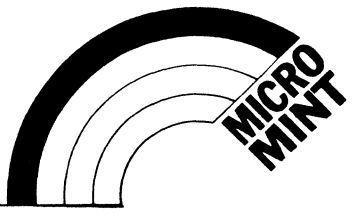
# SB180

## SINGLE BOARD COMPUTER

## Users Manual




**THE MICROMINT, INC.** 25 Terrace Drive, Vernon, Connecticut 06066

**Rev. 1.0**

TIP SHEET 10/23/85

First time users may find the following list of tips useful in setting up and using their system:

1) The system provided must be booted on a double-sided-double-density 48 tpi disk drive with the terminal set for 9600 baud. The system may later be configured by the user to boot from 96 tpi drives and at a different baud rate. (Use CONFIG to change the baud rate.)

2) ZCPR3 allows the use of up to 32 user areas on each disk. (The system provided has been set up to use areas 0 through 15 only. Should the user need more areas than this, the system source files on disk three must be modified and reassembled.) The different areas may be accessed by typing the desired area number followed by a colon and a return at the system prompt. For example, to move from area 0 to area 15 type '15:' at the 'A0:BASE>' prompt.

3) A user area may also have a name associated with it. When the 'System Master' disk is used to boot the system, user area 0 is named 'BASE' while area 15 is named 'ROOT'. To move between areas using their names, type the area name instead of its number, followed by a colon and a return. Another way to move to area 15 in the above example is to type 'ROOT:' at the 'A0:BASE>' prompt.

4) To see a list of currently defined named directories, type 'PWD' followed by a return. To get a list of all the files on a disk in all the user areas, put the disk in drive B: and type 'XDIR B: U' followed by a return. (SB180-20 software only.)

5) All of the files on disks two, three, and four may be found in user area 0. (SB180-20 software only.) However, most of the files on disk one are in user area 15. This is so the user doesn't have to see a list of system tools every time he does a directory.

6) As provided, the system uses a disk head step rate of 10ms. (If this means nothing to you, go on to the next paragraph.) A quicker step rate may be used with some drives to increase the overall disk access rate. Check your disk drive's manual to determine if a faster step rate can be used. (e.g. TEAC FD-55B-20-U drives can use a 6ms step rate.) Run CONFIG to make this change.

7) Besides the system tools on the 'System Master' disk, there are also a number of built-in commands. These include commands in the command processor (CP), which are GO, SAVE, GET, JUMP, and NOTE, and commands in the resident command package (RCP) such as TYPE and POKE. For a list of RCP commands, type H at the system prompt. For more details, see the ZCPR3 book.

8) If you plan to mount your SB180 in an enclosure, be sure to use nylon or plastic washers between any metal hardware and the SB180's circuit board. Any metal allowed to touch the board my cause short circuits and prevent the board from operating properly.

9) If you have a printer plugged into your system, be sure the power on the printer is turned on before booting the system. If the printer is turned off when the computer is turned on, the system may hang and refuse to boot until the printer is unplugged or turned on.

10) The table in the manual describing the correct jumpering for a TEAC 55B drive is for a TEAC FD-55B-20-U drive. We have found at least two other 55B drives which need different jumpers installed. Also, the TEAC 55F 80-track drive has its own set of required jumpers. These are listed below:

TEAC FD-55B-20-U
install ML, UR, DSx, and terminators
TEAC FD-55B-01-U
install HL, UR, PM, DSx, and terminators
TEAC FD-55BV-06-U
install HL, RY, DSx, and terminators

TEAC FD-55F-03-U
install HL, UR, PM, DSx, and terminators

(Terminators should be installed only on the last drive on your cable. The DSx designation above refers to the DS0, DS1, DS2, or DS3 jumper, depending on whether you are configuring drive A:, B:, C:, or D:.)

11) There are over 40 remote access systems (RAS), also called bulletin board systems, across the country which run Z-System as their operating system. Called Z-Nodes, they specialize in supporting Z-System with updates to existing tools, developing new tools, distributing useful public domain software, and in answering any and all questions dealing with Z-System. The main Z-Node, supported by Echelon, is Z-Node Central. The number there is (415) 489-9005. Located on that board is a list of all the other Z-Nodes so you can find the one closest to you.

12) Micromint also has a RAS called the Circuit Cellar BBS. Designed to support users of Micromint manufactured Circuit Cellar projects, it runs on an SB180 and its primary support is for the SB180. Call it anytime to obtain advise, ask questions, download useful public domain utilities, and read advance announcements of other Micromint products. The number is (203) 871-1988. It uses 8 bits, 1 stop bit, no parity, and runs 300 and 1200 baud. It is available 24 hours a day and, like Z-Node Central, makes the list of Z-Nodes across the country available to users.

13) There is an SB180 users' group being formed and is actively recruiting new members. The North American One-Eighty Group (N.A.O.G.) publishes monthly newsletters and will be making available to members disks full of useful utilities, programs, and hints for the cost of the media and shipping. More information plus a membership application is available on the Circuit Cellar BBS or write or call:

North American One-Eighty Group
P.O. Box #2781
Warminster, PA 18974
(215) 443-9031

# ERRATA

Page 20, Step 2 and Figure 2.7-2

Current production boards no longer have jumpers supplied for JP10. Instead JP10 is hardwired for simultaneous 3.5", 5.25", and 8" operation as shown in the center drawing of Figure 2.7-2. This option provides motor control for 5.25" and 3.5" drives, and also allows selection between 3.5"/5.25" and 8" drives.

You can reconfigure JP10 if desired by removing the existing wire (or, on some boards, cutting the existing trace) and installing your own jumpers.

Page 25, Figure 2.7-8

The figure at the bottom of the page is mis-labeled as "Figure 2.7-8". It should be labeled as "Figure 2.7-6".

Page 26, Figure 2.7-8

The figure in the center of the page is mis-labeled as "Figure 2.7-8". It should be labeled as "Figure 2.7-7".

Page 26, Section 2.7.2

Addendum: To add an 8" Shugart 850/860 DS/DD drive to an SB180 with two 5.25"/3.5" drives, the following jumpers should be in place:

850
2S
Z
A
B
I
R
IW
S2
IT
C
RS
HLL
M
NF
DS3

There should be no terminator on the drive.

# SB180 Technical Manual

Release 1.0

Copyright (C) 1985

The Micromint Inc.
25 Terrace Drive
Vernon, CT 06066

All rights reserved

## Copyright Notice

Copyright (C) 1985 by The Micromint Inc. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in any form or by any means, manual or otherwise, without the prior written permission of The Micromint Inc., 25 Terrace Drive, Vernon, CT, 06066.

## Disclaimer

The Micromint Inc. makes no representations or warranties with respect to the contents hereof. Further, changes are periodically made to the information contained herein. The Micromint Inc. reserves the right to incorporate these changes in new editions of this publication without obligation to notify any person of such revisions or changes. Mention in this document of specific product(s) compatible with the SB180 does not constitute an endorsement of the product(s); rather the information regarding specific product(s) is given for illustrative purposes.

## Trademarks

CP/M, CP/M-80 and MP/M are registered trademarks of Digital Research, Inc. MS-DOS is a registered trademark of Microsoft, Inc. Z80 is a registered trademark of Zilog, Inc. IBM, IBM PC, and IBM Personal Computer are trademarks of the International Business Machines Corporation. Z-System is a registered trademark of Eschelon, Inc. ZCPR is a copyright of R.L. Conn. SB180 and COMM180 are copyrights of The Micromint, Inc.

Date of this release: July, 1985

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

## WARRANTY

The Micromint, Inc. extends the following warranty:

A factory manufactured circuit board or assembly caries with it a one year warranty covering both parts and labor. Any unit which is found to have a defect in materials or workmanship will be repaired or replaced at the option of The Micromint, Inc.

No credit will be given for units which show damage due to user modification or neglect.

Units returned for repair must have prior authorization from The Micromint, Inc. A return authorization number may be obtained by phone or letter. Please retain a record of the the return authorization number as most subsequent correspondence will refer to that number. Under no circumstances is any product to be returned to The Micromint, Inc. without prior authorization. The Micromint, Inc. will assume no responsibility for unauthorized returns.

All returns must be shipped prepaid. Insurance is recommended as losses by a shipping carrier are not the responsibility of The Micromint, Inc. Repaired units will be rturned with postage paid.

For repair of units which have expired their warranty, a minimum inspection fee must be prepaid. Contact The Micromint, Inc. for information on current minimum charges.

NO WARRANTY is extended on USER ASSEMBLED systems or kits. However, assembled kits will be inspected and repaired with charges based on the current minimum one hour charge. However, in the event that repair charges would exceed a reasonable amount, the user may be consulted for a determination. The Micromint, Inc. retains the right to refuse to repair any USER ASSEMBLED item. This right is at the sole discretion of The Micromint, Inc.

Repairs on USER ASSEMBLED items must be prepaid.

Return authorization must be obtained prior to any return.

The Micromint, Inc. reserves the right to change any feature or specification at any time.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

TABLE OF CONTENTS


| SECTION | DESCRIPTION | PAGE |
|---|---|---|
| 1.Ø | SB18Ø System Overview | 1 |
| 1.1 | Notational Conventions | 2 |
| 2.Ø | SB18Ø Installation Instructions | 3 |
| 2.1 | Installation Overview | 3 |
| 2.2 | Connecting the Power Supply | 8 |
| 2.3 | Connecting the Main Console I/O Device | 11 |
| 2.4 | Turning On Power | 13 |
| 2.5 | Connecting An Auxillary Serial I/O Device | 15 |
| 2.6 | Connecting a Parallel Printer | 16 |
| 2.7 | Connecting Floppy Disk Drives | 18 |
| 2.7.1 | Installation of Teac 5.25-Inch Disk Drives | 25 |
| 2.7.2 | Installation of Shugart 8-Inch Disk Drives | 26 |
| 2.8 | The Expansion Bus | 27 |
| 2.9 | Installation of User EPROM | 28 |
| 2.1Ø | SB18Ø Installation Checklist | 29 |
| 2.11 | Turning On Power with Disk Drives Attached | 29 |
| 2.12 | In Case of Difficulty | 34 |
| 3.Ø | Hardware Technical Descriptions | 4Ø |
| 3.1 | The Hitachi HD64180 | 4Ø |
| 3.2 | SB18Ø Design Criteria | 42 |
| 3.3 | The SB18Ø Hardware | 42 |
| 3.3.1 | CPU | 43 |
| 3.3.2 | RS-232 Interface | 43 |
| 3.3.3 | Memory Interface | 43 |
| 3.3.4 | 256K Bit Dynamic RAM | 45 |
| 3.3.5 | Centronics Printer Interface | 45 |
| 3.3.6 | Floppy Disk interface | 46 |
| 3.3.7 | Expansion Bus | 48 |
| 3.3.8 | Power Supply | 48 |
| 4.Ø | SB18Ø Monitor | 49 |
| 4.1 | I/O Devices | 49 |
| 4.2 | Disk Format | 49 |
| 4.3 | RESET | 49 |
| 4.4 | Console Baud Rate | 5Ø |
| 4.5 | Console I/O | 5Ø |
| 4.6 | Commands | 51 |
| 4.6.1 | ASCII Table | 52 |
| 4.6.2 | Bank Select | 52 |
| 4.6.3 | Copy Disk | 52 |
| 4.6.4 | Display Memory | 52 |
| 4.6.5 | Emulate Terminal | 52 |
| 4.6.6 | Fill Memory | 53 |
| 4.6.7 | Goto Program | 53 |
| 4.6.8 | Hexmath | 53 |
| 4.6.9 | Input Port | 53 |

| | | |
|---|---|---|
| 4.6.10 | Klean (Format) Disk | 53 |
| 4.6.11 | Move Memory | 53 |
| 4.6.12 | New Command | 53 |
| 4.6.13 | Output Port | 53 |
| 4.6.14 | Printer Select | 54 |
| 4.6.15 | Query Memory | 54 |
| 4.6.16 | Read Disk | 54 |
| 4.6.17 | Set Memory | 54 |
| 4.6.18 | Test System | 54 |
| 4.6.19 | Upload Hex File | 55 |
| 4.6.20 | Verify Memory | 55 |
| 4.6.21 | Write Disk | 55 |
| 4.6.22 | Examine CPU Registers | 55 |
| 4.6.23 | Yank I/O Registers | 55 |
| 4.6.24 | Z-System Boot | 55 |
| 4.7 | Error Messages | 56 |
| 4.7.1 | FDC Error | 56 |
| 4.7.2 | Disk R/W Error | 56 |
| 4.7.3 | Disk Seek Error | 56 |
| 4.7.4 | Disk Not Ready | 56 |
| 4.7.5 | Bad Command | 56 |
| 4.7.6 | Bad Parameter | 56 |
| 4.7.7 | Not Enough Parameters | 57 |
| 4.7.8 | Invalid Interrupt | 57 |
| 4.7.9 | Bad Opcode Trap | 57 |
| 4.7.10 | CTS0* HIGH | 57 |
| 4.7.11 | DCD0* HIGH | 57 |
| 4.7.12 | No ACK* | 57 |
| 4.8 | Disk Format | 57 |
| 4.9 | Monitor ROM Modification | 58 |
| 4.10 | Key Variable Block | 58 |
| 4.10.1 | STARTBYTE | 58 |
| 4.10.2 | CNTLA0 | 59 |
| 4.10.3 | CNTLA1 | 59 |
| 4.10.4 | CNTLB0 | 59 |
| 4.10.5 | CNTLB1 | 59 |
| 4.10.6 | STAT0 | 60 |
| 4.10.7 | STAT1 | 60 |
| 4.10.8 | DCNTL | 60 |
| 4.10.9 | RCR | 60 |
| 4.10.1 | SPCF1 | 60 |
| 4.10.1 | SPCF2 | 60 |
| 4.11 | The "N" NEW Command | 61 |
| | | |
| 5.0 | Schematics | 62 |
| 5.1 | Parts List | 67 |

## LIST OF FIGURES


| NUMBER | DESCRIPTION | PAGE |
|---|---|---|
| 1.Ø-1 | SB18Ø Functional Organization | 2 |
| 2.1-1 | SB18Ø Silkscreen | 4 |
| 2.1-2 | Recommended Mating Connectors for the SB18Ø System Board | 6 |
| 2.1-3 | Orientation of 2Ø-pin Headers | 6 |
| 2.1-4 | SB18Ø Block Diagram | 7 |
| 2.2-1 | SB18Ø System Board Power Requirements | 8 |
| 2.2-2 | J7 Power Signal Specifications | 9 |
| 2.2-3 | Side View of Power Connector J7 | 9 |
| 2.3-1 | J3 Serial Interface Signal Assignments | 11 |
| 2.6-1 | J2 Printer Signal Interface Specifications | 16 |
| 2.7-1 | Jumper Selection for Disk Drives | 19 |
| 2.7-2 | JP1Ø Jumper Setup | 21 |
| 2.7-3 | J9 5.25" Interface Signal Specifications | 21 |
| 2.7-4 | J8 8" Interface Signal Specifications | 22 |
| 2.7-5 | Disk Drive Connectors Specifications | 24 |
| 2.7-6 | Teac 55B Configuration Guide | 25 |
| 2.7-7 | Shugart SA85Ø Configuration Guide | 26 |
| 2.8-1 | Expansion Bus Signals | 28 |
| 2.1Ø-1 | Installation Checklist | 3Ø |
| 2.12-1 | Troubleshooting Chart | 35 |
| 3.1-1 | Comparison of 8 Bit Processors | 4Ø |
| 3.1-2 | Block Diagram and Pin-Out of the HD64180 | 44 |
| 3.1-3 | Block Diagram and Pin-Out of the 9266 | 47 |
| 4.6-1 | Monitor Command Summary | 51 |
| 5.Ø-1 | SB18Ø Schematic Diagram | 62 |
| 5.Ø-2 | Sample Cable Assemblies | 66 |
| 5.1-1 | Parts List for the SB18Ø | 67 |

# 1.Ø SB18Ø System Overview

The Micromint SB18Ø is a single board computer featuring a new generation 8-bit microprocessor which maintains software compatibility with the Zilog Z8Ø while incorporating advanced design features in a single 64 pin chip. The SB18Ø uses just 29 chips on a 4" by 7 1/2" printed circuit board to provide a powerful, low cost processing system which is well suited for a wide range of appliations - from dedicated process control computers to personal computer systems. Figure 1.Ø-1 shows the functional organization of the SB18Ø:

central processing unit
memory interface
RS-232 interface
Centronics parallel interface
floppy disk interface
XBUS expansion bus
power supply

The SB18Ø has the following features:

* Hitachi HD64180 microprocessor running at 6.1 MHz. Supports a superset of the Z8Ø instruction set.
* 256K bytes on board RAM memory (can be partitioned as 64K byte system memory and 192K byte RAM disk or as paged system memory).
* Full 8K byte ROM monitor with disk support (format, read, write, copy, and boot). Can support up to 32K byte ROM on board.
* 2 RS-232 serial ports, one with auto-baud rate detect.
* 1 Centronics parallel printer port.
* Single/double density programmable floppy disk controller. capable of handling a mix of up to four 3 1/2", 5 1/4", or 8" drives. Different size drives can run concurrently.
* Supports 4 external and 8 internal interrupts.
* Requires just +5 V (and +12 V only for RS-232 operation).
* Has an I/O expansion interface

The SB18Ø's ROM monitor is designed to use the (optional) Echelon Inc. Z-System disk operating system, an enhanced, compatible superset of Digital Research's CP/M 2.2 operating system. However, the SB18Ø can also use the CP/M 2.2, CP/M Plus, MP/M II, TurboDOS, or Oasis operating systems (if properly customized).

The SB18Ø is a virtually complete system on a single board. You need only add a power supply, a serial terminal, and one floppy disk drive (4Ø track or 8Ø track DS/DD) to form a complete functional system. To operate the system, simply turn on the power, insert a Z-System disk, and start the bootstrap operation.

Section 2 of this manual provides complete installation instructions, and Section 3 gives a complete description of the hardware logic components which comprise the SB18Ø board. The ROM monitor is described in Section 4.

## 1.1 Notational Conventions

Active low logic signals (those which are true when at a logic low level of Ø volts) are indicated in this manual in two ways. First, active low signals are denoted by the presence of a "*" character following the signal name (e.g., SYSMEMRD*). These signals are also shown with a bar over the signal name, particularly on the logic diagrams. Both notations mean exactly the same thing. On the other hand, active high logic signals (those which are true when at a logic high level of 2.4 volts) are indicated by just the signal name (e.g., READY).

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

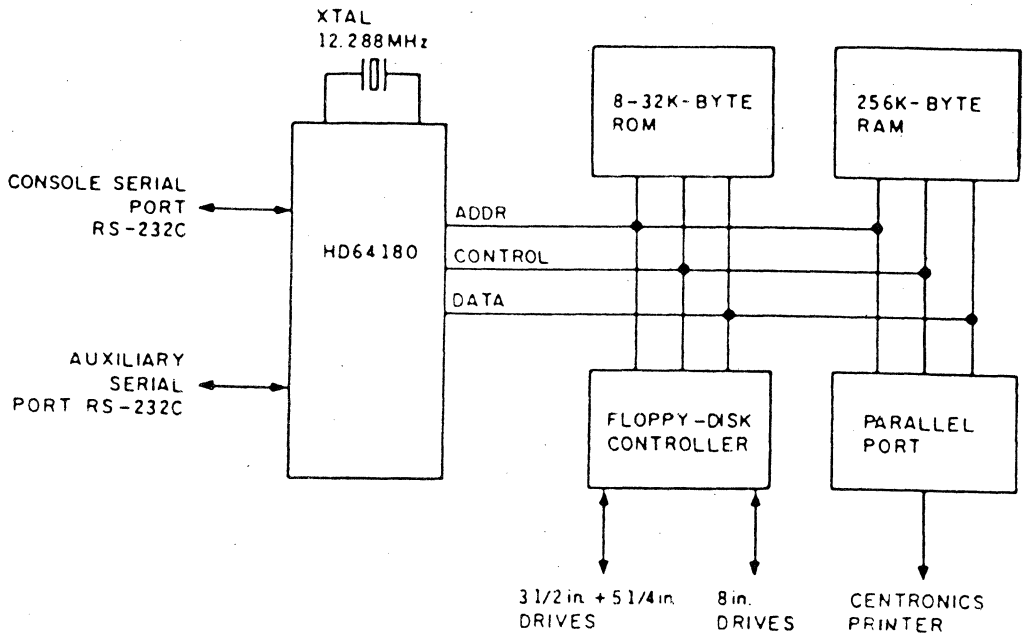



Figure 1.Ø-1 SB18Ø Functional Organization

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

## 2.0 SB180 Installation Instructions

The intent of this section is to be a guide to the installation of an SB180 system. Section 2.1 provides an overview of the installation process and should be read prior to the actual installation. Information presented in sub-sections 2.1 through 2.11 should be used to initially install the SB180 system including associated peripheral I/O devices. Section 2.4 provides a guide to follow when power is first applied if no disk drives are connected to the SB180 board and just the ROM monitor is being used. Section 2.11 assumes that a 40 track DS/DD 5.25" disk drive is connected to the SB180 and that you have the Z-System boot disk.

## 2.1 Installation Overview

The SB180 is designed to be relatively simple to set up and operate. A "bare bones" SB180 system consists of the following hardware components:

* SB180 system board with 256K RAM and 8K ROM monitor
* RS-232C compatible CRT terminal
* Power supply

A more complete system (and a more typical one) would add:

* one or more 5.25" 40 track (or 80 track) DS/DD floppy disk drives (or equivalent 3.5" drives) (Note: the SB180 monitor must have a 40 track double sided drive for Z-System boot up.)

* Centronics compatible parallel printer

Of course once the SB180 is operating in its minimum mode with at least one disk drive, other drives may be added. Additional information in Section 2.7 will detail the addition of 3.5" and/or double sided 8" drives.

It is suggested that the SB180 be **initially installed** and tested for proper operation **without a disk drive.** Once the operation of the ROM monitor has been verified, a floppy disk drive may be added to boot the operating system.

Although the SB180 has been designed to be compatible with "industry standard" peripheral interfaces, it is the responsibility of the user to ensure that any peripheral devices purchased separately meet the SB180 interface specifications. These are defined in the appropriate installation sections and by the actual peripheral devices themselves. Of particular importance, the interconnecting cables must match the interface at both ends. Improper cables will be the cause of system malfunctions in almost every instance, so the time spent in verifying the cable connections will be well worthwhile. The basic interface connectors of the SB180 were designed to use flat ribbon cable and insulation displacement connectors to simplify cable preparation.

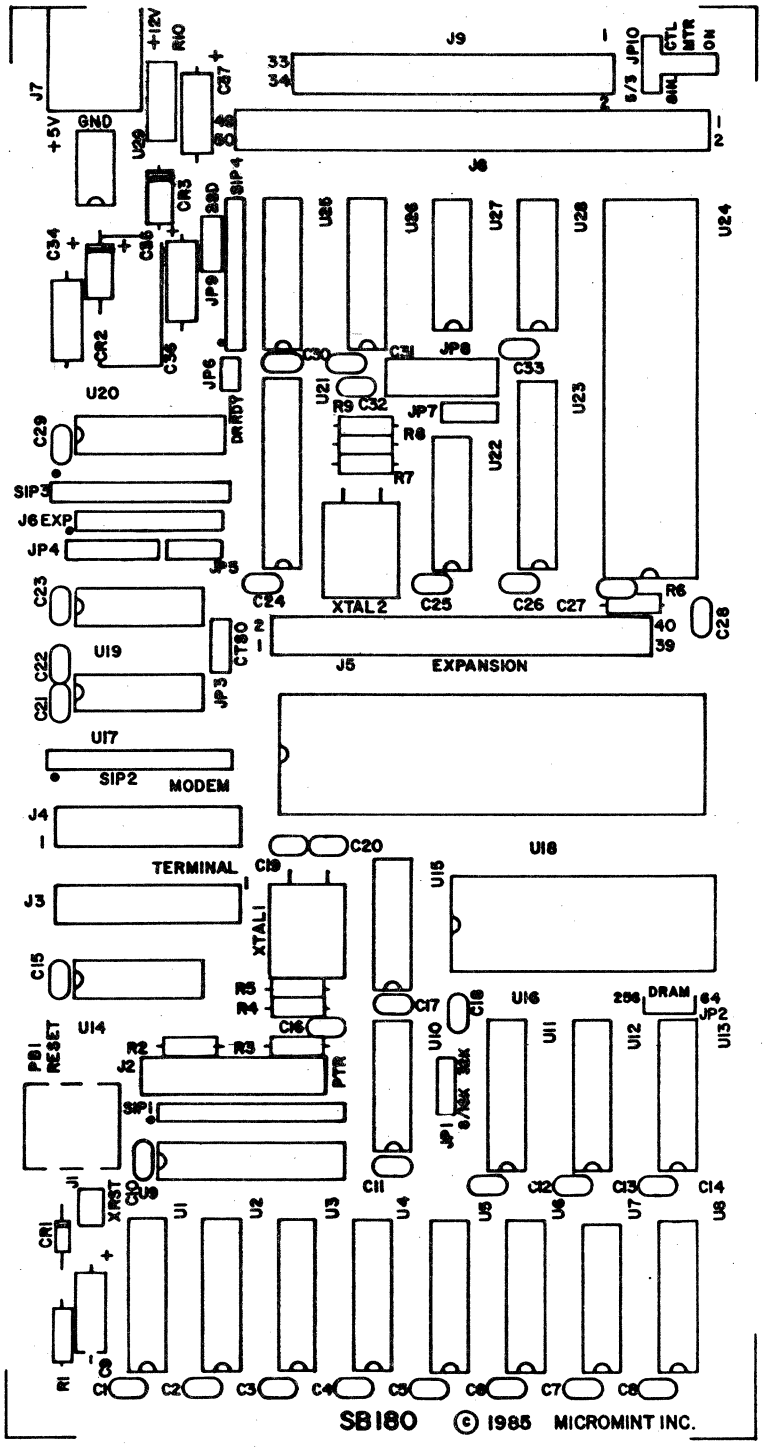



Figure 2.1-1 Silkscreen of SB180 Board

The information contained in each of the installation sections in this manual is geared towards the novice computer user. It is strongly suggested that all users, including those experienced in the installation of computer systems, should read the installation procedures completely before applying power to an SB180 system. Failure to follow the recommended procedures may potentially result in damage to components on both the system board and/or the peripheral devices. In particular, **reversal of the +5v and +12v power leads to the power connector results in permanent damage to all but one IC on the system board!** Figure 2.1-1 shows the silkscreen legend which appears on the SB180 system board. This picture should be referred to for all of the installation procedures as an aid in locating specified connectors, jumpers, IC sockets, etc. Figure 2.1-2 contains part numbers for recommended mating connectors for those on the SB180 system board. Connectors for the peripheral ends of interconnecting cables are dependent on each particular peripheral, and are specified in the manuals which should have come with them. A block diagram of a typical SB180 system using 5.25" floppies, a 3.5" floppy, and an 8" floppy is shown in figure 2.1-4

```
**************************************************************
*                                                            *
*  CAUTION - DO NOT apply power to the SB180 system until the *
*            Installation Checklist (Section 2.10) has been  *
*            completed and you are instructed to do so ! ! !  *
*                                                            *
**************************************************************
```

| SB180 CONNECTOR | DESCRIPTION | MATING CONNECTOR PART NO. | QTY. | MFG. |
|---|---|---|---|---|
| J1 Ext reset | 2 pin solder pads | Hard wire norm. open pushbutton | - | - |
| J2-J4 Ser/Par | 20 pin IDC flat cable connector | 609-2000M | 3 | T&B ANSLEY |
| J5 Expan. | 40 pin header receptacle | 929975-01-20 | 1 | APTRONICS |
| J6 Expan. | 8 pin header receptacle | 929974-01-04 | 1 | APTRONICS |
| J7 Power | 4 pin right angle pin header | 22-01-2041 08-50-0114 | 1 4 | MOLEX |
| J8 8" drive | 50 pin IDC flat cable connector | 609-5000M | 1 | T&B ANSLEY |
| J9 3/5" drive | 34 pin IDC flat cable connector | 609-3400M | 1 | T&B ANSLEY |

Figure 2.1-2 RECOMMENDED MATING CONNECTORS FOR THE SB180 SYSTEM BOARD

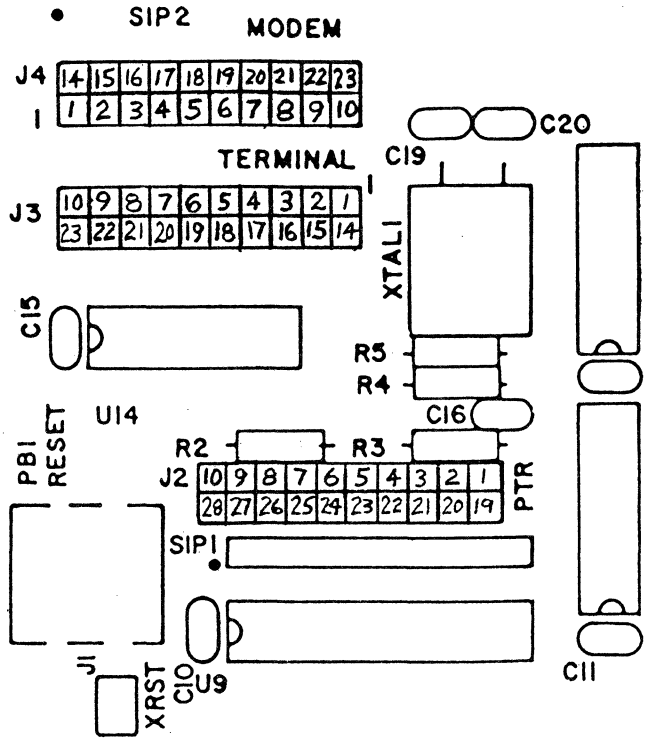



Figure 2.1-3 Orientation of 20-Pin Headers

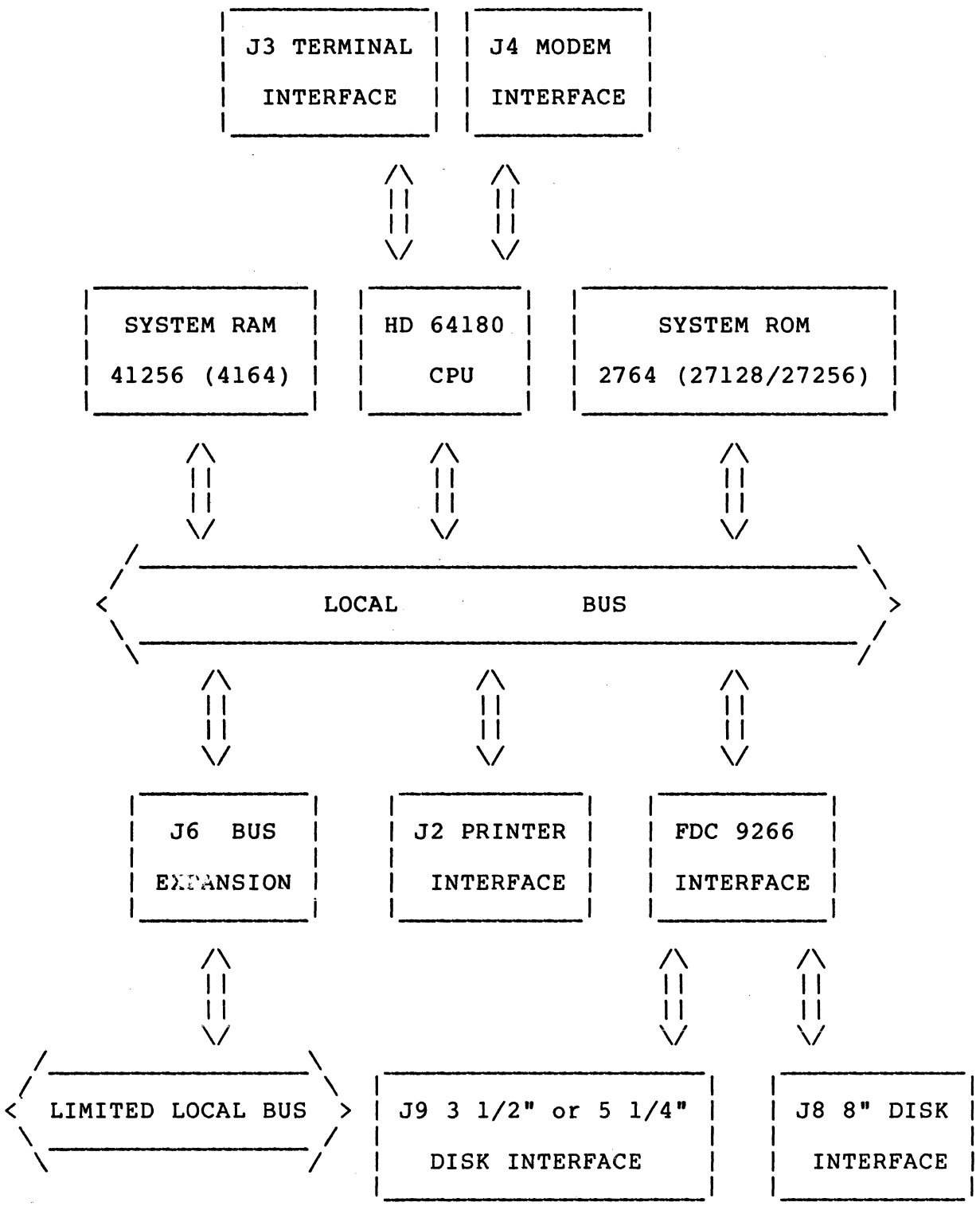



Figure 2.1-4 SB180 Block Diagram

## 2.2 Connecting the Power Supply

The information presented in this section is somewhat generalized since the actual installation of the power supply will be dependent upon several factors such as the number and type of disk drives, and the number of, and the power requirements of any expansion cards which will be installed. An example is given of a typical installation which uses a single power supply (available from Micromint) to power the SB180 system board and two 5.25" floppy disk drives. Recommended part numbers for mating power connectors are given in figure 2.1-2 for the SB180 and in figure 2.7-6 for typical flexible disk drives.

STEP 1 VERIFY THE POWER SUPPLY RATING

Before connecting a power supply to an SB180 system, verify that the supply is capable of providing enough current for all the devices which it will power. This is extremely important, since the SB180 system will probably not function correctly if the supply is operating at reduced output voltage due to overload. At a minimum, the power supply must be able to supply enough current for the SB180 system board. Figure 2.2-1 gives the power requirements for the system board. In addition, if the main power supply is to be used to power any floppy disk drives, then it must be large enough to handle the disk drives as well as the system board. Thus, the first step prior to connecting the power supply is to total up the current requirements for all of the loads. If the total current requirements exceed the rating of the power supply, it will be necessary to replace it with another one that is capable of handling the required load.

| SUPPLY VOLTAGE | TYPICAL OPERATING CURRENT | ALLOWABLE VOLTAGE RANGE |
|---|---|---|
| + 5 VDC | 0.500 amperes | +4.75 to +5.25 |
| + 12 VDC | 0.040 amperes | +11.4 to +12.6 |

Figure 2.2-1 SB180 SYSTEM BOARD POWER REQUIREMENTS

STEP 2 GET/MAKE UP A POWER CABLE

The power supply cable attaches to the SB180 system board at the 4 pin connector, J7. Part numbers for components of a recommended mating connector are given in figure 2.1-2. Signal specifications for the power pins are listed in figure 2.2-2 along with a suggested color coding scheme which will be helpful in avoiding the connection of a wire to an improper voltage level. **WARNING!** IF THE POWER CABLE IS CONNECTED BACKWARDS, IT WILL DESTROY ALL SYSTEM BOARD COMPONENTS! Figure 2.2-3 illustrates J7 from a side view with each pin identified with its corresponding voltage level.

| PIN NO. | SIGNAL NAME | COLOR | NOTES |
|---|---|---|---|
| 1 | +12 V | YELLOW | 40 ma power |
| 2 | +12 V RETURN | BLACK | ground |
| 3 | + 5 V RETURN | BLACK | ground |
| 4 | + 5 V | ORANGE | 500 ma power |

Notes:
1. All wires should be 24 AWG minimum
2. RETURN is the same as GROUND
3. BLACK can be used for both GROUNDs
4. Cable is available as Micromint P/N SB180-P

Figure 2.2-2 J7 POWER SIGNAL SPECIFICATIONS

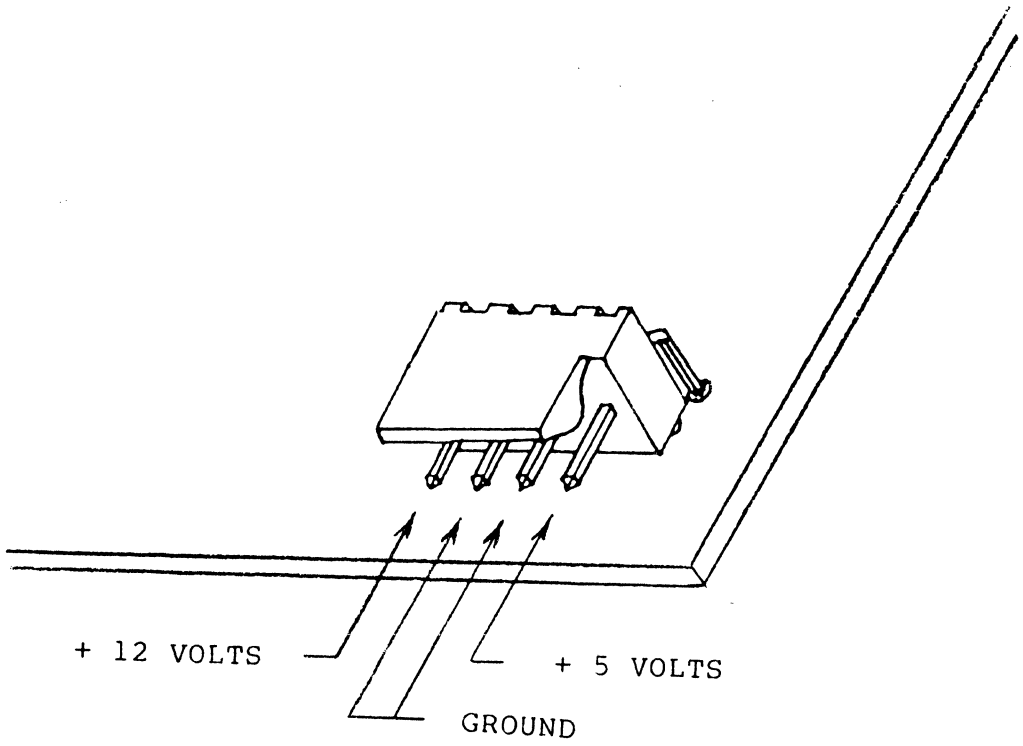



Figure 2.2-3 Side View of Power Connector J7

The type of connector, if any, which attaches at the power supply end of the cable depends upon the particular power supply to be installed. In general, each manufacturer tends to use a different connector, and in most instances a mating connector is not included with the power suply itself. If a mating connector cannot be acquired, the wires can usually be soldered directly to the connector pins, although this is not a recommended practice. In addition, some power supplies use screw type connectors which attach to automotive style spade lugs. These spade lugs would be soldered or crimped to the power leads, and can be used to form a good, reliable power connection which can be easily disconnected if necessary.

If your power supply will operate with no load (some switching supplies require a minimum load), this would be a good point to verify its proper operation. Checking voltages at the connector for the system board insures proper pin outs.

```
**********************************************************************
*                                                                    *
*      CAUTION: UNPLUG ALL AC POWER CORDS BEFORE PROCEEDING          *
*                                                                    *
**********************************************************************
```

STEP 3 ATTACH THE POWER CABLE

Refer to figures 2.1-1 and 2.2-3 for the location and orientation and attach the 4-pin connector to J7 being sure to orient pin number 1 correctly. Make sure that all the pins in the connector and J7 are aligned properly before making the connection. Next, attach the opposite end of the cable to the power supply. Be sure that the correct wires are hooked up to the proper voltages. If a color coding scheme such as the one suggested was followed, this should not be a problem.

```
**********************************************************************
*                                                                    *
*  CAUTION - MAKE SURE THAT THIS CONNECTOR IS INSTALLED IN THE       *
*            CORRECT ORIENTATION TO PREVENT DAMAGE TO THE            *
*            SYSTEM BOARD.                                           *
*                                                                    *
**********************************************************************
```

This completes the SB180 system board power supply installation. Note: DO NOT plug in the AC power cord to a line outlet at this time.

## 2.3 Connecting the Main Console I/O Device

The main console I/O port on the SB180 is designed to attach directly to a standard RS-232C compatible CRT data terminal, such as a Televideo 950. The console serial device is attached to the SB180 system board at the 20-pin, dual row header, J3, which mates with standard 20-pin female flat cable connectors, such as the one recommended in figure 2.1-2. Signal pin assignments for J3 are listed in figure 2.3-1. The connector at the terminal end of the cable is typically a 25-pin DB-25 "D" style connector, and can be either a male or a female connector depending upon the requirements of the particular serial I/O device that is being used for the console.

The SB180 board as shipped is configured such that the console I/O port operates as Data Communications Equipment (DCE). In most instances where CRT (video display) data terminals are used, the serial device is set up to operate as Data Terminal Equipment (DTE), and no special configuration is needed. In this case, the pins on J3 tie directly to corresponding pins on the connector of the serial device. On the other hand, if the serial device is also operating as DCE, then the signal pins must be swapped in the cable assembly (reverse pin 2 with 3 at one end).

| PIN# | EIA RS-232C SIGNAL NAME | I/O(DTE) | I/O(DCE) |
|---|---|---|---|
| 2 | TRANSMITTED DATA (TXD) | O | I |
| 3 | RECEIVED DATA (RXD) | I | O |
| 7 | SIGNAL GROUND (GND) | - | - |
| 1,4-6,8-20 | NOT USED | | |

Notes: 1. Signal direction at J3 is for DCE operation.
2. If hardware handshake is required for the console, J3 may be configured to provide an interface gate for pin 5 (CTS) by changing jumpers JP4 and JP5.
3. The console cable is available as Micromint P/N SB180-T

Figure 2.3-1 J3 SERIAL INTERFACE SIGNAL ASSIGNMENTS

In addition to the problem of matching the cable at each end to the correct interface signals, both the data terminal device and the SB180 system board must be set up to operate with the same set of parameters, such as number of data bits, number of stop bits, type of parity bit, baud rate, etc. This is accomplished via DIP switch settings on the terminal, though some older terminals may use hard-wired jumpers. The SB180 features auto baud rate detect for 300, 1200, 9600, and 19,200 baud, so the terminal may be set for any of these baud rates. The default values for the SB180 are: 8 data bits, 1 stop bit, and no parity. This code configuration may be changed after the SB180 is operating but **must be adhered to for initial operation.** Keeping these things in mind, you are now ready to configure and install the main communications device.

STEP 1 TURN OFF POWER

Disconnect power to both the SB180 system board and to the serial device which is being installed as the main console device. Removing power is best accomplished by unplugging all AC power cords.

STEP 2 CONFIGURE THE MAIN CONSOLE DEVICE

The first step is to read the installation instructions which should be included in the manual for the CRT data terminal (or other serial device if applicable). Follow the instructions given and set up the data terminal for operation with these parameters:

8 data bits
1 stop bit
no parity
baud rate: 300, 1200, 9600, or 19,200

If the serial device is configurable for either DCE or DTE operation, set it up as DTE according to the instructions given in the manual for the device. Most CRT terminals are already configured for DTE operation.

STEP 3 GET/MAKE UP THE CONSOLE CABLE

If a cable was not purchased with the SB180 system board, one must be constructed. The connector which mates with the J3 header should be a 20 pin (2x10) female flat cable connector on .100" centers. Refer to figures 2.1-1 and 2.1-4 for the location of J3 and the position of pin number 1. The connector which interfaces with the serial device will be specified in the manual for the device, but it is usually a 25-pin "D" connector. The actual construction of the cable depends on both the type of connectors and cable being used. The entire flat cable might be crimped at one time or individual pins may need to be crimped or

soldered, and then inserted into a connector shell. Most serial terminals conform to the RS-232 DTE standards (i.e., they transmit data on pin 2 and receive on pin 3). The SB180 mates as a DCE device (receives data on pin 2 and transmits on pin 3). If the serial device is not operating as DTE equipment as previously mentioned, reverse pin 2 with pin 3 on one end of the cable.

STEP 4 ATTACH THE CONSOLE CABLE

Refer to figures 2.1-1 and 2.1-4 and connect the RS-232C cable at the SB180 system board, J3, which is a 20 pin dual row header. Be sure that pin number 1 of the cable is oriented correctly with pin number 1 of J3. Now attach the cable to the serial device. If the connector is one of the the D style connectors, it will only go on one way. If not, use the information given in the manual for the device to verify that the connector is properly installed.

This completes the installation of the console device.

## 2.4 Turning On Power

This section describes the procedure for turning on power to the SB180 system for the first time. The SB180 board, if purchased fully assembled, has been "burned in" and fully tested prior to shipment. If any problems occur during the initial power up procedure and operational tests, the source of the problem will almost always be due to either an improper cable connection to the terminal or an incorrect option configuration on the terminal.

The information presented is intended to aid those users who have purchased a fully assembled and tested version of the SB180 system board. Due to the complexity of the circuitry on the board and the sophistication of some of the IC's, problems which are caused by errors in construction might require a full trouble-shooting effort with some sophisticated test equipment. Difficulties on this scale are beyond the scope of the text presented here. However Micromint does offer an inspection and repair service if needed.

STEP 1 SET UP THE POWER SYSTEM

The ideal arangement of the power system will have all AC power outlets associated with the SB180 system controlled by a single ON/OFF switch or circuit breaker. If this is the case, first turn off the main switch, and then turn on the power switches to the system power supply and the main console I/O device. Now the main switch can be used to turn the SB180 system on and off. Otherwise the individual components will have to be controlled independently of one another. This discussion assumes that this is the case.

Position the ON/OFF switch on the power supply and on the main console I/O device to the OFF position. Unplug any power supply or device which does not have an ON/OFF switch. Plug the AC power cords of all devices which are switched (e.g., have an ON/OFF switch) into a wall outlet. Do NOT apply power to the system at this time.

STEP 2 LAST MINUTE CHECKS!

Recheck all cable connections to the SB180 system board and to the main console I/O device. This includes both I/O interface connections, and power cable connections. Make sure that all connectors are installed correctly and are fully mated.

It would be a good idea at this time to verify that all IC's on the system board are fully inserted in their sockets and are oriented in the proper direction. Sometimes during shipping or during rough handling, ICs may work themselves loose in their sockets. Use your finger to press them all the way down in the socket. If you position the SB180 system board so that the label "SB180" is on the bottom edge of the board facing you, the power supply connector will be at the top left side of the board. With the board in this orientation, most of the ICs are oriented vertically with pin 1 being at the bottom right. The remaining ICs are oriented with pin 1 to the bottom left. The top of the IC is usually indicated either by a notch or by a small circular dot on the IC.

STEP 3 READY TO APPLY POWER!

Now you are ready to apply power to the system. This step is sometimes referred to as "smoke testing" the system, since components have been known to burn up due to improper installations. If any smoke is observed during this step, turn off power immediately and investigate the cause before proceeding. Possibilities here include incorrect cable connections, and ICs which are installed upside down in their sockets.

If you are at all unsure of the power supply connections, now is the time to check them for the proper voltages as listed in figure 2.2-2. First, read section 2.12, "Troubleshooting". Disconnect the power supply cable from J7, apply AC power to the system power supply only, and use a multimeter to ensure that the voltages are correct. If the voltages are correct, remove power to the power supply and reconnect the power supply cable to J7 on the SB180 system board.

It is assumed that a video display terminal has been installed in the system as the console I/O device. When power is first applied to the SB180 system, what is first seen on the main console I/O device depends upon a number of factors such as the type of terminal and baud rate. You may see a completely blank screen with a cursor in the upper left corner, or you may see a few random characters or letters on the screen.

Turn on power to the console I/O device first and then to the SB180. Normal responses will be:

1) A cursor should appear on the screen in 4 to 5 seconds.
2) As mentioned above, you may see one or more random characters on the terminal screen.
3) Press the RETURN or ENTER key on the terminal keyboard. This tells the SB180 the baud rate at which the terminal is operating.
4) On the screen will appear the message "Micromint ROM Monitor Version xx.xx"

These responses indicate that the SB180 system board has successfully completed initialization, recognized that no disk drives were attached to it, waited until a key was pressed on the terminal, analyzed it to determine the baud rate, set the baud rate, and then turned control over to the ROM monitor. If the system responded correctly as indicated above, go on to the next step. Otherwise, first try pushing the reset button, PB1, on the system board. If there is still no response, turn off power to the system, wait a few seconds, and try again. If the system still does not respond, go to the "IN CASE OF DIFFICULTY" section.

STEP 4 TESTING THE ROM MONITOR

The SB180 system should now be waiting for you to enter a command. The monitor prompt is "Ø>" where "Ø" denotes the fact that you are currently using the first 64K bank of memory. If you enter a "?", you will see displayed a full page "help" screen showing all of the monitor commands. Since a disk drive is not connected at this time, some of the commands will return an error status code when used. Using the information in Section 4.6 as a guide, you can try these commands:

A,B,D,F,H,M,Q,S,T,V,X, and Y

In particular, the "T" command (without additional parameters) performs a continuous memory test until terminated by a Control-C or Control-X. The SB180 could test memory until you are ready to attach disk drives, a parallel line printer, and an auxillary serial device such as a modem, as detailed in the next section of this manual.

## 2.5 Connecting an Auxillary Serial I/O Device

The auxillary I/O port is very useful in adding an external modem or serial printer to the SB180 system. Most modems usually operate as Data Communications Equipment, and J4 is set up as Data Terminal Equipment, so a standard pin-to-pin cable assembly should work fine. If you do not own an external modem, Micromint has an expansion board (COMM180) for the SB180 which contains a 300/1200 baud modem connected directly to the system data bus, thus it does not use the auxillary I/O port on the system board.

Installing an auxillary serial I/O device requires the same steps as the installation of the console serial device. The differences are that the interface connector is located in a different spot, has a different number, and that the pin numbers are oriented 180 degrees from the console serial interface connector. As such, refer to the information given in section 2.3 for installing the main console serial I/O device and follow those same procedures to install an auxillary device. Substitute J4 in place of J3 where ever it occurs. Figure 2.1-1 shows the location of J4. As you can see J3 and J4 are situated adjacently on the SB180 system board. If you have already installed the console device this step should be straight-forward. It is recommended that the entire procedure given in section 2.3 be read in its entirety prior to actually installing the auxillary serial I/O device.

## 2.6 Connecting a Parallel Printer

The SB180 system board supports a standard Centronics compatible parallel printer interface. The printer interface cable attaches to the SB180 system board at the 20-pin dual row header, J2. Refer to figures 2.1-1 and 2.1-4 for the location and orientation of J2. Figure 2.1-2 gives a part number for a mating connector. The connector at the printer end of the cable varies between different printers. A typical connector which is compatible with Centronics style printers is a 36-pin male Amphenol part number 57-40360. This connector is also designed to use flat ribbon cable with pin 1 connecting to pin 1 on J2; unused or open pins on the Centronics connector fall toward the pin 36 end of the connector. Many of the newer printers tend to use the 25-pin "D" style connectors similar to the ones used by the serial I/O devices. Figure 2.6-1 below lists the signal specifications for J2.

| J2 PIN NO. | SIGNAL NAME | CENTRONICS PIN NO. | I/O |
|---|---|---|---|
| 1 | DATA STROBE* | 1 | O |
| 2 | DATA1 | 2 | O |
| 3 | DATA2 | 3 | O |
| 4 | DATA3 | 4 | O |
| 5 | DATA4 | 5 | O |
| 6 | DATA5 | 6 | O |
| 7 | DATA6 | 7 | O |
| 8 | DATA7 | 8 | O |
| 9 | DATA8 | 9 | O |
| 10 | ACKNOWLEDGE* | 10 | I |
| 11-20 | SIGNAL RETURNS | 19-30 | - |

Note: This cable available as Micromint P/N SB180-PR

Figure 2.6-1 J2 PRINTER SIGNAL INTERFACE SPECIFICATIONS

STEP 1 DISCONNECT POWER

Check to make sure that power is not applied to either the SB180 system board, or to the printer device. Power should be removed by unplugging all AC power cords associated with the SB-180 system, including those of all peripheral devices.

STEP 2 VERIFY PRINTER INTERFACE SIGNALS

First, read the manual which came with the printer, particularly the section which discusses the installation of the printer in a computer system. Next, verify that the signals on each pin are the same as the signals available at J2 on the SB180 system board. Write down those which are on a different pin so that a printer cable can be constructed in a later step. In most cases the only signals needed are the data lines, the data strobe, the data acknowledge, and a signal return path (ground).

STEP 3 CONFIGURE THE PRINTER

Follow the instructions in the installation section of the printer manual, and set it up for operation as desired. Choices here may include options such as page size, type style, page margins, automatic line feed, character set, etc. (The SB180 monitor normally sends a line feed after each carriage reurn.) Also, if the printer can be configured for the polarity of the strobe and acknowledge signals, make them both active low. The polarity of the printer interface control signals can also be changed by writing a software routine if necessary. Source code for the ROM monitor is available on disk.

STEP 4 GET/MAKE UP A PRINTER INTERFACE CABLE

If a printer interface cable was not purchased with the SB180 system board, one must be constructed. If one is available, use the signal specifications listed in figure 2.6-1 and information from the printer manual to verify the wiring. Change any wires which are incorrect. You should have a list of these from Step 2 above. If a new cable must be constructed from scratch, first verify that the two connectors mate properly with both the 20 pin header at J2 and at the printer's interface connector. Next, wire the two connectors together such that each signal connects to the proper pin number at both ends of the cable. The J2 header pin layout mates directly with insulation displacement connectors and flat ribbon cable.

STEP 4 SELF-TEST THE PRINTER

Many printers have a self-test function which continuously prints all the printable characters in a line across the paper. If the printer has this capability, follow the instructions in

the printer manual and run the self-test. Be sure to turn off power to the printer and to disconnect the AC line cord before proceeding.

STEP 5 ATTACH THE CABLE

Attach the cable to the appropriate connectors at each end. Be sure to align pin 1 on J2 correctly (see figures 2.1-1 and 2.1-4). If the printer connector is a "D" style connector, it will only go on one way. If it is not, refer to the printer manual for the proper orientation of the mating connector.

This completes the installation procedure for the parallel printer device.

## 2.7 Connecting Floppy-Disk Drives.

The SB180 system board has been designed to interface to the standard 5.25 inch flexible (usually called minifloppy) disk drive, to the 8 inch floppy disk drive, and to the newer 3.5 inch microfloppy disk drive. (Note: there are several different types of 3.5 inch floppy drives. The SB180 can use only the 40 or 80 track drive that is pin compatible with the 5.25 inch minifloppy.) A thirty-four pin flat cable connecter is installed on the system board to allow for up to four drives of either 5.25 inch and/or 3.5 inch type to be attached in a daisy-chained fashion at a time. Space is provided for the fifty pin flat cable connector needed for 8 inch double sided disk drives, but it is not installed since most users will not be using this size drive. This connector is available from Micromint (P/N SB180-8X). Although instructions are given for using 8 inch drives, the Z-System disk operating system is delivered only on 5.25 inch diskettes and a **5.25 inch double sided drive (40 track) MUST be connected to the SB180 initially to start up.** Any 3.5 inch drives are considered equivalent to 5.25 inch drives and may be daisy chained along with them. Eight inch drives (double sided only) may be added simultaneously or later (but you must add the 8 inch connector (J8) as mentioned above).

The interface connectors for both types of drives support the "industry standard" interface specifications for floppy disk drives, and thus can attach directly to many of the standard 8 inch and minifloppy drives currently being used. It is the responsibility of the user to ensure that the particular drive to be installed adheres to the interface specifications of the SB180 flexible-disk controller interface. In addition, there are several jumpers on the system board which may be installed or removed depending on the size of the disk drives used. Figure 2.1-1 shows the location and orientation of connectors and jumpers associated with the floppy-disk interface. Refer to it as required during the installation procedures.

The installation of floppy-disk drives to an SB180 system can be complex. If at all possible, an OEM manual for the drives which are being installed should be obtained. Since manuals are not always readily available, the discussions which follow detail the installation of the Teac 55B 5.25 inch and Shugart SA850 8 inch disk drives as examples. The 3.5 inch drive is for this purpose the equivalent of a 5.25 inch drive. (Some manufacturers of 3.5 inch drives use a 34 pin header rather than the standard edge connector, but the pin assignments are identical.)

Figure 2.7-1 lists the jumpers associated with the floppy-disk drive interface section of the SB180 system board which are dependent on the drive size, 5.25 inch (or equivalent 3.5 inch) or 8 inch. The installation descriptions given are generalized for both types of floppy drives, and the user is directed to the appropriate figures and tables for each type of drive as needed.

In the following discussion, references to signals at the SB180 system board interface refer to disk drives as number 0, 1, 2 or 3. The Z-System DOS, however, refers to these drives as A, B, C and D, respectively. Sometimes disk drive manuals refer to the different drive selection options as drives 1, 2, 3 and 4.

| JUMPER | PURPOSE |
|---|---|
| JP6 | Required for drives without READY line. Generally older drives do not provide this line. |
| JP7 | Hard wired on back of board for fixed write pre-compensation. Cut on circuit side and install a wire jumper for controlled pre-compensation. Controlled pre-comp is applied only on inner tracks while fixed pre-comp is applied during write operations on all tracks. 8" drives may require write pre-compensation. |
| JP8 | Hard wired on back of board for NO write pre-comp. See SMC 9266 manual for full specifications. |
| JP9 | Allows use of only single sided drives; in mixed systems, a single sided drive would have a jumper installed to enable use of this multiplexed status line. Z-System software does not support one side operation |
| JP10 | For mixed drive size operation. Allows the processor TXS line to control 5.25" drive motors or selection of 5.25"/8" data transfer rates to disk controller. |

Note: These jumpers are intended to implement all of the advanced features of the disk controller and for special configurations. **No changes required for most standard drives.**

Figure 2.7-1 JUMPER SELECTION FOR DISK DRIVES.

From one to four soft-sectored floppy-disk drives can be attached to one of the two SB180 system board flexible-disk interface connectors. J9, a 34-pin right angle flat cable connector, is used to attach 5.25 inch minifloppies (and 3.5 inch microfloppies). J8, a 50-pin cable connector (optionally installed by the user), is used to attach standard 8 inch floppy disk drives. Both connectors mate with standard flat cable connectors such as the ones recommended in figure 2.1-2. Interface signal definitions are shown below in figures 2.7-2 and 2.7-3 for 5.25 inch and 8 inch drives, respectively. Although the signal pin-outs have been designed to directly interface with many of the drives commonly in use today, the SB180 system installer must verify that all signals match the interface requirements for the particular drive which is being installed. In some instances, it may be necessary to change some of the wires on the drive interface cable in order to match up the interface signals between the drives and the SB180 system board. If the drives being installed are compatible with the Teac 55B minifloppy or the Shugart SA850 8 inch drive interfaces, there should be no problems associated with attaching the drives to the SB180 system board, and getting Z-System up and running.

STEP 1 TURN OFF POWER

Before doing anything else, ensure that all power is turned off to both the SB180 system board and to all of the floppy disk drives. The safest procedure to follow is to unplug all AC power cords associated with the SB180 system, including those of peripheral devices.

STEP 2 CONFIGURE FOR 5.25-INCH (AND 8-INCH DISK DRIVES)

Refer to figure 2.1-1 for a picture of the location of the SB180 system board jumpers associated with the flexible-disk drive interface, Next, use the table in figure 2.7-1 to select the appropriate jumpers for the type of floppy-disk drives which are to be installed. A diagram of the flexible-disk interface area of the SB180 system board with jumpers installed at JP10 for 5.25 inch, for 8 inch, and for mixed size drives is given in figure 2.7-4. Using the appropriate pictorial view as a reference, configure the SB180 system board for the desired drive type by installing the jumpers as indicated in the table. Note that these jumpers are configured based solely on the size of disk drive, and are the same regardless of which manufacturer's drive is used.

Page 20, Step 2 and Figure 2.7-2

Current production boards no longer have jumpers supplied for JP10. Instead JP10 is hardwired for simultaneous 3.5", 5.25", and 8" operation as shown in the center drawing of Figure 2.7-2. This option provides motor control for 5.25" and 3.5" drives, and also allows selection between 3.5"/5.25" and 8" drives.

You can reconfigure JP10 if desired by removing the existing wire (or, on some boards, cutting the existing trace) and installing your own jumpers.

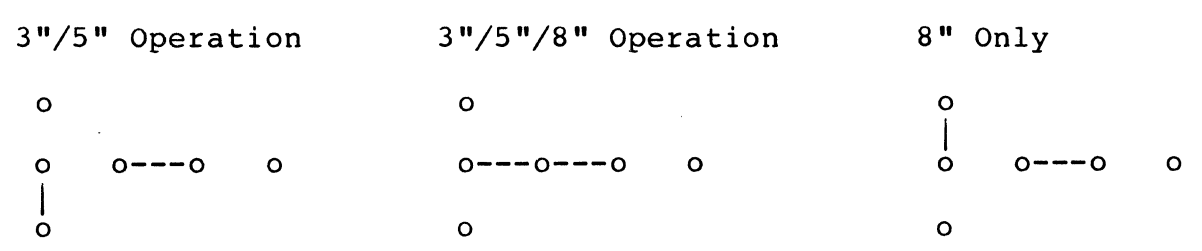



Figure 2.7-2 JP1Ø Jumper Setup

STEP 3 VERIFY DISK DRIVE INTERFACE SIGNALS

The next, and perhaps the most important step to take, is to verify that all of the signals at the floppy-disk drive match those of the SB18Ø floppy-disk drive interface specifications as listed in figure 2.7-2 for 5.25 inch drives, or in figure 2.7-3 for 8 inch drives. If such is not the case, the disk drive interface cable must be altered such that all signal names agree at both ends. Also noted are pins which are directly compatible with the Teac 55B or Shugart SA85Ø disk drives. Note that disk drives are generally connected with flat ribbon cable. Due to the pinouts of the connectors, adjacent signal lines have an interposing ground line between them in the cable. Failure to provide this grounding virtually guarantees problems!

| PIN NO. | SIGNAL NAME |
|---|---|
| 4 | HEAD LOAD/IN USE* |
| 6 | DRIVE SELECT 3* |
| 8 | INDEX* |
| 1Ø | DRIVE SELECT Ø* |
| 12 | DRIVE SELECT 1* |
| 14 | DRIVE SELECT 2* |
| 16 | MOTOR ON* |
| 18 | DIRECTION |
| 2Ø | STEP* |
| 22 | WRITE DATA* |
| 24 | WRITE GATE* |
| 26 | TRACK Ø* |
| 28 | WRITE PROTECT* |
| 3Ø | READ DATA* |
| 32 | SIDE SELECT (Ø/-1) |
| 34 | READY* |

ALL ODD PINS SIGNAL RETURNS (REQUIRED)

Note: This cable is available as Micromint P/N SB18Ø-DSK

Figure 2.7-3 J9 - 5.25" INCH DRIVE INTERFACE SPECIFICATIONS.

The head load function should be configured at the disk drive end of the interface cable according to instructions given in the drive manual. Use of this capability can greatly extend the useful life of the flexible diskettes. Not all drives support a head load option at the interface.

It is quite likely that the floppy disk drive does support a READY* control function. In case your drive does not support a READY* line, jumper JP6 on the SB180 system board must be installed for the disk controller to function. Most newer 5.25 inch drives do support a READY* control function. READY* generally indicates that a diskette is installed and seated in the drive. On some drives READY* only becomes active when index pulses indicate that the disk is up to speed.

| PIN NO. | SIGNAL NAME | |
|---|---|---|
| 2 | LOW CURRENT* | |
| 4 | FAULT RESET* | (not used) |
| 6 | FAULT* | (not used) |
| 8 | MOTOR ON 2* | (not used) |
| 10 | TWO SIDED* | |
| 12 | MOTOR ON 1* | (not used) |
| 14 | SIDE SELECT | (0/-1) |
| 16 | MOTOR ON 0* | (not used) |
| 18 | HEAD LOAD 0* | |
| 20 | INDEX* | |
| 22 | READY* | |
| 24 | HEAD LOAD 1* | (not used) |
| 26 | DRIVE SELECT 0* | |
| 28 | DRIVE SELECT 1* | |
| 30 | DRIVE SELECT 2* | |
| 32 | DRIVE SELECT 3* | |
| 34 | DIRECTION | |
| 36 | STEP* | |
| 38 | WRITE DATA* | |
| 40 | WRITE GATE* | |
| 42 | TRACK 0* | |
| 44 | WRITE PROTECT* | |
| 46 | READ DATA* | |
| 48 | HEAD LOAD 2* | (not used) |
| 50 | HEAD LOAD 3* | (not used) |
| ALL ODD PINS | SIGNAL RETURNS | (REQUIRED) |

Figure 2.7-4 J8 - 8" DRIVE INTERFACE SPECIFICATIONS

STEP 4 CONFIGURE THE DISK DRIVES

This step is highly dependent on the particular drive type to be installed. As an aid in configuring the disk drives, section 2.7.1 describes a typical configuration for the installation of two Teac 55B minifloppy disk drives, and section 2.7.2 discusses typical option selections for installing Shugart SA850 eight inch drives. Note that there are two main differences between the two drives in either case: each drive has a different drive select control line enabled (thus these are called "radial" lines), and only the drive which is physically located at the end of the interface cable has terminator networks (usually DIP resistor networks) installed for the "multiplexed" signals (those which share a single cable wire). The SB180 uses a 330 ohm terminator network allowing up to 1.5 meters of total cable length. For some of the newer low power (CMOS) drives, this termination may require a higher value terminator. Use of a 1k terminator reduces allowable cable length to 1 meter. Where longer cables are required, a 150 ohm terminator may be installed to allow up to 3 meters of cable. If SIP4 (the terminator) must be changed, it must be desoldered from the SB180 board.

STEP 5 CONFIGURE THE SB180 SYSTEM BOARD

Configuring the SB180 system board for disk drives consists of setting the appropriate jumpers. If only double sided 5.25 inch (and 3.5 inch) drives are being used, then no changes are required. Note that the operating system software supplied initially **requires** the use of 5.25 inch double sided drives; you may then change jumpers, modify the BIOS and monitor EPROM to boot from 8 inch drives. No BIOS or EPROM changes are required to add 8 inch double sided drives to a system which has 5.25 inch drives as well.

The connectors on the "T"-shaped block JP10 (upper right hand corner of the SB180 system board) must be set as shown in figure 2.7-4. Finally, if the disk drives do not support a multiplexed READY* control signal, then install a shorting jumper wire in JP6; this signal must be active before the floppy-disk controller will attempt to read a disk. Most 8 inch drives and newer 5.25 inch drives tend to support this control signal while some older 5.25 inch drives do not.

STEP 6 GET/MAKE UP DISK DRIVE CABLES

Floppy-disk drives require both a power cable (in some cases two of these are needed) and an interface cable. The typical 5.25 inch (or 3.5 inch) minifloppy drive uses a 34-pin dual row card edge type of connector for the interface, and a 4-pin power connector for +5 and +12 VDC power. The typical 8 inch disk drive requires a 50-pin dual row card edge type of connector for the interface signals, a 6-pin power cable for DC power, and a 3-pin cable for AC power. Figure 2.7-5 lists mating connector part numbers for the Teac 55B minifloppy and the Shugart SA850 8

inch floppy-disk drives. For other manufacturers' drives consult the drive manual to verify mating connector part numbers as well as signal specifications.

| CONNECTOR TYPE | MFG. | 3.5 INCH PART # | 5.25 INCH PART # | 8 INCH PART # |
|---|---|---|---|---|
| INTERFACE | ANSLEY | 609-3400M | 609-3415M | 609-5015M |
| | 3M | - | 3463-001 | 3415-0001 |
| | AMP | 1-499566-9 | 1-499560-2 | 1-499566-2 |
| DC POWER | AMP | - | 1-480424-0 | 11-480270-0 |
| AC POWER | AMP | N/A | N/A | 1-480303-0 |

Notes: 1. Pins for the AMP housings are AMP number 60619-1
2. Part numbers given are typical, but may not match all drives. Consult the drive manual.

Figure 2.7-5 DISK DRIVE CONNECTORS SPECIFICATIONS.

If cables were not purchased with the SB180 system board, or are not already available, they will have to be constructed for the floppy-disk drives. When making the interface cable be sure that all of the signals are the same at the SB180 flexible disk interface connector as at the disk drives. The number of connectors needed which mate with the disk drives obviously depends on the number of drives which are being installed. Of course, a cable with connectors for all four drives can be made up even if all four drives will not be installed at this time. This would simplify later expansion of the system for additional drives.

STEP 7 ATTACH THE DISK INTERFACE CABLE

Attach the 34- or 50-pin flat cable connector, as appropriate, to the SB180 system board connector, J9 or J8, respectively. Make sure that pin 1 of the cable connector matches up with pin 1 of the disk interface connector on the SB180 system board. If the SB180 system board is held such that J9 and J8 are located at the top of the board, pin number 1 is the upper right pin. Now connect the edge card connectors to the floppy disk drives. The order in which drive numbers are attached does not matter, except that the last drive located at the far end of the interface cable must be the one which has the terminators installed for the multiplexed signals as previously described.

STEP 8 ATTACH THE DISK DRIVE POWER CABLE(S)

Depending on the requirements of the particular drive which is being installed, one or two power cables may be needed. Attach the DC power cables at both ends. These cables are usually constructed so that the wires are daisy-chained from one drive to the next. For small SB180 systems, the end of the DC power cable which attaches to the power supply may be part of the power connector for the system board, and may have already been installed from section 2.2. Next attach the AC power cable if one is required, but do not plug the AC source into a power outlet at this time. In most instances the power end of the AC line cord will probably tie into a terminal block and a single plug or power switch will be used to power the entire SB180 system.

## 2.7.1 Installation of Teac 55B 5.25 inch Disk Drives

This section describes the installation of two Teac 55B 5.25 inch flexible disk drives. These drives are double-sided, and can record in either single- or double-density formats. The two drives are set up for multiple drive operation. All interface signals are TTL compatible with a logic-low of +0.4V maximum and a logic-high of +2.4V minimum. A logic-low indicates a "true" or active condition, while a logic-high indicates a "false" or inactive condition. The maximum length of the interconnecting cable, from the SB180 system board connector, J9, to the last drive on the cable is 4.5 feet. The recommended cable is standard flat ribbon cable with a characteristic impedance of 100 ohms, or equivalent twisted pairs. Figure 2.7-6 lists option selections on the 55B's in a typical two-drive installation.

| OPTION DESIG. | 55B DESCRIPTION | DRIVE A | DRIVE B | SOCKET PINS |
|---|---|---|---|---|
| -- | TERMINATOR NETWORK | R | I | (SOCKET J3) |
| HS | HEAD SOLENOID | R | R | 1-16 |
| DS0 | DRIVE SELECT 0 | I | R | 2-15 |
| DS1 | DRIVE SELECT 1 | R | I | 3-14 |
| HM | HEAD MOTOR CONTROL | R | R | 4-13 |
| DS2 | DRIVE SELECT 2 | R | R | 5-12 |
| DS3 | DRIVE SELECT 3 | R | R | 6-11 |
| MX | MULTIPLEX OPERATION | R | R | 7-10 |
| UR | LED OPTION 1 (SEL+RDY) | I | I | 1-16 |
| ML | MOTOR ON | I | I | 2-15 |
| IU | IN USE | R | R | 3-14 |
| HL | HEAD LOAD | R | R | 4-13 |
| SM | HM/HS ENABLE | R | R | 5-12 |
| U0 | LED OPTION 2 (IN USE) | R | R | 6-11 |
| U1 | LED OPTION 3 | R | R | 7-10 |
| RE | RECALIBRATE | R | R | 8- 9 |

Note: I=installed, R=removed

Figure ~~2.7-8~~ 2.7-6 TEAC 55B CONFIGURATION GUIDE

## 2.7.2 Installation of Shugart SA850 8 inch Drives

This section illustrates the installation of two Shugart SA850 8 inch flexible-disk drives. Figure 2.7-7 lists the option selections which are typically installed on the disk drives. Note that jumper wires must be added at each drive to select the head load options since a common signal wire at pin number 18 on the interface cable is used for all drives.

| TRACE DESIG. | SA850 DESCRIPTION | DRIVE Ø | DRIVE 1 |
|---|---|---|---|
| T1 | HEAD LOAD TERMINATOR | I | I |
| T2 | DRIVE SEL. TERMINATOR | I | I |
| T3-T6 | TERM. FOR MULTIPLEX INP. | R | R |
| DS1 | DRIVE 1 SELECT INPUT | I | R |
| DS2 | DRIVE 2 SELECT INPUT | R | I |
| DS3 | DRIVE 3 SELECT INPUT | R | R |
| DS4 | DRIVE 4 SELECT INPUT | R | R |
| R,RR | RADIAL READY OUTPUT | I | I |
| RI | RADIAL INDEX OUTPUT | I | I |
| X | HEAD LOAD OPTION | R | R |
| A,B | HEAD LOAD OPTIONS | I | I |
| C | HEAD LOAD OPTION | I | R |
| WP | WRITE PROTECT | I | I |
| NFO | STOP AT TRACK Ø | I | I |
| DDS | DRIVE DECODE OPTION | R | R |
| DC | DISK CHANGE OPTION | R | R |
| HL | STEPPER PWR-HEAD LOAD | R | R |
| DS | STEPPER PWR-DRIVE SEL. | R | R |
| NP | NO WRITE PROTECT | R | R |
| Y | IN-USE FROM HEAD LOAD | R | R |
| Z | IN-USE FROM DRIVE SEL. | I | I |
| TS | TRUE FM DATA SEPARATION | E | E |

Notes: R = jumper removed, I = jumper installed,
E = removed or installed,

Figure 2.7-7 SHUGART SA850 CONFIGURATION GUIDE.

The SA850 8 inch drives are double-sided, and can record in either the MFM mode (double-density) or in the FM mode (single-density). All signal lines are TTL compatible. Outputs are driven by open-collector drivers capable of sinking a maximum of 40 ma at a logic zero level (or true state) with a maximum voltage of Ø.4V at the driving device. Collector current when the driver is at a logic one (or false state) and thus off is a maximum of 250 microamperes. These specifications are typical of most drives as well as the devices in the SB18Ø system board flexible-disk interface area.

Page 26, Section 2.7.2

Addendum: To add an 8" Shugart 850/860 DS/DD drive to an SB180 with two 5.25"/3.5" drives, the following jumpers should be in place:

850
2S
Z
A
B
I
R
IW
S2
IT
C
RS
HLL
M
NF
DS3

There should be no terminator on the drive.

Installation of more than two drives is essentially the same as for two drives. Set up the drives for drive select numbers from Ø on up in ascending order, and install the resistor terminator jumpers in the drive which is physically the last drive in the daisy-chain. On mixed size drive systems (both 34 pin and 5Ø pin connector cables) install the drive terminator on the longer of the two cables.

Installation of newer models of 8 inch drives is generally simpler than for the SA85Ø.

Power requirements for the SA85Ø disk drives are +24 VDC at 1.3A, +5 VDC at Ø.8A, -5 VDC at Ø.Ø5A (newer drives do not use this), and 115 VAC at Ø.5 amperes. DC currents are typical values, while AC current is a maximum value.

## 2.8 The Expansion Bus

The SB18Ø system board can support expansion cards through its I/O expansion bus which is accessed through the 4Ø pin connector J5 and the 8-pin connector J6. Figure 2.8-1 illustrates the signal pin-out of these connectors. Because all of the major peripheral devices which are needed to support a high performance Z-System based microcomputer system are supported by on-board controllers, the expansion bus is only needed for expansion peripherals such as a hard disk controller, a "smart" modem, custom I/O interfaces such as data acquisition controllers, a local area network (LAN) interface, or graphics display controller. The only factor which might limit the use of the expansion connector is the rating of your system power supply, so be sure that your power supply capacity is adequate for continued reliable operation.

When designing custom interfaces, serious consideration must be given to the fact that the busses on the SB18Ø are operating at 6-9 MHz. Long extensions to the busses and bus overloading must be avoided.

As always, the first step to take prior to the insertion or removal of any expansion card(s) is to ensure that POWER HAS BEEN REMOVED from the SB18Ø system board.

The next step is to thoroughly read the installation procedures which come with the expansion card. Follow the procedures given and install the card using connector J5. If all goes well the card should now be up and running.

EXPANSION BUS

| DESCRIPTION | PIN # | | DESCRIPTION |
|---|---|---|---|
| +5V PWR | 1 | 2 | +5V PWR |
| GND | 3 | 4 | GND |
| -RD | 5 | 6 | PHI |
| -WR | 7 | 8 | -RESET |
| E | 9 | 10 | -LIR |
| -NMI | 11 | 12 | -EXP SEL (E0-FF) |
| -WAIT | 13 | 14 | NC |
| -INT0 | 15 | 16 | -HALT |
| ST | 17 | 18 | NC |
| A0 | 19 | 20 | A1 |
| -TEND0 | 21 | 22 | A2 |
| A3 | 23 | 24 | A4 |
| -DREQ0 | 25 | 26 | -I/O ENABLE |
| 8.0 MHZ | 27 | 28 | RESET |
| D7 | 29 | 30 | D6 |
| D5 | 31 | 32 | D3 |
| D4 | 33 | 34 | D2 |
| D1 | 35 | 36 | D0 |
| GND | 37 | 38 | GND |
| NC | 39 | 40 | NC |

J5

Notes: 1. Micromint expansion bus 40 pin header is P/N SB180-EX
2. J6 and its driver IC are generally not populated on the SB180. Expansion boards which require the address decode function will supply the chip and connector.

Figure 2.8-1 EXPANSION BUS SIGNALS

## 2.9 Installation of User EPROM

The SB180 contains a 28-pin socket for a capacity of up to 32K bytes of JEDEC standard ROM or EPROM devices. The standard SB180 system board is shipped with a stand alone monitor installed in a Erasable Programmable Read Only Memory (EPROM) device. This consists of one 8Kx8 2764 or 16Kx8 27128 type EPROM. This EPROM contains the power-on jump vector.

A single jumper is associated with the type of EPROM device which is installed on the system board. This is JP1. Refer to the silkscreen drawing of figure 2.1-1 for the location of this jumper. If the standard 2764 EPROM or a 27128 EPROM is used, the jumper should be in the factory wired position. If a 32K 27256 EPROM is ever installed, the jumper must be cut on the circuit side of the board and a wire installed in the opposite position.

## 2.10 SB180 Installation Checklist

This section is intended to serve as an overall guide to the sequence of steps which should be taken during the installation of an SB180 system. Before commencing the actual installation of the hardware components, sections 2.1 through 2.10 should be read to get an idea of the scope of the project about to be undertaken. Once this has been done, proceed to the checklist given below in figure 2.10-1. After all steps in the checklist have been completed, use section 2.11 to start up the SB180 system and verify correct operation.

## 2.11 Turning On Power With Disk Drives Attached

This section describes the procedure for turning on power to the standard SB180 system after the completion of the system installation procedures given in section 2.4 of this manual and after installation of disk drives as detailed in section 2.7 of this manual.

It is assumed in this section that the Z-System operating system has been purchased with the SB180 system board, and that the user is somewhat familiar with the terminology used by Z-System. Users who are not already conversant with Z-System should read the operating system user's guide before trying to use the SB180 system.

| STEP | DESCRIPTION | DONE |
|---|---|---|
| 1. | Read sections 2.1 through 2.1Ø. | ( ) |
| 2. | Unpack the SB18Ø system board and inspect it for damage. If damage is evident, return to vendor. | ( ) |
| 3. | Using section 2.2 as a guide, install the system power supply(s). | ( ) |
| 4. | Using section 2.3 as a guide, install the system console RS-232C serial device. | ( ) |
| 5. | Using section 2.4 as a guide, check out the SB18Ø ROM monitor with the console device connected. | ( ) |
| 6. | Using section 2.5 as a guide, install the system auxillary RS-232C serial device if needed. | ( ) |
| 7. | Using section 2.6 as a guide, install the system listing device (parallel printer) if needed. | ( ) |
| 8. | Using section 2.7 as a guide, install the system flexible disk drive(s). | ( ) |
| 9. | Using the information in section 2.9, verify the jumper configuration for the EPROM devices. | ( ) |
| 1Ø. | Use section 2 to verify correct functional operation of the basic SB18Ø system. | ( ) |
| 11. | Using section 2.8 as a guide, install expansion cards as required. | ( ) |
| 12. | If expansion cards were added in step 11, repeat step 1Ø. | ( ) |
| 13. | Now operate the SB18Ø under Z-System!!! | |

Figure 2.1Ø-1 INSTALLATION CHECKLIST

STEP 1 & STEP 2 SAME AS SECTION 2.4, STEPS 1 & 2

Follow the instructions as given in section 2.4 of this manual, steps 1 and 2.

Open the door(s) of the flexible disk drive(s) and remove the piece of cardboard which may be inserted in place of a diskette for shipping, if not previously done.

If this is the first time the SB180 system is being powered up, it is recommended that any expansion board be removed until the basic system is up and running. If cards are to be installed on the expansion connector at this time, check them to ensure that they are seated properly into the 40 pin and (optionally 8 pin) connector(s) with the component side(s) facing up.

STEP 3: TURN ON POWER AND BACK UP THE Z-SYSTEM DISK

The SB180 system should now be ready for you to make back up copies of the Z-System system diskette using drive number 0 (or drive A as it is referred to by Z-System). Z-System is a disk operating system, or DOS as they are commonly referred to, which was designed by Echelon, Inc. to be fully compatible with CP/M 2.2. This is simply a collection of programs stored on the diskette which will enable you to create and execute (run) programs on the SB180 system.

**A FEW WORDS OF CAUTION**

Before proceeding any further, the flexible-diskettes which are used for program and data storage by the SB180 system will be discussed. Users familiar with the use and handling of diskettes should skip the next several paragraphs.

While diskettes can handle a large amount of information, they are somewhat fragile and need to be treated with respect. Although they are flexible, they can be easily damaged if they are bent or scratched, or if any foreign matter such as dust, hair, or grease from fingerprints is allowed to touch the surface of the diskette itself. Diskettes should only be handled by the black plastic cover which protects them, and should be stored in the paper covers they come in when not in use. Don't leave them lying around where they will collect dust or be dropped on the floor. In addition, diskettes should be kept away from extremes in temperature and away from magnetic fields. Always use felt-tip pens to write on the diskette labels to avoid damaging the surface of the diskette.

To insert diskettes into a disk drive, first open the drive door (different drives have slightly different door mechanisms). This is usually done by pulling outward on the edge of the door, or in some cases, by pushing in on a door release mechanism. Diskettes can then be inserted into the slot in the drive with the end of the diskette which has the oval cutout leading the way. The side of the diskette which has the manu-

facturer's label usually faces the drive door, and always enters the drive last. Diskettes should always be GENTLY pushed into the drive, taking caution not to bend them; this can result in permanent damage. Once the diskette is fully inserted into the drive, close the drive door by pushing it down. Removing diskettes is accomplished by opening the drive door and carefully pulling the diskette out of the drive, again taking caution not to bend the diskette in the process. Finally, it is never a good practice to remove a diskette when the drive is being accessed since this can accidentally destroy the programs which are stored on it. Most drives have an "in-use" light which indicates that they are being accessed.

Since you certainly don't want to accidentally erase any of the Z-System operating system programs which were just purchased, the Z-System master diskette should be "write protected". For the 5.25" disk on which the Z-System is supplied, this is done by installing a foil tab which is placed over a "write protect" notch in the upper right hand side of the diskette. When the foil tab is in place, the diskette is "write protected" and the diskette cannot be written on. Eight-inch drives, on the other hand, are write enabled when the foil tab is installed (on the lower right side of the diskette), and write protected when it is removed.

Since only one copy of the Z-System operating system master diskette is provided, it is advisable to make a back-up copy of the diskette to use when actually operating the SB180 system. It is a standard practice to operate microcomputer systems from back-up copies of diskettes instead of the originals, which are usually stored in a safe place. This is just a safety precaution in that a new copy can be made if the working diskette becomes damaged or worn out.

### START THE SYSTEM FROM THE MONITOR ROM

Open the door of the disk drive, insert a blank diskette (make sure it is not write-protected), but **keep the door of the drive OPEN.** The SB180 system monitor which is stored in the on-board EPROM devices, will sense the baud rate of your attached console device only when the monitor mode is entered, and the SB180 will power up in the monitor mode only if it senses that the disk drive is NOT ready. **(If you close the disk drive door before applying power, the system assumes a 9600 baud rate for your console and will directly boot Z-System upon application of power.)** You will now use the monitor's "K" and "C" commands to format two blank disks and make two back up copies of your Z-System disk. You will then boot Z-System for the first time from one of the copies.

Turn on power to the console I/O device, to the flexible disk drive(s), to the printer if one is attached and to the SB180 system board, in that order. As described in section 2.4, the ROM monitor will be waiting for you to enter a character from the keyboard. Do so, and you will see the familiar monitor prompt, 0>. Enter "K0 40 <CR>" to format the blank disk in drive 0. The monitor will ask for confirmation and then format the disk.

After the first disk is formatted, remove it and repeat the operation with the second blank disk. Remove this disk and place the **write protected** Z-System disk in the drive and enter "CØ Ø" to tell the monitor to make a single drive copy. The monitor will prompt you to "swap" source (Z-System) and destination (the copy) disks as necessary. Repeat this operation so that you have two copies of the Z-System disk, one for back up and the other for a working master copy. Insert the working master copy into the drive.

STEP 4 BOOT THE OPERATING SYSTEM

You should still see the familiar monitor prompt. Close the disk drive door. Now enter the "Z" command to boot the Z-System operating system. If the Z-System operating system was successfully read from the diskette and loaded into system RAM, the following message will be printed on the screen:

SB18Ø 56K Z-System Ver x.x

If the above message does not appear on the screen, open the door of the disk drive, turn power to the SB18Ø system off, wait a few seconds, and try to start the system up again. Sometimes the diskette will not center completely in the drive when the drive door is closed. If the Z-System prompt still does not appear, go to the "IN CASE OF DIFFICULTY" section. The x.x shown above represents the version number of the operating system. You will also see information relating to the terminal configuration program which will make available terminal-specific information to the operating system for use by utility programs.

You will be presented with a list of terminals. Use the + and - keys to move among the pages of terminal names. When you find the name of your terminal, enter the letter displayed next to it. This will create the file MYTERM.Z3T which contains information about your terminal to be used later by various system utilities. If you did not find your terminal on any list, press ESC to abort and enter TCMAKE. This allows you to describe your terminal to the system in detail for later use. (You should have your terminal manual available for this step.) Note: if at any time you wish to change your terminal description, use the ERAse command to destroy the MYTERM.Z3T file. Reboot the system and you again will able to select a new terminal.

After you have successfully configured Z-System to your terminal, you should then see "A0:BASE>"; this symbol is the Z-System system prompt that tells you that Z-System is ready to receive a command from the console device keyboard, and that drive A, user area Ø is the system default drive. Unless Z-System is told to do otherwise (with the PATH command), it will try to find all program and data files on drive A, user area Ø.

At this point the basic SB18Ø system should be up and running. If it is not, run the SB18Ø system from the monitor, make a new copy of the master Z-System disk, and try again. If

the back-up copy still will not boot up the system, refer to the "IN CASE OF DIFFICULTY" section. You might want to try running some of the example programs in the Z-System operating system users guide in order to become familiar with your SB180 system.

STEP 5 CHECK THE PARALLEL PRINTER DEVICE.

If your system does not have a printer device skip this step. The Z-System operating system will now be used to test the installation and operation of the printer device. We will use the printer toggle, CTRL-P (or ^P) to send all of the characters typed in at the console serial input device to the printer. To enter a CTRL-P, you use the key marked CTRL as if it were a Shift key, holding it down while pressing the "P" key next. From this point on, everything typed in at the keyboard will also appear on the printer. To "toggle" the printer off, enter another CTRL-P.

Type in several characters such as letters and numbers (try to avoid control keys which might be non-printing characters) and verify that they are echoed onto the printer device. If they are not, or if they do not match those which you have typed in go to the "IN CASE OF DIFFICULTY section. Characters will continue to be sent to the printer device until a second CTRL-P character is entered which tells Z-System that the printer should be toggled off.

## 2.12 In Case of Difficulty

If the SB180 system is not functioning correctly, this section may be useful in correcting the problem. The information given here is not intended to encompass full-fledged troubleshooting of the SB180 system board. That is best left to technicians who have the necessary test equipment required because of the complexity of the logic on the system board. This section is meant to help you isolate common installation mistakes to get your system up and running for the first time. The information given in figure 2.12-1 should direct you to the appropriate step to go to for help.

| SYMPTOMS | POSSIBLE CAUSES | STEPS |
|---|---|---|
| System appears to be totally dead. | 1. Power problem. | 2 |
| | 2. Console cable. | 1,3 |
| | 3. Console configuration. | 1,3 |
| | 4. Expansion cards. | 1 |
| | 5. Defective console device. | 3 |
| Disk drive goes on but no characters appear on the display. | 1. Console brightness adjust. | 3 |
| | 2. Console cable. | 1,3 |
| | 3. Console configuration. | 3 |
| | 4. Defective console device. | 3 |
| Wrong/garbage characters appear on display. | 1. Console configuration. | 3 |
| | 2. Defective console device. | 3 |
| Correct power-on message but no Z-System prompt; disk I/O error messages. | 1. Disk drive power problem. | 2,4 |
| | 2. Diskette inserted wrong. | 4 |
| | 3. Disk interface cable. | 1,4 |
| | 4. Disk configuration. | 4 |
| | 5. Bad system diskette. | 4 |
| | 6. Defective disk drive(s). | 4 |
| | 7. No termination resistor. | 4 |
| Printer types wrong characters or none at all. | 1. Printer cable. | 1,5 |
| | 2. Printer configuration. | 5 |
| | 3. Defective printer device. | 5 |

Figure 2.12-1 TROUBLESHOOTING CHART.

STEP 1: CHECK CONNECTIONS.

Most of the difficulties which will arise in the initial installation of an SB180 system can be traced to either an I/O interface cable with wires on the wrong pins, or to an improper configuration of the SB180 system board option jumpers or those of an external I/O device. In this step just check to see that all cables are properly attached to their mating connectors.

Turn off power to the SB180 system and attached peripherals, and check all cables for a good connection. Then repeat the procedure given in section 2 for starting up the system.

STEP 2: VERIFY POWER SUPPLY VOLTAGES.

It is always a good idea to verify that the system power supply voltages are in the required voltage range, usually plus or minus 5% of the nominal value. Sometimes systems will appear to work if the voltages are close to the required values, but will typically "crash" often, or act strangely during operation. A voltmeter or an oscilloscope will be needed to measure the power supply voltages.

Measure all DC voltages for both the SB18Ø system board and for the disk drives, and verify that the voltage levels are within the specified tolerances of the power supply. The voltages on the SB18Ø system board can be measured at the I/O expansion connectors. If voltage readings are OK, the problem lies elsewhere. If the power supply readings are incorrect, but within a volt or two of the nominal value, adjust the voltage adjustment potentiometer(s) on the power supply, if there is one, such that the output(s) is at the correct level. On switching type power supplies you may have to add more load to the +12 volt output to bring the +5 volt output into regulation. If this adjustment corrects the power supply readings, go back to section 2 and try to run the system again.

If voltages under full load are low, recheck the rating of your power supply to insure that it can handle both the average (normal) load as well as the peak load (see your disk drive manual).

If the power supply reading is way off, for instance, +12V instead of +5V, or ØV, the power supply cable is probably wired to the wrong pins. If this is the case, you may have destroyed the components on the SB18Ø system board. Recheck the power supply cable pin-outs and correct any wiring errors if any are found. Turn power back on and measure the voltages again. If the readings are now OK, go back to section 2 and try to run the system again. If the readings are still incorrect, the likely cause is the power supply itself. Disconnect the power cable(s) from the supply and measure the voltages under a no-load condition. If the supply will not regulate without a load (this should be stated on the specification sheet for the power supply) a "dummy load" which is usually just a power resistor of sufficient wattage must be placed across the output(s). If the voltage readings are still not correct, replace the power supply.

STEP 3: CHECK THE CONSOLE DEVICE.

Problems associated with a serial device are almost always due to an incorrect cable or jumper configuration. The important thing to remember is that the device at one end must be operating as Data Communications Equipment (DCE), and the device at the opposite end of the cable must be operating as Data Terminal Equipment (DTE), or the signals must be reversed in the cable assembly. If there is no response at all then suspect either the

cable, or the configuration of the operating mode, DCE or DTE. If characters are appearing but make no sense, suspect one of the configuration settings such as baudrate, parity or stop bits.

Operation of the console serial device can be easily verified if it is placed into a local-echo mode of operation. In this mode, each keystroke is echoed back on the display screen as it typed. Turn off power to the terminal and put the terminal into a local-echo mode if it has one. Another way to accomplish this function is to make a test connector which has pin numbers 2 and 3 shorted together, pins 4 and 5 shorted, and pins 6 and 20 shorted. This test connector would be attached to the terminal instead of the console serial cable of the SB180 system.

First ensure that the terminal is plugged into an AC outlet and that power is turned on. At this point, even if the SB180 system board was not connected to the terminal, a cursor should appear somewhere on the screen. The cursor is typically a blinking box. If no cursor appears, try to adjust any brightness or contrast controls (these may be located on the rear of the terminal) until the cursor becomes visible. If no cursor appears at all, the problem is probably in the terminal. In that case refer to the manual for the terminal for troubleshooting procedures.

If a cursor is present on the display screen, and the terminal has been placed in a local-echo mode, try typing in some characters. These should be displayed somewhere on the terminal screen. If no characters appear, disconnect the serial cable attached to the SB180 console I/O connector if it is still attached, and try typing in some more characters. If characters now appear, there is probably an incorrect connection in the cable. If there still aren't any characters displayed, it is likely that there is something wrong with the terminal, so refer to the manual for the device and follow the troubleshooting procedures given there.

Now that characters are appearing on the display terminal in the local-echo mode, turn off power to the system and reconnect the console serial cable to the SB180 system board. Run the system again as described in section 2 of this manual. If the messages which should be displayed are a meaningless string of characters, or perhaps the terminal just does weird things including a lot of beeping, there is probably a mismatch between the SB180 and the terminal in one of the operating parameters (such as baud rate, parity, etc.). Turn off power to the SB180 system and check the switch/jumper settings on the display terminal device and ensure that the baudrate, parity and stop bits are correct. Correct any which are in the wrong position or setting.

Turn power back on and follow the start up procedure given in section 2. If the SB180 system is still not displaying the correct start-up messages at this point, there may be something wrong with the hardware, and the system board should be returned for repair. Call for a return authorization number prior to returning any equipment to Micromint.

STEP 4: CHECKOUT THE DISK DRIVE(S).

Problems in interfacing the SB18Ø system board with the flexible disk drives are usually attributed to four areas: a bad interface cable/configuration; wrong jumper/switch configurations on the disk drive(s), the SB18Ø system board, or both; incorrect power supply voltages; or non-functioning diskette (inserted wrong, or a damaged diskette). During initial installations, the first two areas will usually be the cause of the problem.

The first thing to do is to verify the power supply connections, and to measure the voltages to check that they are all within the prescribed tolerances, usually plus or minus 5% of the nominal value. This should have been done in step 2 above. If it wasn't, refer to step 2 again and make the voltage measurements and adjustments if needed. The actual voltages used depends on the type of disk drive as indicated in the disk installation section (2.7). If any adjustments were made, go back and repeat the procedures given in step 2.

No matter what kind of problem occurs in the disk drive interfacing, about the only thing which can be done to find and correct the problem is to once again recheck the wiring of the cable and the position of configuration switches and jumpers. Both of these are described in section 2.7. There are no user adjustments in the disk controller logic circuitry of the SB18Ø system board.

Avoid attempts at disk drive "adjustments" until you have a fully operational SB18Ø system and the proper calibration and test equipment.

Using the drive's manual or specification sheet, and the signal pin-out for the appropriate interface connector on the SB18Ø system board, verify signal connections on the disk drive interface cable. If any errors are found, correct them and go back and re-try the start-up procedures of section 2.

Using the drive's manual if it is available, and the example installation information of section 2.7.1 or 2.7.2 as a guide, verify that each disk drive is configured correctly. If a drive manual is not available, and the drive type is not the same as the one used as an example in the installation section, write the manufacturer of the drive to find out how to configure the drive. In general, the disk drives should be set up in the following manner:

1) Multiple drives attached in a "daisy-chain",

2) Terminators installed on all "radial" signals,

3) Terminators installed only on the drive at the end of the interface cable for "multiplexed signals,

4) Radial drive select signals,

5) All other signals must be multiplexed,

6) 8-inch drives must have stepper motor power on at all times unless a motor-on control function is available on the drive unit.

Radial signals are those for which a separate signal exists for each drive. Multiplexed signals are those which are shared by all of the disk drives in the system. The drive currently selected must be the only one to activate the multiplexed signal lines.

Now use the jumper configuration table for the SB180 system board given in section 2.6 to verify the option selections on the system board. Most of the jumper options are associated with the drive type, 5.25 inch (and 3.5 inch), or 8 inch, and have mandatory positions for each type as listed in figure 2.7-1. The only jumpers which you have to decide whether to install or remove are those numbered JP6-10. These jumpers are totally dependent on configuration of the drive units, and will vary from installation to installation. These jumpers are discussed in section 2.7. If any configuration errors were discovered, correct them and then retry the start-up procedures of section 2.11. If the disk circuitry is still not functioning correctly, it is time to get help!

## STEP 5 CHECKOUT THE PRINTER DEVICE

There isn't a whole lot which can go wrong in the installation of a printer. Either the wiring of the interface cable is not correct, or the polarity of control signals is reversed.

First check to see if the printer is plugged into an AC wall outlet, and that the power is turned on. Now use the printer manual and check that any operational switches such as the printer select switch is in the correct position. Of course, the printer must have paper in it! If you found anything wrong up to this point, make the indicated corrections and then go back to step 7 of section 2 and test the printer again.

If the printer still does not work, use the printer manual and the installation notes of section 2.6 in this manual to verify printer cable wiring. If any errors are found, correct them and try the printer test procedure again.

The last area to check is the logic signal polarity of the printer control signals. Of particular importance is the DATA STROBE* and DATA ACKNOWLEDGE* signals, both of which are initialized by the SB180 as active low. If any corrections need to be made here, they must be done via a call to a software routine which must be written for that purpose.

## 3.0 Hardware Technical Descriptions

The information in this section provides technical descriptions of the SB180 hardware components. Schematics for the SB180 board hardware are included in Section 5. Separate hardware components such as disk drives, printers, or serial devices are not described in this manual. Refer to the technical or user manuals for other devices if technical data is required for them.

## 3.1 The Hitachi HD64180

The power of the SB180 is made possible by the Hitachi HD64180 - a microcoded execution unit based on advanced CMOS manufacturing technology. It provides the benefits of high performance, reduced system cost and low power operation while maintaining complete compatibility with the large base of standard CP/M software.

Performance is derived from a high clock speed (6 MHz now, 9 MHz in the near future), instruction pipelining, and an integrated Memory Management Unit (MMU) with 512K bytes memory address space. The instruction set is a superset of the Z80 instruction set; twelve new instructions include hardware multiply, DMA, and a SLEEP instruction for low power operation.

Considered to the Z-80 what the 80188 is to the 8088, system costs are reduced because many key system functions have been included on-chip. Besides the MMU, the HD64180 boasts a two channel Direct Memory Access Controller (DMAC), wait state generator, dynamic RAM refresh, two channel Asynchronous Serial Communication Interface (ASCI), Clocked Serial I/O port (CSI/O), two channel 16-bit Programmable Reload Timer (PRT), a versatile 12 source interrupt controller, and a "dual" (68xx and 80xx families) bus interface all on one 64 pin chip. Table 3.1-1 compares the HD64180 with other 8 bit processors.

| | HD64180 | 8080/Z80 | NSC800 | Z800 | 80188 |
|---|---|---|---|---|---|
| Process | CMOS | NMOS | CMOS | NMOS | NMOS |
| Power | 100mw | 1W | 100mw | 2W | 2W |
| Max. Clock | 10 MHz | 8 MHz | 4 MHz | 10 MHz | 8 MHz |
| Address Space | 512 K | 64 K | 64 K | 512 K | 1 M |
| UARTs | 2 ch. | no | no | 1 ch. | no |
| DMAC | 2 ch. | no | no | 4 ch. | 2 ch. |
| TIMERs | 2 ch. | no | no | 4 ch. | 2 ch. |
| Clocked SIO | yes | no | no | no | no |
| CS/Wait Logic | yes | no | no | yes | yes |
| DRAM Refresh | yes | yes (Z80) | no | yes | no |

Note: The availability of the Zilog Z800 at this time is unknown and specifications on the Z800 are subject to change.

Table 3.1-1 COMPARISON OF SOME 8-BIT PROCESSORS

The HD64180 CPU is comprised of five functional blocks:

o Central Processing Unit - The CPU is microcoded to implement an upward compatible superset of the Z80 instruction set. Besides the twelve new instructions, many instructions require fewer clock cycles for execution than on a standard Z-80.

o Clock Generator - The clock generator generates the system clock from an external crystal or external clock input. The clock is programmably prescaled to generate timing for the on-chip I/O and system support devices.

o Bus State Controller - The bus state controller performs all status/control bus activity. This includes external bus cycle wait state timing, RESET*, DRAM refresh, and master DMA bus exchange. It generates "dual-bus" control signals for compatibility with both 68xx and 80xx family devices.

o Interrupt Controller - The interrupt controller monitors and prioritizes the four external and eight internal interrupt sources. A variety of interrupt response modes are programmable.

o Memory Management Unit - The MMU maps the CPU's 64K byte logical memory address space into a 512K byte physical memory address space. The MMU organization preserves software object code compatibility while providing extended memory access and uses an efficient "common area - bank area" scheme. I/O accesses (64K bytes I/O address space) bypass the MMU.

The integrated I/O resources comprise the remaining four functional blocks:

o Direct Memory Access Controller - The two channel DMAC provides high speed memory-to-memory, memory-to-I/O, and memory-to-memory-mapped I/O transfer. The DMAC features edge or level sense request input, address increment/decrement/no-change, and (for memory-to-memory transfer) programmable burst or cycle steal transfer. In addition, the DMAC can directly access the full 512K bytes physical memory address space (the MMU is bypaseed during DMA) and transfers (up to 64K bytes in length) can cross 64K byte boundaries. At 6 Mhz, DMA is 1 Mbytes per second.

o Asynchronous Serial Communication Interface - The ASCI provides two separate full duplex UARTs and includes programmable baud rate generator, modem control signals, and a multi-processor communication format. The ASCI can use the DMAC for high speed serial data transfer, reducing CPU overhead.

o Clocked Serial I/O Port - The CSI/O provides a half duplex clocked serial transmitter and receiver. This can be used for simple, high-speed connection to another microprocessor or microcomputer.

o Programmable Reload Timer - The PRT contains two separate channels each consisting of 16-bit timer data and 16-bit timer reload registers. The time base is divided by 20 (non-programmable) from the system clock, and one PRT channel has an optional output allowing waveform generation.

## 3.2 SB180 Design Criteria

With all this functionality on one chip, only a few additional chips are needed to implement a truly sophisticated 8-bit single board computer in a small space (less than 30 sq. in.). In terms of the original Altair micro of less than 10 years ago, the functionally equivalent machine would have taken about 35 S-100 boards for a total of 1750 sq. in. (using 8K memory boards!).

In order to reduce chip count further, an enhanced floppy disk controller chip from Standard Microsystems Corporation, the FDC 9266, was chosen. This 40-pin DIP chip is software compatible with the industry standard NEC 765A floppy controller and adds an on-chip digital data separator to the functions of the FDC 9229 floppy disk interface chip as well. It is compatible with single and double sided 3 1/2", 5 1/4" (40 and 80 track), and 8" drives; the data separator handles both single density (FM encoded) and double density (MFM encoded) data. This means that it can be programmed to read and write almost all soft-sectored CP/M disk formats (and MS-DOS disk formats).

With the HD64180's two channel ASCI built in, two serial ports were included into the design automatically, and provision was made for a Centronics parallel printer port as well. Since 256K DRAM chips are now plentiful and inexpensive, 8 of these were used for memory (64K DRAMs may also be used). Because only 64K bytes of this is usually used for the logical memory space, the user can optionally designate the other 192K bytes as a RAM disk in the operating system. Of course, it may also be used for other purposes (such as implementing banked memory for CP/M Plus).

## 3.3 The SB180 Hardware

Figure 5.1 is the schematic of the SB180 computer. Its design is primarily characterized by the high performance, high density MOS devices including 256 Kbyte DRAMs.

The SB180 system design implements the following functional blocks:

CPU
Memory Interface
RS-232 Interface
Centronics Printer Interface
Floppy Disk Interface
XBUS Expansion Bus
Power Supply

### 3.3.1 CPU

The HD64180 is a high system integration device which combines a CPU execution unit with a number of basic system and peripheral functional blocks. These include:

CPU
MMU - Supports 512KB address space
DMAC - 2 channels
ASCI - 2 channel UART with baud rate generator
CSI/O - 1 channel clocked serial I/O
PRT - 2 channel, 16 bit programmable reload timer
Wait State Generator
DRAM Refresh Controller
Interrupt Controller - 12 interrupt sources

The HD64180 requires operation at specific frequencies in order to generate standard baud rates. Standard operating frequency for the SB180 is 6.144 MHz (12.288 MHz crystal). Other operating frequencies which maintain standard baud rates are 3.072 MHz, 4.608 MHz and (later) 9.216 MHz.

### 3.3.2 RS-232 Interface

The HD64180 ASCI two channel UART is connected to 1488/1489 RS-232 line drivers/receivers to provide two separate ports. ASCI channel 1 is used for the CONSOLE, while ASCI channel 0 is used for AUXILIARY RS-232 devices such as printers, plotters and modems. This distinction is made because modems require the extra handshakes which are available with ASCI channel 0, while terminals do not. All primary RS-232 parameters (baud rate, handshaking, data format, interrupts) are software programmable.

### 3.3.3 Memory Interface

The SB180 incorporates a 28 pin JEDEC boot ROM socket which can be jumpered to hold 8Kx8, 16Kx8 and 32Kx8 memory devices. The boot ROM (contains disk boot and ROM monitor) occupies the bottom 256K bytes of the HD64180 physical address space since it is selected whenever A18/TOUT (note: the TOUT timer output function is not used) is LOW. Thus, the boot ROM contents (whatever its size) is simply repeated in the lower 256KB. The boot ROM output (OE*) is enabled by the HD64180 ME* (memory enable) signal. (As configured, the maximum RAM memory on the SB180 is 256K. To support larger memories, additional address decoding would be required to designated RAM and ROM areas in the current 256K boot ROM space.)

The critical ROM timing parameter is Tce (access time from CE*). 200ns (and marginally, 250ns) ROMs can operate with 1 wait state.

At RESET*, the HD64180 begins execution at physical address 00000H, the start of the boot ROM.

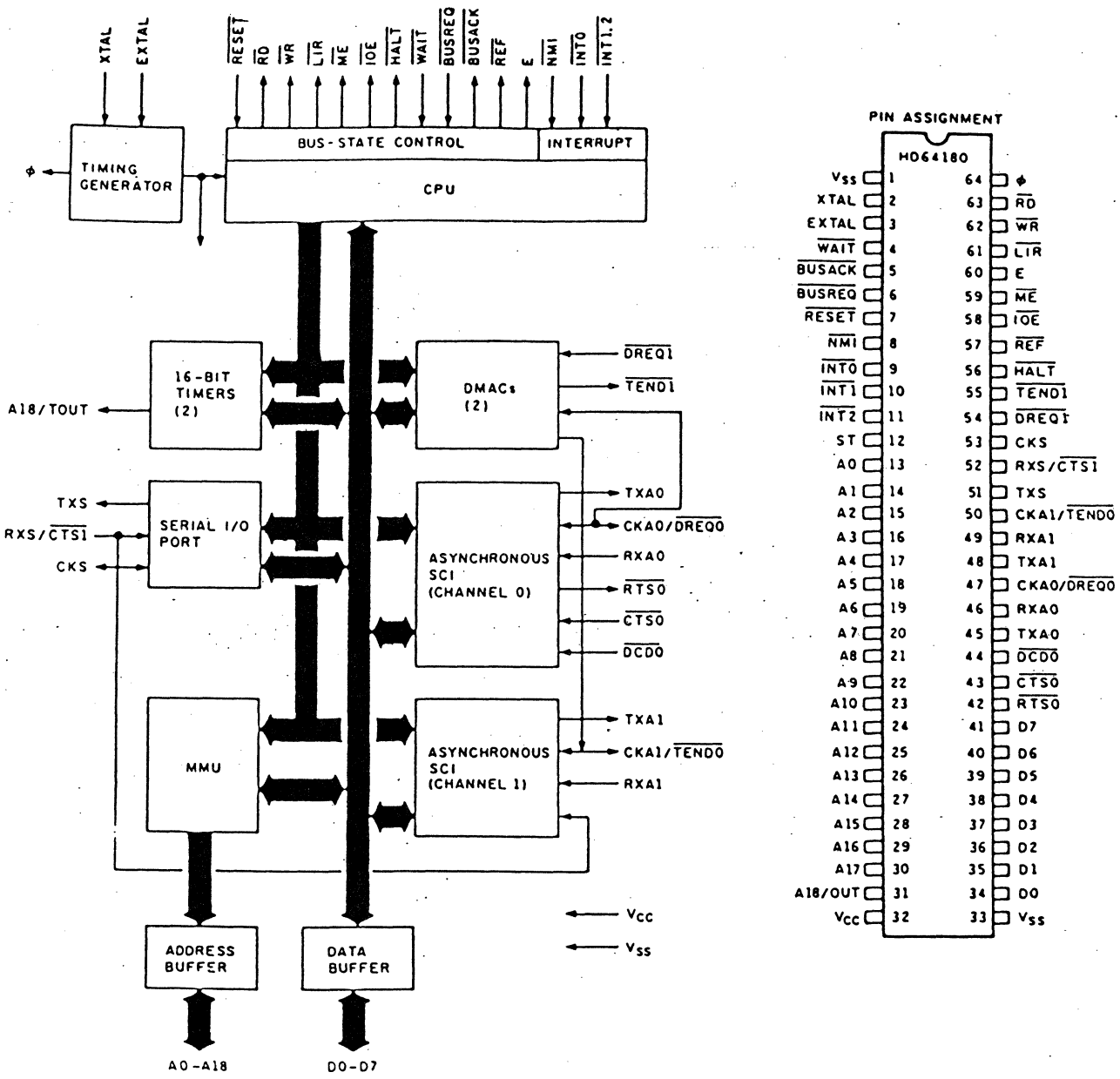



Figure 3.1-2 BLOCK DIAGRAM AND PIN-OUT OF THE HD64180

### 3.3.4 256K Bit Dynamic Ram

Standard 256 Kbit 150 nsec DRAMs, requiring 256 refresh cycles (8 bit refresh address) every 4 ms are used. These RAMs occupy the top 256K bytes of the HD64180 512KB physical address space.

The interface is quite straightforward. Complete DRAM refresh control is provided by the HD64180 in conjunction with control logic U16 and U18 and address muxes U12, U13 and U17.

The HD64180 WR* output directly generates DRAM WE*. The HD64180 ME* output directly generates RAS*. During normal read/write cycles (A18 HIGH, REF* HIGH) CAS* goes LOW at the next rising edge of phi following the rising edge of E (Enable). This provides plenty of set-up time for the address muxes since the rising edge of E switches the address muxes from row to column addresses.

RAS* only refresh is used. The HD64180 generates the refresh addresses. During refresh cycles (REF* LOW), ME* generates RAS* while CAS* is suppressed at U16.

The HD64180 can be programmed to generate refresh cycles every 10, 20, 40 or 80 phi cycles as well as selecting two or three clock refresh. Since the DRAM requires a refresh cycle every 15.625us (4ms/256), the HD64180 is programmed for 80 cycle refresh request since 80 x (1/6.144 MHz) = 13.02 us. Two cycle refresh is also programmed. Thus, refresh overhead is only 2.5% (2 cycles every 80 cycles).

### 3.3.5 Centronics Printer Interface

The Centronics printer interface is comprised of 8 bit latch U5 and F/F U15. The Centronics port is decoded at I/O address 0C0H by U4. To write to the printer, the following sequence is used:

Write data to port 0C1H.

This sets-up the data to the printer and asserts STB* LOW.

Write data to port 0C0H.

This de-asserts the printer STB* signal HIGH>

When the printer has processed the data, it will return the ACK* signal which generates an external interrupt (INT 1*) to the HD64180. The interrupt handler clears the interrupt by performing a dummy output to port 0C0H.

Write (dummy) data to port 0C0H

This clears the INT 1* interrupt request.

The printer interface is not buffered, so compatibility with all printer/cable setups cannot be guaranteed. However, in practice, problems should be rare since the software scheme provides adequate data setup and hold times. Also, note that this printer interface is interrupt driven which allows high performance operation. In a more primitive polling design, excessive overhead limits acceptable performance in such applications as background print spooling.

### 3.3.6 Floppy Disk Interface

The SMC9266 FDC manages almost all details of the drive interface, including data separation and (with external logic U20 and U21) programmable write precompensation. The SMC9266 actually combines a NEC 765/Intel 8272 FDC with SMC's popular 9229 digital data separator. Thus, from the host CPU side, the SMC9266 looks just like these popular devices, including hardware and software compatibility.

The SMC9266 clock is generated by an 8 MHz oscillator comprised of a crystal and U20. Jumpers are provided to select write precomp and allow 8" floppy disk drives to be interfaced.

On the CPU side, the key requirements are interfacing the SMC9266 with both programmed I/O (CS*) for initialization, status check, etc. and with DMA (DRQ, DACK*) for data transfer.

Programmed I/O is straightforward, with CS* generated for I/O address 80H and RD* and WR* directly generated by the HD64180. This is the same scheme used to interface with other '80 family peripherals.

DMA is a little more involved. First, DMAC channel 1 is used for the FDC since dedicated handshake lines (DREQ1*, TEND1*) are provided on the HD64180. Since DMAC channel 0 control lines are multiplexed (with ASCI clocks), DMAC channel 0 is used for memory-memory DMA. This means the ASCI clock functions are available although they are not currently used in this design.

For disk DMA, the 9266 asserts DRQ which in turn causes HD64180 DREQ1* assertion. The HD64180 performs DMA read/writes to I/O address 0A0H, which causes the 9266 DACK* to be asserted, completing the transfer cycle. After the DMAC programmed number of reads/writes has completed, the HD64180 TEND1* output is asserted, and after inversion, causes the 9266 TC (Terminal Count) input to be asserted, completing the DMA operation. This is typically followed by the 9266 generating an HD64180 INT 2* external interrupt. This interrupt service routine can read the 9266 status to determine if errors occurred, etc.

However, there is one 'gotcha', fixed by F/F U18 which conditions the 9266 DRQ output. It turns out that if 9266 DRQ directly generates HD64180 DREQ1* the HD64180 may respond too quickly. This is because HD64180 DREQ* input logic was designed to minimize latency, and thus DREQ* can be recognized at a machine cycle breakpoint. Unfortunately, the 9266 requires that

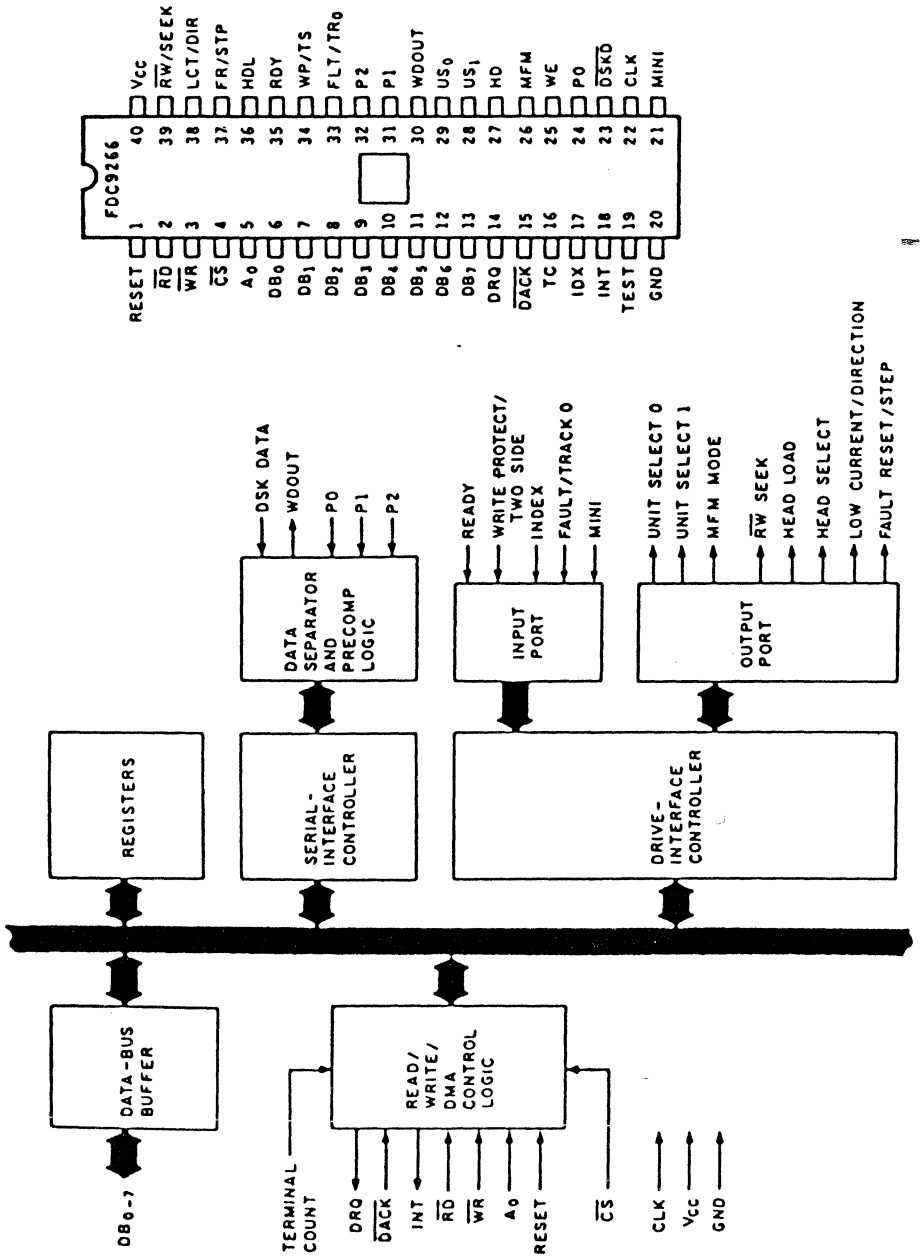



Figure 3.1-3 BLOCK DIAGRAM AND PIN-OUT OF THE 9266

at least 800ns elapse from the time it asserts DRQ before the DMA transfer (DACK*) actually occurs. In other words, when the 9266 'asks' for service, it really doesn't want it...yet! To prevent accessing the 9266 too quickly after DRQ, DRQ from the 9266 is delayed at U18 before issuing the DREQ1* to the HD64180. DRQ is delayed by one REF* cycle time.

Minifloppy double density (MFM) data transfers occur at a 250khz data rate. Thus, each byte must be read within 32us. The disk driver software reprograms the refresh request rate from every 80 phi cycles to every 40 phi cycles prior to disk DMA, and then reassigns it back to 80 phi cycles after the disk DMA is completed. The 9266 DRQ is delayed from between 40 phi clocks to 79 phi clocks. This is about 6-14us. Therefore, the 800ns delay and 32us data transfer constraint are both met. Note that 8" floppy double density (MFM) is twice as fast (500khz) and requires service every 16us. This may require refresh rate increase to every 20 phi cycles to be safe.

### 3.3.7 Expansion Bus

The spare CS* from address decoder U4 (I/O addresses 0E0H-0FFH), along with all major busses (address, data, control) are routed to the XBUS. This allows an I/O expansion board capability. The full complement of HD64180 control signals (IOE*, E*, RD*, WR*, etc.) allows easy interface to all standard peripheral LSI including 80XX, 68XX and 65XX devices. Example expansion boards could include a hard disk controller, 1200 baud modem, or a LAN interface (SIO, SCC or other LAN chips).

### 3.3.8 Power Supply

The SB180 requires +5V and +12V power. A negative voltage is generated on board which is only used by the RS232 driver. The negative voltage is obtained by using a Zener diode to obtain +9V from +12V, which is then inverted using an Intersil 7660 converter. The +12V power is also only used for the RS232 driver. Thus, the SB180 only uses significant power from the +5V supply. Typically, this may be from 0.8 to 1.5 A (depending on the proportion of the TTL and memory devices which are CMOS) - about the same as a 5.25" floppy.

## 4.0 SB180 Monitor

The SB180 monitor provides commands to assist the design and debugging of SB180 related hardware and software. The monitor also serves as a stand-alone training vehicle for the HD64180 high integration CPU.

### 4.1 I/O Devices

The monitor supports the following I/O 'devices':

CON: - Console RS-232 serial port
AUX: - Auxillary RS-232 serial port
CEN: - Centronics parallel printer port
DSK: - Floppy disk storage devices

### 4.2 Disk Format

The monitor supports two disk drive types with the following specification:

5 1/4", 48 TPI, 40 track, double sided, double density
5 1/4", 96 TPI, 80 track, double sided, double density

Note that equivalent double sided 3 1/2" drives can also be used.

During initial system check-out, a 40 track DS/DD drive must be connected to verify operation of the disk interface. After check-out, different disk drives (as supported by the SB180 Z-System DOS implementation) can be connected.

The high capacity format provides 5K bytes per track (10K bytes per cylinder) resulting in a formatted capacity of 400K bytes and 800K bytes for 40 and 80 track drives respectively. Differences in sector sizes between 40 and 80 track formats provides a simple method of drive identification.

### 4.3 RESET

The RESET sequence (from power-up or a reset switch) is as follows. The monitor first initializes the system and performs some diagnostics. Normally (diagnostics OK), the monitor then enters a loop waiting for a disk to be loaded in drive #0 or a carriage return to be entered from the console. If a disk is loaded, the DOS boot routine (same as the 'Z' command) is started. If a carriage return is sensed, the baud rate is determined (see the following section) and the monitor signs on.

Two diagnostic failures cause the above sequence to be changed. First, if a RAM failure is detected, the monitor waits for a carriage return to be entered. In response, a string of 8 bits will be displayed on the console (last displayed is LSB). Bit positions with a '1' represent bad RAM chips. After the display, the monitor HALTs and requires another RESET to restart.

Second, if a problem with the SMC 9266 FDC is detected, the monitor will wait for a carriage return to be entered. In response, the monitor will sign on. However, instead of the normal sign-on message, an error message will be printed.

In either of the above cases the monitor will not try to boot a disk, even if the drive is ready. Thus, if a disk doesn't boot when the system is RESET, enter a carriage return to see the diagnostics results.

## 4.4 Console Baud Rate

The auto baud rate selection described above requires the console baud rate to be either 19200, 9600, 1200, or 300 baud. Note that the baud rate auto-sense routine requires a CPU clock rate of 6.144 MHz.

The console should be configured for the following data format:

8 data bits, 1 stop bit, no parity

Note that the console CTS1* modem control input to the HD64180 is grounded by a trace on the board. CTS1* can be connected to the console by cutting and jumpering JP4-JP5. If so, CTS1* must be asserted by the console, or the system will appear inoperative since the monitor will be unable to transmit to the console.

## 4.5 Console I/O

The monitor prompt is "n>" where "n" represents the currently selected memory bank (see the "B" command).

Commands consist of a command code, followed by 0 to 4 parameters and terminated by a carriage return. Parameters are separated by a " " or a "," and leading blanks are ignored.

Numeric parameters are assumed to be in HEX (with the exception of the #-of-tracks parameter for the "K" disk format command and the baudrate specifier for the "E" terminal emulation command). Leading "0"'s are ignored. For commands which require 16 bit parameters, the last four hex digits are recognized. For commands which require 8 bit parameters, the last two hex digits are recognized.

Console entry may be upper or lower case.

For example, the following command lines have the same result.

```
Ø>D Ø F            ;Display memory from ØØØØ to ØØØF

Ø>d Ø f

Ø>DO,F

Ø>D12340000 5678000F

Ø>D     Ø,     F
```

## 4.6 Commands

The following table is a summary of the monitor commands; a complete description of each command follows.

| | |
|---|---|
| ASCII Table | **A** |
| Bank Select | **B**bank# (bank# = Ø to 3) |
| CopyDisk | **C**source-drive# destination-drive# |
| Display Memory | **D**[start-addr] [end-addr] |
| Emulate Terminal | **E**[baudrate] |
| Fill Memory | **F**start-addr end-addr data8 |
| Goto Program | **G**[go-addr] or GB break-addr [go-addr] |
| Hexmath | **H**data16 data16 |
| Input Port | **I**port-addr |
| Klean Disk | **K**drive# #-of-tracks (4Ø or 8Ø) |
| Move Memory | **M**start-addr end-addr destination-addr |
| New Command | **N**[command#] (command# = Ø-FF hex) |
| Output Port | **O**port-addr data8 |
| Printer Select | **P** |
| Query Memory | **Q**data8 [data8] [data8] [data8] |
| Read Disk | **R**drive#,dest-addr,start-sect#,#-of-sects |
| Set Memory | **S**start-addr |
| Test System | **T**device |
| Upload Hex File | **U**[C] |
| Verify Memory | **V**start-addr end-addr destination-addr |
| Write Disk | **W**drive#,start-addr,start-sec,#-of-secs |
| Examine CPU Regs | **X** |
| Yank I/O Regs | **Y** |
| Z-System Boot | **Z**[drive#] |

Figure 4.6-1 MONITOR COMMAND SUMMARY

### 4.6.1 ASCII Table - >A

Prints an ASCII code table.

### 4.6.2 Bank Select - >Bbank# (bank# = Ø to 3)

Selects a 64K memory bank. The currently selected bank is indicated in the command prompt. All commands which reference memory operate on the currently selected bank. The bank offset is only applied to logical addresses between 2ØØØhex and EFFFhex. Addresses Ø-1FFFhex always refernce the monitor (based at physical address ØØØØØhex), while addresses FØØØ-FFFFhex always reference the monitor data/stack area (physical addresses 4FØØØ-4FFFFhex) This memory management scheme allows 56K of banked memory with a 4K common area at the bottom of ROM memory and a 4K common area at the top of RAM memory.

### 4.6.3 CopyDisk - >Csource-drive# destination-drive#

Source-drive# and destination-drive# can take the values Ø to 3 and correspond to the physical drive address (the jumper on the drive). Systems with 256K bytes RAM can perform single drive copies (i.e., CØ Ø), in which case a "swap disk" prompt will be issued. The Copy command requires that both disks be of the same type (i.e., 4Ø or 8Ø track).

### 4.6.4 Display Memory - >D[start-addr] [end-addr]

Displays memory in hex and ASCII. If start-addr is omitted, the display will start with the address following the last invocation's end address (or address Ø if the first invocation). If end-addr is omitted, the display will end 8Øhex bytes following the start address.

### 4.6.5 Emulate Terminal - >E[baudrate]

Console keyboard input is echoed to the AUX: RS-232 output, and AUX: RS-232 input is echoed on the console display. Baud rate is specified as 15Ø, 3ØØ, 6ØØ, 12ØØ, 24ØØ, 48ØØ, 96ØØ, 192ØØ, or 384ØØ. Note - the baud rate option only works if the CPU is operating at the standard (6.144 MHz) clock rate. If you have a non-standard clock rate configuration, use the "O" Output Port command to directly reprogram the baud rate or modify your monitor (EP)ROM (see the monitor ROM modification section). The command prompts for a key which will exit the terminal mode and return to the monitor. At system start up, the AUX: port is initialized to 192ØØ baud, 8 data bits, 1 stop bit and no parity.

The AUX: port supports RTSØ* modem control output and the DCDØ* and CTSØ* modem control inputs (inputs and output are relative to the HD64180). The RTSØ* output is always asserted. The CTSØ* and DCDØ* inputs must be asserted by the connected device, or grounded on the board.

### 4.6.6 Fill Memory - >Fstart-addr end-addr data8

Memory from start-addr to end-addr is filled with data8 (8 bit data, Ø-ffhex). Care should be taken to avoid writing to the monitor program (Ø-1FFFhex) and stack/data (FFØØ-FFFFhex) areas.

### 4.6.7 Goto Program - >G[go-addr] or GB break-addr [go-addr]

CPU registers are initialized and program execution continues at go-addr. The GB format sets a breakpoint at break-addr. If go-addr is omitted, program execution continues at the saved PC (see the "X" command).

### 4.6.8 Hexmath - >Hdata16 data16

Prints the 2Ø bit sum and difference, and the 32 bit product of the two arguments. (data16 is Ø-FFFFhex)

### 4.6.9 Input Port - >Iport-addr

Prints the 8 bit data input from port-addr in hex and binary.

### 4.6.1Ø Klean (Format) Disk - >Kdrive# #-of-tracks (4Ø or 8Ø)

Asks for confirmation and then formats and verifies the specified disk.

### 4.6.11 Move Memory - >Mstart-addr end-addr destination-addr

Moves the memory block between start-addr and end-addr to destination-addr. Care should be taken to avoid writing to the monitor program (Ø-1FFFhex) and stack/data (FFØØ-FFFFhex) areas.

### 4.6.12 New Command - >N[command#] (command# = Ø-FF hex)

Loads the A register with the command# (Ø if no command# specified). If an extended (EP)ROM is installed (16KB or 32KB), the extended ROM space (2ØØØhex to 4ØØØhex or 8ØØØhex) is enabled. A CALL to address 2ØØØhex is executed. To return to the monitor, the new command should terminate with a RET instruction. If extended (EP)ROM was enabled, it is disabled upon return.

### 4.6.13 Output Port - >Oport-addr data8

Data8 byte is output to port-addr. Note that most aspects of the HD64180 operation (baud rates, data format, wait states, etc.) can be configured by output to chip registers.

### 4.6.14 Printer Select - >P

Toggles the printer selection between the Centronics parallel port and the Auxillary serial (RS-232) port. The initial value is Centronics.

### 4.6.15 Query (Search) Memory - >Qdata8 [data8] [data8] [data8]

Searches memory for the memory pattern comprised of one to four bytes and prints addresses at which the pattern is found.

### 4.6.16 Read Disk - >Rdrive#,dest-addr,start-sector#,#-of-sectors

Reads the specified sectors from drive# into memory at dest-addr. The first sector on the disk is "1" and the last sector on the disk is 190hex (400 decimal) and 320hex (800 decimal) for 40 and 80 track drives, respectively.

### 4.6.17 Set Memory - >Sstart-addr

Displays the memory contents at start-addr and allows new data to be entered. Entering carriage return proceeds to the next address. Entering "." terminates the command.

### 4.6.18 Test System - >Tdevice

Tests various system devices. TA specifies the Auxillary serial (RS-232) port which prompts for input or output test. If input, serial input is echoed on the console. For input, the DCD0* AUX: modem control input must either be asserted by the connected device or grounded with the jumper on the board - otherwise an error message is printed. If output, a test pattern is transmitted. For output, the CTS0* AUX: modem control input must either be asserted by the connected device or grounded with the jumper on the board - otherwise an error message is printed. TC specifies the Centronics parallel port to which a test pattern is transmitted. If the printer doesn't respond in a reasonable time (approx. 5 seconds) an error message is printed. TD specifies a disk seek and read (non-destructive) test. If a bad sector is found, the disk test is aborted and the contents of the 9266 status registers (identifying the type of error, track, head, sector, etc.) are displayed. If no device is specified, a memory test is performed. The memory test is non-destructive and will print a "." after each 256KB pass.

All tests can be terminated with CTL-X or CTL-C.

### 4.6.19 Upload Hex File - >U[C]

An Intel format hex file is uploaded. If the [C] option is specified, the data is uploaded from the Console serial port, otherwise the data is uploaded from the Auxillary serial port. Note that upload termination requires reception of a CTL-X (1A hex). Thus, if the PIP command were used to download a .HEX file for a CP/M system, the [H] option should be specified. The command terminates by printing the address of the last byte loaded.

### 4.6.20 Verify Memory - >Vstart-addr end-addr destination-addr

The contents of the memory block from start-addr to end-addr is compared with the block at destination-addr. When source and destination data differ, the addresses and data values are printed.

### 4.6.21 Write Disk - >Wdrive#,start-addr,start-sector,#-of-sectors

Writes the specified sectors to disk from memory at start-addr. The first sector on the disk is "1" and the last sector on the disk is 190hex (400 decimal) and 320hex (800 decimal) for 40 and 80 track drives respectively.

### 4.6.22 Examine CPU Registers - >X

Displays the main and alternate CPU registers and prompts for modification of the main registers. Entering a carriage return proceeds to the next register while entering a "." terminates the command.

### 4.6.23 Yank I/O Registers - >Y

Displays the HD64180 on-chip I/O register contents.

### 4.6.24 Z-System Boot - >Z[drive#]

Boots the Z-System DOS (or other suitably configured operating system) from the specified drive.

## 4.7 Error Messages

### 4.7.1 FDC Error

Displayed at RESET if the Monitor cannot correctly initialize the 9266 FDC. This indicates a hardware fault such as a bad FDC, bad address decoder, etc. Use the I and O commands to verify FDC input/output operations. The FDC status port is 80H, the data port is 81H.

### 4.7.2 Disk R/W Error

Following this message, the contents of the 9266 FDC status registers are printed along with id information (track, head, sector). Typically, this indicates the disk is not formatted correctly (i.e., non-"native" format), but may also result from faulty media or hardware. Corrective action includes retrying the operation, reformatting the disk, resetting the system, testing memory, swapping disk drives, etc.

### 4.7.3 Disk Seek Error

Explanation and correction - see above.

### 4.7.4 Disk Not Ready

The drive is not loaded with the door closed and the motor on. Check to insure your drive provides a "READY" signal. If not, connect the "No Drive Ready" jumper on the board. Check to insure your drive responds to the "Motor On" signal. If not, either connect the "Motor (always) On" jumper on the board, or jumper your drive to achieve the same effect. Note that some drives are typically jumpered to enable the stepper and spindle motor based on drive select. However, depending on the motor control circuit, these drives may not be able to tolerate the 9266 FDC "Scan" function which toggles drive select at high speed when the drive is otherwise idle. In this case, the drive must be jumpered so that the motors are always on.

### 4.7.5 Bad Command

An invalid command has been entered. Use the "?" command to see a list of available commands and their syntax.

### 4.7.6 Bad Parameter

An invalid parameter has been entered. Remember that most commands require hex parameters. Use the "?" command.

### 4.7.7 Not Enough Parameters

The command requires more parameters than were entered. Use the "?" command.

### 4.7.8 Invalid Interrupt

The HD64180 has received an internal or external interrupt for which no interrupt handler is provided. The only interrupts the HD64180 recognizes are external interrupts INT1* (Centronics interface) and INT2* (Disk interface). External interrupts INT0* and NMI* may be shorting to ground. Internal interrupts (DMAC, timers, etc.) may be inappropriately enable by a program crash or incorrect use of the "O" output port command.

### 4.7.9 Bad Opcode Trap

The HD64180 has encountered an invalid opcode. This may be the result of a user program crash - confirm your program. This may also occur due to slow or faulty memory - perform a memory test. In this regard, note that the HD64180 has stricter access time requirements for opcode fetch than other read/write cycles and this is not checked by the memory test. Try reprogramming the on-chip wait state generator (DCNTL register, I/O address 32hex) or using faster memory chips.

### 4.7.10 CTS0* HIGH

Displayed during the >TA (Test AUX: port) output command if the CTS0* modem control input is not LOW at the HD64180. Check the connected device and cable.

### 4.7.11 DCD0* HIGH

Displayed during the >TA (Test AUX: port) input command if the DCD0* modem control input is not LOW at the HD64180. Check the connected device and cable.

### 4.7.12 No ACK*

Displayed during the >TC (Test CEN: port) command if the printer does not return ACK* within about 5 seconds after a byte is sent to the printer. Check the printer state (on line/off line, etc.) and cable.

## 4.8 Disk Format

The monitor "R" and "W" (Read and Write disk) commands treat the diskette as containing "virtual" 1KB sectors. Actually the format for 40 and 80 track drives is defined as follows:

40 Track Double Sided, Double Density

The disk contains 40 cylinders, each consisting of two sides/tracks. Each track is made up of 10 sectors of 512 bytes. Thus, total formatted capacity is 512 bytes x 10 sectors x 2 sides x 40 cylinders = 400K bytes. Sector numbers start at 11hex, with an interleave factor of two. Actual order on the track is (hex) 11, 16, 12, 17, 13, 18, 14, 19, 15, 1A. During formatting the 9266 FDC GPL (gap length) parameter is 24 decimal, while GPL is 14 decimal during normal read/write operation.

80 Track Double Sided, Double Density

The disk contains 80 cylinders, each consisting of two sides/tracks. Each track is made up of 5 sectors of 1K bytes. Thus, total formatted capacity is 1K bytes x 5 sectors x 2 sides x 80 cylinders = 800K bytes. Sector numbers start at 11hex, with an interleave factor of two. Actual order on the track is (hex) 11, 14, 12, 15, 13. During formatting the 9266 FDC GPL (gap length) parameter is 99 decimal, while GPL is 14 decimal during normal read/write operation.

## 4.9 Monitor ROM Modification

The monitor is organized to allow easy modification to support non-standard CPU clock rates (standard is 6.144 MHz), unique console and auxillary port baud rates and data formats and optimized disk timing parameters. Also the monitor can be extended by using 27128 or 27256 devices in conjuction with the "N" new command. **Never reprogram or erase your original monitor EPROM!** Instead, read it on another system, save the object file, make a copy of the object file, modify that copy, and then burn a new (EP)ROM.

## 4.10 Key Variable Block

Nine bytes, located starting at address 35hex, in the monitor allow changing a number of operating parameters as defined below.

### 4.10.1 STARTBYTE - Address 35hex, default value = FFhex

Normally FFhex, the upper and lower nybbles of STARTBYTE control two aspects of the monitor start sequence.

Changing the upper nybble of STARTBYTE allows disabling the console baud rate autosense routine if either a non-standard baud rate (i.e., not 300, 1200, 9600 or 19,200) console baud rate is required or if a non-standard (6.144 MHz is standard) CPU clock is used. In either case, change the upper nybble of STARTBYTE to 0. Note that STARTBYTE is related to CNTLB1 byte (see below). If STARTBYTE is not changed (i.e., equals Fx hex to use autosense) then CTTLB1 default value of 1 must also not be changed.

Changing the lower nybble of STARTBYTE to Ø allows disabling the DOS autoboot function so that the monitor will always sign on independent of whether a disk is loaded. This is especially useful if the "No Drive Ready" jumper is installed on the board to prevent hanging in the DOS boot loop if a disk is not loaded.

| STARTBYTE | Meaning |
|---|---|
| FFh | Baud rate autosense, DOS autoboot (default value) |
| ØFh | Fixed baud rate (based on CNTLB1 value), DOS autoboot |
| FØh | Baud rate autosense, DOS autoboot disabled |
| ØØh | Fixed baud rate (based on CNTLB1 value), DOS autoboot disabled |

### 4.1Ø.2 CNTLAØ - Address 36hex, default value = 65hex

Contains the value programmed into the HD64180 CNTLAØ register which defines, among other things, the AUX: port data format (i.e., # data bits, stop bits, parity, etc.). The default value also asserts RTSØ* low.

### 4.1Ø.3 CNTLA1 - Address 37hex, default value = 75hex

Contains the value programmed into the HD64180 CNTLA1 register which defines, among other things, the CON: port data format (i.e., # data bits, stop bits, parity, etc.).

### 4.1Ø.4 CNTLBØ - Address 38hex, default value = 1

Contains the value programmed into the HD64180 CNTLBØ register which defines, among other things, the AUX: port baud rate. Note that the "E" emulate terminal command allows the baud rate to be set from the console. However, the "E" baud rate function only works when the CPU operates at the standard (6.144 MHz) clock rate.

### 4.1Ø.5 CNTLB1 - Address 39hex, default value = 1

Contains the value programmed into the the HD64180 CNTLB1 register which defines, among other things, the CON: port baud rate. If your console does not operate at the standard baud rates (3ØØ, 12ØØ, 96ØØ, 19,2ØØ are standard) or the CPU clock is non-standard (6.144 MHz is standard) both STARTBYTE (see above) and ZCNTLB1 must be changed. STARTBYTE must be set to Ø (disables the baud rate autosense routine) and CNTLB1 set to the appropriate value given your console baud rate and CPU clock rate.

4.10.6 STAT0 - Address 3Bhex, default value = 0

Contains the value programmed into the HD64180 STAT0 register which enable or disables AUX: port interrupts, which are normally disabled.

4.10.7 STAT1 - Address 3Chex, default value = 4

Contains the value programmed into the HD64180 STAT0 register which enable or disables CON: port interrupts as well as enabling or disabling the CON: port CTS1* modem control input. CON: port interrupts should normally be disabled and the default value enables the CTS1* function. Thus, CTS1* must be grounded on the board (default case) or asserted by the console if connected.

4.10.8 DCNTL - Address 3Dhex, default value = 7Chex

Contains the value programmed into the HD64180 DCNTL register which defines, among other things, the number of memory and I/O wait states generated by the HD64180 on-chip wait state generator. Normally, the least significant four bits of this value should not change since they define DMA parameters associated with basic disk operation. Change the upper four bits to account for faster or slower CPU clock rate and/or memory-I/O devices.

4.10.9 RCR - Address 3Ehex, default value = 82hex

Contains the value programmed into the HD64180 RCNTL register which defines the interval and duration of HD64180 generated DRAM refresh cycles. Normally, this value needs to be changed only if the CPU clock rate is reduced below 3.072 MHz.

4.10.10 SPCF1 - Address 3Fhex, default = 9Fhex

Contains the value used as the first parameter of the 9266 FDC SPECIFY command which defines the step rate and head unload time for the floppy disk drive. This can be changed if your disk has higher performance than the conservative default value. Note that a DOS BIOS can reSPECIFY in a "soft" manner, so it may be wise to leave the conservative default in the ROM to allow for easy connection of drives with poor or unknown performance characteristics.

4.10.11 SPCF2 - Address 40hex, default value = 28hex

Contains the value used as the second parameter of the 9266 FDC SPECIFY command which defines the head load time and data mode for the floppy disk drive. As above, this can be changed, but change may not be required since the DOS BIOS can reSPECIFY. Note, the data mode must be "DMA".

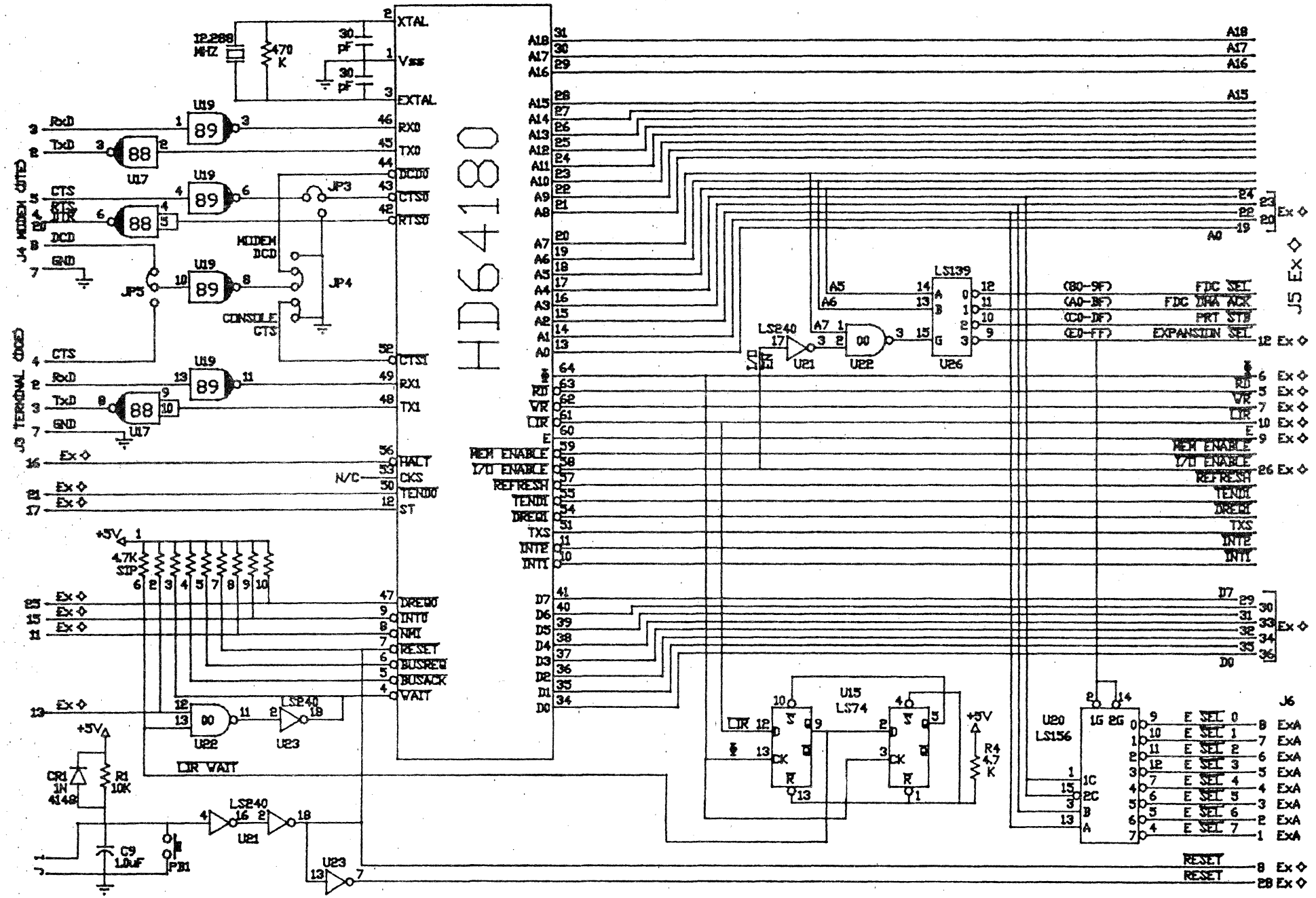
## 4.11 The "N" NEW Command

The monitor, which requires 8K bytes of (EP)ROM can co-reside with other system software in a larger 16K byte or 32K byte (EP)ROM. The extra 8K bytes or 24K bytes can contain additional software such as BASIC, Forth, a DOS, or ???

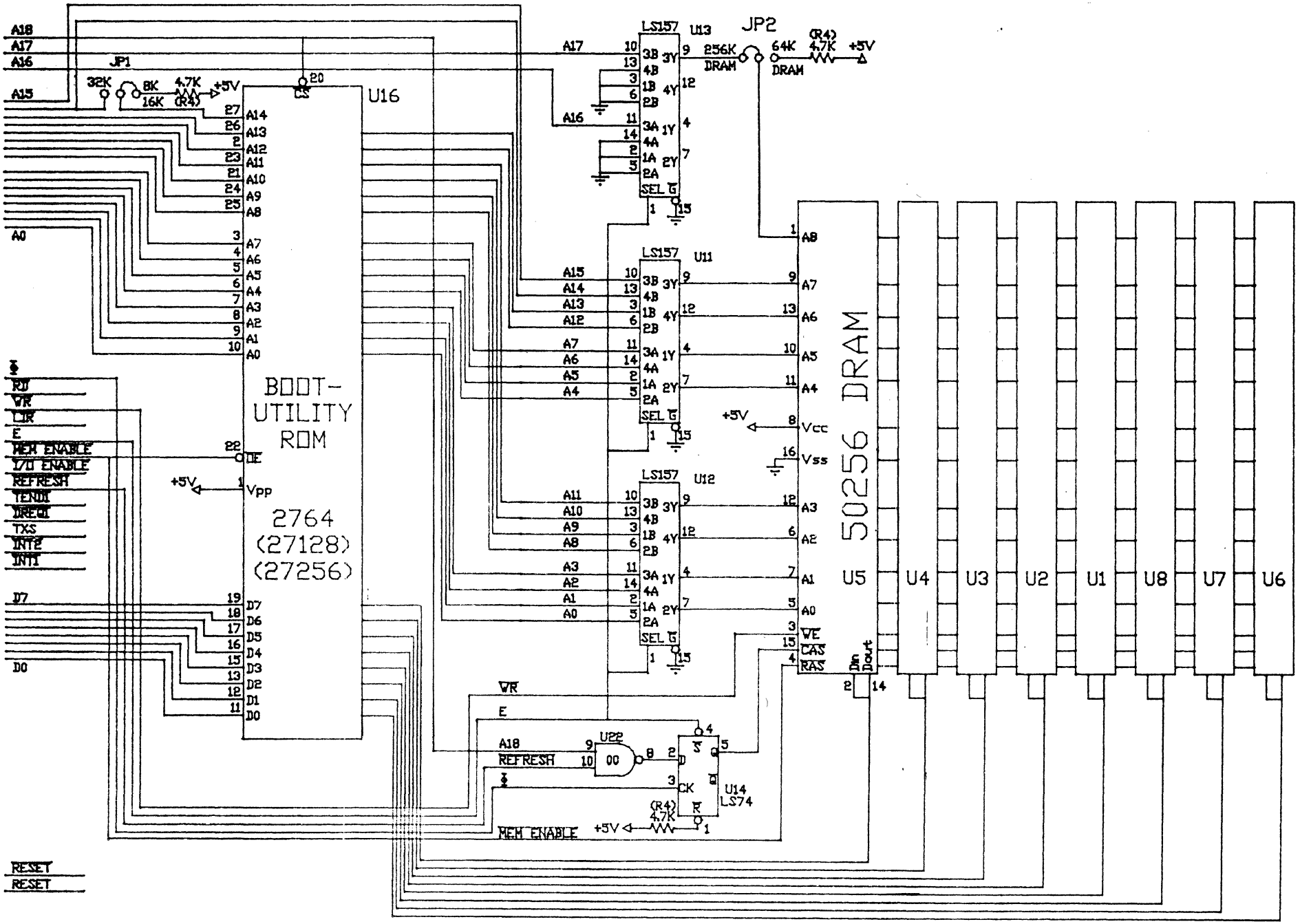
At RESET, the monitor determines how big the installed ROM is. The "N" NEW command can then "phantom" in an extended (EP)ROM. Also, an optional parameter on the "N" command line is loaded into the Accumulator (A) for passage to the extended routines. When the extended routine terminates with a RETurn instruction, the monitor regains control and "phantoms" out the extended (EP)ROM to allow access to overlayed RAM. Note that this return mechanism requires the extended routine to save the return address if it sets up a different stack. Also, the extended routine should avoid writing to the physical address area 4FF00-4FFFFhex since this area contains monitor data structures.

To implement an extended (EP)ROM simply requires assembling your routines to start at the "end" (address 2000hex) of the monitor. Developing the code is made easier by the fact that, with only the 8K byte monitor installed, the "N" command will jump to address 2000hex without performing the "phantom" function. After the code is tested in RAM, burn it and the monitor in a new (EP)ROM, install it (remember to adjust the ROM size on the SB180 system board), and operation will be exactly the same as during debug.

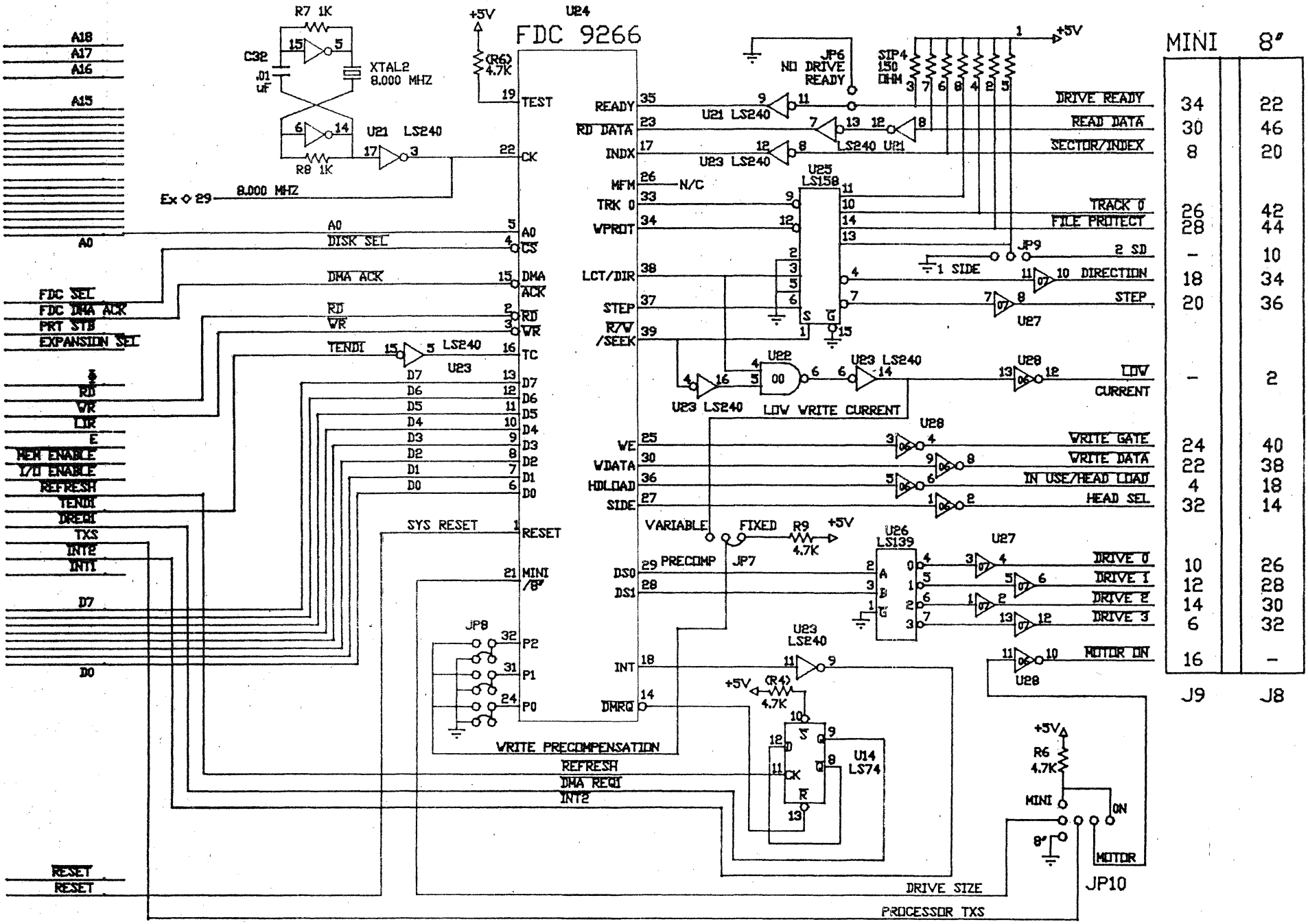
For further information on the ROM monitor, consult the monitor source code on the appropriate system disk.




SB180 COMPUTER PG 1 OF 4 (PROCESSOR) COPYRIGHT 1985, CIARCIA'S CIRCUIT CELLAR 7/23/85

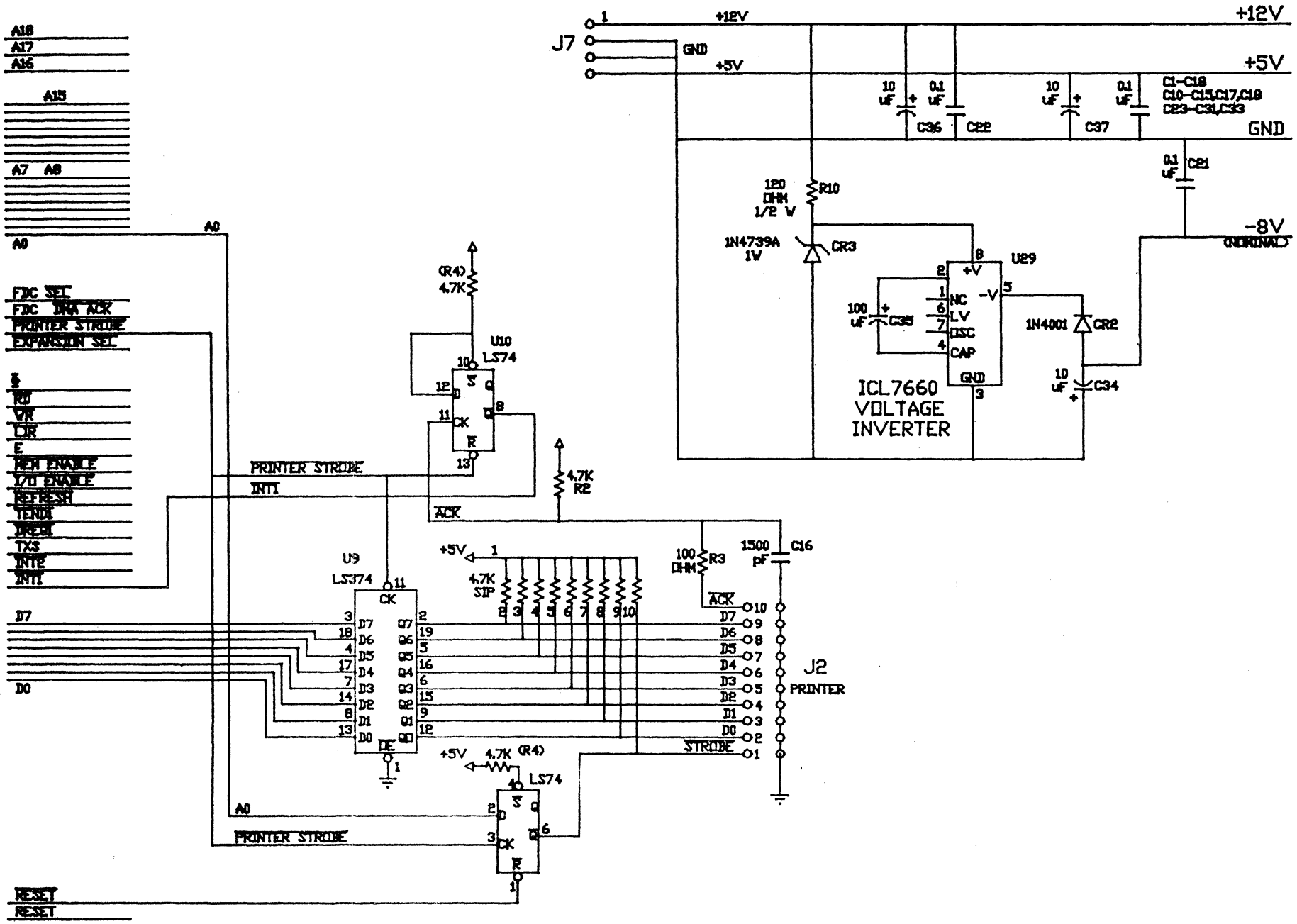



SB180 COMPUTER PG 2 OF 4 (MEMORY) COPYRIGHT 1985, CIARCIA'S CIRCUIT CELLAR 6/6/85






SB180 COMPUTER PG 3 OF 4 COPYRIGHT 1985, CIARCIA'S CIRCUIT CELLAR 6/6/85




SB180 PG 4 OF 4 (PRINTER & PS) COPYRIGHT 1985, CIARCIA'S CIRCUIT CELLAR 6/6/85

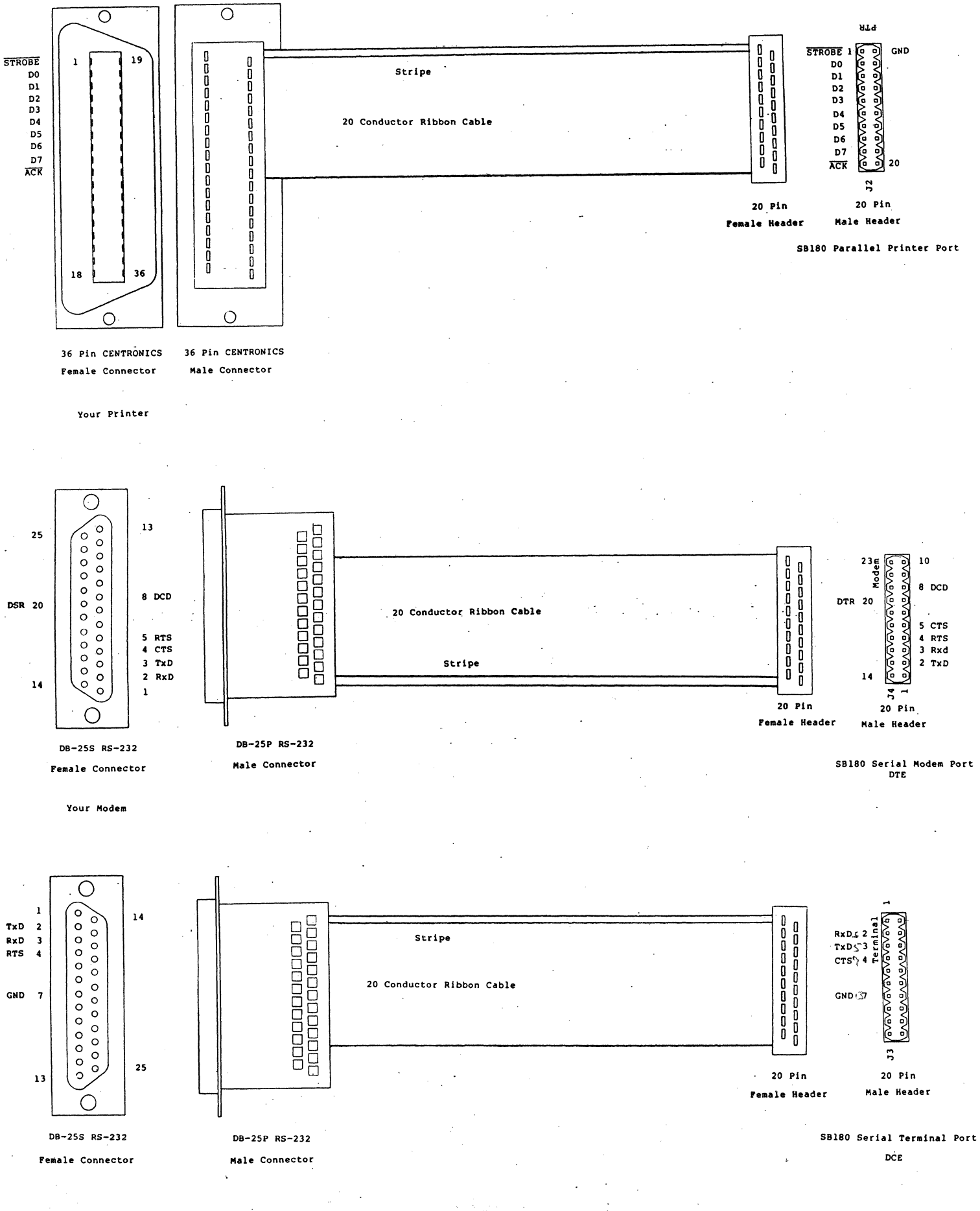



Figure 5.1 Sample Cable Assemblies

## COMPONENT LIST

## SB180 - SINGLE BOARD 64180 SYSTEM

### CAPACITORS

| | |
|---|---|
| C1-C8 | .1 MFD 50V MONOLITHIC |
| C9 | 1 MFD 25V ELECTROLYTIC |
| C10-C15 | .1 MFD 50V MONOLITHIC |
| C16 | 1500 PFD 50V MONOLITHIC |
| C17,C18 | .1 MFD 50V MONOLITHIC |
| C19,C20 | 27 PFD 50V MONOLITYIC |
| C21-C31 | .1 MFD 50V MONOLITHIC |
| C32 | .01 MFD 50V MONOLITHIC |
| C33 | .1 MFD 50V MONOLITHIC |
| C34 | 10 MFD 25V ELECTROLYTIC |
| C35 | 100 MFD 25V ELECTROLYTIC |
| C36,C37 | 10 MFD 25V ELECTROLYTIC |

### DIODES

| | |
|---|---|
| CR1 | 1N4148A, SIL, FAST SWITCH |
| CR2 | 1N4001, SIL, BLOCKING |
| CR3 | 1N4739A, SIL, ZENER, 9.1V, 1W |

### INTEGRATED CIRCUITS

| | |
|---|---|
| IC1-IC8 | 41256, DRAM, 256K, (150NS OR 90NS) |
| IC9 | 74LS374, OCTAL LATCH, TRISTATE |
| U10 | 74LS74, DLATCH, DUAL |
| U11-U13 | 74LS157, MUX, QUAD, NONINVERT |
| U14,U15 | 74LS74, DLATCH, DUAL |
| U16 | ROM, (8K STANDARD)<br>2764 8K X 8 OR<br>27128 16K X 8 OR<br>27256 32K X 8 |
| U17 | 1488, TRANSMITTER, LEVEL SHIFT |
| U18 | 64180, MICROCOMPUTER |
| U19 | 1489, RECEIVER, LEVEL SHIFT |
| U20 (OPTION) | 74LS156, OCTAL DECODE (DUAL 1 OF 4) |
| U21 | 74LS240, HEX INVERT |
| U22 | 74LS00, QUAD NAND |
| U23 | 74LS240, HEX INVERT |
| U24 | FDC9266, FLOPPY DISK CONTROLLER |
| U25 | 74LS158, QUAD MUX, INVERT |
| U26 | 74LS139, 1 OF 4 DECODE, DUAL |
| U27 | 7407, HEX BUFFER, NONINVERT |
| U28 | 7406, HEX BUFFER, INVERT |
| U29 | ICL7660, VOLTAGE INVERTER |

### RESISTORS

| | |
|---|---|
| R1 | 10 K, 1/4 W, 5% |
| R2 | 4.7 K, 1/4 W, 5% |
| R3 | 100 OHM, 1/4 W, 5% |
| R4 | 4.7 K, 1/4 W, 5% |
| R5 | 470 K, 1/4 W, 5% |
| R6 | 4.7 K, 1/4 W, 5% |
| R7,R8 | 1.0 K, 1/4 W, 5% |
| R9 | 4.7 K, 1/4 W, 5% |
| R10 | 120 OHM, 1/2 W, 5% |
| SIP1-SIP3 | 4.7K, 9 ELEMENT, PIN 1 COMMON |
| SIP4 | 330 OHM, 7 ELEMENT, PIN 1 COMMON |

### MISCELLANEOUS

| | |
|---|---|
| J1 | 1 X 2, .1 CTR, (HARDWIRE) |
| J2-J4 | 2 X 10, .1 CTR, ST, .230 L |
| J5 | 2 X 20, .1 CTR, ST, .320 L |
| J6 (OPTION) | 1 X 8, .1 CTR, ST, .230 L |
| J7 | 1 X 4, .1 CTR, RT, .320 L |
| J8 (OPTION) | 2 X 25, .1 CTR, SHROUDED, ST |
| J9 | 2 X 17, .1 CTR, SHROUDED, RIGHT |
| JP1-JP3 (OPT) | 1 X 3, .1 CTR, ST, .230 L |
| JP4 (OPTION) | 1 X 5, .1 CTR, ST, .230 L |
| JP5 (OPTION) | 1 X 3, .1 CTR, ST, .230 L |
| JP6 (OPTION) | 1 X 2, .1 CTR, ST, .230 L |
| JP7 (OPTION) | 1 X 3, .1 CTR, ST, .230 L |
| JP8 (OPTION) | 2 X 6, .1 CTR, ST, .230 L |
| JP9 (OPTION) | 1 X 3, .1 CTR, ST, .230 L |
| JP10 | 1 X 3(2), "T", .1 CTR, ST, .230 L |
| JUMPER (2) | 1 X 2 HEADER JUMPER |
| PB1 | PUSHBUTTON, (RESET) |
| XTAL1 | CRYSTAL, 12.2880 MHZ |
| XTAL2 | CRYSTAL, 8.0000 MHZ |

### SOCKETS

| | |
|---|---|
| 1 | 8 PIN |
| 8 | 14 PIN |
| 14 | 16 PIN |
| 3 | 20 PIN |
| 1 | 28 PIN |
| 1 | 40 PIN |
| 1 | 64 PIN (SHRINK DIP) |

# Z R D O S

Version 1.0

**Echelon Z-System Disk Operating System**

**PROGRAMMER'S GUIDE**

by

Dennis L. Wright

1 January 1985

ZRDOS, its utilities, and documentation files are Copyright 1984 and 1985 by Dennis L. Wright and Echelon, Inc. ZCPR3 is Copyright 1984 by Richard Conn and Echelon, Inc. CP/M and MP/M are registered trademarks of Digital Research. No part of this guide may be reproduced in any way or by any means without prior written permission from Echelon, Inc.

ZRDOS is a Z80 coded CP/M 2.2 compatible Disk Operating System. Use of Z80 code allows addition of many new features. This document explains these features and differences between ZRDOS Version 1.0 and CP/M 2.2.

# ZRDOS PLUS

## ADDENDUM

## TO ZRDOS VERSION 1.0 PROGRAMMER'S GUIDE

ZRDOS Plus is the re-entrant version of ZRDOS Version 1.0. All features and function calls are identical to those outlined in the ZRDOS Version 1.0 Programmer's Guide with the following addition.

Programs that intercept BIOS calls from ZRDOS Plus can be written to make calls to ZRDOS without destroying the original DOS callers pointers and parameters.

---

Re-entrance can be accomplished by first saving the current ZRDOS Plus buffers. This is done by copying the ZRDOS Plus buffers to a user assigned save buffer area of 147 bytes. Once the DOS data has been saved the user program is free to make any DOS calls necessary. Before returning to the original DOS caller, the ZRDOS Plus buffers must be restored. The beginning of the ZRDOS Plus parameter-Buffer area is located at ZRDOS Plus base + 5 (ZRDOS Plus base is the address specified in system page zero, hex location 06) and is 147 bytes in length.

The main purpose of making ZRDOS re-entrant is to allow the efficient programming of ZCPR3 I/O Packages, packages (modules of 1.5k-bytes length) that redirect Device Record (Console, List, Reader and Punch) input and output to and from disk files. Echelon, Inc. offers several IOPs that make use of this feature.

## EXAMPLE METHOD OF SAVING AND RESTORING ZRDOS PLUS BUFFERS

The following routines demonstrate a method that can be used to save and restore the ZRDOS Plus buffers to allow re-entrant calls to ZRDOS Plus.

```
;
DOS     EQU     5
BUFOFF  EQU     5       ; Offset from beginning of ZRDOS Plus to
                        ; internal dos buffers.
;
; This routine gets the address of the ZRDOS Plus parameter buffer.
;
GETBUF: LHLD    06      ; Get dos address.
        LXI     D,BUFOFF ; Add offset to ZRDOS Plus internal buffer.
        DAD     D
        SHLD    DOSBUF  ; Save as dosbuf pointer.
        RET
;
; This routine saves ZRDOS Plus parameters to allow re-entry.
;
SAVDOS: LHLD    DOSBUF  ; Save ZRDOS Plus parameter buffer.
        LXI     D,DOSSAV
        CALL    MOVIT
        MVI     C,47    ; Function 47, get current dma address.
        CALL    DOS
        SHLD    CURDMA  ; Save it.
        RET
;
; This routine restores original ZRDOS Plus parameters.
;
RSTDOS: LDED    CURDMA  ; Restore dma address
        MVI     C,26    ; Function 26, set dma address.
        CALL    DOS
        LXI     H,DOSSAV ; Restore ZRDOS Plus parm buffer.
        LDED    DOSBUF
MOVIT:  LXI     B,147   ; Move 147 bytes.
        LDIR
        RET
;
CURDMA: DW      0       ; Save area for current DMA address.
DOSBUF: DW      0       ; Save area for pointer to ZRDOS Plus parms.
DOSSAV: DS      147     ; Save area for ZRDOS Plus parm buffer.
;
```

# Z R D O S

Version 1.0

**Echelon Z-System Disk Operating System**

**PROGRAMMER'S GUIDE**

by

Dennis L. Wright

# T A B L E  O F  C O N T E N T S

**1. DIFFERENCES FROM THE STANDARD CP/M CCP..... 1**
1.1. ZCPR3 Utilities and Features..... 2

**2. DIFFERENCES FROM CP/M 2.2 BDOS..... 3**
2.1. Disk Change..... 3
2.2. Read Only Disk Status..... 3
2.3. Read Console Buffer..... 4
2.4. File Archiving..... 4
2.5. Wheel Protection..... 4
2.6. Error Messages..... 5

**3. ZRDOS EXTENDED FUNCTION CALLS..... 6**
3.1. Function 47: Get Current DMA Address..... 6
3.2. Function 48: Return ZRDOS Version Number..... 6
3.3. Function 50: Set Warm Boot Trap..... 6
3.4. Function 52: Reset Warm Boot Trap..... 6

**4. ZRDOS VERSION 1.0 FUNCTION CALLS..... 7**
4.1. FUNCTION 0: SYSTEM RESET..... 8
4.2. FUNCTION 1: CONSOLE INPUT..... 8
4.3. FUNCTION 2: CONSOLE OUTPUT..... 8
4.4. FUNCTION 3: READER INPUT..... 9
4.5. FUNCTION 4: PUNCH OUTPUT..... 9
4.6. FUNCTION 5: LIST OUTPUT..... 10
4.7. FUNCTION 6: DIRECT CONSOLE I/O..... 10
4.8. FUNCTION 7: GET I/O BYTE..... 11
4.9. FUNCTION 8: SET I/O BYTE..... 11
4.10. FUNCTION 9: PRINT STRING..... 12
4.11. FUNCTION 10: READ CONSOLE BUFFER..... 12
4.12. FUNCTION 11: GET CONSOLE STATUS..... 13
4.13. FUNCTION 12: RETURN VERSION NUMBER..... 13
4.14. FUNCTION 13: RESET DISK SYSTEM..... 13
4.15. FUNCTION 14: SELECT DISK..... 14
4.16. FUNCTION 15: OPEN FILE..... 14
4.17. FUNCTION 16: CLOSE FILE..... 15
4.18. FUNCTION 17: SEARCH FOR FIRST..... 15
4.19. FUNCTION 18: SEARCH FOR NEXT..... 16
4.20. FUNCTION 19: DELETE FILE..... 16
4.21. FUNCTION 20: READ SEQUENTIAL..... 17
4.22. FUNCTION 21: WRITE SEQUENTIAL..... 17
4.23. FUNCTION 22: MAKE FILE..... 18
4.24. FUNCTION 23: RENAME FILE..... 18
4.25. FUNCTION 24: RETURN LOGIN VECTOR..... 19
4.26. FUNCTION 25: RETURN CURRENT DISK..... 19
4.27. FUNCTION 26: SET DMA ADDRESS..... 19
4.28. FUNCTION 27: GET ADDR(ALLOC)..... 20
4.29. FUNCTION 28: WRITE PROTECT DISK..... 20
4.30. FUNCTION 29: GET READ ONLY VECTOR..... 20
4.31. FUNCTION 30: SET FILE ATTRIBUTES..... 21
4.32. FUNCTION 31: GET ADDR(DISK PARMS)..... 22
4.33. FUNCTION 32: SET/GET USER CODE..... 22

4.34. FUNCTION 33: READ RANDOM..... 23
4.35. FUNCTION 34: WRITE RANDOM..... 24
4.36. FUNCTION 35: COMPUTE FILE SIZE..... 25
4.37. FUNCTION 36: SET RANDOM RECORD..... 25
4.38. FUNCTION 37: RESET DRIVE..... 26
4.39. FUNCTION 40: WRITE RANDOM WITH ZERO FILL..... 27
4.40. FUNCTION 47: RETURN CURRENT DMA ADDRESS..... 27
4.41. FUNCTION 48: RETURN ZRDOS VERSION NUMBER..... 27
4.42. FUNCTION 50: SET WARM BOOT TRAP..... 28
4.43. FUNCTION 52: RESET WARM BOOT TRAP..... 28

**5. DIRECTORY CODES..... 29**

**A. INDEX..... 30**

## L I S T O F F I G U R E S

4-1: Console Buffer Format..... 12
4-2: FCB format..... 17
4-3: Login Vector Bit Map..... 19
4-4: Read Only Vector Bit Map..... 20
4-5: File Attribute Format..... 21
4-6: Use of FCB bytes 'r0','r1' and 'r2'..... 24
4-7: Active Drive Vector Bit Map..... 26

5-1: Example Directory Sector..... 29

## L I S T O F T A B L E S

4-1: ZRDOS Version 1.0 Function Calls..... 7
4-2: IOBYTE Format..... 11

# 1. DIFFERENCES FROM THE STANDARD CP/M CCP

ZRDOS is a Z80 coded, CP/M 2.2 compatible disk operating system designed to be used with Echelon Z80/HD64180 Command Processor ZCPR3 written by Richard Conn and auto-install Z3-Dot-Com by Joseph Wright.

## 1.1. ZCPR3 Utilities and Features

ZRDOS is compatible with all of the ZCPR3 utilities. Named Directories, Redirectable I/O, and all other ZCPR3 features can be used with ZRDOS. For installation of these additional features please refer to original ZCPR3 documentation package.

# 2. DIFFERENCES FROM CP/M 2.2 BDOS

## 2.1. Disk Change

ZRDOS allows the changing of disks without the need of a warm boot. CHANGED DISKS WILL BE AUTOMATICALLY LOGGED IN.

NOTE: The above mentioned auto login will not occur if a file was open when the disk was changed and a write operation is attempted to that file on the new diskette instead the following error message will be printed:

Disk Changed Error On Drive B:

It should also be noted that the automatic logon may or may not be able to handle changes in density or number of sides. This depends on how your bios handles deblocking and double sided disks. However if the disk is of the same density and number of sides as the disk it is being swapped with there will be no problems.

---

## 2.2. Read Only Disk Status

Under ZRDOS a disk can only be set to R/O status by executing function call 28 (Protect Drive).

ZRDOS function call 37 (Reset Drive) is different in that it will only reset the Read Only bits for the drive(s) specified in the user passed drive map in the register pair (DE). With CP/M function 37 will also reset the bits for any drive not currently active.

Function call 13 (Reset Disk System) is different in that it will not reset drives that are set to Read Only but will instead reset the disk changed vector.

## 2.3. Read Console Buffer

The Read console buffer routine (Function 10) for ZRDOS is different in the following ways:

- o Rubout (DEL) is treated the same as a backspace.
- o The Control-R edit function is not implemented.

NOTE: In CP/M these are teletype oriented edit commands and were felt not to be desired in ZRDOS.

---

## 2.4. File Archiving

ZRDOS supports the use of the file archive attribute. The support of this feature is compatible with both CP/M 3.0 and MP/M. This bit when set indicates an archived file. That is a file which has not been altered. The bit can be set by using a Function 30 (Set File Attributes) function call. Any update to this file once the bit has been set will cause the bit to be reset. This can then be used by a copy utility to indicate the need to backup the file. The utility that backs up the file should then set the archive bit to indicate the file has been backed up.

---

## 2.5. Wheel Protection

ZRDOS uses the ZCPR3 wheel byte and a new file attribute bit to protect files from non-wheel users. This bit when set will write protect the file as long as the wheel byte is off. If the wheel byte is set the file is treated as a normal file.

If a non-wheel user attempts to change a wheel protected file the following error message will be displayed:

```
A0>File W/P Error on A:
```

## 2.6. Error Messages

| Error Number | ZRDOS | CP/M |
|---|---|---|
| 1 | Read Error On A: | Bdos Err On A:Bad Sector |
| 2 | Drive Select Error On A: | Bdos Err On A:Select |
| 3 | Disk R/O Error On A: | Bdos Err On A:R/O |
| 4 | File R/O Error On A: | Bdos Err On A:File R/O |
| 5 | Disk Changed Error On A: | n/a |
| 6 | File W/P Error On A: | n/a |

With ZRDOS all non-retryable errors jump directly to warm boot after the error message has been printed. Read Errors allow the user the option of retrying the operation by pressing any key but control-c or aborting by pressing control-c. Error numbers shown above are returned in the (A) register. The selected drive number is returned in register (E).

NOTE: If the warm boot trap (see FUNCTION 50) is set ZRDOS jumps directly to the warm boot vector and no error messages are displayed. (User program stack pointer not returned.)

CP/M handles the errors in the same manner except a key must be pressed before CP/M will return from any type of error and CP/M doesn't return an error or drive number. Nor does CP/M have a warm boot trap function.

# 3. ZRDOS EXTENDED FUNCTION CALLS

## 3.1. Function 47: Get Current DMA Address

This function will return the currently assigned DMA address in the register pair (HL).

## 3.2. Function 48: Return ZRDOS Version Number

This function works the same as CP/M function call 12 except the ZRDOS Version number is returned instead of the CP/M version number. To maintain CP/M compatibility ZRDOS will return version number 2.2 on a function 12 call. As with function 12 function 48 uses the register pair (HL) to return the version number. If user programs that use the extended ZRDOS functions are written this function should first be used to determine if the program is being run under ZRDOS.

## 3.3. Function 50: Set Warm Boot Trap

A new function call is provided that allows the user to set a trap on warm boot to a user specified address. The trap is set by executing a function 50 call with the trap address in the register pair (DE). The Warm boot jump address at location 001H is replaced with the user supplied trap address. Warm boots executed after the trap is set will cause a jump to the trap address. ZRDOS error messages are suppressed allowing the user to print his own error messages. As noted in the ZRDOS error section above errors detected by ZRDOS return an error number and the active drive number which the user can then use to determine how best to handle the error.

WARNING: Caution should be exercised when using this function as the results will be unpredictable if a program that has set the trap terminates without first resetting the trap. See FUNCTION 52 below.

## 3.4. Function 52: Reset Warm Boot Trap

This new function call will reset the warm boot trap set by function call 50. The trap is reset by executing a function call 52. The Real bios warm boot address is restored to location 0001H. If function call 50 is used in a user program a function call 52 should be executed before control is returned to the operating system.

# 4. ZRDOS VERSION 1.0 FUNCTION CALLS

| FUNCTION NUMBER | DESCRIPTION OF ZRDOS OPERATION PERFORMED |
|---|---|
| 0 | System Reset |
| 1 | Console Input |
| 2 | Console Output |
| 3 | Reader Input |
| 4 | Punch Output |
| 5 | List Output |
| 6 | Direct Console I/O |
| 7 | Get I/O Byte |
| 8 | Set I/O Byte |
| 9 | Print String |
| 10 | Read console Buffer |
| 11 | Get Console Status |
| 12 | Return Version Number (CP/M) |
| 13 | Reset Disk System |
| 14 | Select Disk |
| 15 | Open File |
| 16 | Close File |
| 17 | Search For First |
| 18 | Search For Next |
| 19 | Delete File |
| 20 | Read Sequential |
| 21 | Write Sequential |
| 22 | Make File |
| 23 | Rename File |
| 24 | Return Login Vector |
| 25 | Return Current Disk |
| 26 | Set DMA Address |
| 27 | Get Allocation Vector Address |
| 28 | Write Protect Disk |
| 29 | Get Read/Only Vector |
| 30 | Set File Attributes |
| 31 | Get Disk Parameter Block Address |
| 32 | Set/Get User Code |
| 33 | Read Random |
| 34 | Write Random |
| 35 | Compute File Size |
| 36 | Set Random Record |
| 37 | Reset Drive |
| 40 | Write Random With Zero Fill |
| 47 | Return Current DMA Address |
| 48 | Return Version number (ZRDOS1) |
| 50 | Set Warm Boot trap |
| 52 | Reset Warm Boot trap |

**Table 4-1: ZRDOS Version 1.0 Function Calls**

## 4.1. FUNCTION 0: SYSTEM RESET

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 00H | None |

Function to terminate program and reset the system. Same results as performing a jump to location 0000H. The disk system is reset; that is disks marked as changed are cleared and the directory check information is discarded.

## 4.2. FUNCTION 1: CONSOLE INPUT

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 01H | Register (A): ASCII Character |

Function to get character from console device. A byte from the device currently assigned to CON: is returned in register (A). The byte is echoed to the terminal. If no byte is ready at the time the call is made, the calling program is suspended until a byte becomes available.

## 4.3. FUNCTION 2: CONSOLE OUTPUT

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 01H<br>Register (E): ASCII Character | None |

Function to output character in register (E) to the device currently assigned to CON: and expand tabs if necessary. A Control-S (pause) and Control-P (echo to printer) console input test is also performed.

## 4.4. FUNCTION 3: READER INPUT

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 03H | Register (A): ASCII Character |

Function to get character from the reader device. The next byte from the device currently assigned to RDR: is returned in register (A). All 8 bits are returned. The calling program is suspended until a byte is ready.

## 4.5. FUNCTION 4: PUNCH OUTPUT

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 04H<br>Register (E): ASCII Character | None |

Function to output a character to the punch device. The byte in register (E) is sent to the device currently assigned to PUN:. The program is suspended until the device is ready to accept the byte.

## 4.6. FUNCTION 5: LIST OUTPUT

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 05H<br>Register (E): ASCII Character | None |

Function to output a character to the list device. The byte in register (E) is sent to the device currently assigned to LST:. The program is suspended until the device is ready to accept the byte.

## 4.7. FUNCTION 6: DIRECT CONSOLE I/O

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 06H<br>Register (E): 0FFH (input)<br>0FEH (status)<br>ASCII Char (output) | Register (A): ASCII Char<br>or<br>status |

Function to perform direct console i/o. If register (E) contains (FF) then this is an input request. If register (E) contains (FE) then this is a status request. Otherwise the character in register (E) will be sent to the device currently assigned to CON:. This request bypasses all control character checks.

## 4.8. FUNCTION 7: GET I/O BYTE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 07H | Register (A): I/O Byte Value |

Function to return the i/o byte. The current value of the IOBYTE (memory location 0003H) is returned in register (A).

## 4.9. FUNCTION 8: SET I/O BYTE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 08H<br>Register (E): I/O Byte Value | None |

Function to set the i/o byte. The value in (E) is set as the current IOBYTE (memory location 0003H). It changes control of the output direction immediately.

The I/O byte located at memory location 0003H is made up of the four fields shown in the following table:

| IOBYTE: | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|
| bit value | list (LST:) | punch (PUN:) | reader (RDR:) | console (CON:) |
| 00 | TTY: | TTY: | TTY: | TTY: |
| 01 | CRT: | PTP: | PTR: | CRT: |
| 10 | LPT: | UP1: | UR1: | BAT: |
| 11 | UL1: | UP2: | UR2: | UC1: |

**Table 4-2: IOBYTE Format**

## 4.10. FUNCTION 9: PRINT STRING

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 09H<br>Register (DE): String address | None |

Function to send the character string pointed to by (DE) to the device currently assigned to CON:. The printing of the string to the console device will continue until a '$' is encountered in the string. Console input control character checks are made and tabs are expanded.

## 4.11. FUNCTION 10: READ CONSOLE BUFFER

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 0AH<br>Register (DE): Buffer address | Console Characters in Buffer |

Function to execute a buffered read. ZRDOS notes the current cursor position as it knows it, then reads characters from the console device until a CR or LF is received, or until the maximum number of characters have been received.

ZRDOS unlike standard bdos treats the rubout (del) key the same as a backspace. Also the control-R function has been eliminated.

The form of the read buffer is as follows:

BASE = Address in (DE)

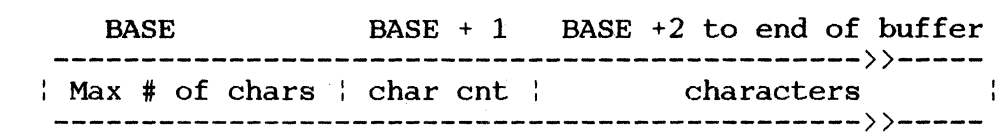



**Fig. 4-1: Console Buffer Format**

## 4.12. FUNCTION 11: GET CONSOLE STATUS

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 0BH | Register (A): Console Status |

Function to interrogate the console device. The device currently assigned to CON: is polled. If a byte is ready for input, a nonzero value is returned in register (A), otherwise 00H is returned.

## 4.13. FUNCTION 12: RETURN VERSION NUMBER

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 0CH | Register (HL): Version Number |

Function to return the current version number. Version number 2.2 is returned to maintain CP/M compatibility. Function 48 should be used to get the ZRDOS version number.

## 4.14. FUNCTION 13: RESET DISK SYSTEM

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 0DH | Register (A): 0FFH if the current default drive contains a file name beginning with a $ 00H if not. |

Function to reset the disk system. All active drives are reset to an unknown condition. Drive A is relogged in and the DMA address is reset to 80H. Unlike CP/M ZRDOS does not reset the read only vector of drives that have been set to read only status but instead resets the ZRDOS disk changed vector for any drives that are marked as changed.

## 4.15. FUNCTION 14: SELECT DISK

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 0EH<br>Register (E): Selected Disk | None |

Function to set the active disk number. Register (E) contains a number in the range 0 - 15, signifying disk A - P respectively. If the selected drive is not the current default drive, it is made the default drive. If it has not been selected since the last warm start or disk reset, its directory is scanned and new allocation and check vectors are built.

## 4.16. FUNCTION 15: OPEN FILE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 0FH<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to open a specified file. The drive code, if not zero, is used to select a drive. The directory is scanned for the first match to the filename and extent number in the fcb pointed to by the register pair (DE). The filename may contain wildcards. A matching directory entry is then copied into the specified fcb and register (A) is return with the directory sector location code 0 - 3 (See section on Directory Codes). If no match is found register (A) will contain 0FFH.

## 4.17. FUNCTION 16: CLOSE FILE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 10H<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to close a specified file. The filename and extent number in the fcb pointed to by the register pair (DE) are located in the directory. The file name may contain wildcards. If they are found, the record count and data map from the specified fcb are copied into the directory entry and the directory location code 0 - 3 is returned in register (A). If the filename can not be found 0FFH is returned in register (A).

## 4.18. FUNCTION 17: SEARCH FOR FIRST

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 11H<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to return the first occurrence of a specified file name. The directory of the default drive is scanned for an entry that matches the filename and extent number in the fcb pointed to by the register pair (DE). The filename may contain wildcards. If a match is found, the directory location code 0 - 3 is returned in register (A). If no match is found 0FFH is returned in register (A). If the extent number contains 00H, only the first extent for a file can be matched. If the extent number contains a question mark, the first entry found is returned. If the drive number of the specified fcb contains a question mark, all directory entries of any user code, and entries of any type including those not in use, are compared.

## 4.19. FUNCTION 18: SEARCH FOR NEXT

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 12H<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to return the next occurrence of a file name. This function performs the same as function 17 except that the search begins with the entry following the one returned by the last search (function 17 or 18). For this function to work correctly it must be proceeded by a search function.

## 4.20. FUNCTION 19: DELETE FILE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 13H<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to delete a file by name. The drive code, if not zero, is used to select a drive. The directory is scanned for all entries that match the given filename (which may contain wildcards). Only files in the active user area are considered.

## 4.21. FUNCTION 20: READ SEQUENTIAL

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 14H<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to execute a sequential read of the specified record number. The drive code, if not zero, is used to select a drive. The 128-byte record referenced to by the (cr) byte is read and placed into the current file buffer. The (cr) byte is incremented. If it then equals the (rc) byte, the entire extent has bee read; the directory entry describing the next extent of the file is copied into the FCB and (cr( is zeroed. If there are no further extents the extent map in the referenced fcb is set to zero. If the record is successfully read, register (A) is returned containing 00H. If an end of file occurs, register (A) is returned containing 0FFH. The format of the referenced fcb is shown below:

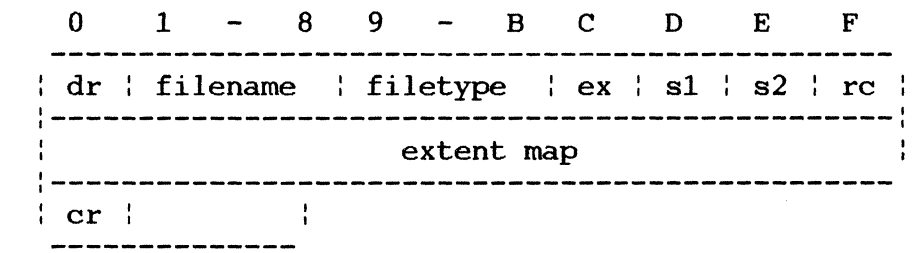



**Fig. 4-2: FCB format**

## 4.22. FUNCTION 21: WRITE SEQUENTIAL

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 15H<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to write the next sequential record. The drive code if not zero, is used to select a drive. If no block has been allocated to the record referenced by (cr) of this extent, one is allocated and entered in the bit map. The record in the current file buffer is written into the position referenced by (cr). The FCB bytes (cr) and (rc) are then incremented. If the extent is then full, the FCB is copied into the matching directory entry and a new entry is made for the next extent, the (cr) and (rc) bytes are reset to zero as is the data map area of the FCB. If the write was successful, 00H is returned in register (A) otherwise a none zero value is returned.

## 4.23. FUNCTION 22: MAKE FILE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 16H<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to create a file. A directory entry is created for the filename specified by the fcb pointed to by register pair (DE). The newly created entry will contain a pointer to the first extent but with no space allocated to it. Upon return register (A) will contain the Directory Code for the new fcb if the operation was successful or 0FFH if no more directory space is available. A successfully created file can be treated as open.

## 4.24. FUNCTION 23: RENAME FILE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 17H<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to rename a file. The drive code if not zero, is used to select a drive. The directory is scanned and all entries for the explicit filename in bytes 01H - 0BH of the fcb are changed to that in bytes 11H - 1BH. If no such directory entry is found, 0FFH is returned in register (A) else the Directory Code is returned in register (A).

## 4.25. FUNCTION 24: RETURN LOGIN VECTOR

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 18H | Register (HL): Login Vector |

Function to return the login vector. A bit map of the drives that are currently active is returned in the (HL). The bits of the map stand for drives as follows:

| | Register (H) | | | | | | | | Register (L) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit numbers: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Drive ID: | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |

**Fig. 4-3: Login Vector Bit Map**

## 4.26. FUNCTION 25: RETURN CURRENT DISK

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 19H | Register (A): Current Disk |

Function to return the current disk assignment. The disk number of the currently logged in drive is returned in register (A). The number returned ranges from 0 to 15 and corresponds to drives A through P respectively.

## 4.27. FUNCTION 26: SET DMA ADDRESS

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 1AH | Register (DE): DMA Address |

Function to set the dma address to the address supplied in the register pair (DE). The Direct Memory Address used to address a 128 byte file buffer for disk read/write transfers is set to the address specified in the register pair (DE). The default DMA address used by ZRDOS is 0080H.

## 4.28. FUNCTION 27: GET ADDR(ALLOC)

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 1BH | Register (HL): ALLOC Address |

Function to return the allocation vector. The address of the allocation vector for the currently logged in drive is returned in the register pair (HL).

## 4.29. FUNCTION 28: WRITE PROTECT DISK

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 1CH | None |

Function to write protect the current disk. The default drive is set to read-only status. Under ZRDOS1 unlike CP/M the protected drive will retain this status until it is reset with a function call 37 or cold boot.

## 4.30. FUNCTION 29: GET READ ONLY VECTOR

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 1DH | Register (HL): R/O Vector Value |

Function to return the read-only status vector. A bit map of the drives that are currently marked read-only is returned in the (HL). The bits of the map stand for drives as follows:

```
                    Register (H)        Register (L)
                         |                   |
                  ---------------- ----------------
Bit numbers:      7 6 5 4 3 2 1 0   7 6 5 4 3 2 1 0
                  | | | | | | | |   | | | | | | | |
Drive ID:         P O N M L K J I   H G F E D C B A
```

**Fig. 4-4: Read Only Vector Bit Map**

## 4.31. FUNCTION 30: SET FILE ATTRIBUTES

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 1EH<br>Register (DE): FCB Address | Register (A): Directory Code |

Function to set the file attributes. The attributes f1-f4 can be used by the user for any purpose. The next three are reserved for future use. Attribute t1 is the File Read Only attribute and is used to prevent a file from being written to. The t2 attribute is the system attribute it alerts the ZCPR3 DIR command that this file is not to be displayed. The t3 attribute is the file archive attribute and is used to indicate whether a file has been updated. Attributes are set by turning on the high order bit of the specified byte and reset by turning it off. The f8 attribute is the Wheel Protect attribute. If this bit is set and the ZCPR3 Wheel byte is off the file can not be written to nor can the file's attributes be changed. If the Wheel byte is set the file is treated as any other file.

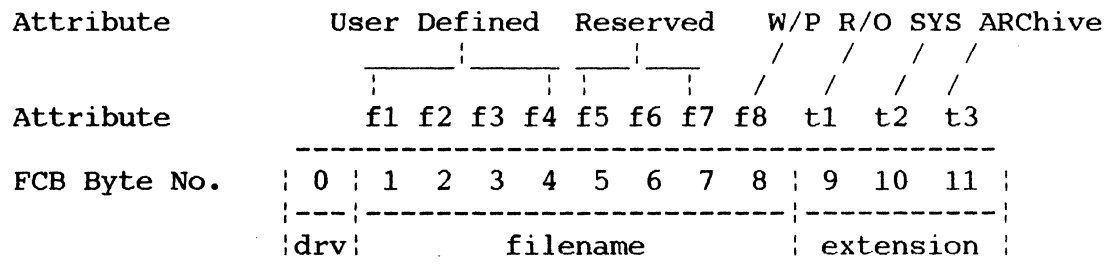



**Fig. 4-5: File Attribute Format.**

## 4.32. FUNCTION 31: GET ADDR(DISK PARMS)

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 1FH | Register (HL): DPB Address |

Function to return the address of the disk parameter block for the current drive. The address of the Disk Parameter Block is returned in the register pair (HL).

## 4.33. FUNCTION 32: SET/GET USER CODE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 20H<br>Register (DE): 0FFH (get) or<br>User Code (set) | Register (A): Current Code<br>or no value |

Function to get or set the user number. If (E) was 0FFH then this is a request to return the current user number. Else set the user number from (E).

## 4.34. FUNCTION 33: READ RANDOM

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 21H<br>Register (DE): FCB Address | Register (A): Return Code |

Function to read a random record from a file. The 'r0', 'r1' and 'r2' bytes used to construct the fcb pointer to the specified record number (see fig.4-6 on next page). Unlike a sequential read operation, the record number is not advanced. Thus, if the calling program does not increment the record number subsequent random read operations will continue to read the same record.

As each random read operation automatically sets the extent and record values into the specified fcb the file can then be sequentially read or written, starting from the currently accessed position.

Upon return from a random read operation register (A) contains 00H if the operation was a success or one of the following error codes:

01 - Reading unwritten data
03 - Cannot close current extent
04 - Seek to unwritten extent
06 - Seek past physical end of disk

## 4.35. FUNCTION 34: WRITE RANDOM

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 21H<br>Register (DE): FCB Address | Register (A): Return Code |

Function to write a random record to a file. This operation is similar to the Read Random operation, except that data is written to the specified record from the currently defined DMA address. If the addressed extent or record has not yet been allocated, an automatic allocation will be performed before the data is written.

Upon return register (A) contains 00H if the operation was successful or an error code if not. The error codes are the same as those returned for a Random Read operation with the addition of the following code:

05 - Directory Overflow

For random R/W, the fcb for the desired record number is set per the 'r0,r1,r2' bytes. These bytes in the fcb are used as follows:

| | fcb+35 | fcb+34 | | fcb+33 |
|---|---|---|---|---|
| Byte | r2 | r1 | | r0 |
| Bit # | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | | 7 6 5 4 3 2 1 0 |
| | overflow | extra extent and 's2' | extent number | record number |

**Fig. 4-6: Use of FCB bytes 'r0','r1' and 'r2'.**

## 4.36. FUNCTION 35: COMPUTE FILE SIZE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 23H<br>Register (DE): FCB Address | Random Record Field Set |

Function to compute the size of a random file. The directory is scanned to find the highest numbered extent of the filename in the fcb specified by register pair (DE). The direct address of the specified file's last record, plus one, is set in the record address field of the specified fcb.

## 4.37. FUNCTION 36: SET RANDOM RECORD

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 24H<br>Register (DE): FCB Address | Random Record Field Set |

Function to return the random record position of a given file which has been read in sequential mode up to now. The extent number and current record number of the fcb specified by register pair (DE) are used to calculate the direct address of the record returned by the last sequential read operation.

## 4.38. FUNCTION 37: RESET DRIVE

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 25H<br>Register (DE): Drive Vector | Register (A): 00H |

Function to allow a program to log off any drives. On entry, set (DE) to contain a word with bits set for those drives that are to be logged off. The log-in vector and the write protect vector will be updated. Drives to be reset are specified in the register pair (DE) as follows:

| | Register (D) | | | | | | | | Register (E) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit numbers: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Drive ID: | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |

**Fig. 4-7: Active Drive Vector Bit Map.**

NOTE: This function differs from CP/M in that only those drives specified in the (DE) will be unprotected and not drives which are not currently active as is done in CP/M.

## 4.39. FUNCTION 40: WRITE RANDOM WITH ZERO FILL

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 28H<br>Register (DE): FCB Address | Register (A): Return Code |

Function to write random records with zero fill. When Direct Access Write (Function 34) is used to build a file, unwritten records within an allocation block contain unpredictable garbage. This request fills the unwritten records of each new block with binary zeros.

## 4.40. FUNCTION 47: RETURN CURRENT DMA ADDRESS

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 2FH | Register (HL): DMA Address |

Function to return the current DMA Address.

## 4.41. FUNCTION 48: RETURN ZRDOS VERSION NUMBER

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 30H | Register (HL): Version number |

Function to return the current ZRDOS version number.

## 4.42. FUNCTION 50: SET WARM BOOT TRAP

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 32H<br>Register (DE): Trap address | None |

This function replaces the warm boot jump address at 0001H with a trap address. Warm boots will then be diverted to the trap address.

## 4.43. FUNCTION 52: RESET WARM BOOT TRAP

| Entry Parameters | Returned Value |
|---|---|
| Register (C): 34H | None |

Function to reset the warm boot trap. The real warm boot address is stored at 0001H. Warm boots will now be directed to the real warm boot. This function will take affect only if the warm boot trap was previously set by function 50.

# 5. DIRECTORY CODES

Many of the ZRDOS functions return a directory code as a return parameter. The Directory Code is actually a multiplier to be used in determining the directory entry location in the default file buffer. The default buffer (location 80H) contains 128 bytes (one sector) of the directory entries read off of the specified disk. There are four 32 byte directory entries to a sector of directory information. The returned Directory Code points to one of these entries. The specified entry can be found by multiplying the Directory Code times 32 and adding this offset to the beginning address of the default buffer (80H). Below is shown a hex ASCII image of a typical directory sector loaded into the default buffer:

```
DEFAULT BUFFER ADDRESS
      /
     /                HEX IMAGE                    ASCII IMAGE
    /                      :                           :
  _/    ___________________:___________________   _______:________
 /  \  /                                       \ /               \
0080   00444454 20202020  20434F4D 00000026  .DDT     COM...&
0090   07000000 00000000  00000000 00000000  ................
00A0   00454449 54202020  20434F4D 0000004C  .EDIT    COM...L
00B0   0A0B0000 00000000  00000000 00000000  ................
00C0   00474F54 4F202020  20434F4D 00000006  .GOTO    COM....
00D0   91000000 00000000  00000000 00000000  ................
00E0   0048454C 4C4F2020  20202020 00000009  .HELLO       ....
00F0   3D000000 00000000  00000000 00000000  =...............
```

**Fig. 5-1: Example Directory Sector.**

The Directory code for the directory entry EDIT.COM is 1. So multiplying the directory code by 32 gives us 32 decimal. 32 decimal is 20 hex. Adding 20 hex to 80 hex gives us A0 hex.

# A. INDEX


**A**

Allocation Vector, 20
Archive, 4
Automatic logon, 3

**B**

BAT:, 11

**C**

CON:, 8, 10, 11, 12, 13
CP/M BDOS Errors, 5
CRT:, 11
Close File, 15
Compute File Size, 25
Console Input, 8
Console Output, 8
Control-C
  ^C, 5
Control-P
  ^P, 8
Control-R
  ^R, 4, 12
Control-S
  ^S, 8
Current Disk, 19

D

DMA, 13
DMA Address, 19, 27
DPB Address, 22
Delete File, 16
Direct Console I/O, 10
Directory Code, 15, 16, 17, 18, 21, 29
Disk Change, 3
Disk Changed Error, 3, 5
Disk R/O Error, 5
Drive Select Error, 5
Drive Vector, 26

E

Echo to printer
^P, 8
Error messages, 5, 6
Error numbers, 5

F

FCB, 14, 15, 16, 17, 18, 21, 23, 24, 25, 27
File Archiving, 4
File R/O Error, 5
File W/P Error, 5
Function 0:, 8
Function 1:, 8
Function 2:, 8
Function 3:, 9
Function 4:, 9
Function 5:, 10
Function 6:, 10
Function 7:, 11
Function 8:, 11
Function 9:, 12
Function 10:, 4, 12
Function 11:, 13
Function 12:, 6, 13
Function 13:, 3, 13
Function 14:, 14
Function 15:, 14
Function 16:, 15
Function 17:, 15
Function 18:, 16
Function 19:, 16
Function 20:, 17
Function 21:, 17
Function 23:, 18
Function 24:, 19
Function 25:, 19
Function 26:, 19
Function 27:, 20
Function 28:, 3, 20
Function 29:, 20
Function 30:, 4, 21
Function 31:, 22
Function 32:, 22
Function 33:, 23
Function 34:, 24
Function 35:, 25
Function 36:, 25
Function 37:, 3, 26
Function 40:, 27
Function 47:, 27
Function 48:, 6, 13, 27
Function 50:, 6, 28
Function 52:, 6, 28

**G**

Get ALLOC Address, 20
Get Address (Disk Params), 22
Get Console Status, 13
Get DMA, 27
Get I/O Byte, 11
Get Read Only Vector, 20

**I**

IOBYTE, 11

**L**

LPT:, 11
LST:, 10, 11
List Output, 10
Login Vector, 19

**N**

Named Directories, 2
Non-retryable errors, 5

**O**

Open File, 14

**P**

PTP:, 11
PTR:, 11
PUN:, 9, 11
Pause
  ^S, 8
Print String, 12
Punch Output, 9

R

RDR:, 9, 11
Read Console Buffer, 4, 12
Read Error, 5
Read Only Disk Status, 3
Read Random, 23
Read Sequential, 17
Reader Input, 9
Redirectable I/O, 2
Rename File, 18
Reset Disk System, 3, 13
Reset Drive, 3, 26
Reset Warm Boot Trap, 6, 28
Return Code, 23, 24, 27
Return Current DMA Address, 27
Return Current Disk, 19
Return Login Vector, 19
Return Version Number, 6, 13
Return ZRDOS Version Number, 6, 27
Rubout (DEL), 4, 12

S

Search for First, 15
Search for Next, 16
Select Disk, 14
Set DMA Address, 19
Set File Attributes, 4, 21
Set I/O Byte, 11
Set Random Record, 25
Set Warm Boot Trap, 5, 6, 28
Set/Get User Code, 22
System Reset, 8

T

TTY:, 11

U

UC1:, 11
UL1:, 11
UP1:, 11
UP2:, 11
UR1:, 11
UR2:, 11
User Code, 22

**W**

Warm Boot Trap, 5, 28
Warm boot, 5, 6
Warm boot trap, 6
Wheel Protection, 4
Wheel Protection Error, 4
Wheel byte, 1
Wildcards, 14, 15, 16
Write Protect Disk, 3, 20
Write Random, 24
Write Random with zero fill, 27
Write Sequential, 17

**Z**

ZCPR3, 1, 2, 4
ZRDOS Errors, 5

# ZDM/ZDMZ/ZDMH -- Z-System Tools

# Z80/HD64180 DEBUGGER and MONITOR

# USER'S GUIDE

by

Robert Doolittle

ZDM/ZDMZ/ZDMH is Copyright 1985 RD SOFTWARE. No part of this document may be reproduced in any way or by any means without prior written permission of publisher. Address requests to Echelon, Inc., 101 First Street, Los Altos, CA 94022.

TABLE OF CONTENTS


| Section | | Page |
|---|---|---|
| | Author's Note | ii |
| I. | **INTRODUCTION** | 1 |
| II. | **ZDM COMMANDS** | 3 |
| 1. | The D (Display) Command | 3 |
| 2. | The DI (Disable Interrupt) Command | 3 |
| 3. | The EI (Enable Interrupt) Command | 3 |
| 4. | The F (Fill) Command | 3 |
| 5. | The G (Go) Command | 4 |
| 6. | The H (Hex Math) Command | 4 |
| 7. | The I (Input) Command | 4 |
| 8. | The L (List) Command | 5 |
| 9. | The M (Move) Command | 5 |
| 10. | The QI (Query Input) Command | 5 |
| 11. | The QO (Query Output) Command | 5 |
| 12. | The R (Read) Command | 6 |
| 13. | The S (Set) Command | 6 |
| 14. | The T (Trace) Command | 7 |
| 15. | The U (Untrace) Command | 7 |
| 16. | The X (Examine) Command | 7 |
| 17. | The & (Alternate Register) Command | 8 |
| 18. | The B (Block Search) Command | 8 |
| 19. | The V (Verify) Command | 9 |
| 20. | The P (Print) Command | 9 |
| 21. | The J (Jump) Command | 9 |
| III. | **INSTALLATION PROCEDURES** | 11 |
| **APPENDIX A** | **PATCHING NOTE** | 13 |
| **APPENDIX B** | **ZDM MNEMONICS** | 15 |
| **APPENDIX C** | **HITACHI HD64180 MNEMONICS** | 17 |
| **APPENDIX D** | **ZDM/ZDMZ/ZDMH COMMAND SUMMARY** | 19 |

## Author's Note

These programs have been thoroughly tested and are believed to be correct. If you find something not to your liking let us know. We welcome your comments, criticisms or questions. Please call or write if you experience any problems.

Robert Doolittle
Echelon Z-Team Member
Telephone 213/454-8270
1290 Monument Street
Pacific Palisades, CA 90272

# I. INTRODUCTION

ZDM, ZDMZ and ZDMH are Z80 and HD64180 machine language debuggers and monitors designed to run under Z-System (and CP/M*) operating system environment. They recognize and debug 8080, Z80 or HD64180 code although they only run on Z80 and HD64180 machines. The command types and command structure are nearly identical to those of the DDT module provided by Digital Research as part of their CP/M operating system. The major difference is that Z80/HD64180 code can be debugged and ten new commands with variations have been added. ZDM/ZDMZ/ZDMH supports all the DDT commands except the in-line assembly command A. Also, only one breakpoint is implemented.

ZDMZ/ZDMH support Zilog Z80 mnemonics whenever instruction mnemonics are displayed. Additionally, ZDMH supports the enhanced Hitachi instruction set of the HD64180. ZDM uses extended Intel 8080 mnemonics similar to TDL (Xitan) which retains all standard 8080 mnemonics. Please refer to the Appendices for a definition of the mnemonics used by ZDM. Refer also to Section III Installation Procedures before attempting to run ZDM, ZDMZ or ZDMH.

Throughout the remainder of this manual all references to ZDM apply equally to ZDMZ/ZDMH except where otherwise explicitly noted.

ZDM is invoked by typing one of the following three forms at the console:

```
ZDM
ZDM filename
ZDM filename.filetype
```

where "filename" is the name of the file to be loaded. ZDM will then sign-on and relocate itself to overlay the CCP and reside directly below ZRDOS (or BDOS). The jump to ZRDOS at location 5 is altered to address the base location of ZDM which, in turn, contains a jump to ZRDOS or BDOS. Note that ZDM provides an additional page of available transient memory compared to DDT for a given size Z or CP/M system. Like DDT, transient programs loaded for debugging can overwrite the disassembler module. In this case the L command is disabled and the instruction field for the X and T commands is replaced by the corresponding hexadecimal bytes of the instruction.

ZDM has an additional feature to prevent overlaying the ZDM nucleus itself. If this is about to happen, ZDM aborts the load and print an "OUT OF MEMORY" message. It then returns to command level so that those portions which were loaded may be examined.

---

*CP/M is registered trademark of Digital Research. Z-System is trademark of Echelon.

The second and third forms of the console command line result in the named file being loaded after ZDM is entered.

After the sign-on message and program loading, if specified in the command line, ZDM will respond with the prompt character "-" and wait for input commands. Each command consists of either one or two characters, as defined in Section II, which determines the command type. These characters may also be followed by additional parameters. No delimiter should be used between the command type characters and the first parameter except as described for the G command. Subsequent parameters are delimited by a comma or a single space. In all cases, if the command expects a final parameter and this parameter is omitted, ZDM will assume it is zero.

All command lines are terminated by a carriage return. All keyboard input to ZDM and output from ZDM is in hexadecimal. ZDM will accept either upper or lower case letters. A single character (?) is printed if an error occurs. To exit ZDM and return to ZCPR3 or CP/M command level either a Control-C or a G0 (jump to loation 0) may be executed.

## II. ZDM COMMANDS

Details of each command are given in this Section.

1. The D (Display) Command. The D command permits the operator to view the contents of memory in hexadecimal and ASCII formats. The forms are:

```
D
Ds
Ds,f
```

In the first case, memory is displayed from the current display address (initially 100H) and continues for the number of lines specified at initialization. Subsequent display addresses are initialized to the value of the program counter following an X, T, U, or G command.

The second form of the D command is similar to the first except that the display address is first set to address s. The third form displays from address s through f. In all cases a subsequent issue of the first form will start with the display address following the last address displayed, resulting in a continuing display. Long typeouts can be aborted with the rubout key.

2. The DI (Disable Interrupt) Command. This command takes the single form:

```
DI
```

The default condition, whenever the target program is entered via the G, T, or U command, is that interrupts are enabled. (Interrupts are always disabled when returning to ZDM). The DI command overrides this default condition. The DI command will remain in effect until a subsequent EI is issued.

3. The EI (Enable Interrupt) Command. This command restores the default interrupt condition. See the DI command description. It takes the single form:

```
EI
```

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

```
Fs,f,c
```

where s is the starting address, f is the final address, and c is a hexadecimal byte constant. If c is omitted then it is assumed to be zero. This command fills the block of memory from s to f

inclusive with the constant c. If f is less than s an error message (?) will occur.

5. The G (Go) Commands. Program execution is started using the G command with one optional breakpoint address. The G command takes four possible forms:

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

The first form starts execution of the target program at the current value of the program counter and in the current machine state with no breakpoints set. The only way for ZDM to regain control is through a RST 7 execution. The second form is similar to the first except that the program counter is first set to s before execution begins. Third form is the same as the second except that a breakpoint is set at address b. Program execution is stopped and control is returned to ZDM. The instruction at address b is not executed when the breakpoint is encountered. The fourth form starts execution from the current program counter and machine state and sets breakpoint at address b.

Upon entering a breakpoint, ZDM types *d where d is the stop address. The machine state can be examined at this point using the X or &X command.

6. The H (Hex Math) Command. The H command takes the form:

```
Ha,b
```

where a and b are hexadecimal constants from 1 to 4 digits. The sum a+b and the difference a-b are displayed in hexadecimal in the form:

```
a+b a-b
```

7. The I (Input) Command. The I command allows the operator to insert a file name into the Z-System default file control block at 5CH. The default FCB can be used by the program under test as if it had been passed by the Console Command Processor. This command must also be used prior to the R command when reading additional HEX or COM files. The forms of the I command are:

```
Ifilename
Ifilename.filetype
```

If the filetype is anything except HEX then ZDM will assume it is a COM file and the R command will read it into memory starting at 100H. (See the R command for further details).

8. The L (List) Command. The L command is used to list assembly language mnemonics. The three forms of the command are:

```
L
Ls
Ls,f
```

The first form lists the number of lines specified in the initialization and starting at the current address of the program counter. The second form lists the same number of lines but starts at the address s. The third form starts at the address s and continues for f lines. All three forms can be continued with a subsequent L command similar to the D command. Also, like the D command, the starting address if not specified is always initialized to the program counter following an X, T, U or G command. Long typeouts can be aborted with the rubout key.

9. The M (Move) Command. The M command will move a block of memory from one location to another. The form of the M command is:

```
Ms,f,d
```

where s is the start address of the move, f is the final address and d is the destination address. If f is less than s, an error (?) will occur.

10. The QI (Query Input) Command. The QI command allows the operator to read an input port and display the value at the port address. The form of this command is:

```
QIa
```

where a is a one byte port address in hexadecimal. The value is printed immediately following execution of this command. Note that if a is omitted, it is assumed to be zero.

11. The QO (Query Output) Command. The QO command allows the operator to output a specified byte to a specified port address. The form of this command is:

```
QOa,b
```

where a is the port address and b is the byte to be output. If either a or b is omitted, it is assumed to be zero.

12. The R (Read) Command. The R command is used in conjunction with the I command to read COM and HEX files from disk into memory. There are two forms of this command:

```
R
Rb
```

where b is an optional offset address which is added to each program or data address as it is loaded. If b is omitted then it is assumed to be zero. Note that if the file name in the FCB from a previous I command is not a HEX type then ZDM assumes it is COM and will load it at 100H or 100H+b if the parameter b is included. If the file cannot be opened or an error occurs in reading, ZDM responds with the error indicator (?). Otherwise at completion of the load a message is issued:

```
NEXT  PC
nnnn pppp
```

where nnnn is the next address following the program just loaded and pppp is the first address of the program just loaded. For HEX files pppp is taken from the last record of the HEX file and will be zero unless an END statement followed by the start address has been included in the source program prior to assembly.

13. The S (Set) Command. The S command allows memory locations to be examined and optionally altered. The form of the command is:

```
Ss
```

where s is the hexadecimal starting address for examination or alteration of memory. ZDM will print the address followed by the byte stored at that address. A carriage return will advance to the next address, displaying the next address and the next byte. If a new byte value is typed followed by a carriage return, this new value will be stored at that address and ZDM will advance automatically to the next address. To terminate the command a period is typed rather than a byte value. The command will also terminate if an invalid hexadecimal value is entered.

14. The T (Trace) Command. The T command permits single step instruction tracing of program execution for 1 to 65535 steps. The forms of this command are:

```
T
Tn
```

where n is an optional step number. The first form assumes an implied n equal one. The CPU state is displayed and the next program step is executed. The termination address is displayed as *hhhh where hhhh is the next address to be executed. The format for the CPU state display is otherwise identical to that of the X command.

Program tracing is discontinued at the interface to Z-System and resumes again after return from Z-System to the target program. Long tracing with the Tn command can be stopped with the rubout key. ZDM will continue tracing from this break if another T or Tn command is issued.

15. The U (Untrace) Command. The U command is identical to the T command except that the CPU state is not displayed. The forms of the command are:

```
U
Un
```

All conditions of the T command apply to the U command. The last CPU state is displayed following the execution of a U or Un command.

16. The X (Examine) Command. (See also the & Command). The X command permits selective display and alteration of the current CPU state at any time. The forms are:

```
X
Xr
```

where r is any of the Z80 registers or flags.

| | | |
|---|---|---|
| C | Carry Flag | (0/1) |
| Z | Zero Flag | (0/1) |
| M | Minus Flag | (0/1) |
| E | Even Parity Flag | (0/1) |
| I | Interdigit Carry | (0/1) |
| A | Accumulator | (0-FF) |
| B | BC register pair | (0-FFFF) |
| D | DE register pair | (0-FFFF) |
| H | HL register pair | (0-FFFF) |
| S | Stack Pointer | (0-FFFF) |
| P | Program Counter | (0-FFFF) |
| X | X-index register | (0-FFFF) |
| Y | Y-index register | (0-FFFF) |

In the first case the CPU register state is displayed in the format

```
CfZfMfEfIf A=bb B=dddd D=dddd H=dddd P=dddd S=dddd
X=dddd Y=dddd instruction
```

where f is a 0 or 1 flag value, bb is a byte value and dddd is a double byte corresponding to a register pair. The instruction field contains the disassembled instruction at the location addressed by the program counter.

The second form of the X command permits display and optional alteration of register or flag values specified by r. If a carriage return is typed following an Xr command then the command is terminated with no changes taking place. Otherwise ZDM accepts input for register or flag changes. If a hexadecimal number in the proper range is typed then that flag or register is correspondingly altered.

17. The & (Alternate Register) Command. The & command takes one of three forms:

```
&
&X
&Xr
```

The first form unconditionally exchanges all CPU registers and flags to the Z80 alternate register set.

The second and third forms of this command are identical to the X and Xr commands except that the operations take place on the alternate register or flag set. Upon termination of these latter two forms the CPU state prior to command execution is restored. The display associated with the &X command replaces the X and Y registers by the vector interrupt register value V. This register value may also be altered by an &XV command followed by the byte value to be stored in the vector interrupt register. The A, B, D and H registers and register pairs are labeled by prime (') symbols whenever the alternate set is being displayed or altered. Note that primes are not used for the flag register except during alteration.

18. The B (Block Search) Command. The B command permits the user to search memory for all occurrences of a byte string. Strings are limited to ten bytes. A second form of this command is initiated by BT rather than B. This second form will accept an ASCII or text string. The form of this command is:

```
Bs,f or BTs,f
```

where s is the start address and f is the final address of the memory block to be searched. Following the carriage return you will be prompted to enter the string. The B form expects the string as a series of HEX bytes. The delimiter may be a space or a comma. The BT form expects a single ASCII string. The input

string is terminated by a carriage return following the last entry. The start address of each occurrence of the string from s to f will be displayed.

19. The V (Verify) Command. The V command will verify if two blocks of memory are identical. The form of this command is:

Vs,f,b

where s is the start address and f is the final address of one block and b is the start address of the other block. If the

match fails, the address is printed out followed by the byte at the corresponding address in the second block.

20. The P (Print) Command. The P command is a toggle which does not expect any arguments. The effect is to send all output to the LIST device as well as to the console. It is turned off by a subsequent P command. Whenever the P toggle is on, a 'P' will be displayed as part of the X or T display.

21. The J (Jump) Command. The J command is a toggle which does not expect any arguments. It only affects subsequent T or U commands. If J has been executed then the T command will display only conditional and unconditional CALLS, JUMPS, RETURNS, RESTARTS, PCHL (IX or IY) and relative JUMPS. The Tn form of the T command is usually used where n represents the actual number of instructions to be traced. As usual this command can be aborted with the rubout key. Whenever the J toggle is on, a 'J' will be displayed as part of the X or T display.

(THIS PAGE INTENTIONALLY BLANK)

## III. INSTALLATION PROCEDURES

The user should first make a copy of ZDM and keep the original master as a back-up. Do not write on the master. Using the copy type "ZDM". ZDM will respond with the question "What is your terminal width (in hex)". A carriage return in response to this question will default to a width of 80(50H). Otherwise, type in the character width of your terminal in hex followed by a carriage return. Next, the number of lines desired for the D and L command displays are requested. A carriage return at this point defaults to 21(15H) lines. (This is the recommended size for 24 line terminals.) Otherwise, type in the number of lines desired, in hex, followed by a carriage return.

Finally, ZDM will ask the question "Is this correct? (Y or N)". Do not respond with "Y" at this time. Following an "N" response ZDM will sign-on, print the prompt character "-", and await a command. You should now test the D and L commands to determine if these displays are sized properly for your terminal. If not, return to Z-System command level by typing G0 or Control-C and repeat the above procedure. When you are satisfied that the displays are properly sized, then respond with "Y" when asked "Is this correct? (Y or N)". After a "Y" response ZDM will automatically create your custom installed file. If you are using ZDM then the installed file name will be ZDI.COM. If you are using ZDMZ or ZDMH then the installed file name is either ZDIZ.COM or ZDIH.COM. You may rename these to whatever names you desire.

ZDM is now properly configured for your terminal. Subsequent invocations of ZDM will proceed directly to the sign-on message. If, at some later time, you wish to change these display parameters you will have to start once again with a copy of the master uninstalled file.

(THIS PAGE INTENTIONALLY BLANK)

## APPENDIX A

Some users have experienced difficulties with ZDM/ZDMZ when used with interrupt driven systems. (ZDMH has interrupts enabled as default condition. You can still disable interrupts with the DI command when entering the target program but they will be enabled again when ZDMH regains control.) The following patches are recommended when running ZDM/ZDMZ on such systems.

Using ZDMH or other debugger, load an image of ZDM/ZDMZ into memory starting at 100h and change the following bytes (addresses apply to version 3.2):

| ZDM ADDRESS | ZDMZ ADDRESS | FROM | TO |
|---|---|---|---|
| 0C61H | OC31H | 0F3H | 00H |
| 0C7AH | 0C4AH | 0F3H | 00H |
| 0DC1H | OD91H | 0F3H | 00H |

Return to Z-System or CP/M without disturbing the memory image and save 22 (or 16h) pages with the SAVE command.

Another frequent user request has been to change the ZDM RESTART address, curently RST 7 at 38h, to a different RESTART. The following patches will accomodate this change. (RST 0 cannot be used under Z operating system or under CP/M.)

| ZDM ADDRESS | ZDMZ ADDRESS | ZDMH ADDRESS | FROM | TO |
|---|---|---|---|---|
| 0D9DH | 0D6DH | 0DEDH | 38H | new RESTART address |
| 0DA5H | 0D75H | 0DF5H | 39H | new RESTART address + 1 |
| 1007H | 0FD7H | 1057H | 0FFH | new RESTART opcode |

(THIS PAGE INTENTIONALLY BLANK)

# APPENDIX B

The disassembler module of ZDM uses a mnemonic set which is similar to the Technical Design Laboratories (TDL) mnemonics. All Intel 8080 mnemonics are preserved. The Z80 peculiar instructions differ from the Zilog mnemonics as shown in the accompanying table. The ZDM mnemonic set is nearly identical to that released by Digital Research as Z80.LIB to be used with their CP/M Macro Assembler "MAC". The following conventions are used in the table.

```
    r - any register or memory
   rr - any register pair or stack pointer
   nn - 8 bit immediate data (0 to 255)
    d - 8 bit signed displacement (-128 to 127)
 nnnn - 16 bit address or immediate data (0 to 65535)
    b - bit number (0 to 7, 7 is most significant)
 addr - 16 bit address within PC+127 through PC-128
```

In cases involving a displacement, d, this parameter is always last one in operand field.

| ZDM | ZILOG | ZDM | ZILOG |
|---|---|---|---|
| LDX r,d | LD r,(IX+d) | LDIR | LDIR |
| LDY r,d | LD r,(IY+d) | LDD | LDD |
| STX r,d | LD (IX+d),r | LDDR | LDDR |
| STY r,d | LD (IY+d),r | CCI | CPI |
| MVIX nn,d | LD (IX+d),nn | CCIR | CPIR |
| MVIY nn,d | LD (IY+d),nn | CCD | CPD |
| LDAI | LD A,I | CCDR | CPDR |
| LDAR | LD A,R | ADDX d | ADD (IX+d) |
| STAI | LD I,A | ADDY d | ADD (IY+d) |
| STAR | LD R,A | ADCX d | ADC (IX+d) |
| LXIX nnnn | LD IX,nnnn | ADCY d | ADC (IY+d) |
| LXIY nnnn | LD IY,nnnn | SUBX d | SUB (IX+d) |
| LBCD nnnn | LD BC,(nnnn) | SUBY d | SUB (IY+d) |
| LDED nnnn | LD DE,(nnnn) | SBBX d | SBC (IX+d) |
| LSPD nnnn | LD SP,(nnnn) | SBBY d | SBC (IY+d) |
| LIXD nnnn | LD IX,(nnnn) | ANAX d | AND (IX+d) |
| LIYD nnnn | LD IY,(nnnn) | ANAY d | AND (IY+d) |
| SBCD nnnn | LD (nnnn),BC | XRAX d | XOR (IX+d) |
| SDED nnnn | LD (nnnn),DE | XRAY d | XOR (IY+d) |
| SSPD nnnn | LD (nnnn),SP | ORAX d | OR (IX+d) |
| SIXD nnnn | LD (nnnn),IX | ORAY d | OR (IY+d) |
| SIYD nnnn | LD (nnnn),IY | CMPX d | CP (IX+d) |
| SPIX | LD SP,IX | CMPY d | CP (IY+d) |
| SPIY | LD SP,IY | INRX d | INC (IX+d) |
| PUSHIX | PUSH IX | INRY d | INC (IY+d) |
| PUSHIY | PUSH IY | DCRX d | DEC (IX+d) |
| POPIX | POP IX | DCRY d | DEC (IY+d) |
| POPIY | POP IY | NEG | NEG |

| ZDM | ZILOG | ZDM | ZILOG |
|---|---|---|---|
| EXAF | EX AF,AF' | IM0 | IM 0 |
| EXX | EXX | IM1 | IM 1 |
| XTIX | EX (SP),IX | IM2 | IM 2 |
| XTIY | EX (SP),IY | DADC rr | ADC HL,rr |
| LDI | LDI | DSBC rr | SBC HL,rr |
| DADX rr | ADD IX,rr | OUTI | OUTI |
| DADY rr | ADD IY,rr | OUTIR | OTIR |
| INXIX | INC IX | IND | IND |
| INXIY | INC IY | INDR | INDR |
| DCXIX | DEC IX | OUTD | OUTD |
| DCXIY | DEC IY | OUTDR | OTDR |
| BIT b,r | BIT b,r | RLCR r | RLC r |
| SET b,r | SET b,r | RLCX d | RLC (IX+d) |
| RES b,r | RES b,r | RLCY d | RLC (IY+d) |
| BITX b,d | BIT b,(IX+d) | RALR r | RL r |
| BITY b,d | BIT b,(IY+d) | RALX d | RL (IX+d) |
| SETX b,d | SET b,(IX+d) | RALY d | RL (IY+d) |
| SETY b,d | SET b,(IY+d) | RRCR r | RRC r |
| RESX b,d | RES b,(IX+d) | RRCX d | RRC (IX+d) |
| RESY b,d | RES b,(IY+d) | RRCY d | RRC (IY+d) |
| JR addr | JR addr | RARR r | RR r |
| JRC addr | JR C,addr | RARX d | RR (IX+d) |
| JRNC addr | JR NC,addr | RARY d | RR (IY+d) |
| JRZ addr | JR Z,addr | SLAR r | SLA r |
| JRNZ addr | JR NZ,addr | SLAX d | SLA (IX+d) |
| DJNZ addr | DJNZ,addr | SLAY d | SLA (IY+d) |
| PCIX | JP (IX) | SRAR r | SRA r |
| PCIY | JP (IY) | SRAX d | SRA (IX+d) |
| RETI | RETI | SRAY d | SRA (IY+d) |
| RETN | RETN | SRLR r | SRL r |
| INP r | IN r,(C) | SRLX d | SRL (IX+d) |
| OUTP r | OUT (C),r | SRLY d | SRL (IY+d) |
| INI | INI | RLD | RLD |
| INIR | INIR | RRD | RRD |



APPENDIX C

## HITACHI HD64180 MNEMONICS

| Object Code | Source Statement | Operation |
|---|---|---|
| ED3805 | IN0 A,(nn) | Load register with input from |
| ED0005 | IN0 B,(nn) | port (nn). |
| ED0805 | IN0 C,(nn) | |
| ED1005 | IN0 D,(nn) | |
| ED1805 | IN0 E,(nn) | |
| ED2005 | IN0 H,(nn) | |
| ED2805 | IN0 L,(nn) | |
| ED4C | MLT BC | Unsigned multiplication of |
| ED5C | MLT DE | each half of the specified |
| ED6C | MLT HL | register pair with the 16-bit |
| ED7C | MLT SP | result going to the specified register pair. |
| ED8B | OTDM | Load output port (C) with location (HL), decrement HL, B, and C. |
| ED9B | OTDMR | Load output port (C) with location (HL), decrement HL, B, and C. Repeat until B=0. |
| ED83 | OTIM | Load output port (C) with location (HL), increment HL and C. Decrement B. |
| ED93 | OTIMR | Load output port (C) with location (HL), increment HL and C. Decrement B. Repeat until B=0. |
| ED3905 | OUT0 (nn),A | Load output port (nn) from |
| ED0105 | OUT0 (nn),B | register. |
| ED0905 | OUT0 (nn),C | |
| ED1105 | OUT0 (nn),D | |
| ED1905 | OUT0 (nn),E | |
| ED2105 | OUT0 (nn),H | |
| ED2905 | OUT0 (nn),L | |
| ED76 | SLP | Enter sleep mode. |
| ED3C | TST A | Non-destructive AND with |
| ED04 | TST B | accumulator and specified |
| ED0C | TST C | operand. |
| ED14 | TST D | |
| ED1C | TST E | |
| ED24 | TST H | |
| ED2C | TST L | |
| ED6405 | TST nn | |
| ED34 | TST (HL) | |
| ED7405 | TSTIO nn | Non-destructive AND of nn and the contents of port (C). |

(THIS PAGE INTENTIONALLY BLANK)

# APPENDIX D

## ZDM/ZDMZ/ZDMH COMMAND SUMMARY

| Function | Form | Definition |
|---|---|---|
| Display | D[s,f] | display screen of memory in hex and ASCII |
| Disable Interrupt | DI | disable interrupts, normal default |
| Enable Interrupt | EI | enable interrupts, default if entering from G, T, and U |
| Fill | Fs,f,c | fill range of memory with declared byte value |
| Go | G[s,b] | execute program with optional breakpoint |
| Hex Math | Ha,b | obtain sum and difference of two hex numbers |
| Input | Ifilename | set up file control block to receive file name |
| List | L[s,f] | list to screen assembly language mnemonics |
| Move | Ms,f,d | move data from one area of memory to another |
| Query Input | QIa | display input byte from indicated port a |
| Query Output | QOa,b | output byte b to indicated port a |
| Read | R[b] | read in file set up with I command, optional offset |
| Set | Ss | examine and optionally alter memory |
| Trace | T[n] | single step program execution, up to 65535 steps |
| Untrace | U[n] | similar to T, but CPU state not displayed |
| Examine | X[r] | examine CPU register values |
| Alternate Register | &[X][r] | examine Z80 alternate register values |
| Block Search | B[T]s,f | find ASCII or hex string in declared memory range |
| Verify | Vs,f,b | verify if two blocks of memory are identical |
| Print | P | send all screen output also to printer |
| Jump | J | display only branch statements: calls, jumps, returns, etc. |

**Legend:** items in [ ]'s are optional; s=start address; f=final address; c=hex byte value; a=hex value or port address; b=hex value or offset, breakpoint or block start address; d=destination address; n=step number; r=register letter, a for accumulator, b for bc pair, s for sp, etc.

# R E L O C A T I N G  M A C R O  A S S E M B L E R

# A N D  L I N K E R

for

## Z 8 0  A N D  H D 6 4 1 8 0

by

Patrick O'Connell

Zas, Zlink, Zlib, Zcon, Zref are Copyright 1984/85 by Mitek. No part of this document may be reproduced in any way or by any means without prior written permission of the publisher. Address requests to Echelon, Inc., 101 First Street, Los Altos, CA 94022.

Rev. 6/25/85

Copyright 1984/85 Mitek
All Rights Reserved

## WARNING

THIS SOFTWARE AND MANUAL ARE BOTH PROTECTED BY U.S. COPYRIGHT LAW (TITLE 17 UNITED STATES CODE). UNAUTHORIZED REPRODUCTION AND/OR SALES MAY RESULT IN IMPRISONMENT OF UP TO ONE YEAR AND FINES OF UP TO $10,000 (17 USC 506). COPYRIGHT INFRINGERS MAY ALSO BE SUBJECT TO CIVIL LIABILITY.

## LIMITED WARRANTY

THIS PROGRAM AND INSTRUCTION MANUAL ARE SOLD "AS IS," WITHOUT WARRANTY AS TO THEIR PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THIS PROGRAM IS ASSUMED BY YOU.

HOWEVER, TO THE ORIGINAL PURCHASER ONLY, ECHELON WARRANTS THE MAGNETC DISKETTE ON WHICH THE PROGRAM IS RECORDED TO BE FREE FROM DEFECTS IN MATERIALS AND FAULTY WORKMANSHIP UNDER NORMAL USE FOR A PERIOD OF THIRTY DAYS FROM THE DATE OF SHIPMENT. IF DURING THIS THIRTY-DAY PERIOD THE DISKETTE SHOULD BECOME DEFECTIVE, IT MAY BE RETURNED TO ECHELON FOR A REPLACEMENT WITHOUT CHARGE.

YOUR SOLE AND EXCLUSIVE REMEDY IN THE EVENT OF A DEFECT IS EXPRESSLY LIMITED TO REPLACEMENT OF THE DISKETTE AS PROVIDED ABOVE. IF FAILURE OF A DISKETTE HAS RESULTED FROM ACCIDENT OR ABUSE ECHELON SHALL HAVE NO RESPONSIBILITY TO REPLACE THE DISKETTE UNDER THE TERMS OF THIS LIMITED WARRANTY.

ANY IMPLIED WARRANTIES RELATING TO THE DISKETTE, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO A PERIOD OF THIRTY DAYS FROM DATE OF SHIPMENT. ECHELON SHALL NOT BE LIABLE FOR INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OF THIS PRODUCT. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTIAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS MIGHT NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

Trademarks: Zas, Zlink, Zlib, Zcon, Zref, Mitek; ZDM, RD Software; DSD, Soft Solutions; Z-System, Echelon, Inc.; Z80, Zilog, Inc.; HD64180, Hitachi; CP/M, DDT, SID, ZSID, Digital Research, Inc.

**TABLE OF CONTENTS**


| | | | |
|---|---|---|---|
| **Chapter 1** | | **INTRODUCTION** | **1** |
| | 1.1 | Overview | 1 |
| | 1.2 | Distribution Files | 1 |
| | 1.3 | Installation | 1 |
| | 1.4 | Software Updates | 2 |
| **Chapter 2** | | **ZAS INVOCATION** | **3** |
| | 2.1 | ZAS Operation | 3 |
| | 2.2 | ZAS Options | 3 |
| | 2.3 | Assembly Statistics | 4 |
| **Chapter 3** | | **PROGRAM FORMAT** | **5** |
| | 3.1 | Label Field | 5 |
| | 3.2 | Operation Field | 5 |
| | 3.3 | Operand Field | 5 |
| | 3.4 | Comment Field | 6 |
| **Chapter 4** | | **EXPRESSIONS** | **7** |
| | 4.1 | Numeric Constants | 7 |
| | 4.2 | String Constants | 7 |
| | 4.3 | Character Constants | 7 |
| | 4.4 | Labels | 8 |
| | | 4.4.1 Label Characteristics | 8 |
| | | 4.4.2 Relocation Bases | 8 |
| | 4.5 | Relocation Counter Reference | 9 |
| | 4.6 | Registers | 9 |
| | 4.7 | Operators | 10 |
| | 4.8 | Precedence of Operators | 11 |
| | 4.9 | Parentheses Versus Brackets | 12 |
| | 4.10 | Expression Restrictions | 12 |
| **Chapter 5** | | **PSEUDO-OPS** | **13** |
| | 5.1 | General Pseudo-ops | 13 |
| | 5.2 | Listing Control Pseudo-ops | 16 |
| | 5.3 | Conditional Assembly Pseudo-ops | 16 |
| | | 5.3.1 IF Pseudo-ops Evaluation | 16 |
| | | 5.3.2 Conditional Assembly Forms | 17 |
| | 5.4 | Linkage Pseudo-ops | 19 |
| | 5.5 | Relocation Base Pseudo-ops | 20 |
| | 5.6 | Macro Pseudo-ops | 22 |
| | 5.7 | Special Function Pseudo-ops | 22 |
| **Chapter 6** | | **MACRO FACILITY** | **23** |
| | 6.1 | Repeat Macros | 23 |
| | 6.2 | Stored Macros | 25 |
| | 6.3 | Exiting Macros | 25 |
| | 6.4 | Local Symbols | 25 |
| | 6.5 | Macro Invocation | 26 |
| | 6.6 | Parameter Evaluation | 26 |

**TABLE OF CONTENTS** (continued)


| | | | |
|---|---|---|---|
| **Chapter 7** | | **ZAS ERROR MESSAGES.....** | **29** |
| | 7.1 | Non-Fatal Errors | 29 |
| | 7.2 | Fatal Errors | 30 |
| | | 7.2.1 General Fatal Error Messages | 30 |
| | | 7.2.2 Macro Fatal Error Messages | 30 |
| **CHAPTER 8** | | **CROSS-REFERENCE GENERATION.....** | **31** |
| | 8.1 | Overview | 31 |
| | 8.2 | ZREF Operation | 31 |
| | 8.3 | Reserved Symbols | 31 |
| **CHAPTER 9** | | **CODE CONVERTER.....** | **33** |
| | 9.1 | Code Converter Operation | 33 |
| | 9.2 | Convertible TDL Pseudo-ops | 33 |
| | 9.3 | Error Messages | 34 |
| **CHAPTER 10** | | **LINKER.....** | **35** |
| | 10.1 | Overview | 35 |
| | 10.2 | ZLINK Operation | 35 |
| | 10.3 | ZLINK Options | 35 |
| | 10.4 | Define Next Free Memory Location | 36 |
| | 10.5 | ZLINK Error Messages | 37 |
| **CHAPTER 11** | | **LIBRARY MANAGER.....** | **39** |
| | 11.1 | Overview | 39 |
| | 11.2 | ZLIB Operation | 39 |
| | 11.3 | ZLIB Options | 39 |
| | 11.4 | ZLIB Messages | 39 |
| | 11.5 | ZLIB Error Messages | 40 |

Appendix A: Z80 Mnemonic Machine Instruction Codes

Appendix B: Software Update Form

Appendix C: Pseudo-op Summary

Appendix D: Hitachi HD64180 Mode

# CHAPTER 1
# INTRODUCTION

## 1.1 OVERVIEW

ZAS (Z80 and HD64180 Relocating Macro Assembler) reads assembly language statements from a disk file and produces either an Intel compatible HEX file or a Microsoft compatible REL file. These files can then be loaded using Echelon supplied MLOAD, or CP/M LOAD, command or any Microsoft object compatible linker. A symbol table file (SYM) is optionally produced that can be used with Echelon DSD or Digital Research SID and ZSID debuggers.

The minimum Z or CP/M system configuration in which to use ZAS is 48k-bytes of RAM with one disk drive.

As soon as you receive ZAS, make backup copies! Then go through the installation process using a copy.

## 1.2 DISTRIBUTION FILES

You will find the following files on your distribution disk:

| File | Function |
|---|---|
| ZAS.COM | Assembler |
| ZLINK.COM | Linker |
| ZLIB.COM | Library Manager |
| ZCON.COM | 8080 to Z80 Code Converter |
| ZREF.COM | Cross-reference Generator |
| TEST.Z80 | Test Assembly File |
| INSTZAS.COM | Installation Program |

## 1.3 INSTALLATION

The installation program was designed to set assembler output options. Type in INSTZAS to invoke the installation program. The options described on the next page will appear on the screen.

**INSTZAS<cr>**

```
ZAS installation options:

1.  Listing to terminal  - off

2.  Listing to disk file - off

3.  Listing to printer   - off

4.  Generate object file - on

5.  Generate symbol file - off

6.  Object file type     - rel

7.  IF trueness based on - least significant bit

99.  Changes complete

Enter option number to change:
```

The preset values for different options is indicated to right of option. To change (toggle) an option value (i.e., **on** to **off**, **rel** to **hex**, or **least significant bit** to **all sixteen bits**), simply enter option number (1 to 7) followed by carriage return <CR>. When desired option changes have been made, type in 99 to end installation program and have ZAS.COM automatically updated.

## 1.4 SOFTWARE UPDATES

You can assist in refining ZAS by recommending enhancements and reporting any software problems on a copy of the Software Update Form, a sample of which is in Appendix B. Software updates will be provided at regular intervals for a nominal fee. You will be notified by Echelon when software updates are available.

# CHAPTER 2
# ZAS INVOCATION

## 2.1 ZAS OPERATION

ZAS is invoked by typing:

ZAS filename.filetype

where filename is the name of the source file to be assembled. If no filetype is specified, then Z80 is assumed. Typing ^C will cancel ZAS operation.

## 2.2 ZAS OPTIONS

A variety of options are available to provide control over the execution parameters of ZAS. They are used once at the end of a command line and spaces are not allowed between options:

ZAS filename {$}options

There are two types of options: non-disk reference options and disk reference options. Using the non-disk reference options reverses the settings supplied by the Install Program and includes the C, H, and L options.

**C: CRT Option.** Setting the C option will page the output of ZAS, at 23 lines per page. Pressing any key allows you to continue to scroll through the output page by page. However, it should be noted that a ^C will abort the assembly.

**H: Hex Option.** When this option is set it will generate Intel compatible hex files instead of Microsoft compatible REL files. Note: When using HEX files, you must have an ORG statement of 100H or higher to prevent an inverted address error from MLOAD or LOAD.COM.

**L: Listing to Printer Option.** Setting the L option sends a formatted assembly listing to Z or CP/M LST: device.

The disk reference options require two characters. The first character is the P, O, or S option characters. The second character indicates the output disk drive for the specified option. The second character must be A-P or Z, where Z (for zero or null) suppresses the output altogether.

**O: Object File Generation** (filename.REL or filename.HEX). The O option specifies the disk for object file output. Depending on the H option, the object file will be a Microsoft compatible REL file or an Intel compatible HEX file.

**P: Listing to a PRN File** (filename.PRN). The P option will send a formatted assembly listing to the specified disk.

**S: Symbol File Generation** (filename.SYM). The S option specifies output disk for Echelon or DRI compatible SYM file.

## 2.3 ASSEMBLY STATISTICS

At the completion of an assembly, ZAS provides several statistics on the program assembled. The output is as follows:

```
Assembly statistics:

 nnnn lines
 nnnn labels
 nnnn macros read
 nnnn macro expansions
 nnnn errors
 nnnn free bytes
```

where nnnn is a decimal number.

# CHAPTER 3
# PROGRAM FORMAT

Acceptable program input consists of a sequence of statements in the form:

label operation operand comment

where each field is separated by one or more spaces and/or tabs. All fields are optional and may begin in any column except for the label field which must begin in column one. The statement is terminated by a carriage return and a line feed is allowed but not necessary. You may also insert blank lines into the program.

The statement may be either upper or lower-case except for macro parameters. For macro parameters, the actual and formal parameters must be in the same case for substitution to take place.

## 3.1 LABEL FIELD

Labels take the form:

label or label:

and are optional except for the SET, EQU, and MACRO assembler directives. The label consists of alphanumeric characters, a ?, an @, or a $ and the first character must not be numeric. If the label exceeds 15 characters then the label is truncated to the right. Labels can be either upper-case or lower-case. The ":" following a label is optional. Examples of labels include the following:

| | | |
|---|---|---|
| a123 | ?a123 | @a123 |
| aLL: | ?ALL: | update_file |
| All? | INDEX | UPDATE$FILE |

## 3.2 OPERATION FIELD

The operation field contains one of the following three: a mnemonic machine instruction code, a pseudo operation code which directs the assembly process, or a macro. The Z80 mnemonic machine instruction codes are listed in Appendix A and Hitachi HD64180 instruction codes are listed in Appendix D. The assembler pseudo-op codes are discussed in Chapter 5, with a summary of the pseudo-ops listed Appendix C. And the macro instructions are discussed in Chapter 6.

## 3.3 OPERAND FIELD

The operand field may contain numeric constants, character constants, ASCII strings, relocation counter references, labels, register references, operators, or expressions containing any combination of the previously mentioned items. Expressions are further described in Chapter 4.

### 3.4 COMMENT FIELD

A comment field is always preceded by a semicolon (;). Comments are ignored by the assembler but are useful for programmer documentation, and later, debugging.

# CHAPTER 4
# EXPRESSIONS

Before the pseudo operations and macros can be described, it is necessary to discuss expressions because of their complexity. Expressions consist of simple operands combined into properly formed sub-expressions by operators. Blanks and tabs are ignored between operators and operands of the expression. Each expression produces a 16-bit value during the assembly. If only 8 bits are needed, the least significant half of the 16-bit value is used.

## 4.1 NUMERIC CONSTANTS

A numeric constant is a 16-bit value in one of several number bases. The base, called the radix of the constant, is denoted by a trailing radix indicator. Any numeric constant which does not terminate with a radix indicator uses the default radix which has been initially set to decimal. The radix indicators are:

| | | |
|---|---|---|
| B | binary constant | base 2 |
| O | octal constant | base 8 |
| Q | octal constant | base 8 |
| D | decimal constant | base 10 |
| H | hexadecimal constant | base 16 |

A constant is a sequence of digits, followed by an optional radix indicator, where the digits are appropriate for the radix, i.e., binary constants must be composed of 0 and 1 digits etc. For hexadecimal constants, the leading digit must be a decimal digit in order to avoid confusing the hexadecimal constant with an identifier (a leading 0 will work). A numeric constant must produce a binary number which can be contained within a 16-bit value.

## 4.2 ASCII STRINGS

String constants represent sequences of ASCII characters, and are represented by enclosing the characters within apostrophe symbols ('). All strings must be fully contained within the current physical line. The apostrophe character itself can be included within a string by representing it as a double apostrophe (''), which becomes a single apostrophe when read by the assembler.

## 4.3 CHARACTER CONSTANTS

Like strings, character constants are composed of 0, 1, or 2 ASCII characters, delimited by an apostrophe (') or quotation (") symbol. One difference between strings and character constants is strings are used only with DB, DC, DEFB, and all macro pseudo-

ops. In all other cases, a character constant is assumed. Another difference is that the value of a character constant is calculated and the result is stored with the low byte in the first address and the high byte in the second address. For example, in the character constant:

```
DW 'AB'
```

the value of A is stored in the second memory location and B is stored in the first memory location. In the string:

```
DB 'AB'
```

the value of A is stored in the first memory location and B is stored in the second memory location.

## 4.4 LABELS

A label is given a value determined by the type of statement it precedes. If the label precedes a macro definition, the label is given a text value, which is the body of the macro definition. If the label precedes an EQU or SET pseudo operation, then the label is given the value of the operand field. If a label precedes any other type of statement, it is given the value of the current relocation counter.

The value of a label is not allowed to change unless the label precedes a SET pseudo-op. In which case, there is no limit to the number of times the label's value may change.

### 4.4.1 LABEL CHARACTERISTICS

Labels fall into one of three categories: public, external, or local. Public labels are labels defined in the current program module and can be referenced in other program modules External labels are labels which have been defined as public in some other program module and are being referenced in the module declaring them external. If a label has not been declared external or public then it is local and cannot be referenced by any other program module.

### 4.4.2 RELOCATION BASES

The symbolic names for independently located memory areas are called relocation bases. These relocation bases may represent ROM, shared COMMON areas, special memory areas such as video refresh, memory mapped I/O, etc. Within each sub-program, each of these memory areas is referenced by a unique name. The actual allocation and mapping of the name to physical addresses is deferred to the link edit and load process. All label references within the assembled program are relative to one of these relocation bases. The four relocation bases and their typical uses are summarized as follows:

**Absolute:** Absolute assembles non-relocatable code. A programmer selects Absolute mode when a block of program code is to be loaded each time into specific addresses, regardless of what else is located at the same time.

**Data Relative:** Data Relative assembles code for a section of a program that may change and therefore must be loaded into RAM. This applies especially to program data areas. Symbols in Data Relative are relocatable.

**Code Relative:** Code (program) Relative assembles code for sections of programs that will not be changed and therefore can be loaded into ROM/PROM. Symbols in Code Relative are relocatable.

**COMMON:** COMMON assembles code that is loaded into a defined common data area. This allows program modules to share a block of memory and common values.

To change the relocation base, use one of the following pseudo-ops in a statement line:

| | |
|---|---|
| ASEG | Absolute |
| DSEG | Data Relative |
| CSEG | Code Relative--default |
| COMMON | COMMON |

## 4.5 RELOCATION COUNTER REFERENCE

The current relocation counter may be referenced as a 16-bit value by use of the symbol $. The value represented by $ is always the relocation counter value at the start of the current statement. For example,

```
JP $
```

will endlessly jump to itself.

## 4.6 REGISTERS

When ZAS encounters a one or two character symbol, it will look up the symbol in the corresponding 8 or 16-bit register table (see the next page). If the symbol is found, then the operand is assumed to be a register reference. Because these single and double character symbols are reserved words, do not use them as labels.

| 8-Bit Registers (Reserved Words) | 16-Bit Registers (Reserved Words) |
|---|---|
| A | BC |
| B | DE |
| C | HL |
| D | IX |
| E | IY |
| H | SP |
| L | AF |
| M | |
| I | |
| R | |

## 4.7 OPERATORS

The operands previously described can be combined in normal algebraic expression using any combination of properly formed operands, operators, and parenthesized expressions. All arithmetic operators (+,-,*,/,MOD,SHL, and SHR) produce a 16-bit unsigned arithmetic result. The relational operators (EQ, LT, LE, GT, GE, and NE) produce a true (0FFFFH) or false (0000H) 16-bit result. And the logical operators (NOT, AND, OR, and XOR) operate bit-by-bit on their operand(s) producing a 16-bit result of 16 individual bit operations. The HIGH and LOW operators always produce a 16-bit result with a high order byte which is zero. The NUL operator produces a true or false result.

The operators for the operand field are given below. In general, the letters x and y represent operands which are treated as 16-bit unsigned quantities in the range 0-65535.

| Arithmetic Operators | Result |
|---|---|
| x+y | arithmetic sum of x and y |
| x-y | arithmetic difference between x and y |
| x * y | unsigned multiplication of x by y |
| x / y | unsigned division of x by y |
| x MOD y | remainder after division of x by y |
| x SHL y | shift left by y, with zero right fill |
| x SHR y | shift right by y, with zero left fill |

| Relational Operators | Result |
|---|---|
| x EQ y, x=y | true if x equals y, false otherwise |
| x LT y, x<y | true if x is less than y, false otherwise |
| x LE y, x<=y | true if x is less or equal to y, else false |
| x GT y, x>y | true if x is greater than y, false otherwise |
| x GE y, x>=y | true if x is greater or equal to y, else false |
| x NE y, x<>y | true if x is not equal to y, false otherwise |

| Logical Operators | Result |
|---|---|
| NOT y | bit-by-bit logical inverse of y |
| x AND y | bitwise logical AND of x and y |
| x OR y, x!y | bitwise logical OR of x and y |
| x XOR y | logical exclusive OR of x and y |

| Special Operators | Result |
|---|---|
| HIGH y | identical to y SHR 8 (high order byte of y) |
| LOW y | identical to y AND OFFH (low order byte of y) |
| NUL line | true if the remainder of the current line is null or contains only space and/or tab characters. Because the NUL operator uses the rest of the current source line as an operand, it must be the last operator on a line. |

## 4.8 PRECEDENCE OF OPERATORS

Without parentheses or brackets operators have an order of application as if they were parenthesized or bracketed. As described below, the operators listed first have highest precedence, and the operators listed last have lowest precedence. Operators listed on the same line have equal priority and are applied from left to right in the expression

| | |
|---|---|
| highest precedence | * / MOD SHL SHR |
| | + - |
| | EQ LT LE GT GE NE |
| | NOT |
| | AND |
| | OR XOR |
| | HIGH LOW |
| lowest precedence | NUL |

The expressions shown below are equivalent:

```
        x + y * z   =   x + [y * z]
x OR y * a SHR b    =   x OR [y * [a SHR b]]
```

Balanced parenthesized or bracketed sub-expressions can always be used to override the order of precedence described above. The last expression could be rewritten to force application of operators in a different order:

```
[x OR y] * [a SHR b]
```

## 4.9 PARENTHESES VERSUS BRACKETS

Parentheses and brackets are not interchangeable. They serve different purposes. Parentheses are used in expressions that have indirect addressing modes. For example,

```
LD HL, (5+1)
```

will load the register pair HL from the contents of memory location six (5+1) and seven.

Brackets are used for all other expressions where the addressing mode is not indirect. Using the above example with brackets,

```
LD HL, [5+1]
```

will load the register pair HL with the immediate value six.

## 4.10 EXPRESSION RESTRICTIONS

The operand field of a statement may consist of a complex arithmetic expression with the following restrictions:

(1) An external may only have an absolute quantity added or subtracted from it. The result will be external.

(2) A relocatable value may have an absolute or another relocatable value (in the same relocation base) added to or subtracted from it. The result will relocatable.

(3) If two relocatable values are subtracted then the result will be absolute.

(4) In all other arithmetic and logical operations, both operands must be absolute. The result will be absolute.

An expression error will be generated if an expression does not follow the above restrictions.

# CHAPTER 5
# PSEUDO-OPS

## 5.1 GENERAL PSEUDO-OPS

**DB:** The Define Byte pseudo-op is used to enter one or more one-byte data values into the program. The statement form is:

```
DB n {,n...}
```

where n is any expression with a valid 8-bit value. More than one byte can be defined at a time by separating it from the preceding value with a comma. All of the bytes defined in a single DB statement are assigned consecutive memory locations. The Zilog mnemonic DEFB can be used instead of DB.

**DC:** The Define Character pseudo-op stores the characters in a string in successive memory locations beginning with the current relocation counter. The most significant bit of the last character will be set to one. The form for the DC pseudo-op is:

```
DC 'string'
```

**DS:** The Define Space pseudo-op reserves an area of memory. The form is:

```
DS  expression {,expression}
```

where the value of the first expression gives the number of bytes to be reserved. The Zilog mnemonic DEFS can be used instead of DS.

To initialize the reserved space, set the optional second expression to the value desired. If the second expression is omitted, the reserved space is left as is (uninitialized). The reserved block of memory is not automatically initialized to zeros. To initialize to zeros give the second expression the value 0.

All names used in the first expression must be previously defined on pass 1. Otherwise, a U error (undefined symbol) is generated during pass 1, and a P error (phase error) will probably be generated during pass 2 because the DS pseudo-op generated no code on pass 1.

**DW:** The Define Word directive is used to enter a 16-bit value into the program. This directive takes the form:

```
DW nn {,nn...}
```

Where nn is any expression with a valid 16-bit value. Multiple 16-bit values may be defined with one DW statement by separating the values with a comma. All 16-bit values defined by the DW pseudo-op are stored in standard Z80 word format with the least significant byte first. The Zilog mnemonic DEFW can be used instead of DW.

**END:** The END statement is optional. All statements following the END are ignored. The form is:

END {expression}

The optional expression is the program starting address. If an Intel compatible hex file is being generated, then this starting address will be included in the last record of the hex file. If a REL file is being generated, then ZLINK will place a JUMP instruction at 100H to the specified starting address.

**EQU:** The EQUate statement is used to name synonyms for particular numeric values. The form is:

label EQU expression

The label must be present and cannot label any other statement. The assembler evaluates the expression and assigns this value to the label. The label is usually a name which describes the value of the expression. Also, this name can be used throughout the program as a parameter or operand.

**.IN:** The INsert (or MACLIB) pseudo-op allows the programmer to use the same section of assembler source code in a number of different assemblies. The format is:

.IN {d:}filename or MACLIB {d:}filename

where d is the optional Z or CP/M disk specifier (defaulting to the logged disk) and filename is the file on disk with the assumed filetype LIB.

This directive causes the specified file to be copied into the assembly in its entirety, and to be treated exactly as if it were part of the original source file. All inserted source lines are flagged with a "+" on the listing. Only one level of insert is allowed, they cannot be nested.

**.LIST:** This pseudo-op resumes a listing which has been suppressed by the .XLIST directive. See the next page.

**PAGE:** The page pseudo-op gives control over the output formatting which is sent to the PRN file and/or directly to Z or CP/M LST: device. The form for the PAGE statement is:

PAGE {expression}

If the PAGE statement is used without the optional expression then a form feed is sent to the output file and/or Z or CP/M LST: device. The form feed is sent before the statement with PAGE has been printed. Consequently, the PAGE command is often issued directly ahead of major sections of an assembly language program, such as a group of subroutines, to cause the next statement to appear at the top of the following printer page.

The second form of the PAGE command is used to specify the output

page size. In this case, the expression which follows the PAGE pseudo-op determines the number of output lines to be printed on each page. If the expression equates to a value between 40 and 90, then the page size is set to the value of the expression. When this value is reached for each page, a form feed is issued to cause a page eject. The assembler initially assumes a 56 line page size and produces a page eject at the beginning of the

listing. Usually, no more than one PAGE statement with the expression option is included in a particular program.

**.RADIX:** The statement form is:

.RADIX n

where n is 2, 8, 10, or 16. This pseudo-op sets the radix to n for all numbers which follow, unless another .RADIX statement is encountered, or the radix is overridden by a suffix radix modifier. Initially, the default radix is set to 10 (decimal).

**SET:** The SET statement is used to name synonyms for particular numeric values. The form is:

label SET expression

The label must be present and cannot label any other statement, except for another SET. The assembler evaluates the expression and assigns this value to the label. The label is usually a name which describes the value of the expression. Also, this name can be used throughout the program as a parameter or operand. The Zilog mnemonic DEFL can be used instead of SET.

**.TITLE and .SBTTL:** The title and subtitle pseudo-ops take the form:

```
.TITLE 'string-constant 1'
.SBTTL 'string-constant 2'
```

where the string-constants are an ASCII string, enclosed in apostrophes, which do not exceed 64 characters. If a .TITLE and/or .SBTTL is encountered during the assembly, then each page of the listing is prefixed with the title and/or subtitle string-constant. The title line will be preceded by a standard ZAS header as follows:

```
MITEK Relocating Macro Assembler vers n.n        page nnn
string-constant 1
string-constant 2
```

where n.n is the ZAS version number, nnn is the current page number and string-constant 1 and/or 2 is the string given in the corresponding pseudo-op. ZAS initially assumes that these pseudo-ops are not in effect. When specified, the title line, along with the subtitle line are not included in the line count for the page. Usually, no more than one .TITLE statement is included in a particular program.

## 5.2 LISTING CONTROL PSEUDO-OPS

**.LALL:** List ALL macro lines, including lines that do not generate code.

**.LIST:** This pseudo-op resumes a listing which has been suppressed by the .XLIST directive.

**.LFCOND:** The List False CONDitionals pseudo-op assures the listing of conditional expressions that evaluate false.

**.PRINT:** The print on console pseudo-op takes the form:

```
.PRINT    pass,text
```

This pseudo-op will output text to the console during the specified pass. The pass can be one of three values:

0 - print text during both passes

1 - print text during pass one

2 - print text during pass two

**.SALL:** Suppress ALL of the macro listing, including all text and object code produced by macros.

**.SFCOND:** The Suppress False CONDitionals pseudo-op suppresses the portion of the listing that contains conditional expressions that evaluate false.

**.XALL:** The EXclude ALL non-code macro lines pseudo-op will list source and object code produced by a macro, but source lines which do not generate code are not listed.

**.XLIST:** This pseudo-op suppresses all list output until a .LIST pseudo-op is encountered.

## 5.3 CONDITIONAL ASSEMBLY PSEUDO-OPS

The next two sections describe the ZAS conditional assembly facility.

### 5.3.1 IF PSEUDO-OP EVALUATION

ZAS has two different methods for evaluating the trueness of an IF expression. One method bases the trueness on the least significant bit of the IF expression, which is compatible with Digital Research's ASM, MAC, and RMAC assemblers. The second method bases the trueness of the expression on the full 16-bit expression value. This method is compatible with the Microsoft M80 assembler.

The default evaluation is set by the installation program (section 1.3). The evaluation method may also be explicitly set by the following two pseudo-ops:

.IF1 - will cause IF expressions to evaluate to true if the least significant bit of the IF expression evaluates to 1.

OR

.IF16 - will cause IF expressions to evaluate to true when the IF expression evaluates to non-zero.

### 5.3.2 CONDITIONAL ASSEMBLY FORMS

The IF, ELSE, and ENDIF pseudo-ops define a range of assembly language statements which are to be included or excluded during the assembly process. The IF and ENDIF statements alone can be used to bound a group of statements to be conditionally assembled thus:

```
IF      expression
statement #1
statement #2
      .
      .
      .
statement #n
ENDIF
```

Upon encountering the IF statement, the assembler evaluates the expression following the IF (all operands in the expression must be defined ahead of the IF statement). Depending on the conditional assembly option in effect, if the expression evaluates to a non-zero value or the least significant bit evaluates to a 1, then statement #1 through statement #n are assembled. If the expression evaluates to a zero, then the statements are listed but not assembled.

The ELSE statement can be used as an alternative to an IF statement, and must occur between the IF and ENDIF statements. The form is:

```
IF      expression
statement #1
statement #2
      .
      .
      .
statement #n
ELSE
statement #n+1
statement #n+2
      .
      .
      .
statement #m
ENDIF
```

If the expression produces a non-zero (true) value, then statements 1 through n are assembled. However, statements n+1 through m are skipped in the assembly process. When the expression produces a zero value (false), statements 1 through n are skipped, while statements n+1 through m are assembled. As an example, the conditional assembly shown in Listing A could be rewritten as shown in Listing B.

Listing A

```
TTY     EQU     1
CRT     EQU     2
DEVICE  EQU     TTY
TTYOUT  EQU     0F003H
CRTOUT  EQU     0F100H
        IF      DEVICE  EQ      TTY
        CALL    TTYOUT
        ENDIF
        IF      DEVICE  EQ      CRT
        CALL    CRTOUT
        ENDIF
```

Listing B

```
TTY     EQU     1
CRT     EQU     2
DEVICE  EQU     TTY
TTYOUT  EQU     0F003H
CRTOUT  EQU     0F100H
        IF      DEVICE  EQ      TTY
        CALL    TTYOUT
        ELSE
        CALL    CRTOUT
        ENDIF
```

Properly balanced IF's, ELSE's, and ENDIF's can be completely contained within the boundaries of outer encompassing conditional assembly groups. The structure outlined below shows properly nested IF, ELSE, and ENDIF statements:

```
IF      exp#1
group #1
IF      exp#2
group#2
ELSE
group#3
ENDIF
group#4
ELSE
group#5
IF    exp#3
group#6
ENDIF
group#7
ENDIF
```

where group 1 through 7 are sequences of statements to be conditionally assembled, and exp#1 through exp#3 are expressions which control the conditional assembly. If exp#1 is true, then group#1 and group#4 are always assembled, and group 5,6, and 7 will be skipped. Further, if exp#1 and exp#2 are both true, then group#2 will also be included in the assembly, otherwise group#3 will be included. If exp#1 produced a false value, groups 1, 2, 3, and 4 will be skipped, and group 5 and 7 will always be assembled. If under these circumstances, exp#3 is true then group#6 will also be included with 5 and 7, otherwise it will be skipped in the assembly.

Conditional assembly of this sort can be nested up to eight levels (i.e., there can be up to eight pending IFs or ELSEs with unresolved ENDIFs at any point in the assembly), but usually becomes unreadable after two or three levels of nesting. The nesting level restriction also holds for pending IFs and ELSEs during macro evaluation. Nesting level overflow will produce an error during assembly.

## 5.4 LINKAGE PSEUDO-OPS

**EXTRN:** The EXTeRNal pseudo-op identifies symbols which are defined in some other program but are used in the current program. The form is:

```
EXTRN symbol {,symbol...}
```

where symbol is the symbol being declared as external. Multiple symbols may be declared in the same statement by separating them with commas. Also, if a symbol in an expression is suffixed with one or two # signs, then the symbol is treated as an external. EXT is a synonym for EXTRN.

**NAME:** The NAME pseudo-op takes the form:

```
NAME symbol
```

where symbol is the relocatable module name. This name is used by the linking loader and library manager to identify the module for selective loading or manipulation. Only the first six characters are significant in the module name. In the absence of the NAME pseudo-op, up to the first six characters of the program name are used.

**PUBLIC:** The PUBLIC pseudo-op identifies those symbols within the current program which are to be made accessible to other programs as external symbols. This directive has no effect on the assembly process for the current program, but merely records the name and value of the identified symbols on the object file for later use by the linking loader. A public symbol must be defined within the current program as a label.

**.REQUEST:** Request a library search. The form is:

.REQUEST filename {,filename...}

This pseudo-op sends a request to ZLINK or any Microsoft compatible loader to search the filenames in the list for undefined external symbols. The filename in the list should not include filetypes or device designation. ZLINK assumes the default extension .REL and the currently logged disk drive.

## 5.5 RELOCATION BASE PSEUDO-OPS

**ASEG:** The Absolute SEGment pseudo-op never has operands. ASEG generates non-relocatable code.

ASEG sets the location counter to an absolute segment (actual address) of memory. The ASEG will default to 0, which could cause the module to write over part of the operating system. It is recommended that each ASEG be followed with an ORG statement set at 100H or higher.

**COMMON:** COMMON statements are non-executable, storage allocating statements. COMMON assigns variables, arrays, and data to a storage area called COMMON storage. This allows various program modules to share the same storage area. The length of a COMMON area is the number of bytes required to contain the variables, arrays, and data declared in the COMMON block, which ends when another relocation base pseudo-op is encountered.

**CSEG:** The Code SEGment directive never has an operand. Code assembled in Code Relative mode can be loaded into ROM/PROM.

CSEG resets the location counter to the code relative segment of memory. The location will be that of the last CSEG (default to 0), unless an ORG is done after the CSEG to change the location.

However the ORG statement does not set a hard absolute address under CSEG mode. An ORG statement under CSEG causes the assembler to add the number of bytes specified by the expression argument in the ORG statement to the last CSEG address loaded. For example, if ORG 25 is given, 25 bytes will be added to the current CSEG location. Then CSEG will be loaded. The clearing effect of the ORG statement following CSEG (and DSEG) can be used to give the module an offset. Rationale for not allowing ORG to set an absolute address for CSEG is to keep the CSEG relocatable.

CSEG is the default mode of the assembler. Assembly begins with a CSEG automatically executed, and the location counter in the Code Relative mode, pointing to location 0 in the Code Relative segment of memory. All subsequent instructions will be assembled into the Code Relative segment of memory until ASEG, DSEG, or COMMON is executed. CSEG is then entered to return the assembler to Code Relative mode, at which point the location counter returns to the next free location in the Code Relative segment.

**DSEG:** The Data SEGment pseudo-op never has operands. DSEG specifies segments of assembled relocatable code that will later be loaded into RAM only.

DSEG sets the location counter to the Data Relative segment of memory. The location of the data relative counter will be that of the last DSEG (default is 0), unless an ORG is done after the DSEG to change the location. However, the ORG statement does not set a hard absolute address under DSEG mode. An ORG statement under DSEG causes the assembler to add the number of bytes specified by the expression in the ORG statement to the last DSEG address loaded. For example, if ORG 25 is given, 25 bytes will be added to the last DSEG address loaded. Then the DSEG will be loaded. The clearing effect of the ORG statement following DSEG (and CSEG) can be used to give the module an offset. Rational for not allowing ORG to set an absolute address for DSEG is to keep the DSEG relocatable.

**ORG:** The Set ORGin pseudo-op allows the value of a location counter to be changed at any time. The form is:

```
ORG expression
```

Under the ASEG program counter mode, the relocation counter is set to the value of the expression, and the assembler assigns generated code starting with that value. Under CSEG, DSEG, and COMMON relocation bases, the location counter for that base is incremented by the value of the expression. All names used in the expression must be known on pass 1, and the value must either be absolute or in the same relocation base as the current location counter.

**.PHASE/.DEPHASE:** The form is:

```
.PHASE    expression
   .
   .
   .
.DEPHASE
```

where expression is an absolute value. .PHASE allows code to be located in one area, but executed at a different area with a start address specified by expression. .DPHASE is used to indicate the end of the relocated block of code.

The relocation base within a .PHASE block is absolute, the same as the mode of the expression in the .PHASE statement. The code, however, is loaded in the area in effect when the .PHASE statement is encountered. The code within the block is later moved to the address specified by expression for execution.

This example,

```
                        .PHASE      300H
                        CALL        DUMMY
                        JP          ENTRY
              DUMMY:    RET
                        .DEPHASE
              ENTRY:    JP          0
```

assembles to:

```
0300                    .PHASE      300H
0300    CD0630          CALL        DUMMY
0303    C30700          JP          ENTRY
0306    C9        DUMMY:RET
0007                    .DEPHASE
0007    C30000    ENTRY:JP          0
```

## 5.6 MACRO PSEUDO-OPS

Provided here is only a brief description of the macro pseudo-ops. For a more complete description, see the next chapter.

| Pseudo-op | Description |
|---|---|
| ENDM | End Macro |
| EXITM | Exit Macro |
| IRP | Indefinite Repeat |
| IRPC | Indefinite Repeat Character |
| LOCAL | Local Symbol Generation |
| REPT | Repeat |
| MACRO | Macro Definition |

## 5.7 SPECIAL FUNCTION PSEUDO-OPS

**.HD64:** This pseudo-op enables ZAS to assemble the ten extended instructions of the Hitachi HD64180 microprocessor, upward Z80 compatible. The ten instructions and their forms are listed in Appendix D.

# CHAPTER 6
# MACRO FACILITY

A common characteristic of assembly language programs is that many coding sequences are repeated over and over with only one or two of the operands changing. Macros provide a mechanism for generating the repeated sequences with a single statement. The repeated sequences are written with dummy values for the changing operands. A single statement, referring to the macro by name and providing values for the dummy operands, can then generate the repeated sequence.

The coding sequence begins with either the macro definition pseudo-op or one of the repeat pseudo-ops and ends with the ENDM pseudo-op. All of the macro pseudo-ops may be used inside a macro sequence. The one exception is a stored macro which, cannot be defined inside a repeat type macro. Macro nesting is allowed up to 15 levels deep.

The macro facility includes pseudo-ops for:

macro definition:
    MACRO (macro definition)

repetitions
    REPT (repeat)
    IRP (indefinite repeat)
    IRPC (indefinite repeat character)

terminations:
    ENDM (end macro)
    EXITM (exit macro)

unique symbols within macro sequences:
    LOCAL

operators:
    &
    ;;
    ^
    %
    <>

## 6.1 REPEAT (OR INLINE) MACROS

The simplest macro facilities involve the REPT, IRPC, and IRP macro groups. All these forms cause the assembler to repetitively re-read portions of the source program under control of a counter or list of textual substitutions. These groups are listed in increasing order of complexity.

**REPT-ENDM GROUP:** The REPT-ENDM group is written as a sequence of assembly language statements starting with the REPT pseudo-op and terminated by an ENDM pseudo-op. The form is:

```
label: REPT expression
    .
    .
    .
label: ENDM
```

where the labels are optional, and the expression indicates the number of times the sequence of statements between REPT and ENDM will be repeated. The expression is evaluated as a 16-bit unsigned number. If the expression contains an external symbol or undefined operands, an error is generated.

In general, if a label appears on the REPT statement, its value is the first machine code address which follows. This REPT label is not re-read on each repetition of the loop. The optional label on the ENDM is re-read on each iteration and thus constant labels (not generated through concatenation or with LOCAL pseudo-ops) will generate phase errors if the repetition count is greater than 1.

**IRPC-ENDM GROUP:** Similar to the REPT group, the IRPC-ENDM group causes the assembler to re-read a bounded set of statements. The form is:

```
label: IRPC identifier,string
    .
    .
    .
label: ENDM
```

where the optional labels follow the same conventions as in the REPT-ENDM group. The identifier is any valid symbol and string denotes a string of characters, terminated by a delimiter (space, tab, end-of-line, or comment).

The sequence of statements between IRPC and ENDM are repeated once for each character in the string. Each repetition substitutes the next character in the string for every occurrence of identifier in the sequence.

**IRP-ENDM GROUP:** The IRP is similar in function to the IRPC, except that the controlling identifier can take on a multiple string value. The form is:

```
label: IRP identifier, string {,string...}
    .
    .
    .
label: ENDM
```

where the optional labels follow the conventions of the REPT and IRPC groups. The sequence of statements between IRP and ENDM is repeated for each string. On the first iteration, the string is

substituted for the identifier wherever the identifier occurs in the sequence of statements. On the second iteration, the second string becomes the value of the controlling identifier and so on until the last string is encountered and processed.

## 6.2 STORED MACROS

**MACRO DEFINITION:** The form for the macro definition is:

```
macname MACRO dummy{,dummy...}
          .
          .
          .
        ENDM
```

The sequence of statements from the MACRO statement line to the ENDM statement line comprises the body of the macro, or the macro's definition. The macname is any non-conflicting assembly language label. Dummy parameter is a place holder that is replaced by an actual parameter in a one for one text substitution when the MACRO sequence is used.

The prototype statements are read and stored in the assembler's internal tables under the name given by "macname", but are not processed until the macro is expanded.

A comment preceded by two semicolons is not saved as part of the macro definition. But a comment preceded by only one semicolon is preserved and will appear in the expansion.

## 6.3 EXITING MACROS

The EXITM pseudo-op is used inside a MACRO or Repeat block to terminate an expansion when some condition makes the remaining expansion unnecessary or undesirable. Usually, EXITM is used in conjunction with a conditional pseudo-op.

The expansion is exited immediately when an EXITM is assembled. Any remaining expansion or repetition is not generated. If the block containing the EXITM is nested within another block, the outer level continues to be expanded.

## 6.4 LOCAL SYMBOLS

The LOCAL pseudo-op is allowed only inside a MACRO definition. The form for the LOCAL directive is:

```
LOCAL identifier {,identifier...}
```

When LOCAL is executed, ZAS creates a unique symbol for each identifier and substitutes that symbol for each occurrence of the identifier in the expansion. These unique symbols are usually used to define a label within a macro. This eliminates multiple-

defined labels on successive expansions of the macro. The symbols created by ZAS range from ??0001 to ??9999. Users should avoid the form ??nnnn for their own symbols. A LOCAL statement must precede all other types of statements in the macro definition.

## 6.5 MACRO INVOCATION

The form for the macro invocation is:

```
macname parameter{,parameter...}
```

Upon recognition of the macname, ZAS "pairs-off" each dummy parameter in the MACRO definition with the actual parameter text, i.e., the first dummy parameter is associated with the first actual parameter, the second dummy is associated with the second actual, and so on until the list is completed. If more actuals are provided than dummy parameters then the extras are ignored. If fewer actuals are provided, then the extra dummy parameter are associated with the empty string, i.e., a text string of zero length. It is important to realize at this point that the value of dummy parameter is not a numeric value, but is instead a textual value consisting of a sequence of zero or more ASCII characters.

## 6.6 PARAMETER EVALUATION

There are several options available in the construction of actual parameters, as well as in the specification of character lists for the IRP group. Although an actual parameter is simply a sequence of characters placed between parameter delimiters, these options allow overrides where delimiter characters themselves become a part of the text. In general, a parameter x occurs in the context:

```
label: macname  ...,x,...
```

where the label is optional and the macname is the name of a previously defined macro. The ellipses (...) represent optional surrounding actual parameters in the invocation of macname. In the case of an IRP group, the occurrence of a character list x would be:

```
label: IRP id, ...,x,...
```

where the label is optional, and the ellipses represent optional surrounding character lists for substitution within the IRP group where the controlling identifier "id" is found. In either case, the statements could be contained within the scope of a surrounding macro expansion. Therefore, dummy parameter substitution could take place for the encompassing macro while the actual parameter is being scanned.

ZAS follows these steps in forming an actual parameter or character list:

(1) Leading blanks and tabs are removed when they occur in front of x.

(2) The leading character of x is examined to determine the type of scan operation which is to take place.

(3) If the leading character is a string quote, then x becomes the text up through and including the balancing string quote, using the normal string scanning rules: double apostrophes within the string are reduced to a single apostrophe, and upper case dummy parameters adjacent to the ampersand symbol are substituted by their actual parameter values. Note that the string quotes on either end of the string are included in the actual parameter text.

(4) If instead the first character is the left caret (<) then the bracket is removed, and the value of x becomes the sequence of characters up to , but not including, the balancing right caret (>) which does not become part of x. In this case, left and right carets may be nested to any level within x, and only the outer carets are removed in the evaluation. Quoted strings within the carets are allowed, and substitution within these strings follows the rules stated in (3) above. Note that left and right carets within quoted strings become a part of the string, and are not counted in the caret nesting within x. Further, the delimiter characters comma, blank, semicolon, and tab, become a part of x when they occur within the caret nesting.

(5) If the leading character is a %, then the sequence of characters which follows is taken as an expression which is evaluated immediately as a 16-bit value. The resulting value is converted to a decimal number and treated as an ASCII sequence of digits, with left zero suppression (0-65535).

(6) If the leading character is none of the above (quote, left bracket, or percent), the sequence of characters which follow, up to the next comma, blank, tab, or semicolon, becomes the value of x.

There is one important exception to the above rule: the single character escape, denoted by an up-arrow, causes ZAS to read the character immediately following as a part of x without treating the character as significant. However, the character which follows the up-arrow, must be a blank, tab, or visible ASCII character. The up-arrow itself can be represented by two up-arrows in succession. If the up-arrow directly precedes a dummy parameter, then the up-arrow is removed and the dummy parameter is not replaced by its actual parameter value. Thus, the up-arrow can be used to prevent evaluation of dummy parameters within the macro body. Note that the up-arrow has no special significance within string quotes, and is simply included as a part of the string.

Evaluation of dummy parameters in macro expansions must also be considered, although this topic has been presented throughout the

previous sections. Generally the macro assembler evaluated dummy parameters as follows:

(1) If a dummy parameter is either preceded or followed by the concatenation operator (&), then the preceding and/or following "&" operator is removed, the actual parameter is substituted for the dummy parameter, and the implied delimiter is removed at the position(s) the ampersand occurs.

(2) Dummy parameters are replaced only once at each occurrence as the encompassing macro expands. This prevents the "infinite substitution" which would occur if a dummy parameter evaluated itself.

In summary, parameter evaluation follows these rules:

- leading and trailing tabs and blanks are removed
- quoted strings are passed with their string quotes intact
- nested carets enclose arbitrary characters with delimiters
- a leading % causes immediate numeric evaluation
- an up-arrow passes a special character as a literal value
- an up-arrow prevents evaluation of a dummy parameter
- the "&" operator is removed next to a dummy parameter
- dummy parameters are replaced only once at each occurrence

# CHAPTER 7
# ZAS ERROR MESSAGES

There are two types of error messages: Non-fatal errors and fatal errors. Non-fatal errors are indicated by a single letter code to the left of the statement line with the error. Fatal errors kill the assembly and give messages as to why the error may have occurred. Statement lines with errors will not generate object code.

## 7.1 NON-FATAL ERRORS

| Error Code | Explanation |
|---|---|
| **A** | **Argument error.** One of the arguments for the op-code is invalid. |
| **B** | **Balance error.** An ELSE or an ENDIF pseudo-op does not have a preceding IF statement. Or an END macro statement has no preceding macro call and/or macro definition. |
| **C** | **Character is invalid.** ZAS has found an invalid character and it is probably a control character. The invalid character will be replaced by a "^". |
| **D** | **Duplicate error.** A label has been defined more than once. |
| **E** | **Expression error.** The expression is ill-formed and cannot be computed. |
| **I** | **Insert error.** The specified insert file cannot be found or an insert is already in progress. |
| **M** | **Mode error.** The statement contains an addressing mode error. |
| **O** | **Opcode error.** The statement contains an illegal opcode. |
| **P** | **Phase error.** A label has a different value on Pass 2 than it did on Pass 1. |
| **S** | **Syntax error.** The assembly statement is ill-formed and cannot be processed. This error may be due to invalid characters or delimiters which are out of place. |
| **U** | **Undefined symbol.** A label argument has not been defined in the program. |
| **V** | **Value error.** The operand (argument) is out of its allowable range. |

## 7.2 FATAL ERRORS

Fatal error messages have been classified into two categories: errors caused by macros and general errors (or errors not caused by macros).

### 7.2.1 GENERAL FATAL ERROR MESSAGES

(1) **"Filename.filetype not found."**
The specified source file cannot be found on the disk.

(2) **"Invalid option specification."**
One or more of the assembler options specified in the command line is invalid.

(3) **"More than eight IF levels are pending at line nnnn"**
Where line nnnn is the line with the ninth IF. A maximum of eight IF levels can be nested.

(4) **"Unterminated IF!"**
The end of file has been reached with no terminating ENDIF.

(5) **"Memory full at line nnnn"**
The assembler's internal tables have run out of memory.

### 7.2.2 MACRO FATAL ERROR MESSAGES

(1) **"Unterminated macro starting at line nnnn"**
Where line nnnn is the line with the error. This error is caused by a macro definition that has no terminating END macro statement.

(2) **"Local label limit exceeded!"**
The maximum of 9,999 local symbols has been exceeded.

(3) **"Macro nested past 16 levels at line nnnn"**
A maximum of 16 levels of nested macros are allowed.

(4) **"Local table exceeds 127 bytes at line nnnn"**
The total length of all local symbols cannot exceed 127 bytes for a particular macro definition.

(5) **"Macro definition inside an inline macro at line nnnn"**
This message indicates that a macro definition has been placed inside a repeat type macro and that is not allowed.

# CHAPTER 8
# CROSS-REFERENCE GENERATION

## 8.1 OVERVIEW

The cross-reference generator (ZREF) is used to provide a summary of symbol usage throughout a program. ZREF reads the file specified line by line, attaches a line number prefix to each line, and writes each prefixed line to the file filename.XRF. After completing this operation, ZREF appends to the file filename.XRF, a cross-reference report that lists all the line numbers where each symbol in the file appears. It also flags with an *, each line number where the referenced symbol is defined.

## 8.2 ZREF OPERATION

ZREF is invoked by typing

ZREF filename.filetype {$}option

where filename.filetype is the name of the file to be cross-referenced with the assumed filetype .Z80, and option is the letter L, if the output is to the list device instead of a file.

## 8.3 RESERVED SYMBOLS

The following symbols will not be part of the cross reference:

| | | |
|---|---|---|
| A | HI | NUL |
| AF | HL | NOT |
| AND | I | NZ |
| B | IX | OR |
| BC | IY | P |
| C | L | PE |
| D | LE | PO |
| DE | LOW | R |
| E | LT | SHL |
| EQ | M | SHR |
| GE | MOD | SP |
| GT | NC | XOR |
| H | NE | Z |

(THIS PAGE INTENTIONALLY LEFT BLANK.)

# CHAPTER 9
# CODE CONVERTER

## 9.1 CODE CONVERTER OPERATION

The code converter (ZCON) converts 8080 source statements, all of the TDL machine instruction statements, and most of the common TDL pseudo-ops to Z80 source statements (see the next section for a listing of the convertible TDL pseudo-ops). In addition, except for character-constants, ASCII strings, and comments, parentheses are converted to brackets. Also, parity bit (bit 7) is zeroed.

To invoke the code converter, type:

ZCON filename.filetype {$}u

where filename is the name of the source file to be converted. If no filetype is specified, then ASM is assumed. When the "u" option is specified, only upper-case conversion is done. This is useful if you already have a Z80 source file in lower case. When the conversion is completed, the output will be in a file called filename.Z80 and one of two messages will be displayed.

Message 1:
**"nnnn lines converted, with no errors detected."**
Where nnnn is the number of lines converted.

OR

Message 2:
**"nnnn lines converted, with eee errors logged in filename.ERR"**
Where nnnn is the number of lines converted and eee is the number of errors detected.

## 9.2 CONVERTIBLE TDL PSEUDO-OPS

The code converter will convert the most common TDL pseudo-ops. They include the following:

| | |
|---|---|
| .ASCII | .IDENT |
| .BLKB | .INTERN |
| .BLKW | .LIST |
| .BYTE | .WORD |
| .EXTERN | .XLIST |

## 9.3 ERROR MESSAGES

If the code converter detects an error in a statement line, it leaves the line unchanged. There are two types of error messages.

(1) **"*** Syntax error at line nnn, line follows ***"**
error line

Where nnn is the statement line number, and error line is the statement line with the syntax error. Normally, this error should not occur because it indicates that the operand for this particular op-code is syntactically incorrect.

(2) **"*** IF/ENDIF unbalanced ***"**

This error message appears if the IFs and ENDIFs are not paired. For every IF, there should be an ENDIF, and vice versa.

# CHAPTER 10
# LINKER

## 10.1 OVERVIEW

The Z80 Linker (ZLINK) is used to combine Microsoft relocatable object modules into an absolute file ready for execution under Z or CP/M. When completed, ZLINK lists the sorted symbol table, any unresolved or duplicate symbols, and a load map which shows the number of free bytes left and the size and locations of the different segments:

```
LOAD MAP FOR FILENAME.COM

SEGMENT    SIZE    START    STOP
ABSOLUTE
CODE
DATA
COMMON
FREE
```

ZLINK writes the sorted symbol table to a .SYM file suitable for use with Echelon Dynamic Screen Debugger (DSD) and Digital Research Symbolic Instruction Debuggers (SID and ZSID) as described in the S option (see next page). ZLINK also creates a COM file for direct execution under Z or CP/M. If errors are detected, the P option (see next page) will be set automatically.

## 10.2 ZLINK OPERATION

ZLINK is invoked by typing

```
ZLINK filename1{,filename2,...,filenameN}
```

where filename is the name of the object module(s) to be linked. If no filetype is specified, then REL is assumed. If some other filename is desired for the COM and SYM files, it may be specified as follows:

```
ZLINK newfilename=filename1 {,filename2,...filenameN}
```

If ZLINK encounters a starting address which is caused by supplying an optional program starting address to the assembler END pseudo-op then ZLINK will place a JUMP instruction at 100H to the program starting address.

## 10.3 ZLINK OPTIONS

A variety of options are available to provide control over the execution parameters of ZLINK. Except for the / option (library

search option) all of the options are link control options. They are used once at the end of a command line:

```
filename1{,filename2,...filenameN} $Cnnnn,Dnnnn,P,Rnnnn
```

Where nnnn is a hexadecimal number.

ZLINK options include:

**C: Code Segment Origin Option.** The C option is used to specify the load address of the code segment. If it is not used, then ZLINK will put the code segment at the address (100H). Unless the R option indicates otherwise, the relocation value of the code segment will be set to its load address. The syntax for the C option is Cnnnn, where nnnn if the desired code origin in hex.

**D: Data Origin Option.** The D option indicates the load address of the data and common segments. If the D option is used, the address specified must be higher than the load address for the code segment. If it is not used, ZLINK will put the data and common segments immediately after the program segment. The syntax for the D option is Dnnnn, where nnnn is the desired data origin in hex.

**P: Paging Option.** The P option will page the output of ZLINK, at 23 lines per page to the terminal. Pressing any key allows you to continue to output one page at a time.

**R: Relocate Origin Option.** The R option specifies the relocation value for the code segment. If not used, then ZLINK will set the relocation value of the code segment to its load address.

**S: .SYM File Option.** If this option is set, ZLINK will write the sorted symbol table to a .SYM file suitable for use with the Echelon DSD or Digital Research SID and ZSID debuggers.

**/: Search Option.** This option is used to indicate that the preceding file should be treated as a library. ZLINK will search the file and include only those modules containing symbols which are referenced but not defined in the modules already linked. Unlike the link control options which can be used once at the end of a command line, the / option must be used after each filename to be searched:

```
filename1/,filename2/,...filenameN/
```

## 10.4 DEFINE NEXT FREE MEMORY LOCATION

If the public symbol $MEMRY is encountered during the link process, then the two bytes addressed by the value $MEMRY and $MEMRY + 1 are filled in with the address of the next free memory location. The statement labeled $MEMRY must be a DS statement.

For example:

```
                PUBLIC    FREBEG,$MEMRY
FREBEG:         LD        HL,($MEMRY)     ;This routine returns
                RET                       ;the first free byte
$MEMRY:         DS        2               ;of memory
```

## 10.5 ZLINK ERROR MESSAGES

(1) **"Can't find filename.filetype"**

Specified file cannot be found on the disk.

(2) **"Filename.filetype is an invalid REL file!"**

One of the files specified is not a Microsoft compatible REL file.

(3) **"Invalid option specification!"**

One of the options specified is invalid.

(4) **"Memory full!"**

There is insufficient memory to complete the linking process.

(5) **"Undefined symbols:"**

The symbol name(s) following this heading are referenced but not defined in any of the modules being linked.

(6) **"Duplicate symbols:"**

The symbol name(s) following this heading are defined as a PUBLIC symbol in more than one of the modules being linked.

(7) **"***Overlapping segments***"**

ZLINK attempted to write a segment into memory already used by another segment. This error is probably caused by incorrect use of the C and/or D options.

(8) **"Read error!"**

A file cannot be read properly.

(9) **"Syntax error in command line!"**

The command line is ill formed.

(10) **"Multiple main modules!"**
Two or more modules contain a program starting address.

(11) **"Library search limit exceeded!"**
A maximum of ten libraries can be specified from assembler .REQUEST statements.

(THIS PAGE INTENTIONALLY LEFT BLANK.)

# CHAPTER 11
# LIBRARY MANAGER

## 11.1 OVERVIEW

The Library Manager (ZLIB) is used to combine Microsoft relocatable object modules into a library. Libraries are files consisting of any number of relocatable object modules. ZLIB can delete modules from a library, concatenate REL files into a library, re-place modules in a library, and print module names and public symbols from a library.

## 11.2 ZLIB OPERATION

ZLIB is invoked by typing:

```
ZLIB libname=filename{,filename,...} $option
```

where libname is the name of the library with filetype REL and filename is the name of the object module(s). If no filetype is specified, then REL is assumed.

An alternate form of invoking ZLIB when using the M or P option (as described below) is:

```
ZLIB libname $listoption
```

where listoption is the M or P option.

## 11.3 ZLIB OPTIONS

If no option is specified, then the specified modules will be appended to the library. The options include:

**D:** Delete the specified modules.

**M:** Print the module names in the library.

**P:** Print the module names and public symbols in the library.

**R:** Replace the specified modules.

## 11.4 ZLIB MESSAGES

Under the following circumstances ZLIB will produce messages.

(1) When a module is being appended to the library:

**"Appending filename.filetype"**

(2) If the specified library does not exist on disk and the specified option is append:

**"Creating library"**

(3) If a module is being deleted:

**"Deleting modulename"**

(4) If a module is being replaced:

**"Deleting modulename**
**Appending filename.filetype"**

## 11.5 ZLIB ERROR MESSAGES

(1) **"Can't find filename.filetype"**

Specified file cannot be found on the disk.

(2) **"Filename.filetype is an invalid REL file!"**

One of the files specified is not a Microsoft compatible REL file.

(3) **"Invalid option specification!"**

The option specified is invalid.

(4) **"Syntax error in command line!"**
The command line is ill formed.

## APPENDIX A
## Z80 MNEMONIC MACHINE INSTRUCTION CODES

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| 8E | ADC | A,(HL) | Add with Carry Oper- | Leading A Oper- |
| DD8E05 | ADC | A,(IX+d) | and to Acc. | and is Optional |
| FD8E05 | ADC | A,(IY+d) | | |
| 8F | ADC | A,A | | If d is Omitted |
| 88 | ADC | A,B | | 0 is Assumed |
| 89 | ADC | A,C | | |
| 8A | ADC | A,D | | |
| 8B | ADC | A,E | | |
| 8C | ADC | A,H | | |
| 8D | ADC | A,L | | |
| CE20 | ADC | A,n | | |
| ED4A | ADC | HL,BC | Add with Carry Reg. | |
| ED5A | ADC | HL,DE | Pair to HL | |
| ED6A | ADC | HL,HL | | |
| ED7A | ADC | HL,SP | | |
| 86 | ADD | A,(HL) | Add Operand to Acc. | Leading A Oper- |
| DD8605 | ADD | A,(IX+d) | | and is Optional |
| FD8605 | ADD | A,(IY+d) | | |
| 87 | ADD | A,A | | If d is Omitted |
| 80 | ADD | A,B | | 0 is Assumed |
| 81 | ADD | A,C | | |
| 82 | ADD | A,D | | |
| 83 | ADD | A,E | | |
| 84 | ADD | A,H | | |
| 85 | ADD | A,L | | |
| C620 | ADD | A,n | | |
| 09 | ADD | HL,BC | Add Reg. Pair to HL | |
| 19 | ADD | HL,DE | | |
| 29 | ADD | HL,HL | | |
| 39 | ADD | HL,SP | | |
| DD09 | ADD | IX,BC | Add Reg. Pair to IX | |
| DD19 | ADD | IX,DE | | |
| DD29 | ADD | IX,IX | | |
| DD39 | ADD | IX,SP | | |
| FD09 | ADD | IY,BC | Add Reg. Pair to IY | |
| FD19 | ADD | IY,DE | | |
| FD29 | ADD | IY,IY | | |
| FD39 | ADD | IY,SP | | |
| A6 | AND | A,(HL) | Logical 'AND' of | Leading A Oper- |
| DDA605 | AND | A,(IX+d) | Operand and Acc. | and is Optional |
| FDA605 | AND | A,(IY+d) | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| A7 | AND | A,A | Logial 'AND' of | Leading A Oper- |
| A0 | AND | A,B | Operand and Acc. | and is Optional |
| A1 | AND | A,C | | |
| A2 | AND | A,D | | If d is Omitted |
| A3 | AND | A,E | | 0 is Assumed |
| A4 | AND | A,H | | |
| A5 | AND | A,L | | |
| E620 | AND | A,n | | |
| CB46 | BIT | 0,(HL) | Test Bit of Location | If d is Omitted |
| DDCB0546 | BIT | 0,(IX+d) | or Reg. | 0 is Assumed |
| FDCB0546 | BIT | 0,(IY+d) | | |
| CB47 | BIT | 0,A | | |
| CB40 | BIT | 0,B | | |
| CB41 | BIT | 0,C | | |
| CB42 | BIT | 0,D | | |
| CB43 | BIT | 0,E | | |
| CB44 | BIT | 0,H | | |
| CB45 | BIT | 0,L | | |
| CB4E | BIT | 1,(HL) | | |
| DDCB054E | BIT | 1,(IX+d) | | |
| FDCB054E | BIT | 1,(IY+d) | | |
| CB4F | BIT | 1,A | | |
| CB48 | BIT | 1,B | | |
| CB49 | BIT | 1,C | | |
| CB4A | BIT | 1,D | | |
| CB4B | BIT | 1,E | | |
| CB4C | BIT | 1,H | | |
| CB4D | BIT | 1,L | | |
| CB56 | BIT | 2,(HL) | | |
| DDCB0556 | BIT | 2,(IX+d) | | |
| FDCB0556 | BIT | 2,(IX+d) | | |
| CB57 | BIT | 2,A | | |
| CB50 | BIT | 2,B | | |
| CB51 | BIT | 2,C | | |
| CB52 | BIT | 2,D | | |
| CB53 | BIT | 2,E | | |
| CB54 | BIT | 2,H | | |
| CB55 | BIT | 2,L | | |
| CB5E | BIT | 3,(HL) | | |
| DDCB055E | BIT | 3,(IX+d) | | |
| DFCB055E | BIT | 3,(IY+d) | | |
| CB5F | BIT | 3,A | | |
| CB58 | BIT | 3,B | | |
| CB59 | BIT | 3,C | | |
| CB5A | BIT | 3,D | | |
| CB5B | BIT | 3,E | | |
| CB5C | BIT | 3,H | | |
| CB5D | BIT | 3,L | | |
| CB66 | BIT | 4,(HL) | | |
| DDCB0566 | BIT | 4,(IX+d) | | |
| FDCBO566 | BIT | 4,(IY+d) | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| CB67 | BIT | 4,A | Test Bit of Location | If d is Omitted |
| CB60 | BIT | 4,B | or Reg. | 0 is Assumed |
| CB61 | BIT | 4,C | | |
| CB62 | BIT | 4,D | | |
| CB63 | BIT | 4,E | | |
| CB64 | BIT | 4,H | | |
| CB65 | BIT | 4,L | | |
| CB6E | BIT | 5,(HL) | | |
| DDCB056E | BIT | 5,(IX+d) | | |
| FDCB056E | BIT | 5,(IY+d) | | |
| CB6F | BIT | 5,A | | |
| CB68 | BIT | 5,B | | |
| CB69 | BIT | 5,C | | |
| CB6A | BIT | 5,D | | |
| CB6B | BIT | 5,E | | |
| CB6C | BIT | 5,H | | |
| CB6D | BIT | 5,L | | |
| CB76 | BIT | 6,(HL) | | |
| DDCB0576 | BIT | 6,(IX+d) | | |
| FDCB0576 | BIT | 6,(IY+d) | | |
| CB77 | BIT | 6,A | | |
| CB70 | BIT | 6,B | | |
| CB71 | BIT | 6,C | | |
| CB72 | BIT | 6,D | | |
| CB73 | BIT | 6,E | | |
| CB74 | BIT | 6,H | | |
| CB75 | BIT | 6,L | | |
| CB7E | BIT | 7,(HL) | | |
| DDCB057E | BIT | 7,(IX+d) | | |
| FDCB057E | BIT | 7,(IY+d) | | |
| CB7F | BIT | 7,A | | |
| CB78 | BIT | 7,B | | |
| CB79 | BIT | 7,C | | |
| CB7A | BIT | 7,D | | |
| CB7B | BIT | 7,E | | |
| CB7C | BIT | 7,H | | |
| CB7D | BIT | 7,L | | |
| ********** | ********** | ********** | ********** | ********** |
| DC8405 | CALL | C,nn | Call Subroutine at | |
| FC8405 | CALL | M,nn | Location nn if Condi- | |
| D48405 | CALL | NC,nn | tion True | |
| C48405 | CALL | NZ,nn | | |
| F48405 | CALL | P,nn | | |
| EC8405 | CALL | PE,nn | | |
| E48405 | CALL | PO,nn | | |
| CC8405 | CALL | Z,nn | | |
| ********** | ********** | ********** | ********** | ********** |
| CD8405 | CALL | nn | Unconditional Call to Subroutine at nn | |
| ********** | ********** | ********** | ********** | ********** |
| 3F | CCF | | Complement Carry Flag | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| BE | CP | (HL) | Compar Operand | Leading A Oper- |
| DDBE05 | CP | (IX+d) | with Acc. | and is Optional |
| FDBE05 | CP | (IY+d) | | |
| BF | CP | A | | If d is Omitted |
| B8 | CP | B | | 0 is Assumed |
| B9 | CP | C | | |
| BA | CP | D | | |
| BB | CP | E | | |
| BC | CP | H | | |
| BD | CP | L | | |
| FE20 | CP | n | | |
| EDA9 | CPD | | Compare Location (HL) and Acc. Decrement HL and BC | |
| EDB9 | CPDR | | Compare Location (HL) and Acc., Decrement HL and BC, Repeat until BC=0 | |
| EDA1 | CPI | | Compare Location (HL) and Acc., Increment HL and Decrement BC | |
| EDB1 | CPIR | | Compare Location (HL) and Acc., Increment HL, Decrement BC, Repeat until BC=0 | |
| 2F | CPL | | Complement Acc. (1's Complement) | |
| 27 | DAA | | Decimal Adjust Acc. | |
| 35 | DEC | (HL) | Decrement Operand | If d is Omitted |
| DD3505 | DEC | (IX+d) | | 0 is Assumed |
| FD3505 | DEC | (IY+d) | | |
| 3D | DEC | A | | |
| 05 | DEC | B | | |
| 0B | DEC | BC | | |
| 0D | DEC | C | | |
| 15 | DEC | D | | |
| 1B | DEC | DE | | |
| 1D | DEC | E | | |
| 25 | DEC | H | | |
| 2B | DEC | HL | | |
| DD2B | DEC | IX | | |
| FD2B | DEC | IY | | |
| 2D | DEC | L | | |
| 3B | DEC | SP | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| F3 | DI | | Disable Interrupts | |
| 102E | DJNZ | e | Decrement B and Jump Relative if B=0 | |
| FB | EI | | Enable Interrupts | |
| E3 | EX | (SP),HL | Exchange Location and (SP) | |
| DDE3 | EX | (SP),IX | | |
| FDE3 | EX | (SP),IY | | |
| 08 | EX | AF,AF' | Exchange the Contents of AF and AF' | |
| EB | EX | DE,HL | Exchange the Contents of DE and HL | |
| D9 | EXX | | Exchange the Contents of BC,DE,HL with Contents of BC',DE',HL' Respectively | |
| 76 | HALT | | HALT (wait for Interrupt or Reset) | |
| ED46 | IM | 0 | Set Interrupt Mode | |
| ED56 | IM | 1 | | |
| ED5E | IM | 2 | | |
| ED78 | IN | A,(C) | Load Reg. with Input from Device (C) | |
| ED40 | IN | B,(C) | | |
| ED48 | IN | C,(C) | | |
| ED50 | IN | D,(C) | | |
| ED58 | IN | E,(C) | | |
| ED60 | IN | H,(C) | | |
| ED68 | IN | L,(C) | | |
| 34 | INC | (HL) | Increment Operand | If d is Omitted 0 is Assumed |
| DD3405 | INC | (IX+d) | | |
| FD3405 | INC | (IY+d) | | |
| 3C | INC | A | | |
| 04 | INC | B | | |
| 03 | INC | BC | | |
| 0C | INC | C | | |
| 14 | INC | D | | |
| 13 | INC | DE | | |
| 1C | INC | E | | |
| 24 | INC | H | | |
| 23 | INC | HL | | |
| DD23 | INC | IX | | |
| FD23 | INC | IY | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| 2C | INC | L | Increment Operand | |
| 33 | INC | SP | | |
| DB20 | IN | A,(n) | Load Acc. with Input from Device n | |
| EDAA | IND | | Load Location (HL) with Input from Port (C), Decrement HL and B | |
| EDBA | INDR | | Load Location (HL) with Input from Port (C), Decrement HL and Decrement B, Repeat until B=0 | |
| EDA2 | INI | | Load Location (HL) with Input from Port (C); Increment HL and Decrement B | |
| EDB2 | INIR | | Load Location (HL) with Input from Port (C), Increment HL and Decrement B, Repeat until B=0 | |
| C38405 | JP | nn | Unconditional Jump | |
| E9 | JP | (HL) | to Location | |
| DDE9 | JP | (IX) | | |
| FDE9 | JP | (IY) | | |
| DA8405 | JP | C,nn | Jump to Location if | |
| FA8405 | JP | M,nn | Condtion True | |
| D28405 | JP | NC,nn | | |
| C28405 | JP | NZ,nn | | |
| F28405 | JP | P,nn | | |
| EA8405 | JP | PE,nn | | |
| E28405 | JP | PO,nn | | |
| CA8405 | JP | Z,nn | | |
| 382E | JR | C,e | Jump Relative to | |
| 302E | JR | NC,e | PC+e if Condition | |
| 202E | JR | NZ,e | True | |
| 282E | JR | Z,e | | |
| 182E | JR | e | Uncondtional Jump Relative to PC+e | |
| 02 | LD | (BC),A | Load Source to | |
| 12 | LD | (DE),A | Destination | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| 77 | LD | (HL),A | Load Source to | If d is Omitted |
| 70 | LD | (HL),B | Destination | 0 is Assumed |
| 71 | LD | (HL),C | | |
| 72 | LD | (HL),D | | |
| 73 | LD | (HL),E | | |
| 74 | LD | (HL),H | | |
| 75 | LD | (HL),L | | |
| 3620 | LD | (HL),n | | |
| DD7705 | LD | (IX+d),A | | |
| DD7005 | LD | (IX+d),B | | |
| DD7105 | LD | (IX+d),C | | |
| DD7205 | LD | (IX+d),D | | |
| DD7305 | LD | (IX+d),E | | |
| DD7405 | LD | (IX+d),H | | |
| DD7505 | LD | (IX+d),L | | |
| DD360520 | LD | (IX+d),n | | |
| FD7705 | LD | (IY+d),A | | |
| FD7005 | LD | (IY+d),B | | |
| FD7105 | LD | (IY+d),C | | |
| FD7205 | LD | (IY+d),D | | |
| FD7305 | LD | (IY+d),E | | |
| FD7405 | LD | (IY+d),H | | |
| FD7505 | LD | (IY+d),L | | |
| FD360520 | LD | (IY+d),n | | |
| 328405 | LD | (nn),A | | |
| ED438405 | LD | (nn),BC | | |
| ED538405 | LD | (nn),DE | | |
| 228405 | LD | (nn),HL | | |
| DD228405 | LD | (nn),IX | | |
| FD228405 | LD | (nn),IY | | |
| ED738405 | LD | (nn),SP | | |
| 0A | LD | A,(BC) | | |
| 1A | LD | A,(DE) | | |
| 7E | LD | A,(HL) | | |
| DD7E05 | LD | A,(IX+d) | | |
| FD7E05 | LD | A,(IY+d) | | |
| 3A8405 | LD | A,(nn) | | |
| 7F | LD | A,A | | |
| 78 | LD | A,B | | |
| 79 | LD | A,C | | |
| 7A | LD | A,D | | |
| 7B | LD | A,E | | |
| 7C | LD | A,H | | |
| ED57 | LD | A,I | | |
| 7D | LD | A,L | | |
| 3E20 | LD | A,n | | |
| ED5F | LD | A,R | | |
| 46 | LD | B,(HL) | | |
| DD4605 | LD | B,(IX+d) | | |
| FD4605 | LD | B,(IY+d) | | |
| 47 | LD | B,A | | |
| 40 | LD | B,B | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| 41 | LD | B,C | Load Source to | If d is Omitted |
| 42 | LD | B,D | Destination | 0 is Assumed |
| 43 | LD | B,E | | |
| 44 | LD | B,H | | |
| 45 | LD | B,L | | |
| 0620 | LD | B,n | | |
| ED4B8405 | LD | BC,(nn) | | |
| 018405 | LD | BC,nn | | |
| 4E | LD | C,(HL) | | |
| DD4E05 | LD | C,(IX+d) | | |
| FD4E05 | LD | C,(IY+d) | | |
| 4F | LD | C,A | | |
| 48 | LD | C,B | | |
| 49 | LD | C,C | | |
| 4A | LD | C,D | | |
| 4B | LD | C,E | | |
| 4C | LD | C,H | | |
| 4D | LD | C,L | | |
| 0E20 | LD | C,n | | |
| 56 | LD | D,(HL) | | |
| DD5605 | LD | D,(IX+d) | | |
| FD5605 | LD | D,(IY+d) | | |
| 57 | LD | D,A | | |
| 50 | LD | D,B | | |
| 51 | LD | D,C | | |
| 52 | LD | D,D | | |
| 53 | LD | D,E | | |
| 54 | LD | D,H | | |
| 55 | LD | D,L | | |
| 1620 | LD | D,n | | |
| ED5B8405 | LD | DE,(nn) | | |
| 118405 | LD | DE,nn | | |
| 5E | LD | E,(HL) | | |
| DD5E05 | LD | E,(IX+d) | | |
| FD5E05 | LD | E,(IY+d) | | |
| 5F | LD | E,A | | |
| 58 | LD | E,B | | |
| 59 | LD | E,C | | |
| 5A | LD | E,D | | |
| 5B | LD | E,E | | |
| 5C | LD | E,H | | |
| 5D | LD | E,L | | |
| 1E20 | LD | E,n | | |
| 66 | LD | H,(HL) | | |
| DD6605 | LD | H,(IX+d) | | |
| FD6605 | LD | H,(IY+d) | | |
| 67 | LD | H,A | | |
| 60 | LD | H,B | | |
| 61 | LD | H,C | | |
| 62 | LD | H,D | | |
| 63 | LD | H,E | | |
| 64 | LD | H,H | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| 65 | LD | H,L | Load Source to | If d is Omitted |
| 2620 | LD | H,n | Destination | 0 is Assumed |
| 2A8405 | LD | HL,(nn) | | |
| 218405 | LD | HL,nn | | |
| ED47 | LD | I,A | | |
| DD2A8405 | LD | IX,(nn) | | |
| DD218405 | LD | IX,nn | | |
| FD2A8405 | LD | IY,(nn) | | |
| FD218405 | LD | IY,nn | | |
| 6E | LD | L,(HL) | | |
| DD6E05 | LD | L,(IX+d) | | |
| FD6E05 | LD | L,(IY+d) | | |
| 6F | LD | L,A | | |
| 68 | LD | L,B | | |
| 69 | LD | L,C | | |
| 6A | LD | L,D | | |
| 6B | LD | L,E | | |
| 6C | LD | L,H | | |
| 6D | LD | L,L | | |
| 2E20 | LD | L,n | | |
| ED4F | LD | R,A | | |
| ED7B8405 | LD | SP,(nn) | | |
| F9 | LD | SP,HL | | |
| DDF9 | LD | SP,IX | | |
| FDF9 | LD | SP,IY | | |
| 318405 | LD | SP,nn | | |
| EDA8 | LDD | | Load Location(DE) with Location(HL), Decrement DE, HL and BC | |
| EDB8 | LDDR | | Load Location (DE) with Location (HL). Repeat until BC=0 | |
| EDA0 | LDI | | Load Location (DE) with Location (HL), Increment DE, HL, Decrement BC | |
| EDB0 | LDIR | | Load Location (DE) with Location (HL), Increment DE, HL, Decrement BC and Repeat until BC=0 | |
| ED44 | NEG | | Negate Acc. (2's Complement) | |
| 00 | NOP | | No Operation | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| B6 | OR | A,(HL) | Logical "OR" of | Leading A Oper- |
| DDB605 | OR | A,(IX+d) | Operand and Acc. | and is Optional |
| FDB605 | OR | A,(IY+d) | | |
| B7 | OR | A,A | | If d is Omitted |
| B0 | OR | A,B | | 0 is Assumed |
| B1 | OR | A,C | | |
| B2 | OR | A,D | | |
| B3 | OR | A,E | | |
| B4 | OR | A,H | | |
| B5 | OR | A,L | | |
| F620 | OR | A,n | | |
| ********** | | | | |
| ED8B | OTDR | | Load Output Port (C) with Location (HL), Decrement HL and B, Repeat until B=0 | |
| ********** | | | | |
| EDB3 | OTIR | | Load Output Port (C) with Location (HL), Increment HL, Decrement B, Repeat until B=0 | |
| ********** | | | | |
| ED79 | OUT | (C),A | Load Output Port (C) with Reg. | |
| ED41 | OUT | (C),B | | |
| ED49 | OUT | (C),C | | |
| ED51 | OUT | (C),D | | |
| ED59 | OUT | (C),E | | |
| ED61 | OUT | (C),H | | |
| ED69 | OUT | (C),L | | |
| ********** | | | | |
| D320 | OUT | (n),A | Load Output Port (n) with Acc. | |
| ********** | | | | |
| EDAB | OUTD | | Load Output Port (C) with Location (HL), Decrement HL and B | |
| ********** | | | | |
| EDA3 | OUTI | | Load Output Port (C) with Location (HL), Increment HL and Decrement B | |
| ********** | | | | |
| F1 | POP | AF | Load Destination with Top of Stack | |
| C1 | POP | BC | | |
| D5 | POP | DE | | |
| E1 | POP | HL | | |
| DDE1 | POP | IX | | |
| FDE1 | POP | IY | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| F5 | PUSH | AF | Load Source to Stack | |
| C5 | PUSH | BC | | |
| D5 | PUSH | DE | | |
| E5 | PUSH | HL | | |
| DDE5 | PUSH | IX | | |
| FDE5 | PUSH | IY | | |
| ********** | | | | |
| CB86 | RES | 0,(HL) | Reset Bit b of | If d is Omitted |
| DDCB0586 | RES | 0,(IX+d) | Operand | 0 is Assumed |
| FDCB0586 | RES | 0,(IY+d) | | |
| CB87 | RES | 0,A | | |
| CB80 | RES | 0,B | | |
| CB81 | RES | 0,C | | |
| CB82 | RES | 0,D | | |
| CB83 | RES | 0,E | | |
| CB84 | RES | 0,H | | |
| CB85 | RES | 0,L | | |
| CB8E | RES | 1,(HL) | | |
| DDCB058E | RES | 1,(IX+d) | | |
| FDCB058E | RES | 1,(IY+d) | | |
| CB8F | RES | 1,A | | |
| CB88 | RES | 1,B | | |
| CB89 | RES | 1,C | | |
| CB8A | RES | 1,D | | |
| CB8B | RES | 1,E | | |
| CB8C | RES | 1,H | | |
| CB8D | RES | 1,L | | |
| CB96 | RES | 2,(HL) | | |
| DDCB0596 | RES | 2,(IX+d) | | |
| FDCB0596 | RES | 2,(IY+d) | | |
| CB97 | RES | 2,A | | |
| CB90 | RES | 2,B | | |
| CB91 | RES | 2,C | | |
| CB92 | RES | 2,D | | |
| CB93 | RES | 2,E | | |
| CB94 | RES | 2,H | | |
| CB95 | RES | 2,L | | |
| CB9E | RES | 3,(HL) | | |
| DDCB059E | RES | 3,(IX+d) | | |
| FDCB059E | RES | 3,(IY+d | | |
| CB9F | RES | 3,A | | |
| CB98 | RES | 3,B | | |
| CB9A | RES | 3,D | | |
| CB9B | RES | 3,E | | |
| CB9C | RES | 3,H | | |
| CB9D | RES | 3,L | | |
| CBA6 | RES | 4,(HL) | | |
| DDCB05A6 | RES | 4,(IX+d) | | |
| FDCB05A6 | RES | 4,(IY+d) | | |
| CBA7 | RES | 4,A | | |
| CBA0 | RES | 4,B | | |
| CBA1 | RES | 4,C | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| CBA2 | RES | 4,D | Reset Bit b of | If d is Omitted |
| CBA3 | RES | 4,E | Operation | 0 is Assumed |
| CBA4 | RES | 4,H | | |
| CBA5 | RES | 4,L | | |
| CBAE | RES | 5,(HL) | | |
| DDCB05AE | RES | 5,(IX+d) | | |
| FDCB05AE | RES | 5,(IY+d) | | |
| CBAF | RES | 5,A | | |
| CBA8 | RES | 5,B | | |
| CBA9 | RES | 5,C | | |
| CBAA | RES | 5,D | | |
| CBAB | RES | 5,E | | |
| CBAC | RES | 5,L | | |
| CBB6 | RES | 6,(HL) | | |
| DDCB05B6 | RES | 6,(IX+d) | | |
| FDCB05B6 | RES | 6,(IY+d) | | |
| CBB7 | RES | 6,A | | |
| CBB0 | RES | 6,B | | |
| CBB1 | RES | 6,C | | |
| CBB2 | RES | 6,D | | |
| CBB3 | RES | 6,E | | |
| CBB4 | RES | 6,H | | |
| CBB5 | RES | 6,L | | |
| CBBE | RES | 7,(HL) | | |
| DDCB05BE | RES | 7,(IX+d) | | |
| FDCB05BE | RES | 7,(IY+d) | | |
| CBBF | RES | 7,A | | |
| CBB8 | RES | 7,B | | |
| CBB9 | RES | 7,C | | |
| CBBA | RES | 7,D | | |
| CBBB | RES | 7,E | | |
| CBBC | RES | 7,H | | |
| CBBD | RES | 7,L | | |
| C9 | RET | | Return from Subroutine | |
| D8 | RET | C | Return from Subroutine if Condition True | |
| F8 | RET | M | | |
| D0 | RET | NC | | |
| C0 | RET | NZ | | |
| F0 | RET | P | | |
| E8 | RET | PE | | |
| E0 | RET | P0 | | |
| C8 | RET | Z | | |
| ED4D | RETI | | Return from Interrupt | |
| ED45 | RETN | | Return from Non-Maskable Interrupt | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| CB16 | RL | (HL) | Rotate Left Through | If d is Omitted |
| DDCB0516 | RL | (IX+d) | Carry | 0 is Assumed |
| FDCB0516 | RL | (IY+d) | | |
| CB17 | RL | A | | |
| CB10 | RL | B | | |
| CB11 | RL | C | | |
| CB12 | RL | D | | |
| CB13 | RL | E | | |
| CB14 | RL | H | | |
| CB15 | RL | L | | |
| 17 | RLA | | Rotate Left Acc. Through Carry | |
| CB06 | RLC | (HL) | Rotate Left Circular | If d is Omitted |
| DDCB0506 | RLC | (IX+d) | | 0 i Assumed |
| FDCB0506 | RLC | (IY+d) | | |
| CB07 | RLC | A | | |
| CB00 | RLC | B | | |
| CB01 | RLC | C | | |
| CB02 | RLC | D | | |
| CB03 | RLC | E | | |
| CB04 | RLC | H | | |
| CB05 | RLC | L | | |
| 07 | RLCA | | Rotate Left Circ. Acc. | |
| ED6F | RLD | | Rotate Digit Left and Right between Acc. and Location (HL) | |
| CB1E | RR | (HL) | Rotate Right Through | If d is Omitted |
| DDCB051E | RR | (IX+d) | Carry | 0 is Assumed |
| FDCB051E | RR | (IY+d) | | |
| CB1F | RR | A | | |
| CB18 | RR | B | | |
| CB19 | RR | C | | |
| CB1A | RR | D | | |
| CB1B | RR | E | | |
| CB1C | RR | H | | |
| CB1D | RR | L | | |
| 1F | RRA | | Rotate Right Acc. Through Carry | |
| CB0E | RRC | (HL) | Rotate Right Circular | |
| DDCB050E | RRC | (IX+d) | | |
| FDCB050E | RRC | (IY+d) | | |
| CB0F | RRC | A | | |
| CB08 | RRC | B | | |
| CB09 | RRC | C | | |
| CB0A | RRC | D | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| CB0B | RRC | E | Rotate Right Circular | |
| CB0C | RRC | H | | |
| CB0D | RRC | L | | |
| 0F | RRCA | | Rotate Right Circular Acc. | |
| ED67 | RRD | | Rotate Digit Right and Left Between Acc. and Location (HL) | |
| C7 | RST | 00H | Restart to Location | |
| CF | RST | 08H | | |
| D7 | RST | 10H | | |
| DF | RST | 18H | | |
| E7 | RST | 20H | | |
| EF | RST | 28H | | |
| F7 | RST | 30H | | |
| FF | RST | 38H | | |
| DE20 | SBC | A,n | Subtract Operand | Leading A Operand is Optional |
| 9E | SBC | A,(HL) | from Acc. with Carry | |
| DD9E05 | SBC | A,(IX+d) | | |
| FD9E05 | SBC | A,(IY+d) | | If d is Omitted |
| 9F | SBC | A,A | | 0 is Assumed |
| 98 | SBC | A,B | | |
| 99 | SBC | A,C | | |
| 9A | SBC | A,D | | |
| 9B | SBC | A,E | | |
| 9C | SBC | A,H | | |
| 9D | SBC | A,L | | |
| ED42 | SBC | HL,BC | | |
| ED52 | SBC | HL,DE | | |
| ED62 | SBC | HL,HL | | |
| ED72 | SBC | HL,SP | | |
| 37 | SCF | | Set Carry Flag (C=1) | |
| CBC6 | SET | 0,(HL) | Set Bit b of Location | If d is Omitted |
| DDCB05C6 | SET | 0,(IX+d) | | 0 is Assumed |
| FDCB05C6 | SET | 0,(IY+d) | | |
| CBC7 | SET | 0,A | | |
| CBC0 | SET | 0,B | | |
| CBC1 | SET | 0,C | | |
| CBC2 | SET | 0,D | | |
| CBC3 | SET | 0,E | | |
| CBC4 | SET | 0,H | | |
| CBC5 | SET | 0,L | | |
| CBCE | SET | 1,(HL) | | |
| DDCB05CE | SET | 1,(IX+d) | | |
| FDCB05CE | SET | 1,(IY+d) | | |
| CBCF | SET | 1,A | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| CBC8 | SET | 1,B | Set Bit b of Location | If d is Omitted |
| CBC9 | SET | 1,C | | 0 is Assumed |
| CBCA | SET | 1,D | | |
| CBCB | SET | 1,E | | |
| CBCC | SET | 1,H | | |
| CBCD | SET | 1,L | | |
| CBD6 | SET | 2,(HL) | | |
| DDCB05D6 | SET | 2,(IX+d) | | |
| FDCB05D6 | SET | 2,(IY+d) | | |
| CBD7 | SET | 2,A | | |
| CBD0 | SET | 2,B | | |
| CBD1 | SET | 2,C | | |
| CBD2 | SET | 2,D | | |
| CBD3 | SET | 2,E | | |
| CBD4 | SET | 2,H | | |
| CBD5 | SET | 2,L | | |
| CBD8 | SET | 3,B | | |
| CBDE | SET | 3,(HL) | | |
| DDCB05DE | SET | 3,(IX+d) | | |
| FDCB05DE | SET | 3,(IY+d) | | |
| CBDF | SET | 3,A | | |
| CBD8 | SET | 3,B | | |
| CBD9 | SET | 3,C | | |
| CBDA | SET | 3,D | | |
| CBDB | SET | 3,E | | |
| CBDC | SET | 3,H | | |
| CBDD | SET | 3,L | | |
| CBE6 | SET | 4,(HL) | | |
| DDCB05E6 | SET | 4,(IX+d) | | |
| FDCB05E6 | SET | 4,(IY+d) | | |
| CBE7 | SET | 4,A | | |
| CBE0 | SET | 4,B | | |
| CBE1 | SET | 4,C | | |
| CBE2 | SET | 4,D | | |
| CBE3 | SET | 4,E | | |
| CBE4 | SET | 4,H | | |
| CBE5 | SET | 4,L | | |
| CBEE | SET | 5,(HL) | | |
| DDCB05EE | SET | 5,(IX+d) | | |
| FDCB05EE | SET | 5,(IY+d) | | |
| CBEF | SET | 5,A | | |
| CBE8 | SET | 5,B | | |
| CBE9 | SET | 5,C | | |
| CBEA | SET | 5,D | | |
| CBEB | SET | 5,E | | |
| CBEC | SET | 5,H | | |
| CBED | SET | 5,L | | |
| CBF6 | SET | 6,(HL) | | |
| DDCB05F6 | SET | 6,(IX+d) | | |
| FDCB05F6 | SET | 6,(IY+d) | | |
| CBF7 | SET | 6,A | | |
| CBF0 | SET | 6,B | | |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| CBF1 | SET | 6,C | Set Bit b of | If d is Omitted |
| CBF2 | SET | 6,D | Location | 0 is Assumed |
| CBF3 | SET | 6,E | | |
| CBF4 | SET | 6,H | | |
| CBF5 | SET | 6,L | | |
| DBFE | SET | 7,(HL) | | |
| DDCB05FE | SET | 7,(IX+d) | | |
| FDCB05FE | SET | 7,(IY+d) | | |
| CBFF | SET | 7,A | | |
| CBF8 | SET | 7,B | | |
| CBF9 | SET | 7,C | | |
| CBFA | SET | 7,D | | |
| CBFB | SET | 7,E | | |
| CBFC | SET | 7,H | | |
| CBFD | SET | 7,L | | |
| ********** | | | | |
| CB26 | SLA | (HL) | Shift Operand Left | If d is Omitted |
| DDCB0526 | SLA | (IX+d) | Arithmetic | 0 is Assumed |
| FDCB0526 | SLA | (IY+d) | | |
| CB27 | SLA | A | | |
| CB20 | SLA | B | | |
| CB21 | SLA | C | | |
| CB22 | SLA | D | | |
| CB23 | SLA | E | | |
| CB24 | SLA | H | | |
| CB25 | SLA | L | | |
| ********** | | | | |
| CB2E | SRA | (HL) | Shift Operand Right | If d is Omitted |
| DDCB052E | SRA | (IX+d) | Arithmetic | 0 is Assumed |
| FDCB052E | SRA | (IY+d) | | |
| CB2F | SRA | A | | |
| CB28 | SRA | B | | |
| CB29 | SRA | C | | |
| CB2A | SRA | D | | |
| CB2B | SRA | E | | |
| CB2C | SRA | H | | |
| CB2D | SRA | L | | |
| ********** | | | | |
| CB3E | SRL | (HL) | Shift Operand Right | If d is Omitted |
| DDCB053E | SRL | (IX+d) | Logical | 0 is Assumed |
| FDCB053E | SRL | (IY+d) | | |
| DB3F | SRL | A | | |
| DB38 | SRL | B | | |
| CB39 | SRL | C | | |
| CB3A | SRL | D | | |
| CB3B | SRL | E | | |
| CB3C | SRL | H | | |
| CB3D | SRL | L | | |
| ********** | | | | |
| 96 | SUB | (HL) | Subtract Operand | Leading A Oper- |
| DD9605 | SUB | (IX+d) | from Acc. | and is Optional |

| Object Code | Source Statement | | Operation | Notes |
|---|---|---|---|---|
| FD9605 | SUB | (IY+d) | Subtract Operand | If d is Omitted |
| 97 | SUB | A | from Acc. | 0 is Assumed |
| 90 | SUB | B | | |
| 91 | SUB | C | | |
| 92 | SUB | D | | |
| 93 | SUB | E | | |
| 94 | SUB | H | | |
| 95 | SUB | L | | |
| D620 | SUB | n | | |
| ********** | ********** | ********** | ********** | ********** |
| AE | XOR | A,(HL) | Exclusive "OR" | Leading A Oper- |
| DDAE05 | XOR | A,(IX+d) | Operand and Acc. | and is Optional |
| FDAE05 | XOR | A,(IY+d) | | |
| AF | XOR | A,A | | If d is Omitted |
| A8 | XOR | A,B | | 0 is Assumed |
| A9 | XOR | A,C | | |
| AA | XOR | A,D | | |
| AB | XOR | A,E | | |
| AC | XOR | A,H | | |
| AD | XOR | A,L | | |
| EE20 | XOR | A,n | | |

**(THIS PAGE INTENTIONALLY LEFT BLANK.)**

APPENDIX B

**ECHELON SOFTWARE UPDATE FORM**

1. **PRODUCT NAME & VERSION** _____

2. **USER NAME** _____ **DATE** _____

3. **USER'S HARDWARE & SOFTWARE SYSTEM:** _____

_____

4. **REPORT TYPE:**

___ Problem/Possible Error

___ Suggested Enhancement

___ Document Suggestion

___ Other _____

_____

5. **PERFORMANCE IMPACT:**

___ Shuts Down System

___ Impairs System Performance

___ Causes Inconvenience

___ Needs Suggested Enhancement

___ Other _____

6. **PROBLEM DESCRIPTION:** Please describe the problem concisely and how it can be reproduced. If possible, provide your diagnosis and your cure. Attach a listing if available.

7. **RETURN FORM TO:** Echelon, Inc.
101 First Street
Los Altos, CA 94022

**YOUR INTEREST IN Z-TOOLS IS APPRECIATED!**

(THIS PAGE INTENTIONALLY LEFT BLANK.)

APPENDIX C

## ZAS PSEUDO-OP SUMMARY

| | Pseudo-op | Form | Definition |
|---|---|---|---|
| | ASEG | | set absolute segment |
| | COMMON | | set common segment |
| | CSEG | | set code segment |
| | DB(DEFB) | n {,n...} | define byte |
| | DC | 'string' | define character |
| | .DEPHASE | | end .phase |
| | DS(DEFS) | expression {,expression} | define space |
| | DSEG | | set data segment |
| | DW(DEFW) | nn {,nn...} | define word |
| | ELSE | | conditional assembly |
| | END | {expression} | specifies program starting address |
| | ENDIF | | end conditional assembly |
| | ENDM | | end macro |
| LABEL | EQU | expression | equate label to a value |
| | EXITM | | exit macro |
| | EXTRN(EXT) | symbol {,symbol...) | define external symbols |
| | .HD64 | | assemble HD64180 instructions |
| | IF | expression | conditional assembly |
| | .IF1 | | conditional trueness based on lsb |
| | .IF16 | | conditional trueness based on 16-bits |
| | .IN(MACLIB) | {d:}filename | include file |
| | IRP | identifier, string {,string...} | indefinite repeat macro |
| | IRPC | identifier, string | indefinite repeat character macro |
| | .LALL | | list all macro lines |
| | .LFCOND | | list all false conditionals |
| | .LIST | | resume listing |
| | LOCAL | identifier {,identifier...} | define local macro labels |
| LABEL | MACRO | dummy {,dummy...} | stored macro definition |
| | NAME | modulename | define module name |

## ZAS PSEUDO-OP SUMMARY (con't)

| | Pseudo-op | Form | Definition |
|---|---|---|---|
| | ORG | expression | change value of relocation counter |
| | PAGE | {expression} | page definition or eject |
| | .PHASE | expression | relocate block of code |
| | .PRINT | pass,text | print text during assembly |
| | PUBLIC | symbol {,symbol...} | define public symbols |
| | .RADIX | n | set radix default |
| | REPT | expression | repeat macro |
| | .REQUEST | filename {,filename...} | request library search |
| | .SALL | | suppress macro listing |
| | .SBTTL | 'string' | define subtitle |
| LABEL | SET(DEFL) | expression | set label to a value |
| | .SFCOND | | suppress listing of false conditionals |
| | .TITLE | 'string' | define title |
| | .XALL | | exclude non-code macro lines |
| | .XLIST | | suppress listings |

**Legend:** items in ( )'s are aliases; in { }'s, optional.

APPENDIX D

## HITACHI HD64180 MODE

| Object Code | Source Statement | Operation |
|---|---|---|
| ED3805 | IN0 A,(nn) | Load register with input from |
| ED0005 | IN0 B,(nn) | port (nn). |
| ED0805 | IN0 C,(nn) | |
| ED1005 | IN0 D,(nn) | |
| ED1805 | IN0 E,(nn) | |
| ED2005 | IN0 H,(nn) | |
| ED2805 | IN0 L,(nn) | |
| ED4C | MLT BC | Unsigned multiplication of |
| ED5C | MLT DE | each half of the specified |
| ED6C | MLT HL | register pair with the 16-bit |
| ED7C | MLT SP | result going to the specified register pair. |
| ED8B | OTDM | Load output port (C) with location (HL), decrement HL, B, and C. |
| ED9B | OTDMR | Load output port (C) with location (HL), decrement HL, B, and C. Repeat until B=0. |
| ED83 | OTIM | Load output port (C) with location (HL), increment HL and C. Decrement B. |
| ED93 | OTIMR | Load output port (C) with location (HL), increment HL and C. Decrement B. Repeat until B=0. |
| ED3905 | OUT0 (nn),A | Load output port (nn) from |
| ED0105 | OUT0 (nn),B | register. |
| ED0905 | OUT0 (nn),C | |
| ED1105 | OUT0 (nn),D | |
| ED1905 | OUT0 (nn),E | |
| ED2105 | OUT0 (nn),H | |
| ED2905 | OUT0 (nn),L | |
| ED76 | SLP | Enter sleep mode. |
| ED3C | TST A | Non-destructive AND with |
| ED04 | TST B | accumulator and specified |
| ED0C | TST C | operand. |
| ED14 | TST D | |
| ED1C | TST E | |
| ED24 | TST H | |
| ED2C | TST L | |
| ED6405 | TST nn | |
| ED34 | TST (HL) | |
| ED7405 | TSTIO nn | Non-destructive AND of nn and the contents of port (C). |