



NCR 5380 SCSI INTERFACE

PRODUCT BRIEF

SCSI INTERFACE

- Asynchronous data transfer to 1.5 MBPS
- Supports both initiator and target roles
- Parity generation w/optional checking
- Supports arbitration
- Direct control of all bus signals
- High current outputs drive SCSI bus directly

MPU INTERFACE

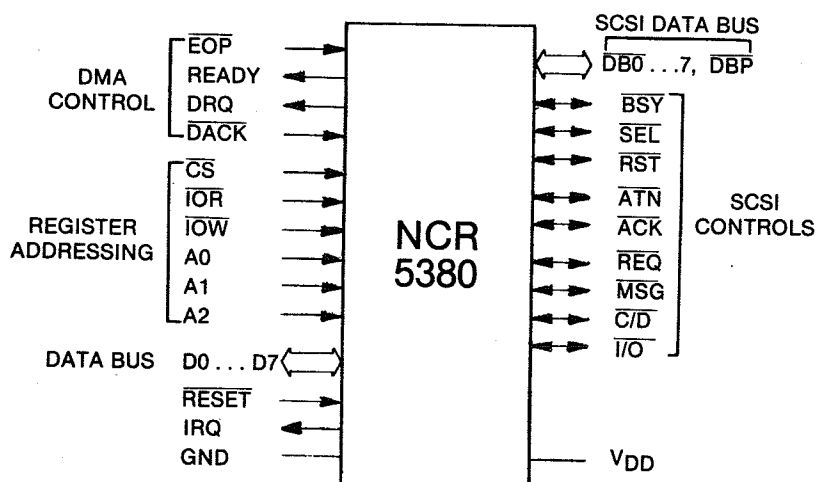
- Memory or I/O mapped interface
- DMA or programmed I/O
- Normal or block mode DMA
- Optional MPU interrupts

The NCR 5380 is designed to accommodate the Small Computer Systems Interface (SCSI) as defined by the ANSI X3T9.2 committee. The 5380 operates in both the Initiator and Target roles and can therefore be used in host adapter and control unit designs. This device supports arbitration, including reselection, and is intended to be used in systems that require either open collector or differential pair transceivers.* It has special high current outputs for driving the SCSI bus directly in the open collector mode.

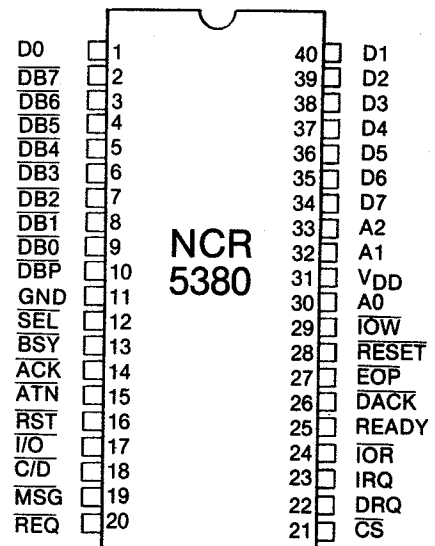
The NCR 5380 communicates with the system microprocessor as a peripheral device. The chip is controlled by reading and writing several internal registers which may be addressed as standard or memory mapped I/O. Minimal processor intervention is required for DMA transfers because the 5380 controls the necessary handshake signals. The NCR 5380 interrupts the MPU when it detects a bus condition that requires attention. Normal and block mode DMA is provided to match many popular DMA controllers.

* Differential pair operation is supported in the NCR 5381 (48 PIN).

FUNCTIONAL PIN GROUPING



PINOUT





PIN DESCRIPTIONS

MICROPROCESSOR INTERFACE SIGNALS

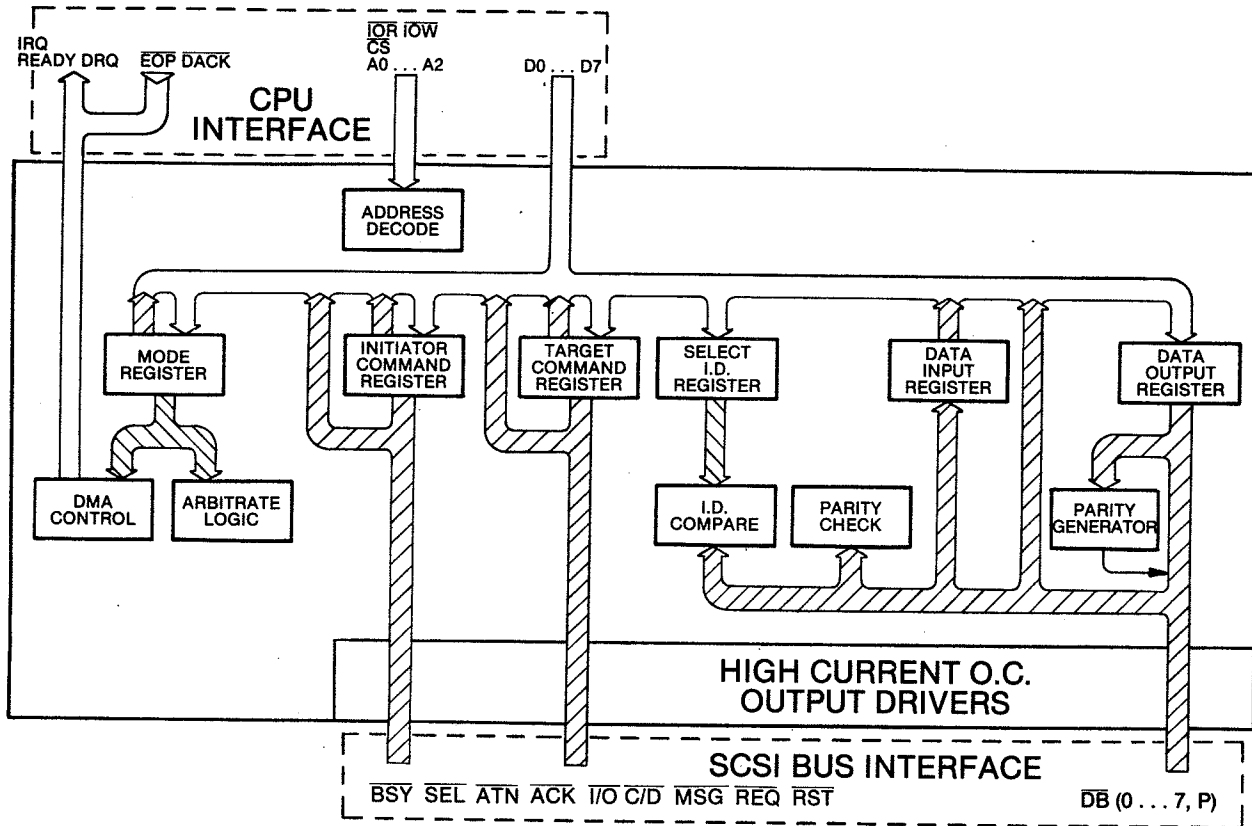
Pin Name	Pin Number	Description
A0...A2	30, 32, 33	INPUTS This address is used with \overline{CS} , \overline{IOR} or \overline{IOW} to address all internal registers.
\overline{CS}	21	INPUT Chip Select enables a read or write of the internal register selected by A0...A2. \overline{CS} is a low active signal.
\overline{DACK}	26	INPUT DMA Acknowledge resets DRQ and selects the data register for input or output. \overline{DACK} is a low active signal.
DRQ	22	OUTPUT DMA Request indicates that the data register is ready to be read or written. DRQ occurs only if DMA MODE is true in the command register. It is cleared by \overline{DACK} .
D0...D7	34...40, 1	BI-DIRECTIONAL, TRI-STATE Microprocessor data bus Active high
\overline{EOP}	27	INPUT The End of Process signal is true during the last byte of a DMA transfer. This stops additional transfers but allows the current transfer to finish. \overline{EOP} is a low active signal.
\overline{IOR}	24	INPUT \overline{IOR} Read is used to read an internal register selected by \overline{CS} and A0...A2. It also selects the data register when used with \overline{DACK} . \overline{IOR} is a low active signal.
\overline{IOW}	29	INPUT \overline{IOW} Write is used to write an internal register selected by \overline{CS} and A0...A2. It also selects the data register when used with \overline{DACK} . \overline{IOW} is a low active signal.
IRQ	23	OUTPUT Interrupt Request alerts the microprocessor of an error condition or an event completion.
READY	25	OUTPUT Ready can be used to control the speed of block mode DMA transfers.
\overline{RESET}	28	INPUT Reset clears all registers. It does not force the SCSI signal \overline{RST} to the active state. \overline{RESET} is a low active signal.

POWER SIGNALS

Pin Name	Pin Number	Description
V _{DD}	31	+5 VOLTS
GND	11	GROUND

SCSI INTERFACE SIGNALS

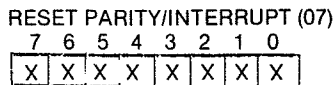
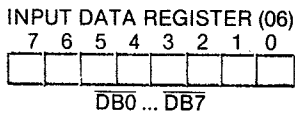
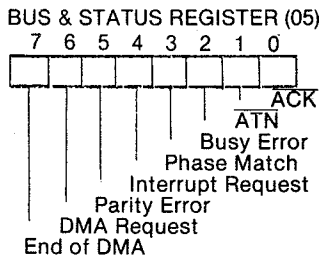
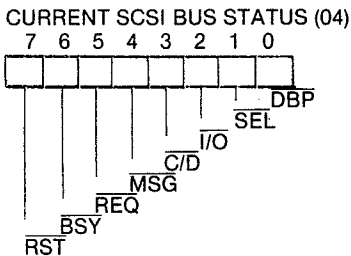
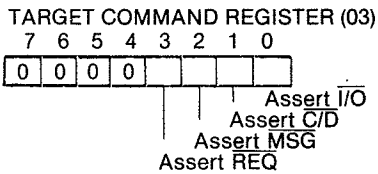
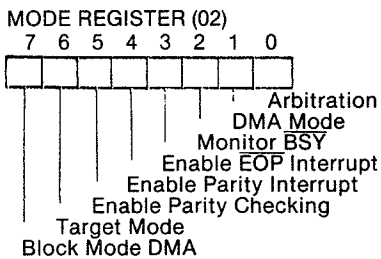
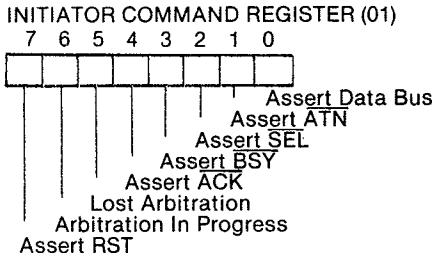
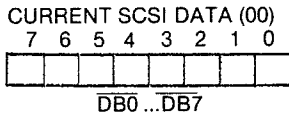
Pin Name	Pin Number	Description
\overline{ACK}	14	BI-DIRECTIONAL, OPEN COL. INITIATOR ROLE: The chip asserts this signal in response to \overline{REQ} for a byte transfer on the SCSI bus. TARGET ROLE: \overline{ACK} is received as a response to the \overline{REQ} signal. \overline{ACK} is an active low signal.
\overline{ATN}	15	BI-DIRECTIONAL, OPEN COL. INITIATOR ROLE: The chip asserts this signal when the microprocessor requests the attention condition. TARGET ROLE: \overline{ATN} is a received signal. \overline{ATN} is an active low signal.
\overline{BSY}	13	BI-DIRECTIONAL, OPEN COL. The SCSI \overline{BSY} signal can be driven and received concurrently. \overline{BSY} is an active low signal.
$\overline{C/D}$	18	BI-DIRECTIONAL, OPEN COL. Command/Data is an input for an initiator, an output for a target. It indicates a command when asserted. $\overline{C/D}$ is an active low signal.
$\overline{I/O}$	17	BI-DIRECTIONAL, OPEN COL. Input/Output is an input for an initiator, an output for a target. It indicates an input to the initiator when asserted. $\overline{I/O}$ is an active low signal.
\overline{MSG}	19	BI-DIRECTIONAL, OPEN COL. Message is an input for an initiator, an output for a target. It indicates a message when asserted. \overline{MSG} is an active low signal.
\overline{REQ}	20	BI-DIRECTIONAL, OPEN COL. The target asserts \overline{REQ} to request a byte transfer from the initiator. The transfer may be in either direction. \overline{REQ} is an active low signal.
\overline{RST}	16	BI-DIRECTIONAL, OPEN COL. SCSI BUS reset signal \overline{RST} is an active low signal.
$\overline{SB0...SB7}, \overline{SBP}$	2...10	BI-DIRECTIONAL, OPEN COL. SCSI DATA BUS and PARITY These signals are low active
\overline{SEL}	12	BI-DIRECTIONAL, OPEN COL. Select is used for selection and reselect operations. \overline{SEL} is an active low signal.



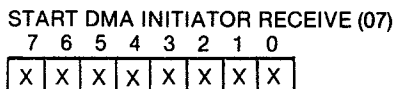
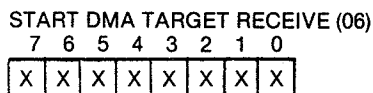
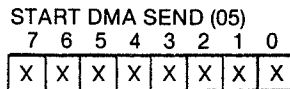
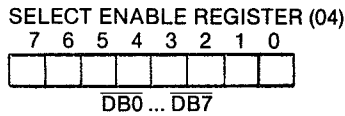
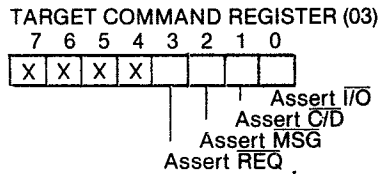
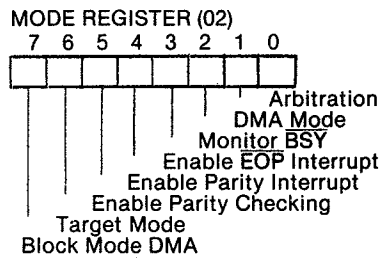
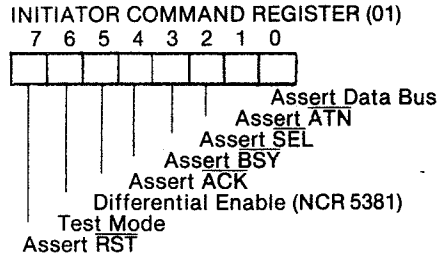
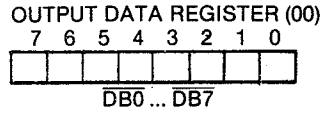
REGISTER SUMMARY

A2	A1	A0	R/W	REGISTER NAME
0	0	0	R	Current SCSI Data
0	0	0	W	Output Data Register
0	0	1	R/W	Initiator Command Reg.
0	1	0	R/W	Mode Register
0	1	1	R/W	Target Command Reg.
1	0	0	R	SCSI Bus Status
1	0	0	W	Select Enable Register
1	0	1	R	Bus & Status Register
1	0	1	W	Start DMA Send
1	1	0	R	Input Data Reg.
1	1	0	W	Start Target Rec. DMA
1	1	1	R	Reset Parity/Interrupts
1	1	1	W	Start Init. Rec. DMA

READ



WRITE



NOTE: X = DON'T CARE

ELECTRICAL CHARACTERISTICS OPERATING CONDITIONS

PARAMETER	SYMBOL	MIN	MAX	UNITS
Supply Voltage	V _{DD}	4.75	5.25	Volts
Supply Current	I _{DD}		145	mA.
Ambient Temperature	T _A	0	70	°C

INPUT SIGNAL REQUIREMENTS

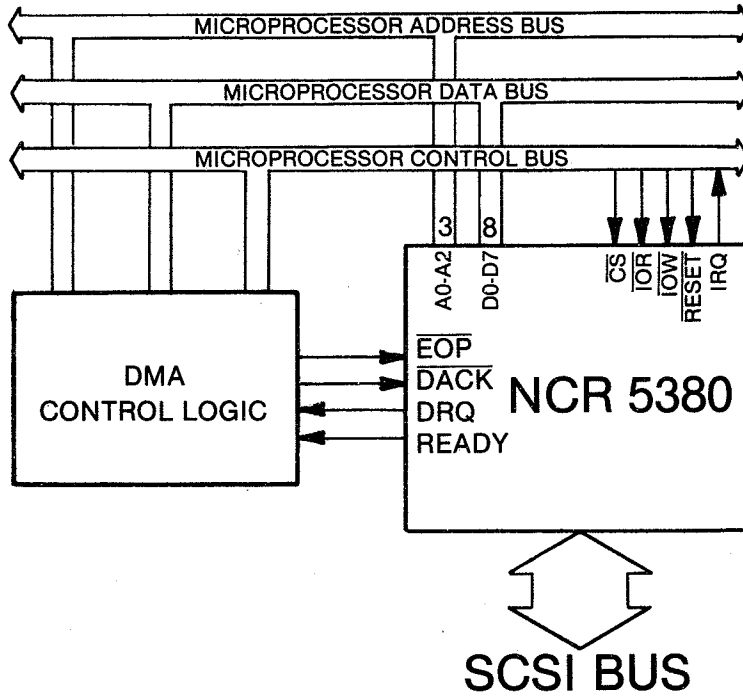
PARAMETER	CONDITIONS	MIN	MAX	UNITS
High-level, Input V _{IH}		2.0	5.25	Volts
Low-level, Input V _{IL}		-0.3	0.8	Volts
SCSI BUS pins 2 . . . 20				
High-level Input Current, I _{IH}	V _{IH} = 5.25 V		50	μa.
Low-level Input Current, I _{IL}	V _{IL} = 0 Volts		-50	μa.
All other pins				
High-level Input Current, I _{IH}	V _{IH} = 5.25 V		10	μa.
Low-level Input Current, I _{IL}	V _{IL} = 0 Volts		-10	μa.

OUTPUT SIGNAL REQUIREMENTS

PARAMETER	CONDITIONS	MIN	MAX	UNITS
SCSI BUS pins 2 . . . 20				
Low-level Output V _{OL}	V _{DD} = 4.75 V I _{OL} = 48.0mA.		0.5	Volts
All other pins				
High-level Output V _{OH}	V _{DD} = 4.75 V I _{OH} = -3.0mA.	2.4		Volts
Low-level Output V _{OL}	V _{DD} = 4.75 V I _{OL} = 7.0 mA.		0.5	Volts

PRELIMINARY

Notice: This is not a final specification.
Some parametric limits are subject to change.



NCR MICROELECTRONICS DIVISION
1635 Aeroplaza Drive
Colorado Springs, Colorado 80916
Phone: 800/525-2252
Telex: 45 2457 NCR MICRO CSP

While the information herein presented has been checked for both accuracy and reliability, NCR assumes no responsibility for either its use or for the infringement of any patents or other rights of third parties, which would result from its use. The publication and dissemination of the enclosed information confers no license, by implication or otherwise, under any patent or patent rights owned by NCR.



**Z80
HARD DISK
SOFTWARE**

USER'S MANUAL

P/N: A74015-A



PREFACE

This manual is for users and integrators of systems based on AMPRO Z80-based single board computers, utilizing the CP/M operating system. It describes the optional hard disk software contained on the AMPRO Z80 Hard Disk Software diskette. Here is an overview of what is included in this manual:

Chapter 1 - GENERAL INFORMATION - Features of the Z80 hard disk software. Conventions used in program operation descriptions.

Chapter 2 - SOFTWARE INSTALLATION - Hardware setup and software configuration procedures.

Chapter 3 - PROGRAM DESCRIPTIONS - Detailed descriptions and operating instructions for each of the hard disk software utilities.

Chapter 3 is intended to provide a convenient user reference. The program descriptions are arranged alphabetically, and each program's name appears on the bottom of the page, to aid you in locating the program description.

Each AMPRO program has a version number, and revision level. For example "Version 2.3" represents program Version 2, Revision 3. The version number is changed when a new program description is required. In most cases, the programs display their version number when you run them. This way you can tell if you have the right program description for the version of a program you are using. As new (improved or enhanced) versions of programs become available, replacement program description sheets for your manual will also be provided.

PLEASE NOTE

Specifications and descriptions are subject to change without notice. Updates are available from AMPRO at nominal charge. The contents of this document are believed to be accurate. If errors are found, please notify AMPRO at the address shown on the title page of this document.

Trademarks and registered trademarks used within this document - Z80: Zilog, Inc.; CP/M: Digital Research, Inc.; IBM: International Business Machines; ZCPR3: Richard L. Conn; LITTLE BOARD, LITTLE BOARD/PLUS, SCSI/PLUS: AMPRO.

No part of this document may be reproduced in any form, for commercial purposes, without the express written consent of AMPRO Computers, Inc.

Copyright (C) 1985, AMPRO COMPUTERS INCORPORATED

PLEASE NOTE

The AMPRO Z80 Hard Disk Software diskette (P/N A60057) now includes a file containing the new SCSI BOOT EPROM code, in "Intel HEX" format (SBT-1-0.HEX). Only auto-initializing SCSI hard disk controllers (e.g. Adaptec and Shugart) can be used for automatic SCSI booting. If your controller is of this variety, you can generate your own SCSI BOOT EPROM from this file. The new EPROM is also available from AMPRO, for \$15 plus shipping/handling. Follow the attached procedure in installing your hard disk system for automatic SCSI boot.

LITTLE BOARD/PLUS SCSI BOOT EPROM

November 1, 1985

A. Introduction

The AMPRO SCSI/PLUS bus interface represents a significant advance in single-board-computer architecture, by providing a general purpose, high performance, interprocessor data channel. The most obvious use of the SCSI/PLUS bus interface is for the addition of SCSI (SASI) hard disk controllers and drives. However, this is not the only use for the SCSI/PLUS bus.

B. Boot Algorithm

In order to provide a general purpose "hook" for a variety of SCSI/PLUS applications, the Little Board/PLUS SCSI BOOT EPROM allows the board to "boot" directly from SCSI, without the need for an attached floppy disk drive. Here is what the new SCSI BOOT EPROM does when the board is RESET:

- (1) Checks for the presence of a bootable floppy. If one is present, it attempts to boot from the floppy. If this fails...
- (2) Reads its SCSI ID (0-8) from the ID Input Register, and either:
 - (3) If the board's ID is 7, performs an SCSI bus reset and then attempts to boot from SCSI device ID 0. If this fails, return to step (1).
 - or -
 - (3) If the board's ID is not 7, does not perform an SCSI bus reset. Attempts to boot from SCSI device ID 7. If this fails, return to step (1).

This boot algorithm provides a means to easily establish a master/slave hierarchy, where the device which is jumpered to ID 7 is the system master. It must be responsible to provide the bootstrap software for all devices which are jumpered to addresses other than 7. This process is very straight forward; the bus master simply emulates an SCSI hard disk controller as far as the slave processor is concerned.

C. Installation of Hard Disk Autoboot

In keeping with the general "philosophy" of SCSI, the SCSI BOOT EPROM does not make any assumptions about the type of disk controller it is booting from, nor about any characteristics of the hard disk drive (i.e., tracks, heads, step rate, etc.). For this reason, automatic booting from a hard disk can only be done when using SCSI controllers which initialize themselves automatically on power-up.

Of the controllers supported by the AMPRO Hard Disk Software, the

following may be used for automatic hard disk booting:

Adaptec ACB-4000 and ACB-5000 series
Shugart 1610-4 controller
Xebec OWL combination drive/controller

The Xebec 1410 and 1410A, and the DTC 510A and 510B are not self-initializing. These controllers require the addition of installation-specific commands in either the SCSI BOOT EPROM of the boot strap loader (contained in the HGEN utility), in order to initialize the controller prior to drive access. Hooks have been provided in the SCSI BOOT EPROM code for this purpose. Refer to the source code, available from AMPRO for details.

Here is how to prepare a Little Board-based system for automatic booting from a hard disk. Replace your old BOOT EPROM with the new SCSI BOOT EPROM, and verify that your system still boots from floppies in the normal manner.

Then perform the following steps:

(1) Hardware Setup: Install your system's hardware as described in Chapter 2 of the AMPRO Z80 Hard Disk Software User's Manual. In addition, for hard disk autoboot, SCSI ID's should be set as follows:

- o Jumper the Little Board/PLUS to SCSI ID 7, as indicated in Chapter 2 of the Little Board/PLUS technical manual. (ALL jumper blocks off.)
- o Jumper the SCSI disk controller to SCSI ID 0, according to the controller's installation manual. Also, the drive to be booted from must be connected to Logical Unit Number (LUN) 0 on this controller.

(2) Initial Software Setup: Install and test the standard AMPRO Z80 Hard Disk software and hardware as described in Chapter 2 of the Hard Disk Software User's Manual. This includes:

- o Running HFORMAT, to format the drive.
- o Running HINIT, to initialize your system's BIOS.
- o Running SYSGEN, to write the operating system to the first hard disk partition. This is the "F" drive prior to running SWAP.
- o Running SWAP, to re-assign drive letters the way that you want them. Be sure you have assigned drive letters so that the first hard disk partition is now drive "A."

(3) Final Software Installation: This is the easy part. All you have to do now is run the AMPRO HGEN utility, and respond to the program's prompt with a "Y." That's all there is to it! Remove your floppy diskettes from your system, and press RESET. After a brief delay (5 to 10 seconds) to allow for floppy timeout, your system should boot directly from your hard disk.

* * NOTE * *

Never use the AMPRO SYSGEN utility to copy your operating system from your hard disk boot drive to a floppy, as HGEN has modified the system loader present on the hard disk boot drive

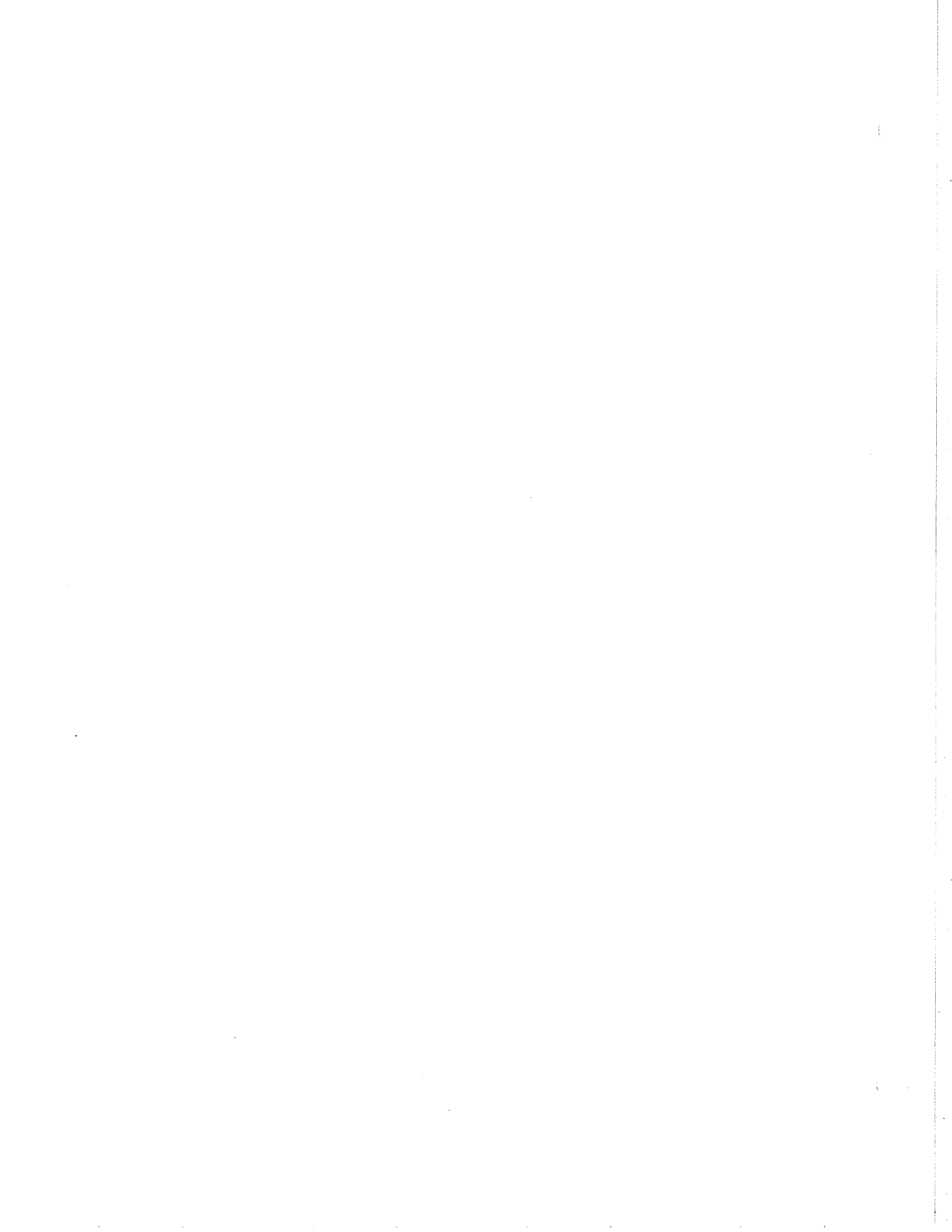


TABLE OF CONTENTS

CHAPTER 1 - GENERAL INFORMATION

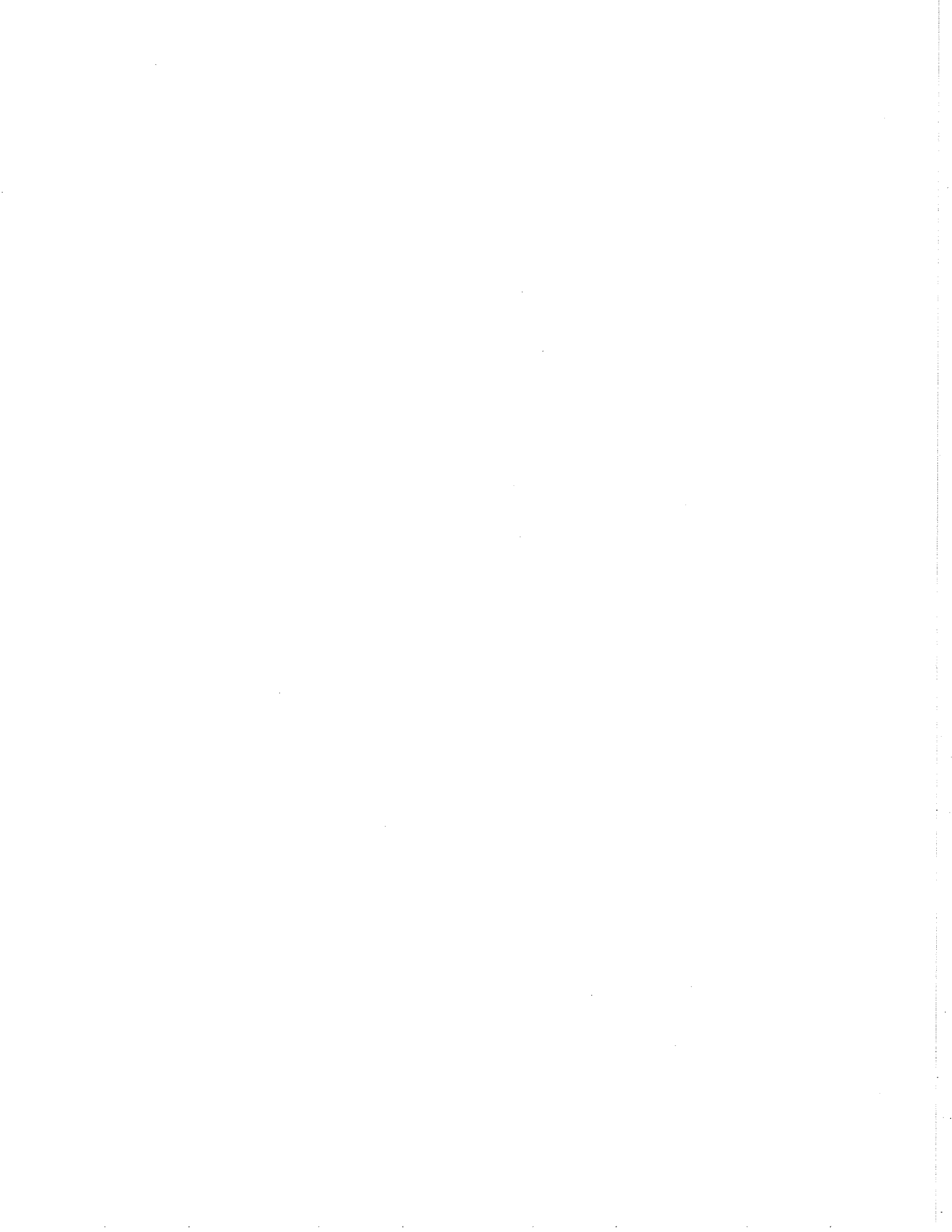
1.1 Introduction	1-1
1.2 Software Description	1-1
1.3 Conventions	1-2
1.4 References	1-2

CHAPTER 2 - SOFTWARE INSTALLATION

2.1 Introduction	2-1
2.2 Hardware Preparation	2-1
2.3 Software Installation	2-2

CHAPTER 3 - PROGRAM DESCRIPTIONS

3.1 Introduction	3-1
3.2 Program Descriptions	3-1
FINDBAD	Bad sector lockout
HINIT	BIOS and controller initialization
HFORMAT	Hard disk formatter
HPARK	Moves hard disk head to landing zone



CHAPTER 1

GENERAL INFORMATION

1.1 INTRODUCTION

This chapter primarily describes the optional hard disk software available for the AMPRO Z80-based single board computers and computer systems. Conventions to be used in program operating instructions are defined, and references for further information are listed.

1.2 HARD DISK SOFTWARE DESCRIPTION

The AMPRO Z80 hard disk software provides support for a wide variety of hard disk system configurations. Use of the hard disk software requires either a system containing an AMPRO Little Board with an SCSI/PLUS Z80 Adapter, or an AMPRO Little Board/PLUS (with built-in SCSI/PLUS interface). AMPRO BIOS Version 3 is also required; the Version 3 BIOS contains generic SCSI (SASI) support, and has been designed to maximize the flexibility of your system's hard disk interface.

The required Version 3 BIOS is available on the Z80 System Software diskette (AMPRO P/N A60101); the hard disk software utilities necessary to format and install hard disk drives and controllers are contained on the Z80 Hard Disk Software diskette (P/N A60057). If you do not already have the Version 3 BIOS, a software update is available from AMPRO for a nominal charge. Source code for the BIOS and utilities is also available from AMPRO.

Here are some of the features of the AMPRO hard disk software:

- o BIOS supports "generic" SCSI (SASI) controllers, and Xebec 1410/1410A
- o Up to eight hard disk controllers, with up to 11 hard disk drives of any size, up to 88 megabytes total storage
- o Variable CP/M partition sizes: 0.1 to 8 megabytes
- o CP/M drive letter swapping

Although the Version 3 BIOS provides "generic" SCSI controller support, the SCSI format command has a number of "vendor unique" options which are not uniformly implemented. Consequently, only a limited number of SCSI controllers are directly supported by the AMPRO hard disk format utility (HFORMAT). Controllers currently supported are:

Adaptec ACB4000
Shugart 1610-4
Xebec 1410, 1410A, and OWL

If you wish to use an unsupported controller, you you can do this by modifying

HFORMAT to be compatible with the desired controller's format command. This usually involves a small change to the source code, which is available for a nominal charge from AMPRO, in the Z80 Software Source Code package.

Most SCSI (SASI) controllers currently being manufactured initialize themselves on powerup from parameters they write to the drive during the format process. Examples are the Adaptec and Shugart controllers, and the Xebec OWL drive. Some controllers -- like the Xebec 1410/1410A -- are not auto-initializing, and must be initialized each time they are powered on. Like the format command, the initialization command is not standardized. The AMPRO HINIT utility provides optional controller initialization for only the Xebec controllers. If you wish to use an unsupported controller that does require initialization, you will have to modify HINIT to meet your controller's requirements; source code to HINIT is available in the Little Board Source Code package.

1.3 CONVENTIONS

In the descriptions of the use of software utilities, terminal keyboard inputs which you will make to the system are shown underlined. This has been done to make it easy for you to distinguish between the computer's prompts and the operator's keystrokes. For example:

```
A0><u>HFORMAT <RETURN>
```

Also, certain keys on your terminal's keyboard have special uses. The control key, generally labeled CTRL, is meant to be pressed at the same time as one other key. The required control key combination will be represented as follows: <CTRL-C> = control key pressed along with C key.

Two other special keys are the "escape" key, indicated by <ESC> and the "return" key (also called the "carriage return" or "enter" key), indicated by <RETURN>. In general, all commands you enter from the CP/M (or ZCPR3) command prompt require you to press the <RETURN> key to begin the operation, as in the example above.

1.4 REFERENCES

Only brief references are made in this manual to the use and operation of some required software utilities. Whenever a software utility is mentioned, it will either be called an AMPRO utility, a CP/M utility, or a ZCPR3 utility. This way you will know where to obtain further information on the program's use.

Refer to the AMPRO Z80 Software User's Manual (P/N A74006) for further information on other AMPRO-supplied software programs and utilities.

CHAPTER 2

SOFTWARE INSTALLATION

2.1 INTRODUCTION

This chapter provides information on how to configure and install the AMPRO Z80 hard disk software. The required hard disk utilities are contained on the Z80 Hard Disk Software diskette (AMPRO P/N A60057). The AMPRO Version 3 BIOS, present on the Z80 System Software diskette (P/N A60101), is also necessary.

This chapter will guide you through the required hardware and software setup and installation procedures. Refer to Chapter 3 for detailed program descriptions and operating instructions on the hard disk software utilities, and to the AMPRO Z80 System Software User's Manual (P/N A74006) for instructions on the use of other AMPRO software.

NOTE

Any modifications to the system parameters should only be performed using your backup disks. **Do not** modify the disks shipped with your system.

2.2 HARDWARE PREPARATION

The AMPRO Z80 hard disk software assumes the presence of either a Little Board, with the Z80 SCSI Adapter option, or a Little Board/PLUS, with built-in SCSI interface. Be sure that conductor number 1 of the 50-conductor SCSI bus cable is plugged into pin 1 of the SCSI 50-pin connector. Often, the flat ribbon cable will have a red strip on one edge, indicating the location of conductor number 1. Also be sure that two -- and only two -- SCSI bus devices have their resistor terminator networks installed.

You may connect up to eleven drives, on up to eight controllers, to your system. The drives may be any size, up to the system maximum of 88 megabytes. Various drive and controller types may be intermixed.

You will need the following information on each drive to be used:

- o Number of cylinders
- o Number of heads
- o Step rate
- o Cylinder number to begin write precompensation (if needed)
- o Cylinder number to begin reduced write current (if needed)

In addition, each type of hard disk drive usually has a jumper configuration area, generally located near the 34-pin drive cable connector. Consult your drive's documentation to determine how it needs to be jumpered. If only one drive is connected to a controller, it should be jumpered as the first logical unit (LUN0), and connected to the connectors provided for that unit. When two or more drives are connected to a single controller, they are jumpered to

different logical unit numbers, and connected to the appropriate connectors on the controller.

Here are some notes on the installation of several controller types:

Xebec 1410, 1410A, and OWL - set the sector size jumper, labeled "SS" to the 512 byte sector position, labeled "5." Set the controller address. This is a trace cut option on either the 1410 or the OWL. The Xebec controllers are shipped jumpered for SCSI bus address 0, so if you are only using one controller use that bus address.

Adaptec ACB4000 - set the SCSI device ID jumper, J5, for the desired controller ID. The ID is specified as a 3-bit binary code, with the least significant bit corresponding to A-B and the most significant bit E-F. The associated bit is a 0 when the jumper is off. If only one controller is on your SCSI bus, you can select bus address 0 by leaving off all of the address jumpers. The jumpers near J1, labeled T, PU, R and S are used for write precompensation setup. Generally a single jumper should be inserted here, between R and S.

Shugart 1610-4 - set its SCSI bus ID jumper (jumper pairs CU1, CU2, and CU4) for the desired controller address in the same manner as with the ACB4000. CU1, CU2, and CU4 on the Shugart controller correspond to jumper pairs A-B, C-D, and E-F on the Adaptec controller. No other jumpering is required.

Adaptec and Shugart ID Jumpering

SCSI Bus ID	Adaptec			Shugart		
	D-E	C-D	A-B	CU4	CU2	CU1
0	out	out	out	out	out	out
1	out	out	in	out	out	in
2	out	in	out	out	in	out
3	out	in	in	out	in	in
4	in	out	out	in	out	out
5	in	out	in	in	out	in
6	in	in	out	in	in	out
7	in	in	in	in	in	in

NOTE

If you are using multiple controllers, be sure only one of them has its SCSI bus termination resistors installed.

2.3 SOFTWARE INSTALLATION

Make a copy of your normal system diskette, and label it as your "Hard Disk System" diskette. Copy the hard disk software utilities from the Z80 Hard Disk Software diskette onto the new Hard Disk System diskette. Use the following procedure to generate your custom hard disk system configuration.

Step 1: Configure your CP/M system size.

The more hard disk storage your system has, the more memory space CP/M requires for storage of directory-related information. As CP/M's space requirements increase, the space available to programs (called the Transient Program Area, or "TPA") decreases. The AMPRO ZMOVCPM and MOVCPM utilities are used to generate various size systems; ZMOVCPM creates systems containing ZCPR3, while MOVCPM creates systems without ZCPR3.

Here are the required sizes for various options of hard disk storage, based on the AMPRO BIOS Version 3 with the standard (built-in) ZCPR3 support:

Hard Disk System Sizes

Hard Disk Storage	CP/M Size	TPA bytes
(none)	60K	56,070
1-10 MB	59K	55,046
11-42 MB	58K	54,022
43-74 MB	57K	52,998
75-88 MB	56K	51,974

NOTE

A usable 60K system, without hard disk support, cannot be generated with ZMOVCPM or MOVCPM. Instead, the 60K floppy-only system is provided in bootable form on the system tracks of the standard AMPRO Z80 System Software diskette.

Determine the required system size, and use the AMPRO ZMOVCPM (or MOVCPM) utility to create a system image file configured for your requirements. (The required system size will be one or more K bytes smaller if ZCPR3 configurations other than the "standard" built-in one are to be used.)

NOTE

ZMOVCPM.COM (or MOVCPM.COM) must be run directly from the CP/M (or ZCPR3) command line, not from a shell like MENU or FRIENDLY, since it prepares a memory image which might be corrupted by the shell program. Use the shell's exit command (X, <CTRL-C>, etc.) to return to the A0> prompt before running ZMOVCPM (or MOVCPM).

Example, to create a 59K system, use the command:

```
A0>ZMOVCPM 59 *<RETURN>
```

Number "59" provides enough space for up to 10 megabytes of hard disk. e. ZMOVCPM will respond:

CONSTRUCTING 59K CP/M vers 2.2
READY FOR "SYSGEN" OR
"SAVE 41 CPM59.COM"
A0>__

Step 2: Write the new system to your Hard Disk System diskette.

At this point, the new system image is stored in memory. Use the AMPRO SYSGEN utility to write the new system image onto the system tracks of a new (preformatted) diskette, by typing:

A0>SYSGEN<RETURN>

When SYSGEN requests the Source drive, respond with <RETURN> only, to tell SYSGEN to take the source from memory. When SYSGEN requests the Destination drive, enter the desired floppy drive letter. After you specify the Destination floppy drive letter, SYSGEN will prompt you for a <RETURN>. After SYSGEN writes the system to the Destination floppy diskette, the Destination drive prompt will again appear; this time, respond with a <RETURN> or <CTRL>-C to exit SYSGEN.

Step 3: Set the powerup port configurations.

Use the AMPRO CONFIG utility to set the required port configurations and other initialization parameters on your new Hard Disk System diskette. This is required even though your current system disk's parameters may already have been set, because ZMOVCPM has created a system image containing the standard defaults (9600 baud terminal, etc.), which may differ from your system's requirements.

Step 4: Test the new system disk.

Try booting from the new Hard disk System diskette. It should indicate the CP/M system size number (e.g. 59K CP/M), and BIOS Version 3, in the sign-on message. In case of difficulty, re-boot your system with a working system disk and use the CONFIG utility to recheck the terminal port setting, step rates, etc.

Step 5: Format your hard disk.

You can now proceed to format your hard disk drive. Before you can use the AMPRO hard disk formatter (HFORMAT), you must complete the above steps, and you must have booted from the new Hard Disk System disk. HFORMAT will not run otherwise. Refer to the description of the AMPRO HFORMAT utility (Chapter 3).

WARNING! HFORMAT will destroy all data on the hard disk drive.

You need to have the following information handy when you run HFORMAT:

Controller: brand and model

Drive: number of cylinders
number of heads
step rate
cylinder at which to begin write precompensation, if needed
cylinder at which to begin reduced write current, if needed
cylinder to use for landing zone, if needed

In case of difficulty, re-boot from your Hard Disk System disk and verify from the system sign-on message that you have a 59K (or less) size CP/M system, and BIOS Version 3 or later. If these are correct, the trouble may be:

- o Faulty hard disk controller, drive, or cables
- o Incorrect controller or drive jumper settings
- o Incorrect controller or drive information given to HFORMAT
- o Incompatibility between hard disk controller and drive
- o Problem with your system's SCSI interface

Step 6: Test hard disk initialization.

Since the AMPRO BIOS supports a wide variety of SCSI (SASI) controllers and drives, as well as a wide range of CP/M drive partition sizes, the operating system must be initialized for your specific configuration. This will be done immediately **after** system boot, by the AMPRO HINIT utility.

Refer to Chapter 3 for information on HINIT. At this point, you should run HINIT in its interactive "menu mode," responding to the program's prompts according to the characteristics of your hardware. Keep a list of all of your keystroke responses, as you will be making an automatic command file (alias) out of them shortly. You can create as many CP/M letter partitions on each drive as you like, using up the available K bytes of formatted capacity of the drive.

After running HINIT (menu mode), try reading the directory of each of your new (empty) hard disk partitions using the DIR utility. If this doesn't work, or if the directory sizes do not seem correct, re-enter the initialization requirements with HINIT. If you still have problems, refer to the list of possible causes in the previous step.

Step 7: Bad sector lockout.

Since a hard disk drive has an extremely dense recording format, tiny defects can result in storage areas which are unusable. Typically, only several K bytes of disk storage space will be lost due to these defects.

The Public Domain FINDBAD utility (Chapter 3) performs a read of every sector of each CP/M drive letter partition you specify, logging all errors encountered during the process. After the specified drive partition is fully checked, all of the bad sectors are grouped in a single file, called [UNUSED].BAD, in user area 15. Once the defects are grouped into this special file, no attempt to read or write to those areas of the drive will be made (unless you erase the file).

Run FINDBAD once for each drive partition. For example:

```
A0>FINDBAD F;;FINDBAD G;;FINDBAD H;;FINDBAD I:<RETURN>
```

Step 8: Create and test an automatic hard disk initialization alias.

In this step, you will use the ZCPR3 ALIAS utility to create an "alias" command file to automatically initialize your hard disk system. The required alias contains the precise set of HINIT keystrokes you used in Step 6, above, but with a comma used to represent the <RETURN> key and a period used to represent the <ESC> key. A good name for the alias you create is HARDINIT.COM. You can use a separate alias to initialize each drive or controller, or one alias can initialize them all.

For example, the following HARDINIT alias command line might be used to initialize two 5 megabyte drive partitions (F and G) on a single 10 megabyte drive, connected as logical unit 0 (LUN0) on an Adaptec controller:

```
HINIT YD010 AF5000,AG5000,.
```

Refer to the program description of HINIT (Chapter 3) for further details.

After you create the HARDINIT.COM alias required by your system, test it as follows: re-boot from your Hard Disk System diskette, to clear out the effects of Step 6; then type the command:

```
A0>HARDINIT<RETURN>
```

Then try again to access the (empty) directories of all of the drive letters your alias has just defined, using the DIR utility. (Problems? See Step 6.)

Step 9: Write your system tracks to the first hard disk partition.

Use the AMPRO SYSGEN utility to copy the system tracks from your Hard Disk System diskette to your first hard disk partition.

Step 10: Configure your system disk for automatic initialization.

In this step you will configure your Hard Disk System disk to perform all required initialization, including complete transfer of operation to your hard disk partitions. To do this, you create another ALIAS which gets everything going just the way you want it.

Here is an example. Let's assume you have created a system with two 10 megabyte drives, partitioned as four 5 megabyte CP/M drive letters, F through I. The standard system disk contains an alias which is useful for this configuration, called HSTART.COM. It contains the following command line:

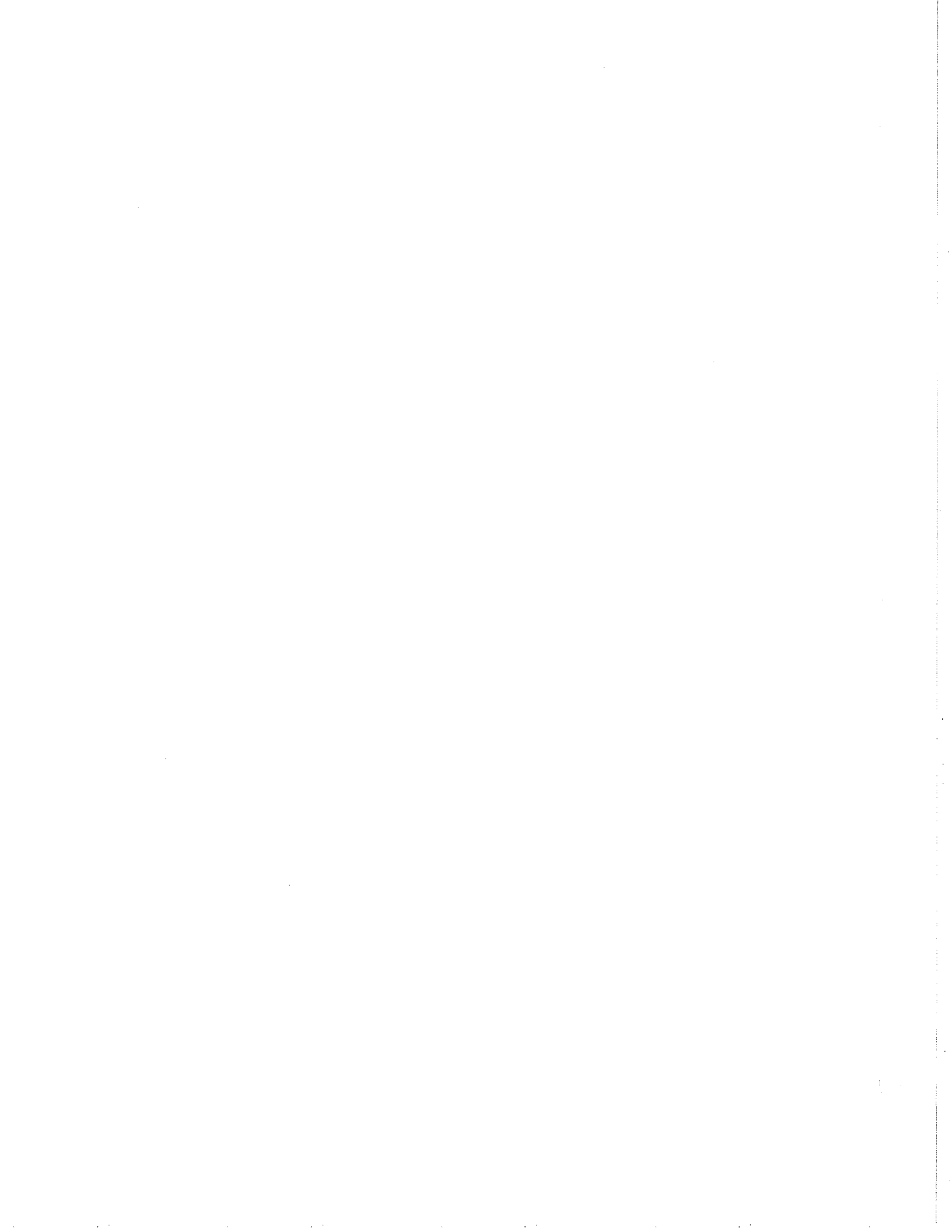
```
HARDINIT; SWAP AF BG CH DI.; STARTUP
```

This performs the following three steps:

- (1) Initializes your hard disk system, according to the alias `HARDINIT.COM` you created in Step 8.
- (2) Reassigns drive letters, such that floppy letters A-D are swapped with hard disk drive letters F-I.
- (3) Continues operation from the first hard disk drive partition (now called "A"), based on the contents of a `STARTUP.COM` alias you put there.

One final touch: set "HSTART" as the system autocommand, using the `AMPRO CONFIG` utility. Now, when you boot from your Hard Disk System diskette, your hard disk system will automatically be initialized, the hard disk drive letters (F-I) will be swapped with the floppy drive letters (A-D), and the command line contained in the `STARTUP.COM` alias on the first hard disk drive partition will run. Try it!

```
.....  
:                                     :  
:               CONGRATULATIONS!     :  
:                                     :  
:   This completes your hard disk system installation. :  
:                                     :  
:                                     :  
:.....
```



CHAPTER 3

PROGRAM DESCRIPTIONS

3.1 INTRODUCTION

This chapter contains detailed information on each of the AMPRO Z80 hard disk software utility programs supplied on the AMPRO Hard Disk Software diskette. Each program's description explains what the program does and how it is used. The utilities are covered in alphabetical order, so this material can serve as a handy operator's reference.

Each program description is identified with a version number. When the utility program is run, its version number (and a revision level) appear in the program's sign-on message, for example:

```
        AMPRO Copy/Format/Verify Utility
        Copyright (C) 1984 AMPRO Computers, Inc.
        Version 1.6
```

In this case the program is Version 1, Revision 6. Revisions of a utility program having the same version number operate in the same manner. If a future version of a particular utility program requires a new description, its version number will be changed, to indicate that the old description is no longer accurate. Program descriptions for the new program version will be available, so that you can update this manual.

3.2 PROGRAM DESCRIPTIONS

The following pages contain the program descriptions of the AMPRO Z80 hard disk software utilities, alphabetically arranged.



FINDBAD

(Version 6)

Description

Since a hard disk drive has a very densely packed storage surface, minute defects can result in small areas which are unusable. Some controllers can automatically "map" these bad spots (defects) out of use during operation, while others can not. The AMPRO hard disk BIOS does not utilize this option, since each controller does this differently. Instead, the FINDBAD bad sector lockout program is used to allocate all of the bad areas of a disk's surface into a single file. As long as you never erase this file, the bad spots on the disk will never be used for program or data storage. Typically only several K bytes of disk storage will be lost due to these defects.

The FINDBAD utility is a public domain program which provides bad sector lockout. FINDBAD performs a read of every sector of the CP/M drive letter you specify, logging all errors encountered during the process. After the specified drive is fully checked, all of the bad sectors are assigned to a single file, called [UNUSED].BAD (in user area 15). Once the defects are grouped into this special file, no attempt to read or write to those areas of the drive will be made (unless you erase the [UNUSED].BAD file).

Operation

Before you can use FINDBAD, your hard disk must be formatted (see HFORMAT), and your system initialized (see HINIT). You must run FINDBAD once for each drive letter you will be using, even if several drive letters represent CP/M partitions on the same drive.

To run FINDBAD, you simply type the program's name along with the CP/M drive partition letter, followed by a colon. For example, to run FINDBAD on drive letter F, enter the following command from the CP/M command line:

```
A0><u>FINDBAD F:</u><RETURN>
```

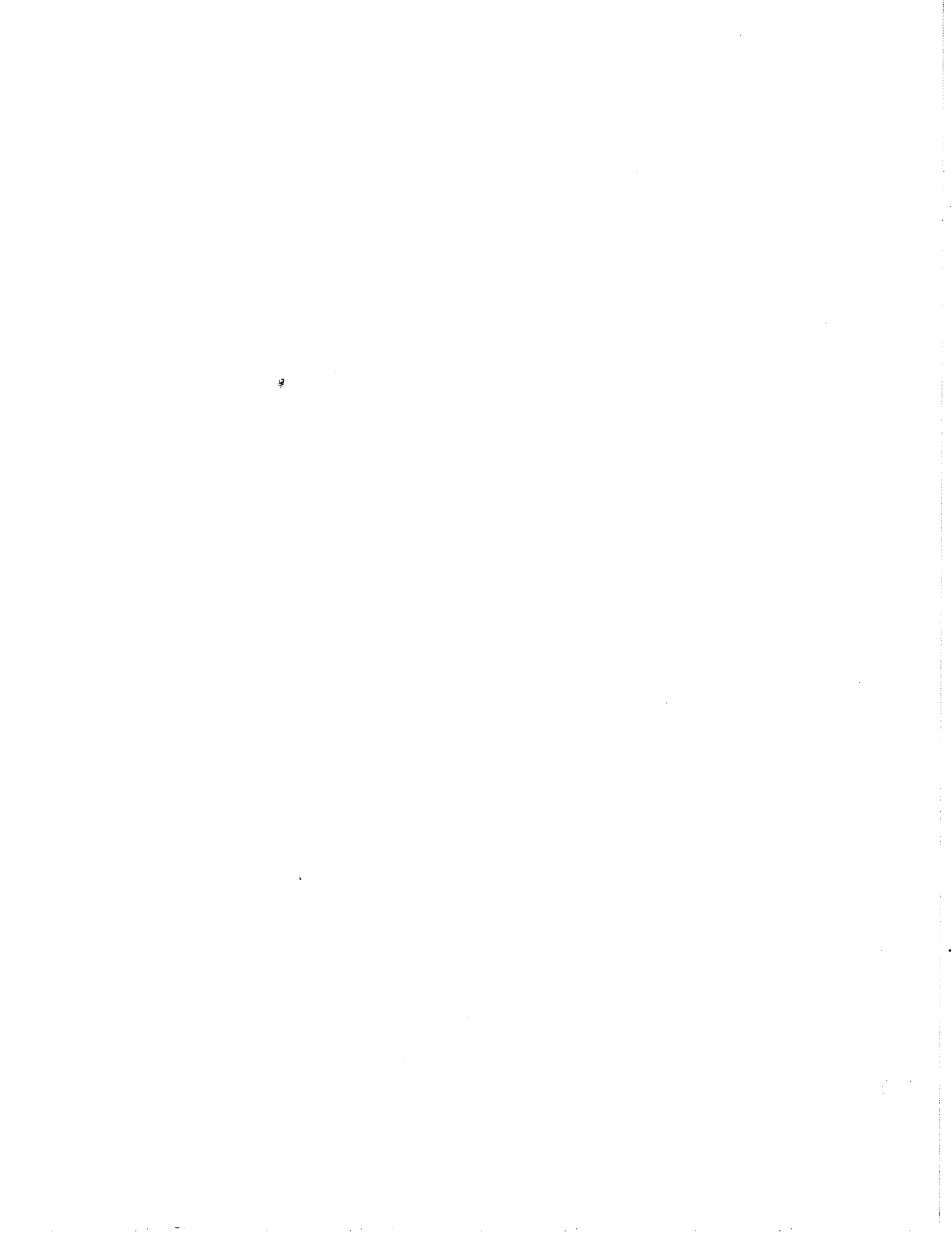
The program will display status messages indicating its progress, and the locations (if any) at which bad disk sectors are found.

NOTE

The file created, [UNUSED].BAD, is placed in user area 15 by FINDBAD. It must be left as is. To protect it, you can use the following STAT.COM command:

```
A0><u>STAT [UNUSED].BAD $R/O</u><RETURN>
```

You can also run FINDBAD on a drive which is not empty, if you suspect that it contains bad sectors. In this case, files already on the drive will not be harmed, and the resulting [UNUSED].BAD file will collect all blocks containing bad sectors at that time.



HINIT

(Version 1)

Description

This utility performs disk controller initialization and CP/M BIOS installation for use of hard disk drives and SCSI hard disk controllers. When your system boots (on power-up or reset), a number of BIOS parameters must be set prior to access of the hard disk drives. Also, some hard disk controllers require initialization before use. HINIT provides the facility for performing these operations. HINIT has two modes of use: Menu Mode, and Command Line Mode.

HINIT provides controller initialization for only one type of hard disk controller, the Xebec 1410 (or 1410A). Most SCSI hard disk controllers currently being manufactured do not require initialization. These self-initializing controllers are referred to as "SCSI Generic." They automatically initialize themselves on powerup from parameters which they write to a reserved area on the hard disk drive when they format the drive. Controllers known to be SCSI Generic are: Adaptec, Shugart, Xebec OWL.

IMPORTANT

Since SCSI Generic controllers depend on initialization information which they write to the disk drives during format, your drives must be formatted first, before you attempt to use HINIT.

The AMPRO Version 3 BIOS classifies SCSI Generic controllers in two ways: those which can support burst data transfer over SCSI, and those which require byte-by-byte handshaking. HINIT prompts you to select either "burst-mode" or "byte-mode" for SCSI Generic controllers. As implemented, burst mode controllers provide a 10-20 percent speed advantage over byte mode controllers. The Adaptec ACB4000 can be used in burst mode, while the Shugart 1610-4 requires byte mode. These designations are specific to the AMPRO application, and are not standard terms, so you won't find this information in your SCSI controller's technical manual. It is recommended that you use any unlisted controller in byte mode, until you thoroughly exercise the system with the burst-mode specified.

Menu Mode Operation

To invoke HINIT's interactive menu mode, simply type the program's name from the CP/M command line:

```
A0>HINIT<RETURN>
```

HINIT will prompt you for all required information. The initial prompt asks:

```
Do you want to clear the existing hard disk assignments (Y/N)?   
```

This provides the option of doing all desired initialization in one run of HINIT, or using HINIT several times to initialize multiple drives or

controllers.

HINIT next displays its main menu:

Options available:

- (D) Define the Current Drive
- (A) Add a partition to the Current Drive

(ESC) Exit the program

What next (D/A/ESC)? ___

The steps you will perform are:

Defining the "Current Drive" - use the "D" option, to define a new "current drive." HINIT will prompt you for the controller's SCSI ID and type, and the drive's logical unit number (LUN), as jumpered on the drive. A single disk drive may contain more than one CP/M drive letter partition, as specified in the next step.

Since the Xebec 1410 and 1410A controllers require power-up initialization, when you specify one of those controllers you will be prompted to supply the same drive-related information required by HFORMAT, namely: number of cylinders, number of heads, RWC cylinder, WPC cylinder, and drive step rate. (See HFORMAT for more information.)

Defining CP/M partitions on the drive - after you define the current drive, HINIT will return you to the main menu options (D/A/ESC). Use the "A" option to define the CP/M drive sub-partitions to be built on the current drive. HINIT will prompt you for a CP/M partition letter (starting from F) and a partition size. The drive partition size, entered in Kbytes, can be anything from 1 to 8192 (Kbytes). (A Kbyte is 1,024 bytes.)

A drive partition can be thought of as a "logical" disk drive, and has its own drive letter (F, G, H, etc., up to P). A single disk drive may have as many CP/M partitions as you wish. However, there are only 11 available partitions, so use them wisely!

After each additional CP/M partition you specify, HINIT will return to its main menu, asking you if you wish to define an additional CP/M partition on the same ("current") disk drive, define a new (additional) "current" disk drive, or exit to CP/M. When you are finished specifying all the required drives and partitions, use the ESC option to exit HINIT.

NOTE

Each time you specify a CP/M drive partition size, HINIT will indicate how much BIOS buffer space remains. If you run out of space, you will need to use the AMPRO ZMOVCPM utility to make a smaller size CP/M system. If you plan to install an alternate ZCPR3 configuration requiring additional BIOS buffer space (1K, 2K, etc.), be sure HINIT indicates sufficient buffer space

remains for your requirement after the highest letter CP/M drive partition is specified.

The total formatted capacity of a given drive depends on the controller to which it is connected. Assuming the controller you are using reserves one cylinder for its own use, you can calculate the available space on each hard disk drive from the number of cylinders (CYLS), the number of heads (HDS), and the number of 512 byte sectors your controller writes on each cylinder (SECTORS), as follows:

$$\text{FORMATTED CAPACITY} = [\text{CYLS}-1] \times \text{HDS} \times \text{SECTORS} \times .5$$

The value you use for SECTORS depends on the particular controller, and, in the case of Adaptec controllers, on the interleave you specified when you used HFORMAT to format the drive. Typical values are:

Adaptec ACB4000:	18	(with interleaves of 2 or more)
Adaptec ACB4000:	17	(with interleave of 1)
XEBEC 1410, 1410A:	17	(all interleaves)
Shugart 1610-4:	17	(all interleaves)

After you run HINIT in its Menu mode, try the ZCPR3 DIR program (Chapter 5) on each CP/M partition you have created. The DIR program will show the space allocated to each CP/M partition. The number of Kbytes of space should be 32K smaller than what you specified in HINIT, due to directory space requirements. After you have established that all CP/M partitions are as you wish, run the Public Domain FINDBAD program to remove any bad sectors from usable disk space. FINDBAD will also let you know if you have attempted to use more capacity than your drive actually has.

Command Line Mode Operation

HINIT would not be a very useful program if you had to manually enter all required information every time you booted your system. However, thanks to HINIT's Command Line mode, you can create a ZCPR3 ALIAS to initialize your system for you.

To create an automatic initialization ALIAS, first run HINIT in its Menu mode, and keep a careful record of all of your keystrokes. After checking to see that everything has been initialized to your satisfaction, create an ALIAS containing a command line containing:

HINIT ¶parameters[®]

where the parameters are characters representing the precise sequence of keystrokes you used when you used HINIT in its Menu mode, with the following exceptions: substitute a comma for the <RETURN> key, and a period for the <ESC> key. Be sure to end with a period, corresponding to the <ESC> used to exit HINIT back to CP/M.

Here is a sample HINIT command line:

```
HINIT YD010 AF5000,AG5000,.
```

It does the following:

```
Y      clear any previous initialization
D      indicates a new "current drive"
0      controller SCSI ID is 0
1      controller type is SCSI burst-mode generic
0      drive logical unit number (LUN) is 0
        (space has no effect; used for clarity)
A      additional CP/M partition on the same drive
F      selects CP/M partition letter F
5000   size of CP/M partition F is 5,000K bytes
,      represents <RETURN>
        (space has no effect; used for clarity)
A      additional CP/M partition on the same drive
G      selects CP/M partition letter G
5000   size of CP/M partition G is 5,000K bytes
,      represents <RETURN>
.      represents <ESC>
```

Spaces can be used in the command line for ease of understanding; they have no effect. Both CP/M partitions defined in this example are on the same physical drive. If the two 5 MB partitions were on separate drives (e.g. LUN0 and LUN1 on the same controller), the command line would have been:

```
HINIT YD010 AF5000,D011 AG5000,.
```

HFORMAT

(Version 2)

This program is used to format hard disk drives using one of several types of SCSI hard disk controllers. You will need to provide the following information concerning the controller and drives:

Controller make and model

Controller SCSI ID - depends on jumper settings on the controller board

Drive "logical unit number" - depends on which connector the drive's data cable is connected to. Drive must also be correctly jumpered according to this number.

Drive characteristics - these are obtained from the drive's technical manual or data sheet. These are:

- Number of cylinders
- Number of heads
- Starting cylinder for reduced write current (RWC), if needed
- Starting cylinder for write precompensation (WPC), if needed
- Landing zone cylinder
- Drive step rate

Operation

HFORMAT is easy to use, providing you have the required information (listed above) available. Run the program by typing its name at the CP/M command line:

A0>HFORMAT<RETURN>

The program requests all of the required information. If your drive does not require reduced write current (RWC) or write precompensation (WPC), respond with a number greater than the highest cylinder number. If your drive does not have a special landing zone (for head positioning prior to power-down), use "number of cylinders + 1" for that parameter.

HFORMAT requires BIOS Version 2 or later. If your BIOS is not configured for hard disk drives, the program will indicate so and will return to CP/M.

After you format a drive with HFORMAT, it is recommended that you immediately run the Public Domain FINDBAD program to group any bad sectors into a reserved file.

HFORMAT supports only a limited number of SCSI controllers. Those supported at the time of this writing are:

- Adaptec: ACB4000
- Shugart: 1610-4
- Xebec: 1410, 1410A, OWL (drive/controller)

HPARK

(Version 1)

Description

The HPARK utility moves the read/write head(s) of one or more hard disk drives to a predefined safety zone on the disk surface, to guard against accidental data loss due to either power on/off glitches in the drive electronics or media damage due to mechanical shock. You should always use HPARK to "park" your drives' heads prior to switching off AC power. After HPARK finishes positioning the disk heads it halts all system operation; the only recovery from system lockup after HPARK is by means of a system reset.

Menu Mode Operation

To use HPARK in its interactive "menu" mode, type the program's name from the CP/M command line:

```
A0>HPARK<RETURN>
```

For each drive you wish to park, you will be prompted the following information:

- (1) SCSI ID of the controller to which the drive is connected (0-7)
- (2) Drive Logical Unit Number (0, 1, 2, or 3)
- (3) Controller type, if listed
- (4) Block number location for head positioning. This step is skipped if your controller type is "listed" (Step 3). If your controller is unlisted, then you are prompted for a block number. Calculate the block number as follows:

```
.....  
:  
: block number = [cylinders X sectors/track X heads] - 1 :  
:.....
```

Where

cylinders = the number of cylinders on the drive; or the special parking zone cylinder number, if the drive provides one

sectors/track = the number of 512 byte sectors the controller formats per track. This depends on the controller. For example...
Xebec: 17; Adaptec: 18 for interleaves of 2 or greater, but 17 for an interleave of 1; Shugart: 17.

heads = the number of heads on the drive

NOTE

The Xebec 1410/1410A controllers do not allow HPARK to seek beyond the last cylinder number defined by the HINIT utility. One way to get around this is to define one more cylinder in your HINIT alias than you are actually using in your CP/M drive partitions. Then use this extra cylinder as your parking cylinder.

Command Line Mode Operation

Once you have tested HPARK on your drives using the program's menu mode of operation, you will probably want to create a ZCPR3 alias to park your drives, in the same way you use an alias to perform the powerup initialization of your hard disk system (see HINIT).

HPARK allows you to specify everything directly from the command line, or from within an alias. The general form of the command line mode usage is:

```
A0>HPARK parameters@
```

As with HINIT, the "parameters" in the command line consist of the keystrokes you use in HPARK's menu mode, except that you substitute a comma for the <RETURN> key. End the parameters with a period.

Use the ZCPR3 ALIAS program to create a command file containing the HPARK command line required to park your system's drives. Here are some examples:

This HPARK command line is for a pair of Miniscribe 3012 drives connected as logical units 0 and 1 on a Xebec controller with SCSI ID 0:

```
HPARK 00#20807,01#20807,.
```

Here is one for the same pair of drives connected to an Adaptec ACB4000:

```
HPARK 00A,01A,.
```

The ACB4000 is listed in HPARK's controller type list, so the letter "A" takes the place of the block number.

Installation and Operation Guide

SMARTCLOCK REAL TIME CLOCK UPGRADE

For Little Board/PLUS

Description

The "SMARTCLOCK" is a low-cost plug in battery backed real time clock module for use with the AMPRO Little Board/PLUS (tm) single board computer. The SMARTCLOCK option is based on a Dallas Semiconductor "Smart Watch" real time clock device, which plugs into a BIOS EPROM socket (under the EPROM).

Since there is no standard system clock function in the CP/M or Z-SYSTEM operating system environments, the use of the date and time information is generally dependent on your specific application software. A utility has been included with the SMARTCLOCK module which allows you to set, read, and test the SMARTCLOCK module. Often, routines to access the SMARTCLOCK module must be incorporated directly into your application program; for this reason, the source code to the utility has also been included.

Use of the SMARTCLOCK with a Little Board/PLUS requires one cut and one jumper on the bottom of the board.

PLEASE NOTE

The SMARTCLOCK option can only be used with the AMPRO Model 1B Little Board/PLUS, and not with the original Model 1A board. The Model 1B board is distinguishable from the Model 1A by the presence of a 28-pin (rather than a 24-pin) EPROM socket.

Installation

CAUTION

The Smart Watch module, the EPROM device, and your computer system are all static sensitive and should be handled with care. Static discharge from your body to these components can damage them irreparably. Therefore, to avoid damage to your system, we recommend that you refer the installation of the SMARTCLOCK option to a qualified technician.

Remove the system boot EPROM (U13), plug the Smart Watch module into the EPROM socket, and then install the boot EPROM on the socket provided by the Smart Watch module. Be sure to orient the Smart Watch device so that pin 1 of the socket on its top corresponds to pin 1 of the board's EPROM socket. Most IC sockets have a semicircular notch at the pin 1 end (between pins 1 and 28).

The following two modifications must be made to the board before the SMARTCLOCK option can be used:

CAUTION

If not performed carefully, these modifications can damage your computer board, and will invalidate your warranty. We therefore recommend that you have this procedure performed by a qualified technician.

- (1) On the back side of the board cut the trace going to pin 20 on the 28 pin EPROM socket labeled U13.
- (2) Solder a jumper between pin 20 of U13 (the EPROM socket) and pin 21 of U4 (the microprocessor socket).

Operation

To verify that your SMARTCLOCK option is properly installed, use the SMARTCLK.COM utility supplied with the device. It has the following commands:

- | | |
|-------------------------------|--|
| A0><u>SMARTCLK</u><RETURN> | Displays current date and time. |
| A0><u>SMARTCLK /S</u><RETURN> | Sets SMARTCLOCK date and time. You will be prompted for the information required. |
| A0><u>SMARTCLK /T</u><RETURN> | Tests the SMARTCLOCK device by continuously reading it and displaying the date and time on your terminal screen. |
| A0><u>SMARTCLK ?</u><RETURN> | Displays help information. |

If your application program requires routines to access the date and time information within the SMARTCLOCK module, use sections of the SMARTCLK.ASM file on the SMARTCLOCK software diskette.

DIFF

(Version 2)

Description

DIFF is used to compare two files. You can use DIFF to simply state if the two files are different (stopping immediately after the first difference is located) or to list all of the differences between two files on a byte-for-byte basis.

Operation

The normal way to run DIFF is by typing a command like:

```
A0>DIFF B0:FILE1.TYP,C5:FILE2.TYP
```

This compares the file FILE1.TYP in directory B0 with the file FILE2.TYP in directory C5: The usual ZCPR3 convention of allowing you to skip the drive letter or user number holds. Here are some more examples:

```
A0>DIFF FILE1.TYP,FILE2.TYP    ...both files are in A0
```

```
A0>DIFF B:FILE1.TYP,C5:FILE2.TYP ...FILE1.TYP is in B0
```

```
A0>DIFF FILE1.TYP,15:FILE2.TYP ...FILE1.TYP is in A0, FILE2.TYP is  
in A15
```

```
A0>DIFF B10:FILE.TYP          ...compares file in B10 to file in A0  
having same name
```

Two options may also be included on the command line:

C - stop at first difference

M - multiple runs: when a comparison is complete, prompt for new disks

DIR

(Version 1)

Description

DIR might be the most commonly used ZCPR3 utility. In the AMPRO operating system, the directory function is based on a disk-based utility, rather than being internal within the operating system itself. As a result, what you get when you type the command

```
A0>>DIR<RETURN>
```

Depends on the features of the disk-based program with the name "DIR.COM".

The ZCPR3 DIR utility has a number of useful features, including alphabetical arrangement of file names, inclusion of file lengths (in K bytes), and display of how much space is available on the disk drive.

Operation

Like most ZCPR3 utilities, DIR allows you to specify the desired directory drive and user area, and to use "wildcards" in the file name specifier. Some variations are:

```
A0>>DIR B12:<RETURN>           ...includes all files in directory b12
A0>>DIR B12:*.COM<RETURN>       ...includes files matching *.COM in B12
A0>>DIR B12:A????.*<RETURN>     ...includes files matching A?????.* in B12
A0>>DIR 15:<RETURN>           ...includes all files in directory A15
A0>>DIR B:<RETURN>             ...includes all files in directory B0
```

In addition, you can include the following DIR options on the command line:

```
A0>>DIR B15:*.*<RETURN> S     ...display "System" files only
A0>>DIR B15:*.*<RETURN> A     ...display "All" files (includes system
                             and non-system files)
A0>>DIR B15:*.*<RETURN> T     ...sort by file Type (extent)
```



DISK7

(Version 7)

Description

DISK7 is a valuable disk housekeeping utility. The program provides a menu from which you can select the operation you wish to perform. All commands consists of single keystrokes. You can copy, delete, rename, view, and print files easily using DISK7.

Operation

You start DISK7 like most other programs.

A0>DISK7<RETURN>

The program then displays its menu of options:

```
                DISK 7.7 -- File Manipulation Program -- 02/11/84
C - Copy file   | D - Delete file   | F - Forward 22   | G - Group copy
J - Jump to fn.ft | L - Length of file | N - New DIRectory | P - Print text
R - Rename file  | S - Stat of disk  | T - Tag file     | U - Untag file
V - View text    | X - Exit to CP/M  | <SP> advances cursor -- B backs up
```

238k bytes free on DIRectory A:

A: AMPRODSK.COM : ___

In the above display, AMPRODSK.COM is known as the DISK7 "current file". If you press the spacebar (<SP> key) the current file indication will move to the next file (alphabetically) in directory A0. These are DISK7's commands:

- C** - Copies the current file to another disk or user area, or the same disk or user area.
- D** - Deletes the current file. You are prompted to answer **Y** or **N**. If you enter **Y**, the file will be deleted. An **N** response cancels this command.
- F** - Moves the file pointer Forward 22 files from current file name. If there are fewer than 22 files on the disk, this command will wrap around.
- G** - Group copy of all tagged files. Tagged files are indicated with a lower-case "t" next to the filename. Destination for this command is the same as for the Copy command.
- J** - Jumps to the file specified by "filename.ext". You must use the entire filename. Wild-card specifiers "?" and "*" cannot be used.

- L - Displays the Length in kilobytes used by the current file.
- N - Selects a New file directory. This can be the same or different disk or different user area.
- P - Prints a text file on the LST device (usually the system printer).
- R - Renames the current file.
- S - Displays the number of kilobytes of Space remaining in the directory of a specified drive. Enter the desired drive designator, followed by <RETURN>.
- T - Tags the current file for use with the Group copy function.
- U - Untags the current file.
- V - Allows you to View the contents of the current file. Text files will be displayed as characters on the screen. This command will not permit you to display a .COM file.
- X - EXit to CP/M.

When DISK7 requests a "DIRectory" you may specify a drive letter only, or a letter/number combination, followed by a <RETURN>. For example:

Copy to DIRectory:B<RETURN>

new DIRectory:A15<RETURN>

LDR

(Version 1)

Description

LDR is used to load ZCPR3 system segments, and your system's terminal definition file (MYTERM.Z3T, etc.). It has no options other than the name of the file you wish to load.

Operation

Uses of LDR include:

A0>> <u>LDR MYTERM.Z3T</u> <RETURN>	...loads terminal definition file
A0>> <u>LDR MYSYS.FCP</u> <RETURN>	...loads Flow Command Package
A0>> <u>LDR MYSYS.IOP</u> <RETURN>	...loads I/O Package
A0>> <u>LDR MYSYS.RCP</u> <RETURN>	...loads Resident Command Package
A0>> <u>LDR MYSYS.ENV</u> <RETURN>	...loads system ENVironment descriptor
A0>> <u>LDR MYSYS.NDR</u> <RETURN>	...loads Named Directory

As with most ZCPR3 utilities, the directory (drive/user) from which LDR is to obtain the file named in the command line can be included in several ways:

A0>> <u>LDR MYTERM.Z3T</u> <RETURN>
A0>> <u>LDR B15:MYTERM.Z3T</u> <RETURN>
A0>> <u>LDR B:MYTERM.Z3T</u> <RETURN>
A0>> <u>LDR 15:MYTERM.Z3T</u> <RETURN>

MCOPY

(Version 4)

Description

MCOPY is a powerful file copy program with many of the copy features of the CP/M PIP program. Both unambiguous (e.g. JULY.TXT) and ambiguous (J???.*) file names can be used. In addition, MCOPY can transfer files from one user to another, since the ZCPR3 drive/user spec (A0, B5, etc.) is included directly in the source and destination file specs. MCOPY verifies unless told otherwise.

Here are some features of MCOPY which are not provided by PIP:

- Easy movement from one user area to another
- Automatic verification (does not require [v])
- Multiple file copy options

On the other hand, PIP does some things MCOPY does **not** do:

- Copy to/from non-disk system devices (CON:, LST:, etc.)
- File modification during copy (i.e., PIP options: [B,D,E,F,...])
- File concatenation
- File renaming during copy

Operation

MCOPY is typically used in a manner similar to PIP. For example:

```
A0><u>MCOPY B12:=A3:*.*<RETURN>
```

copies all files from user area 3 on drive A to user area 12 on drive B. Both asterisks and question mark "wild cards" may be used to specify the names of files to be transferred. You can also transfer several files for example:

```
A0><u>MCOPY B12:=A3:FILE1,A2:FILE2<RETURN>
```

copies the two files (from the two different user areas) to B12.

Other MCOPY options are also available. They are:

E Existence test. Checks to see if a file will be over-written:
Replace -- (Y/N)?. If E not used, replaces without asking.

I Inspect files. As each file name is displayed, the message **(Y/N/S)?** is displayed, meaning:

Y - Yes, replace - copy this file

N - No, do not replace - skip this file

S - Skip all files from this one to end of file list

M Multiple copy permits copying the same group of files to several disks. Pauses after completion for change of disks.

Q Quiet toggle turns off copying messages.

V Verification off.

MENU

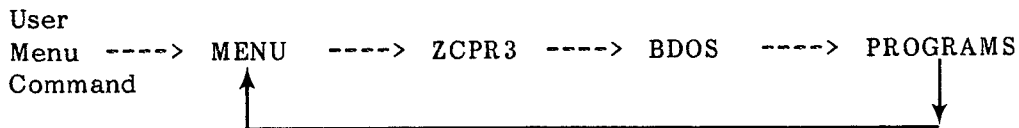
(Version 3)

Description

MENU is one of the ZCPR3 "shell" programs. The purpose of a shell is to temporarily replace or enhance the normal system command line environment. MENU replaces the "A0>" appearance of your system with a full screen display, and substitutes powerful pre-programmed single-keystroke options instead of the normal CP/M or ZCPR3 commands.

MENU is a true shell, in that it automatically reloads itself after any operations invoked from the menu are finished. The best part about MENU is that you can easily create a custom screen display with custom single-keystroke command options, according to your system needs. You can create a variety of MENU's for a variety of applications.

The following diagram shows how MENU works:



The User Menu Command is a single character that you enter, which instructs MENU to perform a function. As MENU processes the selected function, it builds a command line for ZCPR3, optionally requesting input (i.e., filename, drive/user, etc.). MENU then passes the command line to ZCPR3 through the ZCPR3 Command Line Buffer. ZCPR3 in turn passes the command to the BDOS which ultimately interacts with programs. When program execution is complete, ZCPR3 returns control to MENU.

Operation

Once a MENU is set up for a system, it is extremely easy to use. In fact, a system with MENU running is usually much easier to use than without it! As distributed, the AMPRO operating system diskette contains a typical menu, ready for use. All you have to do to run MENU is type the command

A0>MENU<RETURN>

The specific menu appearance and choices depend on the contents of the file MENU.MNU, present on the disk. Modifying the menu is not very hard, and can be done with most any word processing or text editing program. This is discussed below.

One other point: if you plan to use MENU, be sure to customize the terminal characteristics file, MYTERM.Z3T. The system disk contains a "generic" MYTERM.Z3T, which allows immediate use of MENU with any ASCII terminal having an 80 X 24 character display. However, fast screen clearing and screen highlighting are not included. Use one of the two ZCPR3 terminal character-

istics customization utilities -- TCSELECT or TCMMAKE -- to customize the MYTERM.Z3T file for your terminal.

Customizing the MENU File

The nature of each MENU environment depends on the contents of a special MENU file, called MENU.MNU. The MENU file contains up to 22 text lines to be displayed on the screen as a command menu, followed by definitions of the available single-keystroke commands. Here is a sample menu file:

```
-dpx
#
      ^A. ....
      :^B C - Copy a file      ^A:
      :^B D - Directory      ^A:
      :^B E - Erase a file   ^A:
      :^B M - Modem communications ^A:
      :^B S - Status of a disk ^A:
      :^B W - run WordStar   ^A:
      :^B Z - manual command ^A:
      :. .... : ^B
#
C!mcopy "Destination Disk? "=="Source Disk? ":"Filename.typ? "
D!dir "Directory of which Drive/User? ":
E! era "Drive/User? ":"Filename.typ? "
Mmdm740; set port a=9600,8,a,n,y
S! stat "Status of which drive? ":
Z! "Enter command > "
Wws
##
```

Display Options Line

The first line of the file begins with a hyphen. This is the display options line, in this case: -dpx. The characters may be upper or lower case. The choices are:

- d - If present, causes MENU to display the screen text which follows.
- p - If present, causes MENU to scroll the screen before display.
- x - If present, allows exit from MENU with <CTRL-C>.
- c - (Debugging option.) If present, displays command lines as they run.

When MENU is running, a prompt at the bottom of the screen indicates its presence. This prompt varies according to the display options. For example, the prompt:

Command (<CR>=Menu, ^C=ZCPR3) -

indicates that the X option is present, while the prompt

Command (<CR>=Menu) -

indicates that the X option is not present.

Screen Display Lines

The display portion begins and ends with a # character. It can be up to 22 characters in length. The contents of these lines can be any printable ASCII characters. In addition, two control characters, ^A and ^B, can be embedded in the text to control the video highlighting attribute on terminals which provide it. ^A turns highlighting on, while ^B turns it off. An editor such as Wordstar can be used to create text files with embedded control characters.

Command Lines

MENU command lines have the general format:

<command character><option><command>

where:

<Command character> - a single character used to invoke the command. Alphanumeric characters can be upper- or lower-case. The following characters may not be used as the command character:

<SPACE> # % , . < > * ...or any character less than <SPACE>

<option> - an opening option consisting of:

:nn - goto menu nn

! - have MENU wait when the command in this line is finished

<command> - a command line as it would be entered from the keyboard at the normal ZCPR3 command prompt. The line can include a prompt, within double quotes. If so, the string inside the quotes is displayed as a prompt, and the user's response to the prompt is substituted for the quote in the resulting command line.

Here are some fine points regarding the construction of command lines:

The ! Option

If the command line begins with !, MENU waits for a key to be pressed before redisplaying the MENU screen text. In this case, after the menu command completes, MENU prompts the user to "Strike Any Key". Any displayed information remains on the screen until a key is pressed. Without a ! preceding a command line, MENU will clear the screen and display the menu text. This is important in commands which produce a display which must be read, for example:

```
D!dir
```

Without the ! option, the directory would be displayed, but immediately written over by MENU's text.

User Input Prompt Option

The prompt feature allows user input to assist in building a particular command. More than one prompt string can be used in a given command line. Each prompt string is enclosed in double quotes within the command line text, for example:

```
D!dir "Drive/User? (A0,B,15,...) ":"."Type of file? "
```

When the D key is pressed, MENU will prompt with the first message (Drive/User?). After you type your response to the first prompt (followed by a <RETURN>), MENU will prompt with the contents of the second pair of quotes. Your responses to the two prompts are used to build the actual DIR command.

Command Line Variables

The menu command lines can also contain several embedded variables. The variables are defined as follows:

\$D	Current Disk
\$U	Current User
\$Fn	FILENAME.TYP for ZCPR3 System File n
\$Nn	FILENAME for ZCPR3 System File n
\$Tn	fileTYPE for ZCPR3 System File n
\$\$	place a single \$ in command line

The :nn Option

A single menu file can contain several sub-menus (up to 255!). The first menu is number 1. The others follow sequentially, each separated by a # character. The :nn option allows one sub-menu to call up the others. The example given above has only one menu. Here is a highly simplified menu file containing three sub-menus:




```

-dpx
#
                - 1st Menu -

A - Goto Menu 2      3 - Goto Menu 3

#
a:2
3:3
#
                - 2nd Menu -

T - Goto Menu 1      3 - Goto Menu 3

#
3:3
T:1
#
                - 3rd Menu -

2 - Goto Menu2

#
2:2
##

```

Built-in Commands

There are also several built-in commands recognized by MENU at its command prompt, regardless of the contents of the specific menu file. They are:

<RETURN>	Refresh Menu Display
<CTRL-C>	Exit to ZCPR3 (Only if enabled with -x option)
*	Jump to the First Menu
< (or ,)	Jump to the Previous Menu
> (or .)	Jump to the Next Menu
\$	Jump to the System Menu (Password Required)



PATH

(Version 3)

Description

PATH is a utility that allows you to temporarily alter the ZCPR3 command search path. This is especially useful in hard disk systems, when various types of programs and files are divided into separate user areas. PATH can be used to re-define the command search path so that appropriate utilities and programs are available when needed.

Operation

The use of PATH is illustrated by the following example:

```
A0>>PATH A15:B15:$0:A$<RETURN>
```

which sets the path to:

- (1) drive A, user 15
- (2) drive B, user 15
- (3) current drive, user 0
- (4) drive A, current user

The dollar signs stand for "current." The path actually always starts with current drive, current user; this never changes, and need not be included in the PATH command.

To find out what the current path is, use the PATH command, but without any parameters:

```
A0>>PATH<RETURN>
```



TCMAKE

(Version 1)

Description

One of the most powerful features of ZCPR3 is its terminal characteristics definition facility, or "termcap." Programs designed to take advantage of ZCPR3's termcap feature do not require installation for individual terminals, once the ZCPR3 termcap is installed for the particular terminal in use. Instead of installing each program you run, you simply create a termcap file for your terminal; then, all of the programs designed to use ZCPR3's termcap can be run without any installation. The ZCPR3 MENU, VFILER, VMENU, and AMPRO FRIENDLY programs all utilize this feature.

If your terminal is one of the "standard" terminals supported by the TCSELECT utility, then creating your termcap file is simply a matter of selecting the terminal you are using from one of TCSELECT's menu screens. If your terminal is not included in TCSELECT, you must create a custom termcap file using TCMAKE instead.

Operation

These are the terminal functions defined through TCMAKE:

1. Clear Screen Sequences - Byte sequence to clear the screen of your terminal. You can enter from 1 through 255 bytes in this sequence.
2. Cursor Motion Sequence - Determines whether your terminal is Row/Column or Column/Row addressing, requests data for row and column addressing, permits setting three sequences, each up to 255 bytes long -- prefix, middle, and suffix bytes.
3. Clear to end of line sequence - Sequence of bytes that clear the line from the cursor position to the end of the line. This, too can be a sequence of up to 255 bytes.
4. Standout Mode Sequences - Two sequences, standout mode on and standout mode off. Each can be up to 255 bytes in length.
5. Terminal Init/Deinit Sequences - Two sequences, terminal initialization for entry to FRIENDLY, and a de-initialization sequence sent to the terminal when you eXit FRIENDLY. Each can be up to 255 bytes long.
6. Arrow Keys - Permits setting a single-byte character for each keyboard Arrow key. If your terminal outputs a single character when each Arrow key is pressed, simply press each arrow key as requested.
7. Terminal Name - A text string to identify the terminal being installed.

TCMAKE prompts you for what it needs. In answering TCMAKE's questions, you can respond by pressing the keyboard key that corresponds to the value for a byte. For example, pressing an ASCII "A" will store the corresponding character value (41H) for that key. However, it is usually easiest to enter values as a hexadecimal (00H - FFH), or decimal number (0-255). TCMAKE allows you to do this as follows: when the program requests an input, first press the period (.) key, and then enter the value to be stored. For example:

Char #1 - Type Char,.=Number,or <CR>=Done: EnterNumber: 1BH<RETURN>

A period was pressed, followed by 1BH<RETURN>. The hex value "1BH" is the ESC key code, which is the same as 27 decimal. Hexadecimal values must end with "h" or "H". In this case, pressing the ESC key, without pressing a period, would have had the same effect as entering 1BH.

NOTE: Due to space limitations, not all strings can simultaneously be 255 bytes long in the same terminal definition file. It is rarely necessary to have long string sequences, anyway.

Cursor Addressing

Terminal direct-cursor addressing is accomplished in many different ways, depending upon the particular terminal. In each case, the computer sends a string or sequence of characters to the terminal. Usually, there is a lead-in sequence (Prefix Byte sequence), a Row address, and a Column address contained in this string of characters. There may also be a string preceding the Row and Column addresses (Middle Byte sequence), and an ending string (Suffix Byte sequence).

Following are some symbols used with TCMAKE to define the cursor addressing string. The majority of terminals use one of two different sequences. The symbols below must be preceded with a % symbol when used as inputs to TCMAKE. For example:

- d** - Output Row and Column data as decimal with no leading spaces or zeros (e.g., 1, 12, or 79).
- 2** - Output Row and Column data as 2 decimal digits, right-justified (e.g., 01, 09, or 79).
- 3** - Output Row and Column data as 3 decimal digits, right-justified, (e.g., 001, 009, or 131).
- .** - The period (.) outputs the Row and Column data as binary values.
- +[n]** - Add 'n' to Row and Column values. Many terminals use the SPACE character value (20 hex) for 'n'.
-][xy]** - If the value of Row or Column is greater than "x", then add "y" to the Row or Column value.

r - Reverse the order of output sequence to: Column-Row, instead of Row-Column. Row-Column address order also can be determined with the second question of the Cursor Addressing installation process. This merely reverses the order selected. Default order is Row-Column.

i - Set the Row/Column home position to 1,1 rather than 0,0. (The default home position is 0,0.)

Here are two examples of entering the cursor addressing data using TCMMAKE. The first example is for the TeleVideo 925. To install the Heath/Zenith H/Z19, substitute "Y" in place of "=". An upper case "Y" must be used. This is the cursor addressing sequence:

Prefix Byte Sequence: ESC=
Middle Byte Sequence: none
Suffix Byte Sequence: none
Row/Column Order: Row-Column
Row Equation: 20H + Row value (ASCII character)
Column Equation: 20H + Column value (ASCII character)

The TeleVideo 925 cursor addressing sequence is:

ESC=%+<SPACE>%+<SPACE>

The following example shows how these values are entered with TCMMAKE. User inputs are shown underlined.

Cursor Motion Definition

1. Timing Delay:
Enter Delay Time in Milliseconds: <RETURN> <-- Zero delay
2. Enter R if Row/Column or C for Column/Row: R <-- Row/Column addressing
3. Enter Equation for Row: %+ <RETURN> (Note the space after "+")
4. Enter Equation for Column: %+ <RETURN> (Note the space after "+")
5. Enter Prefix Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: Enter Number: 1bh<RETURN>
Char #2 - Type Char, .=Number, or <CR>=Done: Char =<RETURN>
Char #3 - Type Char, .=Number, or <CR>=Done: <RETURN>
6. Enter Middle Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: <RETURN>
7. Enter Suffix Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: <RETURN>

Note that only the Prefix Byte Sequence is required in this example.

The next example is for an ANSI terminal such as the DEC VT52, or the Heath/Zenith H/Z19 in the ANSI mode. The required cursor addressing sequence is:

ESC[%i%d;%dH

Here is how this information is given to TCMMAKE:

Cursor Motion Definition

1. Timing Delay
Enter Delay Time in Milliseconds: <RETURN> <-- Zero delay
2. Enter R if Row/Column or C for Column/Row: R <-- Row/Column addressing
3. Enter Equation for Row: %d<RETURN>
4. Enter Equation for Column: %d<RETURN>
5. Enter Prefix Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: Enter Number: 1bh<RETURN>
Char #2 - Type Char, .=Number, or <CR>=Done: Char [<RETURN>]
Char #3 - Type Char, .=Number, or <CR>=Done: %<RETURN>
Char #4 - Type Char, .=Number, or <CR>=Done: i<RETURN>
Char #5 - Type Char, .=Number, or <CR>=Done: <RETURN>
6. Enter Middle Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: ;<RETURN>
Char #2 - Type Char, .=Number, or <CR>=Done: <RETURN>
7. Enter Suffix Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: H<RETURN>
Char #2 - Type Char, .=Number, or <CR>=Done: <RETURN>

The Row/Column descriptors are constructed from information in a given terminal's technical manual. You need to know:

1. The sequence, if any, preceding, between, and following the Row/Column or Column/Row addressing equations,
2. Whether Row/Column or Column/Row addressing is used.
3. How the Row and Column addressing is implemented on the specific terminal.

A Typical Session with TCMAKE

The following is a terminal dialog with TCMAKE to install a TeleVideo Model 925 terminal. Some possible items were not installed, because some sequences were not needed or wanted.

Run TCMAKE by typing,

```
A0>TCMAKE_MYTERM<RETURN>
```

This will create a termcap file called MYTERM.Z3T when you exit TCMAKE. You can use another name for the file, but the extent should be left off, as the program will add the Z3T part.

The following will be displayed. In the interest of keeping it short, the automatic redisplay of the menu after each parameter entry is not shown.

**** Z3TCAP Main Menu for File MYTERM.Z3T ****

- Define:
1. Clear Screen Sequence
 2. Cursor Motion Sequence
 3. Clear to End of Line Sequence
 4. Standout Mode Sequences
 5. Terminal Init/Deinit Sequences
 6. Arrow Keys
 7. Terminal Name

Status: S. Print Status (Definitions so far) <-- This lets you see what has been entered to this point.

Exit: X. Exit and Write File <-- Leave TCMAKE.COM and write a file with the terminal definitions you have just entered!

Q. Quit and Abort Program without Writing File <-- Forget the whole thing, and quit.

Command? 1

Clear Screen Definition

1. Timing Delay

Enter Delay Time in Milliseconds: <RETURN> <-- RETURN only for zero

2. Clear Screen Byte Sequence

Char #1 - Type Char, .=Number, or <CR>=Done: Enter Number: 1bh<RETURN>

Char #2 - Type Char, .=Number, or <CR>=Done: Enter Number: 2bh<RETURN>

Char #3 - Type Char, .=Number, or <CR>=Done: <RETURN>

... This byte sequence is two characters: "ESC" and "+" (1BH and 2BH).

... You can also check what you have already done. Type an "S", then select from the resulting menu. Let's look at what has been set for the Cursor Motion Definition.

Command? S

**** Z3TCAP Status for File MYTERM .Z3T ****

- Review:
1. Clear Screen Definition
 2. Cursor Motion Definition
 3. Clear to End of Line Definition
 4. Standout Mode Definition
 5. Terminal Init/Deinit Definition
 6. Arrow Key Definition
 7. Terminal Name Definition

Exit: X. Exit to Main Menu

Command? 2

Review of Cursor Motion Data

1. Timing Delay = 0 Milliseconds
2. Row or Column First: R
3. Row Equation: -->%+ <--
4. Column Equation: -->%+ <--
5. Prefix Byte Sequence:
(1) [^][1BH (2) = 3DH
6. Middle Byte Sequence:
-- Empty --
7. Suffix Byte Sequence:
-- Empty --

Strike Any Key to Continue - <RETURN>

Command? X <----- This gets you back to the program menu.

... Now we'll set the sequence that clears the screen from the cursor position to the end of a line:

Command? 3

Clear to End of Line Definition

1. Timing Delay
Enter Delay Time in Milliseconds: <RETURN>
2. Clear to End of Line Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: Enter Number: 1bh<RETURN>
Char #2 - Type Char, .=Number, or <CR>=Done: Char t<RETURN>
Char #3 - Type Char, .=Number, or <CR>=Done: <RETURN>

... The byte sequence ESCt will be sent whenever there is a need to clear the screen to the end of a line.

... Next, we'll set the terminal Initialization & Deinitialization sequences:

Command? 5

Terminal Init/Deinit Definition

1. Terminal Initialization Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: Enter Number: 1bh<RETURN>
Char #2 - Type Char, .=Number, or <CR>=Done: Enter Number: 2eh<RETURN>
Char #3 - Type Char, .=Number, or <CR>=Done: Enter Number: 31h<RETURN>
Char #4 - Type Char, .=Number, or <CR>=Done: <RETURN>
2. Terminal Deinitialization Byte Sequence
Char #1 - Type Char, .=Number, or <CR>=Done: <RETURN>

... The Initialization sequence is ESC.1, which sets the screen cursor to a blinking rectangle. The Deinitialization sequence was left undefined.

... Next, we'll enter the terminal name as any alphanumerical text string, terminated by a <RETURN>.

Command? 7

Input Name of Terminal: tv925<RETURN>

Command? X <----- This exits TCMAKE and writes the terminal definition
file.

Selected Terminal is: tv925 -- Confirm (Y/N)? Y

File MYTERM .Z3T Created

The terminal definition file MYTERM.Z3T just created by TCMAKE will provide satisfactory operation with the TeleVideo Model 925 CRT terminal. The actual filename is strictly a matter of personal preference; the filetype must be Z3T.



TCSELECT

(Version 1)

Description

One of the most powerful features of ZCPR3 is its terminal characteristics definition facility, or "termcap." Programs designed to take advantage of ZCPR3's termcap feature do not require installation for individual terminals, once the ZCPR3 termcap is installed for the particular terminal in use. Instead of installing each program you run, you simply create a custom termcap file for your terminal; then, all of the programs designed to use that feature of ZCPR3 can be run without any installation. The ZCPR3 MENU, VFILER, VMENU, and the AMPRO FRIENDLY programs all utilize this feature.

If your terminal is one of the "standard" terminals supported by the TCSELECT utility, then creating your termcap file is simply a matter of selecting the terminal you are using from one of TCSELECT's menu screens. If your terminal is unlisted, you can still create a custom termcap file, but will need to use the ZCPR3 TCMMAKE utility instead.

Operation

TCSELECT is very easy to use. Before you run TCSELECT, be sure that the disk directory containing TCSELECT also contains a file called Z3TCAP.TCP; this file contains the terminal data required by TCSELECT. Run the program like this:

```
A0><u>TCSELECT MYTERM</u><RETURN>
```

The program will prompt you with instructions. Select the desired terminal from one of the available menus. If your terminal is not in any of the menus, you will need to use the ZCPR3 TCMMAKE program instead.

This creates a file called MYTERM.Z3T, which holds the information regarding your terminal. You can use any name for the file, except that it must have the "Z3T" extent. Before you can use the termcap you have created, it must be loaded into memory by the ZCPR3 LDR utility. This is done with the command:

```
A0><u>LDR MYTERM.Z3T</u><RETURN>
```

Normally this is made part of the system startup aliases.

Another option available through TCSELECT is that you can run the program without specifying a filename on the command name, i.e.

```
A0><u>TCSELECT</u><RETURN>
```

In this case, no file is created when you exit TCSELECT; instead, the selected terminal definition is loaded directly into memory. This is helpful for testing terminal selections.

UNERASE

(Version 1)

Description

UNERASE permits recovery of accidentally deleted files. This program will work reliably only if no write operations have been performed since the file or files were accidentally deleted. ANY write operation can destroy the information used by UNERASE to restore deleted files.

Using UNERASE

UNERASE is easy to use. To restore a specific file, type:

```
A0><u>UNERASE FILENAME.TYP</u><RETURN>
```

You can also restore more than one file at a time. For example:

```
A0><u>UNERASE B5:*.BAK</u> --> Recovers all .BAK type files in user area  
5 of drive B
```

```
A0><u>UNERASE C:MYFILE.*</u> --> Recovers all MYFILE.TYP files, including  
MYFILE, without .TYP specifier
```

For a List of erased files, type:

```
A0><u>UNERASE *.* L</u><RETURN>
```

This will display a list of files that can be unerased, without actually performing the unerase function.

NOTE

In some cases there may be two "erased" files with the same name. Then, UNERASE may restore both versions of the file. It will indicate this has happened by listing the same filename twice when it indicates what it has done. If this occurs, erase the files again with the ERA command immediately.

Always check restored files for integrity!

UNERASE can sometimes "save the day" -- but use it at your own risk!

Z3INS

(Version 1)

Description

Z3INS is the ZCPR3 utility installer. All ZCPR3 utilities supplied on the AMPRO operating system software diskettes are pre-installed. ZCPR3 utilities which you obtain from a user group, bulletin board system, etc., will probably require installation. This is easy to do using Z3INS.

Operation

There are two ways you may use Z3INS to install ZCPR3 programs: you can install a single program, by name; and you can install a group of programs, based on a text file containing their names. In both cases, Z3INS requires the presence of a ZCPR3 "environment descriptor" file. The AMPRO Version 3 BIOS contains a built-in environment descriptor. However, for the purposes of ZCPR3 program installation, a duplicate of the built-in AMPRO environment is contained in the file called ZAMPRO0.ENV, present on the Z80 System Software diskette. This is the file to use with Z3INS.

The most common way to use Z3INS, is to install a single program, by name, as follows:

```
A0>Z3INS ZAMPRO0.ENV MENU.COM<RETURN>
```

Installation of a group of programs is similar, except you need to first create a text file which contains the names of the programs to be installed, and the file should have a file name ending with "INS" -- for example, the following file might be called Z3UTILS.INS:

```
MENU.COM  
MCPY.COM  
PATH.COM  
VFILER.COM  
VMENU.COM
```

Then, to install the five programs in the list, the command is:

```
A0>Z3INS ZAMPRO0.ENV Z3UTILS.INS<RETURN>
```


ZEX

(Version 3)

Description

ZEX is ZCPR3's powerful batch processing facility. In some ways, ZEX is like a combination of CP/M's SUBMIT and XSUB. However, Unlike SUBMIT, ZEX is memory-based. Its input source is located in memory, so that its execution speed is significantly greater. Also, ZEX obtains its command batch from a text file (ending in .ZEX or .SUB) anywhere along the ZCPR3 command search path.

Like XSUB, ZEX intercepts all calls to the BIOS Console Input (and Input Status) routine and provides an input character in its place. There are exceptions to this case, but they will be discussed later.

Operation

ZEX is normally used in much the same manner as the CP/M SUBMIT program. For example:

```
A0><u>ZEX ASSEMBLE</u><RETURN>
```

In this case, a command text file called ASSEMBLE.ZEX contains a sequence of command lines to be executed in batch mode. The command text file can also be called ASSEMBLE.SUB. The command file can be located anywhere along the ZCPR3 command search path.

During ZEX operation, a <CTRL-C> from the console will abort execution of the ZEX command stream. Also, if a command follows ZEX in a Multiple Command Line, ZEX appends this command to the command stream.

ZEX command files may not be nested. ZEX will simply abort if a ZEX command is encountered in the command stream it is processing.

ZEX supports many embedded commands. Combining the facilities of SUBMIT and XSUB in this case, the embedded commands of ZEX reflect the XSUB-like capabilities of ZEX as well as some new ideas.

Like SUBMIT, ZEX can be given parameters on the command line, for example:

```
A0><u>ZEX CMDFILE FILE1 FILE2</u><RETURN>
```

In this case, "FILE1" is substituted for \$1 and "FILE2" for \$2 in the command text file.

A summary of the control commands which may be placed in a ZEX command text file is displayed if you type the command

```
A0><u>ZEX //</u><RETURN>
```


Summary of ZEX Control Commands

Cmd	Meaning	Cmd	Meaning
	insert a <CR>	^	insert <CR><LF>
^:	rerun command file	^.	suppress of characters
^#	toggle ZEX messages	^\$	define default parameters
^?	wait for return key	^/	ring bell; wait for return key
^*	ring bell	^"	allow user input
^<	display chars only	^>	stop display chars
;;	ZEX comment	\$n	parameters (1-9)
\$\$	insert a \$	^\$	insert a
\$	insert a	^c	insert a control character, "c"

The `^*` command simply causes ZEX to ring the bell. It does not insert a BELL character into the command file like a `^G` sequence would. It simply rings the bell and continues processing.

The `;;` command is a ZEX comment. It and all characters following it up to and including the following `<LF>` are not included in the ZEX command stream. They are simply treated as a comment in the ZEX Command File and ignored. Unlike a conventional ZCPR3 comment, the ZEX comment does not take up space in the command stream and does not appear when the command stream is executed.

The `^<` and `^>` commands are used to bracket characters which are simply echoed by the ZEX monitor and not passed back to the calling program. This causes the characters between these commands to be echoed to the system console during execution but not processed by any program. This feature is very good for embedding strings to be displayed at execution time into the command stream. Unlike the ZCPR3 comment form, which is a line beginning with a semicolon, comments enclosed by `^<` and `^>` may appear anywhere, such as within an editor session.

The `^#` command toggles suppression of informative messages generated by ZEX.

The `^.` command causes console output to cease until the next `^.` is encountered. Character input from the ZEX Monitor continues, but the user does not see what it is.

`$n`, where `n` is 1-9, will cause the indicated specified or default parameter to be substituted from the command line.

The `^$` command is used to define or redefine the set of input command parameters. The rest of the line following the `^$` is treated as a set of parameters separated by blanks.

`^?` causes ZEX to stop processing and wait for the user to strike either the space bar or the RETURN key before continuing. The user can take his time and examine the display, and, if he does not wish to continue, a `^C` will abort the command stream. The `^/` command is like `^?`, but it periodically rings the bell at the console, summoning the user in an alarm fashion.

Other ZEX options are available. For further information refer to the ZCPR3 documentation referenced elsewhere in this manual.



CHAPTER 6

PUBLIC DOMAIN PROGRAMS

6.1 INTRODUCTION

The programs listed in this chapter are "public domain" programs which are available for unrestricted free distribution. These programs have been included for your convenience on the the AMPRO operating system software diskette. Each program's description explains what the program does and how it is used. The utilities are covered in alphabetical order, so this material can serve as a handy operator's reference.

A large number of additional public domain programs are available through the many CP/M user groups and bulletin board systems.

6.2 PROGRAM DESCRIPTIONS

The following pages contain the program descriptions of the public domain programs included on the AMPRO Z80 system software diskette, alphabetically arranged.

MDM740

(Program Version 7 - AMPRO Overlay Version 2)

Description

MDM740 is one of the most popular public domain programs. It is a general purpose, powerful communication program, capable of both computer-to-computer and modem operation. It includes the popular "XMODEM" protocol, for error free data transfer, and also can support simple ASCII text transfer.

Supplied on AMPRO system disk are three files relating to MDM740:

1. **MDM740.COM:** the communications program itself.
2. **M7LIB.COM:** a utility program to customize the MDM740 phone list.
3. **M7LB.ASM:** the AMPRO-specific overlay source code.

As supplied on your disk, MDM740 is configured for auto-dialing with a Hayes Smartmodem, or Hayes-compatible, modem. If you have a modem which does not support auto-dialing, or which is not Hayes compatible, you can still use the program -- but with manual phone dialing.

NOTE

Modem control overlays to support other types of modems are available through CP/M user groups and bulletin boards. You can also modify the AMPRO overlay source file, M7LB.ASM, to make the program work with a different type of modem. (Instructions in how to perform this optional customization are contained in the file itself.)

Operation

Since MDM740 is such a powerful program, it naturally has quite a few commands and options. The discussion here will not attempt to explain all operations, but will hopefully help you get started. Contact a local CP/M user group if you need help.

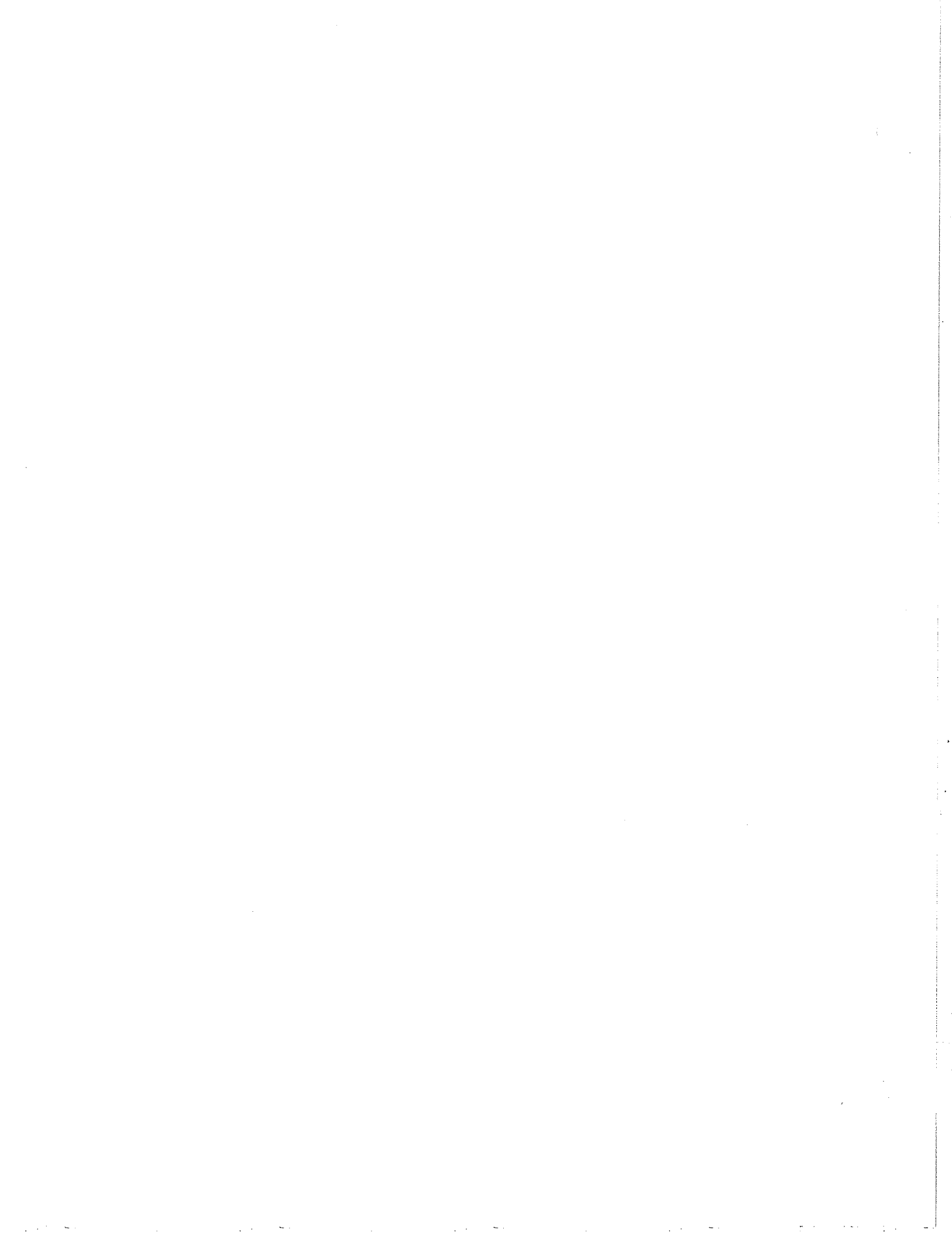
There are two basic modes of operation:

Command - allows control of the modem program functions and parameters.

Terminal - allows your terminal to communicate with a remote computer.

To get started, type the program's name from the CP/M command line:

```
A0><u>MDM740</u><RETURN>
```



When you run MDM740, a message similar to the following will be displayed:

```
MDM740 modem pgm (type M for Menu)
Copyright (c) 1984 - Irvin M. Hoff
Initial baud rate set for 300
```

A>>COMMAND:

This is MDM740's command line (like CP/M's "A>"). From this command line you can dial a number, enter terminal mode, or modify the current communications setup. You can tell you are in the Command mode when you see the prompt:

A>>COMMAND:

MDM740 has three types of commands:

Single Letter Commands - Used in command mode to send or receive an XMODEM file, enter terminal mode, display help information, etc.

Three Letter Commands - used in command mode to set operating parameters, dial a number, exit to CP/M, etc.

Terminal Mode Commands - commands used only in the terminal mode, to turn the text buffer on and off, send or receive a text file, etc.

To display a menu of all command options, type:

A>>COMMAND:M<RETURN>

After you enter the M command, the program will display three screens of help information, representing the three groups of available commands:

Single Letter Commands

- ? - Display current settings
 - Function key intercept character, then (0-9)
- M - Display the menu
- E - Terminal mode with echo
- L - Terminal mode with local echo
- T - Terminal mode
 - For copying text to disk use T (E or L) FILENAME.TYP
 - Start or Stop toggles described on subsequent screen.
- R - Receive CP/M file using Christensen Protocol
- S - Send CP/M file using Christensen Protocol
 - A>>COMMAND: R (or S) FILENAME.TYP
 - R and S can use the following subcommands:
 - B - Bulk transfer using wildcards (e.g., *.*)
 - D - Disconnect when done
 - Q - Quiet mode (no messages to console)
 - V - View <R> or <S> bytes on console
 - X - When done, disconnect, go to CP/M

The single letter commands may also be used on the command line when the program is initially executed.

Three Letter Commands

CPM -Exit from this program to CP/M
DIR - List directory and space free (may specify drive)
ERA -Erase file (may specify drive)
LOG -Change default drive/user no. (specify drive/user)
and reset disks. e.g. LOG A0: or LOG B:
SPD - Set file output speed in terminal mode
TCC -Toggle CRC/Checksum mode on receive
TLC -Toggle local command immediate or after CTL-
TLF - Toggle LF after CR in "L" or "T" mode for a disk file
TRB -Toggle rubout to backspace conversion
TXO -Toggle XOFF testing in terminal mode file output
SET -Set modem baud rate
BYE - Disconnect, then return to CP/M
CAL -Dial number
DSC -Disconnect from the phone line

The following are terminal text buffer commands:

DEL -Delete memory buffer and file
WRT -Write memory buffer to disk file

Local Commands while in Terminal Mode

CTL-@ -Send a break tone for 300 ms.
CTL-E -Exit to command mode
CTL-L -Send log-on message
CTL-N -Disconnect from the phone line
CTL-P -Toggle printer
CTL-Y -Start copy into buffer
CTL-R -Stop copy into buffer

Start & Stop may be toggled as often as desired.
A ";" at start of line indicates buffer is copying.
XOFF automatically used to stop input when writing
full buffer to disk, XON sent to resume.

CTL-T -Transfer ASCII file to remote
CTL-^ -Send local control character to remote

Using a Modem

A really nice feature of MDM740 is auto-dialing. With one simple command, you can access up to 36 predefined telephone numbers from MDM740's internal phone list. It is an easy matter to use the phone list customization program, M7LIB.COM, to enter the names and numbers you wish to use. To use M7LIB, type the following command from the CP/M command line (not from MDM740's command line):

A0>M7LIB MDM740.COM<RETURN>

M7LIB will guide you through the phone list modification process.

Using the AMPRO Bulletin Board System

The following example illustrates how you might log into the AMPRO user's bulletin board system. If you have a Hayes compatible modem, you can use the auto-dialing command, as follows

```
A>>COMMAND: CAL A<RETURN>
```

This assumes that the AMPRO BBS is programmed into your phone list as entry **A**. The modem will set the telephone "off-hook", dial the number, then wait for the remote bulletin board system "answer".

If you do not have a Hayes-compatible auto-dial modem, then simply dial the number of the AMPRO bbs -- (408) 258-8128 -- manually on your telephone.

When the remote device answers, you will hear a high-pitched "carrier" tone if your modem has a speaker. When auto-dialing, MDM740 automatically enters the Terminal mode when it senses the remote modem's carrier tone. Your screen will look like this:

```
A=AMPRO BBS.....408-258-8128 - try #1
```

```
CONNECTED
```

```
(in Terminal-mode now)
```

If you dial manually, you must use the **T** command to enter the Terminal mode, when you notice that the remote system has answered. You will know the remote system has answered by a high pitched tone (if your modem has a speaker, or through the phone earpiece), or if your modem's "carrier detect" light goes on.

```
A>>COMMAND: T<RETURN>
```

Once connected, you type whatever log-in sequence may be required. In the case of the AMPRO BBS, press the **RETURN** key one or two times. The AMPRO bbs will send a message:

```
How many nulls does your terminal require? (0-7)
```

Responding with **0** (zero) for "nulls required" is usually sufficient. The bulletin board system will send some messages and instructions, then prompt you for what to do next. The main BBS commands are:

- S - Summary of messages
- R - Read a message
- E - Enter a message
- G - Goodbye (leave BBS)
- C - Enter remote CP/M
- ? - Display command help

The G command is what you use when you are finished and ready to disconnect from the BBS. If you enter CP/M (C command), here are some common commands:

A0><u>USER 5</u><RETURN> ...to change user areas
A0><u>HELP</u><RETURN> ...for remote CP/M help info
A0><u>SYSMAP</u><RETURN> ...to display a map of program locations
A0><u>BYE</u><RETURN> ...to disconnect from BBS

Downloading and Uploading BBS Files

Downloading and uploading are done from the BBS remote CP/M command line (e.g., "A0>") Downloading a file means that you have the BBS send you a file. Uploading is when you send a file to the BBS.

Here is how you download a file from one of the BBS CP/M directories:

1. At the BBS remote CP/M command line, type:

A0><u>XMODEM S FILENAME.TYP</u><RETURN>

2. Wait for XMODEM to indicate its readiness for sending the file
3. Use <CTRL-E> to exit to MDM740's Command mode
4. At MDM740's command line, type:

A0><u>COMMAND:R FILENAME.TYP</u><RETURN>

Here is how to upload a file (from your system to the BBS):

1. At the BBS remote CP/M command line, type:

A0><u>XMODEM R FILENAME.TYP</u><RETURN>

2. Wait for XMODEM to indicate its readiness for receiving the file
3. Use <CTRL-E> to exit to MDM740's Command mode
4. At MDM740's command line, type:

A0><u>COMMAND:S FILENAME.TYP</u><RETURN>

Capturing ASCII Data

Another nice feature of MDM740 is the capability to "capture" data into a 16 kilobyte memory buffer. Try this:

1. Dial and connect as described earlier.



2. When you are connected and in the Terminal mode, enter <CTRL-E>
3. At the MDM740 command line type:

A>>COMMAND:T FILENAME.TYP<RETURN>

This returns you to Terminal mode, but with a memory buffer for storing text.

4. Now, when you wish to capture a particular set of information, enter a <CTRL-Y>.
5. Any characters displayed on your terminal screen will be stored in the memory buffer.
6. To close the memory buffer, type a <CTRL-R>.

This sequence of <CTRL-Y> and <CTRL-R> can be used at any time, in the Terminal mode, to capture any data you wish. If the buffer becomes filled, MDM740 will send a code to the remote system asking it to wait, while it saves the current contents of the buffer to disk. Each time you return to the Command mode (with <CTRL-E>), the available buffer space is displayed on your screen. To save the buffer contents to disk, return to the Command mode, and type:

A>>COMMAND:WRT<RETURN>

WARNING

If you exit MDM740 without saving the buffer on disk, the buffer data **will be lost!**

Sending ASCII Data

You can also send ASCII text data from a file in your system. When you do this, it appears to the remote system as though you are typing the text manually. Here is how it is done:

1. Dial and connect as described earlier.
2. When you are connected and in the Terminal mode, to send a text file, type <CTRL-T>. MDM740 will prompt you for the name of the file you wish to send, and whether you want delays after each line (some remote systems need this).
3. When the file has been sent, you will be back in normal Terminal mode.

SD

(AMPRO Version 2)

Description

This is the public domain "super directory" utility. It is an alternative to the ZCPR3 DIR utility, and provides some additional capabilities. Files can be specified as in standard CP/M: those in which you use the complete file names (called unambiguous filenames), and those in which you use only part of the filename (called ambiguous), with asterisks or question marks used to represent "wild cards." For instance, AMP* would call up the files AMPle, AMPlify, AMPro, and AMPs; and A???? would call up the files AMPRO, APPLE, ADAMS.

These examples illustrate the options you can use:

A0> <u>SD<RETURN></u>	Display all files.
A0> <u>SD *.ASM<RETURN></u>	Display all .ASM files.
A0> <u>SD *.* \$A<RETURN></u>	Display all files in all user Areas of current drive.
A0> <u>SD *.* \$D<RETURN></u>	Display all files on all Drives in current user area.
A0> <u>SD *.* \$U15<RETURN></u>	Displays the directory of user area 15 only.
A0> <u>SD *.* \$P<RETURN></u>	Send directory to LST: device (Printer)
A0> <u>SD *.* \$F<RETURN></u>	Create a File containing directory contents (The file will always be named SD.DIR).

In addition, the \$-suffixes can be combined to provide further usefulness, e.g.:

A0>SD *.* \$ADFP<RETURN> Create a file and a listing of all files.

One note: since the D option causes all drives to be searched, nonexistent drives may cause your system to hang up.

SWAPCOPY

(AMPRO Version 2)

Description

SWAPCOPY allows you to transfer files from one diskette to another using only one floppy disk drive. You can copy from A to A (both AMPRO format), or from A to E, or E to A (AMPRO to/from non-AMPRO format). The destination disk must already be formatted.

When you use SWAPCOPY, you must provide a file specification similar to that used with DIR.COM. The file specification can either be ambiguous (using the ? and * characters) or not. In addition, SWAPCOPY allows a few optional command line parameters which result in various display options. These are described below.

Operation

To copy an entire disk using a single drive (it must be A), type:

```
A0><u>SWAPCOPY *.*</u><RETURN>          ...copies all files
```

SWAPCOPY first prompts you to select A to A, A to E, or E to A. The second and third of these options allow you to copy to and from non-AMPRO formats.

SWAPCOPY next prompts you to insert the SOURCE disk, then press <RETURN>. The program then reads as much of the source file(s) into memory as possible.

SWAPCOPY then prompts you to insert the DESTINATION disk. You now remove the source disk and insert your destination disk.

CAUTION

Be sure to not accidentally mix the two disks! If you do, you may destroy the SOURCE disk. You may wish to use a write protect tape on your SOURCE disk to protect it in case you accidentally mix it up with the DESTINATION disk.

With your DESTINATION disk in the drive (double check), press <RETURN>. The program will now write the file(s) stored in memory onto the destination disk. This process of exchanging disks will continue until all files are copied. When the program is finished copying, it will prompt you for the SYSTEM disk.

These examples show some of the options you may use in specifying filenames:

```
A0><u>SWAPCOPY FILENAME.TYP</u><RETURN>          ...copies one file  
A0><u>SWAPCOPY *.COM</u><RETURN>                ...copies all COM files
```

For a help screen showing other SWAPCOPY options, type:

```
A0><u>SWAPCOPY</u><RETURN>
```


ed set of mnemonics are simply turned on, resulting in no speed degradation due to the loading of a macro library and macro generation.

SLRMAC is another notable product from SLR Systems. It not only does the job that it's advertised to do but it is also free of the many petty annoyances and inaccuracies that plague so many similar products. Its many advanced features, ease of use, and blinding speed make it a must for anybody in need of an Intel mnemonic assembler.

Your Attention Please

Before progressing further with the discussion of the Ampro Little Board computer begun last month, a few words on Z80 interrupts are in order. The Z80 and its family of related peripheral support chips have an excellent built-in interrupt system. Each support chip that has more than one active element has its own internal interrupt priority scheduler and the capability to be daisy-chained into a system. This structure does not require assistance from interrupt priority schedulers in all but the biggest systems.

The main controlling element in the system, the Z80 CPU, can handle an interrupt in one of three ways depending on the setting of the interrupt mode. This fact permits the software to determine, to a large degree, how complex the interrupt system will be.

The most basic interrupt mode, mode 0, operates identically to that of the 8080 interrupt response mode and was included to provide full compatibility with that earlier and less powerful CPU. This mode is set by executing the IM 0 instruction. With this mode, the interrupting device can place any instruction on the data buss and the CPU will execute it. Thus, the interrupting device, not the memory, supplies the next instruction to be executed.

Mode 1, when enabled by the IM 1 instruction, causes the CPU to execute a restart to memory location 38h in response to an interrupt. This action is identical to that of the non-maskable interrupt except the call location in memory is 38h instead of

66h.

Finally, mode 2, the most powerful interrupt mode, combines three system elements to resolve the effective memory address loaded into the program counter. With a single 8-bit value supplied by the interrupting peripheral device an indirect call can be made to any memory location.

With this mode the CPU forms a 16-bit memory address. The upper eight bits are supplied from the Z80 interrupt register, and the lower eight bits are supplied by the highest priority interrupting peripheral device. This 16-bit address is used as an indirect memory address pointing to a stored table of pointer words that point to the processing routine responsible for servicing the interrupt. Actually, only seven bits are required from the interrupting device because bit 0 is always zero. This structure might seem overly complex until you consider that through software the service routine tables can be changed at will, providing the ultimate in flexibility.

Interrupts and the Little Board

The Ampro Little Board computer satisfied all the requirements I set for my RBBS system except the need for a real-time clock. My first impulse was to attach one through a serial port, but I soon remembered that both ports were busy; one with the CRT and the other with the modem. The lack of any unused I/O ports stymied any chance I had of using an external clock. The only option open to me was to look to the Little Board for a solution.

Digging into the schematics of the Little Board revealed a fairly straightforward solution to my problem. The engineers responsible for designing the Little Board used the Z80 and its related support chips exclusively to minimize interface logic requirements and arranged them to allow full use of their daisy-chained interrupt capabilities. So it seemed that all I needed was to find an unused timer, or some other source of regular interrupts, and a little software to implement a real-time clock.

Fortunately, my search was a short one. Channel 2 of the Clock Timer

Circuit (CTC), used for generating the baud rates for the serial ports, was left totally unused. All that was needed to start it working as a real-time clock was the few lines of code of Listing One (page 120).

My first task was to define a table of pointer values (section 2 in the listing) addressing the interrupt processing routines. The address of the first entry of this table would later be loaded into the interrupt vector register. Following this table are the processing routines. I chose to place the clock values in low memory locations: 08h hours, 09h minutes, 0Ah seconds, and 0Bh tenths of seconds.

Next, I had to insure that the constant flow of interrupts from the CTC would not adversely affect any other time-sensitive system component such as the floppy disk controller. The only positive way to do this was to disable the interrupts (sections 3 through 6 in the listing) prior to entering the data transfer routines. In practice, this proved to be absolutely necessary because of excessive disk errors when running with interrupts enabled.

The side effect of this, however, is to destroy the clock's accuracy. Since it's unknown exactly how long the clock will be stopped during a disk data transfer, it's impractical to add in a fudge factor upon exiting the disk routines. By comparing the computer clock against a stopwatch I found that during a highly disk intensive period the computer clock would run approximately 50% slow. Under more normal circumstances the clock wasn't accurate, but was close enough for my purpose of timing how long a user was on the RBBS.

To achieve my goal of at least 95% accuracy I would have to run the floppy disk controller with interrupts enabled, but, as I already pointed out, that produces an unbearable error rate. My only choice was to modify the hardware slightly to allow both the CTC and the floppy disk controller to run with interrupts enabled. That is a project I am currently working on. Once complete, I will publish the software and other modifications in this column.

Getting back to the real-time clock

The standard program counter maintenance pseudo-ops are available in SLRMAC. These include ORG, CESG, DSEG, ASEG, and COMMON, to name a few. A modified form of COMMON is available to assign up to 5 additional address spaces that can be assigned beginning addresses at link editing time. This can be very important if the program being written is targeted for a machine control environment in which dedicated portions of the address space must be defined. Again, this is an extension to the standard binary output files and as such is only available in the SLR format relocatable file.

When defining data storage areas it is not at all uncommon to need a string delimiter placed at the end of, for example, a console message. The DEFZ data definition statement will automatically generate a byte containing binary zeros at the end of the quoted string. And the DC statement will automatically turn bit 7 on in the last byte of the defined string.

A full complement of conditional assembly pseudo-ops and a full macro capability are provided. Two pseudo-ops that I have found to be especially useful are IFIDN and IFDIF. IFIDN tests for equality between two strings and sets the true condition according to the result. IFDIF performs exactly the opposite function of testing for inequality between the two strings and sets the true condition according to the result.

The method used to process the MACLIB pseudo-op is unique and worthy of comment. Intel mnemonic codes were designed to be used with the 8080, and there is no provision made for the additional instructions available on the Z80. When the program is being written for the Z80 and full use of its instruction set is desired, the extended instruction mnemonics are usually added to the assembly through a macro library. This solves the problem, but at the expense of slower run time and the necessity to maintain another macro library. SLRMAC, on the other hand, directly recognizes the extended set of Z80 mnemonics. Upon encountering the MACLIB Z80 statement the extend-

Thinking of the C Language?

THINK COMPUTER INNOVATIONS

NEW!!

C86 VERSION 2.3 with Source Level Debugging Support

The C language has rapidly become the development language of choice for applications ranging from Operating Systems to Accounting Packages. WHY? Its structured approach and extreme portability make it perfectly suited to today's fast-paced environment.

Of all of the C Compilers available for PC/MSDOS, more programmers choose COMPUTER INNOVATIONS' C86. WHY? Because it's part of a **COMPREHENSIVE** family of C products with an unparalleled reputation for performance, reliability, and stability.

C86 2.3 C COMPILER

C for PC/MSDOS began with C86 and today it remains perhaps the most solid, stable C Compiler available. Even competitor's ads show C86 as a consistent top level performer in benchmark testing.

Version 2.3 offers a host of new features including source level debugging support and a 40% boost in compilation speed. Call for complete specifications.

**COST: \$395 UPDATE TO 2.3: \$35 w/old diskettes NOT COPY PROTECTED
CALL ABOUT VOLUME DISCOUNTS**

LEARN C INTERACTIVELY WITH INTRODUCING C

Intimidated by rumors about the difficulty of learning C? Need to train your staff quickly? INTRODUCING C can help. INTRODUCING C combines a thorough, self-paced manual with a unique C interpreter to provide a fast, efficient method of learning C. Designed for both professional and casual programmers, it provides a comprehensive understanding of important C concepts such as standard K&R syntax and operators, full structures and unions, arrays, pointers, and data types. Requires IBM PC, XT, or AT with one disk drive and 192K bytes of memory.

COST: \$125 - NOT COPY PROTECTED

CI PROBE SOURCE DEBUGGER

Take advantage of C86 2.3 source level debugging support with CI PROBE. Cut down program development time and save money! CI PROBE is highly economical yet has the features of debuggers costing far more.

COST: \$225 - NOT COPY PROTECTED

C-TERP C86 COMPATIBLE INTERPRETER

The C-TERP INTERPRETER is a full K&R implementation that allows you to write code and execute it immediately without the compile and link steps. Once you have your program running with C-TERP you can compile the code (without alterations) with C86 for fast, efficient executable files. C-TERP requires 256K, 512K is recommended.

**COST: C86 version - List Price: \$300, Special Computer Innovations Price \$250.
Combined C86 & Lattice version - List Price: \$400, Special Computer Innovations Price \$350.**

Start With Us, Stay With Us

Computer Innovations offers a complete range of products that let you enter the C environment and create applications with the most advanced set of development tools available. Unparalleled tech support assures that you're always at the height of productivity.

To order call: **800-921-0169**

 **COMPUTER INNOVATIONS, INC.**

980 Shrewsbury Ave., Tinton Falls, NJ 07724 • (201) 542-5920

C-TERP is a trademark of Gimple Software. Prices and specifications subject to change without notice.

Circle no. 96 on reader service card.

routines, the last addition I had to make to the BIOS (sections 7 and 8 in the listing) was to initialize the CTC and the interrupt register with the proper constants at cold boot time.

If you are inclined to make these changes to your machine, be sure that you are running version 2.3 of the

BIOS, also known as the hard disk BIOS. Have a CTC reference manual handy to use when setting initial values.

One final comment: My system has a hard disk that ports into the Little Board through a SCSI peer buss adapter. I have found that this configuration runs completely error

free because the hard disk controller is a buffered device able to transfer data while other tasks are active.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service No. 195.

CP/M Exchange Listing (Text begins on page 114)

1) Add at line number 551 after statement FDALV EQU 50

```
INTRPT SET YES ; SET INTERRUPTS TO NO AS DEFAULT
TIMER EQU YES ; USE CHANNEL 2 OF CTC AS A TIMER
TYPER EQU NO ; USE BUFFERED KEYBOARD
```

2) Add at line number 1028 after statement WTBLN EQU \$-WINCH3

```
IF TIMER
```

```
;=====
;
; MODE 2 INTERRUPT VECTORS FOR THE CTC
;
;=====
```

```
ORG ($+17) AND 0FFF0H
```

TIMER\$INT\$TABLE:

```
DW BAUD$A
DW BAUD$B
DW TIMER$RTN
DW DSK$INT
ENDIF
```

TIMER\$RTN:

```
PUSH PSW ;SAVE THOSE REGISTERS THAT WE DESTROY
PUSH H ;*

LXI H,12 ;POINT AT HUNDREDS
INR M ;INCREMENT IT
MOV A,M ;GET HUNDREDS DIGIT
SUI 10 ;AT LIMIT
JNZ TIMER$EXIT ;NO
MOV M,A ;ZERO HUNDREDS

DCX H ;POINT AT TENS
INR M ;INCREMENT IT
MOV A,M ;GET TENS DIGIT
SUI 10 ;AT LIMIT
JNZ TIMER$EXIT ;NO
MOV M,A ;ZERO TENS

DCX H ;POINT AT SECONDS
INR M ;INCREMENT IT
MOV A,M ;GET SECOND DIGIT
SUI 60 ;AT LIMIT
JNZ TIMER$EXIT ;NO
MOV M,A ;ZERO SECOND

DCX H ;POINT AT MINUTES
```

(Continued on page 122)

Finally. BSW-Make.

The Boston Software Works now brings a complete implementation of the Unix "make" facility to MS-DOS. No more recompiling every file in sight after a small edit; no more wondering if you've really rebuilt every module affected by an edit. Just type "make" and BSW-Make automatically builds your product quickly, efficiently and correctly.

BSW-Make supports:

- most compilers and assemblers
- MS-DOS or PC-DOS v2.00 or later
- macros for parameterized builds
- default rules
- MS-DOS pathnames
- any MS-DOS machine (192K minimum)

Only \$69.95 postpaid (Mass. residents add 5% sales tax)

The Boston Software Works
120 Fulton Street, Boston, MA 02109
(617) 367-6846

Circle no. 126 on reader service card

C Users' Group

Over 45 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details
The C Users' Group

415 E. Euclid • Box 97A
McPherson, KS 67460
(316) 241-1065

Circle no. 17 on reader service card

TURBO PROFESSIONAL™

Turbo Pascal Tools

- SERVICE INTERRUPTS IN PASCAL
- EASY SIDEKICK™-LIKE PROCEDURES
- VARIABLE SIZE WINDOWS
- KEYBOARD MACROS
- DOS PROGRAM EXECUTION & RETURN
- **TI INCLUDE FILES, 109 ROUTINES**
- SOURCE CODE INCLUDED

SUPER MACS™ example program features keyboard macros, save macro files, load macro files, movable window, write screen to ASCII file, print concurrently in DOS 3.0 or greater. Works within other programs. Available separately as .COM file.

SUPER MACS ONLY (.COM)\$10.00
WITH TURBO PROFESSIONAL &
SUPER MACS SOURCE CODE\$49.95
S & H\$ 5.00

MC & VISA orders: (206) 367-0650
Or send check or money order to:
Sunny Hill Software
13732 Midvale North, Suite 206
Seattle, WA 98133

Requires IBM PC, XT, AT, or 100% compatible; DOS 2.0 - 3.1; Turbo Pascal 2.0 - 3.01B for compatibles. Sidekick trademark Borland Intl.

Circle no. 10 on reader service card

u4th

THE UNIX/XENIX-compatible Forth:

- C primitives
- dynamic memory management
- direct-threaded
- UNIX interfaces
- object-oriented Forth-source included!
- 400-page manual and glossary
- no royalties for developers

New Low Prices!

Plexus, Sun, Intel 286: \$495.00
PC XT, AT, Tandy: \$195.00

Now! VAX, AT&T 3B

UBIQUITOUS SYSTEMS
13333 BEL-RED ROAD NE
BELLEVUE, WA 98005
206-641-8030

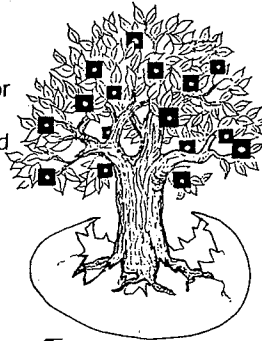
UNIX(TM) AT&T

XENIX(TM) MICROSOFT

Circle no. 152 on reader service card

Tree Shell

A Graphic Visual Shell for Unix/Xenix End-Users and Experts Alike!



Dealer inquiries welcomed.

COGITATE

"A Higher Form of Software"

24000 Telegraph Road
Southfield, MI 48034
(313) 352-2345

TELEX: 386581 COGITATE USA

Circle no. 24 on reader service card

No source code for your REL files?

REL/MAC

converts a REL file in the Microsoft™ M80 format to an 8080 or ZILOG™ Z80 source code MAC file with insertion of all public and external symbols.

- REL/MAC makes MAC source files
- REL/MOD lists library modules
- REL/VUE displays the bit stream
- REL/PAK includes all of the above
- 8080 REL/MAC demo disk \$10.00

REL/PAK for 8080 only \$99.95
REL/PAK for Z80 & 8080 \$134.95
on 8"SSSD disk for CP M™ 2.2

Send check, VISA, MC or C.O.D. to

MICROSMITH
COMPUTER TECHNOLOGY
P.O. BOX 1473 ELKHART, IN 46515

1-800-622-4070

(Illinois only 1-800-942-7317)

Circle no. 41 on reader service card

AT LAST —

A handbook that contains the Programming Codes for 100's of popular printers.

Announcing:

PROGRAMMERS' HANDBOOK OF COMPUTER PRINTER COMMANDS

The handbook gives you:

- Codes for printers made by over 40 Printer Manufacturers.
- Easy to use spiral bound book of over 250 pages of Programming Codes written in table form.
- Codes arranged by Written Code, Hex and Decimal equivalent, and with a brief description of what each code does.
- Codes for either Daisy-Wheel or Dot Matrix Printers (models through 1984).

ONLY \$37.95 + \$2 shpg./hdg. ppd. with a two week approval guarantee. IF NOT SATISFIED, return in original carton for refund of book price only.

FOR MORE INFORMATION OR TO ORDER CALL OR WRITE:

Cardinal Point
INCORPORATED

(812) 876-7811 (9-5 EST)

P.O. BOX 596, ELLETTSVILLE, IN 47429

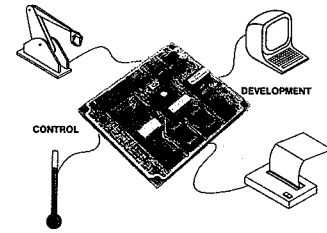
We accept MC, VISA, MO—same day shpg.

COO—\$2 extra. CKs—Allow extra 14 days.



Circle no. 11 on reader service card

SBC88 DEVELOPMENT AND CONTROL SYSTEM



WRITE IT — RUN IT — ROM IT

A single board computer development and control system that is so simple to use, you will be developing applications programs the first day!

- Choice of Basic or FORTH in ROM
- 8 channel, 8 bit analog to digital converter
- Two 8 bit input ports
Two 8 bit output ports
- Time of day
- 8088 16 bit uP
- Onboard EPROM programmer for complete program development
- RS-232 terminal and parallel printer port for program entry
- 7 current sinking outputs rated at 500 mA, 50 VDC
- Up to 32 K of user memory
- Floating point UNIFORTH available

Units start at \$279 (\$99 1000)
Compatible power supplies start at \$49
Vesta Technology, Inc. 7100 W. 44th Ave. Suite 101
Wheat Ridge, CO 80033 (303) 422-8088

Circle no. 156 on reader service card

ICs PROMPT DELIVERY!!!

SAME DAY SHIPPING (USUALLY)

OUTSIDE OKLAHOMA: NO SALES TAX

8087-2 COPROCESSORS \$125.00

DYNAMIC RAM

256K	256Kx1	120 ns	\$ 3.75
256K	256Kx1	150 ns	2.75
128K	128Kx1	120 ns	6.75
64K	16Kx4	150 ns	2.75
64K	64Kx1	150 ns	1.10

EPROM

27C256	32Kx8	250 ns	\$12.75
27256	32Kx8	250 ns	6.00
27128	16Kx8	250 ns	3.30
27C64	8Kx8	200 ns	5.25
2764	8Kx8	250 ns	2.75
2732A	4Kx8	250 ns	2.75

STATIC RAM

6264LP-15	8Kx8	150 ns	\$4.75
-----------	------	--------	--------

640 KByte MOTHERBOARD KITS: IBM PC/XT Zenith 150 & 100, Compaq Portable & Plus: \$69.50

QUANTITY ONE PRICES SHOWN

OPEN 6 1/2 DAYS: WE CAN SHIP VIA FED-EX ON SAT.
MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts **µP**
MICROPROCESSORS UNLIMITED
24,000 S. Peoria Ave. (918) 267-4961
BEGGS, OK. 74421
Prices shown above are for September 2, 1985
Please call for current prices. Prices subject to change. Please expect higher or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 5 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air (c. \$60.00) or Priority One (c. \$15.00)

Circle no. 64 on reader service card

```

INR      M          ;INCREMENT IT
MOV      A,M        ;GET MINUTE DIGIT
SUI      60         ;AT LIMIT
JNZ      TIMER$EXIT ;NO
MOV      M,A        ;ZERO MINUTE

DCX      H          ;POINT AT HOURS
INR      M          ;INCREMENT IT
MOV      A,M        ;GET HOUR DIGIT
SUI      24         ;AT LIMIT
JNZ      TIMER$EXIT ;NO
MOV      M,A        ;ZERO HOUR
    
```

TIMER\$EXIT:

```

POP      H
POP      PSW
    
```

```

BAUD$A:
BAUD$B:
DSK$INT:
    
```

```

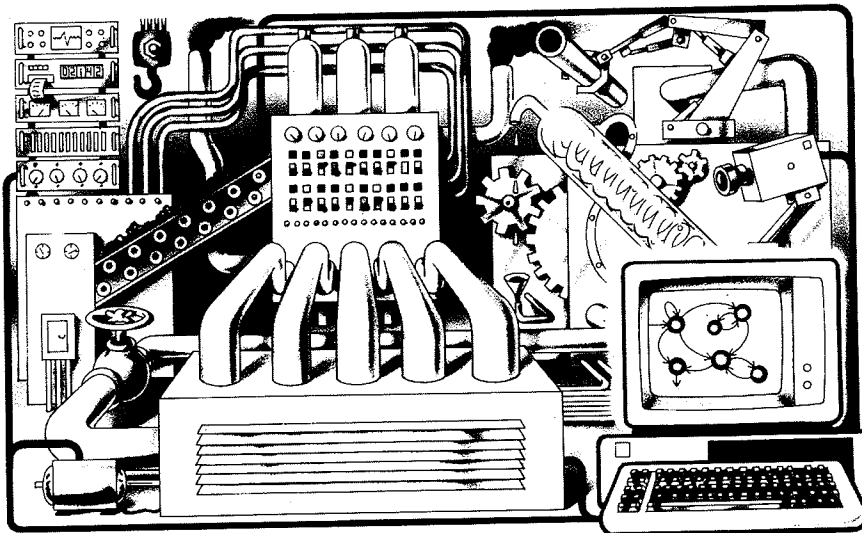
EI
RETI
    
```

3) Add at line number 3144 after label WR:

```

IF      INTRPT
DI
ENDIF
    
```

Csharp Realtime Toolkit



Realtime on MSDOS? Csharp can do it! Get the tools without operating system overhead. Cut development time with C source code for realtime data acquisition and control. **Csharp** includes: graphics, event handling, procedure scheduling, state system control, and interrupt handling. Processor, device, and operating system independent. **Csharp** runs standalone or with: MSDOS, PCDOS, or RT11. **Csharp** runs on: PDP-11 and IBM PC. **Csharp** includes drivers for Hercules and IBM graphics boards, Data Translation and Metrabyte IO boards, real time clock, and more. Inquire for Victor 9000, Unix, and other systems. Price: \$600



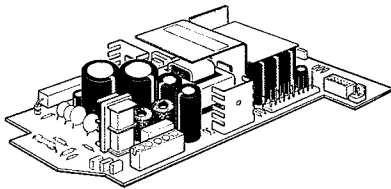
Systems Guild, Inc., P.O. Box 1085, Cambridge, MA 02142
(617) 451-8479

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

Switching Power Supply!

- + 5VDC - 8 Amps
- +12VDC - 5 Amps
- 12VDC - 1 Amp
- (81 WATTS MAX)



\$29⁹⁵ ea.

4 FOR \$99

ADD \$2 EA. UPS

BRAND NEW UNITS, MFG. BY BOSCHERT FOR HEWLETT PACKARD! PERFECT FOR SMALL COMPUTER AND DISK DRIVE APPLICATIONS #XL81-5630. INPUT 110V/220V, 50/60 HZ. NOMINAL OUTPUTS: +5VDC @ 8A, +12VDC @ 5A, -12VDC @ 1A. TOTAL MAX OUTPUT. MUST BE LIMITED TO 81 WATTS TOTAL! (WITH PIN OUT SHEET.) ORIGINAL FACTORY BOXED.

64K S100 STATIC RAM

\$119⁰⁰ KIT

LOW POWER!
150 NS ADD \$10

BLANK PC BOARD WITH DOCUMENTATION \$49.95

SUPPORT ICs + CAPS \$17.50

FULL SOCKET SET \$14.50

FULLY SUPPORTS THE NEW IEEE 696 S100 STANDARD (AS PROPOSED) FOR 56K KIT \$105

ASSEMBLED AND TESTED ADD \$50



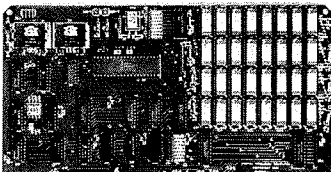
FEATURES: PRICE CUT!

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports IEEE 696 24 BIT Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- * 2716 EPROMs may be installed in any of top 48K.
- * Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- * Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- * BOARD may be partially populated as 56K.

256K S-100 SOLID STATE DISK SIMULATOR!
WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!

- FEATURES:
- * 256K on board, using + 5V 64K DRAMS.
 - * Uses new Intel 8203-1 LSI Memory Controller.
 - * Requires only 4 Dip Switch Selectable I/O Ports.
 - * Runs on 8080 or Z80 S100 machines.
 - * Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
 - * Provisions for Battery back-up.
 - * Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
 - * The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
 - * Compare our price! You could pay up to 3 times as much for similar boards.



BLANK PCB (WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE) \$69⁹⁵
(8203-1 INTEL \$29.95)

(ADD \$50 FOR A&T)

\$149⁰⁰

#LS-100

(FULL 256K KIT)

64K SS-50 STATIC RAM

\$99⁹⁵ (48K KIT)

LOW POWER!
RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION \$52

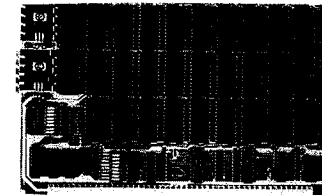
SUPPORT ICs + CAPS \$18.00

FULL SOCKET SET \$15.00

56K KIT \$109

64K KIT \$119

ASSEMBLED AND TESTED ADD \$50



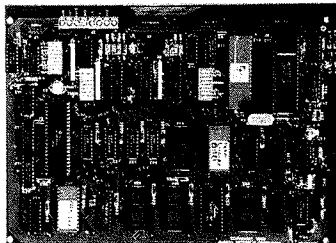
FEATURES:

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- * 2716 EPROMs may be installed anywhere on Board.
- * Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- * One Board supports both RAM and EPROM.
- * RAM supports 2MHZ operation at no extra charge!
- * Board may be partially populated in 16K increments.

THE NEW ZRT-80 CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

- FEATURES:
- * Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
 - * RS232 at 16 BAUD Rates from 75 to 19,200.
 - * 24 x 80 standard format (60 Hz).
 - * Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
 - * Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
 - * Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
 - * 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
 - * Composite or Split Video.
 - * Any polarity of video or sync.
 - * Inverse Video Capability.
 - * Small Size: 6.5 x 9 inches.
 - * Upper & lower case with descenders.
 - * 7 x 9 Character Matrix.
 - * Requires Par. ASCII keyboard.



\$89⁹⁵ #ZRT-80
(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716 CHAR. ROM. 2732 MON. ROM

\$49⁹⁵

SOURCE DISKETTE - ADD \$10
SET OF 2 CRYSTALS - ADD \$7.50

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD \$10

32K S100 EPROM/STATIC RAM

NEW!

FOUR FUNCTION BOARD!

NEW!

EPROM II FULL EPROM KIT \$69.95
A&T EPROM ADD \$35.00



BLANK PC BOARD WITH DATA \$39.95

SUPPORT ICs PLUS CAPS \$16

FULL SOCKET SET \$15

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

- FEATURES:
- * This one board can be used in any one of four ways:
 - As a 32K 2716 EPROM Board
 - As a 32K 2732 EPROM Board (Using Every Other Socket)
 - As a mixed 32K 2716 EPROM/2K x 8 RAM Board
 - As a 32K Static RAM Board
 - * Uses New 2K x 8 (TMM2016 or HM6116), RAM's
 - * Fully Supports IEEE 696 Buss Standard (As Proposed)
 - * Supports 24 Bit Extended Addressing
 - * 200 NS (FAST!) RAM'S are standard on the RAM Kit
 - * Supports both Cromemco and North Star Bank Select
 - * Supports Phantom
 - * On Board wait State Generator
 - * Every 2K Block may be disabled
 - * Addressed as two separate 16K Blocks on any 64K Boundary
 - * Perfect for MP/M* Systems
 - * RAM Kit is very low power (300 MA typical)

PRICES SLASHED!

32K STATIC RAM KIT — \$99.95

For RAM Kit A&T — Add \$40

TERMS: Add \$3.00 postage. Orders under \$15 add 75¢ handling. No C.O.D. We accept Visa and MasterCard. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50 add 85¢ for insurance.

Digital Research Computers

P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

4) Add at line number 3159 after label WREXIT:

```
IF      INTRPT
EI
ENDIF
```

5) Add at line number 3199 after label RD:

```
IF      INTRPT
DI
ENDIF
```

6) Add at line number 3214 after label RDEXIT:

```
IF      INTRPT
EI
ENDIF
```

7) Add at line number 4431 before label LOGMSG: and before JMP GOCPM.

```
IF      TIMER
XRA     A                ;GET A ZERO
STA     8                ;CLEAR TIMER SAVE AREAS
STA     9                ;*
STA     10               ;*
STA     11               ;*
STA     12               ;*
MVI     A, (LOW TIMER$INT$TABLE) AND 0F0H
OUT     CTCA0           ;SET INTERRUPT VECTOR LOW 8 BITS
MVI     A, HIGH TIMER$INT$TABLE
STAI    A               ;SET INTERRUPT VECTOR HIGH 8 BITS
IM2     A               ;SET INTERRUPT MODE 2
MVI     A, 0A7H        ;TIMER MODE - 256 PRESCALER - CONSTANT
                        ;FOLLOWS - RESET - CONTROL

OUT     CTCA2
MVI     A, 156         ;TIME CONSTANT
OUT     CTCA2
ENDIF
```

8) Add at line number 4458 and after label LOGMD:

```
IF      TIMER
DB      CR, LF, 'Real Time Clock Active'
ENDIF
```

End Listing

"C/80 . . . the best software buy in America!"

—MICROSYSTEMS

Now available in MS-DOS

Other technically respected publications like *Byte* and *Dr. Dobb's* have similar praise for **The Software Toolworks' \$49.95** full featured 'C' compiler for CP/M® and HDOS with:

- I/O redirection
- command line expansion
- execution trace and profile
- initializers
- Macro-80 compatibility
- ROMable code
- and much more!

"We bought and evaluated over \$1500 worth of 'C' compilers. . . C/80 is the one we use."

— Dr. Bruce E. Wampler
Aspen Software
author of "Grammatik"

In reviews published worldwide the amazing **\$49.95 C/80** from **The Software Toolworks** has consistently scored at or near the top — even when compared with compilers costing ten times as much!

The optional **C/80 MATHPAK** adds 32-bit floats and longs to the C/80 3.0 compiler. Includes I/O and transcendental function library all for only **\$29.95!**

C/80 is only one of 41 great programs each **under sixty bucks**. Includes: LISP, Ratfor, assemblers and over 30 other CP/M® and MSDOS programs.

For your free catalog contact:

The Software Toolworks
15233 Ventura Blvd., Suite 1118,
Sherman Oaks, CA 91403 or call 818/986-4885 today!

CP/M is a registered trademark of Digital Research.

Circle no. 134 on reader service card.

CONTINUING THE TRADITION

DR. DOBB'S JOURNAL ANNOUNCES THE RELEASE OF BOUND VOLUME 8

Every 1983 issue together in one source



BOUND VOLUME 8

DDJ turns pro. Some of the most powerful, professional programmer's tools ever published in a magazine are in this volume. Jim Hendrix's Small C compiler. Ed Ream's RED screen editor. A microcomputer subset of the Defense Department's official programming language, Ada. C and Forth and 68000 software. Because the magazine increased in size in 1983, this volume is bigger and better than ever.

Vol. 1 1976

The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering," front-panel, machine-language programming which was then the only way to do things. This is always pertinent for the bit crunching and byte saving, language design theory, home-brew computer construction and the technical history of personal computing.

Topics include: Tiny BASIC, the [very] first word on CP/M. Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSAI, and more.

Vol. 2 1977

1977 found DDJ still on the forefront. These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PILOT for microcomputers and a great deal of material centering around the Intel 6800, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NIBL.

Articles are about Lawrence Livermore Lab's BASIC, Alpha Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot & Turtle Graphics, many utilities, and even more.

Vol. 3 1978

The microcomputer industry entered into its adolescence in 1978. This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would

follow. These articles relate very closely to what is generally available today. Languages covered in depth were SAM76, Pilot, Pascal, and Lisp, in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors, and much, much more.

Vol. 4 1979

This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages, innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080/8080, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference.

Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe—more than ever!

Vol. 5 1980

All the ground-breaking issues from 1980 in one volumel Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a subject of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—reprinted here in full. Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler

for the 8080. Further with Tiny BASIC, a Syntax-Oriented Compiler Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler and, as always, even more!

Vol. 6 1981

1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features.

Highlights: Information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

Vol. 7 1982

In 1982 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler, and we continue to publish utility programs and useful algorithms. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi supplied a year of "Clinic" columns while delivering his famous review of JRT Pascal and writing the first serious technical comparison of CP/M-86 and MSDOS. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond, in the provocative essay, "Fifth-generation Computers."

Complete your reference library. Buy the entire set of Dr. Dobb's Journals from 1976 through 1983, Bound Volumes 1-8, for \$195.00. That's \$34.00 off the combined individual prices—a savings of almost 15%!

YES! Please send me the following Volumes of **Dr. Dobb's Journal**.

Payment must accompany your order.

Please charge my: Visa MasterCard American Express

I enclose Check/money order

Card # _____ Expiration Date _____

Signature _____

Name _____ Address _____

City _____ State _____ Zip _____ (please, no P.O. Boxes)

Mail to **Dr. Dobb's Journal**, 2464 Embarcadero Way, Palo Alto, CA 94303

Allow 6 to 12 weeks for delivery.

Vol. 1	_____ x	\$27.75 =	_____
Vol. 2	_____ x	\$27.75 =	_____
Vol. 3	_____ x	\$27.75 =	_____
Vol. 4	_____ x	\$27.75 =	_____
Vol. 5	_____ x	\$27.75 =	_____
Vol. 6	_____ x	\$27.75 =	_____
Vol. 7	_____ x	\$30.75 =	_____
Vol. 8	_____ x	\$31.75 =	_____
All 8	_____ x	\$195.00 =	_____

Sub-total \$ _____
California residents add applicable sales tax _____%

Postage & Handling Must be Included with order.

Please add \$1.75 per book in U.S. (\$4.25 each surface mail outside U.S. Foreign Airmail rates available on request.)

TOTAL \$ _____

TURBO PROGRAMMERS-

... CUTS DEBUGGING FRUSTRATION.

TDebugPLUS is a new, interactive symbolic debugger that integrates with Turbo Pascal to let you:

- **Examine and change variables** at runtime using symbolic names — including records, pointers, arrays, and local variables;
- **Trace and set breakpoints** using procedure names or source statements;
- **View source code** while debugging;
- **Use Turbo Pascal editor and DOS DEBUG commands.**

TDebugPLUS also includes a special MAP file generation mode fully compatible with external debuggers such as Periscope, Atron, Symdeb, and others — even on programs written with Turbo EXTENDER.

An expanded, supported version of the acclaimed public domain program TDEBUG, the TDebugPLUS package includes one DSDD disk, complete source code, a reference card, and an 80-page printed manual. 256K of memory required. Simplify debugging! \$60 COMPLETE.

TURBO EXTENDER™

Turbo EXTENDER provides you the following powerful tools to break the 64K barrier:

- **Large Code Model** allows programs to use all 640K without overlays or chaining, while allowing you to convert existing programs with minimal effort; makes EXE files;
- **Make Facility** offers separate compilation eliminating the need for you to recompile unchanged modules;
- **Large Data Arrays** automatically manages data arrays up to 30 megabytes as well as any arrays in expanded memory (EMS);
- **Additional Turbo EXTENDER tools** include Overlay Analyst, Disk Cache, Pascal Encryptor, Shell File Generator, and File Browser.

The Turbo EXTENDER package includes two DSDD disks, complete source code, and a 150-page printed manual. Order now! \$85 COMPLETE.

TURBOPOWER UTILITIES™

"If you own Turbo Pascal, you should own TurboPower Programmers Utilities, that's all there is to it." Bruce Webster, *BYTE Magazine*

TurboPower Utilities offers nine powerful programs: Program Structure Analyzer, Execution Timer, Execution Profiler, Pretty Printer, Command Repeater, Pattern Replacer, Difference Finder, File Finder, and Super Directory.

The TurboPower Utilities package includes three DSDD disks, reference card, and manual. \$95 with source code; \$55 executable only.

ORDER DIRECT TODAY!

- **MC/VISA Call Toll Free** 7 days a week. 800-538-8157 x830 (US) 800-672-3470 x830 (CA)
- **Limited Time Offer!** Buy two or more TurboPower products and save 15%!
- **Satisfaction Guaranteed** or your money back within 30 days.

For Brochures, Dealer or other Information, PO, COD — call or write:



3109 Scotts Valley Dr., #122
Scotts Valley, CA 95066
(408) 438-8608
M-F 9AM-5PM PST

The above TurboPower products require Turbo Pascal 3.0 (standard, 8087, or BCD) and PC-DOS 2.X or 3.X, and run on the IBM PC/XT/AT and compatibles.

Circle no. 207 on reader service card.

Excerpt from ZRDOS CHECKSUM routine. Code produces 11 bytes and 60 or 66 clock cycles.

```

CHECKSUM:
LD HL,(DREC) ; dir record into HL
LD DE,(CHKSIZ) ; chksum vector to DE
XOR A ; clear A
SBC HL,DE ; compare by subtract
RET NC ; result placed in HL
; ret from CHECKSUM
; if DREC > CHKSIZ
; else continue
    
```

Equivalent excerpt from CP/M 2.2 CHECKSUM routine. Code produces 18 bytes and 92 or 98 clock cycles.

```

CHECKSUM:
LHLD DREC ; Dir record into HL.
XCHG ; Shift to DE.
LHLD CHKSIZ ; Checksum vector to HL.
CALL SUBDH - ; Compare by subtract.
RNC ; Return from CHECKSUM if
; DREC > CHKSIZ else
; continue with CHECKSUM.
    
```

```

SUBDH:
MOV A,E ; Must move both bytes
SUB L ; of directory record into
MOV L,A ; register A for double
MOV A,D ; register subtract with
SBB H ; carry flag set—diff
MOV H,A ; is loaded in HL.
RET ; Return from SUBDH.
    
```

Table 3: ZRDOS-BDOS code comparisons. Comments and labels are from my disassembly.

Excerpt from CP/M 2.2 BDOS SELECTDISK routine. Code produces 22 bytes and 98 or 102 clock cycles.

```

MOVE: ; Necessary in Intel.
INR C ; Set up for loop.

MOVE0:
DEC C ; Get zero if there.
RZ ; End loop if empty.
LDAX D ; Source byte to A.
MOV M,A ; A to destination DPB.
INX D ; Next source byte.
INX H ; Next destination byte.
JMP MOVE0 ; Loop until C=0.
    
```

```

SELECTDISK:
; This subroutine fills
; the 16-byte disk
; parameter block.
; HL points to source.
; Switch source to DE.
; HL=destination DPB.
; C=size of DPB.
; Move it.
; SELECTDISK continues.
LHLD DPBADDR
XCHG
LXI H,SECTPT
MVI C,DPBLIST
CALL MOVE
    
```

Equivalent excerpt from ZRDOS SELECTDISK routine. Code produces 11 bytes and 52 or 57 clock cycles.

```

SELECTDISK:
LD HL,(DPBADDR) ; Source to HL.
LD DE,SECTPT ; Destination to DE.
LD BC,OFH ; DPB size=16 bytes.
LDIR ; Move until done.
; Resume SELECTDISK.
    
```

Table 4: ZRDOS-BDOS code comparisons

more compact Z80 codes for relative and conditional jumps, block transfers, direct loading of double registers, and double-register arithmetic, the Z-System design team have compressed most CCP and BDOS routines tightly enough to add important features. The most common Z80 optimizations in Z-System are the familiar relative and conditional jumps, such as the common substitution of Zilog's *DJNZ LOOP* for the Intel *DEC B-JP NZ, LOOP* pair to control loops, a saving of only 2 bytes and either one or four clock cycles on each pass. Much more impressive optimizations are scattered throughout Z-System and its utilities, however.

In the ZRDOS *CHECKSUM* routine shown in the first part of Table 3, page 46, for example, Dennis Wright (not to be confused with Z-Com developer Joseph Wright) substitutes a single 2-byte, 15-cycle Zilog arithmetic instruction for a cumbersome 10-byte, 51-cycle Intel subroutine (*SUBDH* in the second part of Table 3) to subtract the contents of the *DE* register pair from those of the *HL* register pair. Note also how Wright's direct loading of the *DREC* parameter into registers *DE* eliminates the Intel *XCHG* instruction. Wright's Zilog substitutions in this tiny code sample alone save 7 bytes and 32 clock cycles each time they are used.

The powerful Z80 block-compare and -transfer instructions (*LDI*, *LDIR*, *CPI*, and *CPIR*) are underused in Conn's subroutine libraries and in the Z-System code produced from them. Wright, however, combines *LDIR* with direct loads of double registers to load the disk parameter block in the ZRDOS *SELECTDISK* routine with only 11 bytes and 52 clock cycles of code. Digital Research's comparable Intel code requires twice as many bytes and cycles to do the same thing. Wright's use of the Zilog *LDIR* instruction saves a total of 11 bytes and 46 clock cycles, as you can see by comparing the first and second parts of Table 4, page 46.

The most effective Z-System optimizations have less to do with Z80 features than with sound programming practices. In comparing disassembled source code for BDOS and ZRDOS, I noticed that most of Wright's improvements involved enhanced logic and the elimination of redundancy. By

skillful sequencing of subroutines, he manages to make the flow between calling programs and ZRDOS system calls smoother and more efficient. He uses in-line code in preference to subroutines yet avoids redundancy by elegant relative loops.

The additional space harvested from his thoughtful assembly-language programming allows Wright to add several valuable routines and even four new system calls to ZRDOS. The major alterations in the CP/M 2.2 BDOS eliminate the nagging warm-boot requirement each time a disk is changed and make it easier to use the

read-only disk status feature. ZRDOS also permits wheel protection of individual files and file archiving. Two other useful new system calls are included to add a warm-boot trap option so that users can customize error messages by diverting jumps to location 0000H.

Following the release of ZRDOS, several new utilities have appeared that take advantage of these enhancements, and the list is growing. Most Z-System utilities can work just as well under the CP/M BDOS, but newer ones such as AC (archive copy) and VIEW require ZRDOS instead. Echelon ad-

PolyMake The Leading Make Utility

Are you still using a prehistoric Make? Now, step up to PolyMake, the most powerful and flexible Make utility available for programmers using MS-DOS. PolyMake is like an intelligent assistant that remembers how to rebuild a program when you change one part of the program. PolyMake will automatically invoke your compiler, assembler, linker, librarian or other tools to update a single program or entire software systems when you type — MAKE. PolyMake comes with built-in rules for rebuilding programs but you can also teach it new rules so you don't have to remember the file dependencies in your program. Advanced programmers prefer capabilities like fully recursive makefile processing and the ability to invoke PolyMake with special "flags" and macros that automate a whole range tasks. New Make users appreciate the Step-By-Step tutorial and intuitive commands. Handles source files written in any language. Requires DOS 2.0 & higher. Compatible with LANs, the IBM PC, XT, AT and other MS-DOS PCs. For complete details write for the POLYTRON Programmer's Catalog. **\$99**

PVCS The Most Powerful & Flexible Source Code Revision & Version Control System.

The POLYTRON Version Control System (PVCS) allows programmers, project managers, librarians and system administrators to effectively control the proliferation of revisions and versions of source code in software systems and products. PVCS is a superb tool for programmers and programming teams. (A special LAN version is also available.) If you allow simultaneous changes to a module PVCS can merge the changes into a single new revision. If changes conflict, the user is notified. Powerful capabilities include: Stores and retrieves multiple revisions of text; Maintains a complete history of revisions to act as an "audit trail" to monitor the evolution of a software system; Maintains separate lines of development or "branching"; Provides for levels of security to assure system integrity; Uses an intelligent "difference detection" to minimize the amount of disk space required to store a new version. Requires DOS 2.0 or higher. Compatible with the IBM PC, XT, AT and other MS-DOS PCs. Single User version \$395. **\$395**
5-station LAN version \$1,000, add \$500 for each additional 5 stations.

To ORDER: VISA/MC 1-800-547-4000, Dept. No. 355; Oregon/Outside US, 503-684-3000
Send Checks/POs To: POLYTRON Corp. 1815 NW 169th Pl., #2110, Dept. No. 355, Beaverton, OR 97006

POLYTRON
High Quality Software Since 1982®



***** NOTICE *****
**THIS PACKAGE CONTAINS
PROPRIETARY INFORMATION
WHICH IS THE PROPERTY OF
AMPRO COMPUTERS, INC.**

**PRIOR TO OPENING THIS PACKAGE
A CORPORATE OFFICER OR COMPANY
PRINCIPAL MUST SIGN THE SOURCE
LICENSE AGREEMENT
AND RETURN IT TO:**

**AMPRO COMPUTERS, INC.
Attn: Contracts Dept.
67 E. Evelyn Ave.
Mt. View, CA 94041**

**DO NOT OPEN UNLESS YOU
AGREE TO ABIDE BY THE
TERMS AND CONDITIONS OF
THE SOURCE LICENSE
AGREEMENT**



DIGITAL RESEARCH OPERATING SYSTEM END USER LICENSE AGREEMENT

*Use and possession of this software package
is governed by the following terms.*

1. DEFINITIONS - These definitions shall govern:

- A. "DRI" means DIGITAL RESEARCH INC., P.O. Box 579, Pacific Grove, California 93950, the author and owner of the copyright on this SOFTWARE.
- B. "CUSTOMER" means the individual purchaser and the company CUSTOMER works for, if the company paid for this SOFTWARE.
- C. "COMPUTER" is the single micro-computer on which CUSTOMER uses this program. Multiple CPU systems may require supplementary licenses.
- D. "SOFTWARE" is the set of computer programs in this package, regardless of the form in which CUSTOMER may subsequently use it, and regardless of any modification which CUSTOMER may make to it.
- E. "LICENSE" means this Agreement and the rights and obligations which it creates under the United States Copyright Law and California laws.

2. LICENSE

DRI grants CUSTOMER the right to use this serialized copy of the SOFTWARE on a single COMPUTER at a single location so long as CUSTOMER complies with the terms of the LICENSE, and either destroys or returns the SOFTWARE when CUSTOMER no longer has this right. CUSTOMER may not transfer the program electronically from one computer to another over a network. DRI shall have the right to terminate this license if CUSTOMER violates any of its provisions. CUSTOMER owns the diskette(s) purchased, but under the Copyright Law DRI continues to own the SOFTWARE recorded on it and all copies of it. CUSTOMER agrees to make no more than five (5)

copies of the SOFTWARE for backup purposes and to place a label on the outside of each backup diskette showing the serial number, program name, version number and the DRI copyright and trademark notices in the same form as the original copy. CUSTOMER agrees to pay for licenses for additional user copies of the SOFTWARE if CUSTOMER intends to or does use it on more than one COMPUTER. If the micro-computer on which CUSTOMER uses the SOFTWARE is a multi-user microcomputer system, then the license covers all users on that single system, without further license payments, only if the SOFTWARE was registered for that microcomputer. This is NOT a license to use the SOFTWARE on main-frames or emulators.

3. TRANSFER OR REPRODUCTION

CUSTOMER understands that unauthorized reproduction of copies of the SOFTWARE and/or unauthorized transfer of any copy may be a serious crime, as well as subjecting a CUSTOMER to damages and attorney fees. CUSTOMER may not transfer any copy of the SOFTWARE to another person unless CUSTOMER transfers all copies, including the original, and advises DRI of the name and address of that person, who must sign a copy of the registration card, pay the then current transfer fee, and agree to the terms of this LICENSE in order to use the SOFTWARE. DRI will provide additional copies of the card and LICENSE upon request. DRI has the right to terminate the LICENSE, to trace serial numbers, and to take legal action if these conditions are violated.

4. ADAPTATIONS AND MODIFICATIONS

CUSTOMER owns any adaptations or modifications which CUSTOMER may make to this SOFTWARE, but in the event the LICENSE is terminated CUSTOMER may not use any part of the SOFTWARE pro-

vided by DRI even if CUSTOMER has modified it. CUSTOMER agrees to take reasonable steps to protect our SOFTWARE from theft or use contrary to this LICENSE.

5. LIMITED WARRANTY

The only warranty DRI makes is that the diskette(s) on which the SOFTWARE is recorded will be replaced without charge, if DRI in good faith determines that the media was defective and not subject to misuse, and if returned to DRI or the dealer from whom it was purchased, with a copy of the original registration card, within ten days of purchase. Customer will receive support from the Vendor from whom customer has purchased the software. In addition, support is available from DRI directly, for qualified, registered customers under DRI's then current support policies. DRI reserves the right to change the specifications and operating characteristics of the SOFTWARE it produces, over a period of time, without notice.

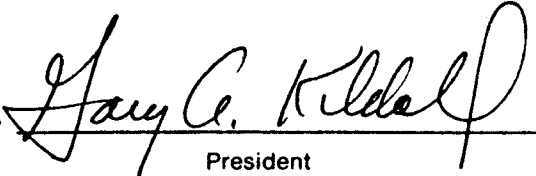
6. DRI MAKES NO OTHER WARRANTIES, EITHER EXPRESSED OR IMPLIED, AND DRI SHALL NOT BE LIABLE FOR WARRANTIES OF FITNESS OF PURPOSE OR MERCHANTABILITY, NOR FOR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES SUCH AS LOSS OF PROFITS OR INABILITY TO USE THE SOFTWARE. SOME STATES MAY NOT ALLOW THIS DISCLAIMER SO THIS LANGUAGE MAY NOT APPLY TO CUSTOMER. IN SUCH CASE, OUR LIABILITY SHALL BE LIMITED TO REFUND OF THE DRI LIST PRICE. CUSTOMER MAY HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE. CUSTOMER and DRI agree that this product is not intended as "Consumer Goods" under state or federal warranty laws.

7. MISCELLANEOUS

This is the only agreement between CUSTOMER and DRI and it cannot and shall not be modified by purchase orders, advertising or other representations of anyone,

unless a written amendment has been signed by one of our company officers. When CUSTOMER opens the SOFTWARE package or uses the SOFTWARE, this act shall be considered as mutual agreement to the terms of this LICENSE. This LICENSE shall be governed by California law, except as to copyright matters which are covered by Federal laws, and is deemed entered into at Pacific Grove, Monterey County, CA by both parties.



by 
President

SAVE THIS LICENSE FOR FUTURE REFERENCE

FROM: AMPRO Technical Support

DATE: 17 June 1986

SUBJECT: AMPRO "Little Board" Power Cable Applications Note

Every month Z-80 and 80186 "Little Boards" are returned for service that have had power applied incorrectly. This results in unnecessary repair, expense, and loss of computer time.

Only ordinary care need be given when constructing power cables to insure the +5 volts and +12 volts go to the correct pins.

These illustrations are to aid in your power cable construction.

