BabyBlue II™

CPU PLUS User's Manual

Version 1.0

Microlog™

A TU INTERNATIONAL COMPANY

222 Route 59
Suffern, New York 10901
(914) 368-0353

Copyright (c) 1984 by Microlog, Inc.
All rights reserved

Baby Blue is a trademark of Microlog, Inc.
CP/M is a trademark of Digital Research
Z-8Ø is a trademark of Zilog, Inc.
Columbia is a trademark of Columbia Data Products,Inc.
Compaq is a trademark of Compaq Computer Corp.
Corona is a trademark of Corona Data Systems
Eagle is a trademark of Eagle Computer, Inc.
TI is a trademark of Texas Instruments, Inc.
IBM is a trademark of International Business Machines, Inc.
Quadram and Quadboard are trademarks of Quadram Corporation
BSTAM is a trademark of Byrom Software
Televideo 95Ø is a trademark of Televideo Systems, Inc.
Hazeltine is a trademark of Hazeltine Corp.

TABLE OF CONTENTS

APPENDICES

FIGURES

TABLES

1. INTRODUCTION

1.1 THE INTELLIGENT MULTIFUNCTION BOARD

Baby Blue II represents a new standard in multifunction boards for the IBM Personal Computer. Built around the high-speed Z-80B microprocessor, it is actually a single-board microcomputer, more powerful and more versatile than many complete systems. Baby Blue II runs in most IBM-PC compatibles.

Baby Blue II offers all the features you expect from a superior multifunction board:

- Up to a 256 Kilobyte Memory Expansion, optional in 64K blocks.
- Clock/Calendar, with battery backup.
- Two Serial Ports (EIA RS-232C).
- Parallel Printer Port (Centronics-compatible).

As ordinary system expansions under the control of the PC's 8088 microprocessor, all facilities conform strictly to the IBM's standards. However, they are all dual-ported, which means that Baby Blue II's own Z-80 microprocessor can control them directly as a completely integrated, independent system.

1.2 SOFTWARE SUPPORT

It's capacity to support a wide variety of complex applications sets Baby Blue II apart from the crowd; like any good computer, it comes with its own system software. The standard package includes: system utilities for enhanced performance; communications software for exchanging information with other computers, including emulation of popular video display terminals; and Microlog's famous CP/M-80 Conversion Software, offering integrated dual operating system capability under PC-DOS.

1.21 SYSTEM UTILITIES

1.211 Print Buffer Spooler

Since Baby Blue II is itself a computer, it's smart enough to do on its own a lot of jobs which would normally tie up your entire system - this is called "background operation". Ever had to go out for pizza while your computer prints a long document? Now you'll have to find another excuse - Baby Blue II is going to run your printer(s) while you go back to work on your PC.

The Print Buffer Spooler accepts print output at high speed, then runs up to three printers simultaneously, while normal operation continues in the "foreground". All functions proceed at full speed, which is very unusual: conventional spoolers must "borrow" processor time from other tasks. Either they degrade sytem performance, print fitfully, or both.

You can get this kind of performance without Baby Blue II, if you purchase a stand-alone print buffer at considerable expense. Only the most expensive of these units can operate more than one printer; none are as easily controlled from the keyboard.

A printer can cost almost as much as your entire system. Fast printers cost much more than slower ones, but without a print buffer the extra expense is justified, because when the printer is running, you're locked out of your computer. Now that speed is not an issue, you can afford to shop for slower, cheaper printers. You may save as much as you spent on Baby Blue II.

You must have PC-DOS 2.Ø to use the Print Buffer/Spooler.

1.212 Ramdisk

A RAMdisk, sometimes called a "pseudo-disk", is an area of memory which your system thinks is a disk drive. It has a drive designation (eg., "c:"), and performs like a disk of equal capacity, except that because it has no mechanical parts, it's much, much faster. It can also save a lot of wear and tear on your disks and drives. It's most impressive when performing operations which are very disk-intensive, such as long data sorts. Try using a RAMdisk for any long operation, especially if you hear the disk drives working overtime.

Create Baby Blue II RAMdisks from the keyboard in any convenient number or size. Store alternate configurations on disk, customized for particular tasks.

You must have PC-DOS version 2.Ø to use Baby Blue II's RAMdisk facility.

1.213 Clock Support

Use Microlog's CLOCK software once to set Baby Blue II's real-time clock, and never type the time or date again. Each time you start your machine, the system clock is automatically synchronized to the real-time clock.

CLOCK also supports automatic execution of commands at preset times, for unattended operation.

1.22 SERIAL COMMUNICATIONS

1.221 Smart Terminal Emulator Package (STEP)

Talk to other microcomputers or connect to larger host computers, as an asynchronous terminal through Baby Blue II's serial ports. Where other "smart terminal" programs limite you to displaying lines of text, STEP offers full emulation of popular video display terminals (the standard package includes Televideo 950 and Hazeltine 1500; IBM 3101, DEC VT100 and many others are available as optional ".TRM" modules).

You can send or receive ASCII text or data files, and with STEP's unique "Sessions Menu", changing your configuration is a keystroke away.

1.222 File Transfer: BSTAM

STEP's file transfer facility is meant for on-the-fly data storage, to save expensive connect time. For transmission of programs and other binary files, as well as large ASCII files, use BSTAM (Byrom Software Telecommunications Access Method).

BSTAM is a sophisticated but easy-to-use communications program, capable of error-free data transmission over a modem on public telephone lines, or at high speeds between computers which have been wired directly together.

1.23 CP/M-80 CAPABILITY

As a descendant of Microlog's original Baby Blue, Baby Blue II is not limited to programs specifically designed for it - it will run most programs written for the CP/M-80 operating system.

The name derives from "Big Blue", IBM's traditional nickname. "Baby" connotes a symbiosis in which Baby Blue handles CP/M-80 code written for the Z-80, while depending on the the host PC's 8088 microprocessor to manage "life support" (operating system) functions - keyboard, screen, disk drives, etc. The closeness of this "mother-child" relationship has been Baby Blue's unique strength: you get dual operating system capability under PC-DOS alone, not the hassle of maintaining two separate operating systems.

If you can operate the PC, you can run CP/M programs - there are no new commands to learn, all peripheral devices work the same way, and all programs use PC-DOS diskettes. You actually can't tell the difference between a "native" program and a CP/M program - in effect, CP/M-8Ø becomes a vast library of time-tested, mature PC-DOS programs in a dizzying variety of applications. It's a whole world - the largest and most professional software resource available for microcomputers - yet it's almost unknown to many PC owners. We think you'll enjoy exploring it.

1.3 MODULAR STRUCTURE

Baby Blue II divides naturally into three modules: the Co-Processor Module, the Device Module and the 192K RAM Module. Each Module is configured separately, as explained under "Installation".

Co-Processor Module

The Co-Processor Module is the "brain" of Baby Blue II, including the Z-8Ø microprocessor and elaborate support circuitry. This Module also includes RAM Bank IV, which is the rightmost of Baby Blue II's four RAM Banks. Bank IV can act as ordinary expansion memory, but it is dual-ported, directly accessible to the Z-8Ø as well as the host system's 8Ø88. Whenever the Co-Processor is engaged in some task, Bank IV is reserved for use by the Z-8Ø.

Device Module

The Device Module includes both serial ports, the parallel port, and the clock. All devices are dual-ported, directly accessible to either the Z-8Ø or the 8Ø88. This means that they may be treated as ordinary system expansions, or as part of the Co-Processor.

192K RAM Module

The 192K RAM Module includes the first three banks of expansion memory, numbered Bank I, Bank II, and Bank III, starting with the leftmost bank. This memory is not dual-ported: it is directly accessible by the 8Ø88 only.

1.4 SYSTEM REQUIREMENTS

Baby Blue II works in an IBM PC or compatible machine, with the following minimum characteristics:

- 64 Kilobytes system RAM (128K recommended).
- One 5" floppy disk drive (one other drive recommended: it need not utilize 5" floppies).
- PC-DOS (or MS-DOS) Version 1.1 or 2.Ø.

1.5 SYMBOLS

The following symbols are used throughout this text:

<CR>	-	Carriage Return, or Enter: press the "Enter" or "Retrn" key when you see this symbol.
< >	-	All characters enclosed by this symbol are non-printing keystrokes used for control purposes: they are typed but will not appear on your screen.
[xxx]	-	items enclosed in square brackets must appear, but are variable depending on context or user response.
Boldface		indicates characters appearing on your screen.
c:	-	Indefinite control drive name. A place-holder showing where to put the name of the disk drive containing the command you are currently using.
s:	-	Indefinite source drive name. A place-holder showing where to put the name of the disk drive from which you are getting a file.
d:	-	Indefinite destination drive name. A place-holder showing where to put the name of the disk drive you are writing to.
e:	-	Another indefinite drive name.

NOTES:

2. BABY BLUE II HARDWARE INSTALLATION

Does the idea of fiddling about in the vitals of a computer make you uneasy? Relax - physically installing Baby Blue II is considerably less complicated than changing an old-style typewriter ribbon. Access to the expansion area involves nothing more than removing the cover of the System Unit, and the board itself literally plugs in. The only tool you'll need is a medium screwdriver.

More complicated is setting the various switches and "jumpers" which configure Baby Blue II for your particular system - fortunately, you can resolve this issue before tearing apart your machine. With so many features on one board, you have a lot of options, probably more than you'll want to face right now. We recommend that you follow our standard configuration, which limits your job to a single set of switches on Baby Blue II, and possibly one more set on the main circuit board (motherboard) of the PC itself. This method will work for practically all systems, excluding a few with very unusual modifications.

A word about static electricity - the kind that gives you a shock when you touch a doorknob or another person - it can damage integrated circuits; the memory "chips" in your computer and on Baby Blue II are particularly vulnerable. Professionals often take special precautions to insure that sparks don't jump from their own bodies to the circuit boards on which they are working. You aren't likely to have trouble if you observe elementary precautions such as "tagging up" on a metal table to discharge yourself before handling any circuit boards. However, if you're in a place where you get a lot of little shocks, it's time to look into antistatic sprays and other products for high-static environments - those jolts aren't doing your computer any good during normal operation.

2.1 CONFIGURATION

User-configurable options on Baby Blue II include three DIP switch blocks, labelled SW1, SW2 and SW3, and three "jumpers", labelled H1, H2, and H3.

The DIP switch blocks are brightly-colored rectangular blocks, containing tiny numbered switches. SW1 and SW3 each contain seven switches, while the smaller SW2 contains only three. The switches are turned "ON" by sliding them in the indicated direction - a pen point is a good tool, but stay away from pencils, because graphite is conductive and could cause a short circuit.

The jumpers are rows of spike-like pins projecting straight out from the board. They are set by connecting the pins together, using a special "jumper" socket which fits over two pins at a time.

2.11 STANDARD CONFIGURATION

Figure 1 shows the configuration in which Baby Blue II is shipped from the factory. With the exception of SW3, which may vary widely, these settings will be standard for most installations. Check to see that all settings are as shown, and reset any that are incorrect.

2.111 RAM Module (SW3)

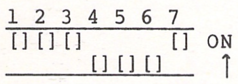
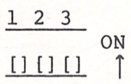
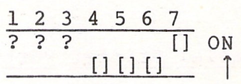
SW3 configures the RAM Module. Since this switch varies with the amount of memory installed in your system, it is the one part of the installation which we can't standardize. We'll explain all the issues in the next Section ("Installing Memory"), but first let's look at the individual switches.

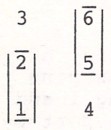
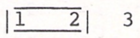
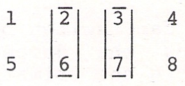
In Figure 1, switches 1, 2 and 3 are defined only as "?", which means "as populated". Turning each of these switches ON enables the corresponding Bank of RAM - switch 1 ON enables Bank I (far left), etc. You enable a Bank only as it is populated, that is when the sockets in that Bank are filled with RAM chips. If you enable an empty Bank, you'll get a system error when you turn on your machine; if you fail to enable a populated Bank, you won't be able to use the additional memory.

You must populate and enable the Banks in numerical order. If a Bank is disabled, the higher-numbered ones will not work properly, even if they are fully populated.

Switches 4 through 7 assign the RAM to a specific position in the scheme of memory addresses. As you will see shortly ("Installing Memory"), deciding this position is the main work of configuring Baby Blue II.

Fig. 2-1: Baby Blue II Factory Configuration

DIP Switches		
SW1	SW2	SW3
		

Jumpers: Sockets Installed Between Indicated Pins		
H1	H2	H3
		

<u>Assignments</u>

<u>Z-8Ø Module (SW1)</u>:

RAM Bank IV:	Page E
Z-8Ø Port Address:	31CH

<u>RAM Module (SW3)</u>:

Base Page (Bank I):	Page 4
Banks Enabled:	? (as populated)

<u>Device Module Port Addresses (SW2, H1, H2)</u>:

COM1:	3F8 - 3FF, INT 4
COM2:	2F8 - 2FF, INT 3
LPT?:	278 - 27F (Alternate Printer Adapter)
CLK:	338 - 33F

1 2 3

11111Ø ØØØ ØØØØØ1

2.112 Device Module

Configuring the Device Module means assigning a <u>Port Address</u> to each of four devices: the two serial ports, the parallel printer port, and the clock. The addresses should not overlap the addresses of other devices already in your system, and they should conform to the standard addresses used by the IBM PC.

Note that we are speaking of hardware addresses, which you must physically configure when installing your boards. Although the MODE command permits you to redirect output from one port to another, for example by setting COM1 equal to LPT1, this does not affect the physical addresses of the ports themselves. No two ports can share the same address.

<u>Parallel Printer Port</u>

DOS uses the name "LPT" to designate a parallel port. It's possible to have as many as three, named LPT1, LPT2 and LPT3. You can't directly assign these names by configuring your hardware, which is why we show Baby Blue II's parallel port as "LPT?" in Figure 1.

What you can assign is one of three possible port addresses, using the jumper at H2. The first of these is reserved for the parallel port on the IBM monochrome board, so we don't provide that option on Baby Blue II. The second and third are called the "Primary Printer Adapter" and the "Alternate Printer Adapter", respectively.

You can switch between the second and third addresses by moving the jumper at H2: Connecting H2's pins 2 and 3 selects the Primary Adapter, and connecting 1 and 2 selects the Alternate Adapter. You can disable the port by removing the jumper socket from H2 altogether.

At boot-time, the operating system looks at each address in turn to see if there is a port installed there. If there isn't, it skips to the next address. The first port found becomes LPT1, the second becomes LPT2, and the third, LPT3. The factory setting assigns Baby Blue II's port to the Alternate Printer Adapter, because this is the last address checked, hence the least likely to be occupied.

Keeping track of the LPT's may be confusing, but the only one that's really important is LPT1, which is the primary system printer, also called the "PRN" device. The best way to use Baby Blue II's parallel port is to invoke the Print Buffer/Spooler, which automatically reassigns the port to LPT1, no matter what it's previous assignment.

<u>Serial Ports</u>

The factory settings configure the two serial ports as COM1 and COM2. The connector at the back panel, labeled "J1" is always COM1, or your primary serial port. Header J2 is always COM2 (secondary serial port).

These ports cannot be switched or selectively disabled. Therefore, with Baby Blue II installed, you cannot have any other ports in your system which are also assigned to COM1 or COM2.

<u>Clock</u>

There is no standard Clock address for the PC, so Baby Blue's real-time clock is assigned to an available address not reserved for one of the standard devices.

2.113 Co-Processor Module

SW1 configures the Co-Processor Module, including both the Z-8Ø Microprocessor and RAM Bank IV. We will discuss other options later ("Installing Memory"), but for now, the factory setting will be the best choice for most installations.

2.2 INSTALLING MEMORY

2.21 MEMORY ORGANIZATION

Your computer's memory is divided into "Pages", or "Segments" of 64 Kilobytes each. There are sixteen in all, beginning with Page Ø - they are usually counted in Hexadecimal notation, which means that where you expect Pages 1Ø-15, you see Pages A-F. It's useful to make a chart, as follows:

Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Each Page represents a physical bank of memory. For example, if Baby Blues II's banks are all populated, they occupy four Pages. Page Ø is always on the "mother" (system) board, and certain other Pages are usually reserved for system functions: in particular, Page F is generally used for ROM, and Page A and/or B are customarily assigned to the memory which keeps track of your video display.

Each bank must have a unique Page number, because your system uses it to locate this particular block of memory. Therefore, no two physical blocks of memory can share the same Page number.

You assign Page numbers to Baby Blue II's RAM by setting the board's DIP switches, in particular SW1 and SW3:

SW1 assigns RAM Bank IV, which is the RAM used by the Co-Processor Module.

SW3 assigns RAM Bank I, which is the first bank in the 192K RAM Module - Banks II and III automatically occupy the next two Pages, so the entire Module installs as a single block.

Your main concern is to discover which Pages are available. Let's suppose you have an IBM PC, with a fully populated 64K motherboard, and no other memory expansions. We'll use an asterisk ("*") to indicate existing memory, and an "R" to indicate reserved areas:

Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
*										R	R				R

↑

In theory, you could choose between twelve available Pages (1-9, C-E). In fact, you would probably choose to put the 192K RAM Module as low as possible, immediately after the last occupied Page:

Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
*	*	*	*							R	R				R
	↑														

This is because you want the RAM to be part of "System Memory". System Memory is your computer's general-purpose memory: it includes all Pages starting with "Ø" and counting up to the first empty Page, or the first Reserved Page, whichever comes first. When you run CHKDSK, the figure returned for Total Memory includes only the System Memory.

If, for example, we assign our RAM to Pages 2 - 4, we add nothing to system memory. CHKDSK would still only show 64 Kilobytes Total Memory. Why? There would be an empty Page immediately after the first 64K of memory:

Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
*		*	*	*						R	R				R
	↑														

Excluded memory can only be used for special purposes. This is the reason why you must populate and enable Baby Blue's memory banks consecutively - a disabled bank creates an empty Page, rendering further banks useless.

It's a good idea to cross-check by filling in your chart of available Pages as you go - this will quickly show you if you have accidentally assigned two banks to the same Page, or have unintentionally left a gap. Remember that the RAM module takes up as many Pages as it has populated banks - fill in a Page on your chart for each enabled bank.

2.22 BASIC PROCEDURE

To add RAM to system memory, first determine how many Kilobytes you already have. Referring to Table 1, match this number under "Existing Memory", and read across to determine your Page number and corresponding switch settings. This will put you on the first available Page (lowest number) after allowing for all memory presently installed in your system. For example, if you have 64K already, this uses up Page Ø, so your first available Page is 1. If you have 64K plus a 256K expansion board, you've used up Pages Ø,1,2,3 and 4 (five Pages of 64K each, or 32ØK), and your first available Page is 5:

Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
*	*	*	*	*						R	R				R
					↑										

To add both the RAM Module and Co-Processor Memory (Bank IV) to system memory, you would perform this operation once for the first set of switches, say SW3. Carrying on our example:

Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
*	*	*	*	*	*	*	*			R	R				R
					↑	↑	↑								

Now recalculate to install the second module (SW1), taking into account the memory added by the first:

Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
*	*	*	*	*	*	*	*	*		R	R				R
								↑							

2.23 "HIDING" MEMORY

Sometimes, we want "hide" a part of RAM from the system, reserving it for some special purpose. The Co-Processor Module, which uses RAM Bank IV to carry out background operations, is a perfect example. If you want to use Baby Blue II's Print Buffer Spooler, the system must not attempt to use Bank IV while the printer is in operation. We prevent this by excluding the bank entirely from system memory - programs like the Spooler, which run on Baby Blue II's Z-8Ø, will find Bank IV, but others can't.

We can hide a Page by deliberately leaving a gap below it. The simplest method is to choose an available Page above the first Reserved Page(s). The factory setting assigns Bank IV to Page E, which is the highest numbered Page not reserved by the operating system. Page E is seldom occupied, and definitely guarantees that Baby Blue II will function as a background processor. Therefore, we recommend that you leave SW1 at its factory setting (Fig. 1).

2.24 SYSTEM BOARD SWITCHES

The switches on the mother board tell the computer's operating system how much memory is available in the system, and how it is organized; also, some of the switches reflect your hardware configuration (how many disk drives, what kind of monitor, etc.). Each time you turn power on, the operating system interrogates the switches and proceeds according to the information it finds there. The information required will vary from machine to machine.

For example, the IBM PC needs to be "told" both how much memory is installed directly on the mother board, and also how much total memory is available, including any expansion boards. By contrast, you tell the PC/XT how much memory is on the mother board, but it figures out how much total memory is in the system without reference to any switches. There are also two versions of the PC: an older one which can socket 64K of RAM on the mother board, and a newer version which sockets 256K on the mother board. The switch blocks on these two machines appear to be similar, but have different meanings, so you must know which machine you have. Some systems have no switches at all.

Don't touch any switches on the mother board unless you change your total system memory. For example, the factory setting for SW1 excludes Bank IV from System Memory. If this is the only populated bank on Baby Blue II, the standard installation won't affect the motherboard.

2.25 AVOIDING "RESERVED" MEMORY

Some machines effectively reserve low-numbered Pages for memory chips to be installed in sockets directly on the mother board. For example, the IBM PC-2 reserves the first four Pages (Ø, 1, 2, and 3) for the 256 Kilobytes which can be installed on the motherboard - your lowest available Page is 4.

Note that if the mother board has not been fully populated, a gap will appear in the sequence of memory Pages: for example, if you have 128K or RAM on the motherboard, there is a gap at Pages 2 and 3:

Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
*	*	R	R	*	*	*				R	R			*	R
		↑	↑												

Since you can't assign Baby Blue II to the reserved Pages, you can't use your extra memory, even though you may have more than enough to fill two Pages. Note that the same restriction applies to all memory expansions - you can't add memory to the system until the motherboard is fully populated.

The Configuration Notes show the Pages reserved by specific machines. For machines not covered, you must find this information in the manufacturer's documentation before attempting to install Baby Blue II.

2.26 HIDDEN CONFLICTS

Sometimes a hidden conflict with some element in your system will force you to depart from the standard procedure. The culprit may be another board, or it may be some piece of special software, especially custom installations to the operating system itself. Scan these notes to see if anything rings a bell, but don't go out of your way looking for trouble. Unless you're sure a problem exists, just go ahead and follow the standard procedure.

This section will help you understand the issues, but it isn't strictly necessary to know the cause of a problem. If a conflict develops, turn to the universal troubleshooting procedure, under "Diagnostics" in the Appendix: in almost every case, a new set of Page assignments will do the trick.

2.261 Concealed Memory

Suppose you run CHKDSK, and use the figure returned under "total memory" as your basis for installing Baby Blue II. This figure should be the same as the total physical memory in your computer, but what if there's some RAM which has been excluded from system memory? In this case, you can only determine your actual memory if you know what boards you have previously installed, together with their Page assignments and the amount of RAM on each. Remember, you have to account for all memory physically installed in your machine, including not only RAM, but also any ROM (Read Only Memory).

The instructions account for factory installed memory and reserved areas, but you'll have to keep track of expansion boards. Be sure that a physical count of your expanison memory agrees with the figure returned by CHKDSK; if it doesn't, you must determine the assignments of the "phantom" Pages.

One special case to watch out for: some boards have been designed to automatically "wrap around" the Reserved Pages at A or B, and continue on up into high memory, using Pages C, D or E. The high Pages, which are usually used for a RAMdisk, will not be part of System Memory, so CHKDSK won't count them. This is a rare case, but important because it conflicts with the recommended setting for SW1, which maps Bank IV into Page E.

2.262 Software Installations

Some accessories, purchased as system utilities or as installations to DOS, may attempt to use Baby Blue II in unexpected ways, especially when Bank IV has been included in system memory.

For example, most RAMdisk software is designed to use all of system memory above some predefined Page. If Bank IV is at the top of system memory, the RAMdisk may attempt to use it at the same time as Baby Blue II's Z-8Ø microprocessor. The system will see contradictory information and shut down in confusion. In this case, the most elegant solution would be to exclude Bank IV from system memory, preferably at Page E, as recommended. Alternately, set Bank IV as low as possible in system memory, that is, immediately after the memory on the motherboard. Then tell the RAMdisk to begin using memory starting somewhere above Bank IV.

Microlog's RAMdisk for DOS 2.Ø automatically utilizes the lowest available Pages, so it requires the opposite course. To use the RAMdisk, you should install Bank IV at the <u>top</u> of system memory, or, as recommended, exclude it completely.

2.263 Port Conflicts

Your system maintains not one, but two addressing schemes. We have discussed the organization of memory; the other scheme locates Input/Output (I/O) devices by defining a separate system of unique <u>Port Addresses</u>. It's possible to develop contentions with boards containing no memory, if the I/O address of a newly installed device overlaps some port already in use. Most of Baby Blue II's devices are contained in the Device Module. However, the Z-8Ø is also a device; its port address is tied to the Co-Processor RAM, varying with the Page assignment of Bank IV.

I/O contentions involving the Co-Processor Module are best solved by assigning a new Page number - and hence a new port address - to Bank IV. Actual conflicts may also involve the Device Module - generalized troubleshooting is treated under "Diagnostics" in the Appendix.

2.264 Quirks

Many RAM boards will not tolerate any system memory installed above them unless they are fully populated, even if the empty banks are theoretically disabled. This is not the case with Baby Blue II, except that Bank IV cannot be disabled and must always be populated.

Even when populated, some boards will still not permit you to assign the Page numbers above them to additional system RAM. The Quadram 256K Quadboard and the IBM 32K expansion memory board are known to cause trouble unless you position them above Baby Blue II's RAM in system memory.

Note that in all cases, we are speaking only of RAM included in system memory. This type of conflict will not occur if Bank IV is excluded as we recommend.

2.3 INSTALLATION

This section will guide you step-by-step through installing and configuring your Baby Blue II. Sections include:

<u>Step-by-Step Instructions</u>

How to get into your system, insert Baby Blue II, and close up. Although these instructions are specific to the IBM PC, other machines are very similar - follow these instructions in conjunction with the manufacturer's documentation for your computer.

<u>DIP Switch Settings</u>

A master chart of possible settings for Baby Blue II's DIP switch blocks SW1 and SW3.

<u>Configuration Notes</u>

The fine points of configuring for different machines, including motherboard DIP switches and Reserved Pages, as follows:

(2.42) IBM PC 1
(2.43) IBM PC 2
(2.44) IBM PC/XT
(2.45) Other IBM-Compatible PC's

We will not cover customized port addresses in this section. If you want to assign your serial and parallel ports to some device other than the standard COM1/COM2 and LPT's, turn to "Hardware Technical Reference".

Before going on, you should note your "system configuration" as it affects Baby Blue II, including especially:

- exactly which system and model you have, so that you can refer to the proper set of instructions.
- how much memory you have in Kilobytes, and how it is distributed between the "mother" board and expansion memory boards.
- the manufacturer and model name of any expansion boards, particularly memory expansions and disk drive interfaces.
- type of monitor (screen) and video interface.
- any installations made in software to your operating system, for example whether you are using disk emulator software (often called "RAMdisk", or "pseudo disk"), or a print spooler.

2.4 STEP BY STEP HARDWARE INSTALLATION

2.41 IBM PC - ALL MODELS

2.411 Begin

Turn the system OFF. Disconnect all power from the System Unit, then disconnect all peripheral devices (be sure you know how to reinstall them). Take your monitor off the top of the System Unit cabinet.

2.412 Remove Cover

At the bottom corners of the System Unit rear panel you will find at least two pan-headed screws that hold the cover in place - later models have three more screws, two at the top corners and one at top center (don't fiddle with the hex-headed ones: they retain internal components). Remove the screws, then remove the cover by sliding it towards the front of the System Unit and then up.

2.413 Verify Switch Settings

First, check to see that Baby Blue II's switches and jumpers match the setings given in Figure 1. Now follow the instructions for your machine in the Configuration Notes for your machine. If you change switches on the mother board, make a note first of the original positions. Pay special attention to Reserved Pages when setting Baby Blue II's switches - hopefully, you will only change SW3, to map the RAM Module into an unoccupied area of memory, as described in "Installing Memory".

Although we don't recommend it, you may use the same instructions to alter SW1 (Co-Processor Module/RAM Bank IV), to include Bank IV in system memory. We have also explained how to change the address of the parallel port (J3) by moving the jumper socket at H2 (See "Standard Configuration").

If you contemplate further changes, be sure you have read and understood "Hardware Technical Reference".

2.414 Install Cables

Now, before you forget, is the time to install the two ribbon cables which came with Baby Blue II. They plug into the twenty-six pin headers J2 and J3, which are the second serial port and the parallel port, respectively. The smaller connector on each cable fits over the pins - the colored stripe down one edge of the cable goes on the right, at pin "1", which is labeled directly on the board.

THIS PAGE INTENTIONALLY LEFT BLANK

The two cables are identical, except that one terminates in a male DB-25 connector, which is similar to the connector at Baby Blue II's mounting bracket - a double row of pins projects from its face. This cable goes on the serial port at header J2. The other cable ends in a female DB-25, similar to the male except that it contains sockets to mate with the pins on the male.

When you close up the cabinet, make sure you lead the free ends of the cables out the DB-25's out the back, so you'll have access to them. You can go through any available opening in the back-panel, or else drape the cable over the top before you replace the cover - the flat cable will nest securely under the cover's edge, for a neat installation. If you have difficulty figuring out how to route the cable, look at your disk drive cables. When turning a corner, fold the cable back over itself at ninety degrees, so that it lies flat.

2.415 Choose Expansion Socket

You will be working in the open area on the left side of the System Unit, as viewed from the front - to the rear of this area you will find the system expansion sockets, sticking up from the mother board. They are made to receive Baby Blue II's "edge connector" - the row of thirty one gold contacts projecting from the either side of the board's lower edge. Some sockets will already be occupied - Baby Blue will work in any open socket, but avoid the leftmost one if you can.

At least one of your present boards will be long enough to reach the front panel of the System Unit, where you'll notice that it's supported by a plastic edge-guide. Baby Blue II come with a similar edge-guide: just press the two plastic pegs through the matching holes in the front panel, directly in line withg the expansions socket you've chosen.

Also in line with the expansion socket, you should see an L-shaped piece of metal about an inch wide, fastened with a screw to the top of the back panel - it covers a wide slot. Unscrew the retaining screw, then lift the slot cover clear. Save the screw.

2.416 Install Baby Blue II

Bolted to Baby Blue II is a metal mounting bracket designed to replace the slot cover. Grasping the board's top corners, start it's free edge into the plastic edge-guide. Then lower the board into the System Unit, easing the mounting bracket's tongue into the gap between the mother board and the back panel. Carefully press Baby Blue II into the expansion socket.

It's a tight fit, and if the board is cocked in any direction it may not go in. Try rocking it lightly, end-to-end and side-to-side, while applying steady downward pressure. The board should seat without excessive pressure, and you should feel that it has gone all the way into the socket - it should sit square in the chassis and not appear cocked. Be careful not to disturb any other expansion boards in the process.

Center the back panel screwhole within the mounting bracket's elongated hole, then replace the screw which you saved. Look to see that the board is still square to the chassis. Check for accidentally dislodged plugs, particularly the speaker connection at the front left of the System Unit.

2.417 Reconnect Cables and Test System

Before closing up, let's make sure everything is OK. Reconnect your peripheral devices to the System Unit - if you're not sure where everything went, see Section 2 (Setup), of your IBM <u>Guide To Operations</u>. Reconnect power cables last of all, then turn on the system to make sure it boots (displays the cursor, beeps, activates the A drive, then comes up asking you for the date, etc.), and responds normally to DOS commands.

Make a duplicate copy of your Microlog master diskettes and put the originals away. Now find the file, "TESTBB2.EXE", and put the disk containing this file in drive A:. Type:

```
          A:TESTBB2 <CR>
```

This initiates the Baby Blue II diagnostic tests - if no errors are shown, your Baby Blue II is properly installed and working. If TESTBB2 shows an error, or if your system behaves abnormally, turn to the Appendix under "Diagnostics".

2.418 Finishing Up

Everything works? Good - turn the power back off, and make sure the free end of Baby Blue II's ribbon cables are outside the system unit. To replace the cover, start by tipping it downward, then leveling it as it slides on. Take care not to bump any boards seated in the expansion sockets. The cover has to go all the way on - if that black tab at the center of the back panel protrudes above the cover, back off and try again. Replace and tighten the cover retaining screws.

This completes the hardware installation. Check again that your system boots normally, and proceed to the next chapter.

Table 2-1: DIP Switch Settings: SW1 and SW3

Existing Memory	Switch 4	5	6	7		Assigns Page/ Segment	Existing Memory	Switch 4	5	6	7		Assigns Page/ Segment
64K	[]	[]	[]		ON	1	512K		[]	[]	[]	ON	8
				[]	↑			[]				↑	
128K	[]	[]		[]	ON	2	576K		[]	[]		ON	9
			[]		↑			[]			[]	↑	
192K	[]	[]			ON	3	64ØK		[]		[]	ON	A*
			[]	[]	↑			[]		[]		↑	
256K	[]		[]	[]	ON	4	7Ø4K		[]			ON	B*
		[]			↑			[]		[]	[]	↑	
32ØK	[]		[]		ON	5	768K			[]	[]	ON	C*
		[]		[]	↑			[]	[]			↑	
384K	[]			[]	ON	6	N/A			[]		ON	D*
		[]	[]		↑			[]	[]		[]	↑	
448K	[]				ON	7	N/A				[]	ON	E*
		[]	[]	[]	↑			[]	[]	[]		↑	

* One or more Pages in the range A through F are reserved by all machines. For specifics, find your computer in the Configuration Notes.

SW1

Configures Co-Processor Module, including RAM Bank IV.

Switches 1, 2 and 3, always ON.

SW3

Configures 192K RAM Module: RAM Banks I, II and III. The indicated Page number is assigned to RAM Bank I, with Banks II and III automatically occupying the next two Pages.

Switches 1, 2 and 3 enable corresponding RAM Banks: switch 1 ON enables Bank I (far left), etc. Enable each Bank as it is populated; Banks must be enabled consecutively.

2.42 IBM PC-1

The PC-1 was IBM's first Personal Computer. It sockets 64K on the mother board, was supplied with DOS 1.1, and has five expansion slots. You can distinguish it from other models by the following label on the left edge of the mother board: "16KB-64KB CPU".

Settings for Baby Blue II's SW1 and SW3 are found in Table 1. SW1 assigns a Page number to Bank IV. SW3 assigns Bank I, but remember that Banks II and III, if enabled, automatically occupy the next two Pages.

The column marked "Existing Memory" is a calculation-saver, assuming that you want to add Baby Blue II's RAM to System Memory. Just find out how many Kilobytes you already have ("Existing Memory") - the corresponding switch setting assigns your RAM to the first available Page, right at the top of currently installed memory.

If you are not adding RAM to system memory, or if you have already selected a Page, find your setting by Page number in the rightmost column.

Now find the two DIP switch blocks on the PC's mother board, faintly labelled "SW1" and "SW2" - SW2 is to the right, almost dead center in the System Unit, and SW1 is to the left.

Examine SW1, switches 3 and 4 - they should both be OFF. If they are not, go no farther: this indicates that you do not have 64 Kilobytes of memory installed on the system board and your system does not meet the minimum system requirements for using Baby Blue II (are you sure you're looking at the right switches?).

You must set SW2 to reflect any change in the amount of System Memory. Calculate the new figure, including additions, but don't count RAM you intend to exclude (See 2.23). Find this figure under "Total Memory" in Table 1, and set SW2 accordingly.

The PC-1 addresses a maximum of 544K as System Memory - the break occurs at the middle of Page 8. Any RAM mapped above that point simply won't appear as system memory, even when you dutifully follow our instructions.

RESERVED PAGES (Hex): A, B, F.

Table 2-2: IBM PC-1: Mother Board SW2 Settings

New Total Memory	1	2	3	4	5	6	7	8
128K	ON	OFF	ON	ON	OFF	OFF	OFF	OFF
192K	ON	ON	OFF	ON	OFF	OFF	OFF	OFF
256K	ON	OFF	OFF	ON	OFF	OFF	OFF	OFF
320K	ON	ON	ON	OFF	OFF	OFF	OFF	OFF
384K	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF
448K	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF
512K	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
544K+	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Switches 5,6,7 and 8 are always OFF.

2.43 IBM PC-2

This is the second-generation IBM PC, featuring a mother board which can hold up to 256K of RAM before it is necessary to install additional memory boards. You can distinguish it from earlier models by the following label on the left edge of the mother board: "64KB-256KB CPU". It's not a PC/XT because it has only five expansion slots to the PC/XT's eight, and does not come with an internal hard disk drive.

Settings for Baby Blue II's SW1 and SW3 are found in Table 1. SW1 assigns a Page number to Bank IV. SW3 assigns Bank I, but remember that Banks II and III, if enabled, automatically occupy the next two Pages.

The column marked "Existing Memory" is a calculation-saver, assuming that you want to add Baby Blue II's RAM to System Memory. Just find out how many Kilobytes you already have ("Existing Memory") - the corresponding switch setting assigns your RAM to the first available Page, right at the top of currently installed memory.

If you are not adding RAM to system memory, or if you have already selected a Page, find your settings by Page number in the rightmost column.

The PC-2 reserves the first four Pages of system memory (Ø,1,2 and 3), or the first 256K, for memory installed directly on the mother board. Therefore, your first available Page is 4. If your calculated figure for "Existing Memory" is less than 256K, it means that the mother board is not fully populated and your machine won't recognize additional system memory. You can still use Bank IV, since we recommend excluding it in any case, but the RAM Module won't work until you finish populating your mother board.

Find the DIP switch blocks on the PC's mother board, faintly labelled "SW1" and "SW2" - SW1 is to the left, and SW2 is to the right, almost dead center in the System Unit. When you change the amount of system memory, you must reset SW2. Calculate the new figure, including additions, but don't count RAM you intend to exclude (See 2.23). Find this figure in Table 3, under "Total Memory", and set SW2 accordingly.

Do not assign any of Baby Blue II's RAM to the Reserved Pages listed below. Be sure that Banks II and III, which automatically occupy the two Pages above Bank I, do not overlap the Reserved areas.

RESERVED PAGES (Hex): 1, 2, 3, A, B, C, F.

Table 2-3: IBM PC-2: Mother Board SW2 Settings

Total Memory	1	2	3	4	5	6	7	8	
64K	[]	[]	[]	[]	[]				ON
						[]	[]	[]	↑
128K	[]		[]	[]	[]				ON
		[]				[]	[]	[]	↑
192K	[]	[]		[]	[]				ON
			[]			[]	[]	[]	↑
256K	[]			[]	[]				ON
		[]	[]			[]	[]	[]	↑
320K	[]	[]	[]		[]				ON
				[]		[]	[]	[]	↑
384K	[]		[]		[]				ON
		[]		[]		[]	[]	[]	↑
448K	[]	[]			[]				ON
			[]	[]		[]	[]	[]	↑
512K	[]				[]				ON
		[]	[]	[]		[]	[]	[]	↑
544K					[]				ON
	[]	[]	[]	[]		[]	[]	[]	↑
640K	[]		[]	[]					ON
		[]			[]	[]	[]	[]	↑

Switches 6,7 and 8 are always OFF.

2.44 IBM PC/XT

This is the new IBM super-PC, which comes standard with an IBM-installed Winchester hard disk, DOS 2.Ø, 128 Kilobytes of memory, and eight expansion slots. The mother board has sockets to receive 256K of memory, and the left edge is marked: "64KB-256KB SYSTEM BOARD".

Settings for Baby Blue II's SW1 and SW3 are found in Table 1. SW1 assigns a Page number to Bank IV. SW3 assigns Bank I, but remember that Banks II and III, if enabled, automatically occupy the next two Pages.

The column marked "Existing Memory" is a calculation-saver, assuming that you want to add Baby Blue II's RAM to System Memory. Just find out how many Kilobytes you already have ("Existing Memory") - the corresponding switch setting assigns your RAM to the first available Page, right at the top of currently installed memory.

If you are not adding RAM to system memory, or if you have already selected a Page, find your settings by Page number in the rightmost column.

Do _not_ change any switches on the mother board. The PC/XT uses software, not switches, to determine the amount of memory in the expansion slots. The switches on the mother board reflect only the amount of memory directly installed there and have nothing to do with Baby Blue II. Also, although the XT sockets 256K on the mother board, this region of memory is not strictly reserved. Even if the mother board is not fully populated, you can map Baby Blue into the Page just above presently installed system memory, and the system will recognize the additional 64K as general purpose memory.

Do not assign any of Baby Blue II's RAM to the Reserved Pages listed below. Be sure that Banks II and III, which automatically occupy the two Pages above Bank I, do not overlap the Reserved areas.

RESERVED PAGES (Hex): A, B, C, F.

2.45 OTHER PC'S

Settings for Baby Blue II's SW1 and SW3 are found in Table 1. SW1 assigns a Page number to Bank IV. SW3 assigns Bank I, but remember that Banks II and III, if enabled, automatically occupy the next two Pages.

The column marked "Existing Memory" is a calculation-saver, assuming that you want to add Baby Blue II's RAM to System Memory. Just find out how many Kilobytes you already have ("Existing Memory") - the corresponding switch setting assigns your RAM to the first available Page, right at the top of currently installed memory.

If you are not adding RAM to system memory, or if you have already selected a Page, find your settings by Page number in the rightmost column.

Find your machine in the following sections for detailed information on configuring Baby Blue II. Refer to the manufacturer's documentation for general instructions on installing new memory, particular with respect to switches on the mother board.

2.451 Columbia

The first two Pages (Ø, 1) are reserved for memory to be installed on the mother board; some models may reserve the first four Pages (Ø - 3). All models accept system memory only in blocks of 128 Kilobytes, or two Pages at a time. You can add 128K to system memory, or 256K, but not 192K.

It's particularly desirable to use the standard configuration for Baby Blue II's SW1 (Co-Processor Module), because various conflicts have arisen when Bank IV is installed in low memory, especially in Columbias equipped with hard disks. Where conflicts arise, Bank IV should be installed at Page 9 or above.

RESERVED PAGES (Hex):

1, B, F.

2.452 Compaq

The Compaq comes with 128 Kilobytes installed on the motherboard. No hidden conflicts have been reported.

RESERVED PAGES (Hex):

1, A, B, F.

2.453 Corona

One of Corona's major attractions is the large capacity of the mother board - there are sockets for 512 Kilobytes of memory. In addition, the floppy disk controller, monochrome monitor adapter, a parallel port, and a serial port are all included on the mother board, leaving plenty of open expansion sockets. However, you may install expansion memory boards in the first 512K of system memory.

The Corona accepts system memory only in blocks of 128 Kilobytes, or two Pages at a time. You can add 128K, or 256K, but not 192K.

Don't install Bank IV (Co-Processor Memory, SW1) below Page 8, or conflicts will arise. The standard configuration is preferable.

RESERVED PAGES (Hex):

A, B, F.

2.454 Eagle PC

Don't install Bank IV (Co-Processor Memory, SW1) below Page 8, or conflicts will arise. The standard configuration.

Do not confuse the Eagle PC with the Eagle 16ØØ. Baby Blue II will not work properly in the Eagle 16ØØ.

RESERVED PAGES:

A, B, F.

NOTES:

3. SYSTEM UTILITIES

3.1 CLOCK

Baby Blue II's Real-time Clock/Calendar keeps constant track of the time and date, even when your computer is turned off.

Find the file CLOCK.EXE, on your Baby Blue II distribution diskette, and copy it to your system disk (never work directly from your original Microlog diskettes, and be sure you keep them in a safe place). Use this file to control the clock, with the following commands:

CLOCK :	Sets the system clock according to Baby Blue II's clock, so you don't have to type in the time and date.
CLOCK S :	"Set" Baby Blue II's clock from the keyboard.
CLOCK R :	"Read" Baby Blue II's clock and display the contents on your screen.
CLOCK W :	"Wait" to autostart at a preset time - used for unattended operation. <Esc> Aborts
CLOCK H :	"Help" - displays all CLOCK commands.

3.11 SETTING THE TIME

The first time you use Baby Blue II's clock, you'll have to set it. With CLOCK.EXE in the default drive, type:

```
CLOCK S <CR>
```

At the prompt, enter the month, date and year, using either "-" or "/" as delimiters, e.g.:

```
8-1-83 <CR>       or       8/1/83 <CR>
```

Now enter hours and minutes, using ":", "-", or "/" as delimiters, e.g.:

```
15:2Ø <CR>  or  15-2Ø <CR>  or  15/2Ø <CR>
```

It's a twenty-four hour clock, so "15" means "3:ØØ P.M."; "2Ø" means 2Ø minutes, so the above example reads, "3:2Ø P.M.". The clock also keeps track of seconds, but they are automatically set to zero when you enter the time.

From now on, to display Baby Blue II's current time on the screen, type:

```
CLOCK R <CR>
```

3.12 SETTING THE SYSTEM CLOCK

Various programs and utilities rely on the DOS system clock, which is not the same as Baby Blue II's clock. To synchronize the DOS clock with Baby Blue II's clock, type:

```
          CLOCK  <CR>
```

The system clock must be reset every time you boot your machine. If you want to automatically set the system clock, create an AUTOEXEC.BAT file on your system disk as follows:

```
          COPY CON: AUTOEXEC.BAT <CR>
          CLOCK <CR>
          <CTL-Z>
```

Booting your machine from this disk will automatically execute the CLOCK command in the AUTOEXEC file, setting the system clock.

- If you are already using an AUTOEXEC.BAT file, be sure to include all commands from it in your new file (there are other methods for creating and editing .BAT files - see your DOS manual for details).

3.13 AUTOSTART AT A PRESET TIME

The "W" command causes CLOCK to suspend system operation until a preset time has been reached. At the appointed time, CLOCK returns control to DOS. If you put the Wait command in a .BAT file, the system will execute the next command in the file at the specified time.

Should you need to return to DOS before the preset time is reached, pressing <Esc> will abort the Wait.

3.131 Absolute Mode

The following command:

```
     CLOCK W 9:15: <CR>
```

causes CLOCK to Wait until 9:15 A.M., today. You can also specify a date - the trick is to use "/" as the delimiter between month, day and year. For example:

```
     CLOCK W 5/18/84  9:15:20 <CR>
```

means "Wait until quarter after nine A.M. on May 18, 1984, plus twenty seconds." In this example, any delimiter other than "/" would cause CLOCK to interpreted the date as a time. Since "84" is not a valid number of seconds, you'll get an error message.

If you specify a time already past, CLOCK returns immediately to DOS.

3.132 Relative Mode

You can also tell clock to Wait for a given amount of time by inserting a "+" before the time. For example:

```
CLOCK W + 380::13  <CR>
```

means "Wait three hundred and eighty hours and thirteen seconds, starting now." You can't enter days or weeks in this mode, only hours, minutes and seconds. Be sure to leave a space before and after the "+".

In relative mode, CLOCK calculates a time and date to wait for, assuming the length of a month to be the length of the current month. This means that if the waiting time spans two month boundaries, the calculation may be in error. For this reason, you should limit yourself to a maximum of 672 hours (twenty-eight days); for longer Waits, use absolute mode.

3.14 ERRORS

CLOCK returns two possible errors. If it understands your command, but can't interpret the time or date, it says:

```
Invalid time   or   Invalid date
```

If the command itself is unrecognizable, e.g.:

```
CLOCK Q <CR>
```

you'll get the appropriate response:

```
huh?
```

3.2 PRINT BUFFER/SPOOLER

You must have PC-DOS version 2.Ø in order to run the Print Buffer/Spooler. It is actually two programs, one for Baby Blue II's parallel port, and one for the serial ports; you put them on your system disk so that they are installed to your operating system each time you boot your machine. Now all system print output is sent first to Baby Blue II, which takes care of sending it on to the printer.

With the Spooler installed, your system dumps its output very quickly - much more rapidly than the printer itself can accept characters - and as soon as it's done, you can resume normal operation. How much time you save depends on the speed of the program which is sending the characters.

Even while accepting characters, Baby Blue II sends them on to the printer - the first characters appear to have been sent direct. It's only after you're back at work and your document continues to print, and print, and print - that you notice something different going on.

This works for parallel or serial printers connected to Baby Blue II's ports - up to three printers simultaneously. You control spooling to each port selectively, or all at once, using simple commands.

Since Baby Blue II buffers 63 Kilobytes of text, if you print a longer file you'll have to wait until only 63K is left to print. You don't have to wait for a file to stop printing in order to send another - the Spoolers will accept data as soon as there is space available in the buffer. When operating more than one printer at the same time, buffer space is dynamically allocated between the ports on a first-come, first-serve basis - all ports together still buffer a maximum of 63K, but any individual port may use as much of the buffer as is available, up to the maximum of 63K.

3.21 INSTALLATION

The Spooler programs install as "device drivers" under DOS 2.Ø. You'll need two files:

```
SER.SYS
PRL.SYS
```

Copy these files into the root directory of your boot (system) disk and include them with DEVICE commands in your configuration file (See Appendix, "DOS 2.Ø Configuration File"). To install spooling for all three of Baby Blue II's ports, your CONFIG.SYS would contain the commands:

```
DEVICE=SER.SYS
DEVICE=PRL.SYS
```

Also copy the Baby Blue II file:

```
SPOOL.EXE
```

You don't need this file for the installation, but you'll use it to control the Spooler later on.

When you make the Spooler part of your system, Baby Blue II's parallel port becomes LPT1, no matter what its normal assignment; any other parallel ports become LPT2 or LPT3, in address order. This is done so that whatever state the system is in, normal print output always comes out the same port. Even if you turn the Spooler OFF, as explained below, LPT1 is still Baby Blue II's parallel port.

Remember that if the Spooler is not installed at boot-time, LPT1 reverts to its normal assignment, which may not be Baby Blue II's port. For example, if you have an IBM monochrome adapter, it's parallel port is normally LPT1, and Baby Blue II's port is LPT2; the Spooler reverses these assignments: the IBM port becomes LPT2, and Baby Blue II's port becomes LPT1.

If you've run MODE to accomodate a serial printer, your characters come out Baby Blue II's COM1 or COM2. Some programs allow you to send characters to LPT2 or LPT3. This output will not be spooled.

The Spooler adds the following names to the DOS list of devices: "PRL" for Baby Blue II's parallel port, "SER1" for the primary serial port (at the mounting bracket), and "SER2" for the secondary serial port (ribbon cable from header J2). The new devices obey the same rules as standard devices like "LPT1:" and "CON". For example, the following syntax is valid for spooling a file through the parallel port:

```
COPY filespec PRL:
```

With the Spooler installed, output meant for COM1 goes to SER1, output for COM2 goes to SER2, and output for LPT1 (or the PRN device) goes to PRL.

3.22 SPOOLER COMMANDS

Control the Spooler with SPOOL.COM, as follows:

```
c:SPOOL command  <CR>
```

where "c:" is the drive containing SPOOL.COM, and "command" can be any one of the following:

OFF :	Disables the Spooler, directing print output to a system port. Any text currently in the buffer is lost.
ON :	Re-enables the Spooler, directing print output to BabyBlue's port(s).
ABORT :	Aborts the current print - the printer stops and any text remaining in the buffer is lost.
PAUSE :	The printer stops immediately, but the contents of the buffer remain intact, so the document can be completed.
RESUME :	Resumes printing after a PAUSE.

You may enter all the letters of a command, or only the first two. For example,

```
          SPOOL ABORT <CR>
```

and

```
          SPOOL AB <CR>
```

will both abort printing.

SPOOL OFF redirects print output directly to the ports, without going though the Co-Processor. The text appears at the same printer, but it's handled in the foreground, so you have to wait for printing to finish before resuming normal operation.

SPOOL ON is used to re-enable the Spooler after using SPOOL OFF. The installed Spooler is enabled immediately on bootup, so there's no need to manually turn it ON at that time.

A second form of the SPOOL command is used to selectively control the spoolers for the individual ports:

```
          d:SPOOL dev: command  <CR>
```

where "dev:" may be "SER1:", "SER2:", or "PRL:". When SPOOL is used in this way, only the specified port is affected. For example,

```
          d:SPOOL PRL: ABORT  <CR>
```

aborts the current print and clears the buffer of text intended for the parallel port only. Spooling continues unabated through the serial ports, immediately taking advantage of the newly freed buffer space, if necessary.

3.23 MODE AND THE SPOOLER

MODE works in conjunction with the Spooler, but shows certain peculiarities. The first use of MODE after booting up will turn all spooling "OFF"; subsequent MODEs won't, but you may want to follow them all with SPOOL ON, just to be sure.

The MODE parameters for number of columns and lines per inch are passed correctly to the Spooler, but the screen will return a "Printer Fault" message - ignore it. Real printer faults won't be displayed, because your system can't see "through" the Spooler to the printer itself - indeed, once Baby Blue II has accepted the output file, your system thinks the printer has stopped. However, the printer will complain loudly when there's a problem, and the possible faults are highly visible (e.g. out of paper, no ribbon, etc.).

3.24 CP/M PROGRAMS WITH THE SPOOLER

The Spooler gets its speed by using Co-Processor memory to buffer print output. If the Z-80 is using Bank IV to run a program (e.g. STEP), the buffer is not available, so the Spooler is effectively OFF. For this reason, a program running on Baby Blue II doesn't print any faster with the Spooler installed.

For the same reason, you can't print in the background while running a program on Baby Blue II. As soon as you invoke the program it will take over Co-Processor memory, wiping out the buffer and aborting the print.

3.25 MESSAGES AND ERRORS

SPOOL always tells you when it executes a command, for example:

```
'PAUSE' command sent to device 'PRL:'
```

If you've commanded all devices at once, you'll get three lines in response, one for each possible device (PRL, SER1, SER2). If any device has not been installed by your CONFIG.SYS file, SPOOL mentions this as well:

```
Unknown device 'SER1:'
```

You will only consider this to be an error if you had meant to install the device mentioned. If you make a mistake entering the command itself, SPOOL responds:

```
Unknown command 'XXXXX'
```

where 'XXXXX' is the command that SPOOL didn't understand.

3.3 RAMDISK

You must have PC-DOS version 2.Ø to use Baby Blue II's RAMdisk facility. Copy the file:

```
        RAMDISK.COM
```

from your Microlog distribution diskette and put the original away.

To create a RAMdisk to your specifications, type:

```
    c:RAMDISK
```

Where "c:" is the name of the drive containing RAMDISK.COM. RAMDISK asks how much memory you want to set aside for your pseudo-disk:

```
               Enter size of RAMdisk in kilobytes

                          >>
```

Type in the number of Kilobytes (eg., 2ØØ), then press "Retrn". RAMDISK responds:

```
        Enter filename for your new RAMdisk device driver
                    (Default = RAMDISK.SYS)

                          >>
```

You can simply press "Retrn", automatically creating a new file called "RAMDISK.SYS". However, if you have an existing RAMdisk which you don't want to destroy, you must supply a different filename for the new RAMdisk. It can be any legal filename, but try to use the extension ".SYS", to identify it as a device driver. In this way, you can build as many different RAMdisks as you like, stored under different filenames - the only limitation is the amount of available memory.

Remember that you must leave sufficient memory for normal system purposes. Usually it's desirable to retain about two Pages, or 128 Kilobytes for system memory. You'll have to experiment: some applications may require less, some more, but it's easy to change the size of your RAMdisk(s) by running RAMDISK again.

Simply creating a RAMdisk device driver doesn't mean you can start putting files on your new disk. The driver must be present in the root directory of your system disk at boot time, and it's name must be included as a device in your configuration file (See Appendix: "DOS 2.Ø Configuration File"). For example:

```
        DEVICE=RAMDISK.SYS
```

If you change the size of an existing RAMdisk, you must "reboot" your machine (by pressing Ctrl-Alt-Del) before the change will take effect.

DOS automatically assigns one-letter drive designations in the same order as the DEVICE commands appear in the configuration file. The permanent system disk drives are named first (a:,b:...), and then the installed devices. If you have two floppy drives, and your configuration file reads:

```
DEVICE=RAMDISK1.SYS
DEVICE=RAMDISK2.SYS
```

The floppies are drives "a:" and "b:", so RAMDISK1 becomes drive "c:", and RAMDISK2 becomes drive "d:".

If your RAMdisk is so large that it exceeds the limits of available memory, you'll get a "bad or missing device" message at boot time, for example:

```
Bad or missing RAMDISK.SYS
```

and the driver won't be installed. If you try use the RAMdisk anyway, the system will respond:

```
Invalid drive specification
```

NOTES:

4. COMMUNICATIONS

4.1 SMART TERMINAL EMULATOR PACKAGE (STEP)

Microlog's Smart Terminal Emulator Package makes your Personal Computer "at home" in virtually any asynchronous communications environment. STEP offers the full power of a true video display terminal: cursor control, graphics, and other special features.

STEP fully exploits Baby Blue II's comprehensive hardware, integrating terminal, communications, and microcomputer functions. Control port initialization and system facilities from within STEP, and save your setups in the "Sessions Menu" for effortless flexibility in passing from one environment to another. Use KEYFIX to program extended command sequences (up to 8Ø characters) for semiautomatic operation at the touch of a function key.

4.2 THE SESSIONS CONCEPT

All STEP activities revolve around the Session, a task-oriented, interactive control structure which extracts a library of proven applications from your hands-on experience. To build a Session, you get on line with a remote, and gradually customize your terminal for that particular environment. You then tell STEP to record the optimized configuration under a name of your choosing; STEP enters the name in the Sessions Menu, so that you can recreate the entire setup with a single keystroke.

Anyone who has tried to work out a communications interface will appreciate the integrity of the Session. All relevant parameters are recorded in the Session definition, including port initialization, terminal type, and a series of "pathway toggles" which direct the flow of data to system peripherals (e.g., Screen ON/OFF, Echo ON/OFF, Full/Half Duplex). You can display all parameters, or record them to disk, at will.

There are pathways which redirect communications to your printer, and to or from disk files. Information can be manipulated locally, limiting communications to the time necessary for transmission only.

STEP always comes up defined as the last Session used, keeping frequently used configurations readily at hand. You change configurations by invoking the Sessions Menu and choosing a new option.

4.3 OPERATING FUNDAMENTALS

4.31 BEGIN: COMMAND MODE

Before beginning, make a working copy of your STEP files, and put the original away for safekeeping. Your working copy must include at least four files:

```
        STEP.COM
        SESSION.DAT
        HELP.FUL
        TV950.TRM (and/or any other .TRM files you will use)
```

Now insert the working diskette in the logged-in drive and type:

```
        STEP <CR>
```

The titles appear, then

```
        The current session is:
```

followed by the Session name. This will always be the name of the last Session used. You are now in Command Mode, signified by the "star" prompt

```
        *
```

which appears after the Session name. When you see this prompt, you may invoke any of the STEP commands described in this chapter.

When you invoke STEP, it must find SESSIONS.DAT, HELP.FUL, and at least one TRM file on disk. It looks first on the default drive, and then on drive A:. If the required files are not in either place, STEP displays an error message and returns control to DOS.

To exit Command Mode and return to DOS, type:

```
        *EX <CR>
```

You should always exit STEP with this command - it resets Baby Blue II's hardware, and preserves the integrity of any saved data. Should you exit in any other way (e.g. CTRL-ALT-DEL), you will probably have to cycle power before using Baby Blue II's again, and data may be lost.

If at any point you can't remember a command, type

```
        *HE <CR>
```

which will bring up the HELP file, listing all STEP commands.

4.32 CONNECTING TO THE REMOTE: TERMINAL MODE

No actual communication takes place in Command Mode - all operations are local to your computer, preparing it to emulate some particular device. You link up with a remote using the following command:

```
         *OC <CR>  (Online Connect)
```

When the link is established, Command Mode terminates and the word "CONNECT" appears on your screen. You are now in Terminal Mode: your computer effectively becomes a terminal, responding exactly as would a device of the type specified for this Session. It will not respond to STEP commands until you re-enter Command Mode, but you can do so temporarily without losing continuity in Terminal Mode.

The remote may be any device connected to one of Baby Blue II's serial ports: a larger computer, another microcomputer, or an external modem.

4.33 RETURNING TO COMMAND MODE: THE ESCAPE SEQUENCE

Although STEP will not accept Command Mode sequences while in Terminal Mode, there is one command you can give it: you can tell it to let you out. Strictly speaking, this is not a command, but an "Escape Sequence" which you use when you want to get back to Command Mode. You have to remember it, because we can't HElp you with reminders until you are in Command Mode.

Invoke the Escape Sequence by pressing CTRL-\ and CTRL-A in succession, or simply by pressing function key F1Ø, which has been KEYFIXed to send these keystrokes automatically. Remember, you use this Sequence in Terminal Mode only. If you try to use it in Command Mode, STEP won't know what you're talking about.

The Escape Sequence won't appear as characters on your screen; instead, you'll see the star prompt ("*"), indicating that you have returned to Command Mode. You have not, however, lost your connection - the remote thinks you're still there, but that you're not sending it anything. You can issue any number of commands to STEP, and then go back into Terminal Mode by typing

```
         *O <CR>
```

which tells STEP to "go back on line in present state", that is, to pick up where you left off. You can use this procedure whenever you need access to STEP commands in the middle of a Session, for example when:

- preparing to send or receive data to/from a disk file, or to a printer.

- changing a characteristic of your terminal (e.g. Screen Enable).

NOTE: STEP will always "trap" the first CTRL-\ you type, thinking that it may be part of the Escape Sequence. If you want to send a CTRL-\ as part of your message to the remote, you must type it twice.

4.34 DISCONNECTING FROM THE REMOTE

You can also return to Command Mode by disconnecting from the remote, using the following command:

```
        *OD <CR>  (Online, Disconnect)
```

Unlike the Escape Sequence, this command breaks the connection between your terminal and the remote device; reestablishing the link requires an *OC, not just an *O. What you find then depends on the remote. Some will still be waiting, just as if you had never left; others will require restarting. For example, many modems automatically "hang up" the telephone line when the terminal disconnects, forcing you to redial.

4.4 COMMAND SUMMARY

Command Mode functions fall into three groups:

1) Operating Functions: enter and leave Terminal Mode, exit program, get help, and repeat last command. (Section 3.31)

2) Session Definition/Configuration: terminal type, transmission rates, port configuration, modem state, capabilities/devices enabled. Includes command to record complete setups under multiple "Session" names for future use. (Section 3.32)

3) Pathway Toggles: Redirection of the data stream - screen enable, full duplex, echo.

4) Data Storage (Send/Receive): direct incoming data stream (ASCII text) to a printer or disk file. Send outgoing data from a disk file. (Section 3.33)

4.41 OPERATING FUNCTIONS

4.411 *OC : go Online, Connect

This is the same as an "O" command, except that it also executes a Connect sequence, resetting all Session parameters. It is usually used when going initially on line through the RS-232 port; you could use it to reset all parameters in the middle of a Session, but you might lose your initial connection. Note that resetting Session parameters is not the same as resetting the remote - in many cases, it is possible to execute an *OC, or even exit to DOS, bring up STEP again, and return to Terminal Mode, without changing the state of the remote - so don't forget to log off.

4.412 *O : go Online in present state.

Use this command to return to Terminal Mode after exiting temporarily to Command Mode. *O does not execute a Connect sequence: Session parameters are not reset by this command, and you will not lose your original connection. If a CONNECT has not already been established, *O will return the error, "NO CARRIER".

4.413 F1Ø (CTRL-\ CTRL-A) : Exit Terminal Mode

Strictly speaking, this is not a command, but the "Escape Sequence" which you use when you're in Terminal Mode and you want to get back temporarily to Command Mode. The sequence is generated by pressing CTRL-\ and CTRL-A in succession, or simply by pressing function key F1Ø, which has been KEYFIXed to send these keystrokes automatically.

4.414 *OD : Online, Disconnect

This is the opposite of the *OC command; it returns you to Command Mode, but is different from the Escape Sequence because it breaks the connection to the remote. An *OC is required to re-establish the link. Technically, *OD turns off the serial port's DTR (Data Terminal Ready) line.

4.415 *EX : EXit to operating system

This returns you to DOS. Always exit using this command (or *CH, below) - if you don't, you will have to cycle power before using Baby Blue II again, and data may be lost from disk files you have opened.

4.416 *CHd:filename : CHain

"Filename" must be the name of a COM or EXE file one the named drive "d:". This command exits STEP, but instead of returning you to DOS, it transfers control to the specified program. Upon exit from the called program, control returns to DOS, not to STEP. Use this program to configure a port under STEP instead of MODE, before invoking a program which must use that port.

4.417 */ : repeat last command

This command re-executes the entire previous command line.

4.418 *HE : HElp

This command brings up the HELP file, which is a brief listing of STEP commands. Use it for quick reference when you can't remember what comes next.

4.42 SESSION DEFINITION (CONFIGURATION)

These commands allow you to define all parameters relevant to a specific application. The configuration can be transitory, or you may elect to save it under a name of your choosing (e.g., "Compuserve", "Talk to Arthur", "Electronic Mail"), in which case it will be preserved as a selection in the Sessions Menu.

The parameters listed here make up the Session definition; all of them are recorded to disk when you Write the definition. The recorded parameters take effect for a particular Session whenever you execute a Connect (*OC). This means that if you change any parameters, you must Write them to disk before executing a Connect, or they will be erased. You can of course change parameters after a Connect by using the Escape Sequence and *O to enter and leave Terminal Mode (See 3.316).

4.421 *CO : Configure

Displays the state of all parameters for the presently invoked Session, and menu options for resetting them. The new settings become active immediately, but are not recorded for future use until you Write the Session definition (*W). The requirements of the remote determine the proper choice for all parameters.

Terminal: Assembles a menu of available terminal emulators by inspecting the currently logged diskette for .TRM files. STEP comes standard with emulators for the Televideo 950 (TV950.TRM) and the Hazeltine 1500 (HT1500.TRM) and many others are available. The terminal emulator enables your computer to interpret display commands from a remote, such as "clear screen" or "position cursor".

Baud Rate: This is the transmission rate, generally equivalent to the number of bits per second, for the Communications Port, as defined below. This parameter does not affect the baud rate of the other serial port, which may be initialized at the system level to support a printer at a different baud rate. Although fundamentally defined by the requirements of the remote device, your choice of Baud rate may depend on several factors, including the capacity of your machine and the remote to capture information, whether you are transmitting through the modem or through the RS-232 port, and the characteristics of the line itself. You must always use the same Baud rate as the remote.

Stop Bits: These are used as delimiters between characters in asynchronous transmissions. If you choose more than one stop bit, the machine decides to give you one and a half or two, depending on the number of character bits.

Character Bits: The number of bits used to encode a character, optionally five to eight. Typically seven with parity checking, or eight without.

Parity: Odd, Even, or none. This is a basic error checking mechanism which may or may not be implemented.

Communications Port: You may choose COM1, which is the connector at Baby Blue II's mounting bracket, or COM2, which is the ribbon cable plugged into header J2. Only the port chosen is initialized by STEP. The remaining port may be configured completely independently at the system level to drive a printer or other device.

4.422 *SP : Enter Sessions Menu

Displays name of current Session, and posts a menu of previously defined Sessions. You may choose a different Session, or return to the command prompt with no change. None of the new Session parameters take effect until a Connect sequence is executed, (*OC).

4.423 *SA : Session Add

Prompts you for a new Session name, and makes this the current Session. The new Session's name appears in the Sessions Menu, and the definition is automatically Written to disk - it is a "snapshot" of your configuration at the moment you type "SA", which means that you can choose an existing Session to act as a template for most of your variables.

4.424 *W : Write Session definition

Records all parameters to disk, indexed under the name of the current Session. Your changes are not saved for future use until you execute a Write - they will be lost if you exit STEP or execute a Connect sequence.

4.43 PATHWAY TOGGLES

A set of software "switches" which enable the data stream to reach different parts of your system (1 = ON, Ø = OFF). The states of all toggles listed here become part of the recorded Session definition when you execute an *W (Write).

4.431 *SC[Ø/1] : SCreen enable

Normally ON. All incoming data is displayed on your screen. You might want to turn off the screen if you are losing characters when receiving data to disk at high transmission rates - although not as slow as printing, the character translation to the screen does take some time, and may reduce your ability to capture incoming data at very high speeds.

4.432 *F[Ø/1] : Full duplex

Service/host dependent. When ON, all characters displayed are sent from the remote, meaning that your keystrokes are displayed not as you send them, but as you receive their "echoes" from the remote. This is possible in full duplex mode because the remote can send at the same time you do, and has the advantage that any characters lost or garbled in transmission will appear that way on your display. With full duplex OFF, your keystrokes are sent directly to the screen.

4.433 *E[Ø/1] : Echo

Normally OFF. When enabled, incoming characters will be retransmitted, or "echoed" back to the remote. Usually, echo is a function of the host - the complement of the Full Duplex function.

4.44 DATA STORAGE - SEND/RECEIVE

These are the commands which enable local storage of data. Since they are normally invoked after entering Terminal Mode, the toggles are not recorded in the Session definition. When you first invoke STEP, all toggles are OFF ("Ø"); afterwards they remain as you set them, even after a Connect.

To record incoming data on your printer, you simply turn it on and off using *PE. Similarly, to send and receive to/from a disk file, you use *RE or *SE to enable the transfer of data, but in addition you must open and close the file. Here's a typical sequence:

```
<CTRL-\><CTRL-A> or <F1Ø> ; exit Terminal Mode
*RD/SDfilespec            ; open file
*RE1/SE1                  ; enable data transfer
*O                        ; go back on-line in present state
:                         ; (transfer data)
:                         ;
<F1Ø>                     ; exit Terminal Mode
*AT[REØ/SEØ]              ; stop transfer and close file
```

Note that no data actually moves until you go back on line (*O). Also note that you must prepare the remote to send or receive data by issuing the proper commands in Terminal Mode.

These commands are primarily intended to limit expensive telephone connect time by emulating a fast typist, not for universal file transfer. They will work best at low (3ØØ-12ØØ Baud) transmission rates, and where the remote has a large input buffer to receive characters. It is possible to transmit large files at higher rates, but some tinkering may be necessary to avoid losing characters.

4.441 *RDfilespec : Receive - Designate file

Creates a new file under the indicated file specification to receive incoming data. Data is not actually written to the file until an *RE1 (Receive Enable) is executed.

> NOTE: if the same filename is already present on the indicated disk, the old file is overwritten without warning, destroying the original contents.

4.442 *RE[Ø/1] : Receive - Enable

*RE1 (Receive Enable ON) directs all incoming data to the disk file defined by the last *RD. *REØ turns Receive Enable OFF, and closes the file - you must close the file with *REØ before exiting from STEP, or you will lose the recorded data. After you close the file, you can still append further data with another *RE1 - this reopens the file at its end, preserving the original contents. Once you exit the Session, however, you will need an *RD to begin receiving again, so you could not append further data to this same file without erasing previously stored data.

4.443 *SDfilespec : Send - Designate file

Opens an existing disk file from which data is to be sent. No data will actually be sent until an *SE1 is issued.

4.444 *SE[Ø/1] : Send - Enable

*SE1 enables data to be sent from the last file designated with *SD. Transmission begins as soon as you execute an *O. *SEØ disables sending and closes the file. No harm results from failure to close the file before exiting STEP.

4.445 *PE[Ø/1] : Printer Enable

*PE1 records all subsequent incoming data in hard copy on your MS-DOS printer. PEØ stops further data stream from reaching the printer. This toggle is not written with the Session definition to disk.

4.446 *Tfilespec : Type file

This command displays the contents of the named file on the screen, twenty lines at a time. The display pauses between screens until you tell it to continue. Use this command to examine the contents of files you are sending or receiving.

4.5 BSTAM: FILE TRANSFER

For transfer of COM and other binary files, and efficient transmission of large ASCII files, Baby Blue II comes with BSTAM (Byrom Software Telecommunications Access Method). This is a sophisticated communications utility capable of very accurate file transfer, over either the public telephone system, or over a local line at rates up to 9600 baud.

BSTAM consists of two COM files, RECEIVE.COM and TRANSMIT.COM. To send a file, type:

```
TRANSMIT filespec <CR>
```

To receive a file, you may type:

```
RECEIVE d: <CR>
```

In which case the remote computer will supply the filename, or you may specify the filename:

```
RECEIVE filespec <CR>
```

Wildcard filenames are permissible, e.g.:

```
TRANSMIT *.COM <CR>
```

You must establish the link to the remote computer before invoking BSTAM. You can use MODE to initialize the port, but you may find it easier to use STEP. Create a Session with the proper parameters, then go on line and perform any necessary chores, such as dialing through a modem. Finally, use the Chain command to invoke BSTAM, for example:

```
*CHTRANSMIT filespec <CR>
```

Refer to the enclosed BSTAM manual for in-depth documentation - operating instructions begin on page 9. Disregard the discussion of hardware requirements and installation - BSTAM comes fully configured for Baby Blue II.

STEP

NOTES:

5. RUNNING CP/M-80 PROGRAMS

5.1 GETTING STARTED

5.11 DOS COMMANDS

In normal operation Baby Blue II is meant to be completely "transparent", which means that it fits seamlessly into your present operating system. When you run a CP/M-80 program, it will appear to be a native PC-DOS program: you'll use the same commands and procedures, and execution speeds will be similar.

We're going to assume that you are already familiar with your operating system. You should know how to physically insert a diskette in a disk drive, "boot" your system from a diskette, and perform common file operations using the following DOS utilities:

```
FORMAT
CHKDSK
COPY
DIR
RENAME
```

If you are at all unsure of these basic procedures, practice them now before you begin, referring to the documentation which came with your computer.

5.12 THE CONVERSION UTILITIES

Although Baby Blue II is now physically installed in your computer, your operating system must be extended to run CP/M programs, using the Microlog diskette labelled "Baby Blue II Conversion Software". First, make a backup copy of the Conversion Software on a new diskette. The command:

```
COPY A:*.* B: <CR>
```

will copy all files, where the original Microlog diskette is in drive A, and the new diskette is in drive B.

Put the original Microlog diskette away for safekeeping, and take a DIRectory of the new disk. You should see the files:

```
HEADER
CONVERT.COM
BIND.COM
STRIP.COM
KEYFIX.COM
SAMPLE.CPM
```

Each of these files becomes a new command in your operating system, with the exception of HEADER, which you will notice lacks the extension "COM". They:

- Convert CP/M-8Ø programs to run on Baby Blue II under PC-DOS.

- Transfer files between selected CP/M formats and PC-DOS diskettes.

- Provide user-programmable function keys for CP/M programs.

Here is a brief description of the Baby Blue II utilities - for detailed information see "The Conversion Utilities" in the Appendix.

<u>HEADER</u>

HEADER is a large program, practically an operating system in its own right. The "meat" of the Conversion software, it is paradoxically the one utility you never command directly. It does all its talking to your computer and you are aware of it only through its effects - it makes CP/M programs run on your machine.

Before a CP/M program will run on Baby Blue II, it must have HEADER attached to it - this is called "binding" the program. Binding is carried out using either CONVERT or BIND, as outlined below.

<u>CONVERT</u>

CONVERT transfers disk files in either direction between PC-DOS and selected CP/M formats. It can copy files, and display directories of both PC-DOS and CP/M diskettes. It also automatically binds HEADER to COM files as they are written to a PC-DOS disk, and removes it when copying to a CP/M disk.

CONVERT insures that you can purchase CP/M programs in at least one format which you will be able to read. Actually, you can buy most CP/M software already on a PC-DOS diskette, in which case you won't need CONVERT at all - this is often called the "Baby Blue" format. If you must purchase a CP/M-formatted diskette, make sure that it is one of the formats which CONVERT can read.

<u>BIND</u>

Like CONVERT, BIND attaches HEADER to CP/M COM files; unlike CONVERT, it works only with PC-DOS, not CP/M diskettes. It's used when you need to attach HEADER to a CP/M program already on a PC-DOS diskette, which is the recommended way to buy software.

If BIND finds a HEADER already attached to a file, it removes it before attaching another one - this is how you update your files with a new revision of HEADER.

STRIP

STRIP is the opposite of BIND - it removes HEADER from a bound program. It's used when you want to export a program from Baby Blue II to a native CP/M system.

KEYFIX

KEYFIX allows you to define over fifty function keys for each CP/M program - a single keystroke becomes a shorthand way of entering as many as 8Ø separate characters. KEYFIX saves the definitions on disk, in the HEADER attached to each program.

SAMPLE

SAMPLE is a short CP/M-8Ø program, which is already on your PC-DOS diskette, but has not yet been bound with HEADER. All it does is post a line of text on your screen, but if you can BIND it and get it to run, you're ready to use run any CP/M program on Baby Blue II.

5.13 OPERATING FUNDAMENTALS

Let's try running the SAMPLE program. Type:

```
BIND SAMPLE <CR>
```

When the system prompt returns, take a DIRectory. You should see two SAMPLE files:

```
SAMPLE.CPM
SAMPLE.COM
```

Notice that SAMPLE.COM is much larger than SAMPLE.CPM - that's because it contains HEADER, and is ready to run. Type:

```
SAMPLE <CR>
```

The response should be a congratulatory message. SAMPLE is a very simple program, but you've just seen how to make a CP/M program run. SAMPLE is now permanently a PC-DOS COM file: even if you turn off your machine, the next time you power up you can still run the program just by typing SAMPLE.

SAMPLE illustrates Baby Blue II's special simplicity: once you bind HEADER to a CP/M program, it is effectively a PC-DOS program and will run in any computer equipped with a Baby Blue II. You can put the Conversion Software away, and there are no restrictions to PC-DOS. You use the Conversion Software only when transferring files between your system and a CP/M system.

Remember that you only bind CP/M COM files - overlays and interpreted programs which run under the control of a COM file are not bound, nor are text and data files. For example, you would not bind STEP ".OVR" files, because they run under the control of WS.COM. You wouldn't bind CBASIC ".INT" or ".BAS" files, because they run under CRUN.COM or CBAS.COM, respectively.

Don't bind Microlog-supplied applications programs such as Wordstar or BSTAM - they come preinstalled and ready to run. A number of independent vendors also package their software bound with Microlog's HEADER - you need to rebind such programs only when you receive an update of HEADER itself.

Notice that you haven't been asked to learn a separate set of commands or create a separate set of disks in order to run CP/M programs. This means that you can call native PC-DOS programs and CP/M programs from the same disk, and just as important, those programs can freely exchange data or text files. Because you continue to operate under PC-DOS, the peripherals already supported by your system will also work with Baby Blue II (e.g., printer, hard disk, etc.).

NOTE:

> There is one area in which we must depart from standard PC-DOS practice: while running a program on Baby Blue II, <u>do not attempt a "warm boot" using CTRL-ALT-DEL</u>. This will fail to properly reset Baby Blue II's Z-8Ø microprocessor, and you will have to cycle power before using the board again. You may, of course, use CTRL-ALT-DEL when you are not running a CP/M program, even with Baby Blue II physically installed.

5.2 MEDIA COMPATIBILITY: ACCESS TO CP/M DISKETTES

5.21 THE PROBLEM OF STANDARDS

Simply by entering the marketplace, IBM created a new set of standards for the manufacture of personal computers. Although open to criticism on technical grounds, the IBM PC has been invaluable as a serviceable, if somewhat arbitrary, convention through which different manufacturers can insure mutual compatibility of their products.

Incompatibilities have arisen as manufacturers strain against some of the PC's limitations, and it is always possible that IBM will someday violate or completely overturn its own standard, as it has repeatedly done in the past. In the main, however, the standard has been succesful, and in both hardware and software the owner of an 8Ø88 microprocessor-based personal computer enjoys much greater freedom of choice than was possible before the advent of the IBM PC. For example, almost all PC-DOS or MS-DOS machines can exchange 5" floppy diskettes, with little or no difficulty. This may seem natural if you have such a machine, but it represents a tremendous advance.

As a standard operating system, CP/M-8Ø permitted microcomputer software to reach a new level of maturity, since it became possible for a manufacturer to develop application programs for a broad base of otherwise heterogenous equipment. However, except for a single 8" CP/M disk format, no standard medium existed for transferring files from one manufacturer's microcomputer to another's. It was still necessary to publish the same program in a variety of machine-specific disk formats, and it was almost impossible for the average user of 5" diskettes to transfer even data or text files between different machines. Media compatibility remains one of the technically most vexing issues facing users of CP/M-8Ø applications programs.

Baby Blue II avoids the issue as much as possible by using the standard PC-DOS format, which means complete compatibility within your system and maximum flexibility in communicating with other similar systems. Most vendors now offer their CP/M software already on PC-DOS formatted diskettes, in what is often called the "Baby Blue format".

5.22 MICROLOG FILE TRANSFER UTILITIES

5.221 5" CP/M Diskettes

Baby Blue II comes with CONVERT, a utility which enables you to transfer files between PC-DOS disks and a number of popular CP/M-8Ø formats (See Appendix, under CONVERT). You can purchase software in one of these formats, but we strongly recommend obtaining PC-DOS diskettes whenever possible. You should be aware that although you will have a wider choice of available software than you normally would if you owned a CP/M-8Ø based machine, you may still run across an incompatible format, especially if you are trying to transfer files from other machines, some of which may not even be fully compatible with others of the same manufacture. Future updates may extend the list of formats available under CONVERT, but limitations in IBM's disk controller hardware make a number of formats forever problematical.

5.222 8" Diskettes

Microlog manufactures an 8" disk drive controller for the IBM PC which will format, read and write standard 8" CP/M diskettes (single-sided, single-density). This is the only standard medium for file transfer in the CP/M world, and gives you access to practically all published CP/M software. The controller supports up to four standard 8" floppy drives, and can store a maximum of 1.25 Megabytes on a double-sided, double-density PC-DOS diskette. It also supports the PC-DOS standard single-sided, single density 8" format. A separately sold utility, called REFORMATTER, provides access to the IBM 8" floppy diskette format (3741), used as a transfer medium by IBM minicomputers and mainframes.

5.23 SERIAL COMMUNICATIONS

If you must import files from a machine with an incompatible disk format, and you can set up a working serial line between the two machines, you can use BSTAM (Byrom Software Telecommunications Access Method), which comes preconfigured for your Baby Blue II. This is a sophisticated but easy to use communications program, capable of high-speed, error-free data transfer, bypassing entirely the question of media compatibility.

BSTAM will only talk to BSTAM, which means that you must acquire it separately in a suitable version for the other end of the line. This will generally be true of any program which can transmit COM or binary data files, because error-checking must be very precise, necessitating a very specialized protocol.

The requirements are somewhat relaxed for ordinary text or data files, consisting purely of ASCII characters. Dissimilar utilities can exchange such files, but error checking is often rudimentary or non-existent; we don't recommend this mode of transmission for sensitive data. Also, don't try to send COM files in ASCII mode - ASCII transmissions preserve only the first seven bits of each eight-bit data "word" - the loss of one bit out of every eight renders a COM file completely useless.

5.24 OTHER ALTERNATIVES

For a fee, Microlog can transfer your files to a PC-DOS 5" diskette from practically any soft-sectored CP/M diskette. The fee is payable in advance, but will be refunded if the transfer is unsuccessful. Microlog will also assist software producers who wish to publish their CP/M software completely ready to run on PC-DOS diskettes.

5.3 IMPORTING CP/M PROGRAMS: COMPATIBILITY

5.31 DEFINITION

To run on Baby Blue II, a CP/M program must be:

1) Compatible with CP/M-8Ø version 2.2

"Version 2.2" represents an update of the original CP/M-8Ø operating system, with enhanced capabilities. Generally, any program written for an earlier version (lower number) will be compatible. Do not confuse CP/M-8Ø with CP/M-86, which is an alternative to PC-DOS and doesn't need Baby Blue II to run on your computer.

2) Installable to a Televideo 95Ø terminal:

A program controls your video display using codes which are defined not by the operating system itself, but by the manufacturers of the leading display terminals. Since CP/M-8Ø programs are published to work with a variety of terminals, they generally come with an installation module which asks you to choose your terminal from a list of available options.

Thus, in addition to emulating for your program's benefit the environment of a CP/M operating system, HEADER must also pretend to be one of the standard terminals for which CP/M-8Ø programs were written. We have chosen the very popular Televideo 95Ø terminal as our model - you will find that this choice assures compatibility with the widest range of available programs.

In general, Baby Blue II supports programs which have proven transportable between native CP/M-8Ø systems; that is, most reputable commercially published software. Incompatibility arises when a program depends upon nonstandard or questionable techniques which, though workable in a particular implementation of CP/M, would be considered unsound by the CP/M community as a whole.

5.32 TEXT AND DATA FILES

Since Baby Blue II operates under PC-DOS, it writes all files directly to PC-DOS diskettes. Its files are therefore PC-DOS files, available to any program, whether it be a native 8Ø88 program or a CP/M program running on Baby Blue II.

You will find this especially useful in a number of cases where only some of the programs in a particular family are available in PC-DOS versions. You can mix and match CP/M and PC-DOS modules, as long as they communicate by exchanging text/data files.

Keep in mind that not all programs can exchange data files - the file may contain control codes and delimiters which are properly interpreted only by a certain class of programs - for example, Wordstar/Infostar compatible programs won't read DBASEII files without translation, and vice-versa. This is true even for programs running on the same machine under the same operating system, and has nothing to do with transferring files between PC-DOS and CP/M.

5.33 OPERATING CONSIDERATIONS

Some CP/M programs will run on Baby Blue II, but show operational peculiarities, due to differences between CP/M and PC-DOS. Most problems result from "thinking CP/M", that is, when a program's documentation or your own experience leads you to expect features which are either not supported or are handled differently under PC-DOS. The common areas of concern are listed below.

Transient Program Area (TPA)

Baby Blue II's TPA is more than 63 Kilobytes. This is the memory available to a CP/M program, and defines the maximum size of the program you can run. 63K is very large in CP/M terms - you won't run across programs which are too large for Baby Blue II. Note that HEADER is not part of this overhead - it runs in system memory, not in Baby Blue II's TPA.

CP/M Resident Commands

Use PC-DOS commands for operations like rename file, directory, erase, etc. CP/M resident commands are not emulated.

CP/M Transient Commands

DDT, ASM, and LOAD will run. File-oriented commands, such as PIP and STAT may run but with poor or misleading results. Use PC-DOS equivalents (e.g. COPY, CHKDSK). Duplicate the SAVE function by running DDT under DEBUG (See "Applications Notes", in the Appendix).

Line Editing

PC-DOS does not support CP/M line editing commands (e.g. CTRL-U, CTRL-R). Use PC-DOS commands, which are assigned to special function keys.

Entering Responses

You will often have to end a typed response by striking Return, where the same program on a CP/M system would not require it.

Submit

Not supported - use PC-DOS .BAT files for batch operations. This provides a primitive way of chaining when no other means is available, and is one way that a CP/M program can be chained to a native PC-DOS program. The Microlog Extended BDOS Call 247 provides a more elegant method.

Note that although $$$.SUB is not supported, it is possible to edit a PC-DOS .BAT file while executing that same file. If the changes are made to an as yet unexecuted command line, they will take effect during the current execution of the .BAT file. Thus a .BAT file can support conditional chaining

I/O Byte

PC-DOS doesn't distinguish between different logical devices (e.g. printers). Therefore a CP/M program that relies on this distinction, using the CP/M I/O Byte, will only address a single device when running on Baby Blue II. The byte found at location ØØØ3H does not contain the usual parameters for I/O redirection; instead, the high-order nibble contains the Segment number at which HEADER found the board.

Users

PC-DOS doesn't support multiple users. HEADER will always default to User Ø.

Case in File Names

PC-DOS treats all filename characters as upper case. Some CP/M programs rely on the distinction between upper and lower case filenames.

Read Only

Not supported under PC-DOS. CP/M supports a software "write protect" which prevents writing to a disk under certain circumstances, even though it is not physically write protected.

Aborting With CTRL-C

You may use CTRL-C to abort a CP/M program which supports it. This will return you to PC-DOS without rebooting your system. In contrast to normal CP/M usage, a CTRL-C will cause an abort when typed anywhere in a line, not merely at the beginning.

5.4 BABY BLUE II AS A CP/M DEVELOPMENT SYSTEM

Baby Blue II makes an excellent tool for CP/M program development. Most CP/M-8Ø compilers, interpreters, and development utilities (including SID and DDT) have been thoroughly tested on Baby Blue II; their maturity and depth often makes these tools preferable even where PC-DOS equivalents exist. Because of its relatively large Transient Program Area (63K), Baby Blue II can handle larger development files than most CP/M systems.

Since you know that any COM files you produce will need HEADER to run on Baby Blue II, you will want to know whether this is going to be a problem. Do you have to bind the COM files you create? How do you get HEADER off again when you want to work on them? What about chaining between programs?

Again, the rule is transparency - as far as you're concerned, except when transferring programs to and from a native CP/M system, you can forget that HEADER is there. A development tool is like any other program - once you've bound it, it handles operating system transactions automatically, and all files are produced ready to use.

5.41 TRANSPARENCY OF HEADER DEFINED

The following are the formal rules by which HEADER handles files containing HEADER itself. We've expanded the discussion to include the most relevant cases.

Please note that these rules apply only when under the control of HEADER, that is, when running a CP/M-8Ø program on Baby Blue II - native PC-DOS programs will not recognize the presence of HEADER. Also note that in the case of interpreters and pseudo-compilers (e.g. CBASIC) which do not produce COM files, HEADER is not even part of your program files - it is bound only to the run-time module or interpreter.

5.411 Rule I: Creating COM Files

New COM files are automatically written with HEADER attached. The program which creates the file copies its own HEADER to the new file.

A) New files are produced ready to run under PC-DOS.

B) HEADER need not be present as a separate file.

C) Any variables stored in HEADER, such as KEYFIXed function key definitions, will be transferred from the creating program to the new file.

D) Output files with an extension other than "COM" are <u>never</u> written with HEADER attached.

E) Files received from another computer by a CP/M serial communications program, such as the Microlog-supplied BSTAM, will also be bound if they are written to disk with the extension "COM".

5.412 Rule II: Opening Existing COM Files

HEADER is skipped, and the file is opened at the first line of the program itself.

A) when debugging a COM file (e.g. under DDT), everything is where you expect to find it, not offset to account for HEADER's extra code.

B) you can chain to bound CP/M programs.

5.413 Rule III: Copying COM Files

This is not a new definition, but a consequence of the rules for opening an existing file and creating a new one. In copying an existing COM file, the input (original) file is read without HEADER, and the output (copy) is a newly created file which obeys Rule I.

Note: These rules do not apply to COPY or other PC-DOS utilities, which have no provision for any special handling of HEADER.

A) If you copy a COM file to a file with some other extension (e.g. ".CPM"), the new copy won't contain HEADER. This is because the input file is read (opened) without HEADER, and since the output is not a COM file, it is written as is, again without HEADER.

B) If the copy is also a COM file, it will contain not the HEADER of the input file, but rather the HEADER of the program which does the copying - this is true even if the input and output filenames are exactly the same. The distinction is academic unless the two HEADERs are different, either with respect to version number, or to variable information.

1) This is one way to automatically transfer a whole set of function key definitions to a new file - just KEYFIX some COM file capable of performing a COPY operation (most text editors have this facility), and then "graft" the KEYFIXed HEADER on to any number of files simply by copying them.

2) It's important that installation modules be KEYFIXed identically to the applications programs they are meant to serve. Otherwise, when you process (copy) the applications program to install it, you'll lose your function key definitions, since the installed copy will have lost its original HEADER.

5.414 Rule IV: Opening Unbound COM Files

If a COM file does not contain HEADER, a "not found" error is returned when you attempt to open it.

A) A 16-bit program cannot be called as an overlay to a CP/M program.

B) An unbound CP/M program can be called as an overlay, but only if its extension is not "COM".

5.42 EXPORTING PROGRAMS

As you have seen, there is never a need to remove HEADER while a program is running under PC-DOS; however, it must come off before the program will run on a native CP/M system. Any of the following will work:

1) Use the Microlog utility STRIP.

2) Transfer the file to a CP/M diskette under CONVERT.

3) Under the control of a CP/M program (not a PC-DOS utility), copy the file to an extension other than "COM".

4) Transmit the file using a CP/M serial communications program such as the Microlog-supplied BSTAM.

STRIP is a native program and does not require a Baby Blue II to work. The other methods work according to the formal rules set forth above - they all contain HEADER, and produce an output file which is not a PC-DOS COM file. For example, the file actually transmitted by the communications program appears to the operating system under a different name, usually with an extension like ".$$$".

Once HEADER is removed, COM files are fully transportable from the Baby Blue II to other CP/M systems. You must, of course, provide for different terminal standards, and your program must fit within the TPA (Transient Program Area) of the target system, which will typically be much smaller than Baby Blue II's.

NOTES:

6. CONVERSION SOFTWARE TECHNICAL REFERENCE

6.1 INTRODUCTION

The Co-Processor Module functions as an emulated CP/M environment, occupying a single 64K Page (Segment), within the host 8088 microprocessor's memory space. It utilizes Baby Blue II's memory Bank IV, which is dual-ported, directly accessible to either the Z-80 or the 8088. Arbitration circuitry automatically ensures that only one processor has access to the bus at any given time. The Z-80 is addressed separately from memory as a device in the 8088's I/O map, through physically distinct decoding circuitry. Therefore, the 8088 can treat Bank IV as an ordinary 64K memory expansion whenever the Z-80 is not executing a program.

During native PC-DOS program execution, the Z-80 is in a HALTed state, during which it executes a bare memory refresh cycle: dummy "Read" operations on each address in turn, taking no action on the stored information. The effect is to physically maintain the electrical level at each location in Baby Blue II's memory, so that the information there remains intact. The two processors alternate control of Bank IV memory according to the handshaking scheme described in "Hardware Functions".

This chapter explains how HEADER drives Baby Blue II, and conversely, how a CP/M program running on the Baby Blue II gains access to host system functions. Since handshaking and memory arbitration are hard-wired, applications can and have been written which do not use HEADER functions at all; however, the discussion of HEADER illustrates all relevant issues, divided as follows:

Control Functions

Describes overall system layout and flow of control during CP/M program execution.

Operating System Translator

Details the conversion of standard CP/M function calls to their PC-DOS equivalents.

Console Emulation

Describes the action of the Televideo 950 Emulator, with a complete list of implemented control sequences.

Extended BDOS Function Calls

Introduces a special series of CP/M-style function calls, enabling a program running on Baby Blue to utilize host system memory, interrupt facilities, and I/O ports.

6.2 CONTROL FUNCTIONS

The process of running a CP/M program begins when PC-DOS loads the program from disk, into system memory. Execution begins with the first byte of HEADER, which is written in code native to the 8Ø88.

First, HEADER "polls" system memory to find out where the Co-Processor Module is installed. Starting on Page 1, it saves the contents of a short address space, then uses that space to write a program, instructing the Z-8Ø to set a "Found" flag within co-processor memory. The HALT state is lifted, activating the Z-8Ø. If a valid "Found" is returned, HEADER knows it has found the Co-Processor. If not, HEADER restores the original contents of the borrowed locations, and the poll is repeated for the next segment, up to Page E, covering all possible locations. If a valid "Found" is not returned, control returns to the operating system, and the message "No Baby Blue Installed" appears on the screen.

Once the Z-8Ø is found, it enters a tight polling loop starting at location FE2ØH, and waits for a "Start" flag while HEADER constructs a simulated CP/M environment within PC-DOS. The first task is to install a Televideo 95Ø Console Emulator in system memory to handle keyboard and monitor transactions, by rerouting traffic through a new set of console drivers. The host drivers remain intact but disabled.

The Co-Processor's memory receives an abridged CP/M operating system and the CP/M program itself (See Table 1). The bottom 256 bytes hold the usual CP/M system-control parameters: for example, the expected jump table vectors are at ØØØØH and ØØØ5H. The I/O Byte normally at ØØØ3H is not implemented; instead, the high order nibble at this location holds the segment number at which HEADER's polls the Co-Processor. The top 5ØØ or so bytes contain the Z-8Ø portion of the Operating System Translator, which mediates between a CP/M program's function calls and PC-DOS. The bulk of Co-Processro starting at location 1ØØH, is TPA (Transient Program Area) - the area used to run CP/M programs.

Those familiar with CP/M memory layout will notice at once the very large "true" TPA - more than 63K entirely reserved for program execution. In an ordinary CP/M-8Ø environment, the boundaries of the memory map are also 64K wide, because that is the largest memory space which the Z-8Ø can directly address. Normally, large sections of that memory are taken up with elements of the operating system, imposing such severe constraints that a major element of the operating system (the CCP, or Console Command Processor) is routinely overwritten in memory when a transient program is loaded. This increases the available TPA, but means that the CCP must be reloaded from the system diskette every time you exit a program and return to the system level. No such overwrite takes place on Baby Blue II, since the permanently available TPA is definitely large enough to hold any CP/M program.

The source of the extra TPA is that with very minor exceptions, the entire operating system resides in the memory of the host and is managed by the 8Ø88. To the extent that the CCP and other transient routines need not be treated as overlays, execution speeds increase. A collateral advantage is that it is not necessary to introduce an entire CP/M operating system, with the result that to the operator, and for the most part to the rest of the system, the operating system remains the familiar PC-DOS.

At location FFFØH in Co-Processor memory, there is a one-byte register which we will call the "Semaphore": the contents of this byte indicate which processor controls the bus. With the 8Ø88 in control, this byte is filled with "1"'s (FFH), which means that the 8Ø88 is in control. When a CP/M program is fully loaded, the 8Ø88 sets the semaphore to a line of "Ø"'s (ØØH), then toggles a "Start" flag to set the Z-8Ø running.

At this point, the 8Ø88 is free to conduct normal operations, using any segment of system memory. HEADER turns it into a dedicated I/O controller: it locks into a loop of code which causes it to periodically interrupt the Z-8Ø and inspect the contents of the semaphore. As long as the semaphore remains low (ØØH), the Co-Processor runs at full speed, completely independent of the host system - except for the 8Ø88's occasional poll, there is no handshaking to retard execution.

When the CP/M program needs to communicate with the outside world, it issues a function call to the operating system. Although the Z-8Ø has direct access to Baby Blue II's ports for special applications, normally HEADER treats all I/O channels as elements of the host operating system, passing standard system calls to PC-DOS.

The Z-8Ø posts the contents of its internal registers in a table just above the semaphore address, and toggles the semaphore to FFH, surrendering control to the 8Ø88. Handshaking resumes, with the Z-8Ø executing a polling loop of its own to periodically inspect the semaphore.

The 8Ø88 inspects the Z-8Ø register table for the function call number and other parameters. Instructions issued by the CP/M program pass through the Operating System Translator, after which it's business as usual under PC-DOS. Returned values are posted to the Z-8Ø register table and other relevant tables in Co-Processor memory. Finally, the 8Ø88 resets the semaphore to ØØH and lapses into dormancy, polling for another I/O request.

When the Z-8Ø discovers that the semaphore has changed, it resumes program execution. At the end of execution, control returns to the 8Ø88, but not immediately to the system. First

HEADER does a house-cleaning which HALTs the Z-8Ø and returns the host operating system to normal, removing all traces of unusual activity. Only now does HEADER retire, relinquishing control to PC-DOS.

Table 6-1: Co-Processor Memory Map

Hexadecimal		Decimal
FFFF		65535
FFF1	Begin Z8Ø Register Table	65521
FFFØ	Semaphore	6552Ø
	<u>**Z-8Ø Portion of Translator**</u>	
FFØØ	CP/M BIOS Jump Table	6528Ø
FEØ6	CP/M BDOS Jump Table	65Ø3Ø
FDFF*		65Ø23*
	Transient Program Area - Space for User Programs	
Ø1ØØ		256
	<u>Page Zero</u>	
ØØ8Ø	DMA Address	128
ØØ6C	Second Input Filename	1Ø8
ØØ5C	First Input Filename	92
ØØØ5	Jump Vector to BDOS Translation	5
ØØØ3	Not I/O Byte: Contains Co-Processor Segment Number	3
ØØØØ	Jump Vector to BIOS Jump Table	Ø

* Subject to change

6.3 CONSOLE EMULATION

6.31 DESCRIPTION

The Televideo 950 Emulator installs in two parts: an output section, which handles all screen output from a CP/M program, and a keyboard input section, which supports TV950-style programmable function keys as well as Microlog's own KEYFIX facility.

6.32 PURPOSE

The Emulator establishes portability of almost all CP/M programs to Baby Blue II's console - that is, any program installable to the Televideo 950 terminal (or the ADM-3A). The purpose is not to emulate a Televideo 950 with respect to the operator or a remote system - see Microlog's Smart Terminal Emulator Program (STEP) for this facility. Keyboard input passes straight through without translation - control sequences entered at the keyboard will not alter video functions, but appear literally as the values typed.

6.33 VIDEO OUTPUT

6.331 Operation

HEADER replaces the PC-DOS CONOUT interrupt, diverting control to the TV950 Emulator and bypassing the host screen driver. As a result, CP/M console output passes without translation from Baby Blue II to the host system, thence to the Emulator where it is finally interpreted, still without translating the original CP/M output. CONOUT is therefore handled by the 8088 under PC-DOS, but while the CP/M program is running, PC-DOS itself drives the screen through the TV950 Emulator and not through the usual driver.

6.332 Video Control Codes

Table 2 defines the standard set of codes for CP/M programs running under HEADER. Do not confuse them with keyboard entry codes - the TV950 keyboard is not emulated, and the presence of Baby Blue in no way alters the operating features of PC-DOS. The codes are available only to a transient CP/M program using successive CONOUT function calls. The apparent keystroke sequences in the chart are for convenient cross-reference, and should be regarded as mnemonics only. (Use Microlog's Smart Terminal Emulator Program for full emulation of a Televideo 950).

Table 6-2: Televideo 95Ø Video Control Codes

Control Sequences:

Mnemonic	ASCII Decimal	ASCII Hexadecimal	Comment
CTRL G	7	Ø7H	Bell
CTRL H	8	Ø8H	Backspace/cursor left
CTRL I	9	Ø9H	Tab
CTRL J	1Ø	ØAH	Line feed
CTRL K	11	ØBH	Cursor up
CTRL L	12	ØCH	Cursor right
CTRL M	13	ØDH	Carriage down
CTRL V	22	16H	Cursor down
CTRL Z	26	1AH	Clear screen
CTRL ^	3Ø	1EH	Home cursor
CTRL _	31	1FH	New line (carriage return-line feed)

Escape Sequences:

Mnemonic	ASCII Decimal	ASCII Hexadecimal	Comment
ESC $	27, 36	1BH, 24H	Graphics mode on (IBM, not Televideo, graphics set)
ESC %	27, 27	1BH, 25H	Graphics mode off
ESC (	27, 4Ø	1BH, 28H	Set high intensity
ESC)	27, 41	1BH, 29H	Set low intensity
ESC *	27, 42	1BH, 2AH	Clear screen
ESC +	27, 43	1BH, 2BH	Clear screen
ESC ,	27, 44	1BH, 2CH	Clear screen

ESC .a	27, 26	1BH, 2EH	Set cursor attribute, where "a"="attribute", coded as follows:
Ø	48	3ØH	No cursor
2	5Ø	32H	Steady block cursor
4	52	34H	Steady underline cursor
ESC = rc	27, 61	1BH, 3DH	Position cursor, where r and c are row and column, with offset of 32 (2ØH) added to each
ESC ?	27, 63	1BH, 3FH	Transmit current cursor position (row, column)
ESC E	27, 69	1BH, 45H	Insert line
ESC G a	27, 71	1BH, 47H	Set video attribute, where "a"="attribute", coded as follows:
Ø or @	48, 64	3ØH, 4ØH	normal
1 or A	49, 65	31H, 41H	blank
2 or B	5Ø, 66	32H, 42H	blink
3 or C	51, 67	33H, 43H	blank
4 or D	52, 68	34H, 44H	reverse
5 or E	53, 69	35H, 45H	reverse blank
6 or F	54, 7Ø	36H, 46H	reverse blink
7 or G	55, 71	37H, 47H	reverse blank
8 or H	56, 72	38H, 48H	underline
9 or I	57, 73	39H, 49H	underline blank
: or J	58, 74	3AH, 4AH	underline blink
; or K	59, 75	3BH, 4BH	underline blank
< or L	6Ø, 76	3CH, 4CH	underline
= or M	61, 77	3DH, 4DH	underline reverse blank
> or N	62, 78	3EH, 4EH	reverse blink
? or O	63, 79	3FH, 4FH	underline reverse blank
			The first (numeric) set of values for "a" also steps the cursor forward one character. The second (alphabetic) set leaves the cursor stationary.
ESC Q	27, 81	1BH, 51H	Insert character
ESC R	27, 82	1BH, 52H	Delete line
ESC T	27, 84	1BH, 54H	Clear to end of line
ESC N	27, 78	1BH, 4EH	Set page edit mode
ESC O	27, 79	1BH, 4FH	Set line edit mode

ESC V c	27, 86	1BH, 56H	Set color, where "c"="color, coded as follows:
Ø	48	3ØH	foreground black
1	49	31H	foreground blue
2	5Ø	32H	foreground green
3	51	33H	foreground cyan
4	52	34H	foreground red
5	53	35H	foreground magenta
6	54	36H	foreground brown
7	55	37H	foreground white
8	56	38H	foreground grey
9	57	39H	foreground light blue
:	58	3AH	foreground light green
;	59	3BH	foreground light cyan
<	6Ø	3CH	foreground light red
=	61	3DH	foreground light magenta
>	62	3EH	foreground yellow
?	63	3FH	foreground high-intensity white
@	64	4ØH	background black
A	65	41H	background blue
B	66	42H	background red
C	67	43H	background magenta
D	68	44H	background green
E	69	45H	background cyan
F	7Ø	46H	background brown
G	71	47H	background white
H	72	48H	border black
I	73	49H	border blue
J	74	4AH	border green
K	75	4BH	border cyan
L	76	4CH	border red
M	77	4DH	border magenta
N	78	4EH	border brown
O	79	4FH	border white
P	8Ø	5ØH	border grey
Q	81	51H	border light blue
R	82	52H	border light green
S	83	53H	border light cyan
T	84	54H	border light red
U	85	55H	border light magenta
V	86	56H	border yellow
W	87	57H	border high-intensity white

ESC W	27, 87	1BH, 57H	Delete character
ESC Y	27, 89	1BH, 59H	Clear to end of screen
ESC F text CR	27, 1Ø2 (user entry) 13	1BH, 66H ØDH	Load user buffer I (8Ø characters max)
ESC f text CR	27, 1Ø2 (user entry) 13	1BH, 66H ØDH	Load user buffer II (8Ø characters max)
ESC g	27, 1Ø3	1BH, 67H	Display user buffer I (on line 25)
ESC h	27, 1Ø4	1BH, 68H	Display user buffer II (on line 25 - in a real TV95Ø, this buffer contains the status line)
ESC j	27, 1Ø6	1BH, 6AH	Reverse line feed
ESC t	27, 116	1BH, 74H	Clear to end of line
ESC y	27, 121	1BH, 79H	Clear to end of screen

6.34 KEYBOARD INPUT

6.341 Operation

Console input functions (i.e. keyboard) remain largely intact, but are routed through a Keyboard Emulator for two functions:

- a greatly expanded TV95Ø-style operator-definable function key set, yielding not 22 but 56 programmable function keys divided into two tables of 256 characters each (double the TV95Ø 256-character table). The included KEYFIX utility offers complete flexibility to the unsophisticated end-user.

- the standard TV95Ø facility for definition of function keys under program control.

HEADER alters the keyboard interrupt vectors, modifying the effective action of CONIN and CONSTAT when a function key is pressed, but leaving console input functions otherwise intact. Normally a valid CONSTAT results in a single CONIN, returning one value for the depressed key. When HEADER recognizes a function key, however, control passes to a special handler which finds the character string assigned to that key in a table, and determines its length. The handler loops CONIN and CONSTAT for the required number of iterations, returning characters one by one until the end of the table entry is reached, at which point CONSTAT returns to the inactive state.

Keep in mind that although a program may issue the TV95Ø cursor control codes, CP/M resident line edit functions are not valid at the keyboard - since we are operating under PC-DOS, we must use the PC-DOS line editor. This is significant whenever the read buffer (READBUF) is in use, that is, whenever the system waits for a "Retrn" before inputting the values posted on the screen. An operator whose experience or documentation leads him to expect the CP/M line editing sequences may suffer some confusion. Also, remember that where CP/M will automatically close the buffer and enter the string at a specified length, PC-DOS requires a "Retrn" for closure.

6.342 TV95Ø Function Key Programming

CP/M programs can use the following sequence to define twenty function keys (Normal and Shifted F1-F1Ø) - the TV95Ø's F11 is not supported by the IBM keyboard. Text entered during the sequence will be input to the program whenever the designated function key is pressed. The effect is transient with the current execution: when execution terminates, all keys revert to the definitions stored in HEADER and must be reinitialized with each load of the program. This is in contrast to the user-definable mode offered under KEYFIX, which physically logs the operator's definitions to disk.

Table 6-3: TV95Ø Escape Sequence: Load Function Key

Mnemonic	ASCII Decimal	ASCII Hexadecimal	Comment
ESC |	27, 124	1BH, 7CH	Load function keys
FUNKEY	XX	XXH	Get ASCII code from Table 4
1	49	31H	Start of Message
(text)			This will be the programmed input for the designated key.
[CTRL P	16	1ØH]	Optional - precedes any non-print character
CTRL Y	25	19H	End of Message

Table 6-4: TV95Ø Function Key Codes

KEY	CHAR.	ASCII DEC.	ASCII HEX.	SHIFT KEY	CHAR.	ASCII DEC.	ASCII HEX.
F1	1	49	31	S-F1	<	6Ø	3C
F2	2	5Ø	32	S-F2	=	61	3D
F3	3	51	33	S-S3	>	62	3E
F4	4	52	34	S-S4	?	63	3F
F5	5	53	35	S-F5	@	64	4Ø
F6	6	54	36	S-F6	A	65	41
F7	7	55	37	S-F7	B	66	42
F8	8	56	38	S-F8	C	67	43
F9	9	57	39	S-F9	D	68	44
F1Ø	1Ø	58	3A	S-F1Ø	E	69	45

Because all function keys share a sequential buffer, your program must define them in the order shown. The buffer's capacity is 256 bytes, including all text plus one byte for each key programmed (e.g. If F2 is programmed to input the message "Hi!" a total of 4 characters are used up in the table: one for each of the three text characters, and one more for F2 itself).

6.343 Keyboard Defaults

HEADER is shipped with the function key definitions shown in Table 5. Unlike CP/M (which uses CTRL codes) PC-DOS assigns line editing functions to a set of function keys. HEADER is shipped with the definitions shown in Table 5, where the sequences beginning with "^@" (<CTRL @>) are the codes expected by PC-DOS. Since overwriting these sequences disables the corresponding DOS line-editing function, redefined function keys may cause problems if the target CP/M program employs the operating system's edit facility.

Table 6-5: Function Key Default Definitions

Key	Unshifted	Shifted	Control	Alt
F1	^@; (ØØH,3BH)	S-FUN1	C-F1	A-F1
F2	^@< (ØØH,3CH)	S-FUN2	C-F2	A-F2
F3	^@= (ØØH,3DH)	S-FUN3	C-F3	A-F2
F4	^@> (ØØH,3EH)	S-FUN4	C-F4	A-F4
F5	^@? (ØØH,3FH)	S-FUN5	C-F5	A-F5
F6	^@@ (ØØH,4ØH)	S-FUN6	C-F6	A-F6
F7	^@A (ØØH,41H)	S-FUN7	C-F7	A-F7
F8	FUN8	S-FUN8	C-F8	A-F8
F9	FUN9	S-FUN9	C-F9	A-F9
F1Ø	FUN1Ø	S-FUN1Ø	C-F1Ø	A-F1Ø
L ARROW	^@< (ØØH,Ø4BH)		C-LF	
R ARROW	^@< (ØØH,Ø4DH)		C-RT	
U ARROW	UP			
D ARROW	DOWN			
HOME	HOME		CTRL HOME	
END	END		CTRL END	
PG UP	PG UP		CTRL PG UP	
PG DN	PG DN		CTRL PG DN	
INSERT	^@R (ØØH,52H)			
DELETE	^@S (ØØH,53H)			

6.344 Emulating TV950 Keyboard Defaults

The Televideo 950 contains a set of default function key definitions which are active in the absence of any other definitions. Occasionally, instead of reprogramming the keys, a program simply looks for the default definitions to initiate control functions. In such a case, the program must be KEYFIXED according to Table 6 before the function keys will work on the IBM PC. Some of the keys shown are not available on the PC - they are shown so that their definitions can be assigned to some other available key.

Table 6-6: Televideo 950 Function Key Defaults

Key	Unshifted	Shifted
F1	<CTRL A> @ <CTRL M>	<CTRL A> ' <CTRL M>
F2	<CTRL A> A <CTRL M>	<CTRL A> a <CTRL M>
F3	<CTRL A> B <CTRL M>	<CTRL A> b <CTRL M>
F5	<CTRL A> D <CTRL M>	<CTRL A> d <CTRL M>
F6	<CTRL A> E <CTRL M>	<CTRL A> e <CTRL M>
F7	<CTRL A> F <CTRL M>	<CTRL A> f <CTRL M>
F8	<CTRL A> G <CTRL M>	<CTRL A> g <CTRL M>
F9	<CTRL A> H <CTRL M>	<CTRL A> h <CTRL M>
F10	<CTRL A> I <CTRL M>	<CTRL A> i <CTRL M>
F11	<CTRL A> J <CTRL M>	<CTRL A> j <CTRL M>
L ARROW	<CTRL H>	<CTRL H>
R ARROW	<CTRL L>	<CTRL L>
U ARROW	<CTRL K>	ESC j
D ARROW	<CTRL V>	<CTRL J>
HOME	<CTRL 6>	<CTRL 6>
BACKTAB	ESC I	ESC I
PRINT	ESC P	ESC L
LINE INS	ESC E	ESC N
LINE DEL	ESC R	ESC O
CHAR INS	ESC Q	ESC q
CHAR DEL	ESC W	ESC r
LINE ERASE	ESC T	ESC t
PAGE ERASE	ESC Y	ESC y
SEND	ESC 7	ESC 6
CLEAR SPACE	<CTRL Z>	ESC *

6.4 OPERATING SYSTEM TRANSLATOR

6.41 DESCRIPTION

The Translator converts CP/M BIOS and BDOS calls issued by the transient program into their nearest logical PC-DOS equivalent, for execution by the host operating system. Just as the console emulator defines CP/M compatibility for a TV950 standard terminal, the Translator defines program compatibility with respect to the function calls employed.

6.42 PURPOSE

The Translator provides mutual transparency between the transient program and the host operating system. There is no point-to-point correspondence between CP/M and PC-DOS - although they are close cousins there are some fundamental differences, the most important of which concerns access to disk files. Some features of CP/M are not supported under PC-DOS: these function calls will return default values reflecting the state invariably imposed on that function by PC-DOS (e.g., User Code always returns as Ø, because PC-DOS does not support more than one user).

6.43 CP/M BDOS FUNCTION CALLS

All BDOS calls follow standard CP/M procedure. We will treat them under the standard CP/M function call numbers.

Ø: <u>System Reset</u>

Used to terminate program execution. Returns control to PC-DOS (COMMAND.COM), for full normal operation. Before relinquishing control, HEADER performs a general house-cleaning in the BIOS, returning all vectors to their normal values, re-enabling the native PC-DOS console drivers, and resetting the Z-8Ø.

1: <u>Console Input</u> (CONIN - "Get/Read a Console Character")

A straight-line translation to PC-DOS CONIN, except that function keys are handled differently as detailed under "Keyboard Emulator". See "1Ø: Read Console Buffer" for further comments about keyboard entry functions.

2: <u>Console Output</u> (CONOUT - "Write a Console Character")

A straight-line translation to the PC-DOS CONOUT, but note that PC-DOS itself is routed through the TV950 emulator, not through the normal host screen driver - hence monitor controls must conform to the TV950 standard (See Table 4-2).

3: <u>Reader Input</u>

A direct call to the PC-DOS Aux In.

4: <u>Punch Output</u>

A direct call to the PC-DOS Aux Out.

5: <u>List Output</u>

Calls PC-DOS PRINT OUT, routing direct to the PRN device.

6: <u>Direct Console I/O</u>

Fully supported.

7/8: <u>Get/Set I/O Byte</u>

Ignored because PC-DOS does not support I/O redirection at this level - "Get I/O Byte" will always return the default value 0. At location 0003H, where the CP/M I/O byte is normally found, the high-order nibble contains instead the segment number assigned to the Co-Processor's 64K.

9: <u>Print String</u>

Fully supported - a direct translation to the same PC-DOS function.

10: <u>Read Console Buffer</u>

A straight-line translation, but note that this means PC-DOS line editing commands will be in effect, not CP/M, so that an operator expecting to use the CP/M set may be confused. Also note that redefining the function keys may disable PC-DOS line-editing features (See "1: Console Input).

11: <u>Get Console Status</u> (CONSTAT: "Interrogate Console Ready")

A straight-line translation except that like "1: <u>Console Input</u>", this will be handled, and sometimes automatically repeated, by the Keyboard Emulator.

12: <u>Return Version Number</u>

Returns Version 2.2.

13: <u>Reset Disk System</u>

Ignored, since the purpose of this call is always satisfied under PC-DOS (all disks perpetually set to read/write). No incompatibility will result from the use of this command, but it may mask a deeper problem if the program or its documentation depends on the CP/M software write-protect facility (see "28: <u>Write Protect Disk</u>").

14: <u>Select Disk</u>

Direct translation - designates default drive. However, the drive will not automatically go to a read-only state if the disk media is physically changed, as it would under CP/M (See "28: <u>Write Protect Disk</u>").

15: <u>Open File</u>

Fully supported, however some confusion may result if you don't fully understand how the HEADER handles CP/M COM files under development, as explained under OPERATION.

16: <u>Close File</u>

Direct translation to PC-DOS function call. AL returns either ØØH (successful close) or FFH (file not found). When closing a COM file, this call also finds the size of the Z-8Ø code (less HEADER) and stores this number at location Ø1Ø7H in the HEADER attached to the target file. The size of HEADER itself is stored at location Ø1Ø5H.

17/18: <u>Search for First/Next</u>

Direct translation to PC-DOS function calls. Returns ØØH (file found) or FFH (file not found) in AL. The directory image buffered at the DMA address is artificially constructed from the PC-DOS image, with the following surprises:

- In the case of a COM file, the record count returned includes HEADER, accurately reflecting the disk space required, but not the TPA.
- There is only one entry, so AL, if found, is always Ø (not 1, 2 or 3). The remaining 96 bytes, which might ordinarily contain further entries, are filled with E5H.
- The correct number of group entries are filled in, but they are all set to Ø1H, since the actual pointer in PC-DOS is to the file's first entry in the File Allocation Table.

19: Delete File

Direct translation to PC-DOS function call.

2Ø: Read Sequential

Direct translation to PC-DOS function call - for a COM file, the first record returned will be the first line of Z-8Ø code - HEADER is skipped over.

21: Write Sequential

Direct translation to PC-DOS. In the case of a COM file, the presence of HEADER on the disk is automatically accounted for - no special adjustments are required to insure that the write indeed begins at the end of the file.

22: Make File:

Direct translation to PC-DOS. As explained under OPERATION, HEADER is automatically bound when the file to be created is designated as a COM file. HEADER is written and closed immediately, before the COM file is opened, so even if you decide not to write to the file, or not to close it, you'll still find that you've created a file containing HEADER.

23: Rename File

Automatically binds HEADER when the filename extension is changed from something else to COM, and vice-versa. Since in the first case the bound file is larger by the length of HEADER, it's possible there will be insufficient disk space available to write it. Rather than lose the file, we recover by leaving the file unbound, and tagging it with the extension "CPM".

24: <u>Return</u> <u>Log-in</u> <u>Vector</u>

Not supported, because it is irrelevant under PC-DOS. Returns the default value FFFFH.

25: <u>Return</u> <u>Current</u> <u>Disk</u>

Direct translation to PC-DOS function call.

26: <u>Set</u> <u>DMA</u> <u>Address</u>

See BIOS Call FF24H: <u>SETDMA</u>, below.

27: <u>Return</u> <u>Allocation</u> <u>Vector</u>

Not supported. This function, usually not used by allocation programs, returns a value which refers to physical properties of a CP/M diskette. Since PC-DOS diskettes do not share these attributes, the function is meaningless when directed at a PC-DOS diskette.

28: <u>Write</u> <u>Protect</u> <u>Disk</u>

Not supported, since PC-DOS does not support the software write-protect facility offered by CP/M.

29: <u>Return</u> <u>Read</u> <u>Only</u> <u>Vector</u>

Not supported, see "28: Write Protect Disk".

30: <u>Set</u> <u>File</u> <u>Attributes</u>

Not supported, since the attributes themselves reside in the physical directory of a CP/M disk and have no equivalent under PC-DOS. Therefore, this call will also fail to "discover" a file which has been defined as "hidden" under PC-DOS.

31: <u>Get</u> <u>Address</u> <u>of</u> <u>Disk</u> <u>Parameters</u>

Partially emulated. The parameters involved are properties of a CP/M diskette and are not supported by PC-DOS. The address returned points to a dummy parameter table based on an assumed 5" diskette.

32: <u>Get/Set User Code</u>

Returns value Ø. PC-DOS does not support multiple users.

33/34: <u>Read/Write Random</u>

Direct translation to PC-DOS function call. Files created while running on Baby Blue II will not introduce gaps in a random access file, and so will be fully transportable.

35: <u>Compute File Size</u>

Returns true file size. Since gaps are not permitted in a random access file under PC-DOS, "virtual size" is always the physical size of the file. HEADER is subtracted from the physical size of a COM file, giving the size of the Z-8Ø code only. This will be accurate for operations conducted on Baby Blue II or on another CP/M system, since HEADER will not appear in memory in either case. If it is desired to return the size of a COM file including HEADER, the directory image returned to the DMA address by function calls 17 and 18 will contain this information in the record count.

36: <u>Set Random Record</u>

Direct translation to PC-DOS function call.

37: <u>Reset Drive</u>

Ignored as irrelevant to PC-DOS.

38: <u>Not used</u>

39: <u>Not used</u>

4Ø: <u>Write Random With Zero Fill</u>

Translated to Function 34: Write Random. (This function refers to physical properties of a CP/M diskette not duplicated under PC-DOS)

6.44 CP/M BIOS CALLS

HEADER maintains a CP/M BIOS jump table starting at FFØØH in Co-Processor memory, with the standard pointer at ØØØØH. Except for disk-based routines, most calls pass to their BDOS counterparts, which in turn call their direct equivalents in PC-DOS. Because CP/M and PC-DOS locate physical disk sectors very differently, disk-based calls undergo a more complicated translation.

6.441 Logical to Physical Sector Mapping

All disk I/O is based on a conversion of CP/M Track/Sector parameters to corresponding PC-DOS logical sectors, assuming an ideal "CP/M" diskette of thirty-two 128-byte sectors per track. This ideal format is automatically mapped onto a real PC-DOS 5" diskette of eight 512-byte sectors per track, through the following algorithm:

Logical PC-DOS Sector = (32 * T + S - 1) / SCALE

Where:

T = "Track number"

S = "Segment number"

and SCALE is computed automatically upon disk selection, as:

real physical sector size in bytes / 128

The algorithm assumes 4Ø96 bytes per track, with a limit of 1Ø24 bytes per sector. It will find the specified sector on any disk conforming to these parameters, since SCALE automatically accomodates different sector sizes. There is no range check on sector number, but it must be in the range 1 to 255. The first segment on the disk is Track Ø, Sector 1, which becomes PC-DOS Logical Sector Ø - therefore, the physical sector always equals the logical sector plus one.

Remainders are truncated, guaranteeing that the logical sector will always contain the expected 128-byte sector. This is because remainders are only produced when physical sector size is larger than 128 bytes, in direct proportion to SCALE - some remainder "n" is really a pointer to the nth 128-byte block within the physical sector. The physical sector is read into a 1K buffer maintained in HEADER, and deblocked into 128-byte segments for loading at the DMA address.

You can also read a non-conforming format if its parameters are known, and sector size does not exceed 1Ø24, but you must first transpose the target track and sector number into the "ideal" equivalents expected by HEADER. Find the target sector as the "Nth" physical sector, counting from the beginning of the disk:

$$NthSect_{phys} = (SPT * T_{phys}) + S_{phys}$$

Where:

SPT = Physical Sectors per Track on the target disk.

T_{phys} = Physical (literal) track number.

S_{phys} = Physical (literal) sector number.

Multiply this number by SCALE, converting it to the nth 128-byte block ($NthSect_{128}$):

$$NthSect_{128} = SCALE * NthSect_{phys}$$

Now divide by 32. The quotient is the desired track number (T), and the remainder is the segment (S). The combined formula reads:

$$T + S = SCALE * (SPT * T_{phys} + S_{phys}) / 32$$

Passing these calculated values to SETTRK and SETSEC will yield the desired physical sector.

For example, given an 8" single-sided, single-density diskette of twenty-six 128-byte sectors per track, SCALE = 1. Therefore, physical [Track 1Ø, Sector 5] yields:

$$(26 * 1Ø) + 5 / 32 = 265/32 = 8 + 8$$

Or [Track 8, Sector 8]. [Track 7, Sector 4Ø] is also valid, since there is no range check, but not [Track Ø, Sector 265], because the highest allowable sector number is 255.

6.442 BIOS Entry Points

The standard BIOS entry points are listed below in address order. All BIOS calls follow standard CP/M procedure, except as indicated.

FFØØH: <u>COLD BOOT</u>

Not supported - initialization is controlled by HEADER under PC-DOS.

FFØ3H: <u>WARM BOOT</u>

Invokes BDOS call Ø, System Reset.

FFØ6H: <u>CONST</u>

Invokes BDOS call 11, Get Console Status.

FFØ9H: <u>CONIN</u>

Invokes BDOS call 6, Direct Console I/O (input).

FFØCH: <u>CONOUT</u>

Invokes BDOS call 2, Console Output.

FFØFH: <u>LIST</u>

Invokes BDOS Call 5, List Output.

FF12H: <u>PUNCH</u>

Invokes BDOS Call 4, Punch Output.

FF15H: <u>READER</u>

Invokes BDOS Call 3, Reader Input.

FF18H: <u>HOME</u>

Not supported.

FF1BH: <u>SELDSK</u>

Calculates SCALE. The disk parameters are always based on an ideal 40-track diskette, with 32 128-byte sectors per track.

FF1EH: <u>SETTRK</u>
FF21H: <u>SETSEC</u>

Literal physical track and sector numbers are valid for any disk of 4096 bytes per track, and no more than 1024 bytes per sector. Other formats are accessible with translated parameters, as described above. The first physical sector on each track is number 01H.

FF24H: <u>SETDMA</u>

The initial address is the expected 80H. This call invokes BDOS Call 26, which means that either call alters the address set by the other. The usual 128-byte allocation is sufficient, regardless of physical sector size - the physical sector is stored and deblocked from a 1024-byte buffer maintained in HEADER.

FF27H: <u>READ</u>
FF2AH: <u>WRITE</u>

Data is buffered and blocked/deblocked as described above, under SETDMA.

FF30H: <u>SECTRAN</u>

The physical sector always equals the logical sector plus one.

FF2DH: <u>LISTST</u>

Always returns "ready" (FFH in A).

6.5 EXTENDED BDOS FUNCTION CALLS

6.51 DESCRIPTION

Microlog has created a set of new CP/M-8Ø style function calls for use when running under HEADER. They are:

Number	Function
247	Chain
248	8Ø88 Software Interrupt
249	System Memory Block Move (Dec)
25Ø	System Memory Block Move (Inc)
251	Peek Host Memory Byte
252	Poke Host Memory Byte
253	8Ø88 BIOS Call (Subset of # 248)
254	Output to Host I/O Port
255	Input from Host I/O Port

6.52 PURPOSE

The extended BDOS function calls are provided to support true user-designed applications using Microlog's Co-processor boards. By means of these functions a CP/M-8Ø program can gain access to the host system at the following levels:

-8Ø88 software interrupt

-Host memory (block moves and individual locations)

-Direct I/O through host ports

6.53 OPERATION

All extended function calls parallel standard CP/M-8Ø usage.

6.531 Call 247: Chain

Entry Parameters

Register C:	F7H
Register DE:	Starting Address of ASCII Command String
Register B:	Length of Command String

Return:

Exits the current program, then invokes the indicated command file.

The Command String may contain the name of any PC-DOS COM, EXE or BAT file, including any passed parameters (DOS resident commands are invalid); it must terminate with ØDH (<CTRL M>, or <CR>). No provision is made for reentry to the calling program.

6.532 Call 248: 8Ø88 Software Interrupt

Entry Parameters:

Register C:	F8H
Register HL:	Address of pseudo 8Ø88 Interrupt/Register Table

Return:

Executes specified interrupt
Updates 8Ø88 Register Table at address specified by [HL]

Emulates an 8Ø88 "INT" instruction. The HL register pair points in Co-Processor memory to the starting address of a table representing the 8Ø88 registers, as follows:

8Ø88 Interrupt/Register Table

	Interrupt Number	Registers									Flags
		AX	BX	CX	DX	BP	SI	DI	DS	ES	
Byte #:	ØØ	Ø1	Ø3	Ø5	Ø7	Ø9	11	13	15	17	19

Flag Byte #19:

Flag:	SF	ZF	--	AF	--	PF	--	CF
Bit #:	7	6	5	4	3	2	1	Ø

The parameters become active as the specified interrupt is executed. Upon completion, the contents of the 8Ø88 registers are returned to the table.

6.533 Call 249: System Block Move - Decrement

Entry Parameters:

Register	C:	F9
Registers	HL:	Block Move Table Address

Return:

Executes block move in system memory (64K max.)

Upon entry, the HL register pair points to a 1Ø byte table in Co-Processor memory, organized as follows:

	Source Offset	Source Segment	Destination Offset	Destination Segment	Block Size
Byte #:	ØØ	Ø2	Ø4	Ø6	Ø8

Where:

Block Size - total number of bytes to transfer (up to FF Hex or 64K).

Source Offset - 16-bit location of first byte in the block you are moving.

Source Segment - Present memory segment containing the block to be moved. Note that this could be the Co-Processor's segement.

Destination Offset - 16-bit location to fill with first byte of the block.

Destination Segment - Memory segment to which block is to be moved. Note that this could be the Co-Processor's segemnt.

This function parallels the Z-8Ø LDDR block move instruction, or the 8Ø88 REPZ MOVSB with the STD instruction, i.e. it moves the block by starting with the highest location and decrementing. You can move data to or from any area of system memory, including on or off Baby Blue II.

6.534 Call 25Ø: System Block Move - Increment

Entry Parameters:

```
     Register    C:  FAH
     Registers HL:  Block Move Table Address
```

Return:

Executes block move in system memory (64K max.)

Identical in all respects to Call 249, except that it emulates the Z-8Ø LDIR instruction, or the 8Ø88 REPZ MOVSB with CLD, i.e., it moves the block starting with the first location and incrementing.

6.535 Call 251: Peek System Memory Byte

Entry parameters:

```
     Register    C:  FBH
     Registers DE:  Offset number
     Registers HL:  Segment number
```

Return:

```
     Register    A:  Contents of Byte
```

Reads a byte from the location specified in [DE] and [HL]. Enables a Z-8Ø program to read from any location in the 8Ø88's address space, including Co-Processor memory.

6.536 Call 252: Poke System Memory Byte

Entry parameters:

```
     Register    C:  FCH
     Register    B:  Contents of Byte
     Registers DE:  Offset number
     Registers HL:  Segment number
```

Return:

Writes contents of byte to specified location in system memory.

The contents of [B] will be written to the location specified in [DE] and [HL]. Enables a Z-8Ø program to write to any location in the 8Ø88's address space.

6.537 Call 253: 8Ø88 BIOS Call

Entry parameters:

```
     Register    C:  FDH
     Registers HL:  8Ø88 Interrupt/Register Table Address
```

Return:

```
     Executes specified interrupt.
     Updates 8Ø88 Interrupt/Register Table at specified address.
```

This function is included for compatibility with earlier versions of HEADER, and is a subset of Function 248. Its action is identical in every respect except that it passes only the first four registers (AX, BX, CX, DX), and the effective table is only nine bytes long.

6.538 Call 254: Output to Host I/O Port

Entry Parameters:

```
     Register    C:  FEH
     Register    E:  8-bit output value
     Registers HL:  Host system port number
```

This function enables a CP/M program to output values directly to a host system port (under the control, of course, of the 8Ø88) - use this function instead of an OUT instruction.

6.539 Call 255: Input from Host I/O Port

Entry Parameters:

```
     Register    C:  FFH
     Registers HL:  Host system port number.
```

Return:

```
     Register    A: 8-bit input value.
```

Complements Call 254, enabling the Z-8Ø to input values directly from an 8Ø88-controlled port.

7. HARDWARE FUNCTIONS TECHNICAL REFERENCE

7.1 CO-PROCESSOR MODULE

7.11 Z-8Ø PORT ADDRESS DECODING

The assignment of address lines to the Z-8Ø's I/O port is given in Table 8 ("Co-Processor Address Decoding"). Note that the memory page (segment) address lines map onto the low-order bits of the port address by sharing the same switches for signal decoding. This means that the port address could vary from Ø3ØØH to Ø31CH, depending on the base address of Co-Processor memory (Table 9).

In a single Co-Processor system, port address and segment decoding could be separate, but tying them together offers the possibility of running more than one in parallel - mapping the onboard memory into different pages will automatically define separate port addresses, without special accomodations from the host control program. HEADER in fact uses this facility when it polls memory to locate the Co-Processor - once the memory segment is located, the port address is automatically known.

Note that here, OFF = 1 = High, and ON = Ø = Low. Values in "{}" brackets are configured for compatibility with HEADER, but could be set differently to interface a different control program. Since A8 and A9 are hard-wired high ("1"), the high-order nibble of the port address is always 3H. AØ is tied Low (Ø).

7.12 Z-8Ø CONTROL LINES

Z-8Ø control lines available to the 8Ø88 programmer are:

NMI	Non-Maskable Interrupt: Jump to Location 66H. In HEADER this interrupt is serviced by a routine which emulates a Z-8Ø system reset.
INT	Interrupt.
HALT	A special, discretely configured control line which presents a hard-wired HALT instruction (76H) to the Z-8Ø data bus, bypassing RAM. Following activation of this line, the HALT instruction waits to appear on the data bus for the next instruction fetch, permitting the orderly completion of the current machine cycle. When the Z-8Ø is in a HALT state, it executes No-ops, which are essentially bare memory refresh cycles. A HALTed Z-8Ø recognizes only an NMI or an INT (with mask enabled), so one of these must be used to resume processing.

RESET is activated only by a power-up system reset, and is not available to the programmer - use NMI with a Z-8Ø service routine at 66H to emulate any desired RESET functions.

Control lines are accessed through the Z-8Ø's I/O port address of as follows:

[O] Ø3XXH [Control Data Byte]

Where:

[O] = OUT Instruction

Ø3XXH = Port address in hexadecimal, where XX is determined by the setting of SW1 (See Table 4-9: "Co-Processor Segment and Port Assignments").

[Control Data Byte] = information transmitted to the port to select the available control lines. Only bits Ø, 2 and 3 of this byte are significant: the rest are "don't cares" ("x"). It maps onto the control lines as follows:

Table 7-1: Z-8Ø Functions Control Byte

	Data Lines				
	NMI	INT	X	HALT	
Functions	D3	D2	D1	DØ	Decimal
HALT Z-8Ø:	Ø	Ø	x	Ø	Ø
RUN (all off):	Ø	Ø	x	1	1
INTERRUPT:	Ø	1	x	1	5
JUMP to Loc. 66H:	1	Ø	x	1	9

All control lines latch and so must be cancelled explicitly. For example, the "JUMP to Location 66H" cancels HALT with a "1" on DØ so that NMI will execute, but NMI must in turn be be cancelled by a "RUN" instruction (Ø on D3) for the service routine to begin - otherwise the Z-8Ø will continuously execute an NMI.

Table 7-2: Co-Processor Address Decoding

PORT NUMBER (HEX)	PORT ADDRESS LINES	DIP SWITCH	SETTING (BINARY VALUE)	SEGMENT ADDRESS LINES	MEMORY SEGMENT (PAGE)
3	A11	-	X	-	-
	A1Ø	-	X	-	-
	A9	HIGH	(1)	-	-
	A8	HIGH	(1)	-	-
X (Ø OR 1)	A7	SW1	ON {Ø}	-	-
	A6	SW2	ON {Ø}	-	-
	A5	SW3	ON {Ø}	-	-
	A4	SW4	?	A19	Ø THRU E
X (2 THRU C)	A3	SW5	?	A18	
	A2	SW6	?	A17	
	A1	SW7	?	A16	
	AØ	LOW	XX	-	

7.13 CO-PROCESSOR MEMORY ARBITRATION

Either processor may directly read or write to Co-Processor memory. Since the Z-8Ø handles refresh, handshaking is constant, but it is automatically controlled by the board's hardware.

The 8Ø88 has priority access. A validly decoded address, combined with an active /MEMR or /MEMW, presents an active /BUSREQ to the Z-8Ø. The Z-8Ø must respond by relinquishing the bus, but first completes its current machine cycle. The 8Ø88 waits, responding to an active signal on its I/O CHANNEL READY line. An active /BUSACK indicates that the Z-8Ø has relinquished the bus, and lifts I/O CHANNEL READY, permitting the 8Ø88 to complete its cycle. Now /BUSREQ goes high, starting the Z-8Ø and insuring that each memory access by the 8Ø88 is followed by at least one Z-8Ø cycle, to maintain refresh.

Table 7-3: Co-Processor Segment and Port Assignments

Switch Setting								Memory	Address	Z-8Ø Port
1	2	3	4	5	6	7		Segment	Range	Address
[]	[]	[]	[]	[]	[]		ON	1	1ØØØØ-1FFFF	3Ø2
						[]	↑			
[]	[]	[]	[]	[]		[]	ON	2	2ØØØØ-2FFFF	3Ø4
					[]		↑			
[]	[]	[]	[]	[]			ON	3	3ØØØØ-3FFFF	3Ø6
					[]	[]	↑			
[]	[]	[]	[]		[]	[]	ON	4	4ØØØØ-4FFFF	3Ø8
				[]			↑			
[]	[]	[]	[]		[]		ON	5	5ØØØØ-5FFFF	3ØA
				[]		[]	↑			
[]	[]	[]	[]			[]	ON	6	6ØØØØ-6FFFF	3ØC
				[]	[]		↑			
[]	[]	[]	[]				ON	7	7ØØØØ-7FFFF	3ØE
				[]	[]	[]	↑			
[]	[]	[]		[]	[]	[]	ON	8	8ØØØØ-8FFFF	31Ø
			[]				↑			
[]	[]	[]		[]	[]		ON	9	9ØØØØ-9FFFF	312
			[]			[]	↑			
[]	[]	[]		[]		[]	ON	A*	AØØØØ-AFFFF	314
			[]		[]		↑			
[]	[]	[]		[]			ON	B*	BØØØØ-BFFFF	316
			[]		[]	[]	↑			
[]	[]	[]			[]	[]	ON	C*	CØØØØ-CFFFF	318
			[]	[]			↑			
[]	[]	[]			[]		ON	D*	DØØØØ-DFFFF	31A
			[]	[]		[]	↑			
[]	[]	[]				[]	ON	E*	EØØØØ-EFFFF	31C
			[]	[]	[]		↑			
[]	[]	[]					ON	F*	FØØØØ-FFFFF	31E
			[]	[]	[]	[]	↑			

* One or more Pages in the range A through F are reserved by all machines.

7.2 DEVICE MODULE

7.21 I/O DEVICES

The Device Module comprises all of Baby Blue II's I/O devices except the Z-8Ø microprocessor, which is addressed separately (See Co-Processor Reference Manual, "Hardware Functions").

7.211 Serial Ports

The two serial ports are called SER1 and SER2; normally, they will be addressed as the PC's COM1 and COM2, respectively. SER1 is always the DB-25 connector at Baby Blue II's mounting bracket. SER2 is assigned to the 26-pin header J2, and can be led out of the chassis via a mass-terminated ribbon cable - no special cabling is required with standard connectors.

Both ports are configured as standard RS-232C DTE (Data Terminal Equipment) interfaces. Available signals conform to the standard of the IBM Asynchronous Communications Adapter, except that the 2Ø milliamp current loop interface is not implemented.

Table 7-4: Serial Port Pin Assignments

Pin	Signal	Abbr.	In/Out
1			
2	TRANSMIT DATA	(TxD)	O
3	RECEIVE DATA	(RxD)	I
4	REQUEST TO SEND	(RTS)	I
5	CLEAR TO SEND	(CTS)	O
6	DATA SET READY	(DSR)	I
7	SIGNAL GROUND		
8	CARRIER DETECT	(CD)	I
2Ø	DATA TERMINAL READY	(DTR)	O
22	RING INDICATOR	(RI)	I

7.212 Parallel Printer Interface

There is one parallel printer port on the board, which is led out by a standard mass-terminated ribbon cable at the 26-pin header labelled J3. The three-pin jumper is used to configure this port as PRL1 or PRL2. The port can be disabled by removing the jumper entirely.

Fig. 7-1: Printer Port Select: Jumper H2

```
                 PRL1                 PRL2

          1    |2    3|             |1    2|    3
```

PC-DOS dynamically assigns device names to the system parallel ports at boot time, by polling to see if a port has been installed at any of three possible addresses: 3BCH, 378H, and 278H. The first port found becomes LPT1, the second becomes LPT2, and the third becomes LPT3. The first address polled is 3BCH, reserved for the parallel port on the IBM Monochrome Adapter.

With the Device Module in its normal configuration, the next address (378H) corresponds to PRL1; the third polled address (278H) corresponds to PRL2. However you configure the parallel port, the DOS device name will depend on the presence or absence of other parallel ports in the system. The recommended configuration is PRL2, since other boards are most likely to occupy the first polled addresses: for example, the IBM Parallel Printer Interface board is addressed at 378H.

When installed, the Print Buffer/Spooler automatically reassigns LPT1 to Baby Blue II's port, with other ports following in address order.

Pin assignments follow the IBM standard.

Table 7-5: Parallel Port Pin Assignments

Pin	Signal	In/Out
1	/STROBE	I/O
2	DATA Ø	I/O
3	DATA 1	I/O
4	DATA 2	I/O
5	DATA 3	I/O
6	DATA 4	I/O
7	DATA 5	I/O
8	DATA 6	I/O
9	DATA 7	I/O
1Ø	/ACKNOWLEDGE	I
11	BUSY	I
12	PAPER END	I
13	SELECT	I
14	/AUTO FD	I/O
15	/ERROR	I
16	/INIT	I/O
17	/SLCT IN	I/O
18-25	GROUND	

7.213 Real Time Clock

CLK is the onboard real-time clock, not to be confused with the system clock; it has no corresponding DOS device name. Microlog-supplied software is used to operate the real-time clock and synchronize the system clock with it.

7.22 PORT ADDRESS DECODING

Devices are addressed via a twelve-bit code yielding a three-place Hexadecimal address. A1Ø and A11 are not used, so the significant bits are AØ thru A9.

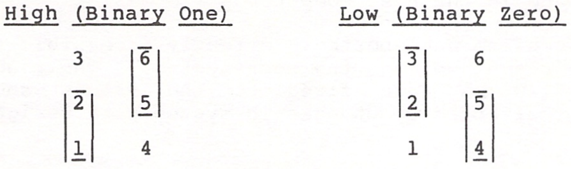
Fig. 7-2: Device Module Address Line Assignments

A9	A8	A7	A6	A5	A4	A3	A2	A1	AØ
H1	XX	XX	XX	S1	S2	S3	XX	XX	XX
Jumper Selec-table	Internal to PAL			Switch Selectable @ SW2			Device Control Lines		

A9, A5, A4 and A3 are user-selectable; a valid decode on these lines selects the Device Module. Normally they are set to correspond to the IBM standard, but can be used for special applications, to alter the address assignment of the entire module. There is no provision for readdressing the individual devices, and except for the parallel port, the devices cannot be disabled.

On SW2, an OFF represents binary 1. H1 is a six-pin jumper, where a binary 1 is represented by connecting pins 1 to 2, and 5 to 6.

Fig. 7-3: A9 Decoding: Jumper H1



For all IBM PC's, set all selectable lines HIGH (to binary 1). All three switches on SW3 should be OFF, and H1 should be jumpered [1<-->2] and [5<-->6]. This will set each device to the IBM standard, except of course for the clock, for which there is no standard I/O address.

Table 7-6: Standard System Port Addresses

Device	Address (Hex)
SER1:	3F8 - 3FF
SER2:	2F8 - 2FF
PRL1:	378 - 37F
PRL2:	278 - 27F
CLK:	338 - 33F

A8, A7 and A6 select individual devices within the Device Module, as shown below.

Table 7-7: Device Select Codes

DEVICE	A8	A7	A6
Not Used	Ø	Ø	Ø
PRL2	Ø	Ø	1
Not Used	Ø	1	Ø
SER2	Ø	1	1
Clock	1	Ø	Ø
PRL1	1	Ø	1
Not Used	1	1	Ø
SER1	1	1	1

A2, A1 and AØ are used to operate the control lines of the particular device (e.g., the USART) enabled by a valid decoding of A9 through A3 - this means that each device occupies an eight-byte wide space in the host computer's I/O map.

All I/O devices are dual-ported, directly accessible to the Co-Processor Module as well as the host system. Their addresses in the Z-8Ø's I/O map are fixed to the values shown below, and are not affected by changes in system port assignments.

Table 7-8: Co-Processor I/O Map

Device	Address (Hex)
SER1:	7Ø - 77
SER2:	3Ø - 37
PRL1:	5Ø - 57
PRL2:	1Ø - 17
CLK:	4Ø - 47

APPENDICES

A. THE CONVERSION UTILITIES

A.1 BIND: THE CP/M-8Ø PROGRAM IN PC-DOS FORMAT

BIND attaches HEADER to CP/M-8Ø programs which are on PC-DOS diskettes, as opposed to CONVERT, which does the same thing to programs on CP/M diskettes. Use BIND when:

- as recommended, you purchase CP/M software published on PC-DOS diskettes, though not yet bound with HEADER.
- you transfer software from a CP/M system by some means which does not directly involve the Co-Processor, e.g. a PC-DOS communications program or the Microlog 8" Disk Controller. (Programs running on the Co-Processor will automatically BIND HEADER to any COM files they write on a PC-DOS disk).
- you update your files with a new version of HEADER.

PROCEDURE

Both BIND.COM and HEADER must be on the same disk in the default, or logged-in drive. Type:

```
c:BIND s:filename.COM d: <CR>
```

BIND first checks for the presence of HEADER in the target file. If the file contains some version of HEADER, it will be replaced with the version currently on your disk. If source and destination are on the same drive, the old filename.COM will be overwritten. This is how BIND is used to update a program with a new version of HEADER.

If filename.COM does not contain HEADER, BIND will respond with the warning:

```
This ".COM" file may be an 8Ø88 file --
if you still wish to bind it,
rename it with extension ".CPM"
```

If you attach HEADER to a native PC-DOS program, the program will no longer run - BIND is making sure that won't happen. If you know you've got a CP/M file, type:

```
RENAME s:filename.COM filename.CPM <CR>
```

Then:

```
     c:BIND s:filename.CPM d: <CR>
```

This will unconditionally attach HEADER to filename.CPM, producing the larger filename.COM. The size of the two files will differ by exactly the length of HEADER.

You may of course rename your file to the CPM extension before running BIND the first time, but be careful: if the file already contains HEADER, it will now be "double-bound", containing two HEADERs, and it won't run. It's safest to probe for the presence of HEADER by attempting to BIND your COM file first, before you RENAME it to CPM.

BIND does not accept global, or "wildcard" filenames, for example:

```
          BIND *.CPM
```

will not match a series of files; instead, it will look for a single file literally named "*.CPM". Since you only BIND each COM file once, the absence of globals shouldn't be a serious handicap.

A.2 CONVERT: ACCESS TO CP/M DISKETTES

Use CONVERT to:

- Move files in either direction between PC-DOS and CP/M.

  To transfer a file, you must have two diskettes, one formatted for PC-DOS (double or single sided), the other for CP/M (single sided only).

- Inspect the directory of a CP/M diskette.

Don't use CONVERT to attach HEADER to a file which is already on a PC-DOS formatted diskette - use BIND instead. Convert requires two disk drives to operate; at least one must be a 5-inch floppy disk drive.

PROCEDURE

Type:

```
c:CONVERT
```

Response:

```
CP/M IBM File Transfer Utility
Version 2.Ø (c) 1982, Microlog Inc.

IBM Disk:____
```

Type the one-letter name of the drive which contains your PC-DOS formatted diskette - no <CR> is necessary. Notice that CONVERT immediately posts your response at the top of the screen. It will continue to do this with each parameter (selection) you supply, forming a "status line" for easy reference. The next prompt is:

```
CP/M Disk:____
```

Type the name of the drive containing your CP/M diskette.

Response:

```
AVAILABLE FORMATS:

1.  NEC PC-8ØØ1
2.  IMS 5ØØØ
3.  DEC VT-18X
4.  Heath/Zenith Soft Sectored
5.  CP/M-86 on the IBM PC

SELECT FORMAT:
```

Select from this list the format that matches your CP/M diskette, and type the appropriate number, 1 through 5. Now the Functions Menu appears:

```
          1.  Copy from CP/M to IBM
          2.  Copy from IBM to CP/M
          3.  Print Directory of IBM disk
          4.  Print Directory of CP/M disk
          5.  Change parameters (restart program)
          6.  Exit program

          ENTER SELECTION:
```

Type a number from 1 to 6 - the entire menu remains on the screen, but the other functions fade to half-intensity, highlighting your choice. Your function remains highlighted until execution is completed.

FUNCTIONS

1. Copy from CP/M to IBM

Type "1" to bring a file into PC-DOS from CP/M. Your screen looks like this:

```
IBM Disk: d:        CPM Disk: s:      CPM format type: formattype

          1.  Copy from CP/M to IBM
          2.  Copy from IBM to CP/M
          3.  Print Directory of IBM disk
          4.  Print Directory of CP/M disk
          5.  Change parameters (restart program)
          6.  Exit program

     ENTER FILE NAME (<return> to exit copy) :
```

You now type:

```
          filespec  <CR>
```

Which is soon replaced by:

```
          COPYING FILE s:filename.ext
```

You may use global parameters in place of filenames and extensions (e.g. *.ext , or *.*).

When the copy is finished, CONVERT says:

```
          ENTER FILE NAME (<return>) to exit) : ___
```

Enter another filename, or type <CR>, returning to the Functions Menu.

2. Copy From IBM to CP/M

Identical to the procedure for Function 1, except of course that now the source is a PC-DOS formatted diskette, and the destination is a CP/M diskette.

3. Print Directory of IBM Disk

This function displays the directory of the disk listed as "IBM disk" at the top of your screen, so you don't have to exit CONVERT to find out which files you've got. If you've got the wrong type of disk in there, you'll get an error. The directory appears at the bottom of the screen and remains there for reference after control returns to the Function Menu.

4. Print Directory of CP/M Disk

Similar to Function 3, except that you get the directory of the disk listed as the "CP/M disk". This is the only way to read the directory of a CP/M diskette under PC-DOS.

5. Change Parameters

When you want to change an entry in the onscreen status line, This function allows you to quickly restart CONVERT from the top, without exiting to system level.

You type "5", the screen clears, and CONVERT begins again with the prompt:

```
          IBM Disk:____
```

6. Exit Program

The screen clears, and the system prompt appears, returning you to PC-DOS command level.

A.3 STRIP: REMOVING HEADER

STRIP removes HEADER from CP/M programs on PC-DOS disks. You won't often use it, because you automatically get the same result when you export files using either CONVERT or a CP/M communications utility (e.g., BSTAM, supplied with your Baby Blue II). It is not necessary to remove HEADER when:

- Debugging a COM file, or for other operations within your system (See "Baby Blue II as a CP/M Development System" in "OPERATION: RUNNING CP/M PROGRAMS").

- Updating your CP/M programs with a new version of HEADER. BIND automatically removes the old HEADER before attaching the new one.

You must use STRIP before exporting a file via some none-CP/M utility, such as a native PC-DOS communications program.

PROCEDURE

Type:

```
c:STRIP s:filename.COM d:
```

where filename.COM is a bound CP/M program. This will produce the output file, filename.CPM on the destination drive. If the COM file does not contain HEADER, STRIP responds with the error message:

```
HEADER NOT ON FILE
```

A.4 KEYFIX: AUTOMATING YOUR KEYBOARD

KEYFIX allows you to program more than fifty "definable" function keys to output any character string (sequence of keystrokes), up to 8Ø characters long. Because the function key definitions reside in HEADER, KEYFIX can only be used with CP/M programs running on Baby Blue II.

PROCEDURE

Type:

```
        c:KEYFIX s:filename <CR>
```

Response:

```
        ENTER THE KEY YOU WISH TO DEFINE, <Q>-TO EXIT
```

Press a "definable" key, as explained below. The screen clears, and you see this:

```
        KEY SELECTED:                    [FUNKEY]

        CURRENTLY DEFINED AS:

        [current designation]

        TO DEFINE A KEY HIT <RETURN> TO LEAVE THE KEY UNCHANGED
        HIT ANY OTHER KEY.
```

Press "Return" (<CR>). All information currently on the screen remains there, and in addition,

```
        To define a key, enter a string of characters. Define-
        able keys may not be used. They will be ignored.
        Control characters are okay. The maximum length of one
        entry is 8Ø. Any characters exceeding this size or the
        total table size will be truncated.

        TO END THE STRING, ENTER THE KEY YOU ARE DEFINING.
```

Now type:

```
        [character string] <FUNKEY>
```

The symbol "<FUNKEY>" means "press the key you are currently defining" - that's how KEYFIX knows you're done. Why not <CR>? Because you might use that as part of your definition - if you do, it will display as "^M".

The screen clears, and you're back at the top, ready to start on another key. This is the only way to exit once you begin to define a key: whatever you see at the bottom of your screen is stored literally as the key definition. If there is nothing at the bottom of your screen, your key will be stored as a "null", meaning that when you press it during program execution, nothing at all will happen.

Restarting KEYFIX

You can always get back to top of KEYFIX by pressing the currently selected key, but remember that once you have started to define a key, the bottom of your screen will be stored as the new key definition, even if there is nothing there.

To Inspect the Definition of a Key

Press the key to display its current status; press the key again, and you're back at the top of KEYFIX.

To Finalize Your Entries

When you've finished setting up your keys, type "Q" ("Quit") at the top of KEYFIX, to return to PC-DOS. Always exit in this way if you want to save your entries - KEYFIX will write them all to disk, in the HEADER attached to the target program. Now, whenever you call that program, your keys will work as you have programmed them.

To Clear a Function Key

You may want to erase the definition of a function key simply to disable it, but the main reason is to clear table space for long entries to other keys, as explained below.

Go to the initial prompt, and press the key. Now press "Retrn": at this point, your cursor is standing on an empty line at the bottom of the screen: immediately press the selected key again, entering a "null", or inactive, string for the selected key.

Correcting Errors

There is no practical way to correct an error, except to start again. Press the selected key twice, then press "Retrn", to start over.

Duplicate Definitions

To KEYFIX the same definitions to a number of programs, rename HEADER to HEADER.COM - now you can KEYFIX HEADER itself. Then change the name back to HEADER, and BIND it to your programs.

Sorry, you can't run KEYFIX on KEYFIX itself - it's not a CP/M-8Ø program.

SCOPE

Definable Keys

You can define a total of 56 different function keys, divided into four registers. The unshifted, or "normal" register consists of:

```
1Ø Function Keys: <F1>-<F1Ø>.
 4 Arrow Keys: <Up>, <Down>, <Left>, <Right>.
 6 Others: <Home>,  <End>,  <Pg Up>, <Pg Dn>,
           <Delete>, <Insert>.
2Ø Total
```

To select an unshifted key, type:

```
<Function key>
```

For example,

```
<F1>
```

displays:

```
KEY SELECTED:               FUNCTION 1
```

The "control" register consists of:

```
1Ø Function Keys <F1> - <F1Ø>
2 Arrow Keys: <Left>, <Right>
4 Others: <Home>, <End>, <Pg Up>, <Pg Dn>
16 Total
```

To select one of the 16 keys in the "control" register, press <CTRL> and the function key simultaneously. For example:

    <CTRL HOME>

displays:

```
KEY SELECTED:            CTRL HOME
```

The "Shift" and "Alternate" registers each contain 1Ø keys:

    1Ø Function Keys <F1> - <F1Ø>
    1Ø Total

For example,

    <SHIFT F1> (or <ALT F1>)

displays:

```
KEY SELECTED:            SHIFT F1  (or  ALT F1)
```

Default Definitions

HEADER comes with all function keys predefined as shown in Table 4-5. The peculiar symbols beginning with "^@" are sequences expected by the PC-DOS line editor: you use this facility, for example, every time you delete a character while typing a DOS command. If you redefine these keys, you will disable the corresponding DOS line-edit function during execution of your KEYFIXED program. This will only be a problem in the rare case where a program does not have its own line-edit functions.

Allowable Strings

You may enter any character as part of the definition for a function key, except that nothing will happen if you try to enter one of the definable keys. This means that one function key cannot call another, nor can a function key call itself.

The so-called "parallel functions" - ALT, SHIFT and CTRL - are always used as part of a two-keystroke combination. If you type <CTRL> nothing happens; however, if you hold <CTRL> down and type another character, for example "C", you get this on your screen:

    ^C

Your system uses a caret ("^") to represent the "hidden" <CTRL> keystroke - don't confuse it with the "real" caret, or <Shift 6> on your keyboard. Your system interprets this "^C" not as two characters, but as one: the normally non-printing command sequence <CTRL C>.

If you type:

    <SHIFT 6><C>

You'll also see:

    ^C

but this is interpreted, and normally printed (or displayed) as the two characters "^", and "C".

Some keys, such as <Tab> and <Retrn> will post peculiar control codes on the screen as you define a function, but don't worry - during program execution your system will properly interpret these as commands, and will not print or display unintended characters.

Space Limitations:

The longest definition you can enter is 8Ø characters - if you enter too many, the following message appears at the bottom of your screen:

    ERROR: NO SPACE AVAILABLE

However, there is also a hidden limitation: your total entries, for all functions, cannot exceed the size of the "table" which has been reserved for them. There are actually two tables, each of 256 characters, divided between the possible function keys as follows:

Table 1   256 Characters Total

```
10 Normal Functions: <F1> - <F10>
10 Shift  Functions: <SHIFT F1 - <SHIFT F10>
20 Keys Total
```

Average 12.8 Characters per Function.

```
Table 2                                      256 Characters Total

        1Ø ALT    Functions: F1 - F1Ø
        1Ø CTRL   Functions: F1 - F1Ø
        6  Cursor Controls: <UP>,<DOWN>,<RIGHT>,<LEFT>,
                            <CTRL RIGHT>,<CTRL LEFT>
        1Ø Other Functions: <HOME>,<END>,<PG UP>,<PG DN>,
                            <INS>,<CTRL HOME>,<CTRL END>,
                            <DEL>,<CTRL PG UP>,<CTRL PG DN>
        ________________________________________________
        36 Functions Total

        Average 7.1 Characters per Function.
```

If you have some really long strings, you may want to use the functions of Table 1, in order to save space in Table 2. In most applications, the Arrow and Other functions tend in any case to be short strings, involving two or three characters, and it is quite natural to save elaborate instructions for the Normal keys F1 - F1Ø.

HEADER's original default definitions occupy most of the table space already - this is why you may suddenly receive a "No Space Available" error after only a moderately long string. To get more space, clear some functions you aren't using by redefining them as nulls.

EXAMPLE

The following exercise programs a hypothetical text editor named TEDIT.COM to output a name and address at the touch of function key F1.

Run KEYFIX on TEDIT:

```
    KEYFIX TEDIT <CR>
```

Select F1:

```
    <F1>
```

Elect to define F1:

```
    <CR>
```

Enter key definition:

```
    Ethel and Rupert Snoot^M35 Tar-Boosh Ln.^MHog-Jaw, N.D.<F1>
```

Exit KEYFIX:

Q

The symbol "^M" appeared each time you pressed <Retrn> while entering the key definition itself. It represents <CTRL-M>, which is properly interpreted as a carriage return by the computer.

Now, whenever you press <F1> during a TEDIT session, the following text appears:

Ethel and Rupert Snoot
35 Tar-Boosh Ln.
Hog-Jaw, N.D.

This occupies 53 of the 256 characters available in Table 1. Note that all symbols count, including "^M" (1 character) and spaces.

Don't limit yourself to simple text entry - you can KEYFIX anything you can type, especially command sequences which can turn a sound but awkward program into a high-performance vehicle. Any often-repeated complex series of keystrokes is a candidate for a function key "mini-program" - a complicated graphic figure for example, or a text editing sequence. With eighty characters at your disposal, you can achieve spectacular results.

Many powerful software packages are so complicated that you end up seldom using many functions simply because it's too much trouble to remember all the codes. A logically arranged KEYFIX is often the answer - you'll find that the function keys fall naturally into groups for easy reference.

NOTES:

B. DOS 2.Ø CONFIGURATION FILES

Each time you boot up (start) a PC under DOS 2.Ø, the operating system searches the root directory of the system disk for a "configuration" file called CONFIG.SYS. This file, like an AUTOEXEC.BAT file, contains a series of commands which are immediately executed before control passes to you, the operator. Unlike the BAT file, however, the configuration file uses only a special set of commands, as detailed in Chapter 9 ("Configuring Your System ") of your IBM DOS 2.Ø Manual.

You can create a new CONFIG.SYS file with the following sequence:

```
COPY CON: CONFIG.SYS <CR>
COMMAND <CR>
COMMAND <CR>
.
.
.
<CTL-Z>
```

If you already have a CONFIG.SYS file, be sure to include all of its commands in the new one. It may be easier to use a text editor to create the file, but be careful: many editors add special, unseen control codes to your file. If your editor has a "non-document" mode, use that. TYPE the competed file at the system level - if you don't see any strange symbols or extra characters, the file is OK.

You'll need to use the DEVICE command to install various Baby Blue II facilities, especially the Print Buffer/Spooler and the RAMdisks you create with RAMDISK.COM. The general form is:

```
DEVICE=filename.ext <CR>
```

where "filename.ext" is the name of the "device driver" file to be installed to your operating system.

The complete list of drivers for Baby Blue II is:

```
DEVICE=PRL.SYS <CR>
DEVICE=SER.SYS <CR>
DEVICE=RAMDISK.SYS <CR>
```

PRL.SYS and SER.SYS appear on the diskette supplied with Baby Blue II; RAMDISK.SYS is the default name given to a RAMdisk driver produced by RAMDISK.COM. You may include additional RAMdisks under names of your own choosing.

All device driver files named in the configuration file must appear in the root directory of your system disk at boot-time, or you'll get the error:

```
        Bad or missing filename.ext
```

See your DOS manual for other features of the CONFIG.SYS file.

C. APPLICATIONS NOTES

C.1 EMULATING THE "SAVE" FUNCTION: DEBUG.DDT

Although DDT and similar utilities work on Baby Blue II, they're not very useful if you can't write the results of your work to disk. Normally, you would use the CP/M resident SAVE command, but this command is not available under PC-DOS. A neat solution is to run DDT under the control of DEBUG, using DEBUG's Write facility in much the same way you would use SAVE. Note that unlike SAVE, DEBUG makes it very easy to compute the file size to write, because it's given simply as the number of bytes in hexadecimal.

The screen display is shown in boldface; comments follow the semicolon.

```
A>DEBUG DDT.COM <CR>       ;run DDT under DEBUG

-G <CR>                    ;start DDT


**********************************************************************
*                         DDT STARTS HERE                            *
**********************************************************************


xxxxxxxxxxxxxxxx           ; DDT's signon message

-I [filespec] <CR>         ; specifies target file

-R  <CR>                   ; DDT reads the file

 NEXT  PC                  ; DDT    responds  with  the  next   free
 nnnn pppp                   address  following the file,  and  the
                             assumed program counter (1ØØH for .COM
                             files).   You can use this information
                             to  determine the size of  the  loaded
                             file.

-[do whatever you want]    ; modify target file under DDT.

                           ; IMPORTANT:  before  exiting DDT,  find
                             out  how many bytes  you want to save,
                             and also the starting memory  location
                             (usually 1ØØH).
```

```
-D3 <CR>                  ; displays  contents  of  Co-Processor
                            memory  location ØØØ3H in first  posi-
                            tion.  Ordinarily,  this would be the
                            CP/M I/O Byte,  which is not implemen-
                            ted in HEADER. Instead, the high-order
                            nibble  contains  the  segment  number
                            occupied  by the Co-Processor's memory
                            (Bank  IV) in the host's memory  (e.g,
                            for  a  Bank  IV  with  base  address
                            EØØØØH,  this command  will   display
                            "EØ".  You'll need this information to
                            Write your file under DEBUG.

-<CTRL-C>                 ; exits DDT, returns to DEBUG

program terminated normally ; DDT signs off;


*****************************************************************
*                         DDT ENDS HERE                         *
*****************************************************************


-N<filespec> <CR>         ; specifies DEBUG output file

-RCX <CR>                 ; calls CX register

-CX:xxxx <CR>             ; enter  number  of bytes to save (HEX)

-Wsegment:offset <CR>     ; write output file to disk:   enter Co-
                            Processor  segment  number,   followed
                            by  colon,  followed  by  the starting
                            address  to save  within  Co-Proces-
                            sor  memory (usually the beginning  of
                            the program, at 1ØØH).

-Q <CR>                   ; exit DEBUG
```

D. DIAGNOSTICS AND TROUBLESHOOTING

D.1 BABY BLUE II CONFIDENCE TEST

TESTBB2.EXE is a diagnostic program which tests all hardware functions on the Baby Blue II board, including memory. Use it any time you suspect a physical malfunction on the board. It is included so that you can distinguish problems related to Baby Blue II from faults in other parts of your system, including possible problems with software.

PROCEDURE

Type:

```
        TESTBB2 <CR>
```

System Response:

```
        BABY BLUE II CONFIDENCE TEST VERSION 1.XX
        COPYRIGHT(C), 1984, MICROLOG, INC.

        1  Z8Ø LOCATION TEST             PASSED
        2  Z8Ø INTERRUPT TEST            PASSED
        3  8Ø88 INTERRUPT TEST           PASSED
        4  MEMORY TEST                   PASSED
        5  8Ø88 ADDRESS LINES TEST       PASSED
        6  Z8Ø ADDRESS LINES TEST        PASSED
        7  COM1 TEST                     PASSED
        8  COM2 TEST                     PASSED
        9  PARALLEL PORT TEST            PASSED
        1Ø CLOCK TEST                    PASSED

        ** TESTING SUCCESFULLY COMPLETED **
```

If you see this, you know the problem is definitely elsewhere, either somewhere else in your system, or in software.

If a test is unsuccessful, you'll see the word "FAILED" in the right hand column. Some tests can't be carried out if another test fails: these will be marked "INVALID".

You can also use standard IBM diagnostics to check all components except the real-time clock and the Z-8Ø. If your board fails TESTBB2, or you suspect a hardware fault in your system, turn to "Troubleshooting", next.

D.2 TROUBLESHOOTING

You're here because your system failed to behave normally after you installed Baby Blue II, or because TESTBB2 returned an error. At worst, you may have to return your board to Microlog for service, but that's going to take some time, so you're hoping to find another answer. Our experience indicates that very few boards actually fail after factory testing, and that apparent faults are usually due to some factor overlooked during the installation. Most boards received by our service department turn out to be in perfect working order.

Here are some common faults:

- At boot-time, an error message appears at the top of your screen (e.g., "1ØAA 2Ø1", Parity 2).

- Your machine won't boot at all. Either you don't get a cursor, or you only get a cursor, or the system locks up as the titles come on.

- You get erratic operation, often associated with a particular utility or peripheral device.

- TESTBB2 returns various errors.

All of these appear to be "hard" errors, indicating defective hardware. Since Baby Blue II is the only new factor, it is natural to assume that the board is defective. However, problems may arise from conflicts between elements in your system, where neither part is in itself at fault. Such conflicts can be resolved, but it's important to know first where the fault lies.

Use the following procedure to isolate the problem. Remember, before you touch any boards, <u>be sure that you turn power OFF, and disconnect the power cable from your System Unit</u>.

First, remove Baby Blue II, and turn all the DIP switches ON and OFF two or three times; then reset them as recommended and try reinstalling the board. DIP switches are the board's only mechanical component, and they sometimes get "tired" - exercising them is often a quick fix. Be sure they're firmly set.

The next step is to isolate Baby Blue II - you don't know the board is defective unless you have eliminated all other factors which might affect it. This means removing as many other boards as possible, stripping your computer down to bare essentials. Obviously, you won't remove your monitor adapter or disk drive controller, since without them your computer won't run anyway, but any extra memory boards or peripheral device interfaces should come out. Also, your boot disk should contain a plain-vanilla operating system - if you've made any software installations to your working copy of DOS, make a new copy of your original operating system for test purposes.

Before you start pulling boards, make some notes, if you haven't done so already. Don't change anything you can't undo, and make sure you've recorded the settings of any switches you can see. On most machines, you'll have to change the switches on the motherboard after removing any memory boards. When you're done, you should have returned your machine to its original factory configuration.

Make sure the machine works normally in every way, then install Baby Blue II. Use the factory switch settings shown in Section XX, but turn OFF switches 1, 2 and 3 on SW3, to disable all three banks in the RAM Module. Don't do any customizing at this point - we're trying to find out whether the board is OK, so keep it simple. You now have a complete isolation test - the only change to your working machine is plugging in Baby Blue II - you haven't even changed a switch position.

If the board doesn't work now, try a different expansion slot. You'd be surprised how often this works, even though all slots are theoretically the same. The manufacturer's documentation won't mention it, but many machines have shown problems with the physical distribution of signals on the expansion bus. The same principle applies if you have an expansion chassis, only more so: try your putting your board in the System Unit, or vice-versa. If the board still doesn't work, it's time to turn to the Warranty section and get help.

If your board works now, enable the RAM Module by turning ON the appropriate switches on SW3, and reinstall Baby Blue II, making all necessary changes to switches on the motherboard. If there's a problem now, either there's something wrong with Baby Blue II's RAM, or you've made some mistake in setting the switches. Review the section entitled "Installing Memory".

If everything checks out OK, you can start reinstalling your various options until you find the one that doesn't like Baby Blue II. The offending board will be trying to use one or more addressesalready assigned to Baby Blue II, either an I/O Port Address, or a memory Page.

<u>Memory Conflicts</u>

You can check for conflicts with the RAM Module by successively disabling RAM banks, starting with Bank III and working down. If the problem goes away, you've found the overlap. Diagnose suspected conflicts with Co-Processor memory by altering SW1 to map Bank IV into some other available Page.

Serial Ports

Does the problem board contain any serial ports? They must be disabled - you can't have more than two serial ports in your system. If you can't disable the ports, you can't use the board.

Parallel Ports

Check for an address conflict. You can change the address of Baby Blue II's parallel port by switching the jumper at H2, but remember, you can't have more than three parallel ports in your system.

Other Port Conflicts

Consult the manufacturer's documentation for any other devices which might interfere with the Clock's port address. The Z-8Ø also has a port address, which you can find in the Co-Processor Reference Manual. This address changes automatically when you alter the Page assignment of Bank IV, so if you suspect trouble here, try setting SW1 for another available Page.

Contact your dealer for any problem you can't fix - he is in the best position to help, since he is on the scene and can directly observe the symptoms. If you can't get satisfaction locally, contact Microlog at:

Technical Support
Microlog Inc.
222 Route 59
Suffern, NY 1Ø9Ø1
914-368-Ø353

When reporting any problem, be sure you include the following information:

- Serial number, dealer's name, and date of purchase.
- System configuration, as outlined in Section 2.26.
- A short history of your attempts to fix the problem, including contacts with your dealer.

E. WARRANTY INFORMATION

DISCLAIMER

Microlog, Inc. makes no representations or warranties with respect to the software programs included herein and specifically disclaims any implied warranties of merchantability or fitness for a particular purpose. Furthermore, Microlog, Inc. reserves the right to revise the software programs included herein and to make changes from time to time in the content thereof. Microlog, Inc. is not obligated to notify any person or organization of such revision or change.

LIMITED WARRANTY

Microlog warrants the original user of this hardware product that it is free from defects in materials and workmanship for a period of ninety (9Ø) days from the date of shipment from Microlog or Dealer to the original end user. If any Microlog product becomes defective within the first ninety (9Ø) days from the date of shipment, Microlog will replace or repair, at its sole option, that unit which proves to be defective. This warranty is void if, in the sole opinion of Microlog, the product has been subject to abuse, misuse, or modification. All warranties are non-transferrable. This warranty is in lieu of any other warranty, expressed or implied, and in any event, is limited to product repair or replacement. Microlog shall not be liable for any incidental or consequential damages of any kind resulting from use of this product.

IF YOUR BOARD FAILS TO OPERATE

Microlog rigorously tests every product to insure that our boards will not fail in the field. However, even with this level of testing, problems do occur. If your board requires repair, please refer to the return procedure outlined below.

RETURN POLICY

All defective products in question, whether purchased directly from Microlog, or through an authorized dealer, must be returned to Microlog for repair or replacement according to the conditions set forth in the limited warranty.

Prior to returning any defective product for replacement or repair, you must receive a RMA (Return Materials Authorization) number from Micrologr. When requesting an RMA number, please provide the following information:

1. A brief description of the problem.

2. Serial number of the unit to be returned.

3. The name of the dealer from whom the unit was purchased.

4. The date of purchase.

Upon receipt of an RMA number from Microlog, pack the unit along with a copy of your proof of purchase and ship it prepaid to Microlog. Items received without proof of purchase cannot be serviced and will be returned at the sender's expense. The RMA number must be marked on the outside of the shipping container.

Repaired units, still in warranty, will be shipped prepaid by UPS surface. Customer requests for any method of shipment other than UPS will be charged to the customer. All requests for air freight will be shipped collect.

All products found to be out of warranty, returned for repair or testing, will be assessed a minimum of $50.00 service charge. If the charge for repair is to exceed $50.00, the customer will be notified for authorization prior to Microlog's repair of the unit. An RMA number is also required for out of warranty repair.

All prices are subject to change without notice.

Ship to:

Microlog Inc.
222 Route 59
Suffern, N.Y. 10901
(914) 368-0353