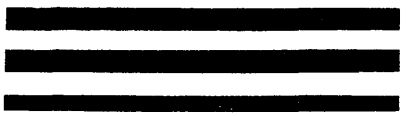
CPU Plus User's Manual

Version 2.0



Microlog™
A TU INTERNATIONAL COMPANY

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

TABLE OF CONTENTS

TABLES

1. INTRODUCTION

1.1 WHO OR WHAT IS BABY BLUE?

Baby Blue is a single-board microcomputer which enables the IBM Personal Computer to run programs written for the CP/M-80 operating system. Although small enough to fit in a single expansion slot, it contains a high-speed Z-80B microprocessor and a full 64 Kilobytes of memory, making it actually more powerful than most first-generation microcomputers.

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, printers, etc. The closeness of this "mother-child" relationship is 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 operate Baby Blue - 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 program which uses Baby Blue - in effect, CP/M-80 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.

By the way, Baby Blue doubles as a 64K memory expansion, although it can be placed outside system memory if you're pressed for space. You also get programmable function keys (KEYFIX), a file transfer utility which gives you access to diskettes in a number of CP/M formats (CONVERT), and a communications program with sophisticated error checking for exchanging files with other computers (BSTAM).

Baby Blue runs in most IBM-PC compatibles; Microlog manufactures a companion product, BabyTex, for use in the Texas Instruments Professional Computer.

1.2 SYSTEM REQUIREMENTS

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

- 64 Kilobytes system RAM (128K recommended for some applications).
- One 5" floppy disk drive (one other drive recommended; it need not utilize 5" floppies).
- PC-DOS (or MS-DOS) Version 1.1 or 2.0.

1.3 ABOUT THIS MANUAL

Although Baby Blue is simple to install and use, it is also a subtle, mature design with a broad history of proven applications. The board's many special features and other esoterica make this a long book, but it's structured to give you easy access to the information you need at any level.

To get you up and running quickly, we begin with a simplified installation procedure, requiring nothing more complicated than physically plugging Baby Blue into an expansion socket. The factory configuration sidesteps the issues of switch settings and memory mapping entirely, avoiding the painful part of most installations. There is one drawback: you won't get a 64K memory expansion with this method, and if this is important to you, you'll have to come back later to reconfigure the board. However, it is definitely the quick way to start running programs on your Baby Blue.

With your board installed, you can skip to Chapter 3, "Operation" - within half a dozen pages, you'll already have run a sample CP/M program. The remainder of Chapter 3 will fill you in on the fine points of Baby Blue's capabilities, and the Appendix contains in-depth reference sections for all the Microlog utilities.

The rest is background and technical information about the inner workings of Baby Blue's hardware and software. It's meant for the interested user as well as the experienced programmer who wants to design his own applications.

1.4 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 file you wish to invoke.

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 HARDWARE INSTALLATION

2.1 THE EASY WAY

If you're gazing with horror at the maze of charts and instructions in this chapter, we've got good news: you probably won't need them. If you've ever installed an expansion board, just read to the end of this section; then open your System Unit, plug Baby Blue into an expansion slot, and close up (if at all possible, avoid the leftmost slot - the one farthest from the power supply).

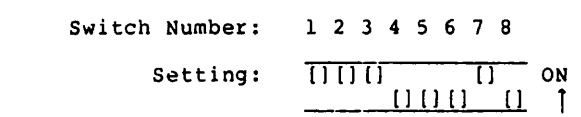
Even if you've never been inside your System Unit, we've included step-by-step instructions for all IBM PC's (See 2.41). This section is also generally valid for IBM-compatibles not specifically covered - use it in conjunction with the manufacturer's documentation for your machine.

The only tool required is a medium blade-type screwdriver.

2.11 CHECK FACTORY SWITCH SETTING

Before beginning, check the "DIP" switch unit at Baby Blue's right center - it's a brightly-colored rectangular block containing eight tiny sliding switches, with numbers to match. It's also marked "ON" and "OFF" - the switches are set by sliding them in the indicated direction, and should come from the factory as shown below. Reset any switches that are incorrect.

Table 2-1: Factory Switch Setting



2.12 RUN DIAGNOSTICS

Just before replacing the System Unit cover, make sure your system powers up and otherwise behaves normally; then insert a working copy (never the original) of your Baby Blue diskette, and type:

TESTZ80 <CR>

This will test your hardware installation, including all circuitry on Baby Blue itself. If TESTZ80 returns any errors, or your system behaves abnormally, turn to the Appendix under DIAGNOSTICS.

2.13 CUSTOMIZATION

The standard installation has one drawback - it doesn't let Baby Blue double as a 64K memory expansion. The rest of this chapter is about alternate switch settings - use them when you want to add Baby Blue's 64K to system memory or when the factory configuration proves unsuitable. As you'll see, choosing the proper setting is complicated, because it involves your total system configuration - the mother board, other expansion boards, and Baby Blue itself. We've given you a configuration which works for all current machines as originally manufactured; only a few unusual systems will absolutely require customized switch settings.

We suggest that everyone take the simple route first, if only to be sure that Baby Blue is functioning properly. Try the board out, run some programs, and get comfortable. Then come back if you like, to customize the installation and pick up your bonus 64K.

2.14 STATIC ELECTRICITY

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 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.

2.2 OPTIONS: THE HARD WAY

You're here because you're not satisfied with the factory configuration: most likely you want to use Baby Blue's 64K for system memory, as well as for running CP/M programs. We'll begin with some background information, paying special attention to unusual factors which may affect your installation. Even if you have some experience, please scan these introductory notes to see if there are any problems you may have overlooked.

2.21 WHAT THE SWITCHES MEAN

Although Baby Blue is a self-contained microcomputer, your operating system sees it as a simple 64K memory expansion. Memory is divided into "Pages", or "Segments" of 64K each; when you set Baby Blue's switches, you assign it to a single Page number such as "1", "2", etc. The number must be unique, because your system uses it to locate this particular block of memory: no two physical blocks of memory can share the same Page number.

"System memory" is general-purpose memory available to the operating system. It includes all Pages starting with "0" (which is always on the mother board) and counting up to the first empty Page. Any memory above the first empty Page is excluded from system memory, meaning that it can only be used for special purposes.

The factory setting assigns Baby Blue to the highest numbered Page not reserved by the operating system, which is Page "E" in hexadecimal notation (Page "14" in our normal way of counting). Since this is outside the normal range of system memory, you don't have to worry about the schemes used by different computers to reserve parts of low system memory, or about other expansion boards (only a few have been designed to use this Page). The result is a simple, universal installation procedure - but you don't get a 64K expansion to system memory.

The following instructions assume that you want to map Baby Blue contiguously at the top of system memory, to gain a 64K expansion.

2.22 BASIC PROCEDURE

To map Baby Blue into system memory, refer to Table 2-2 ("Baby Blue DIP Switch Settings"), after determining how many Kilobytes of memory you have, including Baby Blue. Match this number under "New Total Memory", and read across to determine your Page number and corresponding switch settings. This will put Baby Blue's starting address 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 0, so Baby Blue goes on Page 1. If you have 64K plus a 256K expansion board, you've used up Pages 0,1,2,3 and 4 (five Pages of 64K each, or 320K), so Baby Blue goes on Page 5.

2.23 AVOIDING "RESERVED" MEMORY

Some machines effectively reserve low-numbered Pages for memory chips to be installed in sockets directly on the mother board. In these cases, you will have to pretend you have no less than a certain amount of memory installed when you use the chart to set Baby Blue's switches. 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 possible switch setting will put Baby Blue on Page 4, corresponding to 32ØK total memory. This is true even if you actually have less than 32ØK, including Baby Blue.

Note that if the mother board has not been fully populated, a gap will appear in the sequence of memory Pages: if you have memory assigned to Pages Ø and 1 (128K), with Baby Blue assigned to Page 4, there is a gap at Pages 2 and 3. In this case you can't have a 64K memory expansion anyway, so you might as well stick to the standard installation.

2.24 RESOLVING CONFLICTS

You may need to insert Baby Blue somewhere in the middle of system memory, rather than at the very top. For example, most RAMdisk software is designed to use all of system memory above some predefined Page. If Baby Blue's memory is at the top of system memory, the RAMdisk will attempt to use it at the same time as Baby Blue's microprocessor. The system will see contradictory information and shut down in confusion. The answer is to set Baby Blue as low as possible in system memory, that is, immediately after the memory on the motherboard. Then tell the RAMdisk software to begin using memory starting somewhere above Baby Blue.

If you don't expect any difficulties, just go ahead and install Baby Blue at the top of system memory, but should a problem develop, or if there is any question of compatibility with another expansion board, the best procedure is to remove all expansion memory boards and complete the Baby Blue installation, including testing, as if the other boards did not exist. This will put Baby Blue's memory immediately above the memory on the mother board. Then install the other boards, and reset all switches per the manufacturer's instructions, counting the additional 64K of Baby Blue memory which you have just installed.

The Quadram 256K Quadboard and the IBM 32K expansion memory board are known to conflict with Baby Blue unless you position them above Baby Blue in system memory. Many other RAM boards will not permit any memory to be installed above them unless they are fully populated, even if the empty banks are theoretically disabled.

The factory setting avoids such complications, which arise only when you make Baby Blue part of system memory.

2.25 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, the "PC-2", which sockets 256K of RAM 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 the amount of system memory. Since the factory setting excludes Baby Blue from system memory, the standard installation doesn't affect the mother board.

2.3 CUSTOMIZATION NOTES

2.31 DETAILED INSTRUCTIONS

This Section covers the fine points of configuring for specific machines. Select yours from the following:

(2.34) IBM PC 1
(2.35) IBM PC 2
(2.36) IBM PC/XT
(2.37) Other IBM-Compatible PC's

2.32 BABY BLUE DIP SWITCHES

A master chart of possible Baby Blue switch settings appears in Table 2-2: refer to it for all installations. To map Baby Blue into system memory, find the switch setting which corresponds to the Kilobytes of total memory installed in your system, including Baby Blue's 64K. For other applications, assign Baby Blue to the desired Page (Segment). Switches 1 through 3 must always be ON, switch 8 must always be OFF.

2.33 NOTE SYSTEM CONFIGURATION

Before going on, you should note your "system configuration" as it affects Baby Blue, 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.

Table 2-2: DIP Switch Settings

New Total Memory	Switch Setting 1 2 3 4 5 6 7 8		Baby Blue Memory Page/Segment
128K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ]	ON ↑	1
192K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ]	ON ↑	2
256K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ]	ON ↑	3
320K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ]	ON ↑	4
384K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ]	ON ↑	5
448K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ]	ON ↑	6
512K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ][ ]	ON ↑	7
576K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ]	ON ↑	8
640K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ]	ON ↑ ← ISIS	9
704K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ]	ON ↑	A*
768K	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ][ ]	ON ↑	B*
N/A	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ]	ON ↑	C*
N/A	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ][ ]	ON ↑	D*
N/A	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ][ ]	ON ↑	E*
N/A	[ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][ ][ ][ ][ ]	ON ↑	F*

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

2.34 IBM PC-1

The PC-1 was IBM's first Personal Computer, manufactured until about March 1983. It sockets 64K on the mother board, was supplied with DOS 1.1, and has five expansion slots.

Find the two DIP switch blocks in the middle of the 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 (are you sure you're looking at the right switches?).

If you are mapping Baby Blue into system memory, calculate your total memory in Kilobytes, including the 64K on Baby Blue. Find the resulting figure under "New Total Memory" in Table 2-3, and set SW 2 on the mother board to match the corresponding line of the chart.

The PC-1 addresses a maximum of 544K as system memory. Baby Blue can still be mapped above that point, but you won't see additional system memory. Set Baby Blue's switches according to Section 2.31 (Baby Blue DIP Switch Settings), avoiding the reserved Pages listed below.

RESERVED PAGES (Hex): A, B.

Table 2-3: IBM PC-1: Mother Board SW2 Settings

New Total Memory	Switch Setting 1 2 3 4 5 6 7 8	
128K	[] [] [] [] [] [] [] []	ON ↑
192K	[] [] [] [] [] [] [] []	ON ↑
256K	[] [] [] [] [] [] [] []	ON ↑
320K	[] [] [] [] [] [] [] []	ON ↑
384K	[] [] [] [] [] [] [] []	ON ↑
448K	[] [] [] [] [] [] [] []	ON ↑
512K	[] [] [] [] [] [] [] []	ON ↑
544K+	[] [] [] [] [] [] [] []	ON ↑

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

2.35 IBM PC-2

This is the second generation of IBM PC's, featuring a motherboard which can hold up to 256K of RAM before it is necessary to install additional memory boards. The left edge of the mother board is marked, "64KB-256KB CPU".

Calculate your total system memory in Kilobytes, including Baby-Blue's 64K - this figure is your "New Total Memory". The PC-2 reserves the first four Pages of system memory (0,1,2 and 3), or the first 256K, for memory installed directly on the mother board. Therefore, your first available Page is 4, configured as follows:

New Total Memory	Switch Setting 1 2 3 4 5 6 7 8		Baby Blue Memory Page/Segment
128-320K	[][][][] [][]	ON	4
	[] []	↑	

For total memory greater than 320K, refer to Table 2-2 ("BabyBlue DIP Switch Settings"). Avoid the reserved Pages listed below.

If your calculated figure for "New Total Memory" is less than 320K (i.e., if your mother board is not fully populated), Baby Blue cannot add 64K to system memory, and you shouldn't change any switches on the mother board. If your mother board was fully populated, you now have available 320K or more total system memory, and should set SW2 on the mother board as shown in Table 2-4.

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

Table 2-4 IBM PC-2: Mother Board SW2 Settings

New Total Memory	Switch Setting 1 2 3 4 5 6 7 8	
128K	[] [] [] [] [] [] [] []	ON ↑
192K	[] [] [] [] [] [] [] []	ON ↑
256K	[] [] [] [] [] [] [] []	ON ↑
320K	[] [] [] [] [] [] [] []	ON ↑
384K	[] [] [] [] [] [] [] []	ON ↑
448K	[] [] [] [] [] [] [] []	ON ↑
512K	[] [] [] [] [] [] [] []	ON ↑
544K	[] [] [] [] [] [] [] []	ON ↑
576K	[] [] [] [] [] [] [] []	ON ↑
640K	[] [] [] [] [] [] [] []	ON ↑

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

2.36 IBM PC/XT

This is the new IBM super-PC, which comes standard with an IBM-installed Winchester hard disk, DOS 2.0, 128 Kilobytes of memory, and seven expansion slots. The mother board has sockets to receive 256K of memory, but unlike the PC-2, the XT will accept memory installed in the expansion bus before the mother board is fully populated.

Calculate your total system memory in Kilobytes, including Baby Blue's 64K. Find this number under "New Total Memory", in Table 2-2 "Baby Blue DIP Switch Settings", and set the switches on Baby Blue accordingly, but avoid the reserved Pages listed below.

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. The XT is also unusual in that although it 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.

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

2.37 OTHER PC'S

All the machines listed below reserve varying amounts of system memory for RAM to be installed directly on the mother board. Unless the mother board is fully populated (i.e. unless total memory equals or exceeds the figure given below), Baby Blue won't act as a memory expansion - you should not change any switches on your mother board or otherwise instruct the operating system to recognize Baby Blue's 64K as additional system memory. The standard configuration is probably preferable in such a case.

Set the switches on Baby Blue as shown for your machine. For total memory in excess of the figure given, refer to Table 2-2 ("Baby Blue DIP Switch Settings"), but avoid the reserved Pages listed for each machine.

	New Total Memory	Switch Setting 1 2 3 4 5 6 7 8		Baby Blue Memory Page/Segment
Columbia	: 64-128K	[][][][][][][] [] [] []	ON ↑	2
Compaq	: 64-128K	[][][][][][][] [] [] []	ON ↑	2
Corona	: 64-576K	[][][][][] [][][][] [] []	ON ↑	8
Eagle PC	: 64-576K	[][][][][] [][][][] [] []	ON ↑	8

RESERVED PAGES (Hex):

Eagle PC: 1, 2, 3, 4, 5, 6, 7, A, B.

Columbia: 1, B.

Compaq: 1, A, B.

Corona: 1, 2, 3, 4, 5, 6, 7, B.

2.4 STEP BY STEP HARDWARE INSTALLATION

2.41 IBM PC - ALL MODELS

2.411 Begin

Turn the system OFF. Disconnect 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

For a standard installation, be sure you have read all of Section 2.1, and check Baby Blue's switches against Table 2-1. For a custom installation, read Section 2.2 and refer to the customization notes for your machine in Section 2.3.

2.414 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's "edge connector" - that double row of thirty-one gold contacts projecting from the board's lower edge. You may choose any unoccupied socket, but avoid the leftmost one if you can.

Directly in line with the expansion socket you've chosen, 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.

2.415 Install Baby Blue

Bolted to Baby Blue is a metal mounting bracket designed to replace the slot cover. Grasping the board's top corners, lower it 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 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.

2.416 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 Guide To Operations. 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 the Baby Blue master diskette and put the master away. Now put the copy in drive A: and type:

A:TESTZ80 <CR>

This initiates the Baby Blue diagnostic tests - if no errors are shown, your Baby Blue is properly installed and working. If TESTZ80 shows an error, or if your system behaves abnormally, turn to the Appendix under DIAGNOSTICS.

2.417 Finishing Up

Everything works? Good - turn the power back off, and close up 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.

NOTES:

3. OPERATION: RUNNING CP/M PROGRAMS

3.1 GETTING STARTED

3.11 DOS COMMANDS

In normal operation Baby Blue is meant to be completely "transparent", which means that it fits seamlessly into your present operating system. When you run a CP/M 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.

3.12 THE BABY BLUE UTILITIES

Although Baby Blue 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 Conversion Software". First, make a backup copy of the Baby Blue 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 driveB.

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
TESTZ80.COM

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-80 programs to run on Baby Blue under PC-DOS.
- Transfer files between selected CP/M formats and PC-DOS diskettes.
- Provide user-programmable function keys for CP/M programs.
- Test and diagnose Baby Blue itself in the event of suspected failure.

You have already used TESTZ80 to verify the hardware installation. Here is a brief description of the other Baby Blue utilities - for detailed information see the individual sections in the Appendix.

HEADER

HEADER is a large program, practically an operating system in its own right. The "meat" of the Baby Blue 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, 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.

CONVERT

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.

BIND

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 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 80 separate characters. KEYFIX saves the definitions on disk, in the HEADER attached to each program.

SAMPLE

SAMPLE is a short CP/M-80 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 Baby Blue.

3.13 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'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. You can put the Microlog utilities away, and there are no restrictions to PC-DOS. You use the Microlog utilities 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 Wordstar's ".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 the Baby Blue 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 (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, do not attempt a "warm boot" using CTRL-ALT-DEL. This will fail to properly reset Baby Blue's Z-80 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 physically installed.

3.2 MEDIA COMPATIBILITY: ACCESS TO CP/M DISKETTES

3.21 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 successful, and in both hardware and software the owner of an 8088 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-80 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-80 applications programs.

Baby Blue 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".

3.22 MICROLOG FILE TRANSFER UTILITIES

3.221 5" CP/M Diskettes

Baby Blue comes with CONVERT, a utility which enables you to transfer files between PC-DOS disks and a number of popular CP/M-80 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-80 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.

3.222 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.

3.23 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. 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.

3.24 OTHER ALTERNATIVES

For one hundred dollars per source diskette (less in quantity), 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.

3.3 IMPORTING CP/M PROGRAMS: COMPATIBILITY

3.31 DEFINITION

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

1) Compatible with CP/M-80 version 2.2

   "Version 2.2" represents an update of the original CP/M-80 operating system, with enhanced capabilities. Generally, any program written for an earlier version (lower number) will be compatible. Do not confuse CP/M-80 with CP/M-86, which is an alternative to PC-DOS and doesn't need Baby Blue to run on your computer.

2) Installable to a Televideo 950 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-80 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-80 programs were written. We have chosen the very popular Televideo 950 terminal as our model - you will find that this choice assures compatibility with the widest range of available programs (programs will also work if they're in stallable to the ADM-3A, which is a subset of the Televideo 950).

In general, Baby Blue supports programs which have proven transportable between native CP/M-80 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.

3.32 TEXT AND DATA FILES

Since Baby Blue 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 8088 program or a CP/M program running on Baby Blue.

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.

3.33 OPERATING CONSIDERATIONS

Some CP/M programs will run on Baby Blue, 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'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. Note that HEADER is not part of this overhead - it runs in system memory, not in Baby Blue'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. The byte found at location 0003H in Baby Blue's memory 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 0.

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.

3.4 BABY BLUE AS A CP/M DEVELOPMENT SYSTEM

Baby Blue makes an excellent tool for CP/M program development. Most CP/M-80 compilers, interpreters, and development utilities (including SID and DDT) have been thoroughly tested on Baby Blue; 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 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, 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.

3.41 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 carefully that the rules apply only when under the control of HEADER, that is, when running a CP/M-80 program on Baby Blue - 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.

3.411 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 Micrology-supplied BSTAM, will also be bound if they are written to disk with the extension "COM".

3.412 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.

3.413 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.

3.414 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".

3.42 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 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 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's.

4. BABY BLUE REFERENCE MANUAL

4.1 INTRODUCTION

Baby Blue functions as an emulated CP/M environment, occupying a single 64K Page (Segment), within the host 8088 microprocessor's memory space. Memory 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 Baby Blue's segment 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's memory, so that the information there remains intact. The two processors alternate control of Baby Blue's memory according to the handshaking scheme described in "Hardware Functions".

This chapter explains how HEADER drives Baby Blue, and conversely, how a CP/M program running on the Baby Blue 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.

Hardware Functions:

Describes not HEADER, but the physical structure of the board, covering memory arbitration (handshaking), address decoding, and available control lines (port structure).

4.2 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 8088.

First, HEADER "polls" system memory to find out where Baby Blue 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-80 to set a "Found" flag within Baby Blue's memory. The HALT state is lifted, activating the Z-80. If a valid "Found" is returned, HEADER knows it has found Baby Blue. 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-80 is found, it enters a tight polling loop starting at location FE20H, and waits while HEADER constructs a simulated CP/M environment within PC-DOS. The first task is to install a Televideo 950 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.

Baby Blue's memory receives an abridged CP/M operating system and the CP/M program itself (See Table 4-1). The bottom 256 bytes hold the usual CP/M system-control parameters: for example, the expected jump table vectors are at 0000H and 0005H. The I/O Byte normally at 0003H is not implemented; instead, the high order nibble at this location holds the segment number at which HEADER's polls located Baby Blue. The top 500 or so bytes contain the Z-80 portion of the Operating System Translator, which mediates between a CP/M program's function calls and PC-DOS. The bulk of Baby Blue's memory, starting at location 100H, 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-80 environment, the boundaries of the memory map are also 64K wide, because that is the largest memory space which the Z-80 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, 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 8088. 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 FFF0H in Baby Blue's 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 8088 in control, this byte is filled with "1"'s (FFH), which means that the 8088 is in control. When a CP/M program is fully loaded, the 8088 sets the semaphore to a line of "0"'s (00H), then toggles the "Start" flag to set the Z-80 running.

At this point, the 8088 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-80 and inspect the contents of the semaphore. As long as the semaphore remains low (00H), Baby Blue runs at full speed, completely independent of the host system - except for the 8088'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. Since the Z-80 has no I/O channels at its disposal, it relies on the host system to carry out the transaction. The Z-80 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 8088. Handshaking resumes, with the Z-80 executing a polling loop of its own to periodically inspect the semaphore.

The 8088 inspects the Z-80 register table for the function call number and other parameters. The Operating System Translator translates the CP/M instructions issued by the program into their logical PC-DOS equivalents, after which it's business as usual under PC-DOS. Information is returned to the Z-80 register table and other relevant tables in Baby Blue's memory. Finally, the 8088 resets the semaphore to 00H and lapses into dormancy, polling for another I/O request.

When the Z-80 discovers that the semaphore has changed, it resumes program execution. At the end of execution, control returns to the 8088, but not immediately to the system. First HEADER does a house-cleaning which HALTs the Z-80 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 4-1: Memory Map

Hexadecimal		Decimal
FFFF		65535
FFF1	Begin Z80 Register Table	65521
FFF0	Semaphore	65520
	Z-80 Portion of Translator	
FF00	CP/M BIOS Jump Table	65280
FE06	CP/M BDOS Jump Table	65030
FDFF*		65023*
	Transient Program Area - Space for User Programs	
0100		256
	Page Zero	
0080	DMA Address	128
006C	Second Input Filename	108
005C	First Input Filename	92
0005	Jump Vector to BDOS Translation	5
0003	Not I/O Byte: Contains Baby Blue Segment Number	
0000	Jump Vector to BIOS Jump Table	0

* Subject to change

4.3 CONSOLE EMULATION

4.31 DESCRIPTION

Baby Blue's 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.

4.32 PURPOSE

The Emulator establishes portability of almost all CP/M programs to Baby Blue'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. Keyboard input is passed straight through without translation - control sequences entered at the keyboard will not alter video functions, but appear literally as the values typed.

4.33 VIDEO OUTPUT

4.331 Operation

Before loading a CP/M program onto Baby Blue, 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 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.

4.332 Video Control Codes

Table 4-2 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.

Table 4-2: Televideo 950 Video Control Codes

Control Sequences:

Mnemonic	ASCII Decimal	ASCII Hexadecimal	Comment
CTRL G	7	07H	Bell
CTRL H	8	08H	Backspace/cursor left
CTRL I	9	09H	Tab
CTRL J	10	0AH	Line feed
CTRL K	11	0BH	Cursor up
CTRL L	12	0CH	Cursor right
CTRL M	13	0DH	Carriage down
CTRL V	22	16H	Cursor down
CTRL Z	26	1AH	Clear screen
CTRL ^	30	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, 40	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	30H	No cursor
2	50	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 (20H) 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	30H, 40H	normal
1 or A	49, 65	31H, 41H	blank
2 or B	50, 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, 70	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	60, 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" will also step 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:
0	48	30H	foreground black
1	49	31H	foreground blue
2	50	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
<	60	3CH	foreground light red
=	61	3DH	foreground light magenta
>	62	3EH	foreground yellow
?	63	3FH	foreground high-intensity white
@	64	40H	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	70	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	80	50H	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	27, 102	1BH, 66H	Load user buffer I
text	(user entry)		(80 characters max)
CR	13	0DH	

ESC f text CR	27, 102 (user entry) 13	1BH, 66H 0DH	Load user buffer II (80 characters max)
ESC g	27, 103	1BH, 67H	Display user buffer I (on line 25)
ESC h	27, 104	1BH, 68H	Display user buffer II (on line 25 - in a real TV950, this buffer contains the status line)
ESC j	27, 106	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

4.34 KEYBOARD INPUT

4.341 Operation

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

- a greatly expanded TV950-style operator-definable function key set, yielding not 22 but 56 programmable function keys divided into two tables of 256 characters each (double the TV950 256-character table). The included KEYFIX utility offers complete flexibility to the unsophisticated end-user.

- the standard TV950 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 TV950 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 <CR> 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 <CR> for closure.

4.342 TV950 Function Key Programming

CP/M programs can use the following sequence to define twenty function keys (Normal and Shifted F1-F10) - the TV950'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 4-3: TV950 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 4-4
1	49	31H	Start of Message
(text)			This will be the programmed input for the designated key.
[CTRL P	16	10H]	Optional - precedes any non-print character
CTRL Y	25	19H	End of Message

Table 4-4: TV950 Function Key Codes

KEY	CHAR.	ASCII DEC.	ASCII HEX.	SHIFT KEY	CHAR.	ASCII DEC.	ASCII HEX.
F1	1	49	31	S-F1	<	60	3C
F2	2	50	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	40
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
F10	10	58	3A	S-F10	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).

4.343 Keyboard Defaults

HEADER is shipped with the function key definitions shown in Table 4-5. 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 4-5, 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 4-5: Function Key Default Definitions

Key	Unshifted	Shifted	Control	Alt
F1	^@; (00H,3BH)	S-FUN1	C-F1	A-F1
F2	^@< (00H,3CH)	S-FUN2	C-F2	A-F2
F3	^@= (00H,3DH)	S-FUN3	C-F3	A-F2
F4	^@> (00H,3EH)	S-FUN4	C-F4	A-F4
F5	^@? (00H,3FH)	S-FUN5	C-F5	A-F5
F6	^@@ (00H,40H)	S-FUN6	C-F6	A-F6
F7	^@A (00H,41H)	S-FUN7	C-F7	A-F7
F8	FUN8	S-FUN8	C-F8	A-F8
F9	FUN9	S-FUN9	C-F9	A-F9
F10	FUN10	S-FUN10	C-F10	A-F10
L ARROW	^@< (00H,04BH)		C-LF	
R ARROW	^@< (00H,04DH)		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 (00H,52H)			
DELETE	^@S (00H,53H)			

4.344 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 4-6 before the function keys will work on Baby Blue. Some of the keys shown are not available on the IBM PC - they are shown so that their definitions can be assigned to some other available key.

Table 4-6: 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 *

4.4 OPERATING SYSTEM TRANSLATOR

4.41 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.

4.42 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 0, because PC-DOS does not support more than one user).

4.43 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.

0: System Reset

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-80.

1: Console Input (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 "10: Read Console Buffer" for further comments about keyboard entry functions.

2: Console Output (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: Reader Input

A direct call to the PC-DOS Aux In.

4: Punch Output

A direct call to the PC-DOS Aux Out.

5: List Output

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

6: Direct Console I/O

Fully supported.

7/8: Get/Set I/O Byte

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 Baby Blue's location 0003H, where the CP/M I/O byte is normally found, the high-order nibble contains instead the segment number occupied by Baby Blue's 64K.

9: Print String

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

10: Read Console Buffer

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: Get Console Status (CONSTAT: "Interrogate Console Ready")

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

12: Return Version Number

Returns Version 2.2.

13: Reset Disk System

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: Write Protect Disk").

14: Select Disk

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: Write Protect Disk").

15: Open File

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: Close File

Direct translation to PC-DOS function call. AL returns either 00H (successful close) or FFH (file not found). When closing a COM file, this call also finds the size of the Z-80 code (less HEADER) and stores this number at location 0107H in the HEADER attached to the target file. The size of HEADER itself is stored at location 0105H.

17/18: Search for First/Next

Direct translation to PC-DOS function calls. Returns 00H (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 0 (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 01H, 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.

20: Read Sequential

Direct translation to PC-DOS function call - for a COM file, the first record returned will be the first line of Z-80 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: Return Log-in Vector

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

25: Return Current Disk

Direct translation to PC-DOS function call.

26: Set DMA Address

See BIOS Call FF24H: SETDMA, below.

27: Return Allocation Vector

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: Write Protect Disk

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

29: Return Read Only Vector

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

30: Set File Attributes

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: Get Address of Disk Parameters

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: Get/Set User Code

Returns value 0. PC-DOS does not support multiple users.

33/34: Read/Write Random

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

35: Compute File Size

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-80 code only. This will be accurate for operations conducted on Baby Blue 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: Set Random Record

Direct translation to PC-DOS function call.

37: Reset Drive

Ignored as irrelevant to PC-DOS.

38: Not used

39: Not used

40: Write Random With Zero Fill

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

4.44 CP/M BIOS CALLS

HEADER maintains a CP/M BIOS jump table starting at FF00H in Baby Blue's memory, with the standard pointer at 0000H. 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.

4.441 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 4096 bytes per track, with a limit of 1024 bytes per sector. It will find the specified sector on any disk conforming to these parameters; 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 0, Sector 1, which becomes PC-DOS Logical Sector 0 - 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 1024, 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 10, Sector 5] yields:

$$(26 * 10) + 5 / 32 = 265/32 = 8 + 8$$

Or [Track 8, Sector 8]. [Track 7, Sector 40] is also valid, since there is no range check, but not [Track 0, Sector 265], because the highest allowable sector number is 255.

4.442 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.

FF00H: COLD BOOT

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

FF03H: WARM BOOT

Invokes BDOS call 0, System Reset.

FF06H: CONST

Invokes BDOS call 11, Get Console Status.

FF09H: CONIN

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

FF0CH: CONOUT

Invokes BDOS call 2, Console Output.

FF0FH: LIST

Invokes BDOS Call 5, List Output.

FF12H: PUNCH

Invokes BDOS Call 4, Punch Output.

FF15H: READER

Invokes BDOS Call 3, Reader Input.

FF18H: HOME

Not supported.

FF1BH: SELDSK

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

FF1EH: SETTRK
FF21H: SETSEC

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: SETDMA

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: READ
FF2AH: WRITE

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

FF30H: SECTRAN

The physical sector always equals the logical sector plus one.

FF2DH: LISTST

Always returns "ready" (FFH in A).

4.5 EXTENDED BDOS FUNCTION CALLS

4.51 DESCRIPTION

Microlog has created a set of new CP/M-80 style function calls for use on Baby Blue. They are:

Number	Function
247	Chain
248	8088 Software Interrupt
249	System Memory Block Move Down
250	System Memory Block Move Up
251	Peek Host Memory Byte
252	Poke Host Memory Byte
253	8088 BIOS Call (Subset of # 248)
254	Output to Host I/O Port
255	Input from Host I/O Port

4.52 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-80 program can gain access to the host system at the following levels:

-8088 software interrupt

-Host memory (block moves and individual locations)

-Direct I/O through host ports

4.53 OPERATION

All extended function calls parallel standard CP/M-80 usage.

4.531 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.

4.532 Call 248: 8088 Software Interrupt

Entry Parameters:

Register C:	F8H
Register HL:	Address of pseudo 8088 Interrupt/Register Table

Return:

Executes specified interrupt
Updates 8088 Register Table at address specified by [HL]

Emulates an 8088 "INT" instruction. The HL register pair points in Baby Blue's memory to the starting address of a table representing the 8088 registers, as follows:

8088 Interrupt/Register Table

	Interrupt	Registers								
	Number	AX	BX	CX	DX	BP	SI	DI	DS	ES	Flags
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 8088 registers are returned to the table.

4.533 Call 249: System Memory Block Move Down

Entry Parameters:

Register C: F9
Registers HL: Block Move Table Address

Return:

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

Upon entry, the HL register pair points to a 10 byte table in Baby Blue's memory, organized as follows:

	Source Offset	Source Segment	Destination Offset	Destination Segment	Block Size
Byte #:	00	02	04	06	08

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 Baby Blue's memory.

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 can be anywhere in system memory, including Baby Blue.

This function parallels the Z-80 LDDR block move instruction, or the 8088 REPZ MOVSB with the STD instruction, i.e., it moves the block by starting with the lowest byte and incrementing. You can move data to or from any area of system memory, including on or off Baby Blue.

4.534 Call 250: System Memory Block Move Up

Entry Parameters:

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

Return:

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

Identical in all respects to Call 249, except that it emulates the Z-80 LDIR instruction, or the 8088 REPZ MOVSB with CLD, i.e., it moves the block starting with the last location and decrementing.

4.535 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-80 program to read from any location in the 8088's address space, including Baby Blue's memory.

4.536 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-80 program to write to any location in the 8088's address space.

4.537 Call 253: 8088 BIOS Call

Entry parameters:

```
Register   C:  FDH
Registers HL:  8088 Interrupt/Register Table Address
```

Return:

```
Executes specified interrupt.
Updates 8088 Interrupt/Register Table at specified address.
```

This function is included for compatibility with earlier versions of HEADER, and is a subset of Call 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.

4.538 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
```

Since the Baby Blue has no ports of its own, all I/O must pass through the 8088. This function enables a Z-80 program to output values directly to a port (under the control, of course, of the 8088) - use this function instead of an OUT instruction.

4.539 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-80 to input values directly from an 8088-controlled port.

4.6 HARDWARE FUNCTIONS

4.61 Z-80 PORT ADDRESS DECODING

The assignment of address lines to the Z-80's I/O port is given in Table 4-8 ("Blue DIP Switch 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 0300H to 031CH, depending on the base address of Baby Blue's memory (Table 4-9).

In a single Baby Blue 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 Baby Blue - once the memory segment is located, the port address is automatically known.

Note that here, OFF = 1 = High, and ON = 0 = Low. Numbers in "{}" brackets are set 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. A0 is tied Low ("0"), so all addresses are given as even, even though the low order bit is really a "don't care".

4.62 Z-80 CONTROL LINES

Z-80 control lines available to the 8088 programmer are:

NMI (Non-Maskable Interrupt: Jump to Location 66H)

In HEADER this interrupt is serviced by a routine which emulates a Z-80 system reset.

INT (Interrupt)

HALT

A special, discretely configured control line which presents a hard-wired HALT instruction (76H) to the Z-80 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-80 is in a HALT state, it executes No-ops, which are essentially bare memory refresh cycles. A HALTed Z-80 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-80 service routine at 66H to emulate any desired RESET functions.

Control lines are accessed through the Z-80's I/O port address of as follows:

[O] 03XXH [Control Data Byte]

Where:

[O] = OUT Instruction

03XXH = Port address in hexadecimal, where XX are determined by DIP switch settings on the Baby Blue (See Table 4-9: "Baby Blue DIP Switch Decoding").

[Control Data Byte] = information transmitted to the port to select the available control lines. Only bits 0, 2 and 3 of this byte are significant: the rest are "don't cares" ("x"). It maps onto the control lines as follows:

Table 4-7: Z-80 Functions Control Byte

	Data Lines				
Functions	NMI D3	INT D2	X D1	HALT D0	Decimal
HALT Z-80:	0	0	x	0	0
RUN (all off):	0	0	x	1	1
INTERRUPT:	0	1	x	1	5
JUMP to Loc. 66H:	1	0	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 D0 so that NMI will execute, but NMI must in turn be be cancelled by a "RUN" instruction (0 on D3) for the service routine to begin - otherwise the Z-80 will continuously execute an NMI.

Table 4-8: Address Decoding

PORT NUMBER (HEX)	PORT ADDRESS LINES	DIP SWITCH	SETTING (BINARY VALUE)	SEGMENT ADDRESS LINES	MEMORY SEGMENT (PAGE)
	A11	-	X	-	-
	A10	-	X	-	-
3	A9	HIGH	(1)	-	-
	A8	HIGH	(1)	-	-
	A7	SW1	ON {0}	-	-
	A6	SW2	ON {0}	-	-
X	A5	SW3	ON {0}	-	-
(0 OR 1)	A4	SW4	?	A19	0
	A3	SW5	?	A18	THRU
	A2	SW6	?	A17	E
X	A1	SW7	?	A16	
(2 THRU C)	A0	LOW	XX	-	

4.63 MEMORY ARBITRATION

Memory access is as straightforward as writing or reading a location within Baby Blue's memory segment. Since the Z-80 handles refresh, handshaking is constant, but it is automatically controlled by the board's nardware.

The 8088 has priority access to Baby Blue's onboard memory. A validly decoded address, combined with an active /MEMR or /MEMW, presents an active /BUSREQ to the Z-80. The Z-80 must respond by relinquishing the bus, but first completes its current machine cycle. The 8088 waits, responding to an active signal on its I/O CHANNEL READY line. An active /BUSACK indicates that the Z-80 has relinquished the bus, and lifts I/O CHANNEL READY, permitting the 8088 to complete its cycle. Now /BUSREQ goes high, starting the Z-80 and insuring that each memory access by the 8088 is followed by at least one Z-80 cycle, to maintain refresh.

Table 4-9: Segment and Port Assignments

Switch Setting 1 2 3 4 5 6 7 8	Memory Segment	Address Range	Z-80 Port Address
[][][][][][][][] ON ↑	1	10000-1FFFF	302
[][][][][][][][] ON ↑	2	20000-2FFFF	304
[][][][][][][][] ON ↑	3	30000-3FFFF	306
[][][][][][][][] ON ↑	4	40000-4FFFF	308
[][][][][][][][] ON ↑	5	50000-5FFFF	30A
[][][][][][][][] ON ↑	6	60000-6FFFF	30C
[][][][][][][][] ON ↑	7	70000-7FFFF	30E
[][][][][][][][] ON ↑	8	80000-8FFFF	310
[][][][][][][][] ON ↑	9	90000-9FFFF	312
[][][][][][][][] ON ↑	A*	A0000-AFFFF	314
[][][][][][][][] ON ↑	B*	B0000-BFFFF	316
[][][][][][][][] ON ↑	C*	C0000-CFFFF	318
[][][][][][][][] ON ↑	D*	D0000-DFFFF	31A
[][][][][][][][] ON ↑	E*	E0000-EFFFF	31C
[][][][][][][][] ON ↑	F*	F0000-FFFFF	31E

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

A. THE BABY BLUE UTILITIES

A.1 BIND: THE CP/M-80 PROGRAM IN PC-DOS FORMAT

BIND attaches HEADER to CP/M-80 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 8088 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 s:filename

Response:

CP/M IBM File Transfer Utility
Version 2.0 (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-8001
2. IMS 5000
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 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 80 characters long. Because the function key definitions reside in HEADER, KEYFIX can only be used with CP/M programs running on Baby Blue.

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. Defineable keys may not be used. They will be ignored. Control characters are okay. The maximum length of one entry is 80. 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-80 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:

```
10 Function Keys: <F1>-<F10>.
 4 Arrow Keys: <Up>, <Down>, <Left>, <Right>.
 6 Others:<Home>, <End>, <Pg Up>, <Pg Dn>,
          <Delete>, <Insert>.
20 Total
```

To select an unshifted key, type:

```
<Function key>
```

For example,

```
<F1>
```

displays:

```
KEY SELECTED:              FUNCTION 1
```

The "control" register consists of:

```
10 Function Keys <F1> - <F10>
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 10 keys:

10 Function Keys <F1> - <F10>
10 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 80 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

10 ALT Functions:	F1 - F10
10 CTRL Functions:	F1 - F10
6 Cursor Controls:	<UP>,<DOWN>,<RIGHT>,<LEFT>, <CTRL RIGHT>,<CTRL LEFT>
10 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 to be short strings, and it is quite natural to save elaborate instructions for the Normal keys F1 - F10.

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:

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.

A.4 DIAGNOSTICS: TESTZ8Ø

TESTZ8Ø is a diagnostic program which tests all hardware functions on the Baby Blue board, including memory. Use it any time you suspect a physical malfunction on the board. It is included so that you can distinguish Baby Blue related problems from faults in other parts of your system, including possible problems with software.

PROCEDURE

Type:

TESTZ8Ø <CR>

System Response:

```
Z8Ø CO-PROCESSOR CONFIDENCE TEST VERSION 1.Ø2
COPYRIGHT(C), 1983, MICROLOG, INC.

1  BABY BLUE LOCATOR PASSED
2  INTERRUPT TEST 66 PASSED
3  8Ø88 MEMORY TEST PASSED
4  8Ø88 ADDRESS LINES PASSED
5  INTERRUPT TEST 38 PASSED
6  Z8Ø ADDRESS LINES 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 your board fails TESTZ8Ø, or you suspect a hardware fault in your system, continue reading through "Troubleshooting", below.

TROUBLESHOOTING

You're here because your system failed to behave normally after you installed Baby Blue, or because TESTZ8Ø 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.

- TESTZ80 returns various errors.

All of these appear to be "hard" errors, indicating defective hardware. Since Baby Blue is the only new factor, it is natural to assume that the board is defective. However, as explained in Section 2.2, 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, be sure that you turn power OFF, and disconnect the power cable from your System Unit.

First, remove Baby Blue, 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 Baby Blue's only mechanical component, and they sometimes get "tired" - exercising them is often a quick fix. Make sure they're really set - push hard.

The next step is to isolate Baby Blue - you don't know the board is defective unless you have removed 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 wouldn't remove your video interface 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 testing 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, using the factory switch settings shown in Section 2.11. 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 - 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 some help.

If your board works now, you can start reinstalling your various options until you find the one that doesn't like Baby Blue. When you find the problem, Section 2.2 may give you some idea of what's causing it. The way to fix a conflict with another board is to change the Page assignment of one or both boards - this will remove possible overlaps in the memory map.

Changing Baby Blue's memory Page also changes its "port address", which is a separate addressing scheme used by your system to locate peripheral devices such as printers and disk drives. This means that you can also resolve conflicts involving non-memory boards, simply by changing Baby Blue's Page assignment.

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 10901
914-368-0353

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.

NOTES:

B. APPLICATIONS NOTES

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

Although DDT and similar utilities work on Baby Blue, 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 (100H 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 100H).
```

```
-D3 <CR>                    ; displays contents of Baby Blue
                              location 0003H in first position.
                              Ordinarily, this would be the CP/M I/O
                              Byte, which is not implemented in
                              HEADER. Instead, the high-order nibble
                              contains the segment number occupied
                              by Baby Blue in the host's memory
                              (e.g, a Baby Blue with base address
                              E0000H will  display "E0" with this
                              command). 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 Baby
                              Blue's segment, followed by colon,
                              followed by the starting address to
                              save  within  Baby  Blue's  memory
                              (usually the beginning of the program,
                              at 100H).

-Q <CR>                     ; exit DEBUG
```

C. 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 (90) 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 (90) 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 Microlog. 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, if 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 returned for repair or testing and found to be out of warranty will be assessed a minimum $55.00 service charge. If the charge for repair is to exceed $55.00, 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