The M68000 Educational Computer Board

A look at Motorola's $495, 68000-based single-board computer

by Robert W. Floyd

BYTE magazine, October 1983

If you're interested in getting acquainted with Motorola's 68000 16-bit microprocessor but can't part with $5000 or more, Motorola offers a usable system for only $495. For that price, don't expect a disk drive and a Unix operating system, but do expect a 68000-based single-board computer with 32K bytes of RAM (random-access read/write memory) and what may be the best monitor program in ROM (read-only memory) ever developed.

At $495 per unit, Motorola is not going to get rich by selling its Educational Computer Board (ECB). Obviously the strategy is to educate the coming generation of engineers and programmers about the 68000, with the expectation that they will design products that use the 68000. In addition to its intended audience of educators and students, this board is of interest to both hobbyists and people involved in developing 68000-based products. It is not a development system, but its interpretative assembler and disassembler make it handy to quickly test short routines.

The 68000

The 68000 is becoming an increasingly important microprocessor in today's market, as evidenced by the frequent announcements of products using the 68000. The chip is expandable by design, and Motorola has announced that a full 32-bit version will be available in 1984. Before jumping on the bandwagon, however, you should know that the 68000 is not just a bigger version of the old 8-bit microcomputer; it is considerably more complex than many of the 16-bit minicomputers popular since the 1970s.

Before reviewing the system, I'll briefly highlight some of the major differences between the 68000 and the old 8-bit devices we're familiar with. The machine language supports byte, word (16-bit), and long-word (32-bit) operations. It has eight data registers and eight address registers, each 32 bits wide. Except for one register, A7, which is always the current stack pointer, all address and data registers are treated identically.

The processor operates in two states: user and supervisor. Certain instructions are legal only in the supervisor states. Once in the user state, the processor will stay there until an *exception* occurs. Exceptions may be caused by resets, interrupts, bus errors, a variety of runtime errors, and executions of TRAP. Besides some of the interrupt conditions, all exceptions are auto-vectored; when the exception occurs, the processor gets the address of the exception-handling program from a specific location in memory. These exception vectors take up the first 1K bytes of memory and should all be initialized immediately after a reset occurs if you don't want your processor going off in strange directions.

The large number of 16-bit op codes makes a greater assortment of instructions possible. In the 68000, these possibilities appear as both new operations and a variety of addressing modes for the standard

operations. Most of these operations fall into the general scheme of supporting higher-level languages. These include instructions such as a branch subroutine (BSR), which allows relative branches to any address within 32K words above the program counter's address; the LINK and UNLINK instructions, which ease the construction of reentrant subroutines; and the move-multiple-registers instruction (MOVEM) for quickly saving registers. In addition, 12 addressing modes contain many instructions. These different modes are available for data transfer and logical and arithmetical operations. They can also be used in some of the transfer-of-control operations. In other words, a jump to subroutine (JSR) doesn't have to go to an absolute address--it might go to the address calculated by adding a data register to an address register plus an immediate value included in the instruction (by the way, that's one I haven't used yet). The result of this addressing arrangement is that it takes skill to write a program shorter than 64K bytes that is *not* relocatable, and it is very easy to write reentrant subroutines.

If Motorola simply added eight more address and eight more data lines to an 8-bit processor, only 56 pins would be needed. You well might ask what other functions have been added to require 64 pins. First, the 68000 is asynchronous in operation; it will not terminate any bus cycle until it recognizes that data transfer is complete, which is normally done by having the memory or peripheral device assert a data transfer acknowledge (DTACK). Because not all devices run asynchronously, the 68000 has a provision for synchronous devices, particularly those designed for the old 6800 microprocessor. In this case, a line called valid peripheral address (VPA) is used instead of DTACK.

When the 68000 detects this signal, it replies with a valid memory address (VMA) signal and synchronizes the bus operation to the Motorola 6800 E cycle. To provide more flexibility in the interrupt structure, three interrupt control lines produce seven levels of interrupts. Three bus arbitration lines allow DMA (direct memory access), refresh, and so on. One last note--RESET is bidirectional, so the 68000 can execute an instruction that resets all peripheral devices but does not affect operation of the processor.

Hardware

Let's take a look at what you get for $495. The ECB is a single-board computer measuring 7-1/2 by 10-1/2 inches. The quality of construction is excellent. The board's edge contacts as well as its traces are gold plated. The ECB provides a 68000 operating at 4 MHz, a 16K-byte ROM monitor, 32K bytes of dynamic RAM, two RS-232C serial interface ports, and the MC68230 parallel interface/timer (PIT) that provides a cassette tape recorder interface, a Centronics-type printer interface, a 24-bit timer, plus some uncommitted I/O (input/output) pins for a user to play with (see photo 1).

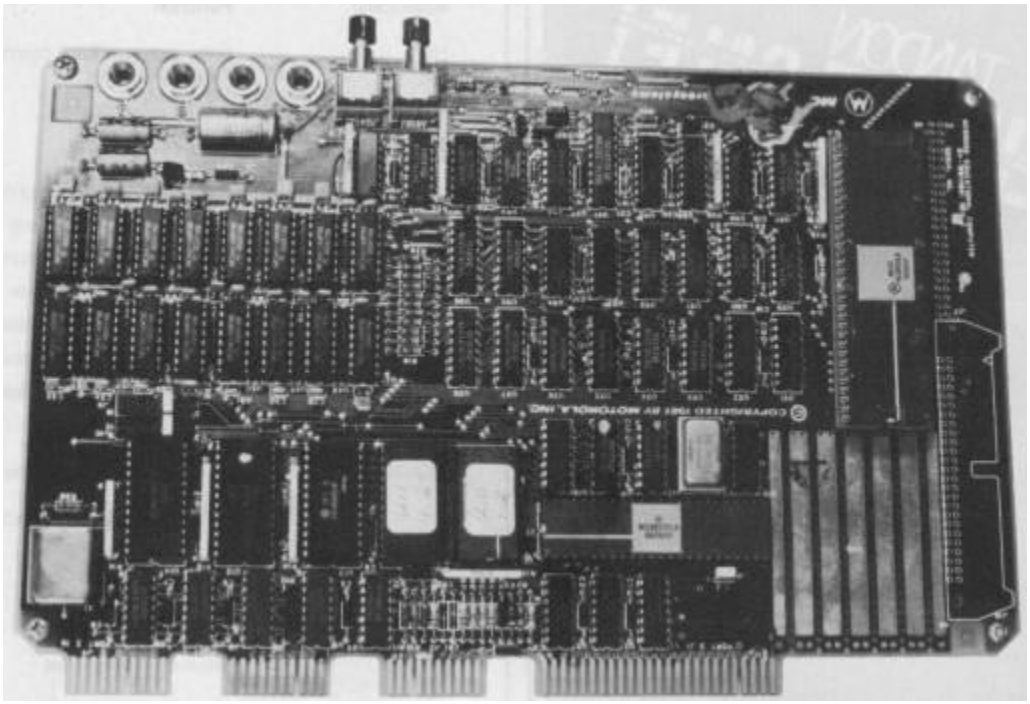

Photo 1: The very large integrated circuit (IC) in the upper right is the 68000. The other large IC near the wire-wrap area is the 48-pin MC68230 parallel interface/timer. The two serial interface ICs are at lower left. The edge connectors on the bottom are, from left to right, serial port 2 (host), serial port 1 (terminal), port 4 (cassette recorder), and port 3 (Centronics-type parallel printer).

The MC68000L4 microprocessor in this kit is the slowest version Motorola makes of this chip, running at 4 MHz (compared to 12-1/2 MHz for the fastest version). Remember: this board was designed for educators, not benchmark freaks. Yet even at this leisurely pace its minimum instruction time is only 1 microsecond. Because any given instruction on a 16-bit chip generally does a lot more than a single instruction on an 8-bit microprocessor, the throughput of a slow 16-bit chip is several times greater than that of the fastest 8-bit machine.

The 32K bytes of RAM are located in the lowest memory space and consist of sixteen 16K by 1-bit dynamic RAM chips. The monitor program, which resides in two 8K by 8-bit ROMs, is located in address space 8000 to BFFF (hexadecimal), making it awkward to expand the RAM because any additional memory will be noncontiguous with the original RAM. (Because it should be no more difficult to have the monitor reside at higher memory, why not do so and give hobbyists more flexibility?)

Communication to the ECB is through one of its two RS-232C serial ports. The data rate is user-selected from 110 to 9600 bits per second. Although any RS-232C terminal can be used, a video terminal is definitely preferable. For example, if you are doing a program trace, displaying eighteen 32-bit registers can burn a lot of paper and time. Also, the assembler works by first disassembling an instruction, letting

the programmer change it, then overwriting the original instruction with the new. This makes for a pretty messy display if you're using a teletype.

In addition to the transmit and receive lines (pins 2 and 3), serial port 1 must supply the DTR (data-terminal ready) signal. Pin 1 is not connected on the board, so make sure ground is tied to pin 7 on your DB-25 connector. The other serial port, the designated host port, allows the ECB to act like a terminal to a host computer, which enables programs to be downloaded from (or uploaded to) a host computer. The serial ports on the ECB come out to 20-pin edge connectors instead of to a DB-25 connector. If you don't want to make or modify your own cables, Motorola sells cables that mate from the edge connector to a standard DB-25.

The operation of the MC68230 PIT, a remarkable device in itself, requires a 32-page manual. The chip provides three 8-bit parallel ports with handshake and a 24-bit timer. Separate interrupt vectors for both the parallel ports and the timer may be stored in the PIT This chip supports a Centronics-type parallel printer port (which drives my Epson MX-80) and provides a cassette-recorder interface.

In addition to the peripherals, you need a power supply for the ECB. The board requires only 5 volts at 750 milliamperes and +12 volts at 100 milliamperes. All the components are tied together on the board on a simple, unbuffered bus. Although there is no edge-of-board connector for system expansion, most of the 68000's pins are brought out to a wire-wrap area, and a provision exists for inserting a 50-pin connector for expansion by the user. Because these lines are unbuffered, three-state buffers should be installed if more than a couple of extra chips are added to the bus.

Software

The ECB contains a monitor in ROM called Tutor. In addition to the usual monitor functions, it provides a disassembler and a line-by-line assembler. Compared to what else is on the market, the monitor alone is worth the price of the board. Table 1 lists the monitor commands provided in Tutor. Most of them resemble those of run-of-the-mill monitors, but a few require some explanation. Note the variety of addressing commands for Tutor. Most other monitor commands work only in the immediate mode (the user enters the explicit physical address), but Tutor has eight separate addressing modes. The most important of these are absolute, absolute plus the contents of an offset register, address register indirect, address register plus displacement indirect, and memory indirect.

Mnemonic	Function
HE	help; displays Tutor commands
MD	memory display
MM, M	modify memory

MS	store into memory
.A0-.A7	display/set address registers
.D0-.D7	display/set data registers
.PC	display/set program counter
.SR	display/set status register
.SS	display/set supervisor stack pointer
.US	display/set user stack pointer
DF	display formatted registers
OF	display offset registers
.R0-.R6	display/set offset registers
DC	convert decimal to hexadecimal
BF	block of memory fill
BM	block of memory move
BS	block of memory search
BT	block of memory test
BR	set a breakpoint
NOBR	remove breakpoint
GO, G	execute user program
GT	execute until breakpoint
GD	execute without setting breakpoints
TR, T	trace
TT	temporary breakpoint trace
DU	dump memory to a port
LO	load memory from a port
VE	verify memory load/dump

PA	printer attach
NOPA	reset printer attach
PF	port format
TM	transparent mode
*	send message

Table 1: The commands available in the Tutor monitor.

One of the more sophisticated monitor commands is .Rn, which enables you to display and modify any of the eight special registers. Remember that the 68000 programs are intrinsically relocatable. These registers help users write position-independent code by being automatically added to addresses specified in Tutor commands. For example, it's generally easier to write assembly-language programs starting at an even address, such as 1000 hexadecimal. If you are following the book *68000 Assembly Language Programming* by Gerry Kane, Doug Hawkins, and Lance Leventhal (Berkeley, CA: Osborne/ McGraw-Hill, 1981), you will notice that all the programs in the book start at 4000 hexadecimal. However, if more than one program is being used, the others have to reside somewhere else in memory.

Assigning the starting addresses of the routines to offset registers makes testing the routines a lot easier. Let's try an example. Assume you have a routine that starts at 4F7E hexadecimal and you'd like to test a piece of code hexadecimal 1A6 bytes from the origin. If adept at this sort of thing, you could perform hexadecimal arithmetic, come up with 5124, then type GO 5124. An easier way of doing this is to set R1=4F7E and then type GO R1+1A6 (or whatever displacement you need). The same technique is used for looking at the *n*th item of a buffer and checking the value in a peripheral chip.

The block search command (BS) lets you search through any block of memory looking for either ASCII (American National Standard Code for Information Interchange) strings or binary data. This is useful when looking for particular memory references, such as I/O locations. The block fill command (BF) is useful for zeroing memory or buffers or for setting ASCII buffers to all space characters. The GO command starts program execution at a specified location, but a return from subroutine (RTS) will not return to the monitor. To do so, you have to use a special TRAP instruction.

In transparent mode (TM), the ECB acts as a dumb terminal to a host computer. Everything received on port 1 is immediately transmitted on port 2. I was able to use the TM command to good advantage when installing the ECB in my Heath H-19 video terminal. Internally, I wired the output of the H-19 to the terminal port of the ECB, and the ECB's host port to the input of the H-19. When first powered on, the H-19 talks to the ECB; but once TM is executed, it behaves like a normal H-19.

DU (dump) and LO (load) commands let you dump automatically formatted binary files either to or from memory. Normally this command is used with port 4 for a tape recorder or with port 2 to a host computer. The VE (verify) command reloads a program saved using the DU command and verifies that it matches what is is in memory. The contents of memory can be either dumped or loaded using the DU and LO commands, respectively. Memory is dumped in what Motorola designates "S" records, which consist of header and data dumps in hexadecimal ASCII. You can use the DU command with all supported I/O channels (terminal, host, printer, and cassette tape). The LO command supports input from all but the printer. Because data is transferred in hexadecimal ASCII instead of straight binary, 2 bytes need to be saved for every single byte of actual data, which can be annoying when you have to wait two minutes to load a 4K-byte file.

Probably the two most important parts of the monitor are the assembler and disassembler. These features are of much greater necessity than their counterparts on an 8-bit computer. Many times I've hand-assembled 100 or so lines of 6502 code, but I wouldn't even think of hand-assembling more than one line of 68000 code. Each instruction must be coded on a bit-by-bit basis, and the results must be converted to hexadecimal.

The assembler is a line-by-line interpretative routine, invoked by typing the MM (memory modify) command, the starting address, and the DI option. The assembler first disassembles the code at the current location, then prompts the programmer to enter the new instruction to be assembled. The new instruction is then written over the original. This process requires a video terminal because a printing terminal would type the new instruction over the old, resulting in illegible copy. Because it is a line-by-line assembler, labels can't be used in the operand field. Previously defined offsets, however, can be used. For forward references (address unknown), the * can be used. This generates the code for a jump to the location of the operation just assembled. When the entire routine has been written, it can be disassembled and the correct memory location inserted at all instructions that branch to themselves.

The disassembler may be invoked in the same manner as the assembler: it disassembles one line at a time. To get a complete listing, the MD (memory display) command can be used with the DI option (see photo 2). The listing can be directed to all four I/O ports. However, the software does not support reading anything other than "S" files from the tape recorder, and I have not yet found a way of retrieving disassembled listings from tape.

```
 ------------------------------------------------------

|                                      |

| 004030    FFFF              DC.W    $FFFF ?.     |

|                                      |

| TUTOR  1.1 > MD 4000 2F;DI                        |
```

```
| 004000   307C6000        MOVE.W  #24576,A0   |
| 004004   4280            CLR.L   D0          |
| 004006   1018            MOVE.B  (A0)+,D0    |
| 004008   6724            BEQ.S   $00402E     |
| 00400A   43E80001        LEA.L   1(A0),A1    |
| 00400E   08810000        BCLR    #0,D1       |
| 004012   5340            SUBQ.W  #1,D0       |
| 004014   600E            BRA.S   $004024     |
| 004016   B308            CMPM.B  (A0)+,(A1)+ |
| 004018   630A            BLS.S   $004024     |
| 00401A   1420            MOVE.B  -(A0),D2    |
| 00401C   10E1            MOVE.B  -(A1),(A0)+ |
| 00401E   12C2            MOVE.B  D2,(A1)+    |
| 004020   08C10000        BSET    #0,D1       |
| 004024   51C8FFF0        DBF.L   D0,$004016  |
| 004028   08010000        BTST    #0,D1       |
| 00402C   66D2            BNE.S   $004000     |
|                                              |
| TUTOR  1.1 > MM 402E;DI                      |
| 00402E   4E73            RTE     ? RTS_      |
|                                              |
```

Photo 2: The assembly and disassembly functions, invoked by typing DI after a memory display or memory modify command. Assembly is done line by line--the code at the current memory is first disassembled and followed by a question mark. The programmer then types in the correct code, and the original code is overwritten with the new. Any value not recognized as a valid op code is disassembled as the declare word (DC.W) pseudo op.

Motorola does not supply source listings of the Tutor program, which could be a pain for anyone writing I/O routines, but it does support a large number of user-available routines that can be accessed through the TRAP #14 instruction. The TRAP function is the only graceful way to enter supervisor mode from user mode. The function desired is passed as a parameter in register D7. Currently, 28 different I/O and utility routines are available to the user through the TRAP, and 127 additional numbers are available for user-defined functions.

Tutor is the only software available for the ECB. As dedicated software hackers get the board, this situation will change. I plan to purchase the FORTH source code for the 68000, along with the installation manual, from the FORTH Interest Group (FIG).

Documentation

When you first open the shipping container, the board seems dwarfed by the nearly 500 pages of documentation that come with the system (see photo 3). The documentation consists of a 240-page programming manual for the 68000, a 130-page manual for the educational board, and data booklets on the three major LSI (large-scale integration) chips on the board--the MC68000 microprocessor, the MC68230 parallel interface adapter and timer, and the MC6850 asynchronous communications interface.

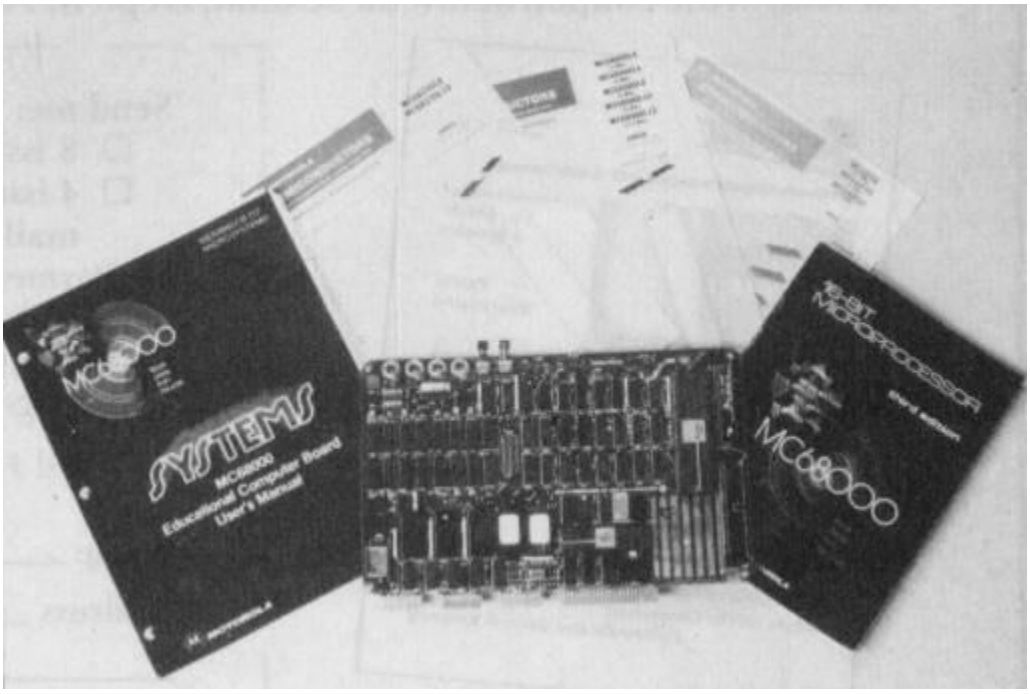

Photo 3: The documentation seems to dwarf the ECB. Clockwise from left are the Educational Computer Board Users' Manual, data books on the MC68230 timer, the 68000, the MC6850 serial interface adapter, and the MC68000 Programmer's Manual.

I've read a lot of computer documentation and have found, as a rule, that Motorola documentation is the most thorough and intelligible. The documentation for the ECB is no exception. Only the source listing for Tutor is lacking. The documentation is not simple because the computer and its software are complex, but so far I haven't come across any problems that couldn't be solved by reading the book. Overall, the material reflects a standard of excellence I would like to see other manufacturers emulate.

Homebrew Capabilities

Although the board is designed primarily for educational purposes, it has great potential as the basis of a homebrew project. In fact, that was my prime motivation in buying the board. Because it starts out with an excellent monitor, 32K bytes of RAM, and some basic I/O functions, I was already on the way to having a system with reasonable capabilities.

I wanted video capabilities, so I installed the board inside a Heathkit H-19 terminal (see photo 4). The ECB draws only a few watts, thus there is no problem with it using the H-19 power supply. The serial output of the H-19 board goes directly into the ECB board, and port 2 of the board hooks to the DB-25 connector on the back. In this way, the system acts like a normal H-19 terminal when the program is in transparent mode.

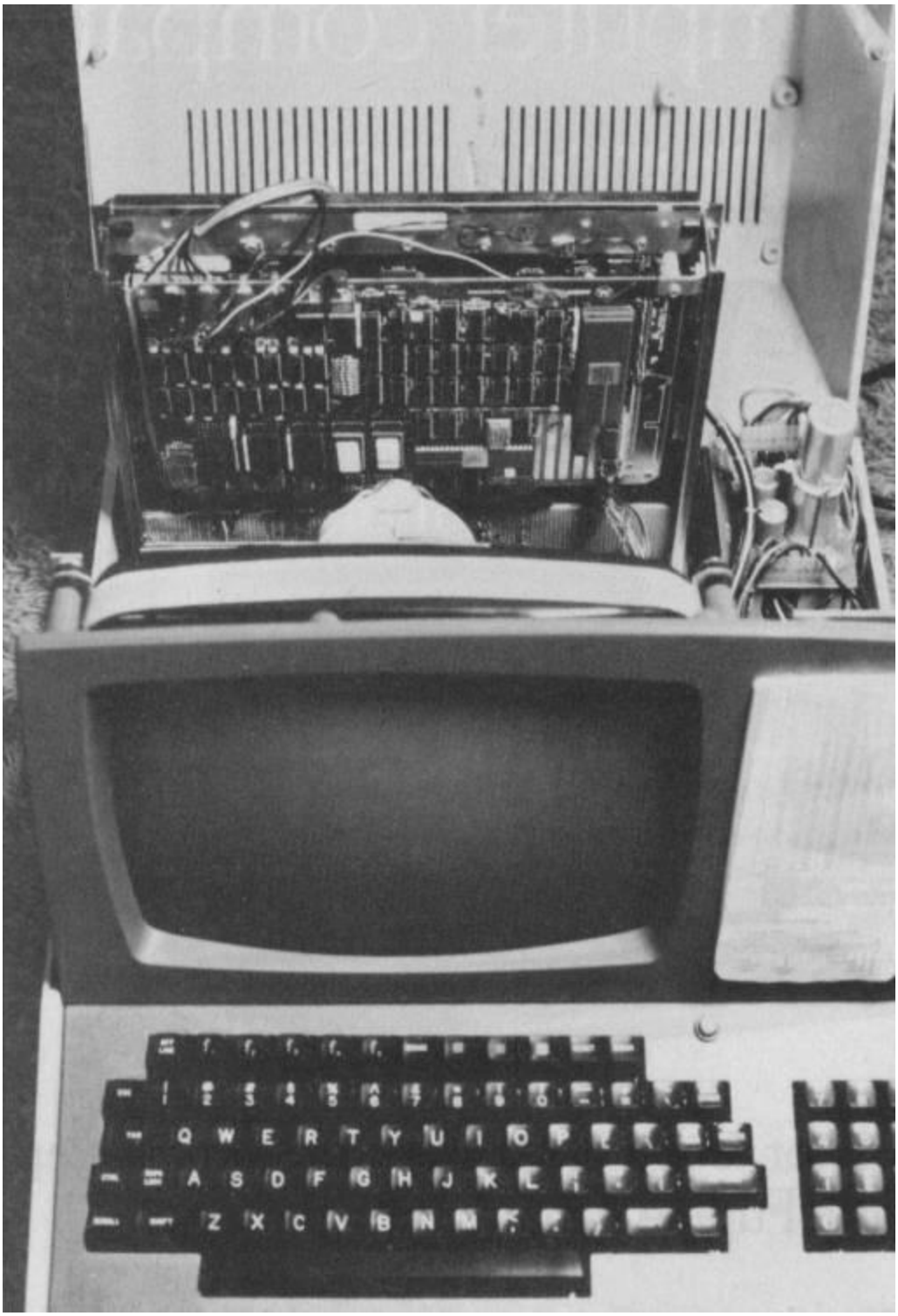

Photo 4: The ECB fits neatly into a Heathkit H-19 terminal. Regulated power is obtained from the H-19 through the wires in the upper left. The serial output of the terminal goes directly to the ECB, and the host port of the ECB is wired to the serial output port of the H-19. Separate holes punched in the rear chassis are for the cassette and disk I/O connectors.

Because I soon tired of loading programs from tape and had an old 8-inch disk drive I had picked up at a swap meet, I decided to construct a disk controller. The design I used is a modification of Nicholson and Camp's "Super Simple Floppy-Disk Interface" (see May 1981 BYTE, page 360). The interface turned out to be even simpler on the ECB because the 68000 is fast enough to perform some functions in software that had been implemented in hardware on the original design.

Conclusions

If you are interested in learning about the 68000, and particularly about 68000 assembly language, the ECB is hard to beat. Even if you already have a 68000 development system, the ECB is valuable for writing and testing short routines.

The monitor program alone, which includes an assembler and disassembler, is worth the price of the board. I expect a 16K-byte monitor to be very powerful, and this one is.

The Motorola documentation is excellent but lacks a source listing of the monitor program. However, most of the important I/O and conversion routines are available through the TRAP #14 function.

At a Glance

Name
M68000 Educational Computer Board (Motorola part # MEX68KECB)

Manufacturer
Motorola Inc.
Microsystems Division
3102 North 56th St.
Phoenix, AZ 85018
Distributed by Hamilton/Avnet Electronics

Hardware
Single-board computer. 6-1/2 by 10-1/2 inches, containing a 68000 microprocessor running at 4 MHz; 16K-byte monitor ROM; 32K-byte dynamic RAM; two serial I/O ports with data rate individually selectable from 110 to 9600 bps; MC68230 parallel port/timer that provides three 8-bit parallel ports with handshaking; audio cassette I/O; Centronics-type printer port; and wire-wrap area. Operation requires an external power supply (+5V at 750 mA, +12V at 100 mA). RS-232C-compatible video terminal, and connecting cable

Software
16K-byte monitor ROM with most of the functions of the Motorola Macsbug plus interpretative

assembler, disassembler, printer and tape recorder functions, and a series of TRAP #14 functions that allows user program access to most of the monitor data conversion and I/O routines

Options
Interface cables to connect computer board to terminals, printers

Price
$495

Warranty
90-day warranty includes parts, labor, and return shipping

Documentation
Over 450 pages, including data sheets on LSI circuits, a software manual for the 68000, and a manual on the ECB hardware and monitor

Audience
Educators, students, hard-core hobbyists. and 68000 systems developers who would like to use the ECB as a supplementary aid