BUILD THE SB180 SINGLE-BOARD COMPUTER

PART 1: THE HARDWARE

BY STEVE CIARCIA

This computer reasserts 8-bit computing in a 16-bit world

Newer, faster, better. These words are screamed at you in ads and reviews of virtually every new computer that comes to market. Unfortunately, many of the proponents of this rhetoric are going on hearsay evidence. While advertising hype has its place in our culture, a more thorough investigation may lead you to alternative conclusions.

Generally speaking, quotes of increased performance are basically comparisons of CPU (central processing unit) instruction times rarely involving the operating system. The 68000 is indeed a more capable processor than the 6502, but that doesn't necessarily mean that commercial application programs always run faster because the CPU has more capability. People owning 128K-byte Macintoshes have discovered this.

The bus size of the processor is only one factor in the performance of a computer system. Operating-system design and programming styles contribute much more to the overall throughput of a computer. It is not enough to simply compare 8 to 16 bits or 16 to 32 bits. For example, the Sieve of Eratosthenes prime-number benchmark runs faster in BASIC on the 8-bit 8052-based controller board presented in last month's Circuit Cellar than it does on a 16-bit IBM PC.

The ultimate performance of a computer is the sum of its subsystem interaction times and not just the CPU execution speed. Using a simple database-management program involves interaction among the user input device, operating system, disk drives, firmware, system memory, and user output devices. Slow communication or a bottleneck between any two of these subsystems can degrade the performance of the entire computer.

In my opinion, the processor/operating system connection contributes most to user satisfaction. In the days before the IBM PC, the de facto computer standard was the 8080/Z80 and CP/M 2.2. Unfortunately, software developers considered it difficult to use, especially in turnkey applications. The frustration of having to warm-boot the system merely to change disks and the lack of a PATH command created many ready-and-willing PC-DOS customers. If only there were an 8-bit operating system with the capabilities and friendliness of MS-DOS.

(continued)

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. He is the author of several books about electronics. You can write to him at POB 582, Glastonbury, CT 06033.

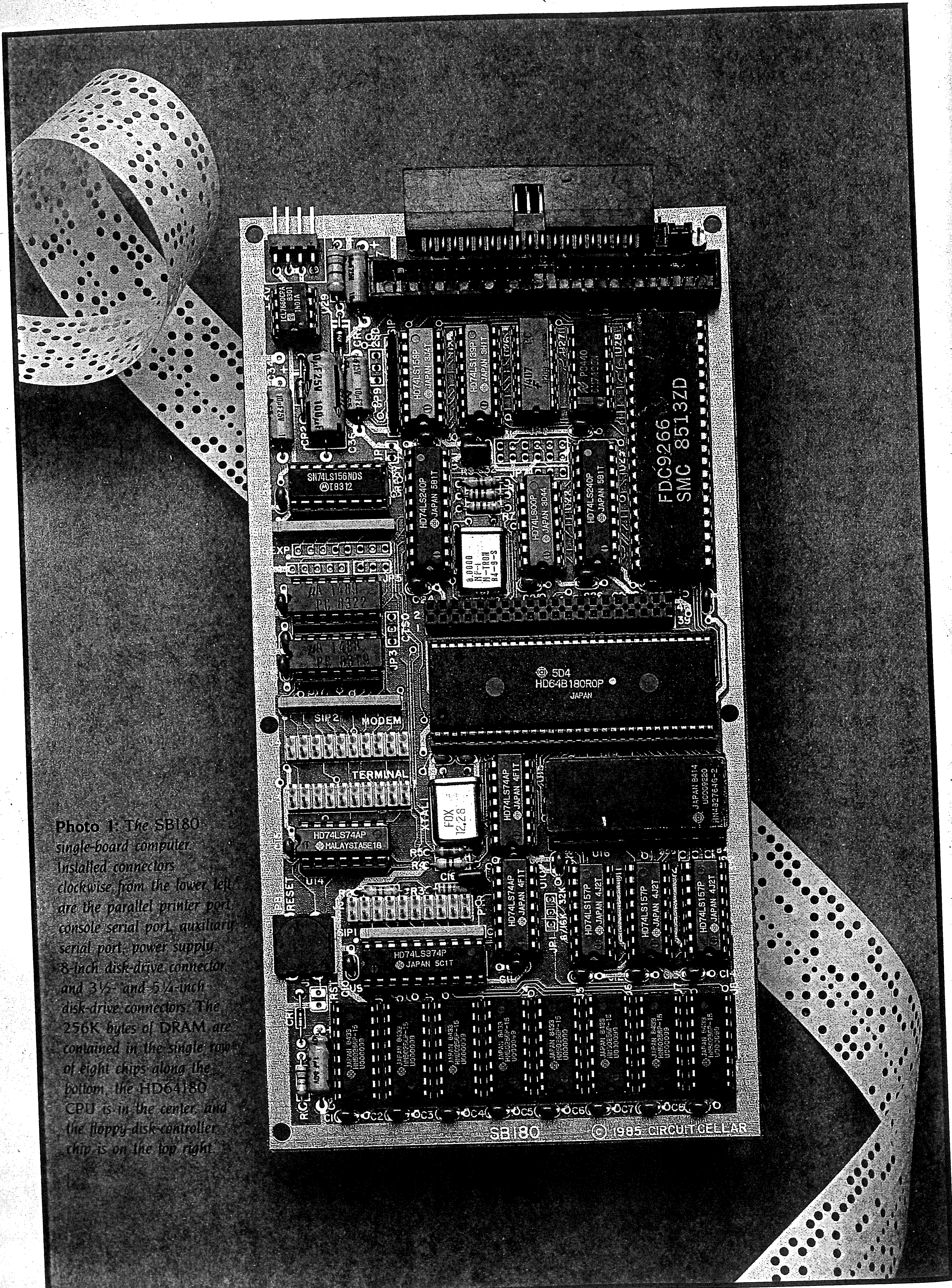


Photo 1: *The SB180 single-board computer. Installed connectors clockwise from the lower left are the parallel printer port, console serial port, auxiliary serial port, power supply, 8-inch disk-drive connector, and 3½- and 5¼-inch disk-drive connectors. The 256K bytes of DRAM are contained in the single row of eight chips along the bottom, the HD64180 CPU is in the center, and the floppy-disk-controller chip is on the top right.*

PHOTOGRAPHED BY PAUL AVIS.

The Z80 is still considered a high-performance CPU. In reality, the 8-bit processor is a cost-effective and efficient choice for personal computers and industrial controllers. (Remember that peripheral support chips such as PIAs [peripheral interface adapters] and floppy-disk controllers are 8-bit devices, as are many memory chips.) One problem is that the Z80 has an address limitation of 64K bytes, which discourages the use of 50K-byte BASIC interpreters and 100K-byte integrated spreadsheet programs common to IBM PC users. While additional memory has been added through hardware bank switching, it has never been properly integrated into the CP/M operating system, and its function is kludgy. If only someone would design a Z80 chip that directly addresses more than 64K bytes.

THE CIRCUIT CELLAR SB180

Hitachi has recently developed a CMOS (complementary metal-oxide semiconductor) Z80-code-compatible processor, designated the HD64180, that directly addresses 512K bytes. Echelon Systems has developed an operating system for the processor that is an amalgam of CP/M, MS-DOS, and UNIX. This operating system is called the Z-System. Using the HD64180 and the Z-System, I have produced a computer that reasserts 8-bit computing in a 16-bit world and outperforms a standard IBM PC by 20 to 100 percent.

In this two-part article, I take a look at the Hitachi HD64180 and the evolution of CP/M as embodied in the Z-System. The hardware project is a single-board computer (4 by 7½ inches) called the SB180, which is the 29-chip equivalent of a large S-100 system (see photos 1 and 2). Refer to figure 1 for a block diagram of the SB180 board.

The Circuit Cellar SB180 has the following capabilities:

- 256K-byte on-board RAM (random-access read/write memory), which can be partitioned as 64K-byte system memory and 192K-byte RAM disk or as paged system memory
- 32K-byte EPROM (erasable programmable read-only memory)
- full ROM-based monitor with system-disk boot
- two RS-232C serial ports, one with automatic data-transmission-rate detection
- a Centronics parallel printer port
- single/double-density programmable floppy-disk controller capable of handling a mix of up to four 3½-, 5¼-, or 8-inch drives; different-size drives can run concurrently
- requires only +5 volts (V) and +12 V for operation (+12 V only for RS-232C)
- I/O (input/output) expansion bus
- fits on a 4- by 7½-inch board that mounts directly to a 5¼- or 3½-inch floppy-disk drive

Disk-based software includes the Z-System DOS (disk operating system), a debugger, and HD64180 assembler.

While this is in fact a hardware project, true functionality would be an exercise left to you readers were it not for the operating system and BIOS (basic input/output system) specifically written and adapted to the SB180. The combination of the HD64180 and Z-System is what gives this tiny computer so much power. Much of the project description therefore has to be the software that uniquely sets performance levels far above traditional 8-bit computer designs. I don't want to diminish the significance of the hardware, but I am a realist.

This month, I'd like to describe the HD64180 chip and give an overview of the changing evolution of CP/M as it pertains to this project. After that, I'll describe the design of the SB180. Next month will be dedicated to the operating-system software and BIOS.

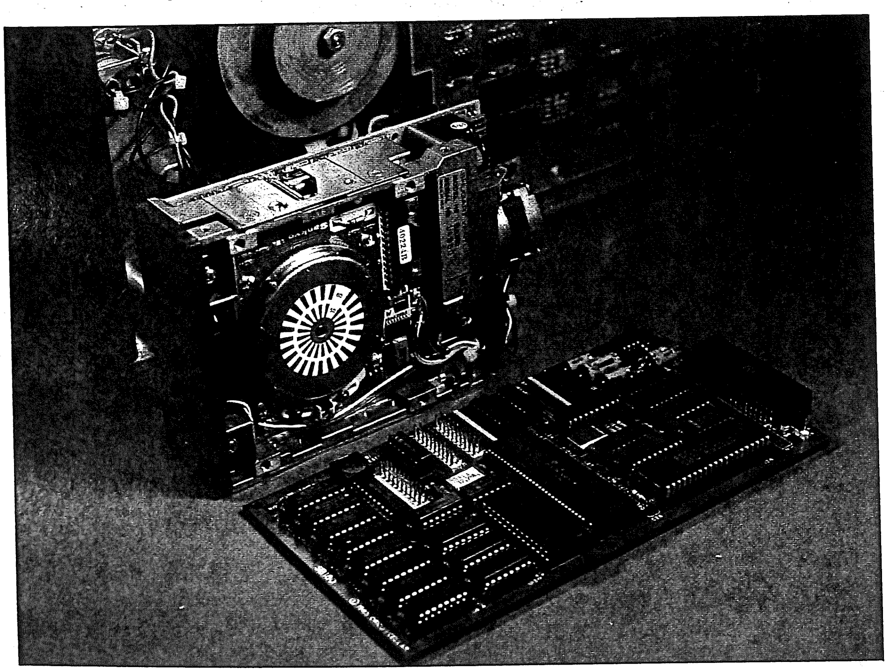


Photo 2: *The SB180 shown beside a 3½-inch disk drive and an 8-inch disk drive.*

CP/M AND BEYOND

Anyone who has been involved with microcomputers for more than two years acknowledges the tremendous impact of CP/M upon the history of personal computing. While hobbyists were still debating whether the "standard" tape format should be Kansas City CUTS (cassette user tape system) or Tarbell, Gary Kildall made CP/M available at a reasonable price. It quickly became the de facto standard DOS for 8080- and Z80-based microcomputers. With a "standard" operating system and a "standard" 8-inch disk format, entrepreneurial programmers saw the opportunity to write serious business applications that needed disk capability to be viable and that could be sold to thousands of CP/M machine owners.

The first major commercial release of CP/M was version 1.4. In an effort to fix some bugs and improve its file-handling capabilities and limits, it was upgraded first to version 2.0 and quickly to version 2.2. Version 2.2 has been a stable product that is familiar to most readers of BYTE. Version 3.0, or CP/M Plus, has been available for about two years, but it hasn't matched the popularity of version 2.2. While CP/M Plus does offer some advanced features, it is not significantly better than version 2.2.

CP/M has many quirks and shortcomings. Having to warm-boot the system when changing disks is a major inconvenience, and named directories would be more convenient than trying to remember disk designators and user-area numbers. Almost no file or system security is provided. CP/M 2.2 does not support redirection of I/O devices, as MS-DOS and UNIX do. Even the MS-DOS batch facility beats CP/M's more primitive Submit and XSUB two ways from Sunday; CP/M lacks conditional testing at the command level.

Many system integrators decry CP/M's lack of a good menu utility to integrate stand-alone, executable programs. Command-line editing is primitive, and multiple commands on one line would be appreciated; creating a single-name "alias" for multiple commands would be even better. Since CP/M has no search path (à la PC-DOS and UNIX), you must either keep multiple copies of often-used programs in many different user areas (wasting disk space) or patch CP/M with a software kludge to make user area 0 accessible from all user areas (even that doesn't help if you need to access a file from user area 3 while you are in user area 7). Many CP/M users supplement their system utilities with more useful public-domain programs like XDIR, DU, CRC, Help, Unerase, and Diff. It's too bad that CP/M wasn't upgraded more regularly so that it could have evolved into a more mature product.

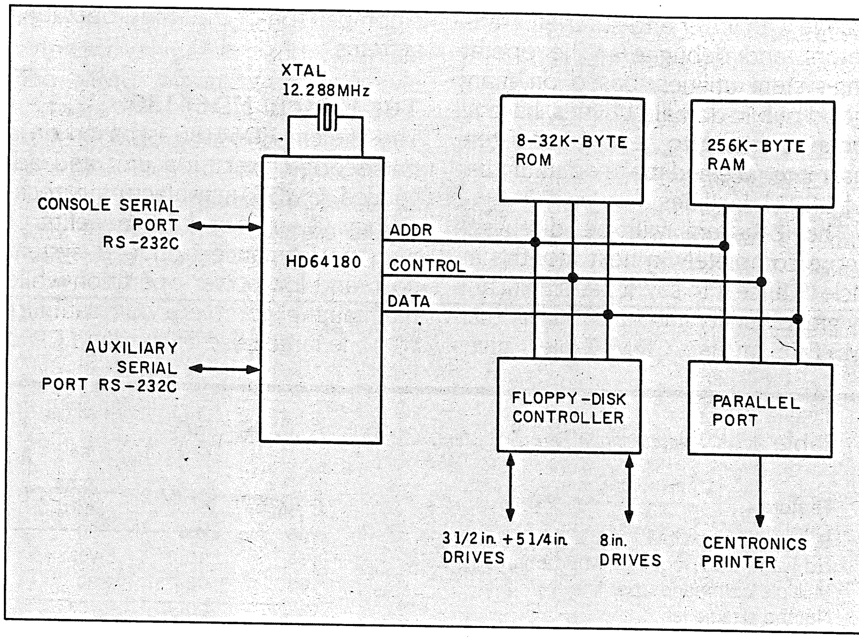


Figure 1: *Block diagram of the* SB180 *single-board computer.*

These observations are certainly not unique. When millions of people use a computer and its operating system in an attempt to become more productive in some way, obstacles to efficiency and productivity are bound to show up. Computer users who have had experience with many different operating systems will miss certain features to which they had become accustomed. And creative individuals of all types always can think of a "better way" to perform some function already provided.

One of the best-known attempts to improve CP/M was ZCPR (Z80 command processor replacement), developed under the direction of Rick Conn. ZCPR was written to replace the console command processor (CCP) supplied with CP/M. This public-domain software featured scaled-down features of UNIX appropriate for a single-user CP/M system. However, since it was designed to fit into the same 2K-byte space as the CCP, it was limited in what it could offer. It did make user area 0 "public" so that executable programs there could be invoked from other user areas. It also changed the prompt so that it indicated the user area as well as the drive currently logged in. As a result of user feedback, ZCPR2 evolved as an extended version of ZCPR. However, it required much more technical sophistication to install it into a CP/M system, and it started patching into and replacing parts of CP/M itself.

Within the past year, Echelon Inc. released ZCPR3, a much-improved version of ZCPR2, which provides solutions to all the problems with CP/M 2.2 and adds more features. Echelon has also developed a complete replacement for CP/M 2.2 in the form of the Z-System. It is composed of two major sections: ZCPR3 and ZRDOS. ZRDOS completely replaces CP/M's BDOS (basic disk operating system).

Written entirely in Z80 assembly language, the Z-System offers the benefits derived from the expanded Z80 instruction set (CP/M is 8080 code only) and from fixing bugs in CP/M 2.2 itself. It is downward-compatible with all CP/M software and takes advantage not only of the Z80 instruction set but of the Hitachi HD64180 CPU as well. Echelon provides both the operating system and a macro-relocating assembly lan-

(continued)

guage with linker and librarian, translators, and debuggers. The operating-system utilities, based on many good public-domain utilities, have all been rewritten to have a consistent user interface and make optimum use of ZCPR3 facilities.

The Z-System will be discussed more completely in next month's article. Suffice it to say, it has significantly greater utility and functionality than MS-DOS or plain CP/M. Table 1 gives a comparison of the three operating systems.

Table 1: A *comparison of the Z-System,* CP/M 2.2, *and* MS-DOS.

Feature	Z-System	CP/M 2.2	MS-DOS
Software-compatible with CP/M 2.2	yes	yes	no
No warm boot required when changing disks	yes	no	yes
Multiple commands per line	yes	no	no
Named directories	yes	no	yes
Password protection for directories	yes	no	no
Dynamically variable user privilege levels for commands	yes	no	no
Searching of alternate directories for invoked programs and files	yes	no	partial
Terminal-independent video capabilities	yes	no	no
Input/output redirection	yes	no	yes
Conditional testing and execution at the operating-system level (IF/ELSE/ENDIF)	yes	no	no
Shells and menu generators with shell variables	yes	no	no
Tree-structured on-line help and documentation subsystem	yes	no	no
512-megabyte file sizes, 8-gigabyte disks	yes	no	no
Complete error trapping with recovery, customizable messages, and prompts	yes	no	no
Screen-oriented file manipulation and automatic archiving and backup	yes	no	no
Full-screen command-line editing with previous-command recall and execution	yes	no	no

THE HITACHI HD64180

The Hitachi HD64180 is based on a microcoded execution unit and advanced CMOS manufacturing technology. It provides the benefits of high performance, reduced system cost, and low-power operation while maintaining complete compatibility with the large base of standard CP/M software. The HD64180 can be combined with CMOS VLSI (very-large-scale integration) memories and peripheral devices to form the basis for process-control applications requiring high performance, battery-power operation, and standard software compatibility.

Performance is derived from a high clock speed (6 MHz now, 9 MHz in the near future), instruction speedup, and an integrated memory-management unit (MMU) with 512K bytes of memory address space. The instruction set is a superset of the Z80 instruction set; 12 new instructions include hardware multiply, DMA (direct memory access), I/O, TEST, and a SLEEP instruction for low-power operation.

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

System costs are reduced because many key system functions have been included on this 64-pin chip. Table 2 compares the HD64180 with other 8-bit processors.

Table 2: A *comparison of some 8-bit processors.*

	HD64180	8080/85/Z80	NSC800	Z800	80188
Process	CMOS	NMOS	CMOS	NMOS	NMOS
Power	100 mW	1 W	100 mW	2 W	2 W
Maximum clock	10 MHz	8 MHz	4 MHz	10 MHz	8 MHz
Address space	512K bytes	64K bytes	64K bytes	512K bytes	1 megabyte
UARTs	2-ch.	no	no	1-ch.	no
DMAC	2-ch.	no	no	4-ch.	2-ch.
Timers	2-ch.	no	no	4-ch.	2-ch.
Clocked SI/O	yes	no	no	no	no
CS/wait logic	yes	no	no	yes	yes
DRAM refresh	yes	yes (Z80)	no	yes	no

Note: The availability of the Zilog Z800 is unknown, and specifications on it are subject to change.

The block diagram in figure 2 shows that the HD64180 CPU is composed of five functional blocks:

- Central processing unit: The CPU is microcoded to implement an upward-compatible superset of the Z80 instruction set. Besides the 12 new instructions, many instructions execute in fewer clock cycles than on a standard Z80.
- Clock generator: This generates the system clock from an external crystal or external clock input. The clock is programmably prescaled to generate timing for the on-chip I/O and system-support devices.
- Bus-state controller: This controller performs all status/control bus activity, including external bus-cycle wait-state timing, RESET, DRAM (dynamic RAM) refresh, and master DMA bus exchange. It generates "dual-bus" control signals for compatibility with

both 68xx and 80xx family devices.

- Interrupt controller: The interrupt controller monitors and gives priorities to the four external and eight internal interrupt sources. A variety of interrupt-response modes are programmable.
- Memory-management unit: The MMU maps the CPU's 64K-byte logical-memory address space into a 512K-byte physical-memory address space. The MMU organization preserves software object-code compatibility while providing extended memory access and uses an efficient common area/bank area scheme. I/O accesses (64K bytes of I/O address space) bypass the MMU.

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

- Direct-memory-access controller (DMAC): The two-channel DMAC provides high-speed memory-to-memory, memory-to-I/O, and memory-to-mem-

(continued)

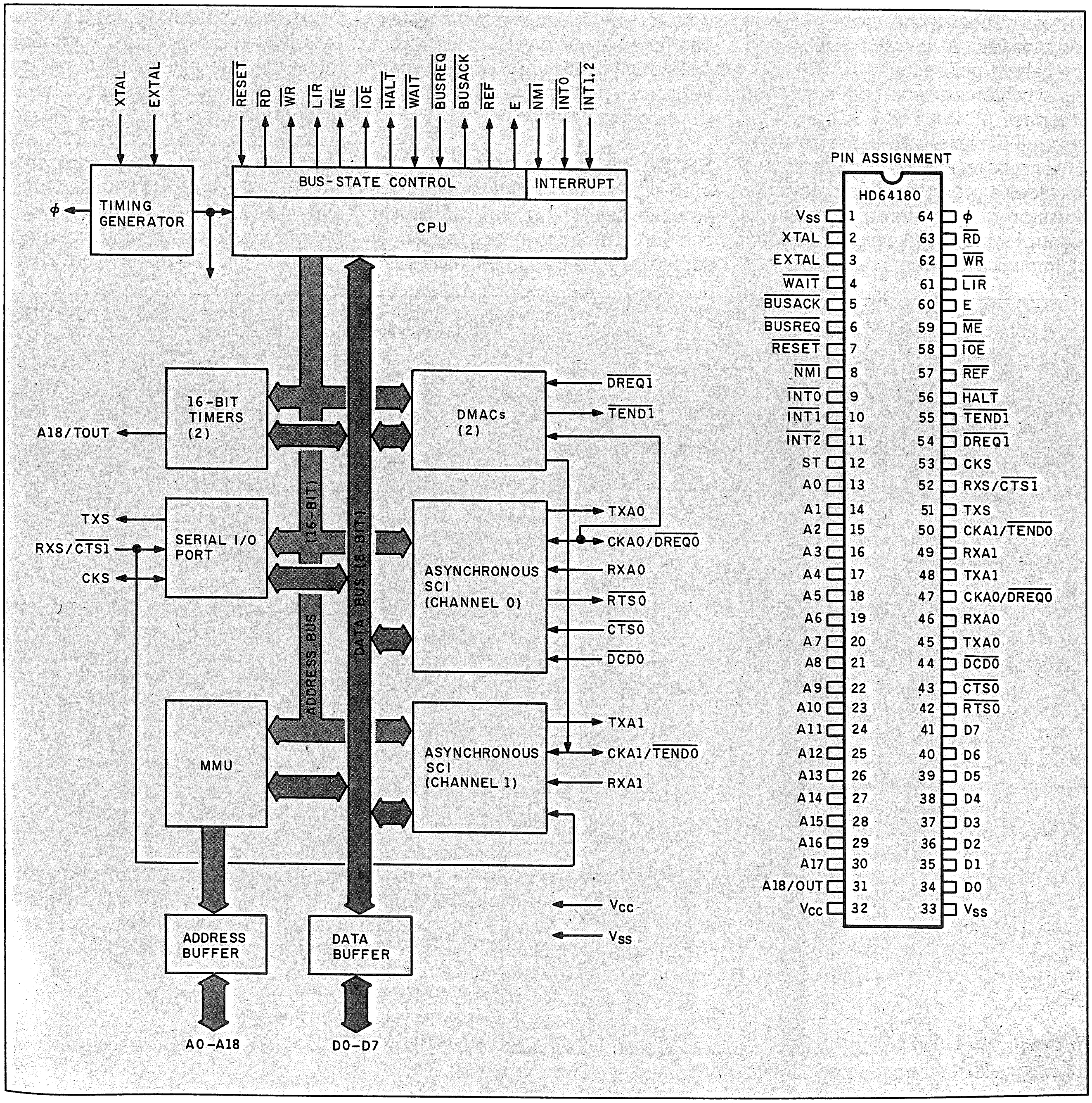


Figure 2: *Block diagram and pin-out of the* Hitachi HD64180.

ory-mapped-I/O transfer. The DMAC features edge or level sense-request input, address increment/decrement/no change, and (for memory-to-memory transfer) programmable-burst or cycle-steal transfer. In addition, the DMAC can directly access the full 512K bytes of physical-memory address space (the MMU is bypassed during DMA) and transfers (up to 64K bytes in length) can cross 64K-byte boundaries. At 6 MHz, DMA is 1 megabyte per second.

- Asynchronous serial communication interface (ASCI): The ASCI provides two full-duplex UARTs (universal asynchronous receiver/transmitters) and includes a programmable data-transmission-rate generator, modem-control signals, and a multiprocessor communication format. The ASCI can use the DMAC for high-speed serial data transfer, reducing CPU overhead.
- Clocked serial I/O port (CSI/O): The CSI/O provides a half-duplex clocked serial transmitter and receiver. This can be used for high-speed connection to another microcomputer.
- Programmable reload timer (PRT): The PRT contains two separate channels, each consisting of 16-bit timer data and 16-bit timer-reload registers. The time base is divided by 20 from the system clock, and one PRT channel has an optional output allowing waveform generation.

SB180 Design Criteria

With all this functionality on one chip, you can see why so few additional chips are needed to implement a truly sophisticated 8-bit single-board computer in such a small space (less than 30 square inches). In terms of the original Altair microcomputer of 10 years ago, the functionally equivalent machine would have taken about 35 S-100 boards for a total of 1750 square inches (and that's using 8K-byte memory boards!).

In order to reduce chip count further, I decided to use an enhanced floppy-disk-controller chip (FDC) from Standard Microsystems Corporation, the 9266 (see figure 3). This 40-pin DIP (dual in-line package) chip is software-compatible with the industry-standard NEC 765A FDC and is actually an integrated combination of SMC's 9229 digital data separator and an NEC 765A FDC. It is compatible with single- and double-sided 3½-, 5¼- (40- and 80-track), and 8-inch

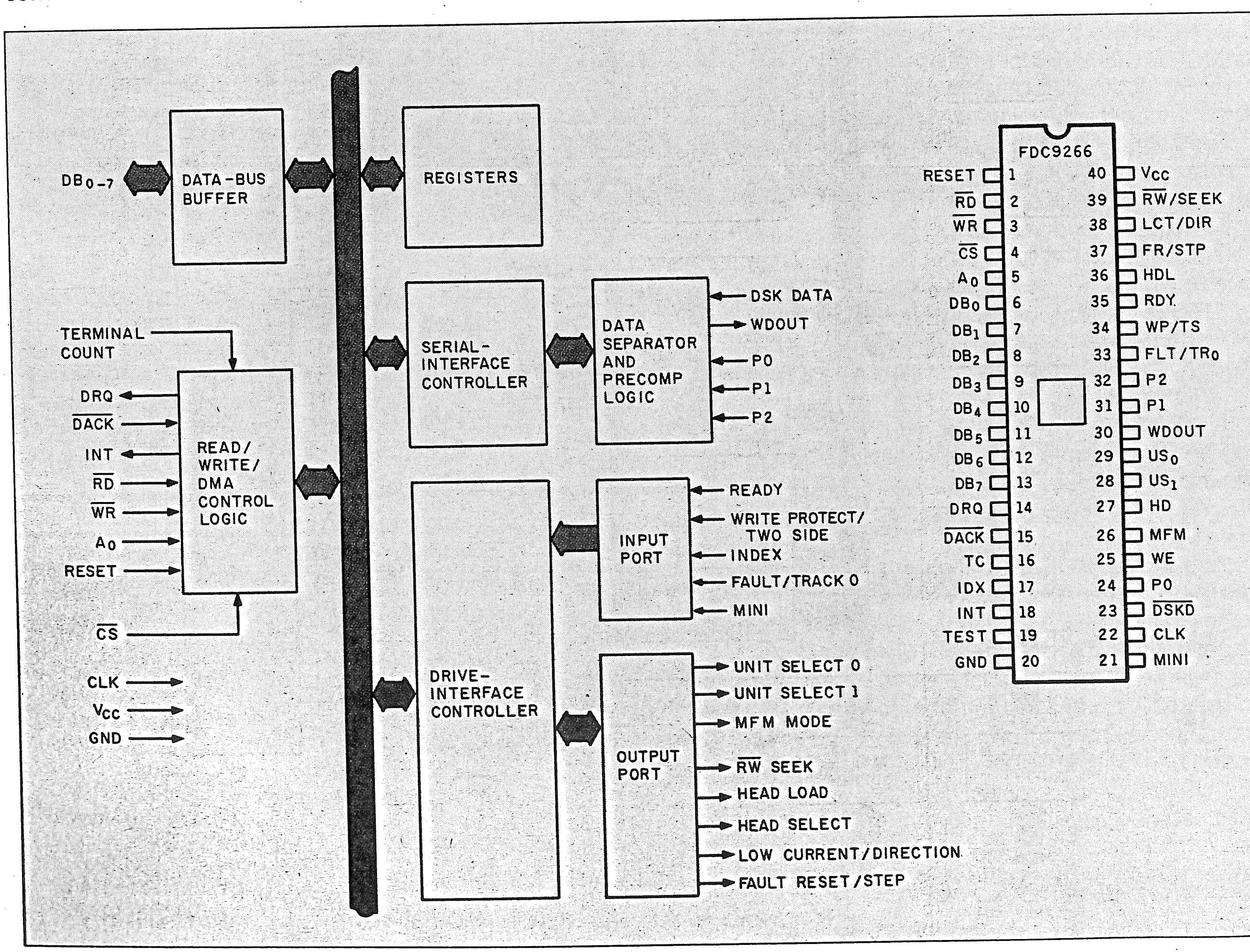


Figure 3: *Block diagram and pin-out of* SMC's 9266.

drives; the data separator handles both single- and double-density data. This means that it can be programmed to read and write most soft-sectored CP/M disk formats.

With the HD64180's two-channel ASCI built in, two serial ports were included in the design automatically. Provision was made for a Centronics parallel printer port as well. Since 256K-bit DRAM chips are now plentiful and inexpensive, eight were used for memory (64K-bit DRAMs can also be used). Because only 64K bytes is used for the logical-memory address space, you can optionally designate the other 192K bytes as a RAM disk in the operating system, or you may prefer to use if for other purposes (such as implementing banked memory for CP/M Plus).

THE SB180 HARDWARE

Figure 4 is the schematic diagram of the SB180 computer. Its design is primarily characterized by the high-performance, high-density MOS (metal-oxide semiconductor) devices, including 256K-bit DRAMs.

In addition to the CPU elements previously discussed, the SB180 system design implements the following functional blocks: RS-232C interface, memory interface, Centronics printer interface, floppy-disk interface, XBUS expansion bus, and power supply.

RS-232C INTERFACE

The HD64180 ASCI two-channel UART is connected to 1488/1489 RS-232C line drivers/receivers to provide two separate ports. ASCI channel 1 is used for the console; ASCI channel 0 is used for auxiliary RS-232C devices like printers, plotters, and modems. This distinction is made because modems require the extra handshake signals that are available with ASCI channel 0, while terminals do not. All primary RS-232C parameters (data-transmission rate, handshaking, data format, interrupts) are software-programmable.

MEMORY INTERFACE

The SB180 incorporates a 28-pin boot-ROM socket that can be jumpered to hold 8K by 8-bit, 16K by 8-bit, and 32K by 8-bit memory devices. The boot ROM (contains disk boot and ROM monitor) occupies the bottom 256K bytes of the HD64180 physical-memory address space since it is selected whenever A18/TOUT is low. (Note: The TOUT timer output function is not used.) Thus, the boot-ROM contents (whatever the size) are simply repeated in the lower 256K bytes. The boot-ROM output ($\overline{\text{OE}}$) is enabled by the HD64180 $\overline{\text{ME}}$ (memory enable) signal. (As configured, the maximum RAM on the SB180 is 256K bytes. To support larger memories, additional address decoding would be required to designated RAM and ROM areas in the current 256K-byte boot-ROM space.)

ROMs of 200 nanoseconds (and, marginally, 250 ns) can operate with one wait state.

At $\overline{\text{RESET}}$, the HD64180 begins execution at physical address 00000, the start of the boot ROM. [*Editor's note: All addresses are in hexadecimal.*]

256K-BIT DYNAMIC RAM

Standard 256K-bit 150-ns DRAMs, requiring 256 refresh cycles (8-bit refresh address) every 4 milliseconds (ms) are used. These DRAMs occupy the top 256K bytes of the HD64180 512K-byte physical-memory address space.

The interface is quite straightforward. Complete DRAM refresh control is provided by the HD64180 in conjunction with control logic IC14 and IC22 and address multiplexers IC11, IC12, and IC13.

The HD64180 $\overline{\text{WR}}$ output directly generates DRAM $\overline{\text{WE}}$. The HD64180 $\overline{\text{ME}}$ output directly generates $\overline{\text{RAS}}$. During normal read/write cycles (A18 high, $\overline{\text{REF}}$ high), $\overline{\text{CAS}}$ goes low at the next rising edge of ϕ following the rising edge of E (enable). This provides plenty of setup time for the address multiplexers since the rising edge of E switches the address multiplexers from row to column addresses.

$\overline{\text{RAS}}$-only refresh is used. The HD64180 generates the refresh addresses. During refresh cycles ($\overline{\text{REF}}$ low), $\overline{\text{ME}}$ generates $\overline{\text{RAS}}$, while $\overline{\text{CAS}}$ is suppressed at IC22.

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

CENTRONICS PRINTER INTERFACE

The Centronics printer interface is composed of the 8-bit latch IC9 and flip/flop IC10. The Centronics port is decoded at I/O address 0C0 by IC26. To write to the printer, the following sequence is used:

Write data to port 0C1

This sets up the data to the printer and asserts $\overline{\text{STB}}$ low. The following sequence:

Write data to port 0C0

deasserts the printer $\overline{\text{STB}}$ signal high.

When the printer has processed the data, it will return the $\overline{\text{ACK}}$ signal, which generates an external interrupt ($\overline{\text{INT 1}}$) to the HD64180. The interrupt handler clears the interrupt by performing a dummy output to port 0C0. This clears the $\overline{\text{INT 1}}$ interrupt request.

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

FLOPPY-DISK INTERFACE

SMC's 9266 FDC manages almost all details of the drive interface, including data separation and (with external logic IC22 and IC23) programmable

(continued)

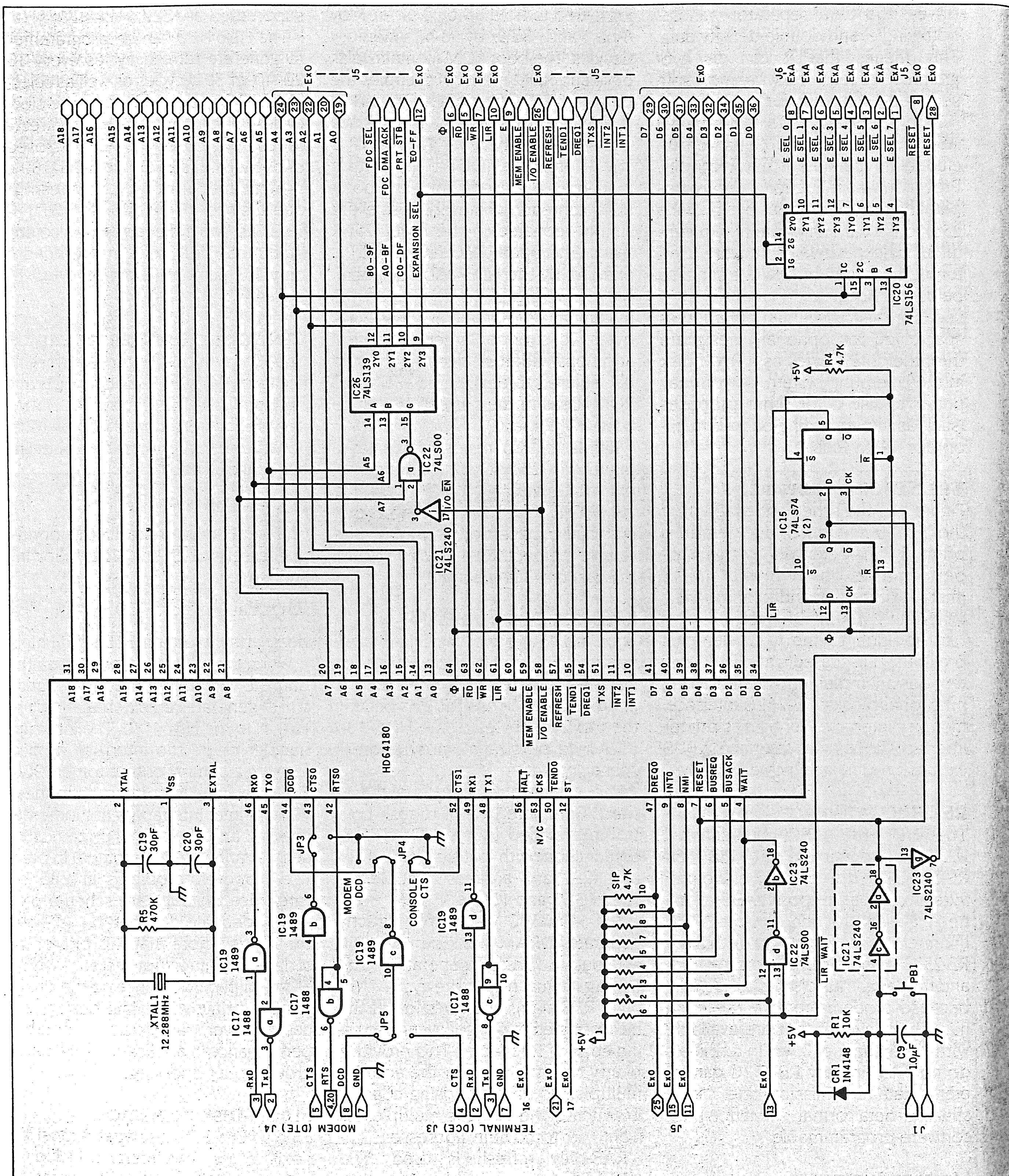


Figure 4: *Schematic diagram of the SB180 computer.*

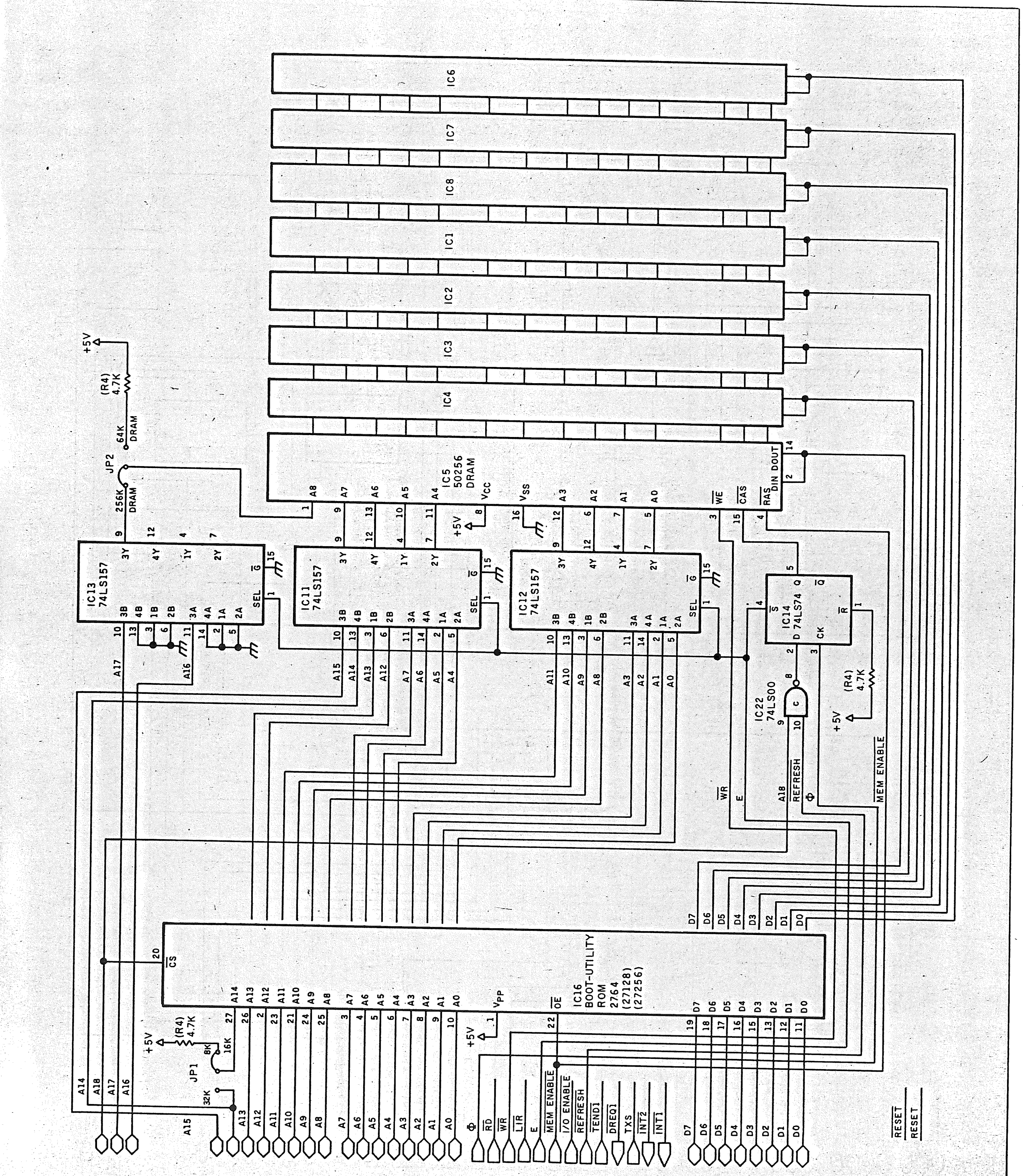
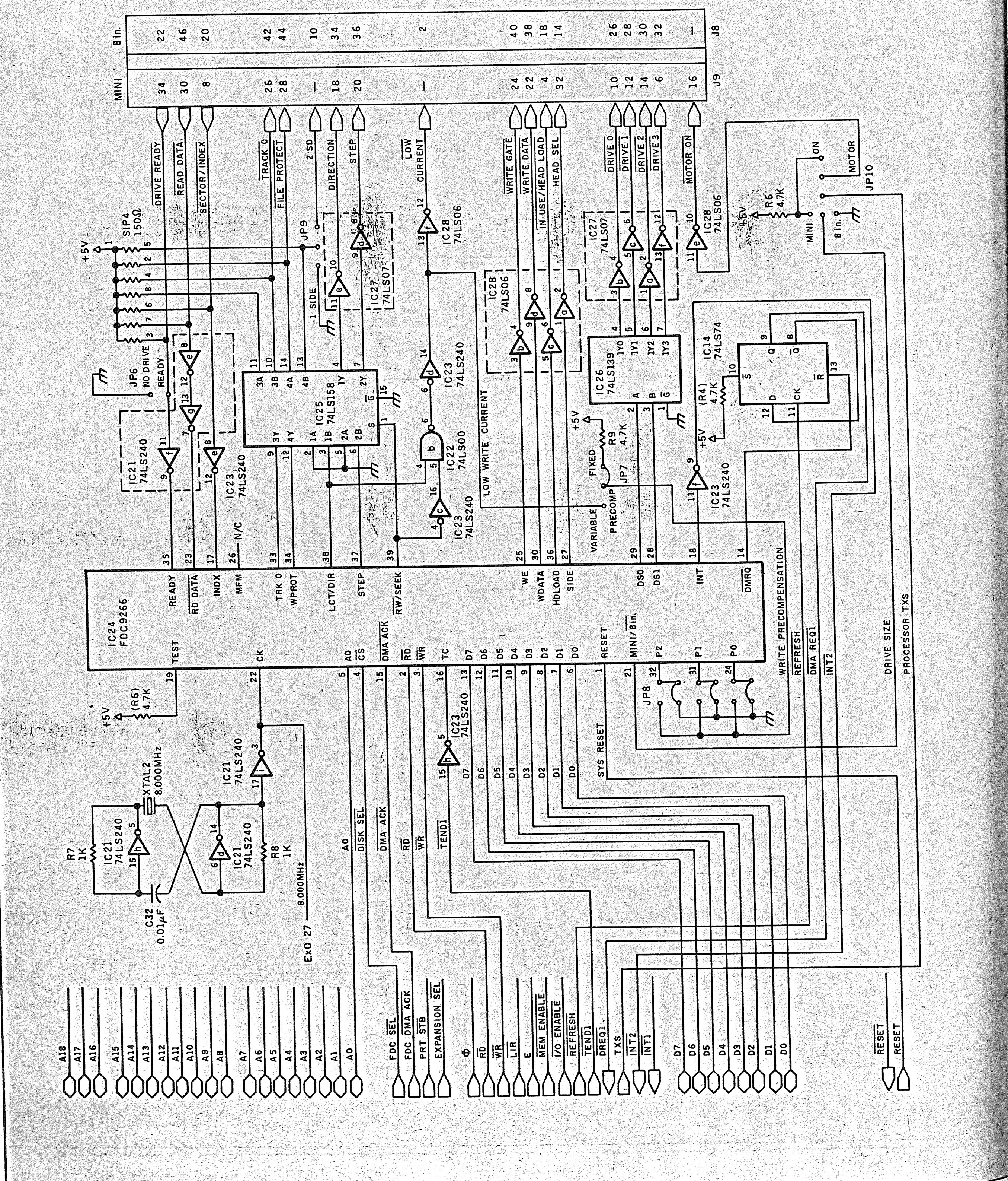
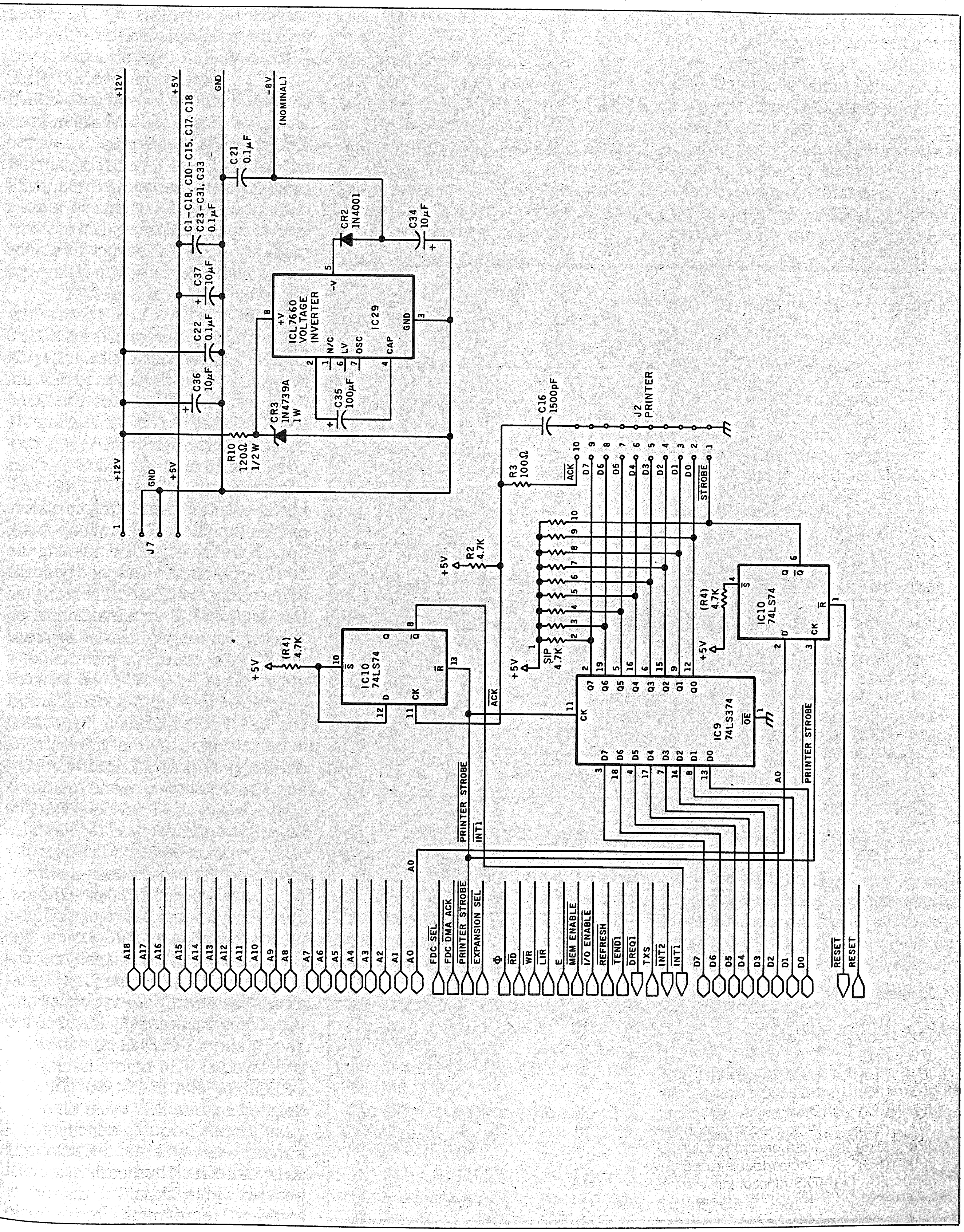


Figure 4 continued on page 96

Figure 4 continued





ICL7660 VOLTAGE INVERTER
IC29
IC10 74LS74
IC9 74LS374
J2 PRINTER
J7
+12V
+5V
GND
-8V (NOMINAL)
C1-C18, C10-C15, C17, C18
C23-C31, C33
0.1μF
C21 0.1μF
C34 10μF
C37 10μF
C22 0.1μF
C36 10μF
C35 100μF
C16 1500pF
CR2 1N4001
CR3 1N4739A 1W
R10 120Ω 1/2 W
R3 100Ω
R2 4.7K
(R4) 4.7K
SIP 4.7K
PRINTER STROBE
INT1
ACK
STROBE
FDC SEL
FDC DMA ACK
EXPANSION SEL
MEM ENABLE
I/O ENABLE
REFRESH
TEND1
DREQ1
TXS
INT2
RESET

write precompensation. The 9266, as mentioned earlier, combines the NEC 765A/Intel 8272 FDC with SMC's 9229 digital data separator. Thus, from the host CPU side, the 9266 looks just like these devices, including hardware and software compatibility.

The 9266 clock is generated by an 8-MHz oscillator composed of a crystal and IC21. Jumpers are provided to select write precompensation and allow 8-inch floppy-disk drives to be interfaced.

On the CPU side, the key requirements are interfacing the 9266 with both programmed I/O ($\overline{CS}$), for things like initialization and status check, and with DMA (DRQ, $\overline{DACK}$), for data transfer.

Programmed I/O is straightforward, with $\overline{CS}$ generated for I/O address 80, and $\overline{RD}$ and $\overline{WR}$ directly generated by the HD64180. This is the same scheme used to interface with other 80xx family peripheral devices.

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

For disk DMA, the 9266 asserts DRQ, which in turn causes HD64180 $\overline{DREQ1}$ assertion. The HD64180 performs DMA reads/writes to I/O address 0A0, which causes the 9266 $\overline{DACK}$ to be asserted, completing the transfer cycle. After the DMAC's programmed number of reads/writes has completed, the HD64180 $\overline{TEND1}$ output is asserted and, after inversion, causes the 9266 TC (terminal count) input to be asserted, completing the DMA operation. This is typically followed by the 9266 generating an HD64180 $\overline{INT\ 2}$ external interrupt. This interrupt-service routine can read the 9266's status to determine if errors occurred, etc.

However, one "gotcha," fixed by flip/flop IC14, conditions the 9266 DRQ output. It turns out that if 9266 DRQ directly generates HD64180 $\overline{DREQ1}$, the HD64180 may respond too quickly. This is because HD64180 $\overline{DREQ}$ input logic was designed to minimize latency, and $\overline{DREQ}$ can thus be recognized at a machine-cycle breakpoint. Unfortunately, the 9266 requires that at least 800 ns elapse from the time it asserts DRQ before the DMA transfer ($\overline{DACK}$) actually occurs. In other words, when the 9266 "asks" for service, it really doesn't want it . . . yet! To prevent accessing the 9266 too quickly after DRQ, DRQ from the 9266 is delayed at IC14 before issuing the $\overline{DREQ1}$ to the HD64180. DRQ is delayed by one $\overline{REF}$ cycle time.

Minifloppy double-density data transfers occur at a 250-kilohertz (kHz) data rate. Thus, each byte must be read within 32 μs. The disk-driver software reprograms the refresh-

Table 3: *Power connections for figure 4.*

		+5 V	GND	+12 V	−8 V
IC1	50256 DRAM 150 ns	8	16	—	—
IC2	50256 DRAM 150 ns	8	16	—	—
IC3	50256 DRAM 150 ns	8	16	—	—
IC4	50256 DRAM 150 ns	8	16	—	—
IC5	50256 DRAM 150 ns	8	16	—	—
IC6	50256 DRAM 150 ns	8	16	—	—
IC7	50256 DRAM 150 ns	8	16	—	—
IC8	50256 DRAM 150 ns	8	16	—	—
IC9	74LS374	16	8	—	—
IC10	74LS74	14	7	—	—
IC11	74LS157	16	8	—	—
IC12	74LS157	16	8	—	—
IC13	74LS157	16	8	—	—
IC14	74LS74	14	7	—	—
IC15	74LS74	14	7	—	—
IC16	2764 monitor ROM 200 ns	28	14	—	—
IC17	1488	—	7	14	1
IC18	HD64180	32	1,33	—	—
IC19	1489	14	7	—	—
IC20	(74LS156) optional	16	8	—	—
IC21	74LS240	20	10	—	—
IC22	74LS00	14	7	—	—
IC23	74LS240	20	10	—	—
IC24	FDC 9266	40	20	—	—
IC25	74LS158	16	8	—	—
IC26	74LS139	16	8	—	—
IC27	7407	14	7	—	—
IC28	7406	14	7	—	—
IC29	7660 regulator	—	3	—	—

Jumpers

JP1	(1x3)	ROM size
JP2	(1x3)	RAM size
JP3	(1x3)	CTS0 disable
JP4	(1x5)	RS-232C gate output
JP5	(1x3)	RS-232C gate input
JP6	(1x2)	Drive without ready
JP7	(1x3)	Write precomp mode
JP8	(2x6)	Write precomp value
JP9	(1x3)	Single-/double-sided drive
JP10	2 − 1x3	TXS function/drive size

request rate from every 80 ϕ cycles to every 40 ϕ cycles prior to disk DMA and reassigns it back to 80 ϕ cycles after the disk DMA is completed. The 9266 DRQ is delayed from between 40 ϕ cycles to 79 ϕ cycles. This is about 6 to 14 μs. Therefore, the 800-ns delay and 32-μs data-transfer constraint are both met. Note that 8-inch floppy double-density data transfer is twice as fast (500 kHz) and requires a refresh-rate increase to every 20 ϕ cycles.

Expansion Bus

The spare $\overline{CS}$ from address decoder IC26 (I/O addresses 0E0 to 0FF) and all major buses (address, data, control) are routed to the expansion bus. This allows an I/O expansion-board capability. The full complement of HD64180 control signals ($\overline{IOE}$, $\overline{E}$, $\overline{RD}$, $\overline{WR}$, etc.) allows easy interface to all standard peripheral LSI (large-scale integration) devices, including the 80xx, 68xx, and 65xx families. Example expansion boards could include a hard-disk controller, 1200-bits-per-second (bps) modem, or LAN (local-area network) interface.

Power Supply

The SB180 requires +5-V and +12-V power. A negative voltage is generated on board, which is used only by the RS-232C driver. The negative voltage is obtained by using a Zener diode to obtain +9 V from +12 V, which is then inverted using an Intersil 7660 converter. The +12-V power is used only for the RS-232C driver. Thus, the SB180 uses only significant power from the +5-V supply. Typically, this may be from 0.3 to 0.6 ampere (A) (depending on the proportion of the TTL [transistor-transistor logic] and memory devices that are CMOS)—about the same as a 5¼-inch floppy disk.

The SB180 ROM Monitor

While my initial discussion espoused the merits of the Z-System, every good single-board computer needs a ROM monitor that should be more useful than just booting the operating system from a floppy disk. The SB180 8K-byte ROM monitor includes commands for everything you need—from A to Z. It is supplied on an 8K-byte EPROM. (The SB180 also supports 16K- and 32K-byte ROMs, so additional commands or application programs can be supported.)

A ASCII table: Prints an ASCII table
B Bank select: Selects a 64K-byte memory bank
C Copy disk: Systems with 256K bytes of RAM can perform single-drive copies
D Display memory: Displays memory in hexadecimal and ASCII
E Emulate terminal: Console keyboard is echoed to the auxiliary RS-232C output, and the RS-232C input is echoed on the console display
F Fill memory: Any portion of memory is filled with a data byte
G Go to program: Starts program execution at specified address and optionally includes a breakpoint
H Hexmath: Prints the 20-bit sum and difference and 32-bit product of the two arguments
I Input port: Prints the 8-bit data input from specified port
K Klean disk: Formats a specified drive
M Move memory: Moves a block of memory
N New command: Enables new commands from extended ROM space
O Output port: Byte is output to specified port address
P Printer select: Toggles printer selection between the Centronics parallel port and the auxiliary RS-232C port
Q Query memory: Searches memory for pattern of 1 to 4 bytes
R Read disk: Reads specified sectors from drive into memory
S Set memory: Displays memory contents and allows new data to be entered
T Test system: Tests various system devices
U Upload hexadecimal file: Uploads Intel hexadecimal file from auxiliary or console serial port
V Verify memory: Compares two blocks of memory
W Write disk: Writes specified sectors to disk from memory
X eXamine CPU registers: Displays main and alternate CPU registers and prompts for modification of main registers
Y Yank I/O registers: Displays the HD64180 on-chip I/O registers
Z Z-System boot: Boots the Z-System from disk

Figure 5: *A list of the monitor commands.*

The SB180 ROM monitor provides commands to assist the design and debugging of SB180-related hardware and software. Also, it serves as a stand-alone training vehicle for the HD64180 CPU.

The monitor supports the following I/O devices:

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

The monitor supports one to four 5¼-inch, 48- or 96-tpi (tracks per inch), 40- or 80-track, double-sided double-density disk drives. During initial system checkout, such a drive should be connected to verify operation of the disk interface.

At $\overline{RESET}$, the monitor first checks for fatal RAM failure. If the RAM is bad, the monitor waits for a carriage return to be typed on the console in order to determine the console data-transmission rate and then prints 8 binary digits—each corresponding to one DRAM chip. A chip associated with a 1 is faulty. The RAM-check se-

(continued)

quence is repeated each time a carriage return is typed.

If the RAM is okay, the stack is set up, and the monitor checks to see if a disk is loaded in drive #0. If so, the DOS boot-load sequence (same as the Z command) is started.

The monitor commands are shown in figure 5.

THE LUNCHBOX COMPUTER

The only things you need to turn the SB180 into a full-fledged stand-alone microcomputer are a +5-V and +12-V power supply (or a 12-V battery and 5-V regulator), at least one 5¼-inch floppy-disk drive (initially), and a serial terminal.

You can mount such a small computer in many ways (see photos 3 and 4). If you use a half-height drive, the SB180 will fit on top of the drive inside a single full-height disk-drive chassis including power supply! Since the console serial port can automatically detect the terminal's data-transmission rate, you can carry your SB180 with you and connect it to any terminal (or computer emulating a terminal) running at 300, 1200, 9600, or 19,200 bps (other rates are optionally selected).

It is possible to fit the SB180, power supply, and two 3½-inch floppies into a lunchbox. If you want to get fancy, you could fit the SB180, power supply, one or two half-height floppies, the Micromint Term-Mite terminal board, and a keyboard into a small attaché case. Use any handy video monitor, and . . . voilà! You have just out-Osborned Adam Osborne!

I can imagine many unusual ways to package the SB180. When good LCDs (liquid-crystal displays) come down in price a bit more, I think the SB180 can form the basis of a functional notebook computer. The SB180 makes minimal use of the +12-V supply (and only for RS-232C operation) and can use less than 1 A at +5 V, so battery power is a real possibility. With just one floppy disk, the SB180 can become a super data logger. While my Z8 BASIC or FORTH single-board computer might ordinarily be used for such an application, not everyone likes to program in FORTH or BASIC. With a CP/M-compatible computer, a developer can now choose Pascal, C, FORTRAN, or even PL/I for applications. Since the data is already in CP/M format, it makes data analysis convenient.

The Z-System is provided with the complete SB180 board and software package. While it comes with a utility to read several other common 5¼-inch formats (like Kaypro and Osborne), its native format is identical to the well-optimized 386K-byte

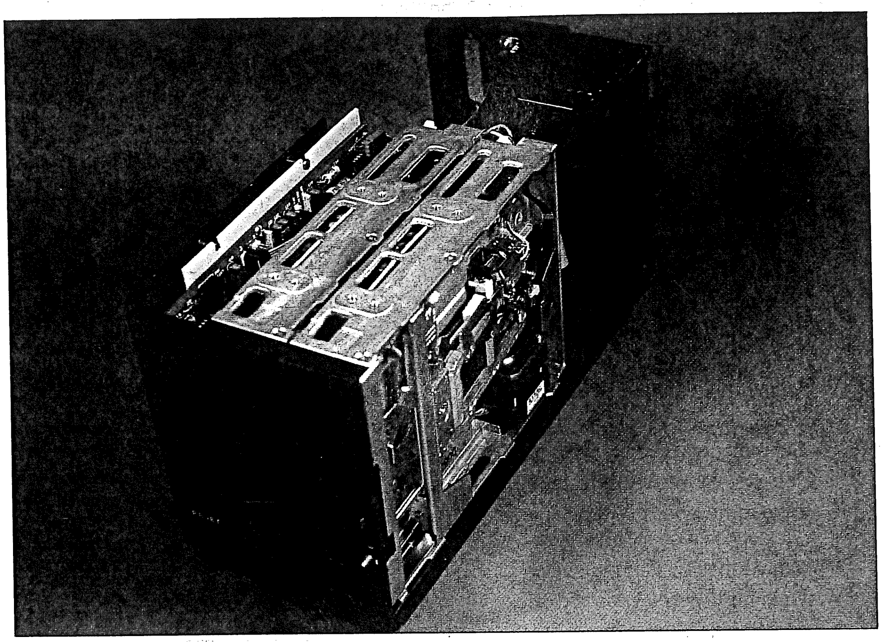


Photo 3: *The* SB180 *can be packaged in many off-the-shelf enclosures advertised in* BYTE. *It fits comfortably to the left of a pair of half-height* 5¼-*inch disk drives in a case available from* Disk Plus. *This enclosure also includes the power supply.*

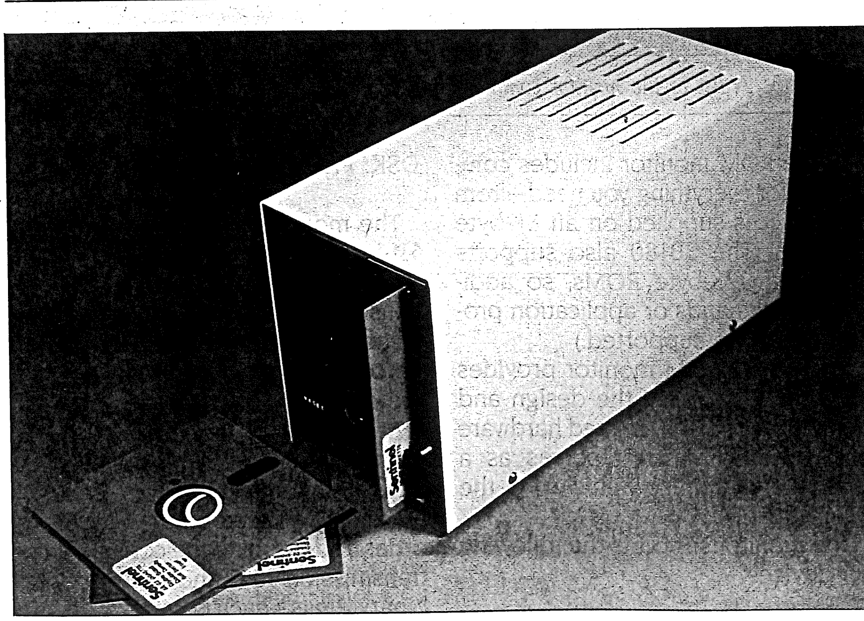


Photo 4: *The enclosure from photo 3 is shown with the cover on. The only additional hardware necessary to use the system is a serial terminal.*

double-sided double-density format of the Little Board from Ampro Inc. Provisions have been made to support 8-inch drives as well. Of course, it is possible to implement CP/M 2.2, CP/M Plus, MP/M II, TurboDOS, or Oasis if you prefer.

Although simple benchmarks are not to be taken as the last word in describing a computer's performance, I did run the BYTE Sieve of Eratosthenes prime-number program on the SB180 and the MPX-16 (8088-based with a 4.77-MHz clock, same as the IBM PC) using appropriate versions of Microsoft BASIC. For one iteration, the MPX using GW-BASIC took 203 seconds, while the SB180 using MS-BASIC took 147 seconds. An empty FOR . . . NEXT loop with 20,000 iterations took 27 seconds on the MPX and only 18 seconds on the SB180. Don't forget that these results are based on the 6-MHz implementation of the HD64180; with a 9-MHz clock, the results will be spectacular—not quite the speed of an IBM PC AT, but close!

EXPERIMENTERS

As always, I try to support the computer experimenter by rewarding diligence. If you build the SB180 from scratch, send me a picture, and I'll send you a copy of the BIOS and ROM monitor on disk (SB180 format, double-sided double-density) at no charge, provided it is for your personal use. A printed listing is available for a modest charge.

If you build, buy, or otherwise assemble an SB180 system, I would like to know about it. I will be designing expansion boards for the SB180 and can notify you of them in advance of publication. In addition, having your name will greatly simplify the organization of any users groups that might arise.

CIRCUIT CELLAR FEEDBACK

This month's Circuit Cellar Feedback is on page 416.

NEXT MONTH

In part 2 of this article, I'll describe the software for the SB180. ■

Special thanks to Tom Cantrell, Merrill Lathers, and Bob Stek for their contributions to this project.

The following items are available from

The Micromint Inc.
25 Terrace Dr.
Vernon, CT 06066
(800) 635-3355 for orders
(203) 871-6170 for information

1. SB180 computer board with 256K bytes of RAM. Complete with user's manual and ROM monitor.
   assembled and tested . . . SB180-1, $369
   complete kit SB180-2, $349
2. SB180 boot disk. Z-System DOS with limited utilities and BIOS. Provided on one 5¼-inch SB180 format double-sided double-density disk SB180-10, $49
3. Z-System, including ZRDOS, ZCPR3, an editor and utilities, ZAS assembler, BIOS source, and ZDM debugger. Complete with manuals. Provided on four 5¼-inch SB180 format double-sided double-density disks SB180-20, $190
4. HD64B180 chip (6 MHz) with data manual and 12.288-MHz XTAL $50
5. BYTE readers' special. Complete SB180 computer board with 256K bytes of RAM, user's manual, ROM monitor, and all the software listed in item 3. (Available through December 31, 1985.)
   assembled and tested . SB180-1-20, $499
   complete kit SB180-2-20, $479

All boards are complete with the exception of the 50-pin, 8-inch drive and 44-pin expansion headers, which are not populated. They are optionally available. Printer, power, disk, and terminal cables are available separately. Call for pricing.

Please include $10 ($7 less on item 4) for shipping and handling in the continental United States, $18 elsewhere. Connecticut residents please include 7.5 percent sales tax.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in book form from BYTE Books, McGraw-Hill Book Company, POB 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume* II covers December 1978 through June 1980. *Volume* III covers July 1980 through December 1981. *Volume* IV covers January 1982 through June 1983.

To receive a complete list of Ciarcia's Circuit Cellar project kits, circle 100 on the reader-service inquiry card at the back of the magazine.

BUILD THE SB180 SINGLE-BOARD COMPUTER

PART 2: THE SOFTWARE

BY STEVE CIARCIA

This computer reasserts 8-bit computing in a 16-bit world

The SB180 computer system represents the state of the art in 8-bit systems (for detailed specifications of the SB180, see last month's Circuit Cellar column). It also elevates the power-per-square-inch ratio to a new high (see photos 1 through 6). However, much of the hardware's potential would be wasted if the software were not as advanced. This month, I'll continue my discussion of the SB180 with emphasis on the DOS (disk operating system).

I began with some general ideas about what the software should do. I wanted a DOS, but it had to accommodate the new 3½-inch disk drives as well as older 5¼- and 8-inch units. A primary requirement was that it needed to be compatible with the most widespread "8-bit" DOS, CP/M 2.2. However, it needed to be free of the many restrictions and quirks of CP/M 2.2 and should represent a step forward in the logical development of operating systems.

The operating system and its utilities should not be separately developed and then stitched together, like a crazy quilt, but they instead should be developed concurrently so that they use a consistent command structure. Finally, since it is a Circuit Cellar project, the system must facilitate a high degree of user customization. It must be flexible enough to operate at 100 percent of the system's potential in one application, yet it must allow a terminal to be connected and a user to interact with it even if no disk drives are connected in another application.

ADVANCED FEATURES OF THE HD64180

From a programming point of view, the HD64180 microprocessor resembles its predecessor, the Z80, but also executes 10 additional instructions. The mnemonic names of these instructions are SLP, MLT, IN0, OUT0, OTIM, OTIMR, OTDM, OTDMR, TSTIO, and TST.

The SLP instruction puts the microprocessor into a "sleep" mode that uses little power; it would not be used in a DOS situation but is available for use in a user's own programs.

The MLT instruction is an impressive feature of the HD64180. It multiplies two 8-bit quantities and results in a 16-bit product. Again, this instruction is not usually used in operating-system software.

(continued)

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. He is the author of several books about electronics. You can write to him at POB 582, Glastonbury, CT 06033.

Photo 1: A *standard lunchbox can be converted into a powerful computer with the SB180.*

PHOTOGRAPHED BY PAUL AVIS

The remaining instructions perform functions like block output with increment, decrement, and repeat; input or output of any register to an immediate I/O (input/output) port address; and nondestructive AND logic operations on the various registers, I/O ports, and immediate data.

This last group of eight instructions would be convenient for use in a DOS based on an HD64180 microprocessor. Most of these instructions could be used in accessing the on-chip peripheral hardware like the asynchronous RS-232C ports or the memory-management unit (MMU).

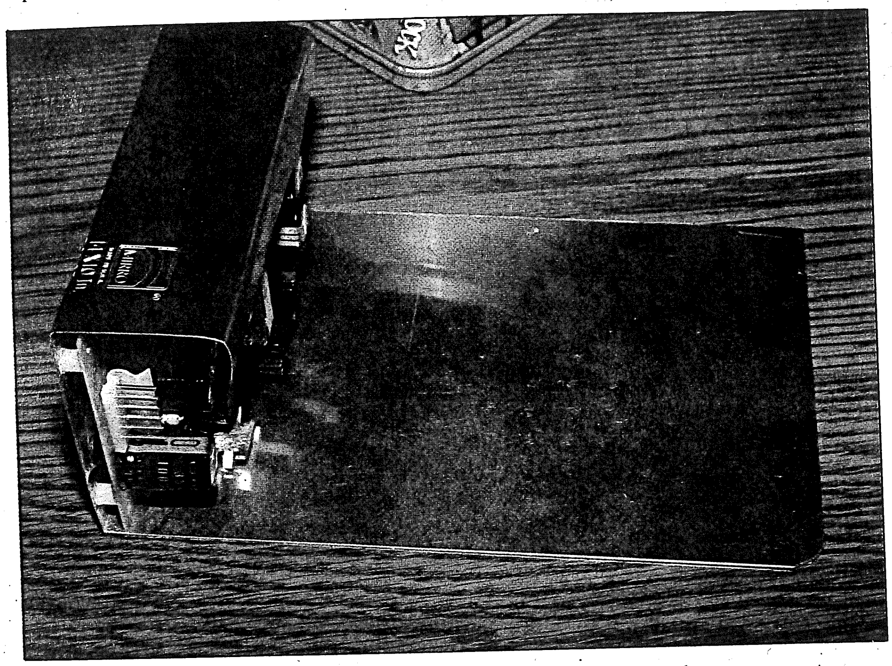


Photo 2: A *section of an aluminum cake pan is cut and bent to form a support for a small switching power supply.*

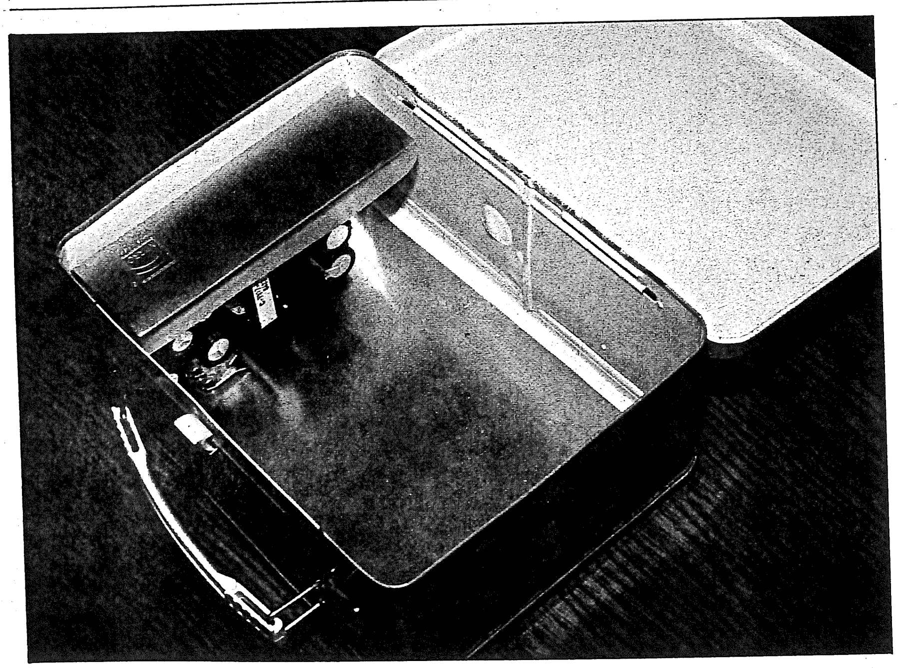


Photo 3: *The power supply is installed in the corner of the lunchbox.*

MATING HARDWARE AND SOFTWARE

I eventually found an operating system that met these challenging requirements in the Z-System from Echelon Inc. of Los Altos, California. The Z-System is compatible with programs that run under CP/M 2.2 and contains a multitude of improvements. The Z-System and its utilities were developed together in a common environment and share a common command structure. The system utilities can be used together in different ways to create new, powerful commands.

The system is adaptable, with several of the utilities being menu-driven. It even has menu processors that allow you to personalize the entire operating system to whatever level of sophistication you require. Best of all, extensive amounts of the source code to the system will be made available to those users who have the knowledge and desire to customize the system at the fundamental levels.

As I explained last month, better and faster microprocessors mean absolutely nothing if they are bound by inefficient operating systems. The SB180 with the Z-System is an unbeatable combination that seriously challenges the advertising-hyped credibility of 16-bit computers.

INTRODUCTION TO THE SOFTWARE

The most visible component of the Z-System is the command processor, called ZCPR3. It is the most visible component because ZCPR3 acts as the user interface to the rest of the operating system: interpreting commands, loading programs that are to be executed, and more.

ZCPR3 is more than just a command processor; it is also more than 70 utility programs, all of which make use of the special features of the environment provided by ZCPR3. These utilities can be used together in many ways to create new system commands to accomplish more powerful tasks. It is similar to UNIX in the way that individual programs can be combined to make a single new program.

DOS ARCHITECTURE

First, I'll define some terms. An *application program* is one that is not an intrinsic part of the operating system. Examples of this are WordStar or the ZCPR3 utilities. A *user area* is a way of partitioning the storage capacity of a disk and was originated in CP/M 2.2. Up to 32 user areas are on each disk drive; only user areas 0 through 15 are usually accessible. *Transient program area* (TPA) is the segment of memory, beginning at address 100 hexadecimal, where application programs are loaded by the command processor. A *file type* refers to the file's intended use and is indicated by the last 3 characters of a filename. In the Z-System, a filename has a total of 11 characters. (As an example, the filename FILENAME.TYP has the file type TYP.)

Figure 1 outlines the memory map of the SB180 software environment.

ZCPR3

To understand ZCPR3, you need an understanding of what a command processor is. A minimum definition is that it acts as the interpreter between you and the rest of the operating system. A command processor is the component of the operating system that prompts you for a command and then attempts to execute it.

ZCPR3 does everything mentioned above and much more. One of it's most powerful features is its ability to act as an interpreter for application programs that want to generate operating-system-level commands, not just for the user. This means that programs like the ZCPR3 utilities can generate new commands that are then fed to the operating system and processed as though you had typed the command at the console.

Another aspect of ZCPR3 that needs to be understood is the concept of system segments. Six system segments are in a fully implemented ZCPR3 system. A system segment is a file that is loaded into a predetermined area of memory.

The ZCPR3 command processor and utilities can call upon a system segment to perform a function or provide information. Memory-resident segments can be overlaid with a new segment at any time, providing the ZCPR3 command processor and utilities with extended functions.

ENVIRONMENT DESCRIPTOR

The first, and most important, segment is the Environment Descriptor

(continued)

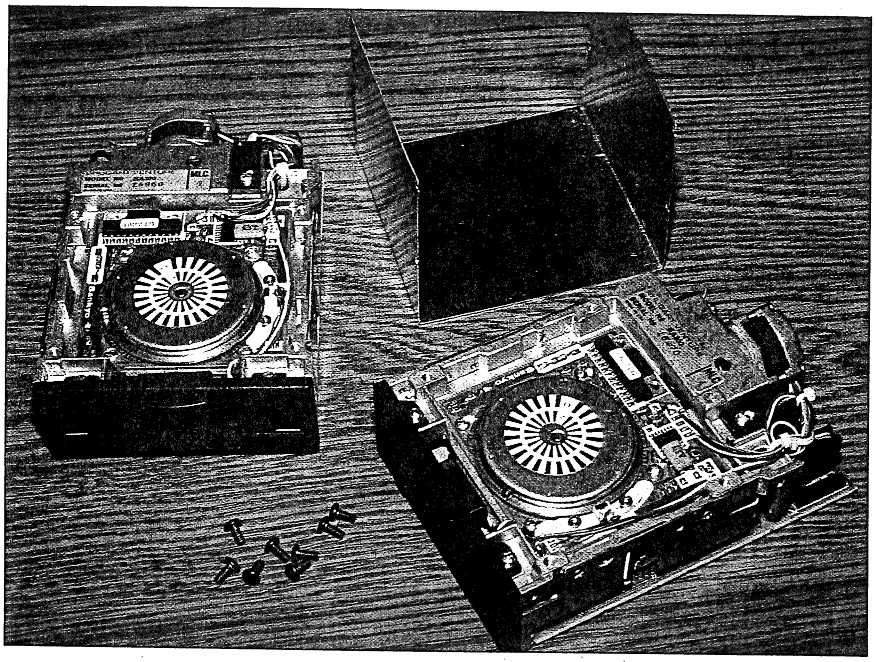


Photo 4: *Another section of cake pan is bent into a "U" shape and drilled to support two 3½-inch disk drives.*

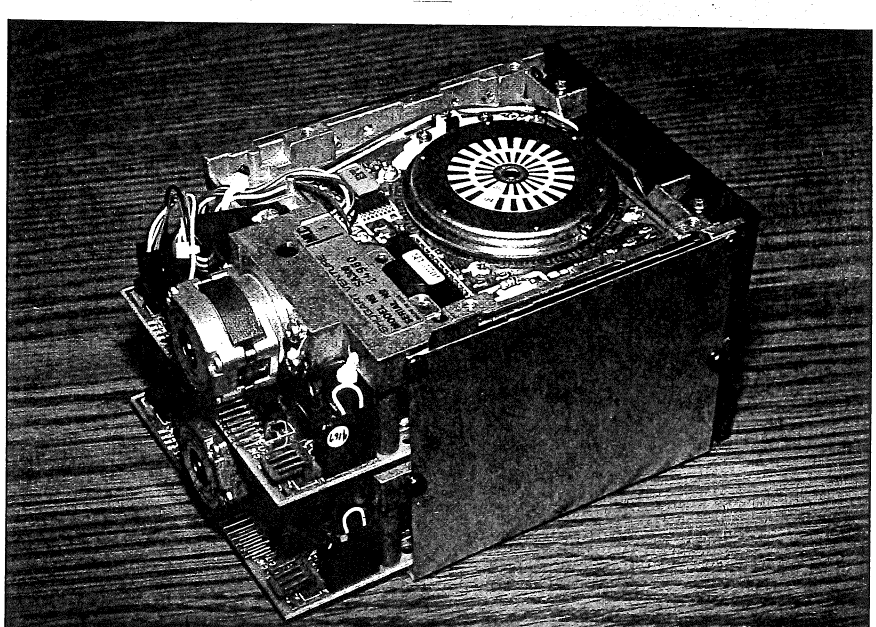


Photo 5: *The disk drives are installed in the cake-pan housing.*

(ENV), which occupies addresses 0FE00 through 0FEFF hexadecimal in the memory map (see figure 1). [*Editor's note: The addresses in this article are in hexadecimal.*] Because of the many possible choices in exactly how ZCPR3 can be implemented, ENV "describes" how that particular ZCPR3 implementation is configured. The ZCPR3 command processor and utilities use the information supplied by the ENV to determine the CPU (central processing unit) clock rate, number of disk drives installed in the system, where the other system segments can be found in memory, and more. When the Environment Descriptor segment is stored in a disk file, the file type is ENV.

Named Directory

The next segment is the Named Directory (NDR). In the memory map, NDR occupies addresses 0FC00 through 0FCFF. This segment also supplies information to the ZCPR3 command processor and utilities, as ENV does.

The NDR segment assigns symbolic names to disk drives and user areas of the system. This means that a name like BASIC may become associated with a particular user area on a disk drive. The ZCPR3 command processor and utilities will then refer to that disk drive and user area whenever a command is executed that contains the directory name BASIC. This gives you the ability to assign names to specific sections of your disk drives, which you can easily reassign by loading a new NDR segment. The file type of a file containing a Named Directory segment is NDR.

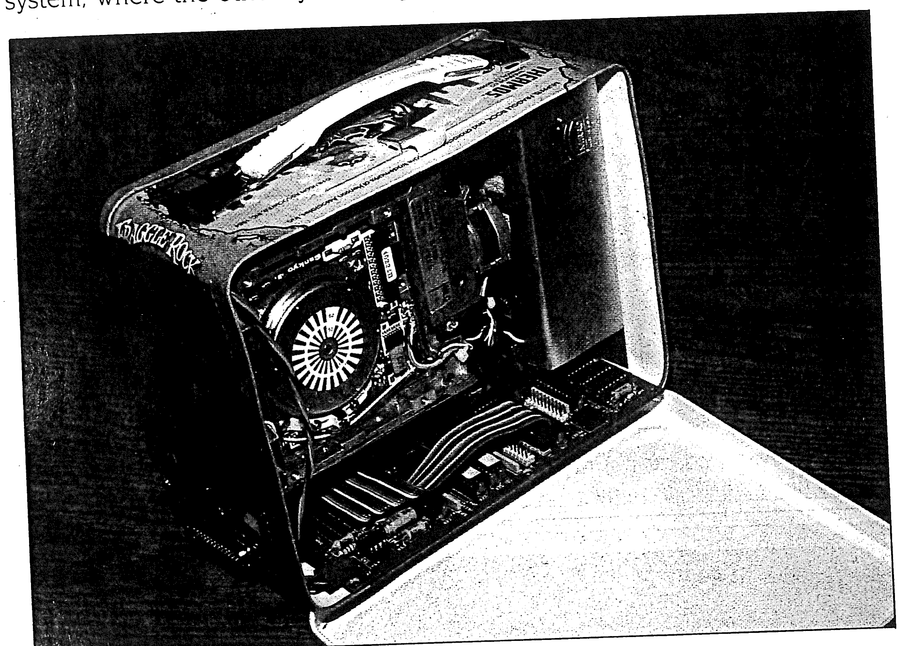


Photo 6: *The side of the lunchbox is cut out so that the drives protrude, and the drive assembly is bolted to the power-supply bracket. The SB180 is installed underneath, and cables are run for power and disk drives.*

Listing 1: *An example of the flow commands in a possible* STARTUP.COM *alias. Only if the operator responds with a "Y" in step 4 will the* MDSK /I *command in step 5 be executed (which will initialize the RAM disk). This is handy for rebooting the system after a program crash and preserving the contents of the RAM disk.*

Command	Comments
1: LDR SYS.RCP,SYS.FCP,SYS.NDR,SYS.Z3T;	| Load system segments
2: WHEEL SYSTEM;	| Set wheel byte
3: ECHO SHOULD THE RAM DISK BE INITIALIZED?;	| Ask question
4: IF INPUT;	| Get yes or no
5: MDSK /I;	| If yes, initialize RAM disk
6: FI	| Terminate IF (ENDIF)

Resident Command Package

The Resident Command Package (RCP) segment is a collection of subroutines that extend the intrinsic commands of the operating system. An intrinsic command is a routine that resides in memory and that can be executed without disturbing the TPA. As an example in the CP/M 2.2 environment, the DIR command is an intrinsic command, but the STAT command (which loads the STAT.COM program into the TPA and therefore disturbs the TPA) is not.

In the software supplied with the SB180, the intrinsic commands that reside in the ZCPR3 command processor are GO, SAVE, GET, and JUMP. The intrinsic commands added by the RCP are CP, ERA, TYPE, LIST, P (PEEK), POKE, PROT, and REN. As you see, the RCP adds many commands. Additional commands are available within an RCP, but the RCP segment must fit within a 2K-byte area of memory. The above commands make it just a few bytes short of this limit.

If you include another command, you would have to disable one of the existing commands in order for the RCP to fit into its assigned area. In the memory map, the RCP occupies addresses 0F200 through 0F9FF. The file type of a file containing a Resident Command Package segment is RCP.

Flow Control Package

The Flow Control Package (FCP) resides between addresses 0FA00 and 0FBFF. It is unique to ZCPR3; no comparable feature is found in any other microcomputer operating system. The FCP adds conditional testing to operating-system-level commands.

An example of conditional testing is the IF. . .THEN . . . ELSE statement in high-level languages like BASIC. The FCP gives the ZCPR3 command processor this testing capability. It is not usually used while you are entering commands at the console, but you can take full advantage of it in batch-processing operations. The file type of a file containing a Flow Control Package segment is FCP.

To understand how the flow commands are useful, you must first know about the ZCPR3 flow state, which is either true or false. While the flow state is true, the ZCPR3 command processor will execute all commands. If the flow state is false, the ZCPR3 command processor will ignore all commands except ELSE and FI.

The IF command is capable of setting the flow state to either true or false. The IF command can evaluate a number of tests: the existence of a file on disk, whether or not a file is empty, the state of the wheel byte (explained in detail later), and more.

A good example of how to use the IF command is shown in listing 1. ZCPR3 allows you to nest the IF/ELSE/FI (ENDIF) flow commands up to eight levels deep.

INPUT/OUTPUT PACKAGE

The Input/Output Package (IOP) segment acts as a traffic cop in routing input and output to and from peripheral devices. The print spooler supplied with the SB180 is an example of an IOP. Other IOPs let you set up programmable function keys or capture characters in a disk file that are normally sent to the console or list device.

The IOP occupies addresses 0EC00 through 0F1FF in the memory map, and the file type of a file containing an Input/Output Package segment is IOP.

TERMINAL CAPABILITIES

The Terminal Capabilities (TCAP) segment is actually contained within the

(continued)

```
 ------------------------------------  3FFFF Hex
 | RAM Disk (192K bytes)             |
 ------------------------------------  0FFFF Hex
 | ZCPR3 External Stack              |
 ------------------------------------  0FFD0 Hex
 | ZCPR3 Multiple Command            |
 | Line Buffer                       |
 ------------------------------------  0FF00 Hex
 | ZCPR3                  | TCAP     |
 | Environment            ----------- 0FE80 Hex
 | Descriptor Segment                |
 ------------------------------------  0FE00 Hex
 | ZCPR3 Wheel Byte                  |
 ------------------------------------  0FDFF Hex
 | ZCPR3 External Path               |
 ------------------------------------  0FDF4 Hex
 | ZCPR3 External File               |
 | Control Block                     |
 ------------------------------------  0FDD0 Hex
 | ZCPR3 Message Buffers             |
 ------------------------------------  0FD80 Hex
 | ZCPR3 Shell Stack                 |
 ------------------------------------  0FD00 Hex
 | ZCPR3 Named Directory             |
 | Segment                           |
 ------------------------------------  0FC00 Hex
 | ZCPR3 Flow Control                |
 | Package Segment                   |
 ------------------------------------  0FA00 Hex
 | ZCPR3 Resident Command            |
 | Package Segment                   |
 ------------------------------------  0F200 Hex
 | ZCPR3 Input/Output                |
 | Package Segment                   |
 ------------------------------------  0EC00 Hex
 | SB180 BIOS (Basic Input/          |
 | Output System)                    |
 ------------------------------------  0DA00 Hex
 | ZRDOS Disk Operating              |
 | System                            |
 ------------------------------------  0CC00 Hex
 | ZCPR3 Command Processor           |
 ------------------------------------  0C400 Hex
 | TPA (Transient Program            |
 | Area)                             |
 ------------------------------------  00100 Hex
 | Page 0 Buffers and                |
 | Reserved Locations                |
 ------------------------------------  00000 Hex
```

Figure 1: *The memory map of the SB180's software system, as initially configured. You can modify the system to meet your own needs.*

Environment Descriptor segment, although it can be loaded independently. It resides at addresses 0FE80 through 0FEFF in the memory map. Information stored here describes characteristics of your terminal, specifically the strings that invoke the terminal's clear screen, cursor addressing, highlight on/off, and other functions.

Also stored in this segment are the codes generated by any arrow keys on the terminal. The ZCPR3 command processor and utilities use this information to enhance interaction with you, by offering flashy displays and using the arrow keys for various functions. The important thing to understand about the TCAP segment is that it is easily changed if you attach a different terminal. Because the ZCPR3 utilities refer to the segment for their information, they do not need to be changed as well. This feature can be described as terminal independence. The file type of a file containing a Terminal Capabilities segment is Z3T.

Other ZCPR3 Concepts

The path, originally incorporated in ZCPR2, is a ZCPR3 concept that provides a tremendous amount of flexibility. The path lets ZCPR3 search other directories (disk drives and user areas) if the program or file to be invoked is not in the active directory.

An example: The path is set up for ZCPR3 to search (in the following order) the current drive and user area, drive A/user area 0, and drive A/user area 15. When you issue a command that is not an intrinsic command, the ZCPR3 command processor begins searching for the file of that name in the current drive and user area. If the file is found, it is loaded and executed. However, if the file is not found, the path instructs the ZCPR3 command processor to continue the search at drive A/user area 0. Again, if the file is found, it is loaded and executed. If the file is not found, the ZCPR3 command processor searches drive A/ user area 15. Once again, if the file is found, it is loaded and executed. The SB180 software allows up to five levels of search.

The flexibility derives from the fact that the path is, like the system segments, changeable at any time. What this means to you is that your frequently invoked programs can be stored in a specific drive and user area, usually A15, and can be invoked from any currently active drive and user area without needing to specify the disk drive, as long as the path points to the appropriate directory.

The path is also used by many of the ZCPR3 utilities. For example, the Help utility will search along the path when looking for HLP files.

The Wheel

The last important component of ZCPR3 that I should describe is the wheel byte, which resides in the distribution software for the SB180 at address 0FDFF. If the RAM (random-access read/write memory) at that address contains a zero value, the wheel byte is considered reset (off); if the address contains a nonzero value, the wheel byte is considered set (on).

The wheel byte functions as a security system. All the intrinsic ZCPR3 commands can be set up so that they check the status of the wheel byte before they execute. This is ideal for situations where a security function is necessary, like a public computerized bulletin-board application. If "dangerous" commands like ERA (erase files) are set up to check the wheel byte, and a user who does not have wheel privileges attempts to use ERA, all that will happen is that the message "No Wheel" will be displayed.

Several ZCPR3 utilities will function only if the wheel byte is set on; otherwise, they abort immediately. Also, both intrinsic commands and a utility program let you manipulate the status of the wheel byte; both require a password to operate.

The ZCPR3 Utilities

All ZCPR3 utilities (there are more than 70) are included in the full SB180 software package (a subset is included with the boot disk only). See the "Z-System Utilities" text box on page 93. About 20 percent of the utilities correspond to intrinsic commands. Consequently, if you elect to omit, for example, ERA as an intrinsic command, you can use the ERASE.COM utility to perform the same function.

ZCPR3 utilities all share many common features, the most significant of which is that they reference the Environment Descriptor segment to determine information about the system configuration, e.g., to determine the location of the Terminal Capabilities segment or the Named Directory segment. However, since the Environment Descriptor segment is not necessarily located at the same addresses in every ZCPR3 configuration, the ZCPR3 utilities must be installed for that particular configuration. This is an easy task because, as you might surmise, a ZCPR3 utility will do it for you! (Note that the software supplied with the SB180 does not require this installation; it is preinstalled for the memory map of the SB180 default configuration.)

Finally, if you get lost, you can always find help. A help screen for any of the ZCPR3 utilities can be called by invoking it with a command-line parameter of //, so that LDR // as a command calls up a help screen for the LDR utility. Other command-line options are usually preceded by a single slash or a space character.

Shells and Aliases

To understand the concept of a shell, think of your computer system as an onion. It is made up of various layers of software and hardware, with the microprocessor at the very center. The outermost, and visible, layer would be an application program like WordStar. When WordStar is executing you are presented with its displays, and the computer will process your input in accordance with the commands of WordStar. When you exit WordStar, the outermost layer of the onion is removed and you see the next inner layer, which is the ZCPR3 command processor. If you looked deeper, you would see ZRDOS as the next layer, then the BIOS (basic input/output system).

While each layer has its own appearance and commands, it has to rely on deeper layers to execute these commands. A shell is an additional layer that fits between the application-program layer and the ZCPR3 command processor. A shell can present its own displays and process your input in accordance with its own commands, and it relies on the ZCPR3 command processor (the next deeper layer) to actually execute the commands. A common use of a ZCPR3 shell is that of a translator between you and the ZCPR3 command processor that converts single keystrokes you type to command sequences that are then executed by the ZCPR3 command processor.

Finally, a ZCPR3 shell can be nested. Multiple layers of shells can be simultaneously active, with only the outermost layer visible to you. This gives one shell the capability of invoking another shell. After the second shell is finished executing, control is returned to the shell that invoked it. A shell knows how and when to return to itself.

An alias is a .COM (executable binary program code) file created by the Alias program that contains one or more commands that are to be placed in the multiple command line buffer and then executed. Command strings can be built into an Alias program and then invoked by a single command. This is an impressive and powerful feature of ZCPR3.

As a simple example, you create numerous files with file types of BAK, HEX, and SYM, which are no longer required. Rather than repetitively entering the commands ERA *.BAK, ERA *.HEX, and ERA *.SYM, an alias named CLEANUP.COM is created. CLEANUP.COM contains the commands ERA *.BAK, ERA *.HEX, and ERA *.SYM. When CLEANUP.COM executes, it places its command string into the ZCPR3 command line buffer, and then the ZCPR3 command processor executes those commands, erasing the files specified.

Aliases can be considerably more powerful than this example because they are the technique used under

(continued)

Z-SYSTEM UTILITIES

The following is a list of some of the more interesting Z-System utilities. Keep in mind that there are more than 70 utilities in all, so this is just a small sample to give you an idea of the capabilities available.

AC: This stands for archive/copy. This utility copies a file from one directory to another, with the option of copying only files that have been modified since last archived.

CLEANDIR: Clean directory removes all deallocated references to files on the disk and sorts the remaining active filenames in either ascending or descending order. Used often, and you're nearly guaranteed a successful **UNERASE** (see below).

CONFIG: This menu-based utility is used to configure BIOS parameters like I/O port speeds, set up the printer as serial or parallel, alter the number of CPU wait states, and more. (This utility was written specifically for the SB180.)

DPROG: This is a device-programming utility that is capable of sending predefined byte sequences to peripheral devices like printers and terminals.

FINDF: Find file searches for a file or files in all disks and user areas of the system and reports their location(s).

HELP: Invokes the help subsystem. Entering HELP ZCPR3 will invoke the ZCPR3 on-line documentation. Other help information can be created with a text editor and displayed using this utility. (This utility uses the TCAP system segment to enhance its displays.)

HELPCHK: This utility checks files to be used with the **HELP** program for proper structure and syntax.

MENU: This utility invokes the menu subsystem under ZCPR3. Menu files can be created with a text editor according to the instructions in the help file. This is a Z-System shell. (This utility also uses the TCAP system segment.)

PAGE: This utility sends the contents of a file or files to the console for viewing. The data is "paged," filling only one screen at a time and then waiting for the operator to strike a key. (Also uses TCAP.)

PWD: The print working directories utility shows currently available named directories.

SHOW: This is a menu-oriented display of the status of the ZCPR3 environment, which includes all system segments, the path, and more. (Also uses TCAP.)

TCMAKE: This menu-oriented utility allows creation of TCAP system-segment files in case your terminal is not already handled by the SB180 system. (An associated utility, TCCHECK, checks TCAP system-segment files for errors in structure or syntax.)

UNERASE: Does just what its name implies; it allows the recovery of accidentally deleted files if run immediately after the deletion. Usually successful if **CLEANDIR** is run frequently.

VFILER: This is an extremely useful utility to manipulate files in various ways, such as sending contents to printer, displaying on console, copying, unsqueezing, etc. Command entry occurs merely by pointing to the filename and selecting a command. **VFILER** can be personalized with up to 10 additional user-determined commands.

Z3INS: Use this utility to install all ZCPR3 utilities if any changes are made in the location or structure of the ZCPR3 system. The supplied utilities are already installed for the SB180 environment but will need to be reinstalled if the system is reconfigured in any way.

ZAS: A relocating macro assembler.

ZCPR3 to create new commands, using whatever other programs that may exist on disk in different ways to add easily invoked powerful functions. Aliases support nesting of other aliases within their command string and also support parameter substitution so that programs invoked by an alias can be fed parameters specified when the alias is invoked, in a fashion similar to Digital Research's SUBMIT.COM utility. This parameter substitution allows the command sequence contained in the alias to operate on different filenames or with different options.

ZRDOS AND BIOS

ZRDOS is the core of the SB180 operating system. It occupies space in the memory map from 0CC00 to 0D9FF. ZRDOS, like the CP/M 2.2 BDOS (basic disk operating system), creates the standard virtual machine that application programs are written for. This lets software vendors write one version of a software package, which will execute on more than one type of hardware configuration. The virtual-machine environment is provided via standardized system functions, such as sending a character to the console or checking the status of the list device. The CP/M 2.2 BDOS contains 39 functions; ZRDOS provides these same functions, thereby maintaining compatibility, and adds four more.

Two aspects of ZRDOS are visible to you in comparison with CP/M 2.2. The major significant difference of ZRDOS is that when a new disk is placed in a disk drive, it is not necessary to type Control-C to log in the new disk, as in CP/M 2.2. The other difference is improved error messages. Instead of the cryptic Bdos Err on A:, you see Read Error on A:, which is much more meaningful. ZRDOS also includes file archive handling compatible with CP/M Plus and MP/M, which can be used to make automatic backup copies of a file that has been changed. ZRDOS also recognizes what are known as wheel-protected files and does not allow modification to those files unless the wheel byte is set.

The BIOS for the SB180 handles several important functions not found in most computers and uses the on-chip hardware of the HD64180 to the fullest. It was written specifically for the SB180, with emphasis on rapid and efficient code. Disk operations are extremely quick in comparison with other machines.

The SB180 BIOS resides between addresses 0DA00 and 0EBFF in the memory map. Like the BIOS in any microcomputer, its function is to act as the interface between the software and hardware. Another way of saying this is that, because different hardware configurations may be used, a certain part of the operating-system software exists that is specially customized for that hardware configuration. Because of the customization of the BIOS, the same DOS can function on many different computer types, despite the fact that there may be significant differences between the machines.

The BIOS is responsible for interfacing all peripheral devices like floppy disks and video terminals that are used on a computer to the standard virtual-machine environment created by the operating-system software.

For example, an application program requests ZRDOS to send a character to the console. ZRDOS is the same no matter what machine it is operating on and has no way of knowing which I/O port address the console might be found at, or anything else about the console. Yet it does know that the request has to do with I/O, and ZRDOS passes this request to the BIOS to send the character to the console. Finally, the BIOS is the component that actually transmits the character to your terminal, because it has been configured to know that the "console" is actually a terminal connected to I/O port number two. This example shows how the peripheral devices attached to the machine (console attached to I/O port number two) are interfaced to the software of the machine (request to output a character to the console).

The SB180 BIOS incorporates this type of standard software/hardware interfacing and several special features. The most important feature is the integrated RAM disk. The upper 192K bytes of the memory can be set up by the BIOS to be used as an extremely fast file-storage device, which, to application programs and the operating system, looks like disk drive M. This integrated RAM disk is a powerful tool, one that gives the SB180 an incredible performance advantage. One of the two direct-memory-access controllers (DMACs) of the HD64180 is dedicated to the RAM disk, providing the best performance possible.

Table 1 lists programs that have been tested to run in the software environment of the SB180. The programs listed are not the only programs that will run; they represent all that was available to be tested.

It is important to understand that, although the Z-System is compatible with programs designed to run under CP/M 2.2, it is a case of upward compatibility. This means that these pro-

Table 1: A *list of programs tested and known to be compatible with the SB180's operating system.*

MicroPro:	WordStar 3.0, WordMaster, MailMerge, StarIndex, SuperSort
Microsoft:	Multiplan, Macro-80, BASIC-80, BASIC compiler
Digital Research:	MAC, SID, ZSID, CB-80, Pascal/MT+
Sorcim:	SuperCalc2
Ashton-Tate:	dBASE II
Borland International:	Turbo Pascal 2.0
Manx:	Aztec C 1.05g
CompuView:	Vedit
Others:	T/Maker III, The Word Plus, Punctuation and Style, MIX, Modem7, MDM7, MEX, BDS C, C/80

grams usually cannot use most of the advanced features of the Z-System but simply perform as they would in the environment they were intended for. Programs written for CP/M 2.2 can benefit from some aspects of the Z-System, but other aspects of the Z-System cannot be utilized.

STARTING THE SYSTEM

Two modes of operation are available when starting the system. The first mode is the SB180 monitor. It will be invoked if the computer is powered on without floppy-disk drives connected or if disk drives are connected but no disk is in drive A. The monitor has its own command set to perform such functions as examining and changing the contents of memory, transmitting and receiving byte values to and from the I/O ports, and more.

The monitor software resides in the on-board system EPROM (erasable programmable read-only memory) and is used for debugging the system hardware and as a bare-bones operating system for SB180 users who have not added floppy-disk drives. If the system is used in this way, simply strike the Return key so that the monitor can determine the console's data-transmission rate. (A listing of the monitor commands was provided last month.)

The second mode of operation involves attaching one to four floppy-disk drives (3½-, 5¼-, or 8-inch) to the appropriate drive connector on the SB180 and simply placing a disk that has the operating system on it into drive A and powering on, or resetting, the system.

This is referred to as cold booting and loads the operating system into the SB180's memory. Several messages are displayed in the process of cold booting, most of which are originated by the ZCPR3 utility LDR.COM. When the system has completed the cold-boot process, you are presented with the system prompt, which is A0:BASE> if you are using the distribution software. The system is now ready to accept your commands.

The cold-boot process has many stages. An important one is the execution of the STARTUP.COM program. The SB180 operating system searches for the STARTUP.COM program as the last step of the cold-boot process, and if it is found, it is executed. STARTUP.COM is created by the Alias program, and its major role in the cold-boot process is to load the ZCPR3 system segments by placing the command LDR SYS.ENV,SYS.RCP, SYS.FCP,SYS.NDR into the multiple command line buffer.

This is not the only role of the program. You can easily customize STARTUP.COM so that whenever the computer goes through the cold-boot process, an additional series of commands are executed automatically. Because each disk can have its own personalized STARTUP.COM, it is possible to create turnkey systems for specific applications. Disks could be set up specifically for word processing by using STARTUP.COM to automatically load and execute WordStar, or for a turnkey database operation. An unattended remote-access computerized bulletin board could have a STARTUP.COM set up so that if power failed and then was restored, all the needed commands to start the system again would be executed. The STARTUP.COM concept gives you a great deal of flexibility and convenience.

RAM-DISK INITIALIZATION

The RAM disk is an exciting feature of the SB180. It will improve system performance many orders of magnitude when used. Like all RAM disks, it has some characteristics that should not be overlooked. The RAM disk, unlike a floppy disk, does not retain its contents when power is removed, so you must be sure to make floppy-disk copies of files used in the RAM disk. It is quite possible to use aliases to make the process convenient, so that when you edit a file that resides in the RAM disk, it is automatically copied onto a floppy disk at the conclusion of the edit session.

The SB180 BIOS is written to *not* initialize the RAM disk when a cold boot is performed. This is so that when the Reset button of the computer is activated, the RAM-disk contents are retained. You may need to use the Reset button if a buggy program goes into an endless loop, and it is nice to reset the computer without losing the contents of the RAM disk!

A utility program called MDSK (the command is MDSK /I) is used to initialize (format) the RAM disk. It is important that MDSK /I not be used in the STARTUP.COM alias, because this will destroy the contents of the RAM disk whenever the Reset button is used. An alternative is to use the flow commands of ZCPR3 to query the user, so that the STARTUP.COM alias may perform the command or not, depending on the user's response.

CHANGING THE DRIVE/USER AREA

The Z-System accepts the generic commands found in both MS-DOS and CP/M for changing the currently active disk drive. For example, to select the B drive as active, simply

(continued)

Table 2: *The different floppy-disk formats that can be processed with the* SB180 BIOS *and their associated storage capacities. You do not need a conversion program to format, read, or write these formats (tpi=tracks per inch; dsdd=double-sided double-density; ssdd=single-sided double-density).*

Format	tpi	Sides	Capacity per disk
SB180 native	96	2	782K bytes (dsdd)
SB180 native	48	2	386K bytes (dsdd)
Hitachi QC-10	48	2	286K bytes (dsdd)
Kaypro 2	48	1	191K bytes (ssdd)
Ampro	48	1	188K bytes (ssdd)
Osborne 1	48	1	183K bytes (ssdd)

type B:. To select the integral RAM disk, type M:. When a drive is selected as active, it becomes the default drive. In other words, it is where programs and files will be searched for first, unless specified otherwise by the user or application program.

These generic commands are actually a subset of the commands available under the Z-System to select the active disk and user area. Two major types of commands are available under the Z-System for this purpose: the DU: form and the DIR: form. Both forms are recognizable by the trailing colon character.

The SB180 BIOS can automatically recognize different floppy-disk formats, so exchange of disks with dissimilar formats becomes easy. See table 2 for a list of supported formats. To change to a disk of different format, you may have to type Control-C.

The DU: form is made up of two components. D is a disk drive, and the acceptable range for it is A through P. U is a user area, and its acceptable range is 0 through 15. Thus, DU: forms of A0:, B6:, and M15: are allowed, but forms like Z0: or G33: are invalid. So, to move to the A2: drive/user area, simply type A2:. Another way of using the DU: form is to realize that the D and U are optional. This means that forms such as 3: and A: are valid.

The DIR: form is derived from the Named Directory system segment, which associates a symbolic name with a specific drive and user area. A directory name is up to eight characters in length. For instance, if the symbolic name ROOT is defined in the Named Directory segment as being associated with A15, and if you type ROOT:<cr> at the system prompt, A15 will be selected as active.

To have the list of defined directory names printed for you, use the PWD (print working directories) utility. The default configuration of the SB180 system software allows 14 directory names to be defined, but this number can be changed (see the "User Customization" section).

A characteristic of the DIR: form is that a password can be associated with a directory name. The password can be up to eight characters in length and is also defined in the Named Directory segment. If you type a DIR: form that is passworded, a prompt of PW? is presented, and if the proper password is not entered, you are not allowed to enter that directory. (Note that the DU: form, if enabled, allows password-free access to that directory. To make the DIR: passwords come into full force, reconfigure the system to not accept the DU: form.)

SECURE SYSTEMS

The SB180 system software has been configured with a minimal amount of security. This was done because most users will probably not be in a situation where public access to their SB180 will be allowed.

However, the system does possess extensive options for security. If you intend to use the SB180 as a public-access computerized bulletin board, you can reconfigure the software into a more secure system. For example, you could deny use of the ERA command to "ordinary" users.

I previously discussed how the DU: form of changing the active disk drive and user area bypasses any passworded Named Directory entries. In a secure system, the DU: form may be disabled, so that the only way to refer to any other area would be to use the DIR: form. Then, because the DIR: form requires a password if one has been defined, critical areas of the system can be password-protected.

These changes are implemented by editing the source code supplied with the system to set the new options and then merging the changes into the operating-system software.

USER CUSTOMIZATION

You can customize a number of other areas in the SB180 system software. The first is the BIOS, using the CONFIG utility.

The BIOS source code is included in the full software package to make easy installation of significant changes in the hardware-configuration information. The assembler, ZAS, lets you regenerate the operating system into new and different forms. An example is if you want to rewrite your RCP so that some of its commands will respond to the wheel byte. Another example is expansion of the Named Directory system segment to handle more than 14 directory names.

Although some of the above examples require programming knowledge, a great degree of customization can be done without specific programming knowledge through the use of Alias programs and the shell utilities.

The STARTUP.COM alias is a likely candidate for personalization. It is normally executed only when the system does a cold boot. When using the SB180 for word processing, I use a STARTUP.COM command sequence that automatically copies the files I will be using to the RAM disk. Once the files I am using are copied into the RAM disk, execution of commands is nearly instantaneous. Having STARTUP.COM automatically place the appropriate files in the RAM disk to gain the benefit of the speed of access can be used in many other areas besides word processing.

SUPPORT

A full set of manuals are included with the SB180 operating-system software. This includes manuals for ZAS, ZDM, EDIT, ZRDOS, and some SB180 utilities. Documentation for the ZCPR3 utilities and ZCPR3 itself are included in the HLP files.

EXPERIMENTERS

As always, I try to support the computer experimenter by rewarding diligence. If you build the SB180 from scratch, send me a picture and I'll send you a copy of the BIOS and the ROM monitor on disk (SB180 double-sided double-density format) at no charge, provided it is for your personal use.

If you build, buy, or otherwise assemble an SB180 system, I'd like to know about it. I will be designing expansion boards for the SB180 (the first one is a 300/1200-bps modem) and can notify you of them in advance of publication. In addition, having your name will greatly simplify the organization of any users groups that might arise.

The SB180 is a fully supported Circuit Cellar project. I have arranged for the hardware to be available in kit or assembled form, and I contracted with Echelon Inc. to write the BIOS and integrate it into the operating system. Echelon has telephone technical assistance available and also has affiliations with more than 40 public remote-access bulletin-board systems, called Z-Nodes. Z-Nodes are located throughout the nation, and there may be one in your area. You can find ZCPR3 utility-program updates and informative newsletters about the Z-System on a Z-Node for the price of a phone call.

In Conclusion

This has been a big project, even though it's only a small board. Sports cars have a lbs/HP rating. It's too bad there isn't an MPS/sq. in. rating that could be used to truly compare the capabilities of the SB180 to other computers.

Like the Z8 and 8052, the HD64180 has joined the Circuit Cellar preferred processor list, and you can expect it to be well supported with a variety of expansion peripheral devices. A hard-disk controller and modem are currently in the works. As soon as I figure out how to design them and gather together the software people who know how to glue it together, I'll let you know. Until then, I'll just keep plinking along with simple projects like home controllers, intelligent terminals, and voice-recognition systems. My PC board designer needs a vacation.

Circuit Cellar Feedback

This month's feedback is on page 388.

Next Month

I'll build a single-chip 1200-bps modem. ■

Special thanks to Tom Cantrell, Frank Gaude', Merrill Lathers, Dave McCord, Joe Wright, and the people at Custom Photo and Design Inc. and Tech Circuits Inc. for their contributions to this project.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in book form from BYTE Books, McGraw-Hill Book Company, POB 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume* II covers December 1978 through June 1980. *Volume* III covers July 1980 through December 1981. *Volume* IV covers January 1982 through June 1983.

The following items are available from

The Micromint Inc.
25 Terrace Dr.
Vernon, CT 06066
(800) 635-3355 for orders
(203) 871-6170 for information

1. SB180 computer board with 256K bytes of RAM. Complete with user's manual and ROM monitor.
   assembled and tested . . . SB180-1, $369
   complete kit SB180-2, $349
2. SB180 boot disk. Z-System DOS with limited utilities and BIOS. Provided on one 5¼-inch SB180-format double-sided double-density disk SB180-10, $49
3. Z-System, including ZRDOS, ZCPR3, an editor and utilities, ZAS assembler, BIOS source, and ZDM debugger. Complete with manuals. Provided on four 5¼-inch SB180-format double-sided double-density disks. SB180-20, $190
4. HD64B180 chip (6 MHz) with data manual and 12.288-MHz XTAL $50
5. BYTE readers' special. Complete SB180 computer board with 256K bytes of RAM, user's manual, ROM monitor, and all the software listed in item 3. (Available through December 31, 1985.)
   assembled and tested . SB180-1-20, $499
   complete kit SB180-2-20, $479

All boards are complete with the exception of the 50-pin 8-inch drive and 44-pin expansion headers, which are not populated. They are optionally available. Printer, power, disk, and terminal cables are available separately. Call for pricing.

Please include $10 ($7 less on item 4) for shipping and handling in the continental United States, $18 elsewhere. Connecticut residents please include 7.5 percent sales tax.

To receive a complete list of Ciarcia's Circuit Cellar project kits, circle 100 on the reader-service inquiry card at the back of the magazine.