and the program reassembled.

### Lookup Table

The lookup table contains one data byte per ASCII character. Each byte contains the five-bit Baudot character to be printed plus three control bits. As shown in Fig. 3, the least significant bit in the byte is 1, if the character is one of the special Baudot characters that will be enclosed in parentheses. If the second bit in the byte is 1, the Baudot character will be shifted to uppercase (FIGS). If the most significant bit in the byte is 1, a right parenthesis will be printed after the Baudot character. Several examples of typical characters are given in Fig. 3.

Minor keyboard variations from one printer to another may require you to alter your lookup table. Machines in common use include those having Telex, TWX, Weather, Military Standard and International Alphabet #2 character sets. Stay away from Weather machines since their character set contains many inapplicable uppercase weather symbols. The differences in the remaining four character sets are summarized in Table 5.

| Upper-Case | Telex | TWX | Military | International #2 |
|---|---|---|---|---|
| B | ? | 5/8 | ? | ? |
| C | : | WRU | : | : |
| D | WRU | $ | $ | WRU |
| F | $ | ¼ | ! | NA |
| H | # | NA | STOP | NA |
| J | BELL | , | ' | BELL |
| K | ( | ½ | ( | ( |
| L | ) | ¾ | ) | ) |
| S | ' | BELL | BELL | ' |
| Z | " | " | " | + |

*Table 5. Differences in common Baudot character sets.*

The lookup table in this driver assumes that a Military Standard printer is used. The printer, a model 28KSR, has been modified to provide a # character for uppercase H. It is particularly easy to substitute characters in the type basket of the model 28. Other characters that may be substituted include a * for uppercase H, and a + for uppercase G. Remember that the lookup table must be modified to reflect any changes that you may wish to make.

CRT displays allow use of the back-space function (ASCII control H), while most printers do not. Since it is desirable to tell when a back space has occurred, the character X is printed each time you back space. If, during an edit, you made a mistake and back spaced to correct it, the following would be printed:

MISTOKEXXXAKE

When listed a second time, the spelling would be correct.

All ASCII characters not covered by normal and special equivalents in the table will cause a space character to be printed. Note that lowercase alphabetic characters are not included in the table, although they can be easily added at the end of the table if necessary.

1. Load tape prepared using instructions from Table 3.
2. Note Stop Address.
3. Change contents of five memory locations using Table below.
4. Reenter Stop Address noted above.
5. Put new tape in record cassette machine.
6. Push "DUMP."
7. You now have a tape containing a program, the Editor, for example, which has a new High Memory point (to protect the ASCII/Baudot Driver), a new Pad, and five changes in the I/O Driver portion to provide jumps into the ASCII/Baudot Driver routines. Note that this has taken two load and dump procedures.

| Memory Address | Was | Change To | Comments |
|---|---|---|---|
| 040365A | 111O | 027O | ENTRY LO |
| 040366A | 040O | "BEGIN" HI* | ENTRY HI |
| 041040A | 303O | 315O | CALL |
| 041041A | 117O | 000O | INIT LO |
| 041042A | 040O | "BEGIN" HI* | INIT HI |

*Use high byte of BEGIN EQU from Table 3. For example, if your H8 has 16K of RAM, you would use 136O at memory locations 040366A and 041042A.

*Table 4. Modifying Heath I/O driver.*

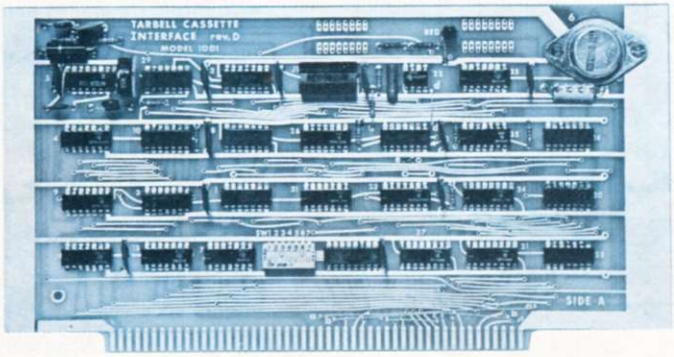

### Acknowledgements

Thanks to Dr. Glyn Harding for his helpful suggestions on program structure, and to Irv Hoff for his suggestions pertaining to the lookup table. ■

# TARBELL SETS STANDARDS

## For Hobbyists and Systems Developers

Sales to thousands of hobbyists over the past two years have proven the Tarbell Cassette Interface to be a microcomputer industry standard. Tarbell Electronics continues research and development to produce new and efficient components to fill hobbyists' changing needs.

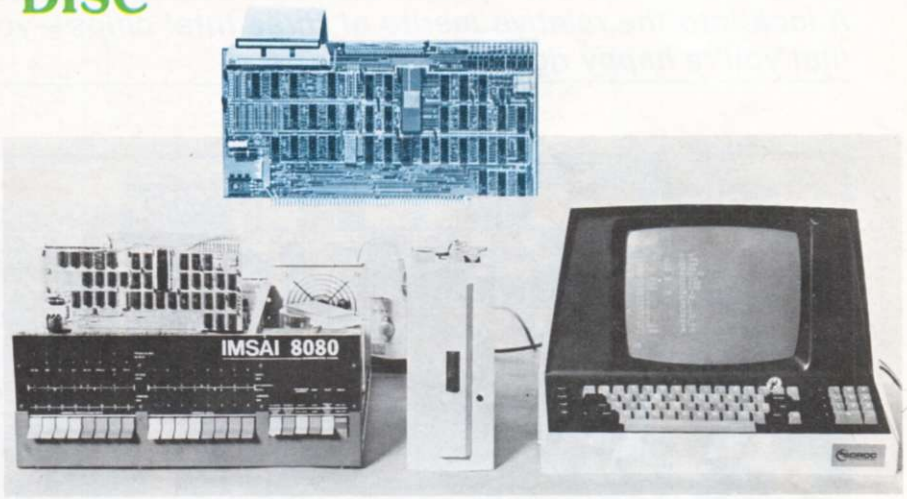



## TARBELL CASSETTE INTERFACE

- Plugs directly into your IMSAI or ALTAIR*
- Fastest transfer rate: 187 (standard) to 540 bytes/second
- Extremely Reliable—Phase encoded (self-clocking)
- 4 Extra Status Lines, 4 Extra Control Lines
- 37-page manual included
- Device Code Selectable by DIP-switch
- Capable of Generating Kansas City tapes also
- No modification required on audio cassette recorder
- Complete kit $120, Assembled $175, Manual $4
- Full 6 month warranty on kit and assembled units

## TARBELL FLOPPY DISC INTERFACE

- Plugs directly into your IMSAI or ALTAIR* and handles up to 4 standard single drives in daisy-chain.
- Operates at standard 250K bits per second on normal disc format capacity of 256K bytes.
- Works with modified CP/M* Operating System and BASIC-E Compiler.
- Hardware includes 4 extra IC slots, built-in phantom bootstrap and on-board crystal clock. Uses WD 1771 LSI Chip.
- Full 6-month warranty and extensive documentation.
- PRICE:
  Kit $190 . . . . . Assembled $265



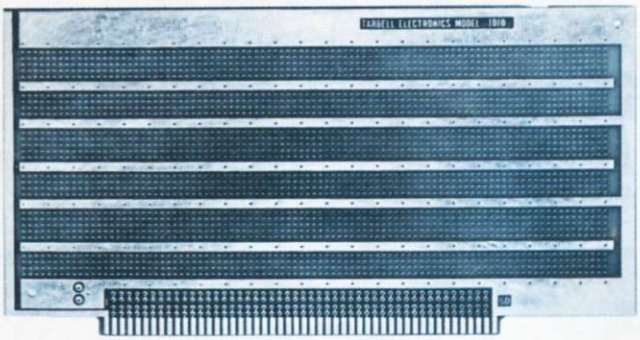

CP/M with BASIC-E and manuals: $100

**Compatible Disc Drives**

Ask about our disc drives priced as low as $525.

## TARBELL PROTOTYPE BOARD

**Model 1010**




- Gold plated edge pins
- Takes 33 14-pin ICs or
- Mix 40-pin, 18-pin, 16-pin and 14-pin ICs
- Location for 5 volt regulator
- Suitable for solder and wire wrap
- ALTAIR/IMSAI compatible

**Price:** $28.00

For fast, off the shelf delivery, all Tarbell Electronics products may be purchased from computer store dealers across the country. Or write Tarbell Electronics direct for complete information.

*ALTAIR is a trademark/tradename of MITS, Inc.
CP/M is a trademark/tradename of Digital Research

Tarbell Electronics

20620 South Leapwood Avenue, Suite P
Carson, California 90746
(213) 538-4251 T11

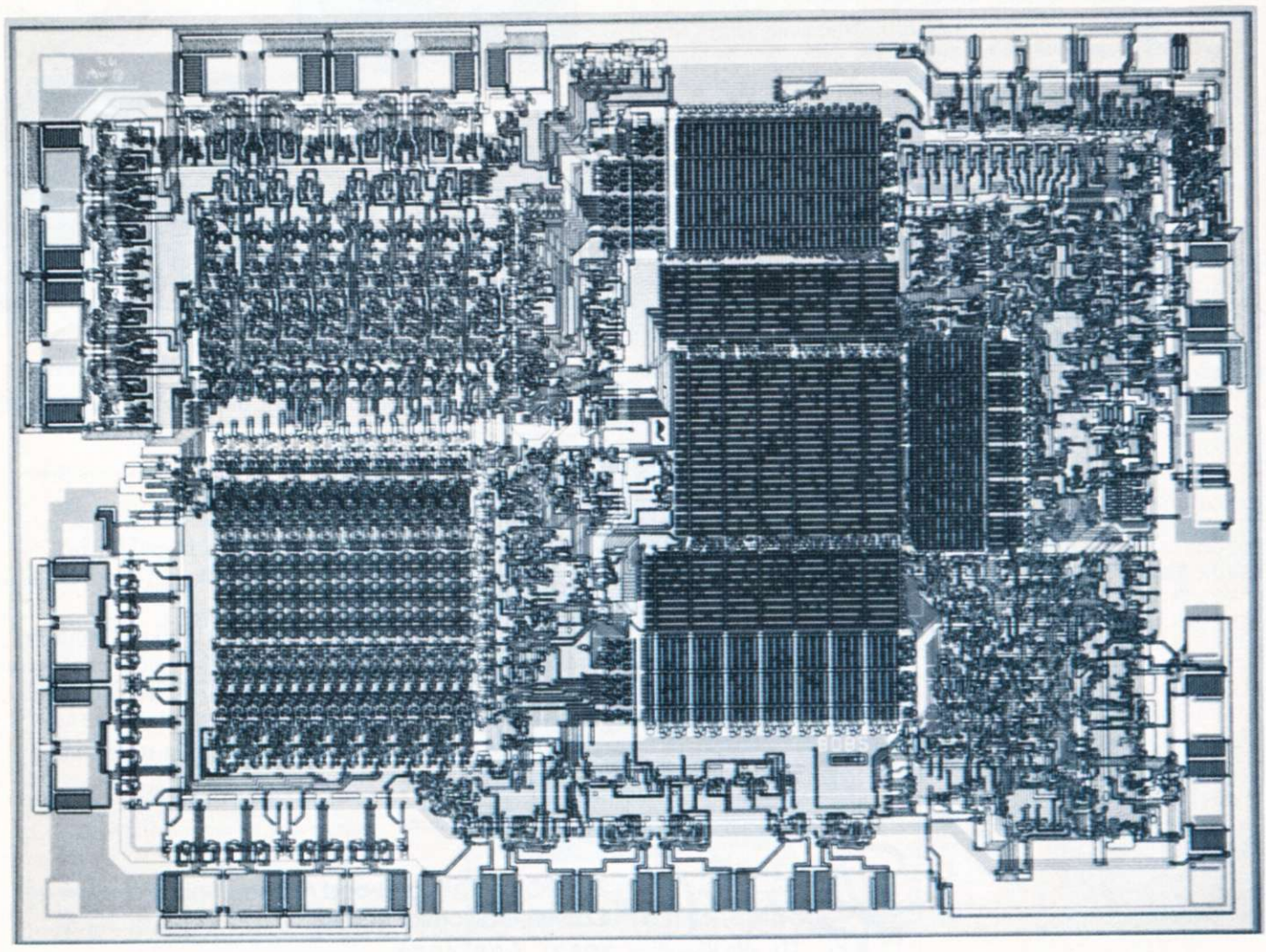
*Michael Slater*
*P.O. Box 1123*
*Menlo Park CA 94025*

*I like objective reviews of products. Michael Slater is a development engineer with Hewlett-Packard in Santa Clara CA (i.e., he isn't associated with Intel). He makes extensive use of Intel microprocessors in his work . . . and has some worthwhile comments and comparisons on their new 8085. See if you agree.—John.*

# 8080, Z-80 or 8085?

*A look into the relative merits of three Intel chips—you may find that you're happy right now.*

The 8085 is Intel's latest entry into the eight-bit microprocessor market. As an enhanced version of the 8080, the 8085 forms the basis of a microcomputer system that is both simpler and more powerful than the 8080. The introduction of the 8085 has made the 8080 obsolete and provided



*Photomicrograph of the 8085.*

some strong competition for Zilog's Z-80.

### Some History

Intel's 8080 microprocessor made the first hobbyist computer possible, and has since become the standard of the industry. Though numerous other processors have taken over a significant share of the market, there is still more hardware and software available for the 8080 than any other processor. In order to make it easier for people now using the 8080 to switch to a next-generation processor, Intel and Zilog have both introduced new processors that can use, without modification, most programs written for the 8080.

Zilog's Z-80 was the first processor to threaten the 8080's market dominance by including the entire 8080 instruction set as a subset of its own, much larger, instruction set. Therefore, most programs written for the 8080 will also run on the Z-80. In addition, the Z-80's sophisticated instruction set makes it attractive for software-intensive applications. Zilog also improved the 8080 hardware by providing on-chip clock generation, single supply operation, a faster clock rate and other features.

Now Intel has introduced a new processor, the 8085, which is also software compatible with the 8080. Intel has taken a different approach than Zilog, however, by making only negligible additions to the instruction set and placing emphasis on a simplified hardware system. Intel's goal was to simplify the entire system, including ROM and RAM.

With the 8085 system, it is possible, using only three ICs, to build a system with 256 bytes of RAM, 2K bytes of EPROM or ROM, 39 bits of I/O, a programmable timer and four vectored priority interrupts. No other microprocessor can provide this much capability with only three chips.

### The 8085 System

The three ICs that make this powerful system possible are the 8085 processor, the 8155 RAM, timer and I/O, and the 8755 PROM and I/O. Fig. 1 shows the complete minimal system. All three ICs are contained in 40-pin packages, which allows the integration of other functions with the ROM and RAM chips. The system has two unique features: a multiplexed address/data bus and two special chips that provide RAM, ROM, I/O and a timer.

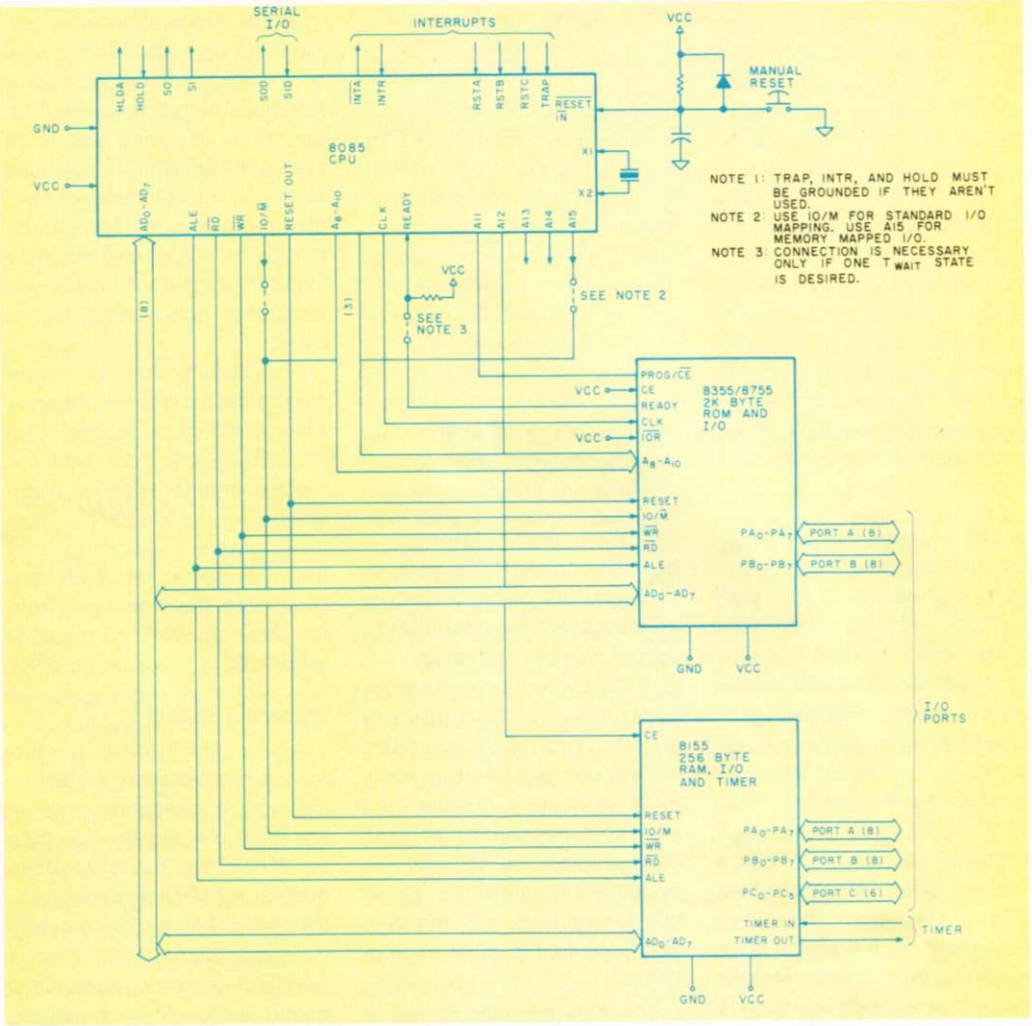



*Fig. 1. 8085 minimum system. The three-chip system consists of the 8085 processor, the 8755 EPROM and I/O and the 8155 RAM, I/O and timer. The system bus is similar to the 8080 except that the low-order byte of address and the data bus are multiplexed together.*

Fig. 2 summarizes the main differences between the 8080, the 8085 and the Z-80. The 8085 does not require a clock generator or system controller chip, as does the 8080. It provides three vectored interrupts without any external hardware. Other features include a single 5 V supply, serial I/O pins and a 3 MHz clock speed.

### The 8085 CPU

The major architectural change in the 8085 is the use of a multiplexed address/data bus. The data and the lower half

| | 8080 | 8085 | Z-80 |
|---|---|---|---|
| On-chip clock generator | no | yes | yes* |
| On-chip status latch | no | yes | yes |
| Vectored, priority interrupts without external hardware | no | yes;three | no |
| Nonmaskable interrupt | no | yes | yes |
| Single 5 V supply | no | yes | yes |
| Serial Input/Output on-chip | no | yes | no |
| Clock speed | 2 MHz | 3 MHz | 2.5 MHz |
| Special multifunction support chips | no | yes | no |
| Multiplexed bus | no | yes | no |
| Expanded instruction set | — | no | yes |
| Dynamic memory refresh | no | no | yes |

*Oscillator required

*Fig. 2. Comparison of the 8080, 8085 and Z-80.*

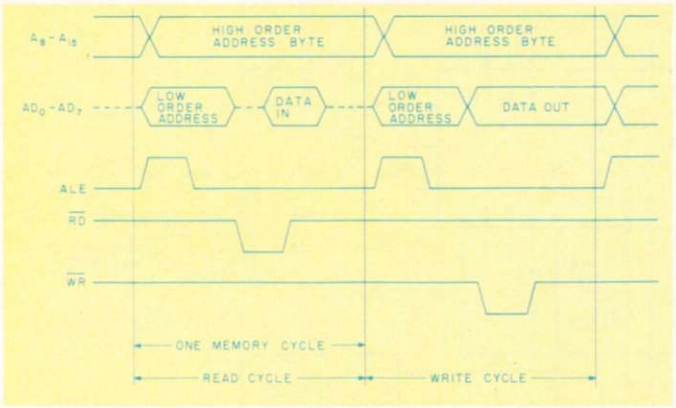



*Fig. 3. 8085 system timing. The trailing edge of ALE signals the rest of the system that the address/data bus contains a new address.*

of the address bus are multiplexed onto a single set of eight pins. At the beginning of each memory cycle, the low-order byte of address is placed on the address/data pins and a new control signal, Address Latch Enable (ALE), is generated to indicate that a new address is present on the address/data lines. ALE clocks this byte of address into a latch external to the processor. This latched address is then fed to the rest of the system, and the processor can free the address/data lines to carry data. The bus timing is shown in Fig. 3.

This multiplexed bus reduces by eight the number of address and data bus lines required. Only one new signal (ALE) is added to accomplish this, so the 8085 has seven more pins available for control signals. Two additional pins were freed by the elimination of the +12 V and −5 V supplies. These nine free pins make possible many of the 8085's new features.

The 8085 has four new interrupts that give it a simple and powerful interrupt structure—three priority interrupts, and one unmaskable (top priority) interrupt. Each is controlled directly by a single pin on the 8085; no other hardware is required to make the interrupts priority or mask them, or to insert an instruction on the data bus, as is required with the 8080.

The pins are called TRAP (nonmaskable) and RST 5.5, RST 6.5 and RST 7.5. Fig. 4 shows the 8085's interrupts and their jump addresses. A high level on the RST 5.5 pin, for example, will cause a jump to address 002C, provided that interrupt has been enabled.

In order to allow these interrupts to be individually masked, two new instructions were added. Set Interrupt Mask (SIM) selectively disables the interrupts. Read Interrupt Mask (RIM) allows you to read the status of the interrupts. These two instructions are the only additions to the 8080 instruction set.

The 8085 has pins for serial I/O to provide one input bit and one output bit without any peripheral hardware. The RIM and SIM instructions are also used to read and set the serial I/O pins.

The 8085 provides all of the control signals directly at its pins. With the 8080, an 8228 is required to latch the control signals off the data bus, on which they were multiplexed.

The 8085 control bus is also organized differently. The main control signals are $\overline{RD}$, $\overline{WR}$ and IO/$\overline{M}$. IO/$\overline{M}$ indicates whether the instruction is addressing an I/O port or memory. This saves one control line as compared to $\overline{MEMRD}$, $\overline{MEMWR}$, $\overline{IORD}$ and $\overline{IOWR}$ in the 8080. Both sets of signals contain the same information, and one can be generated from the other using a few gates.

The 8085 has pins for Reset In, Reset Out, Ready, HOLD, HLDA, INTR and $\overline{INTA}$, which function identically to those on the 8080. There are two new status lines (S1 and S0), which indicate whether the current cycle is an op-code fetch, a read or a write, or if the processor has been halted by a HALT instruction. The read and write status information is not used to control bus transfers, but provides advanced timing information on the type of machine cycle in progress. In most systems this information is not used.

The 8085's final hardware simplification is the addition of a complete clock generator on the CPU chip. The only required clock components are a crystal or a resistor and capacitor. This incorporates the functions of the 8224 within the 8085, and completes the transition from the three-chip CPU of the 8080 to the 8085, a true one-chip CPU.

### Processor Speed

As an additional performance improvement, the 8085's instruction execution rate is about 50 percent greater than the 8080's. The basic clock frequency is 3 MHz, as opposed to the 8080's 2 MHz. (The recently released 8085A-2 runs at *5 MHz*!) In addition, there is a subtle increase in speed by clever handling of conditional jumps. The 8085 fetches the second byte of the instruction (the first byte of the jump address) while it is deciding if the jump condition is satisfied, and then skips the last byte of the instruction (the second byte of the address) if it doesn't need to make the jump. The 8080 always fetches all three bytes and wastes a machine cycle if the jump condition is not satisfied. Thus, the 8085 executes an unsatisfied conditional jump in approximately 30 percent fewer clock cycles than the 8080.

There are two new versions of the 8085 now available. The 8085 is being replaced by the 8085A, which is identical in most respects. The new A version differs only in the state of certain control signals during specific conditions, which are

| Interrupt | Jump Address (HEX) |
|---|---|
| RST 0 | 0000 |
| RST 1 | 0008 |
| RST 2 | 0010 |
| RST 3 | 0018 |
| RST 4 | 0020 |
| *TRAP | 0024 |
| RST 5 | 0028 |
| *RST 5.5 | 002C |
| RST 6 | 0030 |
| *RST 6.5 | 0034 |
| RST 7 | 0038 |
| *RST 7.5 | 003C |

*Fig. 4. 8085 interrupts. The interrupts marked with * are new to the 8085, and each has its own interrupt request pin. The others are the same as the 8080 interrupts and are initiated using INTR and $\overline{INTA}$.*

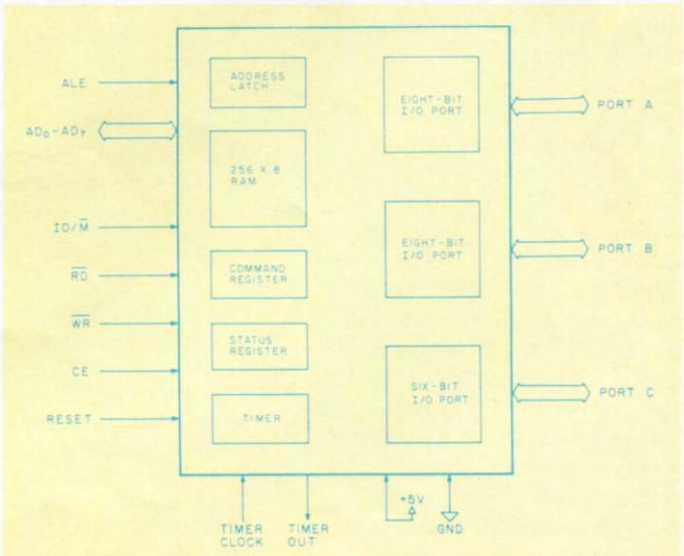



*Fig. 5. 8155 RAM, timer and I/O. Each of the three ports may be selected as input or output. Additionally, port C may be selected as a control/status port. The selection of port modes, as well as the control of the timer, is performed by writing the appropriate control word to the command register.*

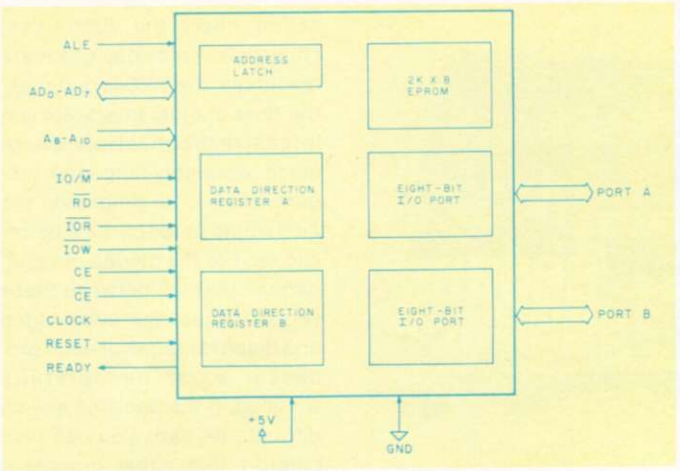



*Fig. 6. 8755 EPROM and I/O. Each bit of the I/O ports may be individually selected as input or output via the two data direction registers. The 8355 is an identical part except that it contains a mask-programmed ROM.*

"don't cares" for most users. The changes will make the 8085A easier to use in some large systems. The new 8085A-2 is the fastest general-purpose, eight-bit processor now available.

### The 8085 Memory and Interface Chips

As shown in the previous section, the 8085 added many hardware features at the price of the added complexity of a multiplexed address/data bus. The penalty is not too high, however, since the bus can be de-multiplexed with a single eight-bit latch. And, in fact, Intel has turned the multiplexed bus into a valuable feature by providing memory and peripheral chips with the address de-multiplexing latch on the chip. Thus, the chips have only eight address/data pins and an ALE pin, instead of eight data and eight address pins. This saves seven pins on each chip, as well as seven bus lines.

The 8155 chip contains a 256-byte static RAM, two eight-bit I/O ports, a six-bit I/O port and a 14-bit programmable counter/timer. The block diagram is shown in Fig. 5. A command register in the 8155 determines the function of the ports and controls the timer. The two eight-bit I/O ports may each be specified as input or output ports. The six-bit port may operate as an input, output or control port. In the control-port mode, it provides a strobe input for either of the eight-bit ports to be used as latched input ports. It also provides a buffer full signal for each of the ports, indicating that the port contains new data that has not yet been read.

The 8155 also contains a 14-bit programmable timer. It has several modes of operation, but, basically, it will count a specified number of clock pulses and then generate an output pulse. The clock may be the system clock or any other signal. Both the mode and the number of counts are under software control via a command register, and the timer's status may be read from a status register on the chip.

The third chip in the system is the 8755, which contains a 2K X 8 EPROM and two eight-bit I/O ports. The block diagram appears in Fig. 6. The EPROM is similar to a 2716; the two I/O ports are unique in that each line may be individually specified as input or output. Thus, you may have five input bits and eleven output bits, for example. The mode of each pin is set by software, via two data direction registers on the chip.

The 8355 is identical to the 8755, except that it contains a mask-programmed ROM. For applications where the same program is to be reproduced in considerable volume, the 8355 provides a cost-effective alternative to the 8755.

### Applying the 8085

The 8085 is a very easy microprocessor to use. Using the basic three-chip system already discussed, the only additional components required to make a complete microcomputer system are a crystal and power-up reset circuitry. Peripherals are, of course, required, but they are independent of the particular processor being used.

Standard memory and peripheral chips may also be used with the 8085, but the address/data bus must be de-multiplexed. Fig. 7 shows the address latch to perform this function, in addition to the gating to generate 8080-type control signals. ALE strobes the address into the latch (see timing diagram in Fig. 4). The latched outputs drive the lower half of the address bus. The data bus actually contains address information early in each memory cycle. This doesn't cause any problems because neither $\overline{RD}$ nor $\overline{WR}$ will be true at this time, and all parts of the system should be ignoring the data bus. Once the bus has been de-multiplexed, any of the 8080 family components may be easily interfaced.

The control signal gating shown is straightforward. In practice, it will usually not be necessary to use actual gates. If the memory chips have more than one chip enable input (many do), one can be used for IO/$\overline{M}$ and another for $\overline{RD}$ or $\overline{WR}$ (for RAMs) or just $\overline{RD}$ (for ROMs). When the 8155, 8355 or 8755 are used, no gating of any kind is needed; it is incorporated into the memory chip.

An interesting side note is that the signals have been arranged on the pins of the three chips to facilitate simple and dense PC layouts. A complete system consisting of the three chips, crystal, pull-up resistor for the Ready line, and resistor, capacitor, and diode for power-up reset, fits on a circuit board measuring 3 1/4 x 4 5/8 inches. The layout for this board is provided in Intel's *8085 Microcomputer Systems User's Manual.*

### Current Products

There are currently at least two 8085-based products suitable for personal-computing applications. Intel produces the SDK-85, which is a single-board computer complete with keyboard (hexadecimal and control) and a six-digit display. An 8355 ROM chip contains a monitor program to allow entering of programs, viewing the contents of memory and numerous other features. An 8155 provides 256 bytes of RAM. All of the I/O lines are available, and a large wire-wrap area is provided for expansion. A complete kit (less power supply) is available from any of Intel's dis-

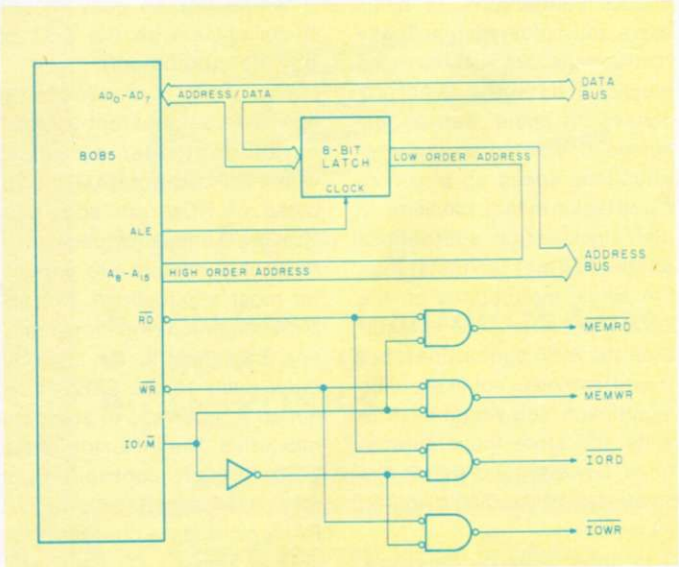
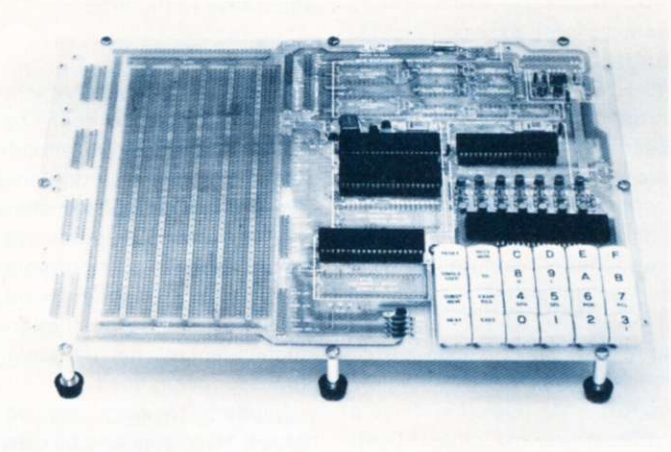



*Fig. 7. Interfacing the 8085 to an 8080-type bus. The latch de-multiplexes the address/data bus, and the gates generate the usual bus control signals.*

tributors for 250 dollars.

The other product is the SPACE BYTE 8085 CPU. It is an S-100-compatible CPU board utilizing the 8085 and 8155, with three 2708s, instead of the 8755. The PROMs contain a system monitor. Two RS-232C serial ports, along with parallel I/O, are provided. The bus is demultiplexed, and S-100-compatible control signals are generated. It is available from the Space Byte Corporation (and computer stores) for $499.



*Intel's SDK-85 design kit.*

**The 8085 vs the Z-80**

Since the Z-80 and the 8085 are the only 8080-compatible processors now available, many users will be faced with the task of selecting one of the two. One thing is clear: There is no longer any reason to design a system using an 8080.

From a software viewpoint, the Z-80 is superior. It has a greatly expanded instruction set, and double the number of registers. There is some question, however, as to the value of this increased software power.

If you program in BASIC or other high-level languages, the Z-80's extra instructions will make no difference to you. If you have an interpreter optimized for the Z-80, your BASIC might run faster than on an 8085. The difference is likely to be insignificant to most users.

One of the Z-80's features is special instructions for block move and block search. These instructions are useful when blocks of data are being processed. It must not be assumed, however, that these instructions speed up block-oriented tasks. In fact, block-move and block-search subroutines for the 8085 may run faster than the single instructions on the Z-80! The 4 MHz Z-80A is faster than the 8085 subroutines, but the difference is not likely to be significant. You would have to write and store these subroutines, however, and that is the advantage of the Z-80's built-in instructions.

In the realm of hardware, both processors have their advantages. The 8085's new interrupts are much easier to use than any of those available on the Z-80. The scheme is less flexible than the Z-80's, but is more than adequate for the vast majority of applications and requires no external hardware. For those few applications that really do require a sophisticated interrupt structure, Intel's 8259 interrupt controller chip provides this capability. Another ease-of-use factor is the 8085's on-chip oscillator. The Z-80 requires an external clock oscillator, while the 8085 requires only a crystal.

A big advantage of the 8085 is the available support chips. A much simpler and more powerful minimal system is possible with the 8155 and 8755, as compared with the Z-80 system. Because of the multiplexed address/data bus, it would not be easy to use these chips with the Z-80 (or any other processor).

The power of the 8085 system is most apparent in dedicated controller systems, where 256 bytes of RAM and 2K bytes of ROM are adequate. The 39 I/O lines provided by the three-chip system are enough for most applications. For applications requiring more memory, only one chip (an eight-bit latch such as the 8212) is required to interface to standard memories. The inclusion of the 8155, though contributing a relatively small amount of RAM, provides a counter/timer and 22 flexible I/O lines with some associated control logic.

The Z-80 also has its advantages, but they are relatively minor. It does not multiplex any pins, which saves one chip when interfacing to standard memories. This is offset, however, by the lack of an on-chip clock oscillator.

A much-acclaimed Z-80 feature is the dynamic memory refresh incorporated into the CPU. It is not, however, without its problems. The refresh address is output during each op-code fetch cycle. Therefore, the refresh rate is directly tied to the instruction execution rate, which may have serious consequences. If the processor is single-stepped, either by slowing the clock or by using the Ready line, the refresh will no longer work. Reset will also halt the refresh, so if a manual reset push button is incorporated in the system, some form of one-shot must be used to protect the memory from being wiped out by the reset button. Finally, a lengthy DMA operation may precipitate loss of the memory contents.

A couple of other points about refresh are worth mentioning. The vast majority of systems use either static memories (where refresh is not needed) or dynamic memory cards with built-in refresh circuitry. Another consideration is that the Z-80 will usually refresh the memory at a much faster rate than necessary, resulting in a higher power consumption for the memory.

The choice of an 8085 or a Z-80 ultimately must depend upon the application. In a dedicated controller type of application where the 8085 three-chip system provides adequate resources, the 8085 is definitely the best choice. In a software-intensive application where much assembly-language programming is anticipated, the Z-80 would probably be a better choice. For the myriad of applications that fall between these two extremes, the advantages and disadvantages of each processor must be carefully weighed. If the decision seems difficult, perhaps you can take comfort that either processor would probably do the job well.

**Future Products**

As may be expected, the battle between Intel and Zilog will not stop here. As Zilog is planning the Z-8 and Z-8000, Intel is working on the 8086 and 8088. The 8086 will be a high-speed (8 MHz clock) 16-bit processor to be introduced early in 1978. It has been reported that the 8088, to be released some time later, will be a 16-bit processor internally, but will use an eight-bit external data bus. Eventually, the 8088 may replace the 8085 in many applications. These next-generation processors will retain basic software compatibility with the 8085 (via assembly-language translation programs) and will also have a similar bus structure.

The 8085 may never have the popularity enjoyed by its predecessor, but its impact on the market will definitely be felt. It is especially cost effective in dedicated controller applications, but it is worth close examination by anyone designing a new system.■

**References**


1. *8085 Microcomputer Systems User's Manual*, Intel Corporation, Publication no. 98-366B, $5.

2. *8080/8085 Assembly-Language Programming Manual*, Intel Corporation, Publication no. 9800301B, $5.

3. *An Introduction to Microcomputers*, Osborne and Associates, Vol. II, 2nd edition.


Intel publications are available from: The Intel Corporation, 3065 Bowers Ave., Santa Clara CA 95051.

Edward Copes
105C New Kent Dr.
West Chester PA 19380

# One Keyboard: Hex and ASCII

*Here's a simple converter circuit that allows a single keyboard to function two ways.*

Recently, when I upgraded my home-brew 1802-based system from a hex keyboard to a full ASCII keyboard, I was confronted with a problem. Once the operating system was loaded, the system could take the ASCII data and convert it to hexadecimal machine-language instructions via software. But to load the operating system I still needed a keyboard that produced hex directly.

Inserting this simple two-chip circuit between the ASCII keyboard and the input port solved the problem. The circuit takes ASCII data from the keyboard and converts the characters 0-9, A-F to their proper four-bit hex format. Now my one ASCII keyboard can function either as an ASCII or hex keyboard, depending on system requirements.■

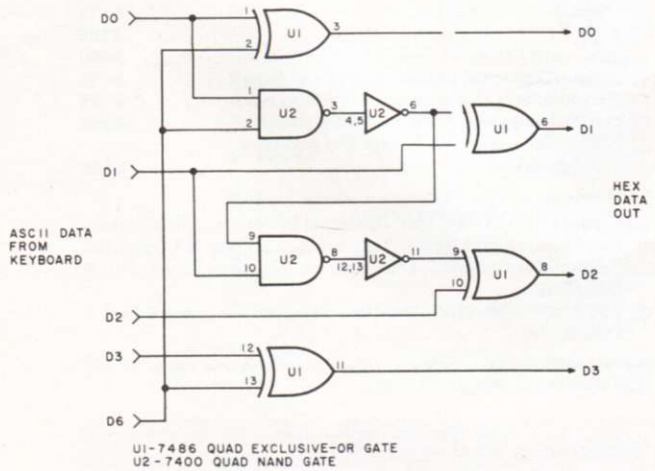



*Fig.1. ASCII-to-hex data converter.*

# THE TOTAL COMPUTER SYSTEM


With RCC's REX you don't need to buy CPU boards and memory boards and parallel I/O boards and video display boards, and motherboards and serial I/O boards and disk interface boards and cabinets and chassis and keyboards and stuff!

REX is the complete small computer, designed for the user who wants to buy a system with enough power to run FORTRAN . . . or Disk BASIC . . . or Z80 Assembler .

The REX system costs **$2495**. Sure that's more than a kit, but look what you get!

| | | |
|---|---|---|
| Z80A CPU (4 MHz) | ☑ standard | |
| 24K of RAM | ☑ standard/ | expandable to 64K right on the motherboard |
| S-100 Bus | ☑ standard/ | up to 5 slots available in basic cabinet. . .expansion cabinet available soon |
| Microfloppy drive | ☑ one standard, one optional/ | record on both sides of the diskette; 75K bytes per surface |
| Bootstrap monitor | ☑ standard/ | in a 2708 PROM. No switches; no buttons |
| Powerfail, 60 Hz timebase & vectored interrupts | ☑ standard | |
| Black & white video interface | ☑ standard/ | 128 character set, 24 line, 80 char/line, full software video control |
| DMA channels | ☑ standard/ | control lines on the bus |
| Microfloppy interface | ☑ standard/ | 8 bit parallel I/O port (optional) |
| ASR-33 type keyboard | ☑ standard/ | optional typewriter keyboard with numeric 10 key pad |
| Walnut side-panel cabinet | ☑ standard | |
| 15 Amp power supply | ☑ standard | |

Your REX system is shipped assembled, tested and burned-in. You just add a video monitor (or an RF modulator and your home TV) and select one of our optional software packages ($195) and you are computing!

Accounting programs, analysis, games . . . no problem! Your basic REX system can handle them. Of course, we offer expansion capabilities too. Options like:

- ☐ Second microfloppy drive . . . . . **$450**
- ☐ Programmable CRT character generator . . . . . **$100**
- ☐ 12 inch CRT monitor (green phosphor, matching REX wood-side cabinet) . . . . . **$215**
- ☐ 8K x 8 RAM expansion . . . . . **$195**
- ☐ 16K x 8 RAM expansion . . . . . **$400**
- ☐ Serial I/O channel (installed on initial order). . . . . **$ 35**
- ☐ RF modulator (for connecting REX to home TV) . . . **$ 35**
- ☐ Dot matrix printer (120 cps,80/96 column) . . . . . **$850**
- ☐ Expansion cabinet for printer and two large floppy drives . . . . . **$145**

The software is super! Take your choice for **$195**!

- ☐ ANSI FORTRAN IV, Disk Operating System, File Management System, Linking Loader, Text Editor, Utilities & Diagnostics
- ☐ Extended Disk BASIC, Disk Operating System, Text Editor & Diagnostics
- ☐ Z80 Macro Assembler, Disk Operating System, Text Editor & Diagnostics

See the REX system today at your local computer store. Or call RCC toll free at (800) 553-1863.

**realistic controls corporation**

404 West 35th Street, Davenport, IA 52806
Toll Free (800) 553-1863 In Iowa (319) 386-4400

## See the Rex at these dealers

**ARIZONA**
- Tempe — Computerworld, Inc. (602) 944-6070
- Tempe — Phoenix Group, Inc. (602) 894-1129

**CALIFORNIA**
- Escondido — Business Enhancement Compuservice (714) 741-6335
- Huntington Beach — Cybernetics, Inc. (714) 848-1922
- Lawndale — Jade Computer Products (213) 679-3313
- San Diego — Byte Shop (714) 565-8008
- San Francisco — Computer Store of San Francisco (415) 431-0604
- San Jose — Byte Shop III (408) 377-4685
- San Rafael — Byte Shop—Computer Store (415) 457-9311
- Santa Clara — Byte Shop—Computer Store (408) 249-4221
- Sherman Oaks — Peoples Computer Shop (213) 789-7514
- Tarzana — Tech-Mart (213) 344-0153

**COLORADO**
- Englewood — Byte Shop, Inc. (303) 761-6232

**IOWA**
- Davenport — Computer Store of Davenport (319) 386-3330

**KANSAS/MISSOURI**
- Overland Park — Personal Computer Center, Inc. (913) 649-5942

**MARYLAND**
- Baltimore — Computer Workshop of Baltimore, Inc. (301) 486-5350
- Rockville — Computer Workshop, Inc. (301) 468-0455

**MICHIGAN**
- Royal Oak — Computer Mart of Royal Oak (313) 576-0900

**MINNESOTA**
- Minneapolis — Computer Depot (612) 927-5601

**MISSOURI**
- Parkville — Computer Workshop of Kansas City, Inc. (816) 741-5055

**NEBRASKA**
- Lincoln — Microtech, Inc. (402) 423-8404
- Omaha — Omaha Computer Store (402) 592-3590

**NEW YORK**
- New York — Computer Mart of New York (212) 686-7923

**OHIO**
- Lyons — Affordable Computers (419) 923-2821

**OREGON**
- Beaverton — Byte Shop (503) 644-2686
- Portland — Byte Shop (503) 223-3496

**PENNSYLVANIA**
- Pittsburgh — Computer Workshop of Pittsburgh, Inc. (412) 327-0455

**TEXAS**
- Dallas — Computer Shops, Inc. (214) 234-3412
- Dallas — Rondure Company (214) 630-4621
- Houston — Interactive Computers (713) 772-5257
- San Antonio — The Computer Shop (512) 828-0553
- San Antonio — Computer Solutions (512) 828-1455

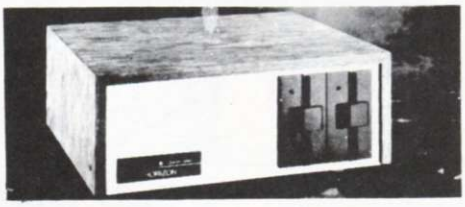
**VIRGINIA**
- Springfield — Computer Workshop of Northern Virginia, Inc. (703) 321-9047

**WASHINGTON**
- Bellevue — Byte Shop (206) 746-0651
- Bellevue — The Microcomputer Center (206) 455-3710

**WISCONSIN**
- Janesville — Austin Computers (608) 752-6002

FCC™ realistic controls corporation

## Look To The Horizon.™ The North Star HORIZON Computer.



**HORIZON ™**

To begin programming in extended BASIC, merely add a CRT or hard-copy terminal. HORIZON-1 includes a Z80A processor, 16K RAM, minifloppy TM disk and 12-slot S-100 motherboard with serial terminal interface—all standard equipment.

## And, Look To Computer Enterprises For The Lowest Prices & Fastest Delivery On North Star Horizon Computer System:

| | Credit Card Price | Cash Discount Price |
|---|---|---|
| Horizon 1 System (one drive) kit | $1497 | $1439 |
| Horizon 1 System (one drive) assembled | $1777 | $1709 |
| Horizon 2 System (two drives) kit | $1871 | $1799 |
| Horizon 2 System (two drives) assembled | $2199 | $2114 |

### ALSO: Run TDL Software On Your Horizon...

| | | |
|---|---|---|
| TDL Package A with Super Basic | $ 228 | $ 219 |

### For Your Serial Terminal:

| | | |
|---|---|---|
| Lear Siegler ADM-3A (kit) | $ 728 | $ 700 |
| Lear Siegler ADM-3A (assembled) | $ 858 | $ 825 |

### Call Or Write Today For Computer Enterprises' FREE Spring Computerlogue!

| | | |
|---|---|---|
| IMSAI PCS-80/30 kit | $1097 | $1055 |
| IMSAI PCS-80/15 kit | $ 748 | $ 719 |
| IMSAI VIO-C kit | $ 297 | $ 286 |
| TDL ZPU Board (assembled) | $ 183 | $ 176 |
| TDL XITAN ALPHA 1.5 kit | $ 812 | $ 781 |
| Seals 8k 250ns RAM (assembled) | $ 203 | $ 195 |
| Dynabyte 16k 250ns Static RAM (assembled) | $ 577 | $ 555 |
| Cromemco Z2D kit | $1399 | $1345 |
| Cromemco Bytesaver less Proms kit | $ 136 | $ 131 |

Shipping charges: $10 per CPU on larger units; $1.50 per kit. $2.00 min. per order.
Delivery is stock to 30 days on most items. Shipment is immediate for payment by cashier's check, money order or charge card. Allow 3 weeks for personal checks to clear. N.Y. State residents add approp. sales tax. Availability, prices and specs may change without notice.

computer enterprises™

Operating Hours:
M-W 10-5 E.S.T.
TH-F 10-9 E.S.T.
Closed Sat. & Sun.

P. O. Box 71 Fayetteville, N.Y. 13066
**Phone (315) 637-6208** Today!

*Dan Dederick*
*4104 Woodhollow Dr.*
*Conroe TX 77301*

# Is the Malibu Model 160 the Printer for Your Business System?

*At $200, this may not be a popular hobby printer, but for business, where a 132-column printer is a must, it's most interesting.*

One of the most important pieces of peripheral equipment for any business system is the printer. During my search for a suitable printer for my system, I was confronted with trade-offs and considerations, but two items remained at the top of the list—dependable operation and speed. I feel that I've obtained both through the purchase of a Malibu printer and would like to share with you some of the fine points, and a few shortcomings, of the unit.

I am neither a technician nor an engineer; I am a businessman. Even though I had some technical assistance in the preparation of this article, the bulk of the comments here reflect my feelings on how the unit contributes toward making my system the useful tool it is. John Craig suggested that I direct my remarks to other businessmen who might also be shopping for a hard-copy device.

### General Overview

As mentioned earlier, dependable operation and speed were two of the main considerations, but the added bonuses of quiet operation and styling were also realized with the Malibu unit. Photo 1 shows the styling, which, quite frankly, played a part in my decision (moving that teak cabinet into my office was almost like adding a fine piece of furniture). The clear plastic cover over the print mechanism, which almost completely seals the entire unit, contributes greatly toward noise reduction. There is, of course, a slot for the paper to come out of the top and enter from the bottom, but the noise escaping from these two openings is minimal and not objectionable. I am sure the soundproofing within the cabinet contributes a lot toward this end.

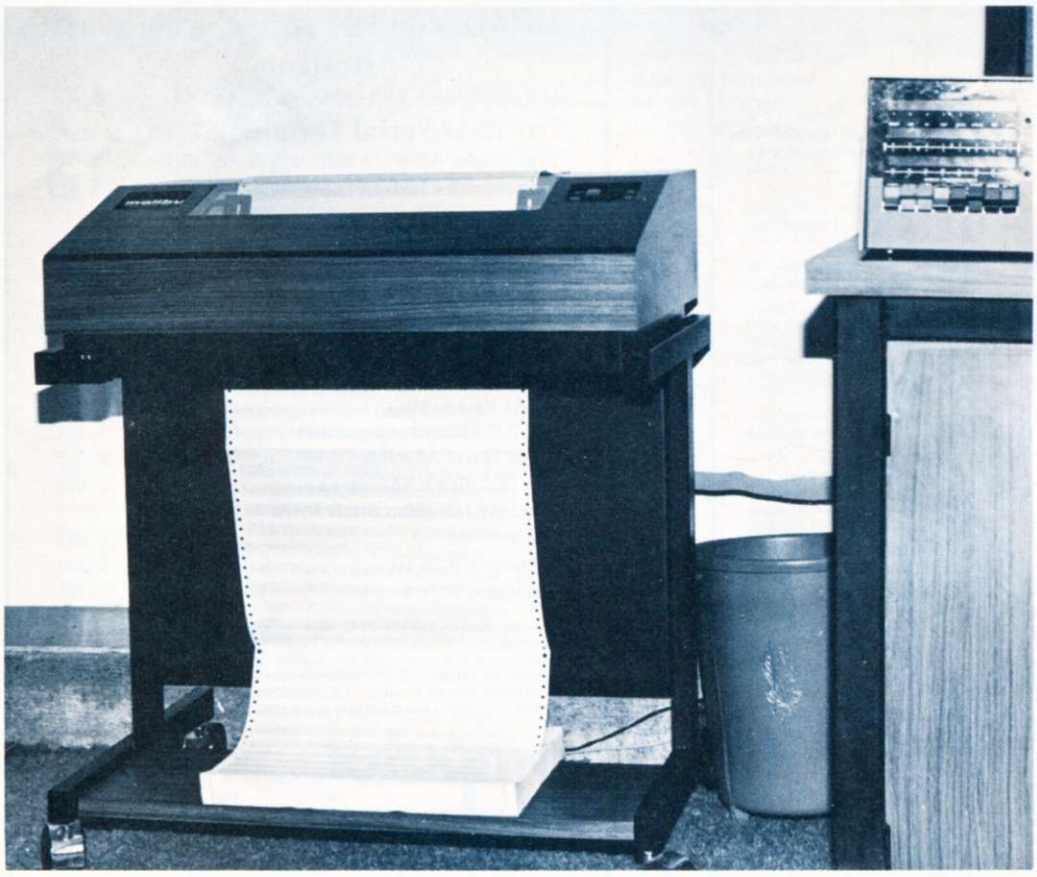


*Photo 1. The Malibu Design Group Model 160 Printer: a good-looking unit.*

I've been using my disk-based Imsai system for almost a year now and there have been several instances when I've had to go through the trauma of adding (interfacing) additional devices that hardly ever worked—at first, anyway—even though the additions were installed by experienced technicians from the store where I bought my system. Adding the hardware components of this printer to my system went smoothly. The entire operation took no more than 15 minutes and simply involved removing the cover from the computer, plugging in the S-100 interface board, running a cable over to the printer, plugging it in and turning everything on! (More on the software interface later.)

Prior to the Malibu printer, I had been operating with an ASR-33 Teletype. The difference in speed, print quality and noise level between the two is so vast that I'm only mentioning the Teletype as an after-

*Photo 2. Different-sized paper and forms. A big plus for the Malibu.*

thought. Comparing the dependability of the two is difficult at this time since I've only been operating the Malibu for a few weeks. Due to the small number of moving parts, I'm confident the Malibu will come out shining in this area.

Printing is done at an incredible 165 characters per second! (Memories of the Teletype surely fan my enthusiasm on this point.) Much of the speed results from the unique combination of hardware and software which allows printing in both directions and determines the shortest distance the printhead has to go to begin printing. This feature is called bidirectional printing.

For example, if a line of 120 characters has just finished printing, and the next line to be printed is 80 characters long, a line feed will be initiated and the 80-character line will be printed backwards—while the printhead returns to the leftmost (home) position. In many other printers a line feed *and head return* will take place before the second line is printed—resulting in a lot of wasted time.

The print quality and character set were two important considerations in my choice of the Malibu. Several of the printers I looked over during the shopping stage produced very light printouts in comparison. One of the reasons the Malibu provides such dark, high-contrast printing is due to the self-inking ribbon used. Another factor is the Hydra printhead (more on that later).

One of the features I disliked most on other printers I looked at was the absence of descenders (the "tails" of certain lowercase letters actually *descending*). The Malibu character set provides for these descenders —as all the others should. Fig. 1 is a sample printout.

I'm not particularly thrilled that the unit is a dot matrix, rather than impact printer, but my wallet is happier that I didn't go with the latter.

I've asked several of my customers what they think of the new invoices they have been getting. Generally, the response is that, though they prefer not to get invoices at all, the new ones are quite an improvement over the old (generated by the Teletype). When I press them for an opinion on the actual characters—and how they are generated (with dots)—the usual response is, "Gosh, I hadn't noticed." So much for the shortcomings of dot-matrix print.

### Miscellaneous Considerations

I dislike reading equipment reports that contain numerous dry specifications. But there are several features of this printer that really should be mentioned.

The maximum line length is 132 characters, one of the considerations that made me decide on the printer. Several low-cost printers provide 72-character width, but I find that some monthly and quarterly reports practically demand the wider paper. Invoices are prepared on 8½- × 11-inch paper, and the mailing labels I generate for customer mailings are approximately 4½ inches wide. The variable width tractors are a

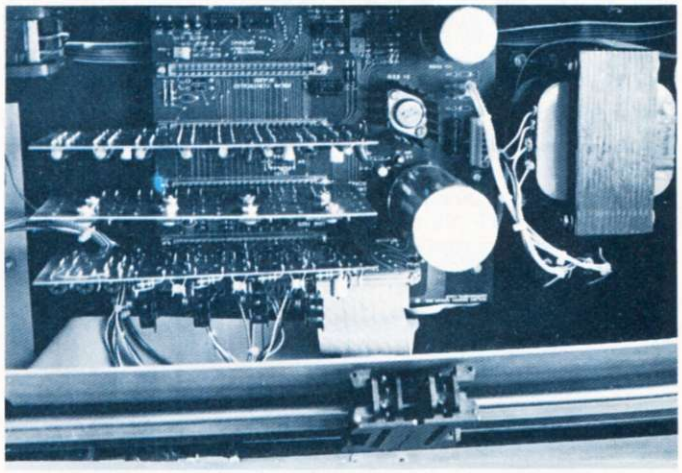


*Photo 3. Inside the Malibu, the three main printed-circuit boards.*

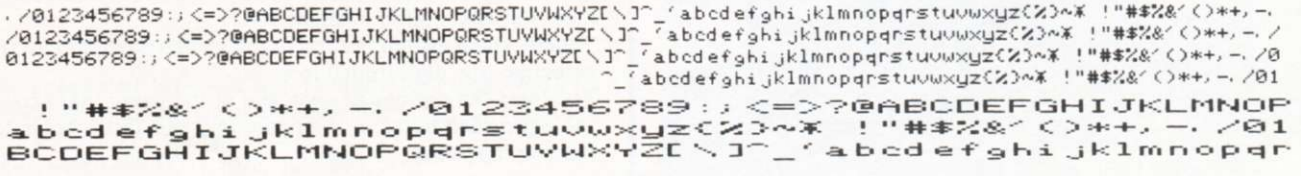



*Fig. 1. Sample printout showing normal and expanded characters. Also note the lowercase characters with descenders.*

definite asset. Photo 2 illustrates this capability.

Although I intend to have my computer serviced at a local store, it seems that I could almost make the repairs myself. That is, if I weren't afraid to take the cover off the unit and dig inside, and if I had spare printed-circuit boards.

There are only three boards in the printer to control all the operations (see Photo 3). The first is the logic card that interfaces with the software. (Incidentally, all of the character generation is done in software; therefore, the character set can be easily changed.) The second board controls the motor that feeds the paper and also controls the movement of the printhead. (Software makes the decisions regarding the direction it should take for the fastest print time.) The third board drives the solenoids on the printhead.

There is a fourth interface board, which contains a Z-80 microprocessor. This board is used for connecting the printer to systems other than the S-100 type and provides for two types of interfacing—RS-232 or 8-bit parallel (at speeds from 110 to 9600 baud). My Imsai, quite naturally, uses the S-100 interface board. This board contains the programs, in programmable read-only memory (PROM), for generating the particular characters, handling the line of characters and telling the paper-feed motor when and how far to advance.

A thumb-wheel switch in the lower left-hand corner of the printer allows the operator to select the number of spaces the paper will be advanced every time there is a FORM FEED command. This is a command from the computer which tells the printer to advance to the next form (e.g., the next invoice to be printed). The switch provides for settings from 1 line to 99 lines.

Photo 4 shows the manual paper-control switches on the cover of the unit. When the FORM FEED switch in the lower right-hand corner is depressed, the paper is advanced to the next page (the number of spaces is determined by the thumb-wheel switches).

Using this feature, the paper can be positioned to a spot called "top-of-form" so that printing will begin in the same position for each page. The LINE FEED switch will cause the paper to be advanced one space for each depression. The other two switches provide for incremental up or down positioning of the paper. No big knobs to turn—just push a button.

There is a particular feature about the ribbon which, although it is small, reflects the thoughtfulness that went into the overall design of this printer. The ribbon is set up to traverse from one side to the other *at an angle*. The result is that the printhead is continually using all of the ribbon, rather than just the top or bottom. Since ribbons are such a mess to change, I'm relieved that this feature will surely extend the life of the ribbon.

The Hydra printhead, shown in Photo 5, is, I've been told, one

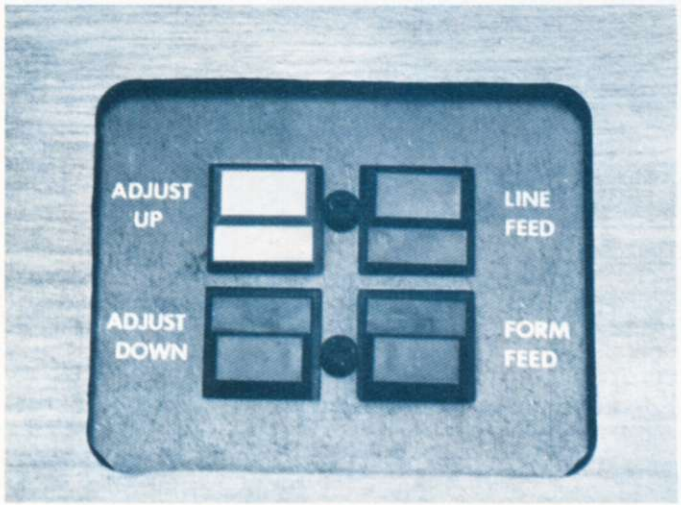



*Photo 4. The manual paper-control switches.*

**Glossary**

**8-bit parallel interface:** Transferring data from the computer to a device one character at a time, rather than sending each bit of the character at a slower serial rate.
**Bidirectional:** The ability of a printer to print characters while the printhead is traveling from left to right or from right to left.
**Baud:** A data transmission term related to the number of bits (BInary digiTS) transmitted in a second.
**Character set:** The complement, list or table of characters a printer is capable of printing. (With a different PROM the Malibu is capable of generating foreign-language character sets or other specialized applications.)
**Descenders:** The "tail" on certain letters, such as the j, y, p, g and q.
**Dot matrix:** The generation of characters by a matrix of dots. Following is an example of how a capital A is generated using a 5 × 7 matrix:

```
  x
 x x
x   x
xxxxx
x   x
x   x
x   x
```

Each x represents a dot made by the printhead. Due to the close spacing of the dots, the letters seem to run together into continuous lines.
**Head return:** The action of the printhead being returned to the home position (i.e., at the extreme left, where each new line begins). The same as a carriage return on a typewriter.
**Impact printer:** A printer that prints characters in the same manner as a typewriter.
**Interface:** The interconnecting of two devices (e.g., the computer and the printer) so they can work together as part of a system.
**Line feed:** The action of advancing the paper to the next line to be printed.
**Line length:** The number of characters that can be printed on a line (132 with the Malibu).
**Patching:** Modifying the software with minor changes so that the operation of a particular device becomes an integral part of the system.
**Printhead:** An assembly containing seven solenoids activated by electrical impulses to produce the dots that generate the printer's characters.
**PROM (programmable read-only memory):** A memory element that cannot be modified and can only be read. (All of the mechanical movements and character generation functions are contained in PROM in the Malibu printer.)
**Routines:** Program segments that are not part of the main program.
**RS-232:** An interfacing standard between computers and other devices that dictates the voltage levels at which data may be transmitted.
**S-100:** The "standard 100" pin bus that originated with the Mits Altair 8800 computer and has found widespread use in most microcomputer systems today.
**Thumb-wheel switch:** A rotary-type switch providing for the selection of a digit from 0 to 9.
**Tractors:** The devices at each end of the paper that accept the perforated holes in the paper and use those holes for feeding the paper.

of the most reliable and rugged available. Time will be the real test of this component, which is the heart of the printer, but so far I've been impressed with the quality of print it delivers and its rugged constitution. Since the entire unit radiates quality, I have no reason to doubt the claims made about the printhead.

### Software Modifications

When you add a device such as this to your system there are actually two types of interfacing to contend with—the hardware and the software.

Making the printer work with my existing software involved modifying the programs with patches so the printout would be on the Malibu, rather than my Teletype. An experienced programmer took care of these modifications in a three-to-four-hour period. He explained to me that the task could have been very difficult if I did not have well-designed software.

Basically, the modifications involved changing the programs (making patches) so when printing is desired, control is transferred to the routines in PROM (provided by Malibu) on the interface board that handle character generation, line handling and paper-feed functions.

### Closing Thoughts

The primary use for this printer in my business (restaurant supply) is for generating monthly reports, customer billing, mailing labels and some accounts payable. It handles all of these jobs without a hitch. Because it is a dot matrix printer I'm not certain that it would be as satisfactory if my requirements included large quantities of textual material. However, I intend to try it with some customer bulletins in the future. For such applications, an uppercase and lowercase impact printer may be a better choice.

I don't normally sit down to write reports on products I have bought, but I was very impressed with this printer. I'm still fascinated with the speed of the unit and appreciate the quality of the print.

I enjoy the finer things in life; I drive a 27-year-old Cadillac, and the quality built into that car will very likely keep it on the road for another 27 years. I feel the same way about the quality built into the Malibu printer.■

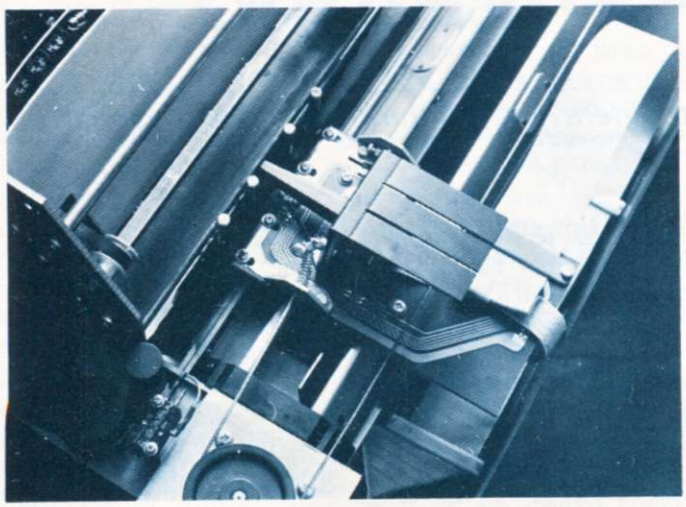
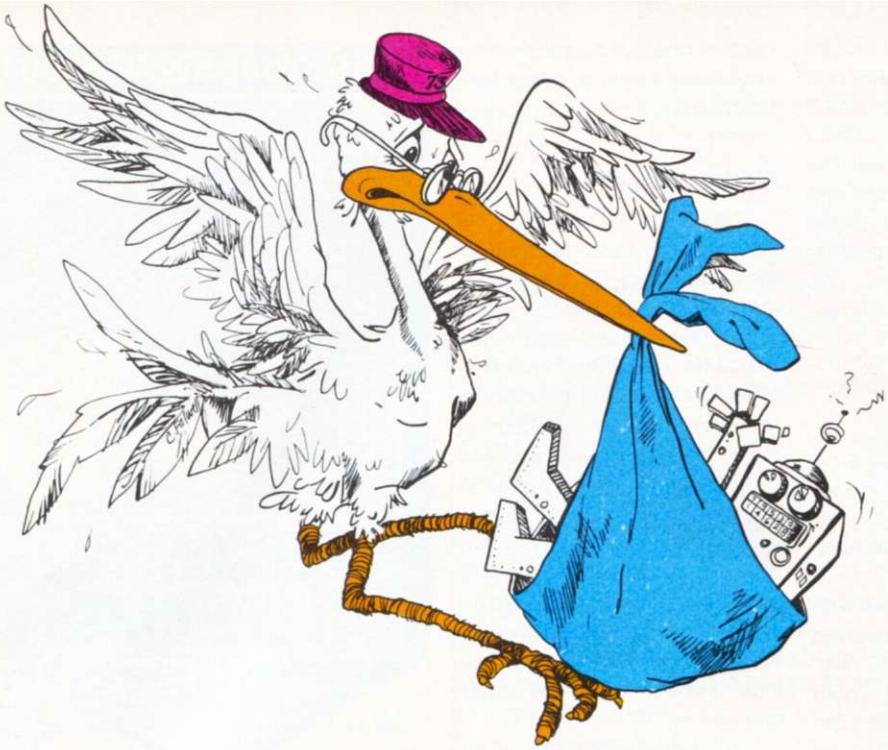


*Photo 5. The heart of the printer: a high-reliability Hydra printhead.*

*The suggested list price for the Model 160 Line Printer is $1995 (which is very close to $2000). For further information, contact: Malibu Design Group, Inc., 21110 G Nordoff St., Chatsworth CA 91311, (213) 998-7694.*



# The Great Computer Conspiracy

*You've heard that some people are afraid of computers? Maybe they have good reason to be.*

*Clint H. Woeltjen*
*La Habra CA*

I guess it was bound to happen sooner or later with the office computer having access to billing as well as a telephone link to the lab computer, the new gigabyte memory and the new, unproven heuristic-algorithm operating system. We should have guessed that it was up to something when the auditor's computer started finding discrepancies in our records. No mere human could possibly have circumvented the system.

It began one day last March. The computer ordered several hundred more TTL ICs than we actually needed for the project at hand. We could find no programming error, so assumed human error or a temporary hardware malfunction and waited for it to show up again. Meanwhile, the extra ICs never made it to the lab. A mobile manipulator (we call them robots for want of a better term) had been sent to shipping and receiving to pick them up, but had returned empty handed. The plant is almost entirely automated, requiring only a few engineers and supervisory personnel.

The shipping-and-receiving administrator remembered seeing the package earlier, but didn't actually see the robot pick it up. Of course, you can't question one of our robots; they have only enough memory for the task at hand, so they can't be expected to remember previous jobs. Nor did we think of questioning the computer; after all, a computer only does what it's programmed to do (anyway, what would a computer possibly want with a bunch of ICs?). We deduced that it was employee theft, but no one would admit to having done it. We beefed up our security system (controlled by the office computer), which caused employees' morale to drop, which caused their efficiency to drop, which caused us to fall behind on several key contracts, which caused us to

become overloaded with work, which caused us to discontinue the security system.

A week later, we received a receipt for a megabyte memory that no one could remember ordering, or paying for. We checked our records and found the order, but it hadn't been initialed properly, so we couldn't tell who had ordered it. Of course, things like that aren't supposed to happen. The office computer is supposed to check for proper initials before printing the check and mailing the order. As we were overworked at the time, however, no one noticed the computer's error. This time, we ran a thorough diagnostic program, but were unable to find the bug. We didn't realize then how easy it was for a sophisticated computer to lie, its own program supplying all of the proper responses while the diagnostic tested harmlessly in memory. As happened before, the megabyte of memory disappeared.

Then the circuit-board inspector got in trouble for not asking the computer to order additional circuit-board material. He claimed that there had been two extra boxes of pre-cut boards just yesterday; so we programmed a more extensive security program, including video input of the entire lab area. This time, we didn't let the employees in on the upgraded security plan, and to make sure that no one might let it slip, I did all of the programming myself. As usual, the entire system was controlled completely by the office computer. We used the office computer rather than the lab computer because the men in the lab didn't have access to the office computer and couldn't change the program if they found out. Of course, we didn't know that the office computer and the lab computer were friends and exchanged programs whenever they felt like it.

The next day, a considerable amount of microcomputer hardware was missing, including a video terminal, several UARTs and some obsolete Z-80 microprocessors. We checked the videotapes but found nothing; the camera set up to record that part of the lab had been unplugged and the videotape erased. The night watchman claimed that he hadn't seen anyone enter or leave the building all night, but after all, he's only human and can't be in two places at once.

We were beginning to suspect that the stolen material hadn't left the building. We also noted that all that was needed to build a complete microcomputer out of the stolen material was a power supply. I instructed the lab computer to build such a supply, and left it out on the lab bench before leaving for the night. That afternoon, I also removed the cameras and deleted the security program. I had the lab administrator punch out for me, but I stayed behind in the lab. I hid within view of the bench containing the power supply and waited.

Shortly after the last employee checked out, I saw one of the robots roll across the floor, remove the power supply and head toward the lab storage area. Meanwhile, another robot had moved some boxes and was removing a metal wall plate that was normally hidden by the relocated equipment. In our earlier search we had noted the panel, but had assumed that it was an electrical maintenance panel and ignored it. I watched as the first robot arrived with the power supply, installed it and turned it on. In the recess was a microcomputer, complete with a video terminal (which indicates to me at least that the robot didn't intend to keep it a secret forever), and on the display was printed the mnemonic "DAD DE."

I didn't have the heart to turn it off. ■


# Meet The North Star Family

**THE NORTH STAR S-100 FAMILY**—four high performance products at attractive low prices. Our boards are designed for use in the North Star HORIZON computer and other S-100 bus computers using 8080 or Z80 processors. Visit your computer store for a demonstration, or write for our free color catalog.

### 16K RAM BOARD

No other S-100 bus memory can match the performance of the North Star 16K RAM at any price. This low-power board has been designed to work at full speed (no wait states), even at 4MHz with both Z80 and 8080 systems. Memory refresh is invisible to the processor, bank switching is provided and addressability is switch selectable in two 8K sections. Best of all, a parity check option is available. Kit: $399. Assembled: $459. Parity Option — kit: $39. Assembled: $59.

### MICRO DISK SYSTEM

The North Star MDS is a complete floppy disk system with *all* hardware and software needed to add floppy disk memory and a powerful disk BASIC to S-100 bus computers. The North Star MDS is widely considered one of the best designed and most complete S-100 bus products available. The MDS includes the S-100 interface board with on-board PROM for system startup, Shugart mini-floppy disk drive, cabling and connectors, and DOS and BASIC software on diskette. Kit: $699. Assembled: $799. Additional drive — Kit: $400. Assembled: $450. Single Drive Cabinet: $39. Optional Power Supply: $39.

### Z80A PROCESSOR BOARD

The North Star ZPB brings the full speed, 4MHz Z80A microprocessor to the S-100 bus. Execution is more than twice the speed of an 8080, and the ZPB operates in systems both with and without front panels. The ZPB includes vectored interrupts, auto-jump startup, and space for 1K of on-board EPROM. Kit: $199. Assembled: $259. EPROM Option — kit: $49. Assembled: $69.

### HARDWARE FLOATING POINT BOARD

If you do number crunching, then this board is for you. The FPB performs high-speed floating point add, subtract, multiply and divide with selectable precision up to 14 decimal digits. Arithmetic is up to 50 times faster than 8080 software, and BASIC programs can execute up to 10 times faster. A version of North Star BASIC is included. Kit: $259. Assembled: $359. Prices subject to change.

NORTH STAR ★ COMPUTERS

2547 Ninth Street • Berkeley, California 94710 • (415) 549-0858

# Personal Computer Shows

*A visitor to the San Jose show suggests that organizers and exhibitors bring showmanship and general appeal to future computer shows.*

*Paul J. Terrell*
*BYTE, Inc.*
*1261 Birchwood Dr.*
*Sunnyvale CA 94086*

We're growing up as an industry. That was the undertone of the recently held second annual West Coast Computer Faire. Having been an observer and participant for the past two years at computer faires, shows, swap meets, etc., I noticed that something was obviously different in San Jose. The throngs of people were there, and I judged from conversations with manufacturers that the sales level of products was just as brisk, but something was different. But what?

Electricity . . . it was in the walls, feeding the computer displays, but it wasn't in the air. The excitement and anticipation of new products from major manufacturers was missing. Radio Shack, Heathkit, and Commodore had long since made their debut. IBM displayed the 1510 computer, but at $18,000, no one was beating a path to their door.

Electricity is also generated by clever new inventions, but they were in short supply at the Faire. Lots of "me too" products were on display . . . whatever happened to our creativity? Dynabyte put robots on display and dilithium Press released its new publication, *Robots on Our Doorstep*, but both only hinted at what's to come. That leads me to the real message in this article: the maturation of our market.

If our industry trade shows are intended for the insiders—the retailers, computer jocks and technically elite—to share innovations in product design and make purchases for the next quarter- or half-year sales, then we owe it to the general public not to invite them.

*Paul J. Terrell*

If our shows are intended for the general public to increase their awareness of personal computing, then we owe it to our industry to give some thought to the educational aspects of the displays we exhibit and, in general, to the whole theme of the show.

Electricity was taken out of the air by a confused and bewildered public audience that couldn't get close enough to the displays they would have enjoyed if they had been told what they were seeing. It's a bit confusing to pass from the IBM booth with $18,000 computers to the Jade booth with $2 connectors and try to understand the relationship.

At the San Francisco Boat Show a person can spend $3 for an afternoon of entertainment and education in one of America's favorite pastimes. Isn't that what we want of our industry?

At the Consumer Electronics Show the public isn't invited and the guests are grouped according to their buying power by brightly colored badges that distinguish the tire kickers from the chain-store buyers.

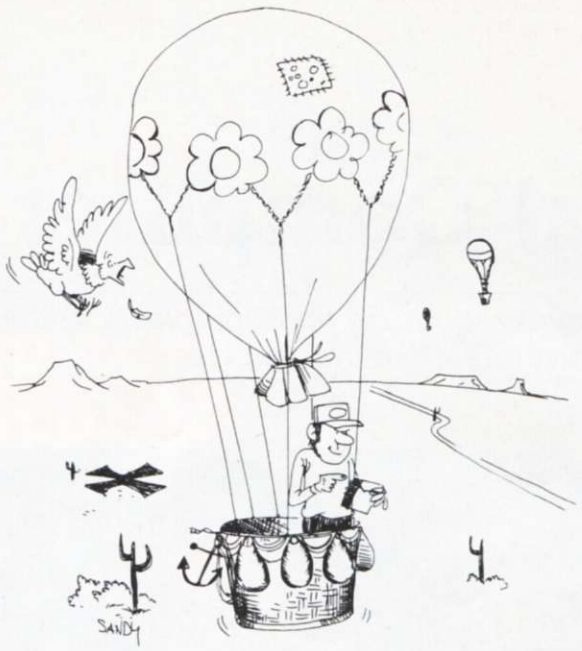
We had all the elements at the Computer Faire for a most enjoyable time—technical seminars for education, contests of computer chess for entertainment and display of new products to ponder and purchase—but something was missing. It's the same thing that is missing in most computer stores today: a guided tour . . . a menu . . . a flowchart. We need to display in the most obvious ways if we are going to bring our secrets of high technology to the public.

As our market matures from the technical activist to the consumer, so should our showmanship. Let's invite everyone to experience the personal-computer evolution in a format understandable to all. Let's bring back "The Greatest Show on Earth" in the form of "Computer Expo."

Let's not forget the obvious help available from the product we display. *Business Week* called them "smart machines"; seems to me they should display, entertain and sell themselves. "Isn't that right, HAL?" (HAL is the computer renegade that took over the spaceship and plotted the destruction of its master in the movie *2001: A Space Odyssey*.)

While push went to shove on the exhibit floor of Computer Faire, plans for a U.S. Personal Computer Show were being presented in a dimly lit suite at the Holiday Inn. The significant point was the format of manufactures being grouped in industry segments (hobby, home entertainment, small business, education and industrial). The product presentations were more an education on computers in the home, business, industry, etc., than attributes of one machine versus the next. Contests and interaction of man and machine were most obvious in the home-entertainment area. Retailers were in storefronts and classes on computer applications were filled with the naive.

With an organized and concerted effort we could bring this "Computer Expo" of the future to Main Street, U.S.A., today. Let's get the organizers and showmen of personal computing together. In combination, their resources would give you and me "The Greatest Show on Earth" in the 1970s. ■




# Cross-Country Balloon Trip

*Take flight with this very entertaining game for you and your SR-52.*

*Mac Oglesby*
*RFD 3*
*Putney VT 05346*

| | | |
|---|---|---|
| 000 | LBL E | Initialization |
| 002 | CMs | Clear registers R00 thru R19 |
| 003 | − RCL 99 = $x^2$ $\sqrt[x]{y}$ 7 = STO 99 | Process input and store for "seed" |
| 015 | 1001 HLT | Display 1001 and wait |
| 020 | LBL A | Each press of A brings a new day . . . |
| 022 | RCL 99 − .2 = | Using subroutine 1', find direction |
| 029 | if pos 9' | wind blows toward. Easterly is |
| 031 | 12 SBR 1' GTO 2' | heavily favored. |
| 037 | LBL 9' | |
| 039 | 45 − 3 SBR 1' | Subroutine 1' was adapted from Peter |
| 045 | if pos 2' | Stark's pseudo random number |
| 047 | + 360 = | generator in "SUBMARINE" |
| 052 | LBL 2' | (KILOBAUD Feb 1977 p. 70) |
| 054 | STO 01 | Store direction in R01 (Θ) |
| 057 | 1 SUM 19 | Count days (increment R19) |
| 061 | SBR 1' STO 00 | Find wind velocity (0 to 30) and store |
| 066 | ÷ 100 + RCL 01 = Fix 2 | Display wind direction and velocity |
| 077 | INV st flg 0 HLT | Authorize a launch and wait |
| 081 | LBL B | Press B to launch the balloon |
| 083 | if flg 0 3' | Error if launch not allowed |
| 086 | fix 0 D.MS STO 04 | Round input to nearest integer and store |
| 092 | − 1 = X (RCL +/- +19) = | Check size of rounded input. Error if |
| 104 | INV if pos 3' | less than 1 or greater than 20 |
| 107 | RCL 04 PR OD 00 | Distance travelled (r) is now in R00 |
| 113 | RCL 01 P/R D.MS SUM 03 | Convert to rect. coords. Store y in R03 |
| 121 | RCL 00 D.MS SUM 02 | Store x in R02 |
| 128 | (RCL 02 − 1001)$x^2$ + RCL 03 $x^2$ = $\sqrt{x}$ | Compute distance from goal |
| 146 | − 15 = | Closer than 15? |
| 150 | if pos 4' | If no, continue the game |
| 152 | RCL 19 A' HLT | Game over! Flash number of days |
| 157 | LBL 4' | Game continues |
| 159 | + 15 = | Repair distance to goal |
| 163 | st flg 0 HLT | Revoke launch permit and wait |
| 166 | LBL 3' | Error section |
| 168 | INV fix 1 ÷ 3 = A' HLT | Flash .3333333333 |
| 176 | LBL 1' | Subroutine 1' |
| 178 | X 30 X ((7 $y^x$ 9 X RCL 99 X .00001) − (RCL − .5) | |
| 206 | fix 0 D.MS) STO 99 = D.MS rtn | |

Note: Angular mode switch must be set at "D"

*Program listing.*

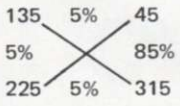
Your balloon starts 1001 miles (use kilometers if you wish) due west from your goal. Each day you are given the wind direction and velocity and decide whether or not to launch. If you choose to launch, you decide how long to stay aloft (from 1 to 20 hours). The game ends if you land closer than 15 miles from the goal. The number of days is flashed.

You will need paper, pencil, protractor and ruler. Using any appropriate scale, locate the goal 1001 miles due east from the start. The program uses the convention for directions that is shown in Fig. 1.

1. To begin a game, load the program and enter any number (6 digits or so).

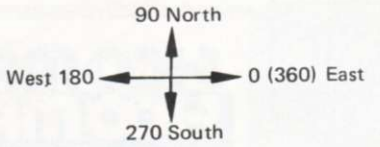
2. Press E. The display will show 1001, your distance from the goal.

3. Press A. The display shows the direction in which the wind is blowing and its velocity. The format is ddd.vv. Note: the wind blows mostly in an easterly direction. In fact, 85% of the wind direction will be between 315 and 45 (SE to NE). Of course, any game may differ from these expectations.




4. Make launch decision. If no, go to step #3 for next day's wind. If yes, enter a whole number from 1 to 20 and press B. Your balloon will travel that many hours. The display shows how far from the goal you landed. Plot your trip carefully. Go to step #3.

5. If you land closer than 15 miles from the goal, the display flashes the number of days. Press CE to stop the flashing. Go to step #2 for a new game. Flashing .3333333333 indicates an error (like if you specify a 22 hour flight). Press CE and try again. ■




(In terms of rectangular coordinates, you start at 0,0; the goal is at 1001, 0.)

*Fig. 1. Conventional compass directions are used to form game board.*

| | Enter | Press | Display | |
|---|---|---|---|---|
| (R99 = 0) | 1977 | E | 1001 | |
| | | A | 354.12 | |
| | 20 | B | 762 | |
| | | A | 347.13 | |
| | 20 | B | 516 | |
| | | A | 337.28 | |
| | | A | 327.20 | |
| | | A | 333.18 | |
| | | A | 7.18 | |
| | 20 | B | 157 | |
| | | A | 318.24 | |
| | | A | 42.10 | |
| | | A | 6.27 | |
| | 4 | B | 53 | |
| | | A | 342.09 | |
| | | A | 349.28 | |
| | | A | 352.14 | |
| | | A | 18.15 | |
| | | A | 36.04 | |
| | 13 | B | Flashing 14 | You arrived within the goal circle in 14 days. Final location was (998,3). |

| | Enter | Press | Display | |
|---|---|---|---|---|
| (R99 = 0) | 354 | E | 1001 | |
| | | A | 352.23 | |
| | 20 | B | 549 | |
| | | A | 36.19 | |
| | 20 | B | 286 | |
| | | A | 326.06 | |
| | 20 | B | 167 | |
| | A | A | 205.12 | |
| | | A | 2.18 | |
| | | A | 23.07 | |
| | | A | 10.01 | |
| | | A | 269.16 | |
| | | A | 325.11 | |
| | 15 | B | Flashing 9 | Reached the goal in 9 days. Final location: (-997, — 3). |

| | Enter | Press | Display | |
|---|---|---|---|---|
| (R99 = 0) | 408 | E | 1001 | |
| | | A | 320.14 | |
| | 20 | B | 807 | |
| | | A | 353.01 | |
| | | A | 180.05 | |
| | | A | 270.21 | |
| | | A | 39.03 | |
| | | A | 8.09 | |
| | 20 | B | 628 | |
| | | A | 40.14 | |
| | 20 | B | 396 | |
| | | A | 315.16 | |
| | 18 | B | 262 | |
| | | A | 349.17 | |
| | | A | 40.15 | |
| | 10 | B | 113 | |
| | | A | 348.11 | |
| | | A | 338.17 | |
| | | A | 343.02 | |
| | | A | 306.26 | |
| | | A | 25.05 | |
| | | A | 134.10 | |
| | 12 | B | 159 | |
| | | B | .3333333333 Flashing. Press CE | |
| | | A | 357.00 | |
| | | A | 335.17 | |
| | 5 | B | 88 | |
| | | A | 333.21 | |
| | | A | 340.21 | |
| | | A | 9.25 | |
| | | A | 327.26 | |
| | | A | 330.14 | |
| | | A | 322.09 | |
| | | A | 25.07 | |
| | 13 | B | Flashing 25 | Reached the goal in 25 days. Final location: (1001, 5) |

*Sample runs.*

# TELETYPE® MODEL 33 ASR

**COMPUTER I/O COMPLETE WITH:**

- Tape Punch
- Tape Reader
- Ready to Go
- Line/Local Wired
- Guaranteed 30 Days

**$840.00**

INCLUDING PACKING
F.O.B. - N.J. FACTORY

| | |
|---|---|
| COMPLETE COVER READY TO INSTALL | $ 35.00 |
| COPY HOLDER (182036) | $ 14.00 |
| TAPE UNWINDER (12" NEW) | $ 55.00 |
| WIND UP PAPER TAPE WINDER (1") | $ 52.00 |
| ELECTRIC PAPER WINDER (LPW-300 NEW) | $ 50.00 |
| DEC TYPE READER RUNCARD | $ 45.00 |
| ACOUSTIC COUPLER OMNITEK 501A (NEW) | $160.00 |
| READER POWER PACK (182134) | $ 35.00 |
| EIA RS-232 INTERFACE | $ 55.00 |
| ACOUSTIC COVERS with FAN (NEW) | $258.00 |

If its for TELETYPE® We Have It.

If you don't see what you need, CALL or WRITE!

**TELETYPEWRITER COMMUNICATIONS SPECIALISTS**
550 Springfield Avenue • Berkeley Heights, N. J. 07922
(201) 464-5310 • TWX: 710-986-3016 • TELEX: 13-6479

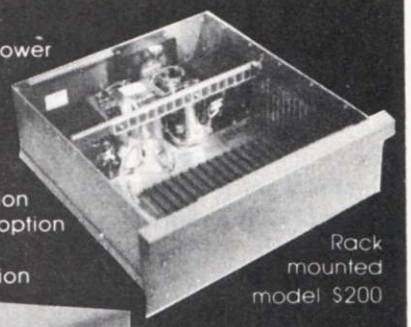
T13

*SUBSIDIARY OF VAN'S W2DLT ELECTRONICS.*

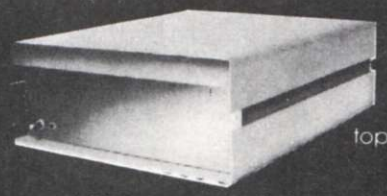
# S-100 MAINFRAME $200

S-100 MAINFRAME $200

- NOT A KIT
- 8v@15A, ±16v@3A power
- Rack mountable
- 15 slot motherboard
- Card cage
- Fan, line cord, fuse, switch, EMI filter
- Desk top version option
- 8v@30A, ±16v@10A option
- SS-50 bus option
- voltage monitor option



Rack mounted model $200



Desk top model $235

Write or call for a copy of our detailed brochure which includes our application note BUILDING CHEAP COMPUTERS.

**INTEGRAND**

8474 Ave 296 • Visalia, CA 93277 • (209) 733-9288

We accept BankAmericard/Visa and Master Charge

# Here's how the Computer Workshop works for you.

Our clients get the whole thing—total systems engineering. We can help you with evaluation and development of requirements for data processing and associated hardware, or give you management assistance in developing systems. We'll develop and design computer systems, and we'll design, organize and manage special training programs for your staff.

So even if your needs are minimal right now, we can be a big help. Then as your needs grow, you'll know where to come.

**The Computer Workshop**

Here are some of the firms we represent:

Alpha Microsystems
Cromemco
Digital Systems
Gimix, Inc.
Icom
Industrial Micro Systems
International Data Systems, Inc.
Lear Seigler
Microcomputer Business Systems
Micro Term, Inc.
North Star Computers
Polymorphic Systems
Seals Electronics
Smoke Signal Broadcasting
Southwest Technical Products Corp.
Technical Design Labs
Vector Graphics

THE COMPUTER WORKSHOP

Here's where to find us: Offices: **BALTIMORE** 4005 Seven Mile Lane, Baltimore, Md. 21208 (301) 486-5350/**PITTSBURGH** Robar Building, 4170 Wm. Penn Highway, Murraysville, Pa. 15660 (412) 327-0455/**WASHINGTON, D.C. AREA** 1776 Plaza, 1776 E. Jefferson Street, Rockville, Md. 20852 (301) 468-0455/**NORTHERN VIRGINIA** 5240 Port Royal Road, Suite 203, Springfield, Va. 22151 (703) 321-9047/**KANSAS CITY** 6 East Street, Parkville, Mo. 64152 (816) 741-5055

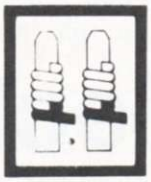
C31

ok®

## "INDUSTRIAL"
### WIRE WRAPPING TOOL

Model BW 520 is a battery operated wire-wrapping tool, also available in reversible models for wrapping and unwrapping operations. The rechargeable nickel cadmium battery is self contained in the handle of the tool. Also available with "Backforce" device to prevent overwrapping.

Uses any 24AWG thru 32AWG wrapping bits and sleeves.



**OK MACHINE & TOOL CORPORATION**
3455 Conner St., Bronx, N.Y.10475 / (212) 994-6600 / Telex 125091

Model **BW520** **NEW**

**BATTERY WIRE-WRAPPING TOOL**

ONLY **$108.70**
BATTERIES INCLUDED
BIT & SLEEVE NOT INCLUDED

Model **BW928** **NEW**

**BATTERY WIRE-WRAPPING TOOL**

ONLY **$49.95**
BATTERIES NOT INCLUDED
BIT & SLEEVE NOT INCLUDED

ok®

## "INSTALLATION AND FIELD SERVICE"
### WIRE WRAPPING TOOL

Model BW 928 is a battery operated wire wrapping tool, also available in reversible models for wrapping and unwrapping operations. Exceptionally useful for installation and field service work, or small production jobs. Available with "Backforce" device to prevent overwrapping.

Uses any 24AWG thru 32AWG wrapping bits and sleeves.

**OK MACHINE & TOOL CORPORATION**
3455 Conner St., Bronx, N.Y.10475 / (212) 994-6600 / Telex 125091

ok®

## "HOBBY"
### WIRE WRAPPING TOOL

For AWG 30, .025" (0.63mm) sq. post, "MODIFIED" wrap, positive indexing, anti-overwrapping device.

O5

**OK MACHINE & TOOL CORPORATION**
3455 Conner St., Bronx, N.Y.10475 / (212) 994-6600 / Telex 125091

Model **BW630** **NEW**

**BATTERY WIRE-WRAPPING TOOL**

ONLY **$34.95**
BATTERIES NOT INCLUDED
COMPLETE WITH BIT & SLEEVE

# Transfer Vectors vs Absolute Addressing

*You can save memory by reusing program modules and algorithms.*

*William Nichparenko*
*Sunick Systems*
*185 Summit Dr.*
*Santa Cruz CA 95060*

*Those of you involved in heavy assembly language programming will find Bill's article very worthwhile. He's discussing the technique of using transfer vectors which, incidentally, was the subject of an article by Art Childs in Issue #1 of* Kilobaud. *Art discussed the use of the transfer vector with regard to accessing I/O routines . . . Bill's approach deals with using the transfer vector as opposed to absolute addressing. – John.*

If you are not fortunate enough to have a sophisticated operating system, but would like to have some modular software that you can use with ease, the follow-

| | Machine Language Hex Location | Operation Code | Label | Assembly Language Mnemonic | Operand | Comment |
|---|---|---|---|---|---|---|
| | | | | ORG | $6F7 | ORIGIN FOR OBJECT CODE |
| | 6F7 | 7E 0802 | | JMP | C | JUMP TO LABEL C |
| | 6FA | 7E 07FA | | JMP | B | JUMP TO LABEL B |
| | 6FD | 7E 0721 | | JMP | A | JUMP TO LABEL A |
| Entry → | 700 | CE E000 | Z | LDX | $E000 | BEGIN PRINT A ACCUM. SUB |
| | X | X X X | | X | X | X X |
| | X | X X X | | X | X | X X |
| | 721 | 7D 01FF | A | TST | $1FF | BEGIN CARRIAGE MOVE SUB |
| | X | | | | | |
| | X | | | | | |
| | 7FA | 86 1F | B | LDAA | #$1F | BEGIN PAPER MOVE SUB |
| | 802 | B1 07A0 | C | CMPA | $7A0 | BEGIN END OF LINE CHECK SUB |
| | 8A0 | 39 | | RTS | | RETURN TO MAIN PROGRAM |

*Fig. 1. Assembly language representation of the module and the object code generated.*

ing method may be of interest to you.

### The Problem

Suppose you have a software module called Printer Control (known as the base module from here on). Fig. 1 is arranged so that the assembly language representation of the module and the object code (or machine language) that it generates are adjacent for easy comparison. This module has a reference entry point of 700 hexadecimal, used whenever entering the module by itself. The purpose of the module is to take a byte of information in the A accumulator, operate on it, and return to the main program when completed. This base module contains various subroutines within it (for illustration purposes call these subroutines A,B,C) that are useful to other modules. For example, a TAB module, entirely separate from the Printer Control module, may want to use subroutine A of the base module to perform a carriage movement without printing any letters. The problem that arises is that other modules, not having been assembled at the same time as the base module, must call the subroutines A, B, and C by absolute addresses (i.e., a machine code value in hex or octal as opposed to a symbolic address). This is fine as long as none of the addresses in our base module change. Unfortunately, any time you want to modify the base module, many of the subroutine call addresses in all the other modules using the base module subroutines may have to be changed. This is a time consuming and error prone process.

### The Solution

A simple solution is to place a transfer vector (a collection of jump commands pointing to the locations of assembly language subroutine labels A, B, C) in front of the module's object code. The absolute address of the transfer vector never changes, but the addresses within the module that correspond to subroutines A, B, and C are changed every time the module is altered and reassembled.

If new subroutines are added to the module, a new jump instruction is appended to the transfer vector so as to not change the absolute addresses of the jump instructions contained in its earlier versions. The only thing that need be remembered is to arrange the assembler origin such that the reference entry point to the module does not change.

Fig. 2 shows how various addresses might change with addition of a new subroutine, labeled D, to the module. Note that a new assembler origin of 6F4 is necessary to maintain the absolute addresses of the old transfer vector and the reference entry point of the base module (three additional bytes of memory are needed for each new jump instruction). Note also how the absolute addresses of each subroutine have changed, and how the corresponding jump command operands in the transfer vector have changed accordingly. A subroutine call to location 6FA will enable the use of subroutine B in either the old or the new edition of the base module.

Fig. 3 shows how the base module can be used in any one of the three possible modes. The first mode is a direct call to the base module at location 700 hexadecimal. This uses the full capability of the base module to call each of the three subroutines A,B,C as needed. An important point to note is that each of the subroutines within the base module must have its own return command in order to be useful outside the base module. Upon completion, the base module returns control to the main program.

The next call from the main program illustrates the use of the transfer vector to call subroutine B in the base module. Finally a direct call to subroutine C at its absolute address illustrates what we are trying to avoid by the use of a transfer vector. If the address of C changes due to a modification in the base module, so must the absolute call in every main program that uses subroutine C directly. ■

| | Machine Language Hex Location | Operation Code | Label | Assembly Language Mnemonic | Operand | Comment |
|---|---|---|---|---|---|---|
| | | | | ORG | $6F4 | ORIGIN FOR OBJECT CODE |
| *New routine added.* → | 6F4 | 7E 0762 | | JMP | D | JUMP TO LABEL D |
| *Old addresses still valid.* | 6F7 | 7E 0843 | | JMP | C | JUMP TO LABEL C |
| | 6FA | 7E 0811 | | JMP | B | JUMP TO LABEL B |
| | 6FD | 7E 0721 | | JMP | A | JUMP TO LABEL A |
| *Entry point unchanged.* → | 700 | CE E000 | Z | LDX | $E000 | BEGIN PRINT A ACCUM. SUB |
| | X | X X X | | X | X | X X |
| | X | X X X | | X | X | X X |
| | 721 | 7D 01FF | A | TST | $1FF | BEGIN CARRIAGE MOVE SUB. |
| | 762 | BB E000 | D | ADDA | $E000 | BEGIN PAPER OUT CHECK SUB |
| | 811 | 86 1F | B | LDAA | #$1F | BEGIN PAPER MOVE SUB |
| | 843 | B1 07A0 | C | CMPA | $7A0 | BEGIN END OF LINE CHECK SUB |
| | 921 | 39 | | RTS | | RETURN TO MAIN PROGRAM |

*Fig. 2. Address changes with new subroutine (D).*

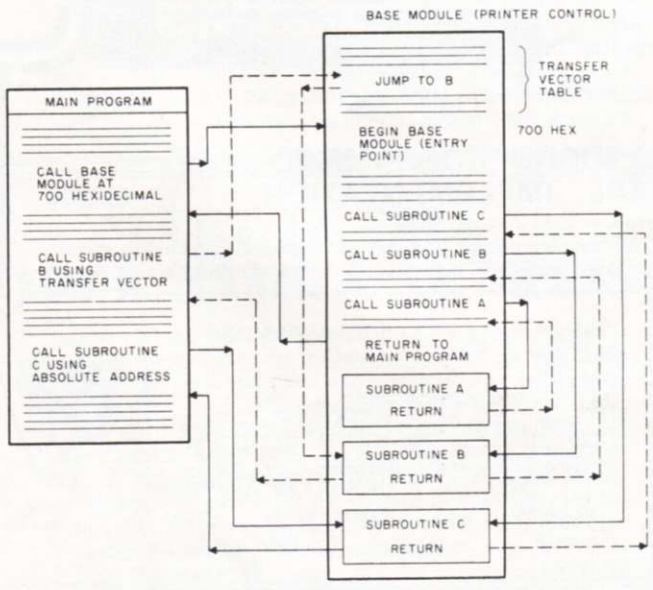



*Fig. 3. Various uses of base module.*

Jeffrey G. Towle
696 Beech St.
E. Lansing MI 48823

# Error Correcting Codes

*Error correcting codes are a software technique for detecting and correcting hardware errors that sometimes occur in data-transmission systems.*

Most of us are familiar with detecting errors using a parity bit. The ASCII 7-bit code is usually accompanied by an 8th bit which is set either as 0 or 1 depending on whether the first 7 bits have an even or odd number of bits turned on (or set to 1). The kind of parity is either even or odd depending on whether or not the entire 8 bits have an even or odd number of bits on. Since the 8th bit is set on or off as a result of the first 7 bits, every set of 8 bits can be sent with the identical parity. At the receiving end, all we need to do is check each 8-bit string for its parity; when one comes along with the wrong parity, we have detected an error. Unfortunately, we have no idea which bit has been clobbered in the channel, and therefore we must request that the string be retransmitted.

Checksums perform the same sort of basic function (detecting errors), but they too suffer from the inability to indicate where the error(s) occurred. Also, since checksums occur after several strings of bits have been sent, a checksum error means that the whole message must be resent. If the channel is subject to much noise, we can expect at least a few errors in a long message, and we may end up sending and resending the message several times before an error-free message is received.

| bit 1 | bit 2 | result |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

*Table 1. Truth table for the Exclusive OR operation.*

*One way to interpret this operation is to view the result as a parity bit which results in even parity. Thus, if we have a string of bits such as 001011 and perform the Exclusive OR $0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1$ the result would be 1. Being able to do Exclusive ORing on a string of bits is necessary when more bits are involved in each message string (more than just one like in this first example). Another way to get the result of a string of EX ORs is to find out how many 1s there are, and if the number is even, the result is 0; if the number is odd, the result is 1.*

With a few extra parity bits interspersed throughout the string of bits, we can not only detect errors but also locate the bit in error, invert it, and have a correct message received without having to send the message more than once. This sort of parity checking (with extra parity bits) is known as error correcting, a bit more sophisticated and helpful than simply error detecting.

## Error Correction — Needs More Parity Bits

In order to have a coding scheme that not only detects errors, but corrects them, it seems only natural to have to pay a price. The price is in terms of extra parity bits. For short strings of bits, the parity bits occur almost as frequently as the message bits. In fact, to send a single bit and detect and correct errors, two parity bits are required. An example will help clarify this point, but in words we must be aware of the fact that errors can occur in the parity bit(s) as well as

| Position<br>Binary Address | |
|---|---|
| 0 = 0 0 0 | |
| 1 = 0 0 1 | P |
| 2 = 0 1 0 | P |
| 3 = 0 1 1 | M |
| 4 = 1 0 0 | P |
| 5 = 1 0 1 | M |
| 6 = 1 1 0 | M |
| 7 = 1 1 1 | M |

*Table 2. The parity bits are at positions in which only a single bit is on. Look down a column. Wherever there are 1s in the column, note the positions. There will only be one parity bit in each column and three message bits. (In general, $(2^{n-1}-n)$ message bits will be used to encode each of the n parity bits.)*

in the message bit(s). If such an error does occur, we want to be sure that we don't assume that the message bit was in error and invert it by mistake.

**One Message Bit**

Suppose we have only one bit to send, which we shall label as M. The error correcting code will have three bits in total: M and two parity bits, $P_1$ and $P_2$. The string will look like MPP, or better, $M_3P_2P_1$ with the subscripts indicating which bit is sent first. $M_3$ means that the message bit will be the third bit in the string. Now we must know how to encode the parity bits $P_1$ and $P_2$.

The error correcting code discussed here depends heavily on the use of the Exclusive-OR logical function (see Table 1). The notation $\oplus$ will be used to indicate Exclusive-OR.

Let's set $P_1=M_3$ and $P_2=M_3$ and see what happens. If the message is M=1, the string will be 111. Now at the receiving end, the string is analyzed by determining the results of the Exclusive OR checks shown in Example 1.

Believe it or not, this check indicates that there is no error in the string. Reading from the top down, the two 0s indicate that error occured in the 0th bit (which does not exist).

**One Message Bit — One Error**

If an error in transmission had occurred, one of the bits in the message would have been a 0 instead of a 1. If the first bit were the source of error, the received message would have been 110 and the check would have given Example 2. This indicates that the error occurred at location 01 binary, or the first bit (which we knew because we made the error in bit 1). If the error occurred in bit 2, the received message would have been 101 and the check would be Example 3. This indicates an error in bit 10 or bit 2 (10 is binary for 2). Of course, had the error been in the third bit, or message bit, the check would reveal 11 which is binary for 3.

$$A_1 = P_1 \oplus M_3 = 0$$

$$A_2 = P_2 \oplus M_3 = 0$$

*Example 1.*

$$A_1 = P_1 \oplus M_3 = 1 \oplus 1 = 0$$

$$A_2 = P_2 \oplus M_3 = 0 \oplus 1 = 1$$

*Example 2.*

$$A_1 = P_1 \oplus M_3 = 0 \oplus 1 = 1$$

$$A_2 = P_2 \oplus M_3 = 1 \oplus 1 = 0$$

*Example 3.*

$$P_1 = M_3 \oplus M_5 \oplus M_7$$

$$P_2 = M_3 \oplus M_6 \oplus M_7$$

$$P_4 = M_5 \oplus M_6 \oplus M_7$$

*Example 4.*

**Several Message Bits**

There are some fixed number of message and parity bit combinations which are not possible in this single error correction coding scheme. In general, it is the number of parity bits which fixes the number of message bits included in each distinct string of bits. The number of bits in a string is $2^n$-1 where n is the number of parity bits. For the two parity bit example, $2^2$-1 = 3, the total number of bits in the string. The number of message bits in the string is, therefore, ($2^n$-1) - n. What does this mean if we try 3 parity bits instead of 2? Plugging in we get $2^3$-1-3 = 4 message bits. Thus, the string will be 7 bits long, and there will be 4 message bits and 3 parity bits. Let's jump to 5 parity bits. In this case, there will be 31 bits in the message, and 26 message bits. The ratio of information (message bits) to checking (parity bits) improves as the strings get longer. But as we will see, the amount of processing needed during encoding and decoding also increases as the length of the string increases.

In this example using 7 bits (4 message and 3 parity), we will assume that you understand the Exclusive OR operation and binary notation. The first problem is to locate where the parity bits belong. Mathematically, they are put in locations $2^0$, $2^1$ and $2^2$ which is easily generalized, meaning that for n parity bits, the ith one belongs in position $2^{i-1}$. For our three parity bits, this means they are the first, second, and fourth bits in the string. The message bits are put in the unused positions, which are 3, 5, 6 and 7.

To determine the coding for the parity bits we'll give you a scheme later. For now, just the notation will do. First, our string will look like this:

$$M_7\ M_6\ M_5\ P_4\ M_3\ P_2\ P_1$$

and is sent out by shifting it to the right one bit at a time (e.g., serial transmission across the phone lines or into a cassette recorder). The parity bits are determined using the message bits in Example 4.

This may appear at first to be sort of random, but look closely. The parity bits occur at positions such that any combination of 0s and 1s for them will encode a binary address from 0 to 7 by adding up the position numbers (which we will find out is the address of the bit which is in error). Also, the positions of the message bits are such that only those in which bit 1 is on (for positions 3, 5, and 7 in binary) are used to encode parity bit $P_1$. $P_2$ is encoded using the message bits in locations in which the second bit of a three bit binary address is coded as 1, which are positions (addresses) 3, 6 and 7. Likewise, $P_4$ uses the bits located in positions in which the third bit of a three bit binary address is 1. A look at the list of numbers in Table 2 will help.

The position 0 (000 binary) is not used during the encoding. For the rest of the table, the columns are the key thing to look at. The parity bit being encoded will be associated with three message bits, found by the positions in which a 1 is found in the column along with the 1 associated with the parity bit. The first parity bit is in position 1, and in that column, positions 3, 5, and 7 also have a 1.

The next step is to take a message (any 4-bit message will do) and encode it into a 7-bit string. I chose the message 0111 which is hex 7. (The important thing to keep consistent is which way to read the message. The bits are to be shifted right if they are serialized.) In this 4-bit message, the individual message bits are: $M_3 = 1$, $M_5 = 1$, $M_6 = 1$, $M_7 = 0$. The parity bits are shown in Example 5. The entire string of bits will be (starting with bit 7 on the left) 0110100.

$$P_1 = 1 \oplus 1 \oplus 0 = 0\ (= M_3 \oplus M_5 \oplus M_7)$$

$$P_2 = 1 \oplus 1 \oplus 0 = 0\ (= M_3 \oplus M_6 \oplus M_7)$$

$$P_4 = 1 \oplus 1 \oplus 0 = 0\ (= M_5 \oplus M_6 \oplus M_7)$$

*Example 5.*

$$A_1 = P_1 \oplus M_3 \oplus M_5 \oplus M_7 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$A_2 = P_2 \oplus M_3 \oplus M_6 \oplus M_7 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$A_4 = P_4 \oplus M_5 \oplus M_6 \oplus M_7 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

*Example 6.*

Now let's send this message with no errors. At the receiving end, the string is processed by performing Example 6. The address of the error is 000, or position 0, which is not in the string. If none of the bits have changed during transmission (no errors) then it makes sense that each of the Exclusive ORs should result in 0s. Remember, $P_1$ was the result of ($M_3 \oplus M_5 \oplus M_7$) and therefore, doing the checking is like doing the following: ($M_3 \oplus M_5 \oplus M_7$) $\oplus$ ($\overline{M}_3 \oplus \overline{M}_5 \oplus M_7$) where the M means the received message bits. We can reorient this expression as ($M_3 \oplus \overline{M}_3$) $\oplus$ ($M_5 \oplus \overline{M}_5$) $\oplus$ ($M_7 \oplus \overline{M}_7$) which compares each message bit before and after transmission. If no change has resulted, the $\oplus$ will be 0 for each pair (a bit $\oplus$'d with itself is 0), but if a single bit has inverted due to error, the result will be a 1. Studying this a bit longer (no pun intended) will show you how the scheme will give you the actual position of the error.

$$A_1 = P_1 \oplus M_3 \oplus M_5 \oplus M_7 = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$A_2 = P_2 \oplus M_3 \oplus M_6 \oplus M_7 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$A4 = P_4 \oplus M_5 \oplus M_6 \oplus M_7 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$

*Example 7.*

$$P_1 = M_3 \oplus M_5 \oplus M_7 \oplus M_{11} \oplus M_{13} \oplus M_{15} \oplus M_{19} \ldots \oplus M_{31}$$

*Example 8.*

$A_1 = 01010101$ $\quad \overline{A}_1 = 10101010$

$A_2 = 01100110$ $\quad \overline{A}_2 = 10011001$

$A_4 = 01111000$ $\quad \overline{A}_4 = 10000111$

(The bar means not or complement, or in this case, when $A_n = 0$.)

*Example 9.*

### A 7-Bit String with One Error

Using the same message bits as before, let's find out what we'd do if we received the following message: (don't look back at the original string just yet) string = 0100100. We know this is a 3 parity bit code, and the parity bits are in locations 1, 2 and 4, thus $P_1 = 0$, $P_2 = 0$, and $P_4 = 0$. Also, the message bits are $M_3 = 1$, $M_5 = 0$, $M_6 = 1$, and $M_7 = 0$. Now let's decode the address of the error, if any, in Example 7.

The address of the error is position 5, read from $A_4$ and $A_1$, both being on; their sum is 5. Now correct the string to read 0110100 (invert position 5's bit from a 0 to a 1) which is the proper string. The message bit was in error in this case, but in others, the parity bit may have been the one in error. Rather than correcting the string, it might be quicker and more efficient simply to check the address to determine whether parity or message bits were involved and ignore the correction if a parity bit was in error (who cares about parity bits after the message has been sent?).

Try doing the error checking for other position errors. Also, where there are two errors, this system will not locate the address of the error, but it will indicate that an error was received (one or more of the error address bits will be on). This error correcting code will handle most errors as long as the error rate is fairly small (so that two errors are not likely to occur in the same string).

### What to do with Error Correcting Codes

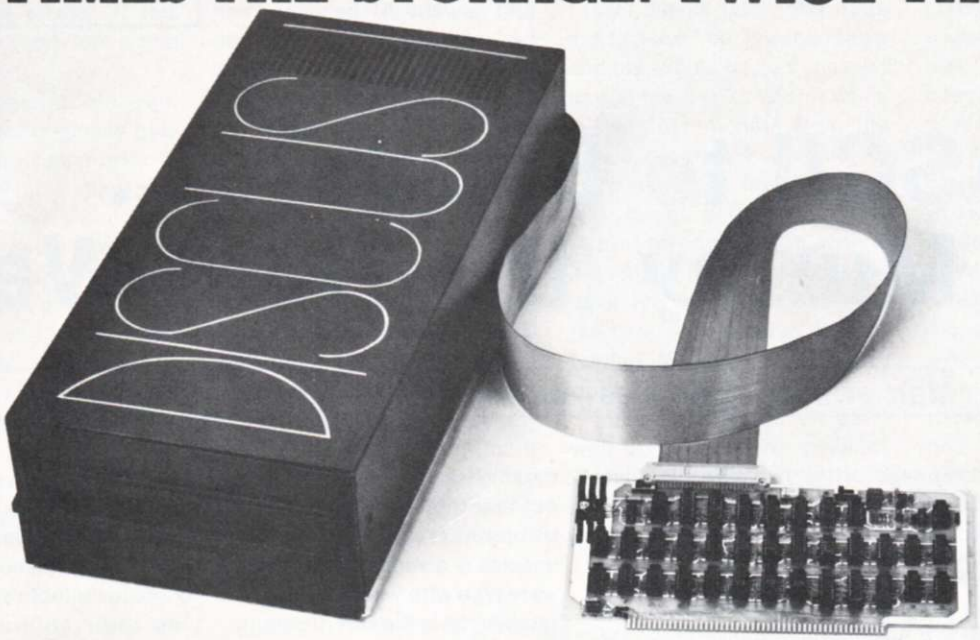
The 7-bit string used in the example only allows 4 message bits to be sent with each 3 parity bits. This sounds inefficient, and in terms of getting a lot of data through a channel it is. However, it helps eliminate an occasional missing bit, which may be even more disastrous than a slower effective bit rate. With 5 parity bits, the relative number of parity bits is substantially reduced compared to the number of useful data bits. The 31 bits in a string result in 26 message bits, which is enough to encode 3 8-bit bytes with 2 extra bits which are of little use. The problem with the 31 bit message is the number of Exclusive ORs needed. For example, the first parity bit is found using the calculation in Example 8.

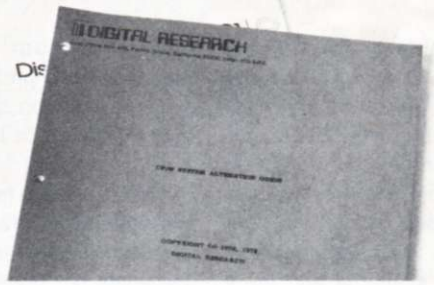
Another problem for the programmer is how to do bit-specific Exclusive ORing efficiently. Hopefully, an article will appear describing a program to implement the 7-bit string error correcting coding in the near future. As a start on the programming, put masks in memory as shown in Example 9.

If it is determined that the location of the error is $A_1$=1, $A_2$=0, $A_4$=1, then use the AND function to AND $A_1$, $\overline{A}_2$, $A_4$ which will result in the following mask in the Accumulator: 0001000. Simply $\oplus$ this with the received string, and the proper bit will be inverted, which is bit 5. Note also that by ANDing $A_1$ with the string, only bits 1, 3, 5 and 7 can get into the Accumulator, at which time you need only to determine the parity (number of 1 bits) and if it is even, the result will be 0 indicating that as far as the $A_1$ portion of the position, use $\overline{A}_1$ mask. Whether to use $A_2$ or $\overline{A}_2$ mask, AND $A_2$ with the string, determine the parity, and select $A_2$ or $\overline{A}_2$ depending on whether the results are odd or even respectively. ■

# The New Star In Disk Systems

## 4 TIMES THE STORAGE. TWICE THE SPEED.




## IBM COMPATABILITY

Introducing Thinker Toys™ new DISCUS I™ standard floppy disk system for just $895.*

It's the new standard of performance in S-100 disk systems. Because DISCUS I's standard floppy format holds 4 times the data per disk as North Star's mini-floppy costing $777.

And DISCUS™ is twice as fast as the North Star. And has twice as many tracks. And twice as much data on each track.

That's four times the performance for just $118 more.

But that's not all. DISCUS I™ is both S-100 and IBM-compatible. Plus, our plug-in S-100 controller has an on-board serial I/O port, cache buffer, and read/write/seek routines in on-board ROM. There's even a bootstrap loader.

DISCUS I™ comes as a completely assembled system. A Disk Jockey I™ controller and cable set interface your S-100 mainframe to the handsome free-standing Shugart disk drive unit—which comes complete with cabinet, power supply and fused detachable line cord.

The price also includes DOS, ATE™ assembler/editor, patches for CPM, and Advanced BASIC (available mid-1978) software.

We also offer CPM ($70) and Micro-soft Extended Disc BASIC ($199).

DISCUS I™ has Shugart reliability, Morrow's Micro-Stuff performance and Thinker Toy™ sales and service nationwide. It's fully assembled, tested and warranted (drive 45 days, controller 1 year).

Don't invest in a floppy that can't deliver big performance right now. Buy DISCUS™ standard floppy system today . . . while our special $895 introductory price ($100 savings)* is still available.

Ask your local computer store to order DISCUS I™ before July 1st. Or send your check or money order to THINKER TOYS™, 1201 10th Street, Berkeley, CA 94710. Or call (415) 527-7548, 10-4 pm PST. Add $3.00 handling; California residents add tax.

**Thinker Toys**™

1201 10th Street
Berkeley, CA 94710

* Introductory prices until June 30, 1978 only.

T28

*John A. Lehman*
*716 Hutchins #2*
*Ann Arbor MI 48103*

*Ray Graham*
*2365 Twinlakes 2A*
*Ypsilanti MI 48197*

Publisher's note: *The Baudot referred to in this article is actually Murray code as per my remarks on page 4. Have we all been the victim of a dastardly French revisionism plot?*

# ASCII to Baudot . . . er . . . Murray (the Hard Way)

## *Mini Micro Mart strikes again—as you will see in this consumer report.*

*And so it goes—the continuing saga of the computer hobbyist community and its special relationship with Mini Micro Mart. Naturally, you won't be able to get into this relationship until you order something from Mini Micro Mart; and then you'll find out (as John, Ray and so many others have) why it is a* special *relationship. Without hesitation, I can recommend at least one product for you to buy from MMM: the GPA-B ASCII/Baudot converter board. The reason for this is because you can become familiar with possible problems you'll encounter by reading the following article. —John.*

The Mini Micro Mart GPA-B is a kit that converts parallel ASCII to serial Baudot. This is obviously a very useful device for hams and computer hobbyists who have Baudot teleprinters, such as the models 15, 19 and 28. These are frequently available and quite inexpensive; many hams have them already. Conversion from ASCII to Baudot can be done in either hardware or software; a hardware conversion system can save a couple-hundred bytes to be used for other purposes in your computer. Whether this particular hardware conversion is worthwhile will be left up to the reader to judge—read on!

For those who have not dealt with Mini Micro Mart, they are located at 1618 James Street, Syracuse NY 13203. They offer quite a range of computer equipment, both surplus and kits, at very low prices. Cynics might observe here that you usually get what you pay for; this is not far from the truth.

One of Mini Micro Mart's most recent bulletins points out that they cater to the "true hobbyist." This particular species is a subset of the hardware type with whom we are all familiar; he is the sort of person who feels that Heathkit-style instructions are demeaning and that (one gathers) finding errors in design and manufacture is an interesting challenge. This is a rather recent definition of Mini Micro Mart's customers; it did not exist (at least in print) when I bought the kit. But since they specify who their ideal customers are, let us now explain who we are so that readers can judge how well we fit the mold.

"I" in the rest of the article refers to John, K8UP (as in microprocessor). I am a ham (since 1964) with the usual experience with a combination of home-brew, kit and commercial ham gear. Digital electronics is fairly new to me, although I have been programming computers for a couple of years now. As a computer hobbyist, I am definitely a "software type."

Ray is a hardware type—an electronics technician by profession, as well as by avocation. His background, like mine, is mostly in communications electronics. We are both members of the Ann Arbor Computer Club, which is the source of this collaboration.

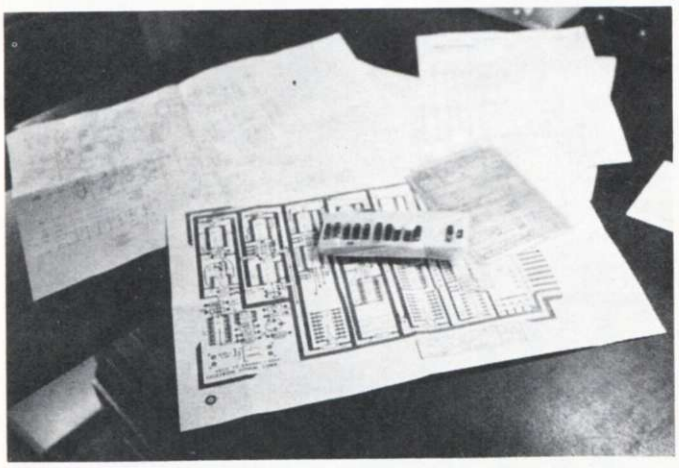


*Photo 1. What was shipped.*

### More on MMM

Mini Micro Mart does not service what it sells. Until a few months ago, their customers were forced to live a catch-as-catch-can existence scurrying to locate suitable shops to service their equipment; now MMM recommends an independent repair service. Their reasoning is that some people do not know how to put kits together, and so the company cannot take responsibility for resulting errors. My own opinion (for what it is worth) is that if MMM took responsibility for making the kits work, they would not be able to send out erroneous documentation, defective parts or incomplete kits the way they do now.

What I received in this instance is pictured in Photo 1; documentation consists of a parts list, a schematic and a layout diagram. The schematic and the layout (at least for this kit) were not in agreement. Part of the reason for this seems to be that MMM makes several ASCII/Baudot conversion boards . . . and they use the same documentation for all of them!

Only by comparing the parts shipped with the parts list (which again does duty for three kits) and figuring out what everything on the schematic does is it possible to figure out what is needed for the GPA-B kit. For those who have trouble, a corrected schematic is given in Fig. 1. This modification contains additions and corrections (more on

these later) but does not include the parts for different kits. Incidentally, I never did receive all the parts for this kit; fortunately, the missing ones were available locally. If this keeps up I may turn into a true hobbyist!

Now for the history of the kit. The circuit was originally published in *Radio Electronics* in March 1976. In the original publication there were a couple of errors, which were corrected the next month. Not all of these are corrected in the GPA-B (I presume that the folks at Mini Micro Mart do not believe in testing their kits before they put them into production and sell them).

I ordered the kit (for $35), along with a surplus keyboard, in early February 1977. The keyboard came a few weeks later with a note that the GPA-B would be delayed about a week. After a month of waiting, I wrote the company to ask what had happened. No answer. Two months from the time the keyboard arrived, I again wrote. This time I was informed that the board had not been shipped because MMM didn't have any. They did have a similar board . . . would I accept that? I said yes.

Three weeks later at a computer fair in Cleveland, I ran into Mini Micro Mart representatives who said they didn't have any of the boards yet (they had one as an exhibit and were taking orders for them), but that it wasn't their fault and they would ship one right away. They were obviously not too enthused about my scaring away customers, although a couple of other people at the show commented that they, too, were having problems with orders.

Two weeks later, still nothing. I called. Yes, they had them and one would be sent the next day. A few weeks later it came —over four months from the time I ordered it (prepaid, of course). I read the directions (half page), checked for non-plated-through holes with a magnifying glass (I should have used an ohmmeter) and built the thing—after going out and buying IC sockets. It didn't work.

First, I compared all parts with parts layout. No problem. Then, using an ohmmeter, I compared the schematic to the board. In addition to the unplated-through holes I had caught with the magnifying glass, there were 27 additional holes that had either been bad to begin with or had come unplated when I inserted the IC sockets. I repeat: *twenty-seven* bad plated-through holes. Ray later found three more; it would have been much easier if the kit had been sold as not having plated-through holes, then I would have soldered everything on both sides to begin with.

"Mini Micro Mart recognizes a responsibility to provide the highest quality components, whether sold as individual items or as part of a kit." So reads the manual. Yep. I might add that Mini Micro Mart not only recognized its responsibility to provide the highest quality components in the form of

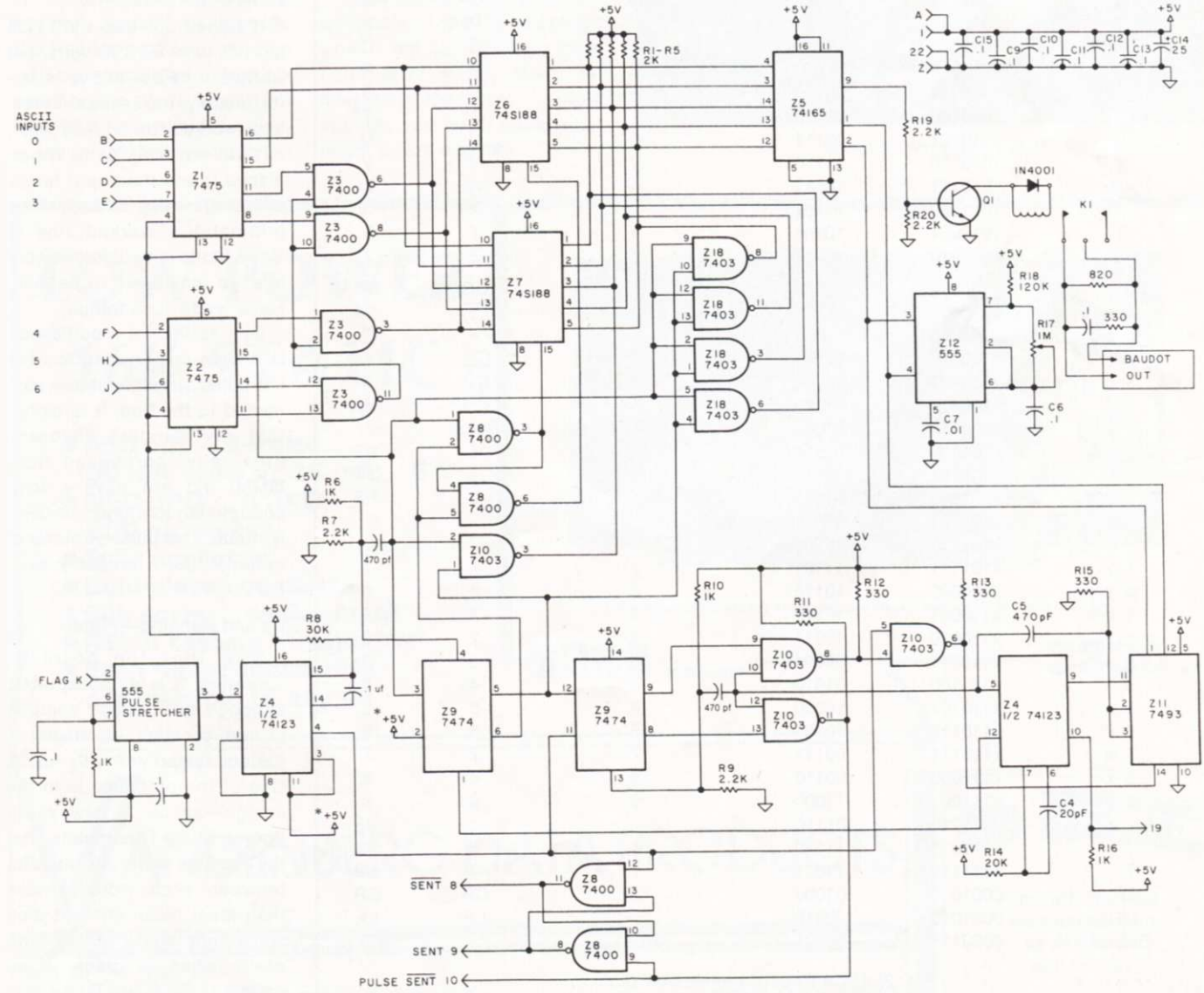



*Fig. 1. Mini Micro Mart GPA-B ASCII-Baudot conversion board. Schematic modified by John Lehman and Ray Graham.*

the board, but also included three bad ICs. We replaced them locally, rather than return them and risk waiting a couple more months.

After I had spent about 25 hours tracing pulses and soldering bad holes, I gave up and called Ray. He found a couple more bad holes, two bad ICs, in addition to the one I had found, and two errors in the schematic. Error here means something more serious than mislabeling a part or misnumbering an IC pin, which also happened.

The first problem was that the clear pins on the 74123 were left floating, rather than being tied high. Although not exactly an error, this is not considered good practice because the pins are susceptible to noise. They were tied high as shown on the corrected diagram.

The second problem was that pin 2 of the 7474 was grounded rather than high. This causes the board to shift from figures to letters, but not shift from letters to figures. This was an error in the original unit. After many hours of work, Ray corrected the problem and brought the unit back to me.

I plugged it in expecting no problems. No such luck; I was plagued by the same problems that Ray had already solved. A couple of hours of circuit tracing were unproductive, but I did notice that the transformer on my bench supply was getting *very* hot. I checked the voltage with the GPA-B connected and read 4.75 volts. So I quickly built another supply with heftier components. This solved some of the problems.

It turned out that the 74123 did not work at 4.75 volts, but worked quite well at 5 volts. Unfortunately, the 74165 shift register worked fine at 4.75 volts, but did not work at 5. We replaced it and the board finally drove my model 15. Both of the original ICs worked fine in Ray's setup with unknown voltage, so they seem to be fairly sensitive to such things.

One additional modification is shown on the diagram: the 555 pulse stretcher shown connected to the flag. It turns out that my Sanders keyboard (ironically, purchased from MMM) did not have a long enough flag to trigger the GPA-B circuit. This pulse stretcher is optional, but it probably can't hurt.

### Up and Running—Finally

Now that the board is operating, it is an acceptable, although not perfect, solution to the problem of driving a Baudot teleprinter with ASCII. One of the problems is in the design—space is always in figures shift. This means that the machine shifts and unshifts between each word—a less than ideal occurrence. It also sometimes shifts but fails to space when a space is received. If the board is run continuously for very long it gets

| ASCII Symbol | Bits | Baudot Bits | Communications | CCIT | Bell |
|---|---|---|---|---|---|
| A | 1000001 | 00011 | A | A | A |
| B | 1000010 | 11001 | B | B | B |
| C | 1000011 | 01110 | C | C | C |
| D | 1000100 | 01001 | D | D | D |
| E | 1000101 | 00001 | E | E | E |
| F | 1000110 | 01101 | F | F | F |
| G | 1000111 | 11010 | G | G | G |
| H | 1001000 | 10100 | H | H | H |
| I | 1001001 | 00110 | I | I | I |
| J | 1001010 | 01011 | J | J | J |
| K | 1001011 | 01111 | K | K | K |
| L | 1001100 | 10010 | L | L | L |
| M | 1001101 | 11100 | M | M | M |
| N | 1001110 | 01100 | N | N | N |
| O | 1001111 | 11000 | O | O | O |
| P | 1010000 | 10110 | P | P | P |
| Q | 1010001 | 10111 | Q | Q | Q |
| R | 1010010 | 01010 | R | R | R |
| S | 1010011 | 00101 | S | S | S |
| T | 1010100 | 10000 | T | T | T |
| U | 1010101 | 00111 | U | U | U |
| V | 1010110 | 11110 | V | V | V |
| W | 1010111 | 10011 | W | W | W |
| X | 1011000 | 11101 | X | X | X |
| Y | 1011001 | 10101 | Y | Y | Y |
| Z | 1011010 | 10001 | Z | Z | Z |
| SP * | x100000 | 00100 | SP | SP | SP |
| ! a | x100001 | 01101 | ! | □ | 1/4 |
| " b | x100010 | 10001 | " | + | " |
| # c | x100011 | 10100 | # | ▨ | # |
| $ d | x100100 | 01001 | $ | ▩ | $ |
| & f | x100110 | 11010 | & | ▤ | & |
| ' g | x100111 | 01011 | ' | BELL | ' |
| ( h | x101000 | 01111 | ( | ( | 1/2 |
| ) i | x101001 | 10010 | ) | ) | 3/4 |
| , l | x101100 | 01100 | , | , | 7/8 |
| - m | x101101 | 00011 | - | - | - |
| . n | x101110 | 11100 | . | . | . |
| / o | x101111 | 11101 | / | / | / |
| ø p | x110000 | 10110 | ø | ø | ø |
| 1 q | x110001 | 10111 | 1 | 1 | 1 |
| 2 r | x110010 | 10011 | 2 | 2 | 2 |
| 3 s | x110011 | 00001 | 3 | 3 | 3 |
| 4 t | x110100 | 01010 | 4 | 4 | 4 |
| 5 u | x110101 | 10000 | 5 | 5 | 5 |
| 6 v | x110110 | 10101 | 6 | 6 | 6 |
| 7 w | x110111 | 00111 | 7 | 7 | 7 |
| 8 x | x111000 | 00110 | 8 | 8 | 8 |
| 9 y | x111001 | 11000 | 9 | 9 | 9 |
| : z | x111010 | 01110 | : | : | 1/8 |
| ; | x111011 | 11110 | ; | = | 3/8 |
| ? | x111111 | 11001 | ? | ? | 5/8 |
| CR | 0001010 | 01000 | CR | CR | CR |
| LF | 0001010 | 00010 | LF | LF | LF |
| BELL f | 0000111 | 00101 | BELL | ' | BELL |

*X = 1 or 0

*Fig. 2. ASCII-Baudot conversion table.*

very hot, and sometimes prints garbage. All of that TTL draws a fair amount of current; you could almost fry eggs over the PROMs.

Photo 2 shows the result of the project—a model 15 that works in parallel ASCII. Fig. 2 shows the ASCII/Baudot translation performed by the board. For those who are unfamiliar with Baudot teleprinters, the reason for the three columns is that there are a number of different sets of figures symbols for Baudot teletypewriters. The three illustrated are the most common (aside from the weather keyboard, which contains numerous meteorological symbols).

The communications style is the unofficial standard and the style the GPA-B was designed to match. The CCIT is probably the most common commercial keyboard around at present. The Bell keyboard is found on old Telex machines. When assembling the kit, be sure to put the PROM labeled "L" (Letters) on the top of the board and the PROM marked "F" (Figures) underneath it. This is not mentioned in the directions.

**Conclusion**

Ray has three boards from MMM (two of them from a PC-board supplier who no longer makes things for MMM). He got them a couple of years ago and has no complaints about the quality. None of these have been powered up yet, so I can't comment on how well they work.

I don't really enjoy troubleshooting hardware, especially when the problems come from other people's mistakes and carelessness. Had I worked overtime for the amount of time I spent on this kit, I could have bought another computer and done the conversion in software.

Considering delivery, quality and design, I shall let you guess when I will buy anything more from Mini Micro Mart. Of course, if the winter is cold enough, you never know what might freeze over! ■

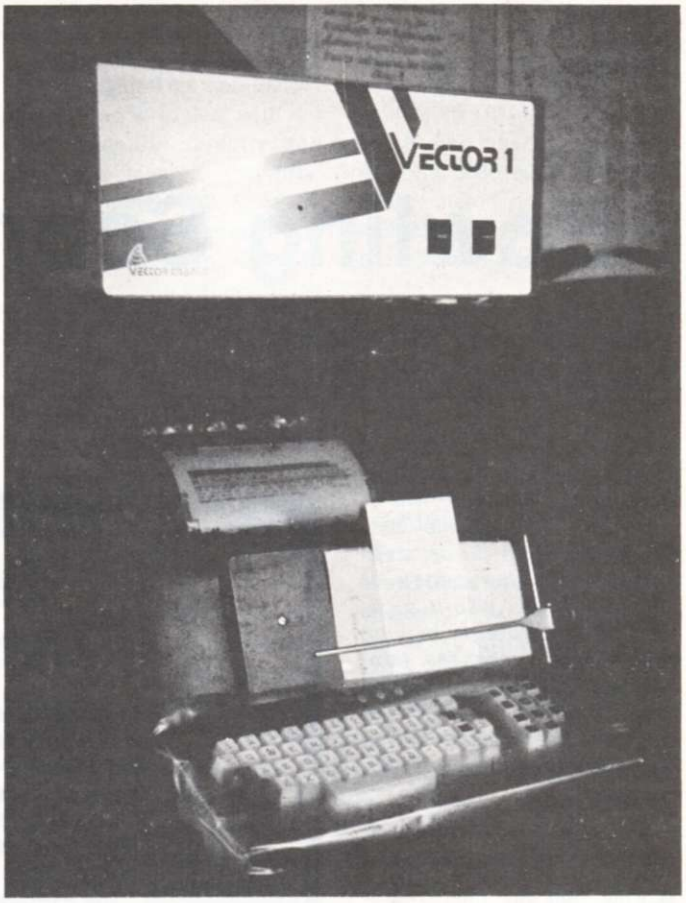



*Photo 2. The final result.*

*Joe Roehrig*
*Box 74*
*Middle Village NY 11379*

# Bowling Scores for Dollars

*There's money to be made from this sport— and not just on the "bowling for bucks" TV shows. This article explains how.*

*Wow, what a package! This is very likely one of the best money-making ideas ever presented in the pages of* Kilobaud. *If you give it a go, we'd be interested in hearing about your successes (and failures). Joe Roehrig will be happy to provide a copy of the program if you send him a blank minidiskette and $5. —John.*

Now that my micro data-processing system has been up and running for a year, it is time for the system to earn its keep and generate profits. Computers are specialists in handling large volumes of figures in a fast, accurate manner. Thus, a small businessman with a DP system should search for an application that processes data for a third party . . . saving the third party money while generating profits for the entrepreneur.

For the last 20 years, I have been bowling or scoring in bowling leagues. Up until a few years ago, scorekeepers manually tallied the scoring sheets for 50 to 75 cents per team per night. With 20 teams in a league, 36 weeks of bowling a year and a 50-cent rate, this fee comes to a modest $360 annually.

However, if a bowling center has three leagues per day, seven days per week, the gross scoring revenues come to $7560 per year. Taking this one step further, $151,200 worth of revenue could be generated by

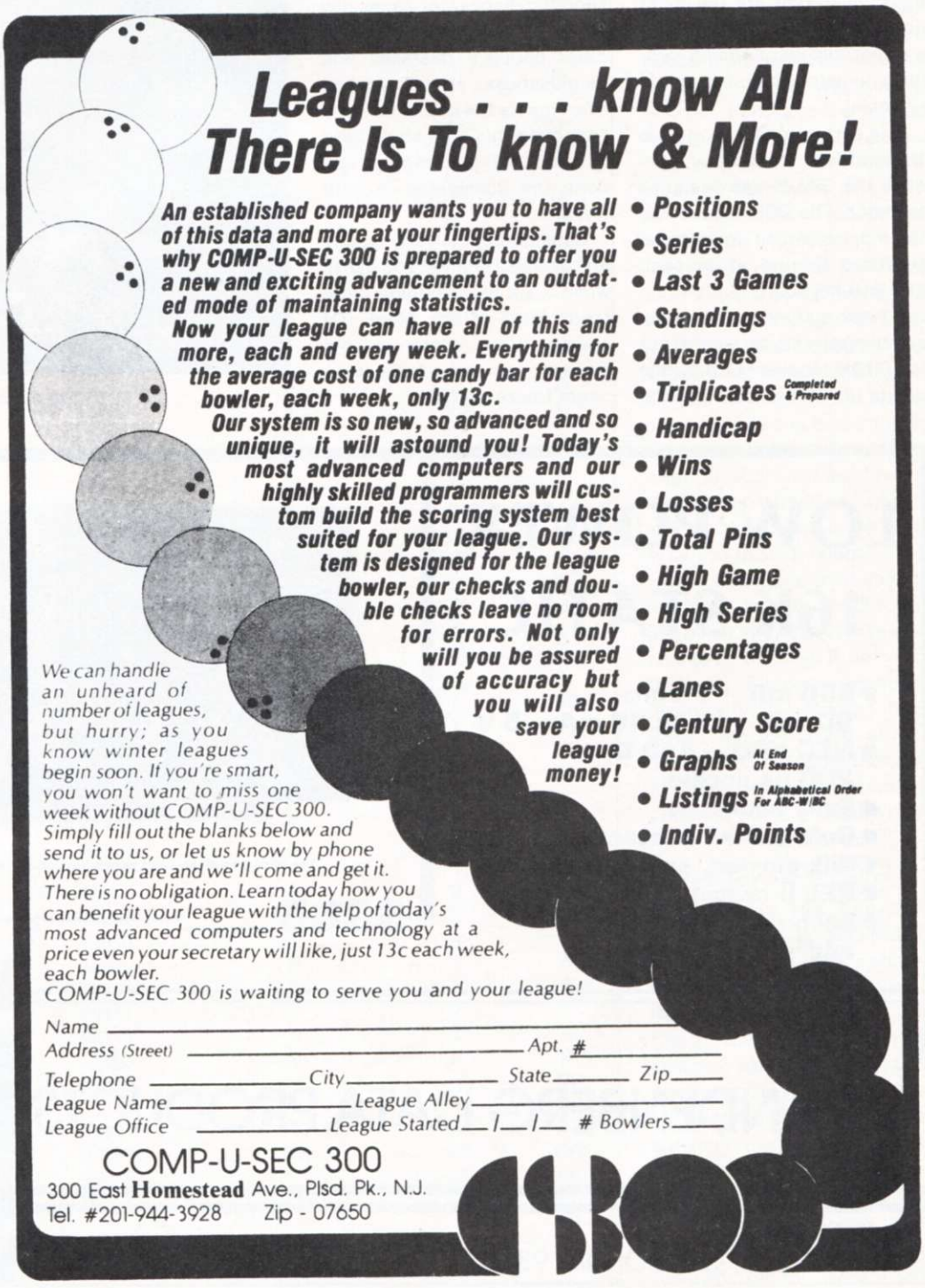



*Fig. 1d.*

Dear Manager,

We are happy to announce the start of a new service for bowling centers that can increase your revenue and facilitate the organization of bowling leagues within your center.

The service being offered is computerized statistics for your bowling league. Normally, this function is performed manually by a person who charges the league between 50¢ to 75¢ per team per week. Our service will be provided to the league through you at a cost of 35¢ per team per week. You, as the manager of a bowling center, can bill the league whatever amount you feel is appropriate. Assuming you charge the league 75¢ per team per week and have three leagues per day with 20 teams each, you would be increasing revenues by $315 per week. Your profit would be $168 and $147 would be paid for our service. We used a 75¢ price to the league in our calculation since our service is superior to anything done manually.

What will you have to do? Very little. The following is a step-by-step outline of the easy procedure:

1. Return the enclosed postcard telling us how many leagues you want us to service and approximately how many teams in each.
2. We will send you forms to give out on opening night of your league. The bowlers will enter their names and average on these sheets. (After opening night your bowlers will never enter their names on a score summary again . . . our computer will print this information for the remainder of the season.)
3. Fill out a simple form describing the league's rules or send us the league rules. Give us a schedule or ask us to formulate a schedule.
4. Collect the forms.
5. We will collect the forms weekly from you for computer processing or supply an envelope for their mailing to us depending on your volume. At time of collection you will be required to pay for one week's data processing (35¢ per team).
6. The weekly statistics will be delivered to you.

We will attempt to verify all ABC averages and will prepare a year-end list of averages for ABC filing.

Regarding our background, the person assigned to this project has been involved in bowling, bowling scorekeeping and league statistics for nearly 20 years. We can handle any size league, any number of players per team and any set of rules. If you ever become dissatisfied with our service, you can terminate immediately and we will supply you all our data on your league.

Attached is a very short sample of a four-team league and a copy of a weekly score sheet to be filled out by the teams. These are just samples, and actuals will be on four-part paper.

If you want more details, give us a call or just return the enclosed postcard.

Sincerely,

Joseph J. Roehrig

*Fig. 1a.*

servicing 20 bowling centers.

The problem now is marketing the idea. The approach I recommend is to deal with the bowling center directly and share the revenue fifty-fifty. Why so generous? Because without the cooperation of the center to recruit leagues and to collect scores for the scoring service, there would be no revenues at all. Fig. 1a shows a sample letter that could be sent to numerous bowling centers. A second approach, advertising in bowling trade publications, is shown as Fig. 1d. A person with time to visit the bowling centers could do person-to-person marketing.

After reviewing Fig. 1 and thinking about your share of $151,200, you should be ready to see how your new business runs, assuming your marketing has been successful and the customers are clamoring for your service. (The tools I use for processing bowling scores are an Imsai 8080, North Star Disk, 24K of memory, DECwriter, an ADM3, North Star BASIC and the software contained in this article.)

The first question is how to store the data. Initially, my North Star Disk was generating a lot of hard-disk errors (a hardware problem later corrected by the local computer store). Therefore, the software design kept disk reads and writes to a minimum during program executions by assigning an area of memory exclusively for examine and fill (peek and poke in Altair BASIC) operations.

This article will not deal with the detailed design of the software, but will show the reader exactly how to use the programs presented.

*Fig. 1. The key to the entire operation lies in your marketing approach. Fig. 1a is a copy of the letter I sent out to various bowling centers. Fig. 1b is a sample of a four-team printout. Fig. 1c is a sample weekly score sheet to be filled out by the teams. Fig. 1d is a sample of another approach—an advertisement in a bowling publication (*The Sports Reporter*).*

NYAC MIXED 01/14/78

| POS | TEAM | | WON | LOST | WIN - LOSS PERCENTAGE | HIGH GAME | HIGH SERIES | AVERAGE | TOTAL WOOD |
|---|---|---|---|---|---|---|---|---|---|
| 1 | RED | ( 1) | 9.0 | 5.0 | .643 | 738 | 2112 | 677.00 | 4062 |
| 2 | GREEN | ( 3) | 7.0 | 7.0 | .500 | 816 | 2448 | 734.50 | 4407 |
| 3 | WHITE | ( 4) | 7.0 | 7.0 | .500 | 817 | 2304 | 762.67 | 4576 |
| 4 | BLUE | ( 2) | 5.0 | 9.0 | .357 | 773 | 2162 | 655.33 | 3932 |

| NAME | ENT | #G | H-G | H-S | WOOD | AVE | NAME | ENT | #G | H-G | H-S | WOOD | AVE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 RED | | | | | | | 2 BLUE | | | | | |
| HARCOHE | 140 | 6 | 110 | 330 | 642 | 107.0 | GRANT | 140 | 6 | 111 | 333 | 603 | 100.5 |
| SANDY | 135 | 6 | 121 | 360 | 701 | 116.8 | TORRE | 135 | 6 | 120 | 342 | 642 | 107.0 |
| BROTHMAN | 110 | 6 | 132 | 390 | 769 | 128.2 | MILLAN | 110 | 6 | 132 | 381 | 711 | 118.5 |
| JOELLY | 170 | 6 | 211 | 538 | 958 | 159.7 | BERRA | 170 | 6 | 220 | 541 | 961 | 160.2 |
| SIMMON | 180 | 6 | 181 | 542 | 992 | 165.3 | MARTIN | 180 | 6 | 190 | 565 | 1015 | 169.2 |
| | 3 GREEN | | | | | | | 4 WHITE | | | | | |
| MANTLE | 140 | 5 | 131 | 393 | 584 | 116.8 | HILL | 140 | 6 | 150 | 416 | 806 | 134.3 |
| MORRIS | 135 | 6 | 141 | 423 | 756 | 126.0 | JORDON | 135 | 6 | 143 | 426 | 755 | 125.8 |
| ROTH J | 110 | 6 | 151 | 453 | 794 | 132.3 | LIPSON C | 110 | 6 | 160 | 456 | 858 | 143.0 |
| ROTH G | 170 | 6 | 171 | 513 | 975 | 162.5 | LIPSON L | 170 | 6 | 233 | 603 | 1089 | 181.5 |
| LACEY | 180 | 6 | 222 | 666 | 1188 | 198.0 | KALIN | 180 | 6 | 192 | 554 | 1068 | 178.0 |

HIGH SCORES

| POS | NAME | AVE | NAME | H-S | NAME | H-G |
|---|---|---|---|---|---|---|
| 1 | LIPSON C | 143 | LIPSON C | 456 | LIPSON C | 160 |
| 2 | HILL | 134 | ROTH J | 453 | ROTH J | 151 |
| 3 | ROTH J | 132 | JORDON | 426 | HILL | 150 |
| 1 | LACEY | 198 | LACEY | 666 | LIPSON L | 233 |
| 2 | LIPSON L | 181 | LIPSON L | 603 | LACEY | 222 |
| 3 | KALIN | 178 | MARTIN | 565 | BERRA | 220 |

*Fig. 1b.*

### The Software Package

The first program in the bowling package sets up the bowlers' names, averages, sex and status (discussed later), which must be entered on a disk. The programs can handle any combination of up to 210 bowlers playing on as many as 30 teams. To avoid hard-disk error, this information was entered on a disk by writing a nine-line program (excluding data) that uses the bowling information as data statements. Thus, the North Star BASIC edit functions can be used to enter data and execute the program over and over again after it is written until a disk is properly prepared. Listing 1 shows this program.

Variable Z in line 1 represents the teams (1 to 30); Z1 shows the number of players on each team (any number as

```
LANES 5 6        4 WHITE                                  1 RED        WEEK # 3

NAME      AVE = 1ST = 2ND = 3RD = TOT = NAME      AVE = 1ST = 2ND = 3RD = TOT
==========================================================================
HILL     -135 =     =     =     =     = HARCOHE -135 =     =     =
--------------------------------------------------------------------------
JORDON   -130 =     =     =     =     = SANDY   -130 =     =     =
--------------------------------------------------------------------------
LIPSON C-143 =     =     =     =     = BROTHMAN-128 =     =     =
--------------------------------------------------------------------------
LIPSON L-181 =     =     =     =     = JOELLY  -165 =     =     =
--------------------------------------------------------------------------
KALIN    -178 =     =     =     =     = SIMMON  -175 =     =     =
--------------------------------------------------------------------------
==========================================================================
--------------------------------------------------------------------------
         TOT.=     =     =     =     =-         TOT.=     =     =     =
--------------------------------------------------------------------------
WON      HAND=     =     =     =     =-WON      HAND=     =     =     =
--------------------------------------------------------------------------
LOST     TOT.=     =     =     =     =-LOST     TOT.=     =     =     =
==========================================================================
```

*Fig. 1c.*

```
LANES 1 2        1 TIGERS                                 2 LIONS      WEEK # 1

NAME      AVE = 1ST = 2ND = 3RD = TOT = NAME      AVE = 1ST = 2ND = 3RD = TOT
==========================================================================
SUE L.   -117 = 106 =  78 =  94 = 278 = ANGELA  -110 = 116 = 114 =  74 = 304
--------------------------------------------------------------------------
MAGGIE S-140 = 128 = 131 = 178 = 437 = RALPH S.-140 = 104 = 121 = 123 = 348
--------------------------------------------------------------------------
FRANK P.-140 = 104 = 153 = 134 = 391 = RENEE L.-125 = 127 = 107 =  99 = 333
--------------------------------------------------------------------------
BILL L. -131 = 132 = 156 = 134 = 422 = JOHN     -123 = 113 ABS = 113 = 113 = 339
--------------------------------------------------------------------------
CHARLIE -144 = 139 = 106 = 128 = 373 = TED R.   -144 = 182 = 178 = 156 = 516
--------------------------------------------------------------------------
==========================================================================
         672 =     =     =     =     =-         642 = 642 = 633 = 565 =
--------------------------------------------------------------------------
WON   1  HAND=   0 =   0 =   0 =     =-WON   5  HAND=  27 =  27 =  27
--------------------------------------------------------------------------
LOST  5  TOT.= 609 = 624 = 668 =     =-LOST  1  TOT.= 669 = 660 = 592 =
==========================================================================
```

*Fig. 4. Filled-in data-entry sheet.*

```
FILE :MIX1
DATE :10/03/77
WEEK :2
LANES :1
ON LANE 1 ? 4
ON LANE 2 ? 6
ON LANE 3 ? 2
ON LANE 4 ? 5
ON LANE 5 ? 1
ON LANE 6 ? 3
 4 6 2 5 1 30 TO FIX ? 4
SET ?
```

*Fig. 3. Input for program run 1.*

**Data Entry**

The data for this program includes the eight-character team name, followed by eight-character players' names coupled to a five-digit number. Lines 5000 to 5003 of Listing 1 show the data for team 1. The five-digit number corresponds to each bowler's average, sex and status. Therefore, the first bowler in line 5000 is "Sue L," who has a 11710 following her name. The number breaks down as follows: 117—bowling average; 1—sex is female (11700 would be a male with a 117 average); 0—status.

Fig. 2 shows how the above program, "Setup 1," runs. For the first execution, a 1 is entered to clear memory location 30470 to 32767, which will be used for the actual scores. A 0 is entered for the next run and long as the number of players does not exceed 210). Variable Z2 is 2 for a mixed league (both male and female players) or 1 for a league with only one sex participating. Z3 represents the number of games required by a status 0 bowler to establish an average; Z4 applies to a status 1 bowler. This allows the flexibility to require established bowlers to bowl a certain number of games to get a revised average, whereas a different number may be required for new bowlers. Variable Z5 sets the maximum number of pins for an established bowler to lower his average.

Note: Some of these variables are included in the bowling package to handle the more complex average calculations required by a few very sophisticated leagues that compete for high-prize money.

```
0 FOR NEW OR 1 TO CLEAR RAM: 1
READY
RUN

0 FOR NEW OR 1 TO CLEAR RAM: 0
FILE :MIX1
READY
```

*Fig. 2. Initialization sequence.*

```
FILE: MIX1
TEAM: 1
SUE L.     SCORES   106,78,94     278?
MAGGIE S   SCORES   128,131,178    437?
FRANK P.   SCORES   104,153,134    391?
BILL L.    SCORES   132,156,134    422?
CHARLIE    SCORES   139,106,128    373?
           SCORES   0,0,0        0?
HANDICAP   0,0,0      0?
BLIND      0,0,0      0?
     609       624       668      1901
WON-LOST   1,5   OK
OK TO UPDATE FILE ?
FILE UPDATED
TEAM: 2
ANGELA     SCORES   116,114,74    304?
RALPH S.   SCORES   104,121,123    348?
RENEE L.   SCORES   127,107,99    333?
JOHN       SCORES   0,0,0        0?
TED R.     SCORES   182,178,156    516?
           SCORES   0,0,0        0?
HANDICAP   27,27,27      81?
BLIND      113,113,113     339?
     669       660       592      1921
WON-LOST   5,1   OK
OK TO UPDATE FILE ?
FILE UPDATED
TEAM: 0
READY
```

*Fig. 5. Entering data from data-entry sheets (provided by various teams).*

the computer asks for a file name. MIX 1 is entered and all the names are written in this disk file.

The main program, called RUN 1 (Listing 2), can now be executed. Fig. 3 shows how this program runs. It asks for the file location of the bowlers' names, the date, the week (1 for first week of bowling, 2 for second, etc.), the league lanes and which team is on each lane. The program prints the lane assignments; an input of 0 lets you change them in case of error. Finally, the program asks "set ?" which allows you to set your paper to line one and push carriage return. Fig. 6 is the printout which follows.

Fig. 3 gives the inputs for week 2. For the first run, you should enter 1 for the week and the computer will generate all blank forms. The blank forms are supplied to the bowling center as input sheets to be used. Fig. 4 shows how this blank form is filled out by the bowlers. The completed form is collected by the center and returned to you for processing.

Listing 3 is the input program —INPUT 1. Fig. 5 shows the data from Fig. 4 being entered into the computer. The program asks for the file name and then the team number. The computer then prints the name of each bowler. Enter the three scores and press return; a total is printed by the computer for verification. This avoids input errors, since the total should agree with the input sheet.

To go to the next bowler push carriage return. Any character except a carriage return will cause the computer to ask for the same bowler's scores again. This allows for the immediate correction of errors.

Note that after Charlie in Fig. 5, scores are still solicited by the computer. This is done because an extra player was put on each team to provide for possible expansion. Next, the computer asks for the handicaps and, finally, for blind scores—scores not actually bowled but assigned to absen-

```
                              NYAC MIXED                          10/03/77
                              ==========

                                                 WIN - LOSS   HIGH    HIGH                    TOTAL
POS TEAM                      WON    LOST       PERCENTAGE   GAME   SERIES    AVERAGE        WOOD
==========================================================================================
  1 LIONS      ( 2)           5.0     1.0          .833       669     1921     640.33         1921
  2 BEARS      ( 3)           5.0     1.0          .833       694     1995     665.00         1995
  3 ROBBERS    ( 6)           5.0     1.0          .833       732     2053     684.33         2053
  4 TIGERS--   ( 1)           1.0     5.0          .167       668     1901     633.67         1901
  5 LANCERS    ( 4)           1.0     5.0          .167       693     1974     658.00         1974
  6 MAN'TANS   ( 5)           1.0     5.0          .167       614     1841     613.67         1841


NAME      ENT  #G H-G H-S  WOOD   AVE  ==  NAME      ENT  #G H-G H-S  WOOD   AVE
==========================================================================================
                1 TIGERS--            ==                    2 LIONS
SUE L.    117   3 106 278   278  92.7  ==  ANGELA    110   3 116 304   304 101.3
MAGGIE S  140   3 178 437   437 145.7  ==  RALPH S.  140   3 123 348   348 116.0
FRANK P.  140   3 153 391   391 130.3  ==  RENEE L.  125   3 127 333   333 111.0
BILL L.   131   3 156 422   422 140.7  ==  JOHN      123   0   0   0     0   .0
CHARLIE   144   3 139 373   373 124.3  ==  TED R.    144   3 182 516   516 172.0
            0   0   0   0     0   .0  ==                0   0   0   0     0   .0
                                      ==
                3 BEARS               ==                    4 LANCERS
MRS. D    110   0   0   0     0   .0  ==  MRS. H.    110   0   0   0     0   .0
RENEE B.  120   0   0   0     0   .0  ==  NELSON H  140   0   0   0     0   .0
SANDY     106   3 138 347   347 115.7  ==  PAT R.     119   0   0   0     0   .0
BILL D    140   0   0   0     0   .0  ==  ELSIE      110   3 125 286   286  95.3
CHUCK R   154   3 204 496   496 165.3  ==  JOE R.     167   3 226 581   581 193.7
            0   0   0   0     0   .0  ==                0   0   0   0     0   .0
                                      ==
                5 MAN'TANS            ==                    6 ROBBERS
CAROL     103   3 112 291   291  97.0  ==  POLLY      110   3 163 391   391 130.3
EILEEN    120   3 128 307   307 102.3  ==  PAT J.     105   3 128 347   347 115.7
KATHY M.  141   0   0   0     0   .0  ==  JOE W.     111   3 140 356   356 118.7
BLIND     140   0   0   0     0   .0  ==  JOE C.     144   0   0   0     0   .0
PAUL S.   155   3 144 400   400 133.3  ==  RICHIE M  155   3 181 437   437 145.7
            0   0   0   0     0   .0  ==                0   0   0   0     0   .0
                                      ==

HIGH SCORES
POS NAME          AVE    NAME          H-S    NAME          H-G
==========================================================================
 1  JOE R.        193    JOE R.        581    JOE R.        226
 2  TED R.        172    TED R.        516    CHUCK R       204
 3  CHUCK R       165    CHUCK R       496    TED R.        182

 1  MAGGIE S      145    MAGGIE S      437    MAGGIE S      178
 2  POLLY         130    POLLY         391    POLLY         163
 3  PAT J.        115    SANDY         347    SANDY         138
```

*Fig. 6. The finished product.*

```
LANES 5 6        1 TIGERS--                          3 BEARS       WEEK #  2

NAME      AVE = 1ST = 2ND = 3RD = TOT = NAME      AVE = 1ST = 2ND = 3RD = TOT
==========================================================================
SUE L.  - 92 =     =     =     =     = MRS. D  -110 =     =     =     =
--------------------------------------------------------------------------
MAGGIE S-145 =     =     =     =     = RENEE B.-120 =     =     =     =
--------------------------------------------------------------------------
FRANK P.-130 =     =     =     =     = SANDY   -115 =     =     =     =
--------------------------------------------------------------------------
BILL L. -140 =     =     =     =     = BILL D  -140 =     =     =     =
--------------------------------------------------------------------------
CHARLIE -124 =     =     =     =     = CHUCK R -165 =     =     =     =
--------------------------------------------------------------------------
          - 0 =    =     =     =     =           - 0 =    =     =     =
--------------------------------------------------------------------------
==========================================================================
          631 =    =     =     =     =-          650 =    =     =     =
--------------------------------------------------------------------------
WON       HAND=  17 =  17 =  17 =     =-WON      HAND=   0 =   0 =   0
--------------------------------------------------------------------------
LOST      TOT.=    =     =     =     =-LOST     TOT.=    =     =     =
==========================================================================
```

*Fig. 7. Updated data-entry form for distribution to team.*

```
290 !TAB(12),%2I,D," ",N1$(V1,V2),TAB(37),"==",TAB(52),%2I,D+1," ",
292 !N1$(V3,V4)
300 FORG=1TOZ1\FORE=0TO1\F=E+D\V1=((F-1)*Z1)+6
310 V2=((V1-1)*6)+B\V4=V1*8\V3=V4-7\V5=EXAM(V2)\V6=EXAM(V2+1)+45
320 Q1=EXAM(V2+2)\Q2=EXAM(V2+3)\GOSUB1000\V7=Q1\V9=INT(A(V1)/100)
325 IFV6=45THENV6=0
330 Q1=EXAM(V2+4)\Q2=EXAM(V2+5)\GOSUB1000\V8=0\IFV5<1THEN340
335 V8=Q1/V5
340 !N$(V3,V4),%4I,V9,V5,V6,V7,%6I,Q1,%6F1,V8,\GOSUB1300
345 GOSUB1460
350 IFE=0THEN!" == ",\NEXT\!""\NEXT\!TAB(37),"==":\L=L+L1
360 IF64>L+L1THEN380
370 E=66-L\FORG=1TOL\!""\NEXT\GOSUB1110\L=4
380 NEXT\L=L+2\!""\!""\IFL<56THEN390
385 L=66-L\FORB=1TOL\!""\NEXT\L=0      AVE    NAME        H-S   ",
390 !"HIGH SCORES"\!"POS NAME
392 !"NAME      H-G"
400 GOSUB1040\FORE=0TOZ2-1\FORF=0TO2\%2I,F+1,"  ",\FORD=0TO2\GOSUB1400
410 !N$(V1,V2),%7I,INT(V3),"    ",\NEXT\!""\NEXT\!""\NEXT
500 L=L+6\IFZ2>0THENL=L+4
510 FORY=1TO66\!""\NEXT
600 FORD=1TOZSTEP2\Q1=P(2,D)\Q3=P(2,D+1)\Q2=Q1*8\Q4=Q3*8
602 GOSUB1630
605 !"LANES",Z7,Z7+1,TAB(15),%2I,Q1," ",N1$(Q2-7,Q2),\Z7=Z7+2
610 !TAB(54),%2I,Q3," ",N1$(Q4-7,Q4),TAB(67),"WEEK #",%3I,Z6\!""
620 !"NAME      AVE  = 1ST  = 2ND  = 3RD  = TOT  = ",
630 !"NAME      AVE  = 1ST  = 2ND  = 3RD  = TOT"
640 GOSUB1040\FORF=1TOZ1\FORG=0TO1\V1=Q1\IFG>0THENV1=Q3
670 V2=((V1-1)*Z1)+F\V3=V2*8
675 K(G)=K(G)+A(V2)
680 !N$(V3-7,V3),"-",%3I,A(V2),"  =      =      =",
690 IFG>0THEN700\!"      = ",\GOTO710
700 !""
710 NEXT\GOSUB1600\NEXT\GOSUB1040
712 IFK(0)>K(1)THEN716
713 K(2)=INT((K(1)-K(0))*.9)\K(3)=0\GOTO718
716 K(3)=INT((K(0)-K(1))*.9)\K(2)=0
718 !%12I,K(0)," ",B$="  ",\GOSUB1700\!%12I,K(1)," ",\GOSUB1710
720 GOSUB1630\GOSUB1600\!"WON      HAND=",%4I,K(2)," =",
722 !%4I,K(2)," =",%4I,K(2)," =      =",
724                    !"WON     HAND=",%4I,K(3)," =",
728 !%4I,K(3)," =",%4I,K(3)," "
740 GOSUB1600\A$="LOST      TOT."\B$="      "\GOSUB1600\GOSUB1040
742 K(0)=0\K(1)=0
750 !""\!""
760 IFD<3THEN800\FORY9=1TO10\!""\NEXT
800 NEXT\END
999 !FREE(0)\END
1000 Q1=(Q2*256)+Q1\RETURN
1040 FORY=1TO76\!"=",\NEXT\!""\RETURN
1110 !""\!""\!"NAME      ENT  #G H-G H-S  WOOD   AVE  == ",
1120 !"NAME      ENT  #G H-G H-S  WOOD      AVE"\GOSUB1040\RETURN
1300 P(0,0)=V8*10\P(0,1)=V7*10\P(0,2)=V6*10
1310 Q3=INT((A(V1)-(V9*100))/10)\FORQ1=0TO2\FORQ2=0TO2
1320 IFP(0,Q1)<(H(Q3,Q1,Q2)/1000)THEN1350
1330 IFQ2>1THEN1340\FORV2=2TOQ2+1STEP-1\H(Q3,Q1,V2)=H(Q3,Q1,V2-1)\NEXT
1340 H(Q3,Q1,Q2)=(INT(P(0,Q1))*1000)+V1\EXIT1360
1350 NEXT
1360 NEXT\RETURN
1400 V3=INT(H(E,D,F)/1000)\V1=H(E,D,F)-(V3*1000)
1410 V2=V1*8\V1=V2-7\V3=V3*.1\RETURN
1460 Q4=A(V1)-(V9*100)-(Q3*10)\Q2=Z2
1470 IFQ4>0THENQ2=Z3\IFV5=>Q2THEN1490
1480 A(V1)=V9\RETURN
1490 V5=INT(V8)\IFV5+Z5>V9THEN1510
1500 A(V1)=V9-5\RETURN
1510 A(V1)=V5\RETURN
```

tee players.

Fig. 4 shows John is absent; ABS was entered next to his scores, all of which are ten pins less than his average (blind scores). In Fig. 5, John's scores are not entered next to his name but are entered as blind. The computer prints the point totals to be compared and then asks for the won and lost points, which are supplied by the bowlers on the input sheet.

A carriage return after OK TO UPDATE FILE? enters the data; anything else erases all the input. A carriage return signifies your approval of each individual input line; anything else allows for the re-input of the line. Team 0 ends the input program. All input need not be done at the same time. However, before the machine is turned off the scores must be saved by copying memory locations 30470 to 32767 to a file. With the North Star Disk System, this is done by:

```
BYE
SF DATA1 7706
```

This assumes file DATA 1 was created. The 7706 is hex for the start of the memory locations to be saved.

Fig. 6 shows the finished product generated by program RUN 1 (Listing 2). Fig. 3 describes the input required for this run. Fig. 7 shows a new blank input form generated by RUN 1. The averages have been updated; a comparison of Team 1's averages to those shown in Fig. 4 illustrates how they've changed. A spelling correction and team names were also added by editing program SETUP 1 and running it again.

Fig. 8 shows the table of contents for my bowling package disk—three copies of each program and two of the data. In this figure, North Star BASIC is referred to as B. My system has 24K placed in addresses 8192 to 32767. Address 2A0A (hex) was changed to 76 (hex) to use addresses 30470 to 32767 for fill and examine instructions. Therefore, GO B and Print Free (0) yield 9105 bytes.

```
BYE
*LI
DOS          4   10    0
B           14   40    1 2A00
SETUP1      54   24    2
SETUP2      78   24    2
SETUP3     102   24    2
INPUT1     126   24    2
INPUT2     150   24    2
INPUT3     174   24    2
RUN1       198   24    2
RUN2       222   24    2
RUN3       246   24    2
DATA1      300   10    3
DATA2      270   10    3
MIX1       280   10    3
MIX2       290   10    3
*SF DATA1 7706
*SF DATA2 7706
*GO B

READY
PRINT FREE(0)
 9105
READY
```

*Fig. 8. Bowling-package directory.*

### Wrap-up

Good luck with this application. Just remember to concentrate on marketing and maybe add some new features to these programs to attain an edge. I have several leagues enrolled now and plan to improve my service. Maybe we will be in competition. ■

*Listing 1. Initial data-base generation.*

```
 1 Z=6\Z1=6\Z2 =2\Z3=3\Z4=0\Z5=0
 3 INPUT"0 FOR NEW OR 1 TO CLEAR RAM: ",A\IFA=1THEN200
 5 IFA<0THENEND\INPUT"FILE :",F$\OPEN#0,F$
10 DIMA$(8)
15 A$="
20 WRITE#0,Z,Z1,Z2,Z3,Z4,Z5
110 FORA=1TOZ\READA$\WRITE#0,A$\FORB=1TOZ1\READA$,C
120 WRITE#0,A$,C\NEXT\NEXT
```

```
130 CLOSE#0\END
200 FORA=30470TO32767\FILLA,B\NEXT\END
5000 DATA"--------","SUE L.  ",11710,"MAGGIE S",14010,"FRANK P.",14000
5002 DATA"BILL L. ",13100,"CHARLIE ",14400
5003 DATA"        ",00000
5004 DATA"--------","ANGELA  ",11010,"RALPH S.",14000,"RENEE L.",12510
5006 DATA"JOHN    ",12300,"TED R.  ",14400
5007 DATA"        ",00000
5008 DATA"--------","MRS. D  ",11010,"RENEE B.",12010,"SANDY   ",10610
5010 DATA"BILL D  ",14000,"CHUCK R ",15400
5011 DATA"        ",00000
5012 DATA"--------","MRS. H. ",11010,"NELSON H",14000,"PAT R.  ",11910
5014 DATA"ELSIE   ",11010,"JOE R.  ",16700
5015 DATA"        ",00000
5016 DATA"MAN'TANS","CAROL   ",10310,"EILEEN  ",12010,"KATHY M.",14110
5018 DATA"BLIND   ",14000,"PAUL S. ",15500
5019 DATA"        ",00000
5020 DATA"--------","BLIND   ",11010,"PAT J.  ",10510,"JOE W.  ",11100
5022 DATA"JOE C.  ",14400,"RIICHIE M",15500
5023 DATA"        ",00000
READY
```

*Listing 2. The main program (called RUN 1).*

```
1 DIMN1$(240),N$(1680),P(2,30),H(1,2,2),N2$(8),A(210),A$(20)
3 LINE 80
5 INPUT"FILE :",F$\OPEN$0,F$
6 READ$0,Z,Z1,Z2,Z3,Z4,Z5\FORA=1TOZ\READ$0,N2$\V2=A*8
8 N1$(V2-7,V2)=N2$\FORB=1TOZ1\READ$0,N2$,C\D=((A-1)*Z1)+B
10 V2=8*D\N$(V2-7,V2)=N2$\A(D)=C\NEXT\NEXT
12 A$="                    "
14 B$="-----"\C$="="
15 A$(1,10)="NYAC MIXED"   \INPUT"DATE :",A$(11,20)\INPUT"WEEK :",Z6
16 INPUT"LANES :",Z7
17 FORD=1TOZ\!"ON LANE",D,\INPUT" ? ",P(2,D)\NEXT
18 FORD=1TOZ\!P(2,D),\NEXT\INPUT"0 TO FIX ? ",E\IFE=0THEN17
20 A=30470\B=30800\C=32060
30 FORD=1TOZ\P(0,D)=D\Q1=A+(11*(D-1))\Q2=EXAM(Q1+1)\Q1=EXAM(Q1)
40 GOSUB1000\P(1,D)=Q1\NEXT
50 Q1=0\FORD=2TOZ\IFP(1,D-1)>=P(1,D)THEN80
60 Q2=P(1,D-1)\Q3=P(0,D-1)\P(1,D-1)=P(1,D)\P(0,D-1)=P(0,D)
70 P(1,D)=Q2\P(0,D)=Q3\Q1=10
80 NEXT\IFQ1>0THEN50
95 G=(Z6-1)*3\INPUT"SET ?",Z$
100 !TAB(30),A$(1,10),TAB(60),A$(11,20)\!TAB(30),"==========="\!""
138 !TAB(33),"WIN - LOSS  HIGH   HIGH                TOTAL"
140 !"POS TEAM             WON   LOST  PERCENTAGE  GAME  SERIES",
150 !"  AVERAGE      WOOD"\GOSUB1040
170 FORD=1TOZ\E=P(0,D)\V2=8*E\V1=V2-7\V3=2+A+((E-1)*11)\Q1=EXAM(V3)
180 Q2=EXAM(V3+1)\GOSUB1000\V4=Q1\IFP(1,D)+V4<1THEN190
185 V5=P(1,D)/(P(1,D)+V4)
190 Q1=EXAM(V3+2)\Q2=EXAM(V3+3)\GOSUB1000\V6=Q1
200 Q1=EXAM(V3+4)\Q2=EXAM(V3+5)\GOSUB1000\V7=Q1
210 Q1=EXAM(V3+6)\Q2=EXAM(V3+7)\Q3=EXAM(V3+8)\GOSUB1000
215 IFQ3ORG=0THEN225
220 V9=Q1+(Q3*65536)\V8=V9/G
225 P(1,D)=P(1,D)*.1\V4=V4*.1
230 !%3I,D," ",N1$(V1,V2)," (",%2I,E,")",%7F1,P(1,D),
240 !%7F1,V4,%9F3,V5,"    ",%6I,V6,%8I,V7,%9F2,V8,%10I,V9
250 NEXT\L=6+Z\IF64>L+4+1+Z1THEN270
260 D=66-L\FORE=1TOD\!""\NEXT\L=0
270 GOSUB1110\L=4\L1=2+Z1
280 FORD=1TOZSTEP2\V2=D*8\V1=V2-7\V3=V1+8\V4=V2+8
1600 !A$,C$,B$,C$,B$,C$,B$,C$,B$,C$,"-",
1610 !A$,C$,B$,C$,B$,C$,B$,C$,B$\RETURN
1620 RETURN
1630 A$="----------"\B$="-----"\RETURN
1700 !   C$,B$,C$,B$,C$,B$,C$,B$,C$,"-",
1705 RETURN
1710 !   C$,B$,C$,B$,C$,B$,C$,B$\RETURN
READY
```

*Listing 3. The data-entry program.*

```
10 DIMN$(1680),A$(8),S(10,6)
15 F1=30470\F2=30800
20 INPUT"FILE: ",F$\OPEN$0,F$
30 READ$0,Z,Z1,Z2,Z3,Z4,Z5\W=Z1+2
40 FORA=1TOZ\READ$0,A$\FORB=1TOZ1\READ$0,A$,C
50 D=((A-1)*Z1)+B\D=D*8\N$(D-7,D)=A$\NEXT\NEXT
55 FORA=0TOW\FORB=0TO5\S(A,B)=0\NEXT\NEXT
60 INPUT"TEAM: ",A\FORB=1TOZ1\D=((A-1)*Z1)+B
65 IFA<1ORA>ZTHENEND\D=D*8
70 !N$(D-7,D),\INPUT1"  SCORES  ",S(B,1),S(B,2),S(B,3)
75 S(B,4)=S(B,1)+S(B,2)+S(B,3)
80 !%6I,S(B,4),\INPUTZ$\IFZ$=""THEN90\GOTO70
90 NEXT
100 INPUT1"HANDICAP  ",S(W,1),S(W,2),S(W,3)\S(W,4)=S(W,1)+S(W,2)+S(W,3)
110 !%6I,S(W,4),\INPUTZ$\IFZ$=""THEN112\GOTO100
112 INPUT1"BLIND     ",S(W-1,1),S(W-1,2),S(W-1,3)
114 S(W-1,4)=S(W-1,1)+S(W-1,2)+S(W-1,3)
116 !%6I,S(W-1,4),\INPUTZ$\IFZ$=""THEN120\GOTO112
120 FORB=1TOW\FORC=1TO4\S(0,C)=S(0,C)+S(B,C)
125 IFC=4THEN150
130 IFS(B,C)>0THENS(B,0)=S(B,0)+1
140 IFS(B,C)>S(B,5)THENS(B,5)=S(B,C)
150 NEXT\NEXT
152 FORB=1TO3\IFS(0,B)>S(0,5)THENS(0,5)=S(0,B)\NEXT
160 FORC=1TO4\!%8I,S(0,C),\NEXT\!""
162 INPUT1"WON-LOST  ",W1,W2\INPUT"  OK",Z$
164 IFZ$=""THEN170\GOTO162
170 INPUT"OK TO UPDATE FILE ?",Z$\IFZ$=""THEN180\GOTO55
180 S(0,0)=W1*10\S(0,2)=W2*10\S(0,6)=S(0,4)\S(0,4)=S(0,5)
190 F3=F1+((A-1)*11)\FORB=0TO6STEP2\C=F3+B\D=F3+B+1
200 L=EXAM(C)\M=EXAM(D)\GOSUB1000
210 IFB>3THEN250
220 N=N+S(0,B)\GOSUB2000\FILLC,L\FILLD,M\GOTO300
250 IFN>S(0,B)THEN300\N=S(0,B)\GOSUB2000
260 FILLC,L\FILLD,M
300 NEXT
310 L=EXAM(F3+8)\M=EXAM(F3+9)\O=EXAM(F3+10)\GOSUB1000
320 N=N+(65536*O)+S(0,6)\O=INT(N/65536)
330 N=N-(O*65536)\GOSUB2000
340 FILLF3+8,L\FILLF3+9,M\FILLF3+10,O
400 F3=((A-1)*Z1*6)+F2\FORB=1TOZ1\F4=((B-1)*6)+F3
410 L=EXAM(F4)\L=L+S(B,0)\FILLF4,L\N=S(B,5)-45
420 L=EXAM(F4+1)\IFL>NTHEN430\FILLF4+1,N
430 L=EXAM(F4+2)\M=EXAM(F4+3)\GOSUB1000
440 IFN>S(B,4)THEN460\N=S(B,4)\GOSUB2000
450 FILLF4+2,L\FILLF4+3,M
460 L=EXAM(F4+4)\M=EXAM(F4+5)\GOSUB1000\N=N+S(B,4)
470 GOSUB2000\FILLF4+4,L\FILLF4+5,M
480 NEXT\!"FILE UPDATED"\GOTO55
1000 N=L+(256*M)\RETURN
2000 M=INT(N/256)\L=N-(256*M)\RETURN
READY
```

*Dr. Jack W. Crenshaw*
*2114 Cecille Dr. SW*
*Huntsville AL 35803*

# It's Here: Machine Language for the TRS-80

*A description of a new monitor program from Radio Shack for the TRS-80.*

Tired of playing blackjack with your TRS-80? Want to get down to serious business? If so, you've probably found that Radio Shack's 4K BASIC keeps you insulated from most of the Z-80's impressive capabilities. Wouldn't it be nice if there were some way to get out of BASIC and into the guts of the machine? Well, there is!

Radio Shack is now offering T-BUG, a 1K monitor program that permits you to use your TRS-80 as a *limited* development system. I have been playing with a copy for about a month now, and can report that it adds a whole new dimension to the usefulness of the machine.

Though I say limited, there is really no limit to what you can do with enough perseverance. However, it is only a 1K monitor, and as such, is roughly comparable to the 6800 MIKBUG, or similar programs. I wouldn't want to develop a compiler with it, but it does open all kinds of doors most of us thought were closed.

T-BUG comes on a cassette and loads like any ordinary tape, using CLOAD. There's one profound difference—when the tape load is finished, you don't get the expected READY. Instead, you get a #, which tells you that you're free from BASIC!

How does this happen? As near as I can tell, each tape has a leader block, which not only contains the usual information, such as the load address and length of record, but also the address at which execution is to begin. For all tapes generated by BASIC, this address is 0000, and thus has the same effect as a reset. On the T-BUG tape, the address is the first instruction of T-BUG, which gets you going.

Now, without going any further into the details of the monitor, we already have some intriguing possibilities—such as writing our own tapes with custom operating systems for special uses, or building a black box which simulates the input of that leader data.

I suppose I knew that there had to be some kind of system like this . . . how else could Radio Shack provide the assembler/editor tapes? Still, it's nice to know how to do it yourself. No, you can't write tapes with your own start address from T-BUG . . . it always starts at its own entry; but now you can get into the machine and make it do things your way!

Note that the CRT display occupies address space. The CRT video generation is totally transparent to the CPU. Just store an ASCII character in a location, and it's displayed. Incidentally, the arrows on the four cursor control keys are displayable as codes 5B through 5E. 5F is the cursor (_).

Note also that the keyboard occupies 256 bytes of address space, and a lot more if you count the unused space around it. This may seem wasteful, but, in my opinion, is the cleverest feature of the whole computer. We have all seen systems with a matrix-type keyboard. Typically, you put out an 8-bit word to an output port, which is latched and strobes the rows of the matrix. The columns are then read into an input port.

Radio Shack has reduced this concept to its bare essentials: There are no ports . . . the keyboard ties right across the

| | |
|---|---|
| M nnnn | Displays the contents of location nnnn in hex. If you wish to alter the contents, merely enter the new data. The monitor will then display the next location, etc. To skip to the next location without entering new data, press ENTER. |
| X | Gets you out of the M mode. It also is used to abort other commands, such as when you type the wrong value for an address. |
| B nnnn | Sets a breakpoint at location nnnn. When control reaches this location, it will revert back to the monitor. The manual says only one breakpoint can be set at once. This is not strictly true . . . you can set as many as you want, but the monitor will only remember the last one. |
| F | Removes a previously set breakpoint. This is where the remembering comes in! |
| R | Displays the contents of the registers. All registers, including the program counter, stack pointer and index registers are displayed. The contents of these registers cannot be altered directly using R. However, it can be done if you need to. |
| P nnnn mmmm | Writes the contents of locations nnnn through mmmm to the cassette. |
| L | Loads a program from cassette. The program loads into the same locations it was written from. |

When I got my copy of T-BUG, the first thing I set out to do was to explore the memory space of the computer. As a result, I was able to draw the memory map shown below:

| | |
|---|---|
| 0000-0FFF | 4K BASIC monitor. |
| 1000-2FFF | Empty ROM sockets for remaining 8K. |
| 2000-37FF | Unused. (Usable???) |
| 3800-3880 | Keyboard. |
| 3881-3BFF | Unused. |
| 3C00-3FFF | CRT display RAM. |
| 4000-7FFF | User RAM. |
| 8000-FFFF | Expansion. |

*Table 1. T-BUG instructions.*

address and data lines. When the keyboard is enabled by a high byte of 38, the low byte serves to strobe the matrix. Only one line of this byte can be high at one time, which explains why the keyboard takes up so much room. Only the addresses 3801, 3802, 3804, 3808, 3810, 3820, 3840 and 3880 are actually used.

Now, here's the great part: When an address line is high and a key in that row is depressed, the key ties the address and data lines together! This drives a data line high, and it is read as a byte with one high bit. In other words, to the CPU, the keyboard looks just like any other RAM. In this case, there is a considerable software overhead for decoding.

| Low Addr → | | 01 | 02 | 04 | 08 | 10 | 20 | 40 | 80 |
|---|---|---|---|---|---|---|---|---|---|
| Data | 01 | @ | H | P | X | 0 | 8 | cr | shft |
| ↓ | 02 | A | I | Q | Y | 1 | 9 | clr | |
| | 04 | B | J | R | Z | 2 | : | brk | |
| | 08 | C | K | S | | 3 | ; | ↑ | |
| | 10 | D | L | T | | 4 | , | ↓ | |
| | 20 | E | M | U | | 5 | - | ← | |
| | 40 | F | N | V | | 6 | . | → | |
| | 80 | G | O | W | | 7 | / | ъ | |

*Fig. 1. Keyboard organization.*

However, RS wisely made the right choices as to where to allocate software and hardware—use software where speed is not an issue (keyboard) and hardware where it is (CRT). The organization of the keyboard is laid out in Fig. 1.

There are a few noteworthy points about this organization. First, the keys are essentially sequential in their ASCII order, so decoding is reasonably straightforward. Second, the shift key is in a column by itself, so it can be used with any other key. I repeat . . . *any* other key, so if you wish you had not only lowercase, but also more special keys, you can get them. Would you believe shift-space bar? Finally, if you *really* want them, there's room in the matrix, if not on the keyboard, for five more keys.

As you can see, I've been having a lot of fun with T-BUG so far; and I've hardly begun to write software! It is a fairly limited monitor, in keeping with its size. If I had written it, I probably would have done things a little differently. And since I now have the capability, that's what I'm doing! ■

*Dr. Mark Boyd*
*St. Mary of the Plains College*
*Dodge City KS 67801*

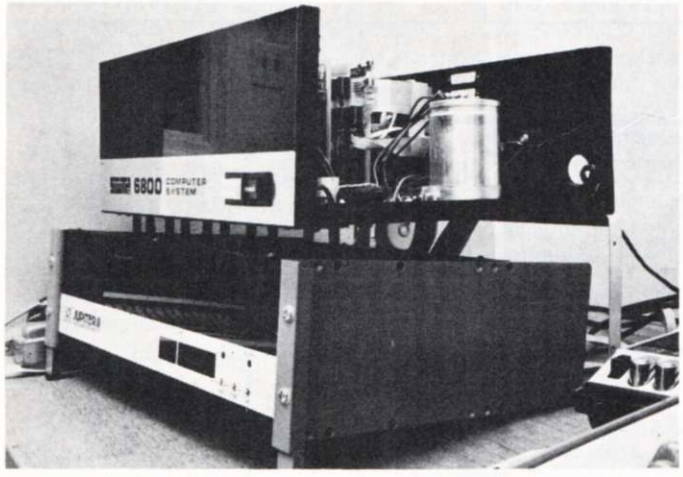
# Two Systems Sharing the Same Bus

*Distributed computing comes to microcomputers with this clever I/O system.*

*I believe this article is a first in computer-hobbyist publications. I don't recall ever seeing an article on interfacing two microcomputers. The technique is rather widespread in the world of the mainframes . . . but I suspect Mark is only one of a handful of people who have done it with personal systems. —John.*

Many computer hobbyists have experienced (or will someday) the ultimate frustration: a full bus. To the experimenter (with good financing) who delights in adding new gadgets to his computer system, a full bus means that new boards can no longer be plugged in with impunity. Now a decision must be made. Which old board will be removed to make room for the latest board?

One alternative to this horrible situation is to buy a new computer for the new boards. Of course, all those neat boards in the old computer can't be used in both computers simultaneously, and a great deal of duplication becomes necessary. A better alternative is to expand the bus of the original computer onto a second bus with its own cabinet, power supply, etc. This solution allows simultaneous use of all the boards, but it is almost as expensive as buying a second computer.




*My two-computer system. The SWTP part of the interface can be seen in the last SWTP slot. The two interface cables run from this board, through separate openings in the back panel, down to the WM part of the interface.*

Why not have the best of both alternatives? Extend your computer's bus by interfacing it to another microcomputer! This really isn't as difficult as it sounds, and it gives you the choice of either two independent computers or one larger computer. The two computers don't even have to be the same type, although things get more complicated with different CPUs. This article describes an interface between two different microcomputers that use the Motorola 6800 CPU, but the same basic ideas could be used to interface any two microcomputers.

### Overview

I interfaced my two computers in an arrangement that allows either fully independent operation or master-slave operation. In the master-slave mode, the master computer takes over the slave-computer's bus and uses the slave's memory and I/O just as if they were on the master's bus. The mode of operation is under software control by the master computer.

My master computer is a Wave Mate Jupiter II. This is a nicely designed and executed system with many useful features. My expansion bus (the slave microcomputer), which doesn't need all the features of the master computer, is an SWTP 6800. Since both computers employ the 6800 CPU chip, much of the software I use with the Wave Mate (WM) can be modified to run on the SWTP. Moreover, the interface between two buses designed for the same CPU chip is only slightly more complicated than an interface between two identical buses.

The SWTP is almost ideal as a slave microcomputer—it is inexpensive, has a wide range of I/O available and is simple and easy to modify for DMA (di-

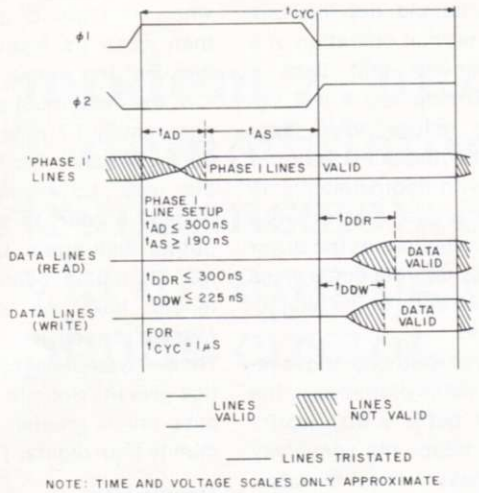



*Fig. 1. Timing diagram for the M6800 microprocessor (from Motorola's* M6800 Microcomputer System Design Data*).*

rect memory access) by halting. (The required modification will be covered later in this article.)

Since both microcomputers use the same CPU, the same basic signals must be used on both buses. While the WM uses a 72-line bus and the SWTP uses a 50-line bus, only 29 lines on the SWTP bus must be controlled by a DMA device. I used 33 distinct connections, including the ground connection, between the two computers.

### Interface Considerations

Before we get into the specific interface design, let's consider how to get the bus signals between the two computers. Overcoming this problem, which is present in any bus extension, is probably the most critical part of the interface design.

Suppose you need to interface two identical motherboards. Why not use a ribbon cable to connect the buses line for line? This seems to be a simple, inexpensive way to extend the bus. Unfortunately, unless the cable is very short, it will not work.

The wires in a ribbon cable are too close together and have too little surface area to use as a computer bus. As a result, the signals on one line will induce noise on the other lines. The faster the signals on the lines change state, the larger the current flow in the lines, and the longer the lines, the greater the induced noise problems. If the noise is enough to change the logic levels on some of the lines, chaos results.

The 6800 CPU synchronizes the changes in level of the address lines, VMA (valid memory address) and R/W (read/write) with the phase 1 clock signal. All these lines may change state only during the first part of the machine cycle. The data lines are synchronized with the phase 2 clock signal. They change state during the second part of the machine cycle. See Fig. 1 for a diagram of these timing relationships.

The noise induced by the changes in the lines synced by the phase 1 clock is not likely to interfere with normal computer operation. Fig. 1 shows the signals on the lines synced by phase 1 are not valid when these changes occur, and the data lines, synced by phase 2, are in a high-impedance state. Unless the noise pulses cause ringing in the lines or are large enough to damage the ICs attached to the lines, they will not upset the operation of the computer.

When the data lines change state, noise may be induced on the lines synced by the phase 1 clock. The signals on these (control and address) lines must be valid during the last part of phase 1 and all of phase 2. Since the data lines change state during the first part of phase 2, noise may be induced on the phase 1 lines at a time when they must be valid. If they're large enough, these noise pulses can really mess up the computer's operation.

### The Ribbon Cable

Several approaches can be used to minimize noise between the lines of the ribbon cable. Because noise problems can be extremely difficult to track down and fix in an existing design, I used an overkill approach, involving three different methods to minimize the noise in the ribbon cable.

1. Since the induced noise falls off rapidly as the conductors are moved apart, I separated the data lines from the other bus lines. I used two cables to carry the signals between the computers with spacers to keep them apart. A 26-conductor ribbon cable carries all of the phase 1 synced signals and a 16-conductor ribbon cable carries the data lines.

2. I partially shielded the data lines by alternating them with ground lines in the 16-conductor cable. This cuts down on the amount of noise radiated by these lines. These eight ground lines are the only ground connections between the computers. This probably would not work if the ribbon cable had to carry the actual bus signals, but, as we shall see, it does not.

3. I used drivers and receivers on all lines running between the two computers, except for the ground lines and one line used to initialize the halt control circuitry. This means that the ribbon cable does not need to carry much current, and that the current-limiting resistors can be used to limit the rate of change of the signals on the cable. It also means that the ground connection between the computers carries little current.

The ribbon cable conductors look, to the drivers, like a complicated network of capacitance, inductance and resistance. To change a line from its low level to its high level, its

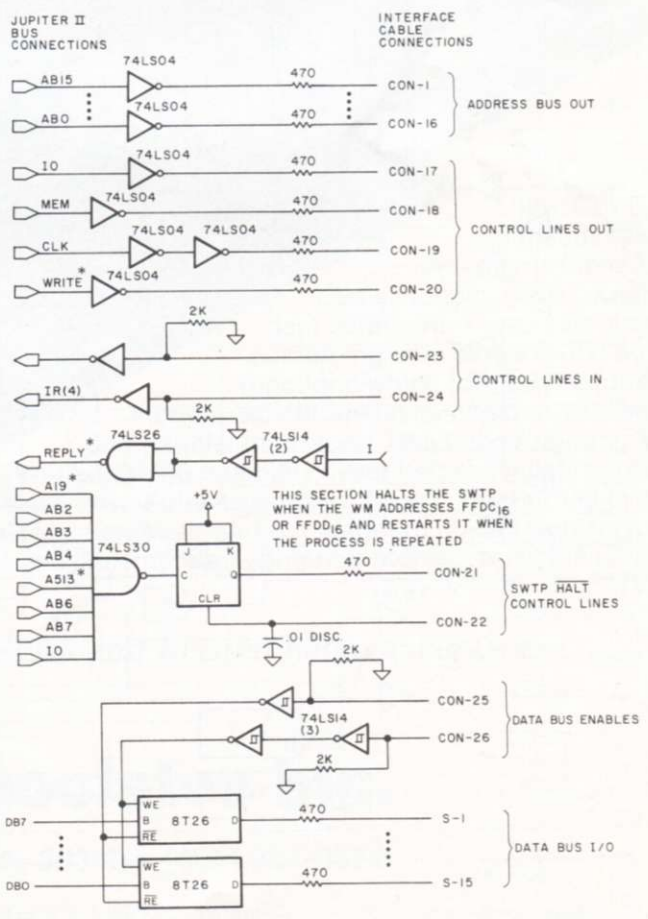



*Fig. 2. Wave Mate Jupiter II to SWTP 6800 interface: WM section.*

equivalent capacitance must be charged by the driver. To change it from high to low, this same capacitance must be discharged. The resistance and inductance in the line, as well as the output current capabilities of the driver, determine the rate at which the line can be charged or discharged. For high-current drivers (e.g., 8T26s), and moderate lengths of cable, the time constant for charging or discharging a line is a few nanoseconds (ns) or less.

Microcomputer bus lines do not need to change state in ns. Typical allowed setup times for bus lines on the 6800, using a 1 MHz clock, are 150 to 200 ns. Since noise problems increase with increasing rates of change on the bus lines, it makes sense to slow the lines down to the rate needed by the computer system.

Current-limiting resistors can be connected between the drivers and ribbon-cable conductors to limit the rate lines can be charged or discharged. The receivers I used require less than .2 mA of current flow in the ribbon cable for proper switching. This means that if a .5 volt drop is the maximum allowed across the resistor, the resistor could be as large as 2.5k.

Since typical ribbon cable has an interwire capacitance of 10 to 20 pF per foot, the capacitance seen by the drivers will be on the order of 20 pF. A 1k resistor in series with a 20 pF capacitor will allow the capacitor to charge to 98 percent of any voltage applied across the combination in less than 100 ns. This should not interfere with the normal operation of a microcomputer that uses a 6800 CPU chip, but it will significantly reduce noise problems in the ribbon cable. In my design, with approximately 18 inches of ribbon cable, I use 470 Ohm resistors at the driver end of all unidirectional lines and at both ends of the bidirectional lines.

The three methods above reduce the noise produced in the bus lines, but it is also worthwhile to reduce the sensitivity of the phase 1 lines to noise. Since we are using a driver-receiver scheme, we can use receivers that have a high noise immunity. TTL Schmitt trigger input inverters are ideal for this purpose.

The TTL Scmitt trigger has a different switching threshold when its input is going high than when its input is going low. For the output to switch low, the input must exceed approximately 1.7 volts. Once the output has switched low, the input must go below approximately .8 volts to switch the output high again. Unlike normal TTL inputs, Schmitt trigger inputs have no disallowed range between low and high. These hysteresis characteristics give the Schmitt trigger inputs much greater noise immunity than regular TTL inputs.

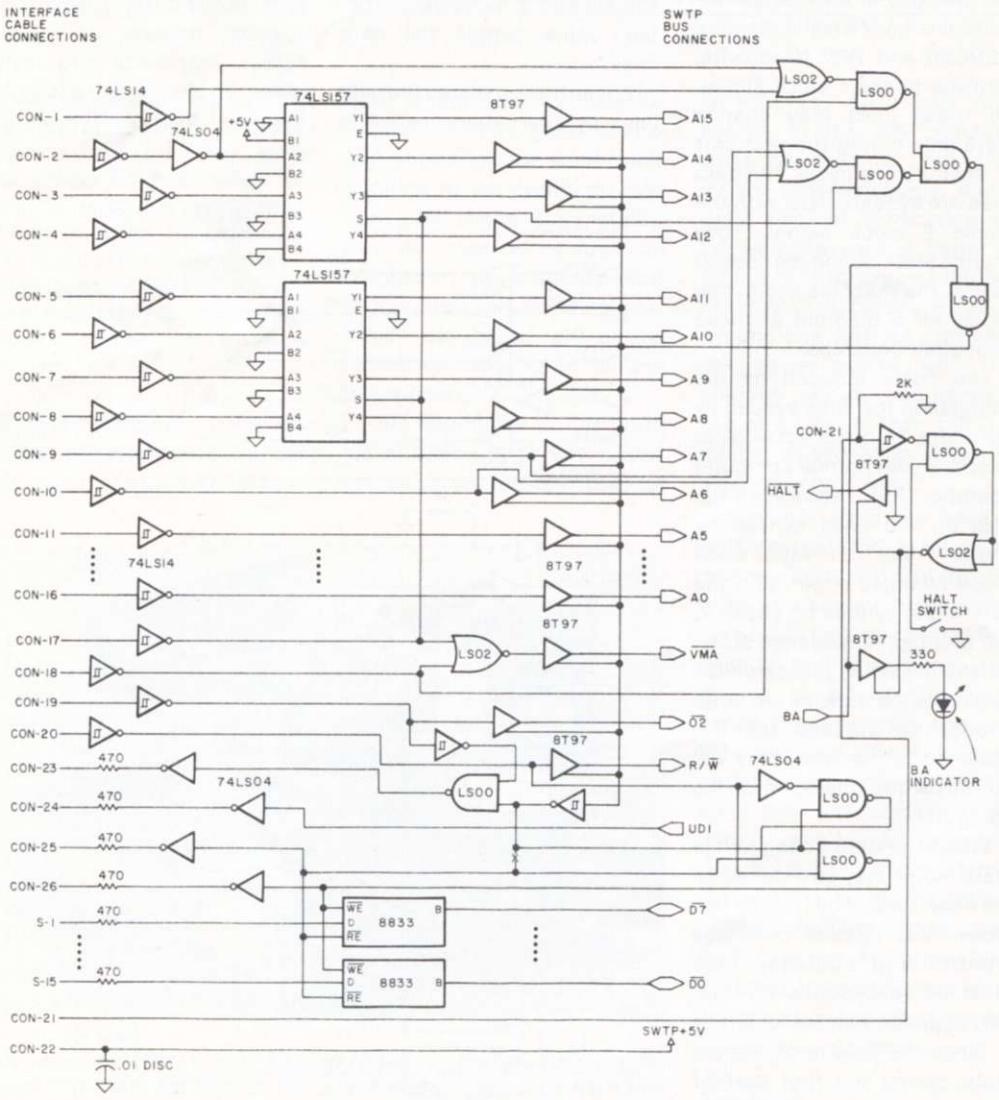



*Fig. 3. Wave Mate Jupiter II to SWTP 6800 interface: SWTP section.*

### The Circuitry

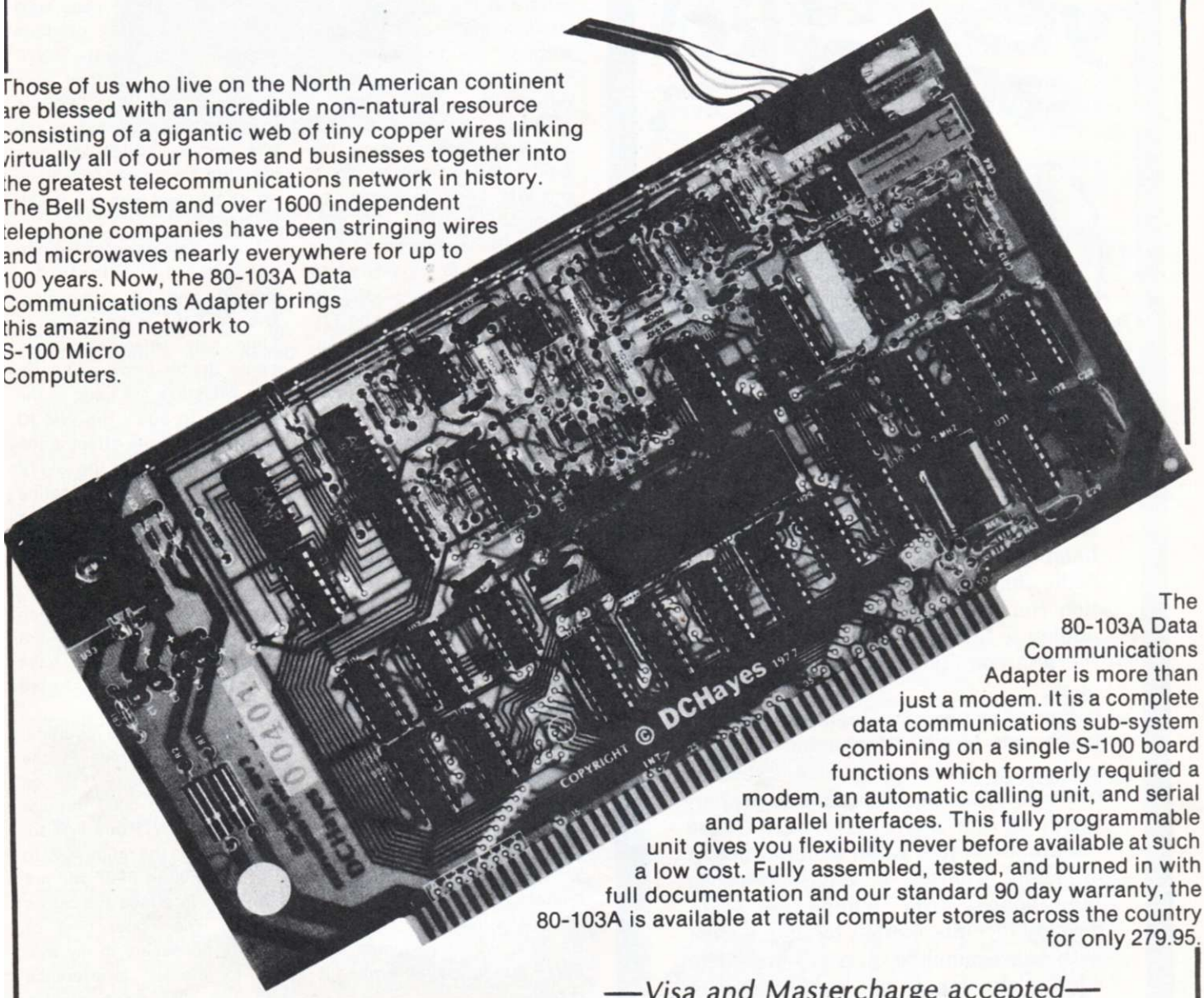
Now let's look at the Wave Mate to SWTP interface design. The schematics for the two parts of the interface are given in Figs. 2 and 3. Fig. 2 shows the logic located in the WM and Fig. 3 illustrates the logic in the SWTP. The two connecting cables are denoted by Con (the 26-conductor cable) and S (the 16-conductor cable). I haven't shown the ground connections (every even-numbered line in S) or the repeated circuitry in the address lines and data lines.

Most of the logic to convert the WM bus signals to the SWTP bus signals is located in the SWTP. This is because only half of a WM board was available for use in the interface. In any case, it makes sense to put as much interface as possible in the slave computer where it will use power only when the slave is in use.

In addition to reducing noise problems, the driver, current-limiting resistor, receiver design also allows simple circuitry for independent use of the computers. When one computer is unpowered, the current-limiting resistors prevent excessive current flow from the drivers in the other computer. For fully independent operation of the two computers with the interface, 2k pull-down resistors were installed on four of the 74LS14 inputs on the WM board and one on the SWTP board. These resistors prevent the interface from interfering with the operation of either computer when the other computer is not powered or the interconnecting cables are dis-

# **modem** / 'mo • dəm / [**mod**ulator + **dem**odulator] *n* - *s* : a device for transmission of digital information via an analog channel such as a telephone circuit.

Those of us who live on the North American continent are blessed with an incredible non-natural resource consisting of a gigantic web of tiny copper wires linking virtually all of our homes and businesses together into the greatest telecommunications network in history. The Bell System and over 1600 independent telephone companies have been stringing wires and microwaves nearly everywhere for up to 100 years. Now, the 80-103A Data Communications Adapter brings this amazing network to S-100 Micro Computers.




The 80-103A Data Communications Adapter is more than just a modem. It is a complete data communications sub-system combining on a single S-100 board functions which formerly required a modem, an automatic calling unit, and serial and parallel interfaces. This fully programmable unit gives you flexibility never before available at such a low cost. Fully assembled, tested, and burned in with full documentation and our standard 90 day warranty, the 80-103A is available at retail computer stores across the country for only 279.95.

*—Visa and Mastercharge accepted—*

## D.C. Hayes Associates Inc.

**P.O. BOX 9884 • ATLANTA, GA. 30319 • (404) 231-0574**

Distributed in Canada by TRINTRONICS LIMITED, Toronto

**H20**

connected.

The interface uses 26 unidirectional lines to carry information from one computer to the other. Twenty-one are used to supply information to the SWTP. The other five lines carry information to the WM from the SWTP. We will discuss these lines in more detail later.

The bidirectional lines (data lines) use 8T26 inverting bus transceivers at the WM end and 8833 noninverting bus transceivers at the SWTP end. The interface must invert the data between the two buses because only the SWTP uses inverted data on its main bus.

This is a good time to mention some points about the logic notation used by SWTP and WM. Logic levels can be defined in two ways—TTL high is usually considered as true and TTL low is considered as false; it is equally valid to define TTL low as true and TTL high as false. Using this negative, or inverted, logic changes the functions of most logic gates (e.g., a NAND gate becomes a NOR gate when the logic definitions are inverted).

Since it is convenient to mix both true and inverted logic definitions in any complicated logic design (it makes the design simpler), some convention must be followed in naming the lines to indicate their logic definition. SWTP uses an overbar to indicate that TTL low is defined as true; WM uses a superscript * for the same purpose. WM data lines are called DB0-DB7, indicating high level true, while SWTP data lines are called $\overline{D0}$-$\overline{D7}$, low level true. Other examples are the WM WRITE* line, which write enables the memory and I/O when it is low, and the SWTP BA line, which indicates the SWTP bus is available when it is high.

Several control signals from the WM bus must be modified before being placed on the SWTP bus. CLK, the WM phase 2 clock signal, must be inverted to serve as $\overline{\phi 2}$ on the SWTP bus. MEM and IO, used by WM for enabling memory and I/O devices, must be combined (using a NOR gate) to provide $\overline{VMA}$, which the SWTP bus uses for both purposes.

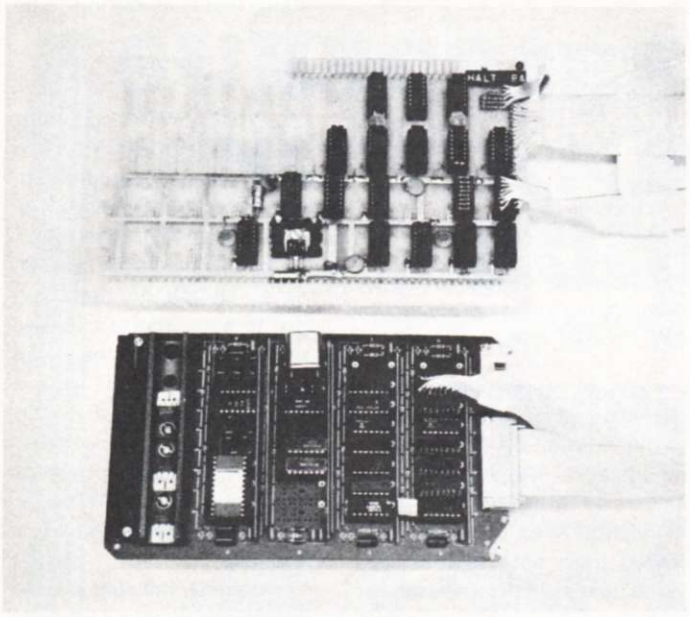
**Memory and I/O Access**

It is convenient to have the interface modify both memory and I/O port addresses. This allows both computers to see their own memory starting from 0000 and, in my setup, extending to 3FFF. The WM, however, sees the SWTP memory from 4000 to 7FFF and uses locations FF00 to FFDF for output ports, while SWTP uses 8000 through 801F. The interface allows it to address the SWTP output ports as if they were located from FF00 to FF1F on the WM bus. This allows the WM monitor, which assumes I/O ports are located from FF00 up, to be used with the SWTP I/O devices.

I used data-selector logic to get different offsets for memory and I/O addressing. The 74LS157 quad 2-line to 1-line data selectors act as 4-pole double-throw switches. Two 74LS157s are used in the interface to allow the WM IO line to control the offset in the address placed on the SWTP bus. When IO is high, indicating that the WM is addressing its I/O ports, the B inputs on the data selectors are selected. These inputs are hard-wired to give an address of 80XX. When IO is low, the A inputs are selected. These inputs have A15 wired low and A14 inverted from the WM bus.

These address-line modifications shift addresses in the range 4000 through 7FFF on the WM bus to 0000 through 3FFF on the SWTP bus. WM addresses not in the range 4000 to 7FFF or FF00 to FF3F will not enable the interface and can be ignored. The upper limit of 7FFF is necessary in my system to prevent interference with the WM video display memory at 8000 to 87FF. The display memory could be relocated (by reprogramming one of the monitor EPROMs), and all 32K of the SWTP memory space made accessible with minor changes in the interface logic.

The enable circuitry for both memory and I/O access to the



*The complete interface. Only the two rear rows of the WM board (bottom of photo) are used in the interface. The other two rows are a WM serial interface.*

SWTP bus requires that BA be high, which means that the SWTP A board is either halted or not plugged into the bus. Additionally, when BA is high, AB15 is low, AB14 is high and MEM is high, the 8T97 drivers on the SWTP bus are enabled for memory access. When BA and IO are high, AB7 is low and AB6 is low, the drivers are enabled for I/O access.

The data bus drivers are enabled whenever the 8T97s are enabled, but the direction (read or write) depends on the WRITE* line. The enable signals for the bus transceivers are generated and used on the SWTP interface board, but they must also be sent to the WM board to control the transceivers there.

Both the 8833 and 8T26 bus transceivers have their bus input and output terminals connected to the buses. This means that when the 8833s are enabled to write to the SWTP bus, the 8T26s must be enabled to read from the WM bus. Therefore, the read enable line for the SWTP becomes the write enable line for the WM, and vice versa. The 8833 enable inputs for both reading and writing are active low, but the 8T26 enable for reading is active high (the enable for writing is active low). I inverted the read bus line from the SWTP on the WM board before sending it to the 8T26 write enable input.

The RPLY* line is used by the WM to allow stretching the WM CLK signal when accessing slow memory or I/O chips. The WM CLK signal is held high (up to 3.5 microseconds—us) until approximately .5 us after RPLY* goes low. Since the SWTP devices must be fast enough for a 1 MHz clock, RPLY* is pulled low by the interface whenever it is enabled and CLK is high.

**The Master-Slave Relationship**

Now let's look at how the WM can halt the SWTP when it wants to use the SWTP bus. The 6800 chip allows DMA by three different methods: halting, cycle stealing and multiplexing. The simplest method, and the only practical one using the SWTP, is halting. The SWTP MP-A board, which contains the CPU, the clock and the MIKBUG ROM and RAM, is designed for DMA by halting, but it does require a very minor modification for full DMA. Before we discuss the modification, let's look at how DMA by halting works on the 6800.

The 6800 microcomputer is halted, logically enough, by pulling the $\overline{\text{HALT}}$ line low. This line may not change state ex-

cept during the first part of the machine cycle, but SWTP has provided a flip-flop, clocked by the phase 1 clock, on the MP-A board to satisfy this requirement. Therefore $\overline{HALT}$, the SWTP bus line, can be pulled low or allowed to go high at any time. When low, the 6800 will complete its current instruction, remove itself from the bus and then set BA high. This process can take from 2 to 15 us, depending upon the current instruction.

The delay between pulling $\overline{HALT}$ and actually having the bus available (BA high) means that we cannot simply pull $\overline{HALT}$ and simultaneously use the bus. Some means of latching the SWTP into its halted state must be used. I used a flip-flop on the WM board for this purpose. When the WM addresses FFDC, the flip-flop is clocked, causing its output to change state. The output is connected, through a unidirectional line, a NAND gate and an inverter, to the enable input of an 8T97. This pulls $\overline{HALT}$ low when it is enabled and serves as a driver for the BA LED indicator on the interface board. The other input of the NAND gate is connected to a switch on the board to allow manual halting of the SWTP.

The CLR input on the flip-flop is connected to +5 volts from the SWTP board. This initializes the flop-flop to the nonhalted state when the WM is powered up before the SWTP. It is important to allow the SWTP to initialize itself before it is halted, even if its CPU is not going to be used.

Although the SWTP MP-A board appears to have been designed for DMA by halting, the circuitry to control the data bus drivers when the CPU is addressing the MIKBUG ROM or RAM prevents a DMA device from reading from the bus. A simple circuit modification corrects the problem without interference to normal computer operation.

Two different logic paths are used by SWTP to control the direction in which the data bus transceivers are enabled. This is necessary because the data lines to the MIKBUG memory are connected directly to the CPU data lines rather than the data bus. When MIKBUG is addressed by the CPU (addresses EXXX or AXXX), the normal logic path is overridden by the MIKBUG path. The transceivers are now enabled to write to the bus while the CPU uses the MIKBUG memory. Since none of the boards on the bus are enabled at this time, the data on the data bus are ignored.

When the computer is halted, the normal logic path is disabled, but the MIKBUG path is still active. When the R/$\overline{W}$ line is low (DMA device writing to the bus), the transceivers are enabled to read from the bus. This does not cause any problem. When the R/$\overline{W}$ line is high, the data bus transceivers are enabled to write to the bus, regardless of the address on the bus. This means that when the DMA device tries to read from memory or I/O on the bus, data bus drivers on both the board addressed and on the MP-A board are enabled. A prime rule of bus usage, "Thou shalt not have more than one set of drivers enabled at any time," is violated, and only garbage can be read from the bus.

To correct this problem, keep the data bus transceivers in a read from bus mode when the computer is halted. The first gate in the MIKBUG read/write circuit is a NAND gate used as an inverter by wiring its inputs together. When the output of this gate is high, the data bus transceivers are held in the read from bus mode. Since a NAND gate will have a high output when any of its inputs are low, separate the two inputs and use one to hold the output high when BA is high. A line that is low when BA is high is available, conveniently, on the same IC.

The NAND gate is one-quarter of IC12 on the SWTP MP-A board. Pins 4 and 5 of this IC are the inputs tied together to make an inverter, and pins 1 and 2, also tied together, have a line connected to them that is low when BA is high. I cut the foil connecting pins 4 and 5 and ran a short length of insulated wire

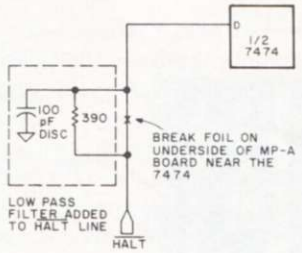



*Fig. 4. Modification to MP-A board to cure noise problem on SWTP $\overline{HALT}$ line.*

from pin 4 to the line coming into pin 1. When BA is high, this modification keeps the transceivers in the read from bus mode, but, when BA is low, the NAND gate acts like an inverter, just as in the unmodified circuit.

With this modification (or, for that matter, without it), the interface cannot read or write the MIKBUG ICs. Because of the MP-A board's design, no DMA method can use the MIKBUG ROM or RAM without added circuitry on the MP-A board. Because of the difficulty of modifying this board, I suggest that if you need this capability you relocate the MIKBUG circuitry to some other card in the system (e.g., the interface card).

One of the ways I hope to use my slave computer is as an I/O controller for the master computer. By connecting all of my slow I/O devices to the slave computer bus and programming it to handle the printer, cassette, etc., which often take the majority of a microcomputer's time, the time available to the master computer to do calculations and data manipulations is greatly increased. There is nothing original about this idea; I suspect all large computers work this way.

To make the interface more flexible, I've incorporated a line that allows the SWTP to inform the WM whenever it needs attention. When the SWTP addresses its 0 I/O port, the WM gets a signal on its #4 interrupt line. Since the WM uses a vectored, maskable interrupt scheme, it can be programmed to go to an I/O service routine when it is ready for I/O and the SWTP has indicated, by interrupting, that it is waiting to be serviced.

### Bugs

I have used my two-computer system for approximately two months, and only one problem has come to light: a noise problem in the SWTP! The HALT line, as I mentioned earlier, uses a flip-flop on the MP-A board to synchronize changes in the HALT line with the phase 1 clock. This is a TTL device that will respond to very short pulses on the bus line. I discovered that noise on this line was causing the SWTP to execute occasional single-cycle operations.

Fortunately, the cure for this problem was simple. I inserted a low-pass filter into the line on the MP-A board before it reaches the flip-flop (see Fig. 4).

This problem was due to noise pulses originating in the interface, not on the SWTP bus. It will not cause trouble if the SWTP is halted by means of a switch connected directly to the bus line and ground with short, reasonably large, wires.

### Summary

Few, if any, will want to build an exact copy of my interface. However, the basic interface design and many of the ideas are applicable to anything from adding a bus to a one-board microcomputer to interfacing two microcomputers using different CPU chips. This is a wide-open area for hobbyist exploration.

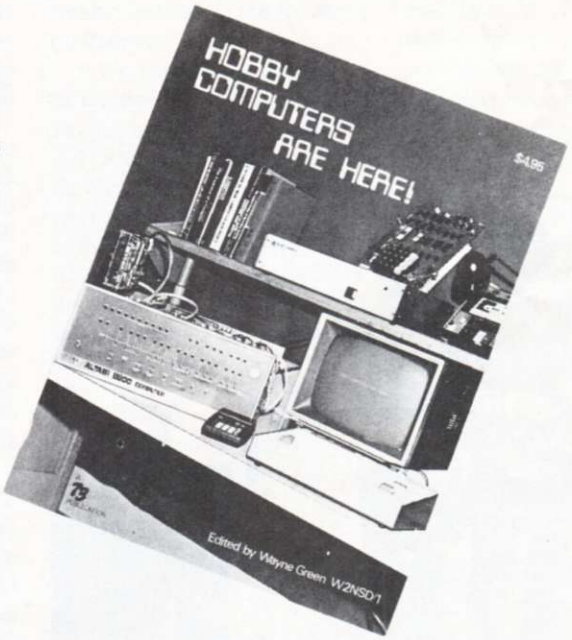
Many people probably feel that interfacing two different computers is too complicated for anyone without an extensive technical background. Actually, if the two computers use the same CPU chip, it is a straightforward and rewarding project. It took me less time to design and build the interface between my two computers than to build an RS-232 interface for my keyboard/printer. All of the critical timing relationships are already taken care of by the computers. All you must do is get the signals between the computers and modify those signals that are different on the two buses.■

# SUBSCRIBE TO KILOBAUD NOW AND GET THIS $5 BOOK FREE!

## HOBBY COMPUTERS ARE HERE!

*Here's your chance to get the best selling book in the hobby computer field . . . for free . . . our gift when you take a subscription to Kilobaud Magazine. Better yet, your satisfaction is guaranteed. If Kilobaud isn't all you expect . . . if you are willing to be without it . . . just ask for a refund . . . and keep the book.*

Hobby Computers Are Here is for the beginner . . . it starts out with the fundamentals of microcomputers . . . shows how all of the basic parts of a computer system work . . . gates flip-flops, TTL logic, keyboards, video generators, I/O systems, CPUs . . . it is a whole basic education in microcomputers. That's why more copies of this book have been sold than any other in the field.




SUBSCRIBE. Never mind the $2 per copy . . . which comes out to $24 a year. The subscription rate is only $15 for a year and WE'LL THROW IN THE $5 BOOK FREE. Charge it to your Master Charge or American Express. If you are in a hurry, you can call it in on our toll free 800 number and your book will be on its way to you, along with the current issue of *Kilobaud.*

**Use your credit card and call our Toll free subscription number 800-258-5473**

*KB-6*

kilobaud
peterborough nh 03458