NEWSLETTER

# HOMEBREW COMPUTER CLUB

**Robert Reiling**, Editor □ Post Office Box 626, Mountain View, Ca. 94042 □ **Joel Miller**, Staff Writer
Typesetting, graphics and editorial services donated by **Laurel Publications**, 17235 Laurel Rd., Los Gatos, Ca. 95030 (408) 353-3609

## RANDOM DATA
*By Robert Reiling*

A survey of hobbyists attending the June 9th meeting reveals the following distributions of CPUs in use: 8080—53, 6502—18, 8008—6, PDP-8—4, LSI-11—3, Z80—2, 4004—1, PDP11/20—1 and TTL—1. That totals 101 systems up and running out the the group. About 250 hobbyists were at this meeting. A similar survey in the October 15, 1975 meeting turned up 38 systems with about 80 hobbyists attending that meeting. This growth is just a sample of the changes taking place everywhere in computer hobbyist groups.

This issue of the newsletter is really filled with interesting material. Robert Purser presents ideas to help get started with word processing—one of my favorite subjects. Walt Hutchinson provides pointers to sources of parts for AMI prototype board users. Mark Garetz has put together a listing program that will be useful to 6502 users. For 8080 users, John Schulein has a Trap routine that should make your program development job much easier. Of course, many other news items are included too.

Don't forget your article can be published too! How about a description of your system and how to use it? Send it to me at P.O. Box 626, Mountain View, Ca. 94042.

## HEY AUTHORS!
*By Joel Miller*

**Laurel Publications** is currently planning production of a book tenatively titled *The Personal Computer* and we are looking for contributors for various chapters. Some of these chapters may include discussions of various systems available such as thoroughly homebrew systems, developmental boards and complete systems; how to get into and have fun with software; peripherals and their interfaces; how microcomputers work. We are looking for talented people with some writing experience, however help with rewriting is available. Rough illustrations will be redrawn by professional draftsman. We are looking for people with diversified interests and specialties to make a well-rounded and interesting presentation. Royalties to be divided among participants. For more information, call **Joel Miller** at (408) 353-3609 or write **Laurel Publications, 17235** Laurel Rd., Los Gatos, Ca. 95030.

## WORD PROCESSING
*By Robert Purser*
*Action Audio Electronics*

An interesting application for homebrew computers is *Word Processing*. A Word Processing unit is basically a semi-intelligent typewriter. Since Word Processing is relatively new, the businessmen who are interested in this field are more adventuresome than most. Many of them would be willing to fund a project to design a Word Processing Unit that is cheaper (and more flexible and expandable) than available commercial units.

A Word Processing Unit is simple in design (much simpler than an accounting system for example). In general, every Word Processing Unit has the following:

*I/O:* A high quality, upper and lower case printer with an upper and lower case keyboard.

*Logic:* Control keys and associated control logic.

*Memory:* A recording media and/or memory.

Before selecting the individual components, remember two things: (1) a Word Processing Unit is used longer and harder than any other office machine. Therefore, it must be durable and reliable. (2) Non-mechanically minded people will be using the Word Processing Unit, so keep all mechanical operations to a minimum.

You have two ways to go for I/O: a separate keyboard and printer or a typewriter terminal. When using a separate keyboard and printer, use only a typewriter style keyboard. For convenience, choose a fully decoded keyboard. For a printer, the *Adapt-A-Typer* is one alternative. For more information on this unit, see the April issue of *Byte* magazine.

Another printer alternative is a daisey wheel or daisey cup printer which are TTL compatible, high quality and high speed. Three manufacturers of these units have offered to provide club members with free literature on their units.

**Qume**—drop a postcard to:
James Soderman
Qume
745 Distel Drive
Los Altos, Ca. 94022

**Diablo**—phone:
Jack Fuller
Diablo
(408) 286-9424

**Interdata**—drop a postcard to:

Dave Carson
Interdata
11222 Lacienega Blvd.
Suite 666
Inglewood, Ca. 90304

Used Selectric or Friden units may be hard to interface and are not recommended. (Editor's note: Even though the Selectric is difficult to interface, it may be worth the hassle considering the availability of many different type fonts including OCR fonts. Also, Totalert Systems, 2001 Karbach, Suite K, Houston, Texas 77018 manufactures a Selectric Interface Controller designed for the PDP 8/E or 8/I.)

The second alternative is a typewriter terminal. Daisey wheel or Selectric terminals are slowly becoming available. The interface is serial and relatively simple. Most units are also available on a rental basis.

For the necessary logic, a microprocessor is ideally suited to the task. The typist interacts with the control program via control buttons. The control program consists of an I/O monitor and a series of subroutines for the different control functions.

The control buttons are no problem. Some keyboards have extra keys which can be used as control buttons. Also, you have the option of huying a separate control keyboard. In most cases however, the control buttons are simulated by either of the following techniques: (1) Have one control button which indicates to the program that the next character to be typed is to be intepreted as a control character. (2) Or have one button which when held down modifies the codes of the regular buttons to make then control codes. (Many keyboards have this feature built in.)

The control program will require at least some PROM memory. No typist is going to load a program through the front panel. Put at least a loader on PROM to load the program via tape. A good idea would be to put a skeleton system (record, play, play character, forward, backspace, reset, store and recall) in PROM with additional control functions (subroutines) to be loaded from tape.

At first do not try to design your own set of control functions. Instead, try simulating a commercially available Word Processing Unit. A list of Word Processing Units plus a brief description of each is available in the November, 1975 issue of *Administrative Management*. Choose the Word Processing Unit you want and obtain the User's Reference Manual for it. This has three advantages: (1) The Reference Manual will serve as a "systems spec". Once you have this, half your work is done. (2) When you are finished you do not have to write your own manual for the typist. (3) Training the typist is easier if the typist is already familiar with the commercial unit.

Memory: In addition to the control program's memory, about 4K of RAM is sufficient for text memory. If you are supplementing your system with a video display, part of the video memory can be used as some of the text memory. A tape cassette or a disk storage device is needed for storing permanent copies for future replay or editing.

Other considerations: As you design your Word Processing Unit, do not ignore the economics of it. A simple Word Processing Unit will cost from $3,000 to $6,000 in hardware. If you are planning to simulate a commercially available unit, choose one which is at least three times as expensive as your hardware ($9,000 to $16,000). Roughly, you should figure 1/3 for the basic hardware, 1/3 for your time and effort and 1/3 for the businessman. No businessman will invest in an experimental machine unless it saves him at least 1/3 the price of a commercial unit.

On the other hand, do not overlook the obvious. IBM designs good office products. They now have a new unit called the *Memory Typewriter* which costs only $4,000. It has a totally new, more intelligent logic design. People like it and are buying it. Even though no one is going to pay $6,000 for hardware to simulate a $4,000 machine, it would be very wise to try simulating the IBM machine first before you attempt to build your ultimate Word Processing Unit.

Robert H. Edmunds is coordinating and stimulating interest in developing homebrew Word Processing Units. His address is P.O. Box 464, Estudillo Station, San Leandro, Ca. 94577. He and I would like to hear what you are doing in this field.

## BULLETIN BOARD

IMSAI RAM 4-1. Low profile sockets, 450ns, 1K expands to 4K, switchable write protect on board, fully assembled, new, $99/best offer. **WM. J. Schenker, M.D.** 2086 Essenay, Walnut Creek, Ca. 94596; 939-6295.

Teletypes. Models 28 through 40, new or rebuilt, RO's, KSR's and ASR's. All available immediately. **National Typewriter Corp.**, 207 Newtown Rd., Plainview, N.Y. 11803. Contact **Joe Gibbons** at (516) 293-0444.

**Kentucky Fried Computers**, a new retail computer kit business in Berkeley makes this offer: *All IMSAI products (except some peripherals) will be sold at 10% off to* **Homebrew Computer Club** *members. Add 2% if shipping is desired (excess is refunded). Orders may be picked up in Berkeley by appointment only. If there is sufficient volume from Homebrew members, we will arrange free delivery for distribution at Homebrew meetings. California residents must add 6% sales tax. The discount offer expires July 31, 1976. Terms: cash.* Kentucky Fried Computers is operated by **Mark Greenberg** and **Charles Grant**. They plan to open a store later this year, but for now they are selling on a mail and telephone order basis. Contact them at **Applied Computer Technology**, 1038 Merced, Berkeley, Ca. 94707; telephone (415) 527-6760.

For Sale: IMSAI system—consists of 1-8080 computer system, 22-EXPM edge connectors, 2-RAM4-4 4K RAMs, 1-PROM 4-4 4K EPROM on board, 1-PGM-1A self-contained system programmed on PROM board, 1-BASIC-4A paper tape, 1-EXP-22 22 slot mother board and 1-SI02-2 2 channel serial I/O board. Total retail price is $3,066.58. The system is completely factory assembled and checked out and will be covered by IMSAI's 90-day warranty, transferrable to the purchaser. Of course, all the IMSAI documentation is included. I am asking $2,000 for the complete system (the unassembled price of this equipment would be $1,867.72). This is a savings of $35%. Contact **Dr. Les Oldenbrook**, 14 Midvale Drive, Daly City, Ca. 94015, telephone (415) 756-5593.

## ON_LINE NEWSLETTER

ON_LINE Newsletter is the only nationwide classified advertising newsletter devoted entirely to the computer hobbyist. ON_LINE is a medium through which hobbyists can buy, swap and sell equipment, programs and services related to the field of home, small business and personal-use computers.

You will probably want to subscribe if . . . you are just getting into the hobby of computers and are looking for good sources of low cost new and used everything (computers, I/O equipment, software, etc.) . . . you are allready "on the air" with a basic system and are looking for bargains in peripherals, supplies, software, memory, etc. to make a larger system . . . you are interested in sharing with other enthusiasts the unique software and/or hardware you developed and also in making some pocket money for "operating expenses".

ON_LINE has fast turnaround time. It includes ads received just 4 days prior to mailing and it is sent out every three weeks.

ON_LINE offers **free ad space** to computer hobbyists wishing to form clubs and to established organizations to announce meetings, special activities and other non-profit events. Also, free ad space is given to clubs or individuals offering information or software available at no charge or for a charge calculated to cover just the cost of distribution.

Each issue contains 6 to 12 pages (3 to 6 sheets, 8½ by 11). Subscription rates are $3.75 for 18 issues (1 year) or $7.00 for 36 issues. Full satisfaction is guaranteed—used portion of subscription will be refunded at any time. Low advertising rates enable individual sellers and small business to offer their wares. Non-commercial advertisements cost $1.50 per line. Commercial advertisements are $3.50 per line.

Send subscriptions to D.H. Beetle, Publisher, 24695 Santa Cruz Hwy., Los Gatos, Ca. 95030.

## ETC OFFERS A COMPLETE SYSTEM
*News Release*

Hawthorne, Ca.—**Electronic Tool Co.** has recently introduced a complete microcomputer system, based on the MOS Technology 6502 CPU, for $675. A spokesman for Electronic Tool Co. said that the **ETC-1000** comes with a 40-key keyboard, a programmable 8-digit display, I/O interfaces, power supply and more. All systems are fully assembled, tested and ready to run.

The ETC-1000 is intended for system development, control, and small-scale data processing applications. As a development system, it provides full system support for hardware and software design work. As a control system, it offers an inexpensive, high-speed computing capability in a studry rack-mountable package. Memory expansion, I/O capacity, and programming flexibility make the ETC-1000 a top choice for stand-alone and communications-oriented data processing.

Software currently available includes a resident assembler, I/O handlers, diagnostics and other support tools. The manufacturer says that BASIC and PLM support are expected to be available during the third quarter of 1976.

Availability of standard configurations is 30-60 days. For more information, contact **Debbie Pye**, Electronic Tool Co., Hawthorne, Ca., (213) 644-0113.

## AMI PROTOTYPE BOARD: PARTS SOURCES
*By Walt Hutchinson*

If you are assembling the AMI board (and are as much a neophyte as I), you may find the following list helpful in saving you time and gas. But please use it only as a guide and always phone before you visit (things change rapidly around here). Also, please excuse me if I've omitted your favorite parts house. If so, tell us about it at the next meeting!

### *LOCAL PARTS SOURCES FOR THE AMI PROTOTYPE BOARD*

**Solid State Music**—2102A Walsh Ave., Santa Clara, Ca. telephone (408) 246-2707. *Socket kit, IC kit, transistor kit, diode kit, Baud rate generator, trim pots, 2.4576 crystal, disc capacitors, tantalum capacitors.*

**James Electronics**—P.O. Box 882, Belmont, Ca., telephone (415) 592-8097. *Sockets, ICs, transistors, diodes, 1 MHz crystal, etc, but no kits expressly for the AMI board. Prices comparable to Solid State Music.*

**Halted Specialties**—729 Evelyn, Sunnyvale, Ca., telephone (408) 732-1573. *Tantalum capacitors, rare resistors, etc. New and used. "Supermarket" style. Well worth visiting just to look.*

**Haltek Electronics**—1062 Linda Vista, Sunnyvale, Ca., telephone (408) 969-0510. *"The" Supermarket. Something of everything. New and used. (Caveat emptor, but will blow your mind.)*

**Weatherford**—3240 Hillview, Palo Alto, Ca., telephone (415) 493-5373. *Local AMI CMOS outlet. (Has trouble keeping these parts in stock, however.)*

**Hamilton Avnet**—575 East Middlefield Road, Mountain View, Ca., telephone (415) 323-7239. *Motorola outlet. Alternative source for CMOS (except for the 6834). Prices differ, however. (Also has stocking problems.)*

**Moltronics**—2300 Owen, Santa Clara, Ca., telephone (408) 244-7600.

**Robinson-Nugent**—*Low-force, 24-pin sockets and other useful things.*

**Acacia Sales**—384 San Aleso, Sunnyvale, Ca., telephone (408) 735-0100. *R-N low-force 24-pin sockets in stock. Can order 8 pF mica capacitors.*

**Capacitor Sales**—253 Polaris, Mountain View, Ca., telephone (415) 964-8880. *Have 8 pF mica capacitors in stock. (Please do group buys, however.)*

**Bogen & Associates**—19752 Bixby Dr., Cupertino, Ca., telephone (415) 257-4461. *Textool no-force sockets. (Make sure you get the right part number. Must file pins.)*

**U-Do Electronics**—1036 Castro, Mountain View, Ca., telephone (415) 968-8894. *Dual in-line PC board switches. Many other necessary things.*

**Sterling Electronics**—1061 Industrial Road, San Carlos, Ca., telephone (415) 592-2353. *Will provide the Amphenol edge connectors. Takes a while.)*

**Problem Areas**: Right angle switches; -50V regulators; cheap (but light and strong) enclosures; inexpensive terminals; money.

## DEBUGGING 8080 SOFTWARE... HOW ABOUT A TRAP ROUTINE?

*By John Schulein*

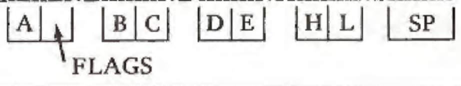
If you have your 8080 system up and running and are starting to write some programs, the trap routine discussed in this article may be just what you need to assist in debugging your own software or other software you have keyed in and/or relocated. This routine is in PROM in my own system and has been invaluable in locating bugs in various programs.

This trap routine is a program that is written as a subroutine and is designed to do the following when it is called: (1) Immediately save the machine state, (2) Print out the machine state on a TVT and (3) Wait for a keyboard input so that the programmer can decide whether to examine and/or modify memory or to restore the machine state and resume execution of the program that called the trap routine.

A typical way to use this trap routine is to change one or more instructions in critical areas of the program being debugged to the RST 7 instruction (FF in hex). Note that this implies that your problem program is in RAM and that if a multi-byte instruction is changed, the RST 7 instruction is placed in the first byte of the multibyte instruction. Several traps can be set at once since the trap location will be printed out each time the trap routine is executed. The trap location will be printed out each time the trap routine is executed. The trap routine is activated (called) whenever the 8080 CPU encounters an RST 7 instruction. Thus, if the trap routine does not start at location 0038 hex, then a JMP TRAP instruction must be present at location 0038 hex. If your PROM memory starts at location 0000, then you can fix the trap routine (or jump instruction and trap rountine) into PROM and you are ready to go. If your PROM is in high memory, then you can fix the trap routine in high memory and you will have to remember to load the jump to trap instructions (3 bytes) into locations 0038 through 003A hex each time you power up the system. An alternate way to get into the trap routine would be to use a standard three-byte call instruction (here no jump instruction would be required). I decided to use the single byte call (RST) mainly for convenience (one byte to change instead of three) and also because my PROM is in low memory.

The output of the trap routine is two lines of printout on a TVT as shown below:

```
        TRAP AT :xxxx:
Axxxx:Bxxxx:Dxxxx:Hxxxx:Sxxxx
```



The first line shows the memory location of the RST 7 calling instruction that resulted in the trap routine being called and the second line displays the machine state (accumulator, flags, the registers B,C,D,E,H & L and the stack pointer) at the completion of the last program instruction executed before the trap calling instruction (RST 7) was encountered.

The trap routine is shown in Figure 1 and the auxilliary subroutines used by the trap routine are described in Figure 2. Listings of these auxilliary subroutines are not provided since most monitor programs will already have these routines and they most likely are system-peculiar since they deal directly with I/O devices. The stack data generated by the trap routine is shown in Figure 3 and will help in understanding the operation of the trap routine. The reader is referred to these three figures during the following explanation of the trap program.

| | |
|---|---|
| PRST | Prints a string of ASCII characters in memory starting at the location pointed to by HL. The string is terminated by a zero byte. |
| PTADR | Prints HL as four hex digits followed by a colon. |
| PTADR1 | Prints HL as four hex digits followed by a colon. |
| CRLF | Sends a carriage return and a line feed to the printer. |
| RDTTY | Reads an ASCII character from the keyboard. Any character other than ESC returns the processor to the calling program with the ASCII character in the A and D registers. An ESC character sends the processor to the monitor program. |
| PRNT | Prints the ASCII character in the D register. |

Auxilliary Suhroutine Descriptions
*Figure 2*

When the trap routine is called by the RST 7 instruction, the program counter is automatically pushed onto the stack. The next four push instructions in the trap routine (lines 1 through 4) save the A,B,C,D,E,H and L registers and the flags in the stack. This machine information will be used at the very end of the trap routine to restore that machine state just before the trap routine return instruction (line 40) is executed. The fifth push instruction (line 5) is used to provide a space in the stack for the value of the stack pointer when the trap routine is called. This value is calculated by adding 000C hex to the current stack pointer value at line 6 of the trap routine. The DAD SP instruction (line 7) does this addition and the desired initial stack pointer value ends up in HL after the DAD SP instruction is executed.

The XTHL instruction at line 8 exchanges the most recent two stack bytes with the contents of HL. Thus after the execution of line 8, the initial value of the stack pointer is on the bottom of the stack (the 8080 stack grows downward) and the contents of HL have been restored to their initial values (the values they had when the trap routine was called). The next four push instructions (lines 9 through 12) save the machine state a second time for the printing operation.

Lines 13 through 14 print the ASCII character string "TRAP AT" in preparation for the printing of the memory address that generated the trap. This information is highly desirable since several traps may have set into the program being debugged and it is nice to know which trap was encountered each time the trap routine is executed. The trap subroutine return address (pushed onto the stack when the trap subroutine was called) is the key to determining the trap location. In order to access this information, lines 15 and 16 cause the HL register pair to point higher up in the stack to the most significant byte of the return address. The return address is put into the DE register pair by the instructions at lines 17, 18 and 19. Note that the value of the DE register pair is decremented by one (line 20) in order to reflect precisely the location of the RST 7 trap generating instruction. This is due to the fact that the return address pushed onto the stack by the RST 7 instruction is the next memory address after the RST 7 instruction. The XCHG instruction at line 21 exchanges the register pairs DE and HL which puts the trap location into HL so that the PTADR subroutine called at line 22 will print the trap location (PTADR prints out the contents of HL as four hex digits surrounded by colons). A CRLF subroutine called at line 23 drives the TVT to the beginning of the next line.

Now things get a little tricky but we are almost

| Line Number | Label | Instruction | Comments |
|---|---|---|---|
| 1 | TRAP | PUSH A | |
| 2 | | PUSH B | Save registers for final return to program being debugged. |
| 3 | | PUSH D | |
| 4 | | PUSH H | |
| 5 | | PUSH H | SP value stack position. |
| 6 | | LXI H,000CH | Stack pointer correction factor. |
| 7 | | DAD SP | HL+SP→HL |
| 8 | | XTHL | Puts corrected SP onto stack and restores HL. |
| 9 | | PUSH H | |
| 10 | | PUSH D | Saves Registers for printing. |
| 11 | | PUSH B | |
| 12 | | PUSH A | |
| 13 | | LXI H,Trap Message | |
| 14 | | CALL PRST | Prints "TRAP AT". |
| 15 | | LXI H,0013H | Stack pointer displacement to return address. |
| 16 | | DAD SP | Points HL to return address. |
| 17 | | MOV D,M | MSP of return address→D |
| 18 | | DCX H | |
| 19 | | MOV E,M | LSP of return address→E |
| 20 | | DCX D | Decrement to trap location. |
| 21 | | XCHG | Trap location ends up in HL. |
| 22 | | CALL PTADR | Prints trap location as 4 hex digits. |
| 23 | | CALL CRLF | |
| 24 | | MVI D,"A" | |
| 25 | | CALL PRREG | |
| 26 | | MVI D,"B" | |
| 27 | | CALL PRREG | |
| 28 | | MVI D,"D" | Print flags, registers and SP. |
| 29 | | CALL PRREG | Axxxx:8xxxx:Dxxxx:Hxxxx:Sxxxx: |
| 30 | | MVI D,"H" | |
| 31 | | CALL PRREG | |
| 32 | | MVI D,"S" | |
| 33 | | CALL PRREG | |
| 34 | | CALL RDTTY | Wait for keyboard input (ESC gives monitor). |
| 35 | | CALL CRLF | |
| 36 | | POP H | |
| 37 | | POP D | Restore machine state. |
| 38 | | POP B | |
| 39 | | POP A | |
| 40 | | RET | Return to program being debugged. |
| 41 | PRREG | CALL PRNT | Prints ASCII character in D register. |
| 42 | | POP H | Pop return address into HL. |
| 43 | | XTHL | Get 2 stack bytes and restore return address. |
| 44 | | CALL PTADR1 | Print 4 hex digits. |
| 45 | | RET | |

**TRAP Routine**
*Figure 1*

through so don't give up now! The next ten lines of program (lines 24 through 33) result in the printing out of the machine state in the form of five groups of characters on the TVT using the PRREG subroutine (lines 41 through 45). Each group of characters consists of a single identifier alpha character, four hexidecimal digits and a colon. Thus the printout of the machine state line takes 30 characters and just fits on a 32 character/line TVT.

Each execution of the PRREG subroutine results in the following operations. First the ASCII character in the D register is printed out on the TVT by the PRNT subroutine. Next the return address for the PRREG subroutine is popped off the stack into the HL register pair by the POP H instruction at line 42. This brings to the bottom of the stack one of the 2-byte values to be printed that was placed on the stack by the instructions at lines 8 through 12. Now the PRREG subroutine return address is restored to the stack (two bytes up from its original position) and the two-byte value at the bottom of the stack that is to be printed is put into HL by the XTHL instruction at line 43. That XTHL instruction sure is nice! Finally, the four hex digit value now in HL is printed out on the TVT by subroutine PTADR1 (called at line 44)* Each execution of the PRREG subroutine causes the stack to be reduced by 2 bytes and after the execution of the last PRREG call instruction (line 33), the stack has been reduced to the upper half shown in Figure 3.

| | | |
|---|---|---|
| *Used to get back to the program being debugged.* | MSP LSP | *Return address for the calling RST 7 instruction.* |
| *Used to restore the machine state.* | A F | *PUSH A @ line 1* |
| | B C | *PUSH B @ line 2* |
| | D E | *PUSH D @ line 3* |
| | H L | *PUSH H @ line 4* |
| *Used to print out the machine state.* | MSP LSO | *Corrected stack pointer* |
| | H L | *PUSH H @ line 9* |
| | D E | *PUSH D @ line 10* |
| | B C | *PUSH B @ line 11* |
| | | *PUSH A @ line 12* |

*NOTE: This figure shows the stack data generated by the trap routine when the processor is executing the instruction at line 13.*

**TRAP Stack Data**
*Figure 3*

After the entire machine state line has been printed out, the trap routine waits for a keyboard input in the RDTTY subroutine (called at line 34). This gives the person debugging a program a chance to digest the trap information before proceeding. Any character other than ESC will return the processor to the trap routine and then to the program being debugged. An ESC will cause the processor to return to the monitor program where the operator can examine and modify memory if necessary. If the character typed was not ESC, a CRLF is generated and then the machine state is popped off the stack by the four pop instructions at lines 36 through 39. After restoring the machine state, the processor will return to the program containing the trap (the program being debugged) and resume execution. Note that you can only use this technique if the program being debugged has the trap routine-calling instructions (RST 7) located in program locations that originally contained NOP's. Most of the time I use the ESC monitor exit so that I can restore the original instruction and move the RST 7 trap-calling instruction to a new location. Then another execution of the problem program can be initiated. This process will continue until the source of a problem is determined and then the program can be modified to (hopefully) correct the problem.

This trap routine (or a similar one) should make debugging much easier as it produces a printout of the entire machine state at a known point in a program. Good luck in finding and removing those bugs.

*This article may be reprinted only with express written permission from the Author.*

| A | B | MASK | 1st XOR | AND | 2nd XOR | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | = B |
| 0 | 0 | 1 | 0 | 0 | 0 | = A |
| 0 | 1 | 0 | 1 | 0 | 1 | = B |
| 0 | 1 | 1 | 1 | 1 | 0 | = A |
| 1 | 0 | 0 | 1 | 0 | 0 | = B |
| 1 | 0 | 1 | 1 | 1 | 1 | = A |
| 1 | 1 | 0 | 0 | 0 | 1 | = B |
| 1 | 1 | 1 | 0 | 0 | 1 | = A |

*ERRATUM*—Volume 2, Issue 5, Page 3, Figure 1 (Byte Packing Truth Table). Bit in third row of rightmost column should be a "1".

## IMSAI MOVES

IMS Associates, Inc. of San Leandro, California, recently moved into new facilities which more than quadruple the company's manufacturing space. The company's new address in San Leandro is 14860 Wicks Blvd., 94577, however the phone number remains the same — (415) 483-2093. The rapid growth of IMSAI has been attributed to the demand for the new IMSAI 8080 Microcomputer introduced earlier this year.

# LIST COMMAND FOR THE 6502

*By Mark Garetz*

```
  JSR   20   728A   Car. rt. line feed subr.
  JSR   20   72E9   Input char subr.
  CMP   C9   4C     Check for "L"
  BEQ   F0   03     If "L" then input next char.
  JSR   20   04B5   Error subr.
  JSR   20   72E9   Input char. subr.
  CMP   C9   49     Check for "I"
  BEQ   F0   03     If "I" then input next char.
  JSR   20   04B5   Error subr.
  JSR   20   72E9   Input char. SR
  CMP   C9   53     Check for "S"
  BEQ   F0   03     If "S" then input next char.
  JSR   20   04B5   Error SR
  JSR   20   72E9   Input char. SR
  CMP   C9   54     Check for "T"
  BEQ   F0   03     If "T" then proceed with program
  JSR   20   04B5   Error SR
  JSR   20   7377   Print space char. SR
  JSR   20   73A4   Addr. input SR.  Input start Addr.
  LDA   A5   EE     Load A with start addr. low byte
  STA   8D   04B1   Store low byte at 04B1
  LDA   A5   EF     Load A with start addr. high byte
  STA   8D   04B2   Store high byte at 04B2
  JSR   20   7377   Space
  JSR   20   73A4   Input end addr.
  LDA   A5   EE     Load A with end addr. low byte
  STA   8D   04B3   Store low byte at 04B3
  LDA   A5   EF     Load A with end addr. high byte
  STA   8D   04B4   Store high byte at 04B4
  INC   EE   04B3   Inc end low byte
  LDA   AD   04B3   Load A with end low byte
  CMP   C9   00     Check to see if INC made byte "00"
  BNE   D0   03     If not, continue with prog.
  INC   EE   04B4   If yes, then INC high byte
  LDY   A0   00     Clear Y reg. for bytes/line counter
  NOP   EA          These NOP's are here because I
  NOP   EA          made a boo-boo
  JSR   20   728A   CR LF  SR
  LDA   AD   04B2   Load A with start high byte
  JSR   20   72B1   SR to type byte in hex
  LDA   AD   04B1   Load A with start low byte
  JSR   20   72B1   Type start low in hex
  JSR   20   7377   Space
  JSR   20   7377   Space
# LDA   AD   04B1   Load A with start low
```

```
STA   8D   047E   Plug into
LDA   AD   04B2   Load A with start high
STA   8D   047F   Plug into
LDA   AD   □□     Get byte to list
JSR   20   72B1   Type byte
JSR   20   7377   Space
INC   EE   04B1   INC start low byte
LDA   AD   04B1   Load A with start low byte
CMP   C9   00     Check if INC made byte "00"
BNE   D0   03     If not, skip next instr.
INC   EE   04B2   If so, INC start high byte
LDA   AD   04B1   Load A with start low byte
CMP   CD   04B3   Compare with end low byte
BNE   D0   0B     If not equal skip next 4 instr.
LDA   AD   04B2   Load A with start high byte
CMP   CD   04B4   Compare with end high byte
BNE   D0   03     If not equal skip next instr.
JMP   4C   7086   End list, JMP to location to return ctrl to TIM
INY   C8          INC bytes per line counter - Y reg.
CPY   C0   10     Check for 16th byte
BNE   D0   03     If not skip next JMP
JMP   4C   0458   JMP to "*"  print next line
JMP   4C   0471   JMP to "#"  print next byte
00,00,00,00       These locations store start and end addr.
JSR   20   7377   Space, start of Error SR
LDA   A9   45     Load A with "E"
JSR   20   7216   Print "E" (print SR)
LDA   A9   52     Load A with "R"
JSR   20   7216   Print "R"
LDA   A9   52     Load A with "R"
JSR   20   7216   Print "R"
LDA   A9   4F     Load A with "O"
JSR   20   7216   Print "O"
LDA   A9   52     Load A with "R"
JSR   20   7216   Print "R"
JSR   20   7377   Type a space
JMP   4C   0400   Go back to beginning & wait for "L"
```

*REQUIRES TIM ROM AT 7000—PROGRAM LENGTH: 215 BYTES*

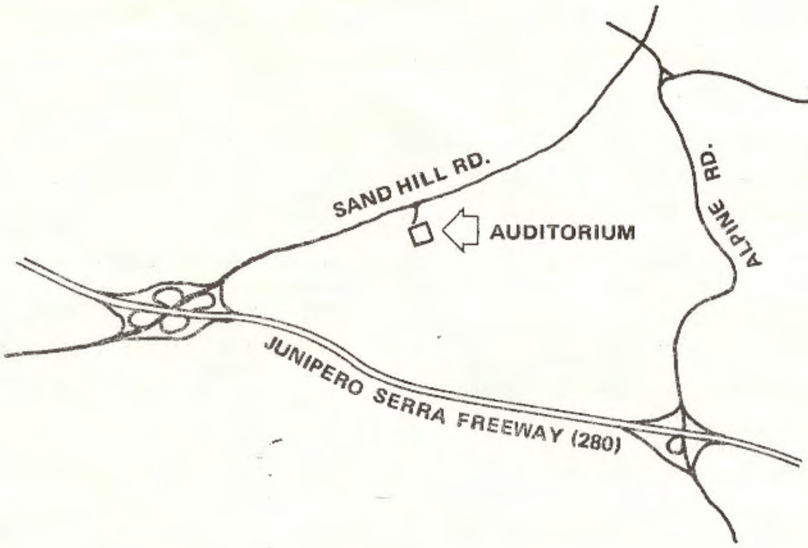
## *PROGRAM OPERATION*

*This program waits for you to type in "LIST". Anything else causes it to type "ERROR" and waits for you to type "LIST" again. After typing "LIST" it will then respond with a space and wait for the address of the start of the list in hex. After typing that in, it responds with another space and waits for the ending address of the list. After typing that in, it responds with a carriage return and line feed, types the starting address, types 2 spaces and 16 bytes each separated by a space, after 16 bytes it does a CRLF, prints the starting address plus 16 and then 16 more bytes until the ending address is reached. At this point, control is returned to TIM which responds with a CRLF and the prompting character "."*

## HOMEBREW COMPUTER CLUB MEETINGS

*Where & When*

The **Homebrew Computer Club** meets every other Wednesday (July 21st, August 4, August 18th, etc.), 7PM at the **Stanford Linear Accelerator Center Auditorium**. Directions: From Freeway Rt. 280, take the **Sand Hill** exit east toward Menlo Park. Turn right at the S.L.A.C. sign. The Auditorium is directly ahead. The parking area is to your right.



**HOMEBREW COMPUTER CLUB NEWSLETTER**
**P.O. Box 626**
**Mountain View, Ca. 94042**

AMF SAN FRANCISCO JUL 12 AM 1976 940

**FIRST CLASS MAIL**

```
JOHN KOSKI
6152 CECALA DR.
SAN JOSE, CA    95120
```