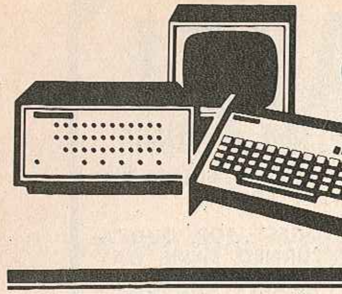

Computer Bits

By Jerry Ogdin

SOFTWARE TOOLS

NO MATTER how small the computer, even the most dedicated programmer will rapidly become bored with binary notation.

Some hobbyists use a Teletype as there are several older models on the market at reasonable prices. Even with a Teletype, though, you need some software to convert those keystrokes into something meaningful in memory. If you can't afford a Teletype, you can almost always use an encoded keyboard. These are frequently on the surplus market for less than $25.00.

A terminal is important, but it is only one tool in the computer hobbyist's kit. Once you've written a program and gotten it into storage, you ought to use the cassette interface (HIT) described in September's column. With this tool, you have to "button in" the program bit-by-bit only once. After it is in storage, you can write it out to be taped and read in the next time you want it. Of course, if your only storage medium is RAM, you'll lose the memory contents when you turn the computer off. So, it's a good idea to copy the latest version of a program out to tape as a backup.

Also, if your only storage medium is RAM, you'll have to reenter the tape reading routine laboriously through the switches (or the terminal) each time the computer power is turned on. That is a good reason for having a small program, called a *bootstrap loader*, kept in read-only memory. This program makes it possible to read data from the tape and then execute that data. Such a read-in program is usually somewhat larger and more powerful, so it reads in several records (perhaps using the bootstrap program as a subroutine), which make up an even larger and more sophisticated program. The effect, then, is to use one group of records to read the next group in, thus "pulling" the program in by its own bootstraps.

```
                ; THIS IS THE POPULAR FLECTRONICS SUPER-
                ; SIMPLE MONITOR.  COMMANDS ARE:
                ;     D XXXX   (DUMP FROM XXXX)
                ;     L XXXX   (LOAD FROM XXXX)
                ;     G XXXX   (GO TO XXXX)
                STACK  EQU  03FFH         ;YOUR STACK ORIGIN
                CRLF   EQU  0106H         ;YOUR CR, LF ROUTINE
                WRCHR  EQU  0103H         ;YOUR OUTPUT ROUTINE
                RDCHR  EQU  0100H         ;YOUR READING ROUTINE
                ;
                ; MONITOR ENTRY POINT
                ;
0040: 31FF03    PEMON: LXI  SP,STACK      ;INITIALIZE
      CD0601           CALL CRLF          ;ISSUE CARRIAGE RETURN,
      3E3F             MVI  A,'?'         ;   LINE FEED AND ?
      CD0301           CALL WRCHR
      CD0001           CALL RDCHR         ;AWAIT COMMAND
      E67F             ANI  07FH          ;STRIP OFF PARITY BIT
      FE4C             CPI  'L'
      CA6300           JZ   LOAD          ;"LOAD" COMMAND
      FE44             CPI  'D'
      CA7400           JZ   DUMP          ;"DUMP" COMMAND
      FE47             CPI  'G'
      C24000           JNZ  PEMON         ;ERROR
                ; THIS THE THE "GO" COMMAND PROCESSOR.  WE
                ; NOW EXPECT A 16-BIT DESTINATION ADDRESS
                ; AND THEN TRANSFER TO IT
005F: CDB700           CALL RDNUM
      E9               PCHL               ;"GO"
                ; THIS IS THE "LOAD" COMMAND PROCESSOR.  THE
                ; USER IS EXPECTED TO TYPE IN A 16-BIT ADDRESS
                ; FOLLOWED BY DATA BYTES TO LOAD INTO SUCCESSIVE
                ; LOCATIONS.  ANY NON-HEX CHARACTER SFPARATES
                ; THE BYTES FROM ONE ANOTHER.  ANY BYTE THAT IS
                ; TERMINATED WITH A COLON WILL BE IGNORED.
0063: CDCD00    LOAD:  CALL RDNR2         ;GET USER'S ADDRESS
      CDB700           CALL RDNUM         ;GET A BYTE
      FE3A             CPI  ':'
      CA6600           JZ   LOAD+3        ;SKIP IF FOLLOWED BY ':'
      7D               MOV  A,L           ;GET LAST TWO HEX DIGITS
      02               STAX B             ;STORE THEM AWAY
      03               INX  B
      C36600           JMP  LOAD+3        ;(GET W/ CPU RESET)
                ; THIS IS THE "DUMP" COMMAND PROCESSOR.
                ; THE USER IS EXPECTED TO SUPPLY A 16-BIT
                ; STARTING ADDRESS.  WHFNEVER THE LEAST-SIG-
                ; NIFICANT FOUR BITS OF THE ADDRESS OF THE
                ; NEXT BYTE ARE ZERO, THE CARRIAGE IS RETURNED
                ; AND THE ADDRESS IS PRINTED (FOLLOWED BY COLON).
0074: CDCD00    DUMP:  CALL RDNR2         ;GET STARTING ADDRESS
      79               MOV  A,C           ;CHECK TO SEE IF
      E60F             ANI  15            ;  NEW-LINE TIME.
      C2A200           JNZ  MORE          ;NO.  JUST PRINT
      CD0601           CALL CRLF          ;START NEW LINE
      78               MOV  A,B           ;GET MSB OF ADDRESS
      1F               RAR
      1F               RAR
      1F               RAR
      1F               RAR
      CDED00           CALL CVTAS
      78               MOV  A,B           ;GET 2ND DIGIT
      CDED00           CALL CVTAS
      79               MOV  A,C           ;GET 3RD DIGIT
      1F               RAR
      1F               RAR
      1F               RAR
      1F               RAR
      CDED00           CALL CVTAS
      79               MOV  A,C           ;DO LAST DIGIT
      CDED00           CALL CVTAS
      3E3A             MVI  A,':'         ;MARK ADDRESS SPECIALLY
      CD0301           CALL WRCHR
      3E20             MVI  A,' '
      CD0301           CALL WRCHR
00A2: 0A        MORE:  LDAX B             ;FETCH BYTE TO DUMP
      1F               RAR
      1F               RAR
      1F               RAR
      1F               RAR
      CDED00           CALL CVTAS         ;ISSUE MSB
      0A               LDAX B
      CDED00           CALL CVTAS
      3E20             MVI  A,' '
      CD0301           CALL WRCHR
      03               INX  B             ;ACQUIRE NFXT CELL
      C37700           JMP  DUMP+3
                ; THIS ROUTINE READS ONE OR MORE HEXADECIMAL
                ; DIGIT CHARACTERS ('0'..'9', 'A'..'F') IN
                ; AND ACCUMULATES A SIXTEEN-BIT NUMBER IN
                ; THE (H,L).  EACH NEW DIGIT IS SHIFTED INTO
                ; THE LEAST-SIGNIFICANT FOUR BITS OF THE (H,L).
                ; CONTROL IS RETURNED WHENEVER THE DEPENDENT
                ; ROUTINE ("RDHEX") SETS THE CARRY BIT TRUE.
00B7: 210000    RDNUM: LXI  H,0           ;START OFF VALUE AT 0
      CDD600           CALL RDHEX         ;GO GET A HEX DIGIT
      DAB700           JC   RDNUM         ;AWAIT HEX DIGIT
00C0: 29        RDNXT: DAD  H             ;SHIFT (H,L) LEFT 4
      29               DAD  H
      29               DAD  H
      29               DAD  H
      B5               ORA  L             ;PLACE NEW FOUR BITS IN
      6F               MOV  L,A
      CDD600           CALL RDHEX         ;GO GET NEXT DIGIT
      D8               RC                 ; THE NUMBER'S FINISHED
      C3C000           JMP  RDNXT         ;GO PROCESS NEXT DIGIT
                ; THIS ROUTINE READS A NUMBER VIA "RDNUM" AND
                ; PLACES IT INTO THE (B,C) PAIR
00CD: CDB700    RDNR2: CALL RDNUM         ;GET THE VALUE IN (H,L)
      44               MOV  B,H           ; THEN MOVE IT
      4D               MOV  C,L
      C9               RET
                ; THIS ROUTINE READS IN AN ASCII CHARACTER,
                ; STRIPS OFF PARITY AND EXAMINES IT FOR
                ; MEMBERSHIP IN THE HEX-DIGIT SET.  IF IT IS
                ; A HEX-DIGIT THE A-REGISTER IS LEFT AT THE FOUR
                ; BIT VALUE APPROPRIATE AND THE CARRY IS CLEARED.
                ; ANY OTHER CHARACTER IS LEFT UNTOUCHED AND THE
                ; CARRY IS SET.
00D3: D630      RDDIG: SUI  '0'           ;TRANSLATE '0'..'9'
      C9               RET
00D6: CD0001    RDHEX: CALL RDCHR         ;***ENTRY POINT***
      E67F             ANI  07FH          ;REMOVE PARITY BIT
      FE30             CPI  '0'
      D8               RC                 ;CHAR LESS THAN '0' (NOT HEX)
      FE3A             CPI  '9'+1
      DAD300           JC   RDDIG         ;IN RANGE 0..9
      FE41             CPI  'A'
      D8               RC                 ;BETWEEN 9 AND A
      FE47             CPI  'F'+1
      3F               CMC
      D8               RC                 ;NOT HEX CHARACTER
      D637             SUI  'A'-10        ;TRANSLATE 'A'..'F'
      C9               RET
                ; THIS ROUTINE CONVERTS A BINARY NUMBER IN THE
                ; LEAST-SIGNIFICANT BITS OF THE A-REGISTER TO
                ; AN ASCII CHARACTER REPRESENTING THE HEX VALUE
                ; AND THEN ISSUES IT AS OUTPUT.
00ED: E60F      CVTAS: ANI  15            ;ISOLATE FOUR BITS
      C630             ADI  '0'           ;SHIFT DIGITS INTO ASCII
      FE3A             CPI  '9'+1         ; SEE IF IT WAS 0..9
      DA0301           JC   WRCHR
      C607             ADI  'A'-'0'-10    ;CONVERT TO 'A'..'F'
      C30301           JMP  WRCHR
```


Using a Monitor. The ability to preserve a program for later recall is important, but it doesn't solve two major nuisances: (1) you still have to key in the program bit-by-bit the first time; and (2) every time you make an error in the program, you have to key in the changes, some of which may be traumatic and complex. One of the best ways to solve this kind of inconvenience is to provide a small *monitor*. A monitor is just another computer program, but one that is designed to make computer use more convenient. This program reads characters from a terminal (or a separate keyboard), with these characters specifying the bit patterns to put into memory.

The simplest monitor has three basic *commands:* Load, Dump and Go. A command is a single letter typed at a time when the monitor is not otherwise engaged in some activity. Typical commands are single letters like "L" for Load, "D" for Dump and "G" for Go. When you type in "L" you are directing the monitor program to accept keyboard input and load it into memory; "D" means you want to display contents of memory on your terminal or display device; "G" is your means of transferring control out of the monitor into the program you have previously loaded.

Most programmers now use the hexadecimal number system for communicating with the machine, although there are "pockets" of users of octal. Hex and octal are, of course, just shorthand notations for binary code. Hex digits allow us to specify four bits with one symbol, octal allows three. The hexadecimal digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. (The letters A through F stand for decimal equivalent values 10 through 15, respectively.) The monitor, for utmost simplicity, uses only hexadecimal digits for the specification of addresses and data byte values.

Each of the command letters (L, D or G) is followed by an address that specifies where to start. For the Load command, that address is where the first byte of data from the keyboard will be stored; for Dump, it is the address from which data will begin being displayed; for Go, it is the address that is to be placed into the CPU's program counter. Whenever the Go command's address has been supplied, control is transferred to that location. Whenever the Dump command's address has been given, data displaying will begin. However, after the Load command's address, the monitor will expect more bytes (one after another) to be loaded into successive locations in memory.

A small monitor for the 8080 microprocessor that you can use as a model is shown opposite. Each command is stopped by resetting the CPU, thus returning control to the top of the monitor. Notice some error correction conventions that have been instituted to save you some time: numbers are assumed to consist of any number of hex digits but if the monitor wants an address, only the least-significant four digits are used. Likewise, for a data byte, only the least-significant two hex digits are preserved. This means that if you've made an error, just keep typing. Hex digits end with any character that is not a hex digit; most people find that the space character is the most convenient.

News Items. The extremely popular 8080, originally from Intel, is now being supplied by other semiconductor makers as well. The TI TMS8080 is identical to the original 8080, which Intel no longer makes. Intel's newer device, the 8080A, is functionally identical but has better current drive capacity. Intel has another part, the 8080A-1, that'll go faster so that AMD's 9080 (which is supposed to run 50% faster than the original Intel part) will have a competitor. So, if you are using the 8080, be sure to check the diagrams to see that your part matches the requirements.

The new MOS Technology 6501 is destined to become a popular CPU among hobbyists, if only because of its dramatically low price ($20 at press time). The device is modelled after Motorola's 6800, although with some major differences. All of the Motorola support parts like memory I/O chips can be used with the 6501, so you can get on board quickly. The Motorola software, however, cannot be executed on the 6501 without revision.

The 6501 is capable of operation at twice the Motorola part's speed; some parts may operate three times as fast. The introduction of this part is likely to start the real price war that has been brewing in the microprocessor business. Even with the new support chip for the 8008 that Intel has announced, it seems unlikely that it can compete with the 6501 for hobbyist use. ◈