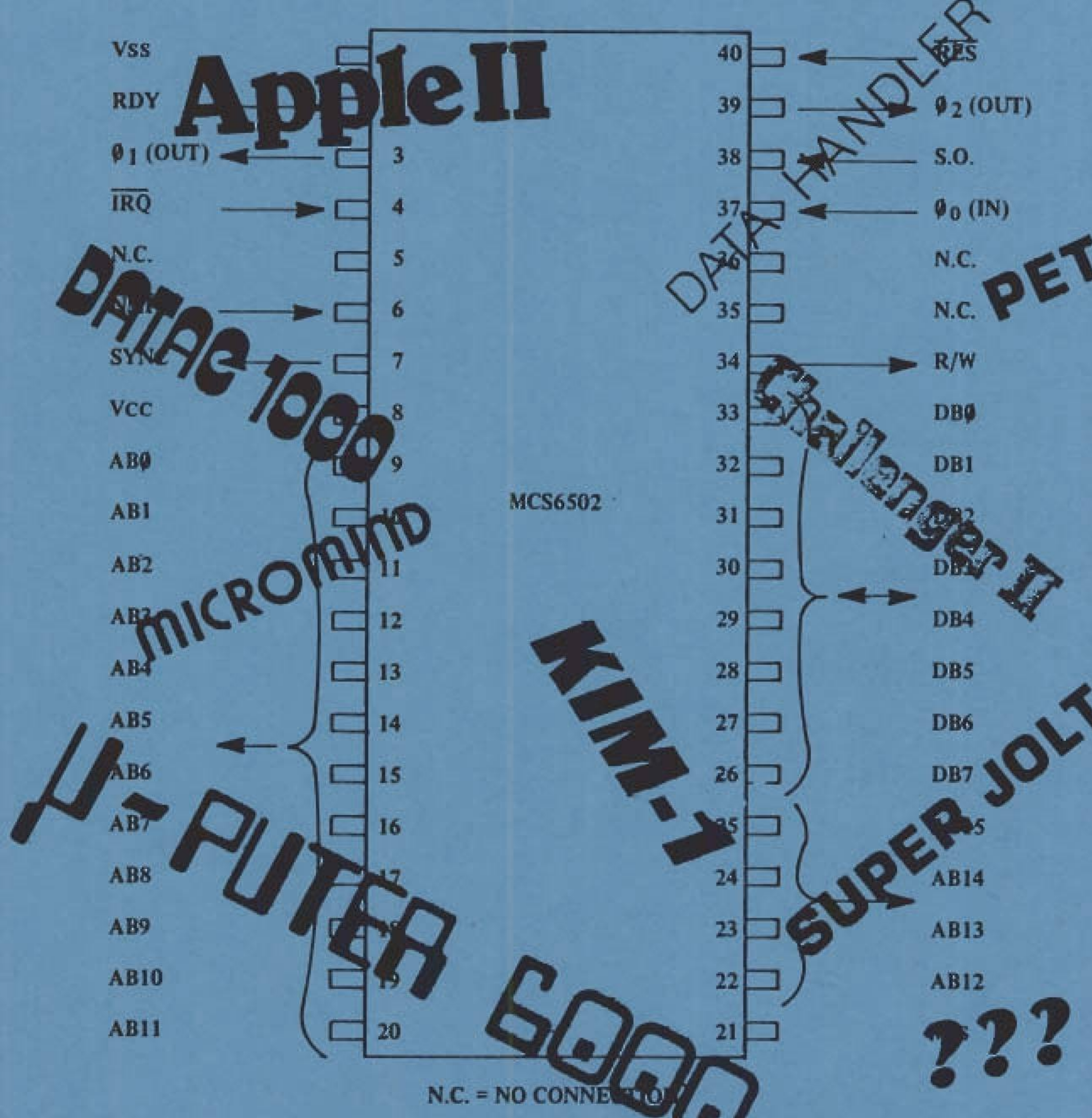
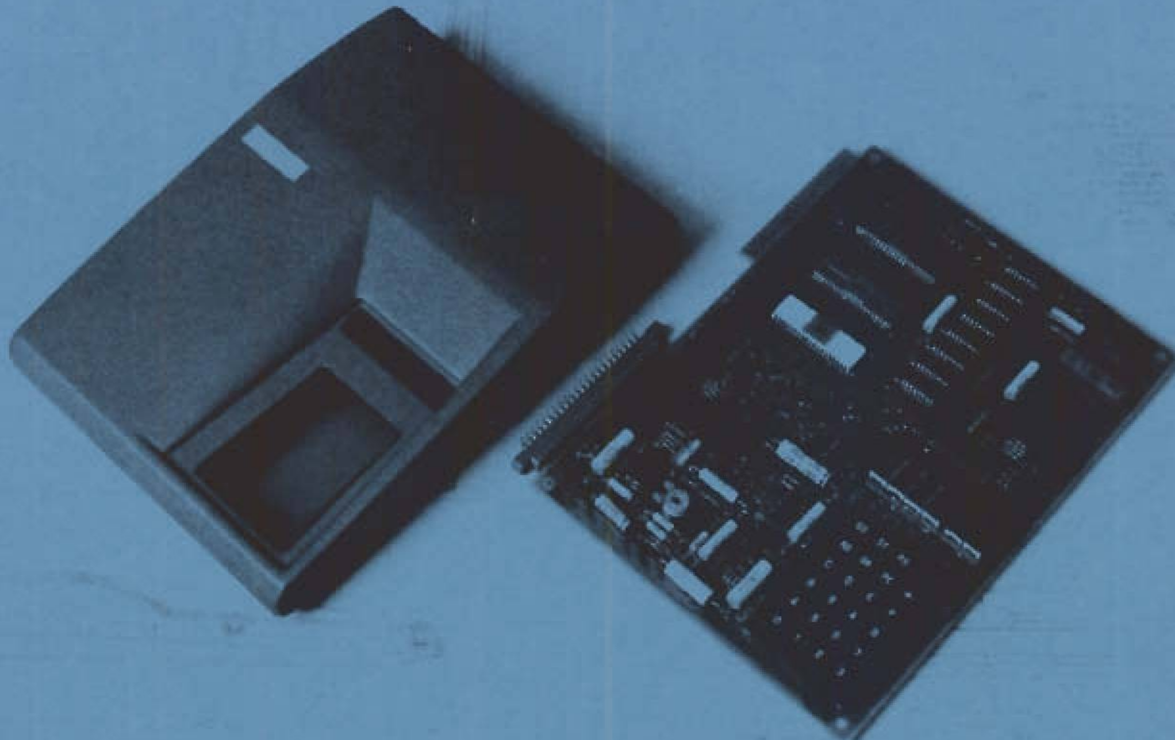
MICRO™

THE 6502 JOURNAL



NO 3 Feb - Mar 1978 $ 1.50

QUICK CHANGE ARTISTRY



ENGINEERED SPECIFICALLY FOR THE KIM-1 MICRO COMPUTER
- Protection of Chips and Other Components
- Viewing Angle of Readout Enhanced
- Improved Keyboard Position for Easier Operation

EASILY ASSEMBLED
- Absolutely No Alteration of KIM-1 Required
- All Fasteners Provided
- Goes Together in Minutes with a Small Screwdriver

ATTRACTIVE FUNCTIONAL PACKAGE
- Professional Appearance
- Four Color Combinations
- Improves Man/Machine Interface

MADE OF HIGH IMPACT STRENGTH THERMOFORMED PLASTIC
- Kydex 100 *
- Durable
- Molded-In Color
- Non-Conductive

AVAILABLE FROM STOCK
- Allow Two to Three Weeks for Processing and Delivery
- No COD's Please
- Dealer Inquiries Invited

TO ORDER: 1. Fill in this Coupon (Print or Type Please)
2. Attach Check or Money Order and Mail to:

NAME ______________________

STREET ______________________

CITY ______________________

STATE ____________ ZIP ____________

Please Ship Prepaid ____ SKE 1-1(s)
@ $23.50 Each
California Residents please pay
$25.03 (Includes Sales Tax)

the enclosures group

55 stevenson, san francisco 94105

Color Desired blue ☐ beige ☐ black ☐ white ☐

* TM Rohm & Haas

Patent Applied For

MICRO

FEBRUARY/MARCH 1978

ISSUE NUMBER THREE

Advertisers Index

MICRO is published bi-monthly by The COMPUTERIST, 8 Fourth Lane, So. Chelmsford, MA 01824. Robert M. Tripp Editor/Publisher. Controlled circulation postage paid at Chelmsford, Massachusetts.

Subscription Rate: $6.00 per year (six issues) in U.S.A.

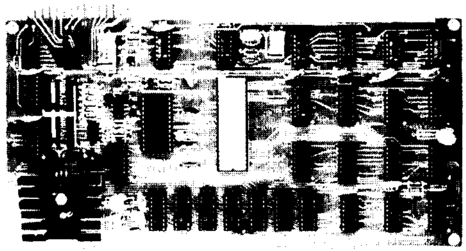
COMPUTER SHOP

288 NORFOLK ST. CAMBRIDGE, MASS. 02139

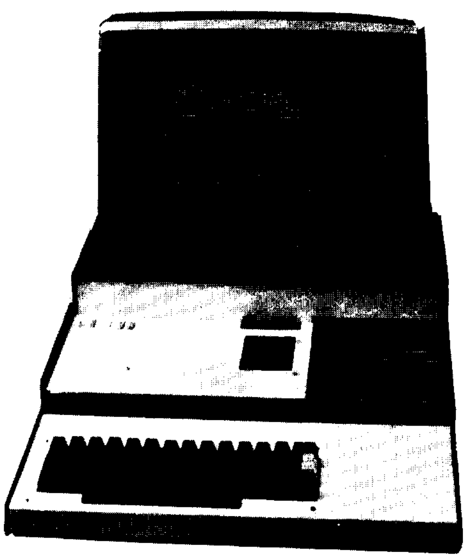
corner of Hampshire & Norfolk St. 617-661-2670

NOW WE HAVE OSI AND YOU CAN GET ONTO THE BUS

You can now adapt you KIM to the OSI Motherboard easily because it has buffers built into the Board. Application notes are available. You cant beat the $39.00 price for an 8 slot motherboard. Once you get onto their motherboard you can get their 4K memory board assembled for $125.00, or their 16 K board for $596.00 assembled. We can supply you with all the other goodies as well such as: Video Graphics, Prom and parallel I/O,cassette and Analog I/O. Or you can switch to their complete Challenger II P for $598.00. It has an 8K ROM Basic , Video 32x64 Cassette,, and a real keyboard. Order Now!



The CS100 VIDEO TERMINAL BOARD IS A 16 LINE BY 64 CHARACTER DISPLAY GENERATOR WITH CURSOR CONTROL AND EDITING Connect a 5V. ASCII Keyboard to it, a Regulated 5 Volt, Unregulat-8 Volts, or 8 Volts AC, and your KIM Teletype port to it along with a video monitor and away you go with all the convenience of a Video Terminal on your KIM.



C2-0 Board assmb. Serial I/O ROM Basic	298.00
C2-1 above with Cabinet &Pwr Supp.	429.00
C2-4P Challenger II P Kbd & Video	598.00
CM-1 4K 1MHz. Memory	125.00
CM-2 4K 2MHz. Memory	149.00
CM-3 16K 1.5 MHz. Ultra Low Pwr. Mem	596.00
CA-7 I/O Board	399.00
480 Backplane..Motherboard	39.00
all above are OSI products available from us.	
VF8 4K Memory assembled & tested.	129.00
for low power RAM add	10.00
same in kit form	74.50
full set of sockets for Kit	10.00
VF8 Motherboard buffered for 4 Boards	65.00
Connector Assembly for KIM to VF8	20.00
8K S100 Memory Board with instructions K	165.00
same but fully assembled and tested	199.00
CS100 Cabinet cut out for KIM	129.00
3 Connector S100 Motherboard Assembly	75.00
CGRS S100 TIM Kit	129.00
CGRS S100 6502 CPU Kit	179.00
CGRS S100 Front Panel Kit	129.00
XITEX Video Terminal Board 16X64 K	155.00
XITEX Video Terminal Board Assembled	185.00
KIM-1	245.00
CS100 with CGRS,Xitex,16KRAM,TV,KB	1529.00
Same but Assembled	1989.00
PS-5 Pwr Supp. 5V5A9V1A-12V1A6x6X2	75.00
PS-5 Assembled	90.00
Total of Order..Circle Items wanted.	$..............
Mass. Residents Sales Tax 5%	$..............
Shipping, 1%($2.00 min.)	$..............
Total Remittance or Charge	$..............

BAC,VISA,MC NO. ....

SIGNATURE..

NAME..

ADDRESS..

CITY..............................STATE.........ZIP......

EMPLOYING THE KIM-1 MICROCOMPUTER AS A TIMER AND DATA LOGGING MODULE

Marvin L. De Jong
Dept. of Mathematics-Physics
The School of the Ozarks
Point Lookout, MO 65726

The interval timers on the 6530 on the KIM-1 microcomputer provide a convenient way to measure the time between two or more events. Such events might include the start and end of a race, the exit of a bullet from a gun and its arrival at a measured distance along its trajectory, the interruption of light to a series of phototransistors placed along the path of a falling object, an animal arriving at this feeding station, the arrival of telephone calls, etc. Some of these measurements will be described in more detail below. Each event must produce a negative pulse which the microcomputer detects and records the time at which the event occurred. The time is stored in memory and later displayed on the 6-digit KIM display.

Description of the Programs

The data logging, timer, and display programs are listed in Tables 1, 2, and 3, respectively. The programs must be used together for the applications described in this paper, but each might be used with other applications, for example pulse generators, frequency counters, temperature logging, light flashing, etc. The events to be timed must produce either a one-shot pulse (positive-zero-positive) whose duration is at least 50 microseconds or a zero to positive transition which must be reset to zero before the next event. These signals are applied to pin PA0 on the KIM applications connector. The programs could easily be modified to detect positive pulses.

The first pulse starts the timer which continues to operate on an interrupt basis. The first pulse is not recorded by the data logging program since it corresponds to t = 0. Successive pulses cause the data logging program to store the six digit time counter in memory. The number of events (not counting the first event) N, to be timed must be stored in location 0003. Remember to convert the number of events, N, to base 16 before entering it in memory. As the program is written, N must be less than 75. = 4B hex.

The function of the timer program is to load the interval timer, increment the six digit time counter, and return to the data logging program. At the end of each timing period the timer causes an interrupt to occur (pin PB7 on the application connector must be connected to pin 4 on the expansion connector), the computer jumps to the timer program, does its thing, and returns to the main data logging program to wait for events.

Table 4 lists several timing intervals which are possible and the numbers which must be loaded into the various timers to produce the given interval. For example, if one wishes to measure time in units of 100 microseconds, then 49 hex must be stored in the divide-by-one counter whose address is 170C. In this case, the numbers which appear on the display during the display portion of the program represent the number of 100 microsecond intervals between the first event and the event whose time is being displayed. To put it another way, multiply the number on the display by 0.0001 to get the time in seconds. The other possibilities listed in the table are treated in the same way.

When all N events have been logged, the program automatically jumps to the display program. When one is ready to record the data, key #1 on the keyboard is depressed. The time of each event, excepting the first which occurred at t = 0 is displayed on the six digit readout for several seconds before the display moves to the time of the next event. This gives the experimenter time to record the data on paper. If more time is required, increase the value of the number stored in location 0289.

Table 4 also lists the measured time interval and gives the percent error between the stated interval (say 100 microseconds) and the actual measured interval (99.98 microseconds). The measurements were made by connecting a frequency counter (PASCO SCIENTIFIC Model 8015) to pin PB7 while the program was running and after the first event had started the timer. If greater accuracy is required for the 10 millisecond and 100 millisecond intervals, then experiment with putting NOP instructions between the PHA instruction and the LDA TIME instruction in the timer program.

Experiments and Applications

The simplest application for the program is a simple stopwatch with memory. Any suitably debounced switch can be used. See pages 213 and 280 in CMOS COOKBOOK by Don Lancaster, published by Howard W. Sams & Co., Inc., 4300 West 62nd St., Indianapolis, Indiana 46268 for some suitable switching circuits.

Being a physics teacher, I originally designed the program to collect data for an "acceleration of gravity" experiment in the introductory physics lab. The technique may be applicable to other problems so it is described herein. Nine phototransistors (Fairchild FPT 100 available from Radio Shack) were mounted on a meter stick at 10 cm intervals. An incandescent (do not try fluorescent lighting) 150 watt flood lamp provided the illumination. The interface circuit is shown in Figure 1.

The 555 timer serves as a Schmitt trigger and buffer which produces a negative pulse when an object passes between the light and the phototransistor. The 500 kilo ohm potentiometer is adjusted so that an interruption of the light to any of the phototransistors increases the voltage at pin 2 of the 555 from about 1.5 volts to at least 3.5 volts; a very simple adjustment which should be made with a VTVM or other high impedance meter.

In the case of a simple pendulum, the relationship between the period and the amplitude can be investigated by allowing the pendulum to "run down" while logging the times when the bob interrupts the light to a single phototransistor. With only one phototransistor the timer-data logging program can also be used as a tachometer if a rotating system of some kind is involved.

Lancaster, in the CMOS COOKBOOK, describes a tracking photocell pickoff which could be used in conjunction with the program for outdoor races and other sporting events. See page 346 in the "COOKBOOK". A simple light beam-phototransistor system could be placed in a cage and the apparatus would record the times at which an animal interrupted the beam, giving a measurement of animal activity.

If you want to measure the muzzle velocity of your rifle or handgun, you will have to be more devious. First, I would modify the program so that one pin, say PA0, is used to start the timing while another pin, say PB0, is used to stop the timing. This can be accomplished by changing instructions 0226 and 022D in Table 1 from AD 00 17 to AD 02 17. Then I would use a fine wire foil to hold the clock input of a 7474 flip-flop low until the wire foil was broken by the exit of the bullet from the gun. The Q output going high would start the timing, so it would be connected to PA0. To end the timing one could use a microphone to detect a bullet hitting the backstop. Of course, the microphone signal would have to be amplified and used to trigger say the other flip-flop of the 7474 to signal the second event. So as not to take all your fun away, that is the last hint except that the distance between start and stop should be at least 10 feet. Please be careful.

I would like to acknowledge the education and inspiration I received at an NSF Chautauqua Type Short Course and a KIM workshop, both conducted by Dr. Robert Tinker.

[Editor's Note: For a related KIM-1 application, see "A Simple Frequency Counter Using the KIM-1", by Charles R. Husbands, on page 29 of this issue.]

Table 1

Data logging program

```
                  DLOG   ORG    $0200

                  LOW    *      $0000
                  MID    *      $0001
                  HIGH   *      $0002
                  N      *      $0003
                  LO     *      $0003
                  MI     *      $0053
                  HI     *      $00A3
                  INH    *      $00F9
                  POINTL *      $00FA
                  POINTH *      $00FB
                  KEY    *      $0271
                  PAD    *      $1700
                  GETKEY *      $1F6A
                  SCANDS *      $1F1F

0200 78           INIT   SEI          DISABLE INTERRUPT
0201 F8                  SED          SET DECIMAL MODE
0202 A2 00               LDXIM $00    SET X = 0
0204 A9 50               LDAIM $50    SET INTERRUPT VECTOR = 0250
0206 8D FE 17            STA   $17FE
0209 A9 02               LDAIM $02
020B 8D FF 17            STA   $17FF
020E A9 FF               LDAIM $FF    INIT COUNTER BY STORING 255 (FF)
0210 85 00               STAZ  LOW    INT THE THREE, TWO DIGIT
0212 85 01               STAZ  MID    MEMORY LOCATIONS OF THE
0214 85 02               STAZ  HIGH   COUNTER
0216 AD 00 17     START  LDA   PAD    READ INPUT PIN PA0
0219 29 01               ANDIM $01    LOGICAL AND WITH PA0
021B D0 F9               BNE   START  LOOP IF PIN IS 1
021D AD 00 17     FLIP   LDA   PAD    IF PIN IS NOT 1, READ AGAIN
0220 29 01               ANDIM $01    LOGICAL AND WITH PA0
0222 F0 F9               BEQ   FLIP   LOOP IF PIN IS 0
0224 58                  CLI          ELSE, ENABLE INTERRUPT AND JUMP TO
0225 00                  BRK          TIMER PROGRAM THEN RETURN
0226 EA                  NOP          PADDING FOR BRK COMMAND
0227 AD 00 17     CHEK1  LDA   PAD    THESE INSTRUCTIONS ARE THE SAM
022A 29 01               ANDIM $01    AS THE START AND FLIP SEQUENCE
022C D0 F9               BNE   CHEK1
022E AD 00 17     CHEK2  LDA   PAD
0231 29 01               ANDIM $01
0233 F0 F9               BEQ   CHEK2
0235 E8                  INX          INCREMENT X FOR EACH DATA POINT
0236 A5 00               LDAZ  LOW    COUNTER CONTENTS ARE STORED IN A
0238 95 03               STAZX LO     SEQUENCE OF LOCATIONS INDEXED
023A A5 01               LDAZ  MID    BY X
023C 95 53               STAZX MI
023E A5 02               LDAZ  HIGH
0240 95 A3               STAZX HI
0242 E4 03               CPXZ  N      COMPARE X TO N.  RETURN TO CHEK1
0244 D0 E1               BNE   CHEK1  IF X IS LESS THAN N
0246 78           DISPLA SEI          ELSE GO TO DISPLAY AFTER
0247 4C 71 02            JMP   KEY    DISABLING INTERRUPTS
```

```
                    TIMER   ORG    $0250

                    TIME    *      $0049
                    TIMEX   *      $170C
                    LOW     *      $0000
                    MID     *      $0001
                    HIGH    *      $0002

0250 48             INTRPT PHA           PUSH ACCUMULATOR ON STACK
0251 A9 49                 LDAIM TIME    START TIMER FOR 49(16) CYCLES
0253 8D 0C 17              STA   TIMEX
0256 A9 01                 LDAIM $01     INCREMENT COUNTER BY ADDINT 1
0258 65 00                 ADCZ  LOW     TO THE TWO LOW DIGITS
025A 85 00                 STAZ  LOW     AND STOR RESULT
025C A9 00                 LDAIM $00     ADD CARRY FROM PREVIOUS
025E 65 01                 ADCZ  MID     ADDITION TO MID DIGITS.  IF
0260 85 01                 STAZ  MID     CARRY OCCURS FROM THE TWO MID
0262 A9 00                 LDAIM $00     FROM THE TWO MID DIGITS, THEN
0264 65 02                 ADCZ  HIGH    ADD THIS TO THE TO HIGH DIGITS
0266 85 02                 STAZ  HIGH
0268 68                    PLA           PULL ACCUMULATOR FROM STACK
0269 40                    RTI           RETURN TO DATA LOGGER
```

Table 2

Timer program

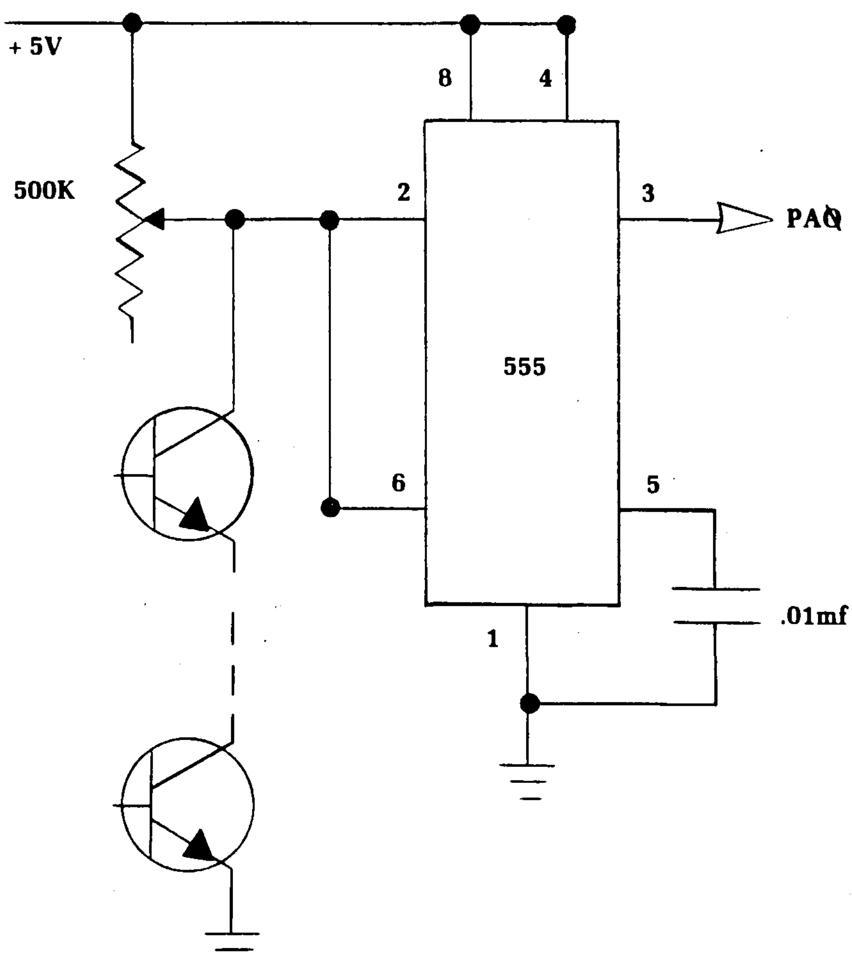


Figure 1

Interface circuit using up to 10 phototransistors. The dashed line represents other phototransistors. The time at which the light to any of the phototransistors is interrupted is recorded by the timer-data logging program.

Table 3

Display program

```
                    DISPLA ORG   $0271

                    N      *     $0003
                    LO     *     $0003
                    MI     *     $0053
                    HI     *     $00A3
                    INH    *     $00F9
                    POINTL *     $00FA
                    POINTH *     $00FB
                    INIT   *     $0200
                    TIME   *     $1707
                    GETKEY *     $1F6A
                    SCANDS *     $1F1F

0271 20 6A 1F  KEY    JSR   GETKEY JUMP TO KIM KEYBOARD MONITOR
0274 C9 01            CMPIM $01    TEST VALID INPUT
0276 D0 F9            BNE   KEY    IF NOT, WAIT FOR INPUT
0278 A2 01            LDXIM $01    INIT X REGISTER TO INDEX
027A B5 03     NXPNT  LDAZX LO     DATA POINTS
027C 85 F9            STAZ  INH    PUN IN KIM DISPLAY REGISTERS
027E B5 53            LDAZX MI
0280 85 FA            STAZ  POINTL
0282 B5 A3            LDAZX HI
0284 85 FB            STAZ  POINTH
0286 8A               TXA          SAVE X WHILE IN SUBROUTINE BY
0287 48               PHA          PUSHING IT ON THE STACK
0288 A0 10            LDYIM $10    TIME TO DISPLAY EACH POINT
028A 98        AGN    TYA          SAVE Y WHILE IN SUBROUTINE BY
028B 48               PHA          PUSHING IT ON THE STACK
028C A9 FF            LDAIM $FF
028E 8D 07 17         STA   TIME
0291 20 1F 1F  REPEAT JSR   SCANDS SCANDS IS KIM ROUTINE WHICH
0294 AD 07 17         LDA   TIME   DISPLAYS DATA IN 00F9, 00FA
0297 30 03            BMI   OVER   AND 00FB.  REPEATED JUMPS TO
0299 4C 91 02         JMP   REPEAT SCANDS PRODUCES A CONSTANT DISPLAY
029C 68        OVER   PLA          RESTORE Y REGISTER
029D A8               TAY
029E 88               DEY          DECREMENT Y BY 1 AND REPEAT
029F F0 03            BEQ   HOP    DISPLAY UNTIL Y = 0
02A1 4C 8A 02         JMP   AGN
02A4 68        HOP    PLA          RESTORE X REGISTER
02A5 AA               TAX
02A6 E0 03            CPXIM N      COMPARE X WITH N.  IF X IS LESS
02A8 F0 04            BEQ   BEGIN  THAN N INCREMENT X AND DISPLAY
02AA E8               INX          NEXT POINT.  ELSE, RETURN TO
02AB 4C 7A 02         JMP   NXPNT  THE BEGINNING
02AE 4C 00 02  BEGIN  JMP   INIT
```

Table 4

Timing intervals for the program.

Time Interval	Value	Address	Measured Interval	% Error
100 microsec	49	170C	99.98 microsec	0.02%
1 millisec	7A	170D	0.9998 millisec	0.02%
10 millisec	9C	170E	10.007 millisec	0.07%
100 millisec	62	170F	100.5 millisec	0.5%

MACHINE LANGUAGE USED IN "LUDWIG VON APPLE II"

C. R. (Chuck) Carpenter W5USJ
2228 Montclair Place
Carrollton, TX 75006

As an Apple II owner, I found the article "Ludwig von Apple II" (by Marc Schwartz, MICRO #2, page 19) quite interesting. The machine language routine used by Marc is put into the BASIC program by use of the POKE statement and I was curious to see the type of program used to activate the Apple II on-board speaker. To do this, I converted the decimal values used for the POKE statements into HEX with my TI Programmer. Then I loaded the values into the computer using the system monitor commands that are part of the Apple II functions.

Once I had the program loaded, I used the monitor commands to list an assembled version of the routine, as shown in Figure 1. The assembler provides a listing of the program and the mnemonics used with the machine language opcodes. This made it easier to determine what was happening in Marc's program. At this point I wanted to see what would happen if I ran the program by itself - as a machine language rouine only.

```
0000-   0F          ???
0001-   00          BRK
0002-   AD 30 C0    LDA   $C030
0005-   A5 00       LDA   $00
0007-   20 A8 FC    JSR   $FCA8
000A-   A5 01       LDA   $01
000C-   D0 04       BNE   $0012
000E-   C6 18       DEC   $18
0010-   F0 05       BEQ   $0017
0012-   C6 01       DEC   $01
0014-   4C 02 00    JMP   $0002
0017-   60          RTS
0018-   00          BRK
0019-   00          BRK
001A-   05 4B       ORA   $4B
001C-   B6 00       LDX   $00,Y
001E-   0F          ???
001F-   08          PHP
0020-   00          BRK
0021-   28          PLP
```

Figure 1

Because it is somewhat easier to call the routine from a BASIC routine, I entered the BASIC routine shown in Figure 2. This way I could also change the values stored in memory location $0000 by using the POKE statement. To initialize the beginning of the routine, I entered a value of $05 into location $0000. According to Marc, this would produce a high frequency output tone and this turned out to be the case.

Now that I had everything set up, I was curious to see why the duration of playing time is not the same for the different tones. To start with, I entered the program with 3 different values at location $0000. As I ran the program I timed the length of playing with a stop watch. The value of 5 played for .18 min., 10 played for .45 min. and 15 played for .85 min. This was in agreement with Marc's findings. As it turns out, the length of time a particular frequency plays is a function of the duration of a cycle. The output continues for a number of cycles and the shorter cycles (higher frequencies) get done sooner. To get the correct musical timing you would need to include variable delay time for each note played. (The time between zero crossings adds up to the same total time per note.)

```
>LIST
  10 POKE 0,5
  99 END

>CALL 2

>10 POKE 0,10
>RUN

>CALL 2

>10 POKE 0,15
>RUN

>CALL 2
```

Figure 2

THE PET VET TACKLES DATA FILES

Charles Floto
267 Willow Street
New Haven, CT 06511

Several people have contacted the PET Vet about their difficulties in recording data files on tape and reading the information back in. Preliminary information on PET BASIC lists the commands to be used, but doesn't tell how to put them together. This makes for a frustrating situation, especially as file handling should be one of the PET's strong points.

The following program is offered as a starting point for development according to your specific application. Reading and writing have been combined in one program for two reasons. First, modifications to one process may call for corresponding changes in the other. Second, this minimizes the need to juggle two cassettes while saving programs on one and data on the other. I recommend that a separate cassette be used for data storage. If you use this program please save it on tape before you try to run it. I have found that while I'm experimenting with data files, the PET is especially liable to go out of control, forcing me to turn off the power. The same memory location that controls the tape drive apparently controls a function essential to BASIC.

To write a data file load this program, have a blank cassette in the tape drive and type RUN. Line 50 clears the screen. Lines 60-300 build a string consisting of: a file name or record number followed by two asterisks; data to be saved that may be broken into data fields by delimiters of your choice; and three consecutive backslashes that mark the end of the record. Lines 90 and 100 cause the keyboard to be read until a key is struck. Then 105 echoes it to the screen and 110 adds it to the string. Use of GET rather than INPUT allows the data file to contain commas and carriage returns. Line 190 warns when C$ is approaching the maximum size; you may wish to have a later or less frequent warning. At the end of the record type three backslashes. These will be detected in line 300, causing 320 to be executed rather than going back to 90 for another character.

Lines 320-400 write C$ onto the tape. You will be instructed (on the screen) to press play and record on the tape drive if you have not already done so. In line 320 the first two numbers indicate that device #1 is tape drive 1. The third 1 indicates a write operation. Compare this to line 1000 where the 0 indicates a read command.

Line 450 provides for creation of the next record in the file. To create the last record simply input the record number and type three backslashes. Then, after it has been written, BREAK IN 500 will appear on the screen.

At this point you're ready to rewind the tape and type RUN 900. Lines 910 to 990 initialize 256 empty strings. Lines 1000-1090 read the tape and build up C$ until three consecutive backslashes are found. Line 2000 prints what has been read while 2850 displays available memory. Then in 3000-3020 C$ is broken down into its individual elements. These can be manipulated further by adding your own lines between 3050 and 9000. Line 9000 will head back to read the next record unless 3050 has detected the last record in a file.

To record numeric data generated in a program rather than entered from the keyboard it must be converted to a string with the STR$ function. Then when it's read back the VAL function can be used on data fields representing numbers. For example, N=VAL(B$(8)+B$(9)+B$(10)) might be used if you knew the eighth, ninth and tenth elements of C$ represented a three-digit number. Of course, it usually won't be nearly so simple as that.

```
50 PRINT CHR$(147)
60 PRINT "ENTER FILE NAME OR RECORD #"
70 INPUT C$
80 C$=C$+"**"
90 GET A$
100 IF A$=""THEN 90
105 PRINT A$;
110 C$=C$+A$
190 IF LEN(C$)>200 THEN PRINT 255-LEN(C$); "BYTES AVAILABLE"
300 IF RIGHT$(C$,3)<>"\\\" THEN 90
320 OPEN 1,1,1,"NAILFILE"
350 PRINT#1,C$
400 CLOSE 1
450 IF RIGHT$(C$,5)<>"**\\\" THEN 50
500 STOP
900 DIM B$(255)
910 FOR J=1 TO 255
920 B$(J)=""
930 NEXT J
990 C$=""
1000 OPEN 1,1,0,"NAILFILE"
1010 GET#1,A$
1020 C$=C$+A$
1030 IF A$<>"\" THEN SL=0:GOTO 1010
1040 SL=SL+1
1050 IF SL<3 THEN 1010
1090 CLOSE 1
2000 PRINT C$
2850 PRINT FRE(0); "BYTES FREE"
3000 FOR J=1 TO LEN(C$)
3010 B$(J)=MID$(C$,J,1)
3020 NEXT J
3030 PRINT FRE(0); "BYTES FREE"
3050 IF RIGHT$(C$,5)="**\\\" THEN END
9000 GOTO 910
```

If you have any problems with specific applications of your PET, drop me a note, preferably giving a phone number where you can be reached evenings and weekends. I'd also be interested to see any information you've been able to pry out of Commodore or discover on your own.

Charles Floto

The PET Vet

The PET Shop

I have five game programs for the PET. I will trade one-for-one with anyone else for other PET programs. Those wishing to trade should send cassettes to me and I will send mine in return.

The games are:

GUESS
AIR ACE
BRACKETS
COMPU-ART
LUNAR LANDER

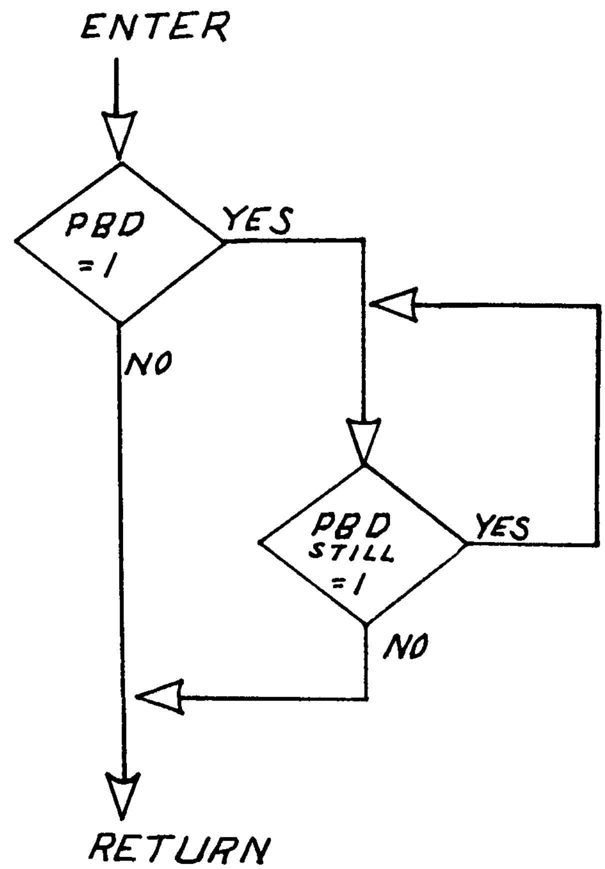
Evan H. Foreman
P.O. Drawer F
Mobile, AL 36601

HOLD THAT DATA

Gary L. Tater
7925 Nottingham Way
Ellicott City, MD 21043

Many programs could be enhanced if the user could stop the data on the video terminal by pressing a key. For instance, during a disassembly or a long directory program, it would be handy to be able to stop new data from coming onto the screen so that the existing data could be carefully examined. This note presents a short subroutine for the 6502 microprocessor which senses when the break key on the terminal is depressed and causes the program to loop until the break key is released. The break key, when depressed, holds the RS 232 line to the computer at a constant logic one.

The flow chart for this subroutine is shown in Figure 1. Typically this routine would be used after a line of data is printed on the CRT. The machine language program is relocatable and can be put in ROM. To use the routine, make a subroutine jump (JSR) to the first location: STOPB. As written, the routine functions with TIM. It can be modified to work with the KIM in either terminal or keypad mode. Using the flow chart and program, you should be able to modify the subroutine as necessary to meet your requirements.



Break Key Test
Subroutine Flow Chart

Figure 1

```
                              ORG   $0500

                    PBD       *     $6E02  PORT B FOR TIM

0500 A9 01          STOPB     LDAIM $01    SETUP TEST BIT
0502 2C 02 6E                 BIT   PBD    TEST I/O PORT
0505 D0 01                    BNE   ENDB   IF 1, BREAK KEY IS ON
0507 60                       RTS          NOT A BREAK KEY

0508 2C 02 6E       ENDB      BIT   PBD    WAIT FOR BREAK KEY
050B D0 FB                    BNE   ENDB   TO BE RELEASED
050D 60                       RTS          THEN RETURN
```

Figure 2

THE MICRO SOFTWARE CATALOG

A Call for Information

MICRO would like to present, in the April-May issue, a comprehensive catalog of software currently available for 6502 systems. While we are aware of many of the programs and packages that are on the market, we do not have anything like a complete listing.

If you have 6502 software for sale (or exchange or free), please send us a complete description which includes ALL of the following information:

Name of program:
6502 systems it works on:
Memory locations required:
Language (Assembler, BASIC,...):
Hardware required:
Description of program:
Number of copies sold to date:
Price:
Package contents (Cassette, Source listings, paper tape, ...):
Ordering information:
Company/Author Name and Address:

Please observe the following requirements in submitting your listings:

Programs must be currently available, and at least 25 copies of the program should have been distributed.

Description should be limited to about one hundred words.

All information should be included.

Material must be received by March 20, 1978 in order to be included.

MICRO READERS CAN HELP

Since all authors of 6502 software are not MICRO subscribers, if you have a favorite program you think should be included, send a copy of this call for information to him.

MICRO
P.O. Box 3, S Chelmsford, MA 01824

PRINTING WITH THE APPLE II

C. R. (Chuck) Carpenter W5USJ
2228 Montclair Place
Carrollton, TX 75006

Hardcopy output from your Apple II is a practical reality. All you need is a TELPAR thermal printer, a simple one-transistor adapter circuit and a machine language printing routine. The printing routine slows the data rate down to 110 (or 300) baud and directs the data stream to ANO (the game paddle connector - annunciator output, port zero). I have the TELPAR PS-40-3C (now PS-48) connected to my Apple II and I am printing everything from Biorhythms to Manpower Planning programs. Here are the details for hooking up the printer.

The TELPAR PS-40-3C

The PS-40 (Photo 1) is a 48 column thermal printer using 5.5 inch width paper. The model I have is a 3 chip F8 controlled unit. The current, more compact models use a single chip F8/3870. Inputs are provided for serial TTL, RS 232 and 20 MA current loop. You can also connect a parallel port to the printer and software controllable options are available. The printer can be used as the only I/O if a keyboard is connected as the parallel source. The paper is not too expensive at $3.00 per 164 foot roll.

Power supply voltages are critical and several are required. (This is the only shortcoming I found with this general purpose printer.) Good regulation is a must from your power supply. Especially the printhead supply voltage (16). Excessive positive deviations here can blow the printhead. Telpar can supply a switching type power supply that will do the job. The connections to the 56 pin edge connector are shown in Figure 1. The connector actually has numbers and letters to designate pins. Somewhere along the line, numbers were assigned to both sides. Be sure you transpose the numbers correctly and connect it to the circuit board properly. Telpar has good repair service, but it still takes time.

Interface Adapter

All that is needed to connect the Apple II to an RS 232 printer input is the adapter circuit shown in Figure 2 (from an Apple application note). I built this circuit on a 16 pin IC header and plugged it in. There is some inconvenience if you want to use the game paddles too, but I think there is a way around this if you choose to do some rewiring.

You can get the -12 volts for this circuit from the main power connector. A short lead and a small connector pin will work. If the pin is small enough, it will slide down inside the -12 volt terminal on the power connector. There are other places like the keyboard where -12 volts is also available. Use caution making this connection.

Making it Print

Now the only part left is a way to get the data slowed down and directed to the ANO output port. Apple has taken care of this detail with the routine shown in Figure 3. You can key in this routine and save it on tape. Each time you have a printing task the program is easily loaded using Apple's system monitor commands. I've used it with machine language programs and both forms of BASIC: Apple's Integer BASIC and Applesoft Floating Point 8K BASIC. The routine is called as follows:

$380G and RETURN in machine language

CALL 896 in Apple Integer BASIC

X=USR(896) in Applesoft 8K BASIC

Note: A line number is not needed to call the print routine. (380 hex = 896 decimal).

Using RESET will stop the print routine in machine language and in Apple Integer BASIC (return to BASIC with the soft entry CONTROL-C). With Applesoft in RAM, exiting via RESET and re-entry the soft way with 0G works sometimes but usually causes a glitch in BASIC and messes up the program. I avoid this problem by waiting to do any printing until the last thing. Any further changes are made at the slower speed. I would speculate that things like this will clear up when Applesoft is in ROM. I'm still looking for a way to get out of the print routine directly from the BASIC program.

The Tale is Told

As I indicated at the beginning, I'm printing most anything I want to. The 5.5 inch paper width presents some limitations but most programs can be formatted to work okay. There are several features and details I've alluded to but an article to do them justice would take several issues of MICRO to cover. Telpar has a technical paper that describes them and would be happy to send you one. For a simple, effective, general purpose printer, I have not found a better choice than my Telpar thermal. I think you would find it a good choice too.

For more info, write to:

Telpar Inc.
4132 Billy Mitchell Road
P.O. Box 796
Addison, TX 75001

[Editor's Note: One problem I have found with this thermal printer is that the print is light blue. This can cause great difficulty if you want to copy the output since most xerox-type copiers and many plate-making films are "blind" to light blue.]

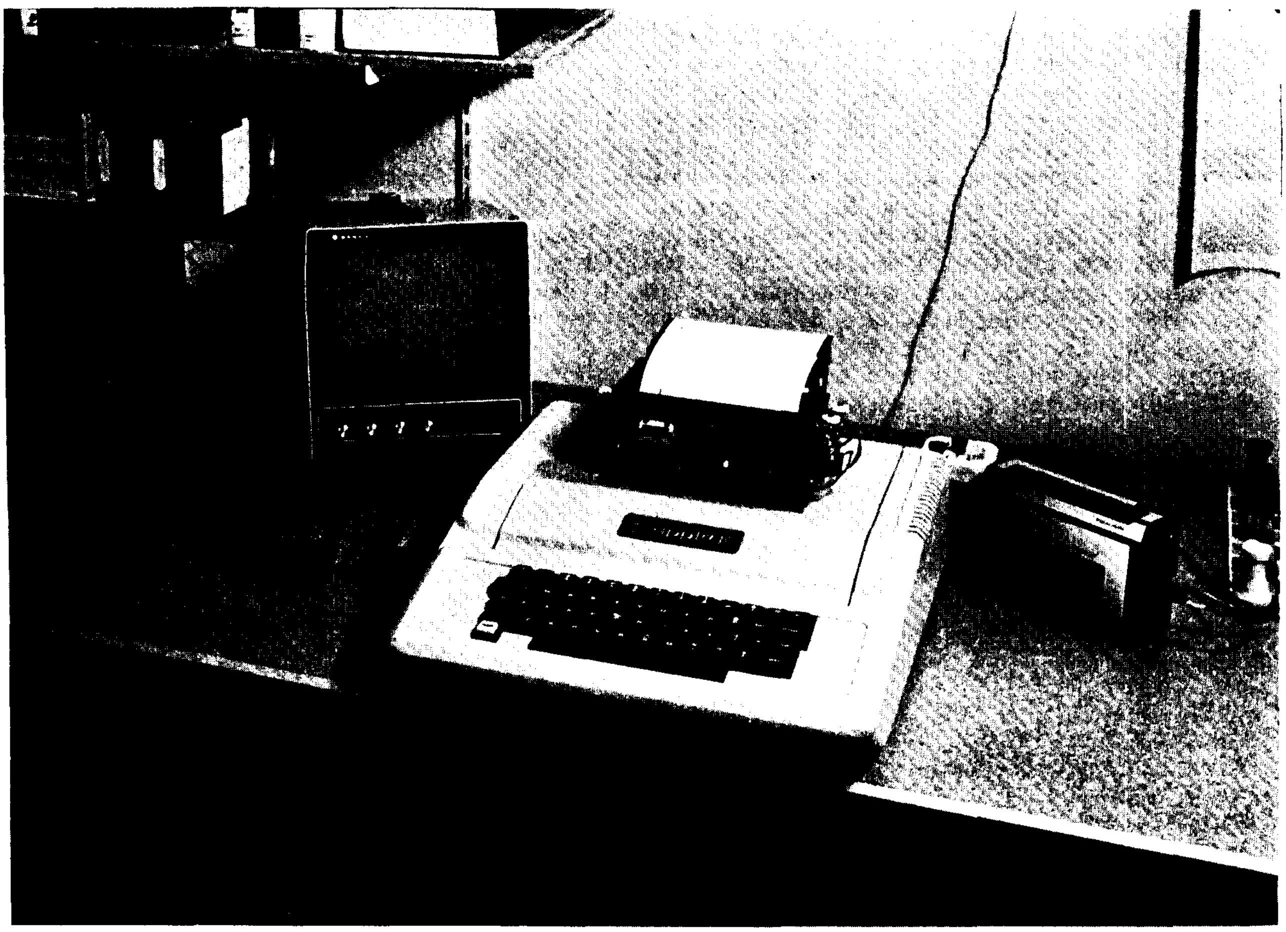


Photo 1 (by Jim Chamberlain)

APPLE II and TELPAR Thermal Printer

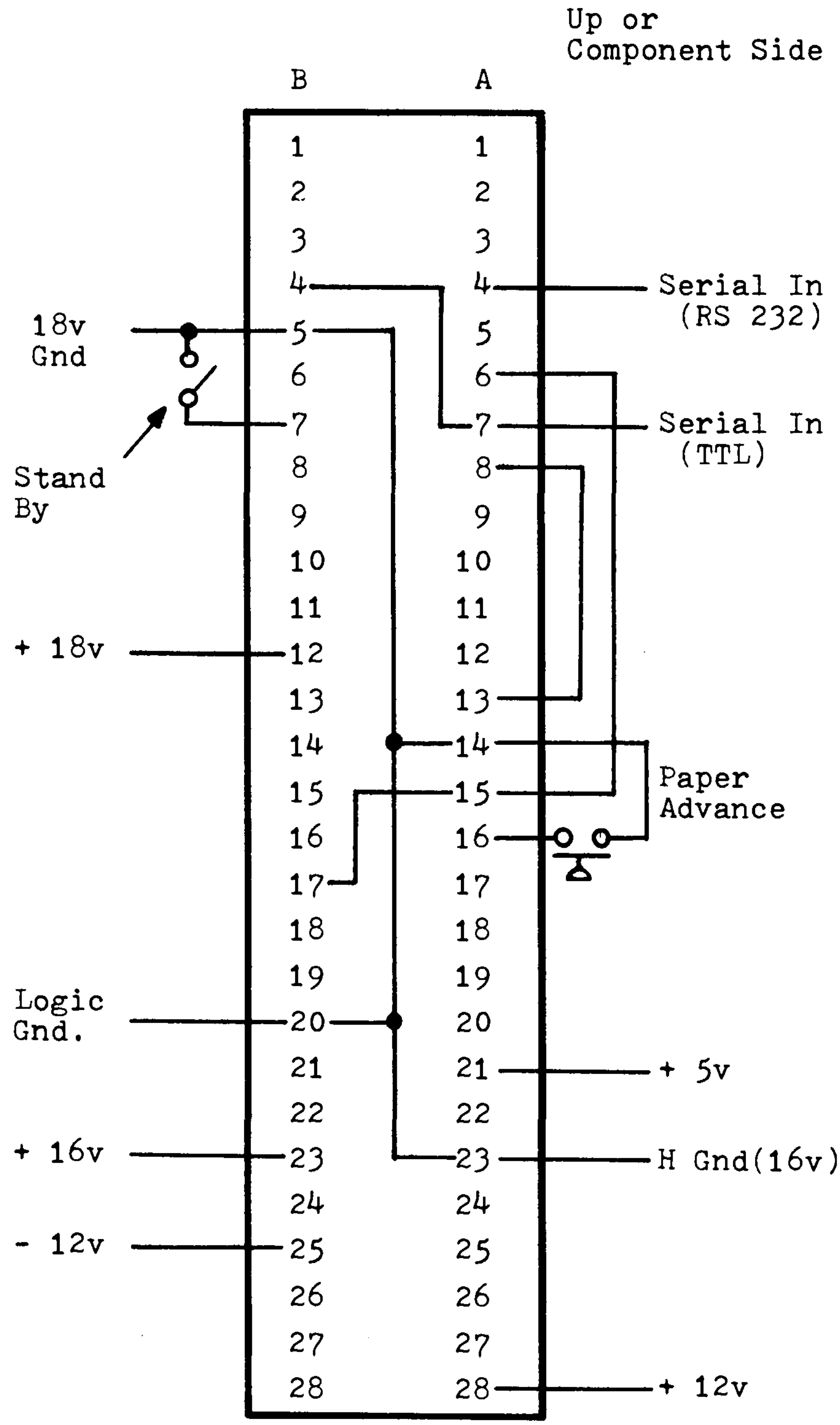


Figure 1

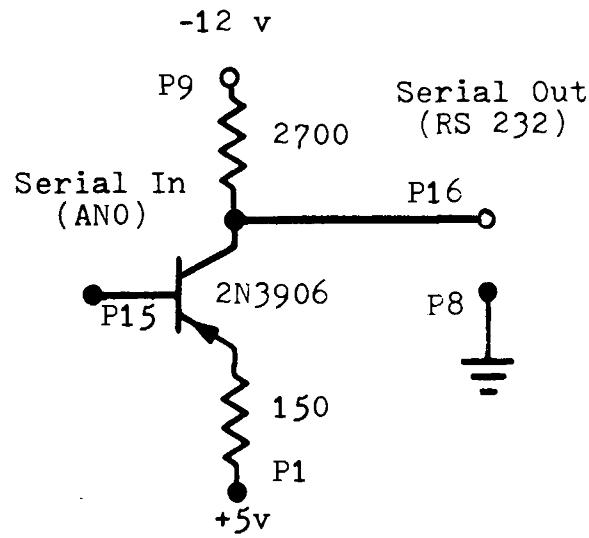
PS-40 Connector Diagram:
Input and Power Connections

```
*380LLL

0380-   A9 89       LDA   #$89
0382-   85 36       STA   $36
0384-   A9 03       LDA   #$03
0386-   85 37       STA   $37
0388-   60          RTS
0389-   84 35       STY   $35
038B-   48          PHA
038C-   20 A5 03    JSR   $03A5
038F-   68          PLA
0390-   C9 8D       CMP   #$8D
0392-   D0 0C       BNE   $03A0
0394-   A9 8A       LDA   #$8A
0396-   20 A5 03    JSR   $03A5
0399-   A9 58       LDA   #$58
039B-   20 A8 FC    JSR   $FCA8
039E-   A9 8D       LDA   #$8D
03A0-   A4 35       LDY   $35
03A2-   4C F0 FD    JMP   $FDF0
03A5-   A0 0B       LDY   #$0B
03A7-   18          CLC
03A8-   48          PHA
03A9-   B0 05       BCS   $03B0
03AB-   AD 58 C0    LDA   $C058
03AE-   90 03       BCC   $03B3
03B0-   AD 59 C0    LDA   $C059
03B3-   A9 D3       LDA   #$D3
03B5-   48          PHA
03B6-   A9 20       LDA   #$20
03B8-   4A          LSR
03B9-   90 FD       BCC   $03B8
03BB-   68          PLA
03BC-   E9 01       SBC   #$01
03BE-   D0 F5       BNE   $03B5
03C0-   68          PLA
03C1-   6A          ROR
03C2-   88          DEY
03C3-   D0 E3       BNE   $03A8
03C5-   60          RTS
03C6-   00          BRK
03C7-   00          BRK

*
```

Apple II AN0 output routine in machine language to provide serial data output at 110 and 300 baud. Change location $3B4 to $4D for 300 baud.



Resistors are in Ohms, 1/2 Watt, 5%
P No's refer to game connector pins -
P9 and P16 are used a tie points.

Figure 2

Single Transistor Adapter Circuit
and Interface

```
*380.3C7

0380- A9 89 85 36 A9 03 85 37
0388- 60 84 35 48 20 A5 03 68
0390- C9 8D D0 0C A9 8A 20 A5
0398- 03 A9 58 20 A8 FC A9 8D
03A0- A4 35 4C F0 FD A0 0B 18
03A8- 48 B0 05 AD 58 C0 90 03
03B0- AD 59 C0 A9 D3 48 A9 20
03B8- 4A 90 FD 68 E9 01 D0 F5
03C0- 68 6A 88 D0 E3 60 00 00
*
```

[Note: This listing and dump were made on the Telpar printer.]

Figure 3

Machine Language Print Routine
and HEX Dump

MICRO INFORMATION

MICRO is published bi-monthly, six issues per year. Single copy price is $1.50. Subscriptions are available at the following rates:

Surface Mail: All Countries $6.00

Air Mail:

Central America	$12.00
Europe & South America	$14.00
Other Countries	$16.00

All subscriptions start with the NEXT issue after receipt of your order.

Back issues are available, while they last, at $1.50 per copy (plus $1.25 for air mail postage overseas or $.50 for surface postage overseas).

Authors will be paid for articles which are printed in MICRO. The payment will be based on the quality of the article, the amount of extra preparation needed to redraw diagrams, etc., as well as type of material. In general, only original material will be considered for publication. There are openings for individuals to contribute on a regular basis: a "KIM Klinic", "Apple Doctor", special interests such as ham radio applications, industrial applications, and so forth.

Advertising in MICRO reaches a lot of people who are seriously interested in the 6502. Advertising deadline for any issue is the 15th of the month prior to publication, e.g. March 15 for the Apr-May issue.

TYPESETTING ON A 6502 SYSTEM

Robert M. Tripp
8 Fourth Lane
S Chelmsford, MA 01824

As Editor/Publisher of MICRO, I was bothered by the need to have typesetting done by an outside company for several reasons. First, of course, was the cost. A typeset page can cost from $12 to $30.00. Second, it takes time have a page set, anywhere from one to five days. Third, once you have the typeset material and are ready to paste up the final copy, it is very difficult to make any changes or corrections. It occurred to me that I should be able to do a reasonable job of typesetting with my existing equipment - a KIM-1 and a Diablo Hytype II based terminal. The results of my efforts are described in this article, and, this entire issue of MICRO has been produced with the equipment and program described.

Actually, "typesetting" is a misnomer for what is being done here. "type" is not being "set". Justification would probably be a better term, but still would not completely cover the features currently implemented. For lack of a term, I named this routine "JUSTIFY".

Features of Justify

JUSTIFY has four modes. The most useful is Full Justification in which a line is set justified at both the left and right margins. The lines you are reading now are an example of a Full Justification. In this mode the width of the column is specified as a parameter to the JUSTIFY routine which then pads the text as necessary to make the text exactly meet the right margin.

The second mode is No Justification. There are a number of instances in which you do not want the material to justified: the last line of a paragraph, source listings, object listings, any type of tables, and so forth. The following listing makes the point quite graphically:

```
0120 A6 DB     JSTIFY LDXZ  CMND
0122 B5 00            LDAZX $00
0124 CA               DEX
```

which, if set with Full Justification would come out as

```
0120  A6  DB       JSTIFY LDXZ   CMND
0122  B5  00                LDAZX  $00
0 1 2 4   C A                      D E X
```

Obviously not what was intended.

The third mode is Center. Title blocks of articles, headers for sections, and so forth need to be centered. The Center mode calculates where to start the text so that it will be properly centered, including spliting a character space in half to get perfect centering.

A
AA
AAA

The last mode currently implemented is actually not a form of justification, but is useful. It is an enhancement in which characters may be printed **slightly** bolder than the surrounding text to make them stand out. This mode is independent of the three justification modes and can be combined with any of them.

Although the JUSTIFY routine was made for typesetting MICRO, we have found it has many other uses. Since the editing portion of the program permits you to make corrections before printing, we can type "perfect" letters.

Justification Algorithm

The justification algorithm, or rules, used is based on certain characteristics of the Diablo printer. This printer "thinks small" - it divides the line into units which are 1/120th of an inch. Each printed character is normally 10 units wide, including the space around the character, giving 12 characters per inch. In TEXT mode, there is no way to space the characters other than next to each other as in regular typing, or separated by a full space. If this was the only method of positioning characters, then the justification would consist of expanding the spaces in a line to pick up the extra units to justify a line. This is the method required for a teletype printer. It looks like this:

This is teletype mode justification.

Note that the spaces between words has been doubled in the first three positions. This is not too bad, and as long as there are not too many spaces to distribute, can be acceptable. Given the Diablo's capability of padding with as little as a space of 1/120 of an inch, much better justification be achieved. If there are only a few units to be distributed over the line, then each normal space may be stretched just a little. For example, in a line which is only one character short of full, there are only ten units of space remaining to be distributed, since each character is 10 units wide. If the line contained five normal spaces, then each space would be stretched by two units, an almost imperceptible amount.

Full justification with an extra unit.
Full justification with no extra units.

As the number of units to be distributed increases, there comes a point at which the spaces become noticeably wide. The way this can be solved on the Diablo is to distribute spaces among the characters as well as the spaces. The calculation is done as:

1. Count number of extra units.
2. If there are more units than characters and spaces, then add one or more units to each character and space.
3. If there are fewer units than characters and spaces, then test just the spaces. If there are more units than spaces, then add one or more to each space.
4. When there are finally fewer units than spaces, distribute the remaining units over the first spaces in the line.

Each character has one unit added.
Characters have not had a unit added.

Close inspection will reveal that the first line above has the individual characters spaced slightly wider than the second line. This algorithm will handle most normal lines, but if a line has too many units to fill, it will look strange.

T h i s   i s   a   v e r y   l o o s e   l i n e .

The JUSTIFY Function

JUSTIFY is written in the form of a HELP Function. HELP is a sort of high level language I have developed and is the basis of the Editor, Mailing List, and Information Retrieval packages sold by The COMPUTERIST, as well as a large number of utilities we use internally for such operations as printing labels for cassette tapes, creating copies of program tapes, and so forth. Each of the Functions is, essentially, a subroutine which is called and passed a set of parameters. If the arguments required are placed in the proper locations - 00D9, DA, and DB - and if the instruction at location 01AB is changed from JMP NXTSTP to RTS, then JUSTIFY may be called as a simple subroutine.

Operation of Justify

JSTIFY uses the pointer in CMND+03 to pick up the full address of the buffer which contains the material to be justified, and stores it in BUFFER and BUFFER+01.

CLEAR puts zero in each of the seven counters, NULLS to TEMP, and then puts a zero at the first location past the end of the buffer as defined by the start of the BUFFER and the length as defined by the parameter CMND+01. This zero guarantees a null for the end of buffer test later on.

MORE starts at the end of the buffer to pick up and test each character in order to get a count of the number of nulls, spaces, and other characters. It also tests for a Control N (0E). A Control N is used to signal that No Justification is required on the current line and control branches to NEXT.

JUSTIFY FUNCTION - 16 JAN 1978

```
                JUSTIF ORG   $0120

                NULLS  *     $00CC
                SPACES *     $00CD
                CHARS  *     $00CE
                COFSET *     $00CF
                SOFSET *     $00D0
                EXCESS *     $00D1
                TEMP   *     $00D2
                POINT  *     $00D3
                BUFFER *     $00D4
                MODE   *     $00D6
                CMND   *     $00D8
                OUTCH  *     $1EA0
                NXTSTP *     $0304

0120 A6 DB      JSTIFY LDXZ  CMND   +03
0122 B5 00             LDAZX $00
0124 85 D4             STAZ  BUFFER
0126 B5 01             LDAZX $01
0128 85 D5             STAZ  BUFFER +01

012A A2 07             LDXIM $07
012C A9 00             LDAIM $00
012E 95 CC      CLEAR  STAZX NULLS
0130 CA                DEX
0131 10 FB             BPL   CLEAR
0133 A4 D9             LDYZ  CMND   +01
0135 91 D4             STAIY BUFFER
0137 88                DEY

0138 B1 D4      MORE   LDAIY BUFFER GET CHARACTER TO COUNT
013A C9 0E             CMPIM $0E
013C F0 59             BEQ   NEXT   NO JUSTIFICATION
013E C9 20             CMPIM $20    TEST SPACE CHARACTER OR LESS
0140 F0 1E             BEQ   SCOUNT EQUAL SPACE
0142 10 1E             BPL   CCOUNT EQUAL CHARACTER
0144 E6 CC             INCZ  NULLS  EQUAL NULL
0146 88         AGAIN  DEY          DECREMENT STRING COUNTER
0147 10 EF             BPL   MORE

0149 C8         TEST   INY
014A B1 D4             LDAIY BUFFER
014C C9 0B             CMPIM $0B
014E F0 16             BEQ   CENTER
0150 C6 CE             DECZ  CHARS
0152 A6 CC             LDXZ  NULLS  TEST ANY NULLS
0154 F0 41             BEQ   NEXT   NO NULLS
0156 A5 DA             LDAZ  CMND   +02
0158 CA         MULT   DEX          CALCULATE UNITS TO EXPAND
0159 F0 22             BEQ   DIVIDE GO TO DIVIDE
015B 18                CLC
015C 65 DA             ADCZ  CMND   +02
015E D0 F8             BNE   MULT   MULT LOOP UNTIL DONE
```

```
0160 E6 CD     SCOUNT INCZ  SPACES
0162 E6 CE     CCOUNT INCZ  CHARS  BUMP SPACES AND CHAR COUNTERS
0164 D0 E0            BNE   AGAIN

0166 E6 D3     CENTER INCZ  POINT
0168 46 CC            LSRZ  NULLS
016A 90 06            BCC   SHIFT
016C A5 DA            LDAZ  CMND   +02
016E 4A               LSRA
016F 20 BE 01         JSR   OFFSET

0172 A9 20     SHIFT  LDAIM $20
0174 20 A0 1E         JSR   OUTCH
0177 C6 CC            DECZ  NULLS
0179 D0 F7            BNE   SHIFT
017B F0 1A            BEQ   NEXT

017D C5 CE     DIVIDE CMPZ  CHARS  TEST CHAR SPACING
017F 30 09            BMI   DIVDON UNITS < CHARS
0181 38               SEC          UNITS >= CHARS
0182 E5 CE            SBCZ  CHARS  HOW MANY UNITS PER CHAR
0184 E6 CF            INCZ  COFSET BUMP COUNTERS
0186 E6 D0            INCZ  SOFSET
0188 D0 F3            BNE   DIVIDE UNCOND. BRANCH

018A C5 CD     DIVDON CMPZ  SPACES REMAINDER TO SPACES
018C 30 07            BMI   SDONE
018E 38               SEC
018F E5 CD            SBCZ  SPACES
0191 E6 D0            INCZ  SOFSET
0193 D0 F5            BNE   DIVDON

0195 85 D1     SDONE  STAZ  EXCESS REMAINDER TO EXCESS

0197 A4 D3     NEXT   LDYZ  POINT  GET STRING POINTER
0199 E6 D3            INCZ  POINT  BUMP FOR NEXT TIME
019B B1 D4            LDAIY BUFFER FETCH CHARACTER
019D C9 18            CMPIM $18    BOLD?
019F F0 43            BEQ   BOLD
01A1 C9 19            CMPIM $19    NORMAL?
01A3 F0 3F            BEQ   BOLD
01A5 C9 20            CMPIM $20    TEST SPACE
01A7 F0 29            BEQ   SPACE
01A9 10 03            BPL   CHAR
01AB 4C 04 03         JMP   NXTSTP
01AE 20 A0 1E  CHAR   JSR   OUTCH
01B1 C6 CE            DECZ  CHARS  CORRECTION FOR LAST CHAR
01B3 30 E2            BMI   NEXT   LAST CHAR
01B5 A5 CF            LDAZ  COFSET FETCH OFFSET
01B7 F0 DE     NTEST  BEQ   NEXT
01B9 20 BE 01         JSR   OFFSET
01BC F0 D9            BEQ   NEXT

01BE AA        OFFSET TAX
01BF A9 10            LDAIM $10
```

```
01C1 20 A0 1E         JSR    OUTCH
01C4 A9 48      BUMP  LDAIM 'H
01C6 20 A0 1E         JSR    OUTCH
01C9 CA               DEX
01CA D0 F8            BNE    BUMP
01CC A9 1C            LDAIM $1C
01CE 20 A0 1E         JSR    OUTCH
01D1 60               RTS

01D2 20 A0 1E   SPACE JSR    OUTCH
01D5 A5 D0            LDAZ   SOFSET FETCH SPACE OFFSET
01D7 A6 D1            LDXZ   EXCESS TEST EXTRA OUTPUT
01D9 F0 05            BEQ    NOXCES
01DB C6 D1            DECZ   EXCESS DECREMENT EXCESS
01DD 18               CLC
01DE 69 01            ADCIM $01     INCREMENT OFFSET
01E0 C9 00      NOXCES CMPIM $00
01E2 10 D3            BPL    NTEST

01E4 18         BOLD  CLC
01E5 69 1E            ADCIM $1E
01E7 AA               TAX
01E8 A9 1B            LDAIM $1B
01EA 20 A0 1E         JSR    OUTCH
01ED 8A               TXA
01EE 20 A0 1E         JSR    OUTCH
01F1 D0 A4            BNE    NEXT
```

TEST first checks to see if the Center Mode has been specified by the Control K (0B) character. It then checks to determine if there are any nulls at the end of the line. If there are no nulls then the line can be printed with no further justification required. It is already justified.

MULT multiplies the number of nulls by the character width provided by parameter CMND+02. This gives the number of units that must be distributed throughout the line to provide left and right justification.

CENTER handles the Center Mode of justification. It bumps over the Control K character and divides the nulls by two so that the nulls will be evenly divided. It tests for an odd or even number of nulls using a BCC after the LSRZ which does the divide. If there are an even number of nulls, then it branches to SHIFT. IF there are an odd number of nulls, it picks up the character width from CMND+2, divides this two to get a one-half character offset to provide more accurate centering. This is output via the OFFSET routine.

SHIFT moves the printer to the start of the centered line by outputting spaces equal to one-half the original number nulls. When finished it branches to NEXT which takes care of printing the text.

DIVIDE allocates the excess units along the line of text to produce the Full Justification. It first tests to see if it can allocate an additional unit to each individual character and space. If so, it increments both the character offset counter (COFSET) and the space offset counter (SOFSET). It then tests whether another unit can be allocated, until it finds that there are fewer units to be allocated than characters and spaces.

DIVDON takes care of any units remaining after the DIVIDE allocation. These are divided among the spaces, incrementing SOFSET until there are fewer units than spaces. The remainder, if any, is stored in EXCESS where it will be used on spaces starting at the beginning of the line.

NEXT handles the printing. It picks up and examines the next character. It branches to BOLD, SPACE, CHAR, or returns to the calling program if a null is encountered.

CHAR outputs the character using the system subroutine, in this case the KIM OUTCH subroutine. It tests for last character and puts out the character offset (COFSET) if non-zero.

OFFSET saves the offset in X, then puts the Diablo printer into PLOT mode by outputting a 10 hex. It then puts out one 'H' for each unit of offset, and finally returns the printer to TEXT mode by printing a 1C hex.

SPACE outputs a space, then combines a unit of EXCESS with the space offset and goes to NTEST to output the offset if not zero.

BOLD converts a Control X to '6' or a Control Y to '7', and then outputs the character after issuing an escape 1B hex. This sets or clears the print enhancement mode.

The DIRECT TYPESETTER

One use of JUSTIFY has been in a HELP program for direct typesetting. In this program a sheet of paper is inserted sideways in the terminal. Material is entered and edited on the left side of the page and typeset on the right side.

The CPRINT Function outputs a Control Comma (CTLCMA) 1C hex which sets the printer in TEXT mode, and then issues a Carriage Return (CR) 0D hex.

The INPUT Function accepts data from the terminal, places it in the buffer defined by FILE (starts at 1780 and is 39 decimal characters long), and supports some editing features.

The next CPRINT causes the printer to TAB to the right side of the page, to the left margin of the typesetting area.

JUSTFY does the actual justification and printing. Its parameters specify that the set line has a maximum width of 39 decimal characters; that the width of each character is 10 units; and the 1E is a pointer to the start of the buffer - FFILE.

The last CPRINT sets the printer back one horizontal unit to provide a closer line spacing.

The BRANCH simply returns control to NEXT and the system is ready for the next line to be input.

DIRECT TYPESETTER - 16 Jan 1978

```
0004  0B1C010D   1  NEXT  CPRINT  CTLCMA  1    CR   TEXT MODE, CARRIAGE RETURN
0008  081C0080   2        INPUT   FILE    0    80   CLEAR AND INPUT TEXT
000C  0B090100   3        CPRINT  TAB     1    0    TAB TO TYPESET AREA
0010  01270A1E   4        JUSTFY  39.     10.  1E   39 CHAR WIDTH, 10 UNITS PER CHAR
0014  0B10014E   5        CPRINT  CTLP    1    "N   PLOT MODE, UP ONE UNIT
0018  03010000   6        BRANCH  NEXT              READY FOR NEXT LINE
001C  20008017   7  FILE  FMAP            FFILE     BUFFER AT 1780
0020  00270000   8  FMAP  00.     39.               FIELD STARTS OFFSET 0, 39. CHAR.
```

[Editor's Note: Please do not judge the quality of the Diablo Printer by this particular article. This was the last one done, and the printer is sorely in need of adjustment. Look at some of the other articles in this issue to see the quality when the printer is in proper working order.]

TIM MEETS THE S100 BUS

Gary L. Tater
7925 Nottingham Way
Ellicott City, MD 21043

Hardly a computer meeting goes by without a discussion of which bus structure is best. While the S100 bus may not be optimum for the 6502 microprocessor, its use does make purchasing RAM and ROM boards easy.

With this in mind, I purchased a 6502 CPU board for the S100 bus from CGRS Microtech. This CPU board is almost a complete system with its onboard 2K RAM and 4K ROM. But in order to use my CT-64 Southwest Technical Products video terminal with this CPU, I needed an S100 terminal interface monitor (TIM) board. While CGRS markets a very nice TIM board, I elected to build a bare bones S100 TIM board which is described in this article.

In addition to serving as a serial I/O port for a terminal, TIM contains an operating system for 6500 microcomputers. The OCT-NOV issue of MICRO (page 5) contains an article on the operation of the TIM program. In summary, TIM is a read-only memory and I/O device that is self adapting to terminal speeds between 10 - 30 cps. With TIM you can display and alter CPU and memory location using a keyboard and video display; you can read and write hex formatted data from a paper tape or a cassette interface such as the Southwest Technical Products AC-30; and you have an eight bit parallel I/O port where each bit of the eight can be programmed as either input or output.

As you can see from the schematic diagram (Figure 2), only the TIM chip (6530-004) and four integrated circuits are needed, excluding voltage regulators. For the perfectionist, buffering could be added to the address lines, data lines, and parallel output port, but two CGRS Microtech systems are now successfully using this TIM design. Integrated circuits U2 and U3 are used during resets to reconfigure TIM memory locations as described in the previously referenced TIM article. The MC 1488 and MC 1489 are Motorola devices which convert TTL levels to RS 232 levels and RS 232 levels to TTL respectively.

A memory map of this TIM design is provided in Figure 1. For proper operation of a 6502 microprocessor and this TIM board, you will need both page zero and page one memory. Page one is needed by the 6502 microprocessor for its software stack. Page zero memory is used in the TIM program to store the baud rate of your terminal (locations 00EA and 00EB).

To operate a TIM based system you need only momentarily ground pin 16 of TIM (pin #75 of the S100 bus) using a switch on your front panel. After you send a carriage return to the computer, you should see a TIM message such as:

```
7052 30 2E FF 01 FF
```

This message contains first the program counter (7052), processor status register (30), accumulator (2E), X register (FF), Y register (01), and stack pointer (FF). The actual values will vary from machine to machine.

7000 - 73FF	TIM ROM	
FFC0 - FFFF	TIM RAM	
6E00 - 6E0F	TIM I/O	
6E02		Serial Port

Figure 1

TIM Board Memory Map

If you have a problem, first check all of your wiring and the +5, +12, and -12 voltages. Then insure that your reset switch is controlling pin 16 of TIM. Next, using an oscilloscope, check for a carriage return character at pin 25 of TIM and pin 24 for the TIM message. With a good signal at pin 25 but no answer at pin 24, the last two things to check are the address lines including pin 21, PB4, and finally, check your TIM chip in a working system. The two systems built using this design on prototype boards came up immediately. Hopefully, you will have the same good fortune.

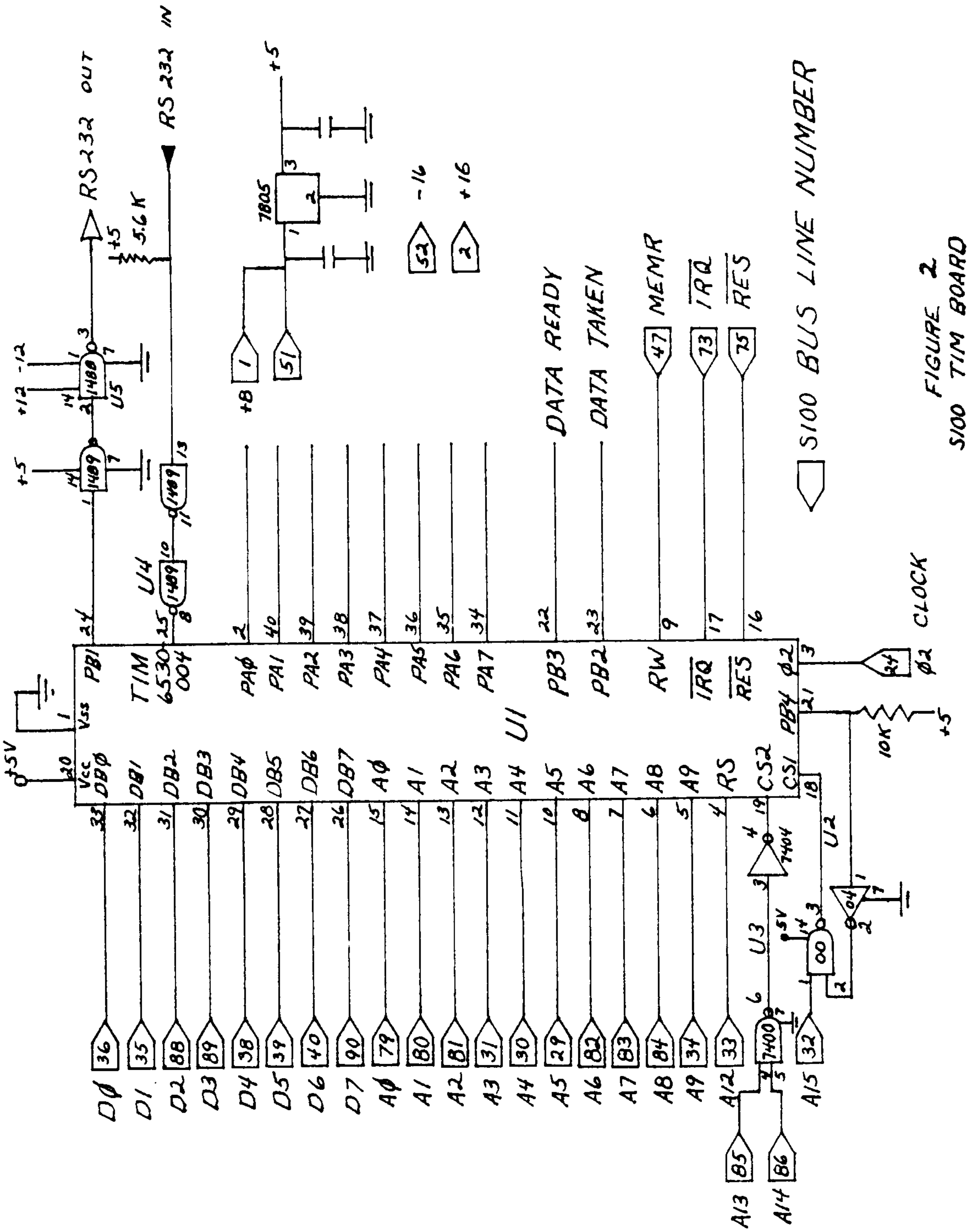


FIGURE 2
S100 TIM BOARD

THE APPLE II POWER SUPPLY REVISITED

Rod Holt
Chief Engineer
Apple Computer Inc.
20863 Stevens Creek Blvd., B3-C
Cupertino, CA 95014

Your review of the Apple II ("Inside the Apple II" by Arthur Ferruzzi, MICRO #1, Page 9) was most gratifying. However, your comment about the "small" power supply invites a reply.

The power supply has no function other than running the Apple II and its peripherals, and as it does this very well, then what's "small"? Apple Computer is far enough along in peripheral card development to state categorically that with an EPROM card, a ROM card, a parallel printer card, a floppy disk card, and several more all plugged in, the power supply isn't even breathing hard.

We do recommend that users keep their designs to a reasonable minimum power. But the reason for this is the same as one of the reasons Apple designed a switching regulator in the first place: to keep temperature rises to a minimum. The general rule of thumb is that a 25 degree C increase in ambient will drop the mean time between failures by a factor of 10. For the user, the watts saved mean literally thousands of hours more of trouble free system operation. The switcher design cuts the input power nearly in half over conventional regulators and the overall temperature rise is reduced by approximately 25 C. And, of course, the use of low-power schottky and a tight and economic hardware design is key as well.

A second point needs to be made. It's quite common to have well over a thousand dollars in semiconducters in an Apple II system. The Apple switcher is designed to protect those semiconducters under all fault conditions (including possible failure modes internal to the power supply itself). Never has an Apple II been damaged by its own power supply. In contrast, Apple can document many cases of blown RAM and other IC's where customers have used homemade or "off the shelf" power supplies. See the sad story in EDN, November 20, 1977 page 232. There are many more such sad stories. The power supply manufacturers of the world are just beginning to see that a supply failure means much more than just an equipment shut-down nuisance. Thus it's important to know what happens when, for example, the +12 volt supply is shorted to the -5 volt supply. What happens to the +5 volts? With the Apple switcher, all supplies neatly go to zero, and they all recover smoothly when the short is removed.

I close by murmuring -

"Small is Beautiful".

MICRO

MICROBES - Tiny Bugs in Previous MICROs

1:13 HYPERTAPE and ULTRATAPE

It has been noted that during loading with ULTRATAPE, the KIM Monitor will go back to a normal tape load if the requested Program ID is not located. The normal use of ULTRATAPE is with a group of files with the same ID, so that this does not normally occur. But, it could happen and would really foul things up.

2:7 Making Music with the KIM-1

First two lines of "Score" should be:

```
0200   60 4A 44 32 24
0205   20 46 40 32 22
```

2:30 Important Addresses of KIM-1 ...

1EA0 OUTCH Reference was omitted.

A SIMPLE FREQUENCY COUNTER USING THE KIM-1

Charles R. Husbands
24 Blackhorse Drive
Acton, MA 01720

A piece of test equipment that is occassionally very useful in the computer laboratory is a frequency counter. This article explains how to use the capabilities of the KIM-1, with a minimum of additional hardware, to provide the functions of such an instrument. The frequency counter described operates over the audio range from 500 Hz to above 15 KHz. To reduce the amount of external hardware needed, the design assumes TTL level input signals. However, the addition of a small amount of analog hardware to the design presented would allow the counter to be used with analog signal sources.

Basic Counter Mechanization

In order to develop a frequency counter from the KIM-1 microcomputer it is necessary to count and display the number of input pulses detected over a specific time interval. The basic time interval chosen was 100 milliseconds. This time interval is established by using one of the two interval timers available on the KIM-1. Transitions in the applied waveform are sensed by the external logic and force non-maskable interrupts to the KIM. As each interrupt is detected a memory location is incremented. Because of the availability of the decimal mode in the 6502 instruction set, the count can be maintained in decimal rather than binary or hexadecimal form. At the conclusion of the 100 millisecond interval the accumulated count is loaded into the display registers and the process is repeated. Figure 1 is a flow chart of the frequency counter program.

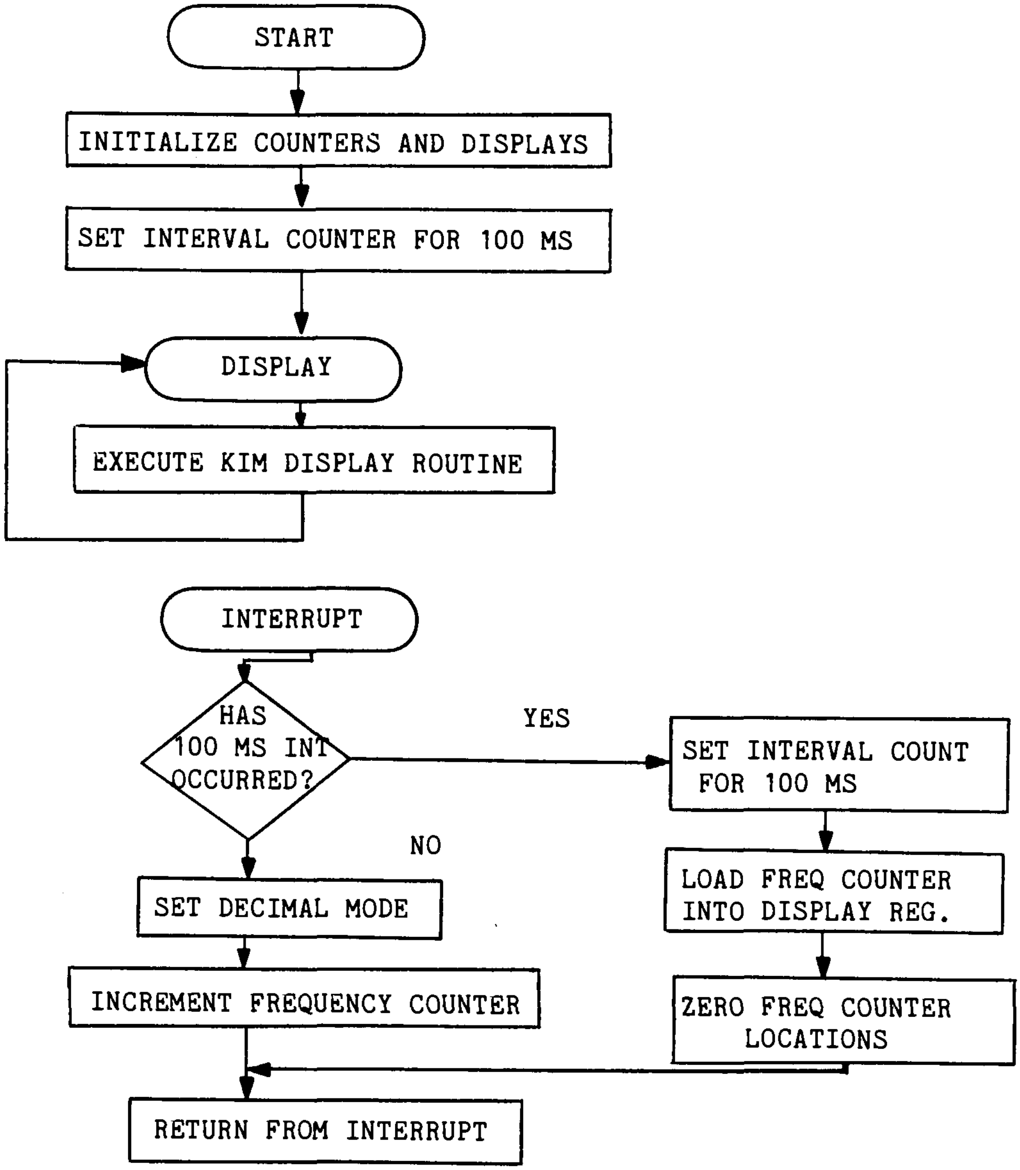
Detailed Software Description

As shown in the flow chart (Figure 1) and in the program listing (Figure 2) the program is started at location 0005 and the frequency counter memory location and display locations are initialized to zero. A Value of 99. is loaded into the interval counter at location 1747. A value stored at this location is decremented every 1024 microseconds. Under these conditions a zero register value will then be realized 101.376 milliseconds after the register is loaded.

After the initialization process the program goes into an idle loop called DISPLAY and waits for an interrupt to occur. The DISPLAY program consists of repeated calls to the KIM display routine which presents the contents of the display registers 00FA and 00F9 on the seven segment display LEDs.

When an IRQ interrupt is sensed, the KIM logic forces program control to the address stored in memory locations 17FE and 17FF. In this mechanization, the value stored in these locations will direct program control to be transferred to the start of the interrupt routine (location 0021). The interrupt program first stores away the values of A and X from the interrupted program. The contents of the interval timer register, location 1746, is then read to establish if the 100 millisecond interval has been completed. A non zero number indicates that the counter is still counting and an input pulse transition has been detected. The logic sets the processor into the decimal mode and adds one to the contents of the frequency counter location. As we wish to detect values above 1 KHz, a second frequency counter register must be employed to count the overflow from the least significant two decimal digits. Having completed the incrementation process, the program restores the the values of A and X and returns to the interrupted program by executing the RTI instruction.

If a zero value is observed when the interval timer register is read, then the 100 millisecond timing interval has been completed. The program reloads the 100 millisecond value into the interval counter, loads the accummulated count in the frequency counter memory locations into the appropriate display



FLOW DIAGRAM FOR FREQUENCY COUNTER PROGRAM

Figure 1

Figure 2

```
                        ORG    $0005

                 INTGER *      $00FA
                 FRACT  *      $00F9
                 PBDD   *      $1703
                 CLOCKX *      $1746
                 CLOCK  *      $1747
                 SCANDS *      $1F1F

0005 A9 00       START  LDAIM $00    INIT COUNTERS AND DISPLAY
0007 85 51              STAZ  CNTONE
0009 85 52              STAZ  CNTTWO
000B 85 FA              STAZ  INTGER
000D 85 F9              STAZ  FRACT
000F 8D 03 17           STA   PBDD
0012 A9 62              LDAIM $62    SET UP 100 MILLISECOND TIMER
0014 8D 47 17           STA   CLOCK

0017 20 1F 1F    DISPLA JSR   SCANDS DISPLAY DATA
001A 4C 17 00           JMP   DISPLA CONTINUOUSLY

0021                    ORG   $0021

0021 48          INTRPT PHA          SAVE A REGISTER
0022 8A                 TXA          SAVE X REGISTER
0023 48                 PHA
0024 AD 46 17           LDA   CLOCKX TEST CLOCK TIMED OUT
0027 30 11              BMI   MILLI  TEST OF 100 MILLISECONDS

0029 F8          COUNT  SED          SET DECIMAL MODE
002A 18                 CLC          CLEAR CARRY BIT
002B A5 51              LDAZ  CNTONE GET FRACTIONAL PART
002D 69 01              ADCIM $01    INCREMENT
002F 85 51              STAZ  CNTONE
0031 A5 52              LDAZ  CNTTWO ADD CARRY BIT IF SET
0033 69 00              ADCIM $00
0035 85 52              STAZ  CNTTWO
0037 4C 4D 00           JMP   EXIT

003A A9 62       MILLI  LDAIM $62    RESET CLOCK
003C 8D 47 17           STA   CLOCK
003F A5 51              LDAZ  CNTONE MOVE DATA TO DISPLAY
0041 85 F9              STAZ  FRACT
0043 A5 52              LDAZ  CNTTWO
0045 85 FA              STAZ  INTGER
0047 A9 00              LDAIM $00    RESET COUNTERS
0049 85 51              STAZ  CNTONE
004B 85 52              STAZ  CNTTWO

004D 68          EXIT   PLA          RESTORE X REGISTER
004E AA                 TAX
004F 68                 PLA          RESTORE A REGISTER
0050 40                 RTI          RETURN FROM INTERRUPT

0051 00          CNTONE =     $00    FRACTIONAL COUNTER
0052 00          CNTTWO =     $00    INTEGER COUNTER
```

registers, and then zeros the contents of the frequency counter locations. The interrupt program is exited by restoring the values of A and X and returning via the RTI instruction.

The Hardware Configuration

Figure 3 illustrates the additional logic required to use the KIM as a frequency counter and shows how that logic is connected to the KIM Expansion connector. The purpose of the 74121 monostable multivibrator is to produce a negative going pulse of short duration onto the IRQ interrupt lines whenever the input to that chip experiences a high-to-low transition. It should be noted that the IRQ is a level rather than an edge sensitive interrupt and that the interrupt line must be held low only long enough to allow the processor to sense the interrupt. Therefore, with the addition of this flip-flop the KIM will experience an IRQ interrupt each time the input source exhibits a high-to-low transition. If a periodic pulse train is being applied to the input, then an IRQ interrupt will be experienced on each cycle.

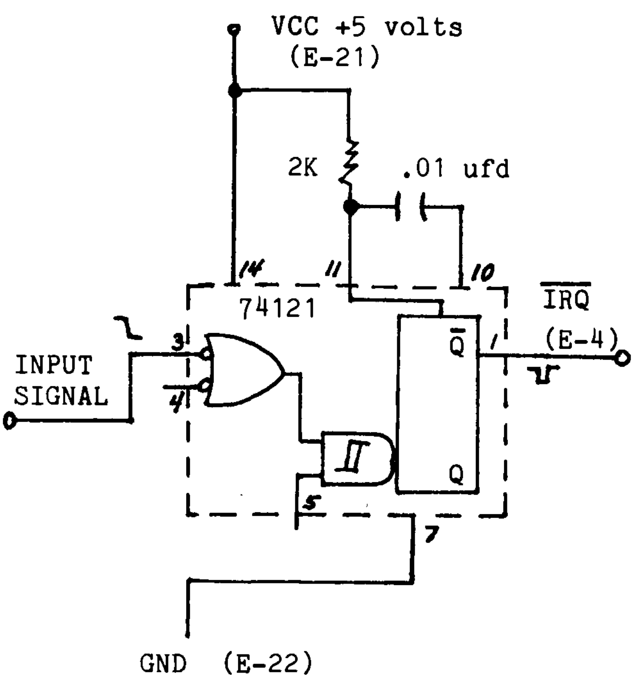


Figure 3

The accuracy of this hardware/software on a KIM-1 for measuring frequencies is shown in the table (Figure 4). A very accurate frequency meter was used to obtain the meter measurements. Since there are probably slight variations in the speed of different KIM-1s, you should calibrate your own unit before using it for any "real" measurements.

Frequency Calibration

Meter	KIM
14.960	15.00
13.961	14.00
12.960	13.00
11.968	12.00
10.966	11.00
9.965	10.00
8.970	9.00
7.977	8.00
6.984	7.00
5.983	6.00
4.985	5.00
3.992	4.00
2.991	3.00
2.003	2.00
1.003	1.00
.902	0.90
.801	0.80
.705	0.70
.608	0.60
.507	0.50

Figure 4

Additional Comments

In addition to entering the values shown in the accompanying listing, the values 0010 should be stored in locations 17FA and 17FB, and 2100 should be stored in locations 17FE and 17FF. The latter value directs program control to the beginning of the interrupt routine when an IRQ is sensed.

The results displayed on the seven segment indicators will be in the form XX.XX KHz. This format was chosen for convenience and the range can be shifted for higher accuracy by software modifications. Additional improvements are left to the reader to create. The author would appreciate being informed of any interesting improvments you come up with.

6502 BIBLIOGRAPHY
PART II

William Dial
438 Roslyn Avenue
Akron, OH 44320

129. Torzewski, Joe, "Apple I Library" On Line 2 No. 12 p. 11 (Sept 14, 1977)
Apple I owners interested in a library for software and hardware should contact Joe Torzewski, 51625 Chestnut Rd., Granger IN 46530.

130. House, Gil, P.O. Box 158, Clarksburg, MO 20734, "6502 Tape Labeler" On Line 2 No. 12 p. 11 (Sept 14, 1977)
Man readable 6502 legible tape labeler for TIM, JOLT, DEMON

131. Cater, J., 11620 Whisper Trail, San Antonio, TX 78230, "Run OSI 6502 8K BASIC on your Tim or Jolt" On Line 2 No. 11, p. 13 (Sept 14, 1977)
Full info and patches to run this super fast BASIC.

132. Staff Article "The PET Computer" Personal Computing 1 No. 5, pp 30-40 (Sept-Oct, 1977)
Interviews with Chuck Peddle of Commodore and with other microcomputer experts.

133. Lancaster, Don, "Hex-to-ASCII Converter for your TVT-6" Popular Electronics 12 No. 4, pp 49-52 (Oct. 1977)
Simple module produces op-code display for entire computer. Describes a board to be connected between the TVT-6 and the KIM-1 microcomputer.

134. Microcomputer Associates Inc., 2368-C Walsh Ave., Santa Clara, CA 95050 Popular Electronics 12 No. 4, p. 100 (Oct, 1977)
New Product Announcement: A 6502 RAP, resident assembler program and TINY BASIC of ROM. Cost $200.

135. CGRS Microtech, P.O. Box 368, Southampton PA 18966, On Line 2 No. 13, p 2 (Oct 5, 1977)
New Product Announcement: EXOS and DATE are two new 6502 software packages. EXOS is an Extended Operating System featuring a number of useful commands and DATE is a disassembler, assembler, trace and debug editor. Available on four programmed 2708 EPROMS or on TIM format paper tape. Programs are each $150 or $295 for both, on EPROMS.

136. Pyramid Data Systems, 6 Terrace Ave., New Egypt, NJ 08533, On Line 2 No 13, p 6 (Oct 5, 1977)
New Product Announcement: XIM is a 1K software package for KIM that adds 17 commands to the KIM Monitor, including a Breakpoint routine. Cassette and 45 page manual is $12 ppd., paper tape is $10.

137. K L Power Supplies, P.O. Box 86, Montgomeryville, PA 18936, On Line 2, No. 13, p. 11 (Oct 5, 1977)
New Product Announcement: Model 512 Power Supply is for the KIM with enough capacity for an extra 8K and other accessories.

138. Matthews, K., "6502 Forum" Kilobaud No. 10, p 11, (Oct. 1977)
Mentions E.C.D. Micromind II based on the 6512 A (related to 6502).

139. Rugg, Tom and Feldman, Phil, "BASIC Timing Comparisons" Kilobaud No. 10 p. 20 (Oct, 1977)
Compares over 30 different hobby computer systems on seven different Benchmark programs in BASIC. Fastest was OSI 8K BASIC using 6502 in a Challenger running at 2 MHz. Actually a late entry which was still a little faster was the HeathKit H-11 with a special Extended Instruction Set and a Floating Instruction Set which are to be offered as accessories for the H-11.

140. Overstreet, Jim, "Try Your KIM-1 on RTTY" 73 Magazine No. 205 pp 88-91 (Oct, 1977)

Has a Baudot Receive Program that takes the output from an FSK converter and runs a video terminal with the KIM board. A CW transmit program is also given in the article.

141. Schawlow, Arthur L., "Search Subroutine for the 6502 Disassembler", Interface Age 2 No. 1, p 146 (Oct, 1977)

A description, listing and sample run of an object code search subroutine for use with the 6502 Disassembler published in the September 1976 issue of Interface Age.

142. Simonton, John S., Jr., "What the Computer does ... an Introduction.", Polyphony 3, No. 1, pp 5-7, 28 (July, 1977)

PAIA Electronics will shortly have a complete KIM-1 package showing how to interface with their 8700 Computer Controller based on a 6503 processor. A large selection of programs for KIM is promised.

143. Simpson, Rick, "KIM Forum", Kilobaud No. 11, pp 16-17, 48 (Nov, 1977)

Caxton Foster of the Computer Sciences Dept. of the University of Massachusetts is the author of a college text on microprocessors and all programming examples use KIM-1. Also R.W. Burhans, E.E. Dept. of Ohio University has some informative comments on the adjustment of the PLL pot VR-1.

144. Butterfield, Jim, "Hyper about Slow Load Times", Kilobaud No 11, pp 66-69 (Nov, 1977)

Butterfield explains the development of his HYPERTAPE (Supertape) program for loading or dumping to a KIM audio cassette at 50 bytes per second, six times the normal KIM-1 rate.

145. Blankenship, John, "Expand Your KIM", Kilobaud No 11, pp 84-87 (Nov 1977)

The first of several articles on expanding KIM to use the S-100 bus to give 13K memory, Cromenco Dazzler, a printer and keyboard, joysticks,etc.

146. Johnson Computer, P.O. Box 523, Medina, OH 44256, On_Line 2, No 14, p 7 (October 26, 1977)

New Product Announcement: KIM-1 8K Basic by Microsoft is available in either a 6-digit or 9-digit precision version which includes full printout of error messages. Prices are $97.50 and $129.00.

147. Rockwell International, P.O. Box 3669, Anaheim, CA 92803, Product Bulletin

Rockwell now has available a number of 6500 family microprocessor chips including r6502, r6505 and others. They also are promoting SYSTEM 65, a floppy disc based powerful development system.

148. Sneed, James R., "Adding an Interrupt Driven Real Time Clock", Byte 2 No. 11, pp 72-74 (Nov, 1977)

An external board drives interrupts at 15 Hz which is used to calculate time for use by the computer.

149. Brader, David, "A 6502 Personal System Design: KOMPUUTAR", Byte 2 No. 11 pp 94-137 (Nov, 1977)

A very detailed constructional article.

150. NCE/CompuMart, 1250 N. Main St., Ann Arbor, MI 48104, Byte 2 No. 11, p 140, (Nov, 1977)

New Product Announcement: A number of Accessories for the KIM-1 including backplane/S-100 adapter, 8K Seals memory, Poly Video terminal interface, Itty Bitty Tiny BASIC, Matrox Video RAMS, Graphics and Alpha-Numerics Boards.

151. The Enclosures Group, 55 Stevenson St., San Francisco, CA 94105, Byte 2 No. 11, p 234 (Nov, 1977)

New Product Announcement: Offers an enclosure to dress up the KIM-1.

152. Apple Computer Inc., 20863 Stevens Creek Blvd., Cupertino, CA 95014, Byte 2, No. 11, p 252 (Nov, 1977)
Apple II is a new entry in the home computer market. At $1298 it offers 6K Basic in ROM video graphics in 15 colors, 4K of programmable memory in RAM, a 2K monitor, cassette interface, floating point package, etc.

153 Anon., "Get the most out of Basic", OSI Small Systems Journal 1, No. 2, pp 4-7 (Sept, 1977)
Note on Basic in general and the OS-65D System.

154. Smith, Gary A., "Contributed Program", OSI Small Systems Journal 1, No. 2, p. 12 (Sept, 1977)
Program displays the memory address and the data contained in HEX.

155. Anon., "OSI 6502 Cycle Time Test", OSI Small Systems Journal 1, No. 2, pp 12-13 (Sept., 1977)
Measures the cycle time using a stop watch and program to record the number of whole cycles.

156. Anon., "Memory Test", OSI Small Systems Journal 1, No. 2,pp15-17(Sept 1977)
A memory test for video and serial-based computers using the 6502.

157. Anon., "1K Corner: Close the Window", OSI Small Systems Journal 1, No. 2, p 18 (Sept., 1977)
Close the Window is a dice game designed to be played on the OSI 65V Computers.

158. The COMPUTERIST, P.O. Box 3, South Chelmsford, MA 01824
MICRO is a new bimonthly publication specializing in information related to 6502 processor based systems.

159. Salzsieder, Byron, "Cheap Memory for the KIM-1", MICRO No. 1 pp 3-4, Oct.-Nov., 1977)
You can add a Veras Systems 4K Byte memory board to your KIM-1 at half the price of the KIM-2.

160. Holt, Oliver, "Terminal Interface Monitor (TIM) for the 6502", MICRO No. 1, pp 1-7 (Oct.-Nov., 1977)
TIM is available on a MOS Technology ROM 6530.

161. Anon., "We're No. 1", MICRO, No. 1, p 6 (Oct-Nov, 1977)
An editorial points out that over 12,000 KIM-1 units are in the field and a thousand more each month are being ordered. Apple I and Apple II systems, plus the OSI units, Jolts, Data Handlers, and other 6502 based systems, plus the huge number of PETs and Microminds that have been ordered, plus a lot of home-brew systems, it all adds up to a lot of 6502 systems. Also Atari has purchased one and one-half million 650X chips for their game units.

162. Ferruzzi, Arthur, "Inside the Apple II", MICRO, No. 1, pp 9-10 (Oct-Nov 1977)
A detailed description of the Apple II.

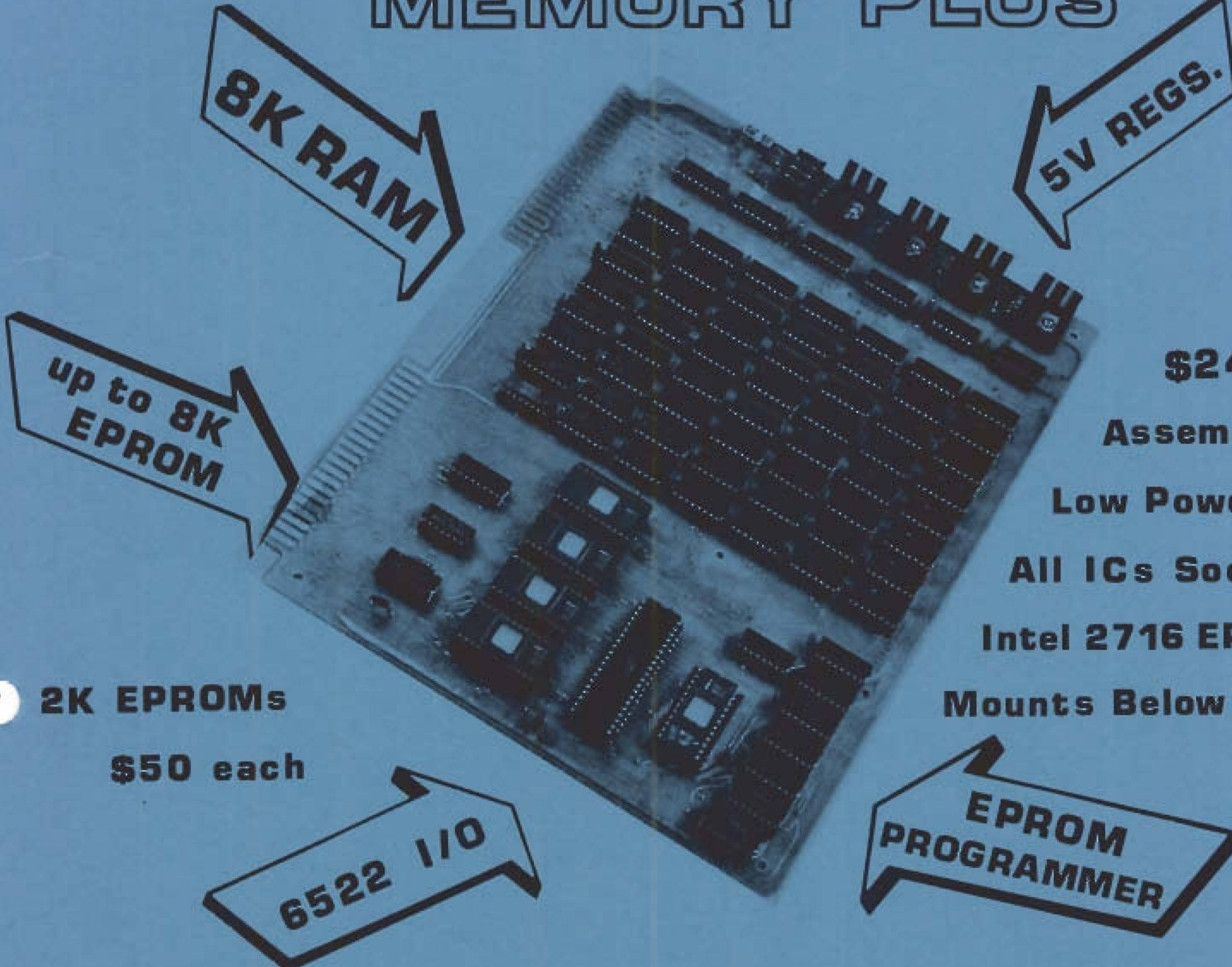
163. Ferruzzi, Arthur, "Rockwell International and the 6502", MICRO, No. 1, p 10, (Oct.-Nov., 1977)
Rockwell is now second sourcing the entire 6502 product line. They have also developed SYSTEM 65, a fancy development system with dual mini floppies, 16K static RAM, text editor, assembler and debug monitor on ROM, serial and parallel interfaces for terminal and printer, hardware breakpoint, etc.

164. Floto, Charles, "The PET's IEEE-488 Bus: Blessing or Curse?", MICRO, No. 1, p 11 (Oct.-Nov, 1977)
Discussion of this feature mentions a rumor that Pickles and Trout may offer a 488 adapter for their new S-100 I/O board, as well as an I/O board for the 488 bus.

165. Anon., "6502 Related Companies", MICRO, No. 1, p12 (Oct.-Nov., 1977)
Lists 28 companies serving 6502 processors.
166. Tripp, Robert M., "Hypertape and Ultratape", MICRO, No. 1, pp13-16, (Oct.-Nov., 1977)
Ultratape runs at 12 times the normal KIM-1 speed, but requires special programs for both loading and dumping.
167. Rowe, Mike, "KIM-Based Degree Day Dispatcher", MICRO, No. 1, pp 17-18, (Oct.-Nov., 1977)
Hundley Controls of Hanover, Mass. is building a number of different KIM-based systems to be used by fuel oil dealers to perform a variety of functions such as meter ticket reading, basic accounting, calculating degree-days by measuring temperature and determining when oil deliveries are to be made, etc.
168. Tripp, Robert M., "Computer Controlled Relays", MICRO, No. 1 p 19 (Oct.-Nov., 1977)
Relays can be used for control of audio assettes, and a variety of other functions. A 7404 Hex Inverter is used to buffer the signals from the KIM's 6530 Port B I/O lines.
169. Dial, William R., "6502 Bibliography", MICRO, No. 1, pp 21-27, (Oct. - Nov., 1977)
128 references to 6502 related articles, programs, etc.
170. Camus, Armand L., "Making Music with the KIM-1", MICRO, No. 2, pp 3-7, (Dec. 1977-Jan. 1978)
How to write music for a DAC such as that recently described by Chamberlain in Byte Magazine, Sept. 1977.
171. Floto, Charles, "Meet the PET", MICRO, No. 2, pp 9-10 (Dec 1977-Jan 1978)
An owners view of the PET 2001.
172. Dejong, Marvin L., "Digital-Analog and Analog-Digital Conversion Using the KIM-1", MICRO, No 2, pp 11-15, (Dec. 1977-Jan 1978)
Experiments with a KIM-1 controlled DAC/ADC.
173 Wallace, Bob, "The PET Vs. the TRS-80", MICRO No. 2, pp 17-18, (Dec. 1977 - Jan 1978)
A feature-by-feature comparison.
174. Schwartz, Marc, "Ludwig von Apple II", MICRO, No. 2, p 19 (Dec 77-Jan 78).
How to write music for the Apple II.
175. Anon., "MICROBES - Tiny Bugs in Previous MICRO", MICRO, No. 2, p 22, (Dec. 1977 - Jan. 1978).
Some corrections for HYPERTAPE and ULTRATAPE and Computer Controlled Relays.
176. Henkel, Joel, "The Challenge of the OSI Challenger", MICRO, No. 2, pp 23-24 (Dec 1977 - Jan 1978)
An owners impressions of the OSI Challenger.
177. MOS Technology, "Improving Keyboard Reliability", MICRO, No. 2, p 25, (Dec 1977 - Jan 1978)
A hardware modification for your KIM-1 to improve action of the "9, D, or C" keys. Based on an Application Note by MOS Technology.
178. Dial, William, "Important Addresses of KIM-1 and Monitor", MICRO, No. 2, pp 27-30, (Dec 1977 - Jan 1978)
A programmers reference card for the KIM-1.
179. Computer Shop, 288 Norfolk St., Cambridge, MA 02139, MICRO, No. 2, p 26, (Dec. 1977 - Jan. 1978)
Advertisement for CS 100 Video Terminal Board for KIM. Includes portable cabinet for the KIM with space for cassette recorder, ASCII keyboard, power supply, extra memory boards, 3-slot motherboard, TIM kit, etc.

Three PLUSes for the KIM-1:

MEMORY PLUS™



$245.00

Assembled

Low Power RAM

All ICs Socketted

Intel 2716 EPROMs

Mounts Below KIM-1

2K EPROMs

$50 each

ENCLOSURE PLUS™ — $30.00

Made by "The Enclosures Group" especially for the KIM-1/MEMORY PLUS combination. The MEMORY PLUS is mounted directly below the KIM-1 providing a compact package about 2.5" high which affords your system a high degree of protection from damage, dust, curious fingers, etc.

POWER PLUS™ — $40.00

Designed specifically for the KIM-1. It has regulated +5V and +12V for the KIM-1 and more than enough unregulated +8V to power the MEMORY PLUS. It is completely enclosed in a black bakelite case measuring about 6.8" by 5.6" by 3". It is fully assembled and tested and weighs about 3 lbs.

MEMORY PLUS is $245 with everything except EPROMs.
KIM-1/MEMORY PLUS Cables are $10.00
Includes 60 page manual, cassette tape, connectors.

The COMPUTERIST
P.O. Box 3
S Chelmsford, MA 01824
617/256-3649

LIGHTING THE KIM-1 DISPLAY

Marvin L. De Jong
The School of the Ozarks
Point Lookout, MO 65726

To light the display, ports A and B data direction registers (SADD and SBDD) must be loaded so that pins SA0-6 and SB1-4 are output pins.

Put $7F in SADD at location 1741
Put $1E in SBDD at location 1743

Numbering the digits on the display from left to right, Table 1 indicates the number to load to light the segment in each digit.

SBD (1742)	Digit Enabled
08	1
0A	2
0C	3
0E	4
10	5
12	6

Table 1

Lettering the segments within the digit as shown in Figure 1, Table 2 indicates the number to load into SAD to select each segment.

To display any character, add the hex numbers in Table 2 which correspond to the segments to be lighted, then add 80. This latter step insures that pin PA7 remains high. Table 3 shows the values required for the sixteen Hex digits.

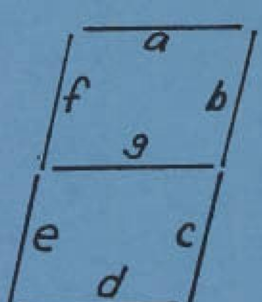


Figure 1

SAD (1740)	Segment Lit
01	a
02	b
04	c
08	d
10	e
20	f
40	g

Table 2

A table with these hex numbers is located in ROM on the KIM-1. It starts at 1FE7 and ends at 1FF6. To access the table, use LDAX TABLE, where TABLE is 1FE7 and X is the number to be displayed, then STA SAD, making sure that the appropriate digit is enabled.

SAD (1740)	Character
BF	0
86	1
DB	2
CF	3
E6	4
ED	5
FD	6
87	7
FF	8
EF	9
F7	A
FC	B
B9	C
DE	D
F9	E
F1	F

Table 3

Table 4 is a character generator for the alphabet and a couple of other characters. Note that some characters simply can not be made with the seven segment display.

Char	Upper	Lower	Char	Upper	Lower
A	F7		O	BF	DC
B		FC	P	F3	
C	B9	D8	Q		E7
D		DE	R		D0
E	F9		S	ED	
F	F1		T		F8
G	BD	EF	U	BE	9C
H	F6	F4	V		
I	B0		W		
J	9E		X		
K			Y	EE	
L	B8	86	Z	DB	
M			?	D3	
N	B7	D4	-	C0	

Table 4