SWTPC 6800 COMPUTER NEWSLETTER
Southwest Technical Products Corporation
219 W. Rhapsody
San Antonio, Texas 78216
Issue No. 2. October 1976

SWTPC WELL REPRESENTED AT PCC '76

Southwest Technical Products had a busy double booth at the Personal Computing Convention on August 28th and 29th in Atlantic City, New Jersey. We had two complete systems up and running and were amazed at the number of people who came by to ask questions and buy their copy of 4K Basic (C). Microcomputer Systems Inc., one of our Florida dealers, helped us out by providing us with some knowledgeable personnel and stock for those customers wanting to take their purchases with them

The convention was obviously an outstanding success. Our only complaint was that the convention was held in Atlantic City. It seemed that most all of the hotels/motels in the area left a lot to be desired. We always had to travel in groups (safety in number algorithm). The lack of air conditioning in the convention exhibit area was of no help either. The equipment worked but the operators didn't. Maybe us Texans leave been spoiled by all of this good living.

Pictured above from left to, right are Bill Thames, Rep. of Microcomputer Systems; Steven Uiterwyk, Software Debugger; Robert Uiterwyk, Author of Microbasic and 4K Basic(C); Gary Kay, SWTPC Engineer; Dan Meyer President of SWTPC; and Joe Deres, SWTPC Engineer. The photo was taken by Jim Stratigos of the Atlanta Area Microcomputer Hobbyist Club. Jim spent most of his time at the convention helping us out at the booth. Also helping were Bill Blomgren, Rep. of Microcomputer Systems; Forrest Hurst, Mgr. of Microcomputer Systems; Warren Startup, Representative of Microcomputer Systems; and last but not least Steven Uiterwyk, Software Debugger.

SWTPC 4K BASIC VERSION 2.0 (C)

We are now distributing 4K Basic Version 2.0 (C), for those of you who bought versions 1.0 (C) or 1.1 (C). The differences and improvements are noted below. At the prices we are charging we cannot afford to continuously update tapes for those customers who bought an outdated version. So if you want version 2.0, send us another $4.95 and we will send you a new cassette tape and manual.

Features

1) An APPEND command has been added which functions as a LOAD command, but does not erase the previous source file.

2) On an INPUT command, failure to enter multiple variables when required is now properly handled. Also, entering an erroneous value such as a null or string is responded to by a reply of "RE-ENTER" instead of a forced exit from the Basic program.

3) Using Control X for line cancel now gives the reply "DEL" for deleted.

4) You may now use either CHR or CHR$ to call the character function. The CHR$ allows you to enter hex values rather than decimal values which are used by the CHR function.

5) FOR-NEXT loops will now allow a premature exit from a nested group without issuing an error 18. It should also be noted that 4K Basic Versions 1.1 and 2.0 (C) allow negative steps. This was not noted in the manual. Example: FOR I=10 TO -1 STEP -1

6) The INT function has been corrected so that values between 0 and -1.0 now correctly return -1.

7) The overflow checking routine (error 14) did not catch an overflow if it wrapped around memory. Example: DIM C4(255,255). This has been corrected.

8) Addition, subtraction and comparisons did not work correctly if one element was zero, and the other less than 1.E-10. This has been corrected.

9) Entry of an incorrectly formatted LIST command (Example: LIST 40,10) will now merely list from line 40 to the end of the source file.

10) Occasionally Basic 1.1 (C) gave an error 13 when it should have been an error 6. This has been corrected.

11) One of the fixes incorporated into Basic 1.1 (C) removed the truncation of some mathematical results to nine digits. This has been restored.

12) The system will no longer insert a null line. A null line is an entry with a line number but no statement following it.

13) 0.001 - 0.01 erroneously gave a positive answer. This has been fixed.

14) An underflow of 1.E-100 or 1.E-101 was not recognized and the value was not set to zero, thus a funny print value was returned. This has been corrected.

15) A multiplication or equivalent division of a small number (Example: 1E-70 * 1E-70) such that the result was less than 1E-127 gave a result of 9.99999999E99 rather than zero. This has been fixed.

The following corrections were made between Basic 1.0 (C) and 1.1 (C):

1) A divide 90 by 9.XXXX gave a result of zero. This was fixed.

2) Repeated multiplications of zero times a number occasionally created an error. This has been corrected.

3) SGN(0) was originally defined to be 1 whereas it should have been 0. This was changed in version 1.1 and you are asked to change your manuals accordingly.

PATCH FOR 4K BASIC (C)

If you have been having problems with the 4K Basic (c) SAVE and LOAD commands not working properly with the AC-30 cassette interface try the following patches: Version 1.0 and 1.1 - make $029D_{16}$ a $1E_{16}$. Version 2.0 make $02AF_{16}$ a $1E_{16}$.

SWTPC 8K BASIC

SWTPC 8K Basic (C) is now in the field testing stage and we expect to start delivering tapes and manuals soon. The price will be $9.95 for the "Kansas City" cassette tape and manual combination. 8K Basic (C) has the features of 4K basic (C) plus string handling capabilities with string arrays, trigonometric functions, exponentials and all sorts of other neat things. It is fundamentally full ANSI Basic with some additions and some small limitations due to the fact that we are using a microcomputer rather than an IBM 370.

MOD FOR MICROBASIC

If you are still using the SWTPC Microbasic package you might want to change memory location 036C from a 08 to a 1B. This will insure more random numbers when using the random number generator. If you have been having problems with your Microbasic not working properly and the memory diagnostics all check good, then you probably transposed some B's with some 8's or vice versa when you loaded Microbasic. The version of Microbasic printed in the first newsletter is accurate and does work:

ANIMALS FOR THE SWTPC 6800

SWTPC is now offering the ANIMALS game for the SWTPC 6800 thanks to the efforts of Doug Domke. The game of "Animals" was originally written in BASIC at DEC by Nathan Teichholz and is presented in "101 Computer Games" as an example of artificial intelligence.

This version of the game unlike the original which uses disk files, operates completely within the RAM of a SWTPC 6800 Computer System. Although the program itself only requires 650 bytes of code, it theoretically has the ability to use up any amount of RAM as it "learns." In practice, however, even a 2K system is large enough to make the game very enjoyable.

The program itself is actually a guessing game. The program attempts to guess what animal the user is thinking of. If it fails, it will request the name of the animal, followed by a request for the user to supply a question which, when answered to the affirmative, would distinguish the correct answer from that guessed (incorrectly) by the program. This information then becomes a part of the program.

In response to the questions "Are you thinking of an animal?", the user may respond in one of four ways:

Y(ES) - the game continues

N(0) - the program exits to MIKBUG

L(IST) - the program lists its vocabulary of animals

C(LEAR) - the program clears everything it has learned and reverts to its initialized (dumb) state.

SWTPC CASSETTE TAPE PROGRAM LIBRARY

The following programs are available from SWTPC on AC-30 (Kansas City) formatted audio cassette tape:

MP-EC Editor/Assembler package with manual. Requires 8K of memory to run. $14.95 ppd. in US.

BAS4C 4K Basic Version 2.0 (C) with manual. Requires at least 6K of memory - 8K preferred. $4.95 ppd in US.

GAMIC Tic-Tac-Toe and Blackjack as listed in the notebook and newsletter. Require 6K of memory to run. $4.95 ppd in US.

BAS8C 8K Basic (Available shortly) with manual. Requires at least 8K of memory12K preferred. $9.95 ppd in US.

ANIMC Animals program. A children's learning game. Requires 2K of memory. $4.95 ppd. in US.

SWTPC BASIC AVAILABLE ON PAPER TAPE

SWTPC has decided to offer 4K and 8K Basic (C) on paper tape:

BAS4P 4K Basic Version 2.0 (C) with manual. Requires at least 6K of memory - SK preferred. $10.00 ppd. in US.

BAS8P 8K Basic with manual. Requires at least 8K of memory - 12K preferred. $20.00 ppd. in US. (Available Nov. 15, 1976.)

MOD FOR THE AC-30 CASSETTE INTERFACE

If you have been having trouble with your AC-30 not loading programs properly to the computer or while operating with the terminal in the LOCAL mode, try changing capacitor C10 on the AC-30 from a 1000 pfd capacitor to a 2700 pfd capacitor. Another suggestion from Rick Hammond of CBS Radio is to change resistor R22 from a 10K ohm to a 1K ohm resistor. These changes insure the reliability of the AC-30 generated UART clock.

In order to prevent damage to the AC-30's reed relays by tape recorders with high motor transient currents, it is suggested that you make the following component additions:

( ) Insert a 2.2 ohm 1/2 watt resistor in series with the wire going to jack J10 terminal A. Install the resistor right at jack J10.

( ) Solder a 0.1 mfd capacitor across jack J10 terminals A and B.

( ) Insert a 2.2 ohm 1/2 watt resistor in series with the wire going to jack J11 terminal A. Install the resistor right at jack J11.

( ) Solder a 0.1 mfd capacitor across jack J11 terminals A and B.

If you have easy access to the REMOTE jack terminals on the inside of the tape recorder, attach a 200 PIV 1 amp diode (1N4001) across the two terminals. Be sure you install the diode so it is oriented with the banded end to the positive terminal. Use a voltmeter to determine which terminal is which. If you put the diode in backwards, the remote circuit will no longer stop the recorder's motor.

CLEAN PAGE MOD FOR THE CT-1024

If you would like to have your CT-1024 come up with an erased frame each time the cursor cycles thru to the top of a new page, you can do so with a one modification from the top ride of the main board:

Using a pair of pointed dykes, carefully cut the pin 3 lead of integrated circuit IC 2 right where the lead goes into the top foil of the main circuit board. Carefully pry this lead up and away from the foil and solder a 12" piece of light gauge wire to it. Attach and solder the other end to pin 6 of IC 28.

Now each time the terminal starts printing at the top of a new page, the screen will first be erased. The erase will be complete if the cursor is not located in the first character position when the new page is selected so take note.

PLEXIGLASS ENCLOSURE FOR THE CT-1024

Microcomputer Systems Inc. of Tampa, Florida has developed an enclosure which will hold a SWTPC CT-1024 Terminal System with power supply. It even has room left over to fit in a GT-61 Graphics Terminal along with its power supply, so the benefits of having both inside the same enclosure may be realized.

The case is made of l/4" PLEXIGLASS and is 15" wide X 17" deep X 6" tall. It comes in two colors: clear and smoked. The smoked color is a dark bronze and makes it difficult to see the components inside the case. There are holes provided for two DB-25 (RS-232) connectors as well as several holes for ventilation. There is a large cutout for the SWTPC KBD-5 keyboard with PLEXIGLASS standoffs for mounting it. The case even has molded feet and polished edges. The cost is $65.00 ppd. in the US. (specify color when ordering)

Order from:

Microcomputer Systems,Tnc.
144 S. Dale Mabry
Tampa, Florida 33609
(813) 879-4301 879-4225

PLEXIGLASS COVER FOR THE PR-40

Microcomputer Systems Inc. of Tampa, Florida is also offering a Plexiglas cover for our PR-40 Alphanumeric Printer. The cover slips right over the PR-40's chassis and is available in clear and smoked versions. The cost is $20.00 ppd. in the US. (specify color when ordering)

Order from:
Microcomputer Systems Inc.
144 S. Dale Mabry
Tampa, Florida 33609
(813) 879-4301 879-4225

METAL ENCLOSURE FOR THE CT-1024

If you have been looking for a metal enclosure for your CT-1024 Terminal System, you might want to contact:

D.W. Ekstrand
P.O. Box 1260E
Southgate, Calif. 90280
(213) 566-1677

This man has been making attractive cases for the terminal system for some time now. The cases are allodyned aluminum with welded seams. Configurations are offered for several keyboards so write the man for complete information and pricing.

SWTPC 6800 PROTOTYPING BOARDS

If you have been looking for prototyping boards for the SWTPC 6800 Computer System, Personal Computing Company is now selling prototyping cards for both the large and small board positions. The PC boards are single sided plated boards and will accept 14, 16, 24, and 40 pin IC sockets. Both boards are provided with holes for SWTPC 6800 compatible connectors on both the top and bottom edges, however the Molex connectors are not supplied. Provisions have been made on each card for an on board 7805 regulator also not supplied. The large board sells for $19.95 while the small goes for $9.95. See their advertisement with photographs on page 112 in the October 1976 issue of 73 magazine.

Personal Computing Company
3321 Towerwood Drive, Suite 107
Dallas, Texas 75234

NEWS FROM MIDWEST SCIENTIFIC

Midwest Scienfific tells us they have two Basic packages, a disassembler, and miniassembler that are SWTPC 6800 compatible. They are also advertising a PROM board and a floppy disk with software Check with them on prices and delivery. See their ad on pages 58 and 59 of the October 1976 issue of Byte magazine.

Midwest Scientific Instruments Inc.
220 West Cedar
Olathe, Kansas 66061

EPROM BOARDS COMING FOR THE SWTPC 6800

We have been notified by two independent firms that they are working on EPROM boards plug compatible with SWTPC 6800 Computer System. Neither manufacturer has a sellable product at this time, however, both should be available by the time of our next newsletter. The PROM boards will use either 2704 or 2708 EPROMS. 1702's are too slow for the SWTPC 6800's memory cycle time.

PR-40 CHARACTER ADDITION

According to David M. Alexander, the underline character ($5F_{16}$) can easily be added to the SWTPC PR-40 Printer's Character font by the addition of two short wires which are tack soldered to the top or bottom of the PR-40 main circuit board. A spare open collector buffer is used to "wire-or" the pin 15 and 16 outputs of the character generator, IC2, into the input circuit of solenoid driver #7. The modification should be made as follows:

1) connect a wire from,IC2 pin 15 to IC7 pin 11.

2) connect a wire from IC7 pin 10 to IC7 pin 6.

MORSE CODE PROGRAM FOR THE SWTPC 6800

If you are interested in a Morse code program for the SWTPC 6800 Computer System, you might want to check out Wayne Sewell's program which appeared on page 42 of the October 1976 issue of Byte Magazine. Although the program was written to run on any 6800 system, it was written specifically for the SWTPC 6800.

FASTER PROGRAM LOADS FROM CASSETTE

SWTPC will be supplying the longer program cassette and paper tapes such as Basic (C) in binary form. By doing this program loading time will be decreased by a factor from two or three to one, depending upon the length of the program. At the beginning of each tape is an ASCII formatted binary load routine. When you start loading your program tape, the computer actually loads in the binary loader program first. The G necessary to start this program is actually recorded on the tape so the program initiates itself. Following the binary loader program is the actual program to be loaded. The program is stored on the tape in binary so its length is about one third that of the same program stored in ASCII. The memory locations used by the binary loader program have been chosen so as not to conflict with the memory locations of the program to be loaded into memory.

MODIFICATION TO BASIC 2.0 (C) TO DRIVE

THE SWTPC PR-40 ALPHANUMERIC LINE PRINTER

By: W. C. Thames
Microcomputer Systems, Inc.

When the following code is added to Basic Version 2.0 (C), the PR-40 printer will be operated directly from the Basic interpreter. Basic will then print data on the terminal first, followed by the same data on the PR-40 printer. Changing the data in memory location 1219_{16} from a 7E to a 39 will eliminate the terminal's printout entirely. If you wish to disable just the PR-40's printout, simply turn off the PR-40. Do not turn off the printer while the computer is outputting data. Doing so will interrupt the normal handshake routine and will cause the computer to lock up. This modification is only usable on the SWTPC PR-40 Alphanumeric Printer and in most cases will not drive other printers.

The patch has been written to output all printer data to a MP-L I/0 board located at I/0 card position #7 of a SWTPC 6800 Computer System. If you wish to relocate this interface to another card position simply change the address in memory locations 1204_{16} and $121D_{16}$ to the address of the desired MP-L parallel interface board.

```
1200 37     121A E1     1234 F5     0101 12
1201 DF     121B D1     1235 8D     0102 1C
1202 1E     121C CE     1236 F3
1203 CE     121D 80     1237 86     0273 12
1204 80     121E 1C     1238 20     0274 2A
1205 1C     121F C6     1239 20     0276 12
1206 A7     1220 FF     123A 0E     0277 33
1207 00     1221 E7     123B 44
1208 C6     1222 00     123C 44     027B 12
1209 36     1223 C6     123D 44     027C 00
120A E7     1224 3E     123E 44
120B 01     1225 E7     123F 84     02AF 1E
120C C6     1226 01     1240 0F
120D 3E     1227 7E     1241 8B     07F9 12
120E E7     1228 08     1242 30
120F 01     1229 0E     1243 81     07FA 4C
1210 6D     122A A6     1244 39
1211 01     122B 00     1245 23
1212 2A     122C 8D     1246 02
1213 FC     122D 0D     1247 8B
1214 E6     122E A6     1248 07
1215 00     122F 00     1249 7E
1216 DE     1230 08     124A 12
1217 1E     1231 20     124B 00
1218 33     1232 0C
1219 7E     1233 8D
```

4K AND 8K BASIC (c) EXTERNAL SUBROUTINE CALLS

Basic Version 1.0 and 1.1

```
0260  7E   E0BF  OUT2H    JMP   $E0BF    DUT2H IN MIKBUG

0263  7E   E0C8  OUT4HS   JMP   $E0C8    OUT4HS IN MIKBUG

0266  8D   09    OUTCH    BSR   BREAK
0269  7E   E1D1           JMP   $E1D1

0268  BD   E1AC  INCH     JSR   $E1AC    INEEE IN MIKBUG
026E  36                  PSH A
026E  20   08             BRA   BREAK0

0271  36         BREAK    PSH A
0272  B6   8004           LDA A $8004
0275  2B   09             BMI   BREAK1
0277  8D   F2             BSR   INCH
0279  81   03    BREAK0   CMP A #$03     .
027B  26   03             BNE   BREAK1
027D  7E   0812           JMP   READY
0280  32         BREAK1   PUL A
0281  39                  RTS
```

Basic Version 2.0

```
0272  7E   E0BF  OUT2H    JMP   $E0BF    DUT2H IN MIKBUG

0275  7E   E0C8  OUT4HS   JMP   $E0C8    OUT4HS IN MIKBUG

0278  8D   09    OUTCH    BSR   BREAK
027A  7E   E1D1           JMP   $E1D1

027D  BD   E1AC  INCH     JSR   $E1AC    INEEE IN MIKBUG
0280  36                  PSH A
0281  20   08             BRA   BREAK0

0283  36         BREAK    PSH A
0284  B6   8004           LDA A $8004
0287  2B   09             BMI   BREAK1
0289  8D   F2             BSR   INCH
028B  81   03    BREAK0   CMP A #$03     .
028D  26   03             BNE   BREAK1
028F  7E   0815           JMP   READY
0292  32         BREAK1   PUL A
0293  39                  RTS
```

RELATIVE ADDRESS CALCULATOR PROGRAM

By: Russel Yost

This Relative Address Calculator Program may be used to calculate relative addresses for branch instructions. This is especially useful when calculating long branches where you are more likely to make an error if you do it by hand. This program lets the computer do the work for you.

To use the program, type in the machine code listing on the next page. The entire program fits inside the scratchpad RAM used by Mikbug®. Be sure to save the program on tape if you have a tape unit connected to your computer. After loading the program type a G for "Go to User Program". The computer will home the cursor and erase the screen on those systems using the CT-1024 Terminal System with the CT-CA option. It will then print out a BA which stands for "branch address". To this you should respond with the address of the branch instruction and not the address following it. The program will then output a T which stands for "TO". Now you type the destination address of the branch instruction. The program outputs a = followed by the relative address. If branching forward, the outputted address will be OOXX and you must be sure not to have XX greater than 7F. If branching backwards, the outputted address will be FFYY and you must be sure to have YY greater than 7F. Only the last two digits of the outputted address are used for the relative address.

If any non-hex character is input at either address, the program jumps to Mikbug® and outputs a *. Upon entering Mikbug®, typing a G will restart the Relative Address Calculator Program. After calculating each relative address, the program prepares itself for new data. When using the CT-1024 Terminal System, the program will home and erase the terminal's screen after each calculation.

Mikbug® is a registered trademark of Motorola, Inc.

```
                         NAM    RELADR
                  * MIKBUG LOCATIONS
 E07E             PDATA1  EQU    $E07E
 E047             BADDR   EQU    $E047
 E0C8             OUT4HS  EQU    $E0C8

 A000                     ORG    $A000
 A000             RAMST   RMB    2
 A002             BRADR   RMB    2
 A004             DEST    RMB    2

 A014                     ORG    $A014
 A014 8E A0 47    BEGIN   LDS    #$A047    Saves BEGIN in  A048,49
 A017 CE A0 6F            LDX    #MSETUP   Clears screen
 A01A 8D 4F               BSR    PDATSR    See subroutine below.
 A01C BD E0 47    NEXT    JSR    BADDR     Gets 4 hex & store in X
 A01F FF A0 02            STX    BRADR     Stores branch addr
 A022 CE A0 77            LDX    #MT       Outputs " T "
 A025 8D 44               BSR    PDATSR
 A027 BD E0 47            JSR    BADDR
 A02A FF A0 04            STX    DEST      Stores dest'n addr.
```

```
A02D CE A0 00          LDX    #RAMST    Prepare for indexed
A030 0C                CLC              Addr. mode.
A031 A6 04             LDA A  4,X       DEST H
A033 E6 05             LDA B  5,X       DEST L
A035 20 13             BRA    CONTN

A048                   ORG    $A048
A048 A0 14             FDB    BEGIN

A04A C0 02     CONTN   SUB B  #02       Subtract 0002 from
A04C 82 00             SBC A  #00       Destination Addr.
A04E 0C                CLC
A04F E0 03             SUB B  3,X       BRA L Subtract Br addr
A051 A2 02             SBC A  2,X       BRA H (Destn. - 2)
A053 A7 69             STA A  $69,X     REL H Store at REL
A055 E7 6A             STA B  $6A,X     REL L
A057 CE A0 7B          LDX    #MEQ      Outputs " = "
A05A 8D 0F             BSR    PDATSR
A05C CE A0 69          LDX    #REL
A05F BD E0 C8          JSR    OUT4HS    Outputs 4 hex's + Sp.
A062 CE A0 71          LDX    #MBA      Outputs CR, LF, "BA "
A065 8D 04             BSR    PDATSR
A067 20 B3             BRA    NEXT
A069           REL     RMB    2
A06B BD E0 7E  PDATSR  JSR    PDATA1    Outputs string
A06E 39                RTS

A06F 10        MSETUP  FCB    $10,$16
A070 16
A071 0D        MBA     FCB    $0D,$0A,$42,$41,$20,$04
A072 0A 42
A074 41 20
A076 04
A077 20        MT      FCB    $20,$54,$20,$04
A078 54 20
A07A 04
A07B 20        MEQ     FCB    $20,$3D,$20,$04
A07C 3D 20
A07E 04
               END

NO ERROR(S) DETECTED

SYMBOL TABLE:
BADDR  E047   BEGIN  A014   BRADR  A002   CONTN  A04A   DEST   A004
MBA    A071   MEQ    A07B   MSETUP A06F   MT     A077   NEXT   A01C
OUT4HS E0C8   PDATA1 E07E   PDATSR A06B   RAMST  A000   REL    A069
```

BINARY LOAD AND PUNCH

To easily decrease the amount of time it takes to load a long tape (Cassette or paper) a BINARY formatting technique can be used instead of the conventional ASCII format used by the punch and load routines in MIKBUG. The two following programs, BILOAD and BIPNCH are two such programs necessary for punching your own binary data tapes. The punch routine is designed to automatically punch a program that is in several sections or a program and its program counter. Load your program and the BIPNCH into your system and set up the following locations in the MIKBUG RAM:

A014	Number of sections to be dumped (02 if you have a one piece program and program counter)
A015	Starting address of program dump (1st block)
A016	
A017	End address of program dump (1st block)
A018	
A019	Starting address of second block (or pgm. ctr.)
A01A	
A01B	End address of second block
A01C	
etc.	

(Above similar to setting up A002-etc. for MIKBUG punch)

A028	The hex value of the program counter for your program. The
A029	data in these locations is transferred to A048 and A049 automatically and punched in binary.

Executing the BIPNCH program at 1E04 will punch your program onto either cassette or paper tape. Note - Be sure to have your READ switch on a cassette tape loader such as the AC-30 in the off position during a binary punch.

Since you are dumping in binary rather than ASCII, do not expect to see the usual S11 format as during a MIKBUG punch. All you will see will be random characters.

Using the BILOAD program is quite straightforward. Simply use it the same way you use the "L" command in MIKBUG except that you are executing a loader program at 1703 instead of typing an L. The loader will give you a register dump when loading is complete.

If you desire, the loader program can be put on the beginning of each binary tape to save you time in loading. Use the following procedure to make such a tape.

1) Load in BILOAD and BIPNCH into memory. Load in the program to be dumped. Set A048 and Ad49 to 1703. Set A002- A005 to 1700 177F. Set up locations A614, A028, etc. as described earlier.

2) Execute the MIKBUG P command. Set A002-A005 to A048 and A049 and execute P.

3) Switch to local and put an S9 G on your tape. Be sure to leave a second or two dead time on both sides of the G.

4) In the remote mode, change A048 and A049 to lE04. Type G and the program will be punched in binary (Be sure to have locations AO", etc. set up correctly as described earlier).

The tape made this way will have the following stored on it:

BINLD FORMATTED IN ASCII	BINLD PGM CTR. (ASCII)	S9	G	USER PROGRAM FORMATTED IN BINARY

IMPORTANT NOTE:

Some terminals (such as the SWTPC CT-1024) will treat a 94_{16} the same as a $l4_{16}$ (Punch off). The BIPNCH is set up to correct for this. If your terminal does not see a 94_{16} as a 14_{16} you must change locations 1EB1-1EB4 to NOP'S (01).

If the program you with to dump or load occupies the same area of memory as either BTPNCH or BILOAD, you will need to re-assemble them to move them to other areas of memory. Be careful because the programs use the EXTENDED addressing mode in several places.

The BILOAD and BIPNCH are very similar to the ones SWTPC will use when formatting long cassette tapes in binary.

```
                            NAM     BILOAD
E1D1                OUTEEE  EQU     $E1D1
E1EF                DEL     EQU     $E1EF
E1F3                DE      EQU     $E1F3
E1E3                IOUT2   EQU     $E1E3
E115                DMPREG  EQU     $E115
E040                LOAD19  EQU     $E040
E1A5                SAV     EQU     $E1A5

1700                        ORG     $1700
1700                CKSM    RMB     1
1701                TW      RMB     2
1703 8E A0 47               LDS     #$A047
1706 8D 49          BILOAD  BSR     LOAD
1708 8D 3C          OVER    BSR     INPUT
170A 81 58                  CMP A   #'X
170C 26 FA                  BNE     OVER
170E 8D 36                  BSR     INPUT
1710 81 31                  CMP A   #'1
1712 27 07                  BEQ     READ
1714 81 39                  CMP A   #'9
1716 26 F0                  BNE     OVER
1718 7E E1 15               JMP     DMPREG
171B 7F 17 00       READ    CLR     CKSM
171E 8D 26                  BSR     INPUT
1720 16                     TAB
1721 5C                     INC B
1722 8D 22                  BSR     INPUT
1724 B7 17 01               STA A   TW
1727 8D 1D                  BSR     INPUT
1729 B7 17 02               STA A   TW+1
172C FE 17 01               LDX     TW
172F 8D 15          STORE   BSR     INPUT
1731 A7 00                  STA A   0,X
1733 01                     NOP
1734 A1 00                  CMP A   0,X
1736 26 0B                  BNE     OUT
1738 08                     INX
1739 5A                     DEC B
173A 26 F3                  BNE     STORE
173C 8D 08                  BSR     INPUT
173E 7C 17 00               INC     CKSM
1741 27 C5                  BEQ     OVER
1743 7E E0 40       OUT     JMP     LOAD19
1746 8D 14          INPUT   BSR     INCHP
1748 36                     PSH A
1749 BB 17 00               ADD A   CKSM
174C B7 17 00               STA A   CKSM
174F 32                     PUL A
1750 39                     RTS
1751 86 11          LOAD    LDA A   #$11
1753 BD E1 D1               JSR     OUTEEE
1756 86 3C                  LDA A   #$3C
1758 B7 80 07               STA A   $8007
175B 39                     RTS
175C 37             INCHP   PSH B
```

```
 175D BD E1 A5           JSR    SAV
 1760 A6 00      IN1     LDA A  0,X
 1762 2B FC              BMI    IN1
 1764 6F 02              CLR    2,X
 1766 BD E1 F3           JSR    DE
 1769 BD E1 EF           JSR    DEL
 176C C6 04              LDA B  #4
 176E E7 02              STA B  2,X
 1770 58                 ASL B
 1771 BD E1 EF  IN3      JSR    DEL
 1774 0D                 SEC
 1775 69 00              ROL    0,X
 1777 46                 ROR A
 1778 5A                 DEC B
 1779 26 F6              BNE    IN3
 177B BD E1 EF           JSR    DEL
 177E 7E E1 E3           JMP    IOUT2
                         END

NO ERROR(S) DETECTED

   SYMBOL TABLE:

BILOAD 1706   CKSM   1700   DE     E1F3   DEL    E1EF   DMPREG E115
IN1    1760   IN3    1771   INCHP  175C   INPUT  1746   IOUT2  E1E3
LOAD   1751   LOAD19 E040   OUT    1743   OUTEEE E1D1   OVER   1708
READ   171B   SAV    E1A5   STORE  172F   TW     1701
```

```
                        NAM     BIPNCH

E1D1            OUTEEE  EQU     $E1D1
E0E3            CONTRL  EQU     $E0E3

A014            MAX     EQU     $A014
A015            BEGA    EQU     $A015
A017            ENDA    EQU     $A017
A028            PCT     EQU     $A028


1DF7                    ORG     $1DF7

1DF7 A7 00      STRT    STA A   0,X
1DF9 B6 A0 29           LDA A   PCT+1
1DFC A7 01              STA A   1,X
1DFE 20 0F              BRA     BIPNCH
1E00 01         CNTR    FCB     01
1E01            TW      RMB     2
1E03            TEMP    RMB     1
1E04 8E A0 47           LDS     #$A047
1E07 CE A0 48           LDX     #$A048
1E0A B6 A0 28           LDA A   PCT
1E0D 20 E8              BRA     STRT
1E0F BD 1E 93   BIPNCH  JSR     PNON
1E12 BD 1E 99           JSR     PNLDR
1E15 FE A0 15   BEG     LDX     BEGA
1E18 FF 1E 01           STX     TW
1E1B F6 A0 18   PUND10  LDA B   ENDA+1
1E1E F0 1E 02           SUB B   TW+1
1E21 B6 A0 17   BEG1    LDA A   ENDA
1E24 B2 1E 01           SBC A   TW
1E27 27 02              BEQ     PUND25
1E29 C6 FF              LDA B   #$FF
1E2B 86 58      PUND25  LDA A   #'X
1E2D BD E1 D1           JSR     OUTEEE
1E30 86 31              LDA A   #'1
1E32 BD E1 D1           JSR     OUTEEE
1E35 37                 PSH B
1E36 5F                 CLR B
1E37 30                 TSX
1E38 8D 69              BSR     PUN
1E3A 32                 PUL A
1E3B 4C                 INC A
1E3C B7 1E 03           STA A   TEMP
1E3F CE 1E 01           LDX     #TW
1E42 8D 5F              BSR     PUN
1E44 8D 5D              BSR     PUN
1E46 FE 1E 01           LDX     TW
1E49 8D 58      PUND30  BSR     PUN
1E4B 7A 1E 03           DEC     TEMP
1E4E 26 F9              BNE     PUND30
1E50 FF 1E 01           STX     TW
1E53 53                 COM B
1E54 37                 PSH B
1E55 30                 TSX
1E56 8D 4B              BSR     PUN
```

```
 1E58 33                PUL B
 1E59 FE 1E 01          LDX    TW
 1E5C 09                DEX
 1E5D BC A0 17  CP1     CPX    ENDA
 1E60 26 B9             BNE    PUND10
 1E62 B6 1E 00          LDA A  CNTR
 1E65 B1 A0 14          CMP A  MAX
 1E68 27 1C             BEQ    EXIT
 1E6A 7C 1E 00          INC    CNTR
 1E6D B6 1E 17          LDA A  BEG+2
 1E70 8B 04             ADD A  #4
 1E72 B7 1E 17          STA A  BEG+2
 1E75 B6 1E 23          LDA A  BEG1+2
 1E78 8B 04             ADD A  #4
 1E7A B7 1E 23          STA A  BEG1+2
 1E7D B7 1E 5F          STA A  CP1+2
 1E80 4C                INC A
 1E81 B7 1E 1D          STA A  PUND10+2
 1E84 20 89             BRA    BIPNCH
 1E86 86 58     EXIT    LDA A  #'X
 1E88 BD E1 D1          JSR    OUTEEE
 1E8B 86 39             LDA A  #'9
 1E8D BD E1 D1          JSr    OUTEEE
 1E90 7E E0 E3          JMP    CONTRL
 1E93 86 2C     PNON    LDA A  #812
 1E95 BD E1 D1          JSR    OUTEEE
 1E98 39                RTS
 1E99 C6 05     PNLDR   LDA B  #5
 1E9B 4F        PNULL   CLR A
 1E9C BD E1 D1          JSR    OUTEEE
 1E9F 5A                DEC B
 1EA0 26 F9             BNE    PNULL
 1EA2 39                RTS
 1EA3 A6 00     PUN     LDA A  0,X
 1EA5 BD E1 D1          JSR    OUTEEE
 1EA8 EB 00             ADD B  0,X
 1EAA A6 00             LDA A  0,X
 1EAC 08                INX
 1EAD 81 14             CMP A  #$14
 1EAF 27 E2             BEQ    PNON
 1EB1 81 94             CMP A  #$94
 1EB3 27 DE             BEQ    PNON
 1EB5 39                RTS
                        END

NO ERROR(S) DETECTED

   SYMBOL TABLE:

BEG    1E15   BEG1   1E21   BEGA   A015   BIPNCH 1E0F   CNTR   1E00
CONTRL E0E3   CP1    1E5D   ENDA   A017   EXIT   1E86   MAX    A014
OUTEEE E1D1   PCT    A028   PNLDR  1E99   PNON   1E93   PNULL  1E9B
PUN    1EA3   PUND10 1E1B   PUND25 1E2B   PUND30 1E49   STRT   1DF7
TEMP   1E03   TW     1E01
```

HIGH SPEED PAPER TAPE READER SOFTWARE

Recently an ad has appeared in several magazines for an optical high speed paper tape reader by Oliver Audio Engineering, 7330 Laurel Canyon Blvd., North Hollywood, Calif. 91605, (213) 765-8080. Some SWTPC 6800 compatible software has been developed for this unit which we are passing along here.

The following programs assume that the required parallel interface is in the #2 card position in the 6800. The program PAPTAP is the loader that takes the parallel data in from the reader through the PIA and stores it in memory. This program is rather long and it is not convenient to load it in by hand each time. If you currently have some type of save/load device (AC-30 cassette, Teletype, etc.) you will have no problems in loading the loader program. If you have no AC-30 or teletype but can steal a few minutes time on someone else's paper tape punch, punch out the program QUICKLOAD using a binary format (no header characters, checksum, address pointers, etc. - just send the data to the punch using OUTEEE). Be sure to use only black paper tape. Do not use the BUNCH program in this newsletter.

If you are using an AC-30, etc. simply load in the PAPTAP program and set the program counter to 1F00. Place the paper tape in the reader and type G. Pull the paper tape through the reader and your program will be loaded. If a software interrupt is encountered an error was seen and the program should be reloaded.

If you are using the binary formatted tape, type in the QUICKLOAD program and insert the binary loader tape in the reader. Start execution at 0000 and pull the tape through the reader. This loads the PAPTAP program from locations 1F00-1F6E. The loading of your program can now be accomplished as described earlier.

Instructions come with the tape reader concerning assembly and use. If you have any questions concerning the mechanics, price, availability, etc. of the loader, please contact Oliver Audio, not SWTPC.

Note: You may have to re-write the loader programs to move them to a convenient area of memory for your computer. Also, the reader has a jumper that must be installed on it - Jumper A to ACK, not -ACK.

```
                NAM    PAPTAP
                *HIGH SPEED PAPER TAPE LOADER PROGRAM
                *DEVELOPED BY DR. CHARLES ADAMS
                *TEXAS A&M UNIVERSITY

E1D1            OUTEEE  EQU    $E1D1
8008            PIA     EQU    $8008

1F00                    ORG    $1F00
1F00 86 2E      ENTER   LDA A  #$2E
1F02 B7 80 0B           STA A  PIA+3
1F05 B7 80 0A           STA A  PIA+2
1F08 8D 28      OV      BSR    SUB1
1F0A 81 53              CMP A  #'S
1F0C 26 FA              BNE    OV
1F0E 8D 22              BSR    SUB1
1F10 81 31              CMP A  #$31
1F12 26 F4              BNE    OV
1F14 7F 1F 43           CLR    CLR1
1F17 8D 39              BSR    SUB2
1F19 80 02              SUB A  #2
1F1B B7 1F 42           STA A  TMP3
1F1E 8D 24              BSR    SUB3
1F20 8D 30      BR2     BSR    SUB2
1F22 7A 1F 42           DEC    TMP3
1F25 27 05              BEQ    BR1
1F27 A7 00              STA A  0,X
1F29 08                 INX
1F2A 20 F4              BRA    BR2

1F2C 7C 1F 43   BR1     INC    CLR1
1F2F 27 D7              BEQ    OV
1F31 3F                 SWI

1F32 B6 80 0B   SUB1    LDA A  PIA+3
1F35 2A FB              BPL    SUB1
1F37 B6 80 0A           LDA A  PIA+2
1F3A 84 7F              AND A  #$7F
1F3C B7 80 0A           STA A  PIA+2
1F3F 39                 RTS

1F40            TMP1    RMB    1
1F41            TMP2    RMB    1
1F42            TMP3    RMB    1
1F43            CLR1    RMB    1

1F44 8D 0C      SUB3    BSR    SUB2
1F46 B7 1F 40           STA A  TMP1
1F49 8D 07              BSR    SUB2
1F4B B7 1F 41           STA A  TMP2
1F4E FE 1F 40           LDX    TMP1
1F51 39                 RTS

1F52 8D 10      SUB2    BSR    SUB4
1F54 48                 ASL A
1F55 48                 ASL A
1F56 48                 ASL A
```

```
 1F57 48                ASL A
 1F58 16                TAB
 1F59 8D 09             BSR    SUB4
 1F5B 1B                ABA
 1F5C 16                TAB
 1F5D FB 1F 43          ADD B  CLR1
 1F60 F7 1F 43          STA B  CLR1
 1F63 39                RTS

 1F64 8D CC     SUB4    BSR    SUB1
 1F66 80 30             SUB A  #$30
 1F68 81 09             CMP A  #$09
 1F6A 2F 02             BLE    RT
 1F6C 80 07             SUB A  #7
 1F6E 39        RT      RTS

                        END

NO ERROR(S) DETECTED

   SYMBOL TABLE:
BR1    1F2C   BR2    1F20   CLR1   1F43   ENTER  1F00   OUTEEE E1D1
OV     1F08   PIA    8008   RT     1F6E   SUB1   1F32   SUB2   1F52
SUB3   1F44   SUB4   1F64   TMP1   1F40   TMP2   1F41   TMP3   1F42



                NAM    QUICKLOAD

 8008           PIA     EQU    $8008

 0000                   ORG    $0000

 0000 CE 1F 00          LDX    #$1F00
 0003 86 2E             LDA a  #$2E
 0005 B7 80 0B          STA A  PIA+3
 0008 B7 80 0A  HERE    STA A  PIA+2
 000B B6 80 0B  LOOP    LDA A  PIA+3
 000E 2A FB             BPL    LOOP
 0010 B6 80 0A          LDA A  PIA+2
 0013 A7 00             STA A  0,X
 0015 08                INX
 0016 7E 00 08          JMP    HERE
                        END

NO ERROR(S) DETECTED

   SYMBOL TABLE:
HERE   0008   LOOP   000B   PIA    8008
```