

# AUGUST '84

## I-M 1 IN A MILLION CLUB NATIONAL NEWSLETTER

### INSIDE...

BASE CONVERSION from L. DOOLEY.  
ESCAPE by ERIC BECKETT in REVIEW  
MICROPROCESSOR GLOSSARY OF TERMS  
A.I.T. HELPFUL HINTS CONTINUED.  
TAPE ASSEMBLER from HEXMART in  
REVIEW.  
DATA SHOP...Condition #3.  
TUNES FROM HAL BLOOM and RUSS  
NEEDHAM.  
DOUBLE ARCADE-BRANCH/JOYSTICKS



**GEO\*GRAFIX LIMITED**  
P O Box 54 • Arrowsmith, Illinois 61722

EDITOR-GEORGE BAKER  
PRODUCED MONTHLY BY-GEO\*GRAFIX LTD.  
PRINTED BY-GLENN VOSS PRINTING

BULK RATE  
U S. POSTAGE  
PAID  
Bloomington, IL.  
Permit No. 298

Send check or m/o to:  
The Program Peddler  
c/o John Mechales  
2830 Townway Rd.  
Danville, IL. 61832



Look! Up on the hill! It's a bus! It's a worm! No! It's...  
IPEED! CENTIPEED! CENTIPEED! CENTIPEED! CENTIPEED! CEN  
The popular ATARI arcade game is here for the APF!

Now \$9.95

#### LIGHT-TRACER

Manuever the "light tracer" within the screen boundaries.

Now \$6.95

#### THREE-FOR-ONE

- 1)SPACE WARS-You and an opponent battle in outer space!
- 2)AIR FORTRESS-Defend the fortress from the kamakazee aliens.
- 3)SQUEEZE PLAY-Get every block moving to escape the room!

All three now \$9.95

#### THE SEARCH



Explore the land in this text adventure game. Find the magic sword and boat ( if you can ! ) , then travel through time to defeat the villain ( the planned sexual. ) The path is not easy! You must face goblins, orcs, dragons, Mother Nature, and other nasties. It's an adventure that uses two-word commands which you musn't pass up!

Only \$9.95

\*\*\*\*\*PROGRAMMED BY JAMES CHAMINGS\*\*\*\*\*

#### STAR-FIRE

Hunt down and destroy the enemy in the differant sectors of the galaxy. All commands are three letters. Special features are found in this game. E.G.-if your scanners are damaged, you don't get a correct view of the sector. If your lasers are damaged, you lose half of your firepower. If your engines are destroyed, you are stranded in space! Specify special version for those with ROMMON chips.

Only \$9.95

A comment from G.R. Jones for CENTIPEED! is:

"Congratulations on a very good game!"

Hey all you out there in COMPUTER LAND!. Have you got a program/program idea that you think would sell? Send it to me! You will be given credit for the idea/program and PAID if I use it. It will be advertised and distributed through me. Royalty pay will be dependant on originality, suffix, etc... All programs/ideas will be subject to some chance.

# GENERAL NEWS

## SPECIAL THANKS!

To BILL BOWMAN, LOUIS DOOLEY, and HAL BLOOM for supplying the club with programs that appear in this issue and in the CLUB PROGRAM LIBRARY.

Listed below is a brief explanation of each program submitted.

BILL BOWMAN-A program that will compare the cost of fuel--OIL vs GAS vs ELECTRIC. Club program designation T6FUELCOMP.

HAL BLOOM-A HARD COPY ONLY of a musical program that contains 14 different tunes that is set up to pull each tune from disk and RUN it. This program is a LIBRARY SELECTION but IS NOT on tape or disk. Eleven pages will be reproduced and sent to you, and counted as ONE selection. The minimum library order is 3, so your other two selections will be sent on cassette. Library designation HCDISKMUS

LOUIS DOOLEY-"APF IM-1 TYPEWRITER Jr." This program allows the user to type data, messages, records, etc. to the screen and store many screens to tape. Reviewing each screen is done by simply tapping the space bar. The menu explains the functions of the program. 8K. Designation T6TYJR.

BILL BOWMAN-A program that will indicate HOW LONG a fixed sum of money, earning a fixed interest rate, would last at a specified monthly withdrawal. Designation T6FIXWDL.

-----

## CORRECTION TIME!

An error (typo) that appeared in the JULY ARCADE is a good example of what happens when EXTREME CARE is not taken when LISTING or ENTERING ML DATA. In this case, I mistyped the very last byte in the short program at address A425. The data entered should have been 0A rather than 04. For those of you who did not catch it in the written instructions and entered the data as it appeared, found your WHITE SQUARE to be QUITE INACTIVE! The intent of the program was to display movement, with delay, from the top of the screen to the bottom.

## MARCH-HELPFUL HINTS-WIRING FOR THE OLDER TYPE GEMINI 10 PRINTERS.

The lead designation should have been PIN 8 to PIN 8 rather than PIN 1 to 8.

## CLARIFICATION

### JULY ISSUE-PROGRAM "PAIRED ITEMS"

The article was weighted heavily with technical information about the operation of the program itself but should have included a brief description of what the program actually IS!

It could be considered a game, more specifically "A MEMORY GAME". The player enters a chosen number of PAIRS that are related to each other such as, STATES/CAPITOLS, POUNDS/KILOGRAMS, DOLLARS/ENGLISH POUNDS, etc. After all pairs are entered, the program will give you the first word of the pair and see if you can match it with its associated word that you entered initially. This can be recalled on a RANDOM or ORDERED basis. The computer will keep track of the % of correct entries. There are many variations that can be achieved and more than one player can be included.

## CLARIFICATION

### JUNE ISSUE-DATA SHOP

As explained, the cable used with the VOLKSMODEM requires wiring changes. This is true only if an 'A' cable is ordered. We have just learned that if a 'B' cable is ordered with the modem, NO WIRING CHANGE IS NECESSARY. Be sure to specify the 'B' cable when ordering this modem.

-----

## ARE YOU MISSING AN ISSUE?

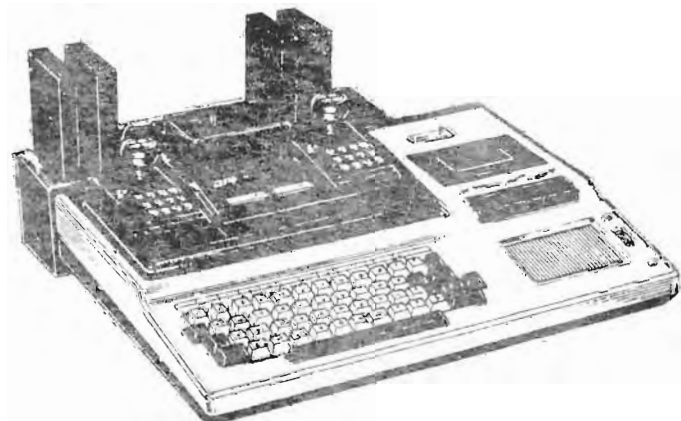
If so, please let us know which month and we'll send one out to you promptly!

-----

## MAY Q & A TOSS UP..NO RESPONSE TO DATE!!???

Please..take another look at the MAY issue Q & A TOSSUP section. If you could respond to ANY of the questions listed therein, we would ALL sincerely appreciate it!

-----



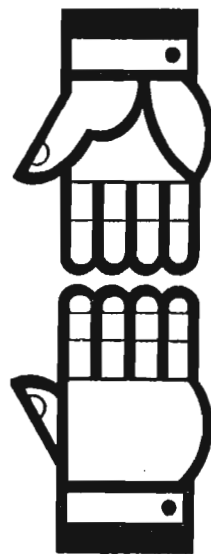
# HELPFUL HINTS

This first *HELPFUL HINT* comes to us from *FRED VANOEVEREN*, Grand Rapids, MI. Fred writes: "I sent along a short program which can be used to convert a *NUMBER* to a *STRING* and back again. Since I keep all my bowling and golf league data as numbers, I have to use routines of this type to get strings for sorting. By concatenating the number strings with *NAMES*, *DISK RECORD POINTERS*, etc., I have been able to produce *ORDERED LISTS* for almost any type of data. Perhaps this routine would be of interest to others."

```
10 REM ....CONVERT A THREE DIGIT OR LESS NUMBER TO A STRING
15 DIM A$(2): REM ....DIMENSION A THREE CHARACTER STRING
20 INPUT "INPUT ANY NUMBER (0-999) : ", N
25 N1= INT (N/100): REM ....GET HUNDREDS UNIT
30 A$(0)= CHR$(48+N1): REM ....CONVERT TO STRING AND PLACE IN VARIABLE
35 N2= INT ((N-N1*100)/10): REM ....GET TENS UNIT
40 A$(1)= CHR$(48+N2): REM ....TO STRING AND INTO VARIABLE
50 N3=N-N1*100-N2*10: REM ....GET ONES UNIT
55 A$(2)= CHR$(48+N3): REM ....TO STRING AND INTO VARIABLE
60 PRINT "NUMBER ";N;" IS NOW STRING: ";A$
100 REM ....THIS ROUTINE WILL CONVERT THE STRING BACK TO A NUMBER
110 N4=( ASC (A$(0))-48)*100
115 N5=( ASC (A$(1))-48)*10
120 N6= ASC (A$(2))-48
125 N7=N4+N5+N6
130 PRINT "... AND AGAIN THE NUMBER: ";N7
```

The next *HELPFUL HINT* was supplied by *BILL BOWMAN*, So. Yarmouth, MA. The program is an example of selecting items from a menu without using an *INPUT* prompt and *RETURN KEY*. Line 70 is the key for this function which is also explained in the *APF BASIC LANGUAGE REFERENCE MANUAL*. Thank you Bill for sending the program in.

```
5 GOSUB 4010
6 PRINT 'PROGRAM TO SETUP "MENU SELECT": PRINT "WITHOUT USING INPUT & RETURN KEY"
7 PRINT : PRINT "LIST TO SEE HOW IT'S DONE AND": PRINT "RUN TO SEE EXAMPLE WORK"
8 PRINT : PRINT "EXAMPLE FOLLOWS:-"
10 DIM NA$(8),NULL$(8)
20 POKE 26112,0: POKE 8193,60
40 PRINT "1-": PRINT "2-": PRINT "3-"
50 PRINT : PRINT "SELECT (1-3)"
60 NA$=NULL$
70 NA$= KEY$ (0)
75 IF NA$=NULL$ THEN 60
76 POKE 909, ASC (NA$)+64: MUSIC "700*700"
80 IF NA$="1" THEN 1000
90 IF NA$="2" THEN 2000
100 IF NA$="3" THEN 3000
200 IF NA$<>"1" IF NA$<>"2" IF NA$<>"3" THEN PRINT "WRONG KEY": GOTO 4000
1000 PRINT : PRINT "# 1 EXECUTED"
1005 GOTO 4000
2000 PRINT : PRINT "#2 EXECUTED"
2005 GOTO 4000
3000 PRINT : PRINT "# 3 EXECUTED"
3001 PRINT : PRINT "YOU SEE..IT'S EASY"
3005 GOTO 4000
4000 FOR I=1 TO 300: NEXT
```

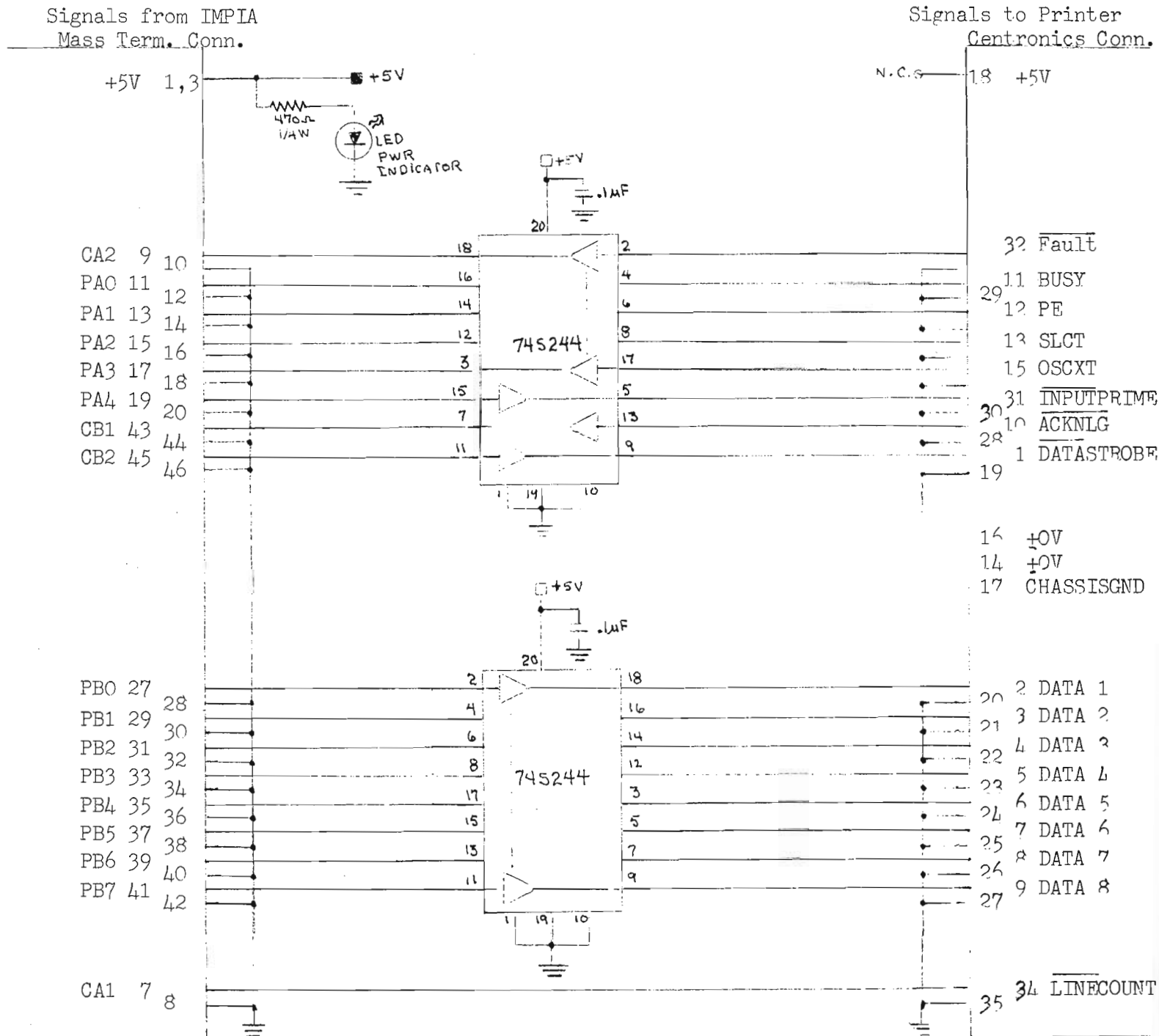


```
4005 GOTO 5
4010 CALL 17046: POKE 40960,2: POKE 40961,0
4020 RETURN
```

The next two pages contain the remainder of the *HARDWARE HINTS* on *CENTRONICS* interfacing, supplied by *DAVID PRESCOTT* of *A.I.T.* Thank you David!

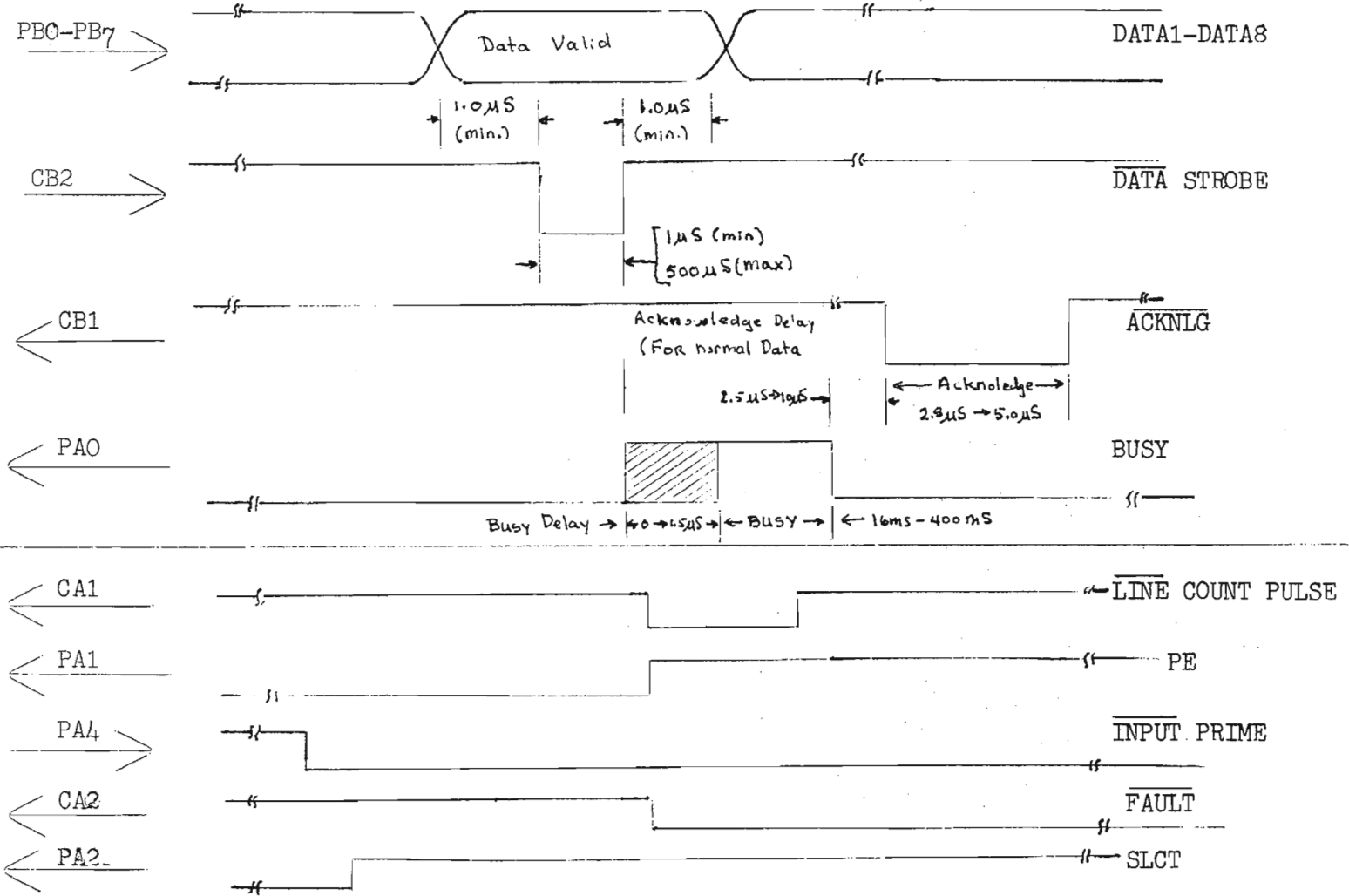
HARDWARE - How to's : Continuation of Interfacing the Centronics parallel printer to the IM-1  
by the Advanced Interfacing Team

Schematic Diagrams;



The LED is used to indicate that the buffer circuit is getting power from the IM-1 through the IMPIA card. The two 74S244 I.C.'s are used to buffer the higher currents required by the printer interface and thereby protect the parallel port inputs and outputs. Although the tri-state of this chip is defeated by grounding the enables, it was chosen because of its ease in straight through wiring and because it has eight drivers per chip saving space. The 0.1 uF caps provide glitch protection for the switching gates. Wire all the grounds together in daisy chain style so that there is a good signal ground between the IMPIA and the printer. The +5 volt supply from the printer is not connected to anything. This is to prevent supply voltage conflicts between the IM-1 and the printer. The I.C.'s supply pin 20 is connected to the five volt supply brought out by the IMPIA edge connector.

Signal timing conditions for programming the parallel port.



Set up the parallel port in the following manner. Data register B side all outputs. Control register B-side in data handshake strobe mode, CB2 strobe low, CB1 accept low pulse from ACKNLG. Because of the speed of the IM-1 clock the proper data set-up and strobe pulse durations should fall right within specifications. The acknowledge pulse from the printer will set the IRQB1 bit in the B-side status register, so you know when to transmit the next character. Set up the A-side data direction register as PA0 - PA3 as inputs and PA4 as an output. PA4 will be used to output a low pulse INPUT Prime to clear and activate the printer. PA0-PA3 will accept the BUSY, PE, SLCT, and OSCLT, if you should desire to use them in software control. Set the A-side Control register with CA1 and CA2 as inputs with active low edge trigger on both. CA1 will then set IRQA1 every time the printer does a line feed so that you can keep track of what line you are on, on your paper. CA2 will set The IRQA2 bit in the status reg. A should a fault occur on the printer.

Because the IM-1 does not have softwares to control a parallel printer interface it will be up to you to right the routine. It is most efficiently done in machine code, however it could be done in basic as well. Good luck and have fun.

All information in this article was addressed to the specifications supplied by Centronics Printer Manual Appendix B- Parallel Interface Specifications, to comply with the Centronics Interface characteristics, and matched to the IMPIA parallel port. A.I.T. will not be held responsible for any errors witch may occur herein nor any damage to either units caused by user wiring errors or abuse. ■

# ORCHESTRA PIT

```
10 :DATACOMP COMPUTER SOFTWARE
20 :36723 STANTON POINT RD.
30 :INGLESIDE, IL 60041
40 POKE 24578,38
50 CALL 17046
55 POKE 40960,2: PRINT "      HAPPY DAZE"
60 SHAPE =9: FOR K=1 TO 25: COLOR =K
70 HLIN 0,31,0: VLIN 0,15,0: HLIN 0,31,15: VLIN 0,15,31
80 FOR L=1 TO 50: NEXT : NEXT
100 MUSIC "100 400 600 *100 *2 *300 *20000"
200 MUSIC " 200 500 +600 *200 *3 *400 *300000"
210 MUSIC " 300 600 ++100 *300 *4 *500 *400000"
220 MUSIC " *2 *200 *300000 *2 *200 *300000"
230 MUSIC " *3 *3 *300 *400 **4 *5000000000000"
235 FOR I=1 TO 3
240 MUSIC " *400 *400 *50 *60000 *500 *400 *50 *40000"
250 MUSIC " *200 *200 *2 *300000000000 "
255 NEXT
260 MUSIC " 4 400 44 4000 6 600 5004000 4"
270 MUSIC " +6 +6 +6 +60 +600 +6000 *200 *2 *100+60000"
280 MUSIC " 500 5 500 50000 700 7 60050000"
290 MUSIC " *100 *1 *100 *5000 *300 *1000"
300 MUSIC " *400 *400 *5 *6000 *500 *400 *5 *4000"
310 MUSIC " *200 *200 *2 *3000 *300 *400 *5 *400000"
320 MUSIC "*2000 *300 *400"
```

```
10 FOR I=1 TO 2
20 MUSIC "2000+2000500040000000 50006000+600050010000000 1000/700010005000000000"
30 MUSIC " "
40 MUSIC "+600060000040020000000 2000+20003000400+200020+2040000000000"
50 MUSIC " "
60 MUSIC "500060050004005006000000000 6000+500000600+600000000000"
70 MUSIC " "
80 MUSIC "*100060000010040000000 500400+2000 200+2000500040000000"
90 MUSIC " "
100 MUSIC "5000600000+60050010000000000 1000/7000100060000000000 6000+6000*1000*2000000000"
110 MUSIC " "
120 MUSIC "+2000*+100000*200*+20000000 *200070000000 *1000*2000*10005000000000"
130 MUSIC " "
140 MUSIC "5000+40005000*+2000000000 5000*200000 *20000000+600000000000"
150 MUSIC " "
160 IF I=1 THEN NEXT I
170 IF I=2 STOP
```

The upper program was supplied by RUSS NEEDHAM of DATACOMP SOFTWARE and is the theme from HAPPY DAYS. The lower program is a sample of the many tunes submitted by HAL BLOOM. You'll recognize the tune immediately!

# SHORT PROGRAMS

At last, a base conversion program that allows you to add, subtract, multiply, and divide in decimal, hexadecimal, octal, or split octal and have the ability to retrieve the answer in ANY of these bases that you desire! OUR THANKS TO LOUIS DOOLEY FOR THIS PROGRAM.

## LEGAL EXPRESSIONS:

"N" = any number between 0 and 65535

"+" = math operator: + - \* /

"C" = number base code:

T = DECIMAL (DEC) (=default code) the program defaults to DECIMAL

H = HEXADECIMAL (HEX)

Q = OCTAL (OCT)

S = SPLIT OCTAL (OCS)

## FORMATS

N+N=  
+N=  
N=C  
CN=C  
C  
N+N=C  
CN+CN=C  
CN+N=  
CN+N=C  
N+CN=  
N=CN=C  
+N=C  
+CN=  
+CN=C

## EXAMPLE

197+381=  
+476=  
1981=H  
H100=T  
Q  
4095+3680=H  
Q377+S17 377=H  
HFF+D2  
T3265+597=H  
4D2+T511=  
FA+Q37=T  
+59=S  
+S57 377=  
+77H

(SPACE PERMITTED ONLY IN SPLIT-OCTAL NUMBER)

("OUT OF RANGE" error print includes input or result greater than 65535 and improper characters) Note: Impractical to check for ALL possible errors in input; user must still be careful and prudent.

## SAMPLE TEST DATA:

DEC	HEX	OCT	OCS
1234	04D2	2322	4322
4095	FFF	7777	17 377
3184	C70	6160	14 160
3568	DF0	6760	15 360
3947	F6B	7553	17 153
65535	FFFF	177777	377 377

## 'BASIC' PROGRAM ROUTINES

LINE	FUNCTION	EXPRESSION
1000	HEX TO DEC	(H\$ to T)
2000	DEC TO HEX	(T to H\$)
3000	OCT TO DEC	(H\$ to T)
4000	DEC TO OCT	(T to H\$)
5000	STRING TO NUMBER	(H\$ to T)
6000	NUMBER TO STRING	(T to H\$)
300+	CALCULATION	(T and V)

```

10 CALL 17046: POKE 40960,2: POKE 40961,0: POKE 24578,38
15 PRINT "NO. BASE CONV/CALC (L. DOOLEY)"
16 PRINT " (FROM HEATH 89)": PRINT
17 PRINT "NO.-BASE CODES(C):"
18 PRINT TAB (3);"T(10), H(16), Q(8), S(8+8)"
19 PRINT "EXPRESSION FORMATS:"
20 PRINT TAB (10);"N= (CONV,)"
21 PRINT TAB (10);"N+N (CONV./CALC.)"
22 PRINT TAB (10);"N (CHAINING)": PRINT TAB (10);"C (CONV. CHAIN.)"
23 PRINT TAB (3);"(N = NO. BETWEEN 0 & 65535)"
24 PRINT TAB (3);"(+ = +, -, *, OR /)"
25 PRINT "N MAY BE PRECEDED BY BASE CODE"
26 PRINT "= MAY BE FOLLOWED BY BASE CODE"
27 PRINT "DEFAULT = T OR LAST-USED CODE"

```

```

30 FOR D=1 TO 5000: IF KEY$(0)=" " THEN 35
31 NEXT
35 DIM X$(1),B$(2),E$(14),H$(6),N$(6),S$(14),Z(4)
36 CALL 17046: POKE 40960,2: POKE 40961,0
40 B=:B$="DEC"
50 M=0: PRINT B$
51 E$:S$:C=-1
60 INPUT "EXPRESSION ",E$
70 FOR A=0 TO LEN (E$)-1
80 X$(0)=E$(A)
90 IF X$(0)="T" THEN B=:B$="DEC": GOTO 180
100 IF X$(0)="H" THEN B=:B$="HEX": GOTO 180
110 IF X$(0)="Q" THEN B=:B$="OCT": GOTO 180
120 IF X$(0)="S" THEN B=:B$="OCS": GOTO 180

```

# SHORT PROGRAMS

```
130 IF X$(0)="+" THEN M=1: GOTO 400
140 IF X$(0)="-" THEN M=2: GOTO 400
150 IF X$(0)="*" THEN M=3: GOTO 400
160 IF X$(0)="/" THEN M=4: GOTO 400
161 IF B=4 THEN IF X$(0)=" " THEN 168
162 IF X$(0)="=" THEN V=T: GOTO 400
163 IF X$(0)="!" THEN PRINT "DONE!!": END
164 IF X$(0)<"0" THEN 500
165 IF X$(0)>"F" THEN 500
166 IF X$(0)>"9" THEN IF X$(0)<"A" THEN 500
168 C=C+1
170 H$(C)=X$(0)
180 NEXT
185 IF M=0 THEN 191
190 ON M GOSUB 300,310,320,330
191 IF T>65535 THEN 500
192 S=T
200 ON B GOSUB 6000,2000,4000,4000
201 IF E=1 THEN 500
210 PRINT TAB (Z1);H$; " ";B$
220 T=S:H$=N$: GOTO 50
300 T=V+T: RETURN
310 T=V-T: RETURN
320 T=V*T: RETURN
330 T=V/T: RETURN
400 IF LEN (H$)>0 THEN ON B GOSUB 5000,1000,3000,3000
401 IF E=1 THEN 500
410 H$=N$:C=-1: GOTO 180
500 PRINT "OUT OF RANGE: ";E$:E=0: GOTO 50
1000 T=0:E=0:X=1
1010 FOR K= LEN (H$)-1 TO 0 STEP -1
1020 H= ASC (H$(K))-48: IF H>9 THEN H=H-7
1030 IF H<0 THEN E=1: RETURN
1031 IF H>15 THEN E=1: RETURN
1040 T=T+(H*X): IF T>65535 THEN E=1: RETURN
1050 X=X*16
1060 NEXT
1070 RETURN
2000 H$=N$:E=0:D=0:X=4096
2005 IF T<0 THEN E=1: RETURN
2006 IF T>65535 THEN E=1: RETURN
2010 H= INT (T/X):T=T-(H*X)
2020 H=H+48: IF H>57 THEN H=H+7
2030 H$(D)= CHR$ (H)
2040 X=X/16: IF X<1 THEN RETURN
2050 D=D+1: GOTO 2010
3000 T=0:E=0:X=1
3010 FOR K= LEN (H$)-1 TO 0 STEP -1
3020 H= ASC (H$(K))-48
3025 IF H=-16 THEN X=256: GOTO 3060
3030 IF H<0 THEN E=1: RETURN
3031 IF H>7 THEN E=1: RETURN
3040 T=T+(H*X): IF T>65535 THEN E=1: RETURN
3050 X=X*8
3060 NEXT
3070 RETURN
```

```
4000 H$=N$:E=0:D=0:X=32768
4005 IF T<0 THEN E=1: RETURN
4006 IF T>65535 THEN E=1: RETURN
4008 IF B=4 THEN X=16384
4010 H= INT (T/X):T=T-(H*X)
4020 H=H+48
4030 H$(D)= CHR$ (H)
4040 X=X/8: IF X<1 THEN RETURN
4045 IF B=4 THEN IF X=32 THEN H$(D+1)=" ":X=64:D=D+1
4050 D=D+1: GOTO 4010
5000 DIM Z(4):L=0:E=0
5030 FOR K= LEN (H$)-1 TO 0 STEP -1
5040 Z(K)= ASC (H$(L))-48
5050 L=L+1: NEXT
5070 T=10000*Z(4)+1000*Z(3)+100*Z(2)+10*Z(1)+Z(0): IF T>65535 THEN E=1
5075 IF T>65535 THEN E=1
5080 RETURN
6000 DIM Z(4):L=0:E=0
6010 IF T>9999 THEN K=4:X=10000: GOTO 6020
6011 IF T>999 THEN K=3:X=1000: GOTO 6020
6012 IF T>99 THEN K=2:X=100: GOTO 6020
6013 IF T>9 THEN K=1:X=10: GOTO 6020
6014 IF T<10 THEN K=0:Z(K)=T: GOTO 6040
6020 Z(K)= INT (T/X):T=T-Z(K)*X
6030 IF X>10 THEN X=X/10
6040 H$(L)= CHR$ (Z(K)+48)
6050 L=L+1:K=K-1: IF K<0 THEN RETURN
6060 GOTO 6014
```

## CLASSIFIED

\*\*\*\*\*WORKING COMPUTERS\*\*\*\*\*

IM-2 with B & W monitor, joysticks,  
schematics, and 11K RAM (exp. to 27K  
with YOUR BB-1 & RAM) w/o BASIC....\$99.  
16K int IM-1 w/o BASIC & AC adptrs..\$89  
Only one BASIC cart for IM-1/IM-2..\$35.

G.R. JONES

419 S. 105 E. Place  
Tulsa, OK 74128

# PRODUCT REVIEW



## ESCAPE by ERIC BECKETT

This fast action HI RES game has been around for quite awhile but there have been a few inquiries about it lately. The game begins by placing you in a maze with no exits. You maneuver a HI RES character around the playing field and try to destroy FOUR security power banks by running over them with your man.

Roving guards occupy the maze with you and will chase and fire bullets at you. You can fire back, and each guard that is destroyed is worth 10 points. If a guard runs over a spot where another guard was killed, a new guard is re-born and BOTH will chase you.

Guards in bunches are almost indestructible and your bullet will not usually penetrate them.

The walls that make up the maze are HIGHLY EXPLOSIVE. You must avoid running into them. You may not fire more than one bullet at a time. Your bullet must hit something and explode before you are able to fire again.

This game has several other distinct features that make it very interesting to play. It's a game that requires quick responses along with a VERY WATCHFUL EYE. The game comes with an instruction sheet and will work on both color and black and white sets.

A few more features to mention are:

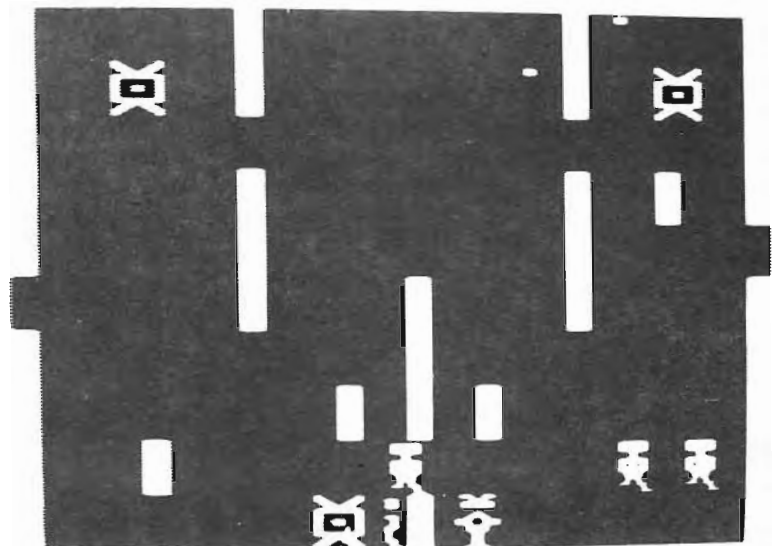
If a dead bullet is blocking your way, DON'T run into it or you may BLOW UP! First shoot the dead bullet then go on through it. After firing, if you run after your own bullet, it will have a better chance of penetrating a guards bullet and destroy him. Try to remember where you kill guards and try to keep the live ones away from that area. It's easy to trick the guards into running into your bullet.

When a guard moves next to you, you must run up or down to escape. If you move right or left, you will most likely get zapped by him!

Each security power bank that is destroyed is worth 100 points. Guards destroyed as mentioned are worth 10 points, and each NEW LEVEL that is achieved by eliminating guards in the preceeding level is worth an additional 1000 points. The game is written in MACHINE LANGUAGE for the play and BASIC for scoring. Some unique sounds are generated with the ML routines.

*Note: The FRONT SCREEN (top left) indicates scoring provided in the game. Photo (bottom right) depicts the maze, power banks (objects that look like TV sets), the guards, YOUR MAN, and a few bullets (upper right).*

**NOT RECOMMENDED FOR THOSE WITH POOR PERIPHERAL VISION!!!!**



# PRODUCT REVIEW

## TAPE ASSEMBLER from HEXMART

If you have been following the ARCADE section over the past months and have been looking up each OP code to find its meaning, you have probably found the task quite burdensome and time consuming. Eric mentioned an ASSEMBLER in the beginning of his instruction but decided to introduce you to the INSTRUCTION SET and use IT for a better understanding of MACHINE LANGUAGE PROGRAMMING. This is a proven method of instruction.

The HEXMART TAPE ASSEMBLER allows a person who is somewhat familiar with machine language to write 6800 assembly language programs without having to spend HOURS looking through the instruction set tables for the right OP codes to use, or counting in HEX to figure out the branching instructions.

The TAPE ASSEMBLER is a LINE ASSEMBLER in that it assembles each line of 6800 assembly code as it is typed in. As an example, if you need to LOAD ACCUMULATOR A with a value, the HEXMART program will look up the OP code for you in its memory and enter this data at a particular address for you automatically. You would simply be required to type in LDAA with the value you want loaded, indicating the addressing mode desired. This makes it a very fast method of writing machine language programs because the end result is actually in MACHINE LANGUAGE.

You might consider this assembler to be INTERPRETIVE, that is, it takes an input such as LDAA IM and interprets the correct HEX value that corresponds to the instruction. This value is then placed in consecutive addresses almost immediately thus causing the entire process to move swiftly and accurately. An important point to remember is that success with this assembler, as with any other, is based primarily upon the USERS KNOWLEDGE of machine instruction and experience with it. The program is written in BASIC with a machine language routine that searches through the entire built in instruction set table to find the correct OP code to use. The average time for the program to assemble a line of code is three to four seconds.

The TAPE ASSEMBLER is particularly useful for assembling relatively small programs but can also be used for larger ones since

it has capacity for 65 labels and 45 deferred operands in 8K.

The program comes with 27 pages of instruction that include an introduction to assembly language programming, how to use the assembler, work sheets, and information taken from the APF TECHNICAL REFERENCE MANUAL for further help in understanding assembly language programming. If you are serious about ML or assembly programming, the TAPE ASSEMBLER will take a lot of the chore out of the work. It is a common tool for virtually all programmers who work with machine language routines and programs.

Note: If you are just starting out in this area, you still need a good idea of how OP codes are used and what functions they perform. This is the intent of the ARCADE section. STAY WITH IT. In order to clearly understand Machine functions, the ARCADE should prove to be very helpful. After you have gotten a good grasp of programming directly, try the assembler and find out how much EASIER it makes the ENTIRE JOB!

## MEMO PADS



Minimum Order 50 Pads  
5x8 100 Sheets To Pad. Photos And  
Logos No Extra Charge.

order From **ARROWSMITH**  
**Glenn's Printing** IL 61722

# THE ARCADE

*MORE ARCADE DATA HAS ARRIVED! The MICRON wheel is ready to go in order to get as much in this issue as possible. Eric begins this issue with BRANCHING techniques.*

If the program appearing in last month's issue was entered correctly you should see a square move smoothly across the screen from left to right and gradually from top to bottom. If not, check your inputs carefully and try again.

## BRANCHING

The machine language branch statements are similar to BASIC IF THEN GO TO. You are checking for a certain condition (depending upon which branch statement is used) and then branching (or jumping) if that condition is met. If it isn't, the program continues on.

Branch statements can be used after any instruction but usually after a compare instruction or a subtract command. It can also be used after an INCREMENT or DECREMENT command as in our previous program example. Branching is best used when you want to go to a relatively close location. The limits are around 128 bytes in either direction from where the program currently is at the time. If you're using a good ML ASSEMBLER your branches are automatically calculated. When using the APF monitor as we are in this instruction, you will have to count how far away you want to go by counting for yourself. If you branch a lot, far away from your current address, a lot of your time will be spent counting bytes. I suggest using the following trick when you want to go to a far location on a certain condition. Let's assume that you want to branch if a certain number is greater than zero. It would look like this in machine code:

```
81 COMPARE ACC. A IMMEDIATE
XX DATA TO COMPARE ACC. A to
2E BRANCH IF (ACC. A-XX) 0
55 WHERE TO BRANCH TO
XX PROGRAM CONTINUES
XX
```

In the example you will be branching 85 bytes FORWARD in the program from where you are. You can change the branch if greater than ( ) 0 to branch if less than ( ) 0. In this way if it is greater than 0 you SKIP and go to a JUMP instead. Look at the following example:

```
81 COMPARE ACC. A IMMEDIATE
XX
2D BRANCH IF LESS THAN 0
03 WHERE TO BRANCH TO
7E JUMP
XX TO THE LOCATION
XX 85 BYTES AWAY
```

This second example will save you time counting all those bytes!

We can branch forward and backward. Branch instructions are TWO bytes long, the second byte is called the OFFSET byte. This byte determines how far away we will branch, forward or backward. To branch forward just count the number of bytes to where you want to go. You DO NOT count the offset byte and the byte after it is number 00. Then comes 01, 02,... etc. You can see how we did this in our example with the moving square. At A420 we wanted to branch over the return code of 39, so we used an offset value of 01. We can only branch forward to around 7F because to branch backwards we start at the offset of FF and count backwards FF, FE, FD, FC,... We did this in our examples at step A416. We wanted to branch backwards to A410 or back 7 bytes, so we used an offset of F8. For convenience, the BRANCH CHART included in this issue should be helpful to you to figure your branching values.

Try experimenting with the moving square example. By changing the delay variable, you can get a real good feel of how fast ML is. There are two variables in accumulator A that are most effective. Start by changing the value at A40F from FF to 7F. This is about HALF THE DELAY from the original version. Do this by using the ML monitor command \*M A40F FF 7F. Now, run the program. Notice the speed change from the original. Try changing A40F to 3F. Then try 01. You will get to a point where it goes so fast across the screen that it no longer looks like the square is moving. It will appear to jump from one place to another.

## JOYSTICKS

If you have ever written any games in BASIC, you know one of the most time-consuming things to do is to READ the joysticks during the program operation. This problem is easily overcome using machine language. To make things easier, APF has included a routine in the ROM that reads the joysticks and places the results at an address. To use these routines we simply JUMP to them using a JUMP TO SUBROUTINE command. When the subroutine is done, it will automatically return to wherever it left off in the program.

To read the LEFT joystick, you would jump to the subroutine at 41BE. Your computer now looks at the joystick. If anything is pressed, a BIT in the CONDITION CODE REGISTER called the CARRY BIT will be SET. This will be explained later. It also READS what was pressed on the joystick and stores the ASCII CODE for this value at address 01F2. Our program would then look at 01F2 to see what was pressed.

# THE ARCADE

This is how we would use this subroutine:

First, clear the carry bit using the OP code 0C which is CLEAR CARRY. This is done to reset the joystick in case it was already moved once. We then jump to the joystick ROM routine as mentioned. When we come back from the ROM routine we use the OP code branch if the carry bit is set, or (25 in HEX). If we used the joystick, we would branch to another routine that reads location 01F2 to see which way the joystick was moved. If we wanted to read the RIGHT joystick, we would do the same thing, only JUMP to a different ROM routine which is located at 41D9. The result would still be found at location 01F2. The result at location 01F2 is the ASCII equivalent. The values that can appear at location 01F2 will be as follows:

NORTH = 4E  
 SOUTH = 53  
 EAST = 45  
 WEST = 57  
 FIRE = 21

We can also use a ROM routine to get a key input from the MAIN KEYBOARD. It will not return a shifted keyword such as the control key and top 2 rows. To use it simply jump to the subroutine at 80CF. When the program returns from the subroutine, the ASCII code for the key pressed will be in ACCUMULATOR A. If ACCUMULATOR A equals 0, then NO KEYS WERE PRESSED.

Next on Eric's agenda is a 150 byte machine language program that moves a white square using the joystick with the ROM routines. There is not enough room in this issue to accomodate the data, so, a small portion will appear herein and the remainder in next months issue.

For this month, enter only the BASIC program that appears below and CSAVE this information to tape so that it may be retrieved easily for next month's instruction.

The first 3 BASIC lines contain 'A' characters after the REM statements. Enter the amount of 'A's shown AFTER the REM.

```
10REM (total of 51 letter A's)
15REM (total of 51 letter A's)
20REM (total of 50 letter A's)
25POKE 24578,38
30CALL17046
40CALL42002
45CALL42012
50GOTO45
```

	F	E	D	C	B	A	9	8	
-	16	32	48	64	80	96	112	0	
F	17	33	49	65	81	97	113	1	
E	2	18	34	50	66	82	98	114	2
D	3	19	35	51	67	83	99	115	3
C	4	20	36	52	68	84	100	116	4
B	5	21	37	53	69	85	101	117	5
A	6	22	38	54	70	86	102	118	6
9	7	23	39	55	71	87	103	119	7
8	8	24	40	56	72	88	104	120	8
7	9	25	41	57	73	89	105	121	9
6	10	26	42	58	74	90	106	122	A
5	11	27	43	59	75	91	107	123	B
4	12	28	44	60	76	92	108	124	C
3	13	29	45	61	77	93	109	125	D
2	14	30	46	62	78	94	110	126	E
1	15	31	47	63	79	95	111	127	F
0	16	32	48	64	80	96	112		
	0	1	2	3	4	5	6	7	

## BRANCH CHART

USE TOP AND LEFT FOR BACKWARD BRANCH  
 USE BOTTOM AND RIGHT FOR FORWARD BRANCH  
 COUNT THE NUMBER OF BYTES TO THE  
 TARGET INSTRUCTION..THEN  
 GO TO THE CHART

### EXAMPLE OF BACKWARD BRANCH:

```
9 8 7 6 5 4 3 2 1
86 23 A7 00 8C 03 00 26 F7
# of BYTES = 9 TOP & LEFT = F7
```

### EXAMPLE OF FORWARD BRANCH:

(Note where count begins)

```
86 33 A7 00 A7 01 8C 03 00 27 06
  | 1 | 2 | 3 | 4 | 5 | 6 |
86 | 22 | A7 | 04 | A7 | 08 | 39 |
# of BYTES = 6 BOTTOM & RIGHT = 06
```

EXAMPLES: FORWARD 122 = 7A  
 BACKWARD 111 = 91  
 FORWARD 33 = 21

# DATA SHOP

Briefly, let's examine the last form of IM-1 to IM-1 communication..NEITHER IM-1 RUNNING A TERMINAL EMULATOR PROGRAM.

After the two computers are connected via modems, the transmitting end configures his IM-1 to PRINT TO THE LINE by typing in the statement PRINT=1. The receiving end relinquishes control of his IM-1 by conditioning it in the PRINT=2 mode. At this point, the transmitting end has full control over the receiving end IM-1.

During our experimentation with this form of communication we found that normal PRINT and CONTROL statements can be sent to the distant end in the IMMEDIATE MODE ONLY, that is, the statements must be typed by the transmitting end without any program operation or influence. For example, to print the word HELLO, the transmitting end (in the PRINT=1 condition as mentioned previously) would merely type PRINT"HELLO" with a carriage return. The distant end would receive the word HELLO on his screen, but, because the word is INVALID by itself, his IM-1 will respond with the word WHAT. This is of no consequence when single sentence conversation is desired in this mode. Transmitting VALID statements as PROGRAM DATA requires a little bit more work and will be explained in detail.

Virtually ALL BASIC commands can be sent to the receiving end as long as they are included within a PRINT STATEMENT. Commands such as CSAVE, CLOAD, POKE, HLIN, VLIN, can also be transmitted to save information stored in the receiving ends memory, or to produce graphics on his screen. For example, to draw a horizontal line across the receiving end IM-1, the transmitting end would type PRINT"HLIN1,30,1". Only the command HLIN1,30,1 would be sent out to the receiving end. His IM-1 would interpret this command as a VALID statement and would produce a line across his screen providing the SHAPE and COLOR was established previously. To the receiving IM-1, this command would seem to have been from the keyboard directly, however, it produces the same results when coming in from the line. It seems a bit EERIE to have someone else CONTROLLING YOUR computer!

If a musical score is to be sent for IMMEDIATE LISTENING, the transmitting IM-1 would include the music within a special print statement such as PRINT'MUSIC"123"' Notice the use of an apostrophe at the beginning and end of the above statement. This allows the needed QUOTATION MARKS to accompany the MUSIC statement so that the receiving IM-1 accepts the command MUSIC"123". Any apostrophe will be eliminated from the transmitted data. This arrangement should also be used when sending line by line program data whenever QUOTES are needed. See example below.

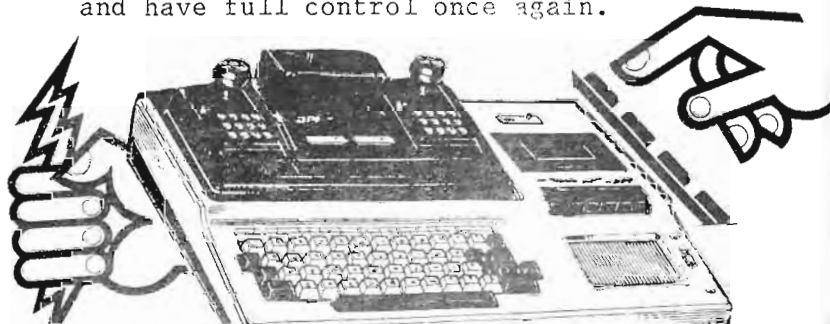
As you can plainly see, there are many things that CAN be accomplished in this mode, but, in an extremely SLOW fashion. For transmitting program data (directly typed), it is much easier to use a terminal emulator program on the transmitting end.

NEXT MONTH: The conclusion of this article and a look at a few of the popular BULLETIN BOARD SYSTEMS throughout the country using TERM-1 with a printer bridged on the line at the same time. See you then!

## SAMPLE PROGRAM THAT CAN BE TRANSMITTED TO THE DISTANT END, LINE BY LINE, THEN RAN ON THE DISTANT END COMPUTER.

```
PRINT'1OPRINT"ITS TIME TO WAKE UP THERE"  
PRINT'2OMUSIC"555666555444555"  
PRINT"3OFORZ=1TO500:NEXT"  
PRINT"4OCALL17046"  
PRINT"5OGOTO10"  
PRINT"RUN"
```

Note: As you type, the program will be entered into the distant IM-1's memory. The command RUN will cause the program to begin running on the distant end! To give control back to the receiving end, type the command PRINT"PRINT=0". The distant end will then be able to CSAVE the program and have full control once again.



# MICROPROCESSOR GLOSSARY

- ACCUMULATOR:** The register where arithmetic or logic results are held. Most MPU instructions manipulate or test the accumulator contents.
- ACCESS TIME:** Time taken for specific byte of storage to become available to processor.
- ACIA:** Asynchronous Communication Inter-face Adapter. Inter-face between asynchronous peripheral and an MPU.
- ALU:** Arithmetic and Logic Unit. The part of the MPU where arithmetic and logic functions are performed.
- ASCII:** American Standard Code for Information Interchange. Binary code to represent alphanumeric, special and control characters.
- ASSEMBLER:** Software which converts assembly language statements into machine code and checks for non valid statements or incomplete definitions.
- ASSEMBLY LANGUAGE:** Means of representing programme statements in mnemonics and conveniently handling memory addressing by use of symbolic terms.
- ASYNCHRONOUS:** Operations that initiate a new operation immediately upon completion of current one — not timed by system clock.
- BASIC:** Beginner's All Purpose Symbolic Instruction Code. An easy to learn, widely used high level language.
- BAUD:** Measure of speed of transmission line. Number of times a line changes state per second. Equal to bits per second if each line state represents logic 0 or 1.
- BAUDOT CODE:** 5-bit code used to encode alphanumeric data.
- BCD:** Binary Coded Decimal. Means of representing decimal numbers where each figure is replaced by a binary equivalent.
- BENCHMARK:** A common task for the implementation of which programmes can be written for different MPUs in order to determine the efficiency of the different MPUs in the particular application.
- BINARY:** The two base number system. The digits are 0 or 1. They are used inside a computer to represent the two states of an electric circuit.
- BIT:** A single binary digit.
- BREAKPOINT:** Program address at which execution will be halted to allow debugging or data entry.
- BUFFER:** Circuit to provide isolation between sensitive parts of a system and the rest of that system.
- BUG:** A program error that causes the program to malfunction.
- BUS:** The interconnections in a system that carry parallel binary data. Several bus users are connected to the bus, but generally only one "sender" and one "receiver" are active at any one instant.
- BYTE:** A group of bits — the most common byte size is eight bits.
- CLOCK:** The basic timing for a MPU chip.
- COMPILER:** Software which converts high level language statements into either assembly language statements, or into machine code.
- CPU:** Central processor unit. The part of a system which performs calculation and data manipulation functions.
- CROM:** Control Read Only Memory.
- CRT:** Cathode Ray Tube. Often taken to mean complete output device.
- CUTS:** Computer Users Tape System. Definition of system for storing data on cassette tape as series of tones to represent binary 1's and 0's.
- DEBUG:** The process of checking and correcting any program errors either in writing or in actual function.
- DIRECT ADDRESSING:** An addressing mode where the address of the operand is contained in the instruction. (Address below 100 in 6800)
- DMA:** Direct Memory Access.
- DUPLEX:** Transfer of data in two directions simultaneously.
- ENVIRONMENT:** The conditions of all registers, flags, etc., at any instant in program.
- EPROM:** Electrically Programmable Read Only Memory. Memory that may be erased (usually by ultra violet light) and reprogrammed electrically.
- EXECUTE:** To perform a sequence of program steps.
- EXECUTION TIME:** The time taken to perform an instruction in terms of clock cycles.
- FIRMWARE:** Instructions or data permanently stored in ROM.
- FLAG:** A flip flop that may be set or reset under software control.
- FLIP-FLOP:** Two state device that changes state when clocked.
- FLOPPY (DISK):** Mass storage which makes use of flexible disks made of a material similar to magnetic tape.
- FLOW CHART:** A diagram representing the logic of a computer program.
- GLITCH:** Noise pulse.
- HALF DUPLEX:** Data transfer in two directions but only one way at a time.
- HAND SHAKE:** System of data transfer between CPU and peripheral whereby CPU "asks" peripheral if it will accept data and "only transfers data if "answer" is yes.
- HARD COPY:** System output that is printed on paper.
- HARDWARE:** All the electronic and mechanical components making up a system.
- HARD WIRE:** Circuits that are comprised of logic gates wired together, the wiring pattern determining the overall logic operation.
- HASH:** Noisy signal.
- HEXADECIMAL:** The base 16 number system. Character set is decimal 0 to 9 and letters A to F.
- HIGH LEVEL LANGUAGE:** Computer language that is easy to use, but which requires compiling into machine code before it can be used by an MPU.
- HIGHWAY:** As BUS.
- IMMEDIATE ADDRESSING:** Addressing mode which uses part of the instruction itself as the operand data.
- INDIRECT ADDRESSING:** A form of indirect addressing which uses an Index Register to hold the address of the operand.
- INDIRECT ADDRESSING:** Addressing mode where the address of the location where the address of the operand may be found is contained in the instruction.
- INITIALISE:** Set up all registers, flag, etc., to defined conditions.
- INSTRUCTION:** Bit pattern which must be supplied to an MPU to cause it to perform a particular function.
- INSTRUCTION REGISTER:** MPU register which is used to hold instructions fetched from memory.
- INSTRUCTION SET:** The repertoire of instructions that a given MPU can perform.
- INTERFACE:** Circuit which connects different parts of system together and performs any processing of signals in order to make transfer possible (ie. serial — parallel conversion).
- INTERPRETER:** An interpreter is a software routine which accepts and executes a high level language program, but unlike a compiler does not produce intermediate machine code listing but converts each instruction as received.
- INTERRUPT:** A signal to the MPU which will cause it to change from its present task to another.
- I/O:** Input/Output.
- K:** Abbreviation for  $2^{10} = 1024$
- KANSAS CITY (Format):** Definition of a CUTS based cassette interface system.
- LANGUAGE:** A systematic means of communicating with an MPU.
- LATCH:** Retains previous input state until overwritten.
- LIFO:** Last In First Out. Used to describe data stack.
- LOOPING:** Program technique where one section of program (the loop) is performed many times over.
- MACHINE LANG:** The lowest level of program. The only language an MPU can understand without interpreter.
- MASK:** Bit pattern used in conjunction with a logic operation to select a particular bit or bits from machine word.
- MEMORY:** The part of a system which stores data (working data or instruction object code).
- MEMORY MAP:** Chart showing the memory allocation of a system.
- MEMORY MAPPED I/O:** A technique of implementing I/O facilities by addressing I/O ports as if they were memory locations.
- MICRO CYCLE:** Single program step in an MPU's Micro program. The smallest level of machine program step.
- MICRO PROCESSOR:** A CPU implemented by use of large scale integrated circuits. Frequently implemented on a single chip.
- MICRO PROGRAM:** Program inside MPU which controls the MPU chip during its basic fetch/execute sequence.
- MNEMONIC:** A word or phrase which stands for another (longer) phrase and is easier to remember.
- MODEM:** Modulator/demodulator used to send and receive serial data over an audio link.
- NON VOLATIVE:** Memory which will retain data content after power supply is removed, e.g. ROM.
- OBJECT CODE:** To bit patterns that are presented to the MPU as instructions and data.
- O/C:** Open Collector. Means of tying together O/P's from different devices on the same bus.
- OCTAL:** Base 8 number system. Character set is decimal 0-8.
- OP CODE:** Operation Code. A bit pattern which specifies a machine operation in the CPU.
- OPERAND:** Data used by machine operations.
- PARALLEL:** Transfer of two or more bits at the same time.
- PARITY:** Check bit added to data, can be odd or even parity. In odd parity sum of data 1's + parity bit is odd.
- PERIPHERAL:** Equipment for inputting to or outputting from the system (e.g., teletype, VDU, etc.).
- PIA:** Peripheral Interface Adapter.
- POP:** Operation of removing data word from LIFO stack.
- PORT:** A terminal which the MPU uses to communicate with the outside world.
- PROGRAMS:** Set of MPU instructions which instruct the MPU to carry out a particular task.
- PROGRAM COUNTER:** Register which holds the address of next instruction (or data word) of the program being executed.
- PROM:** Programmable read only memory. Proms are special form of ROM, which can be individually programmed by user.
- PUSH:** Operation of putting data to LIFO stack.
- RAM:** Random Access Memory. Read write memory. Data may be written to or read from any location in this type of memory.
- REGISTER:** General purpose MPU storage location that will hold one MPU word.
- RELATIVE ADDRESSING:** Mode of addressing whereby address of operand is formed by combining current program count with a displacement value which is part of the instruction.
- ROM:** Read Only Memory. Memory device which has its data content established as part of manufacture and cannot be changed.
- SCRATCH PAD:** Memory that has short access time and is used by system for short term data storage.
- SERIAL:** Transfer of data one bit at a time.
- SIMPLEX:** Data transmission in one direction only.
- SOFTWARE:** Programs stored on any media.
- SOURCE CODE:** The list of statements that make up a program.
- STACK:** A last in first out store made up of registers or memory locations used for stack.
- STATUS REGISTER:** Register that is used to store the condition of the accumulator after an instruction has been performed (e.g., Acc = 0).
- SUB ROUTINE:** A sequence of instructions which perform an often required function, which can be called from any point in the main program.
- SYNTAX:** The grammar of a programming language.
- TRAP (Vector):** Pre-defined location in memory which the processor will read as a result of particular condition or operation.
- TRI STATE:** Description of logic devices whose outputs may be disabled by placing them in a high impedance state.
- TTY:** Teletype
- TWO'S COMPLEMENT ARITHMETIC:** System of performing signed arithmetic with binary numbers.
- UART:** Universal Asynchronous Receiver Transmitter.
- VDU:** Video Display Unit.
- VECTOR:** Memory address, provided to the processor to direct it to a new area in memory.
- VOLATILE:** Memory devices that will lose data content if power supply removed (i.e., RAM).
- WORD:** Parallel collection of binary digits much as byte.



HARDWARE/SOFTWARE UNBELIEVABLE PRICES!!!!

OFFER #

- 1 PAK-MAN, SUPERFROG, ESCAPE!! and 2 FREE APF PROGRAMS ALL 5 ONLY \$8.50 shipped free

---

  - 2 MS. PAK-MAN, FOOSEBALL!! and 2 FREE APF PROGRAMS ALL 4 ONLY \$8.50 shipped free

---

  - 3 ANY 2 OF MINE AND 3 APF PROGRAMS ONLY \$7.50 shipped free

---

  - 4 2 BASIC CARTRIDGES \$25.00 EACH

---

  - 5 8K EXPANSION CARTRIDGES \$20.00 shipped free...ONLY 2 LEFT IN STOCK

---

  - 6 AS IS IM-1's INCLUDES TOP, BOTTOM, J CONNECTOR \$55.00 shipped free (CANADIANS ADD \$5 s/h)

---

  - 7 APF DIAGNOSTIC and FREE APF PROGRAM ONLY \$5.00

---

  - 8 ANY 1 OF MY GAMES: PAK-MAN, SUPERFROG, ESCAPE, MS.PAK, FOOSEBALL ONLY \$5.00. INCLUDES FREE APF

---
- I JUST RECEIVED A NEW SHIPMENT OF SOFTWARE, SO IF I WAS PREVIOUSLY OUT OF A GAME YOU ORDERED, IT'S IN NOW. PLEASE LIST ALTERNATES, THESE GAMES ARE G O I N G F A S T!!!!
- 
- 9 7 APF PROGRAMS ONLY \$5.50 shipped free

---

  - 10 12 APF PROGRAMS ONLY 10.50 shipped free

---

ORDER FORM

MAKE CHECK PAYABLE TO: ERIC BECKETT

8836 W. Waterford Sq. S.  
Greenfield, WI 53228

OFFER #      PRICE

- 1-----\$8.50
- 2-----\$8.50
- 3-----\$7.50
- 4-----\$25.00ea.
- 5-----\$20.00ea.
- 6-----\$55.00
- 7-----\$5.00
- 8-----\$5.00
- 9-----\$5.50
- 10-----\$10.50

APF PROGRAMS AVAILABLE

- |   |   |
|---|---|
| <input type="checkbox"/> ELECTRONIC FILES     | <input type="checkbox"/> MUSIC COMPOSER   |
| <input type="checkbox"/> TYPING TUTOR         | <input type="checkbox"/> SPACE DESTROYERS |
| <input type="checkbox"/> BUDGET MANAGER       | <input type="checkbox"/> HANGMAN          |
| <input type="checkbox"/> PERSONAL BUS.MACH.   | <input type="checkbox"/> SHOOTING GALLERY |
| <input type="checkbox"/> BILLBOARD            | <input type="checkbox"/> CASINO           |
| <input type="checkbox"/> SPACE, SIZE, SURFACE | <input type="checkbox"/> BASEBALL         |
| <input type="checkbox"/> MATH TUTOR           | <input type="checkbox"/> BOXING           |
| <input type="checkbox"/> THE WORD FACTORY     | <input type="checkbox"/> BACKGAMMON       |
| <input type="checkbox"/> SPELLING DUEL        | <input type="checkbox"/> CANTENA          |

JUMBLED UP THINGS

TOTAL      \$

CANADIAN ORDERS PLEASE  
INCLUDE \$2.00 TO HELP WITH  
SHIPPING.

Please list 'A' for ALL ALTERNATES

- |                                  |                                    |                                 |
|----------------------------------|------------------------------------|---------------------------------|
| <input type="checkbox"/> PAK MAN | <input type="checkbox"/> SUPERFROG | <input type="checkbox"/> ESCAPE |
| <input type="checkbox"/> MS. PAK | <input type="checkbox"/> FOOSEBALL |                                 |

