>CURSOR™

COPYRIGHT 1980 THE CURSOR GP.

THE TINY MICRO COMPUTER NEWS SERVICE

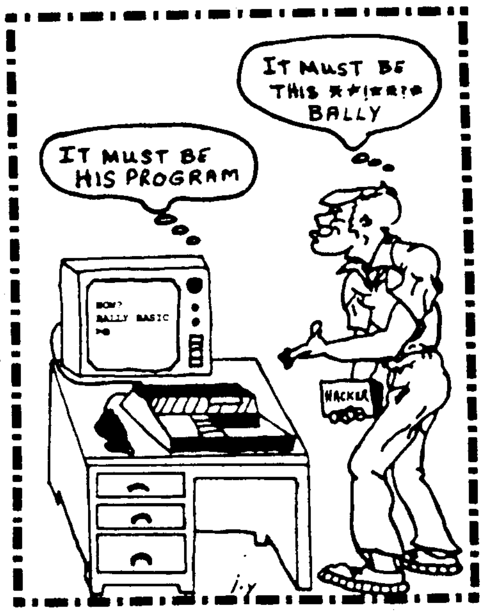
VOLUMN II OCTOBER 1980 ISSUE 3

$9.75 SEMI-ANNUALLY $2.00 PER ISSUE

PEEK n' POKE

The major problem we have faced programming the BALLY has been our inability to put up graphic characters that *do not blink!* The only way we have been able to move characters around the screen has also required us to erase the characters previous location. That method is unfortunately very slow. The lack of speed causes the blinking.

We have known for a long time that the only way to get "real-time" graphics was through handling "screen-interrupts"!! Your computer is capable of executing many many instructions per second; a screen interrupt does just that - it *interrupts* the screen scan at a predetermined rate and during that fraction of a second can perform other instructions that are separate from the program that is written in BASIC.



An example of programming using screen interrupts is the program "CRITTER" by Brett Bilbrey in this issue. "CRITTER" displays a "space invader" type creature on the screen whose speed is controlled by the control knob on Hand Control #1. This creature does not blink and is totally independent of the BASIC program. When the program is "RUN" for the first time, it places a machine language program in the bottom of the screen, executes it, and returns. Which means, you could now eliminate the program for "CRITTER" by keying in each line # and pressing GO until the total program is gone, and the creature will continue bouncing around the screen until you type in ":RETURN".

There is one limitation, we are storing the machine language program in the bottom of the screen, (you will see it twinkling) and if the cursor gets down that far, it will destroy the program. So, pull the cursor up when it starts getting low on the screen with CY=4Ø.

Handling interrupts will allow us to do almost anything we have ever desired, including putting up real time clocks and unlimited colors, in addition to monitoring real time devices such as burgler alarms.

As you have no doubt noticed by now, to write truly sophisticated software requires two things: how to use PEEK n' POKE and also a knowledge of the ON-BOARD ROM SUBROUTINES. Luckily for you, we have written a brand new manual entitled "PEEK n' POKE". This manual will give you beginner level instructions on how to use the PEEK n' POKE commands. This manual also includes a complete explanation of the Input and Output Ports that are utilized in interrupt processing along with a complete explanation of the "CRITTER" program. For those of you who have been waiting a long time for a way of handling floating point decimal numbers (such as dollars and cents) the PEEK n' POKE manual also includes an 11 page tutorial on how to build a complete checkbook program using a decimal routine and the "REM" statements to store literally hundreds of checks and be able to recall individual checks. By using the

MAIL ADDRESS: P.O. BOX 266, NORTH HOLLYWOOD, CA 91603
BUSINESS ADDRESS: 6115 CLYBOURN, SUITE 25, NORTH HOLLYWOOD, CA 91606 • (213) 763-7701

methods in our "PEEK n' POKE" manual in conjunction with the "On-Board" Subroutines manual, you will be able to write programs for the Bally that would require between 8K and 16K if it weren't for our instructions.

The PEEK n' POKE manual to include the "Checkbook" tutorial and Interrupt Handling info is only $7.50. If you want to write sophisticated software, but need complete instructions, this manual is a must. It is available for immediate shipment from CURSOR.

CRITTER
BY
BRETT BILBREY

This program will place a space invader type "CRITTER" on the screen that will bounce from top to bottom and side to side without disturbing anything that is already on-screen. This "CRITTER" will run independent of anything else you wish to do. If you press "HALT", he won't! His speed is controlled by Hand Control Knob #1.

After you have "RUN" this program, do not scroll to bottom line! Use "CY=40" to keep any text away from the area in the bottom of the screen that is "twinkling" (also, do not use "CLEAR").

Once the Basic program has been "RUN", it can be erased and replaced with whatever you want. Use ":RETURN" to stop the routine, and "CALL 19584" to start it up again. One problem is when BASIC tries to print on top of the "CRITTER", small screen glitches appear. You can create an invisable screen by altering the value of Port 15(&(15) Interrupt Line Port), it is set to 99 which is minimum size of invisible screen. The Interrupt Line Port determines the number of lines scanned before the next interrupt(for a complete explanation of all the interrupt ports etc., refer to CURSOR "PEEK n' POKE" manual.

To give you an example of a use for this type of routine, input the following line after you have "RUN" the program: key-in ":RETURN" and hit "GO" before you key in the line:

```
1 CALL 19584;&(15)=99;INPUT A;:RETURN ;
  STOP
```

As you know, when the computer hits an "INPUT" command, it will just sit there, waiting for you to give it a value, it will not allow anything else to happen until you key in a value. With this one line program, it will start the "CRITTER" bouncing around the screen as soon as it hits the input line and will stop the "CRITTER" as soon as you input a value.

So what you ask? Well, instead of having a "CRITTER", we could have a clock decrementing from one minute. If you don't get your answer into the computer before the clock hits zero, you lose your turn and control switches to the next player. This would provide for truly sophisticated software. So, don't lose heart, we are on the opening stages of an exciting software era.

Brett notes that he received a great deal of help from Tom Wood, Dave Ibach, and John Perkins, without whose help he doubts he could have written this program.

```
 1Ø CLEAR ;&(15)=99
 2Ø A=19584;B=A;C=64Ø
 3Ø D=-9741;GOSUB C
 4Ø D=19518;GOSUB C
 5Ø D=18413;GOSUB C
 6Ø D=-813Ø;GOSUB C
 7Ø D=3539;GOSUB C
 8Ø D=-1Ø63;GOSUB C
 9Ø D=2Ø1;GOSUB C
1ØØ A=1968Ø
11Ø D=19683;GOSUB C
12Ø A=19683
13Ø D=-2Ø275;GOSUB C
14Ø D=-3296;GOSUB C
15Ø D=29677;GOSUB C
16Ø D=19568;GOSUB C
17Ø D=28721;GOSUB C
18Ø D=-274Ø;GOSUB C
19Ø D=-1Ø811;GOSUB C
2ØØ D=-8731;GOSUB C
21Ø D=-539;GOSUB C
22Ø D=-9243;GOSUB C
23Ø D=12828;GOSUB C
24Ø D=1977Ø;GOSUB C
25Ø D=255;GOSUB C
26Ø D=6151;GOSUB C
27Ø D=162Ø5;GOSUB C
28Ø D=19768;GOSUB C
29Ø D=19744;GOSUB C
3ØØ D=6151;GOSUB C
31Ø D=589;GOSUB C
32Ø D=-7683;GOSUB C
33Ø D=-7715;GOSUB C
34Ø D=-118Ø7;GOSUB C
35Ø D=-3647;GOSUB C
```

```
360 D=31725;GOSUB C
370 D=19568;GOSUB C
380 D=-13829;GOSUB C
390 A=19736
400 D=14367;GOSUB C
410 D=9293;GOSUB C
420 D=2125;GOSUB C
430 A=19744
440 D=-26624;GOSUB C
450 D=16384;GOSUB C
460 D=0;GOSUB C
470 D=2050;GOSUB C
480 D=-24566;GOSUB C
490 D=-30685;GOSUB C
500 D=-21846;GOSUB C
510 D=-22486;GOSUB C
520 D=8200;GOSUB C
530 D=2080;GOSUB C
540 D=8200;GOSUB C
550 D=0;GOSUB C
560 D=-32735;GOSUB C
570 D=1280;GOSUB C
580 D=0;GOSUB C
590 D=768;GOSUB C
600 D=5;GOSUB C
610 D=0;GOSUB C
620 D=3;GOSUB C
630 CALL B;STOP
640 %(A)=D;A=A+2;RETURN
```

POOR MAN'S MEMORY EXPANSION

BY

C. J. ANDERSON

Flash Foonman, our resident maniac and sometime inventor, came running into the office yesterday looking more wild than usual. "What would you say," he smirked, "if I told you I had just downloaded that 16K BASIC CHESS program into BALLY BASIC and it's up and running on the Arcade right now?" "I'd say you've started drinking early today," I replied, but he had already grabbed my arm and hauled me out of my chair.

Out on the bench sat the Bally. On the screen was a chessboard. White had just opened with P - K4. Nothing was happening, but I noticed that the cassette recorder was running and the red light on the interface was on. Suddenly the screen went blank and a program listing began to appear. "It's working on its opening move," said Flash. "We should have its response in a couple of hours." "A couple of hours?!" I moaned. "Hey, that's good for a chess program,"said Flash. "Atari's VIDEO CHESS, Level 7, takes an average of eight to ten hours per move. Anyway, since we've got plenty of time, let me explain what's happening here." I assured him I was all ears.

"I really do have a 16,000 byte program running here," he began, but I stopped him. "Where's your expanded RAM?" I asked. "Where is your memory add-on?" "Right here," he said, patting the cassette recorder. " And it's not RAM, it's mass memory just like a floppy disk or a stringy floppy. Only it's very slow. I call it a *stringy creepy.*" "You mean you've got a 16,000 byte program on that cassette and the Bally is processing it 1800 bytes at a time?" I gasped. "That's really neat, Flash, but how many times do you have to sit here and rewind the tape looking for the subroutines you need as the program calls for them? You've just made yourself a part of the machine, and I don't call that true computing."

"Not so, mon ami," he replied. "I never touch the machine. It calls for its own subroutines and loads them automatically whenever it needs them thanks to the newly discovered selective load function that Fred Cornett wrote about in Vol. II, Issue 2 of CURSOR. Each subroutine on the tape is prefaced by an identifier number, you know: 257, 258, etc. Each subroutine uses the same line numbers so when a new subroutine is called for it automatically erases the previous one from our constant 1800 byte RAM. When the new subroutine is fully loaded, a :RETURN; GOTO 1000 command on the tape starts the computer running again and it branches down into the new material. Then it decides which subroutine it wants to see next, sets the A variable to that number, and branches to a :INPUT A ;STOP command in RAM. That sends it back to the tape to find the subroutine that corresponds to the value of A. Simple, huh?"

"Hold it!" I said. "That's really brilliant, all right, but what happens when the cassette comes to the end? How do you rewind it to start over?" "I don't have to," said Flash, grinning, "because this is a very special cassette. It has no end and no beginning. It's the Automatic Repeating Cassette, catalog number 60,920 from Edmund Scientific Co., 101 E. Gloucester Pike, Barrington, New Jersey 08007. It sells for a measly $12.95 plus $1.75 shipping and handling, and it boasts 20 minutes of recording time. That's enough to store 20,000 bytes of data or program. I tested the process with a Radio Shack 43-401 which has only 20 seconds of recording time and costs $5.95, but this A.R.C. from Edmund is the cat's pajamas!"

"Now let me get this straight," I said. "This endless cassette just sits here going around and around forever, and the computer keeps jumping to it, waiting for its desired data to come around then loading that data and therefore modifying its own program constantly?" "Yup," said the madman. "No limit to the size of the program you can run as long as time is not an important factor. On a long program like this 16K job you may have to wait up to 20 minutes between each subroutine, which is oka, for a sophisticated chess game or for mailing list or word-processing applications. For shorter programs, but still larger than 1800 bytes, I just load the subroutines in sequence several times on the cassette loop. That speeds up the search. If I can get them on twice, we can search in ten minutes. If I can get them on three times, it's a little less than 7 minutes."

Just then the chessboard reappeared on the screen and the computer made its move: P - K4. " I expected that," mumbled Flash, and reached for his copy of Bobby Fischer Teaches Chess. I left him mumbling something about the Sicilian Defense, but not before I had picked up his notes off the bench. Here's a simple little program that Flash designed to demonstrate his "*stringy creepy*" concept to any skeptical users.

STRINGY CREEPY CASSETTE OPERATING SYSTEM (COS)

by Flash Foonman

Use an ordinary blank cassette, or one of the two described. The program will demonstrate how the computer can continually modify its internal program by acting on its own decisions. If you use an endless cassette, it will do so with no human assistance. Otherwise you must rewind the tape after each response. Type the following line:

```
1ØØ PRINT "FOUND 257";RETURN
```

Now, transfer it to the tape with the following commands input without a line number (before hitting "GO" make sure your tape recorder is recording):

```
NT=1;:PRINT ;TV=1;TV=1;LIST;PRINT ":RETURN
;GOTO 1Ø
```

Stop the recorder. Do not rewind. Now type

```
1ØØ PRINT "FOUND 258";RETURN
```

Transfer it to tape with the previous command, changing the identifier to"...TV=1; TV=2..." etc. Repeat this process twice more, using "...FOUND 259..." "...TV=1;TV=3..." and "...FOUND 26Ø..." "...TV=1;TV=4...".

Rewind the cassette. RESET the Bally and enter this simple program:

```
  5 CLEAR
 1Ø GOSUB 1ØØ;A=RND (4)+256;PRINT "A=",#Ø,
    A
 2Ø :INPUT A;STOP
1ØØ PRINT "START CASSETTE";RETURN
```

Okay, RUN the program. Start the cassette. The screen displays the value it has selected for A, the input port opens, the program stops, and the computer loads subroutine A when it finds it. It then restarts itself and announces that it has found subroutine A.

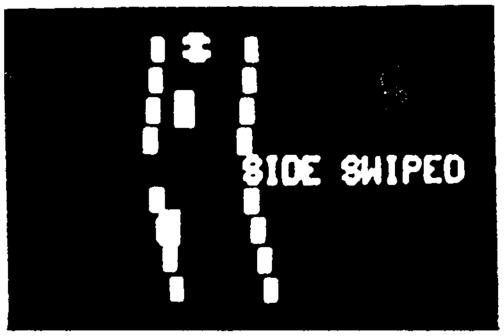
HALT the program at this point, and LIST it on the screen. Notice what has happened to line 1ØØ; out with the old, in with the new. If you're using a repeating cassette you can sit back and watch the computer reprogram itself all day long. The actual resident program is only 44 bytes long, but its effective length is 112 bytes. It's slow, but it's real honest-to-gosh expanded memory.

As for Flash, I just heard him shout: "Hey! If a guy has two cassettes running simultaneously, one playing and one recording, the computer could be pulling data off the first one, modifying it, and rewriting it onto the second one!" I'm going out for a drink!

****

SIDESWIPE
BY
MIKE PEACE
WAVEMAKERS

Editors Note: Your goal is to steer your car through and around the other vehicles on the road at the same time making sure you don't hit the sides of the road. Your car is the one with the broken boxes at the top of the screen. The road moves up toward you from the bottom of the screen as shown in the photograph. Mike as usual has done a very thorough job using very limited memory. This program uses some interesting sounds, and a unique method of movement.Use Hand Control Knob #1.



WHENEVER YOU SEE "ς", PRESS "SPACE".

```
2  .WAVEMAKERSςςςςςςςςςςςςςςς→SIDESWIPE←
5  :RETURN ;BC=1Ø;FC=191;&(2Ø)=9;GOTO 1Ø
8  &(18)=9;FOR N=2ØØTO ØSTEP -1Ø;&(21)=N;
   &(22)=N;&(23)=N;NEXT N;S=S+1Ø;PRINT "C
   OLLISION";RETURN
9  FOR N=1ØTO 25;&(22)=255;&(18)=N;NEXT N;
   &(22)=Ø;PRINT "SIDE SWIPED";S=S+3;RETUR
   N
1Ø CLEAR ;FOR A=-11TO Ø;TV=13;NEXT A;C=Ø;S
   =Ø;NT=Ø
15 PRINT "ςςςςςSTARTςςςςςςςCOURSE",;FOR T=
   1TO 9Ø;E=E+1;TV=13;IF TR(1)RUN
2Ø R=RND (15);IF R>13Q=-RND (2)
21 IF E>5BOX A+RND (26)-13,-32,6,9,1;E=RND
    (3)
25 IF A<-36Q=2
26 IF A>36Q=-2;CX=-72
27 IF A<-2CX=A+17
3Ø A=A+Q;BOX A-15,-32,4,6,1;BOX A+15,-32,4
   ,6,1;IF R<3Q=RND (2)
5Ø C=C+KN(1)÷25
55 &(18)=13Ø+ABS(C);&(22)=15Ø
6Ø BOX C,32,5,7,1;BOX C,31,9,2,1;BOX C,34,
   9,2,1
7Ø IF PX(C,27)GOSUB 8;C=A;GOTO 9Ø
8Ø IF PX(C+4,32)+PX(C-4,32)GOSUB 9;C=A
9Ø NEXT T;CY=Ø;CX=-4Ø;PRINT "FINAL SCORE";
   PRINT #12,1ØØ-S
93 &(22)=Ø;IF H<1ØØ-(S)H=1ØØ-S
95 CX=-62;PRINT "TODAYS HIGH SCORE ",#Ø,H;
   PRINT "ςςςPULL TRIGGER TO RUN
96 IF TR(1)RUN
99 GOTO 96
```

WE THREE KINGS OF ORIENT ARE
Christmas Music by George Moses

Using the music program published in The Cursor in March, 1980, pages 17 thru 19, input the following data. You will need three rem lines of 97 bytes each to store this data. Run the program and input the data one chord at a time (each column of 4 numbers is a chord) progressing from the top down for each of the four numbers, and from left to right for the progression of chords.

```
35   39   44   53   47   44   47   53   0    35   39
44   107  53   107  57   71   57   71   0    44   107
107  47   107  71   80   57   80   107  0    107  47
4    2    4    2    2    2    2    4    2    4    2
.......................................................
44   53   47   44   47   53   0    44   53   39   47
53   107  57   71   57   71   0    53   107  47   120
107  71   80   57   80   107  0    107  44   120  39
4    2    2    2    2    4    2    4    2    4    2
.......................................................
35   44   29   33   35   39   35   39   44   47   53
44   90   39   39   44   53   136  53   142  57   71
90   35   95   95   90   136  53   136  53   142  107
4    2    2    2    2    2    2    2    4    2    4
.......................................................
0    47   39   44   71   90   60   44   53   44   0
0    60   60   71   90   44   71   53   136  60   0
0    80   67   90   44   71   90   136  67   90   0
2    4    2    4    2    4    2    4    2    4    2
.......................................................
44   60   90   60   44   53   44   0    44   53   39
60   90   44   71   53   136  60   0    53   107  47
90   44   60   90   136  67   90   0    107  44   120
4    2    4    2    4    2    4    2    4    2    4
.......................................................
35   33   35   39   35   44   90   60   71   44   53
44   67   44   47   44   90   60   44   90   53   136
90   44   90   120  90   60   44   90   60   136  67
2    4    2    4    2    4    2    4    2    4    2
.......................................................
44   0
60   0
90   0
6    2
.......................................................
```

Add the following four lines to print the title, turn the vibrato on and off at the appropriate times, repeat the stanzas twice and shut off the registers at the end of music.

```
199 CLEAR; CY=5; PRINT"    WE THREE KINGS"; PRINT
    "       OF ORIENT ARE"; PRINT; PRINT"   BY J. H.
    HOPKINS JR.
235 IF C=-24296 IF M<2 A=-24574; M=M+1; &(20)=0; GOTO210
236 IF C=-24437 &(20)=129
240 IF C>-24299 GOTO 250
Change line 260
260 &(17)=0; &(18)=0; &(19)=0; &(20)=0; M=0; A=KP; GOTO 200
```

Also, remember the "Tonic for Bitter Music" I contributed last month on page 61? Here's one that's much improved, using less memory and no IF statements. IF statements slow down the music. Just DELETE LINE 225 and add the following.

```
210 FOR C=A TO A+92 STEP 4; FOR D=1 TO (%(C+3)÷256+127)
     ÷LxT
230 NEXT D
```

L and T are variables that affect timing and duration. T speeds up or slows down all notes equally, while L, being a divisor can be used to set the duration for the shortest notes to one or zero while not similarly affecting the longer notes, using the characteristics of Tiny Basic's division. For the above song they should be entered in line 205 as follows-

```
205 T=3; L=2
```

To start music with title type in GOTO 199 and hit GO. To repeat the song push any button on the keypad. MERRY CHRISTMAS!!

MACHINE LANGUAGE GRAPHICS
USING
THE ON-BOARD SUBROUTINE MANUAL
BY
BRETT BILBREY

Editors Note: The following routines do not use screen interrupts, so the graphics do blink.

You say you want FAST GRAPHICS in BASIC to be used like the "Three Voice Music (see CURSOR Vol I,Issue 3)? Yep, it's me again!!
Well, please read on and let me open the door to a new realm of graphics.

The Bally games all use machine code. Even the Bally BASIC is composed of machine code. Well, all those wonderful on-board sub-routines in CURSOR's manual... Yep. you got it. Machine code. This means if you want fast graphics, you need to know about machine code.

Contrary to popular belief, there are four colors available using BASIC. There is however a drawback, you can't eliminate the garbage at the top of the screen (which is your Basic program when &(9) = Ø). Well, if you are very careful, you can get up four separate colors without the junk. My "Graphics Demo" will do this, but you will see why you have to be careful.

Let's say you want to move a pattern made of four colors around the screen, and that the maximum size of this pattern is 5 pixels deep by 8 pixels wide. First define the colors.

W = White (color #7)
B = Blue (color #240)
R = Red (color #122)
G = Green (color #200)

Take a look at the following chart, we have substituted the first letter of the color for each position of the block. The block as a whole will constitute a small spaceship.

W	W	W	W	B	W	W	W
W	R	W	G	B	G	W	W
W	R	R	B	B	G	G	W
W	R	W	G	B	G	W	W
W	W	W	W	B	W	W	W

Now, we set the color ports and translate the pattern to numbers. Now we get to a different form of number representation called BINARY.

First, lets mark down on paper what colors we are going to assign the color ports and use this for future reference.

PORT COLOR #
&(Ø)=7
&(1)=24Ø
&(2)=122
&(3)=2ØØ

Our computer can only understand Base 2 numbers(but with the help of an interpreter can also understand Base 10 which is decimal and Base 16 which is Hex). Base 2 is also known as Binary. All this means is only zero's and one's are used. For our purposes, we will be using Hex and Binary until we assemble our final program.Please refer to the following chart.

Binary to Hex Conversion Table

Binary=Hex	Binary=Hex	Binary=Hex
0000 = 0	0001 = 1	0010 = 2
0011 = 3	0100 = 4	0101 = 5
0110 = 6	0111 = 7	1000 = 8
1001 = 9	1010 = A	1011 = B
1100 = C	1101 = D	1110 = E
1111 = F		

Now we convert our pattern to numbers the computer can understand (Binary). What we are doing is substituting the port # for the color #.

Port #Ø	becomes	00
" 1	"	01
" 2	"	10
" 3	"	11

Now we write down our pattern the same way we did when we used the first letter of the color except this time we use the binary number for the color port.
You will notice that the pattern has a line dividing it into 2 parts. This is because we must give this information to the computer one byte at a time (a byte is composed of eight bits)(a bit is a zero or a one): looking at the table we would find that 00011010 would equal 1A in Hex. Looking at our pattern in binary we see that it is two bytes wide and the Hex equivalents are located to either side. We also note that our pattern is 5 rows deep. We will need to remember the size as 2x5.

```
HEX    Binary     |   Binary      HEX
 Ø ←00 00¦00 00   | 01 00¦00 00→ 4Ø
23 ←00 10¦00 11   | 01 11¦00 00→ 7Ø
29 ←00 10¦10 01   | 01 11¦11 00→ 7C
23 ←00 10¦00 11   | 01 11¦00 00→ 7Ø
 Ø ←00 00¦00 00   | 01 00¦00 00→ 4Ø
```

Lets explain the number representation system one more time. Our computer only understands Binary, but we must write our machine language programs in Hex, and since our BASIC only understands Decimal, we must convert our Hex programs into decimal. If you find this confusing we strongly suggest you go to a technical book store or a computer shop and buy a very excellent book written by Kathe Spracklen, "Z-80 and 8080 Assembly Language Programming", published by Hayden, Library of Congress Catalog Card Number 79-65355. This book should clear up any confusion.

Also, please refer to CURSOR Volume 1, Issue 2. As we have mentioned in that issue, when converting a Hex program to decimal, me must reverse their order and then put them together as a foursome thereafter converting them to decimal. The conversion to decimal can be accomplished using the Hex to Decimal Converter program in Vol I, issue 2, or you could buy a Texas Instrument calculator designed specifically for computer programmers called the "TI Programmer" for approx. $59.95. Now, lets convert what we have already done to decimal:

	HEXPAIR	HEXQUAD	DECIMAL
Pattern Wide	02	0502	1282
Pattern Deep	05		
Byte #1	00	4000	16384
Byte #2	40		
#3	23	7023	287Ø7
#4	70		
#5	29	7C29	31785
#6	7C		
#7	23	7023	287Ø7
#8	70		
#9	00	4000	16384
#10	40		

Now that we have a pattern and have put it in table form to be used in a basic program, how do we put it on the screen? Well, we have to put the pattern in a program to be POKE'd into memory where it can be called back and displayed by an on-board subroutine. The following is the machine language program to be used from basic.

D5	Push Basic Pointer (Save your place)
FF	RST 38 (Call on-board Sub #)
23	Routine #
0	X Coordinate
20	Y Coordinate
20	Magic Register
E8	Lo Pattern Address
4E	Hi Pattern Address
D1	Return Basic Pointer
C9	Return (to BASIC program)

Next we make a BASIC program out of this by reversing Hex pairs and converting to decimal numbers:

```
Example: D5
         FF  FFD5   -43
         23
         0   0023    35
```

All we have to do is add the decimal info from our previous table and just this easy we have a simple machine language program to put a four color pattern up on the screen from BASIC.

```
 1Ø CLEAR ;&(9)=Ø;&(Ø)=7;&(1)=24Ø;&(2)=1
    22;&(3)=2ØØ
 2Ø A=2Ø18Ø;B=A;C=16Ø
 3Ø X=-43;GOSUB C
 40 X=35;GOSUB C
 5Ø X=8256;GOSUB C
 6Ø X=2Ø224;GOSUB C
 7Ø X=-13871;GOSUB C
 8Ø A=2Ø224
 9Ø X=1282;GOSUB C
1ØØ X=16384;GOSUB C
11Ø X=287Ø7;GOSUB C
12Ø X=31785;GOSUB C
13Ø X=287Ø7;GOSUB C
14Ø X=16384;GOSUB C
15Ø CALL B;&(9)=5Ø;STOP
16Ø %(A)=X;A=A+2;RETURN
```

To move our Spaceship, replace line 15Ø with:

```
15Ø FOR E=1TO 1ØØ;%(2Ø182)=Ex256+35;CALL
    B;FOR I=1TO 15;NEXT I;CALL B;NEXT E;&
    (9)=5Ø;STOP
```

OPEN HOUSE OPEN HOUSE

Come to the Open House at Cursor, 59 E. Orange Grove Ave., Burbank, CA 91502

We will be showing the new Bally cartridge "DOGPATCH" and showing off a MEMORY-ADD-ON. A lot of people will be trading tapes, so if you want access to a computer, it might help to bring a Bally and a TV, even black & white.

I5 NOVEMBER 1980 (SAT)

II AM TO 3 PM

C L A S S I F I E D S

CURSOR SOFTWARE TAPE #1

The following two photographs and descriptions make up the first CURSOR Cassette Tape offering. Both programs are listed on one tape, and include complete documentation. Price is $8.95 complete. Send checks or money orders to: CURSOR, P.O. Box 266, N. Hollywood, CA 91603.

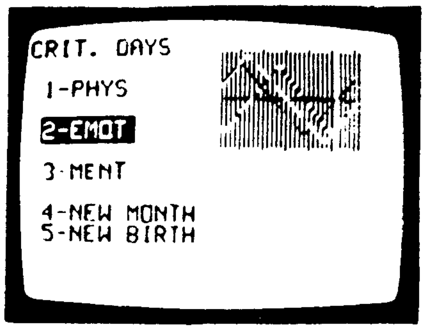
PROGRAM 1: MOON LANDING

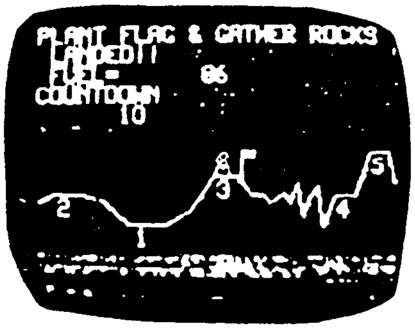
You're in the L.E.M. waiting for the instruction to break away from the mother ship. Once you do, you have to quickly scout for a safe landing spot. You carefully maneuver into a safe landing position; watching the drift and speed. If you successfully land, you have to wait for the countdown to blast-off for re-connection to the mother ship prior to running out of time and fuel. (Software selectable gravity wells.) Program is partly in machine language to generate the fast acting user-defined characters for: Horizontal LEM, LEM banked to the right, LEM banked to the left, 2 explosions (moving). Great sound & graphics.

PROGRAM 2; BIO-RHYTHM

Through this computerized study of biological clocks you can predict physical, emotional and intellectual behavior at peak and critical periods. Bio-rhythm has helped U.S. airlines avoid crashes and athletes to choose their best competive days; it has reduced dramatically the auto accident rate in Japan and increased the performances of sales forces, teachers, and factory workers. Bio-rhythm can help you predict outbreaks of illness, mental depression, days of tireless energy, best times for creative work, peak periods of mental and emotional control.

Very accurate graph format allows you to select and see your critical days individually. No other program like it!

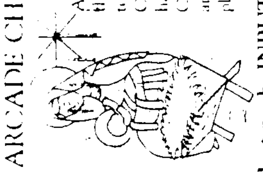




When you're through playing "Space Invaders" . . .

PLAY "SILENT NIGHT" on your Bally Arcade!

PRESENTING 3-PART ARCADE CHRISTMAS MUSIC



A 40 minute cassette tape that programs your Arcade to play 27 beautiful Christmas Carols in 3-part harmony, some with vibrato. Close your eyes and listen and you'll think someone's playing an organ!

Just push: INPUT-GO and they all play consecutively.
All 27 programs on one tape $9.50

Send check or money order to:

GEORGE MOSES CO.
110 E. North St., Brighton, MI 48116
(313) 227-1575

CURSOR
P.O. BOX 266, NO. HOLLYWOOD, CA 91603

NEW CARTRIDGE-DOGPATCH

Open House at CURSOR, 59 E. Orange Grove Ave., Burbank, CA 91502 on 15 November 1980 11 AM TO 3 PM. We will be showing the Dogpatch Cartridge & a memory add-on.

VAN NUYS CA
OCT

ALWAYS
USE

USA 15
THE HOME OF THE BRAVE