MILLERS GRAPHICS

THE SMART PROGRAMMER

FREE MINI SAMPLE

This 4 page FREE MINI SAMPLE of the Smart Programmer newsletter is a condensed version of Volumes 1 & 2. The 16 page monthly subscription newsletter started with the February 1984 issue.

Over the past few months we have received numerous letters and survey forms containing nice comments and questions about the Smart Programming Guide For Sprites. We have enjoyed them very much and we would like to thank you for returning the survey forms.

Many of the questions we received were on how to modify the General Bar Graph program to show a scale that is lower than 0-200. Listed below are the program lines that have been changed and added to allow a low scale of 0-.25. The balance of the program listing is on pages 64 & 65 in The Smart Programming Guide For Sprites.

```
160 SUB GRAPH(G$(),G()):: CA
LL CLEAR :: CALL SCREEN(7)::
 FOR I=1 TO 8 :: CALL COLOR(
I,2,12):: NEXT I :: CALL COL
OR(12,13,4,9,13,4)

170 CALL CHAR(124,"010101010
10101010000000001",62,"00000
0000001",94,"00000001010101"
,96,"00FFFFFFFFFFFFFF")

180 M=0 :: C$=RPT$(RPT$(CHR$
(125),4)&CHR$(124),5):: DISP
LAY AT(1,(29-LEN(G$(0)))/2):
G$(0): : :"  ^>>>>^>>>>^>>>>
^>>>>^>>>>^"

190 FOR I=1 TO 20 :: IF G$(I
)<>"" THEN DISPLAY AT(I+4,4)
:C$ :: M=MAX(M,G(I))ELSE 205

205 IF M>8 THEN C=1 :: GOTO
210 ELSE C=100 :: M=M*100

206 FOR I=1 TO I-1 :: G(I)=G
(I)*100 :: NEXT I

215 IF M>100 THEN M=INT((M-1
)/200)+1 ELSE IF M>50 THEN M
=.50 ELSE IF M>25 THEN M=.25
 ELSE M=.125

220 DISPLAY AT(2,1):"SCALE X
"&STR$(MX)&" > ="&STR$(M*8/C
)

225 IF C=100 THEN DISPLAY AT
(3,3):USING "0 #.## #.## #.#
# #.## #.##":40/MX*M/C,80/MX
*M/C,120/MX*M/C,160/MX*M/C,2
00/MX*M/C :: GOTO 240

240 C=0 :: FOR I=1 TO I-1 ::
 DISPLAY AT(I+4,1)SIZE(3):G$
(I):: IF G(I)<M*8 THEN 260
```

Another popular question was regarding the shooting program on page 35 and how to incorporate joysticks into it. Listed below are the lines that have been changed or added to the program. The IF ABS(X-C)<9 statement in line 115 sets the tolerance to allow a hit or not. To make it harder use a smaller number such as 4, 5 or 6. The actual tolerance = (number-1)*2
ie: (9-1)*2 = 16 pixels tolerance.

```
100 CALL CLEAR :: CALL SCREE
N(2):: CALL CHAR(46,"0000001
818"):: CALL SPRITE(#2,94,16
,180,100)

115 CALL JOYST(1,X,Y):: CALL
 MOTION(#2,0,X*3):: CALL KEY
(1,X,Y):: IF Y=0 THEN 115 EL
SE CALL POSITION(#3,Y,X,#2,R
,C):: IF ABS(X-C)<9 THEN 120
```

```
116 CALL SPRITE(#1,46,16,R,C
,(Y-R)*.49,0):: CALL SOUND(4
76,-3,14):: CALL SOUND(120,1
10,30):: CALL DELSPRITE(#1):
: GOTO 115

120 CALL SPRITE(#1,46,16,R,C
,(Y-R)*.49,(X-C)*.49):: CALL
 SOUND(476,-3,14)
```

Here are two more PEEK addresses that you might find useful. The first one is from the SPCHRD equate (speech read) located at Hex >9000.

```
CALL PEEK(-28672,A)
```

```
IF A=96 THEN..... the  Speech Synthesizer
            is attached to the computer.
IF A=0  THEN..... the  Speech Synthesizer
            is not attached.
```

This one is located in our CPU Scratch pad Ram at Hex >8370. This is a two byte address that contains the highest free VDP Ram address. If you do not have Mem-Expansion your program is loaded into VDP Ram from the bottom up.

```
CALL PEEK(-31888,A,B)
Highest address=A*256+B
```

If you have Mem-Expansion, the Mini Memory cartridge or the Editor Assembler cartridge you can CALL LOAD(-31888,63,255) to shut down ALL of your Disk Drive Files (same as CALL FILES(0) which is not allowed as a command) and then type in NEW or edit your program to open up the memory space. The easiest way to turn your Disk Drives back on is to type in BYE.

CALL LOAD (-31888,55,215) and then RUN, NEW or editing your program in Extended BASIC with Mem-Expansion will also turn your Drives back on. It is possible to use these CALL LOAD's in your programs if you have Memory Expansion, but they haven't been fully tested so make sure that you have backups before you RUN the program or you may lose it!

NOTE: Any Disk accesses with the Drives shut down will lock up your computer and you will have to shut it off and then back on to regain control.

Here is a new address for you to play with >83C2, it is located in the 256 byte area of memory know as Scratch Pad Ram. This address is know as the 'Interrupt Flag' and it is a 1 byte bit mapped address.

If you have memory expansion you can turn the bits on and off by using the following CALL LOAD's. Don't forget about CALL INIT.

Bit	Use
0	On disable all of the following
1	On disable Sprite motion
2	On disable Auto Sound processing
3	On disable the QUIT key (FCTN =)
4-7	not used

```
>83C2 = decimal 33730
 33730-65536 = -31806

CALL LOAD(-31806,128) = bit 0 on
CALL LOAD(-31806,64)  = bit 1 on
CALL LOAD(-31806,32)  = bit 2 on
CALL LOAD(-31806,16)  = bit 3 on

CALL LOAD(-31806,0)   = All bits off

CALL LOAD(-31806,48)  = bits 2 & 3 on
CALL LOAD(-31806,80)  = bits 1 & 3 on
CALL LOAD(-31806,96)  = bits 1 & 2 on
```

If you type in CALL INIT :: CALL LOAD (-31806,16) before you start programming you won't have to worry about FCTN QUIT ever again! If you disable the Auto Sound processing after a CALL SOUND statement the sound will stay on forever. You can then execute only CALL SOUND statements with negative durations such as CALL SOUND (-100,660,3) without locking up your computer.

Listed below is a little program that turns Sprite motion on and off. Even though auto Sprite motion is off all the other sprite related statements still work, such as CALL COINC, CALL LOCATE etc. After you have run the program change the color parameter in the CALL SPRITE to 7 instead of I+2.

```
100 CALL INIT :: CALL SCREEN
(2):: CALL LOAD(-31806,64)::
 CALL MAGNIFY(4):: CALL CHAR
(64,RPT$("F",64))
```

```
110 A=10 :: FOR I=1 TO 10 ::
 CALL SPRITE(#I,64,I+2,I*16,
110,0,I*A):: NEXT I

120 RANDOMIZE :: CALL PEEK(-
31880,B):: IF B<25 THEN CALL
 LOAD(-31806,64):: A=-A :: C
ALL COINC(#1,#2,30,C)ELSE 12
0

130 CALL SOUND(-100,660,(NOT
 C)*-30):: FOR I=1 TO 10 STE
P 2 :: CALL MOTION(#I,0,I*A,
#I+1,0,I*-A):: NEXT I :: CAL
L LOAD(-31806,0):: GOTO 120
```

You can also stop sprite motion by loading -31878 with 0 and then start them back up by loading -31878 with the highest numbered sprite that is in motion. Both -31806 and -31878 work but -31806 brings them to a stop a little faster, on the other hand, by using -31878 you can selectively stop the higher numbered sprites and keep the others in motion.

Since we are going to be doing a lot of talking about PEEK's in future issues of The Smart Programmer here are a few tips on what all those numbers are in memory.

Our computers can access an address as either a byte or a word. A byte is a string of binary digits (bits). In the majority of cases, including the TI 99/4A, a byte is 8 bits long. A 6-bit byte is commonly called a character, not to be confused with the characters that are printed on the screen. A 4-bit byte is usually called a digit or nibble. Our bytes are made up of 8 bits and a word is 2 bytes or 16 bits. A bit can be either on, equal to 1, or off, equal to 0. In a byte, the bits are numbered 0 through 7, with bit 0 being the most significant bit (msb) or highest value bit and 7 being the least significant bit (lsb) or lowest valued bit.

0	1	2	3	4	5	6	7
128	64	32	16	8	4	2	1

A byte can also be split in half to form a nibble or 4 bits, such as it is for the Hex codes in the character definitions.

00000111 in binary equals 7 in decimal and 07 in hex. 00001110 equals 14 in dec. and 0E in hex. 00011100 equals 28 in dec. and 1C in hex.

When you execute CALL PEEK(xxxx,A,B), both A and B contain one byte in decimal form. A and B will always be a value between 0 and 255 since 255 is the largest value that one 8 bit byte can contain. 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255. To convert this into a word multiply A times 256 and add B to it.

The following program is a number converter. You can input a number in decimal, hexadecimal or binary and the program will return the number in the other two number bases. It will also return the PEEKable decimal address if the number is larger than 32767 as well as displaying the number as a word and split into two bytes. The conversion formulas were written as subroutines so that you could easily use them in any of your other programs.

The **SMART PROGRAMMER** is a 16 page monthly newsletter that carries articles on intermediate to advanced Extended Basic programming. There are also columns on programming in TI Forth, The TI Professional Computer and a Question and Answer section where we try to answer our readers questions on computers and programming. In our Peeking Around column we are memory mapping the TI 99/4A and calling out its PEEK and LOAD addresses.

Subscribe today and join the thousands of Smart Programmers that are learning the secrets of the 99/4A.

U.S. .... 12.50 /year
Foreign Surface Mail .. 16.00 /year
Foreign Air Mail ...... 26.00 /year

☐ Yes, please send me a full year of The Smart Programmer at the introductory rate

Name ______________________

Address ______________________

City ______________________

State __________ Zip __________

Country ______________________

Mail subscription forms and payment to:
Millers Graphics
1475 W. Cypress Ave.
San Dimas, CA 91773

Here are the inputs, printouts and conversion formulas listed by line number.

- 130 Decimal input, range checking and subroutine execution.
- 140 Hexadecimal input, range checking and subroutine execution.
- 150 Binary input, range checking and subroutine execution.
- 160 Decimal printout (word and bytes).
- 170 Hexadecimal and Binary printout.
- 180 Hex to Decimal conversion.
- 190 Binary to Decimal conversion.
- 200-210 Decimal to Binary conversion.
- 220-230 Decimal to Hex conversion.

```
100 ON WARNING NEXT :: CALL
CLEAR :: H$="0123456789ABCDE
F" :: PRINT "DEPRESS YOUR AL
PHA LOCK KEY": :"PRESS LETTE
R FOR INPUT BASE": :

110 PRINT : :"D=DEC #   H=HE
X #   B=BIN #": : :: CALL SO
UND(40,660,9)

120 CALL KEY(0,K,S):: IF S<1
 THEN 120 ELSE ON POS("DHB",
CHR$(K),1)+1 GOTO 110,130,14
0,150

130 INPUT "DEC #=":DEC :: IF
 DEC<-32768 OR DEC>65535 THE
N 130 ELSE A,DEC=INT(DEC-655
36*(DEC<0)):: GOSUB 200 :: G
OSUB 220 :: GOTO 160

140 PRINT "HEX #=" :: ACCEPT
 AT(23,7)BEEP SIZE(4)VALIDAT
E(H$):HEX$ :: GOSUB 180 :: G
OSUB 200 :: GOTO 160

150 PRINT "BIN #=" :: ACCEPT
 AT(23,7)BEEP SIZE(16)VALIDA
TE("10"):BIN$ :: GOSUB 190 :
: GOSUB 220 :: GOSUB 210

160 A=INT(DEC/256):: PRINT :
"D=";DEC;TAB(12);A;DEC-A*256
 :: IF DEC>32767 THEN PRINT
"  ";DEC-65536

170 PRINT "H= ";HEX$:"B= ";S
EG$(BIN$,1,8)&" "&SEG$(BIN$,
9,8):: HEX$,BIN$="" :: A,DEC
=0 :: GOTO 110

180 HEX$=SEG$("0000",1,4-LEN
(HEX$))&HEX$ :: FOR I=1 TO 4
 :: A,DEC=DEC+(POS(H$,SEG$(H
EX$,I,1),1)-1)*16^(4-I):: NE
XT I :: RETURN

190 FOR I=1 TO LEN(BIN$):: D
EC=DEC-2^(I-1)*(SEG$(BIN$,(L
EN(BIN$)+1-I),1)="1"):: NEXT
 I :: RETURN

200 A=A/2 :: BIN$=STR$(-(A-I
NT(A)<>0))&BIN$ :: A=INT(A):
: IF A THEN 200

210 BIN$=SEG$(RPT$("00",8),1
,16-LEN(BIN$))&BIN$ :: RETUR
N

220 A=DEC+65536*(DEC>32767)

230 HEX$=SEG$(H$,(INT(A/4096
)AND 15)+1,1)&SEG$(H$,(INT(A
/256)AND 15)+1,1)&SEG$(H$,(I
NT(A/16)AND 15)+1,1)&SEG$(H$
,(A AND 15)+1,1):: RETURN
```

THE SMART PROGRAMMER is published by MILLERS GRAPHICS, 1475 W. Cypress Ave. San Dimas, CA 91773. Each separate contribution to this issue and the issue as a whole Copyright 1983 by MILLERS GRAPHICS. All rights reserved. Copying done for other than for personal use without the prior permission of MILLERS GRAPHICS is prohibited. All mail directed to **THE SMART PROGRAMMER** will be treated as unconditionally assigned for publication and copyright purposes and is subject to **THE SMART PROGRAMMERS** unrestricted right to edit and comment. MILLERS GRAPHICS assumes no liability for errors in articles.

SMART PROGRAMMER & SMART PROGRAMMING GUIDE are trademarks of MILLERS GRAPHICS

Texas Instruments, TI, Hex-Bus and Command Cartridge are trademarks of Texas Instruments Inc.