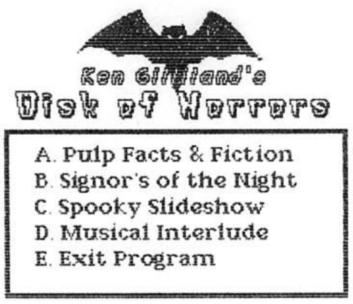
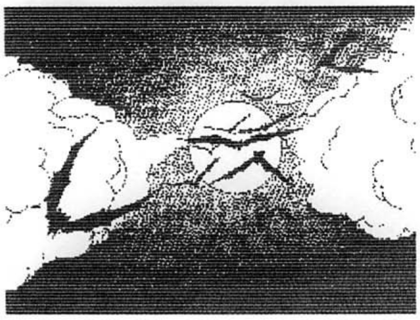
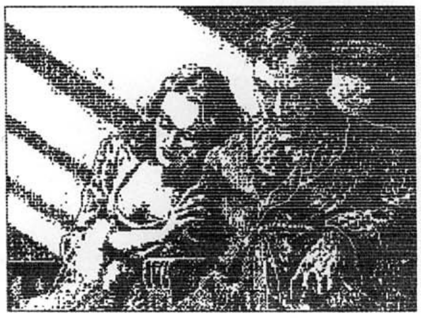
PROGRAM BITEN 92-2







INNEHÅLL

ISSN 0281-1146

REDAKTÖREN

Medlemmar kan beställa en skiva med artiklar (i DIS/VAR 80 för TI-99/4A) om CC-40 (delvis även om TI-74, TI-95 och TI-88) av Charles Good, Lima Ohio UG. Sänd skiva (en DS/SD eller två SS/SD) med frankerat svarskuvert till redaktören. En utskriven kopia kan fås för 20 kr i frimärken (max 5 kr valör). Icke medlemmar kan få skivan för 50 kr och utskriften för 50 kr.

Du kan få en kopia av CALCULATOR-programmet (6 minnen, 6 fönster, 34 funktioner, 14 siffrors noggrannhet) av Jim Peterson, Tigercub, som fanns publicerat i Micropendium jan 1992. Sänd en skiva och frankerat svars-kuvert till redaktören. Du får följande tre filer (TI-99/4A XB):

```
 Filename  Size  Type    Rec
---------- ---- ------- ---
CALC/COMM    99 Int/Var 254
CALC/DOCS    50 Dis/Var  80
CALCULATOR   60 Int/Var 254
```
■

FEST WEST FEB 1992

by Frank Aylstock, Bug News, USA

I went to the FEST WEST in Arizona.

ESD was there but did not have a working model. They are now stating the new HFDC will be ready april 15. This I have to see. They have changed again and now are to have control of 2 floppies and 2 IDE hard drives.

The ACCELERATOR REP was there and said that it had been set aside while they get the MEMEX for the 99/4A with 4 Meg. of RAM (not battery backed) finished and shipping. This MEMEX should be ready about april. The unit uses SIMMS instead of Ram chips. ■

SÄLJES: Geneve 9640, Expansionsbox + TI controller, TI-99/4A + kablage, 360 k diskdrive, 2x90 k diskdrive, IBM AT tangentbord, Myword, Multiplan 1.0 m.fl. program
Allt i ett pris: Skr 5000:-
Thomas Kolk, tel 08-732 4638

SÄLJES: TI-99/4A med expansionsbox, 32 kb-minne, diskkontrollkort och en drive. Ge ett bud!
Arne Wennberg tel 0418-22145

Redaktör: Jan Alexandersson
Medlemsregister: Claes Schibler
Tryckning av tidning: Åke Olsson
Programbankir: Börje Häll

Föreningens adress:
Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 9 NB
S-121 46 JOHANNESHOV, Sverige

Postgiro 19 83 00-6
Medlemsavgiften för 1992 är 120:-

Datainspektionens licens-nr 82100488

Annonser, insatta av enskild medlem (ej företag), som gäller försäljning av moduler eller andra tillbehör i enstaka exemplar är gratis.

Övriga annonser kostar 200 kr för hel sida. Föreningen förbehåller sig rätten att avböja annonser.

Föreningens tillbehörsförsäljning: Följande tillbehör finns att köpa genom att motsvarande belopp insätts på postgiro 19 83 00-6 (porto ingår)

Användartips med Mini Memory	20:-
Nittinian T-tröja	40:-
99er mag. 12/82, 1-5,7-9/83(st)	40:-
Nittinian årgång 1983	50:-
Programbiten 84-89 (per årgång)	50:-
90-91 (per årgång)	80:-
TI-Forth manual	100:-
Hel diskett ur programbanken(st)	30:-

Enstaka program 5:- st + startkostnad 15 kr per skiva eller kassett (1 program=20kr, 3 program=30 kr). Se listor i PB89-3 och PB90-4.

Artiklar sändes till redaktören:
Jan Alexandersson
Springarvägen 5, 3tr
142 61 TRÅNGSUND
Tel. 08-771 0569
För år 1992 planeras fyra nummer.

FÖRENINGEN PROGRAMBITEN
ÅRSBERÄTTELSE 1992-03-14
STYRELSENS ÅRSREDOVISNING

Verksamheten 1991 har bedrivits i form av tidningsutgivning och programförmedling. Antalet medlemmar har minskat från 66 till 56.

STYRELSE OCH FUNKTIONÄRER

Föreningens styrelse:
Claes Schibler, ordf. och registerf.
John Hanssen, kassör
Åke Olsson, sekreterare
Jan Alexandersson, redaktör
Börje Häll, programbankir
Peter Olsson

Under året har 3 protokollförda styrelsemöten hållits.

FÖRENINGENS EKONOMI

Årets förlust uppgår till 1637.61 kr.

FÖRENINGENS VERKSAMHET

Sex nummer av tidningen Programbiten har framställts.

Flexskivor med textfiler och program har kommit från Ottawa UG(Fast XB), Jim Peterson(Tigercub), Jim Swedlow, Tony McGovern (Funnelweb), Alexander Hulpke, Frank Aylstock.

Arvode till funktionärer har ej utgått.

BALANSRÄKNING

INGÅENDE BALANS 1991

TILLGÅNGAR		SKULDER	
POSTGIRO	12578.98	LEV.SKULDER	0.00
VARULAGER	0.00	FÖRBET.MEDL.AVG.	720.00
SKATTEFORDRINGAR	0.00	EGET KAPITAL	11858.98
SUMMA	12578.98	SUMMA	12578.98

UTGÅENDE BALANS 1991

TILLGÅNGAR		SKULDER	
POSTGIRO	10341.37	LEVERANTÖRSSKULDER	0.00
VARULAGER	0.00	FÖRBET.MEDL.AVG.	120.00
SKATTEFORDR.	0.00	EGET KAPITAL	10221.37
SUMMA	10341.37	SUMMA	10341.37

RESULTATRÄKNING

INTÄKTER		KOSTNADER	
MEDLEMSAVGIFTER	6720.00	TRYCKKOSTNADER	6442.00
PROGRAMFÖRSÄLJN.	450.00	PROGRAMINKÖP	345.00
RÄNTOR	398.49	LOKALHYROR	50.00
ÖVERSKJUT.SKATT	960.00	FÖRBR.MAT/PORTO	543.50
ÅRETS FÖRLUST	1637.61	PORTO/KUVERT	2785.60
SUMMA	10166.10	SUMMA	10166.10

STOCKHOLM SOM OVAN

CLAES SCHIBLER	JOHN HANSSEN	ÅKE OLSSON
JAN ALEXANDERSSON	BÖRJE HÄLL	PETER OLSSON

SWEDLOW TI BITS * 17-19 *

by Jim Swedlow, USA

(This article originally appeared in the User Group of Orange County, California ROM)

TIGERCUB

I do not usually promote commercial software in this column but this time I am making an exception. Jim Peterson has been a major contributor to the TI community over the years. His material is first rate. Jim has helped many 4A owners (including yours truly).

There are 130 programs in the Tigercub catalog for $1.00 each (plus $1.50 per order for media). Catalogs are $1, which is refundable from the first order.

There are four disks in the Tips from Tigercub series at $10 each. The three Nuts and Bolts are $15 each. These prices are postpaid.

All of these disks are full of ideas, programs, suggestions and tutorials. You will NOT be disappointed. Novice or experienced, his programs will be of use.

He can be reached at:
Tigercub Software
156 Collingwood Avenue
Columbus, OH 43213, USA

AVATEX 1200hc MODEM

I picked up one of these 1200 baud modems. It is a good modem that works well with the TI. The cable is straight through with pins 2 and 3 switched:

```
   TI--MODEM
    1-----1
    2-----3
    3-----2
    5-----5
    6-----6
    7-----7
    8-----8
   20----20
```

Set the DIP switches as suggested in the manual.

This information applies only to the 1200hc. Other Avatex modems require different cables.

OUCH

Every once in a while I goof and someone catches me. This time it was in TI BITS Number 16 in which I said that you can't chain Include File (.IF) commands in TI Writer.

That is what the manual says and I believed the written word. Not so, says UGOC Pres Bob Harper. He is correct. You can end each file with a .IF command for the next file. I still prefer, however, a master file that has all of the .IF commands.

FUNNELWEB VERSION 4.10 IS OUT

It has some nice new features. The configuration program has been completely rewritten. It uses overlapping windows (just like that computer we don't talk about). You can change more items including four menu choices on each main menu (in version 4.0 you could only change one each).

The McGoverns have added some new keys to the TI Writer Editor. <CTRL Q> and <CTRL A> perform a ROLL UP and ROLL DOWN. <CTRL ;> changes whatever is under the cursor to lower case and <CTRL -> to uppercase.

If you start typing with ALPHA LOCK on, just move your cursor back to the first letter and press <CTRL ;>. The letter will be changed to lower case. Better yet, it is a repeater, so if you keep those keys down, everything you typed will be converted. Very nice.

TELCO 2.1 IS ALSO OUT

Among other things, it now supports

Compuserve B transfer protocol and most parallel and serial printer ports. One feature (not new) that it has is the ability to send an initialization string to your modem. The distribution version has this string:

```
   ATS0=0!
```

This turns off your modem's auto answer feature.

The above string and the following information apply to Hayes modems and true compatibles. Check your modem's handbook before doing anything.

You may want to add to this initialization string. Here is what I use:

```
   ATS0=0 S7=60 S11=55 V1 X4!
```

What does all this mean? Glad you asked.

AT tells your modem that you want to talk to it (as in ATtention).

S0=0 turns off auto answer. Very handy if you have call waiting.

S7=60 changes how long your modem waits for a carrier from the other modem (in seconds). The default is 30 seconds.

S11=55 sets the time between tones when you are dialing (in milliseconds). The default is 70 so this is faster.

V1 tells your modem to display its messages in words (rather than code numbers).

X4 tells your modem to use its full message set rather than an abbreviated one.

! is a carriage return. This tells the modem that it is the end of the command string.

You may find that you needs vary (these work with my Avatex 1200hc) but there are things you can do to control your modem. Experiment.

FILE NAMES

The following applies to TI Controllers ONLY. No guarantees about MYARC or CORCOMP.

The Disk Controller book says that TI file names can contain any character between ASC 32 and 95 except space and period. Having seen other characters used, I decided to test this. I wrote a simple program to open a file, print something, close the file, open it again, read the text, close it and then delete the file:

```
100 FOR I=0 TO 255 :: ON ERROR 190
110 OPEN #1:"DSK1."&CHR$(I)
120 PRINT #1:STR$(I)
130 CLOSE #1
140 OPEN #1:"DSK1."&CHR$(I)
150 INPUT #1:A$
160 IF A$<>STR$(I) THEN PRINT
    "BAD READ IN";I
170 CLOSE #1:DELETE
180 NEXT I :: STOP
190 ON ERROR 210
200 CLOSE #1:DELETE
210 PRINT "FILE ERROR IN";I
220 RETURN 180
```

Note line 170: CLOSE #1:DELETE. The DELETE command causes your disk controller to delete the file after it is closed. This was necessary as your TI will only allow 127 files per disk and if I didn't delete the files, the limit would have been reached.

So what were the unacceptable file names? Everything over 127 bombed out as did 0, 32 (space) and 46 (period). Everything else worked, including lower case.

The TI Manual recommends against using lower case letters in a file name. You can, but there is a danger of saving a file as "DSK1.myfile", trying to read it as "DSK1.MYFILE" and not finding it. These are two different names to your disk controller.

The point is that everything below 128 (except 0, 32 and 46) can be used in a file name (with a TI controller, anyway).

FUNNELWEB VERSION 4.11

Another update to Funnelweb (Version 4.11) reached us recently. There are two major changes.

The first one lets you opt to have the User List menu appear first when you load Funnelweb from Editor Assembler. This will let you use Funnelweb as a user defined menu system.

With this off, when you boot Funnelweb from Editor Assembler, the first menu you get is the Editor/Assembler Menu, which always has options for Editor and Assembler. By activating it, you could use Funnelweb as a menu system for a disk of programs of your choice. There are other programs out there, like BOOT, that also serve this function.

This feature can be activated only through the Configuration program. It is called "UL Immediate".

The other major change is a complete rework of DM 1000. Tony McGovern said that the change was fixing a major bug. In fact, the program has been significantly enhanced. Although it is still called Version 3.5, it is very different from the Version 3.5 that came with Funnelweb Version 4.10. Among the changes are:

Option 1 in File Utilities and in Disk Utilities are now identical. You can print a directory AND copy / move / delete / rename / etc. files. The other changes are in this function.

DM 1000 now recognizes lower case letters. Before, if you pressed <c> instead of <C>, nothing happened. Now it works.

If you press the letter <A>, all the <N>'s are replaced with <C>'s. Handy for coping all files on a disk.

When you press <FCTN 6> your command starts immediately. No more pressing <Y> to get it going.

If there is more than one screen of files, when you get to the bottom or top of one, DM 1000 now automatically moves to the next screen.

When you Type a file (that is, display it on your screen), DM 1000 now returns to the file listing. Before, you had to do a new directory after typing a file.

There is one known bug in this revision to DM 1000. If you change the printer name (using <FCTN 3> from the main menu) and then save the change to disk, your computer will lock up next time you try and load DM 1000. You can change the printer name with a sector editor (it is in first sector in the MG file).

All in all, this version of DM 1000 is a much nicer program to use.

TI WRITER FOOTERS AND HEADERS

I have been working on a TI Writer Reference Guide. Many of us use TI Writer but do not have the Manual. This guide describes all of the Editor Keys, Editor Commands and Formatter Commands. It should be out by the time this is printed. Here are a few things I found about Footers and Headers.

Most of user do not use Footers and Headers in normal work. However, they can be quite helpful on multipage documents.

The basic format is:

```
 .FO t  or
 .HE t
```

Where "t" stands for the actual text. A percent sign (%) in a Header or Footer will be replaced with the current page number. For example:

```
 .FO Page %
```

Will print "Page" and the page number on the bottom line of each page. All Headers and Footers start in column 1 and ignore any margin settings you have made. If you are using a left margin of 5, <.LM 5>, then you need to put five spaces in front of your text:

YAPP VERSION 1.20 FOR 9938

by Alexander Hulpke, Germany

Newly included was the approximation by Bernstein Polynomials (Sergej N. Bernshtejn introduced these polynomials 1912), they allow uniform continuous approximation of continuous functions by polynomials, see [1])

It soon was found, that these polynomials allow the sketching of curves useful in the computer graphics, for example the well-known B'ezier cubics are in fact Bernstein Polynomials of degree 4. The idea to implement these curves in YAPP was inspired by [2].

A Bernstein Polynomial with n supporting points (which are points on the screen) is a mapping f:IR (the set of the real numbers)->IR^2, defined by: (in TeX notation)
f(t)=\sum_{i=1}^n z_i \choose{n}{i} t^{i-1} (1-t)^{n-i}
where each z_i is a 2-coordinate vector.

To draw such a polynomial in YAPP, type 'B' or select the appropriate icon from the color bar. Then select the supporting points. They are (due to speed reasons) shown as points, if you select the crossed lines cursor by pressing CTRL-L, they are connected by lines. If you select a pixel twice in series (it may be helpful to lift the mouse before doing it to avoid unwanted movement) the computer will start to calculate the curve.

Unfortunately even the Geneve is not the fastest computer and the built in 'RADIX100' floating point routines from TI's ROM are very slow, so this may take some time. If you think this too slow, you're welcomed to write new floating point routines. Always remember, that for n selected points a polynomial of degree n has to be evaluated about 2000 times!

It's difficult to describe the look of these Polynomials exactly, I assume the best way to get accustomed to them is to play around with them. You should note, that it can be very helpful to divide a large curve in several smaller parts, since this increases calculation speed immensely.

Literature:
[1] Jongen, Triesch, Analysis II, Skript zur Vorlesung, Teil 2, Aachen, 1989
[2] Donald E. Knuth The METAFONTbook, Reading, 1990
I hope this helps,

Other references:
Micropendium dec 90: YAPP, review
PB 91-1: YAPP, review (in Swedish)
PB 91-3: YAPP version 1.10 ■

```
 .FO      Page %
```

Headers and Footers will print the text that follows the <.HE> or <.FO> on the same line. If you have your Editor margins set at 1 and 38 (so that you can see everything), a long Header might look like this:

```
 .HE User Group of Orange County
  August, 1988 ROM, Page %
```

The Formatter will print only "User Group of Orange County August," as the Header. The other line, "1988 ROM, Page %" will print as a regular line of your document.

If you use Begin Page, <.BP>, to start a page, all Headers and Footers must be reset as <.BP> cancels Headers and Footers. Another way to cancel a Header or Footer is the command without any text (<.HE> or <.FO>).

Headers and Footers do not reduce the number of lines you can print per page. They print on the lines that the Formatter skips for the top and bottom page margins (which you cannot change in TI Writer).

Enjoy. ■

GRAPHIC2 FÖR ASSEMBLER INITIERAD FÖR TEXT

av Jan Alexandersson

GRAPHIC2 FÖR TI-99/4A

Graphic2 delar upp skärmen i rutor med 24 rader och 32 kolumner. Dessa 24 rader delas i tre delar med vardera 8 rader. Varje sådan tredjedel har sin egen uppsättning av 256 olika grafiska tecken. Det finns således totalt 3x256=768 olika tecken så att varje skärmposition kan ha sitt eget unika tecken. Genom att initiera skärmtabellen med alla tecken i nummerföljd så kan man sedan använda skärmen för grafik. Teckendefinitionen för respektive tecken ändras beroende på hur du ritar din grafik. Mer om detta i kommande nummer av Programbiten.

I fortsättningen ska vi använda Graphic2 för text genom att initiera skärmtabellen med blanktecken (ASCII 32) och initiera teckentabellen med vanliga bokstäver ASCII 32-127. Eftersom skärmen är delad i tre oberoende delar så måste detta göras tre gånger. Varje teckenruta består av 8x8 punkter. Varje pixelrad i ett tecken kan ha två färger (8 punkter delar på två färger) så att ett helt tecken kan ha 16 olika färger. Dessutom kan man ha upp till 32 olika sprites som lägges ovanpå grafikskärmen men auto-motion kan inte användas med Graphic2.

TI-FORTH använder Graphic2 i Split Mode så att en del av skärmen initieras för text och resten för grafik. SPLIT använder de övre 16 raderna för grafik och de nedre 8 raderna för text. SPLIT2 har de 4 övre raderna för text och de nedre 20 raderna för grafik. SPLIT2 fås genom att den första tredjedelen av skärmen delas så att rad 1-4 initieras med med blanktecken (ASCII 32) för 4 rader och teckentabellen initieras med bokstäver för ASCII 32-127. Skärmtabellen för rad 5-8 initieras med ASCII 128-255 i nummerordning så att ett unikt tecken finns för varje skärmposition.

Du måste initiera videoprocessorns 8 olika register (#0-#7) för att kunna använda Graphic2 på följande sätt:

#0	>02	binärt 0000 0010
#1	>E0	binärt 1110 0000
#2	värdet x >400	= Screen Table
#3	värdet x >40	= Color Table *
#4	värdet x >800	= Pattern Table **
#5	värdet x >80	= Sprite Attribute
#6	värdet x >800	= Sprite Pattern
#7	högra hexdelen	= Screen Color

(*) Color Table ska ha de sju minst signifikanta bitarna satta till "1" så att >7F adderas till adressen som endast får ha värdet >00 eller >80.

(**) Pattern Table ska ha de två minst signifikanta bitarna satta till "1" så att >03 adderas till adressen som endast får ha värdet >00 eller >04.

TABELL 99/4A	REG	BYTES	PLATSER
SCREEN	2	768	16
COLOR	3	6144	2
PATTERN	4	6144	2
SPRITE ATTRIBUTE	5	128	128
SPRITE PATTERN	6	1024	8
TOTALT		14208	1

```
***********************************
* Graphic2 DEMO PROGRAM 1 for 99/4A
* by Jan Alexandersson
* 1992-02-09
***********************************

       DEF  START

       REF  VMBW,VSBW,VWTR,GPLLNK
       REF  KSCAN,GPLWS

WS     BSS  >20
RTN    DATA 0

START  MOV  R11,@RTN     save return
       LWPI WS
       BL   @G24AT       graphic2/text

****** print message on the screen
       LI   R3,PART
       LI   R0,8+SCRTA2
       LI   R1,TEXT
       LI   R2,16
```

```
LOOP   BLWP @VMBW
       AI   R0,>100
       DEC  R3
       JNE  LOOP

****** wait for key before return
       LI   R0,>2000
KEYLP  BLWP @KSCAN       key scan
       MOVB @STATUS,R1
       COC  R0,R1        test EQ bit
       JNE  KEYLP

****** restore graphic1
       BL   @G14A

       CLR  R1
       MOVB R1,@STATUS
       LWPI GPLWS
       MOV  @RTN,R11
       RT                return

TEXT   TEXT 'Test of GRAPHIC2'
       EVEN

       COPY "DSK2.G2-TAB-4AT"
       COPY "DSK2.G24AT"
       COPY "DSK2.BLKVDP"      PB91-6
       COPY "DSK2.G1-TAB-4A"   PB91-6
       COPY "DSK2.G14A"        PB91-6

       END


*****************************
* G2-TAB-4A tables for 99/4A
* in Graphic2 mode with text
*****************************

PART   EQU  3            3 parts of scr
SCREE2 EQU  6
SCREN2 EQU  24*32
PATTE2 EQU  0
PATEN2 EQU  >800*PART
COLOR2 EQU  >80
COLEN2 EQU  >800*PART
ATTRI2 EQU  >36
SCRCO2 EQU  3            green screen
COLCH2 EQU  >1300        black on green

SCRTA2 EQU  SCREE2*>400
PATTA2 EQU  PATTE2*>800
COLTA2 EQU  COLOR2*>40
ATTAD2 EQU  ATTRI2*>80


***************************
* G24AT initialize
* Graphic2 for 99/4A
* for showing text
* by Jan Alexandersson
* 1992-02-09
* remove the stars before
* the base address for 9938
***************************

*FAC    EQU  >834A       in G14A
*STATUS EQU  >837C          "
*KEYBRD EQU  >8374          "

G24AT  MOV  R11,R10      save return
       LI   R0,>01A0     screen off
       BLWP @VWTR
       LI   R0,>0002     Graphic2
       BLWP @VWTR
****** screen table
       LI   R0,>0200+SCREE2
       BLWP @VWTR
****** colour table
       LI   R0,>0300+COLOR2+>7F
       BLWP @VWTR
****** pattern table
       LI   R0,>0400+PATTE2+>03
       BLWP @VWTR
****** sprite attribute table
       LI   R0,>0500+ATTRI2
       BLWP @VWTR
****** screen colour
       LI   R0,>0700+SCRCO2
       BLWP @VWTR
****** standard keyboard = 0
       CLR  R0
       MOVB R0,@KEYBRD

***** colour table initialize
       BL   @BLKVDP
       DATA COLTA2,COLCH2,COLEN2

****** clear pattern table
       BL   @BLKVDP
       DATA PATTA2,0,PATEN2

****** normal char 32-95
       LI   R3,PART
       LI   R0,PATTA2+>100
       CLR  R1
NCHLP  MOV  R0,@FAC
       MOVB R1,@STATUS
       BLWP @GPLLNK
       DATA >18
       AI   R0,>800
       DEC  R3
       JNE  NCHLP

****** lower char 96-127
       LI   R3,PART
       LI   R0,PATTA2+>300
LCHLP  MOV  R0,@FAC
       MOVB R1,@STATUS
       BLWP @GPLLNK
       DATA >4A
       AI   R0,>800
       DEC  R3
       JNE  LCHLP
```

```
****** new base address for 9938
****** remove the stars (*) for 9938
****** LI   R0,>0E00+BASE+2
****** BLWP @VWTR

****** D0 removes standing sprites
       LI   R0,ATTAD2
       LI   R1,>D000
       BLWP @VSBW

****** new base address for 9938
****** remove the stars (*) for 9938
****** LI   R0,>0E00+BASE+1
****** BLWP @VWTR

****** clear screen table for text
       BL   @BLKVDP
       DATA SCRTA2,>2000,SCREN2

       LI   R0,>01E0     screen on
       BLWP @VWTR
       SWPB R0
       MOVB R0,@>83D4    for KSCAN

       B    *R10         return
```

9938 FÖR 80-KOLUMNSKORT

Graphic2 med 9938 ger möjlighet till 24 eller 26,5 rader. För att klara 26,5 rader så måste skärmen delas i fyra delar, rad 1-8, 9-16, 17-24, 25-27. Varje sådan del använder en unik del av färgtabellen och teckentabellen. De 16 olika färgerna kan väljas ur en palett på 512 färger.

Med videoprocessorn 9938 ska alla VDP-register #0 - #7 initieras som för TI-99/4A ovan. Även VDP-register #1 bör initieras med >E0 även om manualen till 9938 anger >06. Detta är nödvändigt om programmet även ska kunna köras med 9918A/9929A. Dessutom bör följande register initieras:

#8	>88 aktiverar musen
#9	>00 ger 60 Hz skärmväxling
#10	Extended Color Table
#11	Extended Sprite Attribute Table
#14	värdet x >4000 är Base Address

VDP-register #10 och #11 innehåller de mest signifikanta bitarna för adressen till respektive tabell eftersom det behövs 2 bytes för att skriva önskat värde.

TABELL 9938	REG	BYTES	PLATSER
SCREEN	2	864	128
COLOR	10+3	8192	32
PATTERN	4	8192	32
SPRITE ATTRIBUTE	11+5	128	1024
SPRITE PATTERN	6	1024	64
TOTALT		18400	4-7
BASE	14	16384	8

```
**********************************
* Graphic2 DEMO PROGRAM 2 for 9938
* by Jan Alexandersson
* 1992-02-09
**********************************

       DEF  START

       REF  VMBW,VSBW,VWTR,GPLLNK
       REF  KSCAN,GPLWS

FAC    EQU  >834A
STATUS EQU  >837C
KEYBRD EQU  >8374

       COPY "DSK2.G2-TAB-38T"
       COPY "DSK2.G238T"
       COPY "DSK2.G24AT-REV" revised
****** remove the stars (*) in G24AT
       COPY "DSK2.BLKVDP"     PB91-6
       EVEN

WS     BSS  >20
RTN    DATA 0

START  MOV  R11,@RTN    save return
       LWPI WS
       BL   @G238T      graphic2/text

****** print message on the screen
       LI   R3,PART
       LI   R0,SCRTA2+6
       LI   R1,TEXT
       LI   R2,20
LOOP   BLWP @VMBW
       AI   R0,>100
       DEC  R3
       JNE  LOOP

****** wait for key before return
       LI   R0,>2000
KEYLP  BLWP @KSCAN      key scan
       MOVB @STATUS,R1
       COC  R0,R1       test EQ bit
       JNE  KEYLP

****** restore to VDP RAM 0-16383
VDPID  EQU  >8C06       indirect data
       LI   R0,>0E00    base address
       BLWP @VWTR
       LI   R0,>1100    #17 at #0
       BLWP @VWTR
       LI   R0,RSTORE
       LI   R1,12
RSTLP  MOVB *R0+,@VDPID restore reg
       DEC  R1
       JNE  RSTLP

       CLR  R1
       MOVB R1,@STATUS
       LWPI GPLWS
       MOV  @RTN,R11
       RT             return

TEXT   TEXT 'Graphic2 - 26.5 rows'

RSTORE DATA >00E0,>000E,>0106  #0-#5
       DATA >00F5,>8800,>0000  #6-#11

       END


****************************
* G2-TAB-38T tables for 9938
* in Graphic2 mode with text
* 212 pixel lines
****************************

PART   EQU  4
SCREE2 EQU  6+32
SCREN2 EQU  27*32
PATTE2 EQU  0+8
PATEN2 EQU  >800*PART
COLOR2 EQU  >80
COLEN2 EQU  >800*PART
ATTRI2 EQU  >36+>80
SCRCO2 EQU  3          green screen
COLCH2 EQU  >1300      black on green

SCRTA2 EQU  SCREE2-32*>400
PATTA2 EQU  PATTE2-8*>800
COLTA2 EQU  COLOR2*>40
ATTAD2 EQU  ATTRI2->80*>80

COLEXT EQU  1
ATTEXT EQU  1
BASE   EQU  1


***************************
* G238T initialize for 9938
* of Graphic2 with text
* by Jan Alexandersson
* 1992-02-09
***************************

G238T  MOV  R11,R9       save return
****** base address
       LI   R0,>0E00+BASE
       BLWP @VWTR
       BL   @G24AT       99/4A init
       LI   R0,>01A0     screen off
       BLWP @VWTR
       LI   R0,>0888     mouse enable
       BLWP @VWTR
       LI   R0,>0980     212pix,60Hz
       BLWP @VWTR
****** extended color table
       LI   R0,>0A00+COLEXT
       BLWP @VWTR
****** extended attribute table
       LI   R0,>0B00+ATTEXT
       BLWP @VWTR
       LI   R0,>01E0     screen on
       BLWP @VWTR
       B    *R9          return     ■
```

AMORTERING AV LAN

by David Brader, USA

```
70 REM AMORTIZATION SCHEDULE
80 REM 99'ER 83-06.1 XB
90 CALL CLEAR
100 PRINT "AMORTIZATION SCHE
DULE": :
110 INPUT "LOAN AMOUNT? ":LO
AN :: INPUT "NO. OF MONTHLY
PAYMENTS? ":N
120 INPUT "ANNUAL INTERST RA
TE %? ":IN :: PRINT
130 IN=IN/1200
140 PMT=LOAN*(IN/(1-(1+IN)^(
-N)))
150 TOT=INT(PMT*N*100)/100
160 PAY=INT(PMT*100)/100
170 LASTP=TOT-PAY*(N-1)
180 PRINT USING "PERIODIC PA
YMENT= $####.##":PAY
190 PRINT USING "FINAL PAYME
NT=    $####.##":LASTP :: PR
INT
200 INPUT "SHOW SCHEDULE FRO
M PERIOD? ":STRT :: INPUT "T
O PERIOD? ":STP
210 K=(1+IN)^(-(STRT-1)):: L
=1/K*(PMT*(K-1)/IN+LOAN)
220 IF STP=N THEN FLG=1 :: S
TP=N-1 ELSE FLG=0
230 FOR Z=STRT TO STP
240 K=(1+IN)^(-Z):: BAL=1/K*
(PMT*(K-1)/IN+LOAN)
250 PRINT : :"PERIOD";Z
260 I=BAL-L+PAY :: L=BAL ::
PR=PAY-I
270 IMAGE INT=$####.## PRIN=
$####.##
280 PRINT USING 270:I,PR
290 IMAGE BALANCE = $######.
##
300 PRINT USING 290:BAL
310 NEXT Z
320 IF FLG=0 THEN END
330 PRINT : :"PERIOD";N
340 I=LASTP-BAL :: PR=BAL ::
 BAL=0
350 PRINT USING 270:I,PR
360 PRINT USING 290:BAL        ■
```

MINA UPPTÄCKTER I LOGO-SPRÅKET, DEL 1

av Kent Edgardh

1. INLEDNING

Är det fler än jag som skaffat LOGO-modulen? Jobbat lite med den och sedan låtit den bli liggande utan att ha fått någon egentlig insikt om språkets natur.

Det är väl bland de billigaste programspråken som gick att hitta, om man nu bortser från att 32k minnesutbyggnad är nödvändig (LOGO säljs fortfarande av TEX-COMP i USA, Red. Anm.). Du kan använda LOGO med kassettbandspelare eller flexskiveenhet. Hela språket finns i insticksmodulen med editor, interpretator och kommunikation med massminne för att kunna spara program och data. Det medföljande kassettbandet och flexskivan innehåller endast exempel på program så det finns ingen del av själva språket som behöver laddas in före användning (till skillnad mot vad som gäller Editor/Assembler, TI-Writer och FORTH, Red. Anm.).

LOGO är ett språk av listtyp och är av samma språkgrupp som LISP och PROLOG samt i viss mån även APL, till skillnad från språkgruppen beräkningsspråk modell BASIC och PASCAL. Jag hade tänkt att göra vissa enklare jämförelser med BASIC och X-BASIC. Jag tycker man borde kunna utgå ifrån att de flesta är bekant med det mest elementära i de språken. Därför skulle en sådan jämförelse kunna bli den värdefullaste.

När man tittat igenom manualen och det lite tunnare övningshäftet som följde med LOGO-paketet slår det mig att det här handlar ju nästan uteslutande om grafik och vad en så kallad skjöldpadda kan tänkas göra för spår på en sandstrand, den så kallade Turtle Graphics. Det här med sprites och grafik är jag inte så värst intresserad av. Jag har ännu inte lyckats lära mig hur det går till i X-BASIC. Mest kanske för att jag saknat intresse. Det blir ju lätt så och då speciellt när det finns så mycket annat roligt att utforska och undersöka. Tiden vill ju inte räcka till. Sätter man sig ner med papper och penna vid datorn ramlar timmarna raskt iväg. Det är väl fler än jag som upptäckt det vid det här laget.

Det mesta har jag kommit underfund med genom att upprepade gånger läsa i litteratur och prova vid datorn. Mina här nedskrivna erfarenheter kan innehålla en mängd felaktigheter och missförstånd, men hör gärna av dig och berätta om vilka erfarenheter du har gjort. Vissa bitar kanske inte ens är nämnda av mig. Varför inte svara med en egen artikel i Programbiten. Ingen än jag blir gladare om och när något sådant dyker upp.

När jag började med att försöka med LOGO vid en andra omgång efter en längre tids paus, läste jag en bok som beskrev en annan LOGO-dialekt. Den var vad jag tyckte både rolig och lättläst, men åter vid datorn upptäckte jag hur mycket felaktigheter som jag hade fått in i huvudet. Det här med att varva bok- och manualläsandet med praktiskt knappandet vid datorn gällde mer än väl. Nu har jag verkligen förstått och fått reell insikt hur viktigt det är. Den insikten bör man nog själv ha gjort, då förstår man den bäst, vilket många andra i skriftlig form har varnat för.

2. PRIMITIVER

Språket innehåller ungefär 119 editerings- och kommandoprimitiver. Detta beroende lite på hur man räknar. Jag gjorde det lätt för mig genom att räkna igenom dem i appendix B i manualen. Detta utan att ta ställning till om t.ex. primitiven FIRST är den samma som om den opereras på ett ord eller en lista. Det här vad som menas med ord och lista finns det all anledning att återkomma till.

Av dessa primitiver kan knappt hälften, 58 stycken, klassificeras som grafik-primitiver. Återstår alltså 61 stycken att lära sig att hålla reda på. Den längsta finns bland grafik-primitiverna och heter COLORBACKGROUND. 15 stycken tecken som synes. Med tanke på att skärmen bara har 30 teckens vidd för eget instruktionsskrivande per rad, tycker jag att halva utrymmet är i det längsta laget. Den längsta bland icke grafik-primitiver är DIFFERENCE, 10 stycken tecken. Denna har dock ett alternativ nämligen -. Mer om detta senare. Kortaste primitiven är 2 tecken lång. Av denna längd finns det 11 stycken.

Nu har inte det här med längden av primitiverna så stor betydelse. Man kan nämligen göra alternativa tilllägg till de flesta primitiver. Ja, hela LOGO:s språkidé går ut på att just göra sådana egna tillägg. Det har jag gjort för att få de som jag använder ofta kortare. Det går då att få in flera kommandon på sina skärmrader utan att så ofta använda multipla skärmrader.

Fullt möjligt är att sätta in t.ex. svenska ord istället för de engelskspråkiga. En nackdel är att det blir svårare för någon annan att veta vad som utföres. Själv kanske man glömmer bort dem vad de stod för när programkoden blev liggande på flexskivan ett tag. Även LOGO-språkets etiska karaktär blir ju naturligtvis förändrad.

Så här går det definitivt inte att göra i varken BASIC eller X-BASIC. Det närmaste man kan komma torde vara X-BASIC:s SUB-rutiner, men man kommer aldrig ifrån att man jämt måste dras med CALL varje gång det skall användas i ett anrop för att visa att det är ett tillägg i språket som har gjorts.

3. VARIABLER

LOGO:s variabler består av de två nivåerna enkla och sammansatta. De enkla kan delas in i 3 olika typer: binära, heltal och ord. De binära är som sig bör 2 stycken till antalet. De har värdena TRUE och FALSE. Detta bör kunna jämföras med BASIC:s och X-BASIC:s -1 och 0.

Heltalen är till antalet 2^16= 65536 stycken, 32767 positiva, 32768 negativa samt en nolla, alla i talområdet -32768 till +32767. Dessa heltal definieras cykliskt. Här kommer ett exempel på vad jag menar. Tilldela N värde genom att knappa in

```
MAKE "N 32766
PRINT :N + 1
```

Interpretatorn svarar på PRINT med

```
32767
```

Det ser ut att vara både förväntat och riktigt. Fortsätt nu med

```
PRINT :N + 2
```

Nu blir interpretatorsvaret

```
-32768
```

Det här måste man tänka på när man börjat få stora absoluta talvärden i sina beräkningar. Det blir varken felavbrott eller ens ett felmeddelande utan exikveringen går lugnt vidare.

Ord är en sammanhängande teckenuppsättning utan innehåll av blanktecknet. Om något annat tecken inte får ingå finns inte beskrivet i manualen, vilket gör det lite osäkert med en exakt definition. Det tycks finnas 28 stycken bokstäver, enbart versaler ingår i vår LOGO-modul, 10 stycken siffror, en mängd specialtecken samt ett antal småkrafs att kombinera med som man vill. Småkrafs vad är nu det för något? Jo, om man har använt C men skulle vilja använda det en gång till för något annat kan man använda C:et inskrivet i en cirkel. Den hittar man med hjälp av FCTN X. Andra tecken hittar man vid FCTN 1,2,4,5,E,S och D. Här är FCTN 5 och E blankt, utan att för den skull vara det vanliga blanktecknet som man har i den där i sitt slag i särklass största tangent man hittar längst ner på tangentbordet. Dessa speciella blanka mellanrum kan användas om man vill ha ett dubbelnamn med ett mellanrum i sig. En varning är mer än väl befogad här av naturliga skäl för missuppfattningar. Stor försiktighet med andra ord. (Undvik tecken som inte kan skrivas ut med en vanlig printer. Använd endast ASCII 32-127. Programlistningar som innehåller andra ASCII-koder kan inte publiceras i tidningen. Red. Anm.). Orden användes även för namn på variabler.

De enkla variablerna kan fogas ihop med varandra till en lista. Detta liknar mycket BASIC:s fält i form av vektorer och matriser. I BASIC måste man deklarera fält i förväg med typ och storlek och form. Detta görs i DIM-satsen. Det gäller i alla fall att man blir tvungen, när man kommer upp i ett större antal indexvariabler. Typen blir den samma i hela fältet för BASIC:en. I LOGO behövs inte detta och är heller inte möjligt. Man kan blanda olika typer av delelement inuti en och samma lista. Listan kan även inom sig innehålla andra listor som delelement. Som du ser finns en mycket större grad av frihet att själv komponera sina strukturer själv. Tal kan man som bekant operera med. Exempel på operatorer för tal är t.ex. +. Listornas väsentligaste och mest elementära operatorer är hopfogning och borttagning av delelement. Det kan man göra i listans båda ändar. Borttagning kan man dock endast göra med ett av listans element i taget. Att tilldela A en lista med element kan göras med inknappningen

```
     MAKE "A [1 2 :X [P Q]]
```

A:s lista är sammansatt av två heltal, vad som finns i variabeln X, behöver inte nödvändigtvis vara heltal samt listan [P Q]. Den sista listan har två stycken element. Båda är ord.

4.OPERATORER RESP OPERANDER

Generellt sett finns det 3 sorter av typer för operatorer. De kallas prefix, infix och suffix.

Prefix är det när operatorn står före operanden eller operanderna, om de är flera. För flera kan de vara. Ett exempel på det i BASIC är ABS(X). ABS är här operatorn av prefixtyp. X är operanden. Ett annat exempel är PRINT "Storlek = "; X+5; Operatorn är här PRINT medan den följs av två stycken operander. Även de procedurer man själv har möjlighet att tillverka i BASIC är av prefixtyp med eller utan parameterlista, där parameterlistan är operanderna. Jämför med X-BASIC:s SUB-ar eller BASIC:s DEF-satser.

Typiska infixoperatorer är + - * och /. Att använda sig av denna skrivteknik är väl den som kanske känns mest naturlig och behöver därför väl knappast ordas så mycket mer om.

Suffix kan vara svårare att hitta men fakultetsbegreppet som brukar skrivas 4! och som har värdet 1*2*3 *4=24 är utropstecknet ett av suffixtyp. Man kan även tänka sig att semikolon är en suffixoperator. I betydelsen, att efter utskriften av Storlek anger semikolonet hur nästkommande utskrift skall hamna i förhållande till vad som stod före operatorn. D.v.s. först kommer operanden Storlek, därefter operatorn ;.

Vad jag har hittat är samtliga operatorer av prefixtyp i LOGO. Vissa operatorer har även en infixvariant som tilläggsalternativ. Här kommer en tablå med dessa alternativ:

GREATER 8 6	8>6
LESS 3 7	3<7
IS 7 5	7=5
SUM 1 2	1+2
DIFFERENCE 4 2	4-2
PRODUCT 3 2	3*2
QUOTIENT 9 2	9/2

Dessa har i nu nämnd ordning det beräknade värdet TRUE, TRUE, FALSE, 3, 2, 6 och 4.

Om två operatorer av olika typer finns intill varandra prioriteras den med infix i förhållande till prefixtyp. Det här skapade i början en hel del problem för mig. Här kommer ett exempel på detta. Börja först med att tilldela listan A värde. Knappa in följande

```
     MAKE "A [1 2 3 4 ]
```

Fortsätt med detta försök till utskrift

```
     PRINT LAST :A + 10
```

Nu svarar interpretatorn med ett felmeddelande

```
     + DOESN'T LIKE [1 2 3 4 ]
     AS INPUT
```

Det kan ta ett bra tag innan man förstår vad som menas med felmeddelandet, för att sedan kunna förstå vad som är galet. Kasta om operanderna till infixoperatorn med denna inknappning

```
     PRINT 10 + LAST :A
```

Svaret från interpretatorn blir nu 14, vilket är vad man vill ha ut av operationen. Man kan även tänka sig att man inför parenteser i det första exemplet. Då blir man också av med felmeddelandet

```
    PRINT (LAST :A) + 10
```

Hur ska man nu förklara att den första varianten inte fungerar men väl den andra? +, som har en högsta prioritet i printuttrycket, skall ha två operander på var sin sida om sig. Båda skall vara av typen enkel och heltal. I det första exemplet är :A och 10 båda operander, men den ena är av fel typ. I exempel 2 är operatorn omgiven av en annan operator nämligen LAST. Man kan tänka sig att interpretatorn fortsätter till vänster och tar hjälp av :A. LAST :A är en operand, med värdet 4 och dessutom av rätt typ. Har man att göra med den här typen av felutskrift kan man alltså pröva med att kasta om operanderna men säkrast är att sätta parenteser om operanderna, åtminstone för den till höger om infixoperatorn. ■

TIPS FROM TIGERCUB #61

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213

1 Aug. 1990

My stock of Tigercub Software catalogs is depleted and it would not pay me to reprint it. Therefore I have released all copyrighted Tigercub programs, except the Nuts & Bolts Disks, for free distribution providing that no price or copying fee is charged. All of my Tigercub programs have been added to my TI-PD library and are cataloged, by category, in Supplement #8.

My three Nuts & Bolts disks, each containing 100 or more subprograms, have been reduced to $5.00. If I run out of printed documentatiion, it will be supplied on disk.

My TI-PD library now consists of 419 disks of fairware (by author's permission only) and public domain, all arranged by category and as full as possible, provided with loaders by full program name rather than filename, Basic programs converted to XBasic, etc. The price is just $1.50 per disk(!), post paid if at least eight are ordered. TI-PD catalog #3 listing all titles and authors, is available for $1 which is deductible from the first purchase.

This little program won't do any of the fancy things that the sophisticated poster programs do, but it may do a few things they don't. First key in this fontmaker.

```
100 DISPLAY AT(3,1)ERASE ALL
:"Filename? DSK" :: ACCEPT A
T(3,14)BEEP:F$
110 OPEN #1:"DSK"&F$,OUTPUT
120 FOR J=32 TO 126 :: CALL
CHARPAT(J,C$):: CALL HEX_BIN
(C$,B$):: FOR K=1 TO 64
130 IF SEG$(B$,K,1)="0" THEN
 CH$=CH$&CHR$(32)ELSE CH$=CH
$&CHR$(42)
140 NEXT K :: PRINT #1:CH$ :
: CH$="" :: NEXT J :: CLOSE
#1 :: STOP
150 SUB HEX_BIN(H$,B$):: HX$
="0123456789ABCDEF" :: BN$="
0000X0001X0010X0011X0100X010
1X0110X0111X1000X1001X1010X1
011X1100X1101X1110X1111"
160 FOR J=LEN(H$)TO 1 STEP -
1 :: X$=SEG$(H$,J,1)
170 X=POS(HX$,X$,1)-1 :: T$=
SEG$(BN$,X*5+1,4)&T$ :: NEXT
 J :: B$=T$ :: T$="" :: SUBE
ND
```

This program reads the hex code of each character from ASCII 32 to 126, converts it to a 64-byte binary string of 0's and 1's, then changes each 0 to the blank ASCII 32 and each 1 to a printable character, and saves the result to a file of patterns to print characters 8 spaces wide by 8 spaces high.

The 42 in line 130 creates characters composed of asterisks. Change it to J and the characters will be composed of themselves - the A will be made up of A's, etc. Or, check your printer manual and substitute one of the special graphic symbols in ASCII 224 - 255.

The character patterns are designed from the hex codes in memory, so you can first merge in a reidentified char set such as a CHARA1 file or one of the fonts in my Nuts & Bolts disks or in my 127 Screen Fonts disk.

Create as many fonts as you want, then key in this poster maker program.

```
100 OPEN #1:"PIO",VARIABLE 1
36 :: PRINT #1:CHR$(27)&"@";
110 DIM CH$(94):: Q,H=1 :: W
,SP=8 :: DB$,SU$="N" :: D$,E
$="Y" :: GOTO 150
120 F$,CH$(),J,Q$,M$,FLAG,OU
T$,A$,S,SS,PC$,H,T$,L,A,X,K,
T,X$(),SK,ST,DD
130 CALL KEY :: CALL SOUND
140 !@P-
150 DISPLAY AT(3,4)ERASE ALL
:"QUICK & DIRTY POSTERS" ::
DISPLAY AT(5,7):"by Jim Pete
rson"
160 DISPLAY AT(12,1):"Font f
ile? DSK" :: ACCEPT AT(12,15
)BEEP:F$ :: ON ERROR 170 ::
GOTO 180
170 GOSUB 680 :: RETURN 160
180 OPEN #2:"DSK"&F$,INPUT :
: FOR J=1 TO 94 :: LINPUT #2
:CH$(J):: NEXT J :: CLOSE #2
 :: GOTO 190
190 DISPLAY AT(3,1)ERASE ALL
:"Load download font? Y/N N"
 :: ACCEPT AT(3,25)SIZE(-1)V
ALIDATE("YN")BEEP:Q$ :: IF Q
$="N" THEN 230
200 ON ERROR 210 :: DISPLAY
AT(3,1)ERASE ALL:"Filename?
DSK" :: ACCEPT AT(3,14):F$ :
: OPEN #2:"DSK"&F$,INPUT ::
GOTO 220
210 GOSUB 680 :: RETURN 190
220 LINPUT #2:M$ :: PRINT #1
:M$ :: IF EOF(2)<>1 THEN 220
 ELSE CLOSE #2
230 IF FLAG=1 THEN 260 :: FL
AG=1
240 ON ERROR 250 :: DISPLAY
AT(3,1)ERASE ALL:"Output fil
e? DSK" :: ACCEPT AT(3,17):O
UT$ :: GOSUB 670 :: GOTO 260
250 GOSUB 680 :: RETURN 240
260 DISPLAY AT(3,1)ERASE ALL
:"(1) PICA":"(2) ELITE":"(3)
 CONDENSED":STR$(Q):: ACCEPT
 AT(6,1)SIZE(-1)VALIDATE("12
3"):Q
270 IF Q=1 THEN S=80 :: A$=C
HR$(18):: GOSUB 640 :: GOTO
300
280 IF Q=2 THEN S=96 :: A$=C
HR$(27)&"B"&CHR$(2):: GOSUB
640 :: GOTO 300
290 S=136 :: A$=CHR$(15):: G
OSUB 640
300 DISPLAY AT(3,1):"Char wi
dth 1, 6, 7 or 8? "&STR$(W):
: ACCEPT AT(3,26)SIZE(-1)VAL
IDATE("1678")BEEP:W :: SS=IN
T(S/W)
310 DISPLAY AT(3,1)ERASE ALL
:"double width? "&DB$
320 ACCEPT AT(3,15)SIZE(-1)V
ALIDATE("YN")BEEP:DB$
330 IF DB$="Y" THEN SS=INT(S
S/2):: S=S/2 :: A$=CHR$(27)&
"W"&CHR$(1):: GOSUB 640 ELSE
 A$=CHR$(27)&"W"&CHR$(0):: G
OSUB 640
340 DISPLAY AT(3,1)ERASE ALL
:"Double-strike? "&D$ :: ACC
EPT AT(3,16)SIZE(-1)VALIDATE
("YN")BEEP:D$
350 IF D$="Y" THEN A$=CHR$(2
7)&"G" :: GOSUB 640 ELSE A$=
CHR$(27)&"H" :: GOSUB 640
360 IF Q<>1 THEN E$="N" :: G
OTO 380 ELSE DISPLAY AT(3,1)
ERASE ALL:"Emphasize? "&E$ :
: ACCEPT AT(3,12)SIZE(-1)VAL
IDATE("YN")BEEP:E$
370 IF E$="Y" THEN A$=CHR$(2
7)&"E" :: GOSUB 640 ELSE A$=
CHR$(27)&"F" :: GOSUB 640
380 IF DB$="Y" OR E$="Y" THE
N 410
390 DISPLAY AT(3,1)ERASE ALL
:"Superscript? "&SU$ :: ACCE
PT AT(3,14)SIZE(-1)VALIDATE(
"YN")BEEP:SU$
400 IF SU$="Y" THEN A$=CHR$(
27)&"S"&CHR$(0):: GOSUB 640
ELSE A$=CHR$(27)&"T" :: GOSU
B 640
410 IF W=1 THEN 430 :: DISPL
AY AT(3,1)ERASE ALL:"Spacing
? "&STR$(SP)&"  /72"
420 ACCEPT AT(3,10)SIZE(-3)V
ALIDATE(DIGIT):SP :: IF SP>1
27 THEN 420 ELSE A$=CHR$(27)
&"A"&CHR$(SP):: GOSUB 640
430 PRINT #3:PC$;:: PC$="" :
: IF W=1 THEN 450
```

```
440 DISPLAY AT(3,1)ERASE ALL
:"Multiplied height? "&STR$(
H):: ACCEPT AT(3,20)SIZE(-1)
VALIDATE(DIGIT):H
450 DISPLAY AT(12,1)ERASE AL
L:"MAXIMUM LENGTH";SS;"LETTE
RS" :: LINPUT T$ :: L=LEN(T$
):: IF L>SS THEN 450
460 IF W>1 THEN 470 :: T$=RP
T$(" ",(SS-L)/2)&T$ :: PRINT
 #1:T$ :: GOTO 510
470 FOR J=1 TO LEN(T$):: A=A
SC(SEG$(T$,J,1))-31 :: FOR K
=1 TO 57 STEP 8 :: X=X+1 ::
X$(X)=X$(X)&SEG$(CH$(A),K,W)
:: NEXT K :: X=0 :: NEXT J
480 T=(S-L*W)/2
490 FOR J=1 TO 8 :: X$(J)=RP
T$(" ",T)&X$(J):: NEXT J
500 FOR J=1 TO 8 :: FOR K=1
TO H :: PRINT #1:X$(J):: NEX
T K :: NEXT J
510 DISPLAY AT(3,1)ERASE ALL
:"OK? Y/N Y" :: ACCEPT AT(3,
9)SIZE(-1)VALIDATE("YN")BEEP
:Q$ :: IF Q$="N" THEN 540
520 IF W=1 THEN PRINT #3:T$
:: SP=8 :: GOTO 600
530 FOR J=1 TO 8 :: FOR K=1
TO H :: PRINT #3:X$(J):: NEX
T K :: X$(J)="" :: NEXT J ::
 GOTO 600
540 FOR J=1 TO 8 :: X$(J)=""
 :: NEXT J
550 DISPLAY AT(3,1)ERASE ALL
:"(R)edo last line?":"(S)tar
t over?":"Choice? R/S R" ::
ACCEPT AT(5,13)SIZE(-1)VALID
ATE("RS")BEEP:Q$
560 IF Q$="S" THEN 590 :: GO
SUB 650
570 CLOSE #3 :: OPEN #3:"DSK
"&OUT$,INPUT
580 LINPUT #3:M$ :: PRINT #1
:M$ :: IF EOF(3)<>1 THEN 580
 ELSE CLOSE #3 :: GOSUB 670
:: GOTO 620
590 CLOSE #3:DELETE :: GOSUB
 670 :: GOTO 620
600 DISPLAY AT(3,1)ERASE ALL
:"Skip how many lines?  " ::
 ACCEPT AT(3,22)VALIDATE(DIG
IT)BEEP:SK :: FOR J=1 TO SK*
8/SP :: PRINT #1 :: PRINT #3
:" " :: NEXT J
610 DISPLAY AT(3,1)ERASE ALL
:"More? Y" :: ACCEPT AT(3,7)
SIZE(-1)VALIDATE("YN")BEEP:Q
$ :: IF Q$="N" THEN CLOSE #3
 :: STOP
620 DISPLAY AT(3,1)ERASE ALL
:"Load new font? N" :: ACCEP
T AT(3,16)SIZE(-1)VALIDATE("
YN")BEEP:Q$ :: IF Q$="Y" THE
N PRINT #1:CHR$(27)&"@" :: G
OTO 150
630 DISPLAY AT(3,1)ERASE ALL
:"Change codes? N" :: ACCEPT
 AT(3,15)SIZE(-1)VALIDATE("Y
N")BEEP:Q$ :: IF Q$="N" THEN
 450 ELSE 260
640 PRINT #1:A$;:: PC$=PC$&A
$ :: RETURN
650 DISPLAY AT(3,1)ERASE ALL
 BEEP:"Set printer to top of
 page":"and press Enter"
660 CALL KEY(0,K,ST):: IF ST
=0 THEN 660 ELSE RETURN
670 OPEN #3:"DSK"&OUT$,VARIA
BLE 136,APPEND :: RETURN
680 CALL SOUND(1000,110,0,-4
,0):: DISPLAY AT(23,1):"CANN
OT OPEN THAT FILE!" :: FOR D
D=1 TO 100 :: NEXT DD :: RET
URN
```

This program asks you for one of your font files. Next it allows you the option of downloading special characters to your printer, if you have such a file on disk Then you are asked for an output filename; this is necessary because the program rapidly uses up available string storage memory.

Then you are taken through the various printer options. You also have a character width choice of 1, 6, 7, 8. The normal screen font uses only 5 of the 8 pixels of width, so you can select a width of 6 or 7 to get more letters on a line. If your font file used a wider char set, be sure to allow for spacing. If you select 1, you will print a line in the normal printer font.

You are also asked for the line spacing, in 1/72" increments. Characters are normally 8 lines high, but you have the option to print each line multiple times for tall characters or, with closer line spacing, for denser print. Try 3/72" with superscript multiplied by 3, or 5/72" with a solid block graphic character with triple printing.

Finally, you are shown the maximum number of characters according to your options, from 5 double-width 8-wide to 22 compressed 6-wide; you input a line and see it printed. It will be automatically centered.

If you are satisfied with it, the line is saved to disk, you specify the number of lines (8/72" spacing) to skip, and you are taken thru the options (including a new font) for the next line. The previous selections become the default options, so you can skip through quickly.

If the line is not satisfactory, you have the option of advancing the paper to the next page and reprinting the poster up to that point from the disk file and then continuing.

Now, here's the neat part. When you have finished your poster, you can print as many copies as you want. Just key in this program -

```
100 OPEN #1:"PIO",VARIABLE 1
36 :: PRINT #1:CHR$(27)&"@"
110 DISPLAY AT(12,1)ERASE AL
L:"Filename? DSK" :: ACCEPT
AT(12,14)BEEP:F$ :: OPEN #2:
"DSK"&F$,INPUT
120 DISPLAY AT(12,1)ERASE AL
L:"Load a download font? Y/N
 N" :: ACCEPT AT(12,27)SIZE(
-1)VALIDATE("YN"):Q$ :: IF Q
$="N" THEN 150
130 DISPLAY AT(12,1)ERASE AL
L:"Filename? DSK" :: ACCEPT
AT(12,14)BEEP:F$ :: OPEN #3:
"DSK"&F$,INPUT
140 LINPUT #3:M$ :: PRINT #1
:M$ :: IF EOF(3)<>1 THEN 140
 ELSE CLOSE #3
150 DISPLAY AT(12,1)ERASE AL
L:"How many copies?" :: ACCE
PT AT(12,18)VALIDATE(DIGIT):
N :: FOR J=1 TO N
160 DISPLAY AT(12,1)ERASE AL
L BEEP:"position paper, pres
s Enter"
170 CALL KEY(0,K,S):: IF S=0
 THEN 170 ELSE CALL CLEAR
180 LINPUT #2:M$ :: PRINT #1
:M$ :: IF EOF(2)<>1 THEN 180
```

You'll have to reposition the paper after each one.

The poster maker program was written for my Gemini 10X and I have not tried to offer options for other printers, since I don't have them available for testing. However, I think that these are the essential changes for the Epson standard.

```
260 DISPLAY AT(3,1)ERASE ALL
:"(1) PICA":"(2) ELITE":"(3)
 COMPRESSED PICA":"(4) COMPR
ESSED ELITE":STR$(Q):: ACCEP
T AT(7,1)SIZE(-1)VALIDATE("1
234"):Q
270 IF Q=1 THEN S=80 :: A$=C
HR$(18):: GOSUB 640 :: GOTO
300
280 IF Q=2 THEN S=96 :: A$=C
HR$(27)&CHR$(77):: GOSUB 640
 :: GOTO 300
290 IF Q=3 THEN S=132 :: A$=
CHR$(15):: GOSUB 640 ELSE S=
160 :: A$=CHR$(15):: GOSUB 6
40
670 OPEN #3:"DSK"&OUT$,VARIA
BLE 160,APPEND :: RETURN
```

And these changes should make compressed elite available on the Gemini SG10 in Star mode.

```
260 DISPLAY AT(3,1)ERASE ALL
:"(1) PICA":"(2) ELITE":"(3)
 COMPRESSED PICA":"(4) COMPR
ESSED ELITE":STR$(Q):: ACCEP
T AT(7,1)SIZE(-1)VALIDATE("1
234"):Q
270 IF Q=1 THEN S=80 :: A$=C
HR$(18):: GOSUB 640 :: GOTO
300
280 IF Q=2 THEN S=96 :: A$=C
HR$(27)&"B"&CHR$(2):: GOSUB
640 :: GOTO 300
290 IF Q=3 THEN S=136 :: A$=
CHR$(15):: GOSUB 640 ELSE S=
160 :: A$=CHR$(27)&"B"&CHR$(
4):: GOSUB 640
670 OPEN #3:"DSK"&OUT$,VARIA
BLE 160,APPEND :: RETURN
```

Other modifications should be fairly easy. The variable S contains the maximum number of characters per line. In lines 310-400, the option is turned on if it is selected, turned off if it is not.

Almost out of memory,

Jim Peterson ■

PROGRAMMING MUSIC – PART 2

by Jim Peterson, Tigercub, USA

PROGRAMMING MUSIC THE EASY WAY

In Part 1 I showed you how to set up a musical scale to create notes, and how to merge in various little routines to create a variety of musical effects, but I didn't tell you how to figure out what numbers to put in between those GOSUBs. So, here is the little program that makes it all easy.

```
100 CALL CHAR(127,"000F080F0
868F870000F08080868F87000080
8080868F8700008080808689870"
):: CALL CHAR(131,"000000000
0609070")
110 CALL CHAR(132,"0000120C4
83020400000221C0810200000201
0201030200000003CFF"):: CALL
 CHAR(136,"000000FF3C")
120 CALL CLEAR :: S$="GFEDCB
A" :: CALL CHAR(45,"00000000
FF"):: A$=RPT$(S$,3):: FOR R
=2 TO 22 STEP 2 :: IF R=12 T
HEN 130 :: DISPLAY AT(R,1):R
PT$("-",28)
130 NEXT R :: CALL CHAR(98,"
0020202834242830")
140 FOR R=1 TO 21 :: DISPLAY
 AT(R,1):SEG$(A$,R,1);:: NEX
T R
150 DATA 127,127,128,128,129
,129,130,130,131,131
160 DATA 1/16,1/8,1/4,1/2,1/
1
170 FOR R=1 TO 20 STEP 2 ::
READ N :: DISPLAY AT(R,15):C
HR$(N);:: NEXT R :: FOR R=3
TO 19 STEP 4 :: DISPLAY AT(R
,16):".";:: NEXT R
180 C=132 :: FOR R=1 TO 17 S
TEP 4 :: DISPLAY AT(R,17):CH
R$(C);:: C=C+1 :: NEXT R
190 FOR R=1 TO 17 STEP 4 ::
READ M$ :: DISPLAY AT(R,20):
M$;:: NEXT R
200 DATA 35,33,32,30,28,27,2
5,23,21,20,18,16,15,13,11,9,
8,6,4,3,1
210 FOR R=1 TO 21 :: READ N
:: N$=N$&CHR$(N):: DISPLAY A
T(R,6):STR$(N);:: NEXT R
220 G$="b" :: Z=-1 :: GOSUB
320 :: IF F=0 THEN 230 ELSE
GOSUB 330 :: GOTO 240
230 G$="#" :: Z=1 :: GOSUB 3
20 :: IF F<>0 THEN GOSUB 330
240 DISPLAY AT(24,1):"Shorte
st note? 1/" :: ACCEPT AT(24
,18)VALIDATE("12468")SIZE(2)
BEEP:L :: T$="1/"&STR$(L)::
RESTORE 160 :: FOR J=1 TO 5
:: READ L$ :: IF L$=T$ THEN
260
250 NEXT J :: GOTO 240
260 DISPLAY AT(24,1):"Is it
dotted? Y/N" :: ACCEPT AT(24
,19)VALIDATE("YN")SIZE(1):D$
 :: D=1-(D$="Y")
270 T=-3+J*4
280 FOR R=T TO 19 STEP 4 ::
DISPLAY AT(R,11):STR$(D);::
DISPLAY AT(R+2,11):STR$(D*1.
5);:: D=D*2 :: NEXT R
290 GOTO 360
300 FOR R=1 TO 20 STEP 2 ::
READ N :: DISPLAY AT(R,15):C
HR$(N);:: NEXT N
310 GOTO 310
320 DISPLAY AT(24,1):"How ma
ny "&G$&" on upper scale?" :
: ACCEPT AT(24,28)VALIDATE("
01234567")SIZE(1)BEEP:F :: R
ETURN
330 Y$="" :: FOR J=1 TO F ::
 DISPLAY AT(24,1):"On which
letter?"
340 ACCEPT AT(24,18)VALIDATE
(S$)SIZE(1)BEEP:L$ :: IF POS
(Y$,L$,1)<>0 THEN 340 ELSE Y
$=Y$&L$
350 S=1 :: FOR K=1 TO 3 :: P
=POS(A$,L$,S):: DISPLAY AT(P
,2):G$;:: DISPLAY AT(P,6):ST
R$(ASC(SEG$(N$,P,1))+Z);:: S
=P+1 :: NEXT K :: NEXT J ::
RETURN
360 OPEN #1:"PIO" :: FOR R=1
 TO 22 :: FOR C=3 TO 30 :: C
ALL GCHAR(R,C,G):: CALL HCHA
R(R,C,30):: R$=R$&CHR$(G)::
NEXT C :: PRINT #1:R$ :: R$=
"" :: NEXT R :: STOP
```

Get yourself a piece of sheet music and compare it to the screen display from that program. You will see that music is written on two sets of 5 lines. The upper set is marked at the left end with something like a fancy script capital S; it is used to write the higher notes, including the melody, which a pianist plays with the right hand. The lower set, marked with a sort of a backward C, contains the low notes played with the left hand. Your sheet music probably has a wide space between the sets, to make room for the lyrics, but there are really only three notes between them.

The screen display shows letters at the left, which are not on the sheet music. Those are the names of the notes, which we will have to refer to a couple of times to get started; observe that the notes are named A through G and then repeated.

The numbers along the left side are the numbers you will key in to play those notes. However, the screen display is set up in the key of C, which is played entirely on the piano white keys. The sheet music you want to program from may be in a different key, so -
The computer is asking you how many there are of something that looks like a squashed lower case b - I guess that's why they call it a flat? It means that the note will be played a bit lower, on the black key just left of the white key - and we will program it one number lower. So, look next to that capital S and see how many flats there are. If none, type 0. Otherwise, the computer will ask which letters they are next to. Type them in, one at a time, and presto - the computer will put them on the staff and adjust the numbers accordingly.

If there were no flats, the computer will want to know if there are any sharps - those are what you get by typing a shift 3 on the keyboard, and they mean that the note is played on the black key above the white key, and is programmed one number higher.

Now, the computer needs some information in order to help you set up the length of your notes - how long they are sounded. The various notes are depicted at the right. A 1/16 note is a little black egg with a stem (it may go up or down, makes no difference) and two flags on the stem. A 1/8 has only one flag and a 1/4 note has none. A 1/2 note is a hollow egg with a stem and a whole note has no stem.

Those little doodads to the right of the notes are rests, used to indicate a silent pause of the same length as that note - more on that later.

Look through your sheet music and find the shortest note. Tell the computer. It will want to know if any of those shortest notes are dotted - have a little dot to their right, as the screen display shows. A dotted note is played half again as long as normal. Presto again, the computer will show you the duration number to key in for each note. Then, if you have a printer attached, it will print out an XBasic screen dump of that screen - you will have to squash your own b's and sketch in the notes and rests.

If your software library contains an assembly screen dump, delete that last program line and put in a CALL INIT, CALL LOAD and CALL LINK to get a better printout - or ask me for it. If you don't have a printer, why not copy those numbers right onto the corresponding lines and spaces on your sheet music, and number some of the notes.

Now we're ready to make music! Let's keep it simple at first, just a single note melody - and I hope you picked a simple piece of music. Clear the TI's brain with NEW, then merge in that line 100 scale from part 1 by MERGE DSK1.SCALE. In the same way, merge in one of those line 1000 CALL SOUND routines. Put in a temporary stopper line 999 STOP, and a line 110 D=200 to set the duration.

The melody is almost always on the upper set of 5 lines. If a note has 2 or 3 eggs on its stem, as they usually do, the upper one is the melody note - we will get into harmony later.

Start with line 110. Check your chart to see what number denotes the length of the first note - maybe 2, if so key in T=2 :: Then check to see what number applies to the position of the upper egg of that note. Maybe 22, so key in A=22 :: GOSUB

1000 Enter RUN, and if you've done everything correctly, you will hear the note. You might decide already that you want to change that 200 in line 110.

Now for the second note. If it is of the same length as the first, you don't have to type anything - that's what makes this shorthand method so quick and easy. If the note position is also the same, you don't key that in either - just another GOSUB 1000.

If you have EZ-KEYS or another "hot keys" program, you can program a control key to put in the GOSUB 1000 with just one keypress - wish I had thought of that when I was programming music by the diskfull!

So keep plugging along, keying in durations and notes. After every half dozen notes or so, type RUN to see if everything sounds OK so far - it's easier to catch errors before they are too far back in the music.

You can get up to 5 screen lines on one line number, but you might better stick to 3 lines. You will note that the sets of notes are divided by vertical bars. You might program the notes between bars on a separate line, then add a ! followed by the words of the song that go with those notes - I find that a very good way to track down sour notes.

Regarding those bars - it might help you sometime to know this. At the beginning of the music, right after the big script S and the flats and sharps, you will see something like a 3 over a 4, or a 4 over a 4, or whatever - but often a symbol such as a barred C is used instead. A 3 over a 4, for instance, means that the notes between two of those bars will add up to 3/4 - might be three quarter notes, or two eighth notes and two quarter notes, or whatever, but they will add up to 3/4. Sometimes the very first notes will add up short, but in that case the very last ones will make up the difference.

The notes between those two bars make up a bar of music, and the emphasis is on the first note - for instance, that 3/4 is the 1-2-3, 1-2-3 beat of waltz time.

While you are keying in that music, you might come to one of those rests. You can just key in its T= value and then A=0 for a silent note. However, computer notes stop so abruptly that somehow a rest just doesn't sound right, so I often just use the previous note instead.

You may come across one of those flat or sharp symbols next to a note in the music. Give the note a number 1 lower if a flat, one higher if a sharp, and the same for any subsequent occurrences of that note, until you find next to it a symbol that looks like the sharp sign with half its legs knocked off; that means to go back to normal. You might also come across that symbol to tell you to play a normally flat or sharp note as if it was not.

I think that covers all that you absolutely have to know for now, and I have horrified all serious students of music just about enough. There are all kinds of other squiggles on the sheet music but usually they are not essential in programming music.

There is one other time-saving shortcut that I should tell you about right now. Most music consists at least partly of musical phrases, of a series of notes, which are repeated two or more times within a melody. So, the first thing you should do before you start programming a song is to search through the music for such phrases.

If you find one, of more than a few notes, that is repeated elsewhere - and make sure it is repeated exactly the same - mark it off each place it occurs and label it 500. If you find a second repeating phrase, label it 600, and so on.

Then, when you start programming, start with line 500, key in that series of notes first, and end it with RETURN. If you have another phrase, put it in lines starting with 600, again ending with RETURN.

Now, start programming from the beginning of the song in line 120, but

FAST EXTENDED BASIC! 88/03 - SPRINGSONG

(c) 1989 Lucie Dorais, Ottawa TI-99/4A Users' Group, Canada

I wished to do a little column on music, but was unable to find the score for Vivaldi's "Spring"; anyway, Tex is not very good in imitating violins. So I looked into my music books and found a little piece called "Spring Song", by Schulz.

Extended Basic does not add anything to the CALL SOUND statement of BASIC, except that you could use multiple statements lines. I decided not to, because single CALLS are easier to read. And to type??? Well, what about using the REDO function? Just type the first CALL SOUND, line 270, and REDO it as you need; but don't forget to change the line number! (In the program listing, I wrote all the CALL SOUNDS as CS: you will need to type the full statement, I did that only for page setting purposes). A tip: lines 610-710 are an exact duplicate of lines 490-590, just change the line numbers when you redo them. And if you have Triton's SXB, well, just do "COPY 490-590,610,10", presto, all the lines are copied in one statement (10 is for the increment).

The nice title routine comes from our friend David Fink, who used it to fill the page of his renewal letter; thank you Dave! It is in lines 120 to 170, and came as a two-line (multiple statements) routine. You can use it with any program, but the title has to be 12 characters or less to work. To get all the letters in the same color, just give a single color value to the sprites instead of "I+5" in line 160.

I decided to put most of the values into variables at the beginning of the program (lines 190-200 are for the notes, line 210 for duration and volume). Each time Tex encounters a numeric value, it first puts it into a memory address before using it; using variables does it once only, and the program runs a bit faster and smoother.

The names for the notes variables are suggested by Raymond J. Herold in the book "TI-99/4A Sound and Graphics" (Compute!), and follow the names of the notes on a piano: C to B for middle range, LC to LB and LLA-LLB for lower ranges; upper ranges are of course HC to HB, then HHC to... whatever you want. The "S", as in "LFS", means the note is a sharp; a flat would be "F", like in "BF", but we don't use any here. The variable "DUM" stands for DUMMY in line 890 (see below). If you look at the listing, you will find some note frequencies as "40000", which I did not put into variables for aesthetic reasons (to better visualize the score): they are "mute" notes (inaudible), and serve to stop the previous CALL SOUND; they correspond to the ends of the "slurs" in the musical score, and perform a short pause in the music.

Putting the duration of the quarter note into a variable "T" makes it much easier to control the tempo; the eighth note is T/2, the sixteenth T/4. At the end of the piece, the score had some triplets (three notes in the same beat as the basic quarter note), so we have T/3. To make the tempo slower, change T to a higher value, and do the opposite to make it faster. I think 600 is very appropriate to lift our spirits out of the winter blues. TD is for the dotted eighths, equivalent to three sixteenth notes; I made it into a variable because of the long calculation, for fear of slowing down the music.

I did not play too much with the

---

when you come to one of those phrases, just put in GOSUB 500 - the program will jump to that line number, play those notes, and come right back to where it was.

In Part 3, we will get into programming in 3-part harmony, bass notes, auto-chording, and other things. ■

volumes: all the Treble clef (G), i.e. what is played with the right hand, is loud (V1=1), while the Bass (F) clef, played with the left hand, is a bit softer (V2=5). You can of course play with those values to add some "feeling" into Tex's playing.

Before leaving the score itself, a note on line 890: what does it do??? well, it plays a note which has a frequency lower than 110; in this case, the G just below it, with a frequency of 98. I got the tip (and the values) from a little tutorial program by Tom Moran, "Teach Music"; I quote the appropriate screen from this program: "These low notes are created by using noise # -4 in combination with specific frequencies. Here is the program list:

```
100 DATA 1475,1293,1227,1105,990,
    957,840,735
110 FOR I=1 TO 8 :: READ T :: CALL
    SOUND(1000,T,30,T,30,T,30,T,30,
    -4,1) :: NEXT I "
```

Now, don't ask me what these values mean! The program plays down the scale below the lowest A (110). I was able to change the first value with the note and volume I needed in Treble clef, with no audible effect.

You should know about line 220. Lines 230 to 260 start some very crude animation: three crocuses grow slowly. In line 230, we define three characters (in three sets, to get three colors) as the same flower; char. 120 is the green stem. The three flowers are then "displayed" in line 260, a much faster way than with CALL HCHARs.

To simulate the growing, we use a "mask" sprite, character 128, defined as a solid block. We need three sprites, numbered from 13 to 15 because the title use a maximum of 12 sprites. Being the same color as the screen (4), they are "invisible" but all there; being on top of the crocuses, they hide them until put in motion during the playing of the music; a velocity of one, the lowest possible, seemed most appropriate for nature's rhythm. I put the CALL MOTIONs following the "mute" notes to simplify, but you are welcome to change their location. Now, you may ask, how can a single character sprite hide flowers made of two characters vertically placed? Remember, in line 120, we magnify all the sprites to "2", which make them 4x4, so our flowers are conveniently covered.

At the end of this musical masterpiece, you are asked if you want to play it again; note that even if we display a line without a SIZE statement, the letters in the title are not erased: this is because the sprites are on top of the text screen, and are thus not affected by this statement. The three "invisible" sprites are deleted, so that they don't come down far enough to "hide" the displayed text and our full-grown flowers while Tex waits for you to press a key.

SPRING BONUS!

Did you know that you can ACCEPT strings that are longer than the screen width (28 char.)??? Easy, and they can be as long as 255 characters, while INPUT/LINPUT will take only 138 to 140. Only problem: it is always on line 24, but you MUST NOT specify it. However, you can qualify it as usual with BEEP, VALIDATE, etc. Try the following, and enter a text until Tex tells you to stop:

```
100 PRINT "INPUT:" :: INPUT A$ ::
    PRINT LEN(A$)
110 PRINT
120 PRINT "ACCEPT:" :: ACCEPT B$ ::
    PRINT LEN(B$)
```

```
100 REM SPRING SONG (SCHULZ)
 / L. Dorais, Jan. 1988
110 REM title display routin
e by Dave Fink
120 CALL CLEAR :: CALL SCREE
N(4):: CALL MAGNIFY(2):: A$=
"SPRING SONG"
130 S=(6-INT(LEN(A$)/2))*16+
32
140 R=(6-INT(LEN(A$)/2))*16+
1
150 I=1 :: FOR C=S TO ((LEN(
A$)-1)*16)+S STEP 16
160 CALL SPRITE(#I,ASC(SEG$(
A$,I,1)),I+5,R,C)
170 R=R+16 :: I=I+1 :: NEXT
C
180 DISPLAY AT(3,27):"BY": :
TAB(23);"SCHULZ"
190 LLB=123 :: LC=131 :: LD=
147 :: LFS=185 :: LG=196 ::
LB=247 :: C=262
200 D=294 :: FS=370 :: G=392
 :: A=440 :: B=494 :: HC=523
 :: HD=587 :: HE=659 :: DUM=
1475
210 T=600 :: TD=T/4*3 :: V1=
1 :: V2=5
220 GOTO 230 :: CALL CHAR ::
 CALL COLOR :: CALL SPRITE :
: CALL SOUND :: CALL MOTION
:: CALL DELSPRITE :: !@P-
230 A$="0092D6FEFE7C3810" ::
 CALL CHAR(97,A$,104,A$,112,
A$):: CALL CHAR(120,"1010109
2D6FE7C10",128,"FFFFFFFFFFFF
FFFF")
240 CALL COLOR(9,16,4,10,14,
4,11,5,4,12,13,4)
250 CALL SPRITE(#13,128,4,16
1,28,#14,128,4,161,60,#15,12
8,4,161,92)
260 DISPLAY AT(21,3):"a   h
  p":"  x   x   x"
270 CALL SOUND(TD,HD,V1)
280 CALL SOUND(T/4,B,V1)
290 CALL SOUND(T/2,G,V1,LB,V
2,LG,V2)
300 CALL SOUND(T/2,G,V1)
310 CALL SOUND(T/2,A,V1,C,V2
,LD,V2)
320 CALL SOUND(T/4,A,V1)
330 CALL SOUND(T/4,B,V1)
340 CALL SOUND(T,G,V1,LB,V2,
LG,V2)
350 CALL SOUND(T/4,40000,V1)
360 CALL MOTION(#13,1,0)
370 CALL SOUND(T/4,HD,V1)
380 CALL SOUND(T/4,B,V1)
390 CALL SOUND(T/4,HD,V1)
400 CALL SOUND(T/4,B,V1)
410 CALL SOUND(T/2,G,V1,LB,V
2,LG,V2)
420 CALL SOUND(T/2,G,V1)
430 CALL SOUND(T/2,A,V1,C,V2
,LD,V2)
440 CALL SOUND(T/4,A,V1)
450 CALL SOUND(T/4,B,V1)
460 CALL SOUND(T,G,V1,LB,V2,
LG,V2)
470 CALL SOUND(T/4,40000,V1)
480 CALL MOTION(#14,1,0)
490 CALL SOUND(T/4,FS,V1)
500 CALL SOUND(T/4,G,V1)
510 CALL SOUND(T/4,A,V1)
520 CALL SOUND(T/4,B,V1)
530 CALL SOUND(T/2,HC,V1,D,V
2,LFS,V2)
540 CALL SOUND(T/2,A,V1)
550 CALL SOUND(T/2,B,V1,D,V2
,LG,V2)
560 CALL SOUND(T/2,G,V1)
570 CALL SOUND(T/2,A,V1,FS,V
2,LD,V2)
580 CALL SOUND(T/2,D,V1,LD,5
)
590 CALL SOUND(T/4,40000,V1)
600 CALL MOTION(#15,1,0)
610 CALL SOUND(T/4,FS,V1)
620 CALL SOUND(T/4,G,V1)
630 CALL SOUND(T/4,A,V1)
640 CALL SOUND(T/4,B,V1)
650 CALL SOUND(T/2,HC,V1,D,V
2,LFS,V2)
660 CALL SOUND(T/2,A,V1)
670 CALL SOUND(T/2,B,V1,D,V2
,LG,V2)
680 CALL SOUND(T/2,G,V1)
690 CALL SOUND(T/2,A,V1,FS,V
2,LD,V2)
700 CALL SOUND(T/2,D,V1,LD,5
)
710 CALL SOUND(T/4,40000,V1)
720 CALL SOUND(TD,B,V1,G,V1)
730 CALL SOUND(T/4,HC,V1,A,V
1)
740 CALL SOUND(T/2,HD,V1,B,V
1,LG,V2)
750 CALL SOUND(T/2,HD,V1,B,V
1,LG,V2)
760 CALL SOUND(TD,B,V1,G,V1)
770 CALL SOUND(T/4,HC,V1,A,V
1)
780 CALL SOUND(T,HD,V1,B,V1,
LG,V2)
790 CALL SOUND(T/4,40000,V1)
800 CALL SOUND(T/3,HE,V1,LC,
V2)
810 CALL SOUND(T/3,HC,V1)
820 CALL SOUND(T/3,A,V1)
830 CALL SOUND(T/3,HD,V1,LLB
,V2)
840 CALL SOUND(T/3,B,V1)
850 CALL SOUND(T/3,G,V1)
860 CALL SOUND(T/3,HC,V1,LD,
V2)
870 CALL SOUND(T/3,A,V1)
880 CALL SOUND(T/3,FS,V1)
890 CALL SOUND(T*1.2,G,V1,DU
M,30,DUM,30,-4,1)
900 DISPLAY AT(24,3):"AGAIN?
  Y" :: CALL DELSPRITE(#13,#
14,#15)
910 ACCEPT AT(24,11)VALIDATE
("YN")SIZE(-1):A$ :: IF A$="
N" THEN END
920 DISPLAY AT(24,1):"" :: G
OTO 250                    ■
```

THE P-GRAM PLUS WITH CLOCK

by Charles Good, Lima Ohio UG, USA

WHAT CAN YOU DO WITH A GRAM DEVICE?

The PGRAM and PGRAM+ are manufactured by Bud Mills Services. Like other battery backed GRAM devices such as the GramKracker and the Gramulator, this device allows you do three things:

-- 1. You can save any command module to disk and then load the module from disk back into the device. Once loaded, the module remains in memory even if the computer system is powered down because of the battery backup. Minor modifications to modules can easily be made such as changing screen colors, default device names, and (for example in the case of extended basic) default save/load drive names. These modifications can be made to the module disk files using a sector editor, or directly into the PGRAM's memory with the PGRAM's built in memory editor.

-- 2. The PGRAM can be used as a supercart, giving running software such as Y.A.P.P. and some Infocom games needed access to 8K of CPU RAM normally accessed via the cartridge port.

-- 3. GRAM devices can provide some of the same functions as a RAMdisk. They can be used as ROM disks for storage in GRAM memory of EA5 software such as DM1000. The stored software will boot very quickly from a powerup menu. Because loading a Gram device with code is rather cumbersome, and because you usually only have access to that software when you first turn on the computer, gram devices cannot be used to store text files or data files and are not as flexible as RAMdisks, which can store such files. Files created using any of the other GRAM devices, as well as files created by the Geneve's module save software can be loaded into the PGRAM's memory and will work. In addition, many of the various utilities written for other GRAM devices to add TI BASIC programs to the powerup menu and to add extra features to extended basic will work with the PGRAM. Using the now difficult to obtain JP Software program called GRAM PACKER, it is possible to store most (but not all) assembly PROGRAM image software that boots from option #5 of the EA module or loader 2 & 3 of Funnelweb in the PGRAM for instant access at the powerup menu. I had to borrow a copy of GRAM PACKER from a local user group member to do this. GRAM PACKER is a commercial product not available from Bud Mills.

GENERAL DESCRIPTION OF THE PGRAM

Unlike other GRAM devices, the PGRAM resides in the PE box and thus is not affected by what may be the TI system's weakest link, the cartridge (aka grom) port. The other GRAM devices mentioned above plug into the cartridge port.

The regular PGRAM has one bank of memory that simulates GROMs 3-7, plus two 8K banks of RAM to simulate the bank switched ROM that is in many TI cartridges. The "PLUS" version has three extra banks of GRAM (a total of four banks) which can be switched in and out with the PGRAM's memory editor and which can also all be accessed at system powerup using the 99/4A console's built in "Review Module Library" feature.

ADVANTAGES OF PGRAM+ COMPARED TO OTHER GRAM DEVICES:

I have had extensive experience with a GramKracker and am familiar with the Gramulator, so I can make first hand comparisons between these devices and the PGRAM+. I sold my GramKracker and purchased a PGRAM+ because I wanted a real time clock for my TI system and the PGRAM+ seemed a good way to obtain such a clock.

--- MULTIPLE MEMORY BANKS: You can store lots of software in the four memory banks of the PGRAM+, allowing the PGRAM+ to act somewhat like a "romdisk". Other battery backed gram devices have only one bank of memory. Using GRAM PACKER I have loaded up my 4 banks of memory so that the following menus are displayed with I "PRESS ANY KEY TO CONTINUE":

```
PRESS
1 FOR TI BASIC
2 FOR TI EXTENDED BASIC
3 FOR FUNNELWEB
4 FOR PRINTER SETUP
5 FOR REVIEW MODULE LIBRARY
```
(If I press 5 then I get the next menu for the second memory bank)
```
PRESS
1 FOR TI BASIC
2 FOR EDITOR/ASSEMBLER
3 FOR MAC FLIX
4 FOR DV80 WORK COUNT
5 FOR DISK MANAGER 1000
6 FOR REVIEW MODULE LIBRARY
```
(If I press 6 then I get the next menu for the third memory bank)
```
PRESS
1 FOR TI BASIC
2 FOR AMBULANCE
3 FOR TENNIS
4 FOR REVIEW MODULE LIBRARY
```
(If I press 4 then I get the next menu for the fourth memory bank)
```
PRESS
1 FOR TI BASIC
2 FOR SPACE AGRESSOR
3 FOR FIREBALL
4 FOR AUTO RACE
5 FOR GHOST SPELL
6 FOR REVIEW MODULE LIBRARY
```
(If I press 6 then I go back to the first menu above for the first memory bank)

The only actual cartridge software in this list is EXTENDED BASIC and EDITOR/ASSEMBLER. Everything else is either assembly PROGRAM memory image software or short TI BASIC programs. (AMBULANCE and TENNIS are available both as assembly PROGRAMs and as module software.) I don't actually have Funnelweb on my PGRAM since doing so would make Funnelweb hard to configure. The FUNNELWEB menu item is a short TI BASIC "CALL" which loads Funnelweb from my Horizon Ramdisk. PRINTER SETUP is a short TI BASIC program to set up my printer for particular fonts.

Bud Mills includes a public domain program called GMENU you can load into the PGRAM which will simultaneously display all the choices in all four memory banks on one screen. For those who can't get hold of GRAM PACKER, Bud includes some sample multiple item packs that can be loaded directly into the PGRAM. Bud also includes a version of John Johnson's BOOT designed for PGRAM use.

--- ROCK STABLE EXTENDED BASIC: The Extended Basic module is notorious for locking up our computers if it doesn't make perfect contact with all the pins of the cartridge port. Other Gram devices that plug into the module port have gold plated contacts, so when loaded with Extended Basic, they don't lock up the computer AS FREQUENTLY as does the XB module. However, my experience with a GramKracker indicates that XB does lock up all too often with other gram devices because they plug into the cartridge port. Contacts of this port are easily worn and/or abused. Such a lockup requires the user to remove and reinsert the gram device.

Because the PGRAM+ is in the PE box it eliminates any problems caused by worn cartridge ports. I have never had an Extended Basic lockup using my PGRAM.

LIMITATIONS OF THE PGRAM+:

--- ONLY ONE PAIR OF RAM BANKS: Note this section well before purchasing the "plus" version of PGRAM, because what I am describing is potentially a major limitation.

There are four banks of GRAM that simulates cartridge GROM. But there is only one pair of RAM memory banks that simulates the two ROM banks that can be found in many modules. This means that only one module at a time that uses ROM can be loaded into the PGRAM+. If you wonder why much of my PGRAM+ memory is populated with games, this is because some of the modules I would like to use are not simultaneously compatable with each other in the PGRAM+.

Most GRAM device users like to keep Extended Basic in their Gram device most of the time. Other "important" TI modules are TERMINAL EMULATOR II (for unlimited speech), EDITOR/ ASSEMBLER (some software will ONLY load using an EA module that resides starting at grom memory location >6000), MULTIPLAN, and perhaps PLATO and LOGO II. Of these only EDITOR/ ASSEMBLER and MULTIPLAN do not use ROM and can thus reside in a PGRAM+ simultaneously with Extended Basic but in a different PGRAM+ memory bank. To use any of the other "important" modules it is necessary to load them from disk into the PGRAM+ GRAM bank 1, erasing Extended Basic. Later, XB must be reloaded into the PGRAM+ GRAM bank 1 from disk.

I thought when I bought my PGRAM+ that it would be possible to keep Extended Basic and TEII both in the PGRAM+ and have access to unlimited speech from within Extended Basic. It can't be done. The XB and TEII modules compete for the same single pair of 8K RAM banks in the PGRAM+. Many of the better cartridge games manufactured by TI, including most of those with 1982 and 1983 copyrights, use ROM and cannot coexist in the PGRAM+ with Extended Basic. The only TI game I really like that CAN coexist with XB is CAR WARS.

--- NO ACCESS TO GROMS 0, 1, AND 2 WITH A PGRAM+: Gram devices that plug into the console allow the user to modify or completely replace the three grom chips that are on the console motherboard. These comprise the console's operating system (Grom 0) and the TI BASIC interpreter (Grom 1 & 2). Usual mofifications of Grom 0 include a custom title screen ("Charlie's TI computer") and a nice lower case character set with true descenders. Since TI BASIC isn't used very much, other software can be loaded into the Grom 1 & 2 memory space.

These Grom 0, 1, and 2 manipulations are not possible with a PGRAM. I do miss the nice lower case character set I had with my GramKracker, but other Grom 0, 1, and 2 modifications are not important.

--- GMENU LIMITATIONS AND HOW TO SOLVE THEM: If you use the GMENU utility to simultaneously display all software options available from all GRAM banks, and you have Extended Basic in bank 1 (the usual place for XB in a PGRAM), there are some circumstances in which Funnelweb will fail to recognize the presence of Extended Basic. From Funnelweb's Disk Review, if you move the cursor next to an XB PROGRAM file and press "R" to run the program, Disk Review may respond with "XB MODULE NOT FOUND" even though it is there where it belongs in GRAM bank 1. The solution is to load GMENU into whatever GRAM module or pack you intend to keep in GRAM bank 2. With GMENU in bank 2, both GMENU and Funnelweb seem to work properly.

I don't like GMENU's white on light blue default colors. I prefer white on dark blue. To make this change use a sector editor and search both of GMENU's disk files in hex for >F5. Change this to F4. There are two F5's that should be changed, one for the screen border and the other for the central text bearing part of the screen.

--- THE PGRAM MEMORY EDITOR: This has the look and feel of the Gram-Kracker memory editor, but it isn't quite as easy to use. It is, however, in the PGRAM's operating system and thus always available, unlike the Gramulator's memory editor which must be loaded in from disk.

The PGRAM documentation has all the necessary information needed to "turn on" the editor, but this information is scattered over several pages. To properly EXAMINE grom memory, even if you don't intend to do any editing, it is necessary to turn on the editor. Otherwise the screen display of grom memory will be incorrect. Here is what you do. Select MEMORY EDITOR from the menu you get when you type CALL PGRAM from BASIC. Type FCTN/1 to enable you to set the "CRU bits". Type the PGRAM's CRU address, usually 1700. Then in the next field type "12" to change the CRU bits so that memory editing can proceed. THEN TYPE FCTN/1 AGAIN to exit the CRU bit edit mode. This is the confusing part. If you type FCTN/9 (back) you will also exit the CRU bit edit mode but the CRU bits will not be changed and you will not be able to examine or edit memory.

THE PGRAM CLOCK

This battery backed clock shows the day of the week, month, year, hour (only in 24 hour format), minutes and seconds. There is no provision in any of the various 99/4A disk controllers for automatic time/date stamping files, but there are some specific applications that can use the PGRAM clock.

From either BASIC you can get an on screen display of the time by typing CALL PTIME to set the clock or CALL PGRAM to access the PGRAM's loader/ editor. Neither of these gives a clean exit back to BASIC. You can also, from within either BASIC (either in command mode or from within a running BASIC program), display the clock information or set the clock in the same way you would with a CorComp clock or the MPB clock. In these cases you remain in the BASIC environment. Everything is explained in the PGRAM user guide.

The following software will make direct use of the PGRAM clock to display the correct time and/or date. This software is all public domain except for BOOT (which is fairware) and is all available in the Lima User Group library.

-- REMIND ME will, upon booting with a PGRAM clock in the computer, correctly show the correct time, month, and date, and place the cursor on the correct day of the displayed month. To have all this happen, it is necessary for the user to use a sector editor, search the single REMIND-ME file in ASCII for the text "CLOCK", and use the space bar to cover over the word CLOCK with blanks.

-- BOOT will display the correct time and date on screen if a PGRAM clock is in the computer. It may be necessary to space over CLOCK in BOOT as described above for this to happen.

-- Mel Nomina's CHECKBOOK WRITER will automatically put the correct date on your checks if you remove some REM statements.

-- Harold Hoyt has written a program that will display the current date and time, catalog a disk, and optionally time/date stamp any DV80 text file on the disk.

-- Harold Hoyt has also written an extended basic program that will put a time/date display at the top of the screen and then erase itself, leaving the time/date display. This will remain on display through the running of any extended basic program and even if NEW is typed.

CONCLUSIONS:

There are advantages and disadvantages to using a PGRAM as opposed to a GramKracker or Gramulator. For me, the most important advantage is the stability of extended basic and the fact that I may never have to use my console's cartridge port again. I appreciate never having extended basic crash. Since I have a specially modified 99/4A console (modified for use with an AVPC card and with internal speech) I would hate to have to switch to a different console because of an increasingly unreliable cartridge port.

PRICES (revised by the editor)

PGRAM (72 kbytes) kit $150. PGRAM+ (192 kbytes) kit $200. Clock $20. Manufacturer assembly $30 (Built PGRAM+ with clock is $250). Add $5 for oversea air mail (is this a printing error instead of $15 ??).
Available from:
Bud Mills Services, 166 Dartmouth Dr., Toledo OH 43614-2911, USA.
Voice phone 419-385-5946.

REFERENCES (added by the editor)
PB 85-4: Maximem
PB 86-3: Maximem
PB 87-1: Gram Kracker
PB 91-4: RAG Linker, GRAM flag words■

CHICAGO TI FAIRE NOV 1991

by Charles Good, Lima Ohio UG, USA

The Chicago user group did their usual good job of sponsoring a well organized classy event. This year the Faire was at a new location near Chicago's airport at the Elk Grove Village Holliday Inn. Apparently the organizers were pleased with this new venu, because it was announced that next year's Faire will be at the same location in early November 1992.

HARDWARE

The big no-show was THE ACCELERATOR which has been advertised in Bud Mills' MICROPENDIUM ads the last couple of months for $250. Bud says "any day now", and I think he means it. As I understand the situation, production of "THE ACCELERATOR" jointly involves O.P.A. and Bud Mills Services, and Bud is waiting on O.P.A. for something. Bud had Geneve MEMEX memory cards, PGRAMs, and Horizon Ramdisks built up and available for immediate purchase at his table. I bought a PGRAM+ and will report on it in a future article.

O.P.A. had what seems to me to be the most interesting new hardware device up and running at their table. Their POP-CART is about the size of an ordinary TI solid state cartridge and plugs into the 99/4A's grom (cartridge) port. This device is more stable than, and replaces the multi module GIZMO box, which O.P.A. will not be marketing. POP-CARTs contain RAM and GRAM in 256K sized chunks of EPROM and are custom programmed by O.P.A. with your choice of existing TI cartridge software. You can fit 5-7 TI modules of your choice such as Extended Basic, TE2, the EA module, Plato, and Multiplan onto a single 256K POP-CART. O.P.A. can also put specific disk based programs on a POP-CART. Archiver, Rapid Copy, DSKU are among the programs available. Just plug this custom "super module" into any 99/4A console and have access via the console's REVIEW MODULE LIBRARY feature to all of the modules. If Extended Basic and the TE2 are on the same POP-CART you can have unlimited TE2 speech from within XB.

The cost is US$95 plus $4.50 shipping for the 256K model, which includes custom programming of your POP-CART by O.P.A. Additional multiples of 256K are available for more money, up to about 2Meg of TI modules on the same POP-CART. For an extra US$25 O.P.A. will put its SOB operating system on your POP-CART so that you can get a power up menu display of all the modules on your cartridge and don't have to cycle through them one at a time using REVIEW MODULE LIBRARY.

POP-CARTS are hard burned to the specifications of the purchaser. There are no batteries or disks and the device is NOT user programmable. To get one, write O.P.A. and request a list of what TI modules are available. Pick the ones you want on your POP-CART, sign a statement that you already own the requested modules, and send your money and the statement back to O.P.A. They will custom burn your POP-CART and mail it to you.

Another significant piece of hardware shown was Mike Maksimik's Midi Interface (aka MIDI MASTER 99). This is a special cable that connects between the RS232 serial port and an electronic keyboard's Midi I/O jacks, in conjuntion with some software. The software allows you to create music files as DV80 files using TI Writer and then play this music off of a disk via the keyboard. Unlike ordinary audio music tapes records or CDs, the Midi software allows you to alter the tempo and/or octive of the on disk music "on the fly" to create interesting effects. Software version 3.0 of the Midi Interface (available "any day now") will allow one to play the keyboard and have the resulting music stored on disk.

Rave had their expansion chassis and IBM style TI keyboards available. The Geneve chassis is $309 and the TI version costs $379. These look great and do away with much of the cable clutter common to many TI systems, but all they have inside is a power supply. All the important guts (keyboard, computer motherboard, drives, and cards) are extra. Working 99/4A consoles were available at the Chciago show from two different dealers for $10. Used PE boxs with the flex cable, a TI controller, 32K and a SSSD drive were going for $100 at one dealer's table. At these prices I can see no reason other than cosmetic and neatness to buy a Rave Chassis. TI PE boxes were built like M1 tanks. They are tough! Even used TI PE boxes should give years of reliable service.

SOFTWARE

Barry Boone demonstrated SOUND F/X, software that can play canned digitized sound recordings via the 99/4A's sound chip through the monitor's speaker. This is all done with software, which Barry sells for $15. All available RAM (supercart, 80 column device video RAM, RAMBO, Geneve) can be used to store between 10 seconds (on a basic TI + 32K system) and 10 mimutes of digitized sound for playback. "Anything you can digitize," including IBM sound blaster sound files are supported and there are said to be lots of these files available for downloading from bulletin boards. Digitized sound I heard included short music segments and familiar voices from TV and movies. According to Barry, "TI sound resolution is about 1/4 that of an audio CD, Geneve sound resolution is about 1/2 that of an audio CD." Barry had sound files available for $1.25/disk (both SSSD and DSDD the same price). No, you can't create digitized sound files with this software. All you can do is play sound files created by others.

Ken Gilliland (NOTUNG SOFTWARE) demonstrated TI CASINO, enhanced since its release last winter at Fest West. TI CASINO is a real "experience", much more than just individual betting games. You feel that you are actually visiting a casino, complete with 8 casino games, a stage show for entertainment, a restaurant, and various personalities you interract with. If you make a profit, the software prints a personalized check for you. I purchased this ($15) and will review it in a future article.

Representing ASGARD SOFTWARE, David Bishop demonstrated his not quite ready for release MUSIC MAKER CONVERTER. This software can convert 59 sector MUSIC MAKER files to EA source code, C99 code, mergable extended basic, Fortran, or a runnable TI BASIC file. I wonder of David knows about MUSIC SDA, the never released official module that does some of these same conversions.■

PÅSKDAGENS DATUM

(enligt den gregorianska kalendern)

```
100 ! EASTERCALC, Bug News
101 CALL CLEAR
110 INPUT "ENTER YEAR: ":Y
120 IF Y<1583 THEN PRINT "YE
AR MUST BE > 1582": : :: GOT
O 110
130 Y1=Y/19
140 A=INT((Y1-INT(Y1))*(19)+
.001)
150 B1=Y/100 :: B=INT(B1)
160 C=INT((B1-INT(B1))*(100)
+.001)
170 D1=B/4 :: D=INT(D1)
180 E=INT((D1-INT(D1))*(4)+.
001)
190 F=INT(((B+8)/25)+.001)
200 G=INT((B-F+1)/3)
210 H1=(19*A+B-D-G+15)/30
220 H=INT((H1-INT(H1))*(30)+
.001)
230 C1=C/4 :: I=INT(C1)
240 K=INT((C1-I)*(4)+.001)
250 L1=(32+2*E+2*I-H-K)/7
260 L=INT((L1-INT(L1))*(7)+.
001)
270 M=INT((A+11*H+22*L)/451)
280 N1=(H+L-7*M+114)/31 :: N
=INT(N1)
290 P=INT((N1-N)*(31)+.001)
300 N$="APRIL"
310 IF N=3 THEN N$="MARCH"
320 PRINT "EASTER IS ";N$;P+
1: : :: PRINT "WANT ANOTHER
YEAR? (Y/N)"
330 CALL KEY(3,K,S):: IF S=0
 THEN 330 ELSE IF K=89 THEN
100
```
■