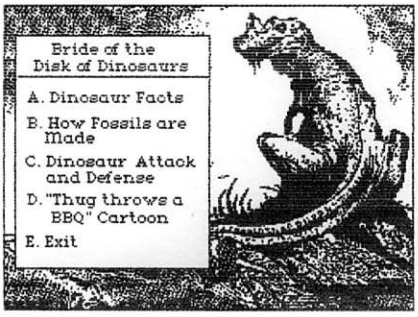
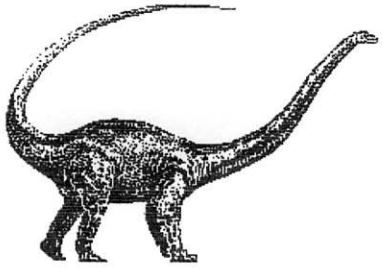
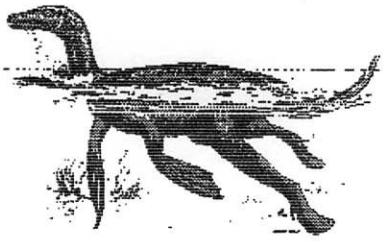
PROGRAM BITEN 92-1







ISSN 0281-1146

KALLELSE TILL ÅRSMÖTE

Medlemmar i föreningen Programbiten kallas härmed till Årsmöte Lördagen den 14 mars 1992 kl.13.00:
Albatrossvägen 76, 1 tr.ned, Haninge
(pendeltåg söder om Stockholm)

Förslag till dagordning:

1. Mötet öppnas

2. Val för mötet av ordförande, sekreterare och två justeringsmän

3. Mötets behörighet
   - utlysning
   - röstberättigade närvarande

4. Beslut om dagordning

5. Styrelsens årsredovisning för 1991. Verksamhetsberättelse och kassarapport delas ut på mötet.

6. Revisorernas berättelse för 1991

7. Om ansvarsfrihet för styrelsen för 1991

8. Val av styrelse för 1992. Årmötet väljer enligt stadgarna en styrelse bestående av ordförande och kassör, vilka samtliga utses till befattning på årsmötet, samt därtill tre och högst sju övriga ledamöter.

9. Val av revisorer för 1992. Mötet väljer två revisorer och en suppleant.

10. Val av valberedning. Mötet väljer två ledamöter varav en sammankallande.

11. Om årsavgift 1992.

12. Årmötet avslutas. ■

TEXAMENTS
53 Center Street,
Patchogue, NY 11772, USA
(porto USD 8.50 extra per order)

TI Artist Plus!	USD 25
TI Base version 3	USD 25
Sound F/X	USD 15
(spelar audiofiler med sång, tal och musik utan Speech Synthesizer)	
Sound Bytes #1:Cartoons	USD 4
Sound Bytes #2:Science Fiction	USD 4
Sound Bytes #3:Miscellaneous	USD 4
Sound F/X(inkl. Sound B.#1-3)	USD 22■

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 licensnummer:
82100488

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:-
1990	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.

PUTTING IT ALL TOGETHER #8

by Jim Peterson, Tigercub, USA

The hard part of learning to program is not in learning what the various commands do - it is in learning how to put them together to do what you want them to do!

Key in this little program and run it to see what it does, then read the explanation of how it does it.

In the early days, when computers had tiny memories, much emphasis was placed on efficient programming - the pioneer David Ahl called it "elegant" programming. The old 99'er magazine published some one-liners. My Tips From The Tigercub contained some one-line programs, even some no-line programs that could be keyed in and run in the immediate mode. In order to cram over a hundred subprograms onto a disk, I made great use of compact programming techniques on my Nuts & Bolts disks. Later, Mike Stanfill originated the name "tinygram" for a program that would fit on one screen, and wrote some great ones. Ed Machonis wrote a diskfull of tiny printer utilities.

Richard Mitchell in his Super 99 Monthly once called me the "king of the one liners", but this title rightly belongs to John Martin. The following one-line disk cataloger is an example.

```
1 IF F THEN INPUT #1:A$,A,J,
K :: IF J THEN PRINT A$;TAB(
12);J;TAB(18);SEG$(B$,ABS(A*
2)+1,2);K;TAB(27);A<0 :: GOT
O 1 ELSE RUN ELSE B$="AVDFDV
IFIVPG" :: INPUT "DSK":F ::
OPEN #1:"DSK"&STR$(F)&".",IN
TERNAL,RELATIVE,INPUT :: GOT
O 1 !BY JOHN M
```

An undefined numeric variable has a value of 0, which is the value of F when the program is first run. IF F THEN is interpreted as "if F is other than 0" so program execution jumps to the first unpaired else. IF J is paired with ELSE RUN so execution jumps to ELSE B$; a string is assigned to B$, the INPUT asks for a disk number, and file #1 is opened, without a filename, as an internal relative file, for input. When it is opened, the first sector of the disk can be read; it contains information regarding the disk and its contents. GOTO 1 goes back to start over. The variable F now as a value other than 0 (from the INPUT disk number) so the values for A$, A, J and K are read from the disk. On the first pass, these are the disk name, a 0, the number of sectors initialized, the number of sectors available, and a 0. IF J THEN is interpreted as "if J is other than 0" and it is because it contains the number of sectors, so the disk name is printed, followed by the number of initialized sectors at tab 12. Since a 0 was read into A, the ABS(A*2)+1 is 0 times 2 plus 1, which is 1, so the segment of "AVDFDVIFIVPG" starting with the first character and consisting of two characters (AV) is printed (meaning "available"), followed by the number of available sectors read into K (preceded by a space because it is numeric). Since a 0 was read into A, the statement A<0 (A is less than 0) is false and has a truth value of 0, so a 0 is printed at tab 27. Execution returns to the beginning, and values are read into the variables again. Now, A$ will be a filename. A will be a number from 1 to 5, indicating the type of file - 1 for display fixed, 2 for display variable, 3 for internal fixed, 4 for internal variable, 5 for a program. If the file is protected, the number will be negative. J will be the number of sectors occupied by the file, and K will be the record length of the file (0 in the case of a program). The filename is printed, and its sector length at tab 12. ABS converts the A from negative to positive, if necessary, and the formula selects the letters DF, DV, IF, IV or PG to print, followed by the record length from K. If the file is protected, A has a negative value and A<0 therefore has a truth value of -1, otherwise a 0, printed at tab 27. Execution goes back to the beginning and this continues until blank records are read. J will then have a value of 0 so execution jumps to ELSE RUN, which re-runs the program, thereby zeroing out the value of F.

John Martin, this is elegant programming to the ultimate! ■

SWEDLOW TI BITS * 14-16 *

by Jim Swedlow, USA

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

A MULTIPLAN APPLICATION
PAYING YOUR BILLS

Home computers were sold for many things, some of which were even possible. One of the big selling points was managing your home finances. That was one of the motives behind my purchase. Well, folks, 'twern't true. A calculator is vastly superior for balancing a check book and, for most of us, the time and trouble required to maintain home records outweigh the benefits.

After many false starts, I developed a Multiplan application that I use for paying my bills. I wanted to automate the manual process of deciding how much goes to whom. That was a pencil, paper and calculator exercise of writing down what I owed, adding it up and then figuring how to dole out what was in my paycheck. Sound familiar?

This application does the following:

o Remembers your fixed expenses (house note, etc).

o Has a place for the monthly expenses that vary (utilities, etc).

o Keeps track of the running balance and the check number for each check.

With a blank Multiplan screen, press <F> for Format and then <D> for Default. Press <W> for Width and set the default width to 11. Then press <ENTER>. For those of you who are not used to Multiplan, in this and following steps, don't press <ENTER> until I tell you to.

Press <F> for Format and then <C> for Cells. Type in C3:4 and then tab (CTRL 2) twice. Press <$> and then press <ENTER>.

Move your cursor to R1C4 and press <F> for Format and then <C> for Cells. Press tab <CTRL 2> twice. Press <D> for DEFault and then <ENTER>. Move the cursor to R9C3 and repeat this process.

Push <O> for Options and then <N> (to turn 'recalc' OFF). Now press <ENTER>.

Move your cursor to R1C1, and press <F> for Format and then <W> for Width. Set the column width for column 1 as 6.

Enter the data for the first twelve rows. Be careful that the numbers are entered as values not as alpha characters. For the dollar amounts, don't enter the dollar sign, Multiplan will add it. Use amounts that fit your situation. It is not necessary to enter the decimal if there are no cents (.00).

1	2	3	4
1		OCTOBER 1987	
2			
3	CHECK DUE TO	AMOUNT	BALANCE
4			
5	Balance	$100.00	
6	Pay Check	$900.00	
7	Service Chg	$6.00	
8	Cash	$20.00	
9	Next Check	1223	
10			
11	MORTGAGE	$234.44	
12	GAS	$57.20	

Now we are ready for some formulas:

Location	Formula
R5C4	=R5C3
R6C4	=R5C4+R6C3
R7C4	=R6C4-R7C3
R8C4	=R7C4-R8C3

What have we done? We made a place for your beginning check book balance. We have a cell for the amount of your pay check, your monthly service charge AND for that handy cash advance you got from the electronic teller on the way home. We have entered formulas for up-

dating your check book balance. Finally, we told Multiplan the number of the next blank check in your check book.

Two more formulas:

Location	Formula
R11C4	=R8C4-R11C3
R11C1	=IF(R11C3=0,R9C3-1,R9C3)

More about that IF function later. For the formulas in row 12 we need to move the cursor. Place the cursor on R12C4 and press <=>. Move the cursor to R11C4 and then press <->. Now move the cursor the R12C3 and then press <ENTER>. Your formula should look like this:

```
R[-1]C-RC[-1]
```

Move your cursor to R12C1 and press <=>. Type 'IF('. [The single quote mark is used here to show what you type in - in this case letter I, letter F and open parens.] Move the cursor to R12C3 and then type '>0,'. Now move the cursor to R11C1 and type '+1,'. Move the cursor to R11C1 again and then type ')' and then press <ENTER>. The formula should read:

```
IF(RC[+2]>0,R[-1]C+1,R[-1]C)
```

This IF function tells Multiplan what check number to put in R12C1. If R12C3 (the amount of the check for that row) is greater than zero (>0 or you are paying that person something), increase the check number for the previous line (R[-1]C or R11C1 in this case) by one, otherwise copy it.

The end is in sight. Figure the highest number of bills you will ever pay and add a few. Lets say your total is 15. Move your cursor to R12C1 and press <C> for copy and then <D> for down. Type in 13 (you already have two) for the number of cells and then tab (CTRL 2) to the 'starting at' field and change it to R12C1:4. Now press <ENTER>.

We need some totals. Now move to R27C1 (use <G> for Go). If your number of bills is other than 15, the location of these total lines will be different. Enter:

R27C2	TOTALS
R27C3	=SUM(R11:25C4)
R27C4	=R25C4

This repeats your final check book balance and totals the amounts you have paid.

The next step is to lock your formulas. Press <L> for Lock and then <F> for formulas. Then press <Y> to confirm.

You must now unlock some cells that your will want to change. Press <L> for Lock and then <C> for Cells. Type in R11:25C2 and then press tab (CTRL 2). Press <U> to Unlock and then <ENTER>.

You should also unlock the month cell. Move your cursor to R1C3 and press <L> for Lock. Press <C> for Cells and then tab (CTRL 2). Now press <U> to Unlock and then <ENTER>.

Now all you have to do is to go back in and enter your numbers and the names of the fine folks you owe money to and how much you owe.

You have built your spreadsheet.

Since automatic recalculation is off, remember to press Recalc (FCTN 8) to update the totals when you change information. You should <T>ransfer <S>ave this sheet before going any further.

About once a week, I update my sheet with the bills that came in the mail (bills and ads seems to be most of the Postal Service's offerings). On payday, I enter my check book balance, the next check number and the amount my employer paid me. I can adjust what I pay to make sure I have enough left over to make it until next payday. As Bill Harms says, you can go "what-if'ing" to work out the best solution.

MAKING A DEGREE MARK IN TI WRITER

An owner in Huntsville Texas wrote me and asked if I know how to make TI Writer type a degree sign on a TI

Impact Printer (it is an Epson MX80). A degree mark is not one of the standard ASCII characters. Although many newer printers can print it, the MX80 can't.

The only way I could figure to do it was to combine TI Writer's transliterate command and the MX80's graphics ability. After a bit of experimentation, I hit on this:

.TL 91:27,76,7,0,48,72,72,72,48,0,0

This redefines the left bracket ([). The first two characters (27,76) tell the MX80 to invoke graphics. The next two (7,0) tell it that there will be seven graphics characters. The last seven characters define the degree mark.

This is not a perfect solution as, if you right justify, the right margin will be a bit uneven. It should work, however, on most Epson and compatible printers.

SOME THOUGHTS ON WORD PROCESSORS

Of late I have occasion to use a number of word processors on other machines. I learned word processing on TI Writer and I wanted to see how the 4A stood up.

TI Writer is limited by the 4A's design. Eighty columns and a full keyboard make text management (warning: buzz phrase alert!!) much easier. Otherwise, TI Writer fairs well.

Just about anything you can do with the big name packages, you can do with TI Writer. Sometimes it is a bit harder, but it can be done. TI Writer is a powerful and flexible tool. It has some abilities, like transliterate, that are superior to other word processors.

The others are slicker because they have much more memory available. They can do things with one or two key strokes that take five or ten with TI Writer - but they can be done on the 4A.

If you are writing a book, it might be worth the cost to move up. But for correspondance, writing this column and similar jobs, TI Writer can do anything you need it to do. And that's a fact.

BLACK FRIDAY PLUS FOUR

It was four years ago this month that TI announced that they were droping the 4A. October 28, 1983. A date that changed everything for 4A owners.

We moved from the main stream of computing to a cul-de-sac. Software and hardware became scarcer and scarcer. Retailers dropped from many to only a few.

And yet a cul-de-sac is not a bad place to live. In fact, they are preferred. Ours turned out to be pretty good. Four years later, exciting software continues to appear. This has been the year of graphics applications, with many innovative programs comming out.

There are some signs of strain. User groups report declining membership and money problems. TI owners are slowly moving to other machines (often with three letters).

The end, however, is not upon us. Our 4A still has strong support from retailers, developers, publishers, user groups and owners. I expect to be writing on the fifth and sixth anniversities of black Friday.

TI SERVICE

I called 1-800-TI-CARES on September 24th and got the current costs for exchanging TI equipment. These prices do not include your cost of sending the item to Lubbock or the amount TI adds for return postage. Call TI before sending them anything.

4A Computer	$30.50
PE Box	$55.00
XB Module	$33.00
Disk Controller	$44.00
32K Card	$44.00
RS232 Card	$33.00

Speech Syn	$30.00
TI Joysticks	$ 9.75

ON DISKS AND DRIVES

A while back the Disk Doctor attended one of our meetings. He had a number of interesting things to say. Since some of you missed it, here are a few of his comments.

o Don't clean your drives until you need to. Your system will tell you when it is time - you will have trouble reading disks.

o When you do clean your drive, use any brand name commercial disk drive cleaner and follow instructions.

o If this fails, you need to have your drive cleaned professionally. If you want to try yourself and you have a double sided drive, be careful with the second read/write head. It is very, very easy to bend the bracket to the point that the head must be realligned.

o He has tested the amount of residue left on heads with brand name disks ($1.00 + each) and the cheepies ($0.25 or so). He found no difference. This doesn't mean that they are of equal quality, only that the cheepies are not dirtier.

o He opposes flippies for single side users. His point is that when you flip the disk and it runs backwards in its cover, dirt is loosened and spun into your drive.

o His overall advise is the first rule of engineering: If it ain't broke, don't fix it.

SOME MORE THOUGHTS ON BACKING UP DISKS

Over the years I have mentioned the importance of backing up your disks. Simply put, disk drives eat disks. On the first weekend of October, I was working on some letters. This was the weekend where the temperature was well over 100 degrees. I blew both my word processing disk and my data disk.

I had a backup of the word processor, but it was not configured. That night, after it cooled down a bit, it took me about half an hour to recreate a working disk. The data files were simply lost.

The moral? Keep two back ups of your program disks. One of the disk as you received it (the master) and one of your configured working disk (back up working disk). Don't forget to back up your data disks every now and then. This will save you time and aggravation next time your drive gets hungry.

TI WRITER's INCLUDE FILE

One of TI Writers nicer features is Include File (.IF). It has a few limitations, but it extends TI Writers capabilities.

TI Writer cannot work on large files. No books in one file here. As you reach the size limit, the time it takes to load and save files increases markedly. Include File to the rescue.

Suppose your have written two chapters of your next book. Your named your files CHAPTER1 and CHAPTER2 (very original). At the end of Chapter 1 (the very last line), add this:

.IF DSK1.CHAPTER2

Name CHAPTER1 for the Formatter and it will print both chapters. All the formatting commands you set for Chapter 1 will be used when Chapter 2 is printed, so you don't have to restate the margins and such.

Ah, you finish Chapter 3. No problem. At the end of Chapter 1, add another line:

.IF DSK1.CHAPTER3

You cannot do this at the end of Chapter 2, as you can't chain these commands. Also note that you must specify the drive number (DSK1 in this case).

I prefer to make a master file (called CHAPTER0) will all of the

BEGINNER ASSEMBLER – 7
LET THERE BE SOUND

by Mack McCormick, USA

As you know from your BASIC programming sounds can be from 110 Hertz to 44,733 Hertz plus 8 noises may be generated. Durations may be from 1 to 4250 milliseconds. (.001 to 4.25 seconds). The volume can be from 0 (loudest) to 30 (quietest). Up to three tones and one noise may be generated simultaneously by the TMS 9919 sound generator controller chip.

Three steps must occur to produce a sound using assembly language:

1. Load the Sound Table which begins at VDP address >83CC with the sound data.

2. Set the least significant bit of the byte at CPU address >83FD to indicate to the computer that the sound table is in VDP RAM.

3. Enable interrupts by using the LIMI 2 instruction.

Once each of these conditions are met you can start the sound generator by placing a value of >01 at CPU address >83CE. This address is used by the interrupt routine as a count down timer during sound generation.

THE SOUND TABLE

You must produce a sound table which describes the characteristics of the sound you wish to produce. The sound generators are numbered 1, 2, and 3. To produce a sound you must enter the following information:

1. Specify the tone generator
2. Frequency
3. Volume
4. Duration

Noises:

1. White or Periodic
2. Shift Rate
3. Volume
4. Duration

All bytes are specification bytes except duration. It takes three specification bytes to hold the generator, volume, and frequency. The frequency must be entered as a code.

.IF commands:

```
.IF DSK1.CHAPTER1
.IF DSK1.CHAPTER2
.IF DSK1.CHAPTER3
```

Before (not after) your .IF lines, put in your format, header and footer instructions. Now you have all of your format commands in one place that is easy to find and edit.

Enjoy. ■

SPECIFICATION BYTES FOR TONES

Byte	Bit#	Contains
ONE	0	Always set to 1
	1-2	Specifies the sound generator
	3	Always 0
	4-7	Contains the 4 least significant frequency code bits
TWO	0-1	Always 00
	2-7	Contains the 6 most significant frequency code bits.

```
THREE   0       Always 1
        1-2     Indicates sound generator used.
        3       Always 1
        4-7     Volume level
-----------------------------------------------------------------
```

Bits 1 and two of all bytes indicate the tone generator: 00 is generator #1. 01 = #2. 02 = #03. 11 = noise.

FREQUENCY VS. FREQUENCY CODE

The frequency code is defined as half the period of the specified frequency. Here's the formula:

```
        111860.8
        -------- = Freq Code
        Frequency
```

Example: To find "middle C" which has a frequency of 523.25. 111860.8/523.25=213.8 . This rounds to 214 or >0D6. Bits 0-5 are placed in bits 2-7 of the second specification byte. The four least significant bits of the freq code are placed in bits 4-7 of the first specification byte. Example to enter a tone of 392 Hz in generator 1 this equates to a frequency code of 285 or >11D.

```
   1000 XXXX   00XX XXXX  = >8---
```

Here we have selected generator 1. Now we take our freq code >11D and place its 4 least significant bits (>D) in bit position 4-7 of the first specification byte:

```
   1000 1101   00XX XXXX  = >8D--
```

Finally we take the most significant 6 bits of the frequency code (>11) and place them in bit positions 2-7 of the second specification byte:

```
   1000 1101   0001 0001  = >8D11
```

We have created the first two specification bytes to generate a tone on generator 1.

VOLUME SPECIFICATION BYTE

Volume is held in bits 4-7 of the third specification byte. It can range from 0 to 30. The 0-3 bits contain the generator number. You must pad the volume on the right with 0 always. For example: A volume of 0 on generator 2 = 1011 0000.

DURATION

Not a specification byte. How long the tone or noise will last. Measured in 1/60 of a second. Can be from >00 to >FF.

LOADING THE SOUND TABLE

You must indicate the number of specification bytes you are going to feed the sound generator. For example:

>03,>89,>3F,>91,30

>03 indicates three specification bytes. Second and third bytes mean generator 1 with a tone of 110 Hz. The forth byte sets the volume at 2 on generator 1. The 30 indicates 30/60 of a second duration.

Provided is a table for generating tones quickly.

So go FORTH and make music!!

```
****************************************
* ACCOMPANIES THE SOUND TUTORIAL       *
*    PLAYS "HOME ON THE RANGE"         *
****************************************
       DEF  START
       REF  VMBW
* SET UP VARIABLES FOR PROGRAM
MYREG  BSS  >20
SOUNDT EQU  >1000        SOUND TABLE ADDRESS
```

```
ONE     BYTE >01
        EVEN                FORCE LOCATION COUNTER TO WORD BOUNDRY
* FIRST EXECUTABLE INSTRUCTION
START   LWPI MYREG
        LI   R0,SOUNDT
        LI   R1,SDATA       SOUND DATA
        LI   R2,274
        BLWP @VMBW
* GENERATE THE SOUND
LOOP1   LIMI 0              LOAD INTERRUPT MASK IMMEDIATE
        LI   R10,SOUNDT     LOAD R10 WITH SOUND TABLE ADDRESS
        MOV  R10,@>83CC     VDP SOUND TABLE
        SOCB @ONE,@>83FD    SET ONES CORRESPONDING BYTE (SOUND TABLE IN VDP RAM)
        MOVB @ONE,@>83CE    START SOUND PROCESSING
        LIMI 2
LOOP2   MOVB @>83CE,@>83CE *WHEN CPU ADDRESS >83CE = 0
        JEQ  LOOP1          *SOUND PROCESSING IS
        JMP  LOOP2          *FINISHED & PROGRAM REPEATS
* SOUND DATA
SDATA   BYTE >03,>8D,>11,>91,40
        BYTE >04,>AD,>11,>9F,>B1,40
        BYTE >03,>A6,>0D,>B1,40
        BYTE >06,>BE,>0B,>AD,>11,>95,>B5,40
        BYTE >09,>8A,>0A,>A6,>0D,>CD,>11,>95,>B5,>D5,60
        BYTE >05,>86,>0D,>91,>BF,>DF,20
        BYTE >03,>82,>0E,>91,40
        BYTE >03,>8E,>0F,>91,40
        BYTE >03,>80,>0A,>91,40
        BYTE >04,>A0,>0A,>9F,>B1,40
        BYTE >09,>80,>0A,>A6,>0D,>CD,>10,>95,>B5,>D5,60
        BYTE >05,>80,>0A,>91,>BF,>DF,20
        BYTE >03,>80,>0A,>91,20
        BYTE >03,>8F,>08,>91,40
        BYTE >09,>8A,>0A,>A6,>0D,>CD,>11,>95,>B5,>D5,40
        BYTE >05,>86,>0D,>91,>BF,>DF,20
        BYTE >04,>A6,>0D,>9F,>B1,40
        BYTE >05,>C6,>0D,>9F,>BF,>D1,40
        BYTE >03,>C2,>0E,>D1,40
        BYTE >03,>C6,>0D,>D1,40
        BYTE >03,>CE,>0B,>D1,80
        BYTE >03,>CD,>11,>D1,40
        BYTE >04,>8D,>11,>91,>DF,40
        BYTE >03,>86,>0D,>91,40
        BYTE >06,>8E,>0B,>AD,>11,>93,>B3,40
        BYTE >09,>8A,>0A,>A6,>0D,>CD,>11,>95,>B5,>D5,60
        BYTE >05,>86,>0D,>91,>BF,>DF,20
        BYTE >03,>82,>0E,>91,40
        BYTE >03,>8E,>0F,>91,40
        BYTE >03,>80,>0A,>91,40
        BYTE >04,>A0,>0A,>9F,>B1,40
        BYTE >06,>80,>0A,>AD,>10,>93,>B3,60
        BYTE >04,>80,>0A,>91,>BF,20
        BYTE >04,>A0,>0A,>9F,>B1,40
        BYTE >09,>8A,>0A,>A6,>0D,>CD,>11,>95,>B5,>D5,50
        BYTE >05,>8E,>0B,>91,>BF,>DF,30
        BYTE >03,>86,>0D,>91,40
        BYTE >09,>82,>0E,>AD,>11,>CD,>17,>95,>B5,>D5,40
        BYTE >05,>86,>0D,>91,>BF,>DF,40
        BYTE >03,>8E,>0B,>91,40
        BYTE >03,>86,>0D,>91,100
        BYTE >01,>FF,0
        END
```

■

SAMLINGSSKIVOR FRÅN PB

Du kan beställa en samlingsskiva från PB-91 för 30 kr som betalas till föreningens postgirokonto 19 83 00-6. Skivan innehåller program från tidningen 1991.

Du kan även beställa samlingsskivor från NN-83, PB-84, PB-85, PB-86, PB-87, PB-88, PB-89 och PB-90. Se skivkatalogen från skivorna. Pris 30 kr per skiva (porto ingår).

PROGBIT-90 Sectors Used = 355 Free = 5 Filecount 42

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
BASKONVERT	4	Program	BX		ORACLE	7	Program	BX	
CHARFIX	19*	Dis/Var	163		OTHELLO	27	Program	BX	
COINC/XB	2	Program	BX		PERISCOPE	28	Program	BX	
DATAFIL-A	2	Program	BX		REMOVER	2	Program	BX	
DATAFIL-B	2	Program	BX		SAMPL-EA/O	6	Dis/Fix	80	
DATAFIL-C	2	Program	BX		SAMPL-EA/S	41	Dis/Var	80	
DATAFIL-D	4	Program	BX		SAMPL-MM/O	6	Dis/Fix	80	
DATAFIL-E	5	Program	BX		SAMPL-MM/S	48	Dis/Var	80	
DATAFIL-F	2	Program	BX		SCHEMA_P	25	Program		
DATAFIL-G	2	Program	BX		SEDUCTION	14	Program	BX	
DATAFIL-H	2	Program	BX		SPELLING2	4	Program	BX	
DATAFIL-I	2	Program	BX		TML-AUTO	4	Program	BX	
DATAFIL-J	3	Program	BX		TML-RITA	5	Program	BX	
DATAFIL-K	3	Program	BX		TML-SPRITE	4	Program	BX	
DATAWRITER	7	Program	BX		TRACK/DOC	3	Dis/Var	80	
DEMO74	6	Dis/Var	80		TRACK/O	3	Dis/Fix	80	
DISKMENU	5	Program	BX		TRACK/S	3	Dis/Var	80	
FIX	2	Program	BX		TRACK/USE	2	Program	BX	
FIX/DOC	9	Dis/Var	80		UCSD/ARC	16	Int/Fix	128	
LOADMAKER	12	Program	BX		UCSD/O	2	Dis/Fix	80	
NUTRITION	4	Program	BX		UCSD/S	4	Dis/Var	80	

PROGBIT-91 Sectors Used = 360 Free = 0 Filecount 41

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
ALSAVE	6	Dis/Fix	80		LOWSOUNDS	3	Program	BX	
BASSNOTES	3	Program	BX		MC-TUT1/O	3	Dis/Fix	80	
BATTLESTAR	19	Program	BX		MC-TUT2/O	4	Dis/Fix	80	
BELLTONES	3	Program	BX		MC-TUT3/O	6	Dis/Fix	80	
BUBBLE	4	Program	EA		MC-TUT4/O	4	Dis/Fix	80	
BUBBLE/ALS	6	Program	BX		MC-TUT5/O	3	Dis/Fix	80	
BUBBLE/BH	10	Program	BX		MC-TUT6/O	5	Dis/Fix	80	
BUBBLE/O	11	Dis/Fix	80		MELTDOWN	41	Program	BX	
BUG-AL6/O	5	Dis/Fix	80		NIM	22	Program	BX	
BUG-AL8/O	5	Dis/Fix	80		NOISE	4	Program	BX	
CONVERTER	10	Program	BX		NUMBER/SS	4	Program	BX	
DATABAS/DS	19	Program	BX		PRIMTAL3	2	Program	BX	
DATAFIL2	23	Int/Fix	32		PSCANFORM	4	Program	BX	
ERASEDEMO	5	Program	BX		SILENTNITE	11	Program	BX	
GPLLNK	5	Dis/Var	80		SOL-PRINT	21	Program	BX	
HELLO	8	Program	BX		SOL-VISA	19	Program	BX	
INIT	10	Dis/Var	80		SORTERING	15	Program	BX	
K-SIFFRA	8	Program	BX		SOUNDTEST	3	Program	BX	
KSCAN	3*	Dis/Var	80		VDP-UTIL	5	Dis/Var	80	
LINK/O	4	Dis/Fix	80		VDPREG/O	3	Dis/Fix	80	
LOVESTORY	9	Program	BX						

NITTINI-83 Sectors Used = 213 Free = 147 Filecount 23

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
ATTACK	14	Program	BX		LOTTO/BG	3	Program	BX	
BASPRITE	3	Program	BX		MG-SPRITE	3	Program	BX	
BOKSTAV	6	Program	BX		MMSPRITE2	3	Program	BX	
CHECK	7	Program	BX		NRSÄNDARE	13	Program	BX	
DISPLAY	3	Program	BX		PERSPEKTIV	4	Program	BX	
DOMINION	8	Program	BX		PUCMAN	32	Program	BX	
HANGMAN	24	Program	BX		QUICKSORT	5	Dis/Var	163	
JAGAD	7	Program	BX		SHELLSORT	3	Dis/Var	163	
JUSTERING	3	Program	BX		SIFFERJAKT	14	Program	BX	
KORSORD	7	Program	BX		STOCKHOLM	33	Program	BX	
KORTLEK	3	Program	BX		TFNRING/XB	9	Program	BX	
LEDTEXT	4	Program	BX						

PROGBIT-84 Sectors Used = 360 Free = 0 Filecount 35

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
ADVENTURE	25	Program	BX		POLSKNOTA	9	Program	BX	
AUTOSPRITE	6	Program	BX		POPPA	3	Program	BX	
BARNVAKTEN	14	Program	BX		PRATOR1	3	Program	BX	
BASKONVERT	4	Program	BX		PRATOR2	5	Program	BX	
CATALOGRUN	6	Program	BX		SCREEN-RAM	2	Program	BX	
CATTEXT	7	Program	BX		SCREENSAVE	12	Program	BX	
COLORDRAW	5	Program	BX		SORT-PROCE	4	Program	BX	
COLOREDIT	6	Program	BX		SORT-SUBPR	4	Program	BX	
CURSOR	3	Program	BX		SPR-MAKER	18	Program	BX	
FILKONVERT	4	Program	BX		TELEFONREG	18	Program	BX	
HEXDECBIN	6	Program	BX		TERRORIST	40	Program	BX	
LISTAPROGR	8	Program	BX		TESTBILD	11	Program	BX	
LOADASMBAS	33	Program	BX		TEXTMODE	5	Program	BX	
MULTICOLOR	10	Program	BX		TI-59	16	Program	BX	
P:TEXTIN	12	Program	BX		TREMOLO	4	Program	BX	
P:TEXTUT	7	Program	BX		TRUNKERING	17	Program	BX	
P:TILLCOMP	14	Program	BX		VDP-REG	4	Program	BX	
PIANO	13	Program	BX						

PROGBIT-85 Sectors Used = 359 Free = 1 Filecount 33

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
CATALOG	3	Program	BX		PEEKV/XB	5	Program	BX	
CHARFILTER	3	Dis/Var	163		PROGLOAD	9	Program	BX	
CSAVE1O	10	Dis/Fix	80		RITAKURVA	8	Program	BX	
CSAVE1S	34	Dis/Var	80		RITAKURVAK	6	Program	BX	
DART	19	Program	BX		SNABBFIL	18	Program	BX	
INDEXSORT	3	Program	BX		STRSORT	3	Program	BX	
JOYST	2	Dis/Var	163		STRSORTXB	3	Program	BX	
LANDER	12	Program	BX		SVENSKABA	2	Dis/Var	163	
LIST28	7	Program	BX		SVENSKAXB	2	Dis/Var	163	
LISTAORD	5	Program	BX		TAPETESTO	3	Dis/Fix	80	
LTWRITMINI	18	Program	BX		TAPETESTS	8	Dis/Var	80	
MINIASSEM	23	Program	BX		TIPSRAD	11	Program	BX	
MINICHIME	33	Program	BX		TOTO-G	15	Program	BX	
MULTISAY	3	Program	BX		TRAIN	8	Program	BX	
NUC	24	Program	BX		TRIPPELPKT	24	Program	BX	
OHMSLAG	16	Program	BX		WIND	4	Program	BX	
ORMAR	13	Program	BX						

PROGBIT-86 Sectors Used = 359 Free = 1 Filecount 14

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
BANKKONTO	35*	Program	BX		MM-MASK	18	Program	BX	
HELIKOPTER	42	Program	BX		MUSIK1	6	Program	BX	
K:INPUT	32	Dis/Var	80		MUSIK2	7	Program	BX	
KALKYLATOR	17	Program	BX		SORTDEMO	60	Int/Var	254	
KORSORD	10	Program	BX		SPR-MAKER2	18	Program	BX	
LADDA/XB	17	Program	BX		STJÄRNKIKA	35	Program	BX	
MINIASMBAS	16	Program	BX		SVERIGE	44	Program	BX	

PROGBIT-87 Sectors Used = 337 Free = 23 Filecount 17

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
42-SPELET	33	Program	BX		LOAD/BERNL	15	Program	BX	
ALFASLANG	27	Program	BX		MENY	3	Program	BX	
AUTOFILE	8	Program	BX		MINEFIELD	19	Program	BX	
CHAR_SET	9	Dis/Fix	80		MITEC;C	34	Dis/Var	80	
CHAR_SETS	23	Dis/Var	80		PÅSK	9	Program	BX	
DIETPROG	8	Dis/Var	80		TECKENSMED	27	Program	BX	
HCHAR:ASS	29	Dis/Var	80		TIWRLOG1S	50*	Dis/Var	80	
HCHAR:REL	5	Dis/Fix	80		TOADY	33*	Program	BX	
HCHARTEST	3	Program	BX						

PROGBIT-88 Sectors Used = 334 Free = 26 Filecount 21

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
AVRUNDNING	2	Program	BX		PRK-PRINT	3	Program	BX	
BASICPRO1	3	Program	BX		PRK-SNABB2	14	Program	BX	
BASICPRO2	3	Program	BX		SATELLIT	10	Program	BX	
CHECKPGM	89	Dis/Var	80		STA-CALL	6	Program	BX	
DUBBELSPEL	21	Program	BX		STA-MERGE	4	Program	BX	
HUNT	30	Program	BX		STARTSPRIT	4	Program	BX	
LISTSKYDD	4	Program	BX		STYRSPRITE	7	Program	BX	
LOUDSPEAK	27	Program	BX		SVERIGE	49	Int/Var	254	
POKEVPEEKV	11	Program	BX		TEX	23	Program	BX	
PRK-CALL	11*	Program	BX		TEX-LOAD	6	Program	BX	
PRK-MERGE	5	Program	BX						

PROGBIT-89 Sectors Used = 340 Free = 20 Filecount 22

Filename	Size	Type	Rec	P	Filename	Size	Type	Rec	P
ARTLES/G6	2	Program	BX		MICRO/S	4	Int/Fix	255	
CR*ADDER	3	Program	BX		MOTION/G6	3	Program	BX	
FILE*PRINT	8	Program	BX		PAYMENT	8	Program	BX	
FILSCR/G7	3	Program	BX		PB/D	66	Int/Fix	85	
LABEL	12	Program	BX		PB/S	4	Int/Fix	255	
LF*STRIPER	3	Program	BX		PRINT-MP/C	4	Dis/Var	80	
LOAD/G6	3	Program	BX		PRINT-RD/C	4	Dis/Var	80	
LOAD/G7	3	Program	BX		R10	37	Program	BX	
LOADER/JAN	8	Program	BX		R11	49*	Program	BX	
LOTTO/SS	15	Program	BX		STYRA-DM99	9	Program	BX	
MICRO/D	79	Int/Fix	85		ZOMBIE	11	Program	BX	■

PROGRAMMING MUSIC - PART 1

by Jim Peterson, Tigercub, USA

A while ago, I wrote an article about music programming in which I said that it was easy but that you almost had to know how to read music. Well, it is still easy to program, but no longer necessary to know how to read it.

Personally, I am about like the country fiddler who admitted that he could read music a little, but not enough to hurt his playing. I know just a little about reading music but that has been all I needed to know to program more than 50 songs. And, if you have ever heard my Tigercub Country or Tigercub Gospel disks, you will know that I have programmed those songs in a wide variety of styles.

Now, I have put together a few little routines to enable anyone to program music on the TI-99/4A very easily, and in many ways. You DON'T need to know how to program and you DON'T need to know how to read music!

First, key in this one-liner and save it as DSK1.SCALE,MERGE

```
100 DIM N(36):: F=110 :: FOR
 J=1 TO 36 :: N(J)=INT(F*1.0
59463094^(J-1)+.5):: NEXT J
:: N(0)=40000
```

Next, NEW to clear memory and then key in this music program, which we will use as an example to experiment with.

```
110 T=2 :: A=13 :: GOSUB 100
0 :: T=1 :: A=18 :: GOSUB 10
00 :: GOSUB 1000 :: T=3 :: G
OSUB 1000
120 T=1 :: A=20 :: GOSUB 100
0 :: A=22 :: GOSUB 1000 :: A
=23 :: GOSUB 1000 :: T=2 ::
A=27 :: GOSUB 1000 :: T=4 ::
 A=25 :: GOSUB 1000
130 T=1 :: A=30 :: GOSUB 100
0 :: A=29 :: GOSUB 1000 :: T
=5 :: A=27 :: GOSUB 1000
140 T=1 :: A=25 :: GOSUB 100
0 :: A=27 :: GOSUB 1000 :: A
=25 :: GOSUB 1000 :: A=22 ::
 GOSUB 1000 :: T=5 :: A=25 :
: GOSUB 1000 :: T=2 :: GOSUB
 1000
150 T=1 :: A=27 :: GOSUB 100
0 :: GOSUB 1000 :: T=3 :: GO
SUB 1000 :: T=1 :: A=22 :: G
OSUB 1000
160 A=25 :: GOSUB 1000 :: A=
22 :: GOSUB 1000 :: T=2 :: A
=20 :: GOSUB 1000 :: T=4 ::
A=18 :: GOSUB 1000
170 T=1 :: GOSUB 1000 :: A=2
0 :: GOSUB 1000 :: T=5 :: A=
22 :: GOSUB 1000 :: T=1 :: A
=18 :: GOSUB 1000
180 A=22 :: GOSUB 1000 :: A=
27 :: GOSUB 1000 :: T=6 :: A
=25 :: GOSUB 1000 :: T=1 ::
A=18 :: GOSUB 1000 :: A=20 :
: GOSUB 1000
190 T=6 :: A=22 :: GOSUB 100
0 :: T=2 :: A=18 :: GOSUB 10
00 :: A=20 :: GOSUB 1000 ::
T=4 :: A=18 :: GOSUB 1000 ::
 STOP
```

Save that by SAVE DSK1.SHEN just so you don't lose it, but keep it in memory, and enter MERGE DSK1.SCALE to get that one-liner back in.

The music you just keyed in is in one voice without harmony. Let's see what you can do with just one voice. Put in a line 105 D=200 and another line - 1000 CALL SOUND(T*D,N(A),0) :: RETURN

Enter RUN, wait a second, and listen. If you didn't make any mistakes in keying in the music, you should hear a fairly pleasant single-note rendition of a beautiful old folk song.

Maybe you would prefer a higher key? Here's the neat part about starting with that formula in line 100 - besides the fact that it lets you key in frequencies in shorthand. To change key, just change that 110 in line 100 to a higher frequency number. They are listed in the "blue book" that came with your computer, but if you lost it they go upward 110, 117, 123, 131, 139, 147, 156, 165, 175, 185, 196, 208, 220. You

can also lower the key, providing you do not cause the lowest note in your music to go below frequency 110. In the piece you keyed in, the lowest note number used was 13 so you could go down 12 steps. The frequencies are not in the book, but they go 110, 104, 98, 92, 87, 82, 78, 73, 69, 65.

Want the music faster or slower? Just change the 200 in line 105.

Now let's see what else we can do with single-note music. Try this -

```
1000 CALL SOUND(T*D,N(A),0,N
(A)*1.01,0):: RETURN
```

Has a richer sound, doesn't it? How about this?

```
1000 CALL SOUND(T*D,N(A),0,N
(A)/2,0):: RETURN
```

Or combine the two -

```
1000 CALL SOUND(T*D,N(A),0,N
(A)*1.01,0,N(A)/2,0):: RETUR
N
```

Multiplying a note by 1.01 in another voice will always give a more resonant sound, and dividing a note by two (providing its note number is not less than 13) will always be in harmony - so will multiplying by two, or by four.

How about some real deep down bass music? The TI's tone generators can only go down to frequency 110, but the noise generator can be tuned far below that. The timber of the sound is different and doesn't blend too well with the tones, so use it with caution - but it's great for a tuba solo. Try this -

```
1000 CALL SOUND(T*D,N(0),30,
N(0),30,N(A)*3.75,30,-4,0)::
RETURN
```

Want to go deeper? Try changing the 3.75 to 1.875 - too deep to even be musical, isn't it? Maybe you could improve it by raising the frequency in line 100.

Try changing the 3.75 to 7.5 - not bad, is it? So try doubling it again to 15 - oops! When you go that high you get some very sour notes!

So, go back to 7.5 and change one of those N(0) to N(A) and change the 30 following it to 0. Pretty good, so try also changing the other N(0) to N(A)*1.01 and the 30 after it to 0.

If any of those effects sound like something you might want to try in a piece of music someday, clear the memory with NEW, key it in and save it with SAVE DSK....,MERGE using a different filename for each one. Then, after you have keyed in some music, you can very quickly merge in different routines and try them. You will find that different ones go better with different songs.

The routines we have been trying all play music with a very strong beat. For a smoother effect, try this -

```
1000 FOR J=1 TO T :: CALL SO
SOUND(-2999,N(A),0):: GOSUB
1100 :: NEXT J :: RETURN
1100 FOR D=1 TO 99 :: NEXT D
 :: RETURN
```

You will notice one thing right away; with this method, a series of the same note gets run together into one long note. Later we will look at ways to get around that.

To change the tempo of the music, just change the value of 99 in line 1100. Try this method in combination with the effects we tried previously.

Here's another one that gives a very nice effect -

```
1000 FOR J=1 TO T :: CALL SO
UND(-999,N(A),0):: GOSUB 110
0 :: CALL SOUND(-999,N(A)*1.
01,0):: GOSUB 1100 :: NEXT J
 :: RETURN
1100 FOR D=1 TO 8 :: NEXT D
:: RETURN
```

Or for a more mournful sound -

```
1000 FOR J=1 TO T*4 :: CALL
SOUND(-999,N(A),0):: CALL SO
UND(-999,N(A)*1.01,0):: NEXT
J :: RETURN
```

You can control the tempo by changing the value of 4, but not as precisely as with the previous method, and it does not work well with bass notes. Try changing the 1.01 to 1.02 - also try erasing the *1.01 and change the following 0 to 8, for a mandolin effect.

Those are just a few of the effects you can create with just a single-note melody - experiment and see what else you can discover.

So, just imagine what you will be able to do using all three voices - coming up in part 2 of this article!■

NOTUNG SOFTWARE
7647 McGroarty Street,
TUJUNGA, CA 91042, USA
(Porto USD 2.50 extra per order)

Bride of Disk of Dinosaurs	USD 12
Son of Disk of Dinosaurs	USD 10
Original disk of Dinosaurs	USD 7
Fonts & Borders IV	USD 8
Disk of Horrors	USD 12
Disk of Pyrates	USD 10
Casino version 3 (01 OCT 91)	USD 15■

ASGARD SOFTWARE
P.O.BOX 10306,
ROCKVILLE, MD 20849, USA
(Porto USD 7.00 extra per order)

ARCADE ACTION GAMES		
Colors(lik Tetris)	disk E9102	USD 13
Rock Runner	disk E9107	USD 13
Starbase Raiders	disk E9108	USD 13
War Zone	disk E9152	USD 13
Boxsteine	disk E9153	USD 13
Karate Challenge	disk E9104	USD 8
Mission Destruct	disk E9105	USD 8
INFOCOM TEXT ADVENTURES		
Plundered Hearts	disk E9141	USD 15
Suspect	disk E9142	USD 15
Hollywood Hijinx	disk E9143	USD 15
Station Fall	disk E9144	USD 15
INFOCOM ADVENTURES FOR SUPERCART 8kB		
The Lurking Horror	disk E9145	USD 15
Leather God.of Phobos	E9146	USD 15■

INVADER - SPEL FÖR XB

```
100 REM ***********
110 REM * N-VADER *
120 REM ***********
130 REM BY J.R.DEW
140 REM 99'ER 82-06 XB
170 CALL CHAR(104,"FF99FFFFA
5A5A5A5"):: CALL CLEAR
210 FOR X=1 TO 20 :: FOR Y=1
 TO 19 STEP 9 :: DISPLAY AT(
X,Y)SIZE(8):"N-VADER" :: NEX
T Y :: NEXT X
240 CALL SPRITE(#1,104,11,1,
1,0,20)
250 DISPLAY AT(22,2):"PRESS
ENTER TO PLAY"
260 CALL SPRITE(#2,104,13,65
,256,0,-20,#3,104,9,73,1,0,1
00,#4,104,4,81,256,0,-99)
290 ACCEPT AT(22,22)BEEP:X$
300 CALL DELSPRITE(ALL)
310 DISPLAY AT(24,1):"INSTRU
CTIONS (Y/N)? N" :: ACCEPT A
T(24,21)SIZE(-1)VALIDATE("YN
"):X$
320 IF X$<>"Y" THEN 490
330 CALL CLEAR
340 PRINT "ALIEN CREATURES A
RE":"ATTACKING THE EARTH!":"
YOU COMMAND THE ONLY DEFENSE
SHIP. ONBOARD COMPUTERS
CONTROL THE LASER WHICH CAN"
350 PRINT "DESTROY THE INVAD
ERS.":"THE INVADERS WILL NOT
 ATTACKYOU, ONLY THE EARTH"
370 CALL SPRITE(#1,104,11,1,
1,0,20)
380 PRINT
390 INPUT "HIT ENTER WHEN RE
ADY":X$
400 CALL CLEAR
410 PRINT "IN THE GAME, YOU
CONTROL":"THE NUMBER OF INVA
DERS,":"THEIR SPEED, YOUR SP
EED &":"THE LASER RANGE OF Y
OUR SHIP"
420 PRINT "SUGGESTED VALUES
ARE:":"INVADERS=6,SPEED=8":"
YOUR SPEED=3,RANGE=25"
430 PRINT "A SMALLER RANGE M
AKES THE":"GAME HARDER."
440 PRINT "YOU ALSO CONTROL
THE LENGTH OF THE GAME. WHEN
 ASKED":"'END OF GAME', ENTE
R THE"
450 PRINT "NUMBER OF TIMES T
HE ALIENS  HIT EARTH FOR THE
 GAME BE   OVER.": :
470 INPUT "ENTER WHEN READY"
:X$
480 CALL DELSPRITE(#1)
490 IF HIT>0 THEN HIT,ZAP1,Z
AP2=0 :: GOTO 750
500 CALL CLEAR :: DISPLAY AT
(1,1):"NUMBER OF PLAYERS? 1"
 :: ACCEPT AT(1,20)SIZE(-1)V
ALIDATE("12"):NP
530 DISPLAY AT(3,1):"PLAYER
1 NAME?" :: ACCEPT AT(3,16):
P1$ :: IF NP=1 THEN 560
550 DISPLAY AT(5,1):"PLAYER
2 NAME?" :: ACCEPT AT(5,16):
P2$
560 DISPLAY AT(7,1):"NUMBER
OF INVADERS? 6" :: ACCEPT AT
(7,21)SIZE(-1)VALIDATE("1234
5678"):INV
```

```
590 DISPLAY AT(9,1):"INVADER
 SPEED? 8" :: ACCEPT AT(9,16
)SIZE(-2)VALIDATE(DIGIT):IS
:: IF IS<1 THEN 590
620 DISPLAY AT(11,1):"DEFEND
ER SPEED(1-9)? 3" :: ACCEPT
AT(11,22)SIZE(-1)VALIDATE(DI
GIT):SPD :: IF SPD<=0 THEN 6
20
650 DISPLAY AT(13,1):"DEFENC
E RANGE? 25" :: ACCEPT AT(13
,16)SIZE(-3)VALIDATE(DIGIT):
RNG :: IF RNG<1 OR RNG>200 T
HEN 650
680 DISPLAY AT(15,1):"END OF
 GAME? 5" :: ACCEPT AT(15,14
)SIZE(-2)VALIDATE(DIGIT):WIN
 :: IF WIN<1 THEN 680
700 DISPLAY AT(17,1):"JOYSTI
CKS (Y/N)? Y" :: ACCEPT AT(1
7,18)SIZE(-1)VALIDATE("YN"):
X$ :: IF X$="Y" THEN JS=1
730 CALL CHAR(96,"0008081C7F
1C0808",100,"FFFFFFFFFFFFFFF
F")
750 CALL SCREEN(2):: CALL CL
EAR
770 CALL COLOR(9,16,16,3,2,3
,4,2,3)
800 CALL HCHAR(22,1,100,96)
830 FOR X=1 TO INV
840 CALL SPRITE(#X,104,3+X,1
,INT(RND*256)+1,INT(RND*IS)+
1,INT(RND*IS)-IS/2)
850 NEXT X
860 IF NP=1 THEN CALL SPRITE
(#9,96,16,100,128)ELSE CALL
SPRITE(#9,96,16,100,56)
870 IF NP=2 THEN CALL SPRITE
(#10,96,15,100,200)
880 FOR X=1 TO INV
890 CALL COINC(#X,#9,RNG,B)
900 IF NP=2 THEN CALL COINC(
#X,#10,RNG,B2)
910 IF B>=0 AND B2>=0 THEN 1
020
920 CALL PATTERN(#X,100)
930 CALL SOUND(-500,-3,0)
940 GOSUB 1360
950 IF B>=0 THEN 980
960 ZAP1=ZAP1+1
970 DISPLAY AT(23,3)SIZE(4):
ZAP1
980 IF B2>=0 THEN 1080
990 ZAP2=ZAP2+1
1000 DISPLAY AT(23,23)SIZE(4
):ZAP2
1010 GOTO 1080
1020 CALL POSITION(#X,V(X),H
(X))
1030 IF V(X)<158 THEN 1080
1040 GOSUB 1360
1050 CALL SOUND(-50,-2,0)
1060 HIT=HIT+1
1070 DISPLAY AT(23,14)SIZE(4
):HIT
1080 IF JS=1 THEN CALL JOYST
(1,JX,JY)ELSE CALL KEYST(1,J
X,JY)
1090 CALL MOTION(#9,-JY*SPD,
JX*SPD)
1100 IF NP=1 THEN 1130
1110 IF JS=1 THEN CALL JOYST
(2,JX,JY)ELSE CALL KEYST(2,J
X,JY)
1120 CALL MOTION(#10,-JY*SPD
,JX*SPD)
1130 IF HIT<WIN THEN 1340
1140 CALL DELSPRITE(ALL):: C
ALL SCREEN(16):: CALL CLEAR
1170 CALL COLOR(3,2,1,4,2,1)
1190 CALL SPRITE(#1,104,7,1,
1,0,25)
1200 PRINT "GAME OVER": :"EA
RTH HITS";HIT
1220 PRINT :P1$;" DESTROYED"
;ZAP1;"ALIENS":TAB(10);INT(1
00*ZAP1/(HIT+ZAP1+ZAP2));"PE
R CENT"
1240 IF NP=2 THEN PRINT :P2$
;" DESTROYED";ZAP2;"ALIENS":
TAB(10);INT(100*ZAP2/(HIT+ZA
P1+ZAP2));"PERCENT"
1260 FOR X=1 TO 10 :: CALL S
OUND(50,440,0):: CALL SOUND(
99,880,0):: NEXT X
1270 FOR X=1 TO 500 :: NEXT
X
1300 CALL DELSPRITE(#1)
1310 CALL SCREEN(8)
1330 GOTO 490
1340 NEXT X
1350 GOTO 880
1360 CALL DELSPRITE(#X)
1370 CALL SPRITE(#X,104,15-X
,1,INT(RND*256)+1,INT(RND*IS
)+1,INT(RND*IS)-IS/2)
1380 RETURN
1390 SUB KEYST(N,X,Y)
1400 CALL KEY(N,K,S)
1410 IF S=0 THEN X,Y=0 :: SU
BEXIT
1420 IF K=2 THEN X=-4 :: Y=0
1430 IF K=4 THEN X=-4 :: Y=4
1440 IF K=5 THEN X=0 :: Y=4
1450 IF K=6 THEN X,Y=4
1470 IF K=3 THEN X=4 :: Y=0
1480 IF K=14 THEN X=4 :: Y=-
4
1490 IF K+1=1 THEN X=0 :: Y=
-4
1500 IF K=15 THEN X=-4 :: Y=
-4
1510 SUBEND                    ■
```

TEXT1 & TEXT2 I ASSEMBLER

av Jan Alexandersson

TEXT1 FÖR TI-99/4A

Text1 delar upp skärmen i rutor med 24 rader och 40 kolumner. Varje sådan ruta kan fyllas med ett av 256 möjliga grafiska tecken. Varje teckenruta består av 8x6 punkter. Hela skärmen kan samtidigt endast använda två färger. Samtliga tecken måste således ha samma förgrunds- och bakgrundsfärg. Du kan dock välja dessa två färger bland 16 olika färger. Text1 kan inte använda sprites.

Du måste initiera videoprocessorns 8 olika register (#0-#7) för att kunna använda Text1:

```
#0  >00         binärt 0000 0000
#1  >F0         binärt 1111 0000
#2  värdet x >400  = Screen Table
#3  användes ej
#4  värdet x 800   = Pattern Table
#5  användes ej
#6  användes ej
#7  förgrundsfärg/bakgrundsfärg
```

TABELL 99/4A	REG	BYTES	PLATSER
SCREEN	2	960	16
PATTERN	4	2048	8
TOTALT		3008	4

I normala fall räcker det med en uppsättning PATTERN. Du kan då i praktiken ha c:a 10 olika skärmar med text eftersom diskbuffert och PATTERN måste ha sin del av minnet.

```
**********************
* Text1 DEMO PROGRAM 1
* for 99/4A
* by Jan Alexandersson
* 1991-09-03
**********************
       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   @T14A      text1

****** print message on the screen
       LI   R0,5*40+13+SCRTA1
       LI   R1,TEXT1
       LI   R2,13
       BLWP @VMBW

       LI   R0,23*40+10+SCRTA1
       LI   R1,TEXT2
       LI   R2,19
       BLWP @VMBW

****** 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

TEXT1  TEXT 'Test of TEXT1'
TEXT2  TEXT 'PRESS ENTER TO EXIT'
       EVEN

       COPY "DSK2.T1-TAB-4A"
       COPY "DSK2.T14A"
       COPY "DSK2.BLKVDP"     PB91-6
       COPY "DSK2.G1-TAB-4A"  PB91-6
       COPY "DSK2.G14A"       PB91-6

       END
```

```
*****************************
* T1-TAB-4A tables for 99/4A
* in Text1 mode
*****************************

SCREE1 EQU  0
SCREN1 EQU  24*40
PATTE1 EQU  1
PATEN1 EQU  >800
FORBAK EQU  >F6      white on red

SCRTA1 EQU  SCREE1*>400
PATTA1 EQU  PATTE1*>800


**********************
* T14A initialize
```

```
* Text1 for 99/4A
* by Jan Alexandersson
* 1991-09-03
***********************

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

T14A   MOV  R11,R10      save return
       LI   R0,>01B0     text1 off
       BLWP @VWTR
       CLR  R0           VDP-reg #0
       BLWP @VWTR
****** screen table
       LI   R0,>0200+SCREE1
       BLWP @VWTR
****** pattern table
       LI   R0,>0400+PATTE1
       BLWP @VWTR
****** character and screen colour
       LI   R0,>0700+FORBAK
       BLWP @VWTR
****** standard keyboard = 0
       CLR  R0
       MOVB R0,@KEYBRD

****** clear screen table
       BL   @BLKVDP
       DATA SCRTA1,>2000,SCREN1

****** clear pattern table
       BL   @BLKVDP
       DATA PATTA1,0,PATEN1

****** normal char 32-95
       LI   R0,PATTA1+>100
       MOV  R0,@FAC
       CLR  R1
       MOVB R1,@STATUS
       BLWP @GPLLNK
       DATA >18

****** lower char 96-127
       LI   R0,PATTA1+>300
       MOV  R0,@FAC
       MOVB R1,@STATUS
       BLWP @GPLLNK
       DATA >4A

       LI   R0,>01F0     text1 on
       BLWP @VWTR
       SWPB R0
       MOVB R0,@>83D4    for KSCAN

       B    *R10         return
```

Du kan även använda Text1 för 9938 och har då möjlighet att välja 24 eller 26,5 rader. Med det större antalet rader användes minnet enligt följande.

TABELL 9938 T1	REG	BYTES	PLATSER
SCREEN	2	1080	128
PATTERN	4	2048	64
TOTALT		3128	32
BASE	14	16384	8

TEXT2 FÖR 80-KOLUMNSKORT MED 9938

Text2 med 9938 ger möjlighet till 80 kolumner och 24/26,5 rader. Fyra färger kan användas samtidigt och de kan väljas ur en palett på 512 färger. Initiera VDP-registren enligt följande.

```
#0  >04         binärt 0000 0100
#1  >70         binärt 0111 0000
#2  värdet x >400  = Screen Table *
#3  värdet x >40   = Color Table **
#4  värdet x 800   = Pattern Table
#5  användes ej
#6  användes ej
#7  förgrunds-/bakgrundsfärg, normal
#8  >88         aktiverar musen
#9  >00         ger 60 Hz skärmväxling
#10 Extended Color Table
#12 förgrunds-/bakgrundsfärg, blink
#13 On time/Off time, Blink Period
#14 Base Address
```

(*) Screen Table ska ha de två minst signifikanta bitarna satta till "1" så att >03 adderas till adressen som endast får ha värdet >00, >04, >08, >0C, >10, >14, >18, >1C o.s.v.
(**) Color Table ska ha de tre minst signifikanta bitarna satta till "1" så att >07 adderas till adressen som endast får ha värdet >00, >08, >10, >18, >20, >28, >30, >38 o.s.v.

Det är möjligt att få bokstäver att blinka mellan färger i #7 och #12. Man markerar i Color Table vilka bokstäver som ska blinka. Varje skärmposition i Screen Table har en egen bit i Color Table för att markera detta. Genom att sätta Off Time i #13 till 0 så kan blinkfärgen vara på hela tiden så att man i praktiken får fyra stadiga färger samtidigt.

TABELL 9938 T2	REG	BYTES	PLATSER
SCREEN	2	2160	32
COLOR	10+3	270	256
PATTERN	4	2048	64
TOTALT		4478	16-32
BASE	14	16384	8

```
**********************
* Text2 DEMO PROGRAM 1
* for 9938
* by Jan Alexandersson
* 1991-09-03
**********************

       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   @T2         text2

****** print message on the screen
       LI   R0,5*80+29+SCRTA2
       LI   R1,TEXT1
       LI   R2,22
       BLWP @VMBW

       LI   R0,25*80+30+SCRTA2
       LI   R1,TEXT2
       LI   R2,19
       BLWP @VMBW

******* second message with blinking
       BL   @BLKVDP
       DATA COLTA2+250+3,>FF00,4

****** 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

TEXT1  TEXT 'Test of TEXT2 for 9938'
TEXT2  TEXT 'PRESS ENTER TO EXIT'
       EVEN

       COPY "DSK2.T2-TAB-38"
       COPY "DSK2.T2"
       COPY "DSK2.BLKVDP"     PB91-6
       COPY "DSK2.G1-TAB-4A"  PB91-6
       COPY "DSK2.G14A"       PB91-6

       END
```

```
**************************
* T2-TAB-38 tables for 9938
* in Text2 mode
**************************

SCREE2 EQU  0
SCREN2 EQU  27*80
PATTE2 EQU  2
PATEN2 EQU  >800
COLOR2 EQU  >28
COLEN2 EQU  27*80/8
COLEXT EQU  0
FORBAK EQU  >13       black on green
BLICOL EQU  >F6       white on red

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


**********************
* T2 initialize
* Text2 for 9938
* by Jan Alexandersson
* 1991-09-03
**********************

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

T2     MOV  R11,R10      save return
       LI   R0,>0130     text2 off
       BLWP @VWTR
       LI   R0,>0004     text2
       BLWP @VWTR
       LI   R0,>0888     mouse on
       BLWP @VWTR
       LI   R0,>0980     212pix,60Hz
       BLWP @VWTR

****** screen table
       LI   R0,>0200+SCREE2+>03
       BLWP @VWTR
****** pattern table
       LI   R0,>0400+PATTE2
       BLWP @VWTR
****** colour table
       LI   R0,>0300+COLOR2+>07
       BLWP @VWTR
****** extended colour table
       LI   R0,>0A00+COLEXT
       BLWP @VWTR
****** character and screen colour
       LI   R0,>0700+FORBAK
       BLWP @VWTR
****** second character colours
       LI   R0,>0C00+BLICOL
       BLWP @VWTR
****** blinking colour always on
       LI   R0,>0DF0
       BLWP @VWTR
```

```
       CLR  R0
       MOVB R0,@KEYBRD  keyboard

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

****** clear colour table
       BL   @BLKVDP
       DATA COLTA2,>0000,COLEN2

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

****** normal char 32-95
       LI   R0,PATTA2+>100
       MOV  R0,@FAC
       CLR  R1
       MOVB R1,@STATUS
       BLWP @GPLLNK
       DATA >18

****** lower char 96-127
       LI   R0,PATTA2+>300
       MOV  R0,@FAC
       MOVB R1,@STATUS
       BLWP @GPLLNK
       DATA >4A

       LI   R0,>0170    text2 on
       BLWP @VWTR
       SWPB R0
       MOVB R0,@>83D4   for KSCAN

       B    *R10        return    ■
```

FAST EXTENDED BASIC!
88/01 - XBCAT

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

By the reactions of my friends in Ottawa, it seems that "real programs" are what they like! So I decided to give you a short utility that I have written a few months ago, then reworked a little bit.

This DISK CATALOGUER was developed when I was going through our dear Berry's disk library: I had to read and print many catalogs, and did not want to leave XB and load DM1000 each time. I also wanted a "no frill" program, that would load and run quickly: one input, then zoom, it does what it has to do.

The cataloguing portion proper is not new: it is the Basic program suggested by Texas Instruments, and found all over the place. I added the printing function and the "avail. space only" and "delete" features, and since I work with DSDD disks, which can be quite long to catalog, I wanted the program to stop at my will.

Quite logically, XBCAT first asks you which disk you want to catalog; you can change the default in line 120 ("A=2"). To get a screen catalog, just enter the disk number. If you want a printout, you precede the disk number with a "P" (P1, P2, P3); if on the other hand you just want the available space, you precede it with an "A" (A1, A2, A3). The program then analyzes your answer and reacts accordingly (line 170-190). If you have made a mistake, like trying to catalog a disk with the door open or with no disk in the drive, the ERROR routine in lines 430-470 will take care of it. More on that later.

While the catalog is printing to screen or printer (you cannot do both at the same time), you can stop it by pressing any key. But you must leave your finger on the key until you hear the sound and see the action line at the bottom, then release your finger. You then have the choice to [C]ontinue or [S]top the catalog. Pressing "S" will bring a new action line: you can catalog [A]nother disk, [D]elete a file (when asked, just enter the filename, Tex knows the disk number) or you can [S]top to exit gracefully. That's all there is to it.

CALL ERROR:

In order to prevent the program from breaking under certain conditions, I have added that feature. The error checking routine starts on line 430, therefore you have to tell Tex at

the beginning: "ON ERROR 430" (line 130). If Tex encounters an error, program goes to line 430. The CALL ERR statement always needs four variables, even if you don't use them. C is the error code, T is the type (1 if an I/O error, -1 if another error in program), S is the severity (always 9, don't ask me why) and L is the line number.

The XB manual gives a list of the error codes in Appendix N. If the variable C is 130, that is the error is an I/O one, line 440 will display a message less cryptic than an error code number. Line 450 tells Tex what to do next... A logical statement would have been "ON ERROR 430 :: RETURN 150" (Tex would have asked you the drive number again), but the program just refused to open the file in line 190, even if I had a disk in the drive, with the door closed; the new error was a "FILE ERROR", which I was unable to correct, so I decided not to give it another thought and simply re-RUN the program. Thanks to our Pre-Scan, it starts again with the blink of an eye.

If the error is not an I/O one, the program warns you and then stops, because some errors could be fatal. Very useful for debugging a program!

CATALOGUING A DISK:

If you are not familiar with the way Tex reads a disk and catalogs it, then try to follow me. In line 160, you have given it the number of the disk to catalog, kept in variable A. Line 200 opens the file, with no filename: this will open sector 0 as a file, INPUT because you want to read only, RELATIVE and INTERNAL... because that is how the original program was, I never questioned those attributes, I only know that they work!

Line 210 reads in the name of the disk into A$; I is always 0, and is not needed further; J is the total number of sectors, and K is the number of free sectors. J-K will thus give us the number of used sectors; the diskette is blank IF J=K, since the avail. sectors equal the total sectors on the disk. Line 230 will print that information where you want it: D is the Device file number, always #0 for the screen, here #2 for the printer since the disk to be read is file #1. If you wanted only the available space, or if the diskette is blank, then the program goes to line 340 to close the files and print the appropriate action line. In Tex's peculiar language, "IF AV" implicitely means IF AV<>0; here it is equivalent to IF AV=1, but think of how useful it could be if you just want to check if there is something in variable AV, no matter what! Another quickie from the XB guru.

Starting on line 250, we read the disk directory; don't worry, Tex knows where to find it. 127 is the maximum files you can have on a disk. For each directory entry, Tex reads the filename into A$, the program type into I (see line 150 for an explanation), the number of sectors into J, and the the length of the record into K. If the filename is non existant, i.e. the length of the filename is "0", then the program exits to line 340.

All that information is then printed; if the file is a program (I=5), Tex prints a space, otherwise it prints the length (K) of the record, e.g. 80 for TIW files. If I is negative, the file is write protected, and Tex tells you so. It then checks to see if a key has been pressed; if yes, it prints the first action line; if not (ST=0), it goes on to NEXT N, next directory entry to read.

AND and/or OR:

AV and BL are both checked together in lines 220 and 230, once with AND, once with OR. In line 220, if AV=0 AND BL=0, i.e. if I want a full catalog (not "avail. only") and the diskette is not blank, then Tex prints the message, and only if those two conditions are filled.

In line 230, we want to exit to line 340 on three different occurences: 1) if I want a full catalog but the disk is blank, OR 2) if I want

"avail. only" even with files on the diskette, OR 3) if I want "avail." and the disk is blank. In all cases, only one condition needs to be true (flag AV or BL is "1"); it is the same if both flags are "1", since at least one flag is not "0". Study this chart:

AV	BL	> 0&0	1or1	
0	0	Y	N	print msg, but don't exit to line 340
0	1	N	Y	I want cat., but disk is blank ¦ don't print
1	0	N	Y	disk not blank, but "av. only" ¦ msg but exit
1	1	N	Y	disk is blank, want "av. only" ¦ to line 340

```
100 REM ** XBCAT / v. 2.1 /
Dec. 87 / L. Dorais
110 REM
120 CALL CLEAR :: CALL CHAR(
92,"00FFFFFFFF",96,"000000FF
"):: L$=RPT$("`",28):: LL$=R
PT$("\",28):: A=2
130 DIM T$(5):: ON ERROR 430
140 GOTO 150 :: CALL KEY ::
AV,BL,C,D,DL,I,J,K,KY,L,N,S,
ST,T,A$,B$,M$ !@P-
150 T$(1)="DIS/FIX" :: T$(2)
="DIS/VAR" :: T$(3)="INT/FIX
" :: T$(4)="INT/VAR" :: T$(5
)="PROGRAM"
160 DISPLAY AT(8,1):"CATALOG
 DISK? ";A: : :"** You may p
recede by:": :"   [P] for PR
INTER": :"or [A] for USED/AV
AIL only"
170 ACCEPT AT(8,16)SIZE(-2)V
ALIDATE("123AP")BEEP:A$ :: I
F LEN(A$)=1 THEN A=VAL(A$)::
 D=0 :: M$="scroll" :: GOTO
200
180 IF SEG$(A$,1,1)="A" THEN
 AV=1 :: D=0 :: A=VAL(SEG$(A
$,2,1)):: GOTO 200
190 IF SEG$(A$,1,1)="P" THEN
 D=2 :: A=VAL(SEG$(A$,2,1)):
: M$="print" :: OPEN #2:"PIO
"
200 OPEN #1:"DSK"&STR$(A)&".
",INPUT ,RELATIVE,INTERNAL
210 INPUT #1:A$,I,J,K :: IF
J=K THEN BL=1 :: B$="* BLANK
 *"
220 IF AV=0 AND BL=0 THEN PR
INT :"Press any key to stop
"&M$:L$
230 PRINT #D:"":"  DSK";STR$
(A);"."&A$;TAB(18);B$:"  Use
d=";J-K;" Avail=";K :: IF AV
 OR BL THEN 340
240 PRINT :: PRINT #D:"filen
ame   size   type     P":"--
------   ----   ----     -"
250 FOR N=1 TO 127 :: INPUT
#1:A$,I,J,K :: IF LEN(A$)=0
THEN 340
260 PRINT #D:A$;TAB(12);J;TA
B(17);T$(ABS(I));
270 IF ABS(I)=5 THEN 290 ELS
E B$=" "&STR$(K)
280 PRINT #D:SEG$(B$,LEN(B$)
-2,3);
290 IF I>0 THEN PRINT #D:TAB
(28);" " ELSE PRINT #D:TAB(2
8);"Y"
300 CALL KEY(0,KY,ST):: IF S
T=0 THEN 330
310 PRINT L$:"  [C]ontinue
 [S]top cat."
320 FOR DL=1 TO 30 :: NEXT D
L :: GOSUB 400 :: IF KY=83 T
HEN 340
330 NEXT N
340 CLOSE #1 :: IF D=2 THEN
CLOSE #2
350 PRINT :LL$:"[A]nother  [
D]elete  [S]top"
360 GOSUB 400 :: IF KY=65 TH
EN CALL CLEAR :: AV,BL=0 ::
B$="" :: GOTO 160
370 IF KY=68 THEN INPUT "DEL
 filename: ":A$ :: DELETE "D
SK"&STR$(A)&"."&A$ :: GOTO 3
50
380 IF KY=83 THEN END ELSE 3
60
390 REM ** sub call key **
400 CALL SOUND(100,2000,0)
410 CALL KEY(0,KY,ST):: IF S
T=0 THEN 410 ELSE RETURN
420 REM ** if error; Code 13
0=I/O error
430 CALL ERR(C,T,S,L):: IF C
<>130 THEN 460
440 DISPLAY AT(21,1)BEEP:"I/
O ERROR:":"no disk, or door
open, or   trying to catalog
 a ramdisk."
450 RUN
460 CALL SOUND(400,-3,0):: P
RINT "ERROR";C;"IN LINE";L :
: STOP
```

■

TIPS FROM TIGERCUB #60

1 June 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 400 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 #2 with Supplement #8, listing all titles and authors, is currently available for $1 which is deductible from the first purchase.

Here are a couple of improvements to the CHARFIX subprogram published in Tips #58.

```
29000 SUB CHARFIX(HX$()):: D
ISPLAY AT(12,1)ERASE ALL BEE
P:"Transliterate punctuation
?" :: ACCEPT AT(12,28)SIZE(1
)VALIDATE("YN"):Q$ :: IF Q$=
"N" THEN 29004
29007 CALL CHARVIEW(HX$())
29009 SUB CHARVIEW(HX$())
```

And call the routine by CALL CHARFIX(HX$()). These changes will avoid unwanted transliteration, and will make it possible to use CHARFIX for ASCII 24-31 and 144-159, if BXB has been merged in, as described in Tips #55.

The Spring 1990 issue of the TI*MES newsletter from England contained an interesting challenge - write a program in any language to find the lowest power of 7 which contains six sevens in succession, i.e. "777777".

The computer cannot solve this by any normal means, because it soon goes into scientific notation in which large numbers are rounded off into long strings of zeros. So, I taught it to multiply the old-fashioned way -

```
100 A$=STR$(7):: Y=1
110 Y=Y+1 :: FOR J=LEN(A$)TO
 1 STEP -1 :: E=(VAL(SEG$(A$
,J,1))*7+X)/10
120 X=INT(E):: F=(E-X)*10 ::
 X$=STR$(F)&X$ :: NEXT J
130 IF X>0 THEN X$=STR$(X)&X
$
140 IF POS(X$,"777777",1)<>0
 THEN 160
150 A$=X$ :: X$="" :: X=0 ::
 GOTO 110
160 PRINT "7^";STR$(Y);"=";X
$
170 PRINT #2:"7^";STR$(Y);"=
";X$
```

The answer? 7^175=78011207
91220815810240464127911180777777188182006932636111839698571603885844026671779915606471699893312656644407347632248554716494939953912586437943

My TI-99/4A computed that in 24 minutes. Would someone like to try it on the 9640?

Anyway, I thought I would use the same method to solve precise multiplication of numbers too large to be computed directly. This routine will multiply two numbers of up to 28 digits each, and

will handle decimals and negative numbers. For even larger numbers, change the ACCEPTs to INPUTs and if necessary change the DIM. The only limitation seems to be that the result cannot contain more than 256 digits and even that could be programmed around.

```
100 DIM C$(100)
110 DISPLAY AT(12,1)ERASE AL
L:"FIRST NUMBER?" :: ACCEPT
AT(14,1)VALIDATE(NUMERIC)BEE
P:A$
120 IF SEG$(A$,1,1)="-" THEN
 A$=SEG$(A$,2,255):: M=1
130 A=LEN(A$):: D1=POS(A$,".
",1):: IF D1>0 THEN A$=SEG$(
A$,1,D1-1)&SEG$(A$,D1+1,255)
:: D1=A-D1
140 DISPLAY AT(16,1)ERASE AL
L:"SECOND NUMBER?" :: ACCEPT
 AT(18,1)VALIDATE(NUMERIC)BE
EP:B$
150 IF SEG$(B$,1,1)="-" THEN
 B$=SEG$(B$,2,255):: M=M+1
160 Y=LEN(B$):: D2=POS(B$,".
",1):: IF D2<>0 THEN B$=SEG$
(B$,1,D2-1)&SEG$(B$,D2+1,255
):: D2=Y-D2 :: D1=D1+D2 :: Y
=Y-1
170 FOR J=Y TO 1 STEP -1 ::
W=W+1 :: B=VAL(SEG$(B$,J,1))
:: FOR K=LEN(A$)TO 1 STEP -1
 :: A=VAL(SEG$(A$,K,1))
180 D=(A*B+X)/10
190 E=INT(D):: F=(D-E)*10 ::
 C$(J)=STR$(F)&C$(J):: X=E :
: NEXT K
200 IF X>0 THEN C$(J)=STR$(X
)&C$(J)
210 C$(J)=C$(J)&RPT$("0",W-1
)
220 X=0 :: NEXT J
230 L=LEN(C$(1)):: FOR J=1 T
O Y :: L2=LEN(C$(J)):: IF L2
<L THEN C$(J)=RPT$("0",L-L2)
&C$(J)
240 NEXT J
250 FOR J=LEN(C$(1))TO 1 STE
P -1 :: FOR K=1 TO Y :: G=G+
VAL(SEG$(C$(K),J,1)):: NEXT
K
260 G=(G+H)/10 :: L=INT(G)::
 G=(G-L)*10 :: D$=STR$(G)&D$
 :: H=L :: G=0 :: NEXT J
270 IF H>0 THEN D$=STR$(H)&D
$
280 IF D1>0 THEN D$=SEG$(D$,
1,LEN(D$)-D1)&"."&SEG$(D$,LE
N(D$)-D1+1,255)
290 IF M=1 THEN D$="-"&D$
300 PRINT D$
```

And this one will add up an almost unlimited number of integers of almost any length - I haven't figured out how to get it to line up decimals.

```
100 CALL CLEAR :: DIM C$(100
)
110 DISPLAY AT(12,1):"Input
from D":"  (D)isk or":"  (K)
eyboard?" :: ACCEPT AT(12,12
)VALIDATE("DK")SIZE(-1):Q$ :
: IF Q$="K" THEN 140
120 DISPLAY AT(12,1)ERASE AL
L:"Filename? DSK" :: ACCEPT
AT(12,14):F$ :: OPEN #1:"DSK
"&F$,INPUT
130 X=X+1 :: LINPUT #1:C$(X)
:: M=MAX(M,LEN(C$(X)):: IF E
OF(1)<>1 THEN 130 ELSE CLOSE
 #1 :: GOTO 160
140 DISPLAY AT(12,1):"Press
ENTER when finished":"":""
150 X=X+1 :: INPUT C$(X):: M
=MAX(M,LEN(C$(X))):: IF C$(X
)<>"" THEN 150 ELSE X=X-1
160 FOR J=1 TO X :: IF LEN(C
$(J))<M THEN C$(J)=RPT$("0",
M-LEN(C$(J)))&C$(J)
170 NEXT J :: FOR J=M TO 1 S
TEP -1 :: FOR K=1 TO X :: G=
G+VAL(SEG$(C$(K),J,1)):: NEX
T K
180 G=(G+H)/10 :: L=INT(G)::
 G=(G-L)*10 :: D$=STR$(G)&D$
 :: H=L :: G=0 :: NEXT J
190 IF H>0 THEN D$=STR$(H)&D
$
200 PRINT D$
```

It is easy to invert characters on the screen simply by making the foreground "on" pixels a lighter color than the background "off" pixels - but when you make a screen dump, you will find that the "on" pixels will print and the "off" pixels will not.

Key this in, SAVE it by SAVE DSK1.INVERSE,MERGE and then merge it into any program by MERGE DSK1.INVERSE, call it at any point by CALL INVERSE(A,B), (A and B are

the first and last ASCII to be inverted), and you will have all "on" pixels turned off and vice versa.

```
31111 SUB INVERSE(A,B):: FOR
 CH=A TO B :: CALL CHARPAT(C
H,CH$)
31112 FOR J=1 TO 16 :: CH2$=
CH2$&SEG$("FEDCBA9876543210"
,POS("0123456789ABCDEF",SEG$
(CH$,J,1),1),1):: NEXT J ::
CALL CHAR(CH,CH2$):: CH2$=""
 :: NEXT CH
31113 SUBEND
```

Here is a truly remarkable discovery by Bill Hudson of the Central Ohio Ninety Niners. This 2-line program will allow you to RUN a variable name such as - A$="DSK1.PROGRAM"

You can write lines before these, after these, and even RES the program. You can also use MOVE from GK UTILITY. You can do anything to the program you want as long as you don't change the content of line 1000. The line number does not even have to be 1000 BUT IT MUST BE THE FIRST LINE THAT YOU KEY IN!! You can merge a program into this but can't merge this into a program. Line 900 can also be a different line number but program execution must go to that line first.

```
900 FOR Z=1 TO LEN(A$):: CAL
L LOAD(-41+Z,ASC(SEG$(A$,Z,1
)),0):: NEXT Z :: CALL LOAD(
-41,LEN(A$)):: CALL LOAD(-44
,4+LEN(A$))
1000 RUN "DSKx.1234567890"
```

It's been a long time since we had a screen display to watch just for the fun of it, so here is a tinygram -

```
100 CALL CLEAR :: FOR SET=1
TO 14 :: CALL COLOR(SET,SET+
1,SET+2):: NEXT SET :: CALL
SCREEN(2):: CALL VCHAR(1,1,3
1,768)
110 FOR CH=32 TO 136 STEP 8
:: CALL CHAR(CH,"FF000000000
000FF"):: NEXT CH
120 X=INT(RND*6+1)*2-1 :: Y=
INT(14*RND+1)*8+32 :: FOR R=
12-X TO 12-INT(RND*X):: CALL
 HCHAR(R,5,Y,R)
130 CALL HCHAR(25-R,5,Y,R)
140 CALL HCHAR(R,28-R,Y,R)
150 CALL HCHAR(25-R,28-R,Y,R
)
160 ON INT(2*RND+1)GOTO 170,
190
170 CALL HCHAR(R,4+R,Y+8,25-
R*2)
180 CALL HCHAR(25-R,4+R,Y+8,
25-R*2)
190 NEXT R :: GOTO 120
```

This is a challenging and educational math puzzler which I think is unlike anything you have seen. I had it in my Tigercub catalog for 7 years and sold just 18 copies. If you don't want to key it in, it is now one of the programs on TI-PD disk No. 1300.1.

```
100 GOTO 140
110 J,K,ST,LV,I,R(),T,X,A,A$
,X$,B,B$,C,C$,D,D$,AY,BY,B@$
,BY$,CY,CY$,C@$,Q,Y(),Y@,X@(
),FLAG,R$,RL,Z,YY,D@(),Q$
120 CALL CLEAR :: CALL CHAR
:: CALL COLOR :: CALL VCHAR
:: CALL SCREEN :: CALL KEY :
: CALL SOUND
130 !@P-
140 CALL CLEAR :: FOR J=1 TO
 12 :: CALL COLOR(J,5,16)::
NEXT J
150 CALL VCHAR(1,3,32,672)::
 DISPLAY AT(5,1):" @$%#*#+#
RITHMATIK #+#%$@ "
160 DISPLAY AT(10,1):" Selec
t difficulty level -": :" Ty
pe 1 or 2"
170 CALL KEY(0,K,ST):: IF ST
<1 THEN 170
180 IF (K<49)+(K>50)THEN 170
190 LV=K-48
200 CALL VCHAR(1,3,32,672)::
 FOR I=1 TO 4 :: RANDOMIZE
210 R(I)=INT(RND*10):: IF R(
I)=0 THEN 210
220 FOR T=1 TO I-1 :: IF R(I
)=R(T)THEN 210
230 NEXT T
240 NEXT I :: X=R(1)*1000+R(
2)*100+R(3)*10+R(4)
250 A=INT(4*RND)+1
260 ON A GOSUB 330,340,350,3
60 :: A$=X$
```

```
270 B=INT(4*RND)+1 :: IF B=A
 THEN 270
280 IF (LV=1)*(LEN(STR$(R(B)
/R(A)-INT(R(B)/R(A))))>2)THE
N 250
290 ON B GOSUB 330,340,350,3
60 :: B$=X$
300 C=INT(4*RND)+1 :: IF C=A
 THEN 300
310 IF C=B THEN 300
320 ON C GOSUB 330,340,350,3
60 :: C$=X$ :: D=10-A-B-C ::
 ON D GOSUB 330,340,350,360
:: D$=X$ :: GOTO 370
330 X$=" 1st " :: RETURN
340 X$=" 2nd " :: RETURN
350 X$=" 3rd " :: RETURN
360 X$=" 4th " :: RETURN
370 AY=R(B)/R(A):: BY=ABS(R(
C)-R(B)^2):: IF BY=0 THEN 38
0 ELSE 390
380 B@$="" :: BY$=" equal to
" :: GOTO 400
390 B@$=STR$(BY):: BY$=" mor
e or less than"
400 CY=ABS(R(D)-R(C)-R(B)-R(
A)):: IF CY=0 THEN 410 ELSE
420
410 CY$=" equal to" :: C@$="
" :: GOTO 430
420 CY$=" more or less than"
 :: C@$=STR$(CY)
430 DISPLAY AT(2,1):" I have
 a 4-digit number ":" with n
o two digits the":" same." :
: DISPLAY AT(6,1):" The";B$;
"digit is";AY;" times the";A
$;"digit."
440 DISPLAY AT(9,1):" The";C
$;"digit is ";B@$;BY$;" the
square of the";B$;" digit."
:: DISPLAY AT(14,1):" The";D
$;"digit is ";C@$;" ";CY$;"
the sum of the other digits"
450 DISPLAY AT(18,1):" What
is the number?" :: ACCEPT AT
(20,2)VALIDATE(DIGIT)SIZE(4)
BEEP:Q :: IF Q=X THEN 530
460 Y(1)=INT(Q/1000):: Y(2)=
INT((Q-1000*Y(1))/100):: Y(3
)=INT((Q/100-INT(Q/100))*10)
:: Y(4)=(Q/10-INT(Q/10))*10
:: IF Y(B)<>INT(Y(A)*AY)THEN
 570
470 IF BY<>0 THEN 490
480 IF Y(C)<>Y(B)^2 THEN 570
 ELSE 500
490 IF (Y(C)<>Y(B)^2+BY)*(Y(
C)<>Y(B)^2-BY)THEN 570
500 IF CY<>0 THEN 520
510 IF Y(D)<>Y(A)+Y(B)+Y(C)T
HEN 570 ELSE 530
520 IF (Y(D)<>Y(A)+Y(B)+Y(C)
+CY)*(Y(D)<>Y(A)+Y(B)+Y(C)-C
Y)THEN 570
530 DISPLAY AT(22,1):" Corre
ct!": : :: FOR J=1 TO 2 :: C
ALL SOUND(100,392,5):: CALL
SOUND(100,440,5):: CALL SOUN
D(100,494,5):: CALL SOUND(10
0,523,5)
540 NEXT J :: CALL SOUND(100
0,523,5,392,5,330,5)
550 DISPLAY AT(24,1):" Hit a
ny key"
560 CALL KEY(0,K,ST):: IF ST
<1 THEN 560 ELSE 200
570 DISPLAY AT(22,1):" Wrong
." :: CALL SOUND(900,30000,3
0,30000,30,400,30,-4,0):: DI
SPLAY AT(23,1):" Type A to t
ry again or Z":" to see the
number"
580 CALL KEY(0,K,ST):: IF ST
<1 THEN 580
590 IF K=65 THEN 450
600 IF K=90 THEN 610 ELSE 58
0
610 DISPLAY AT(22,1):" The n
umber was";X:" " :: GOTO 550
 :: END
```

Nearly out of memory and all out of ideas. More next time, maybe.

Jim Peterson

Tigercub ■

TIGERCUB TIPS I PROGRAMBITEN

#16	PB 84-5.08
#40	PB 91-1.25
#41	PB 91-2.21
#42	PB 91-3.25
#43	PB 91-4.15
#44	PB 91-5.24
#45	PB 91-6.25
#46	PB 88-3.23
#47	PB 89-1.20
#48	PB 89-2.20
#49	PB 89-3.20
#50	PB 89-4.15
#51	PB 90-1.17
#52	PB 90-2.20
#53	PB 90-3.25
#54	PB 90-4.20
#58.1	PB 90-5.24
#59	PB 90-6.18
#60	PB 92-1.24 ■

RÄKNA TIDSAVSTÅND I DAGAR

Fungerar ej med skottdag.

```
10 CALL SCREEN(13):: FOR A=0
 TO 14 :: CALL COLOR(A,16,1)
:: NEXT A
15 CALL CLEAR
20 DISPLAY AT(6,2):"snake co
mputer software": :"       d
ay  counter"
30 DISPLAY AT(3,1):" =======
==================" :: DISP
LAY AT(11,1):"  ============
============"
40 DISPLAY AT(14,1):"Enter i
n the form:":"         yyyy.m
m.dd":"             or":"
     yy.mm.dd": : :"Warning!
!   Leap-days not
    included"
110 PRINT "from": :
120 ACCEPT AT(23,1)VALIDATE(
".",DIGIT)SIZE(10):F$
125 PRINT
130 P1=POS(F$,".",1):: IF P1
 THEN 135 ELSE 110
135 FY=VAL(SEG$(F$,1,P1-1))
140 P2=POS(F$,".",P1+1):: IF
 P2 THEN 145 ELSE 110
145 FM=VAL(SEG$(F$,P1+1,P2-P
1-1))
148 FD=VAL(SEG$(F$,P2+1,2))
149 IF FM>12 THEN 110 ELSE I
F FD>31 THEN 110
150 PRINT "to": :
160 ACCEPT AT(23,1)VALIDATE(
".",DIGIT)SIZE(10):F$
165 PRINT
170 P1=POS(F$,".",1):: IF P1
 THEN 175 ELSE 150
175 TY=VAL(SEG$(F$,1,P1-1))
180 P2=POS(F$,".",P1+1):: IF
 P2 THEN 185 ELSE 150
185 TM=VAL(SEG$(F$,P1+1,P2-P
1-1))
188 TD=VAL(SEG$(F$,P2+1,2))
189 IF TM>12 THEN 150 ELSE I
F TD>31 THEN 150
190 IF FY=TY THEN YDAYS=-365
 :: GOTO 230
200 IF FY+1=TY THEN YDAYS=00
00 :: GOTO 230
210 YDAYS=(TY-FY-1)*365
220 TOTY=(YT-YF)*365
230 ON FM GOSUB 300,320,340,
360,380,400,420,440,460,480,
500,520
240 FANTAL=FDAYS-FD
250 ON TM GOSUB 540,560,580,
600,620,640,660,680,700,720,
740,760
260 TANTAL=TDAYS+TD
270 SUM=YDAYS+FANTAL+TANTAL
280 PRINT "days:    ";SUM: :
:
290 GOTO 110
300 FDAYS=365
310 RETURN
320 FDAYS=334
330 RETURN
340 FDAYS=306
350 RETURN
360 FDAYS=275
370 RETURN
380 FDAYS=245
390 RETURN
400 FDAYS=214
410 RETURN
420 FDAYS=184
430 RETURN
440 FDAYS=153
450 RETURN
460 FDAYS=122
470 RETURN
480 FDAYS=92
490 RETURN
500 FDAYS=61
510 RETURN
520 FDAYS=31
530 RETURN
540 TDAYS=0
550 RETURN
560 TDAYS=31
570 RETURN
580 TDAYS=59
590 RETURN
600 TDAYS=90
610 RETURN
620 TDAYS=120
630 RETURN
640 TDAYS=151
650 RETURN
660 TDAYS=181
670 RETURN
680 TDAYS=212
690 RETURN
700 TDAYS=243
710 RETURN
720 TDAYS=273
730 RETURN
740 TDAYS=304
750 RETURN
760 TDAYS=334
770 RETURN
```

■

KALENDER FÖR VALFRITT ÅR

av Adrian Robinson, Bug News, USA

```
100 ! CALENDAR
110 A$="          CALENDAR
            by Adrian Robins
on"
120 DISPLAY AT(12,1)ERASE AL
L:A$
130 DISPLAY AT(18,1):"  PRES
S ANY KEY TO BEGIN"
140 CALL KEY(3,K,S):: IF S<1
 THEN 140
150 ON WARNING NEXT :: CALL
CLEAR :: DIM MO$(12),D$(12)
160 DATA JAN,31,FEB,28,MAR,3
1,APR,30,MAY,31,JUN,30,JUL,3
1,AUG,31,SEP,30,OCT,31,NOV,3
0,DEC,31
170 L$=RPT$("-",20):: W$="SU
 MO TU WE TH FR SA" :: FOR I
=1 TO 31 :: M$=M$&RPT$(" ",2
+(I>9))&STR$(I):: NEXT I
180 INPUT " YEAR >1582: ":Y
:: IF Y>1582 THEN 200 ELSE P
RINT :"YEAR MUST BE LATER TH
AN 1582"
190 END
200 FOR M=1 TO 12 :: READ MO
$(M),L :: CALL MONTH((Y),M,B
)
210 IF M=2 THEN IF (Y/4=INT(
Y/4))-(Y/100=INT(Y/100))+(Y/
400=INT(Y/400))THEN L=L+1
220 D$(M)=RPT$("   ",B)&SEG$
(M$,1,3*L)&RPT$("   ",42-B-L
):: NEXT M
230 OPEN #1:"PIO" :: PRINT #
1: : : : : :CHR$(27)&"E":C
HR$(14);TAB(3);Y;TAB(18);"CA
LENDAR";TAB(35);Y: : :
240 FOR M=1 TO 10 STEP 3 ::
PRINT #1: : :TAB(14);MO$(M);
TAB(40);MO$(M+1);TAB(66);MO$
(M+2): :
250 PRINT #1:TAB(5);W$;TAB(3
1);W$;TAB(57);W$:TAB(5);L$;T
AB(31);L$;TAB(57);L$ :: FOR
I=0 TO 5
260 PRINT #1:TAB(4);SEG$(D$(
M),1+21*I,21);TAB(30);SEG$(D
$(M+1),1+21*I,21);TAB(56);SE
G$(D$(M+2),1+21*I,21)
270 NEXT I :: NEXT M :: PRIN
T #1:CHR$(12);CHR$(27)&"@" :
: CLOSE #1 :: GOTO 180
280 SUB MONTH(Y,M,B)
290 B=365*Y+31*(M-1)
300 IF M>2 THEN B=B-INT(2.3+
.4*M)ELSE Y=Y-1
310 B=B+INT(Y/4)-INT(Y/100)+
INT(Y/400)
320 B=B-7*INT(B/7)
330 SUBEND
```

ÄKTA SMÅ BOKSTÄVER

```
80 REM RIKTIGA SMÅ BOKSTÄVER
90 REM NITTINIAN 83-4.13
100 CALL CHAR(48,"0038444C54
644438")
110 CALL CHAR(64,"0008107C40
78407C")
120 CALL CHAR(79,"0038444444
444438")
130 FOR I=91 TO 126
140 READ A$
150 CALL CHAR(I,A$)
160 NEXT I
170 DATA 00280038447C4444
180 DATA 0028007C4444447C
190 DATA 00382838447C4444
200 DATA 0028004444444438
210 DATA 3C4299A1A199423C
220 DATA 0008103844784038
230 DATA 00000038043C443C
240 DATA 0040407844444478
250 DATA 0000003C4040403C
260 DATA 0004043C4444443C
270 DATA 0000003844784038
280 DATA 000C101010381010
290 DATA 0000003C443C0438
300 DATA 0040405864444444
310 DATA 0010003010101038
320 DATA 0008001808084830
330 DATA 0020202428302824
340 DATA 0030101010101038
350 DATA 0000007854545454
360 DATA 0000007844444444
370 DATA 0000003844444438
380 DATA 0000007844784040
390 DATA 0000003C443C0404
400 DATA 0000005864404040
410 DATA 0000003C40380478
420 DATA 001010381010100C
430 DATA 000000444444443C
440 DATA 0000004444282810
450 DATA 0000004444545428
460 DATA 0000004428102844
470 DATA 0000004428101010
480 DATA 0000007C0810207C
490 DATA 00280038043C443C
500 DATA 0028003844444438
510 DATA 00100038043C443C
520 DATA 0000280044444438
530 DATA @
```

CALENDAR-programmet på föregående sida är avsett för den gregorianska kalendern som infördes i Sverige år 1753. I katolska länder infördes den redan den 15 okt 1582. Övriga icke katolska länder införde den nya gregorianska kalendern vid olika tidpunkter:

Sverige	1 mars 1753
Danmark och Norge	1 mars 1700
Storbritannien	14 sep 1752
Ryssland	14 feb 1918

Tidigare användes den julianska kalendern som infördes av Julius Caesar år 46 f.Kr. Den julianska kalendern hade en skottdag vart fjärde år så att året blev 365,25 dygn. Den visade dock fel med 1 dygn efter varje 128 år.

Påven Gregorius XIII införde den nya gregorianska kalendern år 1582. Den har skottdag vart fjärde år utom jämna hundratal år som endast är skottår om de är delbara med 400. Således är 2000 skottår medan 1800 och 1900 inte är skottår. Den visar fel med 1 dygn efter varje 3300 år (den dagen den sorgen).

För att krångla till det hela ytterligare hade Sverige en helt egen kalender 1700-1712 så att år 1700 saknades skottdag och år 1712 hade två skottdagar.

1992 **CALENDAR** **1992**

JAN

SU	MO	TU	WE	TH	FR	SA
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

FEB

SU	MO	TU	WE	TH	FR	SA
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

MAR

SU	MO	TU	WE	TH	FR	SA
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

APR

SU	MO	TU	WE	TH	FR	SA
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

MAY

SU	MO	TU	WE	TH	FR	SA
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

JUN

SU	MO	TU	WE	TH	FR	SA
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

JUL

SU	MO	TU	WE	TH	FR	SA
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

AUG

SU	MO	TU	WE	TH	FR	SA
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

SEP

SU	MO	TU	WE	TH	FR	SA
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

OCT

SU	MO	TU	WE	TH	FR	SA
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

NOV

SU	MO	TU	WE	TH	FR	SA
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

DEC

SU	MO	TU	WE	TH	FR	SA
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		