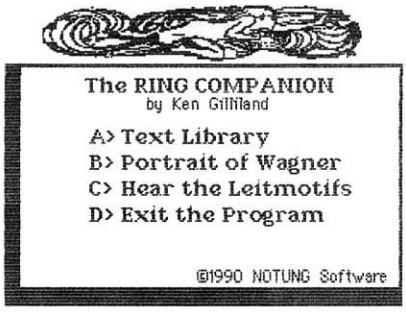
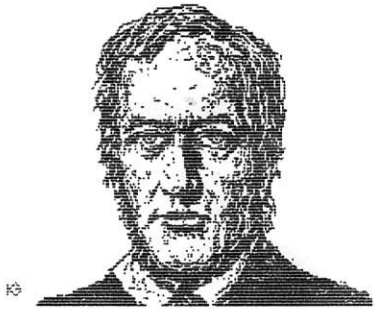
# PROGRAM BITEN 91-5





ISSN 0281-1146

## BUBBLE FÖR MINI MEMORY

```
        AORG >7D00
V1      BSS  >20
V2      BSS  >20
BB      DATA >3C7E,>CFDF,>FFFF,>7E3C
CO      DATA >F333
BL      DATA >A000
SP      DATA >A800
LC      DATA >01DA,>020D,>0271,>02A5
        DATA >02D6,>02E1,>0000
BU      LWPI >70B8
        LI   R0,>394
        LI   R1,CO
        LI   R2,2
        BLWP @>6028
        LI   R0,>D00
        LI   R1,BB
        LI   R2,8
        BLWP @>6028
        CLR  R0
        MOVB @SP,R1
L1      BLWP @>6024
        INC  R0
        CI   R0,>300
        JNE  L1
        MOVB @BL,R1
        LI   R2,LC
L2      MOV  *R2+,R0
        MOV  R0,R0
        JEQ  SC
        BLWP @>6024
        JMP  L2
SC      CLR  R0
        LI   R1,V1
        LI   R2,>20
        BLWP @>6030
        LI   R0,>20
        LI   R1,V2
        LI   R2,>20
L3      BLWP @>6030
        AI   R0,->20
        BLWP @>6028
        AI   R0,>40
        CI   R0,>300
        JL   L3
        LI   R0,>2E0
        LI   R1,V1
        BLWP @>6028
        LIMI 2
        LIMI 0
        JMP  SC
        AORG >7FE8
        TEXT 'BUBBLE'
        DATA BU
        AORG >701E
        DATA >7FE8
        END
```

■

Jan Alexandersson, Springarvägen 5,
3 tr, 142 61 TRÅNGSUND(08-771 05 69)

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 1991 är 120:-

Datainspektionens licensnummer:
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. För lösblad (kopieras av annonsören) som skickas med tidningen gäller 200 kr per blad. Föreningen förbehåller sig rätten att avböja annonser som ej hör ihop med föreningens verksamhet eller ej på ett seriöst sätt gäller försäljning av originalexemplar av program.

För kommersiellt bruk gäller detta: Mångfaldigande av innehållet i denna skrift, helt eller delvis är enligt lag om upphovsrätt av den 30 december 1960 förbjudet utan medgivande av Föreningen Programbiten. Förbudet gäller varje form av mångfaldigande genom tryckning, duplicering, stencilering, bandinspelning, diskettinspelning etc.

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.

# ASSEMBLER REF & ADRESSER

*av Jan Alexandersson*

Det finns flera laddare av assembler DIS/FIX 80 som klarar av att lösa ut REF till BLWP (t.ex. VMBW) och vissa fasta adresser (t.ex. GPLWS). Editor/Assembler, Mini Memory och Funnelweb klarar av dessa enligt följande tabeller. Editor/Assembler kräver för vissa av dessa att den fristående filen BSCSUP laddas in. Extended Basic kan ej lösa dessa REF utan man måste använda EQU till respektive adress.

REF TILL BLWP KAN LÖSAS

| EA LOADER | EA BSCSUP | XB LOADER | MM LOADER | FW LOADER |
|---|---|---|---|---|
| VSBW | - | - | VSBW | VSBW |
| VMBW | - | - | VMBW | VMBW |
| VSBR | - | - | VSBR | VSBR |
| VMBR | - | - | VMBR | VMBR |
| VWTR | - | - | VWTR | VWTR |
| KSCAN | - | - | KSCAN | KSCAN |
| GPLLNK | - | - | GPLLNK | GPLLNK |
| XMLLNK | - | - | XMLLNK | XMLLNK |
| DSRLNK | - | - | DSRLNK | DSRLNK |
| LOADER | - | - | LOADER | LOADER |
| - | NUMASG | - | NUMASG | - |
| - | NUMREF | - | NUMREF | - |
| - | STRASG | - | STRASG | - |
| - | STRREF | - | STRREF | - |
| - | ERR | - | ERR | - |

REF TILL FASTA ADRESSER KAN LÖSAS

| EA LOADER | EA BSCSUP | XB LOADER | MM LOADER | FW LOADER |
|---|---|---|---|---|
| SCAN | - | - | SCAN | SCAN |
| UTLTAB | - | - | UTLTAB | UTLTAB |
| PAD | - | - | PAD | PAD |
| GPLWS | - | - | GPLWS | GPLWS |
| VDPWA | - | - | VDPWA | VDPWA |
| VDPRD | - | - | VDPRD | VDPRD |
| VDPWD | - | - | VDPWD | VDPWD |
| VDPSTA | - | - | VDPSTA | VDPSTA |
| GRMWA | - | - | GRMWA | GRMWA |
| GRMRA | - | - | GRMRA | GRMRA |
| GRMRD | - | - | GRMRD | GRMRD |
| GRMWD | - | - | GRMWD | GRMWD |
| SOUND | - | - | SOUND | SOUND |
| SPCHRD | - | - | SPCHRD | SPCHRD |
| SPCHWT | - | - | SPCHWT | SPCHWT |

BLWP UTILITY EQUATES

| BLWP | EA | XB | MM | FW |
|---|---|---|---|---|
| VSBW | 210C | 2020 | 6024 | 210C |
| VMBW | 2110 | 2024 | 6028 | 2110 |
| VSBR | 2114 | 2028 | 602C | 2114 |
| VMBR | 2118 | 202C | 6030 | 2118 |
| VWTR | 211C | 2030 | 6034 | 211C |
| KSCAN | 2108 | 201C | 6020 | 2108 |
| GPLLNK | 2100 | - | 6018 | 2100 |
| XMLLNK | 2104 | 2018 | 601C | 2104 |
| DSRLNK | 2120 | - | 6038 | 2120 |
| LOADER | 2124 | - | 603C | 2124 |
| NUMASG | - | 2008 | 6040 | - |
| NUMREF | - | 200C | 6044 | - |
| STRASG | - | 2010 | 6048 | - |
| STRREF | - | 2014 | 604C | - |
| ERR | - | 2034 | 6050 | - |

PREDEF SYMBOLS EQUATES

| SYMBOL | EA | XB | MM | FW |
|---|---|---|---|---|
| SCAN | 000E | 000E | 000E | 000E |
| UTLTAB | 2022 | - | 7012 | 2022 |
| PAD | 8300 | 8300 | 8300 | 8300 |
| GPLWS | 83E0 | 83E0 | 83E0 | 83E0 |
| VDPWA | 8C02 | 8C02 | 8C02 | 8C02 |
| VDPRD | 8800 | 8800 | 8800 | 8800 |
| VDPWD | 8C00 | 8C00 | 8C00 | 8C00 |
| VDPSTA | 8802 | 8802 | 8802 | 8802 |
| GRMWA | 9C02 | 9C02 | 9C02 | 9C02 |
| GRMRA | 9802 | 9802 | 9802 | 9802 |
| GRMRD | 9800 | 9800 | 9800 | 9800 |
| GRMWD | 9C00 | 9C00 | 9C00 | 9C00 |
| SOUND | 8400 | 8400 | 8400 | 8400 |
| SPCHRD | 9000 | 9000 | 9000 | 9000 |
| SPCHWT | 9400 | 9400 | 9400 | 9400■ |

## RÄTTELSE: GPLLNK

Av okänd anledning råkade det bli fel i GPLLNK i PB 91-2.15. De två raderna JMP RTNAD och BSS >20 ska strykas. Inledning ska se ut så här:

```
GPLLNK DATA GLNKWS
       DATA GLINK1
RTNAD  DATA XMLRTN
       o.s.v.
```

Även den tidigare versionen fungerar men de extra raderna är meningslösa.

# SWEDLOW TI BITS * 8-10 *

*by Jim Swedlow, USA*

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

FAIRWARE REVIEW: DISK UTILITIES

You may have a favorite disk editor - one that you know and love (?) - one that meets your needs. Mine has been Miller Graphics' Advanced Diagnostics. At least until now. John Birdwell's DISK UTILITIES has jumped to the top of my list. It is easily the best sector editor I have used.

It is what a sector editor should be. You can dump a file to your printer in HEX and ASCII. DISK UTILITIES will follow the file on the disk even if it is fractured. The file dump is like Disk+Aid with HEX on the left and ASCII on the right. The print out can be in condensed print. You can also print a sector or a group of sectors.

You can compare two files or disks. Any sectors that do not match will be dumped to your printer. It can also give you a detailed file report.

DISK UTILITIES supports a string search. You can search a disk, any part of the disk or within a file. The string can be in HEX or ASCII.

The sector editor gives you a full screen editor. The various controls are easy to remember. Pressing CTRL H and CTRL A, for example, switches the screen display between HEX and ASCII. CTRL W will write the sector back to disk. Unlike Advanced Diagnostics, DISK UTILITIES keeps track of the current sector for writing sectors. You can, however, write to any sector on any disk.

Another nice feature is the Disk Report. This prints a disk catalog with two new features. First, the catalog includes each file's sector numbers. Invaluable if have it before you blow a disk directory. Also, DISK UTILITIES hides a short file description in the file header and prints it out as part of the catalog.

This program is a sector editor only. It doesn't have the ability to look into your 4A's memory that Disk+Aid has nor the extensive documentation and diagnostic features of Advanced Diagnostics. But it does have all the features one needs in a disk editor.

There is more, but this should give you an idea of what DISK UTILITIES can do. Without doubt, it warrants your serious consideration.

QUOTES OF THE MONTH

"I cried all the way to the bank."
---Wladziu Valentino (Lee) Liberace
(1919- )

"In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor a yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort."
---John Ronald Reuel (J. R. R.) Tolkien
(1892-1973), from "The Hobbit"

"It is better to know some of the questions than all of the answers."
---James Thurber (1894-1961)

CUSTOMIZING FUNNELWRITER (Funnelweb)

It has been said that FUNNELWRITER may be the most significant program written for the TI. One could argue this point but not easily dismiss it.

I have been working on getting FUNNELWRITER to support the utilities that I normally use.

The first thing I wanted to do was to enable FUNNELWRITER to load FASTTERM. When you press 5 on the main

menu, one of the options that comes up for number 2 is MODEM. I could not find, however, what file name was needed. After a bit of searching (using DISK UTILITIES), I found it: MD.

FAST-TERM comes with two files named UTIL1 and UTIL2. You must rename them (using DM1000) to MD and ME and then copy the files to your FUNNELWRITER disk. Change the names before copying because there already is a UTIL1 on the FUNNELWRITER disk and you do not want to overwrite it.

When you switch item 2 to DISK EDIT, FUNNELWRITER loads Disk Patch, or Disco. This is a bare bones disk sector editor. I wanted to load DISK UTILITIES so I removed Disco from my FUNNELWRITER disk, renamed the two DISK UTILITIES Files (UTIL1 and UTIL2) to DP and DQ and copied them.

I did all of this renaming and copying on back-up copies. My originals are safe and unmodified. Always keep a master copy of important programs.

WORD OF THE MONTH

DIVOT: a piece of turf gouged out with a golf club making a stroke.

"On the one hand, the insecure and anxious youth asks your approval of whatever he says with every y'know. On the other hand, the omniscient stripling's y'knows are simply declaring what he says to be the only possible truth. Indeed, to disagree would be an admission that you had an IQ no higher than an unreplaced divot."
---Edmund Midura, "Um's the Word",
The Inquirer Magazine, 1/30/77

ON GETTING FAIRWARE

On a shelf high above my computer is a large disk box with my masters. One of my prized possesions in that box is an original DISK MANAGER 1000 (V3.3) from the Ottawa TI Users Group. I have a copy from our library but somehow it is not the same. Our library has many fine fairware programs and I (among others) have often urged you to support fairware authors. One way is to request a program directly from the author (and then send some support).

Just a thought.

ALPHABET SOUP

We are constantly bombarded with acronyms. This list is provided as a public service to help you sound like you know what you are talking about!

AI - "Artificial Intelligence" - trying to make computers think like people. A science in its infancy.

ALGOL - "ALGOrithmic Language" - a programming language.

ANSI - "American National Standards Institute"

APL - "A Programming Language" - an interactive programming language that is well suited for handling complex operations on arrays.

ASCII - "American Standard Code for Information Interchange" - and you thought that the II was version two! Pronounced "ask-key".

BASIC - "Beginners All purpose Symbolic Instruction Code" - some suggest that the acronym came after the name!

BBS - "Bulletin Board System"

BIOS - "Basic Input Output System" - the part of CP/M or MS-DOS that allows the CPU to communicate with the keyboard, screen, printer, etc.

C - a programming language developed at Bell Labs. Its predecessors were B (1970) and BCPL (1967).

CMOS - "Complementary Metal Oxide Semiconductor" - a type of IC noted for its low power consumption and resistance to damage. Often used in portable computers. IC's of this type usually have the letter C in their name.

CP/M - "Control Program for Micro-computers" - a family of operating systems that would have been the standard for business had IBM not used PC-DOS (see MS-DOS).

CPU - "Central Processing Unit" - the part of the computer where arithmetic and logical operations are performed and instructions are decoded and executed.

CRT - "Cathode Ray Tube" - the screen on your TV or monitor.

EOF - "End Of File"

IC - "Intergrated Circuit" - a chip with many miniature transistors and other devices.

ISO - "International Standards Organization "

LISP - "LISt Processor" - a programming language often used for AI applications.

MODEM - "MODulator-DEModulator" - a device that encodes and decodes data for transmission over telephone lines, coaxial cable, fiber optics, microwaves, etc.

MS-DOS - "MicroSoft Disk Operating System" - the operating system for computers that use the 8086 or 8088 microprocessor family. MS-DOS is sold by IBM as PC-DOS for the IBM PC.

PROM - "Programable Read Only Memory" - a chip that can be pro-grammed once but not revised. EPROM [Erasable PROM] chips can be erased and reprogrammed.

TTL - "Transistor-Transistor Logic" - a high speed IC that is often used for input-output devices (a TTL monitor, etc).

WYSIWYG - "What You See Is What You Get" - brought to its current potential by the Mac, this means that your item appears on your CRT exactly as it will look when it is printed.

QUOTES OF THE MONTH

"The optimist proclaims that we live in the best of all possible worlds; the pessimist fears that this is true."
---James Branch Cabell

"There are moments when everything goes well; don't be frightened, it won't last."
---Jules Renard

"It's not possible to make things fool proof; fools are too ingenious."
---Edsel Murphy

COMPILED, ASSEMBLED AND INTERPRETED
Or why BASIC is slower than Assembly

A computer language is what you use to tell your computer what to do. It is a common vocabulary. If you have done any programming, you know that your computer believes this language literally.

Your computer don't speak BASIC or Assembly Language. It speaks Machine Language (which is code that the CPU can execute directly).

When a BASIC program is running (also called during 'run time') something called a BASIC interpreter acts as a middle man between the program and the CPU. As each line executes, the interpreter reads the instructions and translates them to Machine Language. This takes time.

In Assembly Language, you write a source program using the Editor and then use the Assembler to assemble it into Machine Language. That's why the module is called Editor/ Assembler. When you run an assembled program, execution is much faster as there is no need for an interpreter.

A compiled program is a hybrid of these two. You write your source program in a higher format. The 'higher' a language is the closer it is to English. The 'lower' it is, the closer it is to Machine Language. BASIC is a high high level language while Assembly is low level.

The language called C looks somewhat

# FAST EXTENDED BASIC!
# 87/11 - HELLO (part 2)



So, did you type and, more important, keep last month's program? It was a simple graphic (and rather superfluous sound) program. Despite its shortness, it included all the subprograms that XB puts at our disposal to use the graphics, less the SPRITES capabilities of course. I save this lesson for a time when I will be more inspired to write a sprity program!

Most CALL statements are shared by TI BASIC: SCREEN, COLOR, CHAR, HCHAR, VCHAR, GCHAR (forgot to put one in my example program...). I will not explain here what they do, I know you know. What I want to stress is rather the new methods that XB gives us so that we spend less time writing the program, and Tex spends less time running it. In addition, XB has two additional and very useful commands: CHARPAT and CHARSET.

CALL COLOR: in BASIC, you write the statement as: CALL COLOR(SET,FGRD, BGRD); if you want to change the colors of more than one set, you have to write a program line for each set, or to use a FOR NEXT loop; both methods are slow. XB uses multiple statements, but why take all that memory when we can simply use

```
290 ... :: CALL COLOR
    (4,1,16,5,1,16,6,1,16,7,1,16)
     S F  B S F  B S F  B S F  B
```

What we do here is change the color of the sets(S) 4 to 7: transparent (1) for foreground(F), white(16) for background(B). This is when we wanted all characters to "disappear" from the screen, which we had made white in line 210. By the way, since there is only one screen, the CALL SCREEN statement cannot get multiple variables... In the remaining of the program, the sets get new colors at a different time, so the statements are simpler (lines 320, 340 and 350).

CALL CHAR: Since the sprites can consist of four characters, XB has a way to define them with a 64-char. long string; if this is too long for your purpose, you can use a shorter one. And, unlike the case of sprites, the ASCII number of the first character to be defined does not have to be a multiple of 4! But unlike BASIC, you must include the trailing "0"s, so that Tex knows when to stop a character and start the next one. Only the last character to be redefined can live without its trailing "0"s, if it has any of course (I included them in the ex. program, lines 250 and 280, I overdid it a bit...). And, to make your programming easier, you can put more than one character definition into your statement:

```
240 call sound(...........) :: CALL
CHAR(56,A$,57,"3F3F3F3F3F3F3F
3F8080808080808080FEFEFEFEFEFFFFFF")
```

is the same as if we had written a statement for each character:

```
240 CALL CHAR(56,A$) :: CALL
CHAR(57,"3F3F3F3F3F3F3F3F") :: CALL
CHAR(58,"8080808080808080") :: CALL
CHAR(59,"FEFEFEFEFEFFFFFF")
```

---

like BASIC but complies into Machine Language.

Now you know.

WORD OF THE MONTH

ALGORITHM: An algorithm is a sequence of instructions that tell how to solve a particular problem. An algorithm must be specified exactly, so there can be no doubt about what to do next, and it must have a finite number of steps. A computer program is an algorithm. Some programs are so complicated that there is no algorithm to solve them.

Enjoy. ■

A little tip here: using variables whenever a string or a number is used more than once saves memory two ways: less characters to write in the program, less running time since Tex does not have to put the string in its memory first. A$ contains the character definition for char. 56, but also for char. 72 and 85, so we have defined it only once, in line 230.

Unfortunately, our other friends of the CHAR family, namely H, V and G, do not allow the use of multiple variables. I don't know why for GCHAR, but I do know that since HCHAR and VCHAR can have the repetition added as an option, poor Tex would get very, very confused.

Now the NEW STUFF: in line 200, we display a sentence on the screen; if you ran the program, you saw "Want to have A surprise?", but soon the character "A" became a shapeless graphic, while "W" and "?" stayed nicely on the screen. The "A" was not an accident, I put it there expressly to show you what I did to the other two. Since we needed the char. "W" and "?" for our graphics, I redefined two other characters, namely "#" and "$", and if you trusted me, you have typed line 200 as "#ant to have A surprise$".

To redefine them as legitimate "W" and "?" that would look exactly like the originals, we could have used a magnifying glass to study those characters on the screen, then used a chardef program to design them, then used a CALL CHAR statement using those definitions. Actually, that is what you have to do in TI BASIC. The wise gurus of XB have designed a much quicker function: CALL CHARPAT, which reads the definition of a character into a string. This does not necessarily read the standard definition, but the definition that the character has when CALL CHARPAT is encountered in the program. In our example, of course, the definition was the standard one, since we had just started our program. So the line

```
170 ... :: CALL CHARPAT(63,A$,87,B$)
:: CALL CHAR(36,A$,35,B$)
```

reads the definition of char. 63, "?", into A$, and that of "W" (ASCII 87) into B$. What to do with A$ and B$ now? Well, we use them right away to redefine our new characters, using the same string variables: "$", ASCII 35, becomes A$, i.e. a "?", while "#", ASCII 35, becomes B$, i.e. a "W".

If you really ran the program through its three screens, you noticed that the last one said "NOW... THAT WAS AN EASY ONE?", and this great phrase was written exactly like that in line 370. How come that we can now use "W" and "?" with confidence? Because of another magic XB statement: CALL CHARSET in line 360. Its purpose is to restore the standard character definitions, but only for characters 32 through 95; that means that the lower case is not restored back (CHARSET will restore the color for char 30-143 but only pattern for char 32-95, ed. note). Note also that it restores the standard character set, not whatever definition was there before the CALL CHAR statement.

I hope this is not too confusing. But, will you say, where are the goodies this month? None in the program itself, but I found a nice postscript on PRE-SCAN, one that is not mentioned in the manual. It comes from Paris, France ("99 Magazine"). If you pre-scan your program, you can do what Tex normally bluntly refuses to do: a FOR NEXT after an IF statement... Try this one:

```
100 PRINT "PRESS <1> or ANY other
key"
110 CALL KEY(0,K,S) :: IF S=0 THEN
110
120 IF K=49 THEN FOR I=1 TO 10 ::
PRINT I :: NEXT I :: GOTO 100 ELSE
FOR I=1 TO 10 :: PRINT "?" :: NEXT I
:: GOTO 100
```

It will not work, Tex will say "* SYNTAX ERROR in 120". Now add this pre-scan line:

```
90 GOTO 100 :: K,S,I :: CALL KEY ::
!@P-
```

and run it. It works! ■

# PROGRAMS THAT WRITE PROGRAMS

*by Jim Peterson, Tigercub, USA*

The first five parts of this series were written long ago, but since then I have found a new method to write programs that really do write programs. I must give Karl Romstedt credit for this idea.

To illustrate this technique, I will use a program which writes an auto-loader to display a diskfull of programs by their complete name rather than the abbreviated filename. This is the LOAD program which I put on all my TI-PD disks.

First, we key in the part which will always be a part of the LOAD program. Do not change the line numbers because there is a reason for them, and leave that REM in line 11 because something else will be plugged in there later.

```
10 CALL CLEAR :: DIM M$(127)
:: CALL SCREEN(5):: FOR S=0
TO 14 :: CALL COLOR(S,2,8)::
 NEXT S :: CALL PEEK(8198,A)
:: IF A<>170 THEN CALL INIT
11 REM
12 ON WARNING NEXT
13 X=X+1 :: READ M$(X):: IF
M$(X)<>"END" THEN 13
14 R=3 :: FOR J=1 TO X-1 ::
READ X$ :: DISPLAY AT(R,1):S
TR$(J);TAB(4);X$ :: R=R+1 ::
 IF R<23 THEN 17
15 DISPLAY AT(24,1):"Choice?
 or 0 to continue 0" :: ACCE
PT AT(24,26)VALIDATE(DIGIT)S
IZE(-3):N :: IF N>X-1 THEN 1
5
16 IF N<>0 THEN 19 :: R=3
17 NEXT J
18 DISPLAY AT(24,1):"Choice?
" :: ACCEPT AT(24,9)VALIDATE
(DIGIT):N :: IF N=0 OR N>X-1
 THEN 18
19 CALL CHARSET :: CALL CLEA
R :: CALL SCREEN(8):: CALL P
EEK(-31952,A,B):: CALL PEEK(
A*256+B-65534,A,B):: C=A*256
+B-65534 :: A$="DSK1."&M$(N)
:: CALL LOAD(C,LEN(A$))
20 FOR J=1 TO LEN(A$):: CALL
 LOAD(C+J,ASC(SEG$(A$,J,1)))
:: NEXT J :: CALL LOAD(C+J,0
):: GOTO 10000
10000 RUN "DSK1.1234567890"
```

Now, save that "source code" by SAVE DSK1.CAT/S,MERGE . Then key in this "assembler" which will convert the "source code" into an "object code."

```
100 OPEN #1:"DSK1.CAT/S",VAR
IABLE 163,INPUT
110 OPEN #2:"DSK1.CAT/O",VAR
IABLE 163,OUTPUT
120 FOR J=10 TO 21 :: LINPUT
 #1:M$ :: PRINT #2:CHR$(0)&C
HR$(J)&CHR$(156)&CHR$(253)&C
HR$(200)&CHR$(1)&"2"&CHR$(18
1)&CHR$(199)&CHR$(LEN(M$))&M
$&CHR$(0):: NEXT J
130 PRINT #2:CHR$(255)&CHR$(
255):: CLOSE #1 :: CLOSE #2
```

Note what this routine does. It reads in each line of the tokenized CAT/S and prints it back out to CAT/O preceded by line numbers 10 to 21 in tokenized two-byte format followed by the tokens for PRINT #2, the tokens for a quoted string followed by the CAT/S record and the CHR$(0) end-of-line indicator. Then it prints the double-255 end-of-file indicator and closes the files.

Now key in the CATWRITER program.

```
1 CALL CLEAR :: CALL TITLE(1
6,"CATWRITER"):: CALL CHAR(1
27,"3C4299A1A199423C"):: DIS
PLAY AT(2,10):"Version 1.4":
;:TAB(8); Tigercub Softwar
e"
2 DISPLAY AT(15,1):"For free
":"distribution":"but no pri
ce or":"copying fee":"to be
charged." :: FOR D=1 TO 500
:: NEXT D :: CALL DELSPRITE(
ALL)
3 DISPLAY AT(2,3)ERASE ALL:"
TIGERCUB CATWRITER V.1.4":;:
```

```
" Will read a disk directory
,":"request an actual progra
m":"name for each program-ty
pe"
4 DISPLAY AT(7,1):"filename,
 and create a merg-":"able Q
uickloader which dis-":"play
s full program names and":"r
uns a selected program."
5 DISPLAY AT(12,1):" Place d
isk to be cataloged":"in dri
ve 1 and press any key" :: C
ALL KEY(0,K,S):: IF S=0 THEN
 5
9 OPEN #2:"DSK1.CAT",VARIABL
E 163,OUTPUT
100 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:N$,A,J,K :: LN=1000 :: FN=1
100
110 DISPLAY AT(12,1):"Disk n
ame?":;:N$ :: ACCEPT AT(14,1
)SIZE(-28):N$ :: LX$=STR$(14
-LEN(N$)/2):: LXLEN=LEN(LX$)
120 PR$=CHR$(0)&CHR$(11)&CHR
$(162)&CHR$(240)&CHR$(183)&C
HR$(200)&CHR$(1)&"1"&CHR$(17
9)&CHR$(200)&CHR$(LXLEN)&LX$
130 PR$=PR$&CHR$(182)&CHR$(1
81)&CHR$(199)&CHR$(LEN(N$))&
N$&CHR$(0)
140 PRINT #2:PR$
145 DISPLAY AT(23,1):"To omi
t a file, press Enter"
150 X=X+1 :: INPUT #1:P$,A,J
,B :: IF LEN(P$)=0 THEN 190
:: IF ABS(A)=5 OR ABS(A)=4 A
ND B=254 THEN 160 ELSE X=X-1
 :: GOTO 150
160 DISPLAY AT(12,1):P$;"
 PROGRAM NAME?" :: ACCEPT AT
(14,1)SIZE(25):F$ :: IF F$="
" THEN X=X-1 :: GOTO 150
170 PRINT #2:CHR$(INT(FN/256
))&CHR$(FN-256*INT(FN/256))&
CHR$(147)&CHR$(200)&CHR$(LEN
(F$))&F$&CHR$(0):: FN=FN+1
180 M$=M$&CHR$(200)&CHR$(LEN
(P$))&P$&CHR$(179):: IF X<11
 THEN 150
190 IF M$="" THEN 210
200 PRINT #2:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&SEG$(M$,1,LEN(M$)-
1)&CHR$(0):: LN=LN+1 :: M$="
" :: X=0 :: IF LEN(P$)<>0 TH
EN 150
210 PRINT #2:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&CHR$(200)&CHR$(3)&
"END"&CHR$(0)
220 PRINT #2:CHR$(255)&CHR$(
255):: CLOSE #1 :: CLOSE #2
230 DISPLAY AT(8,1)ERASE ALL
:"Enter -":;:"  NEW":;:"  ME
RGE DSK1.CAT":;:"  DELETE ""
DSK1.CAT""":;:"  SAVE DSK1.L
OAD"
240 SUB TITLE(S,T$)
250 CALL SCREEN(S):: L=LEN(T
$):: CALL MAGNIFY(2)
260 FOR J=1 TO L :: CALL SPR
ITE(#J,ASC(SEG$(T$,J,1)),J+1
-(J+1=S)+(J+1=S+13)+(J>14)*1
3,J*(170/L),10+J*(200/L))::
NEXT J
270 SUBEND
```

Next, enter MERGE DSK1.CAT/O and that "object code" will pop into place right after line 9. If you list it, it will look like a blown file, because most of the token codes are unprintable, but don't worry. Save the program as CAT-WRITER.

When you run the program, it will open an output MERGE format file called CAT and write those merged lines from CAT/O in MERGE format. Then it will open the disk you are cataloging, read the directory sector, and ask you for a disk name with the existing diskname as default. You can select any disk name you want to title the menu screen, up to 28 characters long. Line 110 computes the position to center the title, and lines 120-140 write to the CAT file a tokenized line 11 (overwriting that REM line) to display your title at the top of the screen.

Line 150 reads each filename from the disk directory, skipping over anything that is not a program (no one yet has been able to tell me how to distinguish an assembly image "program"!). For each filename, it will ask you for a complete program name. If you don't want a program on the menu (such as an XB program that is run from another program, or an image file), just press Enter. Otherwise the program name you select will be printed as DATA by line 170, in tokenized format in lines starting with 1100 (note the FN=1100 in line 100) and incremented by 1. Lines 180-200 assemble the filenames into DATA lines of up to

ten names, and tokenize them in lines beginning with 1000.

When the last filename has been read, line 210 prints one last DATA item "END" to signal line 13 to stop reading, and then prints the double-255 end-of-file. Then you are given instructions to clear memory with NEW, merge in the CAT file, delete it because you don't need it any more, and save it back as LOAD.

When you list the LOAD program, you will find the original CAT/S restored in lines 10-19 and 1000, the line to display the title in line 11, the filenames in DATA lines starting with 1000 and the program names in DATA lines starting at 1100.

When you run the program, it will display the disk name, and read the filenames into an array. Then it will display the program names, numbered, on as many screens as necessary, and ask you to select a program by number. The corresponding filename by number is selected from the array, and lines 19-20 rewrite line 10000 to RUN that filename. List the LOAD program after you have used it to load something, and you will see that it has changed.

That algorithm in lines 19-20 was published in one of the earliest 99'ER magazines, in a letter by A. Kludge. It has been the basis for every XBasic menu loader, and has saved us uncounted thousands of hours. The author had asked me not to reveal his identity, but I think I can now tell you that "A. Kludge" was really the late Dr. Stefan-Romano, who passed away recently at the age of 57. He was a brilliant man who did much for the TI world, at first as editor of the IUG library, and then through the Amnion library and Amnion Helpline. He was of great help to me on several occasions.

Some of you may have obtained from me a copy of CATWRITER which wrote GOSUB 21 in line 12, and CALL LOADs in lines 21-25 to change the cursor to my Tigercub emblem. If you have begun to have problems with the resulting LOAD program or with my previous Tigercub Menuloader which used the same CALL LOADs, I have finally found out the cause. When my Horizon RamDisk is on, any program containing those CALL LOADs will lock up the second time it is run!

■

---

# NIM – SPEL FÖR XB

*by Earl Raguse, Bug News, USA*

```
110 !************************
120 !       NIM(XBASIC)
130 !   A JOLLY? GOOD GAME
140 !     BY EARL RAGUSE
150 !      REV 3/25/86
160 !    BASIC IDEA FROM
170 !    "LEARNING BASIC"
180 !************************
190 CALL CLEAR
200 CALL CHARPAT(42,Z$):: CA
LL CHAR(96,Z$)
210 CALL COLOR(9,16,1):: CAL
L SCREEN(5)
220 A$=RPT$(" ",28)
230 C$(1)=A$&"`     ` ` `
  `"
240 C$(2)=A$&"``    ` ` ``
 ``"
250 C$(3)=A$&"` `   ` ` ` `
 ` `"
260 C$(4)=A$&"`  ` ` ` ` ``
  `"
270 C$(5)=A$&"`    `` ` `
  `"
280 C$(6)=A$&"`     ` ` `
  `"
290 FOR V=3 TO 23
300 FOR U=1 TO 6
310 DISPLAY AT(9+U,1):SEG$(C
$(U),V,28)
320 CALL SOUND(100,330,10,66
0,15,110,6,-4,1)
330 NEXT U
340 FOR I=1 TO 10 :: NEXT I
350 NEXT V
360 FOR I=1 TO 500 :: NEXT I
370 FOR V=23 TO 50 STEP 2
380 FOR U=1 TO 6
```

```
390 DISPLAY AT(9+U,1):SEG$(C
$(U),V,28)
400 NEXT U
410 CALL SOUND(100,400,9)
420 NEXT V
430 CALL SCREEN(6)
440 R=21
450 GOSUB 950
460 DISPLAY AT(3,6)BEEP SIZE
(28):"LETS PLAY NIM !!"
470 DISPLAY AT(10,1)BEEP SIZ
E(28):"DO YOU WANT INSTRUCTI
ON Y:N"
480 DISPLAY AT(12,1)SIZE(28)
:"PRESS <Y> FOR YES"
490 DISPLAY AT(13,1)SIZE(28)
:"PRESS <N> FOR NO "
500 CALL KEY(0,K,S):: IF S=0
 THEN 500 :: IF K=89 THEN 51
0 ELSE 590
510 CALL CLEAR :: GOSUB 950
520 PRINT "THIS IS AN ANCIEN
T CHINESE":"GAME. THERE ARE
21 STONES":"<ASTERISKS> IN A
 ROW; YOU":"CAN TAKE 1,2,OR
3 STONES":"IN YOUR MOVE. ";
530 PRINT "THEN I CAN":"TAKE
 1,2,OR 3 STONES IN MY":"MOV
E. WE ALTERNATE MOVES.": :
540 PRINT "THE PLAYER WHO TA
KES THE":"LAST STONE LOSES!"
: :
550 PRINT "I WILL BROOK NO I
NFRACTION":"OF THE RULES AND
 I AM VERY":"ALERT! DON'T TR
Y TO FOOL ME.": :
560 PRINT " READY; PRESS ANY
 KEY "
570 CALL KEY(0,K,S):: IF S=0
 THEN 570
580 DISPLAY AT(12,4)BEEP SIZ
E(24):"OK! YOU GO FIRST.";
590 R=21
600 CALL COLOR(2,16,1):: GOS
UB 950
610 DISPLAY AT(12,4)BEEP SIZ
E(28):"HOW MANY DO YOU TAKE?
"
620 CALL KEY(0,K,S):: IF S=0
 THEN 620
630 IF (K>48 AND K<52)THEN 6
80
640 X=X+1 :: IF X>6 THEN X=6
650 ON X GOSUB 1240,1260,128
0,1300,1320,1340
660 FOR I=1 TO 1500 :: NEXT
I :: GOSUB 950
670 GOTO 610
680 N=K-48
690 R=R-N
700 DISPLAY AT(14,4)BEEP SIZ
E(28):"YOU TAKE";N;"LEAVING"
;R
710 GOSUB 990
720 FOR I=1 TO 500 :: NEXT I
730 IF R=1 THEN 1660
740 IF FLAG=1 THEN GOSUB 113
0
750 N=4-N
760 R=R-N
770 DISPLAY AT(12,4)BEEP SIZ
E(28):"I TAKE";N;"LEAVING ";
R
780 DISPLAY AT(14,1)SIZE(28)
:" "
790 FOR I=1 TO 200 :: NEXT I
800 GOSUB 990
810 FOR I=0 TO 500 :: NEXT I
820 IF R>1 THEN 610
830 PRINT "  SINCE THERE IS
ONLY ONE":"  STONE LEFT I WI
N!  ";
840 PRINT
850 IF (FLAG=1 AND R=1)THEN
1480
860 CALL SOUND(1000,131,0,16
5,5,196,10,-3,28)
870 CALL SOUND(1000,262,0,33
0,0,392,0)
880 CALL SOUND(1000,524,0,66
0,O,784,0)
890 Z=Z+1 :: IF Z>6 THEN Z=6
900 PRINT
910 ON Z GOSUB 1520,1530,155
0,1570,1590,1610
920 PRINT
930 PRINT "WANT TO PLAY AGAI
N? Y:N >  "
940 CALL KEY(0,K,S):: IF K=8
9 THEN 590 ELSE IF K=78 THEN
 1010 ELSE 940
950 CALL CLEAR :: CALL SCREE
N(2*Z+3)
960 DISPLAY AT(20,4):"221111
111111"
970 DISPLAY AT(21,4):"109876
543210987654321"
980 CALL HCHAR(18,27-R,96,R)
:: GOTO 1000
990 CALL HCHAR(18,6,32,21-R)
1000 RETURN
1010 CALL CLEAR :: CALL SCRE
EN(INT(RND*12)+2)
1020 Y=Y+1 :: IF Y>6 THEN Y=
6
1030 ON Y GOSUB 1360,1380,14
00,1420,1440,1480
1040 PRINT "Y:N ? > "
1050 CALL KEY(0,K,S):: IF S=
0 THEN 1050
1060 IF K=89 THEN 590 ELSE I
F K=78 THEN 1010 ELSE 1050
1070 DISPLAY AT(12,1)BEEP ER
ASE ALL:"OH WELL! I DON'T WA
```

```
NT TO    PLAY WITH SUCH A SO
RE HEAD  ANYWAY!!"
1080 FOR I=1 TO 1500 :: NEXT
 I
1090 GOTO 1690
1100 N=1 :: FLAG=1
1110 R=21 :: CALL COLOR(2,16
,1):: GOSUB 950
1120 R=R-N :: GOTO 770
1130 IF R=13 THEN 1210
1140 IF (R=17 OR R=13 OR R=9
 OR R=5)THEN N=1 :: GOTO 760
1150 IF R>17 THEN N=R-17 ::
GOTO 760
1160 IF R>13 THEN N=R-13 ::
GOTO 760
1170 IF R>9 THEN N=R-9 :: GO
TO 760
1180 IF R>5 THEN N=R-5 :: GO
TO 760
1190 N=R-1 :: GOTO 760
1200 GOSUB 950
1210 DISPLAY AT(9,2)BEEP SIZ
E(28):"ARE YOU JUST LUCKY OR
 HAVE"
1220 DISPLAY AT(10,4)BEEP SI
ZE(28):"YOU FIGURED THIS OUT
?"
1230 GOTO 1140
1240 PRINT "NO!":"ENTER ONLY
 1:2 OR 3":"I'M NOT ASLEEP Y
OU KNOW"
1250 RETURN
1260 PRINT "A JOKER HUH?! EN
TER 1;2 OR 3":"     ***OR EL
SE***"
1270 RETURN
1280 PRINT "WHAT ARE YOU? A
CLOWN PRINCE":" JUST 1;2 OR
3"
1290 RETURN
1300 PRINT "I'M BEGINNING TO
 THINK":"YOU'RE JUST PLAIN S
TUPID"
1310 RETURN
1320 PRINT "YOU'RE NOT ONLY
STUPID":"BUT YOU'RE UGLY TOO
; PLAY":"NICE HUH? 1;2 OR 3!
!!"
1330 RETURN
1340 PRINT "NOT ONLY ARE YOU
 STUPID AND":"UGLY BUT I BET
 YOU HAVE A":"FAT HEAD TOO"
1350 RETURN
1360 PRINT "AW COME ON ! PLA
Y":"JUST ONE MORE!"
1370 RETURN
1380 PRINT "PLEASE PLAY WITH
 ME":"I'M LONELY"
1390 RETURN
1400 PRINT "DONT BE A QUITTE
R":"I'LL GO EASY ON YOU"
1410 RETURN
1420 PRINT "JUST ONE MORE TI
ME":"YOU WILL GET THE HANG O
F IT;":"YOU'LL SEE"
1430 RETURN
1440 PRINT "WILL YOU PLAY IF
 I GO FIRST?":"Y:N  "
1450 CALL KEY(0,K,S):: IF S=
0 THEN 1450
1460 IF K=89 THEN 1100 ELSE
IF K=78 THEN 1470 ELSE 1440
1470 GOTO 1070
1480 PRINT :: PRINT "HA! I W
ON ANYWAY!!":"THAT PROVES I'
M SMARTER THAN":"YOU  PRESS
<E> IF YOU WANT":"AN EXPLANA
TION OF HOW TO DO IT ": :
1490 PRINT "PRESS <Q> IF YOU
 ACTUALLY":"WANT TO QUIT "
1500 CALL KEY(0,K,S):: IF S=
0 THEN 1500
1510 IF K=69 THEN 1630 ELSE
IF K=81 THEN 1710
1520 RETURN
1530 PRINT TAB(10);"AGAIN!!"
1540 RETURN
1550 PRINT TAB(7);"DARN I AM
 GOOD"
1560 RETURN
1570 PRINT "I'M TOO SMART FO
R YOU"
1580 RETURN
1590 PRINT "I AM REALLY GETT
ING BETTER "
1600 RETURN
1610 PRINT "  I AM TRULY INV
INCIBLE!"
1620 RETURN
1630 CALL CLEAR :: PRINT "TH
E FIRST PLAYER CAN'T WIN;":"
IF THE SECOND PLAYER ALWAYS"
:"TAKES 4 MINUS THE FIRST":"
PLAYER'S TAKE; THINK ABOUT":
1640 PRINT "THIS A BIT; I'M
SURE YOU":"WILL FIGURE OUT W
HY!": :
1650 PRINT "DO YOU WANT TO T
RY IT?" :: GOTO 1440
1660 PRINT "ALL RIGHT; YOU W
IN!"
1670 FOR S=-1 TO -8 STEP -1
:: CALL SOUND(300,S,1):: NEX
T S
1680 CALL SOUND(2000,110,0,3
30,1,550,3,-6,0)
1690 PRINT "THIS IS NO FUN;
I'M GOING TO":" TAKE MY STON
ES AND GO HOME "
1700 FOR I=1 TO 1500 :: NEXT
 I
1710 CALL CLEAR
1720 END                    ■
```

# FROM BASIC TO ASSEMBLY
# GRAPHIC 1 MODE

*by Bob August, Bug News, USA*

This month we are going to put a window on the screen with a message in the center of the window. We stayed with ASCII 49 through ASCII 56 to make our window. You may change this to any ASCII number you like. Just use the formula:

( (49*8) + 2048 = 2440 = >0988 )

We used a new command in the program to drop to the next line. This is AI or add immediate. This adds 32 to Register zero to drop down one line. You will also note that we only loaded register two with an eight for the top line of our window. As we never change register two we don't need to load it again. Also we could have added eight to register zero each time instead of using the hex number for the pattern table. In our first loop (aloop) we used CI R0,227+26 to show you that assembly will accept the plus or minus and do the calculation. We could have used CI R0,253 and obtained the same result.

The Extended Basic version of our window program is listed below:

```
100 ! Lesson Number 6
110 CALL CHAR(49,"FFFFC0C0CF
CFCCCC")! LEFT TOP CORNER
120 CALL CHAR(50,"FFFF0000FF
FF0000")! TOP LINE
130 CALL CHAR(51,"FFFF0303F3
F33333")! RIGHT TOP CORNER
140 CALL CHAR(52,"CCCCCCCCCC
CCCCCC")! LEFT SIDE
150 CALL CHAR(53,"3333333333
333333")! RIGHT SIDE
160 CALL CHAR(54,"CCCCCFCFC0
C0FFFF")! LEFT BOTTOM CORNER
170 CALL CHAR(55,"0000FFFF00
00FFFF")! BOTTOM LINE
180 CALL CHAR(56,"3333F3F303
03FFFF")! RIGHT BOTTOM CORNE
R
190 GOSUB 320
200 CALL HCHAR(8,3,49)
210 CALL HCHAR(8,4,50,26)
220 CALL HCHAR(8,30,51)
230 CALL VCHAR(9,3,52,3)
240 CALL VCHAR(9,30,53,3)
250 CALL HCHAR(12,3,54)
260 CALL HCHAR(12,4,55,26)
270 CALL HCHAR(12,30,56)
280 GOSUB 340
290 CALL KEY(0,K,S):: IF S=0
 THEN 290
300 IF K<>13 THEN 290
310 STOP
320 CALL CLEAR
330 RETURN
340 DISPLAY AT(10,3)SIZE(23)
:"PRESS ENTER KEY TO QUIT"
350 RETURN
360 END
```

Both the above program and the assembly program will display a window in the middle of the screen with the message in the center of the window.

HAPPY ASSEMBLING!

```
************************************
* BASIC TO ASSEMBLY Lesson Number 6 *
************************************
*
       DEF  START            Entry point of program
       REF  VSBW,VMBW,KSCAN  Utilities used in program
*
WRKSP  BSS  32               Workspace buffer
SAV11  BSS  2                Save return address buffer
*
TOP    DATA >FFFF,>0000,>FFFF,>0000    Top line of window
BOTTOM DATA >0000,>FFFF,>0000,>FFFF    Bottom line of window
LTCORN DATA >FFFF,>C0C0,>CFCF,>CCCC    Left top corner
RTCORN DATA >FFFF,>0303,>F3F3,>3333    Right top corner
```

```
LBCORN DATA >CCCC,>CFCF,>C0C0,>FFFF    Left bottom corner
RBCORN DATA >3333,>F3F3,>0303,>FFFF    Right bottom corner
LSIDE  DATA >CCCC,>CCCC,>CCCC,>CCCC    Left side of window
RSIDE  DATA >3333,>3333,>3333,>3333    right side of window
*
MSG1   TEXT 'PRESS ENTER KEY TO QUIT'  Prompt to quit
*
       EVEN                Make sure we start on even byte
*
* Start of program
*
START  MOV  R11,@SAV11     Save return address
       LWPI WRKSP          Load the workspace
       BL   @CLEAR         GOSUB CLEAR to Clear the screen
*
* Put window on screen
*
       LI   R0,>0988       Load pattern table address for a one
       LI   R1,TOP         Load top line data
       LI   R2,8           load pattern descripter table
       BLWP @VMBW          Write it to VDP
       LI   R0,227         Load screen location (Row 8, Col. 4)
       LI   R1,>3100       Load ASCII 49 or a one
ALOOP  BLWP @VSBW          Write it to the screen
       INC  R0             Add one to Register zero
       CI   R0,227+26      Compare for end of line or 253
       JNE  ALOOP          Jump if not equal to ALOOP or go on.
*
       LI   R0,>0990       Load pattern table address for a two
       LI   R1,BOTTOM      Load bottom line data
       BLWP @VMBW          Write it to VDP
       LI   R0,355         Load screen location (Row 12, Col. 4)
       LI   R1,>3200       Load ASCII 50 or a two
BLOOP  BLWP @VSBW          Write it to the screen
       INC  R0             Add one to Register zero
       CI   R0,355+26      Compare for end of line or 381
       JNE  BLOOP          Jump if not equal to BLOOP else go on.
*
       LI   R0,>0998       Load pattern table address for a three
       LI   R1,LTCORN      Load left top corner data
       BLWP @VMBW          write it to VDP
       LI   R0,226         Load screen location (Row 8, Col. 3)
       LI   R1,>3300       Load ASCII 51 or a three
       BLWP @VSBW          Write it to the screen
*
       LI   R0,>09A0       Load pattern table address for a four
       LI   R1,RTCORN      Load right top corner data
       BLWP @VMBW          Write it to VDP
       LI   R0,253         Load screen location (Row 8, Col. 30)
       LI   R1,>3400       Load ASCII 52 or a four
       BLWP @VSBW          Write it to the screen
*
       LI   R0,>09A8       Load pattern table address for a five
       LI   R1,LBCORN      Load left bottom corner data
       BLWP @VMBW          Write it to VDP
       LI   R0,354         Load screen location (Row 12, Col. 3)
       LI   R1,>3500       Load ASCII 53 or a five
       BLWP @VSBW          Write it to the screen
*
       LI   R0,>09B0       Load pattern table address for a six
       LI   R1,RBCORN      Load right bottom corner data
       BLWP @VMBW          Write it to VDP
```

```
       LI   R0,381        Load screen location (Row 12, Col. 30)
       LI   R1,>3600      Load ASCII 54 or a six
       BLWP @VSBW         Write it to the screen
*
       LI   R0,>09B8      Load pattern table address for a seven
       LI   R1,LSIDE      Load left side data
       BLWP @VMBW         Write it to VDP
       LI   R0,258        Load screen location (Row 9, Col. 3)
       LI   R1,>3700      Load ASCII 55 or a seven
CLOOP  BLWP @VSBW         Write it to the screen
       AI   R0,32         Add 32 to Register zero for next line
       CI   R0,258+96     Compare for three lines or 254
       JNE  CLOOP         Jump if not equal to CLOOP else go on.
*
       LI   R0,>09C0      Load pattern table address for an eight
       LI   R1,RSIDE      Load right side data
       BLWP @VMBW         Write it to VDP
       LI   R0,285        Load screen location (Row 9, Col. 30)
       LI   R1,>3800      Load ASCII 56 or an eight
DLOOP  BLWP @VSBW         Write it to the screen
       AI   R0,32         Add 32 to Register zero for next line
       CI   R0,285+96     Compare for three lines or 381
       JNE  DLOOP         Jump if not equal to DLOOP else go on.
*
* Put message on screen
*
       BL   @DISPLY       Gosub to display message at Row 10, Col. 5
       DATA 292,MSG1,23   Screen location, Message, Length of message
*
* Call key routine
*
       CLR  @>8374        Clear to zero for CALL KEY(0,K,S)
       CLR  @>837C        Clear status to zero
ELOOP  BLWP @KSCAN        CALL KEY(0,K,S)
       CB   @>837C,@>20   Check for key press  (may not work, ed. note)
       JEQ  ELOOP         IF S=0 THEN ELOOP    (addr.>0020 is >0460, ed. note)
       MOV  @>8375,R0     Move Key press to register zero
       CI   R0,>0D        Compare to 13 or enter key
       JNE  ELOOP         If not enter key goto ELOOP
       CLR  @>837C        Clear status to zero
       MOV  @SAV11,R11    Put return address in register 11
       BLWP @0            Quit (FCTN =)
*
* Clear screen routine
*
CLEAR  LI   R1,>2000      Load Register one with space
       CLR  R0            Clear Register zero to zero
FLOOP  BLWP @VSBW         Write blank space to screen
       INC  R0            Add one to register zero
       CI   R0,767        Compare contents to 767
       JLE  FLOOP         If less then 767 goto FLOOP
       RT                 Return to next line of calling area
*
* Display at routine
*
DISPLY MOV  *R11+,R0      Put screen location into Register zero
       MOV  *R11+,R1      Put message into Register one
       MOV  *R11+,R2      Put length into Register two
       BLWP @VMBW         Write it to the screen
       RT                 Retrun to next line of calling area
*
       END  START         End program with auto start                    ■
```

# BEGINNER ASSEMBLER - 5
# #SPIRITED SPRITES

*by Mack McCormick, USA*

Definition: Any shape or color. Can occupy screen positions independent of any character already present. Once set into motion, can move independently of direct program control. You can magnify or make double size.

How they can be used: Up to 32 sprites on the screen at any one time. Can be used in GRAPHICS and MULTICOLOR modes. Also can be used in BIT MAP mode but not the automatic motion feature. Sprites cannot be used in the TEXT mode.

---

## COLOR DRAW

(från PB 84-5.18)

```
90 !COLOR-DRAW. Ett program
foer lek med faerger
95 !Ur The Smart Programmer
 februari 1984
100 CALL CLEAR :: CALL SCREE
N(2):: K=2 :: W=32 :: FOR S=
88 TO 136 STEP 8 :: CALL CHA
R(S,"FFFFFFFFFFFFFF00"):: 
NEXT S
110 CALL CHAR(42,"00003C3C3C
3C"):: CALL COLOR(1,14,2,8,1
3,3,9,4,5,10,6,8,11,7,9,12,1
0,11,13,12,14,14,15,16)
120 FOR S=88 TO 136 STEP 8 :
: DISPLAY AT(1,K):CHR$(S)&"
"&CHR$(S+1):: K=K+4 :: NEXT
S
130 CALL HCHAR(2,1,32,736):: 
 CALL SPRITE(#1,42,16,17,121
):: Y=3 :: X=16 :: CALL SOUN
D(-100,660,9)
140 CALL JOYST(1,K,S):: X=X+
SGN(K):: Y=Y-SGN(S):: IF Y>2
4 THEN Y=1 ELSE IF Y<1 THEN
Y=24
150 IF X>32 THEN X=1 ELSE IF
 X<1 THEN X=32
160 CALL LOCATE(#1,Y*8-7,X*8
-7):: CALL KEY(1,K,S):: IF S
=0 THEN 140 ELSE IF Y>1 THEN
 CALL SOUND(-90,-2,15):: CAL
L HCHAR(Y,X,W):: GOTO 140
170 IF K=19 THEN 130 ELSE CA
LL GCHAR(Y,X,W):: CALL SOUND
(-90,880,7):: CALL HCHAR(1,2
,W):: GOTO 140
```

■

There are three tables which contain all the information needed to use sprites:

1. SPRITE ATTRIBUTE TABLE

   a. Sprite Position
   b. Sprite Color

2. SPRITE DESCRIPTOR TABLE

   a. Sprite Pattern Identifier
   b. Specify magnified or double sized sprites.

3. SPRITE MOTION TABLE

   a. Define X and Y velocities of Sprites.

DEFAULT LOCATIONS OF SPRITE TABLES

| Table | VDP address |
|---|---|
| SPRITE ATTRIBUTE TABLE | >0300 |
| SPRITE DESCRIPTOR TABLE | >0400 |
| SPRITE MOTION TABLE | >0780 |

Sprites are numbered from 0 to 31. Here's how the screen is defined for Sprites:
Columns are labeled starting from the left from 0 to 255 (>00 to >FF). Rows are numbered from top left, the first row is numbered 256 (>100), followed by the numbers 0 to 190 (>0 to >BE). Each screen location defined in this manner is referred to as a pixel. A pixel is the smallest area of the screen you can turn on or off. Here's the way it looks:
Pixel 1 is in row >100 column >02.
P4 is in row >BE column >01.

Here are the formulas to convert row and column locations to pixel locations:

GRAPHIC TO PIXEL CONVERSIONS

GRAPHIC ROW TO PIXEL ROW
GR*8-7=PR
GRAPHIC COLUMN TO PIXEL COLUMN
GC*8-7=PC

```
PIXEL ROW TO GRAPHIC ROW
       INT[(PR+7)/8]=GR
PIXEL COLUMN TO GRAPHIC COLUMN
       INT[(PC+7)/8]=GC
------------------------------
```

SPRITE ATTRIBUTE TABLE

Begins at VDP >0300 by default. Contains the present position of sprites and their colors. Each sprite takes up four bytes in the table. The first byte is the row or Y position of the sprite. The second byte is the column or X position. The third byte references the pattern of the sprite as to where it is located in the Sprite Descriptor Table. The fourth byte is the early clock attribute and also codes for the color of the sprite.

When the computer moves sprites it updates the information in the sprite attribute table. The more sprites it has to update the longer it takes to execute the program. To shorten this time place a value of >D0 as the Y location of the lowest numbered non-moving sprite. Always let the final unused sprite be undefined by specifying the Y location as >D0.

The third byte references a pattern in the Pattern Descriptor Table. Can range from >00 to >FF. For example if the third byte contained >80 it would point to >0400 through >0407 in the Sprite Descriptor Table.

The forth byte controls the early clock and color. The first four bits control the early clock. If the last bit (3) is reset to zero the early clock is off and the location of the sprite is said to be it's upper left hand corner. This means the sprite will fade in and out on the right hand side of the screen. If bit 3 is on the sprites location is shifted 32 pixels to the left. The sprite can then fade in and out from the left side of the screen.

Bits 4-7 of byte four contain the color. Same as other VDP colors 0 to >F.

Here's an example Sprite Attribute:

```
SAL   DATA   >3356,>8001  Sprite 0
      DATA   >A828,>810F  Sprite 1
      DATA   >D0 - 3rd Sprite Undef
              / /   / /
             Y X   / color
               pattern
```

SPRITE DESCRIPTOR TABLE

Just like the pattern descriptor table for characters. Usually begins at >0400. Addresses >0400 through >0407 are defined as sprite pattern >80.

You can also make sprites magnified or double sized by writing a value to the two least significant bits of VDP register 1.

SPRITE MOTION TABLE

Describes the X and Y velocities of each sprite. This table begins at >0780. Before a sprite can be placed into motion several conditions must be met. Your program must allow interrupts using LIMI 2 but before accessing VDP RAM you must disable interrupts with a LIMI 0. You must indicate how many sprites will be in motion by placing a value at CPU address >837A. For example if sprites 2, 5, and 7 are in motion you must place an >8 at address >837A which will allow motion of 0 through 7. A description of the motion must be placed in the Sprite Motion Table. Each sprite takes up four bytes in the table. The first byte is the Y velocity, the second byte is the X velocity. The third and fourth bytes are used by the interrupt routines, just be sure you leave space for them. The following are allowed as values for X and Y velocities:

A value of >01 will cause the sprite to move one pixel every 16 VDP interrupts. About once every 16/60 of a second.

A thought: Have you ever seen a screen dump program that would dump sprites? It could be done by obtaining their location and pattern and converting to printer bit graphics. Have fun!

```
*******************************************************
* CALL SPRITE, PROGRAM PLACES A HELICOPTER            *
* SPRITE IN MOTION BY ENABLING INTERRUPTS. PRESS ANY *
* KEY TO ALTER MAGNIFICATION, BY MACK MCCORMICK       *
*******************************************************
       DEF  START
       REF  VMBW,VWTR,KSCAN
*
HELI   DATA >007F,>0000,>0107,>0E0E   HELICOPTER PATTERN DESCRIPTION
       DATA >1EBE,>FFBF,>0F07,>020F   BLOCK 2
       DATA >00FF,>8080,>C0F8,>04C2          3
       DATA >DACA,>FEFC,>F8E0,>40F8          4
SDATA  DATA >7080,>8008  INITIAL SPRITE DATA
       DATA >D000        >D0 PREVENTS GHOST SPRITES
*
SPEED  DATA >0A0F,>0000  SPRITE SPEED FOR AUTO MOTION
STATUS EQU  >837C        GPL STATUS BYTE
VDP    DATA >01E0        INITIAL VALUE OF VDP REGISTER 1
MYREG  EQU  >8300        MYREG IN 16 BIT HIGH SPEED AREA OF MEMORY
*
START  LWPI MYREG
       CLR  @>8375       KEYBOARD DEVICE = 0. SCAN ALL.
       MOV  @VDP,R6
       LI   R0,>0400     LOAD  (BASE ADDRESS OF SPRITE DESCRIPTOR TABLE)
       LI   R1,HELI           SPRITE
       LI   R2,32                   DESCRIPTOR
       BLWP @VMBW                             TABLE
*
       LI   R0,>0300     LOAD  (BASE ADDRESS OF SPRITE ATTRIBUTE TABLE)
       LI   R1,SDATA          SPRITE
       LI   R2,6                    ATTRIBUTE
       BLWP @VMBW                             TABLE
*
       LI   R0,>0780     LOAD  (BASE ADDRESS OF SPRITE MOTION TABLE)
       LI   R1,SPEED          SPRITE
       LI   R2,4                    MOTION
       BLWP @VMBW                           TABLE
*
       LI   R1,>0100
       MOVB R1,@>837A    ONE SPRITE IN MOTION
*
LOOP   CLR  @STATUS
       BLWP @KSCAN
       MOVB @STATUS,@STATUS HAS KEY BEEN PRESSED?
       LIMI 2            ENABLE INTERRUPTS FOR AUTO MOTION
       LIMI 0            DISABLE INTERRUPTS SO VDP IS NOT AFFECTED ON READ/WR
       JEQ  LOOP
*
CHECK  INC  R6           R6 IS USED AS A COUNTER TO KEEP
       CI   R6,>01E4     TRACK OF WHICH MAGNIFICATION
       JLT  GO           LEVEL (1 TO 4) WE ARE ON.
       MOV  @VDP,R6
*
GO     MOV  R6,R0        LOAD R0 WITH DATA TO LOAD INTO VDP R1
       BLWP @VWTR        CHANGE THE VDP REGISTER
       B    @LOOP
       END
```

* FOR EXTRA PRACTICE ADD A ROUTINE THAT SHOWS THE X AND Y POSITION OF THE SPRITE
* ON THE SCREEN AS IT MOVES. HINT: Y LOCATION IS 1ST BYTE IN SPRITE ATTRIBUTE
* LIST. X SECOND BYTE. READ THEM, CONVERT TO ASCII DECIMAL AND REDISPLAY WITH
* APPROPRIATE TEXT. WHO'LL BE FIRST? ■

# HYBRIDPROGRAM XB OCH AL

*av Jan Alexandersson*

Det finns flera sätt att gömma ett assemblerprogram i ett Extended Basic program. Som exempel används här BUBBLE/O från PB 91-2 men du kan välja en egen fil. Jag ska här beskriva tre olika program som ordnar omvandlingen:

- ALSAVE av Todd Kaplan som är public domain (se Micropendium sep 90 sid 17)

- SYSTEX av Barry Boone (finns på skivan till EASYMUSIC i programbanken)

- LOADASMBAS av Börje Häll (se PB 84-1)

ALSAVE AV TODD KAPLAN

Du måste ha filen ALSAVE för att kunna göra omvandlingen. Sänd en skiva och frankerat kuvert så sänder jag filen till dig. Du gör omvandlingen på följande sätt:

1. Knappa in programmet ALLOAD;
```
10 CALL INIT :: CALL LOAD
   (8196,63,248):: CALL LOAD
   (16376,65,76,83,65,86,69,255,48)
   :: CALL LINK("ALSAVE")
```

2. Spara det som SAVE DSK2.ALLOADM, MERGE

3. NEW

4. CALL INIT

5. CALL LOAD("DSK2.BUBBLE/O")

6. CALL LOAD("DSK2.ALSAVE")

7. CALL LINK("SAVE")

8. 100 CALL LINK("BUBBLE")

9. MERGE DSK2.ALLOADM

10. SAVE DSK2.BUBBLE/ALS

SYSTEX AV BARRY BOONE

Du gör omvandlingen på följande sätt:

1. CALL INIT

2. CALL LOAD("DSK2.BUBBLE/O")

3. OLD DSK2.SYSTEX

4. RUN

5. Y för YES

6. CALL LINK("SYSTEX")

7. 100 CALL LINK("BUBBLE")

8. SAVE DSK2.BUBBLE/SYS

LOADASMBAS AV BÖRJE HÄLL

Detta kräver AORG och att BSS ej används. Som exempel använder jag här en modifierad version av BUBBLE/SXB från PB 91-2 sid 12 längst ned till höger. Lägg till en rad före COPY med AORG >A000 och ändra COPY till COPY "DSK2.MAIN/BH" och spara den som BUBBLE/SBH. Ändra filen MAIN genom att använda RS (Replace String): /BSS >20/DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0/. Spara detta som MAIN/BH. Assemblera sedan BUBBLE/SBH till BUBBLE/OBH. LOADASMBAS ska sedan omvandla BUBBLE/OBH till DSK1.MERGE. Skriv sedan NEW och MERGE DSK1.MERGE. Lägg till en rad på slutet: 500 CALL LINK("BUBBLE") och spara resultatet som BUBBLE/BH.

```
120 ! Programmet är skrivet
av programmet
130 ! LOADASMBAS
140 ! LOADASMBAS läser en as
sembler OBJEKT-
150 ! kod och gör om objektk
odens data till
160 ! CALL LOAD-satser.
170 ! LOADASMBAS är skrivet
av
180 ! Börje Häll
220 CALL INIT
240 ! * Ladda assemblerprogr
ammet
```

```
260 CALL LOAD(-24576,60,126,
207,223,255,255,126,60,243,5
1,160,168,1,218,2,13,2,113)
270 CALL LOAD(-24558,2,165,2
,214,2,225,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0)
280 CALL LOAD(-24536,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,2,224,160,26)
290 CALL LOAD(-24514,6,160,1
61,14,2,0,3,148,2,1,160,8,2,
2,0,2,4,32,32,36,2,0)
300 CALL LOAD(-24492,13,0,2,
1,160,0,2,2,0,8,4,32,32,36,4
,192,208,96,160,11,4,32)
310 CALL LOAD(-24470,32,32,5
,128,2,128,3,0,22,250,208,96
,160,10,2,2,160,12,192,50,19
2,0)
320 CALL LOAD(-24448,19,35,4
,32,32,32,16,250,0,0,0,0,0,0
,0,0,0,0,0,0,0,0)
330 CALL LOAD(-24426,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0)
340 CALL LOAD(-24404,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0)
350 CALL LOAD(-24382,0,0,0,0
,0,0,4,192,2,1,160,136,2,2,0
,32,4,32,32,44,2,0)
360 CALL LOAD(-24360,0,32,2,
1,160,168,2,2,0,32,4,32,32,4
4,2,32,255,224,4,32,32,36)
370 CALL LOAD(-24338,2,32,0,
64,2,128,3,0,26,245,2,0,2,22
4,2,1,160,136,4,32,32,36)
380 CALL LOAD(-24316,3,0,0,2
,3,0,0,0,16,221,2,0,3,14,4,3
2,32,48,2,0,4,1)
390 CALL LOAD(-24294,4,32,32
,48,2,0,7,243,4,32,32,48,2,0
,1,224,4,32,32,48,4,91)
410 ! * Uppdatera REF/DEF-ta
bellen
420 ! * Referenser
430 ! * BUBBLE
450 CALL LOAD(16376,66,85,66
,66,76,69,160,58)
470 ! * Uppdatera pekaren ti
ll REF/DEF-tabellen
490 CALL LOAD(8196,63,248)
500 CALL LINK("BUBBLE")
```

■

---

## GENEVE ARTICLES

*by Don Jones, Chicago UG, USA*

Well, I have finally compiled ALL that I have in DV/80 format. I have also found that I have exactly 6 which I no longer have the DV/80 files for. This includes my first Geneve Support Article, which was written for the 12/31/87 issue of this newsletter. The other 5 of the 6 missing articles are all from the year 1988. I have all of the others available for the following years. I have "archived" them by year, and they require 1011 sectors of disk storage space. This means that if you only have SS/SD drives, you will require 4 disks to store them. If you have DS/SD, you will require 2 disks. If you have DS/DD, only one disk will be required. (Don't forget, we are talking about approximately 30 articles!) Jim Baird our Geneve librarian will have these disks available for purchase either at the next meeting or by mail. The cost for them all will be a $10.00 donation to our group. You may be asking what am I going to do about the 6 articles which are currently not in DV/80 format. As I said in my last article, I intend on returning them into DV/80 format >OR< I will simply provide a hard copy of these articles to those who purchase the disks. (If you are one of the few members who have already purchased these articles prior to April 15, you won't have the missing 6 and a couple of other articles which I did not initially have access to. If you write to Jim and inform him of who you are, I will ask him to send you the completed package.)

One last thing: By now, it should be fully apparent to my readers that my stuff is really directed towards the non-technical, beginner. If you want something more advanced, I suggest a subscription to Beery Miller's "diskazine,: >9640 News<. It is very well done and it is definitely worth your patronage. Also, don't forget about >MICROpendium Magazine<. They have a lot of interesting and fine articles on Geneve and her older brother, the 4A.

■

# PHM – MODULER FÖR 99/4A

*av Jan Alexandersson*

| | | |
|---|---|---|
| PHM3000 | DIAGNOSTIC | OP |
| PHM3001 | DEMOSTRATION | UV |
| PHM3002 | EARLY LEARNING | UB |
| PHM3003 | BEGINNERS GRAMM. | UB |
| PHM3004 | NUMBER MAGIC | UB |
| PHM3005 | VIDEO GRAPHS | GR |
| PHM3006 | HOME FINANC.DEC. | HA |
| PHM3007 | H.BUDGET MANEGEM | HA |
| PHM3008 | CHESS | SP |
| PHM3009 | FOOTBALL | SP |
| PHM3010 | PHYSICAL FITNESS | UV |
| PHM3011 | SPEECH EDITOR | MU |
| PHM3012 | SECURIT.ANALYSIS | HA |
| PHM3013 | PERS.RECORD KEEP | HA |
| PHM3014 | STATISTICS | HA |
| PHM3015 | EARLY READING | UB |
| PHM3016 | TAX/INVESTM.REC | HA |
| PHM3018 | VIDEO GAMES I | SP |
| PHM3020 | MUSIC MAKER | MU |
| PHM3021 | WEIGHT CONTROL.N | UV |
| PHM3022 | PERS.REAL ESTATE | HA |
| PHM3023 | HUNT THE WUMPAS | SP |
| PHM3024 | SOCCER | SP |
| PHM3025 | MIND CHALLENGERS | SP |
| PHM3026 | EXTENDED BASIC | SK |
| PHM3027 | ADD AND SUB I | UB |
| PHM3028 | ADD AND SUB II | UB |
| PHM3029 | MULTIPLICAT. I | UB |
| PHM3030 | A-MAZ-ING | SP |
| PHM3031 | ATTACK | SP |
| PHM3032 | BLASTO | SP |
| PHM3033 | BLACK JACK/POKER | SP |
| PHM3034 | HUSTLE | SP |
| PHM3035 | TERM.EMULATOR II | SK |
| PHM3036 | ZERO ZAP | SP |
| PHM3037 | HANGMAN | SP |
| PHM3038 | CONNECT FOUR | SP |
| PHM3039 | YAHTZEE | SP |
| PHM3040 | TI LOGO | SK |
| PHM3041 | ADVENTURE | SP |
| PHM3042 | TUNNELS OF DOOM | SP |
| PHM3043 | READING FUN | UB |
| PHM3044 | PERS.REPORT GEN. | HA |
| PHM3045 | ELECTRICAL E.LIB | TE |
| PHM3046 | READING ON | UB |
| PHM3047 | READING ROUNDUP | UB |
| PHM3048 | READING RALLY | UB |
| PHM3049 | DIVISION I | UB |
| PHM3050 | NUMERATION I | UB |
| PHM3051 | NUMERATION II | UB |
| PHM3052 | TOMBSTONE CITY | SP |
| PHM3053 | TI INVAIDERS | SP |
| PHM3054 | CAR WARS | SP |
| PHM3055 | EDITOR/ASSEMBLER | SK |
| PHM3056 | ALPINER | SP |
| PHM3057 | MUNCHMAN | SP |
| PHM3058 | MINI MEMORY | SK |
| PHM3059 | SCH.SPELLING-L3 | UB |
| PHM3060 | SCH.SPELLING-L4 | UB |
| PHM3061 | SCH.SPELLING-L5 | UB |
| PHM3062 | SCH.SPELLING-L6 | UB |
| PHM3064 | TOUCH TYPING TUT | UV |
| PHM3065 | ENGLISH TRAINER | UB |
| PHM3066 | INDIVID ACCOUNT | HA |
| PHM3067 | OTHELLO | SP |
| PHM3069 | EARLY LEARNING | UB |
| PHM3082 | READING FLIGHT | UB |
| PHM3083 | COMP.MATH GAME 1 | UB |
| PHM3084 | COMP.MATH.GAME 2 | UB |
| PHM3085 | COMP.MATH.GAME 3 | UB |
| PHM3086 | COMP.MATH.GAME 4 | UB |
| PHM3087 | COMP.MATH.GAME 5 | UB |
| PHM3088 | COMP.MATH GAME 6 | UB |
| PHM3089 | DISK MANAGER 2 | OP |
| PHM3090 | ADDITION | UB |
| PHM3091 | SUBTRACTION | UB |
| PHM3092 | MULTIPLICATION | UB |
| PHM3093 | DIVISION | UB |
| PHM3094 | INTEGERS | UB |
| PHM3095 | FRACTIONS | UB |
| PHM3096 | DECIMALS | UB |
| PHM3097 | PERCENTS | UB |
| PHM3098 | NUMBER READINESS | UB |
| PHM3099 | LAW OF ARITMETIC | UB |
| PHM3100 | EQUATIONS | UB |
| PHM3101 | MEASUREMENT FORM | UB |
| PHM3109 | TI LOGO II | SK |
| PHM3110 | CHRISHOLM TRAIL | SP |
| PHM3111 | TI WRITER | HA |
| PHM3112 | PARSEC | SP |
| PHM3113 | MULTIPLAN | HA |
| PHM3114 | ALIGATOR MIX | UB |
| PHM3115 | ALIEN ADDITION | UB |
| PHM3116 | DEMOLITION DIV | UB |
| PHM3117 | DRAGON MIX | UB |
| PHM3118 | MINUS MISSION | UB |
| PHM3119 | METEOR MULTIPLIC | UB |
| PHM3122 | PLATO INTERPRETE | UV |
| PHM3131 | MOONMINE | SP |
| PHM3144 | LOGO LEARNING | UV |
| PHM3145 | SNEGGIT | SP |
| PHM3146 | MUNCHMOBILE | SP |
| PHM3148 | MBX:BASEBALL | SP |
| PHM3149 | MBX:SPACE BANDIT | SP |
| PHM3150 | MBX:SEWERMANIA | SP |
| PHM3151 | MBX:BIGFOOT | SP |
| PHM3152 | MBX:METEOR BELT | SP |
| PHM3153 | MBX:SUPERFLY | SP |
| PHM3154 | MBX:TURTLE ADVEN | SP |
| PHM3155 | MBX:I'M HIDING | SP |
| PHM3156 | MBX:HONEY HUNT | SP |
| PHM3157 | MBX:SOUND TRACK | SP |
| PHM3158 | MASH | SP |
| PHM3159 | CHOPLIFTER | SP |

```
PHM3168 TREASURE ISLAND   SP
PHM3169 WORD INVATIONS    UB
PHM3177 FACE MAKER        UB
PHM3178 STORY MACHINE     UB
PHM3185 WORD READER       UB
PHM3189 PIRATE'S ISLAND   SP
PHM3194 JAWBREAKER        SP
PHM3197 SLYMOIDS          SP
PHM3212 SCRABBLE          SP
PHM3213 TI-CALC           HA
PHM3214 NUMBER GULPER     UB
PHM3215 HIDE AND SEEK     SP
PHM3219 DEMON ATTACK      SP
PHM3220 MICRO SURGEON     SP
PHM3222 FATHOM            SP
PHM3224 MOONSWEEPER       SP
PHM3225 STARTREK          SP
PHM3226 BUCK ROGERS       SP
PHM3227 CONGO BONGO       SP
PHM3229 HOPPER            SP
PHM3233 BURGERTIME        SP■
```

---

## KONTROLLSIFFRA

```
100 ! Generator för kontroll
siffra/postgiro/personnummer
140 !**********************
160 ! Tiomodulsmetoden
180 ! 23 692
190 ! 21 212       Börja från
 höger med 2121 osv.
200 ! ------       2*2=4, 1*9
=9, 2*6=12 ger 1 o. 2
210 ! 431294 = 23 Siffrorna
adderas var för sig (12=1 oc
h 2)
220 ! Subtrahera närmast hög
re tiotal
230 ! 30 - 23 = 7
240 ! Om summan blir jämt de
lbar med tio är siffran 0
260 !**********************
280 DISPLAY AT(1,1)ERASE ALL
:"# KONTROLLSIFFERGENERATOR
#"
290 DISPLAY AT(3,1):"  Ange
siffrorna i numret      eller
 talet i en foljd,      utan
mellanrum eller         avdel
ade med tecken."
300 DISPLAY AT(21,1):"Max 11
 siffror                ENT ra
derar allt              = for
exit"
310 C=1
320 FOR B=9 TO 19 STEP 2
330 ACCEPT AT(B,C)VALIDATE(D
IGIT,"=")BEEP SIZE(-11):T$
340 IF T$="" THEN GOTO 280 !
 Radera allt
350 IF T$="=" THEN END 770 :
: GOTO 280 !Exit
360 GOSUB 490 ! Uträkning
370 ! Display av kontrollsif
fran
380 DISPLAY AT(B,C+S)SIZE(-2
):"-";STR$(Z)
390 NEXT B
400 IF C=1 THEN C=15 ELSE C=
1 ! Kolumnändring
410 ! Radera en kolumn
420 FOR CLR=9 TO 19 STEP 2
430 DISPLAY AT(CLR,C)SIZE(-1
3):""
440 NEXT CLR
450 GOTO 320
470 !**********************
490 ! Uträkning
500 S=LEN(T$)! Längden av ta
let
510 P=0 !        Summan
520 L=10 !       Ger L=2 i fö
rsta loopen
530 FOR I=0 TO S-1 ! Loop so
m multiplicerar varannan sif
fra i talet med 2 resp. 1
540 IF L=2 THEN L=1 ELSE L=2
550 TAL=VAL(SEG$(T$,S-I,1))!
 TAL är resp. siffra i talet
560 P=P+L*TAL !
  P är den acc. summan
570 IF L=2 AND TAL>4 THEN P=
P-9 ! Justerar för TAL 10 el
ler högre
580 NEXT I
590 Z=(INT(P/10)+1)*10-P !
  Z är kontrollsiffran
600 IF Z=10 THEN Z=0
610 RETURN                    ■
```

---

```
90 REM DECIMAL-KOMMA ENSAT
92 REM FRÅN PB 84-5.13
95 REM ANVEND RAD 230 ELLER
96 REM 330 SOM RAD 130
97 REM TALET X SKRIVS MED
98 REM DECIMALKOMMAT ENSAT
100 C=10
110 FOR I=1 TO 10
120 READ X
130 PRINT TAB(C-LEN(STR$(INT
(ABS(X))))-(X<1)+(X<0)+((X<0
)*(X>-1)));X
140 NEXT I
150 DATA 3,0.354,-2.4,120,55
55,-300.7,.006,100,56,9,-345
.09
160 END
230 PRINT TAB(C-LEN(STR$(INT
(ABS(X))))-(X<1)+(X<0)+((X<0
)*(X>-1)));X
330 PRINT TAB(C-POS(STR$(X)&
".",".",1)-(X<0));X          ■
```

# TIPS FROM TIGERCUB #44



TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Thanks to Steve Chapman and Bill Wallbank of Stone & Webster Engineering Corp. TIUG for this one. If V=21 you are in Extended Basic, otherwise you are in Basic. I am not sure it will work with all consoles and modules. -

```
100 RANDOMIZE (0)
110 V=INT(RND*100)
```

How can you input a blank (CHR$ 32) with ACCEPT AT? As far as I know, you can't. With LINPUT, just hit the space bar, and with INPUT, type " ". But with ACCEPT AT the space bar gives a null string and " " gives " " ! However, you can code around it -

```
X$=CHR$(34)&CHR$(32)&CHR$(32
)::  ACCEPT AT(1,1):T$ :: IF
T$=X$ THEN T$=CHR$(32)
```

And, to clear up the puzzling behavior of the "quote marks" -

```
100 CALL CHARPAT(34,CH$):: C
ALL CHAR(35,CH$)!written by
Jim Peterson
110 DISPLAY AT(1,7)ERASE ALL
:"THE # PUZZLE":" You can't
enter PRINT # or PRINT ### -
 the computer    demands an
even number of #."
120 DISPLAY AT(5,1):"1 PRINT
 ## !prints a null string (n
othing)":"2 PRINT #*# !print
s *"
130 DISPLAY AT(8,1):"3 PRINT
 #### !prints #":"4 PRINT ##
*## !crashes as   STRING-NUM
BER MISMATCH"
140 DISPLAY AT(11,1):"5 PRIN
T ##*## !crashes as  SYNTAX
 ERROR"
150 DISPLAY AT(13,1):"6 PRIN
T ###### !prints ##":"7 PRIN
T ###*### !prints #*#":"8 PR
INT ###*### !print #**#"
160 DISPLAY AT(16,1):"9 PRIN
T ######## !prints ###":"10
PRINT ####*#### !crashes as
STRING-NUMBER MISMATCH"
170 DISPLAY AT(19,1):"11 PRI
NT ####*#### !crashes as SY
NTAX ERROR":"12 PRINT ######
#### !####"
180 DISPLAY AT(22,1):"13 PRI
NT #####*##### !##*##":"14 P
RINT ######*#####!##*##"
190 DISPLAY AT(24,1):"TRY IT
! LINE NO.(1-14)?" :: ACCEPT
 AT(24,25)VALIDATE(DIGIT)SIZ
E(2)BEEP:LN :: IF LN<1 OR LN
>14 THEN 190
200 CALL CLEAR :: ON LN GOSU
B 230,240,250,260,280,290,30
0,310,320,330,340,350,360,37
0
210 PRINT :;:;:"Press any ke
y"
220 CALL KEY(0,K,S):: IF S=0
 THEN 220 ELSE 110
230 PRINT "" :: RETURN
240 PRINT "*" :: RETURN
250 PRINT """" :: RETURN
260 PRINT ""*"" !crashes as
STRING-NUMBER MISMATCH - the
 * is misinterpreted as a mu
ltiplier!Same with +,-,/
270 !with anything else, inc
luding numerals, crashes as
SYNTAX ERROR - but inserts a
 space before the character!
280 PRINT ""**"" :: !crashes
290 PRINT """""" :: RETURN
300 PRINT """*""" :: RETURN
310 PRINT """**""" :: RETURN
320 PRINT """""""" :: RETURN
330 PRINT """"*"""" !crash
340 PRINT """"**"""" !crash
350 PRINT "":: RETU
RN
360 PRINT """""*""""" :: RET
URN
370 PRINT """""**"""""" :: RE
TURN
```

The method of closing an "ajar" file, described in Tips #28, doesn't always work, but this one seems to be reliable -

```
100 ON ERROR 500 :: OPEN #1:
"DSK1.TEST" :: INPUT #1:A$ :
: PRINT A$ :: STOP
500 ON ERROR 510 :: CLOSE #1
510 INPUT "CHECK DISK AND DR
IVE, PRESS ANY KEY":DUMMY$ :
```

```
: RETURN 100
```

This one is just for the fun of it - it uses the contents of computer memory to create designs -

```
100 DISPLAY AT(3,10)ERASE ALL:"COLORPEEK": :TAB(7);"by Jim Peterson": : :" Watch the computer's memory": :"displayed in color."
110 DISPLAY AT(12,1):"Choose": :"(1) plain colors": :"(2) bars & checks": :"(3) patterns" :: ACCEPT AT(12,8)VALIDATE("123")SIZE(1):Q :: CALL CLEAR :: IF Q=1 THEN 170
120 DISPLAY AT(12,5):"wait, please" :: IF Q=3 THEN 140
130 FOR CH=32 TO 143 :: CALL CHAR(CH,RPT$("F0",8)):: NEXT CH :: GOTO 160
140 RANDOMIZE :: FOR CH=32 TO 88 :: FOR J=1 TO 4 :: X$=SEG$("0018243C425A667E8199A5BDC3DBE7FF",INT(16*RND+1)*2-1,2):: B$=B$&X$ :: C$=X$&C$ :: NEXT J :: CALL CHAR(CH,B$&C$)
150 CALL CHAR(CH+55,B$&C$):: B$,C$="" :: NEXT CH
160 FOR SET=0 TO 14 :: CALL COLOR(SET,SET+1,16-SET):: NEXT SET :: CALL SCREEN(2):: GOTO 180
170 FOR SET=0 TO 14 :: CALL COLOR(SET,SET+2,SET+2):: NEXT SET :: CALL SCREEN(16)
180 FOR J=-1 TO -2000 STEP -1 :: CALL PEEK(J,A):: A=A-(A<33)*(A+32):: A=A+(A>143)*(A/2):: R=R+1+(R=24)*24 :: CALL HCHAR(R,1,A,32)
190 C=C+1+(C=32)*32 :: CALL VCHAR(1,C,A,24):: NEXT J :: GOTO 100
```

Unlike most of the number games played against the computer, you can win this one -

```
100 CALL CLEAR :: CALL SCREEN(16):: DISPLAY AT(3,8):"THE `37' GAME" !by Jim Peterson
110 DISPLAY AT(5,1):" We will take turns picking":"a number from 1 to 5, but":"not the number that was just":"picked."
120 DISPLAY AT(10,1):" The numbers we pick will be":"added to the total count."
130 DISPLAY AT(13,1):" Whoever reaches 37 is the":"winner, but if you go over":"37 you lose."
140 CALL SHOW(20,1,"Press any key to start")
150 CALL KEY(0,K,S):: IF S=0 THEN 150
160 DATA 4,11,17,24,30,37
170 DATA 262,330,392,523,523
180 DATA 1047,784,659,523,523
190 C,P=0 :: CALL CLEAR :: CALL MAGNIFY(2):: R=10 :: FOR J=1 TO 5 :: CALL SPRITE(#J,48+J,5,R,10):: R=R+30 :: NEXT J
200 CALL SHOW(24,1,"(Y)ou or (C)omputer first?"):: ACCEPT AT(24,28)VALIDATE("YC")SIZE(1):Q$ :: DISPLAY AT(24,1):""
210 IF Q$="C" THEN CALL SHOW(22,8,"I pick 4"):: CALL COLOR(#4,1):: P=4 :: C=4 :: CALL SHOW(3,10,"COUNT=4")
220 CALL SHOW(20,8,"Pick your number"):: ACCEPT AT(20,26)VALIDATE("12345"):N :: IF N=P THEN 220
230 IF P>0 THEN CALL COLOR(#P,5)
240 CALL COLOR(#N,1):: P=N :: C=C+N :: CALL SHOW(3,10,"COUNT= "&STR$(C)):: IF C=37 THEN 320 ELSE IF C>37 THEN 340
250 RESTORE 160
260 READ X :: IF C<X THEN B=X-C ELSE IF X<37 THEN 260
270 CALL SHOW(22,8,"I'm thinking..."):: FOR Y=1 TO 700 :: NEXT Y
280 IF B>5 AND B/2=INT(B/2)THEN B=B/2
290 IF B>5 OR B=P THEN B=1-(P=1)
300 CALL SHOW(22,8,"I pick "&STR$(B)):: CALL COLOR(#P,5):: CALL COLOR(#B,1):: P=B :: C=C+B :: CALL SHOW(3,10,"COUNT= "&STR$(C))
310 IF C=37 THEN 340 ELSE IF C>37 THEN 320 ELSE 220
320 RESTORE 170 :: FOR J=1 TO 5 :: READ F :: CALL SOUND(100,F,5,F*1.03,5):: NEXT J :: CALL SHOW(12,8,"YOU WIN!")
330 CALL SHOW(15,8,"Play again? (Y/N)"):: ACCEPT AT(15,26)VALIDATE("YN"):Q$ :: IF Q$="N" THEN STOP ELSE 190
340 RESTORE 180 :: FOR J=1 T
```

```
O 5 :: READ F :: CALL SOUND(
300,30000,30,30000,30,F,30,-
4,5):: NEXT J :: CALL SHOW(1
2,8,"YOU LOSE!"):: GOTO 330
350 SUB SHOW(R,C,T$):: FOR J
=1 TO 10 :: DISPLAY AT(R,C):
" " :: DISPLAY AT(R,C):T$ ::
 NEXT J :: SUBEND
```

A couple more peculiarities of the computer -

```
100 DISPLAY AT(3,8)ERASE ALL
:"POS PUZZLE #1": :"       f
rom Tigercub"
110 DISPLAY AT(9,1):"Why doe
s the computer say":"that X=
1 if you answer the":"prompt
 with the Enter key":"(null-
string) ?"
120 DISPLAY AT(14,1):"110 IN
PUT M$"
130 DISPLAY AT(15,1):"120 X=
POS(""TESTING"",M$,1)::":"PR
INT X :: GOTO 100"
140 !POS PUZZLE #1 - why doe
s the computer say that X=1
if you answer the prompt wit
h Enter (null-string) ?
 - Jim Peterson
150 INPUT M$
160 X=POS("TESTING",M$,1)::
PRINT X :: GOTO 140
```

And -

```
100 DISPLAY AT(3,8)ERASE ALL
:"POS PUZZLE #2": :"       f
rom Tigercub"
110 DISPLAY AT(7,1):"Why doe
s the computer say":"that th
e first position of":"null-s
tring is at whatever":"posit
ion it is told to start":"se
arch at?"
120 DISPLAY AT(13,1):"100 M$
="""""
130 DISPLAY AT(14,1):"110 DI
SPLAY AT(20,1):""POS?"" :: A
CCEPT AT(20,6):P"
140 DISPLAY AT(16,1):"120 X=
POS(""TESTING"",M$,P):: DISP
LAY AT(22,1):""X="";X :: GOT
O 110"
150 M$=""
160 DISPLAY AT(21,1):"POS?"
:: ACCEPT AT(21,6):P
170 X=POS("TESTING",M$,P)::
DISPLAY AT(23,1):"X=";X :: G
OTO 160
```

Here is an improvement to the PRINTSPEAKER in Tips #40 - in lines 130 and 160, change the CHR$(1)&"1" to CHR$(3)&"255" . This will avoid problems if the program being converted opens FILE #1.

Irwin Hott informs me that assembly routines which have been imbedded into XBasic programs, using ALSAVE or SYSTEX, can be saved to cassette and reloaded. This could be very useful for those who have a stand-alone or "matchbox" 32k.

And, a mini-game for you to have fun with or improve on -

```
1 !       2-LINE GAME
        by Jim Peterson
- use S&D keys to paint the
white line on the highway
2 !if it is too easy, change
the 6 in A$=RPT$(CHR$(143),6
) to 5 and the 5 in C>T+5 to
 4
100 CALL CLEAR :: A$=RPT$(CH
R$(143),6):: CALL COLOR(14,2
,2,2,16,16):: CALL SCREEN(4)
:: T=11 :: C=14 :: CALL HCHA
R(22,C+2,42):: RANDOMIZE
110 T=T+INT(3*RND-1)+(T=21)-
(T=1):: PRINT TAB(T);A$ :: C
ALL KEY(3,K,S):: C=C+(K=83)-
(K=68):: CALL HCHAR(22,C+2,4
2):: IF C<T OR C>T+5 THEN ST
OP ELSE 110
```

And finally, one of the best examples of compact programming I have ever seen -

```
1 !JOHN WITTE'S 3-LINE VERSI
ON OF JOHN WILLFORTH'S WAVE
POWER - PUBLISHED IN GREATER
 OMAHA UG NEWSLETTER
100 CALL CLEAR :: A$(1)="ABC
DEFGFEDCBA" :: FOR I=1 TO 7
:: CALL CHAR(72-I,RPT$("0",2
*I-2)&"FFFF",47,"30303EFF7F3
E1E04"):: A$(I+1)=SEG$(A$(I)
,2,12)&SEG$(A$(I),2,1):: NEX
T I
110 CALL SPRITE(#5,47,2,180,
180,-23,0,#6,47,2,80,100,-23
,0):: CALL MAGNIFY(2)
120 FOR I=1 TO 12 :: PRINT A
$(I+(I>7)*2*(I-7))&A$(1+I+(I
>6)*2*(I-6)):: NEXT I :: GOT
O 120
```

Memory full Jim Peterson■

# USER-TREFFEN IN BERLIN 13-15 SEPT 1991

*by Alexander Hulpke, Germany*

For those of you who - be it the large distance, be it other duties - were not able to come to the annual meeting at Berlin this September, I'll try to sum up the highlights and give some report about it. It is of course my personal impressions, and I hope I didn't miss anything and anybody, otherwise sorry! Though the excellent organization of the meeting (but I should note, that we are used to it) by the Berlin user group, especially Franz Neudert and Henry Hillsberg, who also provided some "tourist program" for those who were interested, the audience was significant small compared to the last year, 360 to 100 approximately, if I got the figures right. I assume this is not just falling interest, but also consequence of the very distant and noncentral location of Berlin (which also will be a major disadvantage for it as capital). Nevertheless some guys even came from Belgium and the Netherlands, but for example I did not see anyone from Austria, who came the last years. I won't touch audience and non TI program any more, but describe the various new or modified programs shown:

Starting with the hardware, first I should note, that a lot of people had a modified p-Box; even though it housed the usual cards, you could not recognize it at a TI system at the first glance, also concerning the large varieties of disk drives; from 5 1/4" 90KB up to 3.5" 1.44MB. Consequently also a card with voltage regulators was shown, that allows You to connect more disk drives to the p-Box's internal power supply.

Most systems had an 80 column upgrade, and there were also about 5 Geneve Systems, unfortunately none with the 64k video upgrade.

Not astonishing the new hardware shown had connection to the 9938:

First there was an small modification to the 80 column card to allow the use of a standard Color Look Up Table (CLUT) as used on VGA cards. This allows the 16 or 256 colors to be selected from about 256000 or so, beating the 9958 by length. The modification, which was invented and shown by Sven Dyroff, costs about 30DM in parts and requires the connection of the CLUT to the color bus and the installment of an 8 Byte port for programing the CLUT. As there was no special 80 column card hardware, it should be possible easily to adapt it also to the DIJIT card or the Geneve. The only problem especially on the Geneve) will be to find an 8 Byte area still unused, that could be decoded for this device. Also some power-up software has to be included to reset the standard palette. even though the circuit was build on a breadboard, it worked flawlessly and the video signal was very stable. If I knew a free memory area to decode, this would be the next addition to my Geneve and to YAPP, if anyone knows, please let me know!

The other new hardware was shown by Klaus Wendel and the TI Club Leipzig and consisted of a video digitizer, connected to the 9938. They had built their own video card, that allowed parallel use of the 9918 or 9938 by replacing the video interrupt by an external one. to this card a RGB splitter was connected, that also was connected to a video camera to get a signal. The digitizing speed was virtually real time, and the results astonishing. They provided also the possibility to do a digitized image of yourself. Soon also some people started using YAPP to modify pictures, thus some people eagerly kept the disk with their digitized image for themselves to avoid such tampering...

The same guys also showed modified routines for disk access, that allowed by direct controller

programming and avoiding the TI file system overhead blazing speed, comparable with a good hard disk, when using it on a TI with disk drive. Also they told, but not demonstrated, the use of an AT-Bus hard disk on the TI: A hard disk with this kind of interface could be connected with very few work (about 20DM cost in parts) to the TI, just by providing a port to write to/read from. The only problem would be to write a new DSR, which they had no time for. This kind of hard disk may be difficult to implement on the Geneve, since it does not use the DSR's, but surely would be an alternative for the TI, since an AT-Bus hard disk is priced about the same as one with a ST506 port.

On the other side, there was some software newly demonstrated or now shown in finished versions:

Henrik Wedekind demonstrated a finished version of his platine CAD program, that surprises especially by it's very user-friendly handling, similar to the Macintosh computer by Apple. The program provides nearly all standard paint functions and works neatly on the TI with 80 column device (I did not check it on the Geneve).

Peter Muys from Belgium showed his program "CRASH" which allows to do chart analysis on the Geneve. The program is written completely in Assembler and runs from MDOS mode. I'm not a speciallist concerning shares, but I was told the built in functions are equal or even beat PC programs, that cost several kilomarks.

Also shown (sorry, but my sieve-like brain forgot the name of the author) was a XBasic compiler for the TI. At the moment, it compiles everything except SUB's, DEF's and arrays, the only disadvantage is compiling time, which is still counted in hours, since the program is written in basic and not yet able to compile itself, otherwise really neat.

Asgard Software, which was represented by Jim Fetzner, sold their software programs. Unfortunately a large packet with software did not arrive in time, thus quite a bit had to be ordered to be send by mail. New were TI/Geneve adaptions of some Infocom Games never released for the TI. I've seen those adaptions yet some time ago, but Asgard has purchased not just the rights to distribute it, but also the original packages, which means you get all those great goodies with your software. I guess everyone knows these games, and does not need any explanation, so just a short rating: all the following software is from Infocom; buy, enjoy: SUSPECT, HOLLYWOOD HIJINX, STATIONFALL, LEATHER GODESSES OF PHOBOS, LURKING HORROR PLUNDERED HEARTS and possibly many more!

During the last year, I had not so much time, so I just had to show the newest version of YAPP, with the included hardcopy and the ability to load instances. Again I found that some people did not know about even elementary options of the program, since they did not read the DOC's completely (additionally, but not in the DOC's: enter just the number, when doing a disk's directory). Also the logic functions, that allow for example the mixing of pictures and the use of colored slides are often underestimated.

Winfried Winkler showed his new modified XBasic. This program is not, as many other "new XBasics", just filled with additional routines, be it for graphics or other purposes, but also extensively rewritten. Especially part of the code has been removed from slow GROM and the slowness of interpretation in GPL to fast Assembler. This leads to the only drawback: The module consists of not just 2 ROM Banks at >6000 like the standard XBasic, but 3, thus just running only on the Mechatronic GRAM card, but not on the Geneve or Gram-Kracker or other simulators with just two banks. It is the first module I know of, that makes use of the third bank.

The last program to note was again shown by me, though it was written by Martin Trabi of Austria, but none of the Austrian user groups came to the meeting: A 80 column version of the UCSD p-System for TI and Geneve.

This caused quite some interest in those users tired of slowness and disadvantages of Basic. There also is a so-called Turbo-Pascal, but this contains just a small subset of the Jensen/Wirth standard, excluding all higher data structures as pointers and records, also the syntax is quite different to the standard.

Those, who have worked with the p-system Pascal know, that it resides on a p-box card, which contains some ROMs and GROMs. Martin has done a terrific job in disassembling and rewriting these routines, modifying them to use some module simulator like Gram-Kracker or the Geneve. Doing this work, also some errors were corrected, and a yet non-supported command included. This would get you an (perhaps a bit faster) equivalent of the p-code card, but he did not stop with this: also included is access to the 9938: This will let you use the system in TRUE 80 columns, which is a more than major improvement. The 64k extension RAM of the card can be used (if installed) as an RAMdisk, that is large enough to contain the compiler and some swapping space. Since the compiler does a large amount of swapping itself, this greatly decreases compiling time and thus turnaround times.

Also modified and improved was the SYSTEM.LIBRARY, now not only including rewritten support routines, also including a new one for the mouse, but also an equivalent of the well known TURTLEGRAPHICS package, for example available on the Apple. This allows for example to compile programs, original written for the Apple on the TI/Geneve with just minor modification (for example because the 9938 allows more colors and better resolution than the Apple).

I must admit that this is to my eyes the BEST program I've seen, since I got my Geneve (before this time I had a p-code card), since the reason for me to upgrade was to run Pascal in 80 column mode.

For those of you, who never used the p-system, you can compare the possibilities to those of Turbo-Pascal 4.0 or 5.0, just some units access for graphics etc is done different, since e.g. the PC has no sprites. UCSD Pascal does not include the object-oriented stuff of newer Turbo-Pascal (But I don't think this is a major disadvantage, since this results more or less just in different coding conventions) and functions as parameters, but (and not yet included in Turbo-Pascal) the ability to run procedures parallel, including so called "semaphores" for synchronization.

This excellent program is not yet published, but as I heared, Asgard Software is trying to publish it, since they already obtained the rights for the p-code card and the accompanying programs, hopefully the release will be soon!

That's all for this time, next year the meeting will be held again in Wiesbaden, hopefully attracting again more people. ■

---

## KATALOG AV DISK

(från PB 85-1.17)
*av Jan Alexandersson*

```
100 CALL CLEAR
110 T$="DIS/FIXDIS/VARINT/FI
XINT/VARPROGRAM"
120 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL
130 INPUT #1:A$,J,J,K
140 PRINT A$;"  LED=";STR$(K
);"  ANV=";STR$(J-K):"filnam
n    sekt typ        p": :
150 FOR I=1 TO 127
160 INPUT #1:A$,A,J,K
170 IF LEN(A$)=0 THEN 230
180 PRINT A$;TAB(14-LEN(STR$
(J)));J;TAB(17);SEG$(T$,ABS(
A)*7-6,7);SEG$(" "&STR$(K),1
-(K>99),-3*(ABS(A)<>5));TAB(
28);CHR$(32-10*(A<0))
190 CALL KEY(3,T,S)
200 IF S=0 THEN 220
210 GOSUB 260
220 NEXT I
230 CLOSE #1
240 GOSUB 260
250 END
260 CALL KEY(3,T,S)
270 IF S<1 THEN 260
280 RETURN
```

■

BASIC

# An anniversary column

## Ten interesting years and counting

By REGENA

It has been exactly ten years since we got our first TI99/4 for Christmas of 1980. I have often said that Christmas gift changed our lives more than any other material thing we've received. My husband and I bought our first TI for about $600, so it was slightly less than the $1000 offer of earlier in the year which included the monitor. We hooked it up to our television and were excited about the color graphics and music. We also liked the command module concept for ease of use with children.

The first program I wrote on the TI was "Auld Lang Syne," designed to be RUN right at midnight with 1980 turning into 1981. That program has been updated periodically (and published). In 1981, I learned how a computer addict can get working on a program and end up doing "just one more thing" all through the night. I submitted a few programs to the 99'er Magazine and found out I could actually make money playing with my computer. I decided on the name "Regena" (my actual middle name) and worked with 99'er and met the International Users Group.

There were few TI owners in the early days, so users groups were formed and people shared information freely. We all learned a few little fun things, such as instead of CALL CLEAR, a different effect would be:

```
CALL HCHAR(1,1,32,24*32)
or
CALL VCHAR(1,1,32,24*32)
```

Quite often we would want to PRINT something without the message scrolling up the screen. There are several ways to accomplish this. One way is to change the screen to black, print the letters in black, then change the screen back to cyan.

```
100 REM SAMPLE1
110 CALL CLEAR
120 CALL SCREEN(2)
130 PRINT "HELLO": : : : :
140 FOR J=1 TO 10
150 PRINT J
160 NEXT J
170 CALL SCREEN(8)
180 GOTO 180
190 END
```

**Another way is to use CALL HCHAR in** a subroutine. Before you call the subroutine, specify the row R and the column C where you want the message to start. Put the message in the string M$, then call the subroutine. In this sample, the subroutine starts in Line 500.

```
100 REM SAMPLE2
110 CALL CLEAR
120 M$="THIS IS A MESSAGE."
130 R=7
140 C=4
150 GOSUB 500
160 M$="HERE IS ANOTHER."
170 R=15
180 C=6
190 GOSUB 500
200 GOTO 600
490 REM
500 FOR J=1 TO LEN(M$)
510 CALL HCHAR(R,C+J-1,ASC(S
EG$(M$,J,1)))
520 NEXT J
530 RETURN
600 END
```

I think a lot of early home computer users were excited about the graphics and non-mathematical things a computer could do. Here is a simple effect using blocks of color starting at the center of the screen.

```
100 REM SAMPLE3
110 CALL CLEAR
120 CALL COLOR(9,16,16)
130 C=1
140 FOR R=1 TO 12
150 CALL HCHAR(13-R,17-C,96,
2*C-1)
160 CALL HCHAR(11+R,17-C,96,
2*C-1)
170 CALL VCHAR(13-R,17-C,96,
2*R-1)
180 CALL VCHAR(13-R,16+C,96,
2*R-1)
190 C=C+1
200 NEXT R
210 GOTO 210
220 END
```

Stationary pictures on the TI screen were beautiful, but for games, many of us wanted moving graphics. Since the printing and CALL HCHAR and CALL VCHAR were relatively slow on the TI, programmers would come up with various ideas to try to speed up the illusion of movement. Here is one idea. A diagonal line is defined in characters, one in each of the character sets from 9 to 14. Lines 170-220 place the characters on the screen, but you don't seem them because the colors of those sets have been changed to transparent. When you press a key, lines 260-290 make the characters visible by changing the character set color, then making it invisible again.

```
100 REM SAMPLE4
110 CALL CLEAR
120 C$="010204081020408"
130 FOR S=9 TO 14
140 CALL CHAR(S*8+24,C$)
150 CALL COLOR(S,1,1)
160 NEXT S
170 CALL HCHAR(15,8,96)
180 CALL HCHAR(14,9,104)
190 CALL HCHAR(13,10,112)
200 CALL HCHAR(12,11,120)
210 CALL HCHAR(11,12,128)
220 CALL HCHAR(10,13,136)
230 PRINT "PRESS ANY KEY."
240 CALL KEY(0,K,S)
250 IF S<1 THEN 240
260 FOR S=9 TO 14
270 CALL COLOR(S,7,1)
280 CALL COLOR(S,1,1)
290 NEXT S
300 GOTO 240
310 END
```

One of my favorite uses of the TI is with sounds and music. With one statement you can actually play a chord. In some of the other computers (such as Commodore) it would take several statements and tables of numbers to make one little note. Programmers have been able to make a variety of sounds on the TI, too — all sorts of effects. One thing you may enjoy trying is using FOR-NEXT loops and varying the frequencies. Use a negative number for the duration so the sound statement is executed as soon as it is reached, rather than waiting

(See Page 9)



## REGENA—

(Continued from Page 8)

for the previous duration to finish. You can also try varying the volume in a FOR-NEXT loop. Then try combining frequencies with the "noise" numbers. Here is an example of sound varying in a loop.

```
100 REM SAMPLE5
110 FOR S=400 TO 200 STEP -1
0
120 CALL SOUND(-50,S,2)
130 NEXT S
140 FOR S=200 TO 400 STEP 10
150 CALL SOUND(-50,S,2)
160 NEXT S
170 GOTO 110
180 END
```

One of the main functions I have used in the TI is the random number feature — for example in choosing random numbers for educational quizzes. The following sample program illustrates random numbers in a graphic demonstration. First R(X) is defined in line 110 as a random number from 1 to X. Now everywhere in the program that R() appears, it is really a random number. Line 130 chooses random colors for the color set. Line 140 chooses a random frequency for a tone. Lines 150 and 160 choose random rows, columns, character numbers, and repetitions to place some graphics on the screen. Line 170 ends the loop so all the color sets are used, and line 180 starts the program over. Use Fctn-4 to break the program.

```
100 REM SAMPLE6
110 DEF R(X)=INT(X*RND+1)
120 FOR S=1 TO 16
130 CALL COLOR(S,R(16),R(16)
)
140 CALL SOUND(50,R(1000)+15
0,2)
150 CALL HCHAR(R(24),R(32),R
(8)+S*8+23,R(50))
160 CALL VCHAR(R(24),R(32),R
(8)+S*8+23,R(50))
170 NEXT S
180 GOTO 120
190 END
```

I looked at one of the early programs I wrote and noticed how unorganized it seemed. Through the years I have learned a few tricks and how to use subroutines and how to program more efficiently. I gradually added more and more computers as I was writing programs — I stayed with BASIC programming but with a variety of home computers as they came out. In 1982 I started writing for Compute! and my hobby turned into a business where I could really make money. I wrote six books with Compute! Books and wrote monthly articles in three of their magazines. I also wrote for several other magazines.

About the same time Texas Instruments stopped producing more TIs, many of the magazines disappeared — along with several brands of computers. Many people who were in the home computer business had quite a struggle financially. I was lucky because I have a husband who supports the family, and my income as "extra." I ignored the up-and-down computer world for awhile — and had a baby in 1985. Once again family life took over and computers were on the back burner.

In January of 1987, I started my monthly column with MICROpendium, and that has kept me in touch with the TI world.

Throughout my ten years with my trusty TI, I have been able to travel to users groups throughout the world and made some wonderful friends. I'm looking forward to going to Fest-West '91 hosted this year by the Orange County and Pomona Valley (California) user groups Feb. 16-17. The main reason I go is to revisit long-time TI friends, but I strongly encourage any TI users to go to conventions when they can to see that the TI is still going strong. New software and hardware is still being produced!

Besides the three books containing over a hundred programs specifically for the TI, I have written over 130 articles with programs just for the TI. Our "orphaned" computer is still one of the best available home computers and can do a myriad of things. I still have my original TI99/4 but use a TI99/4A for everyday work. I'm guessing this tenth anniversary is definitely not the end of an era.