Classic 99

The Official Newsletter of the Hoosier Users Group

People Helping People

September - October 1999 | The HUGger's Newsletter | Volume 18 Number 5

Officer's Corner

By Dan H. Eicher

Remember the next meeting is November 21. I have a bunch of things to show and demonstrate from the TI-TREF. I will also be demonstrating how to use PC99's sector read/write utilities to create a fully functional TI disk from a PC99 image. I will also be showing a PAL Video Modulator and a design specification for E.T.'s Adventureon Land by Hank Mishkoff. Hank worked for a company called Looking Glass software. Looking Glass was developing this module under contract for TI.

Don't forget to check out Roger Price's TI support page! It is very well done and informative, you can find it at **http://home.switchboard.com/RogersTiPage.**

Delphi has announced that they will be dropping their dial-in access. They will still support their text based mode, but ONLY through telnet. This means people using their TI's or Geneve's will not be able to use this service anymore as their link to the internet. So, that leaves a number of people scrambling to find internet providers they will let them have 'shell accounts'. Basically a shell account allows people to login to a UNIX computer using a terminal emulator. I have been looking for shell providers around Indianapolis, and I have found one and only ONE small provider that will do this. Two of the larger area providers USED TO, and even then it was on "special setup". Jeff White left an email message that stated "Looks like to access in the Internet in 2000 you will have to have a PC running TCP/IP".

I plan on signing up for this new provider "Blue Mountain" that will give me a shell account and see how it goes. I will also be re-reading Lew King's article on cruising the web on a 4A with a UNIX shell account. Lew's home page can be found at **http://ramsi.rasd.k12.pa.us/~kingt/index.html**

The 17th annual Chicago Fair is set for November 6. For additional information, please email Ernie Pergrem (epergrem@net56.net)

I purchased a replacement monitor for my Magnavox that died. It is analog TV RGB compatible. These are availible for sixty dollars from Tuner Service of Rochester, 485 Spencer Port Rd, Rochester, NY 14606, 716/429/6880, ask for Greg Weber. Hopefully I will have it hooked up to my SNUG system soon! Speaking of my SNUG system, I am very much in-debt to Tony Knerr for making a cable up for me that will allow me to hook a set of TI joysticks to the 25 pin connector on the SGCPU.

Once again, this issue is later than I would have liked. I've started a new job and things have been totally hectic since I got back from the Tref, I hope the size will compensate somewhat. Also, I am still looking for someone to take over as editor....

See You,
Dan

Table of Contents

A Blast From The Past

My name is John Phillips and I'm the author of Moonmine, Hopper, Beyond Parsec, Munchman II, Star Trap, D_Station I, D_Station II, Word Radar, Star Gazer I, II, and III, 4A Flyer, Strike Three!, The Great Word Race, and many others.

It's been a while since I've thought about the TI_99/4A, but if there are people who have questions, I'd be happy to answer them.

John Phillips
VP Information Systems, Chief Information Officer
Processors Unlimited
Office: (972) 980_7825 x286

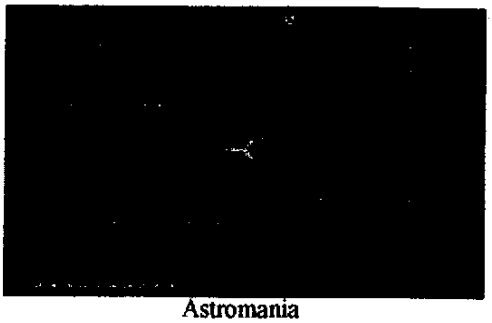
[Editors note: There is now a whole page on Rich's site **http://99er.interspeed.net/jphillips.html** dedicated to John with additional information!]

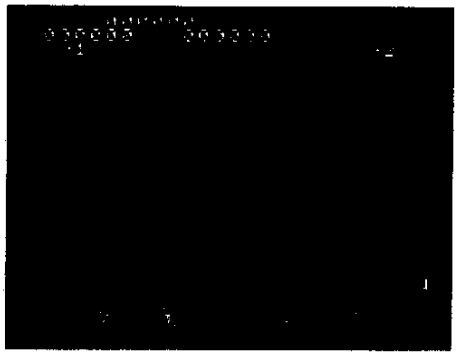
Moonbeam Software

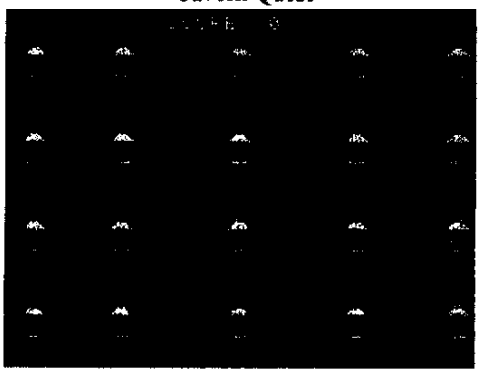
This is to publicly announce that the following software titles produced by my long defunct software company, Moonbeam Software, shall henceforth be considered as being public domain:
Death Drones, Moonvasion, Garbage Belly, Astromania, Moonbeam Express, Cavern Quest, Strike Force 99, Robot Runner, Zero Zone.

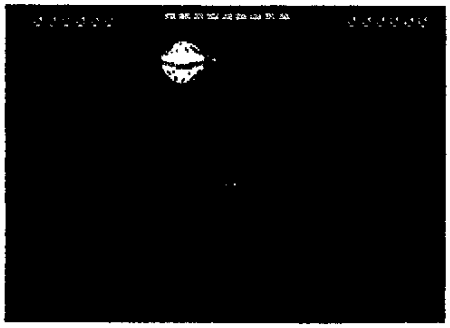
Although, by today's standards, they would be considered lame if not downright pathetic, particularly since they all were either Basic or Extended Basic programs, they were fun to code and some of them were actually fun to play. Joe Macchiarulo (my partner) and I certainly gave the TI99/4A a good workout, pushing it to the extreme limits of what Extended Basic was capable of doing.

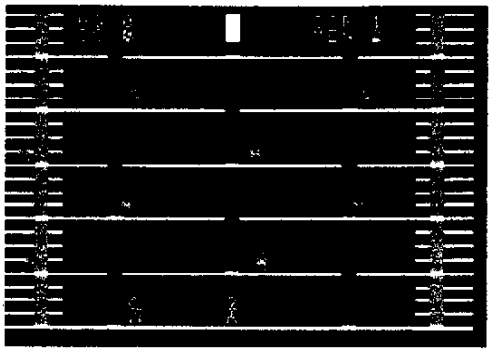
In case you are interested, I will not be making the software available myself. I no longer have copies of any of the original program disks, nor do I have a completely functioning TI99/4A system. Mike Wright, who has copies of all nine titles, has expressed his tentative intention to make the programs available as PC99 files.
Mr Moon
http://www.mrmoon.communitech.net/ti994a/


Astromania


Cavern Quest


Garbage Belly


Strike Force 99


Robot Run

NEWS RELEASE For The Geneve

DDI SOFTWARE announces the release of its version of MYBASIC.(February 1997). Version 4.0 of MYBASIC has 99% of known bugs of Version 3.0 fixed.

To receive your personally registered FREE copy of MYBASIC 4.0, send $15.00 registration fee to the address below. For the registration fee you will receive in addition to the FREE copy of MYBASIC 4.0 the following:

1.Hardcopy of 88 replacement pages, which include commands not included in original manual, new commands, deleted commands and changed commands.
2. Five new Appendixes
   - Call Key ASCII chart for Modes 0_3
   - Assembly Support Information
   - Color Palette and Hexdecimal charts
   - RS232 DSR info
   - Disklayout_Floppy and MFM Harddrive
3. 17 DDI SOFTWARE MYBASIC Utility programs

It is the intent of DDI SOFTWARE to support registered users.

Address for United States
JAMES UZZELL
juzzell@surfsouth.com
704 S. DOGWOOD DR.
NASHVILLE, GA 31639_2027

[Editor's note: Jim has a number of other programs for sale, all written in MyBASIC. MyBASIC is (in my estimation) as fast a GPL on stock TI.]

Oliver Arnold Software Update

Hello TI friends,
I have just added new Software downloads on my homepage. Please have a look on the TI Club_ERRORFREE homepage
The address is: www.planet_interkom.de/oliver.arnold

There you will find the newest version of the MDC with documentation + the SCSI Editor with a backup function. I demonstrated this new software on the International TI Faire in Stuttgart.

These two programs need 80 column cards!

Also the C_ROM v1.1 plus documentation is now available to download. I think a useful development tool for C programs. This C library needs the HSGPL card or the GRAM Card.

Now only 80 columns systems are supported. But if there are some interests, I will modify it for the original TI systems. Thanks to Frederik Kaal with his great Linker program it is now possible to develop large C program files in a very easy way. Please contact him if you need such a linker.

TI_TREF 1999 Review

By Dan H. Eicher

This years Tref was held in Friesburg, Germany. I went over with Mike Wright. Upon arriving, Michael Becker hosted us, he spent a lot of time with us. We are very appreciative of his family's sacrifice of time. After traveling for about twenty hours, we arrived for our first afternoon at the fair.

Some of the Demos:
Wolfgang Bertsch: GIF 99 Viewer, XB-Compiler, Video Disk Controller Demo. Wolfgang has been working on his Extended Basic compiler for many years. This is a true compiler. One interesting feature is the compiler itself is written in Extended Basic. There are many things it won't do, but if you know its limitations you can write some really quick code, really quickly.

Michael Becker and Harald Glaab: SNUG peripherals, the new Speech and Voice Memory Controller (SPVMC) and the new DSR Loader. Harald's new DSR loader is outstanding! It reads the DSR file, tells you if it is compatible with your card, and whether the DSR you are loading is more recent than the version you have loaded!

Oliver Arnold and Martin Zeddies: C-ROM, MDC 1.5, 16-bit Logic Analyzer. I discussed C-ROM with Oliver at some length and brought back a pretty good understanding of how this software works. The software he has written for the 16-bit Logic Analyzer (SPY) is TOP NOTCH. It runs in bitmapped mode and is by far the best use of bitmapped graphics that I have EVER seen on the TI.

Roger Muys: His custom written stock market analysis program for the Geneve. Roger is a very lucky man! He has two sons, who are very good programmers, that have written software for him! Roger was very kind, and hand delivered me a copy of Peter's

continued on next page

EDIT for the Geneve! Peter now runs a firm in Belgium writing internet applications. Peter took the time to dig up the source code for the TI community. Since he couldn't get a copy of Genprog, he wrote his own assembler, which he also included!

Berry Harmsen: Auction of TI Hardware and Software. This was probably one of the most fun segments of the fair, although my not having a command of the German language was a challenge. One thing I didn't know was that both Hiener Martin and Michael Weigand created custom printed circuit boards so that one could build their own ROM/GROM modules. I purchased a built Hiener Martin module with E/A and a debugger built in for 16 marks!

Mike and I were both surprised to find out the following:
TI CALC - This is the spread sheet program that TI should have produced, and did. The story that I was told about TI CALC was that the software was actually produced by TI. Production modules and documentation were all completed. It was put out for sale at two TI stores in Europe. This was after TI announced it was exiting the home computer market. Apparently TI was not suppose to release any new hardware or software after black Friday. This software was pulled after two weeks, when someone realized what had happened. Fortunately for us, a couple of copies had been purchased. Michael Becker ended up with one of these copies. He allowed us to dump the module, and take the manual to be turned into a PDF. Both the manual and the documentation will be made available by CaDD for PC99 users. It bears mentioning that TI CALC was the ONLY module ever produced by TI that had FOUR ROM banks. As a comparison TI Extended basic had two. The SNUG HSGPL card allows this bank switching scheme. Mike and Mark have already modified PC99 to handle this new banking scheme also!

APESoft - APESoft produced many pieces of software for the TI, but they also produced some books for the TI. These books on Assembly (two books) and Extended Basic (one book) were co-released with TI. They bare the official TI Logo you find on books produced in the U.S.A. by the TI learning center.

TI-Review - These are excellent magazines dedicated to the TI-99/4A. They remind me of Home Computer Magazine, during it's best days. Michael Becker gave us a full set of them.

HM (Debugger card) - Built upon a Hiener Martin cartridge.

More Basic - This manual documents an internal TI product that was never released to the consumer market. It appears this module was released between the introduction of the 99/4 and the introduction of Extended Basic. This module either came with or without an additional 2K of RAM. Some of the commands added with this modules are: Block Moves, Program Chaining, Execute CPU code, CRU Access, Packing Utilities, Logical Routines and more. Many of these commands look a lot like the commands provided with the Corcomp Disk Controller in their Tool Shed utilities. Fortunately, this documentation came with the source code listing (GPL) of the modules contents. It may be possible to add these extensions to console basic on a GRAMULATOR or with PC99, as these devices use 8K GRAM's instead of 6K GROMS.

TIPS and Tricks for the 99/4A by DataBecker. Data Becker has produced many very good computer books in the U.S.A. for the Commodore 64, later the Amiga and currently the IBM PC. It appears this book was published during the TI bail-out. Probably because of this, the book was never translated to English and published in the U.S.A.

Another amazing fact was the EXTENT that TI went through to sell their products in other countries! I brought back a warranty card that is printed in ELEVEN different languages!!! I brought back a number of internationalized modules and manuals for everyone to take a look at. I am very sure no other micro computer produced in the early to mid-eighties had so much foreign exposure!

continued on next page

HOOSIER USERS GROUP OFFICERS

Area Code (317)

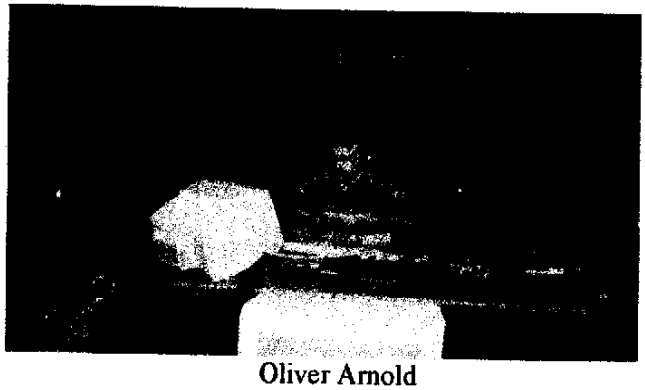
President	Dan H. Eicher	862-1860
email: eicher@delphi.com		
Vice-President	Bryant C. Pedigo	255-7381
email: bpedigo@midlink.com		
Secretary/Treasurer	Greg Larson	783-4575
email: greg.larson@icsbbs.org		

One of the major impressions that Mike and I both walked away with was how far the European users had diverged from the American users! Mike stated it best: "This isn't a TI fair, this is a little black box with lights and switches fair!" Only on a rare occasion did you see a TI logo any where! One of the reason such divergence happened was because of the difficultics European users faced when buying from most American hardware and software providers. There was almost no Myarc or Corcomp equipment to be found. HFDC's were a VERY rare commodity, although a number of Geneve's did show up at the Tref. Because of this, mass storage had been very rare. I saw this in the twenty five to thirty systems that were set up. ALL of them had Zip Drives attached, in fact, most had more than one! I haven't seen that many Zip Drives in one place since I went to CompUSA!

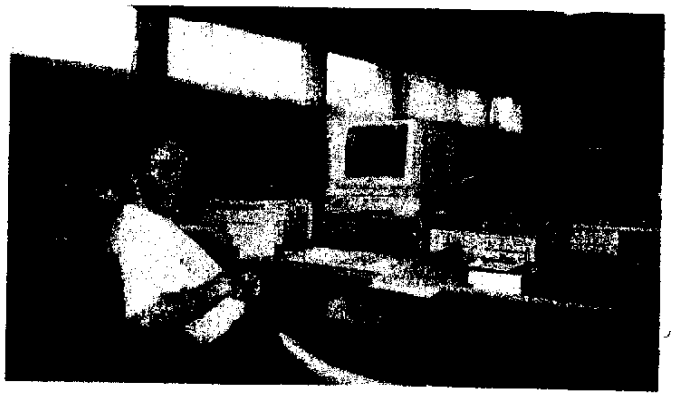
Michael was very busy all three days of the fair, upgrading SCSI cards with his new fixes. The SCSI card once again operates at the speed of a ramdisk. We can thank Jeff White for coming up with the original design of a GAL on the SCSI controller that allowed the SCSI card to act as if it had a DMA controller onboard!

Last but not least, I brought back a number of SNUG cards for U.S. users.
I brought back a number of pictures:

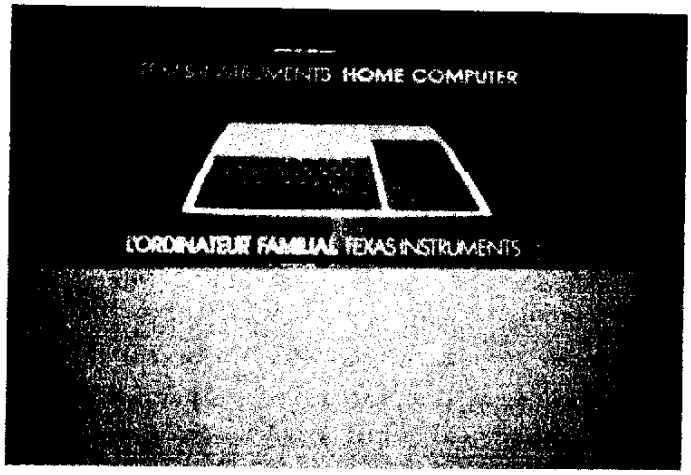
Michael Becker, Richard Twyning, Trevor Steven, Mike Wright



Oliver Arnold



Wolfgang Bertsch



One of the internal Boxes for the 99/4A

Michael Becker, Unknown, Harald Glaab

Roger Muys

Cracking Millers Graphic's Explorer and what I learned from it

By Thierry Nouspikel

Millers Graphics Explorer is one of the most useful programs ever written for the TI_99/4A. Basically, it's a TI_99 emulator within the TI_99! It allows you to executes assembly language step by step, while viewing the registers, editing memory, manipulating the CRU, etc. You can set brake points and use it as a debugger, or just explore your system (hence the name Explorer).

I purchased it from Triton many years ago, and it came on a heavily protected floppy that I could not duplicate even with a track_by_track copy program. That was fairly annoying to me, since I like to have backups of all programs I own. So, after a while I set out to hack the protection, which took me weeks but turned out to be worthwhile when I bought an Horizon ramdisk: I could install my hacked copy in the ramdisk from where it loads in a couple of seconds. But most importantly, the software protection tricks used by Millers Graphics taught me a lot on the TI_99/4A. That's the purpose of the present article: to share this information with you.

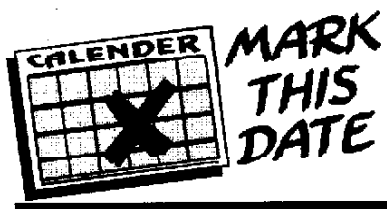
Explorer comes with several loaders: one for the Editor/Assembler cartridge, one for Extended Basic, and their equivalents to be used with the Myarc disk controller. Let's look at the Extended Basic version, since it's the most interesting one.

To load Explorer from XBasic, you just type CALL LOAD("DSK1.XBEXP") and that's it: no need for a CALL LINK. The XBEXP file contains a heavily protected loader that loads Explorer from DSK1. It is a DF80 file, containing assembly language in uncompressed tagged_object format.

A quick inspection of this file with a sector editor showed that it loads at addresses >C000_C2EC (an area overwritten by Explorer itself), at >6000_60FE and >7000_70F0 (two area used by the XBasic catrridge ROM), then at >C02A_c8AC (overwriting itself), etc. The file ends with an auto_start tag pointing at address >C028.

All this is quite puzzling. The key is that only the first part of the file is effectively loaded, at >C000_C270! All the remainder is nothing but a decoy that may cause you to spend

continued on next page



Tentative HOOSIER USERS GROUP Meeting Schedule

November 21
December 12 - Holiday Dinner 2nd Sunday

Mark your calendars!!

Hoosier User Group meeting place TO BE ANNOUNCED prior to meeting. Meetings start at 2:00pm.

HUG supports the following computers:

TI 99/4A and Myarc 9640 Geneve, TI CC-40 and TI-74 BasicCalc.

HUGGER S&T BBS

Hoosier Users Group, Indianapolis, IN
300/1200/2400/4800/9600 8N1
317-782-9942

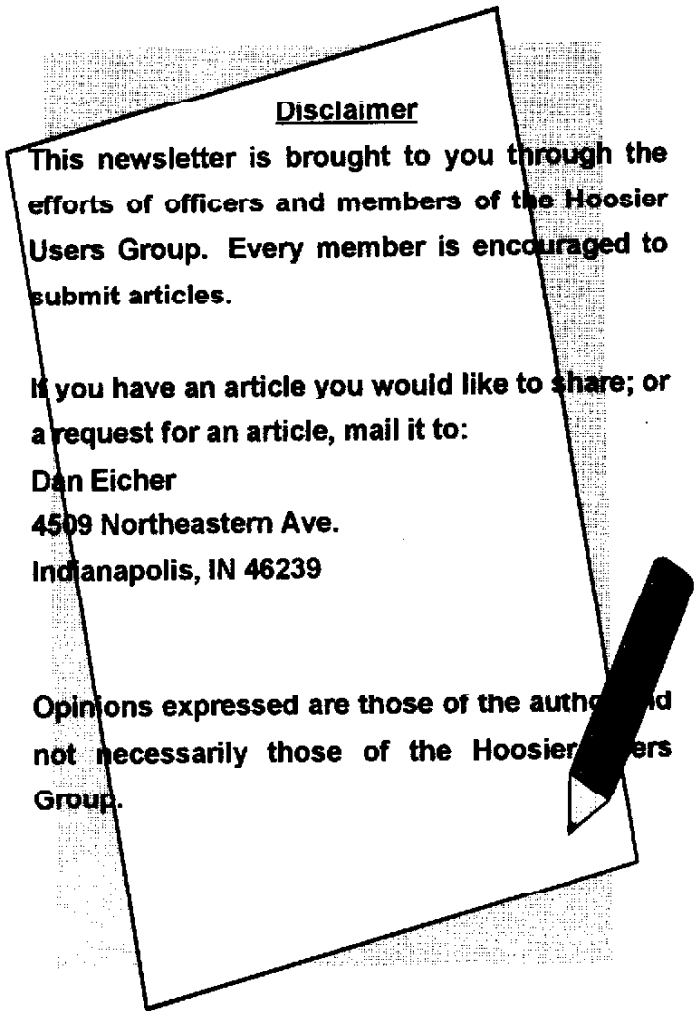
Sysop: William M. Lucid
email: lucid@indy.net

endless hours trying to make sense out of totally absurd code (I didn't). The way XBEXP causes its own loading to abort is by loading >C168 at address >83C4. This address in known as the "interrupt routine hook": whenever a VDP interrupt occurs (60 times per seconds on US consoles) any routine whose address is placed at >83C4 will be branched at. Thus, soon after XBasic has loaded this address, XBEXP will be called.

Note that if you scan the file for a >83C4 address, you won't find one (actually you will in the last record of the file, but it's a decoy as this record is never reached). This is because the program addresses the hook as >81C4. Now the only CPU RAM available in the TI_99/4A is the scratch_pad at addresses >8300_83FF. But... the decoding circuitry in the console is incomplete, so the whole area >8000_83FF maps to the scratch_pad. In other words, >80C4, >81C4, and >82C4 are equivalent to >83C4. That was a new one on me! Ok, so now let's have a look at what XBEXP is doing once it takes over:

```
AORG >C168
* Modify the interrupt hook
AC168  LWPI >81AC
       LI  12,AC1AA
* Change return from CALL LOAD
       LWPI >8180
       LI  6,>176C
       LWPI AC250
       MOV  @>200E,8
       LI  0,AC1E6
       MOV  0,@>200E
* Get pointer from >834E
       LI  0,>41A7
       INC  0
       SLA  0,1
       JGT  AC168
       DECT 0
       MOV  *0,7
       AI  7,>00F9
* Erase all the above
       LI  0,AC168
       LI  1,AC1A4
       CLR  2
AC1A4  MOV  2,*0+
       C  0,1
       JNE  AC1A4
* Next interrupts enter here
* Check it eof record was taken
AC1AA  LWPI >7FFA
       CB  11,@AC1CA
       JEQ  AC1CC
* No: delay loop
       LI  11,>04CF
AC1B8  DEC  11
       JNE  AC1B8
* Place eof record in buffer
       LI  11,>3A20
       LI  12,>2020
       LI  13,>2000
       JMP  AC1DE
AC1CA  DATA >0A0A
* EOF record was processed
```

continued on next page



Disclaimer

This newsletter is brought to you through the efforts of officers and members of the Hoosier Users Group. Every member is encouraged to submit articles.

If you have an article you would like to share; or a request for an article, mail it to:

Dan Eicher
4509 Northeastern Ave.
Indianapolis, IN 46239

Opinions expressed are those of the author and not necessarily those of the Hoosier Users Group.

```
AC1CC LI  11,>0AF0
      LI  12,>F0F0
      LI  13,>F000
* Remove interrupt hook
AC1D8 LWPI >81AC
      CLR 12
* Return to XBasic loader
AC1DE LWPI >83E0
      CLR 8
      B   *11
* CALL LOAD returns here !
* Cover our tracks
AC1E6 LWPI AC250
      MOV 8,@>200E
      LI  0,>A000
      MOV 0,@>838C
      MOV @>838E,1
      AI  1,>0402
      MOVB 0,*1
      SWPB 0
      MOVB 0,*1
* Get decrypting key from sector buffer
      MOV @>80FE,1
      SRC 7,8
      MOVB 7,*1
      SRC 7,8
      MOVB 7,*1
      LI  2,>7DFF
      SRL 1,1
      A   2,1
      SLA 1,1
      LI  0,AC008
      LI  2,>0006
AC222 MOVB *1,*0+
      DEC 2
      JNE AC222
* Decrypt encoded area
      LI  0,AC008
      MOV *0+,1
      MOV *0+,2
      MOV *0+,@AC242
      MOV 2,3
      S   1,3
      CI  3,>0026
      JEQ AC242
      B   @AC13A
AC242 SOCB *14+,*3+
* will be SWPB *R1+
      C   1,2
      JNE AC242
      MOV 2,@AC242
      B   @AC034
* Workspace
AC250 BSS 32
* Cause loading to abort
      AORG >81C2
      DATA >C168
* Decoy program starts here
      AORG >C270
      etc.
```

First of all, XBEXP changes the value of the ISR hook in >83C4. Note the way this is done: by setting the workspace as >81AC and then putting a value in R12, i.e. in >81AC. And we saw above that >81C4 is equivalent to >83C4.

The same LWPI trick is used to change a word at >838C. Why? Now that's a really devilish (and very usefull) programming trick! Let's see how it works. The first thing you should know is that, when the XBasic interpreter branches to a CALL subprogram, it saves the return address (in GROM) and the GROM base on the subroutine stack. This stack is located in the area >8380_839F. As you may already have guessed, the return address for CALL LOAD happens to be saved at >838C. By altering the contents of this address, XBEXP causes CALL LOAD to branch at address >176C in the console GROMs.

So what's in address >176C? We would expect it to contain GPL (GPL is the language the XBasic interpreter is written in). But it's not the case: this area contains data used by the keyboard scanning routines. More precisely, the ASCII codes for the Fctn keys: >0F, >27, etc.

However, the GPL interpreter does not know this, and will try to interpret this data as if it were GPL. Now, >0F is the opcode for XML (execute machine language) an instruction that causes

continued on next page

GPL to call an assembly subroutine. The next byte tells the interpreter where to find the address of this subroutine. XML >27 means that the address is in entry 7 of table 2. Table 2 is located at address >2000 in cpu memory and entry number 7 is therefore at >200E.

Now the following instructions become clear: XBEXP saves the content of >200E and replaces it with an address in XBEXP (>C1E6). Once the CALL LOAD is completed, the GPL interpreted will "erroneously" return to >176C and execute an XML >27, which re_enters XBEXP.

This is a very nice trick to avoid a CALL LINK, while still letting CALL LOAD complete normally, close the file, etc. It's also a very useful feature to know for assembly programmers, because it allows us to call GPL routines and then to come back to assembly without the need for a special cartridge.

The next step is for XBEXP to fetch the content of the memory word >834E. Note the complicated calculations the programmer used to avoid spelling out >834E. Now, what's in that mysterious address? Well, the TI disk controller reads the disk one sector at a time and saves its contents in VDP memory. >834E is a pointer to the top of memory buffer. High level DSRs then extract the data from the buffer (in our case, 80 bytes) and copy it to the area specified by the user.

Here, XBEXP is superbly ignoring the DF80 record and goes directly to the end of the raw sector. This area normally contains garbage in DF80 files, since 3*80 is 240, which leaves 16 bytes unused. But in that very sector the last 7 bytes do not contain garbage at all: they contain key values for a self_decoding subroutine. More on this later, for the moment being XBEXP just memorizes this pointer.

Then comes a self erasing loop, that clears the part of the program that was already executed. Since the ISR hook was modified, all subsequent interrupts will enter XBEXP at address >C1AA, which is where we are moving now.

What XBEXP is doing here is to load an "end_of_file" tag in the area >8310_8315. This area is used by the XBasic loader to decode the loading instructions found in the DF80 file. Placing a semi_column in here will cause the loader to terminate loading and return to XBasic. Provided of course that is was about to read the content of this buffer!

But we don't know when the previous interrupt occured: may be the buffer was about to be written at, which will overwrite our end_of_file tag, or may be it was already read... Never mind, we'll try again at the next interrupt! And again and again, until the value is >8310 is changed from >3A (the semi_column) to >0A. This means that the loader took the bait and started decoding the tag, by substracting "0" (ascii >30) from it.

Note the waiting loop executed >04CF times: this causes a delay of 14.75 milliseconds. On North_american consoles, interrupts occur at 16.667 msec intervalls, on European consoles at 20 msec intervalls. The delay loop ensures that an interrupt is going to occur soon after XBEXP returns to the loader. At this point, >C1AA is re_entered and XBEXP gets another chance to terminate loading.

Once termination is achieved, XBEXP removes the ISR hook, and returns to the loader once and for all. Well, not quite: as we saw above, when the loader has completed its job, the XML >27 trick will cause XBEXP to be entered again at address >C1E6.

The first thing XBEXP does at this point, is to cover its tracks: it restores the proper values in >200E and >838C. It also gets the GROM base that was saved in >838E after the return address, and uses it to set the GROM address to the proper value (which happens to be >A000). This is because Explorer displays the current GROM address, and a nosy user may wonder why the GROM address is >176C...

And now, XBEXP is going to make use of the dummy values stored at the end of the disk sector. Again, note the way the programmer avoids to specify >8C02 and >8800 directly, by fetching them from >83FE via complicated calculations (>83FE is where the GPL interpreter saves the VDP write_address port: >8C02).

From the last seven bytes on the sector, six are used. The first 4 make up the boundaries of an encoded area in XBEXP (>C00E to >C034), the next two contain the decoding instruction: a simple SWPB *R1+. XBEXP patches its own decoding subroutine with this instruction. That's another devilish trick: if we just disassemble the program, we'll never make sense of the SOCB *R14+,*R3+ in the decoding loop. And unless we understood all the above, we'll never figure that the key instruction is stored on disk, at the end of the sector that contains the AORG >81C4 instruction (which is not the last sector in the file!).

continued on next page

After a last check for the size of the area to decode, XBEXP decodes it, and quickly destroys the decoding instruction. The encoded area contains two small subroutines: one that sends a command to the floppy_disk controller (FDC), the second that waits for the completion of this command.

With the help of these routines, XBEXP can manipulate the FDC at will.

Here is what it does:

```
AORG >C000
*
AC000 DATA >A000
AC002 DATA >4800
AC004 DATA >FFFF
AC006 DATA >0200
AC008 DATA 0,0,0
* Encrypted area:
* Send command to the FDC
AC00E MOV  *11+,0
      SBZ  1
      SBZ  1
      MOVB 0,@>5FF8
      SBO  3
      SRC  5,10
      B    *11
* Wait for command completion
AC01E SBZ  2
AC020 MOVB @>5FF0,0
      XOR  @AC004,0
      JLT  AC030
      SRC  0,9
      JOC  AC020
      B    *11
AC030 B    @AC11C
* end encrypted area
* Proceed here after decrypting
* Set unmaskable interrupt vector
AC034 LI   0,AC250
      MOV  0,@>FFFC
      LI   0,AC13A
      MOV  0,@>FFFE
* Turn controller card on, check it
      LI   12,>1100
      SBO  0
      SBO  11
      CLR  0
      MOVB @>4000,0
      SBZ  11
      CI   0,>AA00
      JEQ  AC084
* Patch program for use with CorComp card
      CLR  @>C004
      SBZ  7
      SBZ  8
      SBO  10
      LI   0,>0300
      SZC  0,@>C006
      LI   0,>0100
      TB   18
      JNE  AC078
      SOC  0,@>C006
AC078 LI   0,>0200
      TB   22
      JNE  AC084
      SOC  0,@>C006
* Select DSK1
AC084 SBZ  5
      SBZ  6
      SBO  4
* Set VDP register 1 (screen off)
      LI   0,>9081
      MOVB 0,@>8C02
      SWPB 0
      MOVB 0,@>8C02
* Patch program to match the controller
      SOC  @AC006,@AC0BC
      SOC  @AC006,@AC0D2
      MOV  @AC004,@AC004
      JEQ  AC0B8
      INV  @AC0BC
      INV  @AC0D2
      INV  @AC0FC
```

continued on next page

```
* Command: seek track 0
AC0B8 BL  @AC00E
AC0BC DATA >0800
    BL  @AC020
* Command: seek a given track
    LI  0,>2700
    XOR @>C004,0
    MOVB 0,@>5FFE
    BL  @AC00E
AC0D2 DATA >1800
    BL  @AC020
* Command: read sector
    LI  0,>E000
    XOR @>C004,0
    MOVB 0,@>5FFC
    LI  0,>EF00
    XOR @>C004,0
    MOVB 0,@>5FFA
AC0F0 MOV @AC000,15
    LI  1,>0400
    BL  @AC00E
AC0FC DATA >8C00
    SBO 2
* Read the sector
AC100 MOVB @>5FF6,0
    XOR @AC004,0
    MOVB 0,*15+
    MOVB @>5FF6,0
    XOR @AC004,0
    MOVB 0,*15+
    DECT 1
    JNE AC100
* Check for errors
    BL  @AC01E
AC11C SLA 0,10
    MOVB 0,0
    JNE AC0F0
* Copy drive params to new area
    MOV @AC000,@>A004
    MOV @AC002,@>A006
    MOV @AC004,@>A008
    MOV @AC006,@>A00A
* Self_erasure routine
AC13A LI  0,AC000
    LI  1,AC144
    CLR 2
AC144 MOV 2,*0+
    C   0,1
    JNE AC144
* Set next address, keep erasing
    LI  0,AC164
    LI  1,AC250
AC152 NOP
AC154 NOP
    MOV @AC164,@AC152
    MOV @AC166,@AC154
    JMP AC144
AC164 B   @>A000
* Program continues at >A000: loads remaining sectors, init
* Explorer, run it
```

Once the encrypted area has been decrypted, XBEXP branches at >C034. The first thing it does is to hook the vectors for the non_maskable LOAD* interrupt. This prevents programs like my RIP debugger from kicking in and peeking at the now decrypted routines.

Then XBEXP turns on the disk controller card and determines whether it's the original TI card, or the CorComp one (the Myarc card requires a special loader: MXBEXP). The program patches itself so as to use the proper head_step time values (the time needed for the reading head to settle) and to invert the data bus if needed (the TI card needs it, the CorComp does not). In the process, XBEXP also changes VDP register 1: it turns the screen off, VDP interrupts off and sets text mode.

And here comes the best part: XBEXP is going to use the two decrypted subroutines to talk directly to the floppy disk controller. The FDC can be accessed via 8 ports, mapping in the controller card DSR space:

>5FF0: read status register
>5FF2: read track register
>5FF4: read sector register

continued on next page

>5FF6: read data register
>5FF8: write to command register
>5FFA: write to track register
>5FFC: write to sector register
>5FFE: write to data register

The first command send is "seek track 0". This causes the reading head to move to the outside of the disk until a physical contact is closed indicating that the first track on the disk has been reached.

The second command seeks a given track, whose number is to be placed in the data register. In our case, this is track >27, i.e. 39. The command is written is such a way that the FDC will not read the track to check if it carries the right ID number (the regular command would be >1C00 instead of >1800). We'll see why in a moment.

The third and last command is "read sector". Each sector on disk is preceded with an ID field containing 4 elements:
The disk side (>00 or >01)
The sector ID (normally >00_08)
The track ID (normally matches the track number)
A code for the sector size (>01 for 256 bytes)

However, track 20 to 39 have been formatted in a special way on the Explorer disk. First of all, they contain 1024_byte sectors. This is the largest size that the FDC can handle, and it will speed_up loading.

Second, the track and sector IDs have been set to arbitrary values. Since the FDC absolutely requires the proper IDs to be placed in the track and sector registers, all attemps to read a sector on this disk will fail unless we know these arbitary values! This is also the reason why a standard "seek track" command will fail: the FDC reads the track ID field to see if the proper track was reached. That's probably what crashed the track_by_track copy program.I tried.

The 1024_byte sector is then read directly into the memory area where it must run, in our case >A000_A3FF. The controller parameters are copied to this area so that the program does not have to test the card again. Finally, a self_erasing routine is entered twice: once to erase all that's before it, once to erase all that comes after it. The two NOPs encountered at first pass are changed into a branch instruction for the second pass.

Now, the program goes on with whatever was loaded from track 39. This is basically a clone of the loading routine: it keeps accessing the FDC directly and reads 1024_byte sectors until the whole Explorer program is loaded. For each sector, arbitrary track and sector IDs must be provided. Finally, it performs some initialization and enters Explorer itself.

How did I hack it? I introduced a branch order at the point where loading is completed. This order branched to a SAVE utility that dumped the whole memory range used by Explorer onto a DF128 file. Then I wrote a loader that loads this file, performs the initialization and enters Explorer. I also had to create an extra file to store Explorer options (that are saved/retrieved from a fixed sector on disk) and wrote a small utility to edit that file from XBasic. The whole thing is not very elegant, and every once and a while Explorer crashes upon entry. But I'm satisfied with it the way it is.

The most important thing is how much I learned from this hacking:

1) That >8000_82FF is the same as >8300_83FF.
2) hat the interrupt hook is located at >83C4 (I new this already).
3) The nice trick to return to assembly from GPL (dummy return to XML >27).
4) How to cleanly abort loading within a DF80 file.
5) Where to find the raw sector data for the currently accessed file (the pointer is in >834E).
6) And last but not least: how to talk directly to the floppy disk controller!

That's why I enjoy a little bit of hacking from time to time: not only is it the ultimate intelligence fight between the programmer and the hacker, it also requires (and thereby teaches) an intimate knowledge of your machine.

PS. For additional details on the FDC, the interrupts, etc. visit my website: the TI_99/4A Tech Pages, at **http://www.nouspikel.com/ti99/titech.htm**

Update from Western Horizon Technologies

Don O'Neil

Hello everyone.... I am finally giving a long over due update. Things have been crazy for me this summer with the purchase of my new home, moving, and work weeks of 60_80 hours on the launch of a new product for the company I contract to (Cisco Systems Inc.). Needless to say, I am WAY behind in making the promised upgrades and repairs to SCSI cards, but I think I can get back on track soon.

I have 8 cards currently here in various stages of upgrade and will be getting these out soon to the owners, along with drives and other test equipment I may have of yours.

Some of you have moved since I got the stuff, so if you've moved, and haven't already notified me, please do so ASAP.

For those of you wondering if you can get your card fixed, or buy new cards, the answer is yes, and very soon!! I have parts for nearly 50 new boards, and just need to find the time to build them up. The good news is that we're looking for an assistant for my contracting business which has simply EXPLODED this past summer, so I'll be having some more free time soon! I just can't keep up this pace, I'm going to kill myself!!!

Many of you know that I am in the process of moving my web site and ftp site to a new in house server (quite literally!). My new home has provided me a direct line of sight to Downtown San Jose (we're going to put up à web cam someday, so you'll all get to see!) so I've erected a high speed Microwave link to Above.Net, one of the largest Tier 1 Internet providers. That in itself has taken a lot of my time too because as a result, a partner and myself decided to set up a full blown web hosting service with this link.

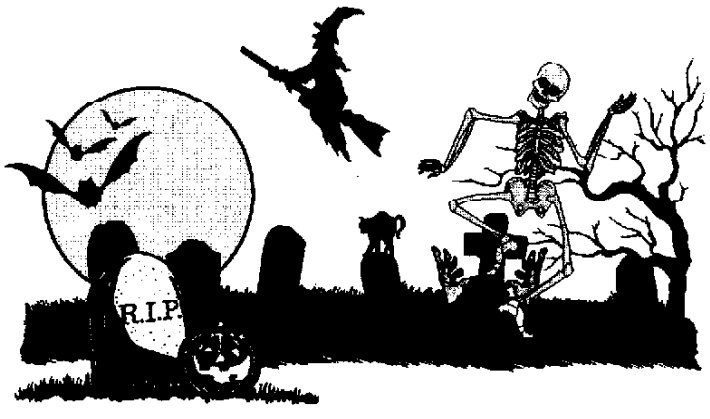
The hosting service is called AmpleSpace, (www.amplespace.com). There's not much on the web site right now, because time has been tight, but we'll be launching formally early next year. So, if you know of anyone who's looking for web hosting, we'd be happy to hook 'em up! We've managed to acquire through "fire sales" of old equipment from Pac Bell shutting down their experimental fiber based home cable & internet access a TON of equipment. We have OVER 700 GB of hard drive space, dozens of ATM Fiber cards, switches, etc... Racks, Unix Servers, just TONS of "fun" stuff. It's all the makings of a World Class ISP to rival the likes of Pair Networks (they only have 300 GB!!). Our microwave link can handle up to 40 MBps before we have to add another dish, and the bandwidth is FAR cheaper than land line based (no local loops!!). We have a ping time to Yahoo.com of LESS THAN 7 ms!! Ping times like that are unheard of, unless you're "right there"! It's like they're on our LAN.

We've had our servers up and testing for about 2 months now, and the new WHT FTP site with OVER 1.5 GB of "TI STUFF" has been accessible for about 2 weeks (ftp://209.172.89.40). LOTS of stuff has been downloaded, we've had over 6.5 GB of downloads in the past week alone!! If I wasn't doing my own hosting it would have cost more than $800 per month on another commercial server! I'm hoping we can make this new site the "site of sites" for the TI.

We now have EVERY issue of Micropendium and other magazines online scanned ready for you to download (Thanks VERY much to Charlie Good and Rich Polivka for taking all the time to do this!!), as well as TONS of software in both TI and PC99 format. We have the complete CompuServe Archives, GenialTravler, and much more! Check it out if you have internet access. If you have any software you'd like to upload, let me know, we sure have the space! ;).

Well, enough of my rambling, you all know what I've been up to now, so you can see why it's taken so long to get these things out. With the weather changing, there are fewer things that can be done outside, so I'll be spending more time inside getting the cards out, and now that the servers and such are nearly done, and we're getting an assistant, I'll have plenty of time to get the cards out.

I hope everyone had a great summer, and I look forward to the great new millennium!



APPLICATION FOR MEMBERSHIP

Below you will find an application for membership in the Hoosier Users Group. Active membership entitles you to the Newsletter, up and down loading rights on the HUG bbs, attendance and voting rights at regular club meetings, access to the HUGger Library of Programs, special club activities and special guest speakers for one year.

New memberships and renewals are $20.00/year. Make check or money order payable to Hoosier Users Group. Send completed application to:

Hoosier Users Group
Dan H. Eicher
4509 Northeastern Ave.
Indianapolis, IN 46239

Cut on Line

Name:_____

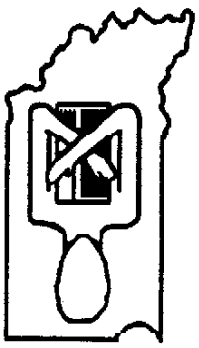
Address:_____

City, State, Zip_____

Phone:_____ E-mail:_____

Next meeting
November 21st!!

Hoosier Users Group S&T BBS
300/1200/2400/4800/9600 Baud 8N1
317-782-9942 24 Hours Daily



Hoosier Users Group
Dan H. Eicher
4509 Northeastern Ave.
Indianapolis, IN 46239

Forwarding and Address
Correction Requested