

GROUP HOOSIER USERS HOOSIER USERS GROUP USERS GROUP HOOSIER GROUP HOOSIER USERS HOOSIER USERS GROUP

HOOSTER USERS GROUP HOOSTER USERS GROUP HOOSTER USERS GROUP HOOSTER USERS GROUP HOOSTER USERS GROUP USERS GROUP HOOSIER USERS

### THE HUGgers HOOSIER USERS GROUP People Helping People

HOOSIER USERS GROUP USERS GROUP HOOSIER GROUP HOOSIER USERS HOOSIER USERS GROUP

USERS GROUP HOOSIER GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS

November/December 1995

The HUGgers Newsletter

Volume 14, Number 6

# Season's Greetings



#### ATTENTION!

#### December Meeting....

Instead of a having our regular meeting on the 3rd Sunday of the month we are planning to get together for a "Christmas Dinner" on the 3rd Saturday, Dec. 16, at the OLD COUNTRY BUFFET at Southern Plaza Shopping Center. We will be meeting there at 4:00. Everyune is welcome,..., so please join us there.

H.U.G. Officers

P.S. Dinner will be on a "Dutch treat" basis.

### ABOUT THE 13th ANNUAL CHICAGO TI FAIRE...

The 13th annual Chicago TI Faire was held on October 28 at the Evanston Public Library. As usual, the Hoosier User's Group had a table with various items for sale. Compared with previous years - before say 1993, attendance was comparitively low and as a consequence so were sales for us as well as regular venders. There was little new software or hardware Venders present included: available. Bud Mills Services, Cecure Electronics, C.a.D.D. Electronics, Rick Bottoms (RDB Enterprises), Ramcharged Computers, Micropendium Magazine. Other users groups included TI Users of Will Co. (IL), Mid-South User Group (TN), Lima User Group (OH), Great Lakes Computer Group (MI), Windy City 99 Club (IL) and Vereniging TI-Gebruikersgroep from The Netherlands.

Seminars included a presentation by Berry Harmsen of the Netherlands users group on a new high-speed GPL card from Germany. This card replaces the console GROMS and provides a 600% The card has 6 banks of speedup. flash rom, two of which can hold application cartridge code. Winford Winkler has deveoped the operating system for the card. The same people are apparently also working on a new "computer on a card" for the P-box; this one will be a TI 9900 mini computer compatible. Sounds interesting.

A new loader program called "Load Master" by Mickey Cendrowski was

Mike Wright, of C.a.D.D.
Electronics showed the newest version
of their PC99 emulator for IBM clones.
New features include function key
info on screen, enhancements to the
"mini screen" rom/ram editor, and new
disk based documentation. Two
versions are available a Lite version
for \$47.00 and the full version for
\$94.00. All TI produced software
disks are now available from C.a.D.D.
Electronics in PC99 format.

demoved by Norm Rokke; and Dr. Charles Good did a demo on the AMS (Asgard Memory System) card, newly available from the Southwest 99er's group. Dr. Good also did a demo of Term 80 by Jeff Brown. This is a terminal emulator program that does 80 columns on a standard TI 99/4a.

Regarding the continuing saga of the Western Horizon Technologies SCSI hard disk controller, Bud Mills in his seminar showed and distributed beta versions of the eprom that is needed to make the SCSI card a viable product for TI 99/4a systems. Hopefully final versions will available before too long.

Overall, the Chicago Faires are still worthwhile. Although at this point it is enjoyable more as an opportunity to talk to other TI and Geneve users and notables. And it is still a good place to find many of those items that you may not have gotten around to getting in the past.

-BCP

Msg#: 49 \*For Sale Messages\*

95/08/02 08:30:50 pm From: Steven Bundy #324

To: ALL

Subj: TI 99/4A Computer

\*\*\*Edited for Newsletter\*\*\*

I have a TI 99/4A computer and accessories for sale. Also a Gramkraker and manuals. Also a large number of programs and TI modules. Call me for prices or call and make me an offer.

My phone number is 317-462-2127.

Msg#: 55 \*For Sale Messages\*

\*\*\*Edited for Newsletter\*\*\*

95/11/22 02:52:14 pm

From: Fred Edstrom, Jr. #10

To: ALL

Subj: Selling of TI 99/4A System

I'am selling all my TI 99/4A system with the following items:

Black and Silver TI 99/4A console

P-box, Flexcable interface, RS232, MBP Clock with analog to digital functions, Corcomp Disk Controller (Double density capability), three full height DS/DD disk drives (two in a external IBM disk drive enclosure), Modem cable for Zoom modem, Hitachi 14 inch color monitor.

Have many modules and manuals included. All computer items are in working order and can be seen upon request. Please contact on HUG BBS (User #10) or call 317-898-7300. Asking \$200 or o.b.o.

# A Christmas Wish

HUG OFFICERS (Area cod	ie 317)
President Ricky Bottoms	392-2312
Vice Pres Bryant Pedigo	255-7381
Secretary Greg Larson	783-4575
Tresurer Greg Larson	783-4575
Librarian Byrant Pedigo	255-7381

## 

## Address Changes

Subscribers who move may have the delivery of their most recent issue(s) delayed unless MICROpendium is notified six weeks in advance of address changes. Please include your old address as it appears on your mailing label when making an address change.

Check each item ordered (or list on separate page) and enter total amount here:				
Check/MO 🗅 🗀 (check one)				
Card No.				
Expiration Date (Minimum credit card order is \$9)				
Signature (Required on credit card orders.)				
No sales tax on magazine subscriptions. Texas residents add 7.75% sales tax on other items, including back issues and disk subscriptions. Credit card orders add 5%.				
Mail to: MICROpendium, P.O. Box 1343, Round Rock, TX 78680				
Name				
Address				
City				
State ZIP				
The set of numbers at the top of your mailing label indicates the cover date of the last issue of your subscription.				



Monthly meeting location White house next to St. Ann's School 2839 S. McClure Indianapolis, IN

December Dining Meeting Sat., December 16, 1995

Next HUG meeting: January 21, 1996

HUG S&T BBS Hoosier User's Group 300/1200/2400/4800/9600 Baud; 24 Hours Daily 317-782-9942 8N1

Now with 80 Megs Storage

# PC99 vs. the Red Baron

## Breaking down the defenses of a worthy adversary

#### By MIKE WRIGHT

Manfred Freiherr von Richthofen was born May 2, 1892. In 1916, during World War I, he became a German fighter pilot. As the war progressed, he grew to fame as the Red Baron, so-called because of his red Fokker triplane, and was credited with shooting down 80 Allied planes in dogfights. He died on April 21, 1918, when he himself was shot down.

In the TI world, the program SPAD XIII from Not Polyoptics puts you at the helm of a SPAD XIII biplane and lets you do battle with the legendary Red Baron. (SPAD is an acronym for: Société Pour Aviation et ses Dérives, a French company run by aviator Louis Bériot, that built the plane.)

SPAD XIII was released in January 1987 according to MI-CROpendium (3:10:6). It was supplied on disk, required 32K memory expansion, and autoloaded under Extended BASIC. However, it was extremely difficult to make backups of the disk since the program was heavily protected.

We at CaDD Electronics were faced with the problem of dealing with this protection when one of our PC99 customers sent in some TI disks to be converted for use with PC99. Using the conversion utility supplied with PC99, we had absolutely no luck with the SPAD disk. Eventually this grew to be a challenge — a leather flying gauntlet slap in the face, so to speak. After all, as far as we know, there are no programs that run on a standard 4A that do not run under PC99. We wanted to preserve our 100 percent record.

The first thing we did was to copy the SPAD disk to a PC using the RSECTOR and WSECTOR utilities supplied with PC99. These utilities copy disks by reading and writing sectors. RSECTOR soon reported an error reading sector 2, and then got an error with every sector from 135 through 359.

We then used Disk+Aid to sector read the original disk. We examined sector 0 and found at offset >10 the bytes >50 and >28. The >50 ("P" in ASCII) means the disk is protected from Disk Manager copies. The >28 means there are 40 (decimal) tracks on the disk. We then used Disk+Aid to confirm what RSECTOR had shown us. Essentially, only the first 15 tracks (15 x 9 = 135 sectors) were formatted. Of these, sector 2 could not be read.

This meant that the byte representing the number of tracks had been patched to fool copy programs into believing that this was a "normal" disk.

If you use the original SPAD disk on a 4A and try to do a Disk Manager catalog, it will report that there are no files on the disk. Similarly, in PC99 you can use the DSKDIR.EXE utility, which allows you catalog TI "disks" from DOS. This, too, follows Disk Manager practice and reports no files on the disk.

The trick used here is to put two null bytes at the start of sector 1—the directory link sector. This contains a series of two-byte entries, pointing to sectors which have File Descriptor Records (FDRs). The Disk Manager catalog routine reads sequentially from the beginning of sector 1 and exits when two null bytes are found. If there are two nulls at the beginning of the sector, then it believes there are no files on the disk. For normal usage,

DSKDIR EXE emulates this behavior. However, DSKDIR.EXE has an override switch for situations like this, If you do: >DSKDIR DSK1 -d -x 4

You will tell DSKDIR.EXE to try to find four FDRs on the disk, even if some of the entries are null. You have to be careful doing this, because you can expect some garbage. In the case of the SPAD disk, DSKDIR.EXE returned the following:

No.	FDR	Filename	Size	Туре	P
001	>000	SPADXIII	21324	PROGRAM	Y
002	>004	LOAD	34	PROGRAM	Y >006 033
003	>005	LOAD24K	97	INT/VAR254	Y >027 096
004	>000	SPADXIII	21324	PROGRAM	Y

Note that entries 1 and 4 are bogus, but that there are two real files on the disk: LOAD, and LOAD24K. DSKDIR.EXE also shows that LOAD starts in sector >006 and is 33 sectors long, while LOAD24K starts in sector >027 and is 96 sectors long.

When you tell Extended BASIC to RUN a program, it uses a different mechanism from the Disk Manager to find the filename. Instead of starting at the front of the directory link sector and searching sequentially, it starts at the middle. If nothing is found, it goes halfway back and so on until it finds an entry. It then looks up this entry and, if the name found is greater than the name it is looking for, it will go halfway back again. Similarly, if the name found is less than the name it is looking for, it will go halfway forward. This binary search is considered to be faster than search sequentially through the FDRs.

So if you put the SPAD disk in DSK1 and start Extended BA-SIC, then XB's autoload mechanism will find DSK1.LOAD and execute it — even though there are apparently no files on the disk.

This also means that you can start Extended BASIC, insert the SPAD disk in DSK1, and do OLD DSK1.LOAD. If you then try to do a LIST, you will get a PROTECTION VIOLATION. This can usually be overcome by:

```
CALL INIT
CALL LOAD (-31931, 0)
```

But, if you do this, Extended BASIC crashes. The reason is that the SPAD LOAD program has had each BASIC line length byte removed and replaced with >00. It is surprising, but Extended BASIC does not need the line lengths to run. However, one or more lines without a length byte just blows LIST away.

At this point we used the DSKOUT.EXE utility on the PC. This allows you to extract a BASIC file from the TI "disk" and create a DOS file. Then we ran the BAS2ASC.EXE (BASIC to ASCII) utility on the extracted file. BAS2ASC.EXE knows about BASIC lines without a length byte, and will re-create the value. So now, we had a listing of DSK1.LOAD.

90 DISPLAY AT(10,7) ERASE ALL: "SPAD XIII Mk. 2" 100 CALL INIT 110 CALL LOAD(8196,255,208)

110 CALL LOAD(8196,255,20 120 CALL LINK("OS")

130 DISPLAY AT(22,11): "LOADING"

## SPAD —

#### (Continued from Page 23)

) RUN "DSK.SPADXIII.LOAD24K"

In addition to these BASIC statements, the file LOAD also contains a large amount of embedded assembly code. In fact, the total file size is about 8K. So when this program is loaded, the line number table and a whole lot of assembly code are stored starting at the end of memory (>FFFF), while the actual BASIC code is stored starting at >A000.

The CALL INIT tests to see that the Memory Expansion is available, and loads TI utilities into low memory (>2000 ->3FFF). These include VSBW, VMBW, VSBR, VMBR, and VWTR. (These are described on page 248 of the Editor/Assembler manual.)

The CALL LOAD is a clever trick. This translates to storing at address >2004 the value >FFD0. Address >2004 is used by the BASIC interpreter to store the last free address in low memory, and is used when searching the REF/DEF table. Usually, this value will never exceed >3FFF. Remember that the 32K memory peripheral consists of two disconnected segments: low memory (>2000 - >3FFF) and high memory (>A000 - >FFFF). This CALL LOAD tells the interpreter that the end of low memory is at >FFD0, which is actually pretty near the end of high memory.

At this stage we ran PC99. At the title screen we entered the mini-screen debugger and set a watchpoint: >2004 = >FF. This means a PC99 break will occur any time the value >FF is written to >2004. We then pressed a key, selected Extended Basic and led for the break to occur.

We then used the one of the PC99 memory windows to examine CPU RAM. We did a search for QS and found that it existed in three places. However, at >FFD0, there was a REF entry which showed that the executable address of QS was >FEBE. This REF was created by the embedded assembly code in LOAD.

We then set a break at >FEBB and when this occurred we went into step mode. This allowed us to write down each instruction that was being executed. Listing 1 shows the disassembled code.

### LISTING 1

```
FEBE 020c LI R12, >1000cru base address
FECO 1000
FEC2 1e00 SBZ 0 turn off
FEC4 022c AI R12, >0100point to disk controller
FEC6 0100
FEC8 1d00 850 0 Lurn It on
FECA c060 MOV 9>400a, R1
                               DSRLNK low level = >4010
FECC 400a
FECE 8821 C 0>0004(R1), 0>feaa>feaa = 0110 = r/w sector
FED0 0004
FED2 feaa
FED4 16f6 JNE >fec2
                       if r/w sector not there, try
FED6 c821 MOV @>0002(R1), @>feaa
                                       = >5b38 = routine
>10 = r/w sector
   1 0002
LLUA feaa
FEDC 0200 LI R0, >040A
FEDE 040a
```

```
>0110
FEE2 8356
FEE4 0200 LT R0, >01ff >01 = drive, >ff - 22 >01 = read
FEE6 01ff
FEE8 c800 MOV RO, @>834c
                               drive in left byte, flag
in right
FEEA 834C
FEEC 0200 LI RO, >1000
FEEE 1000
FEF0 c800 MOV RQ. 9>834e
                               VDP addr of input buffer
FEF2 834e
FEF4 04e0 CLR 0>8350
                       sector number 0
FEF6 8350
FEF6 C260 MOV %>Ieaa, R9
                               saved addr = >5b38 = r/w
sec
FEFA feaa
FEFC 0699 BL *R9
                       read sector 0 (SPADXIII...)
FEFE 1000 NOP skip over bumped return addr
FF00 0200 LI R0, >1010 VDP address
FF04 02al STWP R1
                       CPU start address (>83e0 = R0)
FF06 0202 LI R2, >0002 number of bytes to read
EE08 0002
FF0A 0420 BLWP @>202c (VMBR vector = 2038/24aa)
FF0C 202c
FF0E c2m0 MOV 0>0002, R11
                               [this usemm to do noth-
ingl
FF10 0002
FF12 0280 CI R0, >5028 50 = *P*, >28 = 40 tke/eide
FF14 5028
FF16 1646 JNE >ffa4
                       error exit
FF18 0200 LI R0, >0003 sector number (3 = copyright)
FF1A 0003
FF1C c800 MOV RO, @>8350
                               sector number 3
FF1E 8350
FF20 C260 MOV @>feaa, R9
                               = >5b38 = r/w sec
FF22 feaa
FF24 0699 BL *R9
                       read sector 3
FF26 1000 NOF
FF28 0200 LI R0, >1028 VDP address (>20, >e5, copyright
end
FF2A 1028
FF2C 0201 LI R1, >feac CPU start address
FF2E feac
FF30 0202 LI R2, >0002 number of bytes to read
FF32 0002
FF34 0420 BLWP @>202c VMBR. >feac = >20, >e5
FF36 202c
FF38 6060 MOV @>febc, R1
                               = >feae
FF3A febc
FF3C c2eo MOV @>0002, R11
                               (this seems to do noth-
inal
FF3E 0002
FF40 c831 MOV *R1+, 0>8350
                               sector number. R1=>feb0.
>8350=>0001
FF42 8350
FF44 132f JEQ >ffa4
                       fails if end of table = >0000
FF46 c801 MOV R1, 6>febc
                               = > feb0
```

VDP addr for dummy PAB =

FEE0 c800 MOV RO, @>8356

## SPAD -

```
(Continued from Page 24)
FF48 febc
FF4A c260 MOV 6>feaa, R9
                                = >5b38 \text{ r/w sec}
FRAC food
FF4E 0699 BL *R9
                                             (order
                       read
                                 sector
1.4.5.6.7.8)
FF50 1000 NOP
FF52 0200 LI R0, >0003 sector number (3 = copyright)
FF54 0003
FF56 c800 MOV RO, 4>8350
                                sector number
FF58 8350
FF5A c260 MOV 0>feaa, R9
                                = >5b38 = r/w sector
FF5C feaa
FF5E 0699 BL *R9
                        read sector
FF60 1000 NOP
FF62 0200 LI RO, >1028 VDP address
FF64 1028
FF66 02a1 STWP R1
                        >83e0 = cpu address
FF68 0202 LI R2, >0002 number of bytes to read
FF6A 0002
FF6C 0203 LI R3, >FF88
PP6B ££88
FF70 0420 BLWP 8>202c VMBR
FF72 202c
FF74 2820 XOR @>feac. R0
                                >feac = >20e5: 'real' sec
3 = >20e5
FF76 feac
                "fake" sec 3 = >e5e5
FF78 13df JEQ >ff38
                        if "real" sector found then loop
FF7A 64c0 S RO, *R3
                        R0 = > c500, *R3 = > cc70
FF7C a8c0 A R0, 0>0008(R3)
FF7E 0008
FF80 a8c0 A R0, @>000e(R3)
FF82 000e
FF84 1e00 SBZ 0
FF86 0200 LI, RO, >c397
PP88 -397
FF8A a020 A 9>8384, R0
FF8C 8384
FF8E 0201 LI R1, >5b00
FF90 5b00
FF92 c081 MOV R1, R2
FF94 c110 MOV *R0. R4
FF96 41c4 SZC R4, R7
FF98 cc44 MOV R4, *R1+
FF9A 0640 DEC RO
FF9C 0642 DEC R2
FF9E 16fa JNE >ff94
FFA0 0460 B @>006a
                        return to GPL interpreter
FFA2 006a
FFA4 0200 LI RO, >a000 error exit
FFA6 a000
FFA8 0f40 CLR *R0+
                        clear all of memory
FFAA 0280 CI RO. >ffaa from >a000 - >ffaa
FFAC ffaa
FFAE 16fc JNE >ffa8
FFB0 0420 BLWP 6>0000 branch to title screen
FFB2 0000
```

The first thing the assembly code does is to probe for the disk peripheral (>FEBE - >FED6). When found, the code uses the

hard-coded address >400A in the peripheral ROM to find the address of the card's read-write sector routine. This is saved at a dress > FEAA.

The code then sets up to do a read of sector 0 into VDP memory at >1000. There is a minor irregularity here which uses >01ff instead of >0101 as the drive number and flag (read), but we think this is an attempt to obfuscate the code. The BL \*R9 at >FEFC then reads the sector. (If you do not understand how to do direct sector reads, there was a short article by Dick Vandenberg in Computer Shopper, 84:11:152, that explained the procedure.)

The TI VMBR loaded by CALL INTT at vector >202C is then called. It loads 2 bytes from VDP address >1010 into >83E0, which is the calling workspace's R0. The value read is then compared to >5028. The program is determining if the disk is protected (>50 = "P"), and if it contains 40 (>28) tracks. If not, the code jumps to an exit routine at >FFE4. Since the real disk only has the first 15 tracks formatted, this is another protection device.

Next, sector 3 is loaded. This contains a copyright notice and the code depends on the fact that this is of a specific length. It probes at offset >28 and reads the two bytes into address >FEAC. These two bytes are >20 (a space) and >E5 (the value used by TI to fill a blank formatted sector).

A loop now starts at >FF38. Address >FEBC is a changing value that originally contains >FEAE. The memory starting at >FEAE contains a table:

>00 >01, >00 >04, >00 >05..., >00 >00.

This table is a series of sector numbers: 1, 4, 5, 6, 7, 8, and en with two null bytes. The code now reads the table value. If it is 0, it jumps to the error exit. If nonzero, it uses that value to read a sector. The contents of the sector are then simply ignored.

The code at address >FF52 then sets up to read sector 3 (the copyright sector). The code at address >FF62 then reads two bytes from offset >28 of the copyright sector and compares them with the previously saved value of >20E5. If they match, the code loops back to >FF38.

For quite some time we were totally baffled. The code always failed, since it always reached the end of the sector table starting at >FEAE. We had expected there to be a check for the missing sector 2, but there was none.

We now went back to the 4A and used Miller's Graphics Advanced Diagnostics to read in track 0 of the SPAD disk. We connected a PC to the 4A on a serial line, set up the PC with Pro-Comm Plus (a terminal emulator), and set Diags to output screens to the TI RS232 port. We then dumped each of Diags' buffers to the PC. We did this twice, once in hex format and once in ASCII.

We then checked the inter-sector information. This contains, among other information, an address mark (>FE), followed by four bytes: the track number, side number, sector number, and AU size (>01 = 256 bytes/sector). We used a PC editor called Brief to search for all >FEs in the file and found, sure enough, that there was no sector 2. Instead, there was a second sector 3! We will dinguish these sector 3s by calling them "real" and "fake. The "fake" sector 3 is where sector 2 should have been, according

## SPAD -

#### (Continued from Page 25)

the interlace. We now constructed the following table:

Sector contents in interlace order:

- 0 SPADXIII... DSK
- 7 >01 >0a
- 5 LOAD24K
- 3 Copyright (real)
- 1 >00>00>00>04>00>05
- 8 >04 >E2
- 6 >FB >97
- 4 LOAD
- 3 (2) Copyright (fake)

The contents of the fake sector 3 looked the same as the real sector 3. So we returned to PC99, put SPAD in DSK2, and loaded Disk+Aid from DSK1. Note that there is no magic about Disk+Aid. We just happen to use this as a sector editor since we are familiar with it. We also purchased the source code for this program many years ago allowing us to make changes to the program if we need to.

We then copied real sector 3 to real sector 2 on our "disk." We then used a dump program to find the sector address marks in the PC99 SPAD .DSK file. We then used the PC99 utility PATCH.EXE to change this value from >02 to >03.

We now felt very confident that the program would run. But, once again, the Red Baron put us into a tailspin, as the code returned to the title screen.

By now, we were a little desperate. However, we had a niggling feeling that the program must depend on something in the fake sector 3. We again used Brief to look at the dumped file, but this time we used two windows, so that we could see the two sectors simultaneously. Then we spotted it! There was a one-byte difference in the two sectors. The real sector 3 had an extra space making offset >28 to be >20E5, while the take sector 3 had >E5E5 at offset >28.

Then it dawned on us. The code depends on the hardware and the interlace order. It reads a sector (from the table) and then always reads sector 3. The disk controller eventually reads the fake sector 3 instead of the real one.

The code at >FF7A deserves some mention. While looking through hex dumps of the SPAD disk we didn't see much obvious assembly code. After a while you expect to see things like >0200 (for LI, R0), etc. We finally figured out that large portions of the SPAD code are stored in reverse byte order. Again, this is done as a protection device to prevent disassembly. The code at >FF7A reads the reversed bytes, and puts them into proper order in memory. The count of the bytes depends on the XORed value from the fake sector 3. Very tricky!

The error routine at >FFA4 is also rather nasty. It erases all of memory. This prevents you from using CALL LOAD to examine memory after the program fails.

Armed with our new observations, we once again patched the 'AD .DSK file and changed fake sector 3 to match the original. Now we were cookin'. With guns primed we once again challenged the Red Baron — and once again we were shot down.

We now got to thinking about the way PC99 emulates the disk

controller. We realized that, as the emulation is currently structured, we would never get the SPAD code to execute correctly. PC99 reads in a "track" of information, and then searches sequentially for the first matching sector address. So even though there are two sector 3s on the track, the emulation would never find the second one, since we do not emulate seeks that depend on interlace order.

At this point we were virtually out of ammunition. We decided against rewriting the disk emulation code to handle this one special case. Instead, we decided to patch the SPAD code to defeat the protection. The read from fake sector 3 will return >E5E5 in R0. At address >FF74 this value is XORed with >20E5. The result would be >C500. Since all of the reads are simply designed to wait until the fake sector is read, we figured we could replace the XOR with:

FF74 0200 LI R0, >C500 FF76 C500

This was surely our last chance! We started PC99, set a break at >FEBE and allowed SPAD to load. We then went to a memory window and replaced the XOR with the LI. We then hit continue. Since the debugger is inherently slower than the accelerated version of PC99, there were many anxious moments. The first hint of success was that we saw the message "LOADING" appear at the bottom of the screen. This meant we had exited the LOAD program, and were now running LOAD24K. A few seconds later the SPAD title screen appeared.

We pressed the joystick fire button and saw the guns of our SPAD fire. In our mind, the Red Baron's Fokker triplane plunged to the earth and self-destructed. Our worthy adversary had finally been defeated. To put the seal on the event, we again dumped the SPAD disk and found the bytes that corresponded to the memory patch. We then sector edited them so that the LI permanently replaced the XOR. It was now possible to run SPAD normally under PC99. We had completed our rout of the Red Baron, and maintained our 100 percent compatibility record.

#### SUMMARY OF PROTECTION MECHANISMS

- 1. Only the first 15 tracks of the disk were formatted. The byte containing the number of tracks was then patched to be 40 (>28).
  - 2. The disk was protected against copy by the Disk Manager.
- 3. Track 0 had no sector 2. Physical sector 2 was replaced with "fake" sector 3. Fake sector 3 had a one-byte difference from real sector 3.
- 4. The directory link sector started with two null bytes. This foots the Disk Manager into thinking there are no files on the disk.
- 5. The LOAD program was protected using Extended BASIC protection.
- 6. The LOAD program had all the line length bytes removed, which prevents it from being listed even after the XB protection is removed.
- 7. The last free address in low memory is patched to point to the end of high memory, which ties to code embedded in the LOAD program.
- 8. The CALL LINK calls a small assembly language program that was embedded in the LOAD program. Any errors detected by

## SPAD -

(Continued from Page 26)

the assembly code cause all of memory to be wiped out.

- 9. The assembly code checks to see that sector 3 has the SPAD copyright and that it is the correct length. If not, it fails.
- 10. The assembly code checks to see that sector 0 has the Disk Manager protection and 40 tracks. If not, it fails,
- Most of the loaded assembly code is in byte-swap format making it hard to disassemble.
- 12. The main protection loop depends on the fact that the fake sector 3 will eventually be read by the hardware.

Comments on protection:

We believe the programmer for SPAD was Larry Hughes, formerly of QS Software. We would like to tip our flying helmet at his ingenuity in devising the above scheme. It certainly was a challenge to unravel it.

However, we believe that software developers have a responsibility to users of their product. This responsibility includes the ability to use the software in a convenient and nonintrusive manner. The user should also be permitted to make backups of the software to guard against media failure.

As an example, Millers Graphics Advanced Diagnostics was probably the most sophisticated disk exploration tool available for the 99/4A. Yet the disk is heavily protected and most unsophisticated users would not be able to copy it. Some years ago MG left the 4A world. If you now find that Diags will no longer load you will

probably not be able to get a replacement copy and your investment in this product will have been lost.

It is our contention that software should not be protected, and should be as open as possible. We follow this policy with PC99, even though we know that some illegal copies exist. We prefer to stick to our principles and take the monetary loss rather than inconvenience our users. Resides, nearly all software protection schemes can be broken, so there is little point in using them.

It was not our intention to show you how to "break" SPAD. Even with the above information it would still be quite difficult to do this on a 4A. There are also some "holes" in our knowledge of how certain things were done. For example we do not know how the fake sector was created, or what machinations took place to pack and byte-swap the assembly code. [If anyone knows Larry Hughes, it would be interesting to show him this article and see if, after all this time, he would be willing to reveal this information.]

Instead, we were trying to satisfy a customer's needs — someone who owned a legal copy of SPAD and wanted to see it run under PC99. In doing this, we were able to illustrate the power of the PC99 debugger, and the utilities that are supplied with PC99. Armed with these, a software developer has the most formidable array of tools yet assembled to help in developing sophisticated applications for the 99/4A and allow examination of the inner workings of the machine.

We also do not want to leave you wist the impression that the procedures in volved were quick and easy. The whole process was spread over three weeks and took place at home, over lunch at the office, and during a business trip to Singapore (courtesy of our laptop). However, we would rather have devoted this time to enhancing PC99.

If there were any positives, apart from now being able to run SPAD, it did cause us to make minor improvements to: BAS2ASC.EXE (and the related IV2ASC.EXE, for internal/variable 254 files); DSKDUMP.EXE and DSKCHECK.EXE, which now handle bad sectors; and PC99A.EXE, the accelerated version which had a minor bug uncovered when using Disk+Aid. These will all be incorporated in the next release of PC99.

Finally, we believe the process of uncovering SPAD's secrets shows that distinctions between the 4A world and the PC world tend to become blurred. The two machines are connected together with tumbilical RS232 cable which allows data to be transferred between them. If you need to look at the original disk you use Diags on the 4A. If you need to dump a file in hex it is quicker to use DUMP.EXE on the PC, and so on, allowing you to pick the best available tool to get the job done.

This lets you derive maximum benefit out of the hardware and software you own which, we think most people will agree, is a highly agreeable state of affairs.



Wishing all
Merry Christmas
and Happy
New Year

The complete

Msg#: 343 \*General Messages\*

.95/11/08 12:41:57 am (Read 18 times)

.From: William M. Lucid #1

To: ALL

Subj: 600v9t9.zip

600V9T9.ZIP is a rather large IBM file to download. Hoosier User's Group has copies of the shareware program available for \$3.00, picked up in person. Mail order copies should include another \$1.00 for mailing and handling costs.

Order from: William M. Lucid, 6005 Elaine Street, Speedway, IN 46224-3032.

Make check or money order out to the Hoosier User's Group.

600V9T9.ZIP allows one to use an IBM pc to run TI 99/4A software. Requires at least a 386 machine with a 101 key keyboard.

I will be happy to answer any questions about v9t9 (TI Emulator). This package contains all the utilities you need to dump rom/grom from YOUR TI 99/4A system (This is required to make use of the program, as NO ROM or GROM is included in 600v9t9.zip).

\* ORIGIN: HUG TI 99/4A BBS (317)-782-9942 / 8N1 / 24 hours

Msg#: 342 \*General Messages\*

95/11/08 12:32:33 am (Read 18 times)

From: William M. Lucid #1

To: ALL

Subj: v9t9 Cables

The Hoosier User's Group has eight cables left from the Chicago Faire that are for use with Edward Swartz's v9t9 (TI Emulator) program. These cables are 15 foot serial cables with male/female connectors, plus an special adapter for proper handshaking. Cable and adapter are required for dumping rom/grom from TI 99/4A sytem to the IBM pc. Price of the cable is \$20.00. Shipping and handling \$5.00 or pick at a HUG meeting for no shipping and handling.

If you are not in the Indianapolis area, order from William M. Lucid, 6005 Elaine Street, Speedway, IN 46224-3032.

Make check or money order out to: Hoosier User's Group.

These adapters have been tested and are known to be functional. Buy now before they are gone!!!

autt ug and	DIERSE DATUS
	.0450
,	Date:
New membership:	Иате
	<b>y</b> qqt.ess
Кепемада:	04045/415
222,00	City/State
	byone number

HOOSIER USERS GROUP P.O. Box 2222 Indianapolis, IN 46206-2222

Make check or money order payable to Hoogier Users Group, Send completed appli-cation to:

Below you will find an application for membership to the Hoosier Users Group. Active membership entitles you to the Newsletter, up and download on the HUCbbs, attendance and voting rights at regular club meetings, access to the HUCger Library of Programs, special club the HUCger Library of Programs, special club activities and special guest speakers tot one year.

## APPLICATION FOR MEMBERSHIP



## HOOSIER USERS GROUP

P.O. Box 2222 Indianapolis, IN 46206-2222

Forwarding and Address Correction Requested



May 1996

Dan H. Eicher 2720 Palo Verde Ct. Indianapolis, IN 46227

Hoosier User's Group S&T BBS 300/1200/2400/4800/9600 Baud 8N1 317-782-9942 24 Hours Daily