

Classic 99

People Helping People

The Official Newsletter of the Hoosier Users Group

May - June 1999

The HUGger's Newsletter

Volume 18 Number 3



Officer's Corner

By Dan H.Eicher

The next meeting is May 16th! Hope to see you

A big Hello to our newest member, Oliver Hebert from Brewton, AL. Oliver has been a long time TI user and programmer. Some of his credits include many great programming articles in the Chicago newsletter and beta testing and documentation of the Missing Link package for extended basic.

Two highlights in this issue are the second part of Mike Wrights article on cracking Doom of Mondular so that it will run under PC99 and an article by Guntis Sprenne. Guntis was employed for many years as a field service engineer in TI's Computer Division and has many insights into TI. Thanks Guntis!

We had a great April meeting! Bill setup his PC at home to work as an ftp server, we managed to dial-in with my Geneve and pull down some files. Bryant brought over an Amiga and we attempted to get a TI emulator designed for that hardware platform running (no luck, but Bryant promised to get it working for us, and demo it a later meeting). We were able to use a scan doubler that allowed him to hook up his Amiga to a standard VGA monitor. Hopefully someone will figure out a way of doing this with 9938 and 9958 equipped TI equipment.

We played with the SNUG system. True to computer form, I tried to retype in my basic benchmark, it seemed to get hung in an infinite loop despite looking it over and over, I couldn't find the problem, fortunately, later I found a stored copy of the program and was able to bench mark the SNUG system,

results later in this article. I found the bug was not a mis-type, but instead was an error in the printed listing that was published in January/February 1998 issue. The error was printed 180 If A(K)=A(K+1) THEN 230, it should have been If A(K)<=A(K+1) THEN 230! Lots of good talk and a good time was had by all!

COMDEX this year was rather disappointing. There were only about half as many vendors as there were about 4 years ago. Unfortunately, it was the half that I missed the most, the hardware vendors! Bill Gates keynote address was how Windows2000 was going to make life easier. The highlight of the trip was chatting with Hal Shannafield and Steve Weber of the Chicago TI users group who also were visiting COMDEX.

Speaking of Bill Gates, I just finished a book by Stephen Manes and Paul Andrews called "Gates". It chronicles Bill Gates from his early boyhood to about 1992. It gives some great insights into the early Gates and Microsoft's beginnings! Well worth reading, available from the Marion County Public library.

SNUG Bench Mark

In the Jan/Feb '98 issue, I published a basic program to bench mark speed, I ran the test on a variety of different 9900 and 9995 machines in various basics dialects. Now that I have a new 9900 based system, the SNUG card set, I want to see how fast I could get this system to go. So on the advice of Michael Becker, I

continues on next page

Table of Contents

- 1. MFM DRIVES
- 2. Article Update
- 3. News worthy notes from the TI LIST Server
- 4. Genial Travler now available on the net
- 5. MBX WEBSITE
- 6. TI Reflections
- 7. CPU Reference List
- Doom of Mondular Part 2 of 3

issued the following commands: CALL XB16 runs the Extended Basic GPL interpreter in 16 bit mode, CALL MEM16 allows access to memory (the normal TI 32K) to be accessed at Zero Wait states and 16bits at a time instead of 8 and CALL FAST, which access GROM memory at CPU clock speed instead of GROM clock speed which is 44 Kilohertz.

ABASIC

ABASIC

Here are the results:

GENEVE XBASIC

OLIVE	ADIABLE	11D11D1C	MONDIC
•			(WITH TYPE OF INT)
SORT	7:05	4:22	3:14
PRINT	2:10	:55	:44
TOMMY	TUTOR	BA	SIC
SORT		9	:21
PRINT		2	:10
99/4A E	BASIC	XBASIC	XBASIC+32K
SORT 1	.0:54	13:37	13:18
PRINT 1	1:38	4:30	4:30
SNUG X	BASIC+CAI	LL FAST+C	ALL XB16+CALL MEM16
SORT	11:00		
PRINT	3:44		

SNUG FAQ: The one problem with the snug card set is all the documentation is in German. Now, at least the High Speed GPL cards documentation has been translated. You can find the translation on the web at Alan and Tina-Earl Bray's web site, Http://www.bricktop.demon.co.uk/bricktop/

Because of the lack of English documentation I have passed many mail messages back and forth with Michael Becker, I have collected all of these in a file and changed them into a question and answer format, which I will be releasing shortly as the SNUG FAQ.

Tim Tesch update: First off Tim has been faced with a family tragedy. His older brother and his wife were killed in a tragic traffic accident on March 21. Prior to this Tim had announced because of work pressures (on a good week, 70 hours) and a general feeling of hum out, he was going to cease working on Myarc equipment. Our hopes and best wishes are with Tim at can instead be used to download a text file. It will run without this time. Tim did give me one tip, JDR Microdevices an AMS card but the size of the file which can be transferred www.jdr.com (1.800.538.5000) now has 512Kx8 flash eproms is reduced.

(29C040) on sale for 5.69. These are very close to the parts that were being used by Cecure for doing the PFM upgrades to Geneve's. So if you have a Geneve, you might want to invest in one of these in the hope that someone will write an article on how to add them to the Geneve!

MFM DRIVES

While cruising through March 1999 Computer Shopper, I came across an add by AA Computech, Inc. They have a very good selection of MFM/RLL and SCSI hard drives and controllers. Their price on MFM hard drives is more expensive than you would find at a hamfest, but all their drives are warrantied and tested.

1.800.360.6801 (www.aacomputech.com).

Article Update

This is an update to the article by Jacques Groslouis in the last bi-monthly issue entitled Using Funnelweb with your Horizon Ram Disk. Apparently, some of the article got lost, some place between six different computers. Here is the update. Thanks Jacques!

Hi Dan,

When I read my article appearing in your last newsletter I noticed that a number of lines had been dropped. Apparently I caused this problem when I transferred the article from my TI to my PC. The omissions appear in the paragraphs at the bottom of the first column on page 8 and continue to the top of the next column. The paragraphs which contained missing lines are reprinted below.

If your TI is connected to a PC by means of a serial cable from your RS232 card you can configure your printer name under FWB to be as follows: PIORS232.BA=9600,LF

When you call PF from Text Editor or CTRL P from DiskReview this long line will come up and an error will be sounded if you press ENTER. In order to send your file to your printer space out the dash after PIO and in order to send the file to your PC delete PIO. This saves having to remember the RS232 settings. To receive a file into Text Editor or TIW you must save (SF) the file to RS232.BA=600 but there is no practical way of saving this setting in advance. However Bruce Harrison's new AMS TRANSFER program is very useful and FWB uses high memory location >A000 to >A050 for a 'Mail Box' to store text file names for use by a variety of other programs. If you want to include the same feature in an XBasic program merge the Following program into your XBasic program. You can call it from your program by using CALL MAILBX("TARGET_FILE").

The program REM which you both published suffers from a number of dropped characters. For reasons that I do not understand a number of minuses () and a few blank spaces were dropped when I transferred a LISTing of the program from my TI to my PC using a RS232 cable. My speculation is that I may have to lower the baud rate I use. You may want to make reference to the attachment which contains the affected lines in your next publication.

The same problem has found its way into the program MAILBX where a minus sign should appear in front of 24577 and a space should appear between CALL and LOAD.

```
180 CALL PEEK(2,A)
200 CALL LOAD(2,0)
210 PRINT " Welcome to my TI Computer":
: : : : : :
230 CALL
SAY ("I+UNDERSTAND+THE1+Y+TWO+K+PROBLEM.
DO+YOU")
240 1@P
380 CALL
SAY ("DID+YOU+HEAR+THE1+ONE+ABOUT+THE1
#TEXAS INSTRUMENTS# HOME+COMPUTER")
450 CALL SPRITE(#2,100,7,R+B,CB,SX,SY)
460 CALL SPRITE (#3,100,6,RB,C,SX,SY)
470 CALL SPRITE(#4,100,14,R,C+B,SX,SY)
480 CALL SPRITE (#5,100,2,RB,CB,SX,0)
490 CALL SPRITE (#6,100,8,R+B,C,0,SY)
500 CALL SPRITE(#7,100,4,R,CB,0,SY)
510 CALL SPRITE(#8,100,10,RB,C+B,SX,0)
```

Sorry for any inconvenience these omissions may have caused.

Regards for now,

Jacques

jgroslou@nbnet.nb.ca

News worthy notes from the TI LIST Server

Dear Ti'er,

Due to circumstances beyond our control, we have been forced to reschedule TIMUG99 to Saturday, June 12th. The conference site and hours will remain as originally announced. (7:00 AM to 7:00 PM at Spang Mansion on Kolthoff Drive in Brookpark, Ohio.) Only the date has been changed to Saturday, June 12th. Please accept our sincere apologies for any inconvenience this change may have caused.

We are happy to announce today that we have secured Spang Mansion for Friday evening, June 11th from 7:00PM to 10:00PM for the preconference get together. If you are planning to arrive on Friday, June 11th for TIMUGÆ99 or are a local TIer, why not cometo the get together social at Spang Mansion? This will give you a chance to have a good time renewing old acquaintances and swapping tales, ideas and information as well as finding out just where Spang Mansion is located before the conference on Saturday morning. Snacks and soft drinks will be served.

Please remember Spang Mansion is a Brookpark, Ohio NONSMOKING, public facility and this restriction is strictly enforced. Also, DO NOT bring any equipment or valuables to the get together to be left over night at Spang Mansion. Over night security will NOT be provided.

Glenn Bernasek, Secretary TIChips GBBasics@aol.com

TI Disk Controller Chips Compendium

By Michael Becker

WD1771 – 40pin, singledensity only! The chip in TI's Disk Controller PHP1240, inverted Databus.!

WD1770 = 28pin, doubledensity, chip in MYARC. Not fully compatible with WD1771

WD1772 = 28pin, same as 1770, but faster step rates. late MYARC FDC.

WD1773 = 28pin, doubledensity, commandcompatible with 1771, but noninverted Databus late Corcomp, ATRONIC, BwG.

WD2793 = 40 pin, softwarecompatible with 1773, used in early Corcomp from Anaheim!

All WD17xx have builtin digital dataseparators, sometimes not used (ex: in TI's DC!) all WD27xx have built in analog (PLL) dataseps. (hard to adjust!!! Was the reason CC changed toWD1773 too).

The International TI99/4A and GENEVE Fair

Will be held in 1999 on Friday October 1 through Sunday October 3.

Welcome to the 14th International TI meeting in Europe. This exciting event is centrally located in the heart of Germany near Stuttgart in Freiberg/Neckar.

Friday 10am to 5pm Saturday 8am to 5pm Sunday 8am to 5pm

For more information, Oliver Arnold has created the Regards, following web page:

https://www.planetinterkom.dc/oliver.arnold/dcfault.html

Clipboard99 now available!

Just wanted to let everyone know that I made the old TMS9900 Clipboard files available from anonymous ftp. I was looking for a site to place them on when I just realized I could have done that here at work. I received permission from my supervisor to create and maintain a TI99/4A library here, so it will make a decent repository for the goodies I put together as they become available.

The files are archived using Barry Boone's Archiver v3 03 They are named CLIP1A, CLIP1B, etc. The number denotes the volume (111) and the 'A' or 'B' denotes the side of the disk. The Clipboard was designed to fit on a SS/SD "flippy" disk some of the filenames on each side clash if I remember correctly, so you'll want to make sure each side is on a separate disk. Keep in mind I was just a teenager when I worked on these long ago (about 10 years), so take some of verbiage contained within them with a grain of salt. (I remember staying up late into the wee hours of the morning along with Jon Dyer in order to get the last two or three articles completed.)

The files on this FTP site have a TIFILES header, this means that you will need to transfer them from your PC to your TI99/4a via a terminal program or perhaps using Magic File Manipulator. (If you want high speeds!) If you are using V9T9, you can use the XMDM2TI utility program in order to strip the TIFILES header and make the file usable by V9T9. I would assume that the other TIEmulator has such a utility as well. Make sure you set your FTP client to binary mode before downloading the files. When you have them on your

target machine (emulator) just use Archiver 3.03 to unarchive them to the disks of your choice.

Now, the best for last: the site to ftp to is "gcomm.com". Enter 'anonymous' for user and your email address for the password. The files are located in "library\TI99/4A". This site does not have any upload access, so if you have anything you would like to share, you may have to email it to me. At some point I will create an index file with a list of the files and their descriptions. If you have any problems using the site, please let me know.

Joe Delekto

<idclckto@gcomm.com>

[Editors Note: These articles are great!]

Genial Travler now available on the net

Another Herculean task by Mike Wright! Mike contacted Barry Traver and received permission to convert

the ENTIRE collection of Genial Travler over to PDF's and PC99 and publish them! These can be retrieved, free of charge from Don O'Neil's ftp site ftp.whtech.com/pub/genitrav the two files, both self extracting PC archives, genitrav.exe (15 Megabytes) - a self extracting PC archive file with the text of all articles in Adobe PDF format and gtdisks.exe (2 Megabytes) all the disk in PC99 format! A big hat's off to Mike for rescuing these excellent articles and program from unavailability!

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@freewwweb.com

Secretary/Treasurer Greg Larson 783-45/5

email: greg.larson@icsbbs.org

MBX Website

Here is another interesting web site, with reviews, advertisements and manuals for the Milton Bradley MBX system (as well as other TI game related information)!

Http://www.robpatton.com/ti_mbx.htm

Also, you can download and play mpeg files of a number of cartridges at: ftp://robpatton.com/cvgac/ti994a/ -also- available at the same ftp site is a number of TI related ROMS.

TI Reflections

By Guntis Sprenne

My name is Guntis Sprenne, and I was lucky enough to work for TI's Computer Division as a Field Service Tech for 18 plus years. EICHER, after a few off-topic postings to the TI 99/4A listserver, asked that I write up a short article on working for TI and the 'other' TI computer equipment. The only warning I'll give is that most of this is from the top of my head and may not be all that complete, though it should be fairly accurate.

TI's computer group was known by many names over the years – DSD (Digital Systems Division); DSG (Digital Systems Group), and probably others. I started working for TI right out of college and was happy working on the hardware and software, and got paid every two weeks. So minor details like the official name of the part of TI I worked for weren't real important – until much later.

Anyway, TI's service group was fairly informal in the early years. I started in 1974 and was one the first field techs in the Southeast. I was something like the 14th or 17th in the southeast. Eventually there were upwards of 150 in the Southeast, though that number changed up and down during the years.

Once, while at training in Houston, we got to tour the plant at Stafford. We were shown some of the early TI minicomputers.

As we were told, the computer division started out because of TI's efforts to automate the semi-conductor production lines (which we also saw on our tour). TI started building machines controllers for the semi-conductor equipment, which evolved from dedicated hardware to programmable hardware – using paper tape probably though I can't remember what it really was. These machines were 'named' HSM-xxxx. Our guide told us that no one – except maybe deep in the design area

- knew what HSM stood for, though one of the guesses was High Speed Machine.

As the story goes, that machine evolved into the 960 and the 980 and was commercialized. Parts of the CRU bus and TILINE data/memory buses were evident in these machines. As this was before marketing hype took over, they had a maximum of 64K words of memory, though they used a 16 bit memory word. In today's terms it would have adddressed a max of 128K bytes. Clock speed I can't remember.

The 960's I worked on were used as data concentrators and machine tool controllers. I had one 980 in my area, but was not trained on it; it was used as a process controller in the making of nylon or rayon.

The basis of TI's commercial line was the 990 series. These were nice machines that used the CRU and TILINE. The CRU is a serial input/output mechanism, while the TILINE was a high speed parallel data bus. Since the CRU concept is used in the /4A I'll skip over most of that. Memory capacity was a maximum of 2 megabytes, with a 64K word 'page' limit due to the 16 bit data width. The 2 megabyte memory was addressed by using a paging mechanism that provided a 20 bit backplane addess bus.

One note about the CRU and the chassis. The slots were numbered 1 through 13 (or 6, or 17, depending on the chassis – I'll use the 13 slot chassis for this example). Each slot could hold a full size board (CPU, Memory, Disk Controller, etc) or two half slot boards along with a center support piece. The CRU address was hard wired to the slots, with most (there were exceptions) CRU boards using the slot address. Slot 13 right was address >00, left was >20; slot 12 right was >40, left was >60, and so on.

The TILINE was rather innovative. High speed devices – disk drives, tape drives, memory controllers, then in later years multi-terminal controllers – each had a dedicated memory address, which was selectable by a set of switches. Commands were passed to the controller by addressing (usually) a set of 16 'registers' in memory. Though the memory controller used only 1 or 2 words, as did the status register. These registers – starting at >F800 (TI used the '>' symbol to denote a hexadecimal number; using 378h or

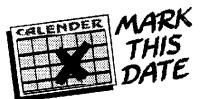
0x378 still seems odd to me) were mapped to high memory, so the theoretical 2 gigabyte limit was reduced by a little bit.

Memory was mapped in three segments, the total of which could not exceed 64K words. This did allow for some interesting programming, as the core of the progam could stay the same in one segment while it 'talked' to different areas of memory in the other segments. Remember I was basically a hardware person, though I was very familiar with the OS – I was one of the very few, if not the only, Field Tech to take the Tech Support O/S training class.

The first operating system was known as DX10. Our best guess for DX was Data eXchange I started with version 3, which had a friendly user interface. Version 2 – which I only saw once – made UNIX look user friendly. It was pre-emptive multitaksing. One of my early systems sustems supported about 8 users and 3 printers, had 128Kbytes of memory (later upgraded to 256Kbytes), 10 meg of disk storage, and did a small companies payroll, receivables, payables, inventory, order tracking, and a few other things. DX10 handled up to about 20 terminals with good response, maxing out around 40 or so, but then the response was fairly slow.

The next operating system was DNOS. For a change, we were told what it stood for - Distributed Network Operating System. It handled probably up to 50 users well. The downside was that with only a few users the response was the same as with 50. Part of the trade-off when selecting which O/S to use was determining the system load - up to a certain number, DX10 was better/faster. After that DNOS did better. And, as the name implies, DNOS had network support and could utilize resources on other machines, and users could log onto their machine, and start a session on another machine and have it appear as if they were logged onto that machine. Though it was nothing like the Internet is today.

Terminals were 'dumb' terminals with an 80 character by 24 line screen – what ever processing power they had (if any) was to control the display and the transfer of data. The original terminals – the 911and 9xx was only a power supply and video receiver. The character generator and video generator were located on the terminal controller in the CPU chassis. The 911 I/O cable was 8 wires – COAX, shield, and three sets of differential control lines using a 9 pin D connector (same as is used for serial ports today). The upside of this terminal was



Tenative HOOSIER USERS GROUP Meeting Schedule

May 16 July 18
September 19 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

the effective data rate was 19.2 Kbytes/sec. Then came serial port board was inserted in the slot, the jumper was removed, and based terminals - starting with the 931. These used a bit of smarts then the controller could take control of the TILINE to package the data coming and going, but still displayed only (memory/data bus) and transfer data straight to/from memory. text. Due to the advances in electronics, they were more The lower the board in the chassis, the higher priority it had configurable. The 924 and the 928 (an HP 700 series terminal for gaining control of the TILINE. with customer ROM) followed.

Another note is that the terminals, even once serial based, used a custom built for each customer. To those of you familiar proprietary protocol. Eventually TI released Terminal emulators with UNIX, the idea was the same, though the front end for the TIPC and for IBM/clone PCs, and at least 2 vendors defining ports, devices, etc - was much friendlier. A released their versions of TI emulators.

Now that I've coverd terminals, I'll back to the CPUs. TI had what could be called a 'beta' 990 CPU known as the 990/9. The original 13 and 17 slot chassis came with programmer's Housed in a multi-slot chassis, it was a TTL logic implementation front panels: 16 LEDs, 16 switches to toggle the LEDs, and of the same 990 instruction set used in the 99/4A home computer 16 switches for various data entry functions. It was possible (which was produced by the Consumer Electronics Group - I'm to read and write disk sectors using just the front panel by not sure if that was the official name - the same group that did entering the controller commands to the correct memory calculators, watches, Speak & Spell, etc). It saw work basically as locations. I had one customer, who while a good a data concentrator. In my area it was used by a national hotel programmer, would skip the testing phase when he generated chain to take the input from several terminals and send it out over a new system. I had to several times walk him through a leased line to the mainframe, where-ever it happened to be. This CPU had a power switch, and a recessed button to reboot. The system ROM contained enough code to download the rest of the O/S from the mainframe. There was no hard drive storage (this is back in the days when a 10 meg hard drive ran about \$10,000 to \$15,000 (more or less)).

The next was the 990/10. Again a TTL implementation, it was housed in a 13 slot chassis. The first two slots were the CPU (slot 1 was the SMI (System Master Interface) card, and slot 2 was the AU (Arithmetic Unit) card), next was the memory controller (16 K words of RAM with ECC - Error Correcting Code; it could correct a single bit error and detect multi-bit errors), then memory expansion boards. Slot 7 was reserved for the disk controller. Then the various terminal and printer controllers and I/O cards.

Back to the TILINE. The 990 chassis supported 16 vectored interrups. 0 to 2 were reserved for the system, 5 was the clock, and the rest were available for other uses, though DX10 and DNOS had standards that were usually followed: The primary disk controller, for example, was int 13. There was an interrupt configuration method where you could assign interrupts to slots. though usually the standard configuration was used. Another feature of the TILINE was direct memory access by the various TILINE controllers. One of the lines, equiped with a jumper, was the TILAG line - or TILine Access Granted line. When a TILINE

The operating systems - both DNOS and DX10 - were SYSGEN - SYStem GENeration - took anywhere from 20 to 50 minutes, depending on the configuration and CPU.

continues on next page

<u>Disclaimer</u>

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 alrequest for an article, mail it to:

Dan Eicher

4509 Northeastern Ave. Indianapolis, IN 46239

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

old version of the OS (at least he was good enough to keep the old system as the secondary).

The 13a chassis came standard with an operators panel which featured a four digit hex display with only 4 switches - top row: Halt, Load and bottom row: Run, Alt. Load. The RUN button is rarely used since the LOAD button performs the RUN function after loading. Hitting the HALT button (after the first HALT) increments the Program Counter (PC), which is where the RUN button comes in handy to resume operation at a new PC though this is only practical if the new PC is near and above the halted PC. There is no decrement PC from this front panel. With the programmers' panel you could enter the PC into the LEDs, then there is a button to enter the LED pattern into the PC. This way any PC could be selected. There were also buttons for reading/setting the WP (Workspace Pinter); reading the status register, etc.

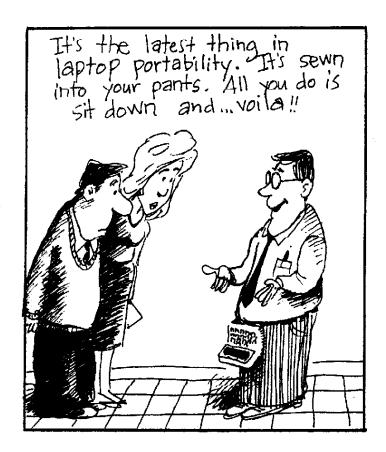
The four digit display helped users tell the service techs the error code. I had the misfortune of having a customer read an error code from a programmer's panel in hex which indicated a memory problem; unfortunately, the customer's binary to hex conversion skills were lacking, and it took me a while to determine that the problem was with the disk controller - of course it didn't help that the problem was heat related and the system had been off for several hours. The Operator's Panel also helped users with the Alternate Load procedure - no more remembering controller address, unit number, etc. Just hit a button and it loads (well, as long as the primary drive is Though techs then had to carry a spare programmer's panel to be able to do other low-level functions.

The OS and hardware also had built-in features for adding expansion chassis. Adding CRU chassis straightforward. CRU addresses just kept going up, and the switches on the expansion boards were easy to set (as long as you understood the CRU). TILINE expansion was another matter. On of my log-time customers was consolidating two systems. This included upgrading a CPU, moving all disk and tape drives to one system, converting the second chassis to a TILINE expansion chassis, and regenerating the system. When I called him to make final arrangements - we started at 4:00 PM on a Wednesday and had to have the system up and running by 8:00 AM the next morning - he told me there was

resetting the flag bit in the 'boot sector' that would boot the a problem. When I asked him what it was, he said the three different engineers had given him three different answers on how the expansion boards should be configured. And these were TI engineers. As things go, that was the hardest part, but we managed, and the working configuration was none of the above three. We even beat the deadline, though I watched the sun come up as I drove home.

> That's about it, though I didn't even mention TI's printers. terminals, laptops, disk drives (yes, TI made its own winchester drives for a short while), and other computers.

[Editor's note: for a list of CPU references, see insert]



Doom of Mondular Part 2 of 3 By Mike Wright of Cadd Electronics

180 CALL INIT::CALL LOAD(31804,0,36)

190 REM ZAP

200 RETURN NEXT

210 NS=NS&CHRS(16)&CHRS(17)::OPEN

#1:N\$,UPDATE, RELATIVE, INTERNAL, FIXED 255

211 INPUT #1, REC 062:N\$

CPU	Description		
HMS	Used as a mchine/process controller in at least the Stafford plant (outside of Houston). Early version of		
{	the 960 & 980, and eventually the 990.		
960	First commercial version. A CRU based computer.		
980	The close second commercial version. The TILINE (or a preliminary version) appeared here.		
990/9	'Beta' Version, saw limited commercial use, mainly as a terminal concnetrator. A TTL implmentation of		
[the 990 instruction set. The chassis was a motherboard/backplane – even the power supply (shileded)		
	plugged on to the backplane. No controls other than a power switch and system load button. Software		
	was downloaded from the mainframe - no hard drives. I don't remember much about this CPU		
990/10	First 990 commercial version. Also a TTL implmentation with the CPU taking up 2 boards. Memory		
	had ECC – Error Correcting Code – it could correct a single bit error and detect multibit errors.		
	Removeable media drives up to 200 megabytes (unformatted) were available. This was before		
990/12	winchester drives. Standard 13 slot chassis, a 6 slot chassis was also available.		
990/12	Bit slice processor implementation. 2 to 10 times faster than the /10, depending on configuration (hardware and software). Also a 2 board CPU Standard 17 slot chassis. Power supply was modular –		
	had +5vdc and +/-12vdc modules; depending on hardware a second or third module could be added (4		
	total – minimumu of 1 cach).		
990/4	Single-chip 990 implementation. I'm not sure if this is the same chip used in the 99/4A. No onboard I/O		
	or memory. Limited use. This was a single board CPU		
990/5	Single chip implementation (TMS9900), also a single board. Had built-in serial I/O (3 ports) and limited		
	on-board memory. Used the 6 slot chassis with disk controller for a complete, if small, system.		
990/10a	Single-chip implementation using the 99000 chip. Essentially the same as the 990/10 except for being a		
	single board and complying with then new FCC RFI rules. Used a newer version of the 12 slot chassis.		
	The /10a was about 1.5 to 2 times faster than the plain /10.		
990/12a	Bit slice processor implementation. Used the same 13 slot chassis the /10a did (see note below). This		
	was a 2 board CPU.		
77x	These were TMS 9900 based computers housed in a combination display/keyboard housing. The first		
	system used tape cartridges (similar in size to the QIC-2120) for program and data storage. Then came		
	versions with hard and floppy drives in an external case. And finally one in a chassis with a separate display. While similar to the 990s, it was slightly different.		
PCs	The TIPC was an Intel 8088 based PC, not totally IBM compatible. There was also a 'portable' version		
103	which was TIPC compatible. Next was the Business Pro, which was an IBM AT compatible that also		
	had the option of being TIPC compatible. After the D-Pro, TI started putting it's name on other vendor's		
	PC's (ACER and AST, maybe others)		
12xx/13xx	Intel CPU / XENIX based multiuser systems		
15xx	68xxx / UNIX System V based multiuser systems		
LISP	First a 68xxx based, then a dedicated LISP processor based CPU used for Artificial Intelligence		
	development. Everything – OS and applications – was LISP.		

Notes

This list is incomplete – this is just what I remember, and not even all of that.

If our guide at the Stafford plant could be believed (probably not) the HMS stood for "High Speed Machine".

All of the 990 series boards starting with the 990/10 were interchangeable – as long as the chassis could power it (the +5vdc module in the 17 slot chassis put out 40 amps!). Of course, using non-FCC boards in the 13a chassis violated some government rules concerning RFI – practically everything worked OK.

The 990 used a dual I/O scheme – the CRU and the TILINE. The CRU was serial based, while the TILINE was a parallel bus

212 PRINT #1, REC 0:N\$ 215CALLPEEK(31952, A, B)::CALLPEEK(A*256+B65534 , A, B) 220 C=A*256+B65534::CALL PEEK(4096,A,B)::CALL LOAD(C, 07)::CALL LOAD(C+6,A,B,0)::ON ERROR 200::GOTO 230 230 RUN "DSK1.AAAAAAAAAA"

There are a couple of tricky things going in this program:

Line 160: Loads the assembly language program BOOT. That meant we had to find the assembly code and disassemble it.

Lines 160 and 210: Manipulates the string N\$ which eventually contains the filename DSK1.ZAPSDATA(16)(17). The last two characters are non ASCII characters.

Line 180: 31804 = 83C4, the address of the userdefined interrupt.

Lines 211212: At this stage the seemingly innocuous INPUT and PRINT statements didn't look dangerous.

Line 220: Calculates the address of the highest line number in the program (230). 4096 = >F000. This address contains >0F0D and was stored there by the program BOOT. CALL LOAD(C,07) changes the length of line 230 to 7 bytes. The b ytes >0F, >0D, >00 are moved into line 230 so that is now reads: RUN "DSK1.(0F)(0D)"(00). (Note that the parentheses show unprintable ASCII characters.)

Line 230: Disguises the next program in the chain. This is a "standard" XB trick. You make the last line in the program RUN"DSK1.1234567890", and then CALL PEEK to find the Basic line number table and the line, and then you poke the filename into the bytes occupied by "1234567890" (or in this case "AAAAAAAAAA").

10. DSK1.BOOT

The last two program names had been LOAD, and GAME. This meant that the search for BOOT would be at entry 15 in sector 1. It was. We used dskoutx to extract the program and, since this was an assembly language program, we ran the PC99 utility eau2asc (Editor/Assembler uncompressed to ASCII). This utility does not, unfortunately, return a disassembled version of the file. However, the file was short enough to hand disassemble. We then entered the

disassembled statements as 9900 source code (BOOT S), added some comments and labels, and then assembled the file.

VERSION 1.2 99/4ASSEMBLER **PAGE 0001** 0001 * 0002 * Symbiotech boot loader

0003 *

0004 DEF BOOT

0005

0006 83C4 USRINT EQU >83C4 * address of

userdefined interrupt

0007

0008 F000 AORG >F000

0009

0010 FNAME

0011 F000 0000 DATA >0000

0012 UDIMSK

0013 F002 C200 DATA >C200

0014

0015 BOOT

0016 F004 0300 LIMI >0000 F006 0000

0017 F008 0200 LI RO,>0D0F* becomes fname 0F0D

FOOA ODOF

0018 F00C 06C0 SWPB R0

0019 F00E C800 MOV RO, @FNAME* save the filename

for Basic F010 F000

0020 F012 0200 LI RO, MYINT* put my interrupt

routine in UDI F014 F01C

0021 F016 C800 MOV RO, @USRINT F018 83C4

0022 F01A 045B B *R11

0023

0024 MYINT

0025 F01C D820 MOVB @UDIMSK, @>8345 F01E F002

F020 8345

0026 F022 02E0 LWPI >83C0* GPL workspace F024

83C0

0027 F026 0380 RTWP

0028

0029 END

0000 ERRORS

If you step through the code you will see that the value >0F0D gets stored at >F000. (We didn't appreciate the significance of this until much later.)

But even more devious is the setting up of a userdefined interrupt routine at >F01C (label MYINT) by plugging this value into >83C4. This routine will now move the byte value >C2 at >F002 into >8345 every 1/60th of a second. >8345 is used by Extended

Basic as a system flag. The flag bits are:

0 = 1 = autonum

l = l = on break next

2 = unused

3 = 1 = trace

4 = 1 = edit mode

5 = 1 =on warning stop

6 = 1 =on warning next

7 = unused

So the value >C2 (1100 0010) sets autonum on, ON BREAK NEXT, and ON WARNING NEXT. This means that even if you think you are defeating the protection by removing the ON BREAK NEXT in the Basic code, if you allow this routine to be called, ON BREAK NEXT will be set until the computer is switched off (or until the userdefined interrupt is moved or changed) and you will not be able to press F4 (BREAK) to see how the program is executing.

11. DSK1.ZAPSDATA

The first thing to remember about this filename is that it has two non ASCII characters appended to it to become DSK1.ZAPSDATA(16)(17). Since the string "ZAP..." is greater than "LOAD", XB will look ahead in sector 1 to find it. Sure enough, at entry 63 + 32 = 95 there is a pointer to sector >0087 (0135). At that sector there is an FDR for ZAPSDATA(16)(17)

The next problem now had to be solved, dskoutx, which was based on dskout, assumes legal filenames. There was no way to enter this name on the command line. There may have been a more elegant way of doing this, but we resorted to simply hardcoding the filename in the program and recompiling. (We ended up doing this a number of times). The extracted file was large, and didn't seem to conform to any recognized structure. It contained FDRs, pieces of the uninitialized disk, and a mixture of Basic and assembly code. Something was not right. But according to the Basic code in DSK1.GAME, record 062 of this file was to be read, and then written to record 0. We then examined the data chain pointer of the FDR more closely. It contained >01, >40, >16. Using the swap trick above, this meant the data for this file began in sector >001 and continued for >164 sectors! We had stumbled across something that we didn't even know could be done in Basic. The statement:

OPEN #1:N\$,UPDATE,RELATIVE,INTERNAL,FIXED 255

with an FDR as set up above, essentially allows Basic to read and write disk sectors!

So this meant that the program was going to read record 062 (= sector 63) and write it to record 0 (= sector 1). But... sector 1 is the disk's File Descriptor Index Record. This meant that all we had learned about the placement of the files was now invalid. It was a new ball game.

12. Time for a regroup

We now decided that things were getting devilishly tricky. So we took a notebook, the dump of the disk, and made a list of what we thought each sector contained. Using the information we had already deduced, we flagged dummy FDRs, bad sectors and so on. The result of this investigation was to find some sectors that looked suspiciously like FDRs, but didn't have any ASCII in their filenames.

The FDRs found were: 0F0D at >0048 (0072) 120D at >0077 (0119) 1B18130d at >007D (125) 1B13180D at >0081 (0129)

We went ahead and extracted them by hardcoding the names into dskoutx and recompiling the code. They all turned out to be Basic programs.

13. Listing for 0F0D

100 GOTO 120 !COPYRIGHT 1984 SYMBIOTECH, INC

ALL RIGHTS RESERVED

110 REM ZAP

120 ON BREAK NEXT::ON WARNING NEXT::GOTO 140

130 REM ZAP

140 ON ERROR 260::OPEN

#1: "DSK1.TITLEDATA" & CHR\$ (16), INPUT, INTERNAL, RE LATIVE, FIXED 255

150 GOSUB 420::CALL

CHAR(092, "919D455D21110C03C0308884324281B9A9B9

AABA848830 C0030C11214C42819D")::N\$="DSK1."

160 FOR R=2 TO 20::RN=R2::INPUT #1,REC

100 10K K-2 10 20..KN-K2..INF01 #1,K

RN:X\$::DISPLAY AT(R,1):X\$::NEXT R
170 GOSUB 290::CALL

COLOR(09,02,15,13,14,15,14,15,02,09,14,15,11,1

5,14)::N\$=N\$&"ZAPS DATA"

180 FOR C=2 TO 7::CALL COLOR(C,02,14)::NEXT C

190 DISPLAY

```
AT(07,09)STZE(14):"LN+DF+HJ+8:+46"::DISPLAY
AT(00,09) SIZE(14). "MO!EG!IK:9;:57"
200 DISPLAY AT (09,14) SIZE(2): "PR"::DISPLAY
AT(10,14) SIZE(2):"QS"
210 DISPLAY AT(11,10) SIZE(11): "46+DF +DF+
<>"::DISPLAY AT (12, 10)
SIZE(11): "57+EG+EG+=?"
220 DISPLAY AT (15,14) SIZE(2):" ] "::DISPLAY
AT(16,14) SIZE(2):"\^"::GOTO 240
230 REM ZAP
240 CLOSE #1::CALL LOAD("DSK1.BOOTS")::CALL
LINK("BOOTS")::N$=N$&CHR$(16)&CHR$(17)::GOT
0 390
250 REM ZAP
260 CALL INIT::CALL LOAD(31804,0,36)
270 REM ZAP
290 CALL
CHAR(52, "007F387C383838383838383838387F0000
EOF 0381C0E060606060E1C3870E000")
300 CALLCHAR (56, "007C387C38383838383838383
8387F00000000000000000000008183878F800")
310 CALLCHAR(60, "007E3F67232120202020202020
20700007EFC1E1C1C1C1C1C1C1C1C1C1C3E00")
330 CALLCHAR(68, "000F183C38383838383838383838
180F000F0183C1C1C1C1C1C1C1C1C1C1C18F000")
340 CALLCHAR(72,"007F387F3F3B3B393938383838
387C000F838F0E0808080808080804040201000")
350 CALLCHAR(76, "007B30783030303133363C3C3C
3C18000DE040E040484C464341C1C1C1C0800")
360 CALLCHAR(80, "000000007E42424242427E0000
0000000000007E40407C40404000000000000")
370 CALL CHAR(43,"00000000000000000")
380 RETURN
390 OPEN
#1:N$,UPDATE, RELATIVE, INTERNAL, FIXED 255
391 INPUT #1, REC 064:N$
392 PRINT #1, REC 0:N$
393 CALL PEEK(31952, A, B)::CALL
PEEK (A* 256+B65534, A, B)::C=A* 256+B65534::CAL
L PEEK(3840, A, B)::CALL LOAD(C, 7)::CALL
LOAD (C+6, A, B, 0)
400 ON ERROR 410::GOTO 480
410 RETURN NEXT
420 CALLCHAR(128,"FF3F0F0300000000FFFFFFFFF
F3F0F03FFCF0C000000000FFFFFFFFFFFFCF0C0")
430 CALL CHAR(136, "FFFFFFFFFFFFFFFFFF")
440 CALL CHAR(096, "FFFFFFFFFFFFFF")
450 CALLCHAR(112,"FFFFFFFFFFFFFFFFFFF3F0F030
460 CALL CHAR(116, "FFFFFFFFFFFFCF0C0")
```

470 RETURN
480 RUN "DSK1.222222222"

Here are some of the highlights of 0F0D:

Line 140: The file TITLEDATA(16) has a nonASCII (16) character appended to it.

Line 240: There is another assembly language program called BOOTS to be loaded.. N\$ eventually contains DSK1.ZAPSDATA(16)(17).

Line 260: Causes a jump to CPU address >0024, which is the start of the console powerup routine. This line is executed if there are any errors because of ON ERROR 260 in line 140.

Line 391: For the second time, a hidden File Descriptor Index Record is retrieved and overlaid on sector 1. This time the source record is 064 (sector 65).

14. Trying a first run

At this stage we didn't feel like spending the time to figure out the file loading sequence. So we guessed that DSK1.GAME called DSK1.0F0D. We now started to make a fresh disk, and used the Disk Manager to copy the files that we had managed to extract. We also changed the Basic and got rid of things like ON BREAK NEXT, REM'ed out things like CALL LOAD("DSK1.BOOT"), and changed the RUN DSK1.XXXX to real filenames. So now we were able to deal with things like TITLEDATA, and not TITLEDATA(16)(17). We also REM'ed out all CALL LOADs that would return to the title screen. We also renamed the LOAD program to ZLOAD so that it would not be autoloaded by XB.

We now started XB, entered OLD DSK1.ZLOAD, and RUN. At this stage we got a black screen with "PRESENTING" and then from 0F0D we got the Doom of Mondular title screen.

16. DSK1.BOOTS

It was now time to deal with DSK1.BOOTS, called out in DSK1.0F0D. Fortunately this turned out to be a clone of DSK1.BOOT. The only difference was that the code was AORG'ed at >F100 and that the hidden file name was >120D.

17. Listing for 120D

100 ON BREAK NEXT::ON WARNING NEXT ! COPYRIGHT 1984

SYMBIOTECH, INC. ALL RIGHTS RESERVED

110 CALL PEEK(31952, A, B)::CALL PEEK (A+ 256+D65534, A, D):: C-A+ 256 | D65534 111 CALL CLEAR::CALL CHARSET::CALL SCREEN (04) 112 CALL CHAR (112, "000000784478404000000038445448340 0000078447848440000003C40380478"1 113 CALL CHAR(116, "0000007C10101010")::N\$="DSK1." 120 CALL CHAR(62, "3C4299A1A199423C000000000081423C00 0000FFFF000000")::N\$="D\$K1." 125 DISPLAY AT(1,8): "WORLD OF DOOM"::TAB(7); "regular version"::TAB(6);"Doom of Mondular" 130 DISPLAY AT (08,01): TAB (7); "Copyright >1984":TAB(5); "ALL RIGHTS RESERVED": TAB(6); "BY SYMBIOTECH.INC" 160 DISPLAY AT(13,1): "This program is protected under the laws of the United States and other countries," 170 N\$=N\$&"ZAPSDATA" 180 DISPLAY AT(16,1): "and illegal distribution may result in civil liability and criminal prosecution." 190 DISPLAY AT(24,1):"Please wait....." 200 ON ERROR 230::CALL LOAD ("DSK1.BOOTER")::CALL LINK("BOOTER")::N\$=N\$&CHR\$(16)&CHR\$(17)::GO TO 270 210 REM ZAP 220 REM ZAP 230 CALL TNITT::CALL LOAD (31804,0,36) 240 REM ZAP 250 RETURN NEXT 260 REM ZAP 270 OPEN #1:N\$,UPDATE, RELATIVE, INTERNAL, FIXED 255 280 INPUT #1, REC 058:N\$ 290 PRINT #1, REC 0:NS 300 CALL PEEK(10240, A, B, D, E)::CALL LOAD(C,9)::CALL LOAD(C+6,A,D,B,E,0)::ON ERROR 250::GOTO 320 310 REM ZAP 320 RUN "DSK1.??????"

We were still not fully comfortable, but it was starting to look a bit like what had gone before.

Line 200: Loads the assembly language program DSK1 BOOTER. Line 280. Yet another switch of sector 1. This time it is overlaid with sector 59.

Line 300: 10240 = 2800. This contains the bytes for the next hidden filename (1B13180D). These values were stored by the assembly langauge program BOOTER.

18. DSK1.BOOTER

This loads the filename 1B13180D at >2800 in low memory. It also resets the user defined interrupt to execute from >292C, which performs the by now standard practice of loading a >C2 into >8345.

19. DSK1.1B13180D

100 GOTO 110 !COPYRIGHT 1984 SYMBIOTECH, INC. ALL RIGHTS RESERVED 110 ON BREAK NEXT::N\$="DSK1." 120 ON WARNING NEXT 130 ON BREAK NEXT: N3-N\$&"ZAPSDATA" 140 ON ERROR 200::CALL LINK("BOOTER")::N\$=N\$&CHR\$(16)&CHR\$(17)::CALL LOAD (31806, 16) 150 CALL PEEK(31952, A, B)::CALL PEEK(A* 256+B65534, A, B)::C=A* 256+B65534::CALL PEEK(10240,A)::A=A+6 160 OPEN #1:N\$, UPDATE, RELATIVE, INTERNAL, FIXED 255 161 Y\$=""::Z\$="019024013032032032032032032000000130001 0000862102540860000000000000000000000000018708000 5000000" 162 FOR I=1 TO LEN(Z\$) STEP 3::Y\$-Y\$@CHR\$(VAL(SEG\$(Z\$,I,3)))::NEXT I 163 PRINT #1, REC 355:Y\$&CHR\$(0)::CLOSE #1::GOTO 220 200 CALL INIT::CALL LOAD(31804,0,36) 210 RETURN NEXT 220 CALL PEEK(10240, D, B, D, E)::CALL LOAD(C, 9)::CALL LOAD(C+6,A,B,D,E,0)::ON ERROR 210 230 RUN "DSK1.??????"

Stayed tuned for the conclusion in the next issue!



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

Forwarding and Address Correction Requested

Next meeting

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

Spone:	E-mail:
Address: City, State, Zip Phone:	
Address:	
лате:	

Dan H. Eicher 4509 Northeastern Ave. Indianapolis, IN 46239

Hoosier Users Group

completed application to:

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

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 HUGbbs, 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.