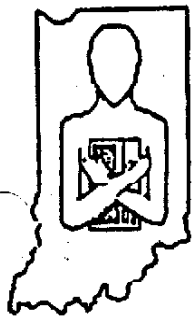

HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP
USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER
GROUP HOOSIER USERS THE HUGgers GROUP HOOSIER USERS
HOOSIER USERS GROUP HOOSIER USERS GROUP
USERS GROUP HOOSIER HOOSIER USERS GROUP USERS GROUP HOOSIER
GROUP HOOSIER USERS People Helping People GROUP HOOSIER USERS
HOOSIER USERS GROUP HOOSIER USERS GROUP
USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER
GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS

AUGUST 1992 The HUGgers Newsletter Volume 11, Number 8

## OFFICERS' CORNER

UFF-DA! (this is a Norwegian technical term)
As you may have noticed, we are having a difficult time getting a newsletter out these days. In a nutshell, we have had problems with our copier at the same time as Gary Jones, with whom we have our service contract, has been out of town for an extended time due to an illness in the family.

Just how much you like BASIC programming articles and typing in code, or music programming, for that matter, will probably determine how well you like this month's (actually last month's, but you know what I mean) newsletter. The reason for the admittedly odd mix is we have to use what we've already copied to paste up the newsletter. As soon as the copier gets fixed, we'll get caught up in the newsletter dept.

In July we had our meeting outdoors. It turns out that St. Ann's is doing some work on the little white house where we meet and one of the things they did was change the locks. Bob Stahlhut reports that we can still meet there but as part of the restoration, St. Ann's has made it a SMOKE FREE ZONE. I wonder how many will be outside for the August meeting too?

We will need to talk over the status of the bulletin board at the August meeting. We've had to help John Powell out on the phone bill a couple of times this year to keep the board online. By meeting time we'll know more about what's what. We may have to decide whether we can afford a full-time bulletin board.

I guess this is the "goodbye letter" newsletter. Bob Stahlhut is printing them up even as I write this. In the past we've sent three issues of the newsletter to members after their expiration date. We haven't been sending out the "goodbye letters" like we used to because the wording was obsolete. Bob's put together a new letter, so if you haven't paid your dues yet it will be in your newsletter. So far, we have 21 paid members for '92-'93.

Upcoming events of interest are: The Dayton Computerfest on Aug 29 & 30. This is like a hamfest without the ham radios. Although they had quite a few user groups set up last year, I didn't see any TI groups. So far, it's got a pretty good reputation going. If you don't want to travel that far, Sept 6 (Labor Day weekend) is the date for the Central Indiana Hamfest/Computerfair at the State Fairgrounds (8am - 4pm, $4).

Of course, the big event for the Fall is the Chicago User Group's TI-99/4A Faire. This year it will be held one week earlier than previously. It used to be the first Saturday in November; this year it's the last Saturday in October (10/31/92). The Milwaukee Faire is always the next day on Sunday. Are they trying to tell us something scheduling it on Halloween?

-GBL

### HUG OFFICERS

| | | |
|---|---|---|
| President | Gregory Larson | 783-4575 |
| Vice Pres | Bryant Pedigo | 255-7381 |
| Secretary | | |
| Treasurer | Fred Edstrom Jr | 898-7300 |
| Librarian | Bryant Pedigo | 255-7381 |

PROGRAMMING MUSIC THE EASY WAY

PART 4

by Jim Peterson

The first three parts of this series were written and published some time ago, so I had better review.

In Part 1, I showed you this one-line routine to set up a musical scale.

```
100 DIM N(36):: F=110 :: FOR
 J=1 TO 36 :: N(J)=INT(F*1.0
59463094 (J-1)+.5):: NEXT J
:: N(0)=40000 ::GOTO 110
101 D,T,A,B,C,V1,V2,V3,J,X,V
102 CALL SOUND
103 !P-
```

That sets up a scale of three octaves beginning with A. If you decide to change the music to a higher key, just change the 110 to 117, 123, 131, 139, 147,156,165, 175, 185, 196, 208 or 220. In fact, for some music you will have to change it, if the program crashes with a BAD VALUE error message.

If you have programmed the music with high notes, you can lower the key by changing 110 to 104, 98, 92, 87, 82, 78, 73, 69 or 65. Again, if you try to go too low you will get that BAD VALUE message.

I have given N(0) a value of 40000, which creates a tone too high to be heard. This can be used to silence a note, but it can also cause a crash when used with some of the following routines. If you are programming three voices and want to play a single note, the easiest way is to give all three notes the same number, such as A,B,C=10. If you need a silent rest, play all the notes at an inaudible volume by V1,V2,V3=30 and then, after the GOSUB, restore their original volume by V1= (whatever is in line 110) and the same for V2 and V3.

Lines 101-103 are a pre-scan routine to start the music playing sooner. There will still be a few seconds delay while that array is set up in line 100. You can perhaps shorten that delay slightly by changing the 36 to the highest note number you have used in programming your piece.

However, Bruce Harrison wrote for me an assembly link which eliminates any delay; this also makes it possible to change key while the music is playing. I won't list the source code here, because everyone is afraid to key in source code anyway, but it is available on my TI-PD disk #1143 and will also be on a tutorial disk on this type of music programming.

Part 2 of this series contained a listing of a program to easily give you the numbers you would need in order to key in a particular piece of sheet music. If you don't have that, you can just take a piece a paper and list the scale A Bf B C C# D Ef etc., on through as many as you will need, and then number them consecutively. For the length of the notes, give the shortest note a value of 1 unless it also appears as a dotted note, in which case it must be 2, and then number the others according to their relative length - for example, 2 for a quarter note, 3 for a dotted quarter, 4 for a half note, 8 for a whole note.

Part 2 showed you how to key in single-note music, and Part 3 showed how to do 3-part harmony. To recap briefly -

First, save yourself a lot of work by identifying any groups of notes in the sheet music that are repeated two or more times. Mark them off wherever they appear. Key them in first, starting with line number 500; at the end, put RETURN. If you find another such series, label it 600 and do the same; you may find several such series. Just stay below line number 1000, which is reserved for mergeable routines. Then, while you are programming the music and come to such a series of notes, just put in GOSUB 500 or whatever.

Start keying in your music in line 120; line 110 is reserved for a line to be merged in. To key in the music, just give T the number for the length of the first note, and give A, B and C the numbers for the melody and first and second part harmony. Then GOSUB 1000. For instance, T=1 :: A=23 :: B=18 :: C=12 :: GOSUB 1000 .

And for each succeeding note, give a new value to whatever changes; if T is still 1 and B and C are still the same,

all you need is. for instance. A=19 :: GOSUB 1000.

Merge in one of the following routines, put in a line 999 STOP, and after every several notes enter RUN and listen to what you have done so far, to catch any errors while it is still easier to find them.

You can merge in any of the following routines to create many different musical effects. The D in line 110 controls the tempo of the music; change it as you wish. V1, V2 and V3 are the volume (loudness) of the three voices; adjust them as you like.

Key this in and save it by SAVE DSK1.

```
110 D=500 :: V1=1 :: V2=5 ::
V3=7
CALL SOUND(D*T,N(A),V,N(B),V
,N(C),V):: RETURN
```

That plays simple 3-part music, all at the same volume, which may sound rather harsh to your years. Try changing the second V to V2 and the 3rd one to V3. Save that as PLAY2.

For a bass accompaniment in the 3rd voice, change that to

```
CALL SOUND(D*T,N(A),V1,N(B),
V2,N(C)*3.75,40,-4,V3)
```

For a bass melody with accompaniment, change the A to C, V1 to V3, C to A and V3 to V1.

For the melody in two voices two octaves apart, change the C back to A and the V3 back to V1. Are you beginning to see how many different effects can be created by making changes in just this one line? Save any ones you like in merge format with a different name for each.

Perhaps those bass notes sound too deep. Try changing the 3.75 in any of those routines to 7.5 . Better yet, change it to X and add :: X=3.75 to line 110. Then you can switch back and forth in your music by simply X=7.5 or X=3.75. Getting interesting, no?

Music played in that way has a strong throbbing beat, so try this method -

```
110 D=4 :: V1=1 :: V2=5 :: V
3=7
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: NEXT J :: RETURN
```

I'll be referring back to this one as the negative duration method. Again, you can change the tempo by changing the value of D, but sometimes not as exactly as I would like. With this method, you will find that a series of the same note runs together into a single long note. To avoid this, use different harmony notes each time, or different volumes for V2 and V3.

There's no law that says the harmony has to be lower than the melody, so try changing N(B) to N(B)*2 or even N(B)*4 or do the same with N(C), or both. Or, use *X, add X=1 to line 110, and then in the middle of your music program you can switch by X=2 or X=4 (don't try 3!)

For a vibrato effect, we alternate a note with the same note multiplied by 1.01 -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: CALL SOUND(-4250,
N(A)*1.01,V1,N(B),V2,N(C),V3
):: NEXT J :: RETURN
```

For vibrato in the harmony rather than the melody, multiply N(C) or N(B), or both, by 1.01 instead - or multiply all three.

For a stronger vibrato, change the 1.01 to 1.02 or even 1.03. Of course, you can also multiply the harmony notes in both CALL SOUNDs by 2 or 4, as above. Or for a "chop" effect, multiply them in one CALL SOUND but not the other. The possibilities are almost endless!

For a tremolo, we alternate the volume rather than the frequency. Add X=3 to line 110 and use this routine -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: CALL SOUND(-4250,
N(A),V1+X,N(B),V2,N(C),V3)::
 NEXT J :: RETURN
```

You can vary the value of X as much as you want (V3+X can't total more than 30) for any amount of tremolo from a flutter to a wobble or a stutter, and

you can put the +X after V1 or V2 or all three. You can even change it in the middle of your music, by X= whatever you want.

And you can multiply any or all by 1.01 for different combinations of vibrato and tremolo.

To enhance a note, play it twice in the CALL SOUND but multiply one of its voices by 1.01 -

```
110 D=4 :: V1=1 :: V2=5 :: V
3=7
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(A)*1.0
1,V1,N(B),V2):: NEXT J :: RE
TURN
```

Of course, with this trick you can only have 2-part harmony, but you can choose to enhance the harmony rather than the melody.

Now, try combining the enhanced note with the vibrato and/or tremolo, for many more effects. For enriched vibrato, use N(A),V1,N(A)*1.01,V1 in the first CALL SOUND and N(A)*1.01,V1,N(A)*1.02, V1 in the second.

The bass notes do not go well with this method because interrupting them through a loop introduces a rattle, but the baritone works well and gives a unique reedy sound. To do this, place the note you want in the 3rd position, multiply it by 7.5, give it a volume of 30, and add the -4 noise at whatever volume you want. You can also combine this with other effects. for instance with vibrato

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C)*7.5,30,-4,V3)
1010 CALL SOUND(-4250,N(A)*1
4,V3):: NEXT J :: RETURN
```

Now for the real fun - the "piano" effects that we get by decreasing the volume gradually. This is the basic routine -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J+V1,N(B),J
+V2,N(C),J+V3):: NEXT J :: R
ETURN
```

Of course, with all of these you must also have that line 110 to define the duration and volume.

If you want a little more percussion in your piano, try this -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J*1.5,N(B),
J*1.5,N(C),J*1.5):: NEXT J :
: CALL SOUND(-4250,N(A),15,N
(B),15,N(C),15):: RETURN
```

And, of course, all those tricks we learned above - vibrato, tremolo, baritone, enhanced, high harmony, chop - can also be used with piano. This will give you the vibrato -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J+V1,N(B),J
+V2,N(C),J+V3):: CALL SOUND(
-4250,N(A)*1.01,J+V1,N(B),J+
V2,N(C),J+V3):: NEXT J :: RE
TURN
```

And an increasing tremolo -

```
1000 FOR J=1 TO T*D :: V=J*2
 :: CALL SOUND(-4250,N(A),J,
N(B),J,N(C),J):: CALL SOUND(
-4250,N(A),V,N(B),V,N(C),V):
: NEXT J :: RETURN
```

And just one more, the "reverse piano" with an increasing volume -

```
1000 FOR J=T*D TO 1 STEP -1
:: CALL SOUND(-4250,N(A),J+V
1,N(B),J+V2,N(C),J+V3):: CAL
L SOUND(-4250,N(A),J+V1,N(B)
,J+V2,N(C),J+V3):: NEXT J ::
 RETURN
```

By the time you get through exploring all the possible combinations of those, you should have a hundred ways of making music. Save each one you like, complete with line 110, in merge format, so you can try them all with each piece of music you create.

I had intended this to be the last part of this series, but I still haven't told you about autochording, so there will have to be one more.

## EXTENDED BASIC - STILL A GOOD CHOICE!

### BY ART BYERS 99PC

There are new guys in the 99/4A neighborhood. The order in which they have arrived is: FORTH, PILOT and SMALL C. They have lots of adherants who talk about "Freedom" (FORTH),"Simplicity" (PILOT), and "Speed and structure" ('c'). They are Compiled languages which means they certainly run much faster than old friend XBasic. SOOOoooo? Why bother with Extended Basic at all? Why not go with the New? The Better? The Faster?

One of the great things about our beloved 99/4A is that even with its limited memory, it CAN support FORTH and C and PILOT. I consider any of the computer languages that will accomplish what is needed to be fine! For me, however, Extended Basic still remains the easiest and best, especially when coupled with Assembly Language subroutines that speed up often used important areas.

Let me try to lead you through a discussion of the many advantages of Extended Basic without "putting down", in the slightest ANY other language for the 99/4A (including Pascal -However Pascal requires a special PEB card and those are hard to find and some early versions have bugs).

Extended Basic has many advantages from a programmer's viewpoint, not the least of which is that it is an interpreted language with a plethora of error degugging routines built in. One of the real swift pains in the neck of a compiled language is that if it is compiled containing errors or bugs, these are extremely difficult to find. This does not mean they cannot be found or that good programmers cannot produce error free compiled code. It is just that debugging, adding to, subtracting from, changing code, etc. is much easier with XB. It is a shame tht TI chose to make XB a "double" interpreted language by writing parts of it in GPL, TI's formerly secret proprietary language. (Which to the best of my knowledge they have NEVER released and should they have chosen to take legal action, they could make trouble for those who have violated their rights by selling GPL programs, books explaining GPL, etc. and etc.). It would have been better if the interpreter had been written in Assembly al la MYARC's XB. The added speed of MYARC's XB is an improvement over TI's XBasic.

To continue, let us discuss the subject of 'speed'. As in any computer, all languages eventually end up being put into machine language in order to be run. For the 9900, the fastest language and one that can do anything and everything that is possible with the computer is, naturally, Assembly - the instruction set of the chip itself. The compiled languages, listed above, allow the use, where the programmer desires, of Assembly language routines. The reason these Assy routines are used is that the compilers do not allow full access to the instruction set, but only use a limited part. This restricts them from equaling true assembly speed, nor is the compiled code written in such a way as to be congruent with the most efficient way of writing Assembly. Where maximum speed is needed, XBasic can also use Assembly routines. In this respect it is the equal of the compiled languages.

One of the biggest advantages of XB is its EASE OF USE AND UNDERSTANDING. BASIC itself was written just for that purpose. It is suppled with such popular computers as Apple, Atari, Commodore, and IBM. This ease of use was most important in bringing better understanding of computers and use of computer languages to large numbers of Americans. We must never forget the KISS principle,( Keep It Simple, Stupid!). For no other reason, the Basic language continues to survive.

As far as the 99/4A goes, another advantage is that the language itself resides outside the RAM areas. It is in ROM and GROM. The cover of the XB manual states that the module contains "32k bytes of preprogrammed memory". Most of the RAM is free. Additionaly,

XB accesses, again with simplicity, clarity and ease, the built in ROM routines such as Device Service (printers, disk drives, modems), screen access and display, setting up of buffers, graphics and sprites, mathematics, etc. Many of the "new" languages save RAM memory by also accessing these same ROM routines which run at about the same speed!

Where blinding lightning speed is required, Asembly programers will write their own Device Service Routines, often accesing the several chips, such as the 9918 (Video display) or the chips in the RS232 card, directly and bypassing the ROM/GROM routines many of which are written in GPL. More often, however, many assembly programmers access the same ROM/GROM routines to save writing code because they don't require the extra speed.

This business of speed, in general, I consider to be highly over rated. For most purposes Extended basic is more than fast enough. Forth, c, or Assembly may pop a full screen of text into view in less than half a second - but what matter if it takes a few minutes for the user to read the text. XB can fill a screen with text in less than two seconds. Surely that is fast enough.

The areas where speed is vital, such as lengthy math calculations, sorting of numerical or string data, searching, etc. are the areas where ALL the languages of the 99/4A most often tie in to Assembly routines.

You may now point out, and with justification, that if I stress the ease of understanding and simplicity of XB, I am ignoring the difficulty of writing Assembly routines. How can the average XB programmer (as apposed to an advanced programmer in general) be expected to write Assembly code.

The answer is that they are not expected to write it, no more than they were expected to write the code in the XB module itself. Today, there are many sources of routines that can be loaded into memory and accessed via "CALL LINK". The programing group led by Barry Traver of Philadelphia has not only put together these routines under the catch name of XXB, (Extended, Extended Basic) but they have devised a way of putting them into memory image (program) form to be loaded simultaneously into memory along with any XB program you have written. Several other Commercial and Fairware authors offer similar material. In the critical areas, XB can use assembly routines to bring its speed to very acceptable levels.

Now lets talk about available memory. Because support for Forth and C must be loaded into the main 32k memory area, they do not have as much memory available as some programmers feel is absolutely necessary. This problem has been solved by using virtual memory - that is disk storage of Forth screens (blocks) or C support routines. XB support resides in console ROM and the module itself, the full 24k upper RAM is available for programs and the 8k low memory for Assembly support routines. As an example, I recently purchased a Disassembler which was written in Forth. The author plainly stated that because of the memory used by Forth itself plus the program, it was not feasible to dissassembly programs from RAM. It did its dissassembly right off the disk!. Obviously the program could read ROM as well as the RAM/ROM set aside for module locations. Not Forth, Pilot or C can match the XB module's extra 32k of support within the memory map!

Some last points! let us look at what we have to work with. We have a machine designed as a HOME computer. For almost every purpose or use at home, memory and speed available through XB are more Than sufficient. We are not tracking satellites, doing high order lengthy math, searching a database the size of the national Social Security register. We have a hundred or so names on our phone list. We do not require massive spread sheets. For our normal practical purposes XB and the 99/4A can suit our needs. In fact I may be accused of HERESY, but I did almost everything with only the XB module and cassette - NO memory expansion or disk!!!

What is more, when I need a special program written to fill a personal need, I write it, dubug it and am using it in a matter of a few minutes to at most an hour. This is possible because the most frequently used XB GOSUB routines and CALL SUBs are saved on disk as MERGE files ready to be placed into a program, easily and quickly. Many programmers overlook this useful feature of XB.

Anyone wishing to ask questions or discuss XB programming with me, please leave me a message on the Forum. If there is enough interest, we can even scedule a formal conference or two on XB programming.

ARTBYERS 99pc.
**********

<END>

## 9640 NOTES

I would like to post a few observations on mdos .97H and the way directory creation is handled on floppys.

1. You can have at most 3 subdirectorys on a floppy disk.

2. Subdirectorys can not be nested.

In the PC World a major bug-a-boo has been handling directorys. If on an XT you do a dir on a floppy drive, the drive will always power up and read the directory before presenting you with the information. This is done to insure that what you are readying is in fact the information in the drive (ie... you didn't change floppys...). On an AT a more sophisticated hardware scheme is used. The AT has a line that goes to the controller card and then to the drive called drive change.... This line tells the computer if the disk has been changed.... So, what this means, if you have an AT class machine and you have just powered up. You do a DIR on the directory. The machine will go out and read it to see what is there and display it for you. Now, you can do directorys all day, and the computer will display the directorys from memory. If you swap out the floppy with another, the machine is smart enough to see the drive change line has changed states, the machine will go out and refresh its directory_information for that NEW floppy. Subsequent reads will be from memory only until the drive is changed again. This procedure works well for the PC world.

In walks MDOS. MDOS is neither smart enough to monitor the drive change line or read from the floppy on each access. What I had happen a couple of time while backing up my hardrive to floppies using MDOS (Why, because a good backup utility to floppys doesn't exist, yet) was that if you create a couple of subdirectorys on a floppy. Change floppys and attempt to create another subdirectoy, MDOS will corrupt the bitmap. My advice to everyone is: BEFORE CREATING A SUBDIRECTORY ON A FLOPPY, PERFORM A DIR.....

These observations could have allowed me to draw the wrong, conclusions. But thats what happens in the light of no support or technical documentation from the manufacturer.....

dan

## ANNUAL PICNIC ?

In last month's newsletter, the idea of having a picnic was mentioned as a possibility on our regular meeting date in August. At this point plans for an "annual picnic" in August have been dropped. However, if there is sufficient interest, one could still be planned for sometime in September. Anyone interested should mention at the August meeting or otherwise let one of the officers know that they would like one to be planned.

-BCP

---

BBS
Hoosier Users Group
Baud rate 300,1200 & 2400
On Line 24 Hours Daily
782-994A

# TI 99/4 - THE EARLY DAYS

by Chuck Neal
PORTLAND USERS NINETY NINES

*Here is an interesting bit of TI history that came over the UNIX user net at work. I rearranged some of the material for clarity. The contributor of thi article was Herbert H. Taylor of the David Sarnoff Research Center. It appears that he worked at Texas Instruments back in the late 1970's when the TI 99 "was happening."*

*First you will find out what the "pack-man door" on your speech synthesizer is for (was for!). Next he talks about some mysterious "wireless peripherals" that the TI 99/4 almost had. Then the replacement of the 9985 microprocessor by the 9900 and the suggestion that there are a lot more interesting stories waiting to be told!*

*The last part of the article goes into the history of the RF modulator. In the early days you had to hook your TI into a television set. And television sets don't have audio and video inputs - they have radio frequency inputs. (Except for some of the new ones.) So the RF modulator is like a miniature TV broadcast station broadcasting on channal 3.*

SPEECH SYNTHESIZER

The plug-in port on the speech module was intended to expand the number of permanent words of vocabulary as in Speak-n-spell. When TE-II worked better than expected, TI lost interest in expansion modules. It is possible that only the first few thousand speech modules have the "hooks" for those modules. I own several vintage speech modules and they all have the hooks, but I never got any plug-in modules and I designed the interface...(please, no questions, it's been 11 years!)

WIRELESS PERIPHERALS

Another historical note: the first few thousand original 99/4's had a hidden plug-in slot on the top flat surface (under the metal overlay) for an IR remote control transmitter/receiver about the size of a Kodak Instamatic. This device supported a number of never introduced wireless peripherals - including a wireless "super" keyboard and joystick.

The wireless peripherals were supported on the systems shown at the June 1979 CES show in Chicago. Ten minutes before the introductory press conference, we were told "...not to show the peripherals...." In any event I am fairly certain the software support for these peripherals was left in GROM - at least until the switch to the 99/4A, which included the new (actually the "original") keyboard.

These peripherals were never introduced because TI thought that they were too expensive for a Home Computer with an intended 1979 price of under $100. (When Apple was $2000!) When the price "skyrocketed" to $1000, it was supposedly "too late" to bring out the wireless peripherals.

The price increase was due to the complete failure of the cheap TMS9985 microprocessor originally designed into the 99/4. This resulted in changing over to a very expensive goldlead ceramic packaged TMS9900, a TTL clock driver, a 256 byte static RAM, and a ton of TTL glue logic. (The true story of how the 9900 ended up in the 99/4 would rival General Hospital!)

VIDEO MODULATOR

Also the first few hundred units had the "hooks" to bring in external video and genlock the TMS9918 - a capability which even today is not generally found in personal computers. This was removed from the connector (despite its less than $1.00 cost in parts) because at the time of the 99/4 introduction, the interface had not been tested sufficiently and there was still too much uncertainty about the FCC implications of genlocking a class 1 TV device.

Remember at that time (1979), any TV game or computer that had an RF modulator had to pass very strict FCC testing. The 99/4 originally had a built-in RF modulator, which was removed shortly before the June 1979 CES show when the decision was made to package the 99/4 with the Zenith color monitor.

In 1978 we built 200 prototype 99/4's with built-in RF modulators and a 9900 "emulation" of the 9985 on a 9"x 4" board sandwiched onto the original PWB and crammed into the original 99/4 tooled case. These 200 units were given to TI executives and board members to play with for six months. I have a wire-wrapped prototype with one of the very few 9985's ever produced, but I would be interested in acquiring any of the 200 prototypes which might be extant.

Everyone was quite shocked when the FCC ruled in TI's favor. That ruling distinguished personal computers from video games.

The decision to remove the RF modulator from the console was initiated when the FCC agreed to test a fiber optic interface we had developed. This optically coupled the computer console with a standalone RF modulator. This interface was dubbed the video light pipe." After the system was sent to the FCC they returned it UNTESTED, stating that they could not accept petitions for rule making to modify the existing rules. The rest is history.

## National Group Adopts Hardware Standards

The National Committee for TI Standards released the following hardware standards after a meeting at the Lima Multi Users Group Conference in May, according to Don O'Neil, facilitator for the group:

Level A: TI99/4A console, TV or monitor, cassette deck and cable.

Level B: Level A plus 32K memory expansion, EA/5 loader (e.g. Extended BASIC, Editor/Assembler, Supercart, TI-Writer, Multiplan).

Level C: Level B plus RS232, double-sided single-density disk drive and controller.

Level D: Level C plus 128K or greater CPU RAM bankable in at least 8K segements.

Level E: Level D plus 9938/58 VDP with 128K VDP RAM (any 80-column card or a Geneve.

The group selected as its first project to write a universal DSR for memory access to all current extended memory cards (Myarc/Cor/Comp 128/256/512K, Rambo, Geneve, RAVE, etc.)

The group plans to exchange information about the current RAM cards at the Chicago TI Faire October 31, O'Neil says. Once the DSR is completed and documented, the group plans to release it in the public domain. Persons with information on accessing RAM cards are asked to contact NCTIS at the Chicago Faire or members of the group which met at Lima.

They include O'Neil; David Connery, Used TI Equipment; Vic Steerup, consumer, Beery Miller, 9640News; Mike Maksimik, Crystal Software; Bud Mills, Bud Mills Services; Mark Wacholz, Media Ware Software; Ken Gilliland, Notung Software, and Mike Sealy, MS Express Software.

*Editor's Note: this article was taken from page 20 of the July 1992 issue of MICROpendium.*

MONTHLY MEETING LOCATION
LITTLE HOUSE NEXT TO THE
ST. ANN'S SCHOOL
2839 S. McCLURE
INDIANAPOLIS, IN
MEETING STARTS
AT 2:00 P.M.
AUGUST 16 1992

DISCLAIMER

This newsletter is brought to you through the efforts of the 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 with the other members mail it to:

Bryant Pedigo
6461 N. Sherman Drive
Indianapolis, IN 46220

Opinions expressed are those of the author and not necessarily those of the HOOSIER USERS GROUP.

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


Forwarding and Address
Correction Requested




**TIME DATED**
August 16, 1992
**MATERIAL**

May 1993

Dan H. Eicher
P.O.Box 605
Mooresville, IN 46158

---

## APPLICATION FOR MEMBERSHIP

Below you will find an application for membership to the Hoosier Users Group. Active memberships entitles you to the Newsletter, up and download 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.

Make check or money order payable to *Hoosier Users Group*. Send completed application to:

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

Please print — cut on line

Check One:

Active Member
New: $20
Renewal: $18
Dues will be due in May of each year. New members subtract $1.50 for each month from Oct. New member minimum $10.00
Amount
Enclosed
#
S

NAME
TODAY'S DATE
ADDRESS
APT #
CITY
STATE
ZIP
PHONE ( ) -
INTERESTS/ COMMENTS
D
O