THE GUILFORD 99'ER NEWSLETTER

VOL.3 NO.11 NOVEMBER 1986

The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to P.O. Box 21691, Greensboro, NC 27420. The Software Library is for dues paying members only. (Herman Geschwind, Editor)

OUR NEXT MEETING

DATE: November 4, 1986. TIME: 7:00 PM PLACE: Glenwood Recreation Center 2010 S. Chapman Street.

Variations in XB. Ever since when TI came out with the Extended Basic Cartridge several years ago, programmers felt that there were still features missing or a general improvement needed. These additions and extensions have taken many forms, some are entirely new cartridges (Myarc's XB and the German XB), some of them are in the form of disk-based enhancements (Super Extended Basic, XXB and Star) and one of them even calls for a new piece of hardware, the GRAM KRACKER for GK Extended Basic. Some of these extensions have been around for a while (SXB), some are brand new (STAR, GK XB and XXB). What will these extensions do for you? Which one has a chance of being the standard for the future? What are prices and availabilities? How compatible are these extensions with existing XB Programs? Bob Carmany will give us a show and tell on this timely subject. (You may want to download STAR from the BBS and study the docs so that you, too, can contribute to the discussion.)

TI SHOPPER

by Bob Carmany

Just a couple of reminders before we get into what is new for the TI. The latest issue of MICROpendium has an advertisement from TACHYON SYSTEMS, 5125 Westwind Way, Salt Lake City, UT 84118. It seems that they are once again producing a 32K standalone with a $50.00 price tag. I have had both of the models that they produced and this second one without the power supply works fine -- at least mine has ever since I "swapped" it out for the old one.

Don't forget the "rock bottom" price on Extended Basic in the Fall TRITON catalog. The $29.95 price for the original TI cartridge is really a bargain! It would make a good "stocking stuffer" for Christmas or just a good deal for a backup!

RYTE DATA MILLENIUM COMPUTERS, 210 Mountain St., Haliburton, Ontario CANADA K0M 1S0, has a wide variety of hardware items from Europe. There are several P-Box cards and other items that are really worth looking at. There is the 128K bRAM card for the P-Box (another GRAM KRACKER clone) for $249.50 as well as a couple of other multi-function cards. They have a DS/DD Disk Controller card with 32K on board for $265.95. There is also an RS232 card (again with on board 32K) for $175.50. As

interesting as those two items are, the real "bomb" is a 1 MEG memory expansion!! This is a standalone unit that plugs into the I/O ports and provides the standard 32K plus 992K of extra memory. It includes a "SUPERVISOR" program to allow the bank switching of the extra memory and it will function as a RAMDISK. This MegaRam peripheral sells for $575.95 --- that is some serious memory expansion!!!

Of course, they also have a BASIC COMPILER for $20.00, the XBII+ cartridge for $75.00, and a GPL ASSEMBLER for $59.95. You might want to write and ask for a catalog or price list if you are interested in these or some of the other products that they have available.

The evolution of memory expansion has been one of the more interesting progressions ever since TI "dropped out of the game". They once told us that th 32K memory expansion was the memory limit for the machine (it is for direct access). There were a variety of 128K cards (FOUNDATION, et. al.) which were son followed by the 128K RAMDISKS. Then, several months ago, CorComp introduced both 256K and 512K cards and standalone memory expansion units. Now, we have a full Megabyte of memory available for our "lowly" TI. Amazing!!! You can expand from 16K (basic console) to a full Megabyte of memory and the only limit is how much you want to spend!!! Who would have guessed it would go this far some three years ago!

Well, I had better get the disk drive cranked up again -- until next month

FOR SALE

A complete TI system for sale as a unit. Two TI 99/4A consoles, three sets of joysticks , PE box with 32K, RS232, internal disk drive, a second external drive, cassette recorder, speech synthesizer, Alphacomm 40 column printer, modem. Dozens of cartridges, tapes, disks, books, including file boxes. Two XB cartridges, Mini-Mem cart and Advanced Diagnostics. All in A-1 excellent condition. Everything except a monitor for one price. Call Dave Cohen at 333-5207 (work) or 656-7632 (Home).

TI WRITER FIXES

by Rick Cosmano

To change the default colors in the Text Formatter, use a sector editor with a search feature (such as DISK+AID or DSKU) and search the FORMAT file for 02 00 07 F5. The F5 is the hex code for the screen color. I changed mine to F0 for white on transparent. Then search for 80 02 01 F5. Again, I changed mine to F0 for white on transparent. For those using the TIGRAMDSK of the Gram Kracker, search for the above strings with the memory editor. I found mine at gB2A5 for the screen color and gB2B4 for the character color.

To change the & and @ symbols so that they are printed as regular characters, search the FORMAT file for 25 21 40 26. Change the 40 26 to 60 5C. This will allow you to use the & and @ as regular text characters. The 'TICK' (FCTN C) is now used for overstrike and the 'BACKSLASH' (FCNT Z) is used for the underscore.

(Editor's Note: When using FUNLWRITER the color change procedure will not be necessary since the default colors can be changed in the XB load program. To change the & and @ as suggested by Rick in FUNLWRITER, look for the string at byte 113 of FORMAT.)

BITS AND BYTES

by Bob Carmany

Remember the days when there were no TI bulletin boards in town? Well, it seems that besides the one that Ben Mann runs (274-5760), there is another one in town. GROUNDSTAR BBS (855-3088) is another that, right now, only has a TI message board up. Dan Post is the SysOp and he tells me that if enough interest is shown, he will be glad to put in a TI section. The BBS currently supports both 300 and 1200 baud and is up 24 hours a day. There is no connect fee and the number is a local call. So, show some interest by calling and leaving a message for the SysOp or other messages in the TI message board!!

SWITCHES

by Brian Kirby

First, let's describe what each of these switches will do for you and the computer.

LOAD interrupt: The load interrupt, when activated will cause the computer to suspend its current operations. Then it will look at a specific memory locations that will tell the computer where to go for the next set of directions. This switch

is useful for several utility type programs. You can have a debugger or disassembler loaded in the memory along with the program you plan to check. When your running program cuts up, you can hit the load interrupt and be put into your debugger program. Then you can go see what happened to your program in the computer's memory. Another use is screen dump routines. You can have a utility loaded up in the computer and your program. When you want a copy of the screen you hit the load interrupt switch and then the screen dump routine takes over and you end up with a hard copy of what was on the screen. You can come up with all kinds of utilities for the load interrupt switch.

In specific a load interrupt causes the 9900 cpu to initiate a interrupt sequence immediately following the instruction being executed. The memory location at >FFFC is used to obtain the vector for the Workspace Pointer and the Program Counter. The old Program Counter (PC), Workspace Pointer (WP) and the Status Register (ST) are loaded into the new workspace and the interrupt mask is set to >0000. Then the program execution resumes using the new PC and WP.

Here is a check, just for grins, that will let you know that the load interrupt works. If you have a memory editor type program (SBUG,MEMORY+AID, GRAM KRACKER,etc) go into memory location >FFFC and change the next four bytes to >83 E0 00 24. The first two bytes are the Workspace Pointer (>FFFC) and the last two bytes are the Program Counter (>FFFE). If you do a load interrupt using these changes the computer will do a power up reset routine. Another is to set the WP and PC to >83C0 and >0900. This is a level one interrupt. When you do it, the system will lock up, but you will note all your P-Box cards lights will be on except the memory.

HOLD: The hold does what it implies. It puts the microprocessor on hold. It's good for stopping the computer dead in its tracks. Works great for games that do not have a pause function. There are times when you do not want to use it. The states you do not want to be in are Input writing out a program in basic or XB. Try it during a game.

RESET: Again it resets the computer. It causes the computer to do the power up routine. This is great when the computer locks up. You hit the reset switch and you are back to the title screen. This saves wear and tear on your power switch and extends the life of the computer's power supply. There have been many articles on the reset switch and not all reset switches work properly. Let me explain why. First the basic form of the reset comes from the cartridge that you plug in the computer. There is a line that runs from the GROM port or cartridge port back to the clock chip that supplies timing for the whole computer. When the GROM port reset line goes low it causes the clock chip to reset and it in turn passes a reset on to the CPU and the 9918 VDP and the 9901 CRU chips. If you have a Widget this is what it uses to reset the computer when you put a new cartridge in. But I'm sure you have noticed that when you have locked up a few times and the reset on the Widget did not do the job, You had to shut the computer off and on to bring it back up. This was due to a lockup in the clock chip and it could not pass the reset along. The modification listed will reset the computer, no if, ands or buts. Thats because it bypasses the clock chip and runs directly to the CPU. When the CPU sees the reset it will in turn, reset all other devices that are resetable.

Specifically, the reset causes the CPU to be reset and inhibits WE and CRUCLK. When the reset is removed the 9900 initiates a level 0 interrupt sequence that acquires WP and PC from locations >0000 and >0002. It also sets all status registers bits to zero and starts execution. Reset also terminates an idle state.

To check it out is easy. When running a program push the reset button. You will be returned to the title screen. If you computer locks up while running a program and does not respond to anything, press the reset button and you will be returned back to the title screen.

Now we will discuss how to install these switches.

First, This modification will require you to open up your computer and solder a few connections directly to the 9900 CPU chip. If you can't solder or open up the case, then this article is not for you. You also may end up drilling holes in the case to mount the switches, but you don't have too. So think about it, your computer will never be a virgin again. Also I cannot be responsible for any problems you may have by trying to implement these modifications. I have supplied you information in good faith but again, its up to you.

First the required parts:

One push button switch,normally open type, use a micro type if you plan to mount it in the console.

Two lever type switches, normally open, again micro types if for the console.

Three 2.2 uF/16V tantalum capacitors.

About 4 feet of small gauge wire for hook up. Wirewrap wire is great if you mount the switches inside the console. If you do not want to drill holes in the console, buy some ribbon cable and a mini box.

Open up the console by removing the screws on the bottom of the console. Note how the door on the I/O port to the P-Box is installed. Then note how the power switch is assembled on the power supply. Remove the screws on the power supply board and set the power supply aside. Remove the plug from the power supply to the computer board. Note how the plug connects. Notice the keyboard and how it connects to the computer. Remove the screws that hold the keyboard and remove the keyboard. The computer is then removed by taking out the remaining screws that secure it. Note its position. Then remove the screws that hold on a shield to the I/O port. Note how that connects. Then remove the remaining screws that hold the shields on the motherboard. Locate the 9900 chip inside. Its the biggest chip and it has 64 pins. We are interested in pins 4 (LOAD), 6 (RESET) and 64 (HOLD). Solder three wires to these pins and mark the wires as to what they are. Be very careful not to splash solder or to short out connections while soldering. Bring these wires out thru a hole in the shield. If you are going to install switches in the console, come out thru a lower hole near the power supply. Reassemble the motherboard with its shields and note all the above that was discussed while taking it apart. If you are going to mount the switches in the console a good place is beside the power supply so the switches stick out beside the I/O port. Be sure to mount them so that they do not short to the power supply and make sure you will have enough room to mount your speech synthesizer. If you are using stand alone devices, you may want to mount the switches in the rear of the console.

Now that you have found a location that works, mount the switches and solder one each of the three wires to each of the switches. Make sure that the reset line goes to the pushbutton. Solder one of the capacitors to each switch across the connections. Make sure the positive side of the capacitor is connected to the line that goes to the computer. On all of the switches run a jumper to the other side that has no connections. Jumper all of them together and run one wire back to a ground on the computer. The shield is a good ground point. Put the computer back together following the reverse of taken it apart.

If you do not want to drill holes you have several options. First you can use ribbon cable and run it out of the rear of the computer to your minibox where you can mount your switches. This way if you decide to remove the switches you can just unsolder your connections and everything will be back to normal. You can also mount the load interrupt switch external to the console, by coming off of the I/O port. You can mount the switch in the speech synthesizer be connecting one side of the switch to edge connector finger number 13 (LOAD) and the other side of the switch to pin 21 or 23 or 25 or 27 (all grounds). But you cannot access the hold or reset thru the I/O port. They do not make it outside of the computer. If you want just a load interrupt, Navarone sells a board that goes between the "firehose" and the console and supplies a load interrupt. It is about $15.

BASIC COMPILER V1.1

by Herman Geschwind

While there have been many exciting developments in new programming languages for the 99/4A during the last three years (Forth in several variations, "C", Pilot and even a MacroAssembler), very little has happened to provide the TI community with a workable Basic Compiler. Compiled basic is readily available in several excellent implementations for the MS/PC-Dos and C/PM world, for the TI where a basic compiler would do the most good, very little has been available.

Why a Basic Compiler? The great benefit of Console or Extended Basic is that a program can be keyed in and then by issuing the "RUN" command, hopefully, the program will execute. If it should not work as expected, it is quite easy to edit the program and try again.

The penalty for this ease of use is that compared to programs written in other programming languages, both Console and Extended Basic will run very slowly. The reason for this is that Basic is "interpreted", which means that every program statement needs to be converted into machine language that the processor can understand and execute. Once the statement has been executed, the machine language conversion is discarded and the next statement must be converted. The inefficiency of this system is particularly apparent when statements within a loop must be converted over and over again until the loop ends. A compiler (such as an Assembler) makes this conversion only once and then the program is presented to the processor in a form which can be directly executed.

From this digression into the nature of "interpreters" versus "compilers" it should be clear that a system that could be both interpreted and compiled really might be the best of both worlds. As long as a program is still in the development

state, it is much easier to test and debug and edit, if necessary, in interpreted mode. Once the program is finished, it could be compiled for optimal running efficiency.

Ideally, a Basic Compiler should make it possible:

1) To use the standard repertoire of Console Basic or Extended Basic keywords, syntax and constructs. There should be no restrictions on the math capabilities , both integer and floating points.

2) The compiler should accept the basic program in its normal, executable form without the need to create Merge or List formats.

3) The compile process should be fast which means that the compiler itself would need to be an Assembler program.

4) The support library that may be required should not be copy protected or encumbered by legal restrictions so that compiled programs can be readily exchanged.

For the 99/4A there are only two compilers (SST and Compiler V1.1). Another product, 9900Basic, is not a true compiler but a product that uses a compiler approach to convert a limited number of Basic statements into Assembler code which then in turn needs to be compiled (assembled).

This review will deal primarily with Compiler V 1.1 by Ryte Data, 210 Mountain Street, Haliburton, Ont. K0M 1S0, Canada, but comparisons to the SST compiler will be inevitable and hopefully useful to put Compiler V 1.1 into perspective.

Compiler V 1.1 is an assembly program with an Extended Basic loader. It will operate with a single disk drive but is much more efficient and user-friendly as a two-drive system. As is to be expected, the system loads quickly and the compile process is lightning fast. By contrast the SST compiler is a basic program and the compile process for even a trivial program can take as long as an hour.

The Ryte Data product will accept normal Console or XB programs either with single or multiple statements per line. Again by contrast, SST will only handle single line statements in MERGE format.

Once the Compiler V 1.1 loads, the user is presented with a menu screen with the options LOAD: 1. BASIC, 2. COMPILER, 3. FP-LOADER, 4. INT-LOADER. Selection option 1. invokes a menu loader which scans the data disk and shows a directory listing of eight files with each file identified by number. Selecting a number will execute the program. Using the combination of SHIFT+NUMBER will load the program without execution and CNTRL+NUMBER will delete the program. FNCT+6 will bring up additional files, again identified by numbers 1 through 8. (This menu selection also applies to the "compile" and "fp-" and "int" loader).

Selecting option 2 will invoke the actual compiler, again with "menu-by-the-numbers" selection of the program to be compiled. Once the compile process is finished, pressing ENTER will bring up the main menu again.

Compiler V1.1 comes with two support libraries, one for floating point applications (FP-LOADER) and one for integer only support (INT-LOADER). Except for the fact that the integer loader will not support decimal math and intrinsic functions such as SIN, LOG or SQR, it is otherwise identical to the floating point loader. For applications where floating point math support is not needed, the INT-LOADER will run programs much faster than the FP-LOADER.

So much for the good news. Compiler V 1.1 will compile Basic and XB programs in your library with blazing speed and no restrictions on program format. The bad news is that if you expect assembler-like speed from compiled programs you will be in for a disappointment.

The main gain is that there is no longer a lengthy delay for the pre-scan, programs will execute almost at the instant when the load process from disk is finished. As for actual run time improvements, results can vary widely. For programs with a lot of text, the performance gain is negligible. On the other hand, programs with elaborate graphics and sprites will benefit substantially from the compile process. Music and speech programs should not be compiled unless you like Donald Duck-like renditions.

Here are some examples to illustrate what is to be expected. Consider an empty loop of the type FOR I=1 TO 10000::NEXT I. Extended Basic will take approx. 35 seconds to run through this loop. The same program compiled and run with the integer

loader will finish in only approx. 11 seconds. So far so good. But now let us introduce a print routine and see what happens with the loop FOR I=1 TO 1000::DISPLAY AT(15,6):I::NEXT I. In this case XB will be finished in a little over 1 minute, while the same compiled program will need 25 seconds longer to finish! We don't want to embarrass the compiler world by telling how fast a pure E/A program would run! Suffice it to say that one of the drawbacks of compilers is that the machine code which is automatically generated is never as efficient as a hand-coded E/A program. In the case of Compiler V1.1 the code appears to be particularly inefficient. A disassembly of the object code for the empty loop shows that 49 assembler statements were needed, not counting the code that is part of the loader. In the case of the SST compiler an execution time of 1.4 seconds is claimed for an empty loop of 1 to 30,000; but the price for this optimized code is a number of other restrictions.

Other bad news is that since Compiler V 1.1 is entirely memory resident, there is a restriction on the size of the program that can be compiled. Again, conditions under which the compiler will abort with the message "Program Too Big" vary. In some cases 20 sector programs refused to compile while in other cases even a 30 sector program would compile and run properly. (A possible solution would be to segment a big program into separately runable components since the compiler will support the "RUN DSKn.XXXX command).

Unlike the SST compiler, which requires a major re-write (all variables must be explicitly declared at the beginning of the program, special syntax for many constructs, limited string and math functions), there are only a few cautions that need to be taken with Compiler V 1.1. Keywords that are not supported are DEF, SUB, OPTION BASE 1, TRACE, ON ERROR, CALL LOAD and CALL LINK. The END or STOP statement must be at the actual end of the program. A more serious limitation is that only one file can be opened during program execution.

A quick way to screen a program for these restrictions is to try a compile run. As unsupported keywords are encountered, the compiler will show "Bad Command Line XXX" where the line number refers to the basic program line number. Since the compile process is so fast, it is an easy matter to let the compiler finish and then to select Basic from the menu and fix and save the offending program.

Another disturbing aspect of Compiler V 1.1 is that the code generated at times is not free of bugs and glitches. In some cases a compiled program appears to execute properly and then all of a sudden the system crashes or the console will lock up completely. At times a re-compile of the same program will result in better code, in other instances it was found that SAVE,MERGEing a program first would result in more stable code. Additionally there also appears to be a hardware dependency. Use of the GRAM KRACKER with customized Upsys and XB resulted in a succession of spectacular crashes. A plain vanilla black and silver console and XB cart. V110 cleared things up somewhat.

The copyright notice indicates that Compiler V1.1 was written in or prior to 1984. The fact that in two years the author did not issue a more optimized code and eliminate the annoying propensity to crash should give cause for concern. We have faulted many Fairware authors for their rapid succession of updated versions (see DM1000 and FUNLWRITER) but at least they made an attempt to improve their product.

The documentation supplied with Compiler V1.1 is all of 3 1/2 normal typewritten pages. Fortunately the user interface of Compiler V1.1 is accomplished enough that not much more is needed to get the program up and running.

Modifications: The default for the data disk drive is on sector 121, bytes 65 and 81 which on the distribution disk point to DSK1. Changing this byte to a 2 will default the data disk to DISK2 which makes for a much smoother operation with a two-drive setup.

One of the stipulations that we set forth earlier was that any associated support programs and loaders should not be encumbered so that compiled programs can be exchanged readily. Unfortunately both the compiler and the load programs of Compiler V 1.1 are copyprotected. So is the SST compiler offering.

To sum things up, Compiler V1.1 certainly is a step in the right direction and compared to SST and 9900Basic deserves high marks for ease of use and user friendliness. Where our quest for the really good compiler for the TI is still unfulfilled is in terms of performance (execution speed) and the rough edges that are evidenced by the many glitches and crashes. With the Fairware market being what it is, maybe an unprotected compiler and loader is too much to hope for. For someone wanting to experiment with a compiler and willing to accept modest performance gains for programs of limited size, Compiler V 1.1 might be acceptable for the small price of $20. For programs of larger size and complexity, you might want to hold on to your money in the hope that someone, someday will come up with a truly acceptable and workable compiler for the 99/4A.

FORTH FORUM

by Bob Carmany

This month, instead of printing a screen or two, let's take a look at what is available in the realm of utilities for FORTH. There are a couple of packages around that are worthy of mention and we will start off with some things that are available for TI FORTH.

In the commercial area, there is the COMPLETE TI FORTH UTILITY SYSTEM by Mike DeFrank, 4374 NW 9th Ave, Pompano Beach, FL 33064. This group of utility screens contains: a 9900 Code Disassembler, a Forth Decompiler, Screen Dump, Sound Utilities, Speech Utilities, improved 40 column and 64 column editors, and some other enhancements. The price of this package is $19.95.

Another commercial entry is an upgrade of TI Forth called SUPER 4TH by DataBioTics (see September MICROpendium) for the details of this application. This one sells for $29.95 and includes a 108 page manual.

There are, of course a number of "freeware" and "fairware" enhancements and improvements to the original TI Forth program. One of the best sources for FORTH applications are the remaining TI-specific magazines still available. THE SMART PROGRAMMER and MICROpendium have FORTH articles appearing from time to time. In fact, Howard Arnold, 210 Beech Valley Rd., Lewisville NC 27023 (the Winston-Salem UG) has a "freeware" series of screens for $5. This rather lengthy series, FORTHFONT, appeared in MICROpendium over several months. Mr Arnold also has a database program available (debugs in September MICRO.).

In Wycove Forth, I still have a series of screens available that were passed on by Tim MacEachern that include a full set of disk utilities (including a disk copier) and some terminal screens and documentation to go along with the whole package. I will be glad to share the entire package with anyone who is interested in it.

Also available are some TI Forth screens with various applications that I have "gleaned" from various sources. Along with those screens is an enhanced version of the original TI Forth system with an XB load and a true lower case character set (selectable from the menu). The editor has also been improved!

Anyone who is interested in either the Wycove of TI Forth screens are invited to contact me for a copy.

Well, that about does it for this month. Next month we will get back to the "good stuff". I needed a break to come up with something of value for the next issue. Besides, Herman was getting tired of all those "at" signs that had to be transliterated by the formatter. Until next month, MAY THE FORTH BE WITH YOU!!!!

(Editor's Note: Since Bob refused to supply a Forth type-in for this issue, I found the following in my collection. Happy gameing!)

```
( SLOTMACHINE PAGE 1 OF 7 ) 57 CLOAD VCHAR 52 CLOAD GRAPHICS BASE->R
: PUTCHAR DUP DUP 2 + SWAP EMIT8 EMIT8 CURPOS @ 30 + CURPOS ! 3 + DUP 2 -
EMIT8 EMIT8 CR ;
: SETSCREEN GRAPHICS 15 0 DO 15 0 I COLOR LOOP 1 SCREEN CLS ;
: FRUIT 6 RND 8 * 128 + ; ( Sets up variables for fruits ) 0 VARIABLE
FRUIT1 0 VARIABLE FRUIT2 0 VARIABLE FRUIT3
: SLOT 64 0 DO I 32 < IF FRUIT DUP FRUIT1 ! 11 9 GOTOXY PUTCHAR ENDIF I 48
< IF 2 0 DO FRUIT DUP FRUIT2 ! 14 9 GOTOXY PUTCHAR LOOP ENDIF 3 0 DO FRUIT
DUP FRUIT3 ! 17 9 GOTOXY PUTCHAR LOOP 0 I 32 * + 0 DO LOOP LOOP ;
( THANKS TO NETWORK99 BY BRIAND SANDERSON )
-->

( SLOTMACHINE PAGE 2 OF 7 ) HEX ( Character Definitions )
: HEART 0000 1C3E 7F7F 7F7F 80 CHAR 3F1F 0F07 0301 0000 81 CHAR 0000 387C
FEFE FEFE 82 CHAR FCF8 F0E0 C080 0000 83 CHAR 9 0 10 COLOR ;
: DIAMOND 0000 1F22 447F 4424 DECIMAL 88 CHAR 1412 0A09 0505 0301 89 CHAR
0000 F844 22FE 2224 8A CHAR 2848 5090 A0A0 C080 8B CHAR 7 0 11 COLOR ;
: CHERRY 0000 0000 001F 3F7F DECIMAL 90 CHAR 7F7F 7F7F 3F3F 1F00 91 CHAR
0000 0608 1020 4080 92 CHAR E0F0 F0F0 F0E0 C000 93 CHAR 6 0 12 COLOR ;
-->
```

```
( SLOTMACHINE PAGE 3 OF 7 ) ( Character definitions cont. )
: LEMON 0000 000F 3F3F 7FFF 98 CHAR FF7F 3F3F 0F00 0000 99 CHAR 0000 00F0
FCFC FEFF 9A CHAR FFFE FCFC F000 0000 9B CHAR A 0 13 COLOR ;
: BAR FFFF FFFF FFCC D5C4 A0 CHAR D5CD FFFF FFFF FFFF A1 CHAR FFFF FFFF
FF47 5747 A2 CHAR 4F57 FFFF FFFF FFFF A3 CHAR 0 F 14 COLOR ;
: BELL 0001 0101 0103 0307 A8 CHAR 070F 0F1F 1F01 0100 A9 CHAR 0080 8080
80C0 C0E0 AA CHAR E0F0 F0F8 F880 8000 AB CHAR 4 0 15 COLOR ;
-->

( SLOTMACHINE PAGE 4 OF 7 ) ( More Character Definitions )
: MACHINE 1818 1818 1818 1818 AC CHAR 0000 0000 0000 FFFF AD CHAR FFFF
0000 0000 0000 AE CHAR 0303 0303 0303 0303 AF CHAR C0C0 C0C0 C0C0 C0C0 B0
CHAR C3C3 C3C3 C3C3 C3C3 B1 CHAR FFFF FFFF FFFF FFFF B2 CHAR E0E0 E0E0
FCFC FCFC B3 CHAR FCFC FCFC E0E0 E0E0 B4 CHAR FFFF FEFE FCF8 F0C0 B5 CHAR
FFFF FFFF FFFF FFFF B8 CHAR 0003 070F 0F1F 1F1F B9 CHAR 1F1F 1F1F 0F0F
0703 BA CHAR 00C0 E0F0 F0F8 F8F8 BB CHAR F8F8 F8F8 F0F0 E0C0 BC CHAR ;
-->

( SLOTMACHINE PAGE 5 OF 7 ) ( Draws The Slotmachine )
: SETMACHINE 4 0 16 COLOR MACHINE 9 6 E AF VCHAR D 9 2 B1 VCHAR 10 9 2 B1
VCHAR 14 6 E B0 VCHAR A 9 2 AF VCHAR 13 9 2 B0 VCHAR B 8 2 AD HCHAR E 8 2
AD HCHAR 11 8 2 AD HCHAR 8 D 2 AC HCHAR E 8 2 AE HCHAR 11 8 2 AE HCHAR A 6
A AE HCHAR A 13 A AD HCHAR C F 6 AD HCHAR C 11 6 AE HCHAR B 10 1 AF HCHAR
12 10 1 B0 HCHAR 16 6 6 B2 VCHAR 14 B 1 B3 VCHAR 14 C 2 B2 HCHAR 14 D 1 B4
VCHAR 16 C 1 B5 HCHAR 16 4 2 B8 VCHAR 15 4 1 B9 HCHAR 15 5 1 BA HCHAR 17 4
1 BB HCHAR 17 5 1 BC HCHAR 6 0 17 COLOR ;
-->

( SLOTMACHINE PAGE 6 OF 7 )
: PULLHANDLE 6 0 DO 15 I 3 + 3 20 HCHAR 16 I 4 + 2 B8 VCHAR 15 I 4 + 1 B9
HCHAR 15 I 5 + 1 BA HCHAR 17 I 4 + 1 BB HCHAR 17 I 5 + 1 BC HCHAR LOOP 15
9 2 20 VCHAR 17 9 2 20 VCHAR 16 6 6 B2 VCHAR 14 B 1 B3 VCHAR 14 C 2 B2
HCHAR 14 D 1 B4 VCHAR 16 C 1 B5 HCHAR 16 4 2 B8 VCHAR 15 4 1 B9 HCHAR 15 5
1 BA HCHAR 17 4 1 BB HCHAR 17 5 1 BC HCHAR 6 0 17 COLOR ;
( PULLHANDLE is the routine that pulls the handle )
: WINY C 10 GOTOXY ." WIN " ;
: WINN C 10 GOTOXY ." LOSE " ;
: WIN? WINN FRUIT1 @ FRUIT2 @ = IF WINY ENDIF FRUIT1 @ FRUIT3 @ = IF WINY
ENDIF FRUIT2 @ FRUIT3 @ = IF WINY ENDIF ;
( You win if any two fruits are the same )
-->

( SLOTMACHINE PAGE 7 OF 7 )
( Main Program Routine ... COIN does what would happen when you put a coin
in the slotmachine ... PLAYSLOT is the loop to detect your joystick button
or the spacebar ... SLOTMACHINE is the master routine! It calls the other
routines. )
: SETGRAPHICS HEART DIAMOND BELL BAR LEMON CHERRY ;
: COIN PULLHANDLE SLOT WIN? ;
: PLAYSLOT BEGIN 1 JOYST DROP DROP 12 = IF COIN ENDIF ?KEY 20 = UNTIL ;
: SLOTMACHINE SETSCREEN SETGRAPHICS SETMACHINE PLAYSLOT ;
R->BASE
```

MODEM TALK

by Herman Geschwind

Most of the files that we listed last month are still available on the FIDO board (274-5760). Ben has been off to California for quite a while and that limited the opportunity to post more uploads. New files since our last listing are as follows:

UNPACKER (6272) This is a utility for those that have only one disk drive and need to "unpack" ARC downloads. UNPACKER works with a sizeable buffer and will prompt for the swapping of master and copy disk. A full SSSD disk can be "unpacked", but will require a few disk swaps.

STARA.ARC (36480) and STARB.ARC (50048). This is the complete set of Assembler Extended Basic Extensions. Documentation and sample programs are included. These are the files that you might want to download and study prior to our November meeting.

GMSS.ARC (61312). Another excellent E/A games collection. This set can be loaded with the Load and Run option of either FUNLWRITER or E/A Cart.

HYP.ARC (34944). This is the hyphenation program that was reviewed in this newsletter several months back. It should be used on all those TIW or FUNLWRITER documents where neatness counts. E/A program with XB loader.

TYP.ARC (25216). Improve your typing skills. Will load with FW (Program File) or E/A Option 5.

Please note that many of the programs posted are "Fairware" where the author is asking for a generally very modest contribution for his effort. Quite often these are excellent programs of a very high quality standard where a commercial version in the store might easily cost $100 or more. A good example is STAR where a similar, commercially available product was selling for $99. Please support our fairware authors to keep the good stuff coming.

MON.ARC. This is a very nice XB/EA version of the monopoly game. This can be downloaded from the ROS board (855-3088).

We have found out that Mass-Transfer will not properly transfer a TI file header when UPLOADING to a MS-DOS board. The author is working on a fix. Until this new version is available, please do not use Mass-Transfer to upload files from your system to FIDO or ROS. For downloads MT is just fine. Fast-Term can be used for uploads.