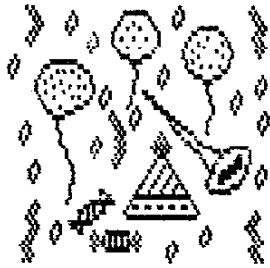
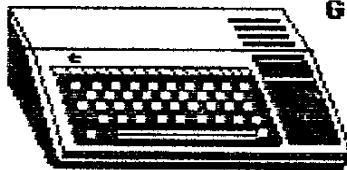


GUILFORD 99'ERS NEWSLETTER



SUPPORTING THE TEXAS INSTRUMENTS TI-99/4A COMPUTER



GUILFORD 99'ERS UG
3202 CANTERBURY DR
GREENSBORO NC
27408

TO:

, Pres.
Tony Kleen, Sec/Treas (924 6344)
BBS: (919)621-2623 --ROS

Bob Carmany, Newsletter Ed (855-1538)
Bill Woodruff, Pgm/Library (228-1892)

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: Tony Kleen c/o 3202 Canterbury Dr., Greensboro, NC 27408. The
Software Library is for dues paying members only. (Bob Carmany Ed)

NEXT THREE MEETINGS

DATE: January 7, 1992	Time: 7:30 PM.	Place: Glenwood Recreation
February 4, 1992		Center, 2010 S.
March 3, 1992		Chapman Street

EDITOR'S NOTES

This issue marks the beginning of a 'new' endeavor. As you might have noticed, we have listed the meeting dates for the first three months of 1992. That also means that the newsletter will be coming out on a quarterly basis as well. Hopefully, the newsletter will be a bit thicker since there will be additional time to gather material. Of course, it also means that you won't be getting your newsletter quite so often.

I had an interesting phone conversation with a fellow in Green Bay, Wisconsin the other day. It seems that he ordered one of the Quest cards last year and had just now gotten around to putting it together. The reason that he called was, of course, because he was having trouble getting it on-line. After some checking, changing a jumper here and there, and finally checking all of his solder joints, he was convinced that it might never run properly. Luckily, he followed up the call with a letter. The problem was that he had one of those "antique" Foundation 128K cards in his system. Once it was removed, everything went just fine. It seems that there is a memory contention between the two of them. When I last heard from him, he was marvelling at the sheer speed of his new Quest RAMdisk.

RETROSPECTIVE

By Tony McGovern

The last issue of the HV-99 Newsletter is an appropriate time for some retrospection, particularly on the early days. I suppose the first thing to ponder on is how a bunch of computer hobbyists in Australia ever came to get mixed up with Texas to such an extent. In our case we have never been there, the nearest being New Mexico on several occasions over the years. Last time in the US we lived through much of the year in what was a Texas culture camp in the summer -- the Colorado Chataqua of Boulder right at the foot of the mountains. In those days EEs I had known years before as analog circuit designers were mucking around at home with computer kits called KIM-1 and the like, and fully packaged computers were around but still expensive. Our friend Ez in Boulder even had a Commodore PET down in the basement and the kids played their first computer game on that. Some would say Will has never stopped. There was some pre-publicity about a consumer level computer from TI, but no real detail.

Several years later back in Australia, Atari games machines were all the

rage among the junior set, and the pressure was on for one of those for Xmas '82. With the usual importer and retailer ripoffs these were hideously expensive (\$275 if I remember correctly) and I said to myself --no way, for that sort of money it has to be a genuine computer even if it did cost more. The first flirtation was with a Dick Smith Wizzard, but that proved so hopeless in the keyboard department that I returned it for a refund (it took more of an argument than Dick Smith advertising would have you believe). I wonder if any of these are still in use? It had a TI 9918 VDP in it though. Then there was a magazine writeup of the new TI-99/4A computer with a BASIC that sounded quite comprehensive, and games cartridges. Xmas was getting close and things were getting desperate. The only known supplier was the then Myer store in Sydney. So that was how we ended up spending twice as much as for an Atari. The selection of a cartridge to buy, and they were scarce that summer - the computer had cost so much that we could only afford one anyway - came to a choice over the phone of Hangman or Munchman. Not knowing what either one was, I chose Munchman which turned out to be one of their best games. [Ed note: The seasons are reversed in the Southern Hemisphere so Xmas falls in their summer]

So off to Queensland for Xmas with Nana. The TI was a big hit, even if not quite an Atari, but various cousins had those anyway. They couldn't program anything, so that equalized it a bit. The TI marketing department had it right there, one cartridge is never enough, so a desperate search started around the Brisbane area. The holy grail was the mythical Extended BASIC, not yet descended from the high plains of Texas. A trek to the north of Brisbane, at an under-the-house (that Queensland all in one word concept) computer workshop yielded Hunt the Wumpus and an unusual cartridge called a Mini-Memory. Another expedition went this time to Ipswich, the town where my mother came from. So there just a couple of doors along from the E. Bostock & Sons building (funny - I don't think the kids have ever come to terms with having a great-great-grandfather called Enoch) we found A-Maze-ing and the dreaded Car Wars. I wonder what it is that makes one cartridge an all-time favorite like A-Maze-ing? The real sleeper of this bunch was the Mini-Mem.

On return to Newcastle we found there was indeed a dealer in town, and they had a bunch of notice slips from someone called Brian Rutherford, which started our association with what was to eventually come to be the Hunter Valley 99ers. Then came several months of waiting for Extended BASIC. When it finally arrived it proved to be a very fine implementation of BASIC, well in advance of what was available on other personal computers until quite recent times, even if slow. A language worthy of some serious programming. One of the things that I found very disappointing was the failure of most published XB programs to use the most powerful feature of the language, the user-defined subprogram. It was like the writers of these programs felt constrained to limit themselves to the parts of XB also found in the primitive BASICs of most other machines instead of using all the power of XB. This set off writing a set of XB programming tutorials, initially in the Sydney group newsletter, and continued in the HV99 Newsletter when our group went its separate way. These have turned up in many newsletters around the world, even in Swedish translation.

Mini-mem, however, allowed access to the assembly language and even at the level of the Line-by-Line Assembler it was clear that there was something well worth pursuing. I had been intending to learn to program at assembly language level, but the 9900 structure and the assembly language looked so much more satisfying to an engineer's outlook that it became the one of choice. Nine years later it still looks good and is still a pleasure to work with. The device independent I/O system still looks more like a real engineered job than most, even today. We were waiting for the 99/8 to appear before making a bigger investment, when Black Friday rolled around. After a little soul-searching, the elegance of TMS-9900 code won out, and we bought

New-Tech's demo /4A system and ordered E/A and TI-Writer.

The next big programming event was the release from TI, with source code as well, of FORTH, a language that promised speed not attainable with XB. When it finally leaked northwards from Sydney, it captured the imagination of several people - just ask Joe and Richard about it - and what is more the source code revealed all sorts of essential information that had not been in the E/A manual. After a brief flirtation with FORTH we went back to mastering Assembler. The great thing about Assembler is that nothing and no-one gets between you and the raw power of the computer. I see the real problem with higher level languages as being the addition of an obscuring layer of someone else's understanding of how the machine should best be used, no matter how helpful it might be a lot of the time. Writing in Assembler on the 99/4A is doubly rewarding because the 99-4A BASICs do not use the 16-bit TMS-9900 directly, but are written in the infamous GPL, an interpreted language simulating an imaginary 8-bit processor. The double interpretation just kills the execution speed, so the improvement obtainable is huge.

So what have we seen in the way of hardware advances over the years? Incremental advances came by way of DSDD capable floppy disk controllers from CorComp and later Myarc --in a sense just a quantitative improvement that TI would have made anyway in time. The first real change, the one that has made much of what the machine is today, was the Horizon RAMdisk and its developments. The programmable auto-booting in particular has meant the 99/4A remains the easiest of all computers to use, and RAMdisks have saved untold disk drive wear and tear for assembly programmers. On other machines it was (and still is mostly) a big deal to have a RAMdisk that survives normal system reset, let alone one that usually survives programs running wild, or even the machine power being turned off. The more recent great event has been the introduction of 80 column units based on the Yamaha V9938 development of the original TI VDP. Firstly this came as the clumsy Mechatronics unit from Germany, then as the more refined DIJIT PE-Box card from San Diego, and now the OPA TIM from Toronto, which looks to be the best of all. Some might vote for the Myarc HFDC and hard disk capability to join this select company, but this device needs another round of developments to fix its flaws, but it looks like it will never get it. Others may well have their own favorite devices of the many produced. The one area which has not yet been satisfactorily resolved with universal agreement is that of RAM extenders. The E/A "super-cart" with its extra 8K is universally recognized, but pre-empts all other cartridges. TI did have in the works a memory expansion system of general applicability, but this was never taken up, a real tragedy for 99/4A users.

I have not included the Myarc Geneve 9640 in this list. It really has been a development off to one side, and kept that way by the non-communicative, closed-system policies of Lou Phillips and the little circle of software developers who were never quite up to delivering the goods. Didn't they learn anything from the TI debacle with all its corporate resources behind it?

The intervening years have been largely filled with the development of the Funnelweb system. There were a few other programs along the way. The first, done as a learning exercise for XB was a pure XB game with sprite handling code so finely tuned to TI XB that Myarc XB-II, which was supposed to be faster cannot run it. The next major one was a multi-column lister for XB programs that grew out of the tutorial articles, and showed just how helpful XB subprograms are to good program writing. Then came the first versions of Funnelweb, born out of fear of the exorbitant rip-off pricing of TI products by Imagic. It was clear by then that TI-Writer was the main reason for keeping with the 99/4A and all of a sudden that GROM cartridge looked very fragile. In retrospect, not a single cartridge has ever gone bad on us, but TI's life sentence on everyone of cartridge swapping forever did

not appeal to us either. So after an Easter long weekend of poring over my disassembly of the Editor, the essential ideas clicked into place --how the code overlay system worked, and the magic little code patch that allowed a full ShowDirectory to be incorporated. What amazes (and depresses me too) is what we found a couple of years later when the original TI source code for the Editor finally leaked to Newcastle, was just how little had been done to it in the meanwhile. The same problem is still bedevilling the TI-99 community -- the original TI source code rarely leaked to those willing and able to do something with it. Assembler and Formatter source availability would have made a great deal of difference to Funnelweb development. Now that Gary Bowser at OPA in Canada has officially licensed TI code, it is to be hoped the programming skills there are adequate in quality and quantity to do something wirthwhile. For most of us it is now getting too late in their piece to care anymore anyway.

So where has it all led to? The HV99 group was fun while it lasted, which was in fact for much longer than anyone would have dared predict at the time of its formation, and its influence has been felt way beyond all expectations. Now the time has come to wind up the HV99 group without it suffering lingering decay, so we all remember it in its prime. 99/4A support by way of the Funnelweb system will continue for a while yet here, but inevitably involvement with newer and brighter things will bring active program development to an end, even if the machine itself remains in use here. Will has now been off on the Amiga for several years, and is still an assembler freak. It is clear in retrospect that the professional level of the E/A facilities, the organized 99/4A system structure, and the elegant TMS-9900 16-bit instruction set all combined to give a training ground for the later computer that was second to none, and far superior to that available on other personal computers of the time. It was a home mini-computer in disguise! This is the basis of the best advice I can give to anyone continuing with their TI or just starting with a newly bequeathed one. It was and is an excellent computer system, and remains an excellent self-education tool in programming and getting to understand computers. There isn't much point coming to it new as a package user --PCs and the like have overwhelmed that scene.

Perhaps the wheel has turned full circle. Now that advanced users are making use of the Yamaha 9938 VDP in their TI-99s, development work at Funnelweb Farm an the Amiga will soon be turning to a new and powerful generation of Texas Instruments graphivs co-processors, starting with a 60 Mhz TMS-34010.

RAVE

A company in Connecticut is marketing two replacement keyboards for the TI. Both keyboards feature standard key layouts which eliminate two-key functions like cursor control, FCTN keystrokes, and "keyfront" keys such as ",?,\, and [. The Alpha Lock function will not interfere with joystick operation on any version of the TI-99. This hardware will work with all software up to simultaneous keystrokes such as FCTN/CTRL/SHIFT. NO P-BOX IS REQUIRED.

Both keyboards have coiled cords for connection to the computer console so you can move your keyboard around without problems from the side connector. You will have one-key entry of word processing commands such as TAB, BACK TAB, DELETE, BACKSPACE, and HOME when using TI-Writer. There is no indication as to how these things will hook up to the 4A. The information I have says that the keyboard interface installs in place of the standard keyboard in the computer console with NO soldering required. The interface has provisions for user upgrade to include SYSTEM RESET and

LOAD INTERRUPT capability from the keyboard.

As near as I can determine, you remove the 4A keyboard from the console and install their interface which connects to the keyboard. The 99/84 keyboard enhancement has an 84 key enhanced version of an IBM PC/AT layout. There is an LED indicator for the CAPS LOCK and NUM LOCK keys. It will also have an adjustable keyboard tilt. There are 10 function keys along the left side for FCTN1-10 and they can be used for CTRL1-10 when SHIFTed. You can optionally use overlay strips with the number key row as we're doing now with the 4A. Price for this keyboard will be \$149.95.

It's a nice looking keyboard and has the 10-key on the right hand side. The model 99/101 is the deluxe package. It is a heavy duty 101 key layout. There are FOUR separate key assignment modes which allow the user to have TWO separate command/programming modes. These modes can place the often used double quote (") in the unshifted key position for ease of use, for example. The QUIT key may be implemented as a single keystroke or may be defeated depending upon the mode selected. There are TWO separate editor modes also. One implements word processing keys for IT-Writer, the other for use with Editor/Assembler.

You have dedicated cursor control and a ten-key on the right. The function keys are arranged in a row at the top of the keyboard, allowing for direct use of standard overlay strips. This keyboard will have a 25-foot coiled cord as an option (\$9.95). The 99/101 will sell for \$164.95. Availability is expected to be about October 1 for both units. For more information, contact: Rick @ (203)242-4012 or John @ (203)872-9272, both after 6 pm Eastern time. These guys can also do custom key assignments such as Dvorak or Multiplan for a minimum of \$15. RAVE 99, 23 Florence St., Bloomfield, CT 06002

PRESCAN-IT

One of the annoying things about TI Extended Basic is the long wait you have between typing in RUN and when the program starts. This is because XB scans the program and sets up memory space for variables, arrays, and data. This wait can be considerable if you have a long program with a lot of arrays and variables. The pre-scan commands are a bit obscure and unless you are an extremely disciplined programmer, they are hard to use. Well, Asgard has just released PRE-SCAN IT! to help us out. You won't find the pre-scan commands listed or discussed in the Extended Basic book, they're in the supplement that came with the book and you really don't get much help as to how to use them.

Using the PRE-SCAN IT! program (hereafter referred to as PSI), a relatively inexperienced XB programmer (such as myself) can use these commands to not only speed up execution time, but your program may end up shortened and even run faster. First, load the XB program you want to work on and RES it so that the line numbers start at 100, then save it in MERGE format. Next run the PSI LOAD program which presents you with the options of using XB in a 16K system configuration, in a generic configuration, or with 32K. Once you select the version which best suits your system, you are asked for the input and output file names. Then, you can customize your pre-scan for whichever version of XB you will be using. There are files for the Mechatronics XB, for Myarc's new XB, and for Craig Miller's revisions. There is also a provision for any new CALLs which may be added to XB at a future date.

Next, you can change up to 5 constants from a numerical value to special punctuation symbols to save memory space (for example changing OPEN #1:DSK1.FILE to OPEN #@:DSK1.FILE). You also have the option of deleting REMs. The program will then write a MERGE file containing the

proper pre-scan syntax. How does it work? Well, the program is written in XB, so it is a bit slow. As a test, I ran my BBS program through it and it took an hour, but the BBS is 80+ sectors long with a large number of variables. However, before using PSI, it took about 45 seconds for the BBS to start once I had told it to RUN. After using PSI, it took 6 seconds!!!! I also got about 5 sectors back.

This program really works fantastic. It is written by J Peter Hoddie who is responsible for getting DM1000 and Fast-Term to sit in GRAM1-2 of GK among other things. So, you probably think that an extremely useful program like this costs an arm and a leg? WRONG....Asgard is selling PSI for only \$10!!!! That is one fantastic value. To get your copy, SMAIL Chris at TI9720 or contact an Asgard dealer. This is a MUST HAVE program for everyone.

SPEECH MOD

Before I got a Triple Tech card, one thing that plagued my system was the speech synthesizer hanging on the side. Most programs would crash due to the poor connection. My solution was to just take the thing out of there and not be bothered. But some programs require that the speech box be hooked up and will not operate without it. Well, you can buy a Triple Tech card if you want the clock and buffer, but I realize that all of you are not as affluent as myself (HA). There has been much talk about putting the speech card in the P Box, but it's not as simple as just plugging in the card (although I have seen breadboard jobs done).

Here's how to put the speech synthesizer where it belongs...in the console. First, take the circuit board out of the shell and remove the big female connector. This is done by carefully prying up each contact as you hit it with a low-wattage soldering iron. Some type of solder-sucker would be very useful whether it is braid or a solder-sucker.

Next, you'll need to take your console apart so that you can mount the card on top of the motherboard case. I won't tell you how to take the console apart since you've probably done it by this time. It is pretty easy if you just pay attention. Get some thin cardboard (like what they package with shirts) and some double-stick tape so you can insulate the circuit board from the shield on the motherboard.

You'll need a piece of 8-conductor ribbon cable and another wire of equal length. Cut the length to whatever is needed to fit the speech board inside where there will be no obstructions when the case is reassembled. You will need to split the ribbon cable a couple of inches or so to do the soldering. Make sure you have neat ends on the cable....neatness counts. Now, refer to the diagram below for the pin designations on the speech board. The pins are labeled as if you were viewing the circuit board from the side, looking at where you took the connector off.

44	TOP	2
+-----+ The top pins are even numbered and the		
43	BOTTOM	1
+-----+		

Connect the ribbon cables to the following pins: TOP 2 12 34 36 38 40 42 44 BOTTOM 1 3 5 19 35 37 39 43 Connect the odd wire to #21 which is the big pad on the bottom where there were 4 wires going to it. This is the earth ground. Now all you have to do is connect these wires to the side connector of the console in the places they would have gone if the speech box were still there. REMEMBER. neatness counts!!!!

Once you're through, use the double-stick tape to mount the circuit

board to the motherboard shield and make sure you use some kind of insulation so that it doesn't short out. Reassemble the console and give it a try. Everything should work as if the speech synthesizer were plugged in the side. NOTE: I take no responsibility for any screw-ups anybody makes doing this..it does work, but don't blame me if you fry your console from sloppy workmanship.

DISK DRIVE POWER SUPPLY

By Ken Hamai

This project requires some skills and knowledge in electronics assembly. Incorrect assembly could result in burning up your disk drive. If you are not sure how to connect the parts, contact me or somebody who can help. In any event, neither I or the ROM will cover you for any damages or losses resulting from the use of this power supply as suggested by this article and you are using it solely at your own risk.

PARTS LIST

1. Radio Shack 277-1016 power supply chassis
2. Radio Shack 273-1511 12.6volt, 3amp transformer
3. 1/2 amp fuse and holder
4. SPST bat switch
5. 5 ft. of lamp cord and plug
6. Three 6 inch lengths of 20-22ga stranded wire, different colors
7. Male plug for disk drive power connector
8. Small piece of heat shrink tubing 1/16th inch size

All of the above items except for item 7 are available at Radio Shack stores. The disk drive connector plug is available from one of our advertisers, R and D Electronic Supply, 100 E. Orangethrope Ave. Anaheim, CA 92901.

When you purchase the power supply board, you will note that it comes with instructions on a suggested wiring scheme. These instructions also recommend the use of an 18volt transformer and 2amp fuse. The reason I recommend the other transformer is because the power supply does not have to work so hard to regulate the output voltages and the lower amperage fuse gives quicker response to an overload. An added plus is that the 12.6volt transformer costs less.

Step 1. - Connect the transformer and the power supply as shown in the Radio Shack diagram that comes with the power supply.

Step 2. - Double check your connections and then plug in your supply to an outlet to test it out. Be sure to turn on the power supply switch located on the board (see fig. 3). Using a suitable volt meter and fig. 2, check that you get the indicated output voltages when you test the +5 and +12 pins and ground. The voltages MUST be pretty close. DO NOT use the power supply if you find it is off by 1/2 volt or more, especially on the +5 volt pin. If the voltages are way off, I suggest you return the board and get another one.

Step 3. - Disconnect the power to the supply and carefully bend the -5 volt pin out of the way or cut it completely off. Then solder one of the 6 inch lengths of 20-22ga wire to each of the remaining pins. Use a piece of heat shrink tubing over each soldered connection for insulation. Use yours or a friend's blow dryer at the High setting to shrink the tubing.

Step 4. - Referring to fig. 1, assemble the three wires you soldered to the pins into the power connector for the disk drive. Double check your wiring and test the connector with your voltmeter to be sure that you have the wires in the correct socket positions.

That's all there is to the wiring. If you connected up your disk drive now, it should work.

One more thing, I have not included plans for a cabinet for the disk drive and power supply. You will need to build one to hold your components together. For my demo model, I attached the transformer, fuse holder, off-on switch, and disk drive to a piece of plywood and covered the whole thing with a piece of cardboard to keep the fingers and dust out. Since various models of 5-1/4 inch floppy disk drives can be used, I suggest you take your own measurements for the design of the cabinet.

Using the above and one of the low power demand disk drives you should be able to set up a double side double density drive AND this power supply for less than \$50.00, including the cardboard and nails.

LINGO

One of the more interesting aspects of my last trip to Australia (and the impending one as well) is the difference in language between the two countries. Allegedly, we both speak the same language but there are enough differences between the Aussie lingo and our American variety to make things a bit troublesome. Everyone knows that "fair dinkum" means "genuine" in some instances but not always exactly!

"Pulling the scab off a tinny" translates into pulling the tab off a can of beer. Or, you can "plow the froth off a couple" and accomplish the very same thing. The phrase "to go walkabout" means that you are going on a journey to explore new areas. From there, it can get a bit more complicated!

"Fish and chips" are what we call fish and french fries. Potato chips (what you may have wrongly surmised from the first phrase) are "crisps". A "biscuit" is a cookie and a "scone" is a bisquit.

A "bum" is the side of your anatomy that you sit on --which gives an entirely new meaning to the American phrase "a bum' rush". Just imagine what colorful phrases can be concocted from innocent American slang!

Through it all, I'll probably have a "bonzer" time and come back with all sorts of information about new developments for the TI. After all, amongst the lot is a fair dinkum plethora of new gear. Look for a report at the March meeting!

FWEB INTERFACES

By Tony McGovern

At various times the HV99 Newsletter has contained articles on how to interface programs to the Funnelweb system. As the TI-99/4A computer contains much of its system in ROM of one persuasion or another, the Funnelweb system has to overlay this, but at the same time it unifies functions that TI assigned to separately sold modules. Like any other substantial operating system it contains data items, memory pointers, and various utility routines, in this case held together by a reference block at the top of high memory. This has grown somewhat haphazardly over the years, but at this stage of the history of the TI-99/4A computer, plans for complete tidy up and rewrite have been shelved although almost all functional improvements and more, previously intended for that grand

revision have been incorporated in Vn 4.40 of Funnelweb. So here is the state of the art for the swansong of the HV99 group.

The information and facilities available include BLWP vectors for a variety of services, indirect BL routine pointers, and various data, pointer, and flag words and bytes. Specifications of various useful BLWP and BL routines and flags, are specified in FEWB/REPT in the Funnelweb documents. The XOP instruction is poorly supported on the 99/4A, and is left for application programs to use.

A) BLWP UTILITY ROUTINE VECTORS
~~ ~~~~ ~~~~~~ ~~~~~~ ~~~~~~

(1) DSRLNK EQU >FFD4

This provides a DSRLNK function equivalent to that in the Ti-Writer Editor but corrected to be compatible with multiple RS232 cards, and with entry data recording as in the E/A DSRLNK. It is for file access only and does NOT take a following DATA 8 or DATA >A instruction. This seemed like a good idea in pre-historic times when Funnelweb was only a Ti-Writer interface and has stuck for reasons of consistency ever since. Never mind, this sort of thing happens all the time --MS-DOS smells like CP/M and Intell's 1486 microprocessor still suffers gigantic hangovers from the 8088.

After the usual setup of a PAB in VDP and PAB pointer in PAD, call DSRLNK as

```
BLWP @DSRLNK
JEQ error handler
```

Further error trapping from the PAB error bits and GPL status is up to the programmer. Funnelweb utilities regularly use this DSRLNK for subprogram access by the code sequence

```
DSRTYP EQU >7A
MOV @DSRLNK+2,R11
INCT @DSRTYP(R11)
BLWP @DSRLNK
JEQ DSERR

DECT @DSRTYP(R11)      sucess exit

DSERR DECT @DSRTYP(R11) failure exit
```

The offset value DSRTYP will be maintained at this value. All exits must restore the offset value correctly.

(2) KSCANA EQU >FFD0

This is an enhanced KSCAN routine which may be used in place of the normal E/A utility. It provides some extra functions transparently setting processor status bits, which may then be tested by various conditional Jump instructions as required. There are 6 condition bits available, and the tests check these seperately.

```
BLWP @KSCANA

JNC --> <escape> --> Jump
```

```

JOP --> <No key> --> Jump
JNO --> <proc'd> --> Jump
JEO --> <Oldkey> --> Jump
JGT --> <Enter> --> Jump
JH -- <Redo> --> Jump

```

<Escape> senses both <Fctn-9> and Ctrl-C> and <proceed> checks both <FCTN- > and <CTRL-A>. Finally an unconditional JMP maybe used to loop on the keyscan after one or more other tests.

```
(3) FILENT EQU >FFCC
```

This provides a single line Editor typically used for entry of file and device names. and may be used in text mode as well from Vn 4.40 onwards. The flashing cursor has auto-repeat with acceleration, and the timing parameters are set on entry to Funnelweb. Editing keys are as described for filename entry. FILENT waits on entry until the previous key is released. Typical calling sequence is

```

BLWP @FILENT
DATA <scrpos>
BYTE <init offset>
BYTE <length>

```

The initial offset gives the initial position of the cursor in the entry window, with null string being the first position. The CPU buffer is assigned after entry to the routine. immediately following the workspace. The workspace is at >8302 in the PAD area, so the maximum buffer length can extend at most to the end of the DSR transient area. A calling sequence that allows another workspace/buffer area to be assigned is

```

BLWP @FWFILN
DATA <scrpos>
DATA <ofs,len>

```

```

FILENT EQU $
LI R11,FEWSP           Workspace for BLWP vector in R11
MOV @FILENT+2,R12     FILENT code pointer in R12
MOV *R14,@FILNT1     Screen pointer
MOV *R14,@FILNT2     Line window data
BLWP R11              Execute F'web FILENT in FEWSP
FILNT1 DATA >0       Data written in
FILNT2 DATA >0       Also
RTWP

FWFILN DATA TRANWS,FILNT   User "FILENT" BLWP vector
DATA >0,>0,>0,>0,>0         R11-R15 of transfer workspace
TRANWS EQU $->20
FEWSP BSS >20
FILBFR BSS <as needed>

```

This is a lot simpler than writing your own routine, and may be necessary if you are already using the PAD area. It is possible to be even trickier in the transfer code but that will do.

Several auxiliary flag locations are associated with use of FILENT.

```
LOWCAS EQU >FF22      Lowercase allowed
HEXDIG EQU >FF24      HEX digits only
ESFLAG EQU >FF60      <escape> flag/pointer
```

When LOWCAS is null, the normal condition, all entries are converted to upper case as in the Editor SF/LF. FILENT always resets LOWCAS to null on exit. SETO of LOWCAS before calling FILENT allows lower case alpha characters to slip through. If HEXDIG is set before entry only valid hex digits can be netered and deletes produce "0" instead of blanks. Write a valid hex entry on the screen first as this is not checked.

If ESFLAG is null the the <escape> key is ignored internally and the key value passed out. SETO of ESFLAG is reserved for internal purposes. Any other value is treated as an <escape> address pointer. First instruction in the <esc> routine should be a LWPI as the <esc> sensing occurs in the internal CURSEOR (see later) routine and will exit with the FILENT registers set.

(4) DELSPR EQU >FFC8

This one just writes byte >D0 to VDP memory at >300 to shut off the sprite list in Graphics Mode I with normal E/A table positions. It is useful in quick cleanups after switching back from Text Mode. Just call as is.

(5) VMBWD EQU >FFBA

VMBWD gives a VMBW function from in-line data, which saves program space when using fixed value data. Null length is just ignored. Call as

```
BLWP @VMBWD
DATA <scrpos>
DATA <CPU data>
DATA <length>
```

The primary Funnelweb workspace is at >FF7C and R9 of this workspace always points to VMBWD so that it may be called from this workspace as BLWP *R9 making a 4 word total call.

(6) VMBRD EQU >FFB6

This is the VDP version and is essentially similar to VMBWD.

(7) VSDRD EQU >FFB2

The calling sequence for this is

```
BLWP @VSDRD
DATA <scrpos>
```

and it returns the byte in VDP memory at <scrpos> to the MSB of R0 (not the usual R1 in E/A code).

(8) VFILL EQU >FFAE

This fills a block of VDP RAM with the byte value in the MSB of R0. The Funnelweb main workspace FWREGS EQU >FF7C maintains R8 as a pointer to

this so it may be called as BLWP *R8 etc. Standard calling sequence is

```
LI R0,<MSB value>
BLWP @VFILL
DATA <scrpos>
DATA <length>
```

- (9) VMBWR EQU >FED4
- (10) VMBRR EQU >FED0
- (11) VSBWR EQU >FECC
- (12) VSBRR EQU >FEC8

This bunch of VDP utilities is equivalent to the standard E/A utility set in usage. The VM routines check for and ignore null length values in R2.

- (13) VSTRW EQU >FEC4

This writes the body of a string with leading length byte to VDP. Call as

```
LI R0
BLWP @VSTRW
DATA <scrpos>
```

If the length byte of the string is null then the call is ignored.

- (14) CURSOR EQU >FEC0

The CURSOR routine is called internally by FILENT, but may also be called externally. The calling sequence is

```
LI R6,<scrpos>
BLWP @CURSOR
```

CURSOR flashes at the screen position specified in the calling R6 and returns the key-value in the MSB of the calling R2, and this is left in raw state at console KEYRT (>8375). See FILENT for a discussion of <escape>. Normally CURSOR provides the up-case function for FILENT, so the LOWCAS flag is effective here also. It is not cleared however except by the <escape> path. If you want to use CURSOR externally with hex digit validation, you will need to supply your own validation routine. A table of hex digits "01 . .EF" is provided at

HXTAB EQU >FEA4

The cursor flash rate and auto-delays are slaved indirectly to the VDP vertical interrupt rate. On entry to Funnelweb the number of console keyscans that occur in an interval of 14 vertical interrupts is counted and stored at

REPETS EQU >FEE4

This does not account for the difference between PAL and NTSC consoles, but this will be minor compared to possible Geneve to 99/4A difference.

(14)DSRREN EQU >FEBC

DSRREN is intended to give a direct DSR re-entry from saved values, as is done in the E/A object loader, or more comprehensively in LINEHUNTER. It re-enters the last DSR ROM at the same entry point directly without having to search for it. All PAB information necessary must be supplied as normal. It is used in DiskReview (80-col) as the new speedup for Vn 4.40 of the Myart file viewer. It is of course not necessary to set the DSRTYP again for sub-program (eg. sector read) re-entry as this is only needed to find the entry point of the original search.

The relevent DSR values are stored in a 4 word block starting at

SAVENT EQU >FF46

on every full DSRLNK call.

(15)QCODE EQU >FEB8

This is mostly of use internally after the AID key has been detected in the path with SETO of ESFLAG (>FF60). It checks the value at AIDFL (>FF3A) and if null returns directly, else it branches to the address pointed to by AIDFL. Usually this is a QD test/load/branch routine.

(16)SETGRD EQU >FEB4

This sets the GROM address using the system GROM pointer in GPL R13, so that Module Library banking is supported. Call as

```
LI R0,<GROM address>
BLWP @SETGRD
```

(B) INDIRECT DL ROUTINE POINTERS
~~~~~

All of these routines are conveniently call by

```
LI @BLPTR,R11
BL *R11
```

together with whatever register and/or in-line data is appropriate for the particular BL call.

(1) CFILE# EQU >FFA6

This takes the trailing DATA item >0n00 where "n" is the number of file buffers to be set aside in VDP as would be done with CALL FILES(n) from BASIC. The routine is compatible with V-9938 based systems, and rewrites MAXMEM (>8370) TO SUIT. A new VDP header is written for Ti/CorComp disk controllers and VDP cleared above that to the top of the disk DSR area. No range checking is done on "n".

(2) KEY EQU >FFA4

This keyloop enables interrupts, follows the previous prescription for <escape> and ESFLAG (>FF60), and calls QDCODE if AID is pressed. The key return is placed in the MSB of R2 and left unchanged at KEYRT (>8375).

(3) RESTR EQU >FFA2

This does various housekeeping tasks to set up the E/A Graphics Mode I for the Funnelweb Central Menu screen. It resets the VDP registers to E/A default, rewrites the color table, clears the screen, calls DELSPR, restores the CHR(>81) pattern to underline, and restores the key-unit to 5.

(4) RDDEV EQU >FFA0

The RDDEV routine handles the details after screen entry of a file or device name, saving the device name to CPU RAM, and building a PAB in VDP for DSRLNK. RDDEV is called externally as

|                     |                          |
|---------------------|--------------------------|
| MOV @RDDEV,R11      | Get pointer              |
| BL *R11             | Go to routine            |
| DATA <screen ptr>   | Entry pointer on screen  |
| DATA <cpu ram bufr> | CPU RAM buffer for name  |
| DATA <vdp PAB adr>  | VDP address for PAB      |
| DATA <cpu PAB data> | PAB data storage address |

The first data item is the VDP pointer for the filename as entered on screen. RDDEV parses this up to the first space and enters it with leading length byte at the CPU RAM buffer pointed to by the second data item. It then builds a PAB in VDP at the address in the third data item. The CPU RAM buffer may well be a continuation of the PAB data, but need not be. While at it, RDDEV clears the GPL status byte (>837C) and loads SCNAME (>8356) for DSRLNK.

(5) QDLOAD EQU >FF3C Uses R0, R1, R2, R10, R12

System files with 2 character filenames in E/A program file format may be loaded without executing by QDLOAD. A typical calling sequence is

|                    |                      |
|--------------------|----------------------|
| MOV @QDLOAD,R11    | Get pointer          |
| BL *R11            | Execute              |
| DATA 'UL'          | 2-char filename      |
| DATA ULPOS         | Load position in CPU |
| DATA ULLEN         | Load length          |
| JMP <error return> |                      |

Successful loads step over the error branch. The normal E/A header is ignored in favor of the load target position and length. The maximum file length allowed is >1100 which is long enough for QD or QF. The file is loaded from the Funnelweb boot path as for other system files.

(4) FILCLN EQU >FEDC Uses R0, R1, R10, R14

This routine cleans up the mailbox (>A000 - >A050) area, for internal system use and may be subject to change in function or auxiliary conditions. Some not already mentioned which may be useful are

NAMBUF EQU >FF62

The space from NAMBUF to FWREGS (>FF7C) is used for a filename buffer by the various loaders. It is long enough to accommodate length byte and 25 character names (DSK.volname.filename or whatever else fits). Increase in this was the one change that would have forced major incompatibilities with previous versions, and so it has been left as is for better or for worse.

BTLN EQU >FF58

This points to the length byte of the boot pathname and all references to system filenames are by offsets from this value. The Editor printer device name, as a survival from much earlier versions, though part of this block, has its own pointer at EDPRNT EQU >FF1E. All these path/file and device names are stored with leading length byte, unlike the workfile name in the mailbox at >A000 which is stored without length byte as required by the Editor.

INCOL EQU >FF26

The word value here is an index range 0-9 into the 10 byte table of color bytes for use in VReg #7, and pointed to by COLRS EQU >FF1C. Standard Funnelweb system practice is always to use this table as the source of color combinations, and to use INCOL as the index. This helps avoid unexpected and jarring color changes on return.

CPUDEL EQU >FEE2

The value here estimates CPU speed as seen from normal expansion RAM by counting a loop against the VDP interrupt timer. It is handy as a value for CPU delay timing of beeps and bleeps. Use REPETS (see earlier) for keyloop timing.



```

100 CALL CLEAR :: PRINT " : 210 IF POS(FN$, ".", 1) <> THE : 390 !L$=STR$(ASC(S$) * 256 + ASC : 630 NEXT J :: PRINT S$ :: IF
'CRUNCH' :: PRINT :: N 230 : (SEG$(S$, 2, 1)) : PHASE=2 THEN PRINT #2: S2$ :
PRINT " by Tim MacEac : 220 FN$="DSK1."&FN$ : 400 COMMAND=ASC(SEG$(S$, 3, 1) : : PRINT S2$
hern" :: PRINT " PD Bo : 230 IF PHASE=1 THEN NTAB=0 : ):: NOTDATA=COMMAND<>147 AND : 640 NEXT I
x 1105" :: PRINT " Dar : 240 DIM TOKEN$(255):: FOR I= : COMMAND<>158 AND COMMAND<>1 : 650 CLOSE #1 :: IF PHASE=2 T
tmouth, NS" : 255 TO 255 :: TOKEN$(I)="& : 54 AND COMMAND<>131 :: FOR J : WHEN PRINT #2: CHE$ :: CLOSE #
110 PRINT " B2Y 48B Ca : STR$(I)&" : =3 TO LEN(S$)-1 :: C=ASC(SEG : 2 :: PRINT "FINISHED, SAVING
nada" :: PRINT :: PRINT :: P : $(S$, J, 1):: IF (C=200)+(C=1 : ";DOSAVE;"BYTES" :: STOP
RINT :: PRINT :: FOR DELAY=1 : 56 :: READ I, S$ :: IF I<0 TH : 99) THEN 360 :
TO 1000 : EN 280 : 410 IF C<>201 THEN 440 : 660 REM PRINT STRING TABLE
120 NEXT DELAY :: CALL CLEAR : 260 TOKEN$(I)=S$ : : 670 INPUT "LISTING VARIABLES
:: PRINT :: PRINT " Purpose : 270 NEXT J : 420 !L$=L$&BL$(STR$(CHAR(J+1 : AND CONSTANTS. ENTER
:" :: PRINT "Shorten Basic & : 280 DEF CHAR(J)=ASC(SEG$(S$, : ) * 256 + CHAR(J+2))) : AND CONSTANTS. ENTER
XBASIC" :: PRINT "programs : J, 1):: CHE$=CHR$(255)&CHR$( : 430 J=J+2 :: GOTO 630 : MINIMUM OCCURENCE COUNT T
by substituting" :: PRINT "s : 255) : 440 IF C<127 THEN 520 : 0 LIST : "MC :: PRINT : "ST
ingle character variable" :: : 290 OPEN #1:FN$, VARIABLE 163 : 450 FOR JJ=J+1 TO 999 :: C=A : TO NTAB :: IF TABLE(1, I) < NC
PRINT "names for longer nam : , INPUT, DISPLAY :: IF PHASE= : SC(SEG$(S$, JJ, 1)):: IF (C>12 : THEN 690
es and" : 1 THEN 380 : 71)+(C=0) THEN 470 : 680 PRINT TABLE(1, I); TAB(3);
130 PRINT "frequently used c : 300 PRINT "TOTAL SAVINGS AC : 460 NEXT JJ : TABLE(1, I)
onstants." :: PRINT :: PRINT : HIEVED WILL BE";DOSAVE;"BYTE : 470 T$=SEG$(S$, J, JJ-J) : 690 NEXT I :: INPUT "PRESS E
" To run:" :: PRINT "(1) En : S." :: INPUT "ENTER NEW FILE : : ENTER TO CONTINUE " : T$
sure that your program" :: P : NAME FOR MERGEFILE: ".FN2$ : : 480 !L$=L$&BL$(T$) : 700 PRINT "SINGLE CHARACTER
RINT "does not have lines nu : : IF POS(FN2$, ".", 1)=0 THEN : 490 IF NOT NOTDATA THEN 510 : VARIABLE NAMES NOT YET IN
mbered" :: PRINT "1 to 10 (o : FN2$="DSK1."&FN2$ : 500 GOSUB 1070 :: IF PHASE=2 : USE" :: : SN$="" :: S$="" :
r so). Statements" : 310 OPEN #2:FN2$, VARIABLE 16 : AND TABLE(2, P) <> "" THEN S2 : : FOR II=1 TO 2 :: FOR I=ASC
140 PRINT "at these lines ar : 3, OUTPUT, DISPLAY : $=SEG$(S2$, 1, J-1+LEN(S2$)-LE : ("A") TO ASC("Z")):: T$=CHR$(I
e generated" :: PRINT "by CR : : N(S$))&TABLE(2, P)&SEG$(S2$, : )&S$ :: FOR J=1 TO NTAB :: I
UNCH if you move a" :: PRINT : 320 ! SET CONSTANT VARS : JJ+LEN(S2$)-LEN(S$, 999) : F T$=TABLE$(1, J) THEN 720
"constant into a variable." : 330 FOR I=1 TO NTAB :: IF TA : 510 J=JJ-1 :: GOTO 630 : 710 NEXT J :: PRINT T$; " :
:: PRINT "(2) Save your pro : BLE$(2, I)=" OR TABLE$(1, I)= : : SN$=SN$&CHR$(I)
gram in" :: PRINT "merge for : "0" OR TABLE$(1, I)=(CHR$(34) : 520 REM TOKENS : 720 NEXT I :: S$="" :: IF I
mat to disk." : &CHR$(34)) THEN 350 : 530 !L$=L$&BL$(TOKEN$(C)) : 1=1 THEN NN$=SN$ :: SN$=""
150 PRINT "(3) Run CRUNCH, p : 340 IF (ASC(TABLE$(1, I))>AS : C("0") AND ASC(TABLE$(1, I))<= : C(SEG$(S$, J+1, 1)) : 730 NEXT II :: PRINT :: PRIN
roducing a" : ASC("9")) OR ASC(TABLE$(1, I) : =ASC(".") OR ASC(TABLE$(1, I) : COMMAND<>158 AND COMMAND<>1 : T "DO YOU WANT TO MODIFY THE
160 PRINT "new merge format : =34 THEN GOSUB 360 : 54 AND COMMAND<>131 :: GOTO : " :: INPUT "PROGRAM? " : ANSWE
file." : 350 NEXT I :: GOTO 380 : 630 : R$ :: IF NO THEN RUN "DSK1.L : " : " :
170 PRINT "(4) Enter NEW in : 360 !XX=IXX+1 :: IF POS(TABL : 560 LN=ASC(SEG$(S$, J+1, 1)):: : 740 REM ###SHORTEN PROGRAM :
XBASIC and" :: PRINT "merge : E$(1, I), CHR$(34), !)=0 THEN T : T$="" :: IF C<>199 THEN 580 : $ :
in the new program." :: PRIN : $=TABLE$(1, I) ELSE T$=SEG$(TA : 570 T$=CHR$(34) : 750 REM PRINT POTENTIAL SAVI
180 NEXT DELAY :: CALL CLEAR : BLE$(1, I), 2, LEN(TABLE$(1, I) : 580 T$=T$&SEG$(S$, J+2, LN)&T$ : NGS
:: DEF PC(X)=((X=36)+(X=34) : -2) : 590 !L$=L$&BL$(T$) : 760 PRINT "POTENTIAL SAVINGS
-(X>47)*(X<58)-(X<64))*(X<91) : 370 S2$=CHR$(INT(IXX/256))&C : HR$(IXX-256)/INT(IXX/256))&A : " :: PRINT :: TOTSAV=0 :: FO
)<> : BLE$(2, I)&CHR$(190)&CHR$(200 : +(T$<>TABLE$(1, I))&CHR$(LEN : (T$))&T$&CHR$(0) :: PRINT S2$ : R I=1 TO NTAB :: GOSUB 910
190 !DEF BL$(Z$)=SEG$( " ", 1, : :: PRINT #2: S2$ :: RETURN : 610 GOSUB 1070 :: IF PHASE=2 : 770 IF SAV>0 THEN PRINT "CHA
PC(ASC(SEG$(L$, LEN(L$), 1)))& : 380 FOR I=1 TO 99999 :: LIMP : AND TABLE(2, P) <> "" THEN S2 : ;SAV;"BYTES"
PC(ASC(Z$))&Z$ : UT #1: S$ :: S2$=S$ :: IF S$= : $=SEG$(S2$, 1, J-1+LEN(S2$)-LE : NGING ";TABLE$(1, I);" SAVES"
200 DEF BL$(Z$)=Z$ :: DEF YE : CHE$ THEN 650 : N(S$))&TABLE(2, P)&SEG$(S2$, : ;SAV;"BYTES"
S=POS(ANSWER$&"Y", "Y", 1)=1 : 620 J=J+LN+1 : J+LN+2+LEN(S2$)-LEN(S$, 999) : 780 TOTSAV=TOTSAV+SAV
: DEF NO=YES=0 :: OPTION BAS : : 620 J=J+LN+1 : :
E 1 :: DIM TABLE(2, 500), TAB : : :
LE(3, 500):: PHASE=1 :: INPUT : : :
"ENTER FILENAME : " : FN$ : :

```

```

790 NEXT I :: PRINT "MAXIMUM 940 REM NUMERIC CONSTANT 1220 DATA 132,IF 1760 DATA 192,>
N TOTAL SAVINGS THAT YOU CA 950 SAV=(LEN(TABLE$(1,I))+1) 1230 DATA 133,60 1770 DATA 193,+
N MAKE BY RENAMING VARIAB 11(TABLE(1,I))-1)-8-14-103(TAB 1240 DATA 134,60TD 1780 DATA 194,-
LES AND CONSTANTS IS ";TOTS LE$(1,I)="0"):: SAV=MAX(0,SA 1250 DATA 135,60SUB 1790 DATA 195,$
AV;"BYTES." :: INPUT "CONTIN V):: IF SAV(<=0 THEN 1060 1260 DATA 136,RETURN 1800 DATA 196,/
UE? "ANSWER$ :: IF NO THEN 960 GOTO 1060 1270 DATA 137,DEF 1810 DATA 197,^
RUN "DSK4.FWFB" 1280 DATA 138,DIM 1820 DATA 202,EOF
800 PRINT "SELECTING ELEMEN 970 REM STRING CONSTANT 1290 DATA 139,END 1830 DATA 203,ABS
TS TO RECODE" :: DOSAVE=0 : 980 SAV=MAX(0,(LEN(TABLE$(1, 1300 DATA 140,FOR 1840 DATA 204,ATN
: FOR I=1 TO NTAB :: J=0 :: I))-2)*(TABLE(1,I))-1)-9-8-12 1310 DATA 141,LET 1850 DATA 205,COS
BEST=0 :: FOR I=1 TO NTAB : *(TABLE$(1,I)=""):: GOTO 10 1320 DATA 142,BREAK 1860 DATA 206,EXP
: GOSUB 910 :: IF SAV>BEST T 1330 DATA 143,UNBREAK 1870 DATA 207,INT
HEN J=1 :: BEST=SAV 1340 DATA 144,TRACE 1880 DATA 208,LDS
810 NEXT I :: IF J=0 THEN PH 990 REM VARIABLE 1350 DATA 145,UNTRACE 1890 DATA 209,SGN
ASE=2 :: GOTO 290 1000 IF SEG$(TABLE$(1,I),LEN 1360 DATA 146,INPUT 1900 DATA 210,SIN
820 TABLE(1,J)=0 :: NUMVER=P (TABLE$(1,I),I)="$" THEN 10 1370 DATA 147,DATA 1910 DATA 211,SQR
OS(TABLE$(1,J),"*",1)=0 AND 40 1380 DATA 148,RESTORE 1920 DATA 212,TAN
POS(TABLE$(1,J),CHR$(34),1)= 1390 DATA 149,RANDOMIZE 1930 DATA 213,LEN
0 :: IF NUMVER AND LEN(NN$)= 1010 REM NUMERIC VARIABLE 1400 DATA 150,NEXT 1940 DATA 214,CHR$
0 THEN 900 1020 SAV=MAX(0,(LEN(TABLE$(1, 1410 DATA 151,READ 1950 DATA 215,RND
830 IF NOT NUMVER AND LEN(SN $),I))-1)*TABLE(1,I):: IF SAV 1420 DATA 152,STOP 1960 DATA 216,SEG$
$)=0 THEN 900 (<=0 THEN 1060 1430 DATA 153,DELETE 1970 DATA 217,POS
840 PRINT "CHANGING ";TABLE$ 1030 GOTO 1060 1440 DATA 154,REM 1980 DATA 218,VAL
(1,J);" SAVES";BEST;"BYTES" 1450 DATA 155,ON 1990 DATA 219,STR$
:: INPUT "DO YOU WANT TO DO 1040 REM STRING VARIABLE 1460 DATA 156,PRINT 2000 DATA 220,ASC
IT? "ANSWER$ :: IF YES THEN 1050 SAV=MAX(0,(LEN(TABLE$(1, 1470 DATA 157,CALL 2010 DATA 221,PI
870 ,I))-2)*TABLE(1,I):: IF SAV 1480 DATA 158,OPTION 2020 DATA 222,REC
850 INPUT "DO YOU WANT TO DO (<=0 THEN 1060 1490 DATA 159,OPEN 2030 DATA 223,MAX
MORE? "ANSWER$ :: IF YES T 1060 RETURN 1500 DATA 160,CLOSE 2040 DATA 224,MIN
HEN 900 1510 DATA 161,SUB 2050 DATA 225,RPT$
860 PHASE=2 :: GOTO 290 1070 REM TABULATE STRINGS 1520 DATA 162,DISPLAY 2060 DATA 232,NUMERIC
870 PRINT "SELECT FROM THE A 1080 GOSUB 1090 :: GOSUB 117 1530 DATA 163,IMAGE 2070 DATA 233,DIGIT
VAILABLE LETTERS:" :: DOSA 0 :: RETURN 1540 DATA 164,ACCEPT 2080 DATA 234,UALPHA
VE=DOSAVE+BEST :: IF NUMVER 1090 P=ROOT 1550 DATA 165,ERROR 2090 DATA 235,SIZE
THEN PRINT NN$ :: AV$=NN$ EL 1100 IF P(<>0 THEN 1140 1560 DATA 166,WARNING 2100 DATA 236,ALL
SE PRINT SN$ :: AV$=SN$ 1110 ROOT=1 1570 DATA 167,SUBEXIT 2110 DATA 237,USING
880 PRINT "ENTER SELECTION: 1120 NTAB=NTAB+1 :: TABLE$(1 1580 DATA 168,SUBEND 2120 DATA 238,BEEP
" :: ACCEPT AT(23,18)VALIDAT ,NTAB)=T$ :: IF P(<>0 THEN TA 1590 DATA 169,RUN 2130 DATA 239,ERASE
E(AV$)SIZE(1):TABLE$(2,J):: BLE(3+(TABLE$(1,P))T$,P)=NT 1600 DATA 170,LINPUT 2140 DATA 240,AT
AV$=SEG$(AV$,1,POS(AV$,TABLE 1610 DATA 176,THEN 2150 DATA 241,BASE
$(2,J),1)-1)&SEG$(AV$,POS(AV $,TABLE$(2,J),1)+1,999) 1130 P=NTAB :: RETURN 1620 DATA 177,TO 2160 DATA 243,VARIABLE
890 IF NUMVER THEN NN$=AV$ E 1140 IF TABLE$(1,P)=T$ THEN 1630 DATA 178,STEP 2170 DATA 244,RELATIVE
LSE SN$=AV$ :: TABLE$(2,J)=T 1150 IF TABLE(3+(TABLE$(1,P) 1640 DATA 179,"" 2180 DATA 245,INTERNAL
ABLE$(2,J))&"$" >T$,P)=0 THEN 1120 1650 DATA 180,";" 2190 DATA 246,SEQUENTIAL
900 NEXT I3 :: PHASE=2 :: GD 1160 P=TABLE(3+(TABLE$(1,P)) 1660 DATA 181,";" 2200 DATA 247,OUTPUT
TO 290 T$,P):: GOTO 1100 1670 DATA 182,")" 2210 DATA 248,UPDATE
910 IF SEG$(TABLE$(1,I),1,1) 1680 DATA 183,( 2220 DATA 249,APPEND
>="A" THEN 990 1690 DATA 184,& 2230 DATA 250,FIXED
920 REM CONSTANT 1700 DATA 186,OR 2240 DATA 251,PERMANENT
930 IF ASC(TABLE$(1,I))=34 T 1180 TABLE(1,P)=TABLE(1,P)+1 1710 DATA 187,AND 2250 DATA 252,TAB
HEN 970 :: RETURN 1720 DATA 188,XOR 2260 DATA 253,$
1190 DATA 129,ELSE 1730 DATA 189,NOT 2270 DATA 254,VALIDATE
1200 DATA 130,"::" 1740 DATA 190,= 2280 DATA -1,-1
1210 DATA 131,"!" 1750 DATA 191,<

```