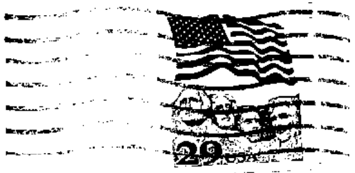
THE R O M NEWSLETTER
USERS GROUP OF ORANGE COUNTY
17161 EDWARDS STREET
HUNTINGTON BEACH, CA 92647



DALLAS TI COMPUTER GROUP (DTIHCG)
PO Box 29863
Dallas
TX 75229

# UGOC ROM

OCTOBER 1991

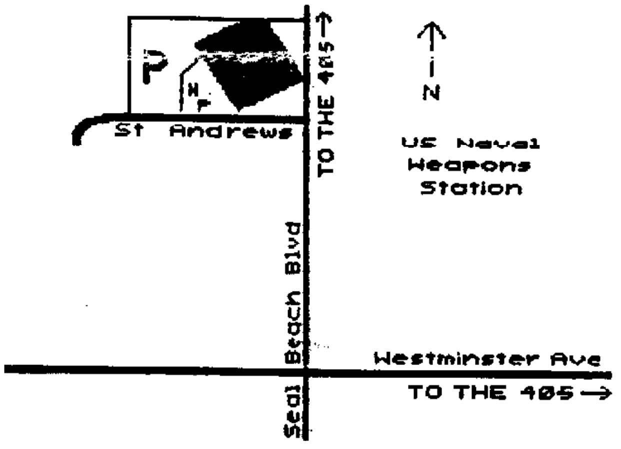
SERVING THE TI 99/4A HOME COMPUTER COMMUNITY

## WE MEET AT FIDELITY FEDERAL

### TIME AND PLACE OF MEETING

The SECOND Monday of each month at
Fidelity Federal Savings
7:30 PM

North of Westminster Ave. at the corner of Seal Beach Blvd and St. Andrews at 13820 Seal Beach Blvd. Parking is availabel west of the building off St. Andrews with additional parking assross the street.
All are welcome.



### U.G.O.C. OFFICERS

| | | |
|---|---|---|
| PRESIDENT | SILES BAZERMAN | 397-2868 |
| VICE-PRES | BILL NELSON | 750-6425 |
| SECRETARY | EUGENE SMITH | 647-3961 |
| TREASURER | ERWIN METZ | 898-2094 |
| PAST-PRES | BEN HATHEWAY | 662-2957 |

### COMMITTEE CHAIRMEN

| | | |
|---|---|---|
| STAN CORBIN | MEMBERSHIP | 892-2818 |
| EARL RAGUSE | NEWSLETTER | 847-5875 |
| BEN HATHEWAY | BULLETIN BOARD | 751-4332 |
| BILL NELSON | HALL OF FAME | 750-6425 |

### SOFTWARE LIBRARY

KNUTE ERSLAND .................. 842-0859

### NEWSLETTER LIBRARY

EARL RAGUSE (PHONE FOR TIME) 847-5875

### PUBLICATION NOTICE

The ROM is published by the Users Group of Orange County, a non-profit organization. Permission is granted to similar groups to reproduce articles published in the ROM provided author and source credits are included. Opinions expressed in this publication are those of the author and do not necessarily represent those of the ROM, it's Editor, or the UGOC, it's officers or members.

We solicit letters and articles of interest to the TI-99/4A user community. Material accepted may be edited for fit and format. No payment is offered nor intended (other than your byline).

### NEWSLETTER CONTRIBUTORS

| | |
|---|---|
| EARL RAGUSE | TI FORTH |
| ADRIAN ROBINSON | ASSEMBLY |
| EARL RAGUSE | CIRCULATION |
| JIM SWEDLOW | AT LARGE |
| BILL NELSON | GRAPHICS |
| SILES BAZERMAN | HARDWARE |

Ra

### TI CLUB ACTIVITIES

| CLUB | ACTION | DATE | INFO |
|---|---|---|---|
| BUG | GENERAL MEETING | | 871-3405 |
| UGOC | LIBRARY. FTNVLY | | 842-0859 |
| UGOC | GENERAL MEETING | | 662-2957 |
| UGOC | LIBRARY. FTNVLY | | 842-0859 |
| UGOC | BOARD MEETING | | 897-2868 |
| UGOC | LIBRARY. FTNVLY | | 842-0859 |
| UGOC | LIBRARY. FTNVLY | | 842-0859 |
| UGOC | NSLETTER LIBRARY | CALL | 847-5875 |

## IN THIS ISSUE
By Editor and Staff

## SEPTEMBER BOARD MEETING MINUTES
By Earl Raguse for Gene Smith

The meeting was held on Sept 19, 1991 at the home of our president, Siles Bazerman. Others present were, Jerry Rash Rash, Erwin Metz, Earl Raguse, and Stan Corbin. The meeting was called to order at 7:30.

Secretary Gene Smith was still singing his favorite song "Back Home In Indiana". Previous meeting minutes published in the ROM were accepted as printed. Earl Raguse acting as secretarty, took these notes on the meeting in Gene's stead.

Treasurer Erwin Metz passed out a report that showed all club bills were currently paid as of that date and we are still solvent.

Stan Corbin, membership chair reported that we have 49 members, a couple of non-renewals had to be dropped. Stan feels we need more elementary TI stuff in the ROM. We will try.

Bill Nelson, was absent as scheduled.

Earl Raguse, ROM Editor, reported that the ROM was nearly ready to put to bed, but that articles would be accepted until Monday, 23 September.

Earl Raguse, Paper Librarian, reported that about 62 newsletters were still coming in every month. Lots of interesting stuff in them.

In order to provide more time to get the minutes ready for printing in the current ROM, Board Meetings will be held on 3rd Thursdays.

Siles will demo DSKU at the October meeting, not DM1000 as reported last month. ~~Jim Swedlow will enlighten us~~ on floppy disk organizational mysteries. In November, Earl will demo XDP a new XB and library enhancement from Australia. In December, Stan will do Christmas cards with TIPS. We plan that at least one library demonstration will be done every month. FW ver 4.4 is out and will be available from the library. We still need a librarian. Any takers?

The meeting was adjorned at 9:15 PM, for the garage SIG, and to to feast on goodies served up by president Siles.

## THE MEMBER SHIP
By Stan Corbin

Now hear this!!!

Saw Ken Hamai at the BUG meeting, told him I was going to send him a notice that it was time to renew his membership. He pulled out his wallett and handed me his membership fees, without further ado. Thanks Ken, we sure like to see that sort of response. With a little stimulation our treasurer, Erwin Metz, came forward with his dues. He gave me his check and then I gave it back so he can deposit it.

By the way did you know that Bill Nelson, our Vice President, will be at the Fest West 92 in Phoenix to represent Asgard Software. Bill has been representing Asgard Software products locally for some time now. Asgard recently awarded him a prize for various pieces of art work he presented to them. If you are interested in graphics programs such as Page Pro, or some artistic fonts, Bill is the man to see. As most of you know, Bill produces the cover page of the Rom with Page Pro and other magic. Bill can be phoned at (714) 750 6425.

Nick Vander Dussen was very generous at the September meeting. Nick had some cartridges and tapes he no longer needed, so rather than hoarding them he contributed them to the Orange Senior Citizen TI99/4A class. Nick has only been with us a year but he has the UGOC "help others" spirit. I hadn't met Nick before the meeting so it was a pleasure getting aquainted with him.

Although I had seen him many times I didn't know who Jules Kerker was, till the September meeting. I have seen his name on the roster many times but was unable to associate him in my mind. I'm pleased to get to know you too, Jules.

Former Membership Chairman, Jim Morris, whom we haven't seen for a long time, re-upped his dues for the coming year. Where ya hidin' out Jim, I'm still waiting to skin you in a game of chess.

Although he lives way up in Downey, Steve Leffler always makes it to the meetings. Steve has a strong interest in working with TI hardware.

The October meeting should be very interesting. It will cover disk drives and sector editing. Most of us can use this information, so if you would like to be enlightened in this area, then be at the meeting, you will learn a lot.

Maintain a good watch and next month we'll be back with more scuttlebut.

## ABOUT THIS ISSUE
From your Editor

Since NO ONE contributed any articles to me this month, I am running an old disk controller article by Jerry Coffey. It is still good stuff even if written in 1987. This is what happened to Hunter Valley of Australia. They used to have one of the best newsletters in the world but, the membership stopped submitting, the Editor can't do it all himself, so no newewsletter, no Group, So Sorry. It could happen here. Think about it. Do not leave it all up to others.

UGOC
BULLETIN
BOARD
300/1200;2400 BAUD
(714) 751-4332

## MAKING A TIPSLABEL LETTERFORM
By Earl Raguse

I have used TIPSLABEL to make border letterforms several times, but I always manage to spoil a few while re-learning what to do, so I am writing it down. Maybe you too can profit from it.

Firstly, it is all done with the Image (only) Option, except the last part, the actual letterhead, which is done with the Text (only) Option. It is assumed you know how to work TIPSLABEL, this is not a beginning tutorial.

1. First, be sure your printer and paper are at TOF (Top of Form), and then (re)-positioned to the line where you want the images to be printed. On all my printers, the way to be sure that that position is in the printer's memory, is to switch power Off/On. I will use TOF to refer to this position in the following, even when it is for printing at the bottom.

2. Select the image for the top row.

3. Select Options, then Multiple.

4. Change the default column 5 to 9.

5. Accept the prompt of 1 space between images.

6. Enter 7 images, no more, no less.

7. Now 7 images should be printed across the top of your paper, starting at column 9 and ending at column 65.

8. Now select FF from the menu to move you to the TOF of next page.

9. Then select Again, from the menu, then 1 to print another.

10. I like to do at least a dozen sheets at a time, so I repeat 8 and 9, but until you are proficient maybe you had better stick to 1 or 2 pages.

11. Rewind your printer to TOF, then move the paper up until you are about ten lines above the bottom. Switch power Off/On again, very important, this is the new TOF.

12. If you plan to put a different image at the bottom of the page, Do FCTN 4 (BREAK) and RUN. This is to prevent TIPSLABEL from printing an underline under all EVEN numbered images. I will tell you an easier way around this in an article on TIPSLABEL, if I ever get it written, else select Again and 1, and go back to 8, else continue.

13. If you now have multiple images on top and bottom of your letterform, lets proceed to do the sides.

14. Again do BREAK RUN, select your new image, then Options, then Image (only), then Single.

15. You will then asked At What Column to Put it?, enter 00.

16. Next, answer 1, to the How Many query. This is a test. The image will be printed on the left side of the paper. If you got your paper lined up right, it will be in line with the previous images, if not, adjust. Then Select Put from the menu, 00, and 10, and you should get images all down the left side.

17. If you need to print more, select FF, do Put 00, 11, and repeat until you have done enough.

18. Rewind the paper to TOF. Now repeat steps 14-17, except Put the image at 71.

19. Again rewind to TOF. Now set the paper to print just under the top images. Be SURE to do power Off/On.

20. Do the BREAK RUN thing again. This time it is not necessary to select an image, you may go directly to Options. Select Text (only). Select New or Old as you choose, just like printing a normal label or lettehead. When ready, select Print, then Head from the menu, accept the default 27 as to the column to Put it query. Enter only 1 to the How Many query until you see if you like it. You will be returned to the menu, to do more if you want.

21 This time select Head, 27 (if you liked that position, else adjust), then enter the number to print. I recommend you print less than a million. The ribbon tends to fade. When done, tear off the paper, separate the sheets, and write letters, letters, letters. I always tear off the tractor holes and use my single sheet feeder.

UGOC ROM OCTOBER 1991

## DISK CONTROLLERS - from TI to MYARC

The following information was on a disk from Ed Machonis. I believe he got it off of a Bulletin Board, hence I am going to reprint it, even if it is copy-writed. My apologies to the author if reproduction was prohibited. I feel the UGOC should know this kind of stuff.

The views expressed in this article reflect the author's personal experience with TI, Corcomp, and Myarc disk controllers. Technical data has been verified wherever possible, but is not publicly documented in some instances. Please bring any errors to the attention of the author.

The disk capacity of the TI99 has increased in just a few years from less than 80K (a single one-sided 35 track drive) to almost 2.9 megabytes (four double-sided, double-density, 80 track drives). The early standalone was replaced by the PEBox system which would support three double-sided 40 track drives (540K). Corcomp introduced their four drive double-density system (1440K), followed by Myarc's similar system with two double-density formats (1280K and 1440K). Then in 1986, Myarc offered its 80 track upgrade which doubled capacity again. Even as capacity was increasing rapidly, the TI and Corcomp controllers differed only modestly in I/O speed. When MYARC introduced its fast DSDD controller, few reviewers did justice to its speed advantage. Early comparisons were done at the standard TI or Corcomp interlace, but the big speed gains required taking advantage of the much tighter sector interlace possible with the high-speed MYARC card. To understand how this works we need to take a look at the way a disk drive performs.

### Disk Drive Fundamentals

A floppy disk drive writes information in concentric rings called "tracks" on a thin plastic disk coated with a film of magnetic particles. Each track in turn is divided into blocks of information called sectors. A blank disk has one (or more) index holes used to synchronize the process of writing to and reading from the disk. The type with many holes are called "hard sectored" since each sector has its position fixed by an index hole. The type of disks used by most computers have only one hole and are called "soft sectored". In this system the computer must write magnetic signposts on the disk to mark out each sector in a process called "formatting" or "initializing" a disk. These signposts take up a subtantial fraction of the space on a track since they include not only sector numbers but buffers (filler bytes) that allow the computer to get into synchronization to read or write sectors of data and to prevent the sector identifier from being overwritten by a drive operating at a slightly different speed from the drive that formatted the disk.

The typical 5.25 inch disk drive has a "stepper motor" capable of moving the drive's read/write head(s) in or out along a radius of the disk in steps of 1/48 of an inch (thus the terminology "48 tpi" = 48 tracks per inch). Since the inner tracks have a smaller circumference, they crowd the bits of information together. Magnetic coatings on a floppy disk are rated by their capacity in bits ~~per inch~~ at standard magnetic flux for the write head. This figure is usually over 5000 bpi for modern floppies, but was somewhat lower a few years ago. The circumference of the inner track of a 40 or 80 track disk is about 10 inches -- which allows about 6250 bytes to be written on the track without exceeding 5000 bpi. For comparison, the Corcomp double density format requires over 6400 bytes per track. Media limitations were the reason that some early 5.25 disk drives only used the outer 35 tracks. The 16 sector (by 256 bytes/sector) format recommended by most drive makers requires only 6250 bytes per track and includes several hundred additional "buffer" bytes to compensate for differences in drive timing.

### Timing is EVERYTHING

With soft-sectored disks, the integrity of the read/write processes require critical timing. The disk rotates at 300 rpm within a small margin. This means there are about 250 thousand magnetic pulses (bits) passing beneath the head each second. In single density format, the majority of these pulses are timing or filler bits -- in double density, many of the timing bits are suppressed in order to double the rate of data bits. In a typical sector read the drive must bring the disk up to speed, recognize the index hole, step out to track zero (to get its bearings), determine single or double density, verify its position, step in to the target track, verify the track number (written in the format operation), detect the sector identifier as it flies past, then immediately read the 256 data bytes into memory. Five of these operations require accurate reading of the magnetic pulses whizzing by at over 250K bits per second.

If you do some quick arithmetic (256 bytes/sector)*8 = 2048 bits/sector into 250k bits/second)..... hmmm..... Why can't the drive read a 125 sector file in one second? Well first many of those overhead to keep things synchronized and allow for timing variation between drives. Second, some time is used moving the head from one track to the next when more than one track must be read. Third, 250K is the instantaneous read rate and the computer must take time to do other things like move the last sector out of its buffer to make room for the next one. In the standard TI protocol for reading a disk, the data is moved into VDP ram (so the drive could be used without the memory expansion) before it goes to the expansion memory. All this thrashing eats great chunks of the time available for reading data. By the time one sector is safely tucked away in the 32K card, several sectors have already passed by the drive's read head. If the sectors were written consecutively on the disk, we would have to wait a full revolution (0.2 seconds) before the next sector would pass under the head. To avoid this inefficiency, the consecutively numbered sectors are spaced out around the disk so that they are separated by just enough time to take care of other business. The actual pattern in which the sectors are scattered is called the "interlace". The idea of the interlace is to spread the sectors out to match the timing needs of the hardware -- both the time needed to stash each sector and the time needed to step from one track to the next and get the the head settled down for some serious (250K bps) reading.

[I had intended this interesting article would be continued over three months, but since I got no articles from our membership, I am running it all this month. Maybe you will contribute something next month. Ed]

## DISK CONTROLLERS - from TI to MYARC
Copywrited By Jerry Coffey 1987

### Interlace and Head Step Times

Life was simple with the TI disk controller. Both the interlace and the head step time were locked into the controller's PROM (that's the programmable chip that contains the control programs for the card). The head step time is the built-in delay between step signals to allow the stepper motor to move the head one "click" in or out. The TI settings are very conservative (read "slow") to allow for slow drives. The step time is 20ms -- if you step from track zero to track 39, it takes 20x39=780ms, almost four revolutions of the drive. The TI interlace lays the sectors down on a track in the order 075318642. This allows all sectors to be read in four revolutions of the disk though the slow head step lets another revolution go by between tracks. Thus the maximum read rate is about 9 sectors per five revolutions (= one second) or 2304 bytes per second.

When Corcomp designed its double density disk controller, allowances were made for the increased speed of later drives by permitting the step rate to be set with DIP switches for each drive. The step rates available are 30, 20, 12, and 6ms (the faster values quoted in the CC manual are referenced to the wrong clock speed). They also provided a choice of interlace options, though only a couple of them are practical. The default interlaces are labeled "7" for single density and "10" for double density. The single density interlace is the same as TI's, but with a faster step setting the head be can moved without losing a revolution and thus reads 20% faster than the TI controller. The double density interlace allows 18 sectors to be read in five revolutions, but it doesn't leave enough margin to stash the last sector and step the head in time to catch the zero sector of the next track (that's why the sector number "hangs" for 0.2 seconds each 18 sectors while verifying a formatted disk -- you are seeing the extra revolution needed to acquire the first sector of the next track). Thus the maximum read rate is 18/1.2 or 15 sectors per second, about 67% faster than the TI controller. Users of the CC controller have probably noticed that it loads its own MANAGER program faster than this. In this case a ~~special~~ loader bypasses VDP and loads directly to CPU RAM -- this faster handling of the data allows the stepper motor to be activated sooner and saves one revolution per track (so the 98 sector file can be read in about 5.5 seconds). This provided a foretaste of the speed that MYARC would achieve with its double density controller.

The MYARC controller bypasses VDP RAM to load directly to CPU RAM. This technique coupled with a buffer RAM chip on the controller card provided a quantum jump in disk I/O speed. The MYARC card reads the TI single density interlace at 11.25 sectors/second (the same as Corcomp) and reads the CC 18 sector/track interlace at 18 sectors/second (the same speed Corcomp reads its MANAGER program), but this is only the beginning. Since the hardware empties its sector buffer faster, consecutive ~~sectors can be~~ placed closer together allowing a track to be read in fewer revolutions, i.e., it supports a faster interlace. With fast drives, the 9 sector/track single density format can be read at interlace "2". (NOTE: In the MYARC terminology, the interlace number represents the number of disk revolutions required to read a track.) this works out to 22.5 sectors/second compared to 9 for the TI and 11.25 for the CC controller. The MYARC 16 sector format can be read at interlace "3", 26.67 sectors/second -- 3 times as fast as the TI controller and almost twice as fast as Corcomp double density. The Corcomp 18 sector format can be read at interlace "3" or "4", but the data rate is the same in either case, 22.5 sectors/second. Interlace "4" is smooth but requires a very quick head step, interlace "3" reads the track in 3 revolutions but forces an extra revolution for the step from track to track ~~because~~ sectors 17 and 0 are adjacent on the disk. Though both interlaces have the same data rate, interlace "3" is safer if you are uncertain about the speed of your stepper motor.

In order to read and write both double density formats, the MYARC system must insert an additional step in some I/O operations -- sector zero must be read to determine whether a double density disk has 16 or 18 sectors per track. This datum is needed to convert the the logical sector numbers used by the TI operating system into track and sector-within-track addresses for the floppy disk controller chip. The TI and Corcomp controllers do not need this step because they do not use the full potential of the TI disk I/O protocol. Once this step, accessing sector zero, is added to the various disk operations, it opens the system up for using more than two formats -- including 80 track formats.

### Beyond Double Density

A two format system can be managed using only the floppy disk controller's inherent ability to sense single and double density recording patterns. To get beyond this limitation, the additional data stored in sector zero must be read, stored, and used to modify the special binary commands sent to the FDC (floppy disk controller) chip. Fortunately the TI99/4A system design already provides for such innovations through the Device Service Routine concept and standard "GPL" calls. The system doesn't care what hardware is attached as long as it plays by the rules -- an interface program stored in a memory chip (PROM) on the peripheral device does the trick. This program handles calls for I/O operations from other programs such as TI Writer or the Basic Interpreters. Another set of rules controls the way disk and file information are saved on a disk. Disk parameters are stored in sector 0, while sector 1 must have a two byte "pointer" (a hexadecimal sector address) for each block (one sector) containing the bookkeeping data for a file. It is these blocks that are scanned in order

~~HOW WOULD YOU~~
Like To Horse
Around A
Little Bit?

## DISK CONTROLLERS - from TI to MYARC

Copywrited By Jerry Coffey 1987

Since the Myarc controller must read sector zero to determine the number of sectors per track, the other parameters in that sector are available to control other variables such as number of tracks. But there were other limitations to overcome. The number of files on a disk is limited by the space available for pointers. 256 bytes at 2 bytes per pointer would give 128 files -- except the pointer list must end with a null word ( >0000 ) so directory routines know where to stop -- so we get 127 files per disk. The pointer itself can address sector numbers as high as 65535, so this is no problem. The real limitation is the bit map in sector 0. It begins at byte 56 leaving only 200 bytes or 1600 bits available to map the disk. Since a bit must be turned on for each sector used, the 1440 sector DSDD 40 track disk is already near the limit. The answer devised for the 80 track DSDD system is to map two consecutive sectors with each bit. It wastes some space but no more than systems that use a standard 512 byte sector.

### Making the Quad System Work

So now lets say we have new code in the disk controller EPROM (an "erasable" version of the PROM chip used by TI) that does all the proper tricks with the bit map and has the FDC commands to control the new 80 track drives we have added to the system. We still have to tell the controller which drives are 80 track and find a disk manager program that can use the new commands. The selection problem can be taken care of using the DIP switches on the card (but in the process you lose their original function -- setting step speed). Since the Eprom responds to standard GPL calls, most functions can be handled by the TI Disk Manager 2 cartridge. The exception is the disk formatting process -- the formatting works OK, but the initial data written into sector zero is for the standard bit map. (This can be fixed by changing byte 56 from >03 to >01 with a sector editor.) Read/write operations from XB or TI Writer work fine since they use the GPL protocols. Myarc has an excellent disk manager program that works beautifully with 40 track drives, but it has suffered from a number of subtle bugs in 80 track mode. This program, like many others designed for high speed I/O, uses assembly language code to handle the FDC -- bypassing some of the routines in the EPROM. Differences in bit map handling, even slight differences in execution times can affect the performance of 80 track drives. The code in the 80 track EPROM has had a lot of attention to proper timing -- the price you pay for higher performance.

### Fine Tuning the Myarc Disk System

Before you start using the Myarc system routinely, there are some experiments that can get maximum performance from your drives. Use the Myarc disk manager to try different interlace settings -- first with your 40 track drives, then with the 80 track drives. Watch for hesitations as each formatted disk is verified, then use the Test option to read the sectors you have layed down. Look and listen for "retries" -- when the sector number pauses with a head seek noise. Use the best disks you have and note the combinations that test smoothly. With fast drives in good condition, you should be able to run 9 sector (single density) format at interlace 2 and 16 or 18 sector double density format at interlace 3. Don't worry if 18/3 pauses at the end of each track -- this is just the extra revolution forced by having sectors 17 and 0 adjacent on the disk.

When you try this with 80 track drives, don't be surprised if the results are different. The time required for the head to settle into a wide standard track may not be adequate to get it reading properly from the narrow tracks on the quad drive. Such subtleties as erase delays and disk quality are also more critical on the skinny, low power tracks. My Mitsubishi 4853s (96 tpi) will support both 16/3 and 18/3 but are unreliable at 18/4, while my TEAC 55Bs support all three at 48 tpi. Don't take chances with any setup that is marginal. The error rate may be low, but it always seems to happen to a file that isn't backed up.

### Hot Rodding

If you want to try for a little more speed, there are two more tricks you can use. The faster WD1772 FDC chip is pin compatible with the standard WD1770 supplied by Myarc. It will try to step the head at 2ms rather than the 6ms setting of the standard chip. (The 80 track EPROM automatically uses the fastest step speed available.) Many of the latest drives can step at 2ms or 3ms even though they are conservatively rated at 4ms or 5ms. The change is noticeable but may not be worth the high price of the WD1772 (it is not a commonly used chip and is rarely discounted). The second fix is cheap and very useful for producing large quantities of copies. The FDC chip's automatic "write verify" function can be defeated by shorting one pin on the controller card to ground. This is best done with a switch so the verify can be enabled for normal operations. The effect of this modification is equivalent of the "turbo" option on the Corcomp controller and should be used only after testing.

There was more to this article, but it was mainly high tech hardware mods. If are truly interesested, see me and I will supply you with the rest of the article.

The above article was downloaded from Bob & Bill's BBS, (BBBBS) Clinton, MD (301) 292-1482, Jerry Coffey 74716,3525 can also be contacted thru this BBS.

The DA thinks Humpty Dumpty may have been pushed

SAVE THE WHALES
~~SAVE THE DOLPHINS~~
SAVE THE EARTH
SAVE EVERYBODY

I wish I could ~~Remember~~ Who I Am

## XBASIC MISCELANY #6

By Earl Raguse

Last month, I revealed a HEXadecimal to DECimal conversion subprogram as a part of PEEK/POKER. I also had written the inverse DEC to HEX at the same time but did not use it in that program. I did not explain how sub DEC worked. This time I will give and explain both. It is frequently much more convenient to input HEX numbers when one is working with computer addresses, but XBASIC doesn't approve, so the program must convert them before execution. I have not had the occasion to use the DEC to HEX conversion in any of my programs, but there must be some need, else why would I have written it?

SUB DEC expects a HEX number string H$, and a variable D in which to pass the decimal value back to the program. The first thing we do is to strip the > from the HEX number string H$, and set D=0. Then using a loop, to loop for a number of times equal to the revised length of H$, we examine each character of the HEX string H$ for its value using X$=SEG$(H$,I,1) and X=POS("0123456789ABCDEF",X$,1) to convert each character to its numerical value in decimal.

We then convert the overall number to decimal by multiplying each previous value of D by 16, and summing the new X value to the old after doing a -1. The latter is necessary, because there are sixteen characters in the POS statement, and we want a range of 0-15, not 1-16. I can't think of a simple way to explain why we must multiply D by 16 each time, but trust me, it is necessary. If this really bothers you, grab me at a meeting, and I will be thrilled to try to explain it to you. When we are through, we compare the number with 32767, and if it is larger, we subtract 65536, to stay within the addressing rules of the 99/4A.

SUB HEX expects a decimal number D and a string H$ in which to pass back the HEX value. The first thing we do is clear H$, and check to see if the number D is negative. If it is we add 65536. Note that this particular subprogram can deal only with numbers equivalent to 16 bits of binary max, or 65536 decimal, or FFFF hex. Here we do almost the inverse of above, we divide our decimal value by 16, (ie N=Q/16) after first saving our decimal value D in Q. We are interested in factoring out only the integer multiples of 16, we save the remainder in R for repeated division. XBASIC does not have a MODulo arithmetic command, so we must make our own with R=(N-INT(N)). R is A value from 0 to 15. Once we have an integer decimal value for R, we convert it to a HEX character using

R$=SEG$("0123456789ABCDEF",R+1,1).

The +1 is there for the same reason we had a -1 above. We then take the HEX character , and insert it into the HEX string H$. H$=R$&H$, Again, I am somewhat at loss to easily explain in writing, why I must append H$ to R$ instead of vice versa. But like the above, if you collar me at a meeting I will do my best to convince you that its necessary. One reason (maybe the reason) is that we are factoring out the More Significant multiples of 16.

```
8000 SUB DEC(H$,D):: H$=SEG$
(H$,2,LEN(H$)-1):: D=0
8010 FOR I=1 TO LEN(H$):: X$
=SEG$(H$,I,1):: X=POS("01234
56789ABCDEF",X$,1):: D=D
+X-1 :: NEXT I
8020 IF D>32767 THEN D=D-655
36
8030 SUBEND
```

```
8050 SUB HEX(D,H$):: Q=D ::
H$="" :: IF Q<1 THEN Q=Q+655
36
8060 N=Q/16 :: R=(Q-INT(N))::
R$=SEG$("0123456789ABCDEF",R
+1,1):: H$=R$&H$ :: IF Q>P
THEN 8060 :: H$=H$&H$
8070 SUBEND
```

## XYZ HARDWARE COMPANY ANNOUNCES
Extended Mouse Support

Mouse balls are now available as an FRU (Field Replacement Unit). If your mouse fails to operate properly or should perform erratically, it may be time for ball replacement. Because of the delicate nature of this procedure, it should be attempted only by trained personnel.

Before ordering, you must determine the type and size of your mouse's balls. This is done by careful examination of the underside of each mouse. Domestic balls will be larger and harder than foreign balls.

Ball removal, as you might well imagine, is a very delicate matter and may differ depending on the type of mouse. Foreign balls can be replaced by using the "pop-off" method and domestic balls using the "twist-off" method. As you can well understand, use of the wrong method can be very detrimental to your mouse. Although mouse balls are not usually static sensitive, excess handling may result in a sudden discharge.

(From the Pomona Valley News who got it from the Northeast Tarrant County TI 99/4A User Group)

SAVE THE WHALES
SAVE THE DOLPHINS
SAVE THE EARTH
SAVE EVERYBODY

## GETIT HAS AN ERROR
By Earl Raguse

A few months ago, I listed some of my subprograms, and among them was GETIT for retrieving a data file which had been saved by SAVIT. Well, I had made some revisions, which I thought would make it more useful than my original version, and I thought surely I had tested it. Evidently I didn't do it well enough, because when I tried to use it, as printed, last week, it didn't work. Stupid error in the subscript counter. I have seen GETIT reprinted in several newsletters, so before the booboo gets too widespread, I shall give out a new corrected version, which I recently used, and know that it works.

Because I have such a poor memory for file names, I have included a CALL CAT option in it, not the same as a CAT CALL, and an error trapping call. I am not including CAT herein because I have made no changes in that. I am usually "improving" my subprograms, and few are still the same as they were published. That doesn't mean that they didn't work, it just means, I found things to change. I change everything. I like to think it is for the better, but there may be some difference of opinion on that. Anyway, GETIT follows. I hope my error did not cause to much inconvenience. Feel free to write or call me if you ever have trouble with any of my programs, they all carry a lifetime warantee.

```
1 ! SAVE DSK1.GETIT,MERGE
4200 SUB GETIT(FIL$,VARBL$()
,N)
4210 CALL CLPUT("Enter Drive
#",12):: CALL GKEY(Q,24):: I
F (Q-48)<1 THEN 4210 ELSE
D$=CHR$(Q):: ON ERROR 4215
4215 CALL PUT("Enter File Na
me",14):: CALL PUT("if you c
an't remember ",20):: CA
LL PUT("Say D for Disk Direc
tory",22)
4220 DISPLAY AT(16,10):FIL$
:: ACCEPT AT(16,10)SIZE(-10)
:FIL$ :: IF FIL$="D" THEN
CALL CAT :: GOTO 4215
4225 OPEN #3:"DSK"&D$&"."&FI
L$,INPUT :: CALL CLPUT("Read
ing Records",10):: R=1
4230 IF EOF(3)THEN 4250 ELSE
 LINPUT #3:VARBL$(R)
4240 R=R+1 :: IF R<N+1 THEN
GOTO 4230
4250 CLOSE #3 :: ON ERROR ST
OP :: SUBEND
```

## RESTORING THE LC CHARACTER SET
By EarL Raguse

I have previously published the little assembly subroutine called CHRSAV that was written for me by Arian Robinson to save the whole character set when one first boots up, and which may be call linked to restore the characters after someone has fouled them up by redefining them for graphics. The only way TI gave us to restore the LC set was to return to the Title Screen. What if you don't want to do that? Suppose you want to RUN another program instead? Well, you could use Adrian's CHRSAV, BUT that requires doing a CALL INIT, and if any program after that does a CALL INIT, you have lost it. Also there are other assembly routines that people use like BXB, that use the same area of memory and they may step on CHRSAV's toes.

I like to use CHRSAV, because it is very fast, but ~~sometimes one can't~~, so I wrote LCSAV/SET as subprograms. If you MERGE them into your finished program, and insert CALL LCSAV(X$()) in the beginning, and CALL LCSET(X$()) just before you need to restore, you are in business. One thing though, you must do a DIM X$(32) sometime before calling LCSAV. X$() may be any string variable.

```
1 ! SAVE DSK1.LCSAV/SET,MERGE
4310 SUB LCSAV(CH$()):: FOR I
=1 TO 32 :: CALL CHARPAT(I+96
,CH$(I)):: NEXT I ::SUBEND

4320 SUB LCSET(CH$()):: FOR I
=1 TO 32 :: CALL CHAR(I+96,CH
$(I)):: NEXT I :: SUBEND
```

## SAVING XB SUBROUTINES
By Earl Raguse

I recently got a letter from the Editor of the TIsHUG newsletter, Rolf Scheiber who is publishing the Beginning Forth articles I wrote for The LA 99ers TopIcs He sent me several disks, in exchange for some of my XBASIC program disks that he wanted. On those disks I found a program that should be of interest to all. In the past I published a program that would save a set of lines out of an XB program, but it was saved to a DV80 file, which had to be converted to an XB program by a separate program. There are several of these, TEXTLOADER being in my opinion the best, but not the only. There is XLATE also which converts DV80 to DV163, which then can be MERGEd into an XB program format.

The following program by David Sheehan, converts your subroutine directly to a DV163 file for MERGEing into your XB programs. Very convenient. It is not a speed demon, but it does the job done painlessly. You just type in the following program, save it in MERGE format, (which is the way they all work), then MERGE it into the program you wish to extract lines from. Assuming that your program has no line numbers less than 10, (if it does, RESequence it).

Now, run the program, and follow instructions. You must know the first and last line numbers of the lines to save. The designated lines will be saved in DV163 format. One final thing, DO NOT save your original program with the above program still MERGEd in it. That's it.

```
1 ! SAVE DSK1.SUBSAVE,MERGE
2 CALL CLEAR :: DISPLAY AT(1
0,1):"START LINE ?":"FINISH
LINE ?":::"FILENAME?" : ACCE
PT AT(10,15):ST :: ACCEPT AT
(11,15):FI
3 ACCEPT AT(14,1):F$ :: DISP
LAY AT(22,12):"SAVING" :: OP
EN #1:F$,OUTPUT,DISPLAY ,VAR
IABLE 163 :: CALL PEEK(-3195
2,A,B):: L=A*256+B-65536
4 CALL PEEK(-31950,A,B):: AD
=A*256+B-65539 5 IF AD<L THE
N
9 ELSE CALL PEEK(AD,L1,L2,A,
B):: LI=L1*256+L2 :: LA=A*25
6+B-65536 6 IF LI>FI THEN 9
ELSE IF LI<ST THEN AD=AD-4 :
: GOTO 5 7 CALL PEEK(LA-1,A)
:: L$="" :: FOR T=LA TO LA+A
-1 :: CALL PEEK(T,B):: L$=L$
&CHR$ (B):: NEXT T :: PRINT
#1:CHR$(L1)&CHR$(L2)&L$ 8 AD
=AD-4 :: GOTO 5
9 PRINT #1:CHR$(255)&CHR$(25
5):: CLOSE #1 :: DISPLAY AT(
22,11):"FINISHED":" COURTESY
OF DAVE SHEEHAN" :: STOP
```