NORTHWEST COMPUTER NEWS

Dedicated to Norm Deletzke and the Orphaned Computers he loved

404 Inverrary Deerfield, Il. 60015 (708)808-7000

HELP FOR 8050 SPEED PROBLEMS by Chris Burgbacher

Some time back I got involved with the speed problems of the 8050 disk drives when one of mine started acting up. The first indication of the problem was the drive declaring disks bad that it was supposed to be formatting. I felt sure something was odd when the other drive formatted them just fine.

I talked to Norm, I bought a copy of Physical Exam for 8050. After adjusting the speed of the drive, it ran fine again. This was not the end of my problems. I discovered that the speed varied with both the season of the year and the amount of access time of the drives. With an increase in heat, the drives slow down. With a decrease in heat, the drives speed up. This launched me into a full scale investigation of the problem.

The drive motor that turns the disk has a small generator built into its end. It is referred to as the 'tachometer' and its voltage output is referred to as the 'tach feedback'. When the motor spins the disk, the tach feedback goes back to the circuit board on drive 0. A diode on the board converts the tach feedback AC voltage to DC.

A Zener reference diode is used to establish a fixed reference voltage. A comparator circuit compares the reference voltage with the tach feedback. The output of the comparator controls the drive transistor that supplies operating power to the drive motor. If the tach feedback should fall a little, the comparator kicks up the power to the motor to maintain its speed. A control is provided to adjust the amount of reference voltage seen by the comparator to allow us to adjust the motor speed.

Now, here is where the system goes astray. The diode used to convert the tach feedback is very temperature sensitive. All diodes are sensitive, to some extent, but this one is bad. Adding insult to injury, Commodore

(Continued on page 11)

Table of Contents

Superbase Programming

by Bruce Faierson

Though promised several years ago, the need for a series of articles on Superbase and SuperOffice programming still exists. Several people have made negative statements regarding Superbase such as it is not very user friendly or easy to utilize. It is time to lay these fears to rest, for both the intimidated computer user and the more skilled Basic programmer. This series of articles is designed with the inexperienced though well read programmer. I will try to add concepts that even an experienced programmer might be unaware of.

This is the fifth article that I have written on Superbase. The first two examined general database information and the Superbase menu functions. The third article discussed methods of recovering data from Superbase and how Superbase stores data. The fourth article appeared in the first issue of this newsletter regarding multi-indexing schemes. If you do not have access to the previous articles, they could be reprinted for a modest charge. It is important to read the first two articles or read the Superbase manual if you do not have much database or programming experience. Warren Swan's Basic tutorial in the CBUG Escape or on library disks is also a very useful tool in understanding Superbase Basic programming. The Superbase language is both an extension and reduction of standard Basic.

It is recommended that you use the current version of Superbase 2 which is 2.06 or the SuperOffice version 2.08. These versions are several revisions and quite a few bugs removed from the Superbase 1.0 that was distributed by Protecto. Please understand that bugs, as unfortunately as it may seem, are a normal byproduct of software development for any computer system. I beta tested a major database that had thousands of bugs that had to be documented and fixed. The number of bugs present in Superbase 2 is minimal.

Superbase is still a viable product for small database applications using from a few to several thousand records. It does have the capability of accessing more data but due to our lack of massive disk storage the 8050 floppy drive sets the limit. For those of you lucky enough to have a hard drive or an 8250 drive, this amount is increased substantially. Among the many features that Precision Software pioneered is the record design screen. Incidentally,this has just been added to most database packages in the last two years. The Superbase design screen is still easier to use.

Superbase or SuperOffice can build any type of application ranging from the simplest name and address list to the

(continued on page 8)

TTL VIDEO MONITORS AND THE B-128

Remember that article I wrote about squeaky wheels? Well this is a direct result of a request for information. If you are looking for something regarding the B, write!

The standard B-128 can provide either composite video or ttl video. There are some remarkable composite monitor bargains to be had from closeout companies, but if you have a ttl monitor on hand there is no reason why you shouldn't connect it to your b. The original Protecto package had a video cable with 4 Rca plugs on the 'monitor end'.

The white plug provides composite video for connection to the dreaded Zenith monitor. Have you ever wondered what the other three plugs were for?

First dust off your Commodore B-128 programmers reference guide and turn to the back of the book. Thumb forward past the software advertisements to the schematics. We will be referring to sheet 13 of 14. The schematic is virtually self-explanatory, although there are three really neat jumpers of interest to anyone contemplating ttl monitor use. I'll return to them in a few paragraphs.

On my Protecto-supplied monitor cable, here are the connections (you should double check your cable with an ohmmeter)

pin1 - ttl video - red plug
pin2 - ground - all plug shells
pin3 - vert sync - black plug
pin4 - composite video - white plug
pin5 - horiz sync - yellow plug

Composite monitor users need connect only one plug, the white one, to their monitors. TTL monitors however, need three connections instead of one.

(If you intend to wire up your own din plug to Rca cable, please make sure you look carefully to ensure that the cable is wired correctly! Din plugs are counted by the random-walk method.)

Having plugged in all three TTL connectors and the Din plug, turn on your B and the monitor. If the screen looks ok after adjusting the monitor controls, quit while you are ahead. However if all is not perfect in paradise, here we go with those jumpers I promised to get back to. Each TTL signal is individually invertible by moving the jumper connecting the relevant 7486 pin from +5 to ground. (I hope I made myself clear. DO NOT connect +5 to ground, unless you want to troubleshoot your power supply! The 7486 pin which was connected to +5 is disconnected from +5 and instead connected to ground)

How do you know which jumpers need to be changed? Look at the following chart:

Bright characters on dark screen - correct

Dark characters on bright screen - move jumper going to u13 pin9

Left hand of screen on the left - correct

Left hand of screen in center with less than 80 characters visible across - move jumper going to u13 pin2

Top of screen at top - correct

Top of screen in middle with less than 25 lines of text visible down- move jumper going to u13 pin13

The jumpers are located between R46, a 1k resistor, and Y1 the 1.8432 MHz crystal. In relation to the drams on the front of the board, they are located behind and to the rear, but in front of the 60- pin expansion port. The jumpers from left to right are connections to pin 13, pin 9 and pin 2. Note that the third jumper is located to the left under Y1, the crystal.

And thats all there is to it. The 7486 is the hardware equivalent to the eor instruction in machine language, so either normal or inverted output is available, in exactly the same way that eor #00 would leave what was in the accumulator alone, but eor #ff flips each bit. Enjoy TTL video!

Mr. Anthony J. Goceliak
RFD #2 - Box 433
Lancaster N. H. 0358407
Sep 91

EDITORIAL

Firstly, let me apologize if there was any confusion regarding the advertising we sent out recently. This was not an issue of the newsletter! The subscription notice inside was directed at former members and customers that had not subscribed to the newsletter. If you paid previously, your subscription is still in good standing for the next two issues. If you paid again, my apologies, your subscription has been extended until the end of 1992 or four issues after your current subscription expires. Starting with the next issue we will have an expiration date on our labels.

The issues will contain between eight and twelve pages of articles and programs, which will depend upon the amount of material we receive. Please send in articles that you think can help anyone. The membership is growing little by little and I can say that we have a lot of the major contributors left in the group.

May I request that articles be submitted in Superscript II with absolutely no formatting commands. It would be nice if there was one space between sentences and only one carriage return between paragraphs. This will reduce the amount of labor in getting these files to the typesetter.

I have decided to lower the price on all library disks except commercial software and royalty bearing disks to $5. I have also decided to set a minimum of $10 for each disk order unless other merchandise is purchased with them. This is meant to keep overhead to a minimum and to give the subscribers a better deal. Remember, library funds help support the publication .

Well, that's it for this issue. I hope you enjoy it as there are so many good articles to read. The next issue should be out in early to middle November. See you then.

Northwest Computer News
Editor Bruce Faierson
Publisher Bruce Faierson
Contributors T. Goceliak, D. Jarvis, A. Bouvier, G. Schumaker, R. Sherman
Northwest Music Center, Inc.
President Bruce Faierson
N.W. Music Center, Inc. assumes no liability for the accuracy of the materials presented.

Business Office
404 Inverrary Lane Deerfield Il. 60015
Advertising Office
404 Inverrary Lane Deerfield, Il. 60015
Phone:
(708) 808-7000
Vice-President Kathy Faierson

RE-DEFINE YOUR FUNCTION KEYS WHILE IN SS2

By Alan Bouvier

FNKEY is my solution to what I felt was one of Superscript II's biggest deficiencies. To me, definable keys ought to be able to be defined at will. Especially within a program involving a lot of typing. Many times I decided that I wanted a different set of key definitions. Rarely was it worth leaving Superscript, changing the loader program, and re-loading Superscript.

So I did something about it. I found a way to get into the Superscript routines and make the function keys programmable again. It is even easier than setting the definitions from BASIC. No more chr$(xxx) codes. I also did something about the carriage return erasing the end of a line. A by product of all this is the program that links my program to superscript. It makes an unprotected copy of Superscript onto a blank disk and copies my routines on it so you can SHIFT/RUN it. Just watching this program in action could prove interesting. It should have no problem working with an already unprotected copy. The last part of all this is a patch for the cursor hesitation we've all experienced. Since this fix is to Superscript instead of the computer's operating system, I believe that it will work with the B-128 serial bus.

I left a lot of room for expansion within Superscript. Anyone with the slightest idea of something they would like should let me know. Even if it seems highly specialized. It could be the spark to yet an other useful program.

The programmability of the function keys is similar to the 'soft keys' of SS3. As I understand it, the 'macros' of SS3 are just a chain of normal editing functions. In this respect, a properly programmed function key could perform the same 'macros' available in SS3. Ever since I got the first version of my enhancement working, I have found more and more uses for it. It has not become one of those cute functions rarely used. There is a good demonstration of the usefulness of this function as well as an explanation of how to use it on the disk. Just get a blank disk, load and run "unprotect ss2" and follow the directions on the screen.

My fix for the carriage return problem was simple. I swapped the actions of the **RETURN** and the **SHIFT/RETURN** keys. This proved quite more disastrous to my editing than I expected. It took quite a while to get used to this. Now I like it. This feature is easily turned on or off at will. Press **escape** and then hold the **control key down** while **pressing x.** I suggest picking one mode and staying with it. Constantly changing modes would probably cause more confusion than anything.

For anyone interested, all of the source code for my routines will eventually be made available. They are well documented so they shouldn't be to difficult to follow. They are quite lengthy, though. I will not include the source code for Superscript which I modified. It is copyrighted material.

Any questions or comments, please write. **You can only benefit from it.**

(Editor's Note: Alan's deprotected Superscript does work with the fast bus implementation. We have tested it.)

Alan F. Bouvier
815 Lake Vermillion Ct.
Slidell, LA 70461-8587

LIBRARY DISK REVIEWED

Alan Bouvier 1.0: Here is a new contributor with some amazing discoveries, useful information and innovations that most B-128 users should find fascinating. Alan, has found a new method of deprotecting Superscript II. He has added a software routine to Superscript itself which eliminates the doubling of characters. He also developed a routine to define function keys within Superscript, without having to use the chr$() function for certain types of input. His method is similar to a programmable macro where you press the escape key and the function key, enter your keystrokes for carrying out the operation and then storing the function key definition. This disk is for legitimate owners of Supersript II and for making archival copies for the original purchasers use only. This is a disk we have all been waiting for. This disk includes the following.

Price $5.00* Freeware. Shipping $2 any quantity.

1.) **Unprotect SS2**
2.) **Function keys description**
3.) **IRQ linking description**
4.) **Clk/IRQ/Restart puts clock on screen**
5.) **Set top of Form program**
6.) **4023 Error codes**
7.) **Sneak Previews, source code and more.**

** ALL DISK ORDERS MUST BE A MINIMUM OF $10..*

SUPERPET 9000 *by Bruce Faierson*

I have often been asked, "What is a SuperPet?". Giving a simple answer is not easy, as it was Commodore's flagship model with more standard equipment than other computers to date. Waterloo University developed this computer, in conjunction with CBM for their computer program. Their strategy was to have several languages available on mainframe, mini and microcomputers. The languages were designed to be portable so programs could theoretically run on any of them without change.

The SuperPet was built upon the 6502 based 8032 model. The SuperPet can utilize all the Commodore drives such as the 8050 and all the IEEE printers like the 4023p. It runs all 8032 diskbased software without modification. Software that requires a rom for operation can be utilized provided the user download the rom and load it into the bank-switchable ram at $9000.

A second processor, the 6809, was added to compliment the 8032 section and to be utilized with the Waterloo languages. The 6502 and the 6809 operate independantly of each other. Both processors have a 64k address space plus access to 64k of bank switchable memory. This memory is switched in 4k segments into the address space at $9000. The Waterloo Software loads into the bank switched ram leaving the low memory for actual user programming.

The SuperPet was the first Commodore computer to have a built in DB25 type RS-232 interface. This port is mounted internally on the 6809 board. There is an opening to feed a cable to the port on the left side of the machine. There is a simple terminal emulation program built in and a communication program called PetCom is available.

Finally, the languages included are interpreted forms of ANS Minimal Basic, Fortran, Cobol, IBM/ACM APL and Pascal. The system also includes an editor, assembler and linking program. A monitor program is included which supports the

(Superpet 9000 continued on page 11)

ONE-LINE 60 %-FLOPPY SPEEDUP *by Gerd Schumacher*

There is a way of speeding up floppy-operations a bit, that most of all CBM-users know, having been described in the Escape before. It is simply to program another time-constant between drive-motor on and data-transfer. Of course, it saves nearly one second per disk-operation, unless the drive was not yet spinning before, but, unfortunately, it sometimes might produce errors during write-operations.

As we are rather unhappy with the CBM-diskhandling - like the directory and the slowness of all disk-operations - as well as with most of all disk-utilities, my friend Dirk Seifert and I thought out a lot of improvements. One result of our investigations was determining the best block-interleave on the disks .

You know, there are two processors in the 8050, 8250, and SFD. One of them is the bus-controller, communicating with the computer, doing all calculations on files, say, doing all of the more "intelligent" things. It tells it's "slave", the disk- controller, what it has to do. The disk-controller only provides data-transfer from the disk into a common RAM of both processors and the opposite way.

The DOS-controller reserves two 256-byte buffers in the common RAM for every opened file, if they are of the prg- usr- seq- or dir-type. Both of the processors use these buffers by turns, which means, while, for example the disk-controller transfers it's data to buffer 6, the DOS-controller transfers data from buffer 7 to the computer. When buffer 6 is full, and buffer 7- transfer finished, the disk-controller writes the next block of the file's data into buffer 7 while the DOS-controller gets buffer 6 to the computer, and so on.

Normally the data transfer to a computer lasts longer than reading the data from a disk. Therefore the blocks of a disk are not used in their numeric order, but with an interleave of some blocks, for example 0,5,10,15,20,25,1,6,11,16,21, and so on. The time, the disk rotates to the next used block ideally is long enough to transfer the data to the computer, but not much longer. If the time is too short, the next actual block cannot yet be read, as there is no reserved buffer free, and the next opportunity to read it, comes after a whole revolution of the disk, and one revolution lasts 0.2 seconds.

I found out, that the DOS 2.7-block interleave does not suit the timing-requirements described above. It provides an interleave- factor 5 while I found 12 to be best.

UP TO SIXTY PERCENT SPEED UP ! ! !

If you use the new block-interleave, saving a file, it will be about 30 % faster. Every prg- usr- or seq file, saved with the new interleave, will be loaded 60 % faster. All you have to do, is to change one disk-RAM-address before you save a file using the following program:

```
open15,un,15
print#15,"m-w"chr$(244)chr$(16)chr$(1)chr$(12)
close15
```

We keep on improving the CBM-disk-handling. We have sent two pre-releases that use the "load*"-routine for reading the directory in less than 5 seconds (even if it contains 224 entries). One of them is the first step to a menu-program second to none. Another favorite project of mine is writing a diskmonitor containing some features, I have never seen before, but most of all I hope, our improved DOS will be available soon.

FAST-BUS *by Dennis Jarvis*

Jim and I are proud to announce that we have completed the ROM version of fast bus (version 3.0). How simple - up to now you were required to buy Gary Anderson's RAM/ROM expansion cartridge which cost $75.00. With our fast bus ROM version you will no longer be required to obtain this cartridge; however, you will need Gary's RAM/ROM socket. Along with this you are no longer required to purchase the FAST BUS SOFTWARE DISKETTE; however, we still recommend that you obtain this disk as it contains massive documentation as well as many useful EXAMPLE programs on using fast bus.

If you are planning on doing any software development on the B series and wish to use our internal routines you will need to obtain this optional RAM version of FAST-BUS 3.0 DISKETTE. Also, if you were one of the many people who bought one of our previous fast bus versions that was on diskette and you would like to exchange it for the version 3.0 DISKETTE (not the rom version) we will give you 50% credit off the new diskette. To obtain this discount, just return the label from your old version of fast bus diskette to us along with your order.

I have included two changes in the new version of FAST-BUS 3.0. We solved the problem involving SUPERSCRIPT, and to circumvent the problems with LIZ-DEAL'S software patches we have come up with a method of turning off the FAST SERIAL BUS part of the serial bus. When you first activate your fast bus software (described below) just hold down the ESCAPE key. When this is done you will still be able to use your serial bus devices, but they will revert to the SLOW SERIAL bus. Also, if you are a software developer, you will find the next feature a life saver. From time to time you will find yourself having to reset or turn off the B series computer several times and you will get tired of seeing our startup screen over and over and over again! To prevent this just hold down the RUN/STOP key and we will not display the startup copyright notice.

INSTALLING FAST BUS:

RAM VERSION - Insure that you have GARY's 24k RAM/ROM cartridge installed.

Now, insert the FAST BUS into DRIVE 0 of DEVICE 8 and then press the **SHIFT** and the **RUN/STOP** keys at the same time and'FAST-BUS will auto load and install for you!

ROM VERSION - Follow GARY's installation procedure for the ROM chip in his cartridge.Now turn on your computer and fast bus will come up all byitself!

As you can see, we are still making improvements in the FAST-BUS stage and will go on doing so as we continue to make the B serial a pioneer into the future!

Currently the B has two COMMODORE firsts.

1) The B series is the first IEEE computer to become SERIAL BUS COMPATIBLE.

2) The B series is the first COMMODORE computer to give you the ability to turn off the FAST SERIAL BUS operation. This option was not included in the C-128 computer which prevented many software and hardware packages from working correctly.

CONCLUSION: With a continuing effort from ALL of CBUG's members we will have everyone stand up (or pass out) and take notice of this quiet little machine!

Dennis Jarvis
30 Louisiana Ave.
St. Cloud, Florida 34769

Editors Note:

Dennis wanted the group to know that he has reduced the prices on the fast-bus materials drastically. He is doing this to promote more interest in this project, which has been dormant for almost two years due to the lack of communication with the CBUG Membership. Please contact him directly regarding these materials and all upgrade information. He has many new ideas in mind, he just needs to know that there is some interest.

During the two years since this article was written, Alan Bouvier developed a new method of deprotecting the original Superscript. He added some new features including a software modification eliminating the doubling of characters. This patch allows Superscript to run unaltered with the older and I presume current fast-bus routines. **Go Alan!**

UK DIAGNOSTICS FIX *by Tony Goceliak*

Some time ago, I modified the entire series of 8032 based diagnostic programs provided by Norm to allow them to run on the B computer. I did not fix many of the bugs in the programs, intentionally so. However, recently a member wrote to me complaining of the results produced by the "uk diagnostic" test. The test as published will report that your (u.s.) basic low rom is bad, with a checksum of A0, when this is perfectly normal for the 'standard' u. s. machine. The test will also report that your keyboard is bad, because, as explained in the introduction to the test, jumper plugs were expected to be connected to the main board instead of a keyboard.

The member had a valid point. Since this is a computer diagnostic test, and since presumably when you are trying to run it you are in doubt of whether your B is working correctly, I have provided the following procedure to eliminate the incorrect 'bad' reports from the Uk test.

1. make sure that a **ram cartridge** is installed on your b. **If not, power down before installing it!**

2. now type in the following, (with a **c-return** at the end of each line), and the B-ieee diagnostic disk in drive #0 of unit 8.

```
load"uk diagnostic",d0,u8,b15,p8192
bank15
poke8788,160
poke8944,0
poke8994,0
scratch"uk diagnostic"
y
bsave"uk diagnostic"+chr$(160)+"+",d0,u8,b15, p8192
to p11264
?ds$
```

Provided that the ds$ message was 00,ok,.... etc., you will now have a copy of the Uk diagnostic which will only report 'bad' on one of our u. s. machines when there is indeed something bad. The keyboard test is compromised, but you can test all that you need to know from basic. If the Uk diagnostic gave your machine a clean bill of health and you see the wrong letters echoed to screen when you attempt to type from basic, your keyboard circuitry is highly suspect.

BACKUP! BACKUP! BACKUP! *by Roy Sherman*

Another case of inaccessable data was brought to my attention recently. In this case it was a read error on the directory track. There were no backup disks, many weeks work down the drain. Some of the data was no doubt gone forever.

Please! Backup your disks! And don't think that this is just a B128 problem. The IBM and compatible community is absolutely paranoid on backups. PCDOS/MSDOS can trash a disk quite easily, with no warning whatsoever. While I'm on the subject, the B, at 2 mhz, is much faster than the XT at 4.77 mhz. In fact it compares quite favorably with the Turbo XT.

Returning to backups, always store the disk you made the backup FROM, your current disk, as the backup disk. Use the disk you backed up TO, the copy, as your current disk.

There is a very good reason for this. You are reasonably sure the disk you were using is good. At least, you've been using it. You have absolutely no idea if the copy is any good at all until you use it! Believe it or not, there could be nothing at all on the copy.

Another thing, never backup a disk on your only backup copy. If something goes wrong in the backup process you could end up with . . . nothing. So, what I'm saying is that two backups are an absolute minumum. Remember, more is better.

Everyone knows how to backup on a dual drive (8050/8250). But, how do you backup a single drive such as the SFD? You can use Jim Butterfield's CopyAll, it's on Liz Deal's disk, IF all your files are **PRG** or **SEQ**. Superbase is another matter since the database itself is essentially random files which **CopyAll** won't copy.

To be able to backup Superbase data on an SFD the first thing is to never use more than half the space on the disk. This is to leave room for the **export** file. Superbase cannot export to another unit, only to the same disk or the other drive in a dual drive unit.

When you are ready to make a backup **export** all Database files. Format a disk. Create your database(s) on it. (database"filename" **return** answer y to the 'create it?' prompt.) Do not, repeat do NOT copy the Database file, the one in all capitals. Exit Superbase and run **CopyAll**. If you have two drives the process will be easier because you won't have to keep swapping disks.

When **CopyAll** is done, reboot Superbase. Call the database. Call each file. Superbase will say 'File does not exist. Create it?'. Type y . Now **import** the file you **exported** above. When you have done this for all databases and files your backup is complete. This will become your current disk. You get a fringe benefit from all this. You will find that it accesses the data much faster than before. The original disk is now your backup.

There is a program called **'Unit to Unit'** in the library that will transfer the data directly from one single drive to another, but I don't recommend it unless your database is very small. It works the disk drives exessivily and is very slow.

That's all for now. Hope this has been of some help to you.

8050 / 8250 FILE UNSCRATCH *by Tony Goceliak*

This article describes how to accomplish file unscratching by disk drive programming. In particular, it illustrates one effective use for a parameter - passing ampersand file. Without the ability to send filenames, variables, etc, to your drive, there is a limit to how much you can ask a drive to do. Although it could be programmed to unscratch the first filename it came to that had a zero filetype (indicating a scratched file), what filetype would you like to assign the resurrected file? Will it be the proper file to resurrect? Will it create an effective mess with 'y' shaped files sharing the last n sectors? Remember you may have scratched two files last week, then expanded some data files until they wrote over the file blocks which the scratched files used to own, and now that you scratched an important file today, which file will a 'mindless' program attempt to re-activate? There are two solutions, both of which I have explored, but this is the better way.

One way communication

Parameter passing lets us instruct the ampersand file to skip any number of active and dead files until it sees the **EXACT** filename which you sent with the print#15 command which invoked the ampersand file in the first place. Assuming the file can be found, (by the way, both '?' and '*' pattern matching are allowed), it will be resurrected with the filetype you select, (even an intentionally wrong type), with everything in english (If you accept rel, del, prg, usr, or seq as english!). Rel files may be unscratched, with both main and side sector chains tracked down and re-allocated on the bam. All files when resurrected, will be traced and their blocks protected, with success guaranteed unless you have written to the disk between the time the file was scratched and now, when resurrection is being attempted.

This does not mean immediately, or even within the same computing session, since the code does not rely on anything other than what is written on the disk. The bam is updated by allocating the chains FROM THE STARTING POINT OF THE BAM ON DISK, so rnd files, machine code designed to be b-e'd, etc. will not need to be exposed to the dreaded collect command, and not incidentally, so we will finish in a reasonable time. Both 8050 and 8250 bam formats are supported automatically, with the exception that an 8250 disk with a file on the 'far side' of the disk cannot of course be traced with an 8050.

How to use the code

The syntax used to activate this disk program is both more exacting and curiously more flexible than the syntax used to run a simple ampersand file. The choice of syntax was more or less natural due to limitations common to both the drive and commodore computers, so here goes.

1. The whole kit and kaboodle must fit into the command buffer.

What good is a file like this if you need to spend 5 minutes 'm-w'ing the filename somewhere? The ampersand command AND filename/filetype designators must all be sent with one print #15 command to the drive.

2. Accepting #1, the command must be unambiguously allowed by both computer and drive.

Some special characters can cause premature termination of the string, or incorrect transmission at least. Obviously, the filename for resurrection carried none of these or it wouldn't have been allowed in the first place, but watch the parameters. Many characters are not allowed when you are running perhaps a wordprocessor or a similar 'take over the whole machine' program. Therefore filename and filetype must both be english.

3. The syntax, although exacting, must not be impossible to remember or use.

FAIR ENOUGH!!! None of those 50 page manuals needed here. No need to simultaneously press shift, esc, ctrl AND reset before pressing 'a'. Just type. All right what is the syntax? First one no-no. open15,(whatever),15, "&unscrat....." (the easy way to invoke an ampersand file) will not be acceptable for two reasons, (briefly due to some cbm computers limitations) that will not be gone into here.

open15,(whatever),15 : print#15,"&unscr*ab?de*,rel" is the correct method to allow the ampersand file to load itself from drive #0 and attempt to find and unscratch a rel type file named ab?de* on drive #1 (wildcards as defined in the drive manuals). The first asterisk must be placed far enough along to uniquely identify the ampersand file &unscrat.nam.typ but is REQUIRED to notify dos that the following characters are not part of the ampersand files' name. Then continue by typing either the full or pattern matched target filename, and separate it from the filetype with a comma.

The asterisk is acceptable with common meaning to dos and your commodore computer, so its choice as the marker dividing ampersand name from filename is a natural, as is the choice of the comma for its use, since it is rarely allowed within a filename (never say never). The code assumes that whatever is sent after the first asterisk is the filename to search for. Syntax of print#15,"&uns**,u" is perfectly acceptable, yielding resurrection of the first file with scratched filetype as a usr file. I do recommend, however, the use of minimal pattern matching, since it is probably what got you in trouble with a scratched file in the first place. That horrible realization sinking in that scratch "a*" yielded TWELVE scratched files!

Here is a good place to note that my implementation of the asterisk is slightly different than commodore's since only one scratched file at a time is resurrected, even when further matches are available. It was a conscious decision, since not all the files may have had the same filetype. Run the ampersand file using the same pattern "a*" until the drive error led turns red to return to your starting point. When in doubt, just pick a filetype. You can always straighten it out later, after you've seen the whole directory.

The Inevitable Warning

It cannot be stressed too much that you should not write to the disk between the action of scratching and the attempt to unscratch. The ampersand file does not demand you to sacrifice whatever is in your computer's memory, or even care much what language you are currently using, as long as you can open the command channel and send the proper string, so there is no worry about lost text or programming being dumped just to rescue a scratched file. During the time the file is scratched, all its blocks are unallocated, and dos is ever too eager to overwrite them and permanently poison the file. Don't give it the chance.

The action of unscratching must not be performed with blind faith, because there are potential pitfalls. The main failure case is as follows.

1. original directory had three files

a	**prg**
b	**prg**
c	**seq**

2. file a is under development, and you, fearing save@, save the better version as a+, scratch a and then rename a+ to a.

3. At this point, you inadvertantly scratch file a.

A virtual directory would show the following filename entries;

'a'	**scratched**
b	**prg**
c	**seq**
'a'	**scratched**

The first 'a' will be the file resurrected, but the file is not the one you desired. Run the ampersand file a second time and the second 'a' gets resurrected, but dos will ignore it until you scratch (USING NO PATTERN MATCHING!!!) 'a' once more. This will finally result in the correct file becoming accessible to dos (and you!). If you rename files in this manner, run an ampersand file like '&no hole dir' to keep the pitfall from ever being set up.

Stop blabbing, where's the Code?

*** (the article on my disk has fully commented source code, my actual assembly file inserted at this point as well as a listing of the basic language file generator program. To conserve space, it has been eliminated here) ***

To sum up, first type in and run the ampersand file generator, save it if you wish, but the ampersand file itself can be copied. Whenever you need to unscratch a file from basic, put the disk with the ampersand file in drive #0, the target disk in drive #1, and type **open 15,8 (or whatever the unit address of the drive is),15**

(From other than basic go to the disk mode or open the command channel, and type all after the print#15,)

Next type **print#15,"&unscra*XXXXXXXX,seq"** - where the X's are replaced with the filename you wish to resurrect, with or without pattern matching, ending the filename with a comma and then typing del - seq - prg - usr - rel - or zero (z-e-r-o not "0"). You **MUST** include the first pattern matching asterisk to inform dos where to stop looking for the ampersand file name, there must be at least one character for a filename, (* is acceptable), and **SOME** filetype must be specified. If you don't know it beforehand, just guess (but don't guess rel unless it is a rel file, or the drive will look for a side sector chain on track #0)

When the ampersand file is done, take a gander at your error led. If green, a file has been unscratched, red indicates not, due to improper syntax, no match found, or a disk fdc failure. If bam conflicts have not occurred, the bam has been updated and rewritten to disk, but if only partial recovery was possible, the activity leds will be blinking, the bam has been left as it was, and the drive will not respond to anything (to alert you that the partial file is still at risk of being overwritten) until you pull the disk partially from the drive. Read or load the file as appropriate, salvage what you can, and when you are ready to jettison the unprotected filename, re-run the &unscratch utility again, assigning filetype zero. **DO NOT SCRATCH** a file which left the leds blinking, or you will expose the 'other owner' of the blocks in conflict to possible corruption.

USING THE 4023 *by Alan Bouvier*

There are things that Commodore forgot to tell us. The following involves the Commodore printer. I am sure it holds true for the 8023 as well as the 4023 since both printers are based on the same operating system.

By now, most of us have found out that turning on paging also sets the top of form. And likewise, turning paging off does a form feed if the paging had been turned on first. Once I found that out, I went wild turning paging on and off to get my printouts to look neater. Now I have a better way. It seems so obvious now, but I just never thought of it.

Back in college I had to learn about the American Standard Code for Information Interchange or ASCII for short. This code had many control characters for printing. One of these commands is the FF or Form Feed character. Although never mentioned in the printer or programmers manuals, sending a form feed character to the 4023 causes the paper to advance to the next top of form. Just as if the paper advance button was pressed! And better yet, it works with paging on or off. No more switching paging on and off.

From basic, **the form feed character is chr$(12).** It can also be included in a text string by typing CTRL-l while in quotes mode.

There is one thing you must be aware of: **the form feed is executed immediately after being received.** Anything still in the printer's buffer will be printed on the next page when it receives a carriage return. All you have to remember is to send a carriage return before sending the form feed.

(The carriage return is also an immediate command. To the printer, a carriage return means: print the line stored in it's buffer, advance the paper one line, and reset all temporary commands. (enhance,quotes mode,case changes,& reverse field).

The shift-ret 'chr$(141)' prints the buffered line without advancing the paper. Reverse and case shifts are reset. Enhance and quotes mode are not. Any following characters will be printed starting at the first column. This will print on top of the already printed line.

A line feed 'chr$(10) or CTRL-j' will act as if it were a carriage return.

I would also like to correct the 'uppercase' and 'lowercase' character definitions as in the manual. **Chr$(145) will switch the printer from the case selected by secondary address to the other.** When turned on, the printer is in graphics mode. Therefore, printing chr$(145) to the printer will cause any text following it to be printed in lower/upper case. **Chr$(17) will only switch the printer into graphics mode.** These changes only remain in effect until a carriage return is received by the printer. It then returns to the case selected by secondary address. Because of the way these control characters act, if you want to switch back and forth between cases on one line, it is best to start off in graphics mode. Otherwise, you will find it impossible to return to lower case by these characters.

I hope this helps make it easier for all of us to make better looking printouts.

(Superbase Programming continued from page 1)
most complex accounting or data entry system you can imagine. SuperOffice goes a step further and can manipulate text files, automate mail merging and anything Superbase and Superscript can do. The limitations are the designer and the storage medium, not the program.

Before we begin the programming tutorial, we will cover some information helpful in setting up a database. The following hints, tips and general information will help you get you off to a good start and help avoid future problems that may occur due to inadequate planning. So, here we go!

Superbase is different from most database systems in that it sets up a control file that will store the important information. The database files, number of records and locations for the start of the files are controlled by this master file. While they call this file the database file it really is only an umbrella for the real database files.

The Superbase database file allows you to relate other files under the same umbrella name to each other. If a file called orders and a file called customers resided in the master database file accounting, you could access data from both if they had a field in common. One of these fields must be the key field. You can not relate two files in different database structures. The common definition of a database is a file or group of records containing related information. Superbase confuses this issue by using the term database in a general sense. There can be unrelated files under a common Superbase database name.

The normal Superbase startup routine follows:

1.) Load program by pressing shift-run.

2.) Program prompts you to enter a data disk or create one.

3.) Will create a data disk if requested or executes step 4.

4.) Program attempts to load the start program.

5.) Start will execute if found and asks for database name.

6.) If start program is not found it will exit to menu 1.

The start program initializes the system parameters, sets up the function keys, displays the name of the program, asks for the database, the file name and in SuperOffice's case will ask for the printer type. The start program can be modified like a normal program and can do anything that you would like, such as displaying a menu or loading an application. We will examine the SuperOffice start program after we have some programming experience.

Let us assume you want to create a simple mailing system or a list of all your friends and relatives addresses. The first step is to decide what specific information you need. Try to keep your files concise and do not add superfluous fields with data that you will never use.

Put information in separate files that would cause redundancy in records. For example, if a customers name and address is entered in the order file, every time you enter an order you would duplicate the name and address. Result fields should be avoided because they require disk space. It is more efficient to write a simple formula to do your calculation.

Since Superbase uses a minimum of 128 bytes per record, care should be taken to keep your character count below 123 bytes. If you need more space per record you might as well set up a second file. Use a key field with the same name as a text field in the first file. This allows you to use the multi-indexing scheme mentioned earlier. Use this concept only when your database is reasonably small and can afford the 256 bytes or more per record. The multi-indexing scheme can work on up to 15 files but is limited on speed, if you have to keep opening new files to use it. Three open files are the maximum allowed with certain limitations.

When you are determining the length of a key field, keep in mind that a longer key field will use more disk space. So keep your key field as short as possible. It is advisable to use only unique keys unless you have a strong reason for duplicate keys. Duplicate keys can not be accessed using key lists or hlists which are integral to many Superbase options.

Field lengths can be adjusted after entering data provided that the size of a field is not decreased below the largest entry to that field. Put your most used fields or forced fields at the top of the form and fields that do not require data entered often at the bottom. You can store the record without entering data in all of the fields, providing all the forced fields have been entered.

For those of you that have used Superbase, the form design screen may be taken for granted. For those of us that have used other database programs, it is a marvel. This screen designer resembles material found in the JCL Workshop. As a matter of fact there are so many resemblances between the two packages, it makes one wonder if Precision Software incorporated material from the JCL Workshop to make Superbase.

The form designer allows you to literally paint the screen with your file format. You may use reverse video and graphic characters for effect. The fields and descriptive text can be placed anywhere on four separate screens. I have never used an easier form designer. To understand all of the options, please read the Superbase manual starting with page R-7 regarding it.

For this tutorial, we will design a record form to work with. Load your Superbase disk and insert the data disk when requested. Type test when asked for the database name. You should default to the file menu after the database has been created. Enter addresses for the name of the file and the format screen should appear.

Enter the following fields in any manner desired. Experiment with different screen layouts and use the inverse screen functions where desired. Press escape-k for the lname field and escape-t for all of the rest. Cursor to the right for the desired number of characters per field. They are displayed on the upper right. Escape-e will erase a field the cursor is on if necessary, though the field name must be deleted manually.

field name	length	type of field
lname	5 characters	key
name	10 characters	text
address	30 characters	text
city	15 characters	text
state	2 characters	text
zip	5 characters	text
phone	12 characters	text

Press escape and then the stop key to finish your screen format. When asked if you want duplicate keys press y for yes.

Please enter a few practice records so we have some data to work with when we start programming. Press 1 to enter the record appending function and enter at least ten records. Press return after the last field of every record to store a record. If you want to store the record without entering all of the fields you may press shift-return. The key field and all other forced fields must be entered for the record to be stored.

After you have designed your format for data entry, the next step is to write a program to manipulate your database. A menu program to control entry of new records, adding new records from existing records, replacing or editing records and deleting unwanted records would be nice. Why should we do that? I can do it all from the Superbase menu!

That is true but you can't control what the user does from the Superbase menu. There are safeguards which can be programmed to ensure that a user can not damage or modify your data files. You can control access to files and deny the user the capability to make modifications to different files. The question arises as to how to program this menu. There are two methods.

The first method is to code display statements for everything you want displayed on the screen. This is a lot of fun and everyone who wants to program should have to do this at least once. The difficulty with this method is that you have to determine what looks best by trial and error. Now all the manly or womanly hardcore programmers are probably snickering, because they know that is the only way to create a menu. There is an easier way but I will give you a couple of examples of display statements first.

The first step in creating a program, in both Superbase and Basic, is to enter a line number. The line number will help you refer to different routines or groups of commands in your program. It also labels the statement at that location. It is preferable to start with a number such as 100 and go up in steps of 10. An example would be 100,110,120 and so on.

The display window has 80 columns and 22 rows to display text, graphics and fields on. The display command consists of several parts which include the @ sign, followed by the column and row that you want to display on. You will see that I use a rem statement after each line. You use a rem statement to enter a comment for the line or routine. When the program interprets the rem, it will not execute anything after it. Type the following examples of display commands. First press return and 5 for program writer from menu 2.

```
10 display @1,1"Hello world":rem displays at column 1 row1
20 display @40,9"My name is":rem displays at col. 40 row 9
30 display @75,20@+"Bruce":rem reverse at col.75 row 20.
40 wait:rem wait for user to press key
50 display @0@13,15@+"Bye world!":rem Reverse.at
col.13 row 15.
60 wait:rem wait for user to press key
70 display chr$(147):rem clears screen
```

Press **esc** and then the **stop** key. Upon return to menu 1, **press 7** to execute the program.

line 10,20 - The results of line 10 and 20 are obvious, the @column,row command displays the appropriate text at the column and row specified.

line 30 - In line 30 we have a new command @+. This command reverses the video for the string or variable displayed following it and including concatenated strings.

line 40 - In line 40 there is a new command called wait. Without the wait command the display will flash before your eyes and be covered by the menu 1 screen.

line 50 - In line 50 a new command @0 is implemented. This command will reset the screen to allow you to write to an area above a previous display statement. If you do not use it you will force a new screen display and erase the previous one. As long as you have your display statements following from left to right and row after row you do not need this command.

line 60 - same as line 40.

line 70 - you can display a chr$ code which will perform its' function or a string or variable.

Now that you have seen the normal method of setting up a screen display, I will show you an easier and far faster method. Not only will you be able to draw a screen faster, but it will display faster and save you programming space. The limitation of this method is you will have to reserve one open file for screen displays.

It is advisable to save programming space in all areas of Superbase. You have 8k of programming space including strings and variables and another 8k for arrays.This is odd, due to the large amount of memory available on the B computer. It appears that since Superbase was ported from the 8000 series, they made no attempt to increase the programming area. You will be amazed at how fast you can use up memory using display statements. If you set up your screen display with display statements, then store those statements in an array. If you store them in a string, they will use the bank 1 programming area. Arrays are stored in bank 2, which won't be used otherwise.

As stated previously, Superbase will allow up to four screens per file and a maximum of three files open at one time. If you use less than the maximum number of fields, you can use the extra screens to draw menus. How is this possible? Superbase requires that you enter at least one field per screen. Therefore, you can set up a screen of menu choices in any manner that you desire, but you must insert a minimum of one letter for descriptive text and a one character field on the screen. You will then cover the field entry with a prompt, displayed immediately following the calling of that screen.To implement this unique feature, access the main menu and type , **file "screen"**

You may use the same layout features that you used for the address file to set up your file display. I will give you the coordinates and data for a menu to use with programming examples that follow. You will have to count by using the cursor keys, as there is no means of telling your location on the screen. I usually press escape-s key to invert the screen and then type the menu heading and the numbers. Then press escape-s key for the normal screen display and type the menu selections. This has a more professional look to it. Type the following at the positions indicated. Do not type the column and row information.

Data	Column	Row
EDITING MENU	**33**	**3**
1 Enter	**33**	**6**
2 Replace	**33**	**8**
3 Add	**33**	**10**
4 Delete	**33**	**12**
5 Exit to Menu	**33**	**14**
s < >	**33**	**17**

To create the last entry type s and press escape-t for a text field and allow only one character. Remember, it is a requirement that a minimum of one field be on any screen format used. Press the escape key and the stop key to store the screen and continue.

Now, press return for menu 2 and press 5 for the programming mode. We will type in and examine a simple program to make use of our menu format.

```
5 sp$="":for c=1 to 50:sp$=sp$+" ":next
10 database "test",8,0
20 file "address"
30 file "screen":select c
40 file "address"
50 display @0@29,17@+"Enter Your Choice Now (1-5)"
60 wait a:a$=str$(a)
70 if a <1 or a >5 then 60
80  display @0@56,17@+&1,0a
90 display @0@29,17"Enter Your Choice Now (1-5)"
100 for c=1 to 3
110 display @0@20,19sp$:for y=1 to 100:next
120 display @0@30,19@+"Is Choice"+a$+" Correct"?
(y/n)"
125 for y=1 to 250:next:next
130 wait y$
140 if y$="n" then display @0@20,17sp$;@20,19sp$:goto
50
150 if y$="y" then 100
160 on a goto 200,210,220,230,240
200 enter:goto 240
210 select f:select r:goto 240
220 select f:select a:goto 240
230 select f:select d
240 menu
```

Rems for the previous lines.

5 - initialize variable sp$ to 50 spaces. This method saves programming memory by appending a space to sp$ each time the for-next loop executes. This variable will be used to erase screen prompts. A for-next loop executes a loop for a number of times as specified in the for statement. The for-next can be used for a delay loop, a counter, building a string or repeating an operation.

10 - opens the database on unit 8 and drive 0.

20 - open the address file.

30 - open the screen file and select the current record. The select c statement displays the menu screen we developed previously.

40 - make the open address file the current file.

50 - display the statement at column 29 and row 17 in reverse video which is symbolized by @+. Note the @0 before the column and row statement resets the screen display. This is necessary if you set a display statement above a previous display statement. This prompt covers up the field displayed on the screen that we entered to design this menu.

60 - Wait for numeric user input. The user must press a number key. The next command converts the number into a string variable.

70 - if a is any number from one to five then proceed to the next line. If a is anything else, then goto line 60 and execute from there. If an if-then is true, it will execute the next statement on the line, if any. If it is false, it will fall through to the following line and begin executing from that point.

80 - display a as a one digit number without any decimal places and in reverse video.

90 - display the prompt at the same location as in line 50 but in normal video. Note that you could have saved programming space by storing the prompt to a string variable such as ch$. You can utilize this anytime that you need to use the same display statement more than once.

100 - start of a for-counter.

110 - displays spaces at column 20, row 19 for the duration of the for-next loop that follows. A delay loop for 100 repetitions helps set up the flashing prompt routine.

120 - displays a prompt at column 30, row 19 and displays it for the duration of the for-next loop that follows. The last next refers to the for in line 100. This loop is repeated three times.

125 - determines the length of time for the previous prompt to display on screen.

130 - wait for the user to press a key. In this case any recognized character that can be entered into a string variable can be pressed.

140 - if the entered character is "n" then clear the prompt lines with the space variable sp$ and goto line 50.

150 - if the entered character is different then "y" then return to line 100 to get another entry. With just two lines we have limited the character entry to "y" or "n" and cleared the prompts.

160 - since a must be an integer from 1 to 5, the line number that corresponds in position will be executed. Therefore, if a=2 then the program will jump to line 210. Though we allowed for an input of numbers from only one through five, the following would happen if we had not specified a range of values. If a=0 or a5 then the program would execute at line 200. If a was negative you would receive an error message.

200 - the enter statement will bring up the record entry screen. You can save the record at any time by pressing shift-return or press return after entry to the last field. If you want to exit without saving press escape-q. After the record or records have been entered, the program will jump to line 240 and reenter the Superbase menu system. The key field must have data entered into it before a store operation is allowed.

210 - selects the first record and allows you to replace data in that record. The record will reflect your changes after storing. You may not alter the key field. If you need to alter the key field, then you would use the add record command which follows.

220 - selects the first record and allows you free control to change any data. The original record will remain intact after the add command.

230 - selects the first record and deletes it without any prompt. If all the records have been deleted or none are present to start with, the current file will be deleted from the database. The only way this statement can be used safely under program control, is by entering the record to delete into a variable. Then display the variable contents and prompt the user to determine whether the record should be deleted. NOTE: If you specify the

select d command in a program line, it will delete the current record without warning.

240 - enter the menu system.

For easier comprehension, I usually put one statement per line in this tutorial. Practically though, in order to squeeze as much as possible into our small memory area, you should put as many statements as will fit easily on the two lines allowed. Remember, every line number takes up space so it is smarter to use as few as possible.

Though commenting source code by using a rem statement is intelligent programming, it is better to print your program and pencil in the rems to avoid the memory loss. If your programs do not require much memory then by all means program properly and rem each line. NOTE: If you were using a compiler, it would ignore the Rems when compiling.

Well that wraps it up for this issue, I will expand upon this short program in a future issue. If you have any suggestions or input please write or call me at:

Northwest Music Center, Inc.
404 Inverrary Lane
Deerfield, Il. 60015
708-808-7000

Superbase is a registered trademark of Precision Software Ltd.

SPINNING YOUR 80XX FOR A CONTROLLED TIME

I frequently advise CBUG members who write to me of intermittant troubles with their 80xx drives to check and see if the heads are clean. Many members are reluctant to open the case on their drives, and considering the potential for damaging some sensitive components by static electricity, their reluctance is probably well advised. For them, there is no alternative to commercial head cleaning kits except shipping off their drives to a repair shop.

Following this article is a short basic program which will spin an 80xx drive selected by you by unit number and drive number for as long as you wish. No 'bumps' whether the unit contains a disk, no disk, or a non-disk such as a cleaning disk.

Some of the commercial cleaners are a bit too abrasive for my own liking, but as a one-shot alternative to several weeks without your drive and a repair bill, the commercial products become quite acceptable.

Dload and run my program titled 'spin 80xx', and after identifying the unit number and drive number at the prompt questions, the selected drive will spin. It will remain spinning until a few seconds after you press 'return'. One reminder, don't leave a cleaning disk in the drive for longer than the manufacturer recommends! *by Tony Goceliak*

Superpet 9000 continued from page 3)

loading of linker produced program files. Complete manuals with tutorials examples are included for each language.

This is the optimum environment for a B-128 or other CBM IEEE user if they desire to learn other languages. They do not have to buy any new equipment such as drives, cables or printers. They can utilize software in the TPUG and CBUG library plus any commercial software still available.

For further information call N.W.M.Inc. at 708-808-7000.

LIBRARY DISKS REVIEWED

Goceliak 1990: Does this guy ever stop amazing us with his new discoveries. If you want to learn what the B machine is capable of, Tony Goceliaks many disks are the way. His understanding of our system is astonishing. One has to wonder whether he sleeps at night or just has his B-128 do it for him.This disk is a must for Goceliak collectors. This disk includes the following.

Price $10.00* Royalty paid. Shipping $2 any quantity

1.) **One disk - two directories**
2.) **Alphabatize your directory**
3.) **Unscratch programs leaving memory intact.**
4.) **File locking program - prevents scratching.**
5.) **Obliterate a file so no one can resurrect it.**
6.) **80xx spinning program for cleaning drives.**
7.) **Prime Factoring program and much, much more.**

Goldcoast Gambit: Fred Peterson one of the many fine contributors to the CBUG library has done it again. After you shift/run your eyes are treated to an interesting video creation for the B. Fred includes not only a complete description of all the programs on the disk but a menuing system to access them. His disk includes his SSII Financial Spreadsheet. Fred states this a satisfactory though simple method of keeping a record of all receipts and disbursments for a small business. This disk has many different types of programs from utilities to games to science and mathmatical challenges. Fred really spent some time putting this one together.

Price $5.00* Freeware. Shipping $2 any quanitiy.

1.) **SSII Financial Spreadsheet.**
2.) **Normopoly - a properties buying game runs on 8432e.**
3.) **Backgammon - 8432 emulator.**
4.) **Chessmate - 8432 emulator.**
5.) **Games - Utilities - Disassembler etc.**

* *ALL DISK ORDERS MUST BE A MINIMUM OF $10*

(8050 continued from page 1)

mounted this diode very near the motor power transistor which can get fairly hot. This heat spreads over the circuit board to the diode. The Zener reference diode chosen by Commodore is overly sensitive to temperature. When the comparator circuit looks at a feedback and a reference that vary with heat, the motor may end up at most any speed.

Now for the technical part. I do not recommend that any of this be attempted by someone not skilled in the repairing of circuit boards. I have changed the tach feedback diodes (CR20 & CR24) to 1N4148 and stood them up on long leads to be 1/2 inch above the board. I installed the precision reference diodes (LM329), mentioned in an earlier Escape article, to replace CR21 and CR25 and changed R49 and R59 to 2700 ohms as necessitated by the reference diode change. Looking at the front of the board, I lifted the left end of R52 and R63 and soldered the anode end of a 1N4148 diode to the hole the resistor lead came from and soldered the cathode end to the now free end of the resistor sticking up in the air. This diode is the same as the one put in for the tach feedback, but is in the other leg of the comparator circuit. The drives were then recalibrated for speed. The net result is that the drives now vary very little. The stood-up diodes are not affected as much by the transistor heat as the old ones were and when they are affected by the heat of the general area, the diodes placed in series with R52 and R63 are also affected about the same amount and trim the reference voltage counteracting the tach change. I still check the drive speed regularly, but I have not had to adjust them in the past year and a half.

YELL FOR HELP!

The people listed below have graciously offered their expertise and time to help fellow CBM computer users. I applaud their generosity and want to thank them for their willingness to share their knowledge.

1.) Please call them only during the hours listed and don't call collect.

2.) Please don't abuse this privilege and only call them when their help is genuinely needed.

NAME	AREA OF EXPERTISE	TIMES AVAILABLE	PHONE OR ADDRESS
Louis Black	Calc Result - Paperclip	7:30-11:30 PM EST	416-728-3244
J. Boyle Electronics	Forth - 65xx ASM	Sat. 7-10 EST	904-539-0506
Alan Bouvier	SSII,ASM,Basic	M-F 6-10 S-S 12-8 CST	504-649-5772
Edwin Bowerman	Basic - SSII	Mail Only w/S.A.S.E.	47 Parsonage Ln Topsfield,Ma. 01983
Art Chick	Basic,SSII,SBII	T-Th 7-10PM PST	916-674-7006
Dennis Jarvis	ASM,B-MSDos,Fastbus	M-F 9-12 AM EST	407-957-2840
Vern Kempfer	General Info	Evening & Sat-Sun CST	608-244-3353
Bob Loeffler	CABS GL-AR-AP	T-W CST	414-294-6412
Fred Lovejoy	SSII	M&W 7:00-8:30 PM	602-946-0202
Dan Mikesell	Basic	7-10 PM EST	616-842-4205
Carter Pawlus	Calc Result	9-9 PM CST	414-457-6100
Fred Peterson	SSII,SBI,SBII	M-F 7-5 PM PST	805-492-0066 **Will call collect if you leave a message**
Tom Rehm	Superbase	M-F 7-9pm CST	708-851-6528
Don Wolf	SSII,Basic	M-F 6-9PM S-S 9-5CST	816-524-8491
Robert Walther	SSII,SSIII	Mail Only w/S.A.S.E. **Phone in future.**	20209 150th Dr. Sun City West, Az. 85375. **Just moved here so I'm not sure.**
John Wright	CP/M-MS Dos for B-128	M,T,TH,F 7-10PM CST	402-339-5728

Anderson Communications Engineering

2560 Glass Rd. NE. ***VERY IMPORTANT,***
Cedar Rapids, Iowa 52402 ***READ THIS!***

This is a plug for Gary Anderson. This man has single handedly developed most of the publicized add on boards for the B-128. Among his many accomplishents are the 1 meg add on board, 24k memory cartridge, the alternate operating system board and he developed a prototype v-2 or suped up 8088 board. Due to his supplier discontinuing production of circuit boards for outside firms, Gary has decided not to retool and has discontinued the manufacturing of these boards. This is it folks, if you need them, you better buy them now!

At the time of this writing he only had a few boards left.

The following products are available while supplies last.
24k ram cartridges @$75.00
B-1024 Expansion boards @$329.00

Hot Info: *June 1991*

Northwest Music Center Inc. reports that their supplies of B-128 computers and 8050 drives are rapidly decreasing. At this time there are less than 60-B-128 computers left and approximately 25-8050 and 8250 drives remaining. It is suggested that if you use your B-128 often and for business applications that you back your system up now.

Fred King of King Communications has announced that he will still perform1 meg upgrades on B-128 & B-256 computers. Contact him directly at **715-341-1149**.

Northwest Music Center Inc. still does repairs on B-128 computers, 8050 drives, 4023p and 8023p printers. Call for details. **708-808-7000**