THE CBUG ESCAPE

NEW THIS ISSUE:

*MULTITASKING!

*FULL GLOBAL DISK SEARCHING

*SPREAD SHEET TUTORIAL

*MULTI-KEY IN SUPERBASE -- HOW TO

*COMPLETE SERIAL BUSS COMPATABILITY
(Other drives & printers)

*DISK DRIVE SPEED UPS

*MAJOR CP/M-86 DEVELOPMENTS

*DISK FORMAT CONVERSION & TRANSLATION

CHICAGO B128 USER'S GROUP -- INTERNATIONAL (CBUG, Inc.)
4102 N. Odell, Norridge, Il. 60634, U.S.A.

Bulk Rate
U.S. Postage
PAID
Desplaines, Il
60018
Permit # 296

!! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !!
!! **POSTMASTER** !!
!! !!
!!**FORWARDING ADDRESS REQUESTED** !!
!! or !!
!! **FORM 3547 REQUESTED** !!
!! ! ! ! ! ! ! ! ! ! ! ! ! ! !!

DATED MATERIAL - DO NOT DELAY

NINTH ISSUE

Winter 1988

* *

THE CBUG ESCAPE is published 4 times a year, more or less, by the Chicago B128 User's Group - International (CBUG, Inc.), an international membership organization in support of applications and usage of the Commodore B128 Computer. Note that some issues are combined in one publication.

CHECK YOUR ADDRESS LABEL FOR EXPIRATION DATE. The expiration date is located at the top of the address label in the form of YYMM, eg. 8712 indicates an expiration at the end of December 1987. CBUG will be unable to mail publications without renewal. **PLEASE KEEP YOUR MAILING ADDRESS CURRENT.**

CBUG is NOT affiliated or allied with any other organization, users' group, business or other entity of any kind, except in support of CBUG chapters.

Advertisements, articles and contents of disks are solely the responsibility of the individual authors. Their existence in a CBUG publication does not imply any endorsement by CBUG. Please report to CBUG exceptional performance, either pro or con.

Publishing address: CBUG, Inc. c/o Norman Deltzke, 4102 N. Odell, Norridge, Il. 60634 USA. 312-456-8720 7pm to 11pm CST.

NOTE: **Due to a fire May 8 at a major phone company switching office, Chicago area phone service has been severely disrupted. Full long distance service may not be restored in our area till mid or late summer. Be persistent or ask for inbound operator assistance. Remember we may not be able to call you either.**

Cover price this issue: $6.00. 1988 subscription rate: $14.00 (bulk rate, U.S. & possessions ONLY); $20.00 (first class, U.S. & possessions, Canada & Mexico); $21.00 (surface <boat> first class, all others); $35.00 (small packet rate air mail, any country).

© 1988 CBUG, Inc.

* *

TABLE OF CONTENTS

CBUG SCRATCH PAD

By: Norman Deltzke

May. 10, 1988

Well we were catching up a little -- this issue will mail only 3 weeks late. Thanks all our contributors who did successfully make the effort to get materials in on a timely basis. Unfortunately, I over estimated my ability to make up for several weeks of illness last fall. With the usual extra-ordinary assistance of family and friends, one more extra push has made it happen.

I hope materials for the next issue will start flowing in shortly so we can minimize the deadline rush. Whilst this issue is late, thank you contributors for doing much better this issue than before.

We've made a couple of changes. By popular request the order form (2 copies) is now in the center of THE ESCAPE instead of inside cover. PLEASE take extra care to neatly and legibly write your address on the order form since we no longer have your mailing label to double check. For the benefit of the post office, the address location has been changed.

Take a look at what is new and coming in this issue. Dennis Jarvis and Gary Anderson have succeeded in doing what all others said was impossible -- to allow the B128 to use all other Commodore periferals such as serial disk drives, printers, etc. More importantly, using the 1571 both text and software from nearly every other system and format can be loaded into the B128. In the case of programs, there remains the conversion and adaption process, but for most text files, the capability should be directly useful without significant effort. There will be a series of future articles born of this important new capability. There are several new capabilities from our author members described in various articles.

From our friends at The Independent Commodore Products Users' Group in London England are two recent articles on Superbase -- how to implement multiple key names. While atleast one person has modified the program itself to do this, the information was considered by that programmer to be proprietary. The ICPUG program is another way, though a slightly consumptive of disk space, to accomplish the same thing. The programs listed were written for the C-128 but should work directly or with minor modificiation on the B128 system. Hey folks, how about some CBUG development work in this area.

Be sure to read the Library Lead article as there are several disk recall/upgrade offers at no cost to members. Most of our new releases are covered in the usual form, however some, such as the Jarvis/Anderson project have been partially printed in the article section of this issue of The Escape, as such seems to be the easiest way to alert everyone to these major works.

Most of us follow the news reports of the value of the US Dollar. That together with the legislation to limit importation of memory chips from several far-eastern countries is currently causing parts shortages and up to 1000% increases in critical parts prices. Meanwhile domestic manufacturers appear to prefer to concentrate their capabilities on "cutting edge" technologies. The net result is that many products are becoming expensive, often unaffordable to manufacture. In the B128 world the first ramifications are shown in Anderson Communication's prices for memory boards, cartridges, etc. While the aborations are profound in newly built products, the traditional B128 and its existing periferals generally will remain uneffected.

As to where we are going, for those interested in MSDOS, CPM and CPM86 capability, tomorrow seems to be getting closer. CPM86 is quite operational; infact the CPM86 version of the most important current word processor, Word Star Professional, appears to be fully operational without any modification. Now with the Jarvis/Anderson Fast Buss available, it will be practical to load and run CPM86 programs via the 1571 then thru the 8050! Future issues of THE ESCAPE will detail this capability along with the progress in offering a CBUG member designed and built co-processor superior to the one designed by Commodore. You know, it seems more and more that when some aus-lander says it can't be done, the CBUGers' just won't listen -- they just go do it anyway!

Ah, but enough for Hi-Tech. In this issue is yet another installment of Warren Swan's Basic Tutorial. A new voice, Fred Peterson is offering two new disks, both intended for the less than technical. Infact, his tutorial should have been around when computers were first introduced to the public -- and ought to be included with every unit delivered! It's a light hearted treatment of just what is the computer, what are the commands and how to use them. Just simply, not a college course. There are several authors in this batch of library disks directing their effort more torward the "utility user" base of our membership. And a few letters to the editor pleading for such help. Things ARE comming together!

SEI reported a few weeks ago they had only 8 drives remaining to repair. If you are one of those and are not receiving adequate responses from SEI, contact Inspector Mike Riddle, U.S. Post Office, Charlotte, N.C. 28228-3000. While a few of the units delivered experienced minor problems likely due to shipping damage (and were promptly repaired by SEI), most reporting customers have stated that the repaired units have performed exactly as represented and are running reliably.

THE PENNSYLVANIA CONNECTION

by: Liz Deal

******* PARALLEL USER PORT *******

The Paraller User Port is burried inside the computer. On all other CBM machines the user port is accessible from the back, but we don't have it so good. The connector we have to use is non-standard; we can't just plug in the old devices. But it is still the good, old, user port.

In my low-profile B128, on the circuit board, to the left and rear (rear is where all the external equipment gets plugged in, things like IEEE-488 cable) are two rows of 13 pins. That's the user port. It's marked P1. Pin 1 is used for proper orientation; it is shown on the board, and it is the rightmost rear pin.

In my high-profile B256 the user port isn't even marked with any number at all, there is no indications of where pin 1 is, I have no schematics and the entire board looks different. However, it is the only 26-pin port, and it lies about the middle of the board. A bit of snooping with voltmeter tells me that the orientation is identical to B128, i.e. Pin 1 is in the rear row, to the right.

Back of computer: power, IEEE, RS232...

```
25 23 21 19 17 15 13 11  9  7  5  3  1
 o  o  o  o  o  o  o  o  o  o  o  o  o
 o  o  o  o  o  o  o  o  o  o  o  o  o
26 24 22 20 18 16 14 12 10  8  6  4  2
```

Front of the computer: keyboard, etc.

The lines that are available on the User Port are quite similar to those found in the Pet, or the C64, but there are some differences, so you better be careful, else you'll fry your computer or the applicances. As far as I can tell from the schematics and actually playing with port the pinouts are:

1,3	ground (see note below!!)
2,4	go to PB2 and PB3 on 6525 TriPort not sure of their function.
5	is PC line on the 6526 CIA2
6	goes to FLAG pin on the 6526 CIA2 (as does the cassete's read line) can cause interrupts
7-14	data lines PB0-7 on the 6526 CIA2 these are the STANDARD user port lines you can use for projects.
15-22	data lines PA0-PA7 on the 6526 CIA2, or IEEE-488 data lines - this is nonstandard. Ideas anyone?
23	is CNT on the 6526 CIA2 (serial port counter on the C64)
24	+5 volts, so be careful with this one Pet DAC used to use cassette's +5 v and ground lines. On the B, I use 24 & 3.
25	is IRQ line going to the 6525 TriPort
26	is SP on the 6526 CIA2 (serial port in the C64)

NOTE: pins 1 and 3 are supposed to be ground lines. In the B128 they are. I have hooked up an old Pet DAC for playing music, and it has worked. The same arrangement in the High-Profile B256 is doomed. These lines aren't grounded. So my DAC can't get 5 volts from the port and nothing else can work. A simple wire bridge between the ground screw of the power supply and pin 3 corrected the problem. I do not know if my B256 is weird, or if they're all like that. If you play with the User Port, it might be a good idea to check it out.

6526 chip is well documented in the C64 literature. Connections to PC, SP, CNT, FLAG are described in the chip specifications, they all have to do with bit-serial I/O. Of special interest to us is the FLAG pin, as it is this pin that makes it possible to read CBM tapes, as Mr. Goceliak has done.

******* COMMAND HISTORY *******

I've written a little routine that may be of interest to some people. I'm sending it to Norm to put in library. The program is for use in BASIC environment. It captures all the commands you type on the keyboard in direct mode and displays them back on the screen when you ask for it. I've seen something like this on the Amiga computer and a Hewlett-Packard at work, and I like it.

Do you really need it? Well, perhaps not. But there are situations where it is nice to know what has caused a problem, or to know what kids are typing in the classroom while you (the teachers) aren't watching, and so on. While in Commodore machines much of the evidence can remain on the screen for a long time, and while such commands on the screen can be reused over and over by just editing them and pushing RETURN, it's nice to be able to recall something that has scrolled off the screen some time ago.

My program is ment to do just that. You can recall commands to the screen and reuse them. You can send them to a printer, using the standard CMD method, or you can write a disk file documenting what you have been telling the computer to do. The capture buffer is failry large, 1000 bytes, and you can change that to any size if you reassemble my code.

The program requires bank 15 memory expansion, only because that's where I put the code and the history buffer. However, I'm enclosing assembly source file so if any of you feel like recoding a bit for another bank, you should be able to do it.

As coded, it also runs in the C64, and the object file is included. There aren't that many instructions, but the few sentences in the source file should get you going.

******* REVIEW: 1200 BPS MODEM *******

Protecto, or as they're now called, COMPUTER DIRECT (same address where you got your B) are selling a Hayes compatible modem for about $80. I think it's a good buy. It's a strange looking device, but it works wonderfully well. Both 300 and 1200 bps are supported, standard RS232 interface is used, and it's an intelligent modem which allows you to set all sorts of things including a real time of day clock!

I've used it with Teleterm80, and so far it is working nicely. AT (shifted A and shifted T) gets the modem's attention, this followed by other letters in caps let you change things like delay between dialing digits, how many rings to wait for, what character to use for delete, day and time of day, echo, duplex, carrier on, off, detect, ignore loss, dial in answer mode, dial in originate mode (reversals allowed), when to have the sound on or off, self test, and so on and so forth. The modem is Hayes compatible, and with the exception of front panel LEDs, it's behaving like a Hayes modem.

The LEDs aren't there, no great loss really. The modem looks a bit awkward. It can plug directly into a wall or an extension cord via a swiveling cord base. It's tiny, the footprint is aout 3-1/4" x 6-3/4" (not counting the RS232 cable's housing, of course). It has two phone jacks, one for the base line, and one for the telephone, sould you wish to attach one. And, the nicest feature of all, when a phone is attached, then when the modem is in the process of transmitting data, it will not garble that data when someone picks up a receiver of the phone.

Instructions are in a nicely bound booklet and are very well written. There is a 'quick use summary' up front, followed by details of all the features you may wish to know about. At the end are instructions about how to use the modem with 3 specific terminal programs. Sorry, you won't find your favorites there (Bee-Line, Bterm, Teleterm...). Instead, you will find two IBM and one Apple reference. Oh, heck, they don't bite really, but considering that the modem is being sold with an interface for the C64, I have a hunch Commodore users won't be too thrilled about that ommission.

But don't let the lack of diplomacy of manual-writers detract you from seriously considering this modem. I think it's a great buy for a four-fold increase in transmission speed, and you know Protecto's (sorry, COMPUTER DIRECT) reputation in standing behind their products. I don't think you can go wrong.

******* KEYBOARD DECODING *******

Machine code people sometimes want to decode the keyboard their own way. It's a bad practice, in that the code that runs on your B has no chance of running on, say, a C64. But if you just can't use the standard GETIN routine ($FFE4), which does everyting so nicely, then you can't.

Here is just one hint about the B computers. Whether you set the lines at $DF00 or $DF01 yourself, or let the system routines do it, when you finally read $DF02, make sure you mask off highest two bits (A AND #$2F). They don't really belong to the keyboard; instead they have to do with what sort of machine you have (128, 256, hi profile, low profile, things related to the CRT controller).

Against my better judgment (and I hope Jim Butterfield and Howard Harrison aren't seeing this) let's play with the five keys on the right side of the numeric pad, (STOP / - +

ENTER). Read $DF02 register. Mask of 3 high bits (A AND #$1F) and play with the rest. STOP key is always important - when pushed, bit 0 becomes 0. The system does LSR, if the carry is clear, STOP was pushed, then they do ROL, and you can take over from there. Slash key is in bit 1, - in bit 2, + in bit 3 and ENTER in bit 4. The values are '1' when not pushed, '0' when pushed, and combinations are permitted, so ENTER and / pushed together set both bits 1 and 4.

Have fun, but don't use this sort of thing unless you really must. It is prone to errors, it makes for incompatible code. I know, I got burned when I wrote some code for the B128 and it collapsed in B256 all because of 1 bit!! Still, I felt like sharing this insanity with you, so this is it.

history.nol

by: Liz Deal
Abridged by CBUG for general information only.
full text/programs in library.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      command history v1.1 - liz deal - 2/21/88        %
% for pal or buddy. if buddy - edit '.if' statements.  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--> the idea to write this program comes from my work
% with a hewlett-packard editor program, and amiga's
% shell program. those computers need command history,
% since they do not have screen editors. but a screen
% editor isn't of much help when the commands vanish,
% hence this suite of routines for you to play with.
% the best of both worlds is at your disposal!
%--> program captures things you type on the keyboard
% and prints the command history on request. coded for
% the b128/256 & c64, conversions to other cbm machines
% are trivial, so long as you have a link to the crunch
% token routine. read about stkv below.
%--> recall command (ju3) can be sent to a file to, for
% instance, create an execute file. it can also list
% to the printer. just use the standard cmd-method.
% without cmd, lines print on the screen. once
% redisplayed, they can be edited and/or reentered.
%--> sys-call to jul connects this code, ju2-disconnects
% ju3-recalls. ju3 can be done in either mode.
%--> in order to make this code compatible with all cbm
% machines, i haven't coded an 'instant key' listing
% routine. i leave it to your usual devices to hook
% routine ju3 to your system. i just use f-keys.
%--> feel free to play with memory allocations for the
% size of history buffer.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

◆◆◆◆◆◆◆◆◆

CBUG letter 1

Dear Norm:

Before starting with a series of questions, I would like to compliment the CBUG organization and the contributers to CBUG for an outstanding job. Without their help programming the beast would be most impossible. Thanks.

Last months article by Anthony Goceliak on renaming 'special files on the B' was certainly interesting. Upon occasion I have renamed files by accident and achieved the same results. Thanks for telling me the right way to rename special files. I would also like to thank Anthony for his program for loading and saving programs on tape, now I can transport programs from my other computers without much trouble.

The JCL Software WORKSHOP is certainly a work of art. The extended basic statments make writing a program less of a chore and more of an adventure. I wish to thank Warren Kernaghan for the software documentation, and hope others will write articles on using this tool.

LOOKING for help!

I do a lot of work in combustion of solids (coal and wood). To monitor the combustion process it is becoming increasingly necessary to employ the assistance of a computer. The computer would be used to gather,and store data for future use and possibly some control. I feel the B128 could be an excellent computer for this purpose because of it's interfacing capabilities. The B would be used to gather both on/off and analog signals (0-5 volts for example) and store the results in either sequential or relative files.

Bing Hart told us how he interfaced analog signals (CBUG vol 7 pp.45) via the rs232 port and an analog to digital converter, this system appeared to meet his requirements. However, I am interested in a more direct interface through an existing expansion port on the B which would allow almost infinite (Mr. Anderson says a million bytes) input capability.

Since I don't have the necessary skills to do the interfacing myself, I am looking for assistance from within the B128 users group. Who knows this may open up a whole new use for our baby. Please contact me at the address below if we can work togeather on this project.

Paul Woessner

◆◆◆◆◆◆◆◆◆

CBUG letter 2

Dear Norm:

In a brief contact with you last year we discussed computer programs which will be available through CBUG. I am interested in programming the beast in other languages to speed up specific scientific calculations. I am still interested in programming in Pascal, Fortran, or other languages as they become available. It would even help if I could efficiently compile the programs developed using JCL software Workshop, (which is an excellent program). I have done some programming in assembly language, but it is tedious and formatting is very difficult. Any help in finding new faster languages for the B128 would be appreciated.

The CBUG organization is certainly growing thanks to tirelsss work and dedication. Since there is a large contingent of B users in the midwest U.S. it would be nice to hold a fest on a weekend and invite speakers on a variety of subjects. I realize CBUG East and West hold meetings on a regular basis, and I should make a point of attending to meet the members. However, through a special meeting on the weekend people from distant places, like Indianapolis, could attend. At this point I don't know how I could help in setting up this 'Super Meeting'. Please let me know if this type of meeting is being considered for this year and if I may be of assistance in setting up the program.

Paul Woessner
4404 Broadway
Indianapolis IN 46205
317-283-1716 (after 6pm)

<<Do I hear anyone wanting to step forward and head up such a meeting effort?>>

◆◆◆◆◆◆◆◆◆

So Spake the Great Unwashed

by: Wesley H. Faust

If you can spare a few lines I would like to personally address this to the CBUG staff and all the others of our many mentors who give so unselfishly of their patience, time

and skill. I won't take up precious space to quote names for each of you know full well the magnitude of your contributions while so many of us stumble along behind, blind in our ignorance of the larger concept you are en-deavoring to share with us, too busy pecking at the gems of your genius that in the narrowness of our understanding, satiate our simple needs.

You see, I am too stupid to become adept at programming. In fact, when I can get SS loaded and ready, it seems sorrowful that no great fanfare of trumpets sound and the world treated to something from Aida. My prompt sheet for SS is the whole Quick Reference, enplasticated and so heavily annotated even I have trouble reading it. I just don't understand it, the program, the machine - you name it, I don't understand it.

It is very difficult to explain, too, but I DO recall my reaction to the lack of understanding in others, way back in the salad days of my youth, when entrusted with the command of one of the largest ocean going vessels, a sad truth was thrust upon me. Inside their little pointed heads, all men are not created equal. In fact, they vary between the ridiculous and the sub-lime and up there around 'sublime' it is so lonely you couldn't find enough people for a two-handed game of poker.

I used to get quite angry and impatient with my men for no matter how simple my orders, when I checked back, things always seemed to be drifting more or less slowly to either port or starboard. While striving toward having things done the BEST way, I found it was most fortunate to get them done at all! Today in computerese it is called "refresh" but it can be directly related to human minds. Some of us have brains so volatile we have to be directed which way to the rest room - time after time after time. You just have to be patient with us, unless all you geniuses want to blackball us out of CBUG, grab it up and run off by yourselves, out of range of our ignorance.

It is sort of painful to state that we are stupid, no matter how many years we've been living with the cold hard fact. We have blanks in our memory. The address is there, and you can write to it, but the data will just disappear because there is no memory.

Let me explain my own sorrowful experience along these lines. I could always read because I lived next door to a little girl who would be three years ahead of me in school and when I went into the first grade, I already knew the third grade reading text - almost by memory. Because I've always been able to draw well, I could print like a typewriter even if I couldn't write! With special dispensation, I finished all the required reading for high school while in the sixth grade - Quo Vadis, Les Miserables, etc., eating Hugo - even Chaucer - like a hungry little mouse. Always placing in the essay contests, I had the world by the tail on a downhill drag!

The momentum I carried into high school had me sliding along on my face because my English grades were negative! My beloved teacher - fresh out of LSU - shed enough tears over me after school every day to wash my grubby little paws a dozen times over - all to no avail. After two or three hours of intense tutoring, I might be able pick out the verb, maybe even differentiate between a noun and pronoun but next day - it was all gone! Only the Principal's logic of "there is no way you can flunk a student in English if he placed second in the State Essay Contest last year!" saved me from ignominy! It didn't bother me at all to be considered an idiot savant of the English language because I could relax in class and listen to those darkly mysterious words bandied about, 'dangling participle' 'split infinitive' 'the subjunctive whatever'. How I envied my classmates, so adept in parsing sentencing and all that good stuff - of which I had not the faintest inkling - while it only made my head hurt. I had to be content with writing 500 words a week and standing up to speak for 3 minutes on whatever the teacher decided my subject to be! It was a piece of cake but - to this day I don't really understand what an 'adjective' is!

I merely try to explain the foibles of "intelligence", Norman, if you can be so generous as to assume I possess even the rudiments. Please, Angel and Liz and Warren and all you other many, many wonderful, generous people to whom my grateful, admiring love goes out, be patient with us. We are so infinitely less talented than you and in our simple hearts you are GIANTS, beings gifted far beyond my meagre comprehension! In our frustration perhaps we scream out that you must simplify the infinite perhaps like we would chide Van Cliburn for being unable to show us how to master "Polonaise" when we can't even do 'chopsticks'! My ignorance is so abyssal I am ashamed to admit that, even as I write this, when I use INS or DEL the text gets squirrelly and the way it prints out convinces me that this machine and I are not operating on the same frequency!

It is most alarming but I am not so stupid as to think I can do anything about it because way down here in the wilds of Texas I am all alone and too, embarrassed to bother comparative strangers on the phone! People are so nice I know they can be depended upon to help you out when they see you drowning but being forced to admit you jumped in without knowing how to swim is a little much! I think perhaps God made a tragic mistake and put a feminine heart inside me since babies, and other beautiful things, tend to make me weepy and I'd rather DIE than admit I hadn't done something RIGHT! I think the Computer Business would have been quite proud to stagger along without having made my aquaintance!

I try, truly I do but sometimes I think I would make much more progress if my endeavors drifted in a more gentle direction, perhaps something on the order of "Home Assembly of Super Novas, from scratch". I lap up every word of ESCAPE because each that falls within my understanding is laid out so beautifully clear and available. There is much that sails over my head but I just consider that is probably you geniuses communicating in machine language and I don't worry about it. You are so obviously trying to bootstrap us lesser mortals up to join the exalted company but speaking for myself you don't seem to understand what poor material you are dealing with! We are just simple folk trying to tiptoe through this vale of tears unscathed, anxiously seeking a culprit to blame for our lack of talent (read - laziness!)

Angel, Tony, Liz, all others of our group, I hope you find a fame even greater than that which burns so brightly in my heart. Recognition is such a nebulous thing. One measure might be that you take all the people of the earth and line them along the path to the moon. There will be over five ranks of human beings stretching over the average 240,000 miles and if you have the privilege (?) to consider that because of one or more of your accomplishments, you can be thought distinctly unique among that mass of humanity, the honor, along with a dollar, will entitle you to a cup of coffee in many of the best places. And yet, there is no measure, in money, of the lift to your heart when a complete stranger rushes up to grab your hand and explain that he had heard about you around on the other side of the Earth and had been waiting a long time hoping to meet you!

The real reward is always, and only, in your heart. When you look up at the dark of the mountain and see the little light you set, way up above all the others, only you know the struggle it cost to put it there. The things you missed, the money it didn't make you, the years of your life it etched away, all the despairing times you looked back and screamed at God, "Stop pushing!" But somebody has to scout the point, lead the way for the lame, the halt and the inept, even though it doesn't pay any better than what they get for tagging along behind, leaving the bleeding to you. Tomorrow someone else will probably set his light above yours but then - was not that the whole purpose? We don't put them there to be envied, we are lighting the way.

To all you lovely people who give so unstintingly of your time, I can say only a heartfelt "thank you" because no

matter how cleverly you search for description, some of the things inside you simply won't come out in words. The world may never acknowledge you with fame and your reward may never be more than what you might feel when you look back down the trail and see us stumbling along behind you and come to realize that perhaps it was God who decided that you would lead us.

Wesley Faust
Texas City, Texas

<<Wesley, you are indeed a talented writer. More importantly for the benfit of our authors and critics alike, your tenor sets forth most poignantly the intent and purpose of CBUG, and surely all other users' groups; and from the many letters and calls many of us receive, the unsung gratitude felt towards the CBUG contributors. With this declaration, let us all move forward with increased vigor both to learn from and help all others in CBUG with even greater dedication.>> <<Minor editing and deletions have been applied to this letter.>>

OOPS, THERE IS A BUG IN BACKUP X-Y

by: Anthony Goceliak

Today, 01 Apr 88, I called Norman on an unrelated matter, and he advised me that a member who had tried to use my program "backup x-y" to backup between two sfd-1001's had not had much success. Bearing the date in mind, I was skeptical, but the member's letter which Norman read was filled with sufficient detail that I immediately attempted to duplicate the failure and succeeeded (in failing!).

The trouble lies in my use of the fdc code involving only some source-drive sfds. Commodore installed two, count'em two indirect jumps in the FDC rom for their IEEE drives, the first of which is used to terminate a successfully executed fdc job, and a second, which on the 8050 is capable of handling both successful AND unsuccessful job terminations. Unfortunately, on some sfd's the second, while superbly suited for failure, is not so well poised for success. The gory details are such that only fdc code operated from a particular buffer ($0500 - fdc address) would be handled correctly if the indirect jump at $fc02 were taken with a job which was done right. The reason is that the code in the sfd's rom at $feef says jmp $fd38 instead of jmp $fd36 which is where the 8050 goes. The difference is crucial, since index .y is set properly to #03 at $fd36 in both roms, and thereby the soon to be executed sta ($19),y places a 'core wars data bomb' instead of setting up for the next sector.

OK, tain't pretty, but exactly as I have always maintained, making your code work by "simplifying life and using pre-existing rom routines" is always the hardest. There is a one-line solution to this which will allow correct operation, listed here.

```
145 bank15:poke 2042,141:poke 2043,5:poke 2044,4:poke
2045,108:poke 2046,0:poke 2047,252
```

If your backups ALWAYS bomb, add this line to the program titled "backup unit x-y" (where there is really an arrow pointing from y to x, but the character is unavailable to the escape typesetters.)

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

<<Any member who would prefer to receive a replacement disk only needs to send CBUG a note and a identifyable portion of the label from the defective disk already received. There is no charge -- just be sure your address is legible.>>

BASIC 4.0+ Tutorial (Abridged)

FORWARD:

by: Warren D. Swan

Some of you readers may be asking yourself "So why is Swan boring us with all of this BASIC garbage? I don't program in BASIC. I use Superscript (or Word Result), Superbase and Calc Result." Well here's a couple of reasons:

1. The B is a BASIC virtual machine. That means that it starts up using BASIC to communicate with the human user. Other machines use a general purpose operating system with a Command Line Interpreter (CLI) (or shell). To those machines, BASIC is just one of the languages loaded into the computer when you ask for it. On the B, BASIC takes over immediately when the system is turned on. The way other languages are used is that they are loaded into memory, and then BASIC is instructed to "turn control over" to them. Even the SHIFT RUN key simply hands a couple of commands to the BASIC interpreter.
   It is not that the B can't run other languages - only that BASIC is the native language of the B. If you can't do something in one of the languages available for the B, such as send commands to a device to change its configuration (behavior), you can always use a different language. What I'm saying is that BASIC is a good choice for such things since it is the native language of the B.

2. Those having to use Superbase for sophisticated purposes soon find out that Superbase's programming language is none other than BASIC itself, with a few extra instructions thrown in for database support and user interface. Furthermore, the books about Superbase assume that you already know BASIC - and that includes Superbase, The Book (which I highly recommend).

3. An understanding of any computer language can help the user to better anticipate the kind of information that the computer must know in order to perform its tasks. A better programmer is a better computer user because he knows how to use the computer to its fullest. Just a small increase in computer knowledge can greatly increase your ability to use it faster, more reliably and more efficiently.

I would be the last person to say that BASIC is the best language, or even a great language, but it does have one strong benefit for the B: Every B owner has it! So on with the show - chapter 4 of the tutorial.

4 STORING DATA IN DISK FILES and SUBROUTINES

4.1 Getting Ready to Read or Write a Disk File [Disk Files 1]

This tutorial does not explore all the nuances of maintaining disk files, which is gargantuan enough of a topic to require another course. We explore some of the fundamentals of reading and writing to files.

In Commodore BASIC the term "files" can refer to devices, such as printers or the RS-232 port, as well as to actual files stored on a diskette. We shall briefly explore both of these.

The unabridged tutorial disk explains why the DOPEN and OPEN statements are used to associate a file or device with a so-called "logical file number." In this abridgment we accept it for granted that we have to DOPEN or OPEN a file or device before reading from or writing to them. DOPEN is only used with disk files, whereas OPEN can be used with disk files or any device. For example, if we wish to DOPEN a disk file to read it, BASIC "looks for" the file when the DOPEN instruction is executed. If we wish to OPEN a file to the RS-232 port, information in the OPEN statement tells BASIC to set up the RS-232 port with the desired baud, parity, duplex, etc.

Commodore uses the IEEE-488 bus. The devices on this bus are "intelligent", since each contains its own internal "computer". When communicating with these devices, BASIC has to tell the bus which device it is communicating to. It uses a "physical device number" to identify the device (even when it is a device not connected to the IEEE bus, such as the screen).

To further confuse matters, it has to tell the device another number, called the SECONDARY ADDRESS, which is used to quickly tell the device what the communication is about. The secondary address is not needed when using the DOPEN command, since BASIC picks an appropriate number for us. However, when using the OPEN command, we have to tell BASIC which secondary address to use. This may require that we look this number up in the manual for the device we are communicating with. Let's start with the simplest command first.

4.1.1 DOPEN

The DOPEN instruction tells BASIC to open a file to the disk. Parameters to the DOPEN instruction tell BASIC which logical file number to give it (abbreviated to lf in this discussion). To open a new file to allow it to be written to we would use:

```
100 dopen#8,"&data",w
```

Here the lf is 8, the file name is &data, and the w parameter means that we are going to write, not read, the file. BASIC doesn't care what order we give this information to it in the DOPEN statement. We could just as well have used:

```
100 dopen w,#8,"&data
```

and the result would be the same. Remember that this just sets up the file to be written to; it doesn't actually write anything to the file yet. Later we would use a PRINT# instruction to send data to the file. For now we'll use a simple example:

```
350 print#8,"This is any old message."
```

Notice that the 8 in the PRINT# statement matches the 8 in the DOPEN statement. BASIC "knows" we mean to write this line of data to the "&data" file.

BASIC allows several files to be opened at the same time. Each may be for reading or writing, and each may even be to a different device. But each file that is opened has to be opened with a different lf number. A given lf number cannot be re-used for a new file until the old file is first CLOSEd (or DCLOSEd). We will learn about that later.

What if the file already exists and we want to blow away the old file and write it over with new data? We can use the same trick we learned with DSAVE in section 3.5; namely, we can place an @ sign at the beginning of the file name:

```
100 dopen "@&data",w,#8
```

The @ is not considered part of the file name to store in the directory. Instead, the disk unit knows that we only meant to open the file "&data" for writing, regardless of whether the file already exists. If it does, the old "&data" is wiped out.

Review section 3.5 to see what constitutes a valid disk file name. The same restrictions apply to DOPEN that apply to DSAVE.

If we had wanted to read data from an existing file, we would use:

```
100 dopen#3,"&names
```

The absence of the w parameter tells BASIC that we wish to read, not write, the file. The choice of 3 for the lf number was arbitrary. We would then use either the INPUT# or the GET# instruction with 3 for the lf in order to read data from the file.

What if the file didn't exist already? This is where we use the information presented in section 3.6 (Checking Statuses - DS, DS$, ST). We could include the statement:

```
110 if (ds=62) then print"Can't find &names file!": stop
```

This is a simplistic "solution". How did we know to check for if ds=62? We looked in the disk manual for the list of errors. If you've never read your disk manual, NOW is the time to do it. Come on; you can't put it off forever!

If not told otherwise, the DOPEN instruction uses drive 0 of device 8. It can be made to use other drives and devices using the d (drive) and u (unit) parameters. Here are some examples:

```
100 dopen d1,#3,"&junk",w: rem drive 1 of unit 8
200 dopen#4,u9,"&mrdata": rem drive 0 of unit 9
300 dopen"&tops",u10,w,d1: rem drive 1 of unit 10
```

The tutorial disk describes how to open a relative file.

4.1.2 APPEND

This instruction is very similar to DOPEN, except (1) it cannot be used with relative files, and (2) it always opens the file for writing instead of reading. The difference here is that it opens an existing file and positions the "file pointer" to the end of file. From there on it acts like as if we had opened the file for write. The result is that we append new data to an existing file. If the file didn't exist, we would get error number 62 here also.

As an example, assume we want to write more data to the &data file after it was closed. We would use:

```
1000 append"&data",#6
```

Again we would use a PRINT# statement to continue writing data:

```
1350 print#6,"Sock it some more data."
```

The use of a preceding @ sign is not only unnecessary, but undesirable. By definition the file specified to the APPEND instruction must already exist.

4.1.3 OPEN

It is hopeless to try and hide machine idiosyncrasies when discussing the OPEN instruction. It requires an understanding of the way in which BASIC communicates with other devices. Furthermore, parameters to the OPEN instruction must be specified in a certain order, unlike DOPEN or any of the other file related instructions added to BASIC 4.0 from BASIC 2.0.

In general an OPEN statement looks like this:

```
50 open lf,de,sa,"filename"
```

where lf is our old friend the logical file number; de is the device number; sa is that mysterious secondary address, and filename is the filename. All but the lf are optional, but each must be given if a following parameter is also going to be specified. If not specified, the device defaults to 1, the secondary address defaults to 0, and the file name defaults to nothing (null).

The list of device numbers is as follows:

0 = Keyboard	4 = Primary printer
1 = Cassette	5 to 7 = Other printers
2 = RS-232 port	8 = Primary disk unit
3 = Screen	9 to 15 = Other disk units

As you can see, the choice of device 1 as the default is a poor choice - a left over from the ancient Commodore machines.

The list of secondary addresses differs with each device. Read your disk and printer manual, please! For keyboard (0) and screen (3) the choice of sa and filename is insignificant - they are ignored. For printers (4 to 7) the filename should be left out. When using the RS-232 port (2) the first 2 characters of the filename determine the baud, parity, duplex, etc. and the name must be 4 characters long

- the latter 2 are ignored but must be given.

The tutorial disk gives the following examples at this point: (1) Communicating with the RS-232 line and (2) Communicating disk commands to a disk unit. Here we summarize a part of the second example, where we interrogate the disk unit for status (without using DS and DS$ with their ST bug mentioned in section 3.6). The secondary addresses for the disk unit are

0,1 = Have special meaning for DLOAD and DSAVE
2 to 14 = Normal disk access, must be unique. DOPEN chooses unique ones for us
15 = Command (output) and status (input) channel.

So we would use:

```
70 open 15,8,15
```

to open logical file number 15 to device 8 (disk unit) using secondary address 15 - the command/status channel. This is a bi-directional file, so we can use it to read the status of our previous disk command:

```
90 input#15,en$,em$,et$,es$,ed$
```

This reads the number, message, track, sector and drive number of the current error message. Review the DS$ variable for the meaning of these. (The ed$ should be left off if the disk unit uses DOS 2.5 or earlier. In fact, it is best to leave it off all the time.) If we then added line 100:

```
100 sd$ = en$+","+em$+","+et$+","+es$+","+ed$
```

then lines 90 and 100 amount to the same thing as sd$ = ds$, except that the INPUT# method is more reliable. The tutorial disk explains why this is so.

4.2 Using Variables or Expressions in Disk Commands [Disk Commands 3]

BASIC 4.0+ allows you to do this although NONE OF THE MANUALS EXPLAIN HOW. The only reason this author knows how to do so is because I remember reading a review of the 8032 several years back. In the review they discussed the additions in BASIC 4.0 and included examples of how to use variables in disk commands.

As mentioned before, BASIC does not care what order you give it the parameters needed for a (BASIC 4.0) disk command. The way it distinguishes them is by the first character of the parameter, as follows:

PREFIX:	MEANING:	VALID PARAMETER VALUES:
none	file name	valid filename string
#	logical file #	1 to 255
b	Bank	0 to 15
d	Drive number	0 or 1
i	disk ID	2 characters - CANNOT BE A VARIABLE OR EXPRESSION
l	Length of record	1 to 254
p	Physical address	0 to 65535
u	Unit number	8 to 15
w	Write file	none - just indicates write mode instead of read

To use an expression for any of the parameters, start each parameter with its prefix letter, then immediately follow that with a parenthesized expression. (The disk ID in a header command cannot be a variable or expression, thus i(id$) is not valid). The file name is funny because it doesn't really start with a prefix letter.

We could have written the above DOPEN examples as:

```
lf = 1: unit = 9: drive = 1: file$ = "data"
dopen(file$),#(lf),u(unit),d(drive),w
```

This is true of any of the (BASIC 4.0) disk instructions:

```
fd = 1: ff$ = "old data": td = 0: tf$ = "data": unit = 8
copy (ff$),d(fd) to (tf$),d(td),u(unit)
```

Again, we could have switched the order of (ff$) and d(fd), and so on. We could have even put the u(unit) before the 'to' instead of after it. Remember that the parameter value may actually be an entire expression, not just a variable. The tutorial disk gives an interesting example of this.

The above applies to all statements in BASIC 4.0 not found in BASIC 2.0. If you are using the old BASIC 2.0 statements (open, close, load, save, print#, input#, get#), you don't need the parenthesis. The reason is that the parameters given to these instructions must always be in the same order; there is no prefix letter as in the 4.0 instructions. Thus, these are all valid statements without any parenthesis:

```
open lf,device,sa,file$          close lf
print#lf                         load prog$,device
input#lf,a$                      save prog$,device
get#lf,b$
```

Notice in particular that the number sign is actually part of the INPUT# and PRINT# keywords. So you cannot use PRINT #15 or INPUT #1 because the space causes BASIC to think you meant a normal PRINT or INPUT and the #15 or #1 would cause a syntax error. Nor can you use ?#15 in place of PRINT#15. However, the number sign may be separated from the GET in a GET# instruction since there isn't a separate "GET#" keyword, only the keyword GET.

4.3 File/Device Input/Output: PRINT#, INPUT#, GET# [Disk Files 2]

We have been learning how to tell BASIC to set up access to a file, and at the same time associate that file with a logical file number, all with the appropriate DOPEN, APPEND or OPEN instruction. We couldn't help but see a little bit about how to get data in and out of files after they're opened. Now we look more closely at this process.

There are 2 ways to send data to a file. One way is to use the PRINT# instruction. The second way is very similar but is covered in a later chapter. It involves using the CMD instruction.

PRINT# is very similar to PRINT. The only exceptions are:

(1) There are no print fields in a file. Print fields are only valid when PRINTing to the screen. Thus, although the PRINT# statement allows them, you should not use commas as delimiters.

(2) BASIC does not even keep track of what "column" a file is currently on. Thus, although the PRINT# statement allows it, you should not use the tab() function. However, you may use the spc() function.

(3) If the logical file number used is from 1 to 127, a single carriage return is sent to the file after a PRINT# that is not terminated with a semicolon or comma. If the logical file number used is from 128 to 255, a line feed is sent along with the carriage return at the end of a PRINT# statement not terminated with semicolon or comma.

Here is an example of printing data to a file. We assume that the appropriate DOPEN or OPEN has already been performed:

```
1050 print#3,"Today's average temperature was: "av
```

The PRINT# must be typed in with no space between the PRINT and the #. Following the number comes the logical file number that was used in the DOPEN or OPEN statement. You may use spaces between the # and the logical file number if you wish. PRINT# follows the same rules as PRINT. Since there is no punctuation between the second quote and the AV, it works as if a semicolon was there - that is, it does not add in any spacing to the space after the was: and the space in the number represented by av (unless it was below zero today). The tutorial disk gives an example of using an expression for the logical file number.

The INPUT# instruction works just like the INPUT instruction, except that it accepts input from a file or

device. The differences between them are..

(1) The INPUT# instruction does not need, nor does it even allow, a prompt string as the INPUT instruction does (INPUT"What is your age";age).
(2) If the INPUT instruction is given nothing but a carriage return as the response, it leaves the variables with the values they had and continues on. If the INPUT# instruction encounters a blank line in the input file, it skips to the next non-blank line.
(3) If the end of the file is encountered when an INPUT# is trying to read data, it sets all the variables in the INPUT# statement list to zero (for numeric variables) or null ("" for string variables), and it sets the End-Of-File bit (value 64 decimal) in the ST variable.
(4) If the INPUT# statement encounters extra data on a line in a file it does not spit out the "extra ignored" message. It just silently ignores the extra.
(5) If an INPUT statement is given bad data (such as a string value when it needs a numeric value) it spits out the "redo" message and allows you to type in correct data. If an INPUT# statement encounters bad data in a file, it causes a BASIC "file data" error (error number 34) and, unless the TRAP instruction is used, it causes the program to halt.

One governing thought should be kept in mind when you are using a PRINT# to store data in a file that will later be read with an INPUT# statement: INPUT# READS DATA FROM A FILE IN THE SAME WAY THAT INPUT READS DATA TYPED FROM THE KEYBOARD. The tutorial disk gives 2 examples to illustrate this important principle - one using numbers, and one using strings.

The GET# instruction gets a single character for each of the variables in its variable list, just as the GET instruction does. Unlike the GET instruction, which doesn't always get a character because one may not have been typed, the GET# instruction almost always returns a character. GET# does not give a character to a variable if:
(1) the end of the file was hit first, in which case it sets the End-Of-File bit in the ST variable (value 64), or
(2) if the character encountered is an ASCII NUL [chr$(0)], GET# clears the variable rather than give it the value chr$(0).

Also, as with the GET instruction, the GET# instruction may be used to get single characters for numeric variables; however, this is rarely done since the only valid characters are those allowed in numbers (digits, +, -, ., e), and all others would cause an error.

4.4 DCLOSE, CLOSE [Disk Files 3]

These instructions tell BASIC that we are done using a file. The B only allows you to have 10 files open at one time. When you DCLOSE a file, you allow other files to be opened. In some cases, such as when sending data to disk, the DCLOSE does more than just allow you to use more files. It tells the disk unit to "finish" the file that you had opened to write data to. To close a file use:

```
990 dclose#(lf)
```

Notice that parenthesis must be used if an expression is used. This tells BASIC that you are done with the file whose logical file number is stored in variable LF. The DCLOSE instruction allows you to close many files at one time:

```
dclose
```

This closes ALL files on unit 8. To close all files on disk unit 9, use:

```
dclose u9
```

The DCLOSE instruction is usually used to close disk files that were DOPENed. The CLOSE instruction is used to close any file. If you want to use a variable or expression in a CLOSE instruction, you do not have to use parenthesis:

```
990 close lf
```

The CLOSE instruction requires a logical file number and can not be used to close multiple files at one time. It is usually used to close files that were opened with the OPEN instruction, although it too can also close files that were opened with the DOPEN instruction.

One point worth noting here is that if you open a file to the command/status channel and then close it, any other disk files on that unit are automatically closed as far as the disk unit is concerned, but not to BASIC's knowledge. This is a feature of the Commodore disk units, not the BASIC in the B. Any attempt to refer to the files that were opened on that unit will probably result in a disk error (check DS$). Thus: .

```
open 15,8,15: close 15
```

closes all the "channels" on disk unit 8. This is intentional. There are some obscure instances where you might want to free up all the disk's channels - such as after a "no channels" disk error.

* * *

There is much more that could be said about disk files, and about the other disk operations (RENAME, COPY, COLLECT, etc.). These are no small undertaking! There should be a course each on both of these areas.

* * *

4.5 GOSUB and RETURN [Program Flow 6]

Let's develop a program to sort a list of numbers read from a disk file. First we have to discuss one of the many ways to sort lists of numbers.

4.5.1 Simple Exchange Sort Technique

Let's assume that we have eight numbers stored in an array NUMS and that they are currently stored in random order as elements NUMS(1) to NUMS(8). The method used to sort these numbers is to make several passes through the numbers, moving the lowest, then the next lowest, and so on, toward the left - that is, toward the lower subscripts of NUMS.

Each pass through the numbers we use the next element from the left to compare with all the numbers to its right, exchanging the numbers if the left one is greater than the right one. If there are N numbers, we only need to do N-1 passes. The tutorial disk gives an extensive pictorial representation of an example of this processes.

4.5.2 Implementing the Simple Exchange Sort

There is one more thing we must realize before we get down to programming our solution to our original problem. We know how to compare two elements of an array:

```
30 if a(i) < a(j) then ....
```

but how do we switch them? The answer is to juggle them! When a juggler is juggling balls the reason he doesn't have to hold them all at one time is because some of them are IN THE AIR at any given time. We can apply this idea to our array elements as follows:

```
40 air = a(i): a(i) = a(j): a(j) = air
```

We stored the value of a(i) in a temporary place (the air) while a(i) was receiving a(j)'s value. Then we pulled a(i)'s old value out of the air to store it in a(j). Got it?

Now back to our original problem: sort a list of numbers read from a disk file. We will read the numbers from the disk until an end of file is encountered. Since we have to store the numbers in an array, and arrays must be DIMed before they are used, we have to pick a reasonably large number for the maximum subscript of our array and just limit

the sort to that many numbers. We use the variable T as the temporary variable for exchanging:

```
 10 rem xsort 1.0a  - reads numbers from a disk file and sorts them.
 100 max = 1000: dim nums(max): n = 0
 120 dopen#1,"&numbers": if (ds > 19) then print ds$: stop
 140 if (n>=max) then dclose#1: print"Too many numbers!": stop
 150 n = n+1
 160 input#1,nums(n)
 170 if (st and 64)=0 goto 140
 180 dclose#1
 190 rem  all n numbers read in.  print them out:
 200 for j=1 to n: print nums(j),: next: print
 210 if (n<2) then end: rem no need to sort 1 number!
 250 rem sort the numbers, printing out the list after each pass:
 300 for i=1 to n-1: rem i is the pass number and which element is comparing
 310 for j=i+1 to n: rem j is which element we are comparing against the ith
 320 if nums(i) > nums(j) then t = nums(i): nums(i) = nums(j): nums(j) = t
 330 next j
 340 rem print out numbers after pass i
 350 for j=1 to n: print nums(j),: next: print
 360 next i: rem next pass
 400 rem now skip some lines and print out the sorted array:
 450 for j=1 to n: print nums(j),: next: print
```

Notice that lines 200, 350 and 450 are identical. BASIC allows us to avoid writing sets of program statements over and over. This is done by using what are called SUBROUTINES. A subroutine is a set of statements that MAY be used several times in a program. We say that we CALL the subroutine when we use it. BASIC uses the GOSUB instruction to call a subroutine. The subroutine also needs one other thing to be complete. It has to know when it has reached the end of the list of statements so that it can go back to where it was called. The RETURN instruction signals the end of the subroutine and tells BASIC to go back to the statement that called the subroutine. When it goes back it picks up right after that GOSUB instruction.

Here is the above program with some of the comments stripped out (now that we understand it) and using a subroutine:

```
 10 rem xsort 1.0a  - reads numbers from a disk file and sorts them.
 100 max = 1000: dim nums(max): n = 0
 120 dopen#1,"&numbers": if (ds > 19) then print ds$: stop
 140 if (n>=max) then dclose#1: print"Too many numbers!": stop
 150 n = n+1
 160 input#1,nums(n)
 170 if (st and 64)=0 goto 140
 180 dclose#1
 200 gosub 1000: rem print the numbers
 210 if (n<2) then end
 250 rem sort the numbers, printing out the list after each pass:
 300 for i=1 to n-1
 310 for j=i+1 to n
 320 if nums(i) > nums(j) then t = nums(i): nums(i) = nums(j): nums(j) = t
 330 next j
 350 gosub 1000
 360 next i
 450 gosub 1000
 500 end
 1000 for j=1 to n: print nums(j),: next: print
 1020 return
```

When line 200 is executed, BASIC "remembers" where it was and jumps down to line 1000. When the RETURN is executed in line 1020, BASIC recalls where it was when this subroutine was called. After continuing on past the GOSUB in line 200, it also "forgets" that line 200 called the subroutine at line 1000 - so it doesn't get confused later; like at line 350, where it again "remembers" where it was, executes the subroutine at line 1000 and returns back to line 360. And so on.

Line 500 was added to keep the program from "falling into" the subroutine at line 1000. Were that to happen it would print the list of numbers an extra time (harmless), but then it would execute the RETURN statement in line 1020. When return is encountered and there was no pending GOSUB, BASIC gives up with a "return without gosub" error. It has no place to return to! Also, BASIC isn't smart enough to "recognize" that it is "falling into" a subroutine so that it can avoid it.

When we first discussed subroutines a couple of paragraphs ago, we said that they MAY be called several times in a program. They don't have to be called more than once. In fact, programmers often use subroutines merely to group statements together that are involved in some particular action. This can lead to much more readable and maintainable programs, even though they are slowed down a smidgeon by the overhead involved in subroutine calling and returning. As an example, let's reprogram the sort routine as a "main" program consisting of several subroutines:

```
 10 rem xsort 1.0a  - reads numbers from a disk file and sorts them.
 100 max = 1000: dim nums(max): n = 0
 120 gosub 500: rem read in numbers
 140 gosub 1000: rem print the numbers initially
 150 if (n<2) then end
 160 gosub 2000: rem sort the numbers
 180 gosub 1000: rem print the sorted numbers
 200 end
 490 -----
 500 dopen#1,"&numbers": if (ds > 19) then print ds$: stop
 510 if (n>=max) then dclose#1: print"Too many numbers!": stop
 520 n = n+1
 530 input#1,nums(n)
 540 if (st and 64)=0 goto 510
 550 dclose#1
 560 return
 990 -----
 1000 for j=1 to n: print nums(j),: next: print
 1020 return
 1980 -----
 1990 rem sort routine:
 2000 for i=1 to n-1
 2010 for j=i+1 to n
 2020 if nums(i) > nums(j) then t = nums(i): nums(i) = nums(j): nums(j) = t
 2030 next j
 2040 gosub 1000
 2050 next i
 2060 return
```

The ----- lines help delineate the subroutines. Although they would cause syntax errors if executed, they won't be executed.

Although reading this program requires jumping back and forth between main program and various subroutines, it is more logically laid out for resolving problems. For example, if an error occurred in reading in the numbers, you know that the problem HAS to be in the subroutine at line 500; if the numbers don't get sorted right, the problem HAS to be in the sort subroutine, and so on.

An important point made by this example is that subroutines can be "nested" - like FOR/NEXT loops. Notice that the sort subroutine calls the printing subroutine from line 2040. Subroutines can be nested many levels deep. BASIC "remembers" the place to RETURN to by storing this information on something called a STACK. Think of this as a pile of papers in which the last paper put on the pile is the first paper taken off the pile to be acted on. (In this analogy, the "return without gosub" error occurs when the pile of papers - the stack - is empty.)

When BASIC encounters the GOSUB 2000 in line 160, it stores the RETURN location on the stack. Then while executing the sort subroutine it encounters the GOSUB 1000 in line 2040 and stores the RETURN location on the stack above the previous stack entry. When the print subroutine

encounters the RETURN in line 1020, it recalls the most recent place to RETURN to from the top of the stack, thus returning to line 2050. It also takes this information off the top of the stack when it recalls it. The process of recalling the information off the top of the stack and removing it at the same time is called POPPING the stack. The inverse operation, that of storing something on the top of the stack, is called PUSHING. The sort subroutine calls the print subroutine N-1 times, each time PUSHING the RETURN location, and the RETURN statement (line 1020) POPS the RETURN location each time. When the sort routine is done, the RETURN at line 2060 POPS off the only remaining RETURN location - the location where the sort routine was originally called - and returns back to line 160 to the REM and on to line 180, etc.

One other interesting thing about GOSUB and RETURN is that when the RETURN is executed, any incomplete FOR/NEXT loops started in the subroutine are terminated anyway. FOR/NEXT uses the same stack that GOSUB/RETURN uses to remember which variable is the FOR variable and what the limiting value and step are. So BASIC "cleans up" the stack by popping off any FOR/NEXT information still pending above the RETURN location for the subroutine.

4.6 ON expression GOSUB line#, line#, line#, ...
[Program Flow 7]

This multiple-way decision making instruction is very useful, and should be very easy to understand since we have already learned the GOTO statement, the ON/GOTO statement, and the GOSUB/RETURN statement pair. ON/GOSUB is to GOSUB what ON/GOTO is to GOTO! Its syntax is the same as that of ON/GOTO except that you replace the GOTO with a GOSUB. Otherwise, it works like a GOSUB.

One of the many uses of the ON/GOSUB instruction is to select one of several subroutines, each of which implements a different "command" as selected by the user when he presses a single key. For example: assume that the user may enter one of A, S, D, or R to cause the program to do one of four things.

```
100 .... display menu ....
110 get c$: if (c$="") goto 110
120 com = instr("asdrASDR",c$): if (com<1) goto 110
130 if (com>4) then com = com-4
140 rem           a     d     s     r
150 on com gosub 1000, 2000, 3000, 4000
160 go to 100: rem get next command
  ...
```

Line 110 waits for a key to be pressed. Line 120 checks to see if the character is one of a, s, d, or r or their capital. If not (com is 0), it goes back to wait for a valid character. Line 130 makes sure com is from 1 to 4 - com is 1 for capital A as for lower case a, etc. Line 140 just shows which line is GOSUBed to for which command, and line 150 takes care of dispatching to one of the subroutines. Whichever one is executed, it RETURNs to line 160 so that the next command can be executed.

The next issue will continue with Chapter 5.

Any questions arising from this tutorial should be sent directly to the author, whose address is given below. Also, you may obtain disks containing the entire tutorial directly from the author. It comes in either a dot-matrix (4023, 4022, 2022, etc.) version, or a letter quality version (6400, etc.). Each version (1 disk) costs $11, or you can obtain both versions for $18 (both disks). Handling is included. Write to:

Warren D. Swan
1 N 114 Woods Avenue
Wheaton, IL 60188

B-128 SERIAL BUS SOFTWARE

by: Dennis Jarvis and James Springer

The B-128 Fast Bus is now available! With this software and Gary Anderson's hardware, namely his SERIAL BUS INTERFACE**!AND!** 24K RAM/ROM CARTRIDGE you will now have a full blown serial bus on your B-128 just as it is available on other Commodore computers. Along with the normal slow serial bus we also support 'Fast Bus' just as it is available on the C-128 computer.

With the B-128 Fast Bus you will have complete access to all serial bus devices such as the 1525, 1526 etc. printers as well as the 1541, 1571 etc. disk drives along with various serial interfaces for Centronics printers (CARDCO and XETEC to name a few.)

If you should be using the 157X or 158X disk drives you will be able to use the fast bus capabilities of these drives on your B-128. Currently the 157X and 158X series of disk drives are the only devices that support Fast Bus.

Just to give you an idea of the speed possible using Fast Bus, the following table shows the LOAD times for various CBM disk drives, comparisons were performed using a 100 block file:

	1541	1571	1581	4040	8050
LOAD	1 min 30 sec	6 sec	4 sec	14 sec	14 sec

As you can see, Fast Bus makes a great difference in file handling speed.

Let's take a moment to clarify the difference between the normal Serial Bus and Fast Bus. Unlike the IEEE-488 Bus which is used with the 4040, 8050 etc. where 1 byte of data is sent out at a time. The normal serial bus sends out the byte one bit at a time using the DATA line to carry the data and the CLK line to indicate when the data bit is valid and ready to be read off of the bus.

Fast Bus operates almost the same way except there is one major difference. Instead of relying on software to shift the data byte out and manipulating the CLK and DATA lines, all the software has to do is store the data byte into the Serial Data Register (SDR) of a CIA chip and the hardware takes over handshaking the byte out on the bus. This method uses the DATA line for one data bit and the SRQ line as the CLK line. The big advantage to this system is that the Operating System doesn't have to spend all that time to:

1) Place the data bit on the DATA line.
2) Toggle the CLK line to indicate data valid.
3) Looping through steps 1 and 2 eight times.

There are several other steps that are performed in shifting out data in the aforementioned manner but they have been omitted for simplicity.

Along with the additional speed available with the 157X disk drives, you also have the ability to Read and Write to various 5 1/4 Double Density MFM types of disks such as OSBORNE, KAYPRO II, TRS-80, CP/M 86, and MS-DOS disks.

These aren't the only formats that the 157X drives can read/write. There are well over 300 different MFM types of disks that the 157X can read.

Now let's move on to how the Serial Bus is implemented on the B-128 computer. After you have installed the serial bus interface device and a RAM expansion cartridge in the address range of $2000 - $2FFF (both are available from Anderson Communications Engineering), you are now ready to start using your B-128 Fast Bus.

Place the software disk into Device 8 Drive 0 and enter the following Basic command in Direct mode: BLOAD "B-128 FAST BUS.1" and press RETURN, then enter: BANK15:SYS 8192, and press RETURN.

At this point you will see the title screen come on to inform you that the Fast Bus is ready for action. Before going any further take a look at your Serial Bus interface as you need to locate a couple of things. First plug your serial bus devices into the serial interface and then locate the switch marked IEEE/SERIAL. This switch is the key to

the power and versatility of your new B-128 Fast Bus system. For now place the switch in the IEEE position then do a DIRECTORY on your disk drive. As you can see you're now obtaining the Directory from the IEEE disk drive. Now flip the switch to the SERIAL position and do another DIRECTORY. Now the Directory is coming from the serial disk drive! That is the procedure to follow when using SUPERBASE or SUPERSCRIPT II. The following guidelines should be used to load up and run programs which you will want to access the serial bus with:

1) BLOAD "B-128 FAST BUS.1"
2) BANK15:SYS 8192
3) Make sure the interface switch is in the IEEE position
4) Load and start your program the usual way

One capability that hasn't been mentioned yet is about the most powerful. When you have the interface switch in the IEEE position you can access BOTH IEEE and Serial Bus devices. Think of it, you can take a SEQ file for example onthe IEEE Bus, print the file out to your IEEE printer, your serial printer and make a backup copy of the file on your serial disk drive AT THE SAME TIME. And all of this from a simple BASIC program! It may sound complicated but it's really not and here's why. Whenever you have the interface switch in the IEEE position we use the device address to select which bus we are going to use. The device numbers are designated as follows:

Device Address
4 - 23 - IEEE Bus
24 - 30 - SERIAL Bus

Simply put device number 4 would be an IEEE printer for example and device number 24 would be a Serial printer.

On the program disk you will find some example programs for copying MFM and GCR disks from BASIC! Along with other examples as the file handling as mentioned above. Also on the disk you will find a text file called Memory Map which gives a detailed description of our memory usage of the B-128 as well as the entry points in our massive Jump table.

One thing to remember, we have attempted to make this software package as compatible as possible, but we are positive there are many programs that this software WILL NOT work with. Currently our version 1.0 is compatible with SUPERBASE and SUPERSCRIPT II, but there may be other versions out there.

B-128 SERIAL BUS HARDWARE

by: Gary L. Anderson

The Commodore Serial Bus can now be connected to the B-128! The hardware consists of an external interface that plugs onto the B-128 internal user port pin field via a ribbon cable. A Commodore Serial Bus cable with 6 pin DIN connector then plugs into the 6 pin DIN receptacle on the external interface to complete the connection. See my ad in this issue.

The design of the external interface was done with the goal of maximum implementation of serial bus features. The hardware will allow the B-128 to act as a serial bus controller, like the C-64 or C-128, and a listener/talker, like a printer or disk drive. It will operate in both slow bus and fast bus modes. Also a serial bus power-on reset and manual reset was included in the design. A three position function switch, accessable from the outside of the case, was included for user selection of manual reset (a momentary position) and two positions for selecting custom features determined by software. The interface comes in a rugged plastic case for protection. A BASIC test program listing is included to self test the interface.

I have heard a report that the high profile CBM-256 main circuit board is missing the ground traces going to the user connector pin field. With this being the case, the interface will not work in the high profile. One would have to go inside the case of the high profile and make the repairs.

As in all computer systems the hardware requires software to control it and presently a software package exists to make use of this hardware interface. Two software experts, Dennis Jarvis and James Springer, have teamed up to write a software package that activates the B-128 Serial Bus Interface as a serial bus controller and runs slow bus and fast bus with present available Commodore Serial Bus peripherals. The hardware and software are available together as a package from CBUG or available seperately, the software from CBUG and the hardware from Anderson Communications Engineering.

Gary L. Anderson
2560 Glass Rd. NE.
Cedar Rapids, Ia. 52402

WHATS NEXT

by: Dennis Jarvis & Jim Springer

Now that you have obtained your new FAST BUS software/hardware, what can you expect in the future? Plenty, some of our future projects include a program to transfer files between MS-DOS disks and COMMODORE GCR disks. This will enable those of you with MS-DOS type machines at work to bring your text file home and use your word processor such as SUPER SCRIPT, etc. to edit your MS-DOS text files then when done place them BACK onto your MS-DOS disk and load them back up correctly with your MS-DOS machine.

How about a program which will allow you to transfer files from CP/M-86 to other CP/M diskettes? A good MFM sector editor? These are just a few of the items we are looking at for future projects. TRANSACTOR which is a TECH NEWS journal for Commodore computers will now start supporting the B series of computers more, and more. We will be releasing this package thru TRANSACTOR, to enable the C-128 computer to keep up with the B!! This package will only include the code to enable them to use BASIC to access the disk drives much the same way the B can now access the disk drives. Once this has occured we will begin releasing thru the PUBLIC DOMAIN various small programs for the 1571 disk drive. We strongly recommend that you obtain a subscription to this journal as we've been seeing the B being talked about more and more by such well known authors as LIZ DEAL, GARY ANDERSON and others. Perhaps we can all make an effort to show the world that the B series is NOT just another HAS BEEN computer. With such packages as this one and others the B will quickly become a formidable computer. Now when you see a program for the 1571 disk drive such as Miklos Garamszeghy's 'Exploring The World Of MFM On the 1571 Disk Drive' (TRANSACTOR Volume 7, Issue 4) you can, with SOME effort on your part, convert these programs over to the B-128 computer. Keep a look out in the CBUG news letter as we will be, from time to time, sending in various programs and tips, etc.

Well that's it for now we hope you enjoy your new FAST BUS capability.

Enjoy,

Dennis J Jarvis
and
James D Springer

UNIVERSAL TRANSFER

<<Following is one of several user instruction files from the Jarvis/Springer FAST BUS disk being introduced in this issue. It is included in THE ESCAPE to show just how powerful these programs are.>>

by: Dennis Jarvis & Jim Springer

This program will give you the capability of transfering various files back and forth over the IEEE, and SERIAL bus with almost NO effort on the USER. With this program you can do all of the following options all at the same time;

1) Print a file to up to 2 printers
2) Copy a file to another disk drive
3) Print a file to the screen

You can perform any or all of the aforementioned items at the same time!

Let's go ahead and use this program to do all of aforementioned items, if you don't have one or more of thos devices don't worry about it.

Go ahead and load up the UNIV. TRANSFER program and run it. When it's up and running you will see the following statement, if our B-128 Fast Bus is in the computer:

IF THE DEVICE IS ON THE SERIAL BUS ADD 20 TO THE DEVICE NUMBER

IF YOU DO NOT WANT TO USE A SPECIFIC DEVICE ENTER 0

What this is telling you is that all SERIAL bus devices start off at 24 and end at 30. All IEEE devices start at 4 and end at 23. The following table will make it clearer to you.

IEEE PRINTERs	4 thru 7	(normally but these COULD be disk drives also)
IEEE DISK DRIVEs	8 thru 23	
SERIAL PRINTERs	24 thru 27	(normally but these COULD be disk drives also)
SERIAL DISK DRIVEs	28 thru 30	

NOTE: If you do not want to use a certain device such as the SERIAL BUS printer then just enter a device address of zero which will inform the program to bypass using that device.

First the program will ask you to enter the DEVICE number of the device to get the SOURCE file off of, the default is currently set up for a SERIAL BUS disk drive with a device address of 8. You must enter a numeric value between 4 and 30 inclusive. If the value is less than 4 or greater than 30 an error message will occur. Then it will ask you to enter the drive number to copy from which has a default drive number of 0. This will be repeated for the destination disk also remember: if you do not wish to use the destination device enter a zero.

Next you will be asked to enter the device and secondary addresses of your printers, after you have selected an active printer you will be asked to enter it's secondary address value. Normally this value will be zero, but certain printers give various options using the secondary address so we passed this option setting on to you the user. If in doubt enter 0.

Finally you will be asked if you wish to print the file out to the screen if you do enter y, if not enter n. That's all there is to it.

Next the computer will check to make sure that there were no mistakes such as entering a device number that does not exist and other such checks. After these are performed either an error message will be printed such as "YOU HAVE SELECTED BOTH PRINTERS AS HAVING THE SAME DEVICE ADDRESS" and other errors. If an error is not critical the program will continue and print out a summary of where each device is located and which bus it's on (IEEE, or SERIAL) and ask for your final aproval. If you answer yes then the program will ask if you need to see a DIRECTORY of the SOURCE drive. And if you do enter 'Y' for yes you will soon see a directory go passing by. If you see the filename you want just press the RUN/STOP key which will cause the DIRECTORY command to ABORT.

Now you will be asked to enter the SOURCE, and DESTINATION filenames. You need not worry about the file type the program will figure that one out for itself! At this point the program will check to make sure the SOURCE filename exists and if it does not it will restart the FILENAME entry section of the program to let you reenter it. If the filename exists on the source drive it will check to make sure it DOES NOT exist on the destination drive. If it already exists it will ask you if you wish to scratch it off of this disk. If you answer yes it will SCRATCH that file from the DESTINATION diskette. If you answer no it will restart the filename entry section of this program.

NOTE: The above references to the DESTINATION files only hold true if you selected a device number to COPY TO.

Finally the program starts to send out the information to the devices selected, if at any time you wish to 'DISABLE' a device because of such things as a PAPER JAM or for any other reason you can think of just turn that device OFF, and you will receive the following error message:

A BUS ERROR HAS OCCURED WITH DEVICE: XX ON THE ?? BUS WOULD YOU LIKE TO

1) IGNORE THE ERROR
2) LOG OFF THAT DEVICE

NOTE: XX is the unit device number
?? is the bus the error occured on (SERIAL or IEEE)

If you want to continue on as if nothing had happened then select option number 1. If you no longer wish to send data to that device select option number 2.

Well that's about it go ahead an use this program and play around with it, make any changes you wish.

FROM COL. WRIGHT

Dear Norm:

Here are the instruction and catalog files from disks 1 and 2. Tony has just provided me with a disk that will give me enough room on the "NATIVE" side to provide these to you in future releases.

I am currently working on a Z-80 emulator. It is working pretty well right now and runs most of the Z-80 stuff for the C-128 and a few others as well. This is what I plan on putting on Disk 003.

I also have a Small C compiler that I am working on. Right now, I am not having too much luck getting it to run. It may take a while, but, hopefully by the end of the summer, I should have it up and running.

I have also found the DRI (Digital Research Institute) GEOS or GIOS, (I can't remember which). I think this is the CP/M-86 version of the popular GEOS system. I will work it, but it may take a "long long" time as it is all in source code and over 4 disks worth of code. I am not to good with source code and we may have a problem since it seems we really don't have a good handle on how this machine processes CP/M.

I am also working on a "Tutorial" disk for the club. I have "accumulated" several books on CP/M over the last few months, unfortunately none on CP/M-86, and am experimenting with what works and what doesn't. It's slow, since I have to test everything out first then try to put it into words. Hopefully, I will have lessons 1 and 2 ready by July.

Tony has been the real "ACE in the Hole" for this movement. His work with the 8050 and his program to convert 8050 to CPM is the only reason that we have anything. Since we don't yet have a Term pgm that will run XMODEM protocole in CPM mode, I have been using TELETERM to download files, then using Tony's program to convert them to CPM. Time consuming, but it works. His new disk access speedup routines will significantly reduce the time we are spending working with CPM.

Which brings me to the "bottom line"! We can use help! Tony is spending his very valuable time working speedups and about everything else associated with the B machine. I, since I don't have either the hardware or software background, am working collecting programs from the many sources that are available to me. What we need, is someone to work software and someone else to research the "mechanics" of the B under CPM control. If you have anyone that may be interested in doing either of these, we may surprise you at what can be done with this "Old, Outdated" machine. After all, what other machine do you know of that can run CPM and MSDOS as well as Native CBM. I venture not many!

To make the B really powerfull, we need faster access, (Tony's working the 8050 end, but a faster "CLOCK" would improve things even more), better software, or at least capability to use MSDOS 2.0 and better, and lastly, a set of terminal programs that use a "standard protocole" like XMODEM. Given these capabilities, the "OLD B" will probably outperform some of the newer machines that are being marketed today. TIME and EFFORT, that's all we need.

Lt. Col John A. Wright
818 Juniper Dr.
Papillion, Ne 68046

◆◆◆◆◆◆◆◆

CALC RESULT — REVIEWED

by: Carter Pawlus

This review is going to be an attempt to aquaint those of you who are just "Beginners". In reading the "ESCAPE" it appears there may be a need for a more simple or basic approach to some of these programs. Well, I'm very simple and am going to walk you through Calc Result (up to a point) later, just as if you had opened the package and wanted to start out creating a Spread Sheet without having to read and fiddle for hours before starting. For the rest of you geniuses, the only reminders needed are, "When all else fails, READ THE INSTRUCTIONS!".

Calc Result is a BIG Spread Sheet highly rated against the competition and appears to be limited only by the imagination of the user. Some of the uses, as an example are, tax forms and their calculations, inventory with ordering and cost projections, and almost any business or sales application that requires some ease of number crunching. I've used it for real estate rental control and analysis among other uses. Later I'm going to walk you through a rental set up, step by step, that will familiarize you with the instructions and give you a base from which to create your particular program.

So how big is this BIG spread sheet? Within each curser there is a maximum of 36 characters available on 63 columns and 254 lines or rows on a page. Now that is a pretty big page! Then you can store 32 pages on a file disk along with preset tabulation and printout records for easy recall. Beyond that, you can get more disks or operate on a Hard Disk. You can also have multi-user set-ups and there is even a built-in commuication mode to tie in to your b/term. Calc Result will handle up to 512k of RAM (random access memory).

Some of the other features are: Columns and rows are designated as co-ordinates in which numbers, words, or formulas may be entered; Recordable tabulation function for speed of inputs; Split paging and windowing for up to 4 pages on the screen; Lots of help screens that can be shown or printed anytime; Editing functions for changing, insertions, and erasing; Protection of formulas; Normal mathematical functions with manual or automatic recalculation for resolving different solutions; Sorting of contents, alphebeticaly or numericly; Screen or printout graphics; Printouts of your work 3 ways, hardcopy, direct, or formated (recordable) any way you want.

Also Calc Result can be purchased along with Word Result which means that you can tie in a Word Processor along with your Spread Sheet. With both programs you will need a minimum of 256K RAM. But this has been covered before in "ESCAPE".

There has to be some negative about Calc Result and from my point of view the manual is it. There may be others but in answering help calls from around the country I havn't heard any. The program WORKS! As far as the manual is concerned I suppose if you have an MBA it's ok but for this novice starting out, it was a zoo. Which is probably why I've been asked to do this.

So let's get started.

1 -Insert the Calc Result Cartridge PROPERLY.
2 -System power on.
3 -Insert the original program disk in drive 0
4 -And (SHIFT) (RUN)

At this point I received a few help calls. The program didn't load up. And the problem was an improperly inserted Cartridge. These embarrassed geniuses went on to do fantastic things with Calc Result so just be sure that Cartridge is seated properly in the correct slot.

5 -As the prompts appear on the screen enter your (2 character) ID.

6 -Put a blank disk in drive 1 and hit a key. A copy is now being made.

7 -When OK flashes on the screen, remove the original disk from drive 0 and put away in a safe place.

8 -Remove the copy just made from drive 1 and insert in drive 0.

9 -(SHIFT) (RUN) and answer the ID as before. Calc Result is now ready to start working on but before we go any further we need a file disc.

10 -Insert a blank disc in drive 1
11 -(CTRL) (D) (N) Fill in file name as called for such as "sample" or whatever.

12 -(RETURN) (Y) (RETURN)
12A-If there is an error or no read out you can INITIALIZE by:

12B-(CTRL) (D) (I) and try steps 11 and 12 again
13-(C=) (F4) And your printer should print out a description of the FUNCTION KEYS:

F 1: Blank.....Cancels contents of the cell under the cursor
F 2: Edit.....Enters the Edit mode on the input line
F 3: Goto.....Moves cursor to desired cell
F 4: Print.....Prints a Hardcopy of the screen
F 5: Recalculate..Forces a recalculation of page
F 6: Formulas.....Displays formulas instead

of values in the cells
F 7: Sync.....Enables/Disables sync. scroll during split screens
F 8: Jump.....Alternates between split screens
F 9: Swap.....Alternates between Main and Extra pages
F10: Shift.....Alternates between LABEL and VALUE

Well, well, here you are already running the program and even have something on paper to show for it. I'm going to have you print out all the help screens at one time to stick up near your work area. It has proven quite handy when starting out. If you don't want a printout at this time just ignore the "F4" instruction and substitute "CTRL" to get you back to the spread sheet.

IMPORTANT NOTE: The "CTRL" key is the command key for Calc Result. It gets you into and out of the Calc Result commands and functions. So if you get into a problem with starting something with "CTRL" hit it again and it will bring you back to your spread sheet. Too many of these computor programs gladly tell you how to start something but if you get into trouble, how do you get out! So here is one that does work for you.

14-(RETURN) (CTRL) (C=) (F4) Gives us the SYSTEM COMMANDS:

C: Communication...Communication with other systems
D: Disk command....For disk communication or User Register
E: Edit command....For screen and printer
F: Format command..Individual cell
G: Global command..Global format and column width
L: Leave.....Title, split-screen and window
O: Order.....Of recalculation (row or column)
P: Page command....Page functions:e.g. add, delete, etc.
Q: Quit.....Quit program
R: Recalculate.....Automatic or manual
T: Tabulation.....To tabulate certain cells
-: .....Automatic repetition of characters at cell under curser

Note: Always use the QUIT command when leaving the program, and REMOVE DISKS before turning off the power. If you don't know why remember internal power surges during power on or off could destroy important information on your disks.

15-(RETURN) (CTRL) (D) (C=) (F4) Gives us the DISK COMMANDS:

B: Backup.....Drive 0 to drive 1
C: Catalog.....Directory of drive 1
D: DIF-file.....For saving and loading of DIF-files
E: Erase.....Scratches file on drive 1
I: Initialize.....Drives 0 and 1
L: Load.....File to work area
N: New.....Disk is formatted in drive 1
S: Save.....Workarea to drive 1
U: User register..Access among other things help screen language, printer type and paper format
V: Visacalc.....Load a VisiCalc-file

Visicalc is a registered trademark of VisiCorp.

16-(RETURN) (CTRL) (E) (C=) (F4) Gives us the EDIT COMMANDS:

A: Assort.....Sorts Rows and columns
C: Copy.....Data area to another area
D: Delete.....row or column
G: Graphics.....Access graphic display or current values
I: Insert.....Row or column
M: Move.....Data area to another area
P: Print.....Worksheet or user-defined format
R: Replicate.....Data area to other areas
S: Split.....Screen (horizontally or vertically)
T: Title.....Places a non-erasable title in the left columns or top rows
W: Window.....Insert window on screen

17-(RETURN) (CTRL) (F) (C=) (F4) Gives us the FORMAT COMMANDS:

G: Global.....Cell : Clears all local formats for the cell
Global : Clears all global formats for the page
M: Maximum.....Sets maximum precision display mode
I: Integer.....Sets integer display mode
$: .....Sets two decimal display mode
N: Number of decimals...Sets 0-7 dicimals display mode
L: Left.....Sets contents at left
R: Right.....Sets contents at right
*: .....Replace integer number with corresponding number of stars, always left adjusted

18-(RETURN) (CTRL) (G) (C=) (F4) Gives us the GLOBAL COMMANDS:

C: Column width....Sets global width in all columns of the screen part, except in protected title columnn
F: Format.....Sets given format in all cells
R: Recalculation...Recalculate several pages by moving the highest column in one page to the alpha-column in the next page

19-(RETURN) (CTRL) (P) (C=) (F4) Gives us the PAGE COMMANDS:

+: .....Add pages, reading values and formulas only
A: Add.....Pages, checking that labels and formulas match
C: Copy.....One page to another
D: Delete.....Page from work area
E: Erase.....Work area
G: Get.....Page from work area
N: Negate.....Changes signs (+ and -) in one page
P: Put.....2nd page from work area (to get extra memory)
R: Renumber.....Page
Z: Zero.....Zero all values on page

20-(RETURN)

You now have all your help screens and remember you can access these any time while working on your spread sheet by "CTRL" "desired sheet" "C=" and "CTRL" to get back.

At this point we should set up your User Register especially for the type of printer you may be using.

21-(CTRL) (D) (U)

Wow! You can run Calc Result in EIGHT different languages the eighth language being of your choice other than whats available.

22-(1) (RETURN) (RETURN)
23-(2) for the 4023 printer or (1) for the 8023P printer (RETURN)

24-(N) (RETURN) (4) (RETURN) (72) (RETURN)
25-(60) for the 4023 printer or (44) for the 8023P printer (RETURN)

26-(79) for 9" paper on either the 4023 or 8023P printer or (132) with 15" paper on the 8023P printer. If you have other printers you will adjust acordingly.

27-(1) (RETURN) (8) (RETURN) (N) (RETURN) (N) (RETURN)

28-(1) (RETURN) (0) (RETURN) (Y) (RETURN)

The next step is actually on the first page. When you have a page really full of numbers and you execute a math excercise I found that when you are in the manual mode your answer comes up rather quickly as compared to automatic calculation.

29-(CTRL) (R) (M) And it shows up in the upper right hand corner of the "Control Panel". It stays there until changed.

Next as all my entries will be in dollar and cents I will set the decimal to two places for the whole page. You can still change decimals for a particular application individually.

30-(CTRL) (G) (F) ($) Every number entered will now show up set to two decimal places. Notice the "Control Panel" shows your choice in the right corner and stays there until changed.

In setting up a 3-unit rental property I use the first column for "transaction" information which contains the most information and is the easiest way for printouts etc. Our headings will be in the LABEL format so always press the "SPACE" bar before starting.

31-(SPASE) and type "1331 N 4 St Transactions" You may use capital or small letters and if you make a typing mistake use the DEL key as usual. You will notice that not all the printing shows up in the curser because the column width is automaticaly set for 8 characters.

32-(RETURN) Now all the printing shows up on the "Control Panel". We will adjust column widths later with the "G" "C" command.

I'm going to define curser movements from here on as "RIGHT"=(Rt); "LEFT"=(Lt); "UP"=(Up); "DOWN"=(Dn); with "RETURN" defined as (RET) and typing entries will be in quotation marks.

33-(Dn) (SPACE) "1988" (RET)

34-(Up) (Rt) "MODE" (RET) I did that on purpose to show what happens when you forget the LABEL setup. To make a correction after filling in some information simply:

35-(F1) (Y) And you are ready to start over. If you have entered the wrong info in a cursor but are not done just hit "CTRL" twice and it will clear the cursor. Remember the "CTRL" gets you out of trouble.

36-(SPACE) (SPACE) "MODE" (RET). Here I just moved "MODE" in one space for easier alignment. This flexibility allows you to set up any kind of printing you desire.

37-(Rt) (SPACE) "DATE" (RET), (Rt) (SPACE) "IN"
38-(RET), (Rt) (SPACE) "OUT" (RET), (Rt) (SPACE) "BAL" (RET)

39-(Rt) (SPACE) "#1 IN" (RET), (Rt) (SPACE) "#1 OUT" (RET)

40-(Rt) (SPACE) "#1 BAL" (RET), (Rt) (SPACE) "#2 IN" (RET)

41-(Rt) (SPACE) "#2 OUT" (RET), (Rt) (SPACE) "#2 BAL" (RET)

42-(Rt) (SPACE) "#3 IN" (RET), (Rt) (SPACE) "#3 OUT" (RET)

43-(Rt) (SPACE) "#3 BAL" (RET), (Rt) (SPACE) "RENT" (RET)

44-(Rt) (SPACE) (SPACE) "AUTO &" (RET), (Dn) (SPACE) (SPACE) "TRAVEL" (RET)

45-(Up) (Rt) (SPACE) "CLEAN &" (RET), (Dn) (SPACE) "MAINT." (RET)

46-(Up) (Rt) (SPACE) "COMM." (RET), (Rt) (SPACE) "INSURANCE" (RET)

47-(Rt) (SPACE) (SPACE) "LEGAL" (RET), (Dn) (SPACE) (SPACE) "FEES" (RET)

48-(Up) (Rt) (SPACE) "MTG." (RET), (Dn) (SPACE) "INTEREST" (RET)

49-(Up) (Rt) (SPACE) "REPAIRS" (RET), (Rt) (SPACE) "SUPPLY" (RET)

50-(Rt) (SPACE) (SPACE) "TAXES" (RET), (Rt) (SPACE) "UTILITY" (RET)

There are four ways to move around this sheet: cursor movement, The "F3" "GOTO" command, "HOME" "HOME" brings you back to the A1 co-ordinate unless title protected, and "CTRL" "L" brings you back to A1 and cancels any titles or windows you have made.

51-(HOME) (HOME) (Dn) (Dn)

52-(CTRL) (-) (-) And you have just excercised the repetition command. After the first "-" you may use any character you wish for repetition.

53-(CTRL) (E) (R) (RET) (RET) (Rt) (RET) (Z3) (RET). And you have just put whatever you have repeated in the first cursor all the way across to underline your heading.

54-(CTRL) (D) (S) wait a moment and (RET) (RET) You have just saved page 1 on the file disk. It is now in three places: computor memory, the work disk in drive 0, and the file disk in drive 1.

55-(CTRL) (P) (R) (2) (RET) You have just renumbered page 1 page 2. Now why did I do that? Because you own more than one property

and this is a great short cut for using your prepared format with minor adjustments. It should be noted that when you changed the page number it also changed on the work disk so that page 1 is only on the file disk You could also use the "COPY" command.

The cursor should be at co-ordinate A1, if not move it there.

56-(F1) (Y) (SPACE) "1626 N 4 St Transactions"
57-(RET), (F3) (P1) (RET), (CTRL) (E) (I) (C), (CTRL) (E) (I) (C), (CTRL) (E) (I) (C). You have just inserted three blank columns ready for more information

58-(SPACE) "#4 IN" (RET), (Rt) (SPACE) "#4 OUT"
59-(RET), (Rt) (SPACE) "#4 BAL" (RET), (Lt) (Lt) (Lt) (Dn) (Dn)

60-(CTRL) (E) (R) (RET) (RET) (Rt) (RET) (R3) (RET). And you have just modified the page for another property.

61-(CTRL) (D) (S) wait a moment (RET) (RET). And you have two pages in the files.

62-(CTRL) (P) (G) (RET) wait a moment and (1)
63-(CTRL) (D) (L) (Dn) (Dn) (RET) (RET) (N). And you are back to page 1. It is important to have the same page number on the screen as what you are going to load from the disk file.

64-(CTRL) (G) (C) (9) You have excercised the column width command. More tricks about this later.

65-(F3) (A4) (RET), (CTRL) (E) (T) (H) You have now locked your heading in place horizontally. No matter what row your in now the heading will stay visable. To remove the Title protection use "CTRL" "L".

66-(SPACE) "1 -JOHN DOE-JAN RENT PAID" (RET), (Rt) (SPACE) "-CASH" (RET), (Rt) (SPACE)

67-"1/2/88" (RET), (Rt) "275" (RET), (Rt) (Rt)
68-(D4) (RET), (Rt) (D4) (RET), (Rt) (Rt) (D4)
69-(RET), (F3) (P4) (RET), (D4) (RET), (F3) (A5) (RET)

There is your first entry. You'll notice that "275" came out as 275.00. Also you're using co-ordiinates to enter numerical values. Within the co-ordiinates you can enter values, co-ordinates, or a mix of both like "3*D4".

70-(SPACE) " -JOHNSON MANAGEMENT CO.-JAN FEES" (RET), (Rt) (SPACE) "-JOHN"

71-(RET), (Rt) (SPACE) "1/5/88" (RET), (Rt) (Rt) "-34.5" (RET), (Rt) "E5+F4"

72-(RET), (Rt) (Rt) "E5/3" (RET), (Rt) "H5+I4" (RET), (Rt) (Rt) "H5" (RET)

73-(Rt) "H5" (RET), (Rt) (Rt) "H5" (RET), (Rt) "H5" (RET)

74-(F3) (S5) (RET), (E5) (RET), (F3) (A6) (RET)

In this entry you did a little division and moved the result to other locations.

75-(SPACE) "2 -LEGAL NOTICE" (RET), (Rt) (SPACE) "-JOHN" (RET)

76-(Rt) (SPACE) "1/5/88" (RET), (Rt) (Rt) "-5" (RET), (Rt) (E6+F5) (RET)

77-(F3) (K6) (RET), (E6) (RET), (Rt) (E6+L5) (RET), (F3) (U6) (RET), (E6)

78-(RET), (F3) (A7) (RET), (SPACE) "3 -CINDY SMITH-JAN RENT PD" (RET)

79-(Rt) (SPACE) "-JOHN" (RET), (Rt) (SPACE) "1/5/88" (RET)

80-(Rt) "325" (RET), (Rt) (Rt) (D7+F6) (RET), (F3) (M7) (RET), (D7) (RET)

81-(Rt) (Rt) (D7+O5) (RET), (Rt) (D7+P4) (RET), (F3) (A8) (RET)

82-(SPACE) " -GAS CO.-HEAT-01172333" (RET), (Rt) (SPACE) "-2411" (RET)

83-(Rt) (SPACE) "1/10/88" (RET), (Rt) (Rt) "-106.06" (RET)

84-(Rt) (E8+F7) (RET), (Rt) (Rt) (E8/3) (RET), (Rt) (H8+I5) (RET)

85-(Rt) (Rt) (H8) (RET), (Rt) (H8+L6) (RET), (Rt) (Rt) (H8) (RET)

86-(Rt) (H8+O7) (RET), (F3) (Z8) (RET), (E8) (RET), (F3) (A9) (RET)

87-(SPACE) "2 -MABEL JONES-JAN RENT PD" (RET), (Rt) (SPACE) "-JOHN" (RET)

88-(Rt) (SPACE) "1/12/88" (RET), (Rt) "300" (RET), (Rt) (Rt) (D9+F8) (RET)

89-(F3) (J9) (RET), (D9) (RET), (Rt) (Rt) (D9+L8) (RET), (F3) (P9) (RET)

90-(D9+P7) (RET), (F3) (A10) (RET)

A lot more entries could be made but enough is a enough. This may seem like a lot of instructions but it really gets easy after while.

91-(CTRL) (-) (=), (CTRL) (E) (R) (RET) (RET) (Rt) (RET) (Z10) (RET) I did that just to show you a different repetition.

92-(Dn) (SPACE) "TOTALS" (RET), (Rt) (Rt) (Rt) "SUM(D4:D9)" (RET)

Ain't that neet! That is the first example of some powerful Calc Result math. Chapter 3.21 gives you all the higher math capabilities of this program.

93-(Rt) "SUM(E4:E9)" (RET), (Rt) (D11+E11) (RET)

If the value at "F11" matches the value at "F9" your OK; if not you made an error somewhere. Just a nice little check.

Aside from having this on your screen and saved on disks how about a printout of your work? You already have used the "F4" command to print your help sheets. We will use it another way now.

The first way to print is using the "F4" command but first a little set-up.

94-(CTRL) (L), (CTRL) (G) (C) (36) (RET), (Rt) (CTRL) (E) (T) (V)

95-(CTRL) (G) (C) (9) (RET) (F4)

Here we created a protected 1st column and reset the other column widths. And you got a print-out of the co-ordinates too. To continue you would cursor down and "F4" again. Remember you get a printing of everything on the screen.

THE CBUG LIBRARY

Lots of superlative new library materials, but this page for other thoughts. Read the next 11 pages for the "goodie list!"

ERRORS: We've received several recent calls regarding Copy-All for the B128 as it appears on Liz Deal's Utilities #17a and Liz's B-Kit #60. The temporary fix was discussed in detail on page 7 of the Fall 1987 issue of THE ESCAPE. If when you try to load and run it, you get "error in 96, then the Copy All program MUST be run under Keytrix -- which in the latest version on the same disks also has a bug. If Keytrix crashes, the change is simple: In essence (working on a duplicate disk) load "tsk", list 180, change the characters near the right hand end of the line from "myb" to "tsk", then scratch "tsk" from the disk, then dsave "tsk".

Tony Goceliak also has an error to report -- discussed on page 5 of this issue which error has to do with the X <-- Y copy program.

A member asked about the difference in directories between the Library listing and the actual disk of CBUG #75 by John Berezinski. Answ. Often the library section directories are from a temporary disk or even a fabrication while the actual disk is in transit. Often there are differences. In the case of #75 we tried to clearly indicate that the Super Office experiment was not complete nor properly operational. One of the notable changes between printing and duplication was that "loader" became two programs, "soloader" and "sbloader" What that means I've not the faintest -- but that portion of the disk is for educational purposes only. The SuperOffice stripper that works is #74.

WHY A LIBRARY? Beyond the obvious, that a program may be valuable to many members as is is a superficial if not an impractical expectation. Rather, to glean bits and pieces, concepts, methods is the higher usage, the very heart of the educational process. There are few books in our small world so we must learn from each other. To study programs such as the dozens within Super Teacher by Jon Whatley will assist members in developing or adapting for their own needs without having to re-invent the wheel. Even utility programs which are of the most general usage have important concepts to teach. Virtually every disk teaches something!

There are only a few things The Library can do for you by itself, but with its help there is a great deal you can do for yourself -- then in turn for others who depend on CBUG for support. Take a second and third look at the bricks which make up, or seemingly must comprise, the library materials. It's a virtual quary of useful objects which can be built into almost anything you can imagine. A repository of highly refined raw material for you to mold into useful tools.

At the risk of abusing the English language, I'm inclined to think in terms of "Constructive Plagerism." Someone once told me that the U.S. Patent Code suggests that the prime purpose of patents is to foster development and to that end encourages disclosure of new ideas so others may learn from them and improve upon them. In this context, the use of building blocks is the basic learning process -- ideas to be assembled in infinite combinations.

There are some common sense considerations. If borrowing even a very small brick here or there, always give credit to sources/authors when providing your work product to others. Let us not engage in the real definition of Plagerism even when it is NOT infringement or otherwise illegal.

THE CBUG LIBRARY is a resource for budding programmer and expert alike. Try to view it for the useful components, not just the finished programs provided. Help yourself, and it WILL help you.

Several comments have been made as to disk labeling. Unfortunately I've little time to make many of the needed determinations, plus label size is sort of cast in atleast sandstone. We do try to indicate when a disk is a Superbase application which ought advise by implication that Superbase (or SuperOffice)MUST be loaded first. Many authors have not learned to set up automatic loaders so the user is confronted with the demand for a database name. Easy to do. Before starting look at the directory, and any files with their names in ALL CAPS are likely to be database names -- but they type in on the screen in lower case!. Help files are always preceeded by an "h", and program files end ".p". Even if a disk is not marked as Superbase, these patterns are dead give aways.

We receive some complaints about lack of instructions. Though this is all too often the case -- either none, inadequate, too complex, etc. From the very beginning of CBUG we have asked that instructions be on disk and whenever possible readable in Superscript. Why? (In part because only one member does not have Superscript.) SS files can be read by many programs and simple readers in library. Universal literacy, eh! It saves CBUG the huge expense of printing and constantly refining instructions, not to mention postage. But the most important reason is that to search an instruction document for a reference, one needs only use the Search function in Superscript!

Then to, some library disks are equiped with SHIFT/RUN menu's. A careful perusal of the directory will give you some hints in that direction. Usually the instruction files and menu files are conspicuously labeled. You can do no harm by loading anything on the disk. The worst thing which can happen is the machine crashing without keyboard control. Just shut it off and start up again. Generaly speaking program files are loaded from basic, and sequential files are loaded into a word processor, etc. There are exceptions of course: Machine language programs are automatically loaded by a "loader" program -- usually a 1 or 2 block program with the file name actually desired. Loading of machine language programs in basic will always crash. The .p files in Superbase can only be used in Superbase. Programs can not be loaded into Superscript or sequential file readers.

A note of thanks to our contributors and authors. The inclusion of annotated directories is much improved. Everyone knows how much effort it is and it is being greatfully received. Informative and supportive articles are always welcome as well.

By popular request I've inlarged the print size of the library to that of the articles. Thus, when practical, the directories have been forced into a 60 column format -- i.e. CR in column 60 prior to our final formatting. We do the editing first in normal 80 column with a manual 60 column directory, then reformat in a 152 column wide screen. If you try it, caution: watch out for the munchkin characters when doing massive range moves. Always check that the end of text is where you think it is (Function 14) and that the right hand margins are clean. This issue I've shrunk several of the directories back to 60 column, while others were left in wider formats. Use your best judgement.

There is nothing I can possibly add to Mr. Goceliak's detailed anotated directory and polished articles.

```
6    "menu program"      prg  shift/runnable way to execute disk's files in b or drive
8    "menu pgm.ins"      seq  instructions on 'how to' use the menu program
0    "----------------" del
2    "read instruction" prg  basic pgm called if needed to quick-read any instructions
0    "----------------" del
5    "dumper demo"       prg  basic program to demonstrate some of screen dump's tricks
35   "dump screen.ins"   seq  SS II file detailing how to use dumps in your programs
1    "screen dump 1792" prg  m/1 code. version for those with no added bank 15 memory
1    "screen dump 6b00" prg  m/1 code. version for those with added bank 15 memory
8    "test screen"       prg  a demonstration screen with "the works", dumped by demo
0    "----------------" del
19   "autolink.ins"      seq  SS II file telling how to use the files below
15   "autolink seq.gen" prg  basic programs generate the linking files without 2 drives
13   "undolink seq.gen" prg  (If you have only sfd's, use this instead of 'copydltod0')
3    "&autolkd0seq.imm" usr  allows reversible, fast disk-wide SS II search&replace
2    "&undo autolk.imm" usr  this file reverses the rename and link performed by &auto*
0    "----------------" del
19   "4023 'bug'.ins"    seq  bugs are small.  I needed a microscope to find these
3    "4023 bug demo"     prg  basic pgm.  by running this you should see 'em without aid
0    "----------------" del
13   "dir reorder.ins"   seq  'documentation' for self-documenting pgm below
21   "dir reorder"       prg  shuffle filenames in directory at will with this pgm
0    "----------------" del
20   "ti$ vagueries"     seq  or why you shouldn't bet ti$ is right
3    "disk speed test"   prg  a non-destructive speed test. makes no disaster tracks!
3    "wp tester 80xx"    prg  real quick way to test your write-protect sensors
6    "test read disk"    prg  drv out of whack? disk out of whack? can they work together
0    "----------------" del
1    "4 col directory"   prg  just to see how much fits in a one-block program.
0    "----------------" del
1    "                " del
1    "cp/m-86 section"   del  the following three file sections pertain to cp/m-86
1    "                " del  ALL *.INS FILES ARE IN SUPERSCRIPT II, NATIVE MODE!
12   "native 2 cpm.ins" seq  how to convert pre-release 8,9,10 into cp/m-86 formats
16   "cpm disk convert" prg  basic program that converts 'em
2    "&cpm convert2"     usr  disk drive pgm automatically called by above
0    "----------------" del
16   "cbm_cpm dir.ins"   seq  how to see a cp/m-86 directory without needing to boot-up
12   "see cpm imm.gen"   prg  basic pgm generates the 'cp/m-86 dir viewer file'
0    "----------------" del
6    "8050 cp/m layout" seq  SS II instructions show what the chart below means
7    "cp/m 8050 layout" prg  prints out a one-sheet map of exactly where any file is
0    "----------------" del
1    "                " del  this ends the cp/m-86 section.
1    "                " del
9    "backup bug -mine" seq  otherwise known as I blew it, but here's how to fix it
1    "                " del
1    "                " del
0    "----------------" del
24   "8050disks on sfd" seq  SS II instruction file to make sfd/8250 bam compatible
1    "&pseudo 8250 bam" usr  disks from an 8050 drive using this &file
0    "----------------" prg
12   "blurb"             seq  thats this.
1722 blocks free
```

This disk may be upgraded as described in the .ins files in the cp/m-86 sec files are copyright 1988 by A. Goceliak.

FAST BUS CBUG #84 NEW RELEASE 2703 #13033

By: Dennis Jarvis and Jim Springer

This is the program portion of a joint project of Messrs. Anderson, Jarvis, and Springer. Mr. Anderson designed and is fabricating the electronics cartridges required, whilst Messrs. Jarvis and Springer have developed the extensive programming for some very significant advancements never before seen in any computer products.

NOTE: You must have two electronics products to use these programs. Mr. Anderson's 24K RAM cartridge which has been availale for nearly two years -- many members already have it. You will also need the new Fast Buss cartridge with connects internally to the B128 User's port. The cartridge(s) would best be ordered directly from Mr. Anderson. (See ads this issue) The software is available only thru CBUG. As a courtesy, CBUG will accept orders for either or both of the Anderson cartridges at the time of the software order. CBUG must, however, add a $5.00 handling charge to whatever is Mr. Anderson's current pricing. Due to critical parts shortages and market volitility, all prices are subject to change without notice. Please allow an extra 10 days (assuming stock availability) for any cartridges trans-shipped via CBUG.

```
0 "b128 fast bus v1" fb 2c
17   "b-128 fast bus.1" prg  - The object file for Fast Bus
1    "<basic programs>" prg
62   "disk copier"      prg  - Examples of using the various
                               routines built into fast bus
64   "make 2 sided"     prg  - Program to convert a single
                               sided diskette to a 1571 double
                               sided diskette
40   "universal trans"  prg  - Example of using the soft
                               device selection of the fast
                               bus software
2    "convert"          prg  - Program to read a SEQ into
                               memory that was LISTed to a
                               disk file
1    "<-instructions->" prg  - following files instructions
20   "&disk copier"     seq    for the example programs
11   "&universal trans" seq    /
23   "&make 2 sided"    seq    /
10   "&convert"         seq    /
1    "<--technical--->" prg
120  "memory map"       seq  - A detailed map of memory
                               usage of the B128 Fast Bus
                               throughout the B
190  "jump table"       seq  - Complete JuMP table listing
                               of the Fast Bus Kernal with
                               full descriptions on using
                               the routines
1    "<-----misc----->" prg
25   "b128 fast bus.ss" seq  - Article on the Fast Bus which
                               is in this issue of The ESCAPE
1521 blocks free
```

CBUG MISC. 0588 CBUG M82 NEW RELEASE #13014

A new collection of goodies. This one shows off quite an improvement by the various contributors. Each author's section is set apart by the dashed lines with the contributor's initials in the middle. Because some submissions have anotated directories and others do not, the listing below is not in the same order as the disk.

Several of the larger entries come with their own menu programs. You can either load the menu program (the first one in the section) then run it, OR, copy off that entire section of the disk (easiest using copy all) to a new disk. The effort is worth it as much more descriptive information files about the contributions are to be found in these manners.

Dr. James Condon brings us a medical application of a mathematical model: "On this disk is a little program I use to calculate the growth rate of lung tumors. It has a nice routine for telling you the day-of-the-week for any date. I've never seen it make a mistake, though I suppose it will someday."

Robert Chordas: "There are a couple of programs on this disk that you may not have in your library. The Julian Date program I wrote myself. The Julian calendar is handy when you have to figure preventive maintainance schedules and expiration dates. It was written using a superpet 9000 and a 4022 printer but it works just fine on the B-128 and a 4023 printer.

The hidden word program was in a book somewhere. Originaly all the words were in data statements. As in this puzzle, the data statements contained a list of all fifty states. However I did modify the program some. It now gives the option to type in the word list from the keyboard. I also centered the puzzle across the page and modified the word list output to make it a little more presentable. To determine the number of rows and the number of columns in the puzzle, you should count the number of letters in your word list and then add about 20%. For instance the fifty states contain about 412 letters. Type 'print sqr(412+(.2*412))' and your computer will tell you that 22 rows and 22 columns should be a large enough matrix for the puzzle. Of course a 21 by 23 or a 20 by 25 would probably do just as well. A puzzle this size takes close to an hour and a half to complete, so you must be patient.

I am not sure where the 'world quiz' program came from but I do think it is a very good program. The author's name pops up on the screen in the first frame. I do remember the program used jiffy timing. Outside of the timing, the only change I remember is to add a musical tone for every correct answer. This may not be a public domain program. If anyone knows of a copyright on this program (none visible) please notify CBUG.

And Ben's Clock. Ben got this clock from a superpet 9000 programmers manual, and modified it some. Ben was an

electronic tech. and worked on the Minuteman Missile. So, I suspect that is a minuteman missile that pops out on the screen and shoots down each minute as it expires. Of course the clock should be set before running this program."

Neil Cumfer: "A few useful programs and a lot of curiosities, some for the B128, some for the 8432 emulator. All programs believed to be in public domain. Those marked T are from the Transactor. All marked Basic are "load and run", though many operate through machine language poked into memory by data statements."

From: Jay Shepherd are 4 programs -- file names are self explainatory.

From: Armand Carrier. This looks like one of the more extensive collections done to date. Obviously lots of things are available for both the amature and the expert alike.

From Matt Goldstein is yet another upgrade of his famous Checkbook program.

Liz Deal's materials are discussed in her article in this issue.

Gerald North is a publisher of a periodical newsletter called Remnant Review. We present here the entire uncut articles he wrote which were previously published in abridged form in The Escape. The topic is electronic communications security -- i.e. telephone/computers taping and adulteration.

- - - - - - - - - - - - - - - - - - -

```
1 "CBUG MISC 0588  " cm 2c     3    "+clk1060"off1063  prg     2    "mirror"8432       prg     38   "cbug3/88"         seq
1    "----------jc------" seq     2    "+date1536"        prg     1    "<-b128amusemts->" prg     40   "history.no1"      seq
27   "doubling time"    prg     2    "+alarm1536"       prg     2    "zoundz"           prg     42   "history.pal"      prg
1    "-above +c+ 1988-" seq     8    "+clockcalscreen"  prg     3    "kaleidoscope"     prg     47   "history.lst"      seq
1    "---ok to give---" seq     11   "pettoascii"       prg     3    "crystal"          prg     2    "+hist.c64.6000"   prg
1    "-but not to sell" seq     8    "asciitopet"       prg     2    "tickertape"       prg     2    "+hist.b128.f6000" prg
1    "-the above prog-" seq     4    "system ram scan"  prg     1    "swords of doom"   prg     2    "+hist.b256.f6000" prg
5    "letter2Norm.ss"   seq     25   "softwareinvntory" prg     28   "cyndi"            prg     1    "--------gn------" seq
1    "--------rc------" seq     8    "great circle"     prg     2    "blurb.nc4"        seq     47   "north abridged"   seq
10   "julian date"      prg     3    "chromatic scale"  prg     8    "adirectory.nc4"   seq     64   "a"                seq
26   "hiden words"      prg     7    "binary tree demo" prg     1    "--------js------" seq     51   "b"                seq
49   "world quiz"       prg     1    "<-8432programs->" prg     102  "checkbook B128"   prg     48   "c"                seq
19   "bens clock"       prg     5    "basic aid.instr"  prg     2    "balance reader"   prg     70   "d"                seq
10   "this disk/ss"     seq     47   "c64basicaid.inst" prg     46   "checkbook B128in" seq     9    "commentary"       seq
1    "--------nc------" seq     18   "basic aid"8432    prg     24   "printer search"   prg     1    "----------------" seq
18   "contents.nc4"     prg     6    "sort demo1"8432   prg     1    "--------mg------" seq     32   "!!!--NOTICE--!!!" seq
17   "ss.contents.nc4"  seq     3    "sort demo2"8432   prg     6    "checkbook v3.2"   prg     347 blocks free
1    "<-b128programs->" prg     12   "sort pet"         prg     185  "%checkbook v3.2"  prg
6    "clock-calendar"   prg     3    "file ripper"8432  prg     1    "--------ld------" seq
```

```
                  ----------nc----------
18"contents.nc4"      prg Basic. info on these programs
17"ss.contents.nc4"   seq SuperScriptII "must" reading
1 "<-b128programs->   prg
6 "clock-calendar"    prg Basic. pokes date, time to screen
3 "+clk1060"off1063   prg    /updated each second         T
2 "+date1536"         prg    /allows alarm if desired     T
2 "+alarm1536"        prg    /'+' files loaded by Basic   T
8 "+clockcalscreen"   prg Binary screen dump (poke codes)
11"pettoascii"        prg Basic. converts seq files
8 "asciitopet"        prg Basic. converts seq files
4 "system ram scan"   prg Basic. updates 4 pages of system
                            variables to screen 60X/second
25"softwareinvntory"  prg Basic. keeps track of software
8 "great circle"      prg Basic. calculates distances
3 "chromatic scale"   prg Basic. plays all notes of scale T
7 "binary tree demo"  prg Basic. allows sorting quickly    T
1 "<-8432programs->   prg
5 "basic aid.instr"   prg Basic. loads "basic aid"8432
47"c64basicaid.inst"  prg Basic. "basic aid" = "keytrix"
18"basic aid"8432     prg Basic with appended machine code
6 "sort demo1"8432    prg Basic. /can sort 975 items in    T
3 "sort demo2"8432    prg Basic. /about 3 seconds          T
12"sort pet"          prg machine language for 8432 only!  T
3 "file ripper"8432   prg Basic. fast seq file reader      T
2 "mirror"8432        prg Basic. amusement for 8432        T
1 "<-b128amusemts->"  prg
2 "zoundz"            prg Basic various frequencies heard  T
```

dec-hex-bin: Converts DEC to HEX or BIN / HEX to DEC or BIN / BIN to HEX or DEC
def fn: Example using def fn with math
delay loop: Use insted of for...next loops. Compiling will not affect it. list/read REMS
disk demo: Tony's demo modified for 8050 disk drive.
disk spinner: I use this when I clean my disk drives (8050)
dynamic 1: Example of dynamic programming
dynamic formula: Enter your own formula at prompt.
feminist's: No explanation necessary - Just load & run
flash titles: Flash screen with program title, name, etc.
flashing screen: Example on how to flash screen from normal to reverse video and back
flashing screen2: Similar to 'flashing screen'
for next hint: Hint on use of for...next
get 2: Hint on using get (no ? and commas are allowed)
get 4: Another hint on using get instead of input
hi-lite demo: Another menu-type program
jump search: Binary search program.
keyboard matrix: Press a key and it will print on the screen. Not what you expect, but the name shown on the key. Such as tab, esc, shift, ctrl, return, enter, etc.
keys: Programs function keys.
mem jogger: Simple database

```
3 "kaleidoscope"     prg Basic interesting graphics      T
3 "crystal"          prg Basic screen patterns           T
2 "tickertape"       prg Basic good subroutine           T
1 "swords of doom"   prg Basic wreck your screen         T
28"cyndi"            prg Basic. Cyndi Lauper gig/no music
? "blurb.nc4"        seq
? "adirectory.nc4"   seq this file
```

----------ac----------

Menu program I wrote for CBUG. Just press shift/run and disk directory will load and appear on the screen in alphabetical order. Enter the number to the left of the program, press return and it will load and run automatically. Other instructions will be on the screen, plus information about the disk. Such as Disk Title and ID, how many blocks free, how many program files on the disk, option to load and display a directory of sequential files.

1 pg dir printer: This is Warren Kernaghan's 2 column directory, modified to print the directory on one page (up to 210 items) in 3 columns, up to 70 items per column by setting up the printer.

alpha slalom: Watch as letters of the alphabet slalom down the ski course (screen)

alpha slalom2: Same idea as above only more to it

alpharound: Prints the alphabet on the screen diagonally from lower left to upper right, then left to right across the center (horizontally) then diagonallytop left to lower right, then vertically in the center of the screen, top to bottom, then diagonally top right to lower left, then right to left in the center (horizontally), then lower right to upper left diagonally, then vertically in the center from the bottom to the top, them to the beginning again and changing, alternating from lower case to uppercase and from normal print to reverse print.

angles (sine of): You enter an angle and it computes the sine

asc finder: Finds ascii values

ascii: You enter an ascii value and it tells what key it is on the keyboard

ave: Simple adding machine that also keeps a running ave of the numbers entered

binary quiz: Shows number on the screen in binary and prompts for the answer in decimal

binary table: Shows decimal number and binary equivelant on screen

book names: Use this program to make a list of anything. It will search on one character so there is no need to remember the whole title, name, date whatever

bubble sort: What can I say. It's a bubble sort

bug: See the bug try to escape. When he stops running the bug eliminator gets'em

cbm 4023: Examples from the printer manual

cbugx: Similar to (flash titles)

centered prting: Example on how to center things on the screen

chr$ codes: Examples to be used with a printer

contents: The program you are currently looking at.

cursor demo: Demo on cursor control

cursor demo 2: Another demo on cursor control

cursor demo 3: Another demo on cursor control

cursor demo 4: Another demo on cursor control

data debugger: This will print the data item and the number of the line it was read from

msx+: Benchmark test from Compute! mag. It shows the B-128 as being the fastest computer (of those listed anyway). Fantastic! It really shines in 8432 mode!

my window: Short demo on using a window.

nrm distribution: This will give the normal distribution, when you enter the mean and the standard for (x) numbers.

patterns: Graphics - Similar to waves - Just load & run

pdirectory: Used by menu program (contains program files directory)

peek: Sends contents of peek location to the printer

phone phun: Load & run - Converts letters to digits [play with phone numbers, etc]

prime factors: Play with prime numbers

printat: Example of a print @ routine

printat+: Similar to above (printat) + a little more.

printer on/off: Short routine will alert you on your printer's status.

profit: Computes profitability of an item

quicksort: And this is a quicksort

quilt: Prints to the screen random characters at random places. (patchwork quilt?)

roman: Prompts you for a decimal number which it converts to roman numerals

screen dump: Will send whatever is on screen to the printer. (Basic, so it's slow)

scrn format: Benchmark program shows the fastest method. [Cursor control, Poke or Tab]

scrolling: Adapted from disk that came with Protecto package

sdirectory: Used by menu program (contains sequential files directory)

shell sort: This is a shell sort

slow printing: This shows how to control the speed at which the computer will print to the screen using for...next loops. This prints 6 lines at different speeds

speedy: One way to speed up polynomial calculations

stop & cont: Hint on how to use these commands as debugging tools

temp conv: Converts farenheit to celcius or celcius to farenheit.

timer: A timer for your programs ,loops, whatever. You can time to 1 tenth of a second accuracy. No more guessing with for...next loops.

trap+: Useful when programming. Not only shows the line the error occured in but by pressing <return> or <enter> key lists the line on the screen for editing.

trap with sound: Use of trap statement with a buzzer. (Wakes you up when programming)

twinkling stars: Create a screenful of twinkling stars! (not really, but almost)

underlining: Example on using underlining with the 4023 printer

utility audit: Audit your utility bills with this one. You will have to change the data to reflect your local rates

vtab: A useful vertical tab command. Use like tab (horizontal)

wait: An example on the use of the wait command

waves: Another demo on graphics

window demo: This is just what it says. A demo on windows

which key: This tells you which keys you are pressing and whether you are pressing more than one at a time. Try pressing ctrl, shift, tab & esc keys one at a time, two at a time, three, and even all four. Could be used as a prompt in prog

GOLDCOAST GAGGLE CBUG #85 NEW RELEASE 2720 #13047

By: Fred Petersen

This is the third Goldcoast submission. This time we have a collection of utility and accounting programs, grouped for easy use, and mostly a new bunch of games with some "oldies, but goodies" included.

I've tried to limit this submission to those programs found helpful to me as one who is trying to become a self - taught programmer. Note that the startup program is particularly helpful in that it sets the function keys to provide most of the basic commands, statements and functions with the press of one key and initializes the 4023 printer for optimum program printout.

The "f-key template" program (along with the accompaning instructions - fkey template.ss) has been enlarged to update the printout and explain how just one function key can be set to save over 100 keystrokes for repetitive use. Also, ready for printout, you will find two "templates"; namely, "prog helps" (for use with my startup program) and "mastermenu" (for use with this entire disk while utilizing Mr. Goldstein's mastermenu program).

The BASIC labels program, enhanced on an earlier Cbug disk by Rev. Schwartzbauer, has been further revised to allow easy printout of any label stored on the data disk. Makes it much easier than using Superbase to get mutiple labels for the same Company or individual.

The "home video" sequential data has been expanded in the hope of convincing you that this is a really great program.

Amongest the Accounting bunch, particularly try "income property", "speed vs. time", "mtg. comparisons", "loan repay/print" and "varied acctgprgs" - all can be very helpful AND they are handy.

For newies in the Games department give the following a go: "baccarat" (note the backs of the cards say B 128), "baseball", "black and blue", "dragon", fortune teller", "freedom", "guess in 7 tries", "inspector clewso", "matador", "not so easy", "pro football", "psyc test", "toss of coin" (not only enlightening, but funny too), "trucker" and "wumpus". The others are "oldies, but goodies" - some with variations.

I've also included a "pet" (no relation to Commodore's name for earlier computers) theory of mine which deals with the overcoming of gravitational force. Please at least read it and feel free to send your comments. If it has no merit please tell me and I'll not waste any more time or thought on it.

Now, here's the annotated directory:

---

0 "GOLDCOAST Progs " 03 2c disk name

1 " loader" prg loads the introductory screens and the mastermenu.

13 "Masterdirectory" rel A support file for the mastermenu.

8 " bootscrn .scn" prg The introductory screens.

8 " instscrn .scn" prg

8 "boot1" prg

36 " mastermenu" prg Author is "guru" Mathew Goldstein.

48 "mastermenu.instr" seq Instructs on the use of the mastermenu; written by Mat Goldstein.

2 "i----------i" prg spacer

12 "INTRODUCTION" seq Please read first. Much data about the whys and wherefores of the various programs included on this disk.

29 "!!--NOTICE--!!!" seq PLEASE READ. It will make you aware of the rules regarding acquisition, submission,usage and dispensation of computer programs.

1 "u----------u" prg spacer

1 " UTILITIES " prg dummy header

6 "startup.instr" seq Tells about the "startup" program - how to use it and what it can do for you.

9 "startup" prg Helpful when entering a new program or revising an old program.

28 "f-key template" prg A prior Cbug submission (revised to fit better over the function keys). This program permits personalized templates for different functions.

27 "fkey template.ss" seq This is the help-mate to the "f-key template" program - with full instructions.

2 "prog helps.temp" seq These are two of the personalized templates - one for use with the "startup"

64 "autoexp analysis" prg Another accounting gem. Keep track of just how much the 'ole lizzy' is costing you.

23 "address book" prg This really doesn't belong with the accounting programs - it just slipped in the wrong slot. But it is a good program to have handy.

9 "mtg. comparisons" prg Two more "super" accounting programs - lets you pick the mortgage that's right for you

10 "loan debt paymnt" prg and helps you decide about that debt payment.

11 "days twix dates" prg Want to know exactly how many days until your anniversary? This gives you the answer. I'm sure "sweetie pie" wants you to know!

17 "envelope address" prg Another one that kinda slipped in here; however I've left it as it is an easy way to multi address envelopes in basic (prints upper/lower).

7 "loan analysis" prg More on "loans". Get comparisons.

27 "conversion chart" prg Very handy - gallons to liters , etc.

14 "loan repay/print" prg How long should it take to pay off that loan at "x" interest , etc.

42 "varied acctgprgs" prg If you haven't found the accounting help you need try this bunch.

13 "utilities audit" prg Keep those utility companies in line!

1 "g----------g" prg spacer

1 " GAMES " prg dummy header

23 "baccarat" prg Ever play this one? Good instructions! You'll find you are playing with "B 128" cards!

16 "baseball" prg Yup - honest to goodness baseball, although a little

			program and the
2	"mastermenu.temp"	seq	other for use with the "mastermenu" program.
7	"renumber"	prg	Excellent when you've completed or are working on a program and find you need greater number spacing between the lines.
10	"copy"	prg	A fantastic utility program which allows transfer of one or more or all programs from one to another disk, with alphabetization or pattern arranging if desired.
7	"file unscratch"	prg	Who hasn't inadvertently "scratched" a program? Let he who is without sin cast the first "unscratch"!
53	"unutilized basic"	prg	A program recently submitted by Mr. Goldstein that tells you what is there, but unnecessary.
7	"dir two col ptr"	prg	Does just what it says - prints the disk directory in side by side columns. Recommended for every new disk - keep the printout in the disk jacket.
14	"double col ptr"	seq	This allows "newspaper" like printing - see my "prospectus" program for an example.
28	"labels"	prg	Not a replacement for the great Superbase labels program, but most useful for single or multiple printings in any number, and - right from basic!
16	"labels instr."	prg	These two programs clearly explain the use of
54	"labels2 instr."	seq	the "labels" program. Read them first.
26	"home video"	seq	This is a fantastic idea discovered by Rev. Mark Schwartzbauer. Try it - it really works.
15	"screen ed - bug"	prg	One or more of these programs are needed when you work on your "home video" pressentation. Practice
7	"screen ed - cbug"	prg	makes peerfect!
1	"a----------a"	prg	spacer
1	" ACCOUNTING "	prg	dummy header
3	"adding machine"	prg	Adds 'em and gives you the results.
7	"finance prog"	prg	An example program written by my daughter, Lynn Bonner.
3	"savings growth"	prg	Try it - you'll like it!
3	"compound int."	prg	ditto for the next three accounting programs - they
2	"cost of purchase"	prg	very nicely give the answers.
12	"income property"	prg	
2	"speed vs. time"	prg	Want to know how long it will take for your next auto trip
			limited. Needs work.
3	"black and blue"	prg	Makes you say "black".
19	"blackjack"	prg	This is an "oldie", but one of my favorites - play at the Butterfield Social & Recreational Club and wait for the "cutie" to arrive!
22	"dragon"	prg	Careful - the DRAGON will get you.
6	"fortune teller"	prg	Do you believe? Try it.
13	"freedom"	prg	Tough to gain it - you keep adding days!
5	"guess in 7 tries"	prg	Easy little program. You can beat this one.
22	"inspector clewso"	prg	The great, the one and only, French INSPECTOR CLEW-SO!
27	"king"	prg	Has been seen under different names, but slightly changed. Don't lose an eye!
26	"matador"	prg	Try your expertise as a "bull fighter".
12	"not so easy"	prg	A test of your reading skill!
72	"oregon trail"	prg	Stop at the forts - watch out for the "injuns".
25	"pro football"	prg	Yup, football at its finest. You call all the plays for your favorite team as both the Quarterback and the Defensive Captain. Two full haves, with two minute warnings.
17	"psyc test"	prg	You'll enjoy this one - maybe!
24	"solitaire-super"	prg	My favorite way to play solitaire.
91	"super star trek"	prg	Never have been able to win, but I keep trying.
12	"synonyms"	prg	Try this one for "likes" and "unlikes".
12	"tic tac toe"	prg	Another version of an old game.
18	"tic toc tac toe"	prg	Complicated - don't let that Computer beat you!
11	"tnt slot machine"	prg	Another version of "slots".
6	"toss of coin"	prg	How many times will it be "tails"?
58	"trucker"	prg	Carry the load cross country and win, but watch out for "smokey".
26	"vegas craps"	prg	This is the true craps game - gives you all the options.
116	"will o' the wisp"	prg	This oldie's good for an afternoon or evening of entertainment.
25	"wumpus"	prg	If the "wumpus" don't get you, the "bats" or the "bottomless pits" may!
1	"t----------t"	prg	spacer
1	"----------"	seq	spacer
3	" THEORY "	prg	header, but with some comments.
15	"anti-gravity"	seq	This is my theory. Please read this and the next
15	"prospectus"	seq	program and give me your thoughts!!
1	"dc----------dc"	prg	spacer
1	"-DISK CONTENTS-"	prg	dummy header
49	"disk contents+"	seq	What you've just finished
463	blocks free		reading.

GOLDCOAST TUTORIAL CBUG #86 NEW RELEASE 2721 #13052

By: Fred Petersen

This is the second Goldcoast Instructional program submitted for CBUG use as an aid to a better understanding of the abilities of the B-128 machine. <<AND certainly should have been done 10 years ago to be available with every computer

sold!>>

The entire disk this time is directed toward a "tutorial" compilation which instructs in the very basic of basics. It deals with the common and frequently used Commands and Statements; explains and, in most cases, illustrates what they accomplish.

The program is targeted on the uneducated in the world of basic language computing and is probably of little value to the advanced student. Any and all constructive criticism by those who study or simple read this work will be most appreciated.

Hope these additional "golden nuggets" will be found helpful.

<<Fred has put in one place quite a collection of easy to understand programming instructions, easy examples of menu construction, special effects, scrolling, etc. Fully enough that virtually any amature can in but a few hours get the "basic" "hang" of it all. In a few more hours (figuratively), you'll be able (by the examples on this disk) to apply fully operational menus, scrolling, borders, and other "professional" touches to your own programs.>>

<<AND, this is a fun disk. Fred has taken a very light and humerous touch to making common sense out of programming. The 23 sections are subtitled to help make sense out of the basic command. For example, "Line Number - or Address & Zip Code;" "Print - or Show Me." Each are followed with a paragraph or three of easy explaination. Many have example programs to assist the learning process.>>

0	"GOLDCOAST Progs " 02		Disk name.
2	" loader"	prg	Loads bootscrn, instscrn, the seven boot screens and the initial menu.
8	" bootscrn .scn"	prg	Introductory screens.
8	" instscrn .scn"	prg	
8	"boot1"	prg	
8	"boot2"	prg	
8	"boot3"	prg	
8	"boot4"	prg	
8	"boot5"	prg	
8	"boot6"	prg	
8	"boot7"	prg	
5	" menu"	prg	Sets up the two main portions of the program and explains how it can be studied.
9	"menu"	prg	This menu controls the loading and running of the 24 "basic" Sections of the program. These Sections, along with the 6 illustration programs are the example portion of the entire presentation.
122	"introduction"	prg	This is the narative presentation. It tells about the computer and its basic functions and then goes on to describe, in some detail, the reasons for and the uses allocated to the various commands and statements, etc.
3	"basic0"	prg	Explains how to proceed through "basic1" to "basic23" and the 6 illustration programs.
13	"basic1"	prg	Self-explanatory "Sessions" regarding the various
13	"basic2"	prg	Commands, Statements, etc. used in basic
12	"basic3"	prg	programing - with some animation.
12	"basic4"	prg	
12	"basic5"	prg	
8	"basic6"	prg	
12	"basic7"	prg	
12	"basic8"	prg	
12	"basic9"	prg	
13	"basic10"	prg	
11	"basic11"	prg	
12	"basic12"	prg	
10	"basic13"	prg	
12	"basic14"	prg	
13	"basic15"	prg	
14	"basic16"	prg	
10	"basic17"	prg	
13	"basic18"	prg	
12	"basic19"	prg	
13	"basic20"	prg	
13	"basic21"	prg	
9	"basic22"	prg	
13	"basic23"	prg	
1	"animation"	prg	Here begin the "illustration programs" each of which can be individually loaded and run.
2	"depreciation"	prg	
3	"address list"	prg	
2	"deviation"	prg	
3	"ranking"	prg	
3	"word game"	prg	
27	"!!--NOTICE--!!!"	seq	User warning notice - PLEASE READ CAREFULLY!!!
24	"disk contents+"	seq	What you are now reviewing.
5	"scroll"	prg	A further "demo" of the B's abilities depicting a message for the user.
5	"marquee"	prg	An "accolade" to all those who have completed a review of this "tutorial" program!!!
7	"startup"	prg	A brief program, written in basic, which mainly sets the Functions Keys for practical use in programming.
7	"flow chart"	prg	Designed as an initial aid in planning a program.
10	"fkey template.ss"	seq	Explains how to create, edit, save and print out your personalized Function Key template.
2	"prog helps"	seq	A template designed for the "startup" program.
28	"f-key template"	prg	The basic program which allows creation of your personalized Function Key template.
1	"----------"	seq	divider
7	"!!read me last!!"	seq	Words of Wisdom for those who really aspire (and probably also perspire) to become seasoned "dyed in the wool" programmers.
12	"seq read/print"	prg	A convenience for reading the sequential files on this disk without the benefit of SS2.
9	"finale"	prg	Provides an easy way back to user selected portions of the entire program. Also some more helpful "WORDS OF WISDOM".

-CATALOG OF CP/M-86 UTILITY DISK 001
PROVIDED BY:

LT COL JOHN A. WRIGHT
818 JUNIPER DR., PAPILLION, NE.
68046
Tele: 402-339-5728

THIS DISK RUNS ONLY ON A B MACHINE EQUIPED WITH THE 8086 CO-PROCESSOR BOARD. TO USE SIMPLY TYPE "SHIFT RUN/STOP."
IN NATIVE B MODE THE DIRECTORY WILL SHOW 12 BLOCKS USED, BUT ZERO BLOCKS FREE.

****
******** NOTICE!!!: All programs copyrited,,and not ********
******** Releasable outside CBUG, nor releaseable for ********
******** for commercial purposes!!! ********
****

001.01	INSTRUCT.DOC	0K	00	00	Disk usage instructions.
001.02	-CATALOG.002	0K	00	00	This file.
001.03	HELP+ .CMD	6K	65	1E	Advanced help utility. Use "HELP+ ov DISK" to obtain help on files contained on this disk.
001.04	HELP .CMD	3K	5C	CE	Simple HELP utility which will read normal help files (files with HLP as the extension).
001.05	LU86 .CMD	19K	17	23	Self documenting Library Utility.
001.06	USQ .CMD	3K	EB	7C	Unsqueeze, Squeezed files.
001.07	UUDECODE.CMD	13K	AF	B1	De-code, uuencoded files.
001.08	DELBR .CMD	13K	6F	0B	De-library library files.
001.09	TY .CMD	3K	6D	A2	Type, even squeezed files.
001.10	DISK .LBR	185K	45	D6	Library of A86 and help files. (For this disk)
001.11	UUENCODE.CMD	13K	87	A1	EN-code files for transmission.
001.12	SHOW .CMD	2K	0E	B3	Improved bidirectional show.
001.13	BIDUMP86.CMD	2K	A3	0D	Bidirectional dump.
001.14	WHATSNEW.CMD	3K	DD	0F	** Show what has been added/deleted to directory. (See note 1 below)
001.15	SAVE0-2 .CMD	2K	76	33	Reserve space in directory.
001.16	FRAG16 .CMD	2K	7F	01	File-ext cleaner, faster, sorts by file.
001.17	QSORT .CMD	2K	4F	78	Quick sort to order the lines of a file.
001.18	LTYPE86 .CMD	2K	A6	E5	LTYPE for CP/M-86.
001.19	GRAB .CMD	5K	06	9A	Finds paragraphs.
001.20	PRINT86 .CMD	2K	31	FD	Print utility.
001.21	ARCHIVE .CMD	19K	48	52	CP/M-86 archiver.
001.22	DU .CMD	8K	FD	83	Disk utility.
001.23	FILE-EXT.CMD	3K	28	42	File-extensions program.
001.24	FIND20B .CMD	3K	3D	95	Locate any string in any part of a file.
001.25	PAUSE .CMD	22K	14	70	Pause even during submit.
001.26	PRINT10B.CMD	2K	50	2A	Print utility
001.27	PUT .CMD	3K	15	C7	"PUT" files in different USER area.
001.28	SD .CMD	5K	F9	5C	Sorted directory utility.
001.29	SQ .CMD	17K	DF	8D	File squeezer.
001.30	UNERA .CMD	3K	C7	5C	Un-erase file utility. Need the file name.
001.31	VFILER .CMD	9K	C1	23	Screen oriented file utility. Setup for the ADM 31 (CBM emulates the ADM 3A)
001.32	LDIR86 .CMD	2K	B3	0C	Directory lister.
001.33	PIP .CMD	8K	15	78	
001.34	DISK .HLP	1K	69	39	Help file used with HELP+

Note 1: This version of WHATSNEW requires an onboard clock. We haven't gotten the B's clock to run properly. The next disk will contain a version that does not require the clock but will let you set the date. Software Tools RCPM - MISC Volume Number - 001, 34 Files cataloged.

This disk was prepared with the "NEW" user of CP/M-86 in mind. There is a help file available which will will provide assistance for many of the utilities. Simply type the following:

A>HELP+ DISK [cr]

You will be provided with a menue to select from. Choose the file you want learn about, and an "instruction" file will appear on your screen. To advance to a new page, simply use "CNTL S" or the Number "1". If it "runs away" as I have found it wants to do on occasion, simply type a "L" to go back a page.

There are many utilities on this disk. I have checked all of them out, and they seem to work fine. The VFILER is a screen oriented directory utility that is set to run on an ADM 31. We use an ADM 3a emulation in CPM mode, so it is a little hard to use. You can use it by following this sequence:

A>^Z (CNTL Z) [cr] <<where "^" is symbol for "up arror", shifted 6 on keyboard>>

That will clear the screen and get you ready. Now just type:

A>VFILER [cr]

Have fun with this disk. I have not tested every pgm thoroughly, so there may be many applications that I haven't explored. If you find something, let us know, we will provide further investigation into what you find. The CP/M movement is alive and well in CBUG. Tony Gooeliak, Bruce Faierson, myself and several others are working to get this effort off the ground.

There is still work to be done, especially in the response speed, and system archeticture, as well as the MS-DOS world. We are working on these as well as finding compilers that will allow us to write; not only BASIC, but Fortran, C, Cobol, etc programs that run on the "B".

If you like what you see, let Norm know. If you want to help, P L E A S E, let us know! Call me at 402-339-5728, and I will let you know where we are and where we are going.

Again, have fun, and let us know how you like what we have and what you would like to see us provide.

JOHN A. WRIGHT, Lt Col, USAF
818 Juniper Dr., Papillion
Nebraska, 68046

CP/M 86 2.002 CBUG #PR17 NEW RELEASE 2723 #13071

-CATALOG OF CP/M-86 UTILITY DISK 001
PROVIDED BY:

LT COL JOHN A. WRIGHT
818 JUNIPER DR., PAPILLION, NE.
68046
Tele: 402-339-5728

THIS DISK RUNS ONLY ON A B MACHINE EQUIPED WITH THE 8086 CO-PROCESSOR BOARD. TO USE SIMPLY TYPE "SHIFT RUN/STOP."
IN NATIVE B MODE THE DIRECTORY WILL SHOW 12 BLOCKS USED, BUT ZERO BLOCKS FREE.

```
*********************************************************************************
********        NOTICE!!!:  All programs copyrited, and not          *********
********        Releasable outside CBUG, nor releaseable for         *********
********        for commercial purposes!!!                           *********.
*********************************************************************************

02.01      !!--NOTI.CE-    1K   90 27 READ THIS FIRST.
002.02      8OT86   .CMD    5K   21 6B CPM-80 to CPM-86 translator
002.03      8OX86   .CMD    6K   27 BE Untried as yet.
002.04      CRCBUILD.CMD    3K   33 15 Used to build this directory
002.05      FINDBAD .CMD    3K   BO 7A Finds bad sectors on disk
002.06      FFYNDE  .CMD    5K   50 32 Find (Haven't completely figured
                                       this one out yet.)
002.07      HELP+   .CMD    6K   65 1E Improved HELP
002.08      INSTRUCT.HLP    2K   75 E9 Intro to this disk
002.09      PUT     .CMD    3K   7A 7B Put files in different user areas
002.10      TYSQ    .CMD    1K   E4 8A Type squeezed files, even in LBRS.
002.11      UNERA   .CMD    3K   28 42 Unerase erased files.
002.12      XDIR    .CMD   19K   23 57 Improved directory utility
002.13      CRC     .CMD    4K   10 E5 Compute the CRC of a file
002.14      CRCK-51 .CMD   11K   30 CF Same as CRC but improved
002.15      DP      .CMD   33K   D1 60 Disk Patch utility
002.16      VFILE(B).CMD    9K   02 DC VFILER modified to run on B.
002.17      MUCHTEXT.CMD    1K   9F E0 Word Count, etc.
```

```
002.18      DISK    .LBR  168K   59 03 Help, Document and Source Files
                                       for this disk.
002.19      SIZE    .CMD   12K   33 19 Get program size.
002.20      TYPES15 .CMD   12K   E8 2C List directory by file extension.
002.21      RUN     .CMD    4K   D1 AF Run pgms from SUB and LBR files.
002.22      DU-V75A .CMD    8K   FD 83 Improved disk utility
002.23      RUN     .SUB    1K   5F D6 (See RUN.CMD)
002.24      TAB     .CMD   24K   C4 FE Places tabs in code.
002.25      UNTAB   .CMD   24K   61 48 Deletes tabs from code.
Software Tools RCPM - MISC Volume Number - 002, 25 Files cataloged.
```

This is DISK 002 of the CP/M-86 library. It, like DISK 001 has been designed with the new user of CPM in mind. All "TEXT" files can be typed out using the "TYPE" command resident in CP/M-86. I have assumed that you have purchased-obtained my DISK 001 and so have the LU utility that will allow you to "DE-LIBRARY" LBR files. If not, no problem, as none of the programs on this disk require that utility to run. It, and USQ, would be needed to work with the SOURCE (A86) files that are contained in the file DISK.LBR.

I have attempted to obtain all the "HELP" and "DOCUMENTATION" files where available. Several of the utilities do not have any of these files available. However, for the most part these are simple to use and "SELF-DOCUMENTING". I have included the "HELP" files in the file "DISK.LBR". They are all "Squeezed", but by using the following sequence, you can gain access without needing any USQ or DE-LBR routines. (This is the same procedure that I used in DISK 001).

A>"help+ disk" [return]

This will give you a menu to select from. The source files (A86) can be viewed but not changed unless you can extrac and unsqueeze them from the LBR file. I have included them because there is often good information in these files on how to work the program.

You may see many files that look the same as those found on DISK 001. These are either new or modified files. VFILE(B) is one such program. I have reworked the code to allow it to run on the B machine. It still isn't perfect, but at least it is usable now.

I think that between DISK 001 and this one, you will have most of the "UTILITIES" that are currently available from the SIGM group. If there is a utility that is not here that you would like to have, let me know and I will see if I can find it.

There are several DBII utility packages available as well as several different kinds of compilers. I will be working on these over the next few weeks/months. When available I will send them to CBUG for distribution.

A reminder, all these programs have been copyrited by the SIGM users group. They cannot be sold or in anyway used for commercial purposes. They are offered free of charge for personnel use. Any other use requires the consent of CBUG and the SIGM groups.

Have fun with this disk. As more files become available, I will be sending them to CBUG. If you have questions, feel free to contact me. If you want to help, let me or Norm know that too. We need help in the CP/M field.

John A. Wright
818 Juniper Dr.
Papillion, Ne, 68046

WINTER 1988 PRINT FILE CBUG #87 NEW RELEASE #13085

The text files to typeset this issue of The ESCAPE.

```
1 "winter88comp    " w8 2c    94  "pawlus"          seq   7  "g.8050 cp/m"     seq  16  "83"              seq
32  "!!--NOTICE--!!!" seq    45  "8432"            seq  25  "g.8050/sfd"      seq  13  "84"              seq
23  "scratch"         seq    26  "fake"            seq  11  "g.bug"          seq  61  "85"              seq
38  "liz"             seq     9  "g.menu"          seq  14  "chick"          seq  32  "86"              seq
8   "liz hist.n"      seq    35  "g.dump"          seq  14  "easynews"       seq  25  "pr16"            seq
16  "woessner"        seq    20  "g.search"        seq  14  "easylessons"    seq  22  "pr17"            seq
39  "faust"           seq    20  "g.4023"          seq   7  "west"           seq   7  "87"              seq
136 "swan"            seq    13  "g.dir"           seq  13  "east"           seq  24  "library lead"     seq
35  "jarvis/springer" seq    21  "g.ti$"           seq  14  "wright"        seq  979 blocks free.
10  "jarvis/text"     seq    13  "g.converting"    seq  22  "hints.tips"    seq
22  "univ.tran"       seq    16  "g.native dir"    seq  66  "82"           seq
```

Jon Whatley has further revised his Superbase programs for the Super Teacher suite. Below are a few examples of reports generated by these programs including graphing!! Jon is showing us that Orwell's prophetic book "1984" may even have anticipated the B-128. The Super-Teacher suite is becoming so advanced that it should be in every B128 users' library for the purpose of using the concept and programs to easily write other applications. Jon notes that as to the GRAPHING FEATURES:

- No manual entry of raw data required
- Automatic collection of data from the files
- Automatic sorting and ranked display from highest to lowest
- Class average of field graphed is calculated and shown at end of graph
- Choose print or display
- Function keys at prompt hold previously entered field name and description

For want of space the directory to this disk is omitted -- it has 56 entries on it and 174 blocks of additional programming. Like its predicessor, this disk resides in Drive 1, and the applicable data disk in Drive 0. The entire set including #77.11 is $24.00 including a package of 10 blank OPUS disks lest you be tempted to run on the originals -- make duplicates and run on them!

If you already have the set, and wish to upgrade #77.1 to #77.11, write in the item on a blank line on the order form. The upgrade charge for #77.11 is $2.00 (plus the usual $2.00 per order S&H charge). To avail yourself of the exchange privledge attach as much as possible of the original disk label to your order. Should you wish to acquire #77.11 alone, the price is $9.00 --- but caution, without the data files there will be little if anything that will run as is. Order stock #12031.11

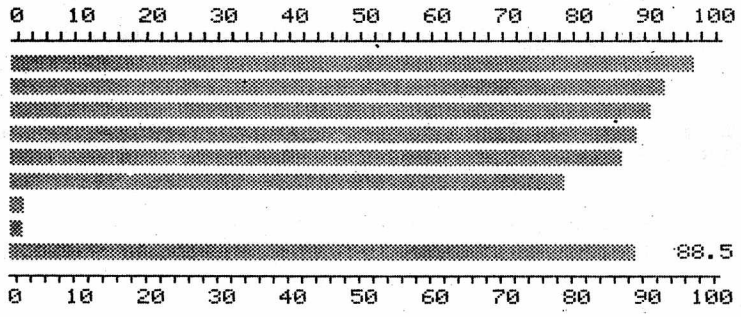
The print outs below have been mechanically edited (razor blade) to get them all on this page. They were beautifully set up for automatic pagination and all on standard 8.5" x 11" paper, (5.5" on vertical center for the notices).

GRADE POSTING RECORD FOR: Period 1, Eco 4651 Posted to: q11

Item: Eco Chapter 1 Quiz

	Item	P1	P2	P3	Av	NOTES
Anderson, Gary	86	86	0	0	86	
Deltzke, Norman	92	92	0	0	92	
Faierson, Bruce	90	90	0	0	90	
Goceliak, Anthony	0	0	0	0	0	Absent
Northrop, Clyde	96	96	0	0	96	
O'Halloran, James	89	89	0	0	89	
Swan, Warren	0	0	0	0	0	Not handed in
Whatley, Jon	78	78	0	0	78	
Class Average	89					

Bar Graph of Eco Chapter 1 Quiz



Ranked Grade List (Cum. Av.)

	P1	P2	P3	Av	
Clyde Northrop	96	0	0	96	
Norman Deltzke	92	0	0	92	
Bruce Faierson	90	0	0	90	
James O'Halloran	89	0	0	89	
Gary Anderson	86	0	0	86	
Jon Whatley	78	0	0	78	
Warren Swan	0	0	0	0	Incomplete
Anthony Goceliak	0	0	0	0	Incomplete
Class Average				89	

John Doe, Instr.

MISSING ASSIGNMENTS REMINDER Student Copy
Period 1, Eco 4651

Anthony Goceliak Eco Chapter 1 Quiz

Quiz makeup will be on the morning of Dec. 25th
TURN IN MAKE-UP WORK AS SOON AS POSSIBLE.
SEE THE TEACHER FOR SPECIAL NEEDS.

John Doe, Instr.

MISSING ASSIGNMENTS REMINDER Student Copy
Period 1, Eco 4651

Warren Swan Eco Chapter 1 Quiz

Quiz makeup will be on the morning of Dec. 25th
TURN IN MAKE-UP WORK AS SOON AS POSSIBLE.
SEE THE TEACHER FOR SPECIAL NEEDS.

MISSING ASSIGNMENTS LIST Bulletin Board Copy
Period 1, Eco 4651

Goceliak, Anthony Eco Chapter 1 Quiz
Swan, Warren Eco Chapter 1 Quiz

The second way to print is in the "DIRECT FORMAT"

96-(CTRL) (E) (P) (D) (A1) (RET) (F12) (RET)

I did that just to show you something good and BAD mayby. Now we got rid of the co-ordinates and here you could print all the way down to row 254 or wherever. But you will notice that it only prints the column widths as a standard of whatever you set all the way across the sheet. No varying column widths permitted.

The third way to print is using the "FORMATTED" commands and provides the greatest flexibilty. At this point you will want to put on a pice of paper just what heading lines (1-3) if any and what column widths you want. As an example on the column widths:

A B C D E F
33 6 8 7 8 7=69
Characters and you have 79 characters available with 9" paper or reseting the "USER" register for 132 characters on 15" paper.

97-(CTRL) (E) (P) (F) (N) (1) (RET) (2) (RET)
(3) (RET) (4) (RET) (12) (RET)

98-(INS) (F) (RET) (7) (RET), (INS) (E) (RET)
(8) (RET), (INS) (D) (RET) (7)

99-(RET), (INS) (C) (RET) (8) (RET), (INS) (B)
(RET) (6) (RET), (INS) (A)

100-(RET) (33) (RET), (RET) (Y) (RET)
"1331 N 4" (RET)

Several things happened here. First off if you made a mistake in entering the print format you can get out of trouble with "CTRL" and start over. You also saved the format on the file disk for later recall. The proof in that is try (CTRL) (E) (P) (F) (O) cursor to the line and (RET) (RET) and it will show you your format to change or just (RET) until (Y) or (N) and prints again.

With formatted printing you can make columns different widths and you can mix up columns in way you desire. You can have headings or no headings. You can save on disks or not. You could create a master format print, save it, recall it and just change line numbers to print to suit a continuing profile.

I didn't plan it this way but I just finished in 100 little instructions. Ha! Ha! There is much more that can be done with Calc Result but that wasn't the plan here, just get you started with something to build on. It is hoped more sophisticated reviews as a take off on this one will be forthcoming in future issues of "ESCAPE". Have fun!

P. S. I just asked my helpmate to proof read this and her comment was
"This is SIMPLE ??" hmmmmmmmmmmmmmmmmmmmmmm!!

P. S. S. It took about an hour and a half to run through this excercise.

8432/e MULTITASKING

Transaltion by: Jurgen Billhoffer

<<It was just brought to my attention that way back in the Summer 1987 issue we neglected to print the second half of the 8432 information files which Jurgen Billhoffer spent hundreds of hours translating from German. Of course, those of you who received the disk have this file already.>>

<<Recently we learned that some of the 8432 disks shipped were defective -- the problem was traced to one of 3 duplication masters. If when you start your 8432 on CBUG #66, your second screen comes up with a glitch in the rectangle at the bottom of the screen, then the disk is bad. Simply pull of the label and send us a note with your name and address. There is no charge for the replacement.>>

The original 8432 operating system was able to create up to four 8032 computer within the 700, but limited to run only one of them at any given time. True multitasking, running more then one program at the same time, was not possible. The new 8432/e enables the 700 to run two programmes at the same time.

To run programmes in unit1 and unit2 in parallel, one has to activate unit1 and two from the main menu and load the programmes into these units.
Then using SHIFT+CBM the user goes back to the main menu and follows this procedure:

1. type "p2RETURN" a "p" appears behind unit2, indicating unit2 is switched now in parallel.

2. select unit2 and type the command "run", then leave unit2 again using SHIFT+CBM.

3. select unit1 and start it with the command "run".

Unit2 running now in parallel is not using the screen, only unit1 is using the screen. The only indication of unit2 being working is the reduced speed of unit1 working (halve the regular speed). Unit1 is therefore called the active unit, unit2 is called the parallel unit. All other combinations of units are possible.

SHIFT+CBM will stop both units and return to the main menu. The parallel unit will have the indicator "p" after its name, as long as it is toggled on to be the parallel unit. Changing the active unit will not effect the parallel unit, but restart it when any other unit is started.

In our example, with unit2 being the parallel unit, another "p2RETURN" will deactivate unit2 as the parallel unit.

The parallel unit can at any time be selected to be the active unit as well. Then the parallel unit is the only unit running and will use the screen. This feature is useful to check the progress the parallel program has made.

If, by starting up the parallel program following the procedure above, step1 and step2 where exchanged then the screen from unit1 would show a strange overlay from the screen of unit2. This can be resolved by typing: SHIFT+CBM 2 SHIFT+CBM 1 .

Using the command expansion peek(....)# and copy# one can observe a running parallel unit from any other unit.

Examples:

1. copy#2,32768,34767
   Will copy the screen memory from unit2

2. copy#2
   Will copy the variables from unit2

3. copy#2,1024:copy#2
   Will copy first the program and then the variables from unit2. This copy can be started using "run.

4. peek(196)#2+(peek(197)#2)*256+peek(198)#2
   Will display the cursor position in unit2

This are only four small examples how the command expansion enable the user to observe and control any

running unit from the outside.

Another very important feature is the switching of a parallel unit under program control.
The parallel unit can be switched on and off from any active unit or from itself. Only two bytes need to be changed.

122 ($007a) in bank 15: 0 if no parallel unit
i if uniti in parallel

65310 ($ff1e) in bank i : 0 if not in parallel.
p (208) if in parallel

The command:
poke#3,65310,208:poke#15,122,3 will switch unit3 in parallel to the active unit. Unit3 will start immediately.

The command:
poke65310,0:poke#15,122,0 will switch the parallel unit3 off under its own control. The switch is done with poke#15,122,0 , therefore this part needs to be the last command in this sequence.

The command:
poke#3,65310,0:poke#15,122,0 would switch the parallel unit off under control of the active unit.

Sometimes when selecting a unit from the main menu the screen will not appear. If this is the case a message appears "press SHIFT+CE" witch will restore the screen.

Warning

It may be fatal to your program, if you try to access a subprogram using gosub# in a unit, while this unit is running a program. This is not the case, while the parallel unit is in "ready" state. But even then it is not very useful, because the performance would be the same as with a not parallel unit, only halve as fast.

Another error might occur, if the two programmes are accessing the disks using the same secondary address. Using different secondary addresses the two units are able to read the same sequential file or to read and write the same relative file. (Do not use the dopen# command, use the open command).

Without special consideration are several applications processed without any problem. The following are only two of many cases:

1. The parallel unit is printing a lengthy disk file, while the active unit is loading, editing and saving a program. All load and save commands can be used.

2. The parallel unit is working on a disk file, while the active unit is running a program with printer output without accessing the disk.

The following error can happen: The parallel unit has after a disk access the required "unlisten" (jsr $f1b9) or "untalk" (jsr $f1ae) missing. If the active unit then is trying to access the bus, it will stop. If the user recognizes the frozen active unit he can press the STOP key. The active unit will then continue to run correctly, but the status of the parallel unit needs to be investigated.

8432 - Utility program "tabler"

Changes the keyboard.

The values of all keys and key combinations can be changed very easily using this program.

Instructions:

1. After 8432 is loaded into the 700, load the program "tabler" into one of the units and start with "run".

2. Select from the menu by typing the appropriate number:

   1 = normal value of the keys
   2 = Shift value of the keys
   3 = CTRL value of the keys
   4 = CTRL+SHIFT value of the keys
   5 = ESC value of the keys
   x = end of changes
   s = save new keyboard

   As one can see, the keyboard has five "levels". Any key can in any of these levels be assigned any value between 0 and 255.

3. After entry of the selected number the message
   "Jetzt Tasten druecken und Werte eingeben"
   (now press key and enter value)
   will appear and the user can now press keys and assign there new value. By pressing the CBM key will the program return to the menu and the user can select a new level.

4. To save the new keyboard the letter "s" must be pressed and the message
   "Kennbuchstabe? (x=Irrtum)"
   (identification letter ? (x= back to menu)) will appear.
   The user selects now a letter or a number to identify this table. The program will then check, if this identifier was used before and if a table with the same name is existing on drive 0. If there is a table existing, the program will ask:
   "existiert schon, soll ich loeschen? (j/n)"
   (table exists, should i delete? (j/n))
   The letter "j" stands for the german term "ja" meaning yes, "n" stands for no. The user can now decide, if he would like to over write the old table or assign a new identifier for this table. The table is saved under the chosen identifier preceded by a "t"; i.e. if the letter "e" was chosen the table will be saved under "te".

5. The new key values are effective immediately, so care needs to be taken, by modifying the function keys.

6. To reprogram the ESC level, one has to use a special technic:

   a) The ESC value is programmed by pressing the ESC key first, releasing it and then press the key to be programmed.

   b) If a key in the ESC level gets the value "0" assigned then the ESC key has no effect on this key, it will act as if ESC was never depressed.

7. Every one of the up to four units in a 720 can work with its own keyboard table. At the startup of the 8432 the user is ask to select one keyboard table for all the units. After the 8432 is started the user can assign a table to each of the units, by loading the table from this unit, following this procedure:

```
sys4        jump in the monitor
.l"te",08   loading the table "te"
.x          leave the monitor
```

The keyboard will then use the new values.
During program control can the keyboard be changed by using load("te"),8. This will open a wide range of applications for this feature.

8432 Benchmark tests.

To compare the computing power of the 8432 operating system we have run the standard benchmark tests as used for several years in the computer literature. Four sets of test where run:

1. CBM-8032 using BASIC-4 (original unit)
2. 8432 in normal mode
3. 8432 with "bank mirror" off
4. CBM-720 using BASIC-256

The following table shows the times in seconds, compared with times of other computers as published in the literature.

	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8	Averg.
1.	1.2	7.9	14.9	16.5	17.8	26.9	42.2	96.3	28.0
2.	0.7	4.9	9.1	10.2	10.6	16.6	26.0	59.3	17.2
3.	0.68	4.7	8.9	9.8	11.0	16.0	25.1	57.3	16.7
4.	1.1	6.2	11.9	12.4	13.5	21.5	35.2	112.0	26.7
ACT-Apricot	1.6	5.2	10.6	11.0	12.4	22.9	35.4	34.4	16.7
Apple II	1.3	8.5	16.0	17.8	19.1	28.6	44.8	107.0	30.4
EPSON QX-10	2.3	6.4	15.8	15.8	16.5	31.9	52.9	65.9	25.9
Hyperion	1.0	5.0	10.0	10.0	12.0	21.0	33.0	35.0	15.9
IBM-PC	1.5	5.2	12.1	12.6	13.6	23.5	37.4	35.0	17.6
Sirius 1	2.0	7.4	17.0	17.5	19.8	35.4	55.9	43.0	24.8
TI Profess.	1.0	4.2	9.3	9.7	10.5	19.0	29.5	31.0	14.3
Tulipsyst. 1	1.0	3.7	6.0	6.1	7.8	15.5	23.3	17.5	10.1

Source for comparative times:

EPSON QX-10 :Personal Computer World 6 (1983) 7,page 172
ACT-Apricot : " " " " " 10, " 156
Hyperion : " " " " " ", " 182
Tulipsystem 1 : " " " " " " " 195
TI Professional: " " " 5 (1982)11, " 111
Apple II : " " " " " ", " 111
IBM PC : " " " " " ", " 111
Sirius 1 : " " " " " ", " 111

The ACT-Apricot and the Tulipsystem 1 are using the 16 bit microprocessor 8086, running at 8MHz. The TI Professional and the Hyperion are using the 8088 microprocessor at 5MHz and 4.77MHz.

The 8 Benchmark tests are included on the 8432 disk.

◆◆◆◆◆◆◆◆◆

FAKE BASIC

by: David Evans

Assembly Language programmers take note! This article describes some techniques that might be of interest to those programming on a B128 or B256 computer.

While working on a multi-user game for my BBS (over 16,000 lines long!), I came upon a need to generate random numbers from assembly language programs. After spending several frustrating days I managed to generate random numbers quite easily using the system timers. The number generated however was not always within the range needed. In basic I would commonly use a function for all my random numbers. It normally looked like this:

```
def fnr(x)=int(rnd(0)*x)+1
```

While the above is EASILY done in basic, try that in assembly language some time! After thinking about it for several days, I decided if basic could do it, why couldn't I do it! After disassembling a lot of basic source code the following technique came to light. To accomplish what I needed to do, you need to do the following:

1: Tell basic where in memory the tokenized "dummy" line is.
2: Set the computer in "program mode".
3: Call the basic execute code from any bank.
4: Return safely to my program.

Several limits came to mind quickly. As far as I can determine at this point, only one statement can be executed at a time. No multi-statement lines are allowed. Line numbers are not used, so GOTO's and GOSUB's are probably out.

To accomplish each of the 4 steps do the following:

STEP 1: Tell basic where in memory the tokenized line is.

First save the basic text pointer (location $85 hex, 133 decimal) in bank 15 (You will need to restore this value after your code has been executed). Next set this value to point to the start of the text. Remember that this pointer is a 3 byte pointer. The first byte is the LSB of the address, the next is the MSB and the next is the BANK the text is in.

STEP 2: Set the computer in "program mode".

Location $43 (hex, 67 decimal) in bank 15, contains the MSB of the current line number being executed. If this value is $ff (hex, 255 decimal) we are in direct mode. To set it to program mode, set this location to zero.

STEP 3: Call the basic execute code.

Location $87AE in bank 15 (B128 machine) is the target address. Before going there, you need to load in the

accumulator the first byte of the text. Getting to bank 15 and back to a non-kernal location takes some tricky programming. First it is necessary to supply a return address (remember return addresses are the desired address less one). Next the desired address PLUS 2 is pushed on the stack. The registers are then set and a call to the bank transfer routines (location $fea7).

STEP 4: Return safely to my program.

This step is the simplest. Simply restore the values saved in step 1 and do a return!

The following program demonstrates the above techniques:

```
equals = $b2   ; token for equals sign
int    = $b5   ; token for integer
rnd    = $bb   ; token for random statement
times  = $ac   ; token for multiplication symbol
plus   = $aa   ; token for addition symbol
peek   = $c2   ; token for peek instruction
poke   = $97   ; token for poke instruction
minus  = $ab   ; token for minus symbol
;
excreg = $0000 ; 6509 execution register
indreg = $0001 ; 6509 indirect register
;
;
lfe = 20       ; location of returned data
tmp = $f0      ; a indirect pointer
;
; routine fnr:
; this routine does the random numbers.  The a register
; has the desired range (1-255).
;
fnr sta lfe     ; set maximum range
    ldx #0      ; force range to 0-255
    stx lfe+1
    jsr calrtn  ; call basic calling routine
    ldx lfe+1   ; return data in a&x registers
    lda lfe
    rts
;
calrtn lda #0
       sta tmp
       sta tmp+1     ; point tmp to address 0
       ldy #$85      ; offset to text pointer
       ldx #15
       stx indreg    ; set indirect pointer to bank 15
       lda (tmp),y   ; get text pointer and save it.
       sta pntsve
       iny
       lda (tmp),y   ; save msb of text pointer
       sta pntsve+1
       iny
       lda (tmp),y   ; save bank pointer of text pointer
       sta pntsve+2
       ldy #$43      ; offset to line number
       stx indreg
       lda (tmp),y
       sta pntsve+3  ; save previous value of line #
       lda #0
       sta (tmp),y   ; set msb of line number to zero
       ldy #$85
       lda #<baslin
       sta (tmp),y   ; point text pointer to my text
       iny
       lda #>baslin
       sta (tmp),y
       iny
       lda excreg    ; make bank pointer point to here
       sta (tmp),y
       sta indreg    ; restore indirect pointer
       lda #<rtn     ; get desired return address
       sec
       sbc #1        ; subtract one(form return address)
       tax
       lda #>rtn
       sbc #0
       pha           ; push my return address msb
       txa
       pha           ; push my return lsb address
       lda #>$87b0   ; destination address +2 on stack
       pha
       lda #<$87b0   ; actually $87ae due to quirk
       pha
       lda baslin
       jmp $fea7
;
rtn    lda #0
       sta tmp
       sta tmp+1     ; point TMP to address zero
       ldx #15
       ldy #$85
       lda pntsve    ; restore text pointer
       stx indreg    ; set indirect pointer to bank 15
       sta (tmp),y
       iny
       lda pntsve+1  ; restore MSB of text pointer
       sta (tmp),y
       iny
       lda pntsve+2  ; restore Bank of text pointer
       sta (tmp),y
       ldy #$43
       stx indreg
       lda pntsve+3  ; restore MSB of line number
       sta (tmp),y
       lda excreg
       sta indreg    ; restore indirect pointer
       rts
;
pntsve .word 0,0
;
; poke20,int(rnd(0)*peek(20))+1
;
;
baslin .byte poke,'20,',int,'(',rnd,'(0)',times,peek
 .byte '(20))',plus,'1',0
;
basvec = baslin-1
```

Should any users have any ideas on this subject or questions, they are welcome to contact me at my address listed below or contact me via my BBS which the phone number is given also. If you call my computer line, you will have to wait to be validated prior to sending the message.

Address : David L. Evans
913 Dearborn
Caldwell, Idaho 83605
Voice Phone: (208) 459-3279
Computer : (208) 454-8421 (24 hours a day)

MENU PROGRAM INSTRUCTIONS

by: Anthony Goceliak

<<Following is a series of papers by Mr. Goceliak explaining products on his latest disk being released in this issue.>>

Those of you who know me realize that the philosophy of 'Menu driven' program seletion is NOT my cup of tea. However, since this is a novel approach, I couldn't resist sharing this.

Shift/running this disk will dload the menu program from drive #0, unit 8. Your Function Keys will all be re-defined, and now allow loading the related file indicated in the key definition from either drive of unit #8. Of course there are two classes of exception. (I never do anything entirely straightforward, do I?)

First are the ampersand files. Pressing the relevant function key will activate drive #0 of unit 8 to load/run the related ampersand disk program. Since the file doesn't

run in the b, it shouldn't load into the b, right?

Second, the 'read instruction' key. It will first display a selected directory of all sequential files from the drive you request and prompt you to input which of them you desire to read. There will be no word-wrap, but if you only need to refer to an instruction file to jog your memory, this saves a bit of bother from the alternative, which is to load Superscript.

Last but not least, key 20 restores all the function keys to their default settings, so you can make sure they are normal, and key 19 restores the screen to it's default setting, which you can relate to a SUPER screen clear.

Should I mention that after having loaded a basic program by pressing it's related Function Key, in order to Run it, type on an empty line the three characters 'run', followed by the return key?

When in doubt, press key #18, which will load a program that can print out a one-line 'definition' of what each function key now does.

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

B SERIES TO PRINTER SCREEN DUMPS

by: Anthony Goceliak

Among the programs released on my first disk was an intentionally imperfect screen dump, and more of you have written to me concerning it than any other file. Intentionally imperfect? Yes, but with good reason. There is no method I know to enquire of the printer whether it has been left in graphics mode or brought to lower case mode, and I was sick and tired of seeing bridge hands instead of english-language spewn forth by my printer (when the printer is in graphics mode and capital letters are encountered). Consequently, my original machine language screen dump left the poor printer in whatever mode it was and then stripped bits from the characters to be printed to make upper case equal lower case, allow punctuation and numbers, and the devil take the graphics symbols.

Why not simply force the printer into lower case and then dump away? Simple enough, but it can leave unintentional booby traps for subsequent operation. Try this if you don't believe me. From basic type the following:

```
open1,4,7 (and return)
print#1   (and return)
close1    (and return)
```

The printer is now correctly set to lower case. It will properly respond to both upper and lower case english letters, etc. Now however, try running Superscript! SS expects the printer to be where it was on power-up, and will not impress anyone with it's features if printing is attempted when the printer is being "hit where it ain't".

All these explanations and people still clamor for a 'faithful' screen dump. O.K. here it is, or rather not just quite.

One and a half characters will Not be faithfully dumped half of the time! (I admit that some of my verbiage isn't of maximal clarity, but the preceeding sentence must rank right near the bottom of the swamp)

One and a half characters: The british pound symbol only on a 4023 (half) and the pi symbol on both 4023 and 8023.

Half the time: Pi symbol is only mis-represented in lower case mode.

OMT (one more time), for those fortunate enough to possess an 8023 printer, there is just one deviation from perfection, that of the pi symbol, and ONLY when in text-mode screen.

On the 4023 printer the british pound symbol has been replaced by Commodore with a backslash symbol, (which looks like divided by reflected in a mirror), while the 8023 types the british pound symbol. I can't figure out whether that means CBM wanted to send beefy printers to the British Isles, or if the bachslash is to be the new unit of currency in the European Common Market, but anyway, there is no point in doing anything about this one since half the printers will be printing this much-used character right anyway.

CBM in it's considerable wisdom, gave us a special key on the b series keyboard devoted exclusively to the mathematic pi symbol, while simultaneously shortchanging us in our printers. In lower case mode, without altering the user-defineable character, there is no way to print a pi, it comes out like a squished checkerboard. Graphics mode handles pi ok, so it wasn't like they forgot about it either.

Therefore if CBM can make a mistake and stubbornly refuse to give in, so can I. You are hereby warned that the lower case pi symbol is printed as exactly the ascii character that the keyboard reflects, and I DARE CBM to fix my 4023 in lower case mode!

Seriously, the new screen dump that I have written has the following features:

1. It comes in two forms, one for those with no added bank 15 memory, and one for those with extra memory at $6b00-$6bff. If you currently run your machine with Ms. Deal's Keytrix, either use the extra memory version, or power down and then up again without keytrix, since the two utilities fight over the same memory.

2. You asked for it, the printer will be automatically set to 'echo' whatever mode the b-screen is in. Remember the warning from above, after a screen dump, your printer may have had it's mode changed!

3. No memory beyond the single page occupied by my code is changed, (and if my dump is to be called from a machine language program, neither are any registers!) Maximum stack depth is ten, (need I say that the stack pointer is left where it was on entry), otherwise a truly stealthy routine! For those who are intending to use the utility from basic, there is absolutely no need to fret over the stack. Your b will already have ground to a halt if there are less than 64 stack bytes available, so if a basic program runs, then this utility has plenty of room on the stack.

4. You may use whatever device number you choose, but please make sure it is a printer! (Poke required for other than device #4).

5. You may dump the entire screen, (the default), or only selected lines, or only selected columns, or indeed any rectangular area of your screen (a window dump). This is entirely independant of any window(s) you may have established on the b screen.

6. You may decide to dump a 'virtual screen' from any bank! Completely independantly of the b screen, a section of memory may be designated as a virtual screen, and although I leave it up to you to determine how to write or draw there, you can print out the results! This 'screen' is completely programmable, even as to number of columns per line.

22 column vic screens, or 40 column pet/64 screens anyone?

7. Within the 1 1/2 exceptions 1/2 the time, 'what you see is what you get'. Included with the screen dump machine language programs, there is a basic program entitled 'dumper demo' and a test screen which will put the dump through it's paces. Whether you are dumping text or graphics, this should turn the trick for you.

8. The utility respects whatever you have done (or not done) with the line spacing on the printer. You may equally print 20 or fewer lines spread out to cover an entire 11 inch sheet, or adjust the spacing to allow lines to touch for

graphic dumps (and paper misers).

9. The utility respects whatever (if anything) you have done with the user-programmable character. If you had gone through the trouble of making it into your own logo, I think you would be justifiably upset if I stole the logo and made it print a letter 'pi' instead of what you had explicitly set it to.

Adding this routine to a basic program is easy. First decide which version to use. If you do not normally use Ms. Deal's Keytrix, select "screen dump 1792". Add a line in your program in the following form, altering the line number to suit you, but early in the program is reccommended.

```
10 bload"screen dump 1792",d0,u8,p1792
```

If you wish to use drive1, substitute d1 for d0, and if you wish to bload the file from a unit other than #8, substitute u# for u8.

Determine the point or points in your program where you wish to perform a printout of what is on the screen. If you wish to print the entire screen to device #4, (the normal printer device#), add this command at the appropriate line or lines (obviously altering the line number appropriately):

```
510 bank15:sys1792
```

For those who use Ms. Deal's Keytrix there is a choice to make. If you have added some memory in bank 15 via Mr. Anderson's cartridge or the modified Calc Result cartridge, the simple route is to select the second version of my routine which resides at bank 15, $6b00. Otherwise add a line (again preferably early in the program), prompting you to reset your b.

The appropriate line in basic to bload the second version of my file would be:

```
10 bload"screen dump 6b00",d0,u8,p27392
```

And the proper way to execute the default dump is:

```
510 bank15:sys27392
```

When or if you ever wish to make use of any of the 'fancy' dumps to your printer, please refer to my basic program "dumper demo" for the proper way to change the default settings. My own favorite is the virtual screen.

From a machine language program, the routine is entirely transparent, even the psw is restored, merely jsr $0700 or jsr $6b00 as appropriate.

As a final point, the infamous square brackets are handled properly, as are printer quotes mode, displayable cursor control characters, reversed screen characters, normal as well as graphic mode b screens, graphic characters beyond the pi and lb. as mentioned above, as well as characters which can be only placed on screen via pokes (there being no way to enter a few values from the keyboard).

Not bad for less than one page of m/l is it?

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

GLOBAL (DISKWIDE) SEARCH AND REPLACE IN SUPERSCRIPT II

by: Anthony Goceliak

Last issue of the Escape saw Norman tout one of my programs which allowed a diskwide 'HUNT' or 'SEARCH AND REPLACE'. It works, but it requires you to exit Superscript, and it runs slowly. Here is a much better alternative, or rather a pair of alternatives, with the following advantages:

1. The operation is reversible. You are NOT stuck with permanently renamed files, or permanently linked files.

2. It operates from within Superscript II, from basic, or from the machine language monitor:

3. It is MUCH faster.

On this disk are a pair of programs to create
&autolkd0seq.imm
and &undo autolk.imm

The basic language programs will execute from any b series computer b-128 or b-256 and create the working ampersand file which runs on any 80xx drive. That includes 8050 (2.5 or 2.7), 8250, or sfd-1001. The reason I have written a 'generator' program instead of the actual files themselves is for the benefit of those who own only an sfd-1001 and who cannot copy files from disk to disk. An ampersand file in my nomenclature with a suffix .imm designates a file which is going to take IMMEDIATE ACTION, on the disk holding the ampersand file, with no pause to allow you to swap disks. In this case, the actions of the file are both benign and reversible, so that there is no real need for precautions of disk swapping.

Dload and then run first one, and then the other generator program. Follow the instructions to create the working versions of the &files on your superscript data file disk or disks.

Presuming you have already created the two &files, from Superscript II, enter disk mode [press key marked rvs], and with the disk which has the ampersand files in drive #0, type either &auto* or &undo* [and return, of course]. The remainder of the operation is entirely automatic, it proceeds at the speed of the 'collect' or 'v0' disk command, and when done, (assuming &autolink.imm was selected) all seq files will have been renamed in a sequence of ascending numbers beginning with 100. The 'old' name, or at least the first twelve characters of it will follow the 'new' name, but don't worry, if the 'old' name was longer than twelve characters, it is preserved though hidden for now. Each seq file will have been read through at machine language speed to the end (the only proper place for the SSII link command) and the appropriate *lk:### will be automatically appended to the file. Prg files, usr files, or rel or del files, and indeed locked seq files will not be disturbed by this, and remain entirely unaltered.

Now you may perform global 'hunts' or 'search and replace' command(s). One suggested use for this was to change the year 1987 to 1988 on all form letters on a disk, although there are many others as well. If you wish, you may remove the disk, and power down your computer if 5:00pm arrives before you are through with whatever file maintenance you intend, since the disk holds everything needed to restore the filenames safely.

Whether you have powered down or not, when you wish to undo what &autolink did, goto SS II's disk mode again with the disk in drive #0 and type the name of the 'undo' file, namely &undolink.imm [and return]. As before, the operation is entirely automatic, the filenames will be restored, and the SS II *lk:### command will be deleted from the end of the file.

From basic, with the 'target' disk in drive #0, type the following:

```
open15,8,15,"&auto*" [and return]
  or
open15,8,15,"&undo*" [and return]
```

and in either case, after the disk activity has stopped, if the error led is green type close15 [and return].

In the event of an unrecoverable error, dos 2.7 drives

will respond to the query ?ds$ with the error information describing the type of error, and track ans sector information. Dos 2.5 will not, but will turn the error led red if an error occurred. Pull the disk out to regain control of the drive for dos 2.5 (makes you aware that a disk error has occurred).

If an error has occurred, it would be VERY wise to make a backup of the disk before doing anything else, since an error tells you that something went wrong! Many people have turned a single scrambled letter into a disk catastrophe by ignoring errors. If the backup command will not operate, try one of my programs such as 'copy disk via q'; which will if needed, continue beyond a sector or sectors with error(s).

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

4023 GRAPHICS BUG

by: Anthony Goceliak

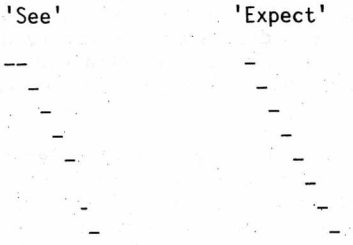
In my article regarding the b-series screen dump, there is one other set of inconsistencies which I would like to point out, even though the majority of users would never notice the imperfection. The difficulty is in a non-systematic series of inconsistencies regarding the graphic symbols which produce horizontal lines. Specifically, if you were to place your b into graphics mode and type ?chr$(164) you would see one of the series, of which there are eight unique members via screen display, but only seven unique representatives on your 4023, with one duplication, and not so obviously, one omission.

[To make things much easier, I have included a short basic program entitled '4023 bug demo' which exposes the inconsistency much better than I will by words alone].

When the b is placed in graphics mode, pressing the appropriate keys, or alternatively, printing the appropriate chr$(...) will display the series of eight possible horizontal bar symbols on screen. [They cannot be displayed from text mode]. If arranged in either ascending or descending order, they can make a 'staircase' of uniform slope. Upon sending the same string of characters to the printer, a similar, but not exact staircase will result. It is human nature to try to make patterns out of nothing, and to try to make things fit, so my demo program follows this up with a different string, including a chr$(254) [the programmable character, suitably defined of course] which should demonstrate the now 'sticks out like a sore thumb' differences. All in all, no less than six of the eight characters are incorrect! It would normally take a microscope to detect the fact that these six characters produce print which is one dot too high, but I recently wrote a program which produced a graph on my b using exactly these characters, and when I producwed a hard copy, it just wasn't right. (I admit, I DID use a microscope!)

This kind of reminds me of the time in school when the teacher was introducing the class to the concept of pi, the ratio of a circle's circumference to it's diameter. She had instructed everyone to make a string of some arbitrary length, say six inches, double it and use it as a guide to draw a circle by sticking your thumb on the two ends, a pencil in the loop, and holding the loop tight, make a circle. That is somewhat harder to do than to describe, but then she indulged herself a bit by asking everyone to use the string as a 'ruler' and measure around the circle, and those who got EXACTLY three 'strings' as an answer to hold up their hands. Well, one by one, hands shot up, some a bit more tentative than others until everyone's hand was in the air except mine. She waited for what seemed to be an eternity, looked me square in the eyes and asked 'Well, what's the matter with YOUR circle?'. I replied somewhat abashed, that it didn't fit three times exactly, whereupon she whirled around to everybody else in the class and roared that IT SHOULDN'T HAVE FIT THREE TIMES EXACTLY! I've never seen any of my classmates since we graduated grammer school, and perhaps some do and some don't remember the value of pi, but I'll bet you there are almost thirty of us who won't hesitate to say the emperor has no taste in clothes.

At any rate, here is a representation of what you see, vs what you expect:



Please bear in mind that this demonstration in the escape becomes magnified in the vertical dimension only by a factor of eight, so the difference which appears at a glance is not so easy to spot when your 4023 spits it out.

Now, as to why I have not 'sorted these things out' in my screen dump. First, notice that I have kept referring to 4023 and not 8023 throughout this article. 8023's handle these characters perfectly! A 'corrected' set would make an 8023 misbehave. Second, there is simply NO character which is in the 4023 set which will produce what chr$(198) prints on a graphics screen. I can usurp your one and only programmable character to patch around it, but only by killing any chance you have of using the programmable character reliably in a program. To me it is too high a price to pay just to straighten out a line displaced the height of one dot on your printer by Commodore.

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

TOTAL CONTROL OF DIRECTORY LISTINGS

by: Anthony Goceliak

The basic language program associated with these instructions gives you complete control over how, in what order, and indeed whether to display the filenames of files on a given disk. You are given complete flexibility to group your directory listings any way that you desire. Quick reference instructions are displayed on-screen at all times, so a detailed description is really not needed here. Your screen will be partitioned into three 'windows', with the left-hand side showing the old directory listing scrolled in a circular fashion, much the same as someone familiar with Keytrix, but done entirely from basic. You may scroll the list past your cursor pointer, or move the cursor pointer to some desired (or undesired) file for selection. Scrolling the list instead of the cursor becomes understandable when you consider long directory listings which do not fit on a single screen.

The Center window is static, and displays each command the program is capable of and the single-keystroke way to invoke it.

a = accept all remaining files in old order BELOW what you selected, and re-write dir.

d = scroll old directory screen downwards past cursor.

e = erase the old filename opposite cursor from the list.

o = Oops! (reject the last selection from new

list)
q = quit program without rewriting.
r = rename the file at cusor position.
s = select the file at cursor position for new directory.
u = scroll old directory Upwards past the cursor.
↑ Cursor up and cursor down move cursor as usual.
l = list the numbers (a diagnostic proceedure)

One point bears mention, an erase (e) removes the filename from the directory, but does NOT de-allocate the blocks which the file occupied as does scratch. If you wish to conceal a file, this is a medium-security way to do so, but if you wish to scratch the file, a few extra blocks free can be gained by Collecting the disk after the program is finished.

The right-hand window contains the list of filenames in the order you have selected them, and can be built-up, or in the event of a mistaken selection, torn back down. When you are finished sorting, you may either re-write the directory to your specified list, or merely abort with the original directory intact.

Filenames as they are selected for the New directory are highlighted by displaying their names in reversed video on the old list, and by pre-fixing the name with an asterisk. Since DOS would never have allowed you to name a file with a leading asterisk, this is never a source of confusion.

The sole machine-language section in the program is the inclusion of the public-domain routine by Mr. Butterfield/Ms. Deal, stringthing, which considerably speeds the reading of the directory at the beginning of the program.

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

WHY TI$ SHOULD NOT BE TRUSTED FOR PRECISION TIMING

by: Anthony Goceliak

The variable ti$ is derived not from the crystal clock running the b computer, but from the 60 cycle [50 in europe] line frequency powering the b via the wall socket. I have known this for some time, as have many of you, but there is now an unprecedented opportunity for error in setting this variable.

First the inevitable brief diversion. Why is the 60 cycle line such a baddie when your kitchen clock keeps such splendid time? The answer lies in the amount of integration applied to the definition of accuracy, or more plainly stated, over what time period are we considering accuracy? The power company, when demand is high, just physically cannot spin the generators quite fast enough to maintain 60 cps (anybody who wants to bring Herr Heinrich into this discussion better be prepared to be hurtin'), since it requires more mechanical work than their energy source can provide [whether it be coal, oil, gas, water, or atomic]. The generators can occasionally slow to 58 cycles, although this much is relatively rare. However, knowing all about the clock that you (and I) keep plugged in to the kitchen outlet, they dutifully speed up their generators during periods of light demand, so that averaged over a day, or a month, your clock maintains perfectly wonderous accuracy. Human activity is such that being ten seconds early or late is rarely noticed, as long as the error is not additive.

The following experiment can be instructive when run at different times of the day, or different times of the year, and well illustrates my point. Your b doesn't get smarter or dumber at different times of the day, but ti$ runs 'fast' or 'slow' slightly.

From basic, enter the following program after having just turned on your b.

```
10 ti$="0000000"
20 for x=1 to 100000:next x
30 ?ti$
```

The time reported for the program to run will vary slightly, beyond the expected [and perfectly normal] variability of +/- one count (having set ti$ to zero, will it be bumped to 0000001 virtually instantaneously or will it have to wait 0.1 second?)

This variability can be exposed by the following program:

```
10 x=0:ti$="0000000"
20 if ti$="0000000" then x=x+1:goto 20
30 ?"x=";x,:input"try again";y$:if left$(y$,1)="y" then 10
```

Note the values printed for x. They represent the number of loops your b performed before 0.1 seconds elapsed as defined by ti$. On my b they vary from 14 to 16, exactly as the +/- one count theory predicts. Exit the program by typing anything except 'y' at the prompt.

One other instructive point for basic language programmers should be mentioned here as well, since it can sometimes significantly add to a program's speed. Try the first demonstration program again, and write down the time reported. Edit line #20 to eliminate the ' x' beyond the keyword next. Line #20 should now read:

```
20 for x=1 to 100000:next
```

Run the program again. Have any light bulbs suddenly lit?

Now the main point. Apparently some CBUG'gers have acquired uninterruptable power supplies in order to maintain critical computer operations in spite of power failure, and some of these supplies have very poor control of the 60 cps output frequency. This error is both large and additive, and can result in all kinds of timing difficulties when ti$ is used to regulate something important instead of merely reporting that it took you 17.2 seconds to shoot down the alian air force. Programs such as my on screen clock, when run on a system so equipped, can yield twenty or more minutes per day CUMULATIVE error, becoming worse than useless in very short order.

On this disk, there is a program titled "80xx speed test" which will report two speeds on a given disk. Once again, 80xx refers to any drive in the series 8050, 8250, or sfd-1001. It uses no machine code, and best of all does NOT destroy any tracks in the process of testing, which means you can test absolutely any of your formatted disks without fear of losing their data. Two tests are performed in order to compare averaged speeds for tracks #1 and #77 in an effort to help pin down belt slippage on whichever disk is giving you trouble. As you may know, the resistance to spinning varies from disk to disk, and especially from brand to brand, (some disks have liners, some don't, etc.), so the opportunity to test speed on a given disk is valuable. The downside to all of this is that since precise m/l timing is unavailable, the speed resolution is somewhat low, +/- 0.5 rpm, and the test will take roughly 2 minutes. Speed should ideally be 300 rpm and should NOT vary from track #1 to track #77. Adjustment of speed is not reccommended using this program, since ti$ is used for timing, and it will take the patience of a saint to wait out the cycles of 'tweak', 'see what it did', and 're-tweak'.

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

CONVERTING NATIVE-MODE 8050 DISKS TO CP/M-86 USEABLE FORM

by: Anthony Goceliak

First, let me say that this program is hopefully doomed. If my efforts to improve the disk access speed of our implementation of cp/m-86 succeed, then this program will of necessity need to be rewritten. In other words, this program is being released in direct violation of my principles of consistency in programming.

However, when / if the access program is released, this program will be simultaneously upgraded, and a disk converter program will be available to copy all information from the current format to the new, (and maybe you will choose not to speed up disk access time anyway!) Consequently, no one will be left with useless disks full of files, and so, properly warned, you can see what this program will do for you.

We all owe Lt. Col. Wright a great debt for devoting many hours time to downloading and de-bugging several disks of cp/m-86 files. This program will convert cp/m-86 files stored as native mode files to the proper form for co-processor equipped b series computers to handle. No additional hardware beyond a b computer, monitor, and an 8050 drive is required. Your machine does not even need a co-processor to execute this program, but one will of course be required to actually use the resulting disk(s).

The program operates in two parts, one as a basic program from within the b, and a second as an m/l program from within the drive itself. Reading, modifying, and re-writing a file are all carried out entirely within the drive, eliminating the time-consuming operation of transfer to and from the b. Only the directory entries are handled in this less-efficient manner, mostly to enable me (and you) to keep tabs of the progress of the program as it proceeds.

The program is set up to specifically handle the pre-release 8, 9, and 10 files, producing four output disks in 8050 cp/m-86 useable form. Additional disks beyond these may require minor modification of the filenames to be skipped on the first pass to accommodate the somewhat wasteful use of disk space in the cp/m-86 format, but the program will otherwise continue to perform. If you have a native-mode terminal program which you like and have cp/m-86 programs available for download, this program will convert the disk files produced into ones suitable for use by an 8088 equipped b.

By the way, as a reward to those who have read the entire article, here are two useful tidbits:

1. An 8050, or for that matter an 8250 or sfd-1001 'cp/m-86 formatted' disk is perfectly duplicated by the native mode command - backup d0 to d1, or by my ieee unit to unit backup program.

2. A 'blank cp/m-86 formatted' disk is similarly capable of duplication by either of the above methods.

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

NATIVE-MODE DIRECTORIES OF CP/M-86 8050 DISKS

by: Anthony Goceliak

First, let me say that this program is hopefully doomed. If my efforts to improve the disk access speed of our implementation of cp/m-86 succeed, then this program will of necessity need to be rewritten. In other words, this program is being released in direct violation of my principles of consistency in programming.

However, when / if the access program is released, this program will be simultaneously upgraded, and a disk converter program will be available to copy all information from the current format to the new, (and maybe you will choose not to speed up disk access time anyway!) Consequently, no one will be left with useless disks full of files, and so, properly warned, you can see what this program will do for you.

Our version of cp/m-86, like all others, assumes the worst in regard to our disk drives. i.e. not one iota of smarts beyond a circular tape recorder, and so every last disk will be produced with the same name and id#, and a cp/m-86 directory tucked away on track #3 using a format quite unlike the cbm directory. However, since it does make sense to the cpm/-86 operating system, what is the big deal? No problem, UNLESS you are in our native mode, and surrounded by five or six cp/m-86 disks and don't feel like having to boot up the 8088 just to figure out which disk goes in what sleeve, or whether to backup this disk or that, because you promised to send a copy of 'wim-wam.cmd' to someone.

What this basic program does is to place a disk file (&see cpm.imm) directly onto a cp/m-86 formatted disk and incidentally also free 12 additional blocks for native mode files, or whatever.

Whenever the ampersand file is activated, it will update the native mode directory to reflect the cp/m-86 files on that disk. They will of course, NOT be available from the native mode, but at least you can tell what is on a given disk. Full instructions regarding how to activate the file are given in the basic program.

The directory of a sample disk is listed below.

```
1 "cp/m-86 disk    " 88 2c
1  "cp/m boot"     prg
10 "cp/m-86"       prg :the 2 files needed to begin cp/m-86
2  "&see cpm.imm"  usr :disk command to update native dir
0  "hjelp   .lbr"  del :
0  "qsort   .lbr"  del :all remaining are cp/m-86 files
0  "ty      .lbr"  del
1  "apc-caln.lbr"  del :note this file is in user area #1
1  "apc-date.lbr"  del
1  "apcserio.lbr"  del
0  "bytype86.lbr"  del :while these are in user area #0
0  "fastvf86.lbr"  del
0  "frag86  .lbr"  del
0  "grab86  .lbr"  del
1  "save0-86.lbr"  del
0  "wc86    .lbr"  del
0  "bidump86.lbr"  del
0  "bishow86.Cmd"  del :this is set to read only [C vs. c]
0  "bishow86.lBr" del :this is set to system file [B vs, b]
0  "ltype86 .lbR"  del :this is set to archive   [R vs. r]
0  "print86 .lbr"  del
0  "typesq86.lbr"  del
0  "!!--NOTI.CE-"  del
12 blocks free.
```

Beyond the first three files, all remaining entries are cp/m-86. The blocks used represents the 'user' area of the associated file, and all filetypes are listed as 'del' to remind you that they are not accessible from this directory. The filename extension will be displayed in lower case for r/w dir non-archived files, and will be displayed in the following manner for the following attributes:

```
.Ccc = read only
.cCc = system file
.ccC = archived file
```

Should you add or erase files, or modify attributes of your cp/m-86 files, merely activating the already-resident disk program from drive #0 will update the directory to the current cp/m-86 status. It is not necessary to re-run the

basic program.

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

CP/M-86 8050 DISK LAYOUT

by: Anthony Goceliak

The following program when run will print out a chart showing the first sector assigned to a cp/m-86 'block', which is really 8 sectors, each in ascending order with an interleave of one. To make that a bit plainer, consider cp/m-86 block #0 [the cp/m directory]. My chart shows that it begins on track #3, sector #0, and therefore block #0 runs sectors 0 -1- -2- -3- -4- -5- -6- -7. There are only two funny points, cp/m-86 believes in sectors only up to 22, and it does NOT believe in tracks #38 or 39.

Block #100 begins on track #37, sector 18- -19- -20- -21- -22 and then continues on track #40, sector 0- -1- -2.

One final item, cp/m won't write voluntarily beyond track #75, but if you manually place information there and show file control block #205, it will be read correctly even though block #205 begins on track #76!

This chart, or one like it is utterly indispensible if you wish to track down and alter information in a file via a 'disk doctor-like' program, since the organization of a cp/m-86 disk dispenses with the treack and sector links of the CBM system, and instead only shows the FCB block numbers in a directory listing.

Mr. Anthony J. Goceliak
32 Cottage Street
Jersey City N. J. 07306

WRITING 8250 BAM ACCEPTABLE DISKS WITH AN 8050 DRIVE

by: Anthony Goceliak

The following code can make life on a system having mixed 8050 and 8250/sfd-1001 disk drives much simpler. It works in the opposite direction from my &8250 lobotomy program by modifying an 8050 disk to have an 8250 bam. Of course the back side of the disk is blank, and all tracks above 77 appear fully allocated, but an 8250 or sfd drive will no longer turn it's nose up in disdain when presented with an 8050 disk modified in this way. Basic (or m/l) programs with embedded dclears or "i0 or i1"s (as appropriate) will also run without reporting illegal track 00 sector 255 when the 8050 disk has been set-up by this program. It is written in the form of an ampersand file to allow use independantly of whatever is resident in your computer. To load & run transmit the string "&pseudo 8250 bam" to the drive with the disk containing the & file in drive 0 and the 'target' in drive #1. Whether you are in basic, monitor, direct mode or program running, even from wordprocessors and the like, the disk utility will be executed.

The code is fairly well self protecting, the 'new' bam blocks at 38/6 and 38/9 will only be created if all unallocated blocks were there before. Note that I said all. Belt and suspenders! Therefore, you may try execution of this disk utility not only on a newly formatted disk, but on a nearly full one as well. If blocks 38/6 AND 38/9 were both free, the program will proceed. Otherwise, the drive will sulk, with error led and drive #1 activity led red and nothing overwritten, until you remove the offending disk. Should the fdc fail to execute any of the steps correctly, the led's will similarly flip, but the drive remains active, so you may examine what went wrong. Error information is retrievable by 'm-r' chr$(6) chr$(16) chr$(1), and the actual track and sector in trouble is in the header block table 'm-r' chr$(59) chr$(16) chr$(2). Although you may not be able to recover when some fdc errors have occurred, there is no point in my shutting you out and keeping you from trying. One more point, the error led may be red, but there is no valid ds$ or error message behind it.

Another thing to keep in mind when using ANY ampersand file is to beware open buffers. The ampersand file described here 'lives' in buffer #7, and makes use of buffers #3 and #13. Had you any half-full write buffers, well, too bad. Ampersand files in general won't give any warning that they are about to load into the middle of your open buffer, so designing in protection against other open buffers becomes moot. Also on a successful exit, the drive is told that it has a new disk in it, so if there were any unwritten bam changes, they too also be lost.

In order to facilitate use of my ampersand files, I have made it a personal convention to make whatever sits in drive #1 the 'target' disk, so that the ampersand file does not have to be copied ad-nauseum every time it is to be used. The ampersand containing disk, which is NOT modified, sits in drive #0. There is no straightforward way to execute ampersand files in drive #1, and so drive 0 targeting involves additional code to allow detection of disk swapping and re-initialization of disk id#'s, or the need to copy the ampersand file to the target disk, but for an application like this, the compulsory use of drive #1 is not onerous, unless, of course your drives are mis-aligned.

The best time to execute this file is right after headering, when the required blocks will be empty and the program should execute barring a drive failure. From basic, place the newly headered disk in drive #1, with this disk in drive #0 of the same unit. Type the following, replacing (unit number) with the appropriate number, (as in open15,8,15)

open15,(unit number),15,"&pseudo*" (and return)

The rest is automatic. If the error led is green, the disk has been successfully modified and you may proceed by typing

close15 (and return)

Should you wish to 'convert' a partly or almost-full disk, the proceedure is somewhat more complicated, but try the above step first. If it fails, as it almost certainly will with a filled disk, you must run the disk editor program of your choice to determine which file owns blocks 38/6 and 38/9. Hopefully one, but sometimes two different files. Copy both files (or the one) to another disk, scratch them on the disk you wish to 8250-ize, and then re-run the &pseudo* utility. Once the utility has been successfully run, the files may be copied back to their original disk, unless there were either one or no blocks free at the beginning. In that case one of the files must be left off the disk to make room for the two extra bam blocks. Should the file be important for the disk, once the &pseudo utility has been run, a less-important file may be copied and scratched to make room for the important file. However consider this, do you want to entrust all that typing, programming and data to a single 60 cent piece of plastic? I like to follow the old wall street adage 'liquidate to the sleeping point'.

Should you collect a disk modified in this way on an 8050, it is reccommended that you again perform the ampersand operation since the 8050 will re-write the original bam links as it collects. No harm but the disk will again behave as an 8050 disk when used in an sfd-1001 or 8250. One anomaly of the 8050 is worth mentioning, you must collect the disk TWICE or the utility will fail. Blocks 38/6 and 38/9 were in the directory chain the first time an 8050 collects an 8250 disk, and so even though the blocks will no longer be in the chain which is to be written to disk after the first collect, the blocks will be marked as allocated on the bam. The second collect will find them no longer in the chain and they will be freed for the utility.

Mr. Anthony J. Goceliak
32 Cottage St.
Jersey City N. J. 07306

SEI COMMENT & A FEW HINTS & TIPS

by: Art Chick

Norm, Sorry I have not written sooner, but the last three months of 1987 was a real barn burner. This is to bring you up to date on the performance of my "resurrected" disc drive. The SEI conversion appears to be working exactly as advertised. The drive swap switch works, the speed holds at 300.4 to 300.9. I have not even lifted her skirts to see how this miracle was done.

On to other things. I recently tried to read the manual on my CBM 8023P printer to make a degree F or C sign (§F/C). On page 27 Commodore tells the reader to lay out a 7x6 matrix. I tried this and could not get the sign to print where I wanted it. On reading further, I found a description on page 33 on storing bit image data into the printer buffer. This describes an 8x8 matrix. I tried this and it works (as you can see). To make it work in Superscript there are a couple of tricks that I have learned, but first the format.

[*](format command)sa5:sa5,0,96,144,144,96,0

To print the sign type [$](esc. $)F or C whenever you want it printed. The trick is that if you try to output to video you get a "format error type A" message. You must place a [*] nb in front of the data statement if you want to check your layout before printing. and then remember to remove the nb before printing. It is a real pain. If there is a way around this, I would like to know it.

Of course any symbol or special character you want can be designed and printed with this method.

In my last letter to you I volunteered to be a "Yell for Help'er" in the fields of basic programming, superscriptII, superbase, and calc result. I do not profess to be an expert, but over the last several years I have managed to drag out of the various manuals and your great "rag" enough information to let me get a lot of work done. I can be reached at (408) 722-8475 evenings (to 10 PM), California time.

I have found a method in Superbase to create a sequential file on drive 1 from a "sorted file" in Superbase. This is not explained very clearly in the manuals but I have found this system works.

```
all fill to 1:"seq.file"[field1][field2]...[field??], or
fill                                          from"oldfile"
to1:"seq.file"[field1][field2]...[field??].
```

Well that is all for now. If I find any more little gems I will try to communicate them.

Thanks again for all the work you are doing for us orphan owners.

Well Here it is the end of February and I am trying again to finish this letter.

Since I started this letter way back in January, I have taken the time to lift the cover of the SEI modified disc drive to see what the workmanship looked like. It is a very "professional looking" job. A neat pc board on the side of the cabinet, and a bunch of wires running to various connectors. The drive reversing switch mounted on the cabinet has a togle switch and a red led to indicate the drives have been switched. works very well.

CBUG West EASYWARE

Remember that the purpose of EASYWARE is to identify some of the CBUG library disks that are very easy to use. They aren't the only good disks, just the ones that we at CBUG West found easy to use right away. We just want to pass on some of our recommendations to you.

First come the disks that contain Superscript files of information, articles or sermons. One of the things that is not always obvious when you obtain some of these disks is that you must load Superscript before these disks can be used. Not all of them contain annotated directories. They aren't always necessary on such disks, since the file names themselves usually give a good indication of the contents. Also, it is very easy to use Superscript to just browse through them and see what's there. Also, if you see a file named superspell.tm, ignore it - it just means that the disk provider used Superspell to check some file's word spellings.

8 Sermons #1
10 CBUG Print File
16 Swan's BASIC 4.0+ Tutorial Course - Instructions at the bottom of the Ready-to-print Version menu were a little confusing at first.
16Q Swan's BASIC 4.0+ Tutorial Course - Only available through its author. See the article elsewhere in this issue.
38 Summer 86 Part 1 Print Files
39 Summer 86 Part 2 Print Files - also given CBUG number 52
59 Winter/Spring 1987 Print Files
63 King James New Testament
64 Sermons 2 - Some files formatted for right margin 85
65 Sermons 3 - Ditto

The following Superbase application disk was identified as EASYWARE. The only criticism we had for this disk also holds for most of the other Superbase application disks, namely: the label didn't indicate that the user must first load Superbase before using this disk as the data disk.

49 Medical Finance #2 - Useful help files, but user has to directory the disk to find them

The following utility program disks are useful for the average user and have either annotated directories (a file on the disk identifying the purpose of all the files on the disk) or a SHIFT/RUN start up program.

3 Swan's Utility - Would be nice if the user could obtain a hard copy of the main start up menu.
28 Casey's Scrubber - Warning: Follow all instructions exactly!
36 London Sampler
37 SUPERPRINT - Good collection of programs
47 dFile Database Program - Menu driven. Instruction files are in Superscript (with no indication).
58 Dittinger's Utilities - Needs more instructions

The following programs are for the more experienced user and/or programmer.

12 Scott's B-Mon - For the machine language programmer
27 Goceliak's Gold Mine - Lots of useful information for experts
51 JCL Workshop & Assembler - User must read all of the extensive documentation (true of any such large package).

That's all for now. We haven't finished reviewing all of the library disks available. So next time we'll have more EASYWARE disks to list.

WHAT WE HAVE LEARNED FROM EASYWARE REVIEWING

by Warren D. Swan

CBUG West members have learned some things while reviewing CBUG library disks for EASYWARE. We'd like to pass them on to software authors or those who provide disks of software collected from multiple sources.

1. Provide an annotated listing of the directory of your disk or some other means of telling the user what each file is there for. It may take even an expert several hours just to figure out what that file

   2 "%a.2tx+v3e" seq

   is on the disk for. Do I need it? What uses it? Should I ignore it?
   Which files do I load and RUN? Which files do I Word Result? Which do I Superscript?

2. When possible, include a SHIFT RUN menu as the first file on your disk. Yes, we know it's not always possible or even desirable. We're just making the point that a SHIFT RUN disk is much easier to use.

   Along the same line, if you make a Superbase application program disk, make the start.p on it either a menu, or sufficient instructions to allow the user to know what to do next. The primary reason that we have not identified more of the existing Superbase application disks as EASYWARE is because the disks left the user (even the experienced user) at a loss as to what to do next. Some of them use the standard start.p, leaving the user confused by a prompt for a database name. What should they type in? Some of the disks have a nice menu, the items of which are not self-explanatory enough for the user to know what they mean. A way to combat this is ...

3. Think like a user when you write the instructions for your programs.

   Are the instructions accurate? Some of our reviewers have tried to follow instructions exactly, only to find the computer displaying the "Hyper Inverted Toenail Sort" screen when the instructions said that it would be displaying the "Quadratic Hurricane Spline" menu. When you change the user interface, update the documentation.

   Do you use consistent, non-jargon terms? Don't call something a "form" in one paragraph and then switch to calling it a "screen" in the next. Also, don't use terms that only you know, like funny names for some of the keys.

4. When converting programs from other Commodore systems to our wonderful B machines, try to finish the job. We at CBUG West must decline to consider disks full of (wonderful, I might add) old C-64 programs heaped onto a disk. Take the time to make the program a B program.

   'POKE 59468,12 (or 14) for graphic mode: PRINT "C-64ColorCharacter": PRINT "ONLY 40 COLUMNS OF DATA PER LINE": PRINT "AND THAT IN ONLY UPPER CASE": print "or only in all lower case": POKE WrongLocationForB'sScreen, SomeCharacterValue'

   Syndrome." (Probably the longest named disease ever - even better than pneumono-ultra-microscopic-silicovolcanoconiosis.)

   If you want to write programs that work on all Commodore systems - fine. BUT, computer programs should conform to user's needs, not vice versa.

Please make your disks good for us and for your reputation's sake.

Thank you.

CBUG WEST MEETING SCHEDULE

YOU CAN GET CBUG LIBRARY DISKS FREE OR FOR $1 above the Library's cost

PLEASE NOTE THE NEW MEETING TIME: 7:00 PM

Come join the growing group at CBUG West where we're deciding which CBUG library disks can be certified as CBUG EASYWARE. Show up and take home a disk for review and it's yours FREE. Also, anyone who attends any of the CBUG West meetings can obtain any of the disks reviewed that night for $1.00 (or $1 over the author's royalty for royalty disks).

Anyone can review a disk. Don't worry. The less you know - the better. We don't want experts to decide what disks are usable by us non-experts! Just grab a disk, follow the instructions on its label (if any), plug it in, and see if it makes sense to you. We'll help you, and we need your help.

CBUG West meets on the second Monday of each month at the First Congregational Church, 5th and Main Street, West Dundee, IL. Main street is State Route 72 (Higgins Road). 5th is about 6 blocks west of the Fox River and 2 blocks east of Illinois 31. Meeting time is 7:00 PM. Please disregard the Fox Valley Commodore Users Group meeting and go straight down to the basement where CBUG meets promptly at 7:00.

For information including meeting contents contact Warren Swan at (312) 665-1514 6 to 9 PM (please no later). For weather cancellation queries, contact either Herb Gross (312) 695-1316, or Warren Swan.

CBUG EAST MEETING SCHEDULE

May 22 - Relative Files

Roy Sherman reviewed relative file construction in April. He will continue his lessons on programming relative files for databases and other uses in May, including finding out why things didn"t work as they should have last month.. We'll get you caught up and have you programming in short order, or at least we'll give you a program you can use without knowing anything about how it works. Bring a formatted disk if you want a copy of what we develop at this meeting.

June 26 - Relative Files, Continued

Since we didn't get started right away at the April meeting, we'll need this session to reach a stopping point.

July 24 - Co-Processor Boards

Bruce Faierson will domonstrate the processor boards, which are available for purchase. This much-discussed add-on to the B-128 is now up and running and many of the problems have been solved. See how we can run CP/M 86 programs on our Bs. For those who've been considering purchase of a clone machine, this offers an alternative. We expect a good attendance at this meeting, so get there on time!

August 28 - Library Demonstrations

By this time, a number of the programs in our library should have been reviewed. We will demonstrate some of the more interesting ones and/or some of those of most general use. Copies will be available.

September 25 - Hardware I - one meg board

Vern Kempfer will demonstrate the installation of a megabyte of memory in the B-128. If you've considered dusting off your soldering iron and trying this, come and see how it's done. Vern has done it before, so he knows what he's doing and can show you how to do it and answer questions for those who hesitate to take their computers apart and do things to them.

October 23 - Hardware II - 8050 upgrade, SFD drive switches, etc.

Vern Kempfer will continue hardware demonstrations, showing how to replace the sometimes unreliable Zener reference diodes in the 8050 disk drive and how to install a switch on the SFD drive that will allow switching from 8050 to 8250 format. As time allows, other things Vern has done to his machines will be demonstrated. If you want to bring your drives and soldering irons (and a couple of diodes) to the meeting, Vern will help you do this relatively simple installation.

CBUG East meets at 2:00 pm, on the fourth Sunday of each month except December, although occasional rescheduling is necessary. In case of rescheduling, all on the local mailing list will be notified. We meet at Bethlehem Lutheran Church, Wesley Avenue and Greenwood Street, in Evanston. Greenwood Street is one block north of Dempster. Wesley Avenue is a block west of Asbury (a continuation of Chicago's Western Avenue). Enter on the north side of the building. Parking on the street and behind the church. If you need more details, call Marilyn Gardner at 866-9159.

HINTS & TIPS, ETC.

From Tony Cureton:

How To Disable The RUN/STOP KEY---
In an earlier issue of THE CBUG ESCAPE, Warren Swan gave one method of disabling the RUN/STOP Key in the B128 compute. This method involved the use of the B-128's basic 4+ TRAP command. While this method will disable the RUN/STOP Key in the B128 to some extent if properly used, it is not fool proof. One can often break out of program control just by taping the run/stop key a few quick times. Just as the C-64 has a simple poke Command to disable its Run/Stop Key I knew the ir must be a similar magic poke to perform the same service for the B128. After a little poking around, Presto! My RUN/STOP Key was rendered useless no matter how often I tapped at the key with the following: poke 789,250. To Restore every thing back to normal: poke 789,249. I'm sure most basic programmers will find these two addresses very useful. I also discovered the address to disable the list command. To disable list poke 647,138. To restore the list command poke 647,137. For a incomplete warmstart ' Clears screen ' Gives power up screen and reset's basic pointers: SYS32768. P.S. This routine will not erase any windows previously created. These Commands may or may not work on all issues of B128/B256. I'm reasonable positive they will perform as stated on all Protecto sold B128's. Now for my BIG QUESTION?? How does one disable the RESET BUTTON??

From Kenneth Bright

I have found a fix for bad write protect sensor problems. The original sensors seem to be impossible to find for the MPI drivies. My fix involves removing both the IR emitter and detector from the errant drive and replacing them with a matched IR emitter/detector pair available from Radio Shack. The holes in the disk guides must be enlarged with a drill forthe new sensors. The IR devices will have to be measured before drilling. I successfully performed this modification on an 8050 drive with a bad IR emitter. Sometimes the 7407 on the write protect board may go bad. Check this and connectors on the baord before replacing the IR pair.

(A sure sign of write protect problems is that the buffer inputs will show a change of logic state when a disk is slid past the sensors. The output of the buffers will not changne. This shows that the logic level at the buffer inputus are not high enough to force the gate to change states. A 7417 may be used in place of the 7407. It has a lower collector rating of 15v and works just as well.)

<<Also be sure the IR pair are clean, free from dust and debree. Excessive dust can cloud the lenses producing the same results as bad parts.>>

Ken Bright is a technician with Judge Computer Corporation, 1930 E. Marlton Pike, Cherry Hill, NJ. 08003, 609-751-1767. He reports experience in servicing 8050 drives and other PET/IEEE Commodore products.

From Dick Forman:

I'd like to see department: Battery powered clock calander interfaced with saves and directory (cartridge?)
Patch to allow spellchecker to check document without locating filename from current list (bring up current name first). <<just type in current name or short form 'filen*' instead of scrolling list>>

From CBUG:

Protecto and others have offered the Silver Reed #895P printer. While the instructions are rather clear, the first time on line is a bit of a trick. From Superscript you select printer type 4 -- "Other non-matrix". In the Silver Reed instructions, pages 67 and 68, the instructions state how to switch to print mode (vs typewriter), and then you press "9" while off line to set line feed. Otherwise all printing will be on a single line.

From Jack Lambert:

I had a problem with my 4023 printer having a variable left margin while using Superscript and graphics to make a form. A ten minute phone call to Roy Sherman belayed my fears and resolved my technical problem with my printer. Roy pointed out that by "backing the paper and labels out of the printer" I had crimped the static strip a the base of the print head. 1A fw smoothing storkes with a flat edge solved my problem.

Since Commodore does not emphasize how paper and labels should or should not be removed from their printers, it may be a good ideal to let us novices know that these printers should "ALWAYS FEED FORWARD".

Thanks, Roy! A ten minute phone call saved me the cost of troubleshooting and replacement repairs, not to mention the downtime.

From V.F. Burke

A note of thanks. The fact this note is not on disk or in Superscript indicate there are still some of us loyal dues paying membmers who really appreciate the understanding of Warren Swan, Mark Schwarzbauer, Warren Kernaghan and particulary Bruce Faierson's new column "From the Beginning".

I still have great lack of understand of "mother boards", "parses", emulators", etc. Congratulations to those advanced contributors who are understanding enough to parenthesize simple English definitions of their high tech termonology for those of us who struggle through in hope of someday understanding. Although I'll never understand "Welche Tastaturtabelle" without the (which keyboard table?). Even Goceliak and Deal might not find that "obvious" or "self evident".

Good Luck!

SUPERBASE - MULTI-KEY ACCESS

by Peter Hauke

There's only one quick way of selecting a record, and that is to use the SELECT KEY option. The other options are too slow for normal purposes. So what do you do if you want to access your records in a variety of different methods? Consider the following: You have decided to put your club's membership onto a Superbase file. Because of possible duplications you decide to give each member a membership number and use this as the key. The other fields include surname, forename, address fields, telephone, date joined, subscription, etc. Once a month, or so, you print out a list of members in membership number order and one in surname order. If a member calls you up, all you do is ask for his membership number, tap it into the computer and up pops his record. If he does not know his number then all you do is look down the surname list, find his name, get the number and tap that in. Of course, you may need more than two lists. You may have been talking to somebody on the phone and he gave you his telephone number, but you forgot to take his name. A list in phone no. order would solve that problem.

In general, you can print out as many lists as you want (paper, time and money permitting) to access your data in different ways. As long as your database is reasonably static then there is little to worry about. A list once a month will suffice. New members can be added to the end of each list as they join. But what do you do if members are joining and leaving more often? One solution is to print out lists more often, once a week instead of once a month. This becomes time consuming, though, and once you start a print run it is not normally acceptable to interrupt it. You could buy a faster printer or even upgrade all your equipment and buy a database with multi-key access. But none of this is necessary. Multi-key access is available with Superbase! It is not easy though. It takes quite a lot of programming and careful design, but at the end of the day, it is worth it, as not only will your computer suddenly become more interactive but it will also save you money - you won't need to print out so many lists for a start!

The theory: Let's design a simple database which holds membership details and allows you to access any record by either the membership number or surname. The membership number is limited to four characters. Firstly set up a file called 'members' and a second file called 'names'. The members file contains all the usual fields and uses the membership number as the key. The

names file has a key field and three other fields: membno, surname and forename. The names file key field is made up of a combination of the first ten characters of the surname and the membership number. This ensures that each record is unique. Of course you could use fewer characters of the surname as this saves memory. Records are entered into the database under program control (see program enter.record). The program sets up two records; one in the members file stored with the membership number as the key; and one in the names file stored with the surname/membno as key. Access to any record is also done under program control (see program select.record). Selecting by membership number is exactly the same as normal direct SELECT KEY and takes exactly the same time. To select by surname is a little more complicated. Firstly you tap in the surname. The names record appears and you have to confirm that you have the right record. If there are several people with the same surname then you can SELECT NEXT or PREVIOUS until you locate the correct record by checking the forename field.

And now finally, the record formats and programs. Both files are set up with no duplicate keys. The programs are reasonably straightforward. Although this was set up on a C128/1570 system, it should work on any other Commodore and Superbase setup with only very slight modification.

File Def: club/members

	name	type	len
1	membno	Key	4
2	surname	Txt	20
3	forename	Txt	20
4	add1	Txt	24
5	add2	Txt	24
6	town	Txt	24
7	county	Txt	16
8	postcode	Txt	8
9	telephone	Txt	16

File Def: club/names

	name	type	len
1	key	Key	14
2	membno	Txt	4
3	surname	Txt	20
4	forename	Txt	20

```
1 rem save "enter-record":menu
2 ask &4"membership number"m$:ask &20"surnames"$:ask &20"forename"f$
3 file "members"
4 clear:[membno]=m$:[surname]=s$:[forename]=f$:store
5 file "names"
6 clear:[key]=left$(s$,10)+m$:[membno]=m$:[surname]=s$:[forename]=f$:store
7 file "members"
8 select m$:select r
```


USING MULTIPLE KEY NAMES IN SUPERBASE

Reprinted by permission of our friends at the INDEPENDENT COMMODORE PRODUCTS USERS GROUP, London, England; Jim Kennedy, Chairman

Note: These articles are based on Superbase for the C128, etc. Commands are virtually identical for the B128 versions and B128 Superoffice.

```
 9 ask &1"y to enter another member"q$:if q$="y"then 2

 10 rem save select-record:menu
 12 ask &1"m/s to select by Member Number or by Surname"q$
 14 if q$="m"then gosub 20:goto 18
 16 if q$="s"then gosub 26:goto 18
 18 ask &1"y to select another"q$:if q$="y"then 12:else menu
 20 file "members"
 22 ask &4"membership number"m$:select m$:nmat 48
 24 select c:return
 26 file "names"
 28 ask &10"surname (max 10 chars)"s$:select s$:nmat 48
 30 select c
 32 ask &1"c/n/p - for correct, next, previous record"q$
 34 if q$="n"then select n:goto 30
 36 if q$="p"then select p:goto 30
 38 if q$<>"c"then return
 40 m$=[membno]
 42 file "members"
 44 select m$:nmat 48
 46 select c:return
 48 display @0,7@0,147"NO SUCH RECORD EXISTS":return
```

Once you have set up the database as above and entered the programs you will be able to enter your member's details onto the database. You will then be able to access each member directly from his membership number or his surname. Of course if your system has enough disk space then you can add further keys if you want. All manipulations of the database must really be done under program control. You can imagine the problems that would follow if you simply went into the members file and changed the spelling of a member's surname! At least two more programs are required; delete.record and edit.record. You may also need check.data as well. You would run this program once in a while to ensure that records existed in both files.

One of the advantages of using Superbase is its flexibility. Unfortunately a little bit of flexibility is lost when you use multi-key access. It is therefore very important that all your programs are properly tested before you start to enter masses of records. Keep the programs simple and short. It is better to write several small programs to do separate jobs even if this means that you are duplicating code.

***

SUPER* CORNER

edited by Peter Hauke

And so the New Year brings a new editor to the Super* Corner. For those of you new to ICPUG, Super* Corner is a regular ICPUG column about the database and wordprocessor programs produced by Precision Software. These include: Superbase 64, Easyscript, Superscript 128, Superbase 128, Superoffice 8096 (Superbase and Superscript on one disk) and any other "Super" programs produced by Precision.

A short history lesson? I can still remember back to the days when I first purchased a Commodore 64 and Superbase and sat in front of a small portable telly designing my first database. Eventually I discovered ICPUG. Bits of advice gleaned from those first ICPUG magazines set me on the correct road. Seeing programs written by others made me look at my own with a different critical eye. New techniques were incorporated. Better presentation. Easier to use programs were the result. Several years and several machines later (C64 to 8296 to C128 and almost to Amiga but not quite) I now find myself editor of the Super* Corner and hopefully in a position to help others in a similar manner.

But first my address. Do not hesitate to contact me by writing on any query regarding a Super* program:

Peter Hauke
1 Montpelier Road
Ealing
LONDON
W5 2QS

A stamp addressed envelope will always elicit a prompt reply. Superbase articles should be sent to me. Please, always send a disk with the article on it and preferably a working demo of the program if relevant. A printout of both the article and any programs would also be appreciated. And please do send in your queries and tips. If you have a problem, chances are that somebody else has already solved it. And if you have solved one, then chances are that somebody else still has the problem.

You may notice at the end the new title Superlink. Superlink is simply the term I shall use for switching between Superbase and Superscript when both are resident in memory. But enough rambling, let's get started...

USING MULTIPLE KEY NAMES IN SUPERBASE

Reprinted by permission of our friends at the INDEPENDENT COMMODORE PRODUCTS USERS GROUP, London, England; Jim Kennedy, Chairman

Note: These articles are based on Superbase for the C128, etc. Commands are virtually identical for the B128 versions and B128 Superoffice.

Superbase

Firstly, there are a couple of small errors in the programs written for the Superbase Multi-Key Access article in the last ICPUG magazine. These resulted from my omission to replace double quotes with two single quotes — apologies to Bill Bremner for the extra work. To move on, line 2 of the first program has the fourth double quote after the 's' whereas it should be before reading "surname"s$. Line 10 of the second program should have double quotes around like so, "select-record".

At present there are no outstanding Superbase queries (indeed any Super* queries at all) so let's start off with a few short programs and techniques to improve the presentation and ease of use of Superbase databases.

Firstly always try to dedicate a disk to each database (together with a backup disk or two). I rarely if ever have more than one database on a disk. When designing a file format for a database file I always bear in mind the amount of memory a record will take up. Look up in the appendix of your Superbase manual and calculate how many bytes the maximum record will take up. Superbase does not store trailing spaces. For instance, in an address field of up to 20 characters, Superbase only uses 7 bytes for the field if you entered LONDON (6 for London and 1 for the field separator). Furthermore, if the maximum record size is 140 bytes, including separators, but the average size is about 105, then most records will be below the 123 limit for one half-block per record. If not, then you may want to consider a different approach to your file format design should you want to maximise the number of records.

I always include a start.p program on every data disk and I always keep all my programs on the same disk as the database itself. But what to include in the start.p program? Very little as you can see from the following program:

```
  1 goto 10
  2 save "start":menu
 10 rem
100 date "00JAN00"
200 database "youthclub"
300 load "menu"
```

Either on the command line or in prog mode, simply type run 2 and the program will automatically be saved to disk. Line 100 simply sets the date format for Superbase to English format. date "JAN0000" would set the date format to American. Line 300 loads a program on the disk called menu which is a loader program for all the programs written for that database. As you can see the program is very simple. This makes it easy to follow, easy to debug

and easy to tailor to another application. Short programs are easy programs. It is useful to include the system parameters in the start program. The appendix to the Superbase manual has an explanation of how to set the margins, paper lengths, etc.

Note that line 10 is always a rem. I include these first three lines in any program I write. If it were a program line then it might be deleted by mistake and then crash the next time it was used. And so for the menu program:

```
  1 goto 10
  2 save "menu":menu
 10 rem
100 display @0chr$(147)
110 display @10, 7@+"0"" Superbase Menu"
120 display @10, 9@+"1"" Enter Record"
130 display @10,11@+"2"" Select Record"
140 display @10,13@+"3"" Sort Records"
150 display @10,15@+"4"" Print Records"
200 wait wt$
300 if wt$="0"then menu
310 if wt$="1"then load "enter-record"
320 if wt$="2"then load "select-record"
330 if wt$="3"then load "sort-record"
340 if wt$="4"then load "print-record"
400 goto 200
```

A very simple program to do a very important job. CHR$(147) is the code for clearing the screen. The program is self explanatory. It can easily be tailored to use the function keys instead of numbers (check the back of your Commodore manual for the character codes) but this program should run on any Superbase system with no alterations necessary. It is always best to write programs which are understandable and if possible keep them as short as possible. When you come back to the program several months later it helps if you can follow the program easily.

After selecting an option the program loads in the next program (this is called chaining) and runs it. When exiting from the second program back to the menu program, you should include LOAD"menu" in place of MENU. but be careful, when developing a program, always leave the line as MENU and only change it when the program has been fully debugged and saved to disk. I have lost many a good program by testing a chained program and then returning to the menu program without first saving the tested program!

USING MULTIPLE KEY NAMES IN SUPERBASE

Reprinted by permission of our friends at the INDEPENDENT COMMODORE PRODUCTS USERS GROUP, London, England; Jim Kennedy, Chairman

Note: These articles are based on Superbase for the C128, etc. Commands are virtually identical for the B128 versions and B128 Superoffice.

ESCENSIA de VIDA

COL.J.E. O'HALLORAN
Rt 2 Owl Creek Road
Hiawassee, Ga 30546

TAX SOFTWARE & SERVICES
Phone 404-896-4342

Mr. Norman Deltzke
CBUG, Inc.
4102 N. Odell
Norridge, Il 60634

20 February 1988

Dear Norm,

I was both surprised and pleased to see that you had published my letters in the last ESCAPE. Thank you.

In the letter pertaining to my Income Tax Program, I mentioned the added in planning ability for the 1988 tax year. This past week I received the final (maybe) information on the TAX RATE SCHEDULES for 1988 which did NOT even resemble the information which had been disseminated in December. Consequently, I spent about ten hours redoing the TAX RATE SCHEDULES in my various programs. If you think that dealing with the IRS once a year on your tax forms is a pain in the neck you should see what it is like to deal with their many changes in forms and information put out as 'proofs' starting almost immediately after the 'tax season'. They design NEW FORMS before they even read or understand the last changes in the tax code which means that trying to get a program ready early means oodles of repititious, last minute and even too late revisions. It galls me to think that we PAY for this kind of boondoggle in every one of the thousands of bureaucracies in Government.

As a result of this foulup, I will be sending a new set of disks to each purchaser of my 1987 Income Tax Program---FREE. Those who did not buy the 1987 program may obtain the 1988 Tax Year Planning Program for $40.00 with any necessary updates for tax preparation in early 1988 for an additional $20.00.

I also have a program for the Federal 1040A for those who would rather not manually calculate their tax. The 1040A is on two discs, runs under CalcResult and costs $10.00.

I plan to also have 1988 programs for users of IBM PC/ATs (80286) and compatibles. They will be on either 5.25"/1.2MB or 3.5"/720K formats. These will run under Lotus 1-2-3, Quatro or other programs which read 1-2-3. I expect the prices to be in the range of the programs for the B128/1024.

Inasmuch as I have another enterprise licensed in Georgia under the ESCENCIA de VIDA logo, I have incorporated my tax program endeavour into that one, hence the new letterhead. I feel that it is more compatible with ESCENSIA de VIDA (a service and distributor function) than with HORSAO FARMS (a Quarter Horse breeding effort).

I enjoyed the last ESCAPE very much, as is the case with all of them, but I miss the small issues of CBUG which used to come between ESCAPES. I will not, however, complain since I marvel at the superhuman effort put out by those responsible for the publication of the ESCAPES. You do good work.

Sincerely,

Jim

Lowell Breunig
Estacada Oregon

U N D E R L I N E

It is possible to UNDERLINE with the 4023 Printer, using Superscrip II, even if the manual states that it can't be done. The secreat is to use the secondayra address (sa6,xx). Secondary address 6 sets the line spacing and by mannipulating the line spacing it is not hard to underline. Line spacing can be set to zero using (sa6,1), which will cause the printer to print over the same line. If the scecondary address is set to (sa6,36), then normal spacing is resumed.

*sa6,1 (set's to zero spacing) & *sa6,36 (set's to normal spacing)

The steps to uderline are as follows:

HEADINGS

*sa6,1;This is a test(ret) Using secondary address sa6,1. ill change the line spacing to 0, and prints the heading. Don't forget the simicolen.

*sa6,36;_____(ret) On the following line set the scondary address back to normal spacing and press the CTRL Key and tap the four Key to get underline character.

*sa6,1;HEADINGS
*sa6,36;_____

INSIDE OF TEXT

The method to underline in text is somwhat harder but not to bad once a tactic is worked out. I have found that it is best to type in the text as you wish it to be and then go back and manipulate the uderling proceedure. The folowing is and example. We will underline the words underline, porject and develop.

PARAGRAPH AS FIRST TYPED

This is a test to see if I can get it to uderline the data I want to print. This is a very hard project and will need to be worked on some more. Will you try to help. I would like to develop this very much as I may have a very great need for this in the future.

SET UP FOR UNDERLING

```
*sa6,1;          test                                  uderline
*sa6,36;This is a ____ to see if I can get it to _________ the data I want to
*sa6,1;                              project
*sa6,36;print. This is a very hard _______ and will need to be worked on some
*sa6,1;                                        develop
*sa6,36;more. Will you try to help I would like to _______ this very much as I
may have a very great need for this for now.
```

The procedure is as follows, insert a new line starting with *sa6,1, and the words that are to be underlined typed in their positions just above where they occured in the text. Type in a return

Now goto the original line and at the start of the line insert *sa6,36 and replace the words to be underlined by holding down CTRL Key and typing in the uderline chacter on the four Key. Go to the position on this line that will start the next line and type a return. Follow this proceedure through all the text you wish underlined. The above paragraph now with uderlining.

This is a test to see if I can get it to underline the data I want to print. This is a very hard project and will need to be worked on some more. Will you try to help I would like to develop this very much as I may have a very great need for this for now.

The first word of a paragraph must be shifted one space to the left, I don't know why, it just the way it works.

COMMERCIAL ADVERTISING

SERVICE CENTER:

TYCOM INC
503 East Street
Pittsfield, MA 01201
(413) 442-9771

Authorized Commodore service center, we repair B128 8032, 4023, 8023, 6400, 4040, 8050, 8250, 2031, C-64, C-128, etc. Rates are $50/Hr plus parts. We will diagnose & quote repair costs for $25. Normal turn-around time is 5 business days. We ship via UPS. Limited supply 8032's and 4023's (new) at $189 each.

* *

Attention CBUG'ers:

NWM Inc. has found the company that bought out all of the CBM 8023p printers from Commodore. If there is anyone that still desires to purchase one of these incredibly well built units there is still time. The company has been converting the units to Centronix but still have some new IEEE units left. Unfortunately they will not sell in small quantities, so an effort to determine interest is necessary. If there is enough interest and advance orders we will attempt to make the purchase. Do not hesitate because these units and other devices are being shipped overseas. CALL NWM, INC. AT 312-520-2540 FOR FURTHER INFORMATION.

* *

SPECIAL TRANSFER SERVICE OFFER

1.) FACILITIES -- Transfer Service

** Transfer programs and files from 1541, 1571 & 1581 to 8050 format, from 8050 to 1541, 1571, 1581 format.

** Convert some program files to sequential files.

** Transfer basic programs & sequential files from C64, C128 & B128 to PC type equipment on either 360K, 5 1/4 disks or 720K, 3 1/2 disks.

** Transfer basic program as sequential files from PC to CBM.

E.L Rhyner, 4852 N. Karlov, Chicago, Il. 60630
MINIMAL CHARGES. 312-286-7901 3:30pm to 6pm, 7pm to 10:30pm CST

* *

ADDITIONAL WANT ADS

1.) FOR SALE: 1 B-128/256 to IEEE-488 cable $26; BRAND NEW: $7.50 each Superscript II & Superbase 1 (2 of each on hand): Warren D. Swan, 1 N 114 Woods Avenue, Wheaton, IL, 60188, (312) 665-1514

2.) B-128 systems $345, w/4023 $445; B & W monitors $69; 14" Samsung color monitor $99. Daisywheel printers: CBM 6400 (C.ITOH F-10) w/s feeder $495; CBM 8300P (Diablo 630) $295, sheet feeder $295; NEC Spinwriter 5510 w/s feeder $295. ADA 1800 interface $49. C/BUG/IEEE-488 interface $25. Word Result $15. Superdisk Doc $10; SSII, A/C, A/P, payroll, order entry, inv. mgmt. $5. 605-642-2622.

3.) Expanded B-128-80 system, $300. $625 with 6400 printer. Used 1-1/2 years Call 701-949-2792 (ND) for details. David Medalen.

4.) B128 LoO-Pro system: 8050 - Drive 0 Disk Id Mismatch; Hi-Res Grn Mon; IEEE to Cen Par Print Interf (no printer); all cables; softw. Little use. Margaret Roytek. Eve: 313 569 5208.