C.U.G.S MONITOR
APRIL (FOOL) ISSUE

Vol. 3 Issue 4

April 1988

OBLIGATORY STUFF

PRESIDENT	Richard Maze	586-3291
VICE-PRESIDENT	Barry Bircher	359-1925
TREASURER	Harry Chong	
LIBRARIAN	Earl Brown	543-2068
ASS'T LIBRARIAN	Steve Bogues	
EDITOR	Ken Danylczuk	545-0644
ASS'T EDITOR	Greg Rezansoff	
MEMBER AT LARGE	Gord Williams	

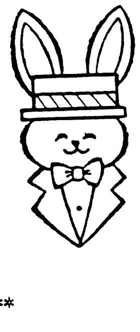
THE MONITOR is published monthly by the COMMODORE USERS' GROUP OF SASKATCHEWAN (CUGS), Regina, Sask., Canada. CUGS meetings are held at 7 pm the first Wednesday of every month (unless otherwise noted) in the North-West Leisure Centre, corner of Rochdale Boulevard and Arnason Street.

Anyone interested in computing, especially on the C64, 128 or 64C, is welcome to attend any meeting. Out of town members are also welcome, but may be charged a small ($5.00) mailing fee for newsletters. Members are encouraged to submit public domain software for inclusion in the CUGS DISK LIBRARY. These programs are made available to members. Any member is entitled to purchase DISKS from our public domain library for a nominal fee. Programs are 'freeware', from computer magazines, or the public domain. Individual members are responsible for deleting any program that he/she is not entitled to by law (you must be the owner of the magazine in which a particular program was printed). To the best of our knowledge, all such programs are identified in their listings. Please let us know if you find otherwise. Contact Earl Brown, 737 Rink Ave.

CUGS is a non-profit organization comprised of C64, 64C, C128, and 128D users interested in sharing ideas, programs, knowledge, problems and solutions with each other. The more members participate, the better the variety of benefits. Membership dues are pro-rated, based on a January to December year.

IN THIS ISSUE

MEETING PLACE:



AGENDA:

DATABASE POTPOURRI - Richard and co. present a batch of the best!

*******coffee****visiting****disk-picking***********

POTPOURRI CONTINUED - DBasing on the 128 and DBasing on a small budget

EDITORIAL:

Since I joined the C.U.G.S. executive 2 years ago, I've been impressed by the general concern on the part of all executive memebers to keep our meetings and our club's offerings up-to-date, practical, interesting and novel. A look back at our meeting agendae will show that clearly. And we ain't done yet! Next month we will be trying something new (again!) for our club members (at least in the past few years).

In the early days of computers (ESPECIALLY COMMODORE computers), software was sparse, even non-existent. Those dedicated few who had machines naturally met together to share their discoveries about their machines (in the absence of decent operating instructions) AND to share software (always in short supply). Thus were planted two seeds - the roots of the COMPUTER USERS' GROUP - a group of users gathering together for growth in understanding (just like us!); and the first attempts at a SWAP SESSION (exchanging user/public domain software for the benefit of all concerned).

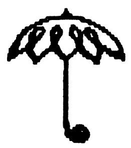
In the past couple of years commercial software has increased in quantity (and quality) to the point where SWAPPING is no longer NECESSARY to obtain a varietyof useful programs. This proliferation of commercial material has had a two-fold negative effect on what should be a time-honoured tradition of us computer lovers. First, the vast expanse of brightly-packaged, advertised, and reasonably-priced software has hidden (NOT eliminated) the work of the PUBLIC DOMAIN programmer from the general computer user. The programs are still being written - they're still being shared - but not as evidently as 6 years ago.

Also, because computers became so widespread, a different kind of computer devotee appeared - the "cracker" (NOT "hacker" as they'd like to be called) - a person dedicated to pirating software and "sharing" it with friends. "Sharing" software took on sinister overtones of devious, arrogant computer wiz's distributing the latest commercial game! This negative aspect also did much to "hide" the legitimate exchange of decent, public domain material.

But the exchange of good, "home-made" computer software IS seeing a revival! PUBLIC DOMAIN software is original work of a programmer which has been given for distribution to anyone interested. FREEWARE and SHAREWARE are variations on PUBLIC DOMAIN software; they are programs offered to the public freely, with the request that anyone trying the work and finding it useful send some amount (usually nominally small) to the author. Most public domain/shareware material carries the restriction that the work MUST NOT be sold, or used as part of works which will be sold commercially.

Anyway, I seem to be rambling. Next month's meeting of C.U.G.S. should be fascinating for anyone interested in extending their software library. C.U.G.S. will operate their first (annual?) SWAP NIGHT for members only! Admission will be a membership card! We'd like to encourage everyone to bring along some blank disks AND some PD software to swap. (No disks - no swaps - short evening!) Please try to avoid bringing material currently in our library (no fair!) and be prepared to "share" it with our librarian, Earl. Once it's checked and sorted it'll be added to our regular disk library! This evening could be the greatest thing since C.U.G.S., or a night we'd rather not remember - it all depends on you! (Gee, sounds like a great line for a song title!) See you next month - signed Trader Dan!

RICHARD'S EASTER ENLIGHTENMENT



The MAY meeting of CUGS will be held WED. MAY 4.

This meeting will be set up as an opportunity for CUGS members to share public domain C64/128 software.

Each member is requested to bring a disk(s) containing your favorite public domain program(s) and a supply of blank disks (limited supply for sale at meeting @ $1.00 each).

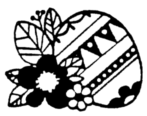
Computers and dual drives will be made available for copying disks. A copy of your program(s) will be made for inclusion in the CUGS library.

To speed up the copying process, we ask that all disks be in 1541 or single-sided 1571 format.

Club members may bring commercial software for sale or trade, as long as the software is original software only and they assume full responsibility for the software (we won't have a supervised sale table or area).

Since this activity is for CUGS members only, you may be requested to show your membership card. (Memberships will be available at the meeting).

Coffee, cookies, conversation and advice will be provided.



BIRCHER ON BITMAPPING!

by Barry Bircher

What is multicolor bitmap mode???
B.Bircher

The VIC Chip sees graphics in 4 parts and reconstructs a picture on the screen. The first (main) part involves 8000 bytes called the BITMAP. The chip looks at the current character position within the BITMAP, and extracts the byte it wants to scan on screen. It breaks the byte into 4 2-bit NIBBLES. Using the 2 bits (corresponding to 2 pixels on the screen), it searches the colour it is to display using the codes held in the nibbles. Since 2 bits are able to count to 4, we are able to display 4 colors within a character position. The nibble code simply tells the chip where it is to get the colour source code corresponding to that character position. We therefore need to store the colour codes to be displayed in another part of memory (within the chip's addressing range). The chip has only 14 lines to address with, so it can only access 16K at a time. What this means is that our info needs to be within that range.

The background byte we are most familiar with (53281/$D020) we will call Color %00. It is common to all screen positions and must be one of the 4 colors to be displayed. Since color ram (1000 bytes from 55296/$D800 to 56295/$DBE7) is not movable, we can count on this being here all the time. We will call this Color %11. This accounts for 2 colors... so where do the other 2 come from??

There are 1000 bytes (usually below the bitmap ram) that are encoded to hold 2 color codes within one byte (since 4 bits can count to 16 we only need 4 bits per color). The upper 4 bits hold the code for what we will call Color %01, and the lower 4 bits Color %10.

So, there we have it - Color 00 01 10 11 - 4 colors. Now, back to the bitmap. This is the heart of the multicolor mode. The VIC chip looks at the bitmap and determines the 2 pixel group it is to display. It decodes the 2 bits and displays the color pointed to by the nibbles.

```
  00 = display color in Background register.
          (53281/$D020) 1 byte
  01 = display color in upper 4 bits of color map
          (1024/$0400) 1000 bytes
  10 = display color in lower four bits
          (1024/$0400)   "    "
  11 = display color in color ram
          (55296/$D800) 1000 bytes
```

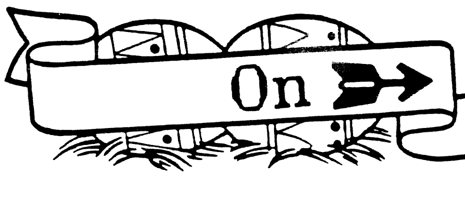
The bitmap as well as the color bitmap is able to be moved around to one of 4 video banks seen by the VIC chip. The background and color ram cannot be moved, so they stay put. A KOALA picture file consists of a first byte of unknown use (if you know what the first byte does, please let me know!); a second byte to hold the border color, then the 8000 byte Bitmap, 1000 byte color bitmap, 1000 byte color ram, then the background color byte (10,003 bytes total).

KOALA READER (BASIC 128)

```
100 BANK15
110 CATALOG"?PIC*"        :REM SELECT AND OPEN FILE
120 PRINT"CURSOR TO FILE & HIT RETURN"
    :INPUTA$:A$=MID$(A$,7,15)
130 OPEN2,8,2,A$
140 GET#2,A$,A$:POKE 53280,ASC(A$):REM LOAD BORDER COLOR
150 GRAPHIC3,1
160 FORJ=DEC("2000")TODEC("3F3F"):REM LOAD 8000 BYTE BITMAP
170 GET#2,A$:POKEJ,ASC(A$):NEXT
180 FORJ=DEC("1C00")TODEC("1FE7"):REM LOAD 1K COL. BITMAP
190 GET#2,A$:POKEJ,ASC(A$):NEXT
200 POKE216,255                   :REM DISABLE SCREEN IRQ
210 POKE1,PEEK(1)AND 254          :REM COLOR BANK (2 AVAIL.)
220 FORJ=DEC("D800")TODEC("DBE7"):REM POKE TO COLOR MEMORY
230 GET#2,A$:POKEJ,ASC(A$)AND15  :REM LOAD BACKGROUND COLOR
240 NEXT:POKE1,PEEK(1)OR1:POKE 216,160
    :REM VBANK1 AND ENABLE SCREEN.IRQ
250 CLOSE2
```

KOALA PICTURE DISPLAY (BASIC+ML 128)

```
10 BANK15:GRAPHIC3,1:GRAPHIC0      :REM CLR SC & MOVE BASIC
20 BLOAD"KOALA-R-ML",D0,P4864
30 PRINT"PLEASE INSERT KOALA DISK":GETKEY A$
40 CATALOG"?PIC*"
50 PRINT"CURSOR UP TO FILE AND HIT RETURN"
60 INPUTA$
70 A$=MID$(A$,7,15)
80 OPEN2,8,2,A$
90 GET#2,A$                        :REM ???
100 ~~GET#2,A$:POKE 53280,ASC(A$)~~  ~~:REM BORDER~~ COLOR
110 GRAPHIC3,1:SYS4864             :REM M.L. TO DUMP FILE
120 GET#2,A$:POKE 53281,ASC(A$)    :REM BACKGROUND COLOR
130 CLOSE2                         :REM CLOSE FILE AND WAIT
140 GETKEYA$:GRAPHIC CLEAR
150 POKE 53280,13:POKE 53281,11:POKE 241,13
    :REM RETURN TO DEF. COLORS
160 PRINT"DISPLAY ANOUTHER? Y/N"
170 A$="Y":INPUT A$
180 IF A$<>"Y"THEN CLR:END
190 GOTO 40
200 END
```



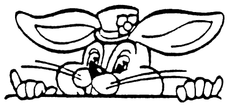
KOALA PICTURE DISPLAY (BASIC+ML 64)

```
10 REM                          :LOAD ML CODE AT 49152
20 IFA=0THENA=1:LOAD"KOALA-R64-ML",8,1
30 REM
40 REM  SET START OF BASIC TO 16384 AND POKE START ZERO BYTE
50 REM
60 IFA=1THENPOKE43,1:POKE44,64:POKE16384,0:CLR:A=2
  LOAD"KOALA DRIVER64",8
70 PRINT"PLEASE INSERT KOALA DISK"
80 GETA$:IFA$=""THEN80
90 GOSUB240                     :REM ENABLE COLOR BITMAP
100 OPEN2,8,2,"?PIC H*"         :REM * * * YOUR FILENAME HERE
110 GET#2,A$                    :REM ???
120 GET#2,A$:POKE 53280,ASC(A$):REM BORDER COLOR
130 SYS49152                    :REM M.L. TO DUMP FILE
140 GET#2,A$:POKE 53281,ASC(A$):REM BACKGROUND COLOR
150 CLOSE2:POKE198,0            :REM CLOSE FILE AND DISPLAY
160 GETA$:IFA$=""THEN 160
170 POKE 53280,14:POKE 53281,6:POKE 646,14
    :REM RETURN TO DEF. COLORS
180 GOSUB 270                   :REM DISABLE BITMAP
190 PRINT"DISPLAY ANOTHER? Y/N"
200 A$="Y":INPUT A$
210 IF A$<>"Y"THEN CLR:END
220 GOTO 70
230 REM POKES TO ENABLE BITMAP MODE
240 POKE 53265,PEEK(53265)OR32
250 POKE 53270,PEEK(53270)OR16
255 POKE 53272,PEEK(53272)OR8
260 RETURN
270 REM POKES TO DISABLE BITMAP MODE
280 POKE 53265,PEEK(53265)AND223
290 POKE 53270,PEEK(53270)AND239
295 POKE 53272,PEEK(53272)AND247
300 RETURN
```

KOALA M.L. L.A.D.S. SOURCE 64

```
10 *= $C000
20 .S
30 .D KOALA-READ64-ML KOALA-R64-ML
40              LDX #2      :JSR $FFC6
50              LDA #$20    :STA STOREIT+2
60              LDA #$00    :STA STOREIT+1
70  BITMAP      JSR $FFE4
80              JSR STOREIT
90              LDA STOREIT+2:CMP #$3F      :BNE DONE1
100             LDA STOREIT+1:CMP #$40      :BNE DONE1
110             JMP COLORMAP
120 DONE1       JMP BITMAP
130 COLORMAP    LDA #$04    :STA STOREIT+2
140             LDA #$00    :STA STOREIT+1
150 COLORBIT    JSR $FFE4
160             JSR STOREIT
170             LDA STOREIT+2:CMP #$07      :BNE DONE2
180             LDA STOREIT+1:CMP #$E8      :BNE DONE2
190             JMP COLORRAMO
200 DONE2       JMP COLORBIT
210 COLORRAMO   LDA #$D8    :STA STOREIT+2
220             LDA #$00    :STA STOREIT+1

230 COLORRAM    JSR $FFE4
240             JSR STOREIT
250             LDA STOREIT+2:CMP #$DB      :BNE DONE3
260             LDA STOREIT+1:CMP #$E8      :BNE DONE3
270             JMP SET216
280 DONE3       JMP COLORRAM
300 STOREIT     STA $2000
310             INC STOREIT+1:BNE DONE
320             INC STOREIT+2
330 DONE        RTS
340 .END KOALA-READ64-ML
```



KOALA M.L. L.A.D.S. SOURCE 128

```
10 *= $1300
20 .S
30 .D KOALA-READER-ML KOALA-R-ML
40              LDX #2      :JSR $FFC6
50              LDX #$FF    :STX #216
60              LDA #01     :AND #$FE      :STA #01
70              LDA #$20    :STA STOREIT+2
80              LDA #$00    :STA STOREIT+1
90  BITMAP      JSR $FFE4
100             JSR STOREIT
110             LDA STOREIT+2:CMP #$3F      :BNE DONE1
120             LDA STOREIT+1:CMP #$40      :BNE DONE1
130             JMP COLORMAP
140 DONE1       JMP BITMAP
150 COLORMAP    LDA #$1C    :STA STOREIT+2
160             LDA #$00    :STA STOREIT+1
170 COLORBIT    JSR $FFE4
180             JSR STOREIT
190             LDA STOREIT+2:CMP #$1F      :BNE DONE2
200             LDA STOREIT+1:CMP #$E8      :BNE DONE2
210             JMP COLORRAMO
220 DONE2       JMP COLORBIT
230 COLORRAMO   LDA #$D8    :STA STOREIT+2
240             LDA #$00    :STA STOREIT+1
250             LDA #$FF    :STA $D8
260             LDA $01     :AND #$FE      :STA $01
270 COLORRAM    JSR $FFE4
280             JSR STOREIT
290             LDA STOREIT+2:CMP #$DB      :BNE DONE3
300             LDA STOREIT+1:CMP #$E8      :BNE DONE3
310             JMP SET216
320 DONE3       JMP COLORRAM
330 SET216      LDA #160    :STA $D8
340             LDA $01     :ORA #$01      :STA $01
350             JSR $FFCC   :RTS
360 STOREIT     STA $2000
370             INC STOREIT+1:BNE DONE
380             INC STOREIT+2
390 DONE        RTS
400 .END KOALA-READER-ML
```

prizesprizesprizesprizes

At each CUGS Meeting during 1988, there will be a computer-generated draw for a winner of a prize.

RULES:

Paid-up members for 1988 ONLY are eligible.

Draw is made at the end of each meeting.

The winner must be present to claim the prize. If a member NOT present is drawn, the draw will be made again until a winner is found.

Prizes are to be accepted "As is" no substitutions permitted!

The membership list will be updated at the break and new members will be eligilble.

Prize for April: THE HOME MANAGER (program)

March winner: BARRY BIRCHER

prizesprizesprizesprizes

Sir Richard's BASIC:

by Richard Maze

In last month's article I examined the procedure involved in creating a relative file. The follow-up to that article is a program that actually uses the relative file that was created. The program that follows is a simple, no-frills relative file handler. It assumes that the file 'ADDRESS' has already been created on disk. This program, along with the one from last month, would constitute the start of a file-handling system. All that needs to be added is more flexibility in creating the file and the ability to report to paper.

```
100 REM USE RELATIVE FILE ** ADDRESS **
130 REM 200 RECORDS
140 REM RECORD LENGTH = 83 CHARACTERS
170 REM SET VARIABLES
180 : POKE 53272,23:REM LOWER CASE
190 : POKE 53280,15:POKE53281,15
200 : DN$="HOME CRSR DOWN 23 TIMES"
210 : PD$="+<24 SPACES"
220 : BL$="<25 SPACES>"
230 : DIM LA$(6),DA$(6):CR$=CHR$(13)
270 REM TITLE
290 : GOSUB 2240
310 REM >>> MAIN LOGIC <<<
330 : REM OPEN FILE
350 :   OPEN 15,8,15:REM COMMAND CHANNEL FIRST
360 :   OPEN 2,8,2,"0:ADDRESS"
380 : REM GENERAL SET-UP
400 :   GOSUB 620
420 : REM MENU
440 :   PRINT "CLR"LEFT$(DN$,9)TAB(6)
         "BLACKWHAT OPTION?  REDSELECT NUMBER"
450 :   PRINT TAB(10)"DOWN RED 1 BLUE  SEE
         /CHANGE A RECORD"
460 :   PRINT TAB(10)"DOWN RED 2 BLUE  ADD A RECORD"
470 :   PRINT TAB(10)"DOWN RED 3 BLUE  QUIT"
480 :   GET G$:IF G$="" GOTO 480
490 :   G=VAL(G$):IF G<1 OR G>3 GOTO 480
500 :   ON G GOSUB 840, 1400, 530
510 :   GOTO 440
530 : REM QUIT
550 :   CLOSE 2:CLOSE 15
560 :   PRINT "CLR"
580 END
590 :
600 REM GENERAL SET-UP
620 : REM GET FIELD LABELS
640 :   FOR  I = 0 TO 6
650 :  READ LA$: LA$(I)=LEFT$(LA$+BL$,13)
660 :   NEXT I
680 : REM FIND EMPTY RECORD
700 :   RN = 1
710 :   GOSUB 1970:REM POSITION TO RECORD
720 :   INPUT#2,LN$
730 :   IF LEFT$(LN$,1)="+" THEN ER=RN:GOTO 780
740 :   GOSUB1970:REM REPOSITON TO RECORD
750 :   RN = RN + 1:IF RN < 201 GOTO 710
760 :   ER = RN
780 RETURN
790 :
800 REM SEE/CHANGE A RECORD
820 : REM GET A RECORD NUMBER
840 :   PRINT "CLR"LEFT$(DN$,11)TAB(8)"RED WHAT
         RECORD DO YOU WANT?"
850 :   PRINTTAB(8)"BLUE DOWN ENTER THE NUMBER (1-200)"
860 :   PRINT TAB(8)"DOWN AND PRESS RETURN";
870 :   INPUT "  * LEFT LEFT LEFT";RN$
880 :   RN=VAL(RN$):IF RN<1 OR RN>200 GOTO 840
900 : GOSUB 1720:REM GET DATA
920 : REM DISPLAY DATA ON THE SCREEN
940 :   PRINT"CLR":GOSUB 1820:REM LABELS
950 :   PRINT"HOME RED RVSON RECORD NUMBER";RN;"LEFT "
960 :   FOR I = 1 TO 6
970 :  PRINT TAB(13)"BLACK";DA$(I)
980 :   NEXT I
1000 : REM SUBMENU TO SELECT OPTION
1020 :   PRINT LEFT$(DN$,9)"BLUE PRESS THE LETTER
          OF THE OPTION YOU WANT."
1030 :   PRINT "RED C BLUE HANGE SOMETHING"
1040 :   PRINT "RED N BLUE EXT RECORD"
1050 :   PRINT "RED P BLUE REVIOUS RECORD"
1060 :   PRINT "RED E BLUE RASE THIS RECORD"
1070 :   PRINT "RED R BLUE ETURN TO MENU"
1080 :   GET G$:IF G$="" GOTO 1080
1090 :   IF G$ = "R" GOTO 1380
1100 :   IF G$ = "E" GOTO 1280
1110 :   IF G$ = "P" THEN RN=RN-1:IF RN<1 THEN RN=1
1120 :   IF G$ = "N" THEN RN=RN+1:IF RN>200 THEN RN=200
1130 :   IF G$ = "P" OR G$= "N" GOTO 900
1140 :   IF G$ <> "C" GOTO 1080
1160 : REM CHANGE SOMETHING
1180 :   PRINT LEFT$(DN$,17)"ENTER THE NUMBER OF
          WHAT YOU WANT"
1190 :   PRINT "TO CHANGE"
1200 :   GET G$:IF G$="" GOTO 1200
1210 :   G=VAL(G$):IF G<1 OR G>6 GOTO 1200
1220 :   INPUT"DOWN ENTER NEW DATA  * LEFTLEFTLEFT";NV$
1230 :   DA$(G)=LEFT$(NV$+BL$,VAL(MID$(LA$(0),G*2-1,2)))
1240 :   GOSUB  1900:REM PRESS S
1250 :   GOSUB  2160:REM CREATE DATA STRING
1260 :   GOTO 1340
1280 : REM ERASE RECORD
1300 :   GOSUB 2050:REM CREATE PADDING STRING
1320 : REM PRINT CHANGES TO DISK
1340 :   GOSUB 1640:REM SAVE CHANGES
1360 : GOTO 900
1380 RETURN
1390 :
1400 REM ADD A RECORD
1420 : RN=ER:GOSUB 1720:REM GET DATA
1430 : PRINT "CLR":GOSUB 1820:REM LABELS
1440 : PRINTLEFT$(DN$,14)"BLUE ENTER DATA
        AS INDICATED."
1450 : PRINT"DOWN PRESS RETURN AFTER
        EACH ENTRY."
1460 : PRINT "HOME RED RVSON RECORD NUMBER ";RN;
        "LEFT  "
1470 : FOR I = 1 TO 6
1480 :   PRINT TAB(13);:INPUT "BLACK  *
          LEFTLEFTLEFT";DA$(I)
1490 : NEXT I
1500 : PRINT  LEFT$(DN$,20);:GOSUB 1900:REM PRESS  S
1510 : GOSUB  2160:REM CREATE DATA STRING
1520 : GOSUB 1640:REM SAVE CHANGES
1540 : GOSUB 700:REM FIND NEXT EMPTY RECORD
1550 : PRINT "DOWN BLUE ADD ANOTHER RECORD?
        PRESS Y OR N"
1560 : GET G$:IF G$="" GOTO 1560
1570 : IF G$ = "Y" GOTO 1420
1580 : IF G$ <> "N" GOTO 1560
1600 RETURN
1610 :
1620 REM SAVE DATA ON DISK
1640 : GOSUB 1970
1650 : PRINT#2,P$
1660 : GOSUB 1970
1680 RETURN
1690 :
1700 REM GET DATA
1720 : GOSUB 1970
1730 : FOR I = 1 TO 6
1740 :   INPUT#2,DA$(I)
1750 : NEXT I
1760 : GOSUB 1970
1780 RETURN
1790 :
1800 REM DISPLAY LABELS
1820 : FOR I = 1 TO 6
1830 :   PRINT "DK GREY";LA$(I)
1840 : NEXT I
1860 RETURN
1870 :
```

```
1880 REM PRESS S MESSAGE
1900 : PRINT "DOWN BLUE PRESS S TO SAVE CHANGES"
1910 : GET G$:IF G$ <>"S" GOTO 1910
1930 RETURN
1940 :
1950 REM POSITION TO RECORD
1970 : HB =  INT(RN/256)
1980 : LB = RN - HB*256
1990 : PRINT#15,"P"+CHR$(98)+CHR$(LB)+CHR$(HB)+CHR$(1)
2010 RETURN
2020 :
2030 REM CREATE PADDING STRING
2050 : P$=LEFT$(PD$,15)+CR$:REM LAST NAME
2060 : P$=P$+LEFT$(PD$,12)+CR$:REM FIRST NAME
2070 : P$=P$+PD$+CR$:REM STREET
2080 : P$=P$+LEFT$(PD$,15)+CR$:REM CITY
2090 : P$=P$+LEFT$(PD$,4)+CR$:REM PROVINCE
2100 : P$=P$+LEFT$(PD$,7):REM P. CODE
2120 RETURN
2130 :
2140 REM CREATE DATA STRING
2160 : P$ = ""
2170 : FOR I =  1 TO 5
2180 :   P$ = P$ + DA$(I) + CR$
2190 : NEXT I
2200 : P$ = P$ + DA$(6)
2220 RETURN
2230 :
2240 REM TITLE
2260 : REM SORRY - NO TITLE
2270 : REM DATA SET-UP OCCURS WHILE TITLE SCREEN
        IS DISPLAYED
2290 RETURN
2300 :
2310 REM DATA (LABEL TITLES)
2330 DATA "151225150407","1 LAST NAME","2 FIRST
      NAME","3 STREET"
2340 DATA "4 CITY","5 PROV","6 P. CODE"
2350 REM FIRST DATA ITEM LA$(0)  CONTAINS FIELD SIZES
```

CAPUTE'S G.D.G. OPCODES AND COMMANDS

by Barry Bircher

CAPUTE's GOSH DARN GOOD OPCODES FOR DUH CPU

Here is one of CAPUTE'S most popular and useful utilities ever published in a magazine since it was first published last week. The programm wedges itself in the most useless place you can think of in the computer so you don't have to worry about crashing it (All it takes to crash it is to be near an electrical outlet). The program can be used in both the monitor and in BASIC as it is wedged in a main KERNAL vector "CRaSH", and therefore is not affected by some idiot hitting RUN/STOP+RESTORE.

Here are some useful opcodes to use when you boot your computer with my new GOSH DARN GOOD MACHINE LANGUAGE, language extention. It can be used with all existing CPU's on the market today. It will auto-configure itself to your system, so there's no need to relocate. Since GOSH DARN GOOD is totally M.L., you will have to use MLXUP (elswhere in this issue) to enter it. When you have MLXUP up and running, it will ask you for a starting address. You should answer 2300 and end it at 2320. To introduce you to my GOSH DARN GOOD operating system (GEOS compatable) here are some useful opcodes to use while in monitor mode. Program listing is on page 9. Film at eleven.

ADC	- ADd with Carry
AND	- Logical operator, used for masking out bits.
ASL	- Automatic Shift To your Left
BCC	- But Can't I Come
BCS	- But Can I Stay
BEQ	- Before Ernie Quits
BIT	- Back In Two jiffies (in the biffie)
BMI	- Beam Me In
BMU	- Beam Me Up (Scottie!)
BNE	- Buy Non-Expensive equipment
BPL	- Branch if Processor Locks (very useful)
BRK	- coffee BReaK
BUT	- Logical operator eg. BUT IF it doesn't then WHYNOT (see also NOT, WHY and HOWcome)
BVC	- Branch if oVer-Crowded
BVS	- Branch if oVerStocked (bypasses commercials)
CLC	- CLear the Carry
CLD	- CLearit Dammit!
CLI	- CLear it, you Idiot!
CLV	- Collect Leftover Variables
CMP	- Call Mom & Pop
CPX	- Collect Pension Cheque
CPY	- CoPY
DEC	- To hit someone
DEX	- To hit someone (plural)
DEY	- DEmand and Yell
EOR	- Winnie the Pooh's donkey (kick start)
IF	- Logical operator eg. IF this AND that BUT NOT this OR this then WHYNOT (see MAY IF, AND,OR,BUTTes)
INC	- Stuff they use in pens
INX	- What the washer does to pens and your clothes
INY	- I kNow Not whY (logical operator used for no knowledge)
JMP	- What you do if this works
JSR	- Junk Status Register
LDA	- Long DAy
LDX	- LoaD and cross
LDY	- LoaD and Yell
LSR	- Last Stupid Remark
MAY	- Logical operator eg. MAYbe this, OR that AND then again MAYbe NOT BUT then MAYbe so (see also NOT,IF,AND,OR BUT)
NOT	- Logical operator eg.NOT this AND that BUT NOT that (see AND)
NOP	- NO Party tonight
ORA	- Logical operator eg. Could be this ORA, this OR maybe NOT this and NOT that (see also BUT MAY NOT MAYbe)
PHA	- Push HArd
PHP	- Party Hardy Partner
ROL	- short for Rolls Royce
ROR	- Lion Sound (used by SID's)
WHY	- Logical operator eg. if WHY then HOW did it NOT work (see also HOW ,NOT)

****

Commands Used by GOSH DARN GOOD in BASIC
B. Bircher

****

LPRINT	- print on line L
HPRINT	- print Hard copy
LQPRINT	- Letter Quality print on dot matrix printers
COLORS	- set up 255 colors from a palette of 65535
CCOLOR	- Select colors from color (see COLORS)
MSIZE	- check on memory size in K's & add if necessary
SDOS	~~-- speed up disk access to 1 Giga byte~~/second
RAMDOS	- Turn disk storage into contiguous RAM (see SDOS)
TERM	- Built in Terminal program with 640K buffer
ETERM	- Speed up 300 baud modems to emulate 2400 baud
GOCP/M	- switch to CP/M plus
GOMS	- switch to MS-DOS
GOBED	- Sets alarm, shuts off computer after 11:00 PM
GO640	- adds 640K bytes to BASIC storage
GO128	- Switch to 128 mode
FFAST	- switch to 4 MHz clock
SFFAST	- switch to 20 MHz clock (requires SDOS activated first)
SP	- Switch disk to regular speed
LP	- (Long Play) slows disk to get more info on it. (see SLP)
SLP	- (Super Long Play) gets up to 3.25 Gigabytes per disk. (see also SP)
AMIG	- Enable multitask mode
RECHRG	- Recharge RAM backup batteries right now
UNCHRG	- Erase your last 5 VISA transactions

JUST FOR THE RECORD

(OR IS THAT A FIELD?)

by Richard Maze

There are many terms that are associated with handling large quantities of data. Often these terms are used incorrectly resulting in confusion and a misunderstanding as to what a particular program will do.

The major terms used are: DATABASE, FILE, RECORD, FIELD, and CHARACTER. These terms represent a hierarchy of data organization. It is often easiest to look at them starting with the lowest level and working up.

CHARACTER: This is one letter or number entered as a part of data. It could be a letter of a person's name or one digit in a sales amount.

FIELD: A group of characters together which make up ONE DATA ENTRY entry. A person's last name or a sales amount might be examples of different fields.

RECORD: A group of fields make up a RECORD. The name, address, city, prov., and postal code for a person could be ONE RECORD.

FILE: A number of records is grouped together to form a file.

DATABASE: The grouping together of related files.

An example of this organization could be constructed as follows: If a company has a set of index cards for its customers, then, the entire set of cards would be a FILE. Each card, which contains information on one customer, would be a RECORD. Each space to be filled in on each card, (name, address etc.) would be a FIELD. Each field would be filled in using CHARACTERS. Now if the company had another parallel set of cards containing other information about its customers, (amount ordered, amount owing, etc.), this parallel set would be another FILE. The TWO SETS together would make a DATABASE.

One common misuse of terms is that programs that are really just file handling programs are called databases. For example, "a database to help you keep track of magazine articles" is really a file handler program - NOT a DATABASE. This is also true of commercial programs - many 'database managers' are really 'file managers'. A true database manager program should allow data to be accessed from two or more files.

Another area that often creates confusion relates to sizes - FILE size, RECORD size, FIELD size. Strictly speaking, FILE size should be the NUMBER OF RECORDS that can be stored; RECORD SIZE is the SUM OF ALL the CHARACTERS that can be entered in all the fields; and FIELD size is the NUMBER OF CHARACTERS that can be entered in one particular field. If a program stores the data in memory, file size is limited by computer memory. Otherwise, unless the program limits it, FILE size should be limited only by disk space available. RECORD size is often limited to 254 characters (a convenient storage and transfer unit - 1 disk sector) but programming techniques can get around this. FIELD size is up to the user to determine but may be limited to 80 characters (input limits this) or 40 characters (screen width). The most important factor is that the sum of all field sizes cannot exceed the record size limit.
If you looking for a database/file program, it might be worthwhile to do some calculations based on the specified sizes. Often the number of possible records that is stated is only possible if there are minimum fields of minimum length. In most cases: larger field sizes and greater numbers of fields means less records that can be stored per file. You can't just calculate the available space on a disk to determine how much can be stored (664 blocks * 254 characters/block = 168656 characters) because many programs create auxillary files to support the file which also take up storage space.

FOR SALE:

POCKET Writer Filer Planner

$70.00 O.B.O.
call BARRY 359-1925

ON-LINE WITH Greg

BBS list as of March 11 ,1988

$ = Pay BBS * = Temporarily Down

(Ed. note) by Greg Rezansoff

The "Set" column below provides information on MODE (Full or Half Duplex), # OF BITS (7 or 8), PARITY (Odd, Even or None), and # OF STOP BITS (usually 1). The "Baud" column indicates the relative speed at which files can be transferred to and from the bulletin board. Most of this information is necessary to properly prepare your system to "log on" to the system desired.

-System Name-----	Phone Num-	Baud-	Set--	comment-----------	
Abyss	584-0721	300	F8N1	Dark Angel's PFBBS	
Bermuda Square	545-7678	2400	F8N1	RBBS	
Bit Bucket	352-3236	2400	F8N1	4 Amiga,Atari,64	
Dialogue	775-3587	2400	F8N1	Atari BBS	
Digtl Dimensions	586-6493	1200	F8N1	Atari 8 bit	
DoubleCheck	525-0807	2400	F8N1	V. good Amiga brd	
Edu-Net	522-1998	1200	F8N1	Brd of Educ.	
EMIS	586-5585	1200	F7E1	Unix multiuser	$
EMIS 2	584-2035	1200	F7E1	Unix multiuser	$
Fernando's	585-0298	9600	F8N1	Pyroto Mtn/PC	*
Infobase	789-5463	1200	F8N1	PC clones only	
Late Night Alt.	586-3285	1200	F8N1	Good convos /PC/Amiga dls	
Magic Fountain	586-2692	1200	F8N1	Small but growing	
Micro City I	584-0747	1200	F8N1	Multiuser system	$
Micro City II	584-0748	1200	F8N1	Multiuser system	$
Negative Zone	565-8538	2400	F8N1	Good echos/convos	
Onyx	347-1015	2400	F8N1	Great CP/M good PC support	
Pirate's Galleon	775-3358	2400	F8N1	Growing PC system	
Polestar Opus	586-1551	2400	F8N1	PC support	
Probable Fate	525-1054	300	F8N1	Convers.&txtfiles	
Regina FIDO	347-4493	2400	F8N1	Great.PC&CP/M	
R.A.C.E. BBS	545-7035	300	F8N1	Atari only BBS(NEW)	
R.A.T. II	949-6105	1200	F8N1	Good Atari 8 bit	
Saskatchewan ROS	789-0690	1200	F8N1	Good Amiga/PC	
Shadowland I	789-7883	2400	F8N1	Home of The Realm	
Shadowland II	789-8989	2400	F8N1	Good Mac pic. area	
State of Mind	352-1455	2400	F8N1	PC based RBBS(NEW)	
SwiTch	569-0883	1200	F8N1	Atari ST based	
Tee Wun Kay	779-1237	2400	F8N1	PC support	
Tesseract	757-5699	1200	F8N1	Excellent Amiga	
Triple Diamond	775-3314	1200	F8N1	C-64 based AEBBS	
Turbo	586-7560	1200	F8N1	Good convos 64&PC files	
UltraFast Systems	545-5717	2400	F8N1	Amiga based (NEW)	
Wizard's Wand	949-0178	2400	F8N1	PC supp./chess	
Datapac 300	565-0111	300	H8N1	Datapac 300	
Datapac 1200	565-0181	1200	H8N1	Datapac 1200	
Datapac 2400	565-6000	2400	H8N1	Datapac 2400	
U. of Regina	584-8085	1200	F8N1	U of R Develswitch	

NEW C.U.G.S. DISK LIBRARY ADDITIONS:

CUGS GAZETTE #26

```
menu
6.<->
sprint ii
 sprint ii.32768
spr ii routines
sprint doodler
needlework editr
 needlework.obj
scrolledit
metascroll
 scrolledit.obj
 metabasic+.36864
6.<-->
mosaic
fast 64.note
 fast 64 mode/128
multi-list
turbo ss
 alter spdscrpt
 turbodisk boot
 turbodisk
 turbosave
 speedscript
imp borders
fire!
easyload
easyload/128
func key magic
boot epson
 grand pix.epson
boot 1525
 grand pix.1525
boot 1526
 grand pix.1526
rads
 rads.obj
6.<--->
basically music
 basic music.obj
 fur elise.demo
big screen
 big screen.obj
sketch-pad+ menu
 sketch-pad.ml
 printsketch.ml
 savesketch.ml
 sketch-pad+.ml
xpresscard/128
human rat race
 human rat.49152
color lister
color lister/128
color lister/16
geos dir printer
ml cloner
oil defense
 oil defense.obj
6.<---->
dgraph loader
 dgraph.obj
dgraph painter
 demos
spy defense
 spy defense.obj
smart val
 smart val.bas
 mini-calc demos
calc goto demo
sel restore demo
bsave demo
directory demo
key clicker
key clicker/128
three-d speedway
 speedway.49152
ramdisk/128
 ramdisk.4864
speed file
 speed file.obj
 contents
screen
```

ARC. GAMES 13 #AM

```
cugs loader
 cugs data
---------------
boot/frus
 boot/hints
 frustration5.03
 frus/hints1.2
hookdodger
 hd
kanga
 boxes
 ml
 k-instructions
hunchback
camel.boot
 starter
 amc
camel caverns
chess.c
 chess-64
 chess/clk
 chs/char
ship to ship
```

CUGS 128 PGMS #13

```
conden'dfont/128
cf costomizr/128
tile pattern/128
hires tilepn/128
pie chart dm/128
new patterns/128
double paint/128
 tilepaint.6656
ss justify
file manager/128
 file manag.doc
yahtzee/128
fast 64.note
 fast 64 mode/128
multi-list/128
easyload/128
xpresscard/128
color lister/128
ml cloner/128
key clicker/128
ramdisk/128
 ramdisk.4864
power poke/128
phantom list/128
colorplot/128
 colorplot.docs
 samples
canvas/128
 canvas help
draw-n-paint/128
commodore demo
 com.spr3
80reader1.1
cw-memo.80
```

TEXT GAMES 8 #TH

```
cugs loader
 cugs data
---------------
charlemagne's
word hunt
jotto
unscramble
the farm game
power poker
dragon's den
tombs.c
headline news
six chess
adv kit
 adv docs
sprint iv
```

CUGS 128 PGMS #14

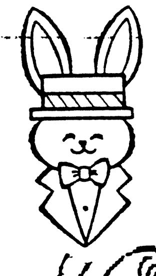
```
ultraterm
 startup
 index
 u-docs
---------------
star bbs v2.5
 star.ml
 star.setup
 starv2.5.docs
 main menu 1
 main menu 2
 sig menu 1
 sig menu 2
 transfers menu 1
 transfers menu 2
 bulletin menu 1
 bulletin menu 2
 email menu 1
 email menu 2
 welcome 1
 welcome 2
 new user 1
 new user 2
 news 2
 news 1
 logoff 1
 logoff 2
--- ------- ---
seq divider
- --------- -
file reader/3.0
```

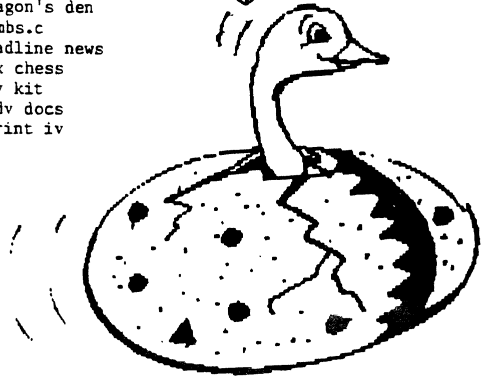
CUGS 128 PGMS #15

```
gas128
 1gas.doc
 2gas.doc
mentor 128
naughty noties
nerdcopy v1.1
disknoser v1.5
diskdoctor128 v2
 dd.doc.clean
80reader1.1
job list 128
```

SOUND 8 #SH

```
cugs loader
 cugs data
---------------
sprint iv
sidpic v3.4
 sidpic.docs v3
 herbies rag.mus
 lilyqueenrag.mus
vailrun.mus
 sarahdance.mus
 hymne.mus
 onojohn.mus
stereoplayer7.0
 stereo document
 stereo schematic
stereo demo
max sings
```





SOUND 9 #SI

```
cugs loader
 cugs data
---------------
world of madness
 a to z
final synth i
```

PRINT UTIL. 2 #PB

```
cugs loader
 cugs data
---------------
small disk label
hiscreen printer
 m7 plot+print
graphic dump/64
label maker.c
fast dump/128/64
seq reader
high res dumper
labeler.src
label loader.c64
label test prog
print ml.c64
screen dump
boldface
printdir.c
calendar.c
hex table
cbm(new.49152)
list ascii $c0.c
list ascii $9d.c
func template
 ft.multiplan
 ft.man-c/r
 ft.man-e/e
 ft.man-arith
sleeve maker
calendar maker
geos note printr
lister filter
printer wedge
 char sets
fontmaker
fontprinter.boot
 fontprinter
calendar 1520
 jjspock
e-z.seq.read
banner.printer
doc.reader
custom labels
lfs form maker
ultra seq ptr
```

BUSINESS 9 #BI

```
cugs loader
cugs data
---------------
microfile
 MF000-1.07A
 MF001-1.07G
 MF003-1.07F
 ?Tape Index samp
 ?Disk Catalog sa
 ?Jam Labels samp
 ?Address Label s
 mfdeletedatafile
 MF002-1.07I
 ?MF-SPECIF-107D
 ?MF-EQUIPM-107D
 ?MF-FEATUR-107C
 ?MF-CHGDAT-107B
?MF-DATTIM-107B
?MF-COLOR -107B
?MF-FREEWA-107C
?MF-WARRAN-107C
?MF-MAIN   -107C
?MF-NEWDAT-107A
?MF-SORT   -107A
?MF-REGIST-107D
?MF-INSTAL-107A
?MF-ENTER -107A
?MF-SCREEN-107A
?MF-START -107A
?MF-CHGHDG-107C
?MF-CREATE-107C
?MF-DISPLY-107C
?MF-SAVE   -107B
?MF-PRINT -107C
?MF-FILES -107H
 ?MF-PROGRA-107E
 ?MF-DELETE-107B
 ?MF-MENU   -107K
 ?MF-ENDING-107B
 ?MF-LOAD   -107B
+---------------+
calcaid
mini-filer
 mini-filer.obj
poll maker
poller
vp docs
c64-averager
```

Scratch 'n' Save

by Earl Brown

For those of you that arrived at a negative number in line 20 of SCHEDULE 10 CHILD TAX CREDIT in the 1987 Income Tax program, the following change in line 403 of the SPECIAL, and line 767 of the GENERAL should be made:

```
403
or    XC=AL-M1:IFXC<0THENXC=0
767
```

Quite often, when writing a program, I type a new number over an existing line number and make the appropriate change to that line. For some reason, I forgot to make all the changes and the 'if' statement contained the wrong variable. Thank you, Richard, for bringing this to my attention.

For those members interested in the database program "MICROPHILE" which Richard is demonstrating this evening, you will find it on CUGS BUSINESS 9 DISK #BI. Also included on this disk is another database program entitled "MINI-FILER" from Gazette Magazine (Feb/86) and a spreadsheet program "CALCAID" from Run Magazine (Nov/86) for those of you who purchase these magazines.

There are five additional disks for the 64 in this month's listings as well as three more for the 128. Also included in the listings is the latest Gazette disk which includes the remaining programs from January, all of February and March, and most of April 1988. The Turbo Speedscript program that is included on this disk for the 64 won't work with my 1571 drive but does work with all 1541 drives. The boot and additional files allows this word processing program to fast-load and fast-save all your processed files.

Finally, one thing I forget to mention each month. The library has in its possession two books which members ~~may borrow (in one month stretches) from~~ the library. They are:

1. THE COMAL HANDBOOK - a 310 page handbook on how to use the COMAL language to your best advantage. We have COMAL program disks in our library.

2. THE BEST OF TORPET MORE FOR THE COMMODORE 64 AND THE VIC-20 - This book covers many aspects of the 64 and Vic-20 and includes a disk with many useful programs on it.

```
***************************************************
*                                                 *
* I picked up a good looking printercover at February's *
* COMPUTERFEST.  The problem is it is too small for my *
* FX-80 EPSON PRINTER.  The dimensions of the cover are *
* 14 1/2 inches wide by 12 inches front to back.  Five *
* dollars is what I paid for this cover manufactured by *
* AMERICAN COVERS INC and if you can use it, it's yours *
* for just THREE DOLLARS! OKAY? .................. *
*                                                 *
***************************************************
```

PRESENTING:

the

Great Software SWAP

PLACE: NORTHWEST LEISURE CENTRE
DATE: WEDNESDAY, MAY 4, 1988 7pm

Open to all Commodore User Group members (Please have your membership card).

Admission will be to card-holding members only. Members should come with PUBLIC DOMAIN or SHAREWARE material to swap.

NO COPYING OF COMMERCIAL SOFTWARE WILL BE PERMITTED.

Members wanting to offer legitimate used software for sale may do so on their own responsibility. We hope to see <u>YOU</u> there!

C.U.G.S MONITOR APRIL FOOL ISSUE

Vol. 3 Issue 4 April 1988