# CoCoNutz!

Volume 1 Issue 3
October 2005


## In this Issue

# Dennis Bathory-Kitsz

Dennis Bathory-Kitsz is the author of the coco maltedmedia list serv that many of us participate in via email. I thought it only fitting to get to know Dennis for this issue of the newsletter. There is a lot to get to know about Dennis, but I have hoped to capture a bit of an overview of him. Dennis is married, with two step kids (not at home), two cats, three horses, one house, one barn, and three gardens. He has many interesting stories to tell, and there are a lot of aspects of his life that I have come to know. I hope that you all check out each of his websites. The list is just a small portion compared to his life and musical talents. He is talented in many ways and I hope that this gets you all interested to learn more through his many websites. I want to add, that for someone who seems so busy he was nice enough to grant me this interview without hesitation. Thanks Dennis for your generosity and time!

First I asked Dennis about his Music. He had this to say:

I think you might want to learn a little more about me before asking questions about my music. It's a pretty intense topic, because I've composed over 700 pieces. :) It's called nonpop, and it runs from electroacoustic music to symphonies. You'll also find lots on my homepage, http:\bathory.org, and here's my radio show homepage (I'm the "Kalvos" of this show): http://kalvos.org/

**CoCoNutz:** Why did you pick the coco instead of Atari or something else back then?

In 1978, two years before the CoCo existed; I had a TRS-80 Model I (before it was called the Model I) and a KIM-1. My book about the Model I was called "The Custom TRS-80", and it was one of the most popular books ever

written about that machine. Unfortunately, it wasn't very good for musical uses except as a sequencer for my analog synth:
http://maltedmedia.com/people/bathory/killer.html

I got an early CoCo because it had a hybrid 8/16-bit processor, and I thought it might be better for music. It was better. I wrote a program called "Quaver", which was the first program to provide fully independent 4-voice polyphony on a CoCo without using external hardware.

**CoCoNutz:** Why did you stick with it?

I didn't, really. I stayed with it for six years, eventually creating a music/sculptural installation in Montana with 5 CoCos:
http://maltedmedia.com/people/bathory/bocca/
Then my company went bankrupt, and I put all my CoCos in storage and returned to the Model I, which I kept until I bought a Windows PC in 1992.

**CoCoNutz:** Were you involved with it business wise did you sell stuff etc.?

I founded Green Mountain Micro, which sold hardware and software. I designed the hardware and wrote the software. I designed the first lowercase mod called the Lowerkit, the Color Burner EPROM burner, and the Data Gatherer analog interface system.

**CoCoNutz:** Have you written any programs for coco?

The most interesting were "Quaver" (which I mentioned above) and "Scroller," which was a high-resolution announcement board that I wrote for the Smithsonian, and the operating system for the Data Gatherer. Also, I wrote, designed and narrated "Learning the 6809", a programmed learning cassette/textbook course for the CoCo's microprocessor.

**CoCoNutz:** Did you ever go to coco fest and rainbow fests...do you still attend the Glenside one?

I was at Glenside as an invited guest a few years ago, and had lots of fun. Back in the "old days" when my company was still around, I went to all the festivals and conferences and trade shows because it was the best way to meet people and learn what products they wanted. My most famous

appearance was at a Princeton Rainbowfest, where I wore a black-and-white striped prison outfit and had shaved my head (before that I had long hair).

**CoCoNutz:** Tell me about this documentary about your family on (was it discovery channel?)

All the info is here: http://bathory.org/
(Can you tell I have lots of websites? :)

**CoCoNutz:** Do you travel outside the US, what is your favorite place to visit?

I travel outside the US at least once a year, and lived in the Netherlands in the early 1990s. I speak French, Dutch and some German, and plan to move the Europe in a few years. My favorite places are New York, Amsterdam, Prague, and southern France.

**CoCoNutz:** You have the coco newsgroup going did you start this yourself? Why....

The old newsgroup was being infected by thousands of spams, so I started the list. Since I lease a fairly large server, I had room and bandwidth for a list, so had the Mailman software installed. After that, it was pretty smooth.

**CoCoNutz:** How many members did you have for the list at first?

I never counted! There are about 200 now, I think.

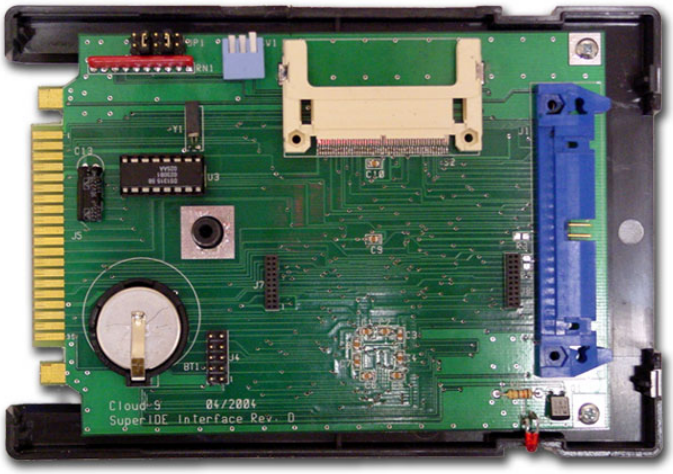
**CoCoNutz:** Did you ever write articles for hot coco or other magazines?

Not Hot Coco. But I wrote for Byte, 80 Micro, Kilobaud, Desktop Computing, The Alternate Source, The Color Computer Magazine, Programmer, Opinion-80, Software Critique, Dynamic Color News, The Rainbow, and UnderColor.
Altogether, I wrote about 500 articles about the TRS-80 Model I, the CoCo, and the MC-10.

**CoCoNutz:** Any possibility I might get one for the newsletter? :) wink

I'll have to look in storage. I don't have any here.

# The Next Generation of Performance...




**The Ultimate CoCo Storage Solution**
Look no further than the SuperIDE Interface from Cloud-9. This innovative product utilizes the latest in electronic design technology to bring you an interface that rocks your CoCo!

**Blazing Fast CompactFlash On-Board!**
Use popular, widely available CompactFlash modules to access up to 1 gigabyte of solid-state, super fast storage. No hard drive crashes to worry about, and fast, reliable data storage that lasts for years.

**It Does Hard Disks, Too...**
If you prefer rotating cylinders, no problem! The SuperIDE Interface sports a standard 40-pin IDE connector for hard drive lovers.

**Boot In A FLASH!**
With 64K of internal FLASH memory, you can house up to four 16K ROMs. Choose from HDB-DOS, Radio Shack Disk BASIC, your favorite DOS, or even a ROM Pak! Easy to use software allows you to program your ROM in-circuit. No more obsolete EPROMs or cumbersome burners needed!

**Got The Time?**
Choose the optional real-time clock for up to the second time on your CoCo. Never wonder again what time it *really* is.

**Available Now!**
Get your CoCo souped up with the SuperIDE Inerface from Cloud-9 today!

www.cloud9tech.com

Coco Abbreviations.

GIME Graphics, Interrupts, Memory Enhancements
HLA High Level Assembler (win 32 or linux)
ASM Assembly Language
PWM Post Width Modulation
EOF End Of File
LSI Large Scale Integration
RAM Random Access Memory
ROM Read Only Memory
CPU Central Processing Unit
VDG Video Display Generator

# Run Any Disk Basic Program with the DOS Command by Robert Gault

Here is a utility that you can use to run any Basic program with the DOS command from Disk Basic 1.1. Actually the program would also work with Disk Basic 1.0 except that version 1.0 does not have a DOS command. However, if you use the *.BAS program that exists on Tandy OS-9 disks as a replacement for the DOS command on version 1.0 system, this program should still work.

Why should this utility be of any interest? Well there may not be too much point to it, but it is a good learning tool for getting an insight into how parts of Disk Basic work and how you can increase its functionality with very simple programs. The normal method of running programs is RUN"PROGRAM" but that requires you to remember the name of every program you want to run. This utility can be used to start the main program on any disk just by entering DOS.

There are two parts to the program, a binary program which does all the work, and a Basic loader that places the binary code on track 34 of a disk. The assembly source code is in EDTASM format but can easily be changed to work with other assemblers such as CCASM in Portal-9; replace .EQU. with = in the conditional (COND) statements.

Here is how the utility is used. Let's say you have a program, MYFILE.BAS, that you want to be run. This program is placed on a disk, the file name is added to DOSBOOT.BAS, and DOSBOOT.BAS plus DOSBOOT.BIN are placed on the same disk as MYFILE.BAS. Running DOSBOOT will alter track 34 so that when the DOS command is issued, MYFILE.BAS will run.

## DOSBOOT.ASM

```
00100 *DOS routine for Basic by Robert Gault June 2002
00110
00120 * This is a generic routine to autostart a Basic program
00130 * with the DOS command. The program must exist on the indicated
00140 * drive and the name of the program must be entered at T34 S2.
00150
00160
00170 * Select your version of Disk Basic.
00180 VERSON      EQU   11          Must be 10 or 11 (ie. 1.0 or 1.1)
00190
00200             COND  VERSON.EQU.10
00201 * This version of Disk Basic does not have the DOS command so
*.BAS
00202 * (from OS-9 disks) must be used instead.
00210 GETNAM      EQU   $D2C4       These are the locations of Disk
00220 LOADR       EQU   $C9B9       Basic routines.
00230             ENDC
00240
```

```
00250             COND  VERSON.EQU.11
00260 GETNAM      EQU   $D3B1
00270 LOADR       EQU   $CA67
00280             ENDC
00290
00300 DOSLOC      EQU   $2600       DOS loads T34 to this location.
00310 OFFSET      EQU   $100        The program name will be at this
offset.
00320
00330             ORG   $7000       Used to compile this code.
00340             FCC   "OS"        * fool the DOS command
00350 START       LDX   #DOSLOC+OFFSET    Point to the program name.
00360             JSR   GETNAM      Make Basic set up the file name.
00370             CLR   DOSLOC      IMPORTANT! Clear byte at $2600
00380             JMP   LOADR       LOAD,R Make Basic load and run
program.
00390             END
```

## DOSBOOT.BAS

```
10 ' DOSBOOT.BAS by Robert Gault    June 2002
20 ' This program will setup the DOS loader in T34 S1
30 ' Change the name of the file in line 120 as needed.
40 CLEAR1000:B$=STRING$(128,CHR$(0))
50 DR=PEEK(&HEB):' Get the drive number this program is on.
60 DRIVE DR:' Set it as the default drive.
70 LOADM"DOSBOOT":'  Get the magic ml code.
80 FOR M=&H7000 TO &H707F:'  More bytes than needed but fills A$.
90 A$=A$+CHR$(PEEK(M)):NEXT M
100 DSKO$ DR,34,1,A$,B$:' Write a sector to track 34, sector 1
110 REM Here is where you decide what to run.
120 C$="MYFILE  BAS":' This name must be exactly 11 characters and
130 '                 BAS must be right justified.
140 C$=C$+STRING$(128-LEN(C$),CHR$(0)):' Fill in the string.
150 DSKO$ DR,34,2,C$,B$:' Write a sector to track 34, sector 2
```

## MYFILE.BAS  a test case

```
10 CLS:PRINT"MYFILE.BAS IS RUNNING!"
```

# “Highly Advanced ECB Programming” or “The Indefinite FOR/NEXT loop”

## by Richard Kelly a.k.a. Retro Rick

Time for some very advanced ECB programming. The ECB manual has left out one detail about specifying variables. Let's say that you have the statement:

```
10 IF S+X=-1 THEN X=-X
```

… but you accidentally leave out the “=-1” part of the statement. Strangely, it still runs fine, as if you left nothing out. What gives?

Well, the “=-1” is put in there automatically. This is the way most versions of BASIC work.

Now, the “=-1” part doesn’t end there. Try this command:

```
A=(X>6)
```

You’d expect an error message, right? Well instead, you’ll get a “-1” if the value of X is less than 6. If the value of X is *not* less than 6, then the value of A will equal Zero. In other words, a “-1” is used as the value if the statement is true, and Zero is returned if the statement is false.

Let’s take advantage of this fact with a program example. Let’s say you want to play sounds 1 through 255, but every time the value of the sound was an even number, you wanted the sound to be double the length of odd-numbered sounds. You might normally type it in as a multiple-line program:

```
10 FOR  S=1 TO 255:IF S/2=INT(S/2)THEN T=2 ELSE T=1
30 SOUND S,T:NEXT
```

But the “IF … THEN” portion is actually not needed. Instead, you could put the statement into the value of T. Then, whenever the S variable was even, T would equal Zero. If S was an odd number, then T would equal -1.

The work is complete once you add 2 to T after the S-related statement. Like this:

```
FOR  S=1 TO 255:T=(S/2)>INT(S/2):T=T+2:SOUND S,T:NEXT
```

Note that the “=” between “2” and “INT” was changed to “>”. If it wasn’t, odd numbers would be twice the length of even ones!

Now for something *really* fancy. The program below shows a common way to view all five PMODEs in two different colors. With each screen, the program waits for a key to be pressed by the user before going to the next screen.

```
10 PMODE 0:PCLEAR 4:FOR A=0 TO 4:FOR B=0 TO 1:PMODE A,1:SCREEN 1,B
20 IF INKEY$="" THEN 20
30 NEXT B,A
```

There's a smarter, more compact way of doing this. And I *don't* mean EXEC 44539.

```
PMODE 0:PCLEAR 4:FOR A=0 TO 4:FOR B=0 TO 1:PMODE A,1:SCREEN 1,B:FOR
I=-1 TO 0:I=INKEY$="":NEXT I,B,A
```

Note the "FOR I=-1 TO 0" portion of the code. I call this "The Indefinite FOR Loop". You see, inside the "I" loop is the "I=" statement. That statement sets Variable "I" to "-1" if no key has been pressed, and Zero if a key *has* been pressed. If that part of the code sets "I" to -1, then the "NEXT I" part of the instruction will set the "I" variable to Zero. Since Zero isn't beyond the boundaries of the "FOR I=" loop that you've specified, the code keeps running the "FOR I=" loop until a key is pressed. Whoever dreamed you could do this with a FOR/NEXT loop! PC users - If you used DO/UNTIL or WHILE/WEND in your BASIC programs and want a substitute for it in ECB, this is it!

Now, suppose you wanted to write a very simple ECB program where a dot is bouncing off the edges of the screen and the program will go on until a key is pressed. You've specified the dot's horizontal direction to be stored in Variable "A", and the vertical direction to be stored in Variable "B". When the dot hits the edges of the screen, the value of one of these variables is multiplied by "-1" to reverse the horizontal or vertical direction of the dot.

Now, since multiplying X or Y by "1" doesn't change its value at all, we could have a new variable - "E" or "F" in this case - to equal "-1" if the dot needs to reverse direction, and "1" if the dot is okay to continue going the direction it's headed. To get such a value, we'd multiply "E" or "F" by 2, then add "1" to it. To scan for the screen's boundaries, a simple "OR" will do the trick there; just like an IF statement, only without the IF or THEN, and without having to use multiple lines of code.

So the one-liner below basically draws a dot bouncing off the sides of the screen, and the program keeps running until a key is pressed. Thanks to the advanced variable tricks I've elaborated, it's been proven that such things can be done in ECB without numerous lines of code. Simply type in this program (or run it as a set of direct statements if you wish!), sit back, and enjoy it! And remember - These things and more can be done in just one line of code!

```
10 PMODE0:PCLEAR1:PCLS:SCREEN1,1:A=2:B=2:X=128:Y=96:FORZ=-
1TO0:E=((X<2ORX>253)*2)+1:F=((Y<2ORY>189)*2)+1:A=A*E:B=B*F:C=A+X:D=
B+Y:Z=INKEY$="":PRESET(X,Y):PSET(C,D,1):X=C:Y=D:NEXT
```

Ready for a challenge? Enhance the code to use to the Speed-up Poke based on what CoCo you're running the program on. Use PEEK(33021) to find the CoCo type. The value of 49 is stored in this area if it's a CoCo 1 or 2 (if the ROM is un-hacked, that is). If you've done it right, the entire program should still fit as a one-liner.

---

**Make your own replacement disk pockets!**

NEEDED: a ruler, a disk and a sheet of paper, 8.5x11 or A size

To make a pocket for a 5.25" floppy or a CD:
* fold the page in half, bottom edge to top edge
* fold down the top 3 centimeters of top leaf only
* turn pocket over and measure 3 cm from left and right edges
* use the ruler as a straightedge to crease up the marked edges
* flip the pocket back over and fold the side leaves back
* insert and center a disk, then do the final edge creases
* tape or glue the side leaves in place - done.

To make TWO pockets for 3.5" floppies:
* turn the initial page horizontal before the first fold
* fold the page in half lengthwise, bottom to top
* cut the folded paper in half vertically, then for each half:
** fold down the top 3 centimeters of the top leaf only
** turn the pocket over and measure TWO cm from each side edge
** crease up each edge using the ruler as a straightedge
** flip the pocket back over, then insert and center a disk
** fold the side leaves back and do the final edge creases
** tape or glue the side leaves in place - done.

# A Partial History of Color on the Color Computer

by

Robert Gault

While it may be hard to remember the origins of the Color Computer, it started with about 4K of RAM memory and a simple Color Basic. The potential for interesting graphics was there but was not supported by Tandy. In fact, if you look at the commands for Color Basic, you won't find any for graphics functions. However to be fair, Tandy did include a chapter in the Color Basic manual which showed how to access the graphic functions. So what did Tandy say the Color Computer was capable of in terms of graphics? Two different types of color were mentioned, semi-graphics and normal (later known as PMODE graphics in Extended Color Basic.)

Semigraphics-24 was the most complex semi mode and never was supported by Tandy.[1] This mode produced a 2x12 graphics block on a 64x192 screen with 9 possible colors on screen at the same time. As you might imagine, a 2x12 pixel would not easy to draw with and that might be why there was no support from Tandy. It is, however, possible to put separate colors in each of the 12 pixel lines making this mode actually a 2x1 pixel size. An example of a semi mode is what you see when you print block graphics characters; PRINT CHR$(m) where m=128-255.

Normal graphics has a maximum resolution of 256x192x2 or 128x192x4. A screen resolution of 256x192, offers a small enough pixel that very good drawings can be made. Good drawings if you like pencil sketches. Black and white is hardly "Color". The four color mode offered green, yellow, blue, and red or buff, cyan, magenta, and orange. These are not the greatest of color palettes and the pixel was larger so results were only fair even for the best of artists. The Color Computer 3 improved the situation considerably with high resolution graphics, but that's another story.

It did not take long for Coco enthusiasts to realize that the Coco could do much more than Tandy said was possible. The reason this was true is that the normal display hardware for the Coco was either a TV or a composite monitor. This meant that, in the United States at least, the Coco was feeding signals to an NTSC device. Why did this make a difference? Well color on an NTSC device is controlled by a color burst signal, chrominance, and phasing while luminance is intensity of signal.

The highest PMODE graphics turned off the Coco's color burst output and this should have resulted in only black or white. However, the resolution of 256x192 resulted in a pixel frequency which made the TV or composite monitor think that color information (chrominance) was being received. Thus was born Coco artifact colors. This was not a new discovery, as the effect was present on other computer of that era, but Coco programmers took the effect and ran with it. A professional effort with great artifact color was Bjork's "The Sands of Egypt". I think the first time I saw an explanation of the effect in a Coco hobbyist magazine was an article by James Wood.

Let's assume that your Coco has Extended Color Basic so you have access to the PMODE commands. You could do the same things with Color Basic using POKEs or assembly code as described in Tandy's "Getting Started with Color Basic". Run the short program

```
10 PMODE4,1:PCLS:SCREEN1,0
20 FOR H=0TO255STEP2
30 LINE(H,0)-(H,191),PSET
40 NEXT
50 GOTO50
```

You should see a black screen with a green border and vertical green lines. This is one of the PMODE4 color sets. If you change line 10 to read SCREEN1,1 (which selects the second color set) you should see a black screen with a white border and vertical white lines. In fact, that is exactly what you will see with an RGB monitor but not a TV or composite monitor. With the latter you will see either a solid red or blue screen with a white border. The color displayed depends on how the Coco booted and the color will switch eventually if the Reset button is pushed enough times. However, it is just as easy to change the color by changing the positions of the lines. This can be made automatic by adding a variable to the above program.

```
10 CL=0:PMODE4,1:PCLS:SCREEN1,1
20 FOR H=CL  TO255STEP2
30 LINE(H,0)-(H,191),PSET
40 NEXT
42 A$=INKEY$:IFA$=""THEN42
43 IFCL=0THENCL=1ELSECL=0
50 PCLS:GOTO20
```

This would just be a novelty unless there was an easy way to determine the line color. This can be done by using mixed PMODEs. If you do all of your drawing in PMODE3, then the Coco will automatically place your pixels at the correct horizontal position to get a fixed color when displayed in PMODE4. Try the following as a demonstration and note the two PMODE commands in line 10.

```
10 PMODE4,1:PCLS1:SCREEN1,1:PMODE3,1
20 CL=3
30 COLOR CL,0
40 LINE(0,0)-(255,191),PSET
50 A$=INKEY$:IFA$=""THEN50
60 IFCL=3THENCL=2ELSECL=3
70 PCLS:GOTO30
```

What is even more interesting is that other combinations of vertically spaced lines will produce colors other than red and blue. It is possible to get colors that approximate green, yellow, violet, and brownish tones.

That's all for now but if there are enough requests from the readership, there might be more.

1. http://home.att.net/~robert.gault/Coco/History/Semi24.htm

# Biorhythms
## by Diego Bariza

As you should know, your life is controlled by 3 biological cycles, known as "Biorhythms"
The physical, of 23 days, the emotional, of 28 days, and the intellectual, of 33.
This program allows you to monitor these cycles and help you take decisions that can change your life.

The routine in the lines 1000-1080 check that the entered dates are valid. If they are not, the "flag" variable F is set.
The one from 1100 to 1180 calculates how many days have passed between the date of birth and the date of prediction (this routine was taken from another program)
The main loop is from 500 to 610.
Values are calculated for 14 days before the prediction date to 15 days after it (500) 520 to 540 give values based on the length of the cycles that are later (550-600) used to calculate the position of the point in the graphic (the *80+81 is to center the graphic in the vertical axis)

```
0 ON BRK GOTO 750
10 WIDTH 40
20 CLS
30 PRINT:INPUT "Date of birth (M,D,Y)";MB,DB,YB
40 GOSUB 1000
50 IF F=1 THEN SOUND 1,1:GOTO 30
60 GOSUB 1100
70 BD=JD
80 PRINT:INPUT "Prediction date (M,D,Y)";MB,DB,YB
90 GOSUB 1000
100 IF F=1 THEN SOUND 1,1:GOTO 80
110 GOSUB 1100
120 PD=JD
130 LD=PD-BD
140
PD$=RIGHT$(STR$(MB),2)+"/"+RIGHT$(STR$(DB),2)+"/"+RIGHT$(STR$(YB),2
)
200 POKE 65497,0
210 HSCREEN 2:HCLS 2
220 PALETTE 9,50:PALETTE 10,55
230 X=25
240 HCOLOR 9:HLINE(25,0)-(319,81),PSET,BF
250 HCOLOR 10:HLINE(25,81)-(319,161),PSET,BF
260 HCOLOR 4:HPRINT(0,30),"Physical":HLINE(56,180)-(74,190),PSET,BF
270 HLINE(165,0)-(165,161),PRESET:HPRINT(17,21),PD$
280 HCOLOR 3:HPRINT(11,30),"Emotional":HLINE(168,180)-
(184,190),PSET,BF
290 HCOLOR 5:HPRINT(25,30),"Intellectual":HLINE(296,180)-
(312,190),PSET,BF
300 FOR A=25 TO 320 STEP 50:HLINE(A,0)-(A,161),PRESET:NEXT A
```

```
500 FOR PP=LD-14 TO LD+15 STEP .1
510 X=X+1
520 PC=PP/23-INT(PP/23):PC=PC*6.28+3.14
530 EC=PP/28-INT(PP/28):EC=EC*6.28+3.14
540 IC=PP/33-INT(PP/33):IC=IC*6.28+3.14
550 VP=SIN(PC)*80+81:VP=INT(VP)
560 HSET(X,VP,4):HSET(X,VP+1,4)
570 VE=INT(SIN(EC)*80+81)
580 HSET (X,VE,3):HSET(X,VE+1,3)
590 VI=INT(SIN(IC)*80+81)
600 HSET (X,VI,5):HSET(X,VI+1,5)
610 NEXT PP
620 POKE 65496,0
630 EXEC 44539
640 RUN
750 POKE 65496,0:END
1000 F=0 ' CHEQUEO FECHAS VALIDAS
1010 IF MB<1 OR MB>12 THEN F=1
1020 IF MB<8 AND MB/2<>INT(MB/2) THEN ML=31
1030 IF MB>7 AND MB/2=INT(MB/2) THEN ML=31
1040 IF MB=2 THEN ML=29
1050 IF ML=0 THEN ML=30
1060 IF DB<0 OR DB>ML THEN F=1
1070 IF YB<100 THEN YB=YB+1900
1080 RETURN
1100 REM COUNT THE DAYS
1110 W=0:IF MB<3 THEN W=-1
1120 JD=INT(1461*(YB+4800+W)/4)
1130 B=INT(367*(MB-2-W*12)/12)
1140 IF B<0 THEN B=B+1
1150 JD=JD+B
1160 B=INT(INT(3*(YB+4900+W)/100)/4)
1170 JD=JD+DB-32075-B
1180 RETURN
2000 REM BIORHYTHM
2010 REM YOUR LIFE IS CONTROLLED BY 3 CYCLES
2020 REM THE EMOTIONAL THAT CIRCLES EVERY 28 DAYS
2030 REM THE PHYSICAL, EVERY 23 DAYS
2040 REM AND THE INTELLECTUAL EVERY 33 DAYS
2050 REM WHEN THE GRAPHIC IS HIGH, YOU ARE DOING GREAT
2060 REM WHEN IS LOW, BE CAREFUL
2070 REM THIS PROGRAM SHOWS YOU THOSE CICLES FOR A 30 DAY
PERIOD
2080 REM CENTERED ON THE PREDICTION DATE
```

**CRYPTO.BAS  by Rodney Hamilton**

Have you ever solved a cryptogram on paper and wished there were an easier way?
Well, here it is!  This program won't solve the cryptogram for you, but it does handle all the routine character substitutions so you can concentrate on solving the puzzle.

The method that works best for me is to enter the cipher text in uppercase and do the substitutions from uppercase to lowercase.
The reverse-video text characters make the changes stand out well, and you can easily see what characters remain to be decoded.
Press CLEAR to erase all changes and start over.

```
0 'CRYPTOGRAM DECODING AID
1 'BY RODNEY HAMILTON 1981
10 CLEAR1000
20 CLS:PRINT"ENTER MESSAGE"
28 POKE282,255'UPPERCASE
30 LINEINPUTD$:N=LEN(D$)
32 POKE282,0'lowercase
40 A$=D$:E$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
50 CLS:PRINTA$:PRINT:PRINT" --";E$;"--"
60 PRINT"CHANGE: ";
70 B$=INKEY$:IFB$=""THEN70
80 IFASC(B$)=12THEN40
90 PRINTB$;" TO ";
100 C$=INKEY$:IFC$=""THEN100
110 PRINTC$
120 IFINSTR(1,A$,C$)=0THEN140
130 PRINTC$;" IS ALREADY IN USE":GOTO60
140 I=INSTR(1,A$,B$):IFI=0THEN160
150 MID$(A$,I,1)=C$:GOTO140
160 I=INSTR(1,E$,B$):IFI=0THEN50
170 MID$(E$,I,1)=C$:GOTO50
```

## Try these one liners, also by Rodney!

COLORS-3.BAS

```
1 IFI THENX=JOYSTK(0):Y=INT(JOYS
TK(1)/4):PRINT@0,"PALETTE"Z;TAB(
12)"COLOR"X;TAB(22)"SCREEN"J;:IF
BUTTON(0)THENPALETTEZ,X:GOTO1ELS
EZ=Y:A$=INKEY$:IFA$=""THEN1ELSEI
FA$="R"THENRGB:GOTO1ELSEIFA$="C"
THENCMP:GOTO1ELSEJ=J+1AND15:POKE
&HFF22,J*8:GOTO1
2 RGB:WIDTH32:SCREEN,0:CLS0:FORI
=1056TO1535:POKEI,255ANDI+96:NEX
T:POKE38345,127:POKE&HFF22,0:GOT
O1
```

COLORS.BAS

```
2 IFI THENIFINKEY$=""THEN2ELSEJ=
1-J:SCREEN0,J:GOTO2ELSEFORI=1024
TO1535:POKEI,255ANDI:NEXT:J=0:GO
TO2
```

### * * *  In Search of 256  * * *
### By Nickolas Marentes

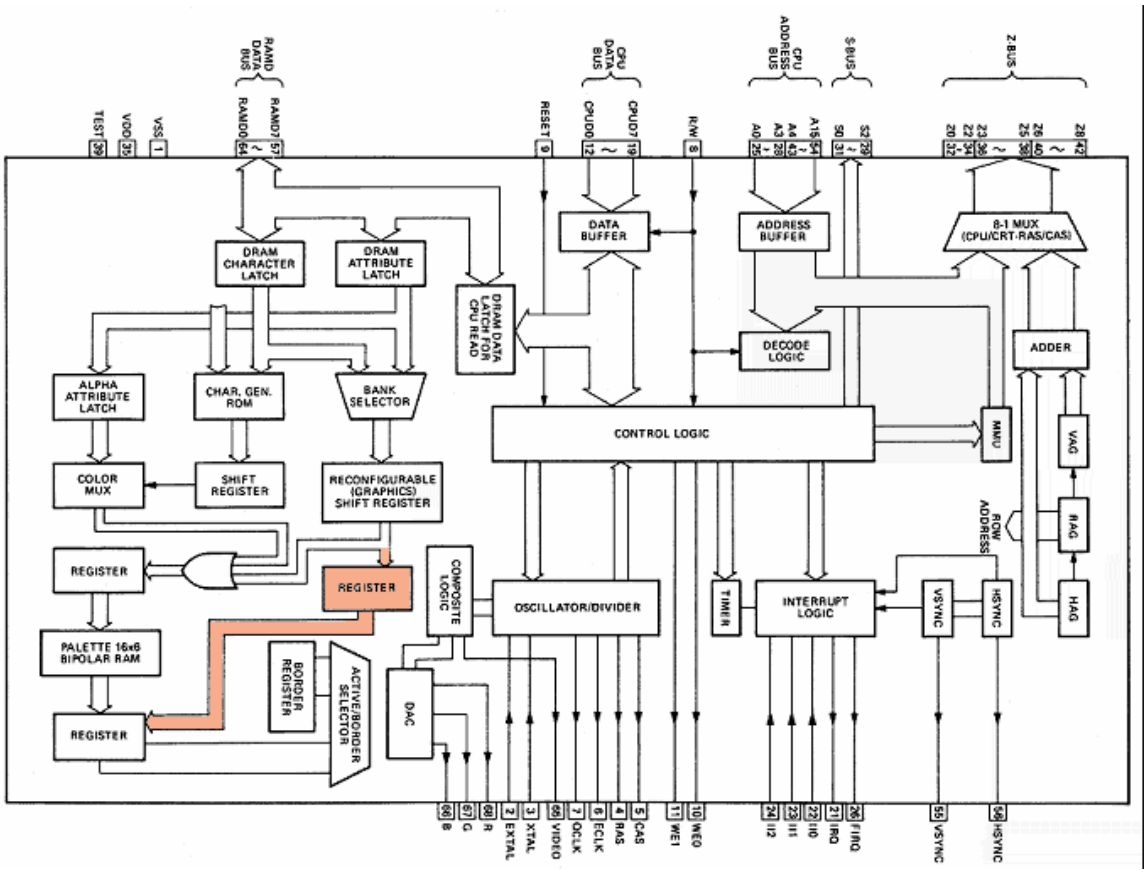
There have been rumours of a secret 256 color mode hidden within the GIME chip architecture of the Tandy Color Computer 3.

This page reveals all the information that I have collected over the last three years with the hope that one day the truth will be revealed.

**INTRODUCING THE GIME CHIP**

The GIME chip within the Color Computer 3 is an incredible piece of work. Not only does it almost completely emulate the functions of the 6847 VDG and 6883 SAM chip architecture of the earlier Color Computers but it adds a lot more! Advanced bank switching of RAM up to 512K, advanced interrupt handling and many more sophisticated graphics modes.

The Color Computer 3 with its GIME chip has expanded upon the earlier models with their more limited 128x192 four color and 256x192 two color modes. The GIME chip has added new modes of 16 colors with a resolution up to 320x225 and 2 color modes with a resolution up to 640x225, the colors being selectable from a total palette of 64.




## AN INTERESTING OBSERVATION!

In 1999, I pointed John Kowalski's attention to the block diagram of the GIME chip on page 96 of the Tandy Color Computer 3 service manual.
John made an interesting observation that got us both curious. The diagram shows that the output of the "Reconfigurable (Graphics) Shift Register" is fed through a gate, followed by a "Register" and then on to the "Palette 16x6 Bipolar RAM", finally ending up at another register then on to the DAC for output. This seems sound and as one would expect. What got John suspicious was that the output of the "Reconfigurable (Graphics) Shift Register" ALSO routed to another unlabeled "Register" which appeared to bypass the "Palette 16x6 Bipolar RAM" and fed in the register that follows it. John came with the suspicion that this register allowed 8 bit data from the shift register directly past the 64 color palette RAM!!

We both puzzled over this but and no one had any ideas of what it could be. Suddenly I came into contact with a very prominent ex-radio shack employee who had a very close association with the development of the Color Computer 3 via one of the CoCo IRC chat sessions. To say that his question to me..."Have you found the 256 color mode?" ...raised the hairs on my back was a gross understatement!!

**"HAVE YOU FOUND THE 256 COLOR MODE?"**

*Here is an edited transcript of what he had to say about this mode...*

*Have you found the 256 color mode?*

There is a real 256 color mode in there. I was hoping someone would discover it. It's a real byte level pixel, 320x200 mode. It uses a yyyyyrgb format, 5 bits of intensity, 3 bits of color.

The mode is rather complicated to get to and did not work reliably on the first run of GIME chips and it’s been too long, I don't remember the sequence anymore and I got overruled by Roach on the disclosure of the mode because it was too much competition with the 1000. It's really the last secret in the Color Computer 3.

You can only have one 256 color page but it was very complicated to set up. It could view 512K and that was one of the problems because of the page at the top of memory. I was not allowed to tell anyone exactly what to look for either.

The only thing I remember about how to get into it is that you have to switch the mode while you are in the IRQ in the protected page at $FE00. It had to be set as a mode on the IRQ, it didn't use the IRQ. While in that routine from an IRQ there is a machine state that changes the meaning of the graphics mode. I don't have a clue what that pattern is anymore.

John Prickett ran out of pins to make it work so it reused some of them in while in the IRQ state.

*I, along with John Kowalski, was able to get in touch with this person for a second time to get a few more questions in and here is the edited transcript ...*

[MR X] It is a machine state, so it does require the right set of events and I am probably the only one who ever turned it on and played with it. My demo program put up changing color patterns. It was designed to be for games.

[JOHN] So, you're saying that an IRQ routine that 'itself' is located in the $FEOO-$FEFF page is used to switch on the mode?

[MR X] That's correct, John and as I remember it's the only place you can do it.

[JOHN] Any hints as to what generates the IRQ signal? Does it matter?

[MR X] I did it on the timer I think.

[NICK] Is the 256 mode visible in both RGB and composite video output?

[MR X] I believe so, it’s a real mode.

[JOHN] So if I set up a TIMER based IRQ that jumps to the $FExx range... If I systematically trash random hardware registers $FFOO to $FFFF, eventually the mode should turn on.. shouldn't it?

[MR X] It sees to me some of the regs had to be written with a given interval so random data is not going to do it and I can tell you the routine is not in Tandy archives anymore either.

[JOHN] What would be the reason for the program code to be in $FExx range? It's not like the GIME can tell where the CPU's PC is.

[MR X] That's not true John, there are times the GIME does know. AII the coco files were destroyed 2 years before I left Tandy. No, once in the mode the computer acts as it

did before, but no way to exit the video mode without a reset because the MMU is part of the addressing so the MMU has to intercept the address.

[JOHN] So.. the GIME remembers the last address issued on the CPU bus, and if that address is not exactly 'some number', the mode will not activate?

[MR X] The special page is treated differently by the MMU and the memory map becomes unstable during the mode change so it has to be one in that page system needs to be in the RAM mode.

[NICK] MMU on or off?

[MR X] On.

[NICK] CoCo1/2 compatible mode on or off?

[MR X] CoCo 3 mode.

**GIME CHIP DESIGNER FOUND!**

My next avenue of pursuit was to locate the man responsible for the design of the GIME chip and I was given the name, John Prickett. My search on the internet located a John Prickett working for AMD at the time and here is his response to my question...

*I'm John Prickett, and I did design the chip you referenced, but that was 14 years and 50 designs ago. I couldn't possibly tell you any of the internal intricacies of the chip now. You probably know more about it than I do right now.*

*I do seem to remember that it was 64 colors only -- we only used a 6-bit palette, and so I very much doubt that there is a 'hidden 256 color' mode. But I could be wrong.*

*Sorry I couldn't be more help, John*

John appears to have moved on and I have lost contact with him.

**MORE PROOF!**

During my job as PennFest 2000 co-organiser, I came into contact with Mark Hawkins of Microware. Mark is one of the 3 people who developed the CoCo3's Super Extended Basic and is one of the "Three Mugsters" that appears on the CTRL-ALT-RESET page. I asked Mark about this mode and he was unfamiliar with it but he said that he would look through the archives at Microware for old documentation about the GIME chip. He found the original "Color Computer Custom Video Proposed Feature List" document which came from Tandy's R&D section. All of the info is as found in the service manual except for the cover page which lists the 256 color mode.

This proves that the mode was at least in the planning stages. What we are unsure of from this page is what the references are about 1 and 2 banks of RAM and how that affects the modes. We have seen references to this in CoCo3 memory maps but it has always been ignored.

(you can see this on my website where it is posted at www.

**FF91: INITIALIZATION REGISTER 1**

Bit 7 - 0 = Two banks of DRAM

Bit 6 - 0 = 64K chips, 1 = 256K chips

Bit 5 - Timer Input Clock Select 1=70ns, 0=63ns

Bit 4 - NOT USED

Bit 3 - NOT USED

Bit 2 - NOT USED

Bit 1 - NOT USED

Bit 0 - MMU Task Register Select

With closer scrutiny of this R&D document, we find only one additional piece of information that is not to be found in the normal CoCo3 memory maps. On page 7 of this document that covers the CoCo1/2 compatible registers of $FF20 to $FF23, we find handwritten at the bottom of the page the following...

*"FF27 Will also be used for power-up system configurations. Bit definitions are TBD."*

Could this be one of the mystery registers or was it an abandoned idea?

**INFINITE SCANLINES MODE**

Another interesting observation that has me curious (but may not be related to the 256 mode) is the function of BITS 6 and 7 in the FF99 Video Resolution Register. The functions are listed in the CoCo3 memory maps as...

**Lines per Field**

| BIT 1 | BIT 0 | |
|---|---|---|
| 0 | 0 | 192 |
| 0 | 1 | 200 |
| 1 | 0 | 210 |
| 1 | 1 | 225 |

The Lines Per Field is the number of vertical scan lines or vertical resolution of the mode. All of these work fine except for the 210 setting.
This does not create 210 lines. Instead it appears to create an infinite number of lines. With this setting, the display stretches past the top and bottom and it seems to pick whatever color the GIME was processing at the time the mode is set. If it was in the border region of the screen, it keeps generating this color otherwise it repeats the current color in the active portion of the screen.

**CURRENT STATUS**

John and I have run tests on the various GIME chip locations, running random data and experimenting with all the areas of the map that are marked as undefined or not used. The mode, if it truly exists, does not take the form of a normal register function. As our friend on the IRC chat explained, the designers ran out of pins to make it work properly and had to re-use pins during a specific machine state of the GIME. This makes it extremely difficult to locate because even if we are looking at the right places to setup the mode, the mode is non-functional and therefore will not present itself unless this specific state is met.

We need more information and vital clues in order to proceed further.
What are those 2 registers that appear to be part of some bypass of 8 bit video data in the GIME chip and how are they accessed? Could the mode have been removed from final production GIME chips or does the mode still exist but has been carefully "hidden" under the request of Tandy management so as not to conflict with the sales of the Tandy 1000 line of PC compatibles in the product line at the time.

For more information on this subject, visit Nick’s website at:
http://members.optusnet.com.au/nickma/ProjectArchive/index.html

---

# "Ninja Warrior!"

**Game author: Charles Forsythe**

**Publisher: The Programmer's Guild**

**Runs in PMODE 1 screen on any 32K+ CoCo**

**Review by Richard Kelly**

**Introduction:** You play the role of the title character, moving left and right, jumping over pits, and getting past rocks either by jumping over them or by crushing them with your foot or your trident. Hmmm … jumping over pits, and destroying or evading rocks. Sound familiar? It should. Just as Pac-droid was a variation of Pac-man, the game Ninja Warrior is a variation of Moon Patrol.

After passing the first level, newer objects and/or enemies appear that you'll have to get past. There'll be fireballs that start out on the ground, and after the level they first appear in, the fire starts falling from the sky. Time it right, and you can actually destroy the fire with your trident before it hits the ground.

On the later levels, rocks will appear in pairs; that is, one rock stacked on top of another. Crushing these two rocks together is a bit trickier; you have to press the Attack key (which would be Spacebar in Keyboard Mode), and time your attack so that your weapon hits the top rock, and then your foot hits and crushes the bottom one.

On higher levels, enemy Ninjas will appear. Like the rocks, you can either jump over them or get rid of them. But this time, only your trident alone can annihilate these turkeys.

A unique feature about "Ninja" is that the game defines the levels not as "Level 1, 2, 3, 4", etc., but as a challenge level for different-colored belts, such as Challenge Level for Yellow Belt, Challenge Level for second Black Belt, etc.. That last level I mentioned is the farthest I've ever gotten, and it takes a lot of time, patience, and skill to get there.

**Graphics:** *C*. The graphics do the job, but they never do it well. I've seen PMODE 1 graphics that look pretty good, but that's not from this game. The graphics come off as simplistic at best, but at least the images aren't appalling. And I've seen far worse. The fact is, the Ninjas look like Ninjas. The rocks look like rocks. The fireballs look like fireballs. You won't mistake one image for something it's not. That's basically what's important in the long run.

**Animation:** *C+*. There's just enough animation to carry out the job. In other words, the Ninja walks along the playfield, and there's enough animation there to make you believe that he's walking across the playfield. So of what animation is there, it's done right.

**Sound:** *A-*. This is one of the better aspects of the game. It has very fun sound. It's part of what makes this game an enjoyable experience. You hear your Ninja waltzing across the playfield when he's on the ground. You hear neat little crush noises when you destroy a rock. You hear a very quick, two-note melody when you score points in particular ways. The only part of the sound that isn't above average is the "end level" melody, and wouldn't you know it - you hear that tune every single time you pass a level. That end-level melody can get a little old, especially after passing enough levels to try for your second Black Belt.

**Game Engine:** *C-*. This is probably where the game is most at fault. The more objects there are on the screen at one time, the slower the action moves, although the frame rate never actually crawls along.

The objects flicker incessantly, even if they are standing still. Even the pits below flicker. Oh, well. At least the sprites don't flicker so much as to distract the player from game play.


But certainly the most annoying part of the game is the title screen, High Score screen, and Player Select screen. Now, don't get me wrong; they *look* fine. But *using* them is another story. These screens have that infamous, annoying UI setup where you can't get the program to respond to the same key twice in a row. Let's say you press the Enter key after the high scores are displayed. When the title screen is drawn, you can't go to the next screen by pressing Enter again. You have to use a different key on the keyboard. And you shouldn't use one of the number keys to go to the Player Select screen, because then you might be unable to select the correct number of players to start the game! And you re-enter these screens every time you lose your last man, don't forget. Anyway, a thousand demerits for this very annoying program glitch.

**Gameplay:** *A-*. All objects chunk around the screen a little, but the game is still pretty playable. There's new enemies and/or scenarios to face with almost every new level, which always makes the player anxious to do better and better in the game; he wants to make more progress.

Since the layout of the objects in each level is random, different results are possible every time you play. An enemy Ninja could appear near the beginning of a level, or near the end. He can appear in a part of the level where it's easy to get him, and sometimes he shows up when the level has left you off-guard, and you're in no position to defend

yourself. So sometimes you're ready for him, sometimes not. That keeps the game play interesting. And it keeps you on your toes.

It also helps that there's many enemies with more than one way to get past them. Suppose an arrow shoots at you. You have the option of destroying it with your trident, or jumping over it. Jumping it is the easiest thing to do, but if you're good enough - and can react quickly enough - you can destroy it instead and get more points.

Also, I don't think this was quite intentional by the author, but if you time it right, you can jump when you're near a rock, make points off of jumping over it, and then make points by crushing that same rock with your foot right when you're landing on top of it. Another scoring feature (a definite bug, I'm sure) is where you can jump the beginning of the "finish line" for the level, and the game awards you a few points for such an action. "Ninja" leaves you with all sorts of ways to make extra points if you just take the time to fine-tune your game-playing skill. You can make it past the highest level in the game, then play again and try to top that feat by trying to make more points and still pass as many levels as before. Of course, you have to accomplish this task in a completely different set of level layouts than in the last game, so you can't rely on your memory to guide yourself through the trickiest parts. Talk about replay value!

And I really like this additional feature - No joystick required. You can play with a joystick if you like, but you can also play by using the keyboard. This is a big plus for me, since my last couple of joysticks kept botching up on me big time.

**Overall:** *A*-. Let's do a quick summary of the game's ups and downs.

**The ups:** This takes the game play of Moon Patrol and expands upon it in ways you'd only dream. There are new enemies in almost every single level. The game has great sound and a high replay value.

**The downs:** The "level completed" theme gets too old too quickly. The graphics are average at best. The game has a fairly chunky game engine that lags a little at times. But the worst flaw of all is definitely that "don't press the same key twice" problem with the High Scores, Title, and Player Select screens.
In a nutshell, "Ninja Warrior!" doesn't quite make it into the "Must-Play" category of CoCo games. Nevertheless, it's a very good game as-is, and I'd recommend checking it out.

Thanks to everyone who made this and the other issues possible. I hope to see the newsletter keep growing until I can have one for each month or every other month. Feel free to email me at coconutznewsletter@yahoo.com for questions, comments, or submission info. I want to give a special thanks to Roger and CoCo3.com for hosting The newsletter!

Editor in Chief
Mary Kramer