



*The third in a series of tutorials for the  
beginner to intermediate machine  
language programmer*

# Machine Language Made BASIC

## Part III: What a Dump!

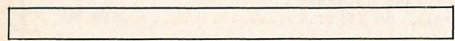
By William P. Nee

**D**umps are simply programs that transfer images from the screen to paper by way of a printer. The programs shown in Listings 1 and 2 are for a seven-dot printer that adds 128 to the total value of the dots used for graphic printing. Listing 1 is the BASIC version and Listing 2 is a machine language version. The seven dots in a column have a value of 1, 2, 4, 8, 16, 32, and 64, starting from the top. The values of the dots you want to print are added together and then added to 128. The total value is sent to the printer as a CHR\$ value. Printer commands used in this program are:

- CHR\$(18) - set for graphics
- CHR\$(27); CHR\$(16) - position the print head
- CHR\$(0); CHR\$(50) - 50 spaces over
- CHR\$(30) - end graphics mode

Check your printer manual for any changes to these CHR\$.

Location \$6F tells the computer



*Bill Nee bucked the "snowbird" trend by retiring to Wisconsin from a banking career in Florida. He spends the long, cold winters writing programs for his CoCo.*

which device will display or receive information as follows:

<b>\$6F</b>	<b>Device</b>
-2(#\$FE)	printer
-1(#\$FF)	tape recorder
0	screen
1 - 15	disk

Locations \$BA and \$B give the location of beginning graphics — usually at \$600, or \$E00 with disk.

The BIT command is a quick way to test each bit in a byte and branch accordingly. We will check each bit to see if it is a zero, and we'll branch if it is. The BIT command ANDs a number in registers A or B with any other number you select; but unlike the AND command, the number in registers A or B remains unchanged — only the values of the condition codes (CC) register are affected. The CC register is the register to which all branches look to see if the conditions for a branch have been met (plus, minus, equal, zero, etc.). The rules for AND are:

0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

Another way to think of this is: any number AND 0 = 0, any number AND 1 = the same number.

Now, how do we test the left bit (Bit

7) in Register A to see if it is a 0 or a 1? We must BITA with the number 128. This is easier to see when written in the binary format (Base 2):

let Register A = 149 = 10010101
BITA with #128 = 10000000
CC register = 10000000

Since the result is not 0, the CC register will not be set to 0 and a BEQ (Branch if Equal to 0) will not execute, so the program will continue with its next command. If you continue to BITA with 64, 32, 16, 8, 4, 2 and 1, you can check each bit and branch accordingly.

Since the printer can type a column seven bytes high, this program will look at the left bit in each of the seven stacked bytes, then the next bit over, and the next, etc., until reaching the right bit (Bit 0).

If a bit is 1, the value of the CHR\$ to be printed is increased by the dot's value according to its location in the column:

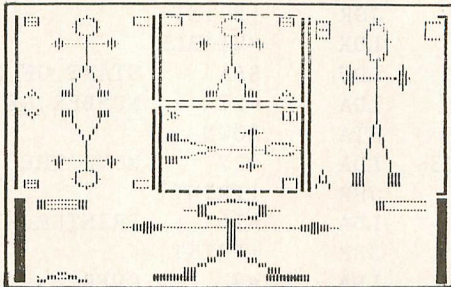
Dot	Value
1	1
2	2
3	4
4	8
5	16
6	32
7	64

Finally, 128 is added to the total value and the result is sent to the printer as a CHR\$(value). The routine at \$A002 sends the value in Register A to whatever device is indicated by Location \$6F. The brackets around PRINT in the program mean to use the values in \$A002 and \$A003 as the JSR address. Those values may differ in various Color Computer models, but the \$A002 should remain the same.

The end of the program uses CLR \$6F to set Location \$6F to 0, the device number for the screen. CLR is the fastest command available to set any memory byte or single-byte register such as A, B or CC to 0. CLRA executes more quickly and uses less memory than LDA #0.

In one portion of the program we used LBNE (Long Branch if Not Equal) instead of BNE. This is because a regular Branch can only move backward -128 spaces or forward +127 spaces (remember "signed" numbers?); a Long Branch can branch anywhere in the program but consumes a little more memory. Use the regular branch whenever possible; EDTASM+ will let you know if a Long Branch is necessary. It is good, though, to use JSR when referring to ROM routines.

The BASIC program for the "dump" takes advantage of the BASIC PPOINT command to see if each bit in the seven-byte column is set or not. If it is set, the value of the CHR\$(value) is increased and 128 is added to the total value. The resulting character is sent to the printer.



Sample printout using a Radio Shack DMP-105

The PPOINT method could have been used in the machine language program, but it is still a slow command. We will, however, use this command in future articles to write programs involving graphics.

Both programs will run for a while without printing anything since the printer buffer must be filled before printing. The buffer stores what the computer has been sending it until

ready. Then it prints it all at once, rather than printing out one CHR\$(value) at a time. Both programs also skip printing the bottom three lines of graphics. You can add your own routines if you want them. Be sure to clear space before running the machine language program from BASIC: CLEAR 200, &H3000-1.

The time difference between the two programs is amazing. The BASIC program can take up to twenty minutes to copy a graphics page, while the machine language program can do the job in about three minutes. It's a longer program, but if you're doing a lot of repetitive dumping (Christmas cards, for example) it is quite a time-saver. The machine language program is designed for PMODE 4, since we're not using a color printer.

Try experimenting with a program that dumps from top to bottom of the page rather than from left to right. This prints the picture sideways but allows you to double its size. You might also try to reverse the picture.

(Questions or comments concerning this tutorial may be addressed to the author at Route 2, Box 216C, Mason, WI 54856-9302. Please enclose an SASE when requesting a reply.) □

#### Listing 1: DUMPBAS

```

10 CLEAR200,&H3000-1
20 'SAMPLE GRAPHICS PROGRAM
30 PMODE 4,1:PCLS5:SCREEN1,1
40 FOR X=0 TO 254 STEP 2
50 LINE(X,0)-(255-X,191),PRESET
60 NEXT
70 FOR Y=190 TO 0 STEP -2
80 LINE(0,Y)-(255,191-Y),PRESET
90 NEXT
100 DRAW"C5BM80,60M+6,+10E5F5M+6
,-10BR6D5ND5R18NU5D5BR6E10F6NL12
F4BR16U10NL10R10"
110 DRAW"BM120,100E10F6NL12F4"
120 DRAW"BM80,130U10R18F2D6G2NL1
8BR8BU2NU8F2R14E2U8BD10BR6M+6,-1
0F5E5M+6,+10BR6U10R18F2D3G2L18"
130 'THE DUMP PROGRAM
140 'EXEC &H3000 OR -
150 PRINT#-2,CHR$(18)
160 FOR V=0 TO 182 STEP 7
170 PRINT#-2,CHR$(27);CHR$(16);C
HR$(0);CHR$(50);
180 FOR H=0 TO 255:P=0
190 FOR N=0 TO 6
200 IF PPOINT(H,V+N)<>0 THEN P=P
+2^N
210 NEXT N
220 PRINT#-2,CHR$(P+128);:NEXT H
230 PRINT#-2:NEXT V
240 PRINT#-2,CHR$(30)

```

HAWKSoft HAWKSoft HAWKSoft HAWKSoft HAWKSoft

DOMINATION.....\$18.00  
MULTI-PLAYER STRATEGY GAME!

Try to take over the planet of YCNAN. Battle other players armies to take control of their provinces and defend yours. Play on a Hi-res map of the planet. Take the "RISK" and be a planet-lord today!!! Requires 1 disk and joystick or mouse. See Rainbow Review JULY 88

MYDOS.....\$15.00

CUSTOMIZABLE! EPROMABLE!!  
The commands Tandy left out!

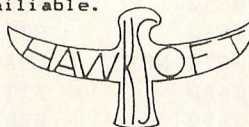
MYDOS is an enhancement to Disk Extended Basic 2.1 on the CoCo 3. One command loadm and execute for M/L programs. Lowercase command entry and display on ALL screens. Screen echo and SAY command for RS Speech Pak. Point and click mouse directory. NEW FEATURES!!!!

Supports double-sided and 40 track drives. Set any palettes you want on power-up (RGB or CMP). Power-up in any screen width and colors (or monochrome) you wish! More options than you can shake a joystick at!!! See Rainbow Review JUNE 87

HAWKSoft KEYBOARD CABLE.....\$25.00

UNCHAIN YOUR KEYBOARD!

Five foot extender cable for CoCo II and 3. Move your keyboard where you want it! Installation instructions and tips included! Custom lengths available.



HAWKSoft P.O. Box 7112  
Elgin, IL 60121-7112  
312-742-3084

S&H always included. Il. orders add 7% sales tax.

Listing 2: DUMPBIN

```

3000          00100      ORG      $3000
          A002      00110 PRINT EQU      $A002      PRINT ROUTINE
3000 86      FE      00120 START LDA      #-2      USING THE PRINTER
3002 97      6F      00130      STA      $6F
3004 86      12      00150      LDA      #18      TEXT TO GRAPHICS
3006 AD      9F A002  00160      JSR      [PRINT]
300A 8E      30CC   00170      LDX      #VTABLE
300D DE      BA      00180      LDU      $BA      START OF GRAPHICS
300F 86      1B      00190      LDA      #27      NUMBER OF ROWS
3011 B7      30CA   00200      STA      DOWN
3014 86      1B      00210 LOOP3  LDA      #27      MOVE THE
3016 AD      9F A002  00220      JSR      [PRINT]
301A 86      10      00230      LDA      #16      PRINTER HEAD
301C AD      9F A002  00240      JSR      [PRINT]
3020 86      00      00250      LDA      #0      OVER
3022 AD      9F A002  00260      JSR      [PRINT]
3026 86      32      00270      LDA      #50     50 SPACES
3028 AD      9F A002  00280      JSR      [PRINT]
302C 86      20      00290      LDA      #32     BYTES PER LINE
302E B7      30C9   00300      STA      ROW
3031 C6      08      00310 LOOP2  LDB      #8      BITS PER BYTE
3033 7F      30CB   00320 LOOP1  CLR      VALUE
3036 A6      40      00330      LDA      0,U     FIRST BIT IN THE COLUMN
3038 A5      85      00340      BITA    B,X     AND A WITH B,X
303A 27      03      00350      BEQ     NEXT2
303C 7C      30CB   00360      INC     VALUE
303F A6      C8 20   00370 NEXT2  LDA      32,U     SECOND BIT IN THE COLUMN
3042 A5      85      00380      BITA    B,X     AND A WITH B,X
3044 27      08      00390      BEQ     NEXT3
3046 B6      30CB   00400      LDA     VALUE
3049 8B      02      00410      ADDA   #2
304B B7      30CB   00420      STA     VALUE
304E A6      C8 40   00430 NEXT3  LDA      64,U     THIRD BIT IN THE COLUMN
3051 A5      85      00440      BITA    B,X     AND A WITH B,X
3053 27      08      00450      BEQ     NEXT4
3055 B6      30CB   00460      LDA     VALUE
3058 8B      04      00470      ADDA   #4
305A B7      30CB   00480      STA     VALUE
305D A6      C8 60   00490 NEXT4  LDA      96,U     FOURTH BIT IN THE COLUMN
3060 A5      85      00500      BITA    B,X     AND A WITH B,X
3062 27      08      00510      BEQ     NEXT5
3064 B6      30CB   00520      LDA     VALUE
3067 8B      08      00530      ADDA   #8
3069 B7      30CB   00540      STA     VALUE
306C A6      C9 0080 00550 NEXT5  LDA      128,U    FIFTH BIT IN THE COLUMN
3070 A5      85      00560      BITA    B,X     AND A WITH B,X
3072 27      08      00570      BEQ     NEXT6
3074 B6      30CB   00580      LDA     VALUE
3077 8B      10      00590      ADDA   #16
3079 B7      30CB   00600      STA     VALUE
307C A6      C9 00A0 00610 NEXT6  LDA      160,U    SIXTH BIT IN THE COLUMN
3080 A5      85      00620      BITA    B,X     AND A WITH B,X
3082 27      08      00630      BEQ     NEXT7
3084 B6      30CB   00640      LDA     VALUE
3087 8B      20      00650      ADDA   #32
3089 B7      30CB   00660      STA     VALUE
308C A6      C9 00C0 00670 NEXT7  LDA      192,U    SEVENTH BIT IN THE COLUMN
3090 A5      85      00680      BITA    B,X     AND A WITH B,X
3092 27      08      00690      BEQ     PRNT
3094 B6      30CB   00700      LDA     VALUE
3097 8B      40      00710      ADDA   #64

```

3099 B7	30CB	00720	STA	VALUE	
309C B6	30CB	00730 PRNT	LDA	VALUE	
309F 8B	80	00740	ADDA	#128	FILL IN THE EIGHTH BIT
30A1 AD	9F A002	00750	JSR	[PRINT]	
30A5 5A		00760	DECB		
30A6 26	8B	00770	BNE	LOOP1	FINISHED THE BYTE?
30A8 33	41	00780	LEAU	1,U	MOVE OVER TO THE NEXT BYTE
30AA 7A	30C9	00790	DEC	ROW	FINISHED THE ROW YET?
30AD 26	82	00800	BNE	LOOP2	
30AF 86	0A	00810	LDA	#10	CARRIAGE RETURN
30B1 AD	9F A002	00820	JSR	[PRINT]	
30B5 33	C9 00C0	00830	LEAU	192,U	SKIP DOWN 7 ROWS
30B9 7A	30CA	00840	DEC	DOWN	
30BC 1026	FF54	00850	LBNE	LOOP3	
30C0 86	1E	00860 FIN	LDA	#30	BACK TO TEXT
30C2 AD	9F A002	00870	JSR	[PRINT]	
30C6 0F	6F	00880	CLR	\$6F	BACK TO THE SCREEN
30C8 39		00890	RTS		BACK TO BASIC
30C9		00900 ROW	RMB	1	
30CA		00910 DOWN	RMB	1	
30CB		00920 VALUE	RMB	1	
30CC	0001	00930 VTABLE	FDB	\$0001	
30CE	0204	00940	FDB	\$0204	
30D0	0810	00950	FDB	\$0810	
30D2	2040	00960	FDB	\$2040	
30D4	80	00970	FCB	\$80	
	3000	00980	END	START	

00000 TOTAL ERRORS

## BOWLING LEAGUE SECRETARY ©1986



Reviewed  
 Sept. 1986 April 1987  
 pg 141 pg 140

Now for the Co-Co 1, 2 or 3 with disc drive, printer, 32K.

- User friendly—full menu driven selections.
- Any number of teams, and over 200 bowlers.
- Calculates and stores all team and bowlers stats.
- Men, women, mixed, scratch or handicap; blinds and substitutes.
- Start up any time in season.
- Full edit capability.
- Automatic backups and weekly, mid-season and end-season resets.
- ABC/WIBC style printouts.
- Includes 20-page instruction manual.
- Upgrade for individual tally sheets. (\$9.95 separate; free when ordered with program.)

Priced at **\$49.95** including Shipping, Handling & Sales Tax. To order, send check or M.O.

Specify Version number (1.0 for men or women; 1.1 for mixed) and number of disc drives.

**TOMELA\*CO**

P.O. Box 2162 • Doylestown, Pa. 18901-2162 • (215) 348-5822

# HELLO/BAS

A DISK DIRECTORY UTILITY by Roy C. Pierce (c) 1988

### WHAT HELLO WILL DO

- \* Display Alphabetically Sorted Directory of any Drive. (0-3)
- \* Print a Hardcopy of Sorted Directory w/Date and Disk Name.
- \* Run ANY BASIC Program with Ease. RUNS ON ANY COCO. (32K Disk Extended BASIC Required)
- \* Single Key Stroke Commands.
- \* Easy to Read Display.
- \* ALL BASIC so it won't mess up your System.
- \* SUPER FAST OPERATION.
- \* Reads Any Drive at Will.
- \* Low Disk Overhead - Only 1 Gran.
- \* Easy to Copy to All your Disks, comes with Handy Diskinit Utility for Auto-booting HELLO/BAS.

**\$19.95 U.S.**

**\$22.95 CDN.**

**INTERNATIONAL \$22.95 U.S.**  
 SHIPPING & HANDLING INCLUDED

**RCP PIERCE**  
**SOFTWARE**

**P.O. BOX 1787,**  
 Main Post Office,  
 Edmonton, AB, Canada  
 T5J-2P2

**PH: (403) 474-8435**