ENGINEERING NOTES
on
Radio Shack Color Computers

JULY 1985
Vol. 2 No. 6

$1.95

DYNACALC

COLOR

NEWS

* CHARACTER GENERATOR PROGRAM
* ALARM CLOCK PROGRAM
* 64K RAM PROGRAM

* LARGE MEMORY PROGRAMS (Part 6)
* WRITING PROGRAMS (Part 5)
* BASIC PROGRAMMING
* COMPUTER GRAPHICS (Part 6)
* NEW PRODUCTS
* PRODUCT REVIEWS

DYNAMIC COLOR NEWS is published monthly by DYNAMIC ELECTRONICS, INC., P.O. Box 896, Hartselle, AL 35640, phone (205) 773-2758. Bill Chapple, President; Alene Chapple, Sec. & Treas.; John Pearson, Ph. D. Consultant; Bob Morgan, Ph. D., Consultant.

 Subscriptions are $15/yr for U.S.A. & Canada, $30 other foreign.

The purpose of this newsletter is to provide instruction on Basic & Machine Language programming, Computer theory, operating techniques, computer expansion, plus provide answers to questions from our subscribers.

The submission of questions, operating hints, and solutions to problems to be published in this nesetter are encouraged. All submissions become the property of Dynamic Electronics if the material is used. We reserve the right to edit all material used and not to use material which we determine is unsuited for publication.

We encourage the submission of Basic and Machine Language Programs as well as articles. All Programs must be well documented so the readers can understand hov the program works. We will pay for programs and articles based upon their value to the newsletter. Material sent will not be returned unless return postage is included. Basic & ML programs should be sent on a tape or disk & comments should be sent as a DAT or BIN file.

****
* *
* DYNAMIC COLOR NEWS *
* *
* JULY 1985 *
* *
* Editor and Publisher *
* Bill Chapple *
* *
* Secretary *
* Deanne Hill *
* *
****

CONTENTS

EDITOR'S COMMENTS

Each month our excitement grows because of the increasing numbers of accessories at reduced prices that are available for COCO owners. We have been involved with Color Computers for about 3 years. My first computer had 4K of RAM and now I am using a 256K Thunder RAM upgrade from Spectrum Projects. We were the first to offer 128K upgrades in the Fall of 1983. They were not very well accepted at first but now there is much interest in larger memories. With the prices of memory chips dropping everyone can afford at least 64K.

In our large memory program series we are discussing methods for utilizing large memories. This month we show how to configure the 64K computers for the ALL-RAM mode. This allows you to modify the ROM software as well as provides about 8K of additional memory.

We improved our graphics character generator program to allow 3 character sizes of 8, 16, and 32 characters per line. Additional instruction is included. So if you did not understand what we did last month maybe you will after reading our editorial.

In an earlier isssue we commented about the heat problem in Color Computers. As a result we received an editorial from Dennis Kitsz President of Gree, Mountain Micro. He was kind enough to send us a copy properly justified so that we could use it without having to do any retyping. His article and our comments on this subject are included. Thanks Dennis for your support. We would appreciate receiving more letters from you our readers. So take a few minutes and write us and let us know what you think about the newsletter and what subjects you would like for us to cover.

We are continuing our series on Basic Programming. For you who are learning you should begin to feel a little more comfortable writing programs.

This month we started our product review section. We are reviewing one product from Spectrum Projects. We need more products to review and if you produce or sell color computer products then send them to us for a free review. Next month we will be reviewing "Thunder RAM" from Spectrum Projects. If you want to know what to expect from this inexpensive 256K RAM then you will want to see our review. To be fair we will not be reviewing our own products. However we will mention them in our New Products section.

LARGE MEMORY PROGRAMS
(PART 6)

In the previous issues in this series we showed how to configure a 64K computer so that Basic programs could be run in both 32K memory banks. For writing large programs we showed how to put part of the program in one bank and put the rest in the second bank. Machine language subroutines that could be called from Basic were given with examples showing how to use them.

Last month we showed how to put data in the second 32K memory bank. We gave an address file example program where the addresses were stored in the second 32K memory bank.

In our first issue, Vol. 1 No. 1 we showed how to stack programs in the computer and select the program to run from a menu. We called this a multi-program manager. An advantage of this is that information does not have to be moved as is the case with a disk drive. Also software is not required. Let's

briefly review how to stack programs in a memory bank.

Stacking Programs

Basic programs have vectors that tell where the program begins and ends. The beginning vector is in locations 25 and 26. To convert vector components to values we multiply the lower memory value by 256 and add the higher memory value. If the values stored in 25 and 26 are 38 and 1, then the basic program starts at 38 * 256 + 1 or 9729. The end of the program can be calculated similarily by using the vector components in locations 27 and 28.

Preceeding each Basic statement is a "0". Therefore in location 9728 there will be a "0". To stack programs we can poke different values into the Basic beginning and ending vectors. Suppose we loaded in a basic program and the values in locations 25-28 are as follows:

38, 1 49, 199 where the number before the comma is the most significant byte and the number after the comma is the least significant byte.

If we want to add an additional program in higher memory then we can leave space to protect our previous program and start the new program. Since the first program ended at 49, 199 if we let the next program start at 50, 1 then there will be some space between the programs. To be safe perhaps we should leave a little more space so let's let the next program start at 51, 1 (add 2 to 49). This leaves at least 256 bytes between programs which is enough for minor editing of the program.

The value in location 26 is always a "1". Write down the values in 25-28 so that you can restore them should you want to edit the first program. To initialize the second program we need to put a "0" in the byte preceeding the program. POKE 256 * 51, 0 to do this. Now enter "NEW" and you are ready to load the second program.

Switching Programs

To go to another program just poke the value of the most significant byte (MSB) in 25. Then you can RUN the first program. If it needs editing then all of the values from 25-28 need to be restored before you do the editing.

For running the programs you need only to poke the MSB of the desired program in location 25. This doesn't take much software because all that is needed is to keep track of is the vectors in 25-28. Of course you might want a directory to keep up with the program names. If you just use basic programs you might want to write your own multiprogram manager software.

64K ALL RAM

This month we want to show how to configure the computer for the all RAM mode. If you have a 64K computer then you can only use 32K at a time. The second 32K memory bank can be switched with the first bank which we have done in our previous editorials. The upper 32K is for read only memory (ROM) chips. The Basic, Extended Basic, and Disk Basic, ROMS occupy the upper 32K.

In the 64K RAM mode the ROMS have to be copied into the upper 32K of RAM. Then you have complete flexibility to modify the ROM software. You also have about 8K of additional RAM that can be used for data. Some programs configure the computer for the all RAM mode.

Machine Language Subroutine

We have developed a machine

language subroutine what we carry within a basic program. The program just requires 26 bytes and we put it at 10000. Following is an assembly listing of the program with comments.

```
10000 ORCCI   80 'MASK OUT THE
      INTERRUPT BITS OF THE
      CONDITIONAL CODE REGISTER
10002 LDX I  32768 'PUT THE
      VALUE 32768 INTO THE X
      REGISTER
10005 LDA X DIR R+0 'LOAD THE A
      REGISTER WITH THE VALUE
      IN MEMORY INDICATED BY X
10007 STA E  65503  'STORE A IN
      65503 TO SWITCH TO THE
      ALL RAM MODE
10010 STA X DIR R+ ' STORE A IN
      MEMORY INDICATED BY X AND
      INCREMENT X (X=X+1)
10012 STA E  65502 'SWITCH BACK
      TO RAM-ROM MODE
10015 CMPX I 65280 'IS X=65280?
10018 BNE 10005  ' GO TO 10005
      IF X IS NOT EQUAL TO
      65280
10020 STA E 65503 'SWITCH TO
      ALL RAM MODE
10023 ANDCI  175 'RESTORE THE
      INTERRUPT BITS TO THE
      CONDITIONAL CODE REGISTER
10025 RTS 'RETURN FROM SUB
```

We use decimal arithmethic with our assembly language listings so that readers who are not too experienced with assembly language programming can better understand what we are doing. If you have an assembler/ disassembler then you can use it after the Basic program is run.

64K RAM PROGRAM

The following program transfers all ROM data to RAM and configures the computer for the 64K all RAM mode. Because there were only 26 bytes required we used READ and DATA statements to handle the machine language program.

The machine language data is transferred to location 10000. The ML subroutine is called from Basic with EXEC 10000. You can switch back to the RAM-ROM mode by POKING 65502 any value. To switch back to the all RAM mode POKE 65503 any value.

64K RAM PROGRAM

(TRANSFERS ROM to RAM)

```
1' EXTENDED BASIC NOT REQUIRED
2' 64K COMPUTER IS REQUIRED
10 PRINT"64K RAM PROGRAM
20 COPYRITE (c) 1985
30 PRINT"dYNAMIC eLECTRONICS
   iNC.
40 PRINT"PROGRAM 7-1-85
50 PRINT"THIS TRANSFERS ROMS TO
   RAM
60 DATA 26, 80, 142, 128, 0,
   166, 132, 183
70 DATA 255, 223, 167, 128,
   183, 255, 222, 140, 255, 0,
   38, 241, 183, 255
80 DATA 223,28,175,57
90 FOR J=10000 TO 10025
100 READ A: POKE J,A: NEXT J
110 EXEC 10000
120 PRINT"COMPUTER IS NOW
    CONFIGURED FOR  64K RAM
```

WRITING PROGRAMS

(PART 5)

This is a series on how to write basic programs. We use the same techniques that we used in our computer classes. If you missed the first parts, back issues are available.

Last month we introduced some new concepts. Learning a computer language is similar to learning a foreign language. As new commands are covered they need to be used several times so that they become second nature. Then they can become a permanent part of our vocabulary. We have used the print command several times since we introduced it, and now it should be easy to write a print statement. Let's expound a little on the two con-

cepts covered last month.

FOR - NEXT Loops

These loops allow repetitive operations to occur. The first part or the FOR J=1 to N tells how many tims the sequence is to be repeated. There can be many statements between the FOR and the NEXT J statements. Each statement in between will be executed N times. This is a programming tool we will be using regulary because of its power.

Suppose you want to slow down the computer. Perhaps you have some instructions that you want to keep on the screen for a few seconds. A very powerful one line statement which is useful for slowing down the computer is as follows:

```
10 N=1000: FOR J= 1 to N: NEXT J
```

Statement 10 causes the computer to go through the FOR - NEXT loop 1000 times. This will slow down the computer so the instructions can be read from the screen. You can vary the delay by choosing different values for N.

Now suppose we want to print a line of stars across the screen. We can write the following program to do this task:

```
5 FOR J = 1 to 31:?"*";: NEXT J
```

Notice the ; at the end of the print statement. Remembe. that the ; means to print the character and leave the cursor in the next space.

There are many other applications for using FOR - NEXT loops and we will be showing more examples in the future.

More on Arrays

Last month we showed how to use arrays for the names of students and their grades. This is a convenient method for keeping information in order. Let's suppose we have an address file of 100 people. We would like to be able to call up a telephone number, a ZIP code, a City, or a name and have all the information about the person printed on the screen. The addresses will be numbered from 1 to 100. Then when we find the number for the address in question, we can print all of the information about the person by using the same number in the argument or parentheses. Let's take an example. Suppose that we assign the variables as follows:

```
N$(X) is the name
A$(X) is the street
S$(X) is the state
T(X) is the telephone number
Z(X) is the zip code.
```

Now it should be obvious that if X is determined then all of the information can be determined with a print command since each part of the address can be called from the array by giving its name and the value of X.

How is X determined? We have to let the computer search for a match on one of the parameters of the address. For example suppose we know the ZIP code. Then the computer can search all ZIP codes in the array until a match is found. Then all of the information for that address can be printed. The search routine will require a FOR - NEXT loop and the loop counter will be equated to the number of the address file (X) when the match is obtained. We will show how to do this next month.

READ & DATA STATEMENTS

In our example program last month we defined our variables within program statements. This involves a lot of typing and is not a desired method when a large amount of data is invol-

ved. information can be carried in DATA statements. The procedure for entering the data is to type in a statement number and then the characters "DATA". After this the variables can be entered. The READ statement tells the computer what label is given to the variables as they are read into the computer. An example of a data statement follows:

```
150 DATA Jim Smith, 326 Jones St. , 3555672
```

Each data element is separated by a comma. The first variable will be "Jim Smith" which is obviously a string variable. The second variable is "326 Jones St." and this is also a string variable. The third variable "3555672" can be a string or numerical variable.

Suppose this is data from a program and represents the 53rd elements in an address array. Then we will want the following definitions:

```
N$(53) = " Jim Smith"
A$(53) = "326 Jones St."
T(53) = 3555672
```

Notice all of the preceeding would have to be typed if the variables were defined within the program. See how much simpler it is to use the DATA statement for carrying the data. Now let's look at how the computer recognizes the data.

READING DATA

A READ statement is required to transfer the data from statements into an array or into a variable the computer can recognize. A read statement could be as follows:

```
100 X=53
110 READ N$(X), A$(X), T(K)
```

If we had 100 addresses we wanted to read then we could use a FOR - NEXT loop as follows:

```
200 FOR K = 1 to 100
210 READ N$(K), A$(K), T(K)
220 NEXT J
```

Statements 200 to 220 would be all that would be required to read the data into the arrays. Notice the programming power of the FOR - NEXT loop.

BRANCHING

To develop our address file program we need to be able to branch out of a loop when a match is found. This will give us the number "X" in the array so we can obtain the rest of the information. If we are looking for a name match, we will start with the first name in the array and compare the name in the array with the name we want to find. If there is not a match we want to continue with the second name. Thus we can have a FOR - NEXT loop and search and compare each name until a match is found or until we exit the loop. When the match is found we need to branch to a part of the program where we can display the results.

The tools for doing branching are IF - THEN commands. They work like this. IF (a condition is true) THEN (Do this part of the program). If the condition is not true, then the THEN part is ignored. For our comparison we will use a test similar to the following:

```
20 IF T=T(J) THEN GO TO 300
```

Assume T is a telephone number and T(J) is the telephone number of the Jth address. Then if these numbers are equal, then statement 20 tells us to go to statement 300 where we can print the results.

We will continue with out address file program next month. This month we want to give you a useful program without using

arrays. So we are including an alarm clock program.

ALARM CLOCK PROGRAM

The following program will display the time and sound an alarm when a preprogrammed time is reached. It does not require extended basic. The time is determined by X in statement 340. You can adjust the value of X for better accuracy. The program is menu oriened so you select the time elements to change from the beginning menu. Comments are included to explain what each section does.

```
10 CLS
20 PRINT"ALARM CLOCK PROGRAM
30 PRINT"COPYRIGHT (c) 1985
40 PRINT"DYNAMIC ELECTRONICS
   INC.
50 PRINT"EXTENDED BASIC NOT
   REQUIRED
60 PRINT"PGM 7-2-85
7^ 'SCREEN DELAY
80 FOR J=1 TO 2000:NEXT J
90 '
100 'PRINT TIME ELEMENTS
110 CLS:PRINT"1 HOURS ="H
120 PRINT"2 MINUTES ="M
130 PRINT"3 SECONDS="S
140 'CHECK FOR ALARM ON
150 IF A=0 THEN PRINT"4 ALARM
    DISABLED
160 IF A=1 THEN PRINT"4 ALARM
    HOURS="AH:PRINT"  ALARM
    MINUTES="AM
170 'DECIDE WHICH TIME TO
    CHANGE
180 P$=INKEY$:IF P$="" THEN 180
190 N=VAL(P$)
200 IF N=0 THEN PRINT"CLOCK IS
    STARTING": GO TO 340
210 'BRANCH TO MAKE CHANGES
220 ON N GOSUB 250,260,270,280
230 GO TO 110
240 'SET TIME ELEMENTS
250 INPUT"HOURS";H:RETURN
260 INPUT"MINUTES";M:RETURN
270 INPUT"SECONDS";S:RETURN
280 INPUT"ALARM HOURS";AH
290 A=1
300 INPUT"ALARM
    MINUTES";AM:RETURN
310 'BASIC TIME EQUATION
320 'ADJUST NUMBER TO TRIM
330 'THE SECONDS
340 X=675
350 FOR J=1 TO X: NEXT J
360 S=S+1
370 'IF WE HAVE 60 SECONDS THEN
380 'WE NEED TO INCREASE THE
390 'MINUTES
400 IF S=60 THEN 470
410 CLS:IF A=1 THEN PRINT"ALARM
    TIME="AH":"AM
420 IF A=0 THEN PRINT"ALARM
    OFF"
430 PRINT@232,H":"M":"S
440 GO TO 340
450 'INCREASE THE MINUTES AND
460 'CHECK FOR 60 MINUTES
470 B=0:S=0:M=M+1: IF M=60 THEN
    H=H+1:M=0
480 'CHECK FOR 12 HOURS
490 IF H=12 THEN H=0
500 IF A=0 THEN 430
510 'CHECK ALARM HOURS
520 IF AH=H THEN 540 ELSE 430
530 'CHECK ALARM MINUTES
540 IF AM<M THEN 430
550 'SOUND THE ALARM
560 FOR J=1 TO 100
570 SOUND RND (255), 1
580 NEXT J
590 'ADJUST THE SECONDS TO
600 'MAKE UP FOR ALARM TIME
610 S=S+8: GO TO 340
```

COMPUTER GRAPHICS
(PART 6)

Last month we showed how to use graphics for writing messages on the screen. Our demonstration program allowed either 16 or 32 characters per line. After selecting the size, as a key was pressed, the character was formed on the screen. There were a couple of features about the program that we did not like.

First of all it was necessary to wait until the character was drawn before entering the next character. Secondly a larger

size for display purposes say 8 characters a line should be included. The third thing was that there were not enough blank rows between the character rows.

This month we updated the program to eliminate these disadvantages. Also we added additional comments so that you can see what the program is doing. With this program you enter the complete string and do not have to wait on the computer.

To generate the characters we used the highest graphic mode for the smallest set. This is PMODE 4 which gives 256 x 192. For the two larger sets we used the PMODE 2 graphics mode which gives 128 x 192 elements.

Let's review how the bytes are displayed in the high resolution graphics modes. The bits are displayed across the screen with the most significant to the left. The characters occupy 5 elements and we left the first 3 elements blank. So we can have 32 characters in the PMODE 4 mode since 32 * 8 = 256. A character requires 7 rows. We carried the row information within a dimensioned string array called K$(N) and let N represent the ASCII value of the character.

GENERATING LARGE CHARACTERS

We decided to make the large characters twice as wide as the medium size characters. To do this we had to operate on the ROW data from the string array. This is what we did.

Let's use "A" as an example. IF we use a "*" to indicate the element on, an A will appear as follows:

```
76543210     VALUE

   *          04
  * *         10
 *   *        17
 *   *        17
 *****        31
 *   *        17
 *   *        17
```

To print double wide we will have:

```
765432107654321

      **
    **  **
  **      **
  **********
  **      **
  **      **
  **      **
```

Notice that each bit has to be printed twice. The numbers above the character elements represent the bit number of the byte. We wrote a subroutine at 1230 in the prorgram that strips off the bits and reformats the two bytes for a double wide character.

With the comments in the program and our discussion here you should be able to understand the generation of characters. If you have a dot matrix printer, then these techniques can be used for generating large characters to be printed on your printer.

CHARACTER GENERATOR

```
10 CLS
20 PRINT"THIS IS PGM 7-3-85
30 PRINT"COPYRIGHT (c) 1985
40 PRINT"DYNAMIC ELECTRONICS
   INC.
50 PRINT"EXTENDED BASIC IS
   REQUIRED
60 PRINT
70 'RESERVE ROOM FOR GRAPHICS
80 PCLEAR 8
90 INPUT"ENTER 1 FOR DISK
   DRIVE" ;D
100 'SET UP A CHARACTER ARRAY
110 DIM K$(100)
120 M=3072
130 IF D=1 THEN M=5120
140 'LET M=5120 FOR DISK DRIVE
150 INPUT"CHARACTER SIZE
```

```
    (1-3)";CS
160 G=CS:IF CS=1 THEN G=2 ELSE
    IF CS=2 THEN G=1
170 'SET UP GRAPHICS MODE
180 PP=2:IF CS=1 THEN PP=4
190 '
200 'CHARACTER DATA FOLLOWS
210 'THE 7 ROWS ARE DEFINED BY
220 'THE DECIMAL VALUES OF THE
230 'PAIRS OF CHARACTERS
240 'OTHER CHAR CAN BE ADDED
250 '
260 K$(42)="04211404142104
270 K$(43)="00040431040400
280 K$(44)="00000000040408
290 K$(45)="00000031000000
300 K$(46)="00000000000004
310 K$(48)="14171921251714
320 K$(49)="04120404040414
330 K$(50)="14170106081631
340 K$(51)="31010206011714
350 K$(52)="02061218310202
360 K$(53)="31163001011714
370 'ASCII 54="6"
380 K$(54)="07081630171714
390 K$(55)="31010204080808
400 K$(56)="14171714171714
410 K$(57)="14171715010228
420 K$(58)="00000400040000
430 K$(59)="00000400040408
440 K$(60)="02040816080402
450 K$(61)="00003100310000
460 K$(62)="08040201020408
470 K$(63)="14170204040004
480 K$(65)="04101717311717
490 '
500 'ASCII 65 IS FOR "A" THE
    BINARY REPRESENTATION FOR
    AN "A" FOLLOWS. WHERE THERE
    IS A "1" THE ELEMENT WILL
    BE PRINTED
550 '
560 ' 00100  (04)
570 ' 01010  (10)
580 ' 10001  (17)
590 ' 10001  (17)
600 ' 11111  (31)
610 ' 10001  (17)
620 ' 10001  (17)
630 '
640 K$(66)="30171730171730
650 K$(67)="14171616161714
660 K$(68)="30171717171730
670 K$(69)="31161630161631
680 K$(70)="31161630161616
690 K$(71)="15161616191715
700 K$(72)="17171731171717
710 'ASCII 73="I"
720 K$(73)="14040404040414
730 K$(74)="01010101011714
740 K$(75)="17a2024201817
750 K$(76)="16161616161631
760 K$(77)="17272121171717
770 K$(78)="17172521191717
780 K$(79)="14171717171714
790 K$(80)="30171730161616
800 K$(81)="30171717211813
810 ASCII 82="R"
820 K$(82)="30171730201817
830 K$(83)="14171614011714
840 K$(84)="31040404040404
850 K$(85)="17171717171714
860 K$(86)="17171717171004
870 K$(87)="17171721212717
880 'ASCII 88="X"
890 K$(88)="17171004101717
900 K$(89)="17171004040404
910 K$(90)="31010204081631
920 'H IS THE NUMBER OF
     CHARACTERS ON A LINE
930 X=0:H=16*G
940 IF G=3 THEN H=16
950 INPUT"ENTER THE MESSAGE";P$
960 L=LEN(P$): FOR W=1 TO L
970 A$=MID$(P$,W,1)
980 PMODE PP,2: SCREEN 1,1
990 'CONVERT THE STRING TO A
1000 'VALUE
1010 A=ASC (A$):N=M
1020 '
1030 'BREAK THE STRING K$ DOWN
      INTO THE SEVEN PARTS FOR
      THE CHARACTER ROW
      INFORMATION
1060 FOR J=1 TO 7
1070 Q=2 * J-1:X$=MID$ (K$ (A),
     Q, 2) :Y=VAL (X$): X=255-Y
1080 'DOUBLE WRITE FOR LARGE CH
1090 IF G=1 THEN POKE N+16,X:
     POKE N+32,X
1100 IF G=3 THEN GO SUB 1230:GO
     TO 1130
1110 IF G=3 THEN 1130
1120 POKE N,X:N=N+32
1130 NEXT J
1140 'LEAVE SPACE BELOW CH
1150 POKE N,255: POKE N+32,255
1160 IF G=3 THEN M=M+1:C=C+1
1170 M=M+1:C=C+1
1180'CHECK FOR END OF LINE
1190 IF C=H THEN GO SUB 1340
1200 NEXT W
1210 GO TO 1210
1220 GO TO 970
1230 FOR K=1 TO 5:AA=K-1:Z=2↑A
1240 X(K)=Z AND Y: IF X(K)>1
```

```
     THEN X(K)=1
1250 NEXT K
1260 B1=3 * X(5)
1270 B2=3 * X(1) + 12 * X(2) +
     48 * X(3) + 192 * X(4)
1275'255 MEANS ALL BRIGHT
1280 B1=255-B1
1290 B2=255-B2
1300 POKE N,B1: POKE N+1,B2
1310 POKE N+16,B1:POKE N+17,B2
1320 N=N+32
1330 RETURN
1340 M=N+1: C=0: IF G=3 THEN
     M=M+1
1345 'LEAVE SPACE BETWEEN ROWS
1350 FOR F=1 TO 16 * H
1360 POKE M,255: M=M+1:NEXT F
1370 RETURN
```

COCO HEAT PROBLEM

by

Dennis Kistz

Mr. Bill Chapple
Dynamic Color News
P.O. Box 896
Hartselle, AL 35640

Dear Bill,

I was just going out the door for a week in another **wil**derness (Montana) when I skimmed **th**rough your recent issue. Your "Editor's Comments" caught my eye. Perhaps, if you wouldn't mind, I could amplify a little bit on your notes and give you a gentle criticism.

Your overview of regulation is correct, but you might note that the regulation is performed differently on various Color Computers. The difference can be significant.

On the original machine (revisions C through F), regulation is performed by an adjustable standard circuit. The actual current flows through a very ·ugged transistor with proper heat ;inking. It should also be noted that :his machine also offers a 12-volt ;upply (using a monolithic, or)ne-chip, regulator) and -5 and -12 'olt supplies of low-current :apability. Disturbing this mix can)e chancy.

)n the Color Computer 2 (USA-made ·evisions through B), regulation is lone by a custom supply-and-level :ranslator (SALT), but still with a ·ugged transistor carrying the current :low. Low-current positive and ıegative voltages for the RS-232 port ıre provided by the SALT for internal ıse.

'inally, in the Korean Color Computer ! (revisions through B), a monolithic one-chip) 5-volt regulator handles :he brute current, with the SALT again)roviding internal positive and ıegative voltages.

ın input of 9 or 10 or 12 or even 15 'olts (representing AC supply 'ariances of up to 50%) is trivial.

On all these machines, the regulators and transistors in question are designed to run with up to 30 volts input and up to a temperature of 85 degrees C (185 degrees F)! In other words, if you can touch it, it's probably running plenty cool on its own terms.

With that background, consider the actual ramifications of heat buildup. First, the performance of many parts, especially capacitors, can be degraded over the long term by heat buildup. Secondly, excessive heat on the regulators or pass transistors can cause their breakdown in the power supply. Third, heat buildup around hot parts (regulators and pass transistors) can oxidize solder joints and cause intermittent failures in the power supply itself. Finally, if you're foolish enough to do so, disks and tapes placed on a hot computer can be damaged.

But I believe the heat question is not significant. With the exception of the defective Synchronous Address Multiplexers (SAMs) in the original Color Computers (which ran excessively ho') I have rarely seen a machine sustaining heat damage. Overall degradation is insignificant, and there is not excessive heat on the regulators and transistors themselves. And the only damage I have seen is oxidized solder joints (in one rare case in a school where the machine had been covered with papers and books). Aside from believing in the motto, "if it ain't broke, don't fix it", there are other reasons for avoiding a "solution" to a heat "problem".

The main solutions proposed are more heat sinks or fans. Can better heat-sinking and cooling fans help? Yes and no. They work differently. Cooling fans, in fact, can be far safer than more heat-sinking to the parts, but they introduce other problems.

Consider this: It is bad practice to OVER-sink a part. It causes what is called "thermal stress". When a die (the "chip" part of a chip) heats up, as it does in transistors and regulators, its heat is carried away through its substrate to the package, which can be a metal can or tab. That can or tab is fastened to a metal heat sink with conductive paste or pads. The heat is drawn away and dissipated by air circulation.

By over-sinking, the heat is drawn away very quickly, especially on power-up, causing the microscopically fine binding wires running from the die to the pin connections to stress and, eventually, to break. The part mysteriously fails!

The heat sinks on the Color Computer are pretty close to ideal design size, and I feel they should not be altered, added to, or moved elsewhere. (For more data on this, refer to National Semiconductor's "Voltage Regulator Handbook")

What about cooling fans? I said they are safer because when they draw heat away, they do so through air circulation around the heat sink, not by direct connection which might cause thermal stress.

But cooling fans are disadvantageous because when they are turned on and off, a magnetic pulse (caused by the field rise and fall in the coils) is produced, together with a power supply "spike" which can glitch memory contents or CPU activities. They should always go on and off with the computer, which is not always the way these retrofits are sold.

Really the safest method of cooling the computer is doing nothing: don't plug it up with cloth underneath, don't pile papers and books on it, don't overload it with internal additions.

But, if you decide you need to dissipate some heat (especially with lots of internal modifications), simply drill holes over the hottest

area. It avoids thermal stress on the power supply components, and avoids spikes and pulses caused by a fan.

I hope this discussion is helpful to your readers.

Sincerely,

Dennis Bathory Kitsz

President, Green Mountain Micro
Publisher and Technical Editor, Under Color

Thank you Mr. Kitsz for your fine article on the Color Computer's voltage regulators. We agree with your editorial except we would like to add a couple of comments. When additional accessories are added inside the computer more heat is generated and the air flow can be restricted. This additional heat causes stability problems with some of the integrated circuits. Our solution is to mount an external 5 volt regulator outside the computer.

```
+ ++ + ++ + ++ ++ ++ + ++ + ++ +
+                               +
+         RENEWAL TIME?         +
+                               +
+ The date beside your name on +
+ the address label indcates   +
+ the last issue you will re-  +
+ ceive. Send in your renewal  +
+ if you want to continue re-  +
+ ceiving technical informa-   +
+ tion on Color Computers.     +
+ This is the last issue for   +
+ those with 7/85.             +
+                               +
+ ++ + ++ + ++ ++ ++ + ++ + ++ +
```

PRODUCT REVIEWS

This section is open to all producers and dealers of color computer products. We will review you product free of charge and write an editorial on the product. We do not use a rating system but will explain what the product does, and what can be expected from it. Any comments about the review from the firm submitting the product will be printed in a later issue.

* * * * * * * * * * * * * * * * * *

SPECTRUM DOS 1.0

SPECTRUM DOS 1.0 is a disk operating system. An operating system is a program that simplifies commands, adds additional features, and makes the operation of the computer easier. Some common operating system are "CP/M","OS-9","FLEX", & "UNIX". Each of these are designed to provide advantages over the standard software that comes with the computer.

DISK DRIVES

In this review we will explain what SPECTRUM DOS 1.0 does and what you can expect from it. First let's consider its disk operating features. The number of tracks for each drive can be specified as well as the default seek rate. This mean that you can mix drives from different manufacturers and select the number of tracks and rates for each drive. SPECTRUM DOS 1.0 can also handle double sided drives.

For printing the disk directory the "DIR" command causes the directory to be printed in two columns plus the number of free granules. If you want the directory to be printed to the screen and printer then use "HDIR".

GENERAL OPERATION FEATURES

The advantages of SPECTRUM

DOS 1.0 really become evident when writing Basic Programs and for normal disk operations. There are 9 functions that can be programmed and recalled by pressing 2 keys. For example commands like "MOTOR ON", "SOUND 100,10" , "LIST", ETC. can be programmed for 2 key operation. A screen dump to the printer can be made by pressing the SHIFT - RIGHT ARROW key. If you have a Basic program that you load and run frequently then name the program "BOOT" when it is saved to a disk. Then it will be loaded and run with the SHIFT - RIGHT ARROW key. A machine language program can be loaded and run by using the "RUNM" command.

HI-RESOLUTION SCREENS

There are 3 Hi-Resolution screens that can display 32,51, or 64 characters with true upper and lower case. The characters can be reversed by the "INVERT" command. For listing Basic programs much more of the program can be displayed using the 51 or 64 character displays.

WRITING BASIC PRGRAMS

There are several commands that are very helpful for writing basic programs. The "AUTO" command allows the automatic generation of line numbers with any increment. For example AUTO 100,10" will start line numbers starting at 100 with increments of 10.

A key will repeat as long as it is held down. This is very useful when a statement is being edited as you can quickly move to any part of the statement with the space bar and left arrow.

A basic line can be copied from one statement to another. For example suppose you want the same information in statement 2050 that is in statement 10. Just type "LCOPY 10,2050" and the information in statement 10 will also be in statement 2050.

A basic line can also be moved to a different location. "LMOVE 10,2050" will move statement 10 to 2050 and delete statement 10.

To slow down the computer we generally write statements as follows:

```
200 FOR J=1 TO X: NEXT J
```

SPECTRUM DOS 1.0 has a command that simplifies this. It is:

```
200 WAIT X
```

X is the amount of delay and is about 1 second if X=1.

For storing 16 bit values or vectors in memory we have to convert to two values consisting of a most significant and a least significant. The formulas are:

```
100 MS=INT (V/256): LS = V -
    256 * MS: POKE M,MS: POKE
    M+1, LS
```

An equivalent using SPECTRUM DOS 1.0 is

```
100 PPOKE M,V
```

For looking at 16 bit values in memory the PPEEK M,V will display the decimal value of the 2 bytes.

Auto key repeat is another feature. This allows quickly editing Basic programs by positioning the cursor with the space bar and left arrow keys.

Lower case commands are accepted. You can type "run". This is handy if you are in the lower case mode and enter a command. With normal DISK Basic you will get an error and have to enter the command again.

Have you ever typed "NEW" to start a new program and wished you had the old one back? Now you can by typing "OLD" and the old program is restored.

If you want information to be

printed on the screen and printer then you can use the "ECHO" command.

Also included is an error trapping routine which transfers control to a designated statement number if an error occurs.

Spectrum DOS 1.0 is advertised to support "FLEX" operating systems. We do not have a "FLEX" system and could not verify this.

In case you forget what commands are available just type "HELP" and a menu will display the commands.

SUMMARY

As with any program there are trade offs. Spectrum DOS 1.0 configures the computer for 64K RAM, transfers the ROMS to RAM, and modifies the disk operating system. Programs that require 64K of RAM will not work with Spectrum DOS 1.0. Instructions are included for burning in a replacement EPROM for the disk ROM with Spectrum DOS 1.0. We did not do this but this would eliminate the need to configure the computer for 64K RAM.

The program accepts commands in lower case or upper case. It appears that it converts all lower case to upper case. This will be a disadvantage if you have character strings in lower case that you do not want changed to capitals.

As with any program it should be used when it best meets your requirements. For normal disk operation and Basic program development, Spectrum DOS 1.0 is a tremendous asset. Its many features are time savers and it is a joy to use. The auto line number, line move, line copy, auto repeat, high resolution screens, plus the many other features make writing programs much quicker and easier using Spectrum DOS 1.0.

Spectrum Projects, P. O. Box 21272, 93-15 86th Dr., Woodhaven, NY 11421 $24.95 disk.

-DCN STAFF-

****
* *
* BACK ISSUES *
* *
* Back issues of DCN are *
* available for $1.95 each *
* or 3 for $5 postpaid. *
* *
* Foreigners other than Can- *
* ada add $2 for Air Mail *
* postage. *
* *
****

CLASSIFIED ADS

1. 10 cents a word, $3 minimum.
2. Name, Address, & Telephone listed free.
3. Send payment with ad.
4. Closing date 1st of the preceeding month. Ex. Nov ad closing is Oct. 1.

X-RATED Adventure Game! A game so real that you will know that your COCO has finally grown up. 16/32K $15 tape/ $20 disk; Y. Kraft. 164 N. Blackstone Ave., Fresno, CA 93701

* * * * * * * * * * * * * * * * *

DISPLAY ADS

Closing 1st of preceeding month.

Pages	1 time	2 times	3 times
1	$25	$22	$20
1/2	18	15	13
1/4	10	8	7

NEW PRODUCTS

This section is available free for producers and dealers of color computer products. These products have not been reviewed by us but are included for our reader's information. Send a description of new products to:

New Products
Dynamic Electronics Inc.
P. O. Box 896
Hartselle, AL 35640

Maxsys Electronics and Software Company has announced the release of its new product COLORBURST. COLORBURST is a memory expansion unit that when plugged into Tandy Company's Color Computer will expand the RAM memory up to ONE MEGABYTE. This is continuous RAM memory without bank switching. COLOR-BURST has the capability of processing 1/2 megabyte of ROM memory. Other features include:

- Parallel interface (bidirectional) for a parallel printer and other parallel devices.

- COLORBURST greatly speeds up the Color Computer's throughput.

- Seven software selectable card slots for ROM cards/ disk controller; six along the top and one out the back for a disk controller. For more information contact

Electronics & Software Co.
P. O. Box 25336
Milwaukee, WI 53225

****
* Please sign me up for one year for the DYNAMIC COLOR NEWS SERVICE. I *
* Want to receive instruction on programming, Computer Theory, Operat- *
* ing Techniques, Computer Expansion plus the Individual Reply to my *
* Computer problems for a special of $10 each. Also I understand that *
* therm will be no charge for letters printed with answers in the *
* Newsletter. Cost $15 USA & Canada, $30 foreign. *
* *
* Name _____ Mail payment to *
* Address _____ Dynamic Electronics Inc *
* City _____ P. O. Box 896 *
* State & Zip _____ Hartselle, AL 35640 *
* Enclosed is a check ___ *
* charge to VISA ___ MC ___ Number _____Exp._____ *
* *
****

DYNAMIC ELECTRONICS INC.
P. O. Box 896 (205) 773-2758
Hartselle, AL 35640

BULKRATE
U.S. POSTAGE
PAID
HARTSELLE, AL
35640
PERMIT NO. 21