ENGINEERING NOTES
on
Radio Shack Color Computers

June / July 1984 $1.95
Vol. 1 No. 5

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


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


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

* ASCII PART III
* INTERRUPTS
* POWERFUL REMARKS

* COMPUTER THEORY
* BASIC PROGRAMMING
* OPERATING HINTS
* MACHINE LANGUAGE PROGRAMMING
* QUESTIONS & ANSWERS

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 newsletter are encouraged. All submissions become the property of Dynamic Electronics 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.

All paid subscribers are entitled to discounts of 10% on hardware. and 20% on software manufactured or produced by Dynamic Electronics Inc. plus "specials" mentioned in the newsletter. To receive these discounts use your DCN number which is at the right of your name on the address label. DCN subscribers may obtain a personalized reply to specific computer problems or advice on purchasing equipment. The charge for this service is $10.

```
********************************
*                              *
*   DYNAMIC   COLOR   NEWS     *
*                              *
*      June / July  1984       *
*                              *
*    Editor and Publisher      *
*        Bill Chapple          *
*                              *
*         Secretary            *
*       Belinda Parker         *
*                              *
********************************
```

CONTENTS

EDITOR'S COMMENTS

Sometimes it is hard to decide on what would be of interest to our readers. Since our stated objective is to provide technical information on color computers, it would seem like there should not be a problem. I have attended a lot of classes in my lifetime and one of the worst feelings that I remember was to be in a class and not understand anything that was going on. Some of you can identify with this. That is not quite the feeling I have because this is the fifth newsletter we have written and I am sure that those who have been with us from the start are versed in computer terminology a little anyway. The problem is what subject should we present next. With this issue we are finishing our presentation on serial communications or ASCII so new material should be considered.

The following incident occurred last month. My nephew, who has a degree from Louisianna State University (LSU), got married in May. He is a reporter for a newspaper in Texas and is a real poet when it comes to writing. My son, Mike, went with me to Baton Rouge, LA for the wedding. If you will look at your road map, you will see that the trip is about 500 miles from Hartselle, AL. Since so much time would be involved in travelling, why not take a computer along and operate it from the car. Surely I could do some writing while Mike was driving. Well how do you get the computer to operate in the car? There are devices called power inverters that will convert car voltage to 115 volts so the computer could be plugged into the inverter along with a television.

I have a microcassette recorder and a set of tape files with it. The recorder works on two AA type batteries so I did not need any power source for it. The inverter was installed and the computer and television turned on for an initial test. Everything worked fine so it looked like I would be able to use the color computer on the trip. The results were very gratifying. I was able to write the first draft on "POWERFUL REMARKS" while Mike was driving. I could have been able to write more except that Mike went out with one of his cousins on Saturday night and I had to drive most of the way back while Mike slept.

Since I haven't read much about using battery power for operating computers, I thought that perhaps this would be a subject of interest. So an editorial appears in this issue that tells you some options available for running your computer on battery power. In an earlier issue I mentioned that a battery backup was being developed for mounting under the keyboard. We are about to release this as a new product and are offering it as a "SPECIAL" to DCN subscribers in this issue. So if you are tired of power failures causing you to loose your programs, then you will want to read that editorial.

Also this month we are starting a series on "POWERFUL REMARKS". Basically what we will be doing is showing how remark statements can be used as a substitute for "DATA" statements. Esentially we can say go to statement number "X" and get the data from it. Statement X will be a remark statement. Basic does not have an equivalent to this and we have used this technique for our invoice program, our accounting program, writing instructions, and writing chain letters. Also we put machine language subroutines in remarks and carry them with our basic programs. So if you want to learn an easy way to handle data then you should read this section.

A customer sent us a computer to upgrade to 64K. It was an old grey type with only 4K of memory. That was what I had to start with and paid $399.95 for it. It was only 3 years ago that I bought my first color computer and I'll have to admit that the 4K memory in the computer refreshed my memory. It took me about a week to realize more memory was needed. I upgraded to 16K, shortly afterwards to 32K and then to 64K. Look at the MC-10 which sold for about $59.95 during Christmas. It is 4K and a 16K module can be purchased for $49.95. This is really a buy for someone wanting to get into computers economically. What is the future for computers? My guess is smaller disk drives, lower priced hard disks, and inexpensive tape drives with operating systems similar to disk drives. But this is just a guess. Of course memory chips are getting larger. I saw an advertisement for 256K chips for $50 each. That's pretty expensive but the prices are slowly dropping. I predict that the next generation of computers will have 256K of random access memory. How does this effect the color computer owners. Not very much because color computers are very expandable. The old 4K D board can be easily upgraded to 128K or larger and is suitable for any other kind of modifications that new technology may introduce.

POWERFUL REMARKS

(Part 1)

WORD PROCESSING

The casual programmer might wonder "what is so great about remarks"? Anybody knows that remarks are used to write comments about the program and basic ignores them. The writing of comments in remark statements is straightforward and simple. In fact it is similar to the operation of word processors because you just type in the information. Now if the statement numbers and the Remark sysmbol "'" could be removed then we would have a word processor with each line being the informtion in the remark statement. Remark statements can also be used to carry data and machine language subroutines. The purpose of these editorials will be to show how to use remark statements for word processing, data, and machine language routines.

Let's look at data statements first. If you will recall a couple of months ago we covered "READ" and "DATA" statements. Remember that the data has to be perfectly ordered and to obtain data from the middle it is necessary to read all data starting at the first data variable up to and including the one you want. Also remember if one data element was put in the wrong order then all the data after it was wrong. These of course were the disadvantages. The advantage is it is easy to use for a small number of variables or when variables are easily ordered.

Now let's look at machine language subroutines. First of all a subroutine is a program that can be called from another program. After the subroutine is run then program control is transferred back to its previous function or the first program. To run a machine language subroutine from basic you just enter EXEC M where M is the starting location of the machine language subroutine. For basic to call a subroutine just enter "GO SUB X" where X is the statement number of the start of the subroutine. A subroutine is written like any other program except the command "RTS" is at the end of a machine language program and "RETURN" is placed at the end of a basic program. Well why would we want to use machine language subroutines with basic? We want basic to run as fast as posible and machine language subroutines are great for sorting data, finding the memory location for a statement number, and terminal programs to give a few examples.

What about remark statements for word processing? Have you noticed how easy it is to write in remark statements. Remember for handling data we could not use a comma because a comma is used to separated different data elements. However when we write in remark statements there are no restrictions. Also Basic handles the information like any basic statement. This means that statements can be eliminated by typing the statement number and pressing the <ENTER> key. Also extended basic's statement editor works well for correcting information in a statement. Everyone should agree that it is easy to write a letter and put a statement number and remark symbol "'" at the beginning of each setence. Now we can use basic to insert new lines or delete lines. If we have extended basic then we can edit any line. The only problem is the removal of the statement numbers and the remark symbol.

Now let's look at a remark statement. Remember a 0 preceedes any basic statement. The next 2 locations contain a vector pointing to the next statement number, the next 2 locations contain the statement number in vector format, and the next 2 contain the ASCII code for remark and the basic token for remark. So if you will count the bytes required to get to the first bit of information in a remark statement, you will get 6.

The format for a basic program with remarks is as follows.

byte 0 - A zero preceedes each statement
byte 1 - MSB of vector for the next statement
byte 2 - LSB of vector for the next statement
byte 3 - MSB of statement number.
byte 4 - LSB of statement number
byte 5 - ASCII code for '
byte 6 - Basic code number for remark
byte 7 - First data byte

Now if we can find the memory where the statement begins (byte 1), then we add 6 to that value and obtain the memory where data starts. That doesn't seem too difficult does it?

Let's write a basic program to find the memory location for any basic statement number.

STATEMENT NUMBER SUBROUTINE

```
60000 'THIS FINDS THE MEMORY FOR STATEMENT NUMBERS
60010 INPUT "ENTER THE STATEMENT NUMBER"; SN
60015 'FIND THE MEMORY LOCATION FOR THE FIRST STATEMENT BM
60020 BM=256 * PEEK (25) + PEEK (26)
60025 'LOCATIONS 25 & 26  CONTAIN THE BEGINNING OF BASIC PROGRAM VECTOR.
60030 'FIND THE FIRST STATEMENT NUMBER AND THE
60035 'LOCATION OF THE NEXT STATEMENT
60040 NS = 256 * PEEK (BM) + PEEK (BM +1)
60050 TS = 256 * PEEK (BM + 2) + PEEK(BM + 3)
60055 PRINT"THIS IS 60055 & NS="NS;"  TS="TS
60060 'NS = NEXT STATEMENT MEMORY LOCATION & TS = STATEMENT NUMBER
60070 'DID WE FIND THE STATEMENT WE WANTED?
60080 IF SN =>TS THEN 60200
60090 'GO TO 60200 IF THE STATEMENT NUMBERS AGREE
60095 BM = NS 'GO TO THE LOCATION OF THE NEXT
60100 'STATEMENT AND LET ITS MEMORY = BM
60110 GO TO 60040
60190 '
```

```
60200 'WE HAVE FINISHED SEARCHING AND THE MEMORY FOR OUR
60210 'STATEMENT NUMBER IS BM.  DATA STARTS AT BM + 6
60220 X = BM + 6
60230 PRINT"THE MEMORY LOCATION FOR STATEMENT NUMBER "SN; "IS "X
60240 RETURN
60250 'WE USE RETURN TO END THIS PROGRAM SO IT CAN BE USED
60260 ' AS A SUBROUTINE.
```

After finding the memory for the remark statements and the beginning of data it is fairly easy to process it. The following subroutine will process the data and print the characters on the screen and /or to an external printer. The comments will show what each section does.

```
60500 'THIS SUBROUTINE WRITES CHARACTERS TO THE SCREEN OR TO A PRINTER
60510 INPUT "BEGINNING STATEMENT NUMBER";X1
60520 INPUT "ENDING STATEMENT NUMBER";X2
60530 'THIS WILL PRINT THE INFORMATION CONTAINED IN
60540 'ALL STATEMENTS BETWEEN AND INCLUDING THE
60550 'BEGINNING AND ENDING STATEMENT NUMBERS.
60560 SN=X1: GO SUB 60020
60570 'FIND THE START OF DATA. X IS THE MEMORY
60580 'WHERE DATA STARTS FROM THE SUBROUTINE.
60590 A=PEEK(X): A$=CHR$(A)
60600 'LET'S CHECK TO SEE IF A=0 INDICATING END OF STATEMENT
60610 IF A=0 THEN 60650
60620 PRINTA$;: IF P=1 THEN PRINT#-2,A$;
60630 'IF P=1 THEN THE PRINTER IS ON.  WE WILL
60635 'DETERMINE THIS FROM OUR MAIN PROGRAM
60640 'GO TO THE NEXT MEMORY LOCATION (X+1)
60645 X=X+1: GO TO 60590
60650 'X IS A ZERO. LET'S CHECK STATEMENT NUMBERS
60660 'TO SEE IF WE HAVE FINISHED.
60665 PRINT: IF P=1 THEN PRINT#-2," "
60670 M=PEEK(X+3):N=PEEK(X+4): AA=256*M +N
60680 'AA IS THE NEXT STATEMENT
60690 X=X+7 'THIS IS THE DATA IN THE NEXT STATEMENT
60700 IF AA> X2 THEN RETURN
60710 'REMEMBER X2 WAS THE ENDING STATEMENT NUMBER
60720 GO TO 60590
60730 'THE ABOVE STATEMENTS TELL THE PROGRAM TO GO
60740 'TO 60590 IF THE NEXT STATEMENT NUMBER AA IS
60750 'NOT GREATER THAN OUR ENDING STATEMENT NUMBER X2
60760 'ALL INFORMATION AFTER ' IN ANY STATEMENT CAN
60770 'BE ELIMINATED AND THE PROGRAM WILL STILL RUN
60780 'BASIC IGNORES THIS AND GOES TO THE NEXT
60790 'STATEMENT.
```

USING SUBROUTINES

When writing Basic or Machine Language programs it is advisable to use subroutines. A subroutine is a program that can be called from another program. It ends with a "RETURN" command in Basic and a "RTS" or return from subroutine command in machine language subroutines. If you have a task you need done then write the task in the form of a subroutine and call it from the main program when it is needed.

Now if we are going to print the information contained between two remark statements then we will need the two subroutines we just wrote. The 2 tasks we need to do are:

1. Find the memory location for the statement number. This is what the subroutine at 60000 does.

2. Print the information contained within the remark statements until a 0 is encountered. Then check the next statement number to see if it is greater than the ending statement number. Return if we have finished. This task is done by the subroutine at 60500.

WORD PROCESSING PROGRAM

Now that we have developed the subroutines for recovering information from remark statements, the writing of the program is fairly easy. For using the subroutines we use the basic commands "GO SUB X" where X is the statement number of the first statement we want to use in the subroutine. For example with the subroutine at 60000 if we "GO SUB 60000" then the program asks us to enter a statement number in 60010. Suppose we want to designate the statement number before going to the subroutine. For example we define the statement number before we go to the subroutine. Then we can enter "GO SUB 60020" instead. All of the subroutine will be executed until a "RETURN" command is encountered. Then the program will return to the next command in the first part of the program after the "GO SUB 60020" command.

The following program prints all information between two remark statement numbers:

```
50000 'THIS PROGRAM PRINTS ALL CHARACTERS BETWEEN 2 REMARK STATEMENTS
50010 INPUT"ENTER A 1 FOR PRINTER";P
50015 GO SUB 60500
50020 'THIS IS ALL WE NEED TO ENTER AS THE SUBROUTINE
50030 'AT 60500 WILL DO THE REST
50090 END
```

Notice the power of Basic. The program consists of only 2 statements 50010 and 50015. These statements set up for printer use and do the "housekeeping". Housekeeping means to keep track on what is going on. By using subroutines the actual task of keeping things in order is greatly simplified. In case you didn't follow everything that was done, all of the program sections need to be entered. When you enter run, the first statement is 50000 which we used for the master program with subroutines in higher statement numbers. Text can be put anywhere in memory below 50000. The first statement in your program should be "GO TO 50000" if you have statements that are not remarks.

Also notice how easy it would be to write a chain letter to multiple addressees. You can use "FOR NEXT" loops and reserve statement numbers for the names and addresses of the people receiving the letter. Also a different group of statement numbers can be used for the body of the letter. There are many ways the program can be organized and basic is very powerful for controlling this.

OTHER USES FOR REMARKS

We use remark files for our bookkeeping, invoices, writing chain letters, and sometimes for printing instructions. We have a master program that operates on a basic remark file program. Instead of using basic to find the memory of the statement numbers, we use machine language subroutines. This speeds up the process considerably since sometimes it takes basic a while to find the memory for the first statement number desired especially if it is preceeded by a lot of statements. Notice that we can obtain the information contained in just one statement number. This is quite an improvement over DATA and READ statements since we don't have to have knowlege of any ordering of information. We will continue with this subject next month.

The following is the complete program with data in statement numbers 20 to 50. We eliminated all comments. You can put anything that you want printed in remark statement numbers before 50000 and can edit your writing by using the basic and/or extended basic commands. For example if you want to change what is in a statement number then retype the statement or use extended basic's editor.

```
1 '
20 'My name is Bill and I live in
30 'Hartselle, AL.
40 'Where do you live?
50 'This is a test program to see what can be done.
50000 '
50010 INPUT"ENTER A 1 FOR PRINTER";P
50015 GO SUB 60500
50090 END
60000 '
60010 INPUT "ENTER THE STATEMENT NUMBER"; SN
60020 BM=256 * PEEK (25) + PEEK (26)
60040 NS = 256 * PEEK (BM) + PEEK (BM +1)
60050 TS = 256 * PEEK (BM + 2) + PEEK(BM + 3)
60055 PRINT"THIS IS 60055 & NS="NS;"  TS="TS
60080 IF SN =>TS THEN 60200
60095 BM = NS
60110 GO TO 60040
60190 '
60200 '
60220 X = BM + 6
60230 PRINT"THE MEMORY LOCATION FOR STATEMENT NUMBER "SN; "IS "X
60240 RETURN
60500 '
60510 INPUT "BEGINNING STATEMENT NUMBER";X1
60520 INPUT "ENDING STATEMENT NUMBER";X2
60560 SN=X1: GO SUB 60020
60590 A=PEEK(X): A$=CHR$(A)
60600 '
60610 IF A=0 THEN 60650
60620 PRINTA$;: IF P=1 THEN PRINT#-2,A$;
60645 X=X+1: GO TO 60590
60650 '
60665 PRINT: IF P=1 THEN PRINT#-2," "
60670 M=PEEK(X+3):N=PEEK(X+4): AA=256*M +N
60690 X=X+7 'THIS IS THE DATA IN THE NEXT STATEMENT
60700 IF AA> X2 THEN RETURN
60720 GO TO 60590
```

ASCII PART 3

This is the last of our discussions on ASCII in this series. In the preceeding discussions we showed that ASCII is used for serial data communications where characters are sent out one bit at a time. This is similar to the familar Morse Code where dits and dahs are sent out and combined to form characters. Early teletypes used serial communications where the character was composed of 5 bits plus start and stop bits. They used a code called "Baudot". With 5 bits there are only 32 possibilities. A code was sent to shift from "figures to letters" and the machine remembered that this had been sent. This allowed about 60 different characters to be available for sending. It is much nicer now with the ASCII code since an 8 bit byte can be sent or all 128 ASCII characters.

INTERFACING ASCII DEVICES

Last month we showed how ASCII is used with Basic for storing numbers representing characters in memory. This month we want to explain what is involved when ASCII devices are connected together. How do you connect a terminal, printer, or modem to a color computer? How do we know that the external device is ready to receive information? There is a signal in most ASCII devices which is called "handshaking". This means that the device sends the handshaking signal when it is ready to receive a character. Let's assume that our device is a serial printer. If the printer is not ready to receive characters from the computer, then the computer is put in a "hold state" until the printer sends the proper handshaking signal. If you have a printer and do not have it "ON LINE" then the computer is put in a wait state until the printer sends the ready signal. If you have a printer then you have probably observed this.

CONNECTING TWO COLOR COMPUTERS TOGETHER

If you have two color computers then they can be connected together and programs or information can be passed from one to the other. A terminal program is required for each computer. A cable needs to be made up to plug into the printer jack on each computer. The cable can be made with flat cable like the type that goes from the disk drive controller to the disk drive. On each end of the cable, 4 pin DIN plugs are required. The wiring of the plugs is as follows:

Color Computer 1				Color Computer 2		
pin #		Color wire	goes to	pin #		Color Wire
1	-	yellow	(not used on either)			
2	-	green		3	-	red
3	-	red		2	-	green
4	-	white		4	-	white

How are programs transferred? Let's assume that a terminal program is loaded into each computer in an area not used for programs. There are several terminal programs available and we will use our "DYTERM" as an example since we are very familiar with it. Now let's load the program to be transferred into the first computer. After the program is loaded either from a cassette or disk, record its beginning and ending basic vectors in locations 25 to 28. Then switch to the terminal program. If you don't know how to stack programs see the multiprogram manager in Volume 1, No. 1. You can pick a memory location that is not being used to load in the second program. To initialize basic for a new memory location do the following.

1. If M is the location for the new program poke a zero into M-1.
2. Let the vector in locations 25 and 26 point to M and type "NEW".

The second computer should be initialized for the same beginning memory location as the program to be transferred. This means that the values in 25 through 28 for the second computer should be the same values as for the first computer. Poke the zero into M-1. Then the terminal program can transfer the loaded program a byte at a time until all of the program is transferred. After the transfer is completed the second computer can run the program.

Before attempting to transfer data between two computers send a few characters to make sure that everything is working properly. You need to initialize the ASCII parameters. That is you need to select the baud rate, word length, parity, and number of stop bits. Use the fastest transfer rate you can so the transfer will be completed quickly. For transferring programs or data you need to send 8 bit ASCII characters.

RS-232

The RS-232 port is generally used for ASCII signals on terminals, printers, MODEMS, and computers. The definitions of the pins is a standard but the way they are used is confusing. Some of the pin connections for a RS-232 port is as follows:

pin #	Function
1	chassis ground
2	TX (transmit data)
3	RX (receive data)
4	RTS (request to send)
5	CTS (clear to send)
6	DSR (dataset ready)
7	SG (signal ground)
8	DCD (carrier detect)
20	DTR (data term. ready)
21	RI (ring indicator)

The definition of the functions of the pins is standard but the use of handshaking signals is not. So when interfacing two RS-232 ports it is important to connect the proper handshaking lines. Examples are usually given in the instructions of the equipment.

This will conclude our discussion of ASCII or serial data transmission. Remember the ASCII code is a standard which assigns a value to each character of the alphabet. Also when bytes are transmitted by ASCII, both the sender and receiver have to be set for the same baud rate. If you need to know more about ASCII then with the information we have given you should be able to read other material and understand it.

UNINTERRUPTED POWER SOURCES

You have been working hard developing that special program. It is almost finished and suddenly the power goes off. All of that work is lost. If a program was being saved to a disk, what happens to the disk? Is there a solution to this dilemma? The use of backup power sources is not well known to most microcomputer owners and users. There are several ways that backup power can be used and we want to show some of the options.

POWER INVERTERS

First let's briefly discuss the power requirements for computers. Most require 115 volts alternating current (ac) at a frequency of 60 hertz. This is what is available in the outlets of most homes and businesses in the United States. The voltage is constantly changing direction and it can be represented by a sine wave equation. This was designed for maximum efficiency back at the turn of the century for transmitting power from generating stations to consumers. The advantage of this type power is that it can be converted into different voltages by transformers. Within each color computer on the left side is a power transformer that converts the 115 volt line voltage into levels needed by the computer. A transformer also provides isolation from the power line making it safer for the operator.

The most common source for backup power is the 12 volt direct current (dc) storage battery as used in automobiles. Now if this could be converted to 115 volts ac then the computer could be powered from the battery. This is true but let's point out some problems that need to be resolved when using this approach.

1. The battery needs a source to recharge it. This is no problem if it is operated in a vehicle with a built in charging system, but can be a problem if operated in a home or office. A regular car battery charger will not work because it will overcharge the battery unless one is selected with overcharge protection.
2. The correct inverter type needs to be selected for the particular requirements. Most inverters put out a square wave voltage with a varying frequency that depends on the load. This will not work with disk drives which require a relatively stable frequency. These are fairly inexpensive and will run a television or monitor and the computer. So if you don't need a disk drive then this is acceptable. Special inverters are available that put out a sine wave at the correct 60 hertz which can be used with disk drives but they are much more expensive.
3. A method of turning on the inverter when power is lost must be designed into the system. You don't have time to throw a switch. It has to be done quickly and automatically so all of the memories in the computer will be saved.
4. An inverter has to be selected that is large enough to power the computer and any other required accessories. A 100 watt inverter will power a color computer and a small television. You figure the total power required by adding the power for each accessory. A television usually has the power required on the back somewhere. You can generally figure 25 watts for the color computer. So if the computer requires 25 watts and the television 30 watts then a total of 55 watts will be required. Get an inverter about twice as large as the power required so you will have plenty of margin for additional accessories.

After you have the inverter, battery source, and control circuit installed, it is a good idea to try it out before power actually fails. Load a program into the computer and remover commercial power. The uninterrupted power system should take over and the information should stay in the computer. Restore power to the computer and the uninterrupted power should disappear.

MOTOR GENERATORS

Motor generator assemblies are available that produce 115 volts ac. These consist of gasoline motors connected mechanically to an electrical geneator. Basically they are not very practical for computer back up because of the noise they make and the problem of getting them to start. Something is needed that automatically starts and takes over before the computer can loose the information in its memory. Motor generators would be useful in case of emergencies where power is lost for a long time or for backing up larger computer systems. For this application the motor-generators should be equipped with an electrical starter and a control circuit to start the motor when power fails and turn it off when power is restored. This might be used with a back up battery system for extended power failures.

BATTERY BACKUP POWER

Since the power transformer converts the 115 volts ac to a low voltage dc, can batteries be installed in the computer to provided the dc voltage required when power fails? For 64K computers the power required is 5 volts dc. There are 6 volt recharageable batteries available that will mount under the keyboard. The battery needs to be charged by the computer, but not overcharged. A control circuit is required that switches the battery to the 5 volt regulator to provide the battery power when commercial power is lost. Also a switch is needed to disable the control circuit when the computer is shut down. This prevents the battery from discharging everytime the computer is turned off. So the requirements for an internal battery backup system are

1. The recharageable battery must supply 5 volts for the required length of time which could be 30 minutes, 1 hour, or more.
2. A switch is needed to disconnect the battery when the computer is shut down.
3. A control circuit is needed to charge the battery when the computer is on, to prevent it from overcharging, and switch the battery to the input of the 5 volt regulator when power fails.

We have developed a back up battery system with the required control circuit. The battery mounts under the keyboard or it can be externally mounted. A switch is included to enable the system while it is being operated and to disable it when the computer is shut down. We are offering this as a "SPECIAL" to DCN subscribers. See our offer near the end of this newsletter.

MACHINE LANGUAGE PROGRAMMING

Last month we introduced interrupts. Perhaps we should continue along this line for a while and show exactly what happens when an interrupt occurrs . As a matter of review the 6809E microprocessor contains the following registers:

X - Index Register (16)
Y - Index Register (16)
U - User Stack Pointer (16)
S - Hardware Stack Pointer (16)
PC- Program Counter (16)
A - Acccumulator (8)
B - Accumulator (8)
DP- Direct Page Register (8)
CC- Condition Code Register (8)

The numbers in parenthesis show the size of the register. When an interrupt occurrs everything about the operation of the microprocessor has to be preserved if it is to continue after the interrupt. To understand what happens we need to look at the hardware stack register (S). The stack register contains a vector that points to a memory location. We store a byte in that location by using the "PUSH" instruction and we recover the byte from a stack by using the "PULL" instruction. The terminology generally used is to say "a byte is stored on the stack or pushed onto the stack". What this means is that the byte is stored in the memory location designated by the stack pointer. After a byte is stored on the stack the stack pointer is decremented (decreased) by 1. To recover information from a stack we use the "PULL" command. The stack pointer is then incremented (increased) by 1.

So when an interrupt occurrs all of the registers are pushed onto the hardware (S) stack or stored in memory locations designated by the stack pointer. So everything about the microprocessor's operation is preserved. This means that you can be running a basic program and an interrupt can cause you to run a different program. After the interrupt is completed, since everything about the microprocessor was preserved, the first program continues like nothing has happened.

The process of storing information on a stack is as follows. The stack pointer points to the memory location where the next byte can be stored. After that byte is stored, the stack is decremented by 1 and points to the next lower memory location. So to push data the stack pointer decreases after each byte. To recover information on the stack, the "PUL" command is used. To Pull data the byte the stack pointer is pointing to is pulled and then the stack pointer is incremented by one. The ordering of information on a stack is "the last byte pushed" is the "first byte pulled". So if we pushed A, B, & DP and wanted to recover the information one byte at a time we would PULDP, PULB, PULA.

Now when the interrupt occurs, the microprocessor automatically pushes all of the registers onto the hardware stack so that everything the microprocessor was doing is saved. The interrupt program is a machine language subroutine which is ended by a return from interrupt command (RTI). When the microprocessor sees the code number for RTI it then pulls all of the registers from the stack so it can proceed with the first task it had before being interrupted.

INTERRUPT EXAMPLES

You might wonder how or when can an interrrupt be used. Last month we showed how to install a hardware interrupt switch. The switch connects pins 1 and 2 together on the microprocessor. This is called the nonmasked interrupt (NMI) because it can not be masked off (disabled) by software. The basic ROM in color computers uses memory locations 265, 266, and 267 to allow a machine language subroutine for the NMI. It is pretty hard to write a machine language subroutine in only 3 bytes but a jump extended command can be placed there. So the command for jump extended (126 or hex 7E) can be placed in location 265 and the vector for the location to jump to can be placed in locations 266 and 267.

Now what kind of program can be executed with the interrupt switch? The answer is any machine language program. The only requirement is that it must end with the RTI command. Here are some examples of interrupt programs.

1. Print the contents of the screen to a printer.
2. Exchange the screen display with the display in a designated memory location.
3. Exchange program vectors so a different basic program can be run.
4. Find the memory for a basic statement.
5. Provide a hard reset for the computer. Last month we showed how to use the nonmasked interrupt for a hard reset. This is the only

hardware interrupt that is not used by the color computers. Have you noticed that when you plug in a cartridge the computer automatically runs the program in the cartridge? This is because a hardware interrupt occurs which forces a machine language program to run at the cartridge memory area. Also the timer used with the extended basic is clocked by a hardware interrupt.

There are 3 software interrupts available which are SWI, SW12, and SW13. Last month we showed that the vectors for these point to memory locations 262, 259, and 256 respectively. There are 3 bytes at these locations available for the software programs. What you have to do is to put a jump extended command in the first byte and the vector for the location of the interrupt subroutine in the second and third bytes. After the software subroutine is run then the program returns to its previous function. The advantage in using these is that all of the registers are pushed onto the hardware stack. This saves everything that was happening in the main program and makes for easy machine language programming.

OPERATING HINTS

LOAD DEFECTIVE PROGRAMS

Sometimes you can load programs that have errors near the end of them by resetting the computer before it reaches the bad part of the program. For example for a cassette you can time the loading until the computer indicates an error. Then rewind the tape and try loading it again. Watch the time and reset the computer before the error occurrs and the good part of the program should be in the computer. You will have to scan the program with a utility program to find a good place to put the end of basic vector (locations 27 and 28). This will also work with disks but is harder because of the fast transfer rate.

ASCII BAUD RATE

Locations 149 and 150 determine the ASCII baud rate. For sending information to a printer, modem, or other terminal the proper baud rate has to be sent and received. The following chart shows the values to poke for various baud rates. You can trim the rate if necessary by slightly varying the values in 15●.

Rate	149	15●	Rate	149	15●
50	4	88	180●	0	25
75	2	227	2●●●	0	23
110	1	246	240●	0	18
150	1	110	360●	0	1●
300	0	180	480●	0	7
60●	0	87	720●	0	3
1200	0	40	9600	0	1

****

You may have noticed that we called this the June / July issue. We were behind on our first issue and dropped further behind with this issue. So we decided to combine June and July. You will still get 12 issues because we will add one month to your subscription. If you order our hardware or software products and want to take advantage of your DCN discount please indicate this when you place your order. On your shipping label next to your name is your DCN number followed by numbers representing the last issue you will receive.

****

DCN SUMMER SPECIALS

These specials will be good through August 1984. Add $2 shipping in US & Canada & $3 for other foreign subscribers.

HARDWARE SPECIALS

Uninterrupted power source (UPS)

A 6 volt, 2 ampere - hour rechargeable battery with electronic control circuitry to charge the battery & prevent overcharge plus a small toggle switch to enable the UPS. Battery can be mounted under the keybord to power your RAMS when commercial power fails. Requires soldering one or two wires. This is a new product. List price $59.95, DCN subscriber's introductory price $39.95. Allow 2 to 3 weeks for delivery.

VIDEO REVERSERS

Video reversers reduce eye strain by providing bright characters against a dark background. They are automatically disabled in the graphics modes. Excellent for normal computer operation.

VR-1 . . An integrated circuit with eyelets that mounts on the 6847 chip. A 3 position toggle switch is included that provides (1) All characters reversed. (2) All characters reversed and displayed as capital letters. (3) The normal display. List $19.95. DCN special $14.95.

VR-2 . . Same as VR-1 except in a plug in module. List $24.95. DCN special $19.95.

VR-3 . . A plug in module that provides all characters reversed. No switch. List $19.95. DCN special $14.95

SOFTWARE SPECIALS

We are increasing our discount on software during this period from 20% to 30%. Now is the time to stock up on the software you need.

DYTERM - inexpensive terminal program allows access to bulletin boards. List $14.95, DCN Special $10.45.

DISASM - 6809 Decimal Machine Language Assembler & Disassembler. List $19.95, DCN Special 13.95.

MPM - Multiprogram Manager allows up to 5 programs to be loaded into a 16K or 32K computer. List $14.95, DCN Special $10.45.

UP-1 - Utility program that does many of the everyday operating requirements. List $14.95, DCN Special $10.45.

```
*************************************************************************
* Please sign me up for one year for the  DYNAMIC COLOR NEWS SERVICE. I *
* understand I will receive a monthly news letter, Discounts on DYNAMIC *
* ELECTRONIC INC. Computer  products  plus the  Individual Reply  to my *
* Computer  problems for a  special of $10 each. Also I understand that *
* there  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