ENGINEERING NOTES
on
Radio Shack Color Computers

February 1984
Vol. 1 No. 1

$1.95

DYNAMIC COLOR NEWS

* COMPUTER THEORY
* BASIC PROGRAMMING
* MEMORY EXPANSIONS
* PRORAMMING STACKING
* MACHINE LANGUAGE PROGRAMMING
* QUESTIONS & ANSWERS
* VECTORS & MEMORY MAPS
* MUCH MORE

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 for 12 issues, $20 outside U.S.A.

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 Inc. and the person submitting material will be recognized 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.

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 obtan a personalized reply to specific computer problems or advice on purchasing equipment. The charge for this service is $10.

****
*
* DYNAMIC COLOR NEWS *
*
* February 1984 *
*
* Editor and Publisher *
* Bill Chapple *
*
* Secretary *
* Jo Ann Shamblin *
*
****

CONTENTS

EDITOR'S COMMENTS

A glance at the calendar tells me that it is already February and we should be working on the March issue. Let me thank those of you who subscribed to this service before seeing what we have to offer. Also I want to thank those who gave us suggestions and asked questions about computer problems. Without this interest we would not have begun this project. I am an electronics engineer and engineers are not supposed to know how to write. However I am certainly not a journalist but I know that the concepts we are going to cover are important and I hope that my teaching experience will help in explaining computer concepts so they can be understood. To make sure it is not too technical, it will be read by my secretary. Also I am an electronic circuit designer. My experience ranges from an electronics technician, to an electronics Jr. College instructor, to an electronic engineer and now computers. I was forced to learn computer hardware design because microprocessor components were taking over digital circuit design. I learned machine language programming before I was exposed to Basic. I knew Fortran fairly well so Basic was not hard to learn. I like Basic better than Fortran because Basic does everything Fortran does plus it allows powerful handling of word files.

The applications for computers are endless. Getting information arranged so that it can be computerized is a big problem. Most of the time the computers are idle waiting for us to enter information. The use of interrupts is not well known. An interrupt causes the computer to stop what it was doing and do a different task after which it returns to its initial state. Many monitoring and control operations can be performed with interrupts without any awareness to the user. The future topics we will cover depend on your response. Write and let us know what you would like to see.

The interest in OS-9 has caused us to take a look at this operating system. Maybe next month we will be in a position to comment on it. We are very familar with CP/M which is an operating system for Z80 based computers. Modifications to these systems are almost imposible, but if they are used for their designed purpose then they do a terrific job. The same is true of word processors, spread sheets, and other dedicated programs. There are some technniques we use for handling information using Basic and Machine Language programming which we will cover in future newsletters. The advantage of using Basic programs is that you can modify them to suit your needs and they can be saved on both tape and disk.

STATEMENT OF PURPOSE

As with most new publications the intention and purpose of the publication is usually stated in the first issue. So as a start we will state our purpose for this new endeavor. We intend to provide technical information on Radio Shack Color Computers, instruction on Computer Theory, Basic Program development, Machine Language Programming, Hardware expansion techniques, plus provide a Consulting Service where we answer questions in this Newsletter or provide individual answers to your questions for a small fee. We want to fill the technical void left by the Magazines.

You may wonder why these Computers are so popular. The answer lies in the versatility of the machines and their powerful microprocesor. They can be learning tools, business Computers, Industrial Controllers, Word Processors, Telephone answers, a means of communicating with the Deaf, music composer or instruments, printer spoolers, burglar alarms, etc. The list is endless and no matter what your need or interest is, these Computers can serve your purpose.

What about the structure of these computers when compared to others? Are they Business Computers or Personal Computers? It would be natural to presume that Radio Shack Color Computers are not as powerful as the so called "Business Computers" in the $2000+ price range. It is true that the screen does not have as many characters as "Business Computers". If you want more than the standard 32 Characters by 16 lines then you can add "Character Expanders" and increase the number of Characters on a line to 42 or 64 or more. If you want better resolution then you can get a high resolution television or a Video adapter for a Video terminal.

How do the integrated circuits compare with the more expensive Computers? Let's answer this with a question. Did you know that the same integrated circuits are used in "Home Computers" and "Business Computers"? The 16K 4116 Memory Chips and the popular 64K 4164 Dynamic RAM Chips are used in both. The microprocessors are similar in speed and performance and Input/Output (I/O) circuits are similar. Well what is so different about them? Generally Computers designed for Business use have powerful Software for accounting and word processing needs. The operation is designed so an untrained Secretary or Clerk can learn to use the Computer's features without knowing programming. Also the units are completely contained in one cabinet with dual disk drives and video display.

Color Computers can support several disk drives and software equal to that used on Business Computers is available for Color Computers. A Color Computer's memory can be expanded so that it has the same or more memory than Business Computers. Because of its ease of expansion Color Computers can be made to equal the performance of computers costing much more. Let's look at another aspect of these computers. The games that can be played on Color Computers are second to none. The joystick ports allow items to be moved horizontally and vertically across the screen. Games come in plug in cartridges, on cassettes, or on disks. You can change games easily by inserting a new cartridge, cassette, or disk.

When we look at the joystick ports from an engineering view

point we see Analog to Digital Converters. Immediately accessories of a different nature from games come to mind. Analog circuits are continuous where Digital Circuits have discrete steps. Voltages from 0 to 5 volts are converted into digital steps of from 0 to 63. These can be used to indicate temperature, pressure, voltage, power, etc. The cassette motor is controlled by a relay which can be used for other things such as controlling temperature in a home in conjuction with the Analog to Digital Converters accessed through the Joystick Ports. The cartridge expansion port is a gold mine when it comes to designing accessories. All of the microprocessor signals plus 3 supply voltages appear on the cartridge connector. Almost anything can be designed using this port which can easily be verified by observing the accessories advertised in leading Color Computer Stores and magazines.

BASIC PROGRAMMING

Each month we want to cover one phase of Basic Programming. Basic is a very powerful programming tool and can be used to write any kind of program ranging from serious business programs, games, learning programs, engineering programs, statistics, word processors plus much more. There are many ways to write Basic Programs and some have advantages over others. So we will point out these advantages as we cover various techniques. This month we are going to cover the Immediate Mode of operation, Vectors, plus give you a powerful Basic Program that will allow you to handle several programs in your computer at the same time.

THE IMMEDIATE MODE

This is a mode of operation that is mostly unknown perhaps because of its simplicity. Most of the Basic Commands are available from the keyboard without having to write a program. The best know are MOTOR ON, AUDIO ON, CSAVE, CLOAD, LOAD, SAVE, + various other Disk Drive commands such as DIR, and ?FREE, etc.

CHECK BOOK BALANCING

The Immediate mode allows the Computer to be a powerful calculator. Variables can be assigned for several entries and then the variables added to form the final sums. This is much better than using a calculator because all of the transactions are displayed on the screen. example: Lets assume a check book has the following transactions. The beginning balance is $255.37 and checks (-) and deposits (+) are +23.35, -57.98, +125, -15.19, -119.5, +598.77, -121.35, -98.20, -7.65, -25.50, -11.25, -3.59, -8.99, +25, -3.21, -45.49, -87.22. Now enter the following: A=255.37 -23.35 -57.98 +125: ?A <ENTER> The Sum of 299.04 will appear on the screen and the Computer has assigned this value to the variable A. Now enter
B= A -15.19 -119.5 +598.77 -121.35 -98.2 -7.65 : ?B <ENTER>
The sum of 535.95 will be displayed. Notice that this is the result up to the last entry or -7.65. Now enter
C= B -25.5 -11.25 -3.59 -8.99 +25 -3.21 -45.49 -87.22 : ?C

<ENTER>

The final sum is 375.78. Notice that all entries appear on the screen so that they can be verified. This is a quick way to check the arithmetic in check book programs or any other application involving a calculator. The values of the Variables A, B, and C will be retained by the Computer until power is removed or a program is RUN. Of course any of the other arithmetic functions such as * or / could be used as well as parenthesis.

THE IMMEDIATE MODE WITH A PRINTER

Suppose you want to write a letter that you do not need to edit. To print to the printer use the command ?#-2, "THIS PRINTS CHARACTERS TO THE PRINTER" <ENTER>. The printer will print the characters inside the quotations. This is the equivalent of an electronic typewriter where you compose a line and then send it to the printer. This method is very handy for personal letters or addressing envelopes. We use it when we need to change the mode the printer is set for. We use an Epson MX-80 and sometimes we need to change the type size or add emphasize or double strike. So we just send the printer commands directly from the keyboard.

BASIC PROGRAMMING THEORY

VECTORS

This month we want to introduce Vectors. We picked the subject of vectors because of their importance. What are vectors? In Physics and Engineering vectors are quantities that give magnitude and direction. For Computers vectors can be defined as variables that "point to memory locations". It takes two bytes to define a vector. Since most microprocessors have 16 address lines then it takes two 8 bit bytes to equal 16 bits. The Computer Vector is broken down into two parts called the least significant byte (LS) and the most significant byte (MS). For Motorola 6800 family microprocessors the most significant byte occupies the lower memory location and the least significant byte occupies the upper memory location. To obtain the value of a vector the following formula is used. V is the value of the vector and MS and LS are the most and least significant bytes.

```
V=256*PEEK(MS)+PEEK(LS)
```

When we designate the location of a vector we give the location of the MS. For example the vector that shows where a BASIC program starts is located in location 25. To find the value of this vector we use the formula

```
BE=256 * PEEK(25) + PEEK(26)
```

The vector that points to the end of BASIC is located in memory location 27. There we can let EN be the end of Basic Vector and calculate it as follows.

```
EN = 256 * PEEK(27) + PEEK(28)
```

If you want to know how long your program is then just subtract BE from EN. It is a good idea to put these vector calculations at the beginning of your program so you can tell how long your program is when it is run. An example of how to do this follows:

```
10 BE=256 * PEEK(25) + PEEK(26): EN=256 * PEEK(27) +
PEEK(28)
15 ?"THIS PROGRAM STARTS AT "BE; "AND ENDS AT "EN: ?"IT
USES "EN-BE ;"BYTES OF MEMORY"
```

VECTORS IN BASIC STATEMENTS

Each Basic Statement has a vector at the beginning preceeded by a "0". The vector points to the memory location of the next basic statement. For example suppose the PEEK(25)=6 and the PEEK(26)=1 then BE=256*6 + 1=1537. Then NS=256 * PEEK(1537) + PEEK(1538) where NS is the start of the next statement. The second 2 bytes are the statement number. After this is the Basic Commands. So Basic is arranged as follows.

BYTE #	FUNCTION
0	a "0" seperates Basic Statements
1	MS of next statement vector
2	LS of next statement vector
3	MS of statement number
4	LS of statement number
5	Start of Basic Commands

This knowlege of the Basic program structure will allow you to patch up defective programs. You can scan the characters in a Basic Program and look for the zeros at the start of each statement. Then the vectors can be verified and changed if necssary to cause the program to work.

MULTIPROGRAM MANAGER (MPM)

With this knowlege of Vectors we wrote the "Multiprogram Manager" which allows programs to be stacked in memory. This program is listed at the end of this section and is available on tape to DCN Readers for $6.95 including shipping or you can type it in. It uses a technique with which we have been very successful. We call machine language subroutines from Basic when we have a task that is best done by machine language programs. About 200 bytes of memory are used before the beginning of the program to store the vectors for the various programs plus the machine language subroutines. To get maximum usage from the program do the following.

POKE 25,8: POKE8*256,0: NEW <ENTER> if not using a disk drive. POKE 25,15: POKE15*256,0: NEW <ENTER> if a disk drive is connected. Load in the multiprogram master (MPM) and type RUN <ENTER>. The vectors for the MPM will be displayed in MS, LS format. You can load another program by picking a beginning vector greater than the ending vector of the MPM. The easy way is to take the MS of the ending vector and add 1 or 2 to it and use this value for the beginning vector of the new program. The idea is to start in lower memory and add programs above the last entered one. This program is user friendly and displays the vector components of each program as programs are added. The MPM was designed to handle 5 programs plus the MPM. When used with our 96KX you can load and run 5 programs in both page 0 and page 1.

After you have loaded the MPM and are running another program it is desireable to return to the MPM if you want to

change programs from the MPM menue. The MPM can then be accessed by EXEC 510 <ENTER>. We wrote a machine language subroutine that saves the program vectors for the program you were running and exchanges them with the program vectors for the MPM. The machine language subroutine also calls the RUN subroutine from the Basic ROM causing the MPM program to run.

MPM INSTRUCTIONS

We designed the MPM not to use a lot of memory. It uses less than 2000 bytes. It is menu oriented but we will give some steps for using it. The program is very useful for both Disk and cassette systems. The steps follow:

1. Type in beginning vectors as discussed above.
2. Load the MPM from a cassette or disk
3. Type "RUN ENTER" . The program menu will be displayed. Notice the values of the vectors for the MPM. They will be listed similar to the following.
   0 PGM MGR 20 201 27 204
4. The next to the last number (27) is the MS of the ending vector. Add 2 to this number for the MS of a new program.
5. Press any key for the menu
6. Select (5) to transfer the machine language program to memory. This only has to be done the first time you come to the menu.
7. To load a new program select (1). It then prompts for the value of the MS of the starting vector for the new program. This is the value determined in (4). After entering this you are returned to Basic.
8. You can load another program, run it, or do anything else you want. If you desire to return to the MPM then
   EXEC 510 ENTER
9. The MPM runs and the vectors from the last program you were running are saved. This program can be combined with the others on the MPM menu by selecting (2) from the menu.
10. To combine a new program you will be asked for a program number. This is a number between 1 and 5. Then you will be asked for an 8 character name. It requires 8 characters so if your name is less than 8 use the space bar to fill in the rest of the characters. The menu is returned with the last program combined.
11. Any program on the menu can be selected by selecting (3).
12. After running another program return to the MPM by EXEC 510
13. A program can be deleted from the menue by selecting (7).
14. All of the programs can be saved by (6). To load them again you must put the beginning and ending of basic vectors at the same beginning location as they were when the programs were saved.
15. The MPM can be used in both pages of a 64K system with our 96KX.

MULTIPLE PROGRAM MANAGER (MPM)
PROGRAM LISTING

```
1' MULTIPLE PROGRAM MANAGER (MPM)
2' COPYRIGHT (c) 1984 by DYNAMIC ELECTRONICS INC.
3' ALL RIGHTS RESERVED
10 ?"MULTIPROGRAM MANAGER (MPM)
```

```
15 A=PEEK(25): B=PEEK(26): C=PEEK(27): D=PEEK(28)
20 PS=256*A+B: PE=256*C+D: U(1)=PS-100: U(2)=U(1)+12:
   U(3)=U(1)+24:U(4)=U(1)+36:U(5)=U(1)+48:U1=U(1)+50:U2=U1+25
25 MS=INT(U1/256):LS=U1-256*MS:POKE510,126:POKE511,MS
30 POKE 512,LS:?"TO ACCESS THIS PROGRAM EXEC"; U1
40 POKE 504,A: POKE 505,B: POKE 506,C: POKE 507,D
100 ?"THESE ARE THE PROGRAMS":?" 0 PGM MGR  ";A;B;C;D
105 FOR J=1 TO 5:M=U(J):A=PEEK(M):IF A>128 THEN 130
110 PRINTJ;:FOR K=0 TO 7:A=PEEK(U(J)+K):IF A>128 THEN 130
115 A$=CHR$(A): PRINTA$;: NEXT K
120 ?"  ";:FOR K=8 TO 11:A=PEEK(U(J)+K):?A;:NEXT K:?
130 NEXT J:?:?:?"PRESS A KEY TO CONTINUE
150 A$=INKEY$: IF A$=""  THEN 150
200 ?"SELECT A FEATURE":?"1 LOAD A NEW PROGRAM?":?"2 COMBINE
    LAST LOADED PROGRAM?":?"3 RUN A DIFFERENT PROGRAM":?"4
    RETURN TO PREVIOUS PROGRAM":?"5 TRANSFER ML PROGRAM TO
    MEMORY"
205 ?"6 SAVE ALL PROGRAMS": ?"7 DELETE PROGRAM FROM MENUE
210 INPUTX:ON X GO TO 400,300,600,250,800,950,1000
250 EXEC U2
300 ?"THIS ADDS LAST LOADED PROGRAM TO THIS SET OF PROGRAMS.
310 INPUT"PROGRAM NUMBER";PN:?"ENTER 8 CHARACTER NAME
325 FOR K=0 TO 7
330 P$=INKEY$:IF P$="" THEN 330 ELSE IF P$=CHR$(13) THEN 330
335 ?P$;: A=ASC(P$): POKE(U(PN)+K),A: NEXT K
340 FOR K=0 TO 3: M=U(PN)+8+K: A=PEEK(500+K): POKEM,A: NEXT K
350 GO TO 10
400 PRINT"THIS CONDITIONS FOR ADDING A NEW PROGRAM.
    ENTER THE MOST SIG. BYTE OF THE START VECTOR?
410 INPUT MS:POKE500,MS: POKE501,1: POKE502,MS: POKE503,3
420 POKE 256*MS,0: ?"LOAD NEW PROGRAM":EXEC U2:END
600 INPUT "NUMBER FOR PROGRAM"; Y
610 FOR J=0 TO 3: PM=U(Y)+J+8: A=PEEK(PM)
620 M2=500+J: POKEM2,A: NEXT J : EXEC U2
800 PRINT"ML PGM GOES INTO RAM JUST BELOW THIS PROGRAM
805 PRINT"WHEN PROGRAMS ARE SWITCHED RAM FROM 500 TO 508 IS
    USED FOR TEMP STORAGE.
810 FOR J=0 TO 38: READ X:M=U1+J: POKEM,X: ?"MEM="M,"VAL="X:
    NEXT J:GO TO 10
900 DATA 158,25,191,1,244,158,27,191,1,246,190,1,248,159,25,
    190,1,250,159,27,126,174,117,18,18
910 DATA 190,1,244,159,25,190,1,246,159,27,126,174,117,57
950 ?"THIS SAVES ALL PGMS AT ONCE AS A ML PROGRAM
960 INPUT"8 CHAR NAME FOR PGMS";C$
970 INPUT"MSB OF ENDING VECTOR OF LAST PGM";MS:INPUT"LSB OF
    ENDING VECTOR OF LAST PGM";LS
975 EG=256*MS+LS
977 INPUT"ENTER 1 FOR CASSETTE AND 2 FOR  DISK";XX
978 IF XX=0 THEN 977 ELSE IF XX=1 THEN 990
980 SAVEM C$, U(1), EG,U(1) :GO TO 30
990 CSAVEM C$, U(1), EG, U(1): GO TO 30
1000 INPUT "NO OF PGM TO DELETE"; X
1010 POKE U(X),255: GO TO 10
```

MACHINE LANGUAGE PROGRAMMING

As we introduce this subject, we want to take an approach that is perhaps different from the standard method of teaching machine language programming. Rather than confuse our readers with the vast number of mnemonics (symbols for commands) we are going to take a small portion at a time and show how to use what we have covered. Since our readers will have a vast difference in knowlege on this subject, we will tailor our presentation so that those with no knowlege will be able to understand the concepts.

The introduction to vectors elsewhere in this newsletter will be beneficial here. Whether we use BASIC or Machine Language, the knowlege of vectors and free memory will be equally advantageous. If you don't know what memory is available then you can't use it no matter what programming language you use. Therefore vectors will also be used here to point to memory locations and will consist of two memory locations (bytes) with the most significant byte in the lower memory and the least significant in the upper memory. Also let us point out that the computer operates on "machine language codes". Basis is just a translation of simple English commands to machine language codes. This is why Basic is slower. It takes time for the translation to take place.

WHAT IS a MICROPROCESSOR?

The heart of microcomputers is the microprocessor. You have heard these terms but exactly what are they? The microprocessor has been called "the computer on a chip" and rightly so. However it takes support circuitry and hardware before it can be useful as a computer. It needs memory chips, input/ output (I/O) chips, a keyboard, a monitor, plus other support items.

Now let's look at the internal structure of the 6809 microprocessor used in Color Computers. It contains the following:

X-Index Register (2 bytes)
Y-Index Register (2 byte)
U-User Stack Pointer (2 bytes)
S-Hardware Stack Pointer (2 bytes)
PC-Program Counter (2 bytes)
A-Accumulator (1 byte)
B-Accumulator (1 byte)
DP-Direct Page Register (1 byte)
CC-Condition Code Register (1 byte)

The X and Y registers are used to hold vectors. Memory locations can be determined and operations performed using these registers as a reference.

The U and S Pointers (vectors) point to memory locations reserved for the microprocessor. These locations are called stacks. The pointers keep track of the last location used so it is not necessary to give a location when storing information. The process of storing information on a stack is called "PUSH" and the process of retreiving information from a stack is called "PULL".

The PC is the program counter and points to the memory location of the next command.

The A and B accumulators do the calculations. They are the working parts of the microprocessor. They can be combined for some operations and the combination is called the "D" register.

The DP register breaks the memory locations into 256 pages of 256 bytes. This reduces the length of code required if programming is done within a 256 byte span.

The CC registor contains "flags" that indicate the result of an operation. Suppose you added two numbers. It would be nice to know if a carry occurred. This is obtained from one of the bits of the CC register.

The procedure for writing machine language commands is similar to programming a calculator. You load a register with the first number and add the second number to it. The result is retained in the register. The 6809 allows addition, subtraction, and multiplication but not division. Most microprocessors do not allow multiplication which is one of the factors contributing to their slowness. The numbers that the A and B register can handle are values from 0 to 255. There are no such variables like "strings" in machine language programming. Everything has to be calculated using integers from 0 to 255.

let's say we want to add the numbers 25 and 140. We can load A with 25 and add 140 to A. Now the sum of 165 will be in A. To eliminate confusion we are going to only use Decimal Numbers and not HEX. It is impressive to other programmers if you show a thorough knowlege of hexidecimal arithmetic and make what you are doing to seem as complicated as possible. We are not trying to impress anyone so will only use decimal arithmetic to make our examples easier to understand.

For the example just discussed the operation we did is similar to

```
10 X=25
20 X=X+140
```

We will try to relate machine language operations to Basic as much as possible. We could have also performed the calculation by loading A with 25 and then adding 140 from a memory location to A. Then the result would have still been in A but our program would have the memory location as part of it. This would be similar to the following Basic program.

```
10 X=25
20 X=X + PEEK(M) 'M contains the value 140
```

Now let's look at ways we can load a register. If we designate in the program the value to put in the register then this is the "Immediate" mode and we will designate this by "I". The load command will be designate by "LD". So to load register A with 25 would be written in machine language mnemonics (symbols) as

```
LDA I  25 It would appear in memory as
1000   machine code for LDA I
1001   25
```

An assembler would translate the LDA I to a numberor machine language code for this command. The value of the machine command would be put in 1000 and the value of 25 would be put in 1001. For the immediate mode the value is always in the next memory location.

Notice the format of the comand in 1000. The left 2 characters designate the command, the third character designates the register and the right character designates the mode for the

operation.

Now suppose you want to load an accumulator with a value in memory somewhere. The "EXTENDED MODE"(E) allows the value to be anywhere in the memory map. So let's consider the Extended mode now. Let's say our data is in memory location 20000. The vectors for 20000 can be calculated as follows.

MS=INT(20000/256):LS=20000-256*MS

Now we can write the command at location 1000 as

1000 LDA E 20000 This command means we are going to location 20000 and get the value stored there and put it into accumulator A. The program memory would look like this

1000 Machine code for LDA E
1001 MS byte of 20000
1002 LS byte of 20000

Don't worry about what is put where. That is what assemblers are designed for. Our decimal assembler (DISASM) looks up the machine code for LDA E and stores it in 1000. It then calculates the MS and LS for the extended location and stores these values in the next 2 bytes as shown in the example. This command takes 3 bytes of memory.

CALLING MACHINE LANGUAGE PROGRAMS from BASIC

We want to pause here to point out that the EXEC Basic command allows machine language programs or subroutines to be called from BASIC. We want to write some elementary programs by poking values into memory and using this EXEC Command to run the programs. From the keyboard you can type

EXEC 10000 <ENTER> and a machine language program at 10000 will be run. Don't do it yet as we don't have a program there. Now let's write a machine language program to add two numbers. Let's put our program at 10000 and put the numbers at 500 and 501 and let our result be at 502. We will write a Basic program to handle the operation. The "STORE" (ST) Command is equivalent to Basic's "POKE" Command. So to put the result of our calculations in a memory location we will need to write

STA E 503 We need to define the ADD Command which will be written as

ADDAE 501 This command means to add to A (the fourth character) the value in 501. The left 3 characters are for ADD, the 4th is the "A" register and the 5th character is the Mode which is Extended for this example. The following pieces are needed:

1. 182 is the machine code for LDA E
2. 185 is the machine code for ADDAE
3. 183 is the machine code for STA E
4. the vetors for 500 are MS=1, LS=244
5. The vectors for 501 are MS=1, LS=245
6. The vectors for 502 are MS=1, LS=246 Here is the assembled version of our program

10000 LDA E 500
10003 ADDAE 501
10006 STA E 502
10009 RTS The RTS means to return from a subroutine, the same as "RETURN" in BASIC. The values stored in memory for the subroutine are:

10000 182 (machine code for LDA E)
10001 1 (MS of 500)

```
10002 244  (LS of 500)
10003 187  (machine code for ADDAE)
10004 1  (MS of 501)
10005 245  (LS of 501)
10006 183 (machine code for STA E
10007 1  (MS of 502)
10008 246 (LS of 502)
10009 57  (machine code for return from subroutine)
```

Type in the following basic program.

```
10 FOR J=10000 TO 10009: READ X: POKE J,X: NEXT J
20 DATA 182,1,244,187,1,245,183,1,246,57
30 ?"THE MACHINE LANGUAGE PROGRAM HAS BEEN ENTERED.
40 ?"NOW ENTER THE VALUES TO PUT IN 500 AND 501.
50 INPUT X,Y: POKE 500,X: POKE 501,Y
60  ?"NOW  CALL  THE  MACHINE  LANGUAGE  SUBROUTINE TO DO THE
CALCULATIONS.
70 EXEC 10000
80 ?"LET'S SEE IF THE CALCULATION IS CORRECT"
90 Z=PEEK(502):?"THE SUM IS "Z
95 ?"PRESS ANY KEY TO RUN THE PROGRAM AGAIN
100 A$=INKEY$: IF A$="" THEN 100
110 GO TO 10
```

Next month we will continue with this approach. You may wonder about the rest of the machine language codes. We will cover them and give you a chart or list. Programs can be assembled by hand as we did but an assembler is much better. For those interested we are putting our decimal assembler (DISASM) on special for $9.95 postpaid. It uses the same approach as we are using here.

QUESTIONS & ANSWERS

These are questions that have been asked us. If you have a question please send it in and we will answer it in a future issue. If we can't answer it then we will ask for help from our subscribers.

Question: I want to purchase an assembler but have heard that they can damage my computer. Is this possible?

Answer: No you cannot damage your computer with a program. The worst that can happen is the computer may latch up and you may have to turn off the power to reset it.

Question: Is it safe to plug in a cartridge with the power on?

Answer: No you can damage the microprocessor or the SAM chip. However if you must do this hold the reset button to keep the computer inactive.

Question: How can you advertise solderless memory expansions when other require soldering?

Answer: We designed our memory expansions so that traces do not have to be cut. We get around this by making a complete assembly with wires soldered to pins as needed.

Question: I have a 64K computer but it only gives about 24000 when I ?MEM. What is the problem.

Answer: 32K is about all you can use at a time. The way you use your extra memory is to write programs in 32K blocks (pages) and exchange them with other memory pages. Another way is to use the other pages for files or data and pull the information you need from them.

Question: How can Basic programs be put in a cartridge? I understand that Basic programs cannot be run in the area reserved for ROMS

Answer: You can run programs in the upper memory area reserved for ROMs. This has a great advantage as it leaves all of your memory free and does not require any loading. The beginning of the cartridge area is at 49152 (192,1). The beginning of basic vectors in 25 and 26 must point to this location.

Question: Can programs be moved down from a cartridge into the computer's random access memory?

Answer: Yes the vectors have to be corrected for the final location of the program. Extended Basic will do this for you with the PCLEAR command. The basic beginning & ending vectors must point to the beginning & ending of the program in the cartridge if the PCLEAR method is used.

Question: Should I buy a disk drive?

Answer: After 2 crashed disks it is tempting to say no. If you have a lot of information you need to handle, long programs, and can afford the cost then they are nice. You can destroy all of the information on a disk at a time. This can not happen with a cassette system because information previously recorded can not be destroyed. We use both and it is nice to let one system back up the other.

OPERATING HINTS

We will publish operating hints you send in. All persons sending in information will be recognized if their material is published.

EASY MOTOR OFF: You can stop you cassette without typing "MOTOR OFF". Just type an unauthorized command such as "Z <ENTER> creating an error and the motor turns off.

DISK DOUBLER: You can double the information you can put on disks by cutting notches in the opposite side and punching extra holes in the middle. Several kits with a template are available for this purpose.

EASY DISK BACKUP: Conditioned your disks for operation on both sides as previously explained. When you save something just flip the disk over and save it on the other side. You then have 2 copies, one on each side.

Easy Cassette Load: You do not have to give a name for the program you are going to load. Just type "CLOAD ENTER" and the next program will be loaded.

DYNAMIC ELECTRONIC INC. Products

DCN Specials good through March 1984. We pay shipping on specials. These are available to DCN subscribers only. Use your DCN subcription number when ordering these specials.

1. Multiprogram Master (MPM) featured in this issue. Cassette tape. $6.95.

2. Decimal Dissassembler- Assembler (DISASM) $9.95 .

NEW PRODUCTS

DCN Subscribers can take 10% off the cost of these new hardware items. Add $2 shipping.

96KX-M . . . Our famous 96KX software in a plug in module that mounts in the extended basic socket and the extended basic chip plugs into the module. Nothing to load just EXEC 57701 when you want to move data or use the other 32K memory bank in 64K systems. As a bonus the powerful software is useful for entering data in decimal, hex, ascii or vector formats plus many other useful features. Uses upper 8K of memory in ROM area and allows basic to be run in either of the 32K banks or pages. Solderless installation for D & E Computers, 2 wires to solder for later version computers. $59.95

96KX-C . . . The same as above except in a plug in cartridge. Excellent for cassette systems or disk systems with multipack expanders. $49.95

Interrupt switch . . . Allows the computer to be interrupted and run a machine language program. Use for computer reset when the normal reset fails. 96KX software includes this reset feature. Mounts in 1/4 " hole with instructions. Only $9.95

ME-128-64 . . . If you have a 64K computer you can upgrade it to 128K with this upgrade kit. Consists of a set of 64K chips with sockets for your 64K chips, control circuit board for hardware or software 64K bank selection, plus a 3 position toggle switch that mounts in a 1/4 " hole for manual bank selection. $199

We want your suggestions. Solutions to software or hardware problems, questions to be answered, topics you would like for us to cover, or any other suggestions will be appreciated.

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 & $20 foreign. Start with _____ issue.

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