ENGINEERING NOTES
on
Radio Shack Color Computers

March 1984
Vol. 1 No. 2

$1.95

DYNAMIC COLOR NEWS

* UNDERSTANDING MEMORY EXPANSION

* COMPUTER THEORY
* BASIC PROGRAMMING
* OPERATING HINTS
* MACHINE LANGUAGE PROGRAMMIN
* 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 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

March 1984

Editor and Publisher
Bill Chapple

Secretary
Belinda Paker

CONTENTS

EDITOR'S COMMENTS

Recently there was a microcomputer show in Huntsville. I thought it would be interesting to attend and see what new equipment was being demonstrated. There were several booths representing Sinclair, Osborne, KAYPRO, IBM PC, Hewlett Packard, Radio Shack and many more. Each had personnel knowledgable about the equipment except for one group. You guessed it. The Radio Shack booth was manned by two incompetent people. I use the term "incompetent" because they did not know anything about their equipment. I am sure that you have experienced this when you have tried to get an answer about their computers. What is more depressing is the fact that these people work out of the "Radio Shack Computer Center" in Huntsville. We have purchased a lot of equipment and parts from Radio Shack but only ocassionally have we met a salesperson who vaguely knew what the equipment would do. This ignorance has cost us a lot of time in finding replacement parts and trying to get the correct answer about RS products. Therefore it is not surprising that some of our customers who have recently purchased computers are confused about what computers are and what they can do. In all of the other booths where I asked questions about the equipment, I received logical answers. To me there are only two answers to a question. The first and best answer is the correct answer. The second is "I don't know but will find out for you".

I looked at computer systems in all price ranges. I guess the most expensive was around $8000. Some of the things they could do were better than the Color Computers. However considering the much higher prices, I didn't think the extras were worth it. The impact of the show on me was to reinforce my beliefs that the Radio Shack Computers are still the best buy because of their versatility and powerful microprocessor.

In this issue we talk about memory expansion. I started out with a 4K Computer. With the computer needing about half of that, I only had about 2K for programs. It doesn't take long to use up 2K of memory. I then expanded to 16K, 32K, 64K, and now 128K. I have never used all of the 128K memory. What I would like is a battery inside the computer to power the memory when power fails. I purchased a rechargeable battery before Chirstmas but never did get it installed. It is just the right size to fit under the keyboard. Maybe I will be able to finish that project and give you a report on its operation in a month or two.

We are looking for ideas and suggestions for this newsletter. Without letters from our readers we can only guess at the things you want us to cover. What about hardware projects you can incorporate into your computer? I can think of a few such as memory upgrades, video reversers, ON-OFF lights, heat reducers, etc. Also there is an endless list of external hardware accessories that could be designed to be controlled by the computer. So let us know what you would like to see.

In this issue we discuss variables and arrays. We are going to build on this in future newsletters to develop file handling programs you can use. Our microprocessor theory is being continued so that next time we will be able to show you how to move data from one area of memory to another. These concepts of course work with anybody's assembler. Techniques for handling information are the same no matter what programming language you use.

BASIC PROGRAMMING THEORY

VARIABLES AND ARRAYS

We picked this topic to show how information can be handled. Organization is very important when there is a lot of information to be processed. By putting variables in arrays, the programming is greatly simplified.

NUMERICAL VARIABLES

These are variables on which the computer can do operations. The computer recognizes the name of variables to be the first 2 characters of the label. If we say TAX=3.56 the computer recognizes TA as the variable not TAX. Other examples of variables are X=5: A1=10.3: AC=553. Variables in arrays are followed by one or 2 numbers in parenthesis. Examples are A(5): X(3): AX(15): Y(6,4). We will define arrays later and these examples are just given so you can become familiar with the form of the variables. The computer can perform numerical calculations with these variables.

STRING VARIABLES

The handling of words and word phrases as well as numbers really makes BASIC a powerful programming tool. Words and word pharases are entered into the computer as string variables. There are two distinctions between string and numerical variables. The first is the use of the $ sign in naming the variable. The second is the use of quotation marks surounding the word or word phrases. Some examples of string variables are A$="JOHN ": B$=""SMITH": A$(1)="Hello". The + operator combines two strings. For the examples given suppose C$=A$+B$. Then C$="JOHN SMITH". The + operator just places the second string next to the first. This is the only arithmetic operator that will work with strings and it just combines the two strings.

ENTERING VARIABLES

There are several options available for getting variables into the computer. Perhaps the simplest is having the computer pause and wait for the variables to be entered with an INPUT command. The disadvantage of this is that the variables have to be entered every time the program is run. This would really be time consuming if there were a lot of variables to enter. Another method is to carry the variables within the program. This can be done by defining each variable in the program steps or reading them from DATA statements. Both methods have their advantages. To use READ and DATA statements requires that the information be ordered in DATA statements. The READ command tells the computer the name of the variable and the computer goes to the DATA Statement to get the value of the variable. An example is

```
10 READ X, Y, A$, Z$
20 DATA 35.67, 998.53, Jim Jones, Huntsville
```

The items in DATA statements are separated by a comma. For the example given, after the variables are read, X=35.67: Y=998.53:

A$="Jim Jones": Z$="Huntsville" Data statements can be anywhere in the program. A data statement can contain several variables each separated by a comma. Lost month we used READ and DATA statements for our example of machine language programming. It is easy to use these for a small number of variables or if they are easily ordered. Another way of entering variables is to define them within program statements. For each variable we would have to type A$="HELLO": or X$(1)="Jane Smith". Defining variables within program steps gets pretty monotonous expecially if there are a lot of short string variables.

ARRAYS

Arrays are a means of handling information without having to name each variable. The members of an array are called elements. An easy way to think of an array is to think of information being put in columns and rows. Suppose we want to enter information from a check book into the computer. Let's think of a chart with the check numbers in the first column, the amounts of the checks in the second column, The name of the recepient of the check as the third column, and the date as the fourth column. To inform the computer that we are going to use arrays we use the DIM command. This is put near the beginning of the program and establishes the size of the arrays. An example of the DIM Command follows:

```
10 DIM N(50), A(50), R$(50), D$(50)
```

The numbers in parenthesis shown the maximum number of elements that can be in the array. For statement 10 we can have up to 50 elements in each of the 4 columns. All arrays used in the program have to be dimensioned in the DIM statement. There can be only one DIM statement in a program. It is convenient to enter the information for one check in a data statemnt. The data statement would be similar to

```
20 DATA 101, 35.65, South Central Bell, 2-25
25 DATA 102, 98.75, Credit Union, 3-1-84
290 DATA ,,,,,,,,
```

Statement 290 is added so the number of checks will not have to be counted. We will use a statement to check if the data is 0 to signify the end of data. All of the checks for a month would be entered in similar DATA statements. Now to get the computer to recognize the data we need a read statement. We can write

```
300 FOR J= 1 TO 50
310 READ N(J), A(J), R$(J), D$(J)
315 IF N(J)=0 THEN 330
320 NEXT J
325'N=Check No.:A=Amount: R=Name of recpient: D=Date
330'
```

Now suppose you want to review the checks. A program could be written as follows:

```
330 FOR J=1 TO 50
340 ?"check #" N(J),"For $"A(J),"to "R$(J), "date "D$(J)
350 NEXT J
```

Notice the small amount of programming required to display the information. We will give a check book program in a future issue. We handle our information by putting it in remark statements most of the time. This has many advantages which we will also discuss in a future issue. The understanding of arrays will be needed in developing information processing programs.

UNDERSTANDING MEMORY EXPANSION

Memory expansion is a very popular subject and justifiably so. A Computer with a small memory is limited in the length of a program it can handle. The purpose of this article is to explain what is involved in expanding the memories of Radio Shack Color Computers.

First it is necessary to observe that the maximum memory that the microprocessor can address is 64K bytes. These are bytes that can be directly addressed by the microprocessor. More can be addressed by a technique called "Bank Selection". As an example of Bank Selection consider the plug-in "Y-Packs" or multi-expanders that plug into the cartridge slot. With a mechanical switch or by software, one of two or more memory devices can be selected to occupy this memory area. The cartridge port occupies the area from 49152 to 57343 for a total of 8192 bytes. If an expander has 5 devices that can be selected then a total of 8192 * 5 or 40K + bytes can be selected through this 8K port. The memories used here are called Read Only Memories (ROM) because the computer can only get information from them and not store information in them.

For Color Computers the memory map generally used allows the top 32K of memory to be reserved for read only memories and the lower 32K is reserved for Random Access Memories (RAM). By random access we mean that the Computer can freely put or store information in memory or it can recall or read the information from memory. Therefore expansion to 32K is easy because of the hardware design. This does not imply that the expansion to 64K of memory is not easy. The actual hardware expansion is fairly simple. We will discuss this later.

After a Computer is expanded to 64K there are two ways that it can be used. Most 64K Computers transfer the Basic and Extended Basic ROMs to the upper 32K of RAM. Since each of these is 8K and the cartridge port is 8K, a total of 24K is required by the system. This only leaves 40K of usable memory. So expanding to 64K did not give us much more memory, only 8K more. Now suppose we want to run Machine Language (ML) Programs in a 64K Computer. Then the Basic and Extended Basic ROMs are not required and all of the 64K of memory can be used.

There is relief for the 64K problem. Fortunately the engineers that designed the Color Computers choose a powerful memory control chip the Synchronous Address Multiplexer (SAM). This powerful chip provides refresh signals to the Dynamic RAM Chips and allows for two modes of operation for the memories. The two modes are (1) The lower 32K is RAM and the upper 32K is ROM. This is the normal power up mode and is called memory map type 0. (2) All of the memory is RAM. This is called memory map type 1. Let's look at (1). If we have a 64K Computer and the top 32K of memory is reserved for ROMs then how can we access all of the memory? First the lower 32K is divided into 2 memory pages of 32K each called page 0 and page 1. Now if we can use both of these pages then we will have 2 pages of 32K RAM plus 32K of ROM giving a total of 96K. This is where we got the name for our 96KX Cartridge. Page 1 can be accessed by pokeing 65492 any value. The Computer goes dead when this is done because there is no operating system in page 1. To get around this BASIC has to be initialized in page 1. The normal power up routine initializes page 0 for Basic and ignores page

1. Our 96KX allows the information in page 0 to be transfered to page 1. This initializes page 1 and allows Basic programs to be run in page 1.

32K MEMORY EXPANSION

The Computer requires about 2K of memory for vectors, strings, and the Microprocessor. Therefore a 16K Computer has about 14K of usable memory for programs. Extended Basic clears 4 Graphic Pages when power is turned on. A graphic page is about 1.5K bytes of memory. You can trade graphic page for more memory by poking 6 into memory location 25. Then type "NEW" and ?MEM and a value around 14000 will be displayed for a 16K Computer. This will not work with a disk drive because the disk drive uses the first page for its own memory. When the computer is upgraded to 32K from 16K then all of the added memory is available for your use. In other words the computer does not require any more memory than for the 16K System.

The upgrade to 32K involves soldering sockets onto 4116 chips and plugging the original 4116 chips into the sockets. This is called piggy backing. The Color Computer 2 does not use 4116 chips so this will not work. Pin 4 of each soldered socket is tied together with wire and also connected to pin 35 of the SAM Chip (MC 6883) through a 33 ohm resistor. The resistor is not required but easies the strain on the SAM Chip. This is the least expensive memory upgrade because the original 16K chips are still used. The Color Computer 2 uses 5 volt only type 2118 chips. The piggy back method will also work but a set of 8 chips is relatively expensive. It is usually better to upgrade the Color Computer 2 to 64K.

64K EXPANSION

When replacing 4116 (16K) chips with 4164 chips wiring changes have to be made to the memory chip sockets. One method of doing this is by cutting traces and modifying the circuit board. Also the 74LS138 and the 74LS02 chips have to be modified. The "D" board is the hardest and requires a jumper between the 6821 PIA Chips. The "E" board is a little easier but not much. The 285 board is the simplest of the original computers since the black jumpers do most of the switching. Our approach to this problem was to make prewired assemblies. The memory chips are removed and our prewired assembly installed. Included with the prewired assembly is a 74LS02 and a 74LS138 for the D and E boards. As with the 32K upgrades a wire is connected to pin 35 of the SAM Chip through a resistor.

128K MEMORY EXPANSION

If you can piggy back 16K chips to give 32K, then why can't 64K chips be piggybacked to give 128K? The answer is this is possible but you have to be careful. The problem is that the SAM Chip was only designed to handle 64K of RAM. The additional circuitry needed for bank switching is not included in the computer's architecture. To overcome this obstacle we designed a control board. This board selects one 64K memory bank and puts the unused bank in a standby mode. In this mode the memory chips draw about 1/10 of their normal operating power. Thus the

whole assembly requires less power than a 32K piggy backed computer. With our 128K assembly we include a 3 position toggle switch that allows selection of either bank. In the center position the banks can be switched with software. When used with our 96KX you have the equivalent of four 32K computers in one package.

256K MEMORY EXPANSION?

Naturally this would be the next jump. However 256K chips are very expensive now and it will probably be a couple of years before they become economical. Well what about piggy backing four 64K chips to give 256K? This is a possibility but space becomes a problem plus a 4 bank circuit board would have to be designed.

WHAT ABOUT THE MC-10?

The MC-10 is a cute little computer that has great potential. However it will be much harder to upgrade than standard color computers because static instead of dynamic RAMS are used. Static RAMS do not require refresh circuits. Therefore there is no need for the SAM Chip. The cost per K byte of Static RAMS is about 16 times that for dynamic RAMS. The reason for using them is cost. Well how can you use chips that cost more to reduce cost? For a small memory system the expensive SAM chip is not required. But to upgrade with static RAMS is expensive.

We have looked at the 16K memory expander for the MC-10 from Radio Shack and have determined that it is a good buy. We don't have any experience with this computer but it has great potential. It uses a microprocessor of the same family as the large Color Computers but not as powerful. Maybe next month we will be able to make some comments on it.

There are other memories that can be used for memory expansion. An example is bubble memories. They were introduced a few years ago but did not catch on very fast. They retain their memory when power is removed but are very expensive. As an example a 128K byte bubble memory costs over $ 500. There is work being done on EEPROMS. These can be programmed and electrically erased. They only hold a few K bytes and are relatively expensive.

MACHINE LANGUAGE PROGRAMMING

Last month we introduced machine language programming and gave an example of a ML program to add two numbers. Using machine language subroutines with Basic makes a very powerful programming combination. Some things will not run fast enough in Basic and machine language programs or subroutines are required. An example is a terminal program. A terminal program entirely written in Basic will run up to about 110 baud. This is too slow for the 300 baud and 1200 baud modems. Another application for ML suubroutines would be for memory relocation. You can move close to 100000 bytes in a second using machine language subroutines. This would take several minutes in Basic.

Also last month we discussed two of the addressing modes, immediate and extended. In the immediate mode the value is included in the next byte. The extended mode takes two bytes and allows any location in the memory map to be included. If we load register A immediate with 25 then the value of 25 is put in register A. If we load A extended with 30000 then the value stored in location 30000 is put into register A. What about negative numbers? We need to digress a little and show how to handle negative numbers because this knowledge is needed when we cover indexed addressing.

2's Complement Numbers

A byte can have a value from 0 through 255. If we are going to represent negative numbers then it is necssary to assign some of the values to represent these numbers. These negative numbers are generally used to designate the number or bytes to skip when branching occurs. We can let values from 0 to 127 represent + numbers and values from 128 to 255 represent - numbers. An equation for calculating the value of a byte for 2's complemnt numbers follows:

V = 256 + N where V is the value of the byte and N is the number. If N= -1 then V = 255 and if N= -2 then V = 254.

INDEXED ADDRESSING

This is one of the outstanding features of the 6809 microprocessor. An address can be determined with respect to any of the 16 bit registers which are X, Y, U, S, and PC. For moving blocks of data we can load a register indexed to the X register and store it indexed to the Y register. The X and Y registers can be automatically advanced (incremented) during the process. By increment we mean to increase the value by 1. We use decrement to decrease the value of a register by 1. The format for using indexed addressing is the first byte contains the op-code and the next byte contains the relative information.

Here is a list of indexed addressing possibilities.

Number	Indexed Addressing Mode
0	R+ (increase value by 1)
1	R++ (increase value by 2)
2	-R (decrease value by 1)
3	--R (decrease value by 2)
4	R + no offset
5	R + B offset
6	R + A offset
7	R + 5 bit offset (-16 to +15)
8	R + 8 bit offset (-128 to +127)
9	R + 16 bit offset (-32768 to 32767)
10	not used
11	R + D offset
12	PC + 8 bit offset
13	PC + 16 bit offset
14	not used
15	address

The list shows 14 possibilities. R stands for any of the 16 bit registers X, Y, U, or S. Now we want to show how the postbyte is formed. For all of the conditions the most significant bit is a 1 except for 7 where it is a 0. The most significant bit carries a weight of 128. The X, Y, U, ad S registers have weights of 0, 32 64, ad 96 respectively. We can calculate the value of the postbyte from the following formula: V = M + R + N

M = most significant byte value which is 128 except for N=7. R = Register weight of 0, 32, 64, or 96. N = number from the preceeding list.

USING INDEX REGISTERS

Let's say we want to move a block of data from one location to another. We will use the X register to point to the original data and the Y register to point to the new location. We can load the A register indexed with X and auto-increment X.

To load X and Y we use the notation LOX and LDY. LDX I 1024 means to load the value of 1024 into X. Also STA N, X, R+ will store A indexed to X and increment X. The notation LDA N, Y, R+ will mean to load A indexed to Y and increment Y. R refers to the indexed register.

Now let's look at a practical problem. Suppose we want to move the information on our screen to a new location. The information on the screen is contained in locations 1024 to 1536 and the location we want to move this to is at 10000. Our plan will be to use the X register to point to the next byte in the original location and the Y register will point to the location to which the byte is to be moved.

Next month we will cover some branch tests to allow us to finish this problem. This operation does not take a lot of bytes and is very fast. We suggest you study these notes and read other approaches to assembly language programming if you wish to become proficient in this area.

OPERATING HINTS

If you have an operating hint you think others would like to know about send it to us and we will publish it here.

DOUBLE YOUR COMPUTERS SPEED

You can make your computer operate at twice its normal speed by pokeing 65495, 0. All of your computer's functions may not operate at this rate. To change back to the normal speed poke 65494, 0.

PRINT YOUR DISK DIRECTORY

You can obtain a printout of your disk directory by entering the following. POKE 111, 254: DIR <ENTER>

SAVE SEVERAL PROGRAMS AT ONCE

You can save anything in your computer by treating it as a machine language program. Notice last month how we saved all of the programs in our MPM program. This works on both disks and cassettes. To save a machine language program enter the following.

(C)SAVEM "8 character name", BE, EN, EX <ENTER>
BE = The beginning address in decimal
EN = The ending address in decimal
EX = The execution address in decimal

QUESTIONS & ANSWERS

If you have a question about color computers send it to us and we will answer it here. If we can't answer it we will ask for help from our subscribers.

Question: How can Basic programs be transferred from one computer to another? I have heard that you can do this with modems.

Answer: A modem just converts electrical voltages into tones so they can be processed over audio circuits. The ASCII port is usually used. This is the same port to which the printer is connected. You need a terminal program such as our "DYTERM" program to allow information to be passed through this port. The terminal program has to be able to handle 8 bit words and must have a memory buffer area to receive characters. For two color computers if the program is loaded into the exact same memory area in the second computer then it should run when the correct values are poked into memory locations 25-28. These are the areas for the beginning and ending vectors. See our Feb. newsletter for information on vectors.

Question: Sometimes my computer hangs up and the reset button on the rear will not reset it. Is there a way to force it to reset without turning it off?

Answer: The key to whether a hard or soft reset will occur dpends on the value store in memory location 113. When power is turned on of course there is a 0 in each memory location and the computer goes through the power up routine. We call this a hard reset where all vectors are initiallized. If a 0 is not in 113 then a soft reset occurs when you push the reset button. You need someway to force a 0 into location 113.

The nonmasked interrupt is used with our 96KX and when it is enabled it forces a 0 into location 113 causing a real reset. Sometimes you can do this from the keyboard by entering the following. POKE 113,0: EXEC 40999 <ENTER>. Location 40999 is the beginning of the reset routine.

DYNAMIC ELECTRONIC INC.

HARDWARE PRODUCTS - DCN Subscribers can deduct 10% from these prices.

96KX expanders allow you to use all 64K of your memory in 64K Computers. Transparent until accessed, it works with disk drives and multicartridge selectors. Simply EXEC 57701 from Basic to access its powerful software. In addition to managing both memory pages, the software performs many of the everyday operating and program writing requirements and aids in troubleshooting defective programs. Your 64K computer becomes two 32K computers. There is nothing to load since the 96KX software is in permanent ROM the same as the Basic and Extended Basic ROMS.

* Initilizes the second 32K memory bank so basic can be run in it.
* Transfer blocks of data within or between either bank.
* Enter data in DECIMAL, HEX, VECTOR or ASCII formats.
* Copy machine language programs from magazines in HEX.
* Allows mixed HEX and DECIMAL entries for data storage.
* Performs HEX to DEXIMAL and DECIMAL to HEX Converions.
* Displays program statement numbers and their memory locations.
* Change statement numbers one at a time.
* Gives the beginning, ending, & execute addresses of ML programs.
* Software is included for our interrupt switch for a hard reset.
* Uses top 8K of memory in the ROM memory area.

96KX-M . . . A plug in module that mounts in the extended basic socket and the extended basic chip plugs into the module. 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. Use with cassette or disk systems with multipack expanders. $49.95

Interrupt switch . . . Allows the computer to be interrupted and run a machine language program. Mounts in 1/4 inch hole with instructions. Only $9.95

Video Reverser . . Reduces eye strain. Solderless assemblies with 3 position switch to give you (1) All characters reversed, (2) All characters reversed and displayed as capital letters & (3) the normal display. $19.95

Sam Buffer - Integrated circuit that mounts on the SAM chip and protects it against shorts due to memory upgrading. $8.95

Solderless & Reversible Memory upgrades without trace cutting. 64K upgrades require a 1.1 ROM. EXEC 41175 <ENTER> for your ROM type. State board or computer type when ordering.

ME-3 . . . D, E, & 285 from 16K to 32K $36.95.
ME-4F . . . F or 285 and COCO 2 to 64K $79.95.
ME-4 . . . D & E Computers to 64K $89.95.

ME-128-64 . . . If you have a 64K computer you can upgrade it to 128K with this upgrade kit. Wired assembly consists of a set of 64K chips with sockets for your 64K chips, control circuit board for hardware

or software bank selection, plus a 3 position toggle switch that mounts in a 1/4 " hole for manual bank selection. $199

YOUR BASIC OR ML PROGRAMS IN A CARTRIDGE . . . We can put your programs in a cartridge and you will have all of your memory. To access these you POKE 25, 192: RUN for Basic or EXEC 49152 for ML programs. Nothing to load, faster than tape or disk. Send a cassette with 2 copies of your program on it. Don't use PCLEAR command. Up to 8K $29.95.

YOUR PROGRAMS IN A MODULE . . . We can put your program in the module designed for our 96KX. Cost for up to 8K - $49.95.

We loan Computers

No need to be without a computer while you have it upgraded. Send a deposit of $200 for a 32K or $250 for a 64K Computer with extended basic + $20 shipping, handling, and labor. When you receive our Computer send us yours in the same box. After yours is returned and you verify that it works correctly then return ours for your deposit.

DYNAMIC ELECTRONICS SOFTWARE - Extended Basic is not required. Programs are on a cassette or in a cartridge and are disk compatible. DCN SUBSCRIBERS TAKE 20% DISCOUNT

DYTERM - A terminal program that has the features of more expensive programs. Excellent for bulletin boards or transferring information between two computers. Baud rates from 300 to 2400, 7 or 8 bit word, 1 or 2 stop bits, variable parity, plus user defineable buffer for composing messages off line or storing received messages.
tape $14.95, cartridge $24.95

DISASM - Decimal Disassembler & Assembler. Uses English mnemonics and decimal arithmetic for simplicity. DISASM is a real time assembler and stores the machine codes in memory after each instruction is entered. No need to remember formats as DISASM askd you for the information needed to complete an instruction. The disassembler displays the mnemonics in English and branch locations and numerical values are displayed in decimal. $19.95

MPM - Multiprogram manager allows up to 5 programs to be loaded into a 32K computer. When used with our 96KX you can load 5 programs into both 32K memory pages. Select the one you want to run from the menue. $14.95

COMPUTERS for SALE with Extended Basic and 90 day warranty.

COCO 2 with 96KX & Video Reverser installed. $ 325

Used TDP-100 with 128K RAM, 96KX, & Video reverser. $395

MC-10 Computer. $60

****
*
* 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