ENGINEERING NOTES
on
Radio Shack Color Computers

March 1985
Vol. 2 No. 2

$1.95

DYNAMIC COLOR NEWS

* LARGE MEMORY PROGRAMS (Part 2)
* WRITING PROGRAMS (Part 1)
* BASIC PROGRAMMING
* COMPUTER GRAPHICS (Part 2)
* QUESTIONS & ANSWERS
* OPERATING HINTS

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 if the material is used. We reserve the right to edit all material used and not to use material which we determine is unsuited for publication.

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


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


CONTENTS

EDITOR'S COMMENTS

We are having a few warm days here in Alabama which is a welcomed relief after the bitter cold we had the first of the year. In a way I enjoy the cold because there is no grass to cut and more time is available for computers. As the seasons change so do computers. My first computer was a 4K non extended color computer. I only had about 2K for programs since the computer required about 2K. Now 16K is the bottom line and 64K is very common. New microcomputers are being designed with 128K, 256K and more memory expansion capability.

We were the first to advertise 128K memory expanders for the Color Computers. However they were not well accepted a year ago. Now some other companies have introduced 128K memories and the interest is increasing for larger memories. We have needs for large memory programs as well as a lot of other people. We send out several hundred samples of this newsletter using bulk mail. We need a large file program with the names of people to whom we have sent newsletter so that duplicate copies will not be sent. As we consider a new name it would be nice if the computer could look through all of the names let us know if we previously sent a newlsetter to that person. When you are dealing in the hundreds of names and addresses, it doesn't take long to fill up 32K of memory. So we are excited about large memory programs and as we cover principles needed for writing large programs, we will be giving example programs you can write for you needs.

The new 256K memory chips are droping in price and the 4164 chip prices are also dropping. Would you like a 256K color computer? Our 128K expanders allow four 32K memory banks. If we doubled this then we would have eight 32K memory banks. When used with a battery back up such as our UPS, then a lot of programs and information could be retained within the computer. This would be more information than is contained on a standard disk drive. What approach should we take for 256K expanders? Would adding an assembly with 3-4164 chips piggy-backed with sockets for the 4164 chips in the computer work? This is a good possibility due to the lower cost of the 4164 chips and the lower power feature built into the chips when they are disabled. We will let you know next month what we intend to do for larger memories.

Not only are dynamic memories dropping in price but Erasable Programmable Read Only Memories (EPROMS) are also dropping. 2764 (8K bytes) EPROMS are now less than $10 and 27128 (16K bytes) are less than $18 depending upon where they are purchased. Our 96KX modules are made with 2764 EPROMS. Recently I combined a Basic and Extended Basic ROM into a 27128 EPROM. This left one socket free for another EPROM. If I put 16K of programs in the other socket then I could have the equivalent of 2 extra cartridge programs built into the computer. I would probably put our machine language Assembler & Disassembler (DISASM) in half of the memory and miscellaneous utilities and ML subroutines in the other half. I can visualize a powerful 256K color computer with many self contained EPROM programs, battery back up, and a small mounted television and cassette recorder. Most of the portable computers have only a small amount of memory.

In this issue we discuss how to initialize the second 32K bank for Basic in 64K computers.

Programs are included so that you can use the "hidden" 32K in your 64K computer. With the information in this issue you will be able to write or load a basic program and swap banks. Then you can load another program in the other bank. At any time you can exchange the banks and run either program and have most of the 32K memory available for both programs. Maybe you have a 16K computer and want to upgrade. The following questions were asked recently. "I have a 16K computer and want to expand my memory. Should I expand to 32K?" Our advice would be to expand to 64K because the 64K chips are cheaper now than they have ever been.

We are continuing our series on Computer Graphics. This is a very involved subject and it will take a while for us to cover it. So please bear with us. Graphics give us the ability to see things. Do you want to know how to make graphs, bar charts, and draw pictures on the screen. These are some of the things we want to cover in future issues.

This is the first in our series on writing programs. We can't cover as much material as we would like in each issue, but if you will study the material each month then you can learn to write programs that will do what you want them to do. Our computer classes last about 10 weeks and if we cover the same material in each newsletter as we do for each class then it will take about 10 months to cover the equivalent material.

Most of the people whose subscription expired in February have renewed their subscription. This is encouraging to us and will allow us to continue and improve the newsletter. I want to thank each of you for your interest and support.

Large Memory Programs
Part 2

Last month we introduced large memory programs and discussed some of the problems associated with writing them. We discussed two methods of utilizing large memories. The first involves initializing the second memory bank for Basic. Then a link can take you to the second bank to run a program. This is the approach we will use this month.

The second method involves moving information from one bank to another similar to the action of a disk drive or cassette where information is placed into the same operating memory area. We will discuss this method later since software to do the moving will be required. This software is not too involved and we will probably cover it next month.

We can store information in the microprocessor's registers as well as the PIA registers when banks are switched. If a machine language program is doing the bank switching then after banks are switch the program needs to continue. We don't want the program to stop just because we switched banks until all required data is transferred from one bank to the other. Therefore we need to do one of two things.

If the machine language program is in the ROM memory area that is common to both 32K memory banks then when we switch banks the program will continue. If the machine language program is in RAM then when the banks are switched the machine language must continue from the RAM in the new bank. This means that the machine language program has to be copied into the same memory area of the second bank before it is executed.

This month we will show you

how to copy the machine language program into the second 32K memory bank and how to run basic or machine language programs in both banks after the second bank is initialized.

You may wonder what needs to be initialized. When the computer is first turned on, there is no operating system. It becomes initialized by the reset subroutine. The computer is automatically reset when it is turned on. The reset routine sets up vectors in the lower 300 or so bytes so that Basic can be run. After the routine is completed the computer displays a version of Basic, Extended Basic, or Disk Basic, plus a blinking cursor. This means that the computer is ready for a Basic command from the keyboard.

The initialization procedure is for the first 32K memory bank. There is no action for the second bank and the values in the memories are arbitrary. Therefore the second bank has to be initialized. The easiest way to initialize the second bank is to copy each byte from the first bank into the corresponding location of the second memory bank. Then anything that will run in the first bank will run in the second bank.

This month we are going to present machine language subroutines that will let you initialize the second bank and run programs in it. We will explain in detail how the subroutines work. We picked a memory area starting at 4015 decimal for our subroutine blocks.

Loading Subroutines into Both Banks

The first thing that we will do is to load the subroutines into both banks. Perhaps we should say that we will write a subroutine that will load all of our subroutines into both 32K memory banks. This subroutine is only 26 bytes and starts at 4015 and ends at 4040. The working subroutines start at 4044. Let's look at this subroutine.

Control Subroutine

```
4015 LDX I   4044 ' put the
     value 4044 into X reg.
4018 LDY I   36812
4022 LDA X DIR R+ 'Load A
     Reg with memory value
     X is pointing to and
     increment X.
4024 STA E   65494 'store
     A in 65494. This sets
     the slowest MPU rate.
4027 STA E   65503 'Switch
     to Map type 1 with
     all RAM.
4030 STA Y DIR R+
4032 STA E   65502 'Switch
     back to Map type 0
4035 CMPXI 4500 'Compare
     X with 4500.
4038 BLT   4022 'Branch if
     Less than to 4022.
4040 RTS
```

The preceeding subroutine takes values starting at 4044 and copies them into corresponding locations in upper memory offset by 32768. Then when the memory is returned to its normal configuration the second memory page will contain the subroutines starting at 4044 that will allow us to transfer programs and data between banks.

The subroutine works by loading A with a value while in map type 0, switching to map type 1, storing A in the upper memory location designated by X, switching back to map type 0, then repeating the operation until all of the information is transferred.

BANK SWITCHING SUBROUTINE

The bank switching subroutine is located at 4044. The procedure we use is to clear both the "A" and "B" registers which is the "D" register. This value of "0" is transferred to the "X" register. Since we are going to exchange all bytes in each bank we will start at 0. We load the "A" register with the value in the first bank, switch banks by poking a value into 65493, load "B" with the value in the second bank, Store "A" in the second bank with X as the memory pointer, switch back to the first bank by storing a value in 65492, and storing B in the memory designated by X and autoincrementing X. We then check to see if we have finished, and if not we repeat the procedure. The following is our assembly language listing with comments for the bank switching subroutine.

```
4044 CLRA
4045 CLRB 'PUT A 0 IN A & B
     WHICH IS THE D REGISTER
4046 TFR I  D ,X  'TRANSFER A
     0 TO THE X REGISTER
4048 LDA X DIR R+0 'LOAD A
     WITH THE VALUE IN THE
     LOCATION X IS POINTING
     TO.
4050 STA  E  65493 'STORE A IN
     65493 TO SWITCH BANKS
4053 LDB X DIR  R+0 'LOAD B
     WITH THE VALUE IN MEMORY
     TO WHICH X IS POINTING.
4055 STA X DIR R+0 'STORE A IN
     MEMORY DESIGNATED BY X.
4057 STA  E  65492 'SWITCH BACK
     TO THE FIRST BANK.
4060 STB X DIR  R+ 'STORE B IN
     THE MEMORY DESIGNATED BY
     X. THIS WAS THE VALUE IN
     THE SECOND BANK.
4062 CMPX I 32767 'CHECK TO SEE
     IF WE ARE FINISHED
4065 BLS 4048 'IF X <= 32767
     THEN BRANCH TO 4048
4067 RTS
```

To exchange banks just EXEC 4044 after the second bank is initialized. We will show this later. Now let's look at a subroutine for transferring the first bank to the second bank.

BANK COPY SUBROUTINE

```
4068 NOP
4069 NOP
4070 CLRA
4071 CLRB
4072 TFR I  D,X 'PUT A "0" IN X
4074 LDA X DIR R+0 'LOAD A WITH
     THE MEM THAT X POINTS TO.
4076 STA E  65493  'STORE A IN
     65493 (SW TO 2ND BANK)
4079 STA X DIR R+ 'STORE A IN
     MEMORY DESIGNATED BY X AND
     INCREMENT X
4081 STA  E  65492 'SWITCH BACK
     TO THE FIRST BANK
4084 CMPX I 32767
4087 BLS 4074 'GO TO 4074 IF WE
     AREN'T THROUGH
4089 RTS
```

HEX		DECIMAL	
	CONTROL SUBROUTINE		
0FAF	8E	4015	142
0FB0	0F	4016	15
0FB1	CC	4017	204
0FB2	10	4018	16
0FB3	8E	4019	142
0FB4	8F	4020	143
0FB5	CC	4021	204
0FB6	A6	4022	166
0FB7	80	4023	128
0FB8	B7	4024	183
0FB9	FF	4025	255
0FBA	D6	4026	223
0FBB	B7	4027	167
0FBC	FF	4028	255
0FBD	DF	4029	223
0FBE	A7	4030	167
0FBF	A0	4031	160
0FC0	B7	4032	183
0FC1	FF	4033	255
0FC2	DE	4034	222
0FC3	8C	4035	140
0FC4	11	4036	17
0FC5	94	4037	148
0FC6	2D	4038	45
0FC7	EE	4039	238
0FC8	39	4040	57
	SPACE FILLERS (NOPS)		
0FC9	12	4041	18
0FCA	12	4042	18
0FCB	12	4043	18

methods of entering ML subroutines into memory. So we will leave it up to you to get the values into memory using whatever approach you are accustomed to using. After typing in the values it is a good policy to make a copy since you will probably want to use the subroutines at a later date. To make a disk or cassette copy enter the following:

(C)SAVEM "ML SUBS", 4015, 4089, 4015.

These are position independent subroutines which means that they will run anywhere in the memory map. To use the subroutines do the following:

1. EXEC 4015
2. EXEC 4070

After doing step 2 OK should appear on the screen. It should take about a second to accomplish the transferring.

3. Load a program. This program can be saved into the second bank by EXEC 4044 or EXEC 4070. Use EXEC 4070 when you want the same thing in both banks and use 4044 when you want to exchange information in both banks.

4. Now you can load another program into the computer. The first program is completely hidden and cannot be erased unless you turn off the computer. You can run the second program and when you are finished you can EXEC 4044 to exchange it with the first program.

96KX COMPATIBILITY

For those of you who have our 96KX software, these are the same subroutines that are used in it. Just add 53000 to the locations we are using and you will have the 96KX subroutines.

Next month we will show more on using the second 32K memory bank in a 64K computer.

COMPUTER GRAPHICS
(PART 2)

Last month we introduced computer graphics and briefly explained how a telvision works. We took a look at the Video Display Generator (VDG) and mentioned the different graphic displays that could be presented on the television screen. There is a lot of material that can be presented on this subject so we will have several editorals on Computer Graphics.

The Normal Display & Graphic Pages

There is much confusion about computers, especially with newcomers. People who have not been exposed to an office environment seem to have the most trouble with computer terminology. The term "Graphic Page" is confusing. What is a "Graphic Page"? Think of your television screen as being the page. What is required to write or print information to the screen?

First let's consider the normal display. By this we mean the display that is available when the computer is first turned on. If you will count the number of characters across each line and the number of lines you will get the following results. There are 32 characters on each line and there are 16 lines. This gives a total of 512 characters that can be displayed with the normal display.

Each character occupies a byte. So 512 bytes are required for the normal display. Where is the normal display in the memory map? A memory map is a map that shows what occupies each block of memory. The normal display occupies the area from 1024 through 1535 for a total of 512 bytes. The display memory can be located elsewhere by poking certain memory locations. For our use we will leave it where it is.

Forming Characters

The VDG does the work of forming the characters. A value for each keyboard character is defined by the American Standard Code for Information Interchange (ASCII) which is pronounced as ASK-KEY. If you will recall from our previous discussion of ASCII, each character is given a number. Examples are A=65, 1=49, space bar=32, etc. When a key is pressed, the ASCII value of the key is stored in memory designated by the cursor. The cursor is the blinking symbol on the screen. The location of the cursor is determined by the vectors in locations 36 and 37. If you want to find where the cursor is pointing then do the following:

```
L=256 * PEEK (136) + PEEK (137) : ? L
```

L will be the memory location of the cursor. As far as the screen is concerned the upper left element is memory location 1024 and the bottom right element is memory location 1535. A representation of the ASCII value of the characters displayed on your screen is stored in corresponding memory locations.

Graphics Memory

Let's assume for graphics we want just 2 conditions, either information or no information in each graphics position. The term "resolution" is used to designate the number of pixels used to form a graphic image. A pixel is a small unit area on the screen. If a picture were composed of small dots, then the smaller the dots the better the picture. Also more dots would be required if they were smaller. The lowest resolution consists of 64H x 32V for a total of 2048 blocks while the highest resolution consists of 256H x 192V for a total of 49152 blocks. Since 8 bits can be stored in a memory location and since only 1 bit is required per pixel, there will be a total of 49152 / 8 or 6144 bytes required for the highest resolution graphics display.

CURSOR MOVE PROGRAM

As previously mentioned the cursor is in locations 136 & 137. The valus in these locations form the cursor vector. We want to design a program using the arrors to move the cursor over the screen. If we use the up arrow we want to subtract 32 from the cursor vector. We add 32 for the down arrow, add one for the right arrow, and subtract one for the left arrow. The ASCII values for the arrows are as follows:

8 Left Arrow
9 Right Arrow
10 Down Arrow
94 Up Arrow
12 Clear the Screen

CURSOR DEMONSTRATION PROGRAM LISTING

```
10 'CURSOR DEMONSTRATION PGM
20 'COPYRITE (c) 1985
30 'DYNAMIC ELECTRONICS INC
40 '
50 CLS
60 A$=INKEY$:IF A$=""THEN 60
70 A=ASC(A$)
80 IF A=12 THEN CLS
90 IF A=8 THEN X=-1:GO TO 150
100 IF A=9 THEN X=1:GO TO 150
110 IF A=10 THEN X=32:GO TO 150
120 IFA=94 THEN X=-32:GO TO 150
130 GO TO 60
```

```
140 'CALCULATOR CURSOR VECTOR
150 U=256*PEEK(136)+PEEK(137)
160 'ADD OFFSET X AND CALCULATE
170 'NEW CURSOR VECTOR
180 V=U+X
190 'FIND MS & LS OF NEW VECTOR
200 MS=INT(V/256):LS=V-256*MS
210 PRINT"O";
220 POKE 137,LS:POKE136,MS
230 GO TO 60
```

Run the program and the screen will be cleared. You can then move across the screen with the arrows. A "O" is printed to show your location. You can draw various patterns on the screen using the arrows. If the "CLEAR" key is pressed, then the screen is cleared and a new pattern can be generated.

Writing Programs
Part 1

INTRODUCTION

Did you know that a microcomputer can be programmed to do just about anything you could imagine for doing calculations and writing? If you do a lot of writing a Word Processor will do wonders for you. With a Word Processor you can edit text, move paragraphs, insert text, delete text, append additional text, etc. It was only a few years back when reports had to be done on a typewriter. If a paragraph had to be added to a paper, it would mess up the page structure of the following pages. With a word processor the page structure can easily be reformatted.

What about areas that involve calculations such as check book and record keeping programs for personal or business uses? Computers have always been known for their calculating ability. So microcomputers are a natural for programs that require calculations. Microsoft Basic is a very powerful tool and programs can be written that will do any kind of calculation and can easily handle word strings for labels. For example a check book program would need to have the names of the persons or organization to whom the checks were written as well as the amounts of the checks. It would also be nice if a code could be entered with each check to indicate the category of the check. For personal use this could be for tax deductions. For business use these categories could be for utilities, advertising, automobile expenses, sales, etc.

What about programs that can be purchased? Everyone will purchase programs at times. The problem with some of the more advanced programs is that they are not easily modified. It takes a long time to learn to use some programs and if the program does not do what you want then it may be very difficult to modify. As an example take a word processor program. It will do a fine job of writing text which is its designed function. However if you want to change the word processor, then it is almost impossible if the program was written in machine language code unless you have expertise in modifying machine language programs.

We have been conducting computer programming classes for a couple of years and want to pass on some of our teaching experience to you. The way we conduct classes is to give a few principles on programming and then give assignments to the students to write program segments using the principles covered. Sometimes we write example programs for the students to use. The main thing is that the students have to be involved in order to learn to write programs. This is very similar to learning mathematics. You will never be good at math unless you work problems. The same is true with programming. You have to practice writing programs in order to learn to write programs.

Perhaps the hardest part about writing programs is organization. You have to be orderly and keep things straight. A program should be complete in itself. By this we mean that the computer should give you options to select and ask you for the information as it is required. You should not have to have instructions written on paper to tell you how to run each part of the program. There are two major elements that we like to use. They are the menu and subroutines. We will show how to write menus and use subroutines plus much more. It will be a little harder for us to write instructions for students to try on their own rather than giving them verbally. However we will try to give exercises that you can do to force you to think in the logical method that is required for writing programs. The computer is very obedient and will do exactly what you tell it to do. There are very few instances where "Computer Error" occurs. Mistakes are usually "Operator Errors".

In summary if you follow our exercises then you can learn to write your own programs. This gives you the advantage of writing programs to do exactly what you want. There are no programs on the market with this characteristic. As stated before it is not hard to learn to program but requires effort so let's get started.

Program Structure

Basic requires statements to be ordered into statement numbers. This helps you keep up with what you are doing as well as telling the computer the order that you want the commands to be performed. The range of statement numbers is from 0 to above 60000. It is generally desireable to use lower numbers because they are easier to type. A statement can contain several commands which are seperated by a colon (:). A statement can be replaced by a new one by typing in the new statement and pressing <ENTER>. We will use <ENTER> to mean press the enter key. You can insert additional statements between previous statements by typing in the new statements and pressing <ENTER>. The computer will put the new statements in the proper numerical order in relation to the old statements. Now let's look at some of the Basic Commands.

NEW COMMAND

From the keyboard type "NEW <ENTER>". This erases any previous program you may have had in the computer. It is a good idea to do this if a program has been previously entered. If you don't do this then you will only replace statements in the first program as you write new statements. So it is a good idea to erase all old program statements with the "NEW" command before starting a new program.

REMARKS

A remark can be entered by typing either "REM" or "'".

The ' symbol is easier to type and it is recommended. A remark statement allows you to put labels within your program for your own use in identifying various sections of the program. The computer ignores information in remarks. At the beginning of each program should be several remarks identifying the program and explaining what it does. Also within the program there should be remark statements that head each major section so that you can know what that section does. The following is an example of using remarks to head up a program.

```
10 'DEMONSTRATION PROGRAM
15 'FOR BASIC PROGRAMMING
20 'FEB. 14, 1985
25 'DYNAMIC ELECTRONICS INC.
30 'EXAMPLE ON USING REMARKS
```

Remarks can also be used for containing data and machine language subroutines. In previous editions of this newsletter we showed how to use remarks for these purposes. For our use in writing programs we will just use them as labels or comments.

LIST & LLIST

The LIST command tells the computer to display the program statements on the screen. To list the entire program type "LIST <ENTER>". You can list part of your program. Examples of list commands follow:

(1) LIST
(2) LIST - 30
(3) List 50-300
(4) LIST 600-

For (1) all of the program will be displayed. For (2) statements up to and including 30 will be displayed. For (3) statements between 50 and 300 will be displayed. For (4) all statements from 600 up will be displayed.

The LLIST command is for the printer. IF you have a printer you can list all or part of the program as previously stated. For the 4 examples given just put an extra "L" in the list statement for listing to the printer.

Next month we will continue. It is always harder introducing a subject because there is no background upon which to build. We will be giving programs and suggestions so you can develop a program writing technique.

LETTERS
to the Editor

Dear Mr. Chapple:

I thoroughly enjoy reading Dynamic Color News and I am continuingly intrigued by the possibility of increasing my COCO's memory with your ME-128-64 memory expander. The ads cannot go into detail concerning the technique for using them and I thought I would ask the question directly. Your response might also be a useful article in Dynamic Color News.

I have read the "buzz" words about "bank switching" but I haven't the vaguest idea what this means concerning programming. Could you comment on what needs to be done if I were to write a program or a file that required more than the present 32K, or so-called Map 0, in terms of program commands? If not perhaps you could recommend a book or article that would help me to better understand how extra memory is accessed. I have the Motorola specifications for the SAM Chip with the map types ("The Facts for the TRS-80 Color Computer, by Spectral Assoc.) but the technical description is not helpful to me.

Your comments would be appreciated. Thank you.

J. Stewart Campbell

COMMENTS: Mr. Campbell's letter was written last Fall. The problem of using all of the memory in Color Computer's is being discussed in our present series on Large Memory Programs. We study the Motorola data sheets and use this information for our designs and discussions in this newsletter. If we haven't answered all of the questions Mr. Campbell presented, we will in our future newsletters.

QUESTIONS & ANSWERS

Qestion: I enter a lot of data and wonder if a numerical keyboard could be added as an accessory to make it easier to enter numbers.

Answer: Yes an additional key pad can be added. One of the PIA chips is programmed for the keyboard software. A 40 pin socket could be used to connect wires to appropriate pins on the PIA chip.

```
****************************************************************************
* Please sign me up for one year for the  DYNAMIC COLOR NEWS SERVICE. I *
* Want to receive instruction on programming, Computer Theory,  Operat- *
* ing Techniques,  Computer Expansion plus the Individual Reply  to  my *
* Computer  problems for a  special of $10 each. Also I understand that *
* 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

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