SANTA CRUZ EDUCATIONAL SOFTWARE

Presents

Tricky Tutorial(tm) #2

Horizontal & Vertical Scrolling

****

HOW TO LOAD

****

TAPE.....

First, if you haven't cleaned your tape recorder heads lately, please do so now. SEE ANY STEREO SHOP FOR THE RIGHT TOOLS. Place the tape with the label side up in your recorder. Make sure it is rewound and also reset the counter to 0. Push PLAY on the recorder. TYPE RUN"C:" AND PRESS RETURN. If this doesn't work you might try CLOAD. We also include a backup program on the shorter programs we sell (not the TUTORIALS), and these sometimes require CLOAD. If the program won't start to load, try positioning the tape forward or backward a little bit at a time. The easiest way would be to LISTEN to the "noise" on the tape with a regular tape recorder. When you find the steady tone that lasts for about 8 seconds, you have the beginning of each program. We recommend you write down the number on your recorders counter as each program starts. This will make it easier to find each part later on if needed. We occasionally get a bad tape from our supplier, so if yours won't load on both your own 410 and a friends (or your DEALERS), call (408) 476-4901 for a replacement.

Once the program starts, it will load in the remaining parts. The multiple parts are needed so that machines with only 16k can enjoy the TRICKY TUTORIALS. With more memory you can go beyond these simple tutorials. After each part, the computer will beep. This is your signal to press return to load in the next part. Some newer programs will start the next part without pressing return. Most programs will run themselves when done loading.

The reason for several methods of running tapes is that as we improve the programs, we change the masters; however, the manuals are printed in large amounts, so changes are VERY hard to get into your manuals!

****

DISK.....

To load & run the disk you first have to turn on the drive. When the busy light goes out place the disk in the drive. Now turn on the computer with the basic cartridge in place and the program will load each part and run by itself (aren't disk drives nice!).

Introduction To Santa Cruz Software

Hi! First let us thank you for purchasing our programs. We want to start out by telling you a little about ourselves and our company. Santa Cruz Educational Software is the result of several local programmers who initially purchased the ATARI computer as a home machine around the time the 800 was first sold. At that time, and even today, we found a lack of information about how to use the power of the machine. As you have seen in a few of the better programs being sold, the ATARI can do more than any other computer in it's price range, but how could we learn the many "tricks" that were contained withen the machine??

Well, our local club, although it had several HUNDRED members, ignored most of our questions. Fortunately other clubs across the country have some excellant newsletters full of programs and facts. Also, ATARI(tm) has given we owners the Operating System and Hardware manuals. Finally, many magazine articles and several books have been published.

All this is great for programmers, but what about the average owner who doesn't understand much of what is said in the magazines? Even as a programmer, I have to spend many hours studying all of the information that is now available inorder to understand some of unique things about our machine.

The story ends when I asked some local club members to submit their best programs for us to offer to others at the lowest possible price, ie. a price WE WOULD be glad to pay to buy these programs. I myself wrote most of the TRICKY TUTORIALS(tm) and the MASTER MEMORY MAP. Now, others across the country have offered programs for us to sell, all meeting the goal of: minimal on fancy artwork and a few spelling errors, but worth every penny.

Write us with new ideas or memory locations to share, and if it's something we can use, a reward will be sent back to you as soon as we can. Also, please remember that this is only a part time business. We are usually late in getting new programs out due to last minute debugging at midnite(no kidding, I do all this after my regular job). This doesn't mean we don't care!

My thanks to all of you who have raved about the Tutorials. I will try to keep a new one comming out every three months or less, and also to get out those bugs that remain.

ROBIN ALAN SHERER (owner, programmer, and janitor)

OK! Back to Work:

Warning!

In most of our examples we wanted to keep the amount of Basic code short, so if you go too far in any direction the pokes we are doing may cause the program to error and stop. To avoid this problem in actual examples you will have to make sure you don't try to poke a number larger than 255 into memory. Some of examples will do this. To restart a stopped example, just press RESET and type RUN.

Here is the code for Example 1, a simple coarse horizontal scroll:

```
10 DL=PEEK(560)+256*PEEK(561)
20 DL4=DL+4:DL5=DL+5
30 PDL4=PEEK(DL4)
40 ST=STICK(0)
50 IF ST=11 THEN PDL4=PDL4+1
60 IF ST=7 THEN  PDL4=PDL4-1
70 POKE DL4,PDL4
80 FOR W=1 TO 50 :NEXT W
90 GOTO 40
```

THAT'S IT!

Another note - Yes we know that if you list EX.1 it has more lines than this. The missing lines above were put in EX. 1 to make it have a title and call EX. 2 when your ready. The same differences will occur with our other examples.

To better understand this method, first a few basics from the Display List Tutorial. The Display List is a set of instructions in memory that the computer uses to find out what to put on the screen. By changing it, we can do many programming tricks that are not only interesting to look at, but useful. Scrolling is one of those tricks.

Line 10 above looks at locations 560 & 561 by use of the PEEK command. The values in these two memory locations combine to give you the location (called the address) of the first byte of the Display List. They are combined into the address as line 10 shows. ANYTIME you have a 2 byte address, the method in line 10 is used. Whenever you use the

"GRAPHICS #" command, the computer creates a standard Display List somewhere in memory and puts the location in 560/561.

Now that we know where the DL(Display List) is, line 20 stores the address of the low part(DL4) and high part(DL5) of a number that is always the 5th and sixth numbers in the DL. Note that we call the 5th number into the DL, DL4, because it is equal to the value at DL plus 4 more, or the 5th number from the start. Confused? Then just remember that these two numbers tell the computer where to go get the data for the screen.

IMPORTANT!

You must understand what these two numbers do. Scrolling is based on changing these values. Let's look at a sample GR. 0 Display List :

```
112 112 112 66 95 34 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 65 89 56
```

The first three numbers are 112's that tell the computer to display 8 blank lines each. The next number is called the load memory scan byte(LMS for short). This is the number we are about to change. The LMS says to the computer "Go get data starting at the location in memory that follows in the next two numbers...and by the way, start the screen with some of this knid of text(or graphics).

Wow. I can't learn this!

After reading my explainations, you may think I DIDN'T learn it either, but it's not that hard. You must practice on your own beyond these examples. Try making modifications to what we have given you. Later we will give you a chart that shows what number goes into the LMS byte (4th # in DL). In this case the chart would say that for a Graphics 0 LMS,we use 66. The next two numbers (5th and 6th bytes) are the low and high parts of the location where the screen data starts in memory. The example above was made up, so your numbers will be different depending on your memory size. That's why we store the value of the 5th #(DL+4 means the 5th number) in lines 20 & 30. We need to know where the data is in YOUR computer.

Continuing on explaining the DL, the 2's tell the computer to put a line of mode 0 on the screen for each 2.

Again the chart will give you these numbers later for other modes and types of scrolling. There are 23 2's plus the line of mode 0 that the LMS created gives 24 lines of mode 0. If you want, look ahead at the chart. Under graphics mode 0 it says to put 2 in the DL,ie. 2 is the instruction that MEANS put some GR.0 on the screen. Finally comes a 65 that means all done, go start all over again using the DL at the location in the next two numbers(usually the same DL).

Ok! So far we have found where the data to put on the screen is located(lines 20-30). Now, in line 40 we read the joystick. If it is held left, then we take the value in DL+4, called here PDL4, and add 1. If it is held right, we subtract 1 in line 60. Now line 70 pokes this changed value back into the DL at DL+4. This says to the computer "start displaying the data on the screen one byte more(or less) than where you did before." SIMPLE...

Line 80 is a delay loop to slow the action down so we can see. Line 90 then goes back to read the joystick to see if you want to move the display some more.

Notes:

1)You can speed up this method by changing or leaving out the delay loop.

2)So far we haven't changed the high part of the data address(DL+5). If you try to move continually in one direction the value of PDL4 will reach 0 or 255. Recall that the ATARI uses 8 bits per word in memory, ...and eight bits in binary only counts to 255. When this happends, the number you Poke into DL+4 will no longer have any effect. Actually it is causing errors that the example is TRAPing to keep the program going. It's sort of like bottoming out. The simple solution is to do some mathamatics to change the high byte(DL+5) when needed. We do this in EXAMPLE 2.

EXAMPLE 2

This time we scroll vertically:

```
10 GRAPHICS 0
20 DL=PEEK(560)+256*PEEK(561)
30 DL4=DL+4:DL5=DL+5
40 NUML=PEEK(DL4)
50 NUMH=PEEK(DL5)
60 ST=STICK(0)
```

```
70 IF ST=14 THEN NUML=NUML+40
80 IF ST=13 THEN NUML=NUML-40
90 IF NUML<0 THEN 140
100 IF NUML>255 THEN 160
110 NUML=NUML-256
120 NUMH=NUMH+1
130 GOTO 160
140 NUML=NUML+256
150 NUMH=NUMH-1
160 IF NUMH<0 THEN 60
170 IF NUMH>255 THEN 60
180 POKE DL4,NUML:POKE DL5,NUMH
190 GOTO 60
```

The only real difference in COARSE vertical scrolling over that of horizontal is that to move vertically we want to add or subtract 40 instead of 1. Do you know why? Of course! It's because we are using the ATARI's standard 40 characters per line for mode 0. Thus to move the LMS address up one line add 40. This is done in line 70. For other modes you may need to use 20 or 10 instead of 40. Look ahead at the chart under the row "Bytes of memory per line". Also this time we include the mathamatics to change both numbers of the address of the data to display. The math won't be explained.....just use it as is. It is standard computer math explained in most referances including Our MASTER MEMORY MAP. Finally, note that no delay loop was used. The extra math delays the computer enough!

EXAMPLE 2A

This is a new example added by suggestion from one of our customers. Some people didn't see how to combine horizontal and vertical scrolling, so here it is. You can later do the same thing for FINE scrolling. See Micheal, I told you I respond to letters!

EXAMPLE 3

Look at the top figure(1). The box drawn represents the screen area which is 40 characters wide by 24 down. You can see that if we attempt to scroll horz. over this area we have nowhere to go, so the data on the last line just moves up to the next line one character at a time. What can we do? Figure 2 shows some data that is now wider(how wide is optional with you). The box that represents the screen now has room to move to either side when horz. scrolled.

Lets sketch out an abreviated screen of data:

Data interpreted 40 bytes at a time

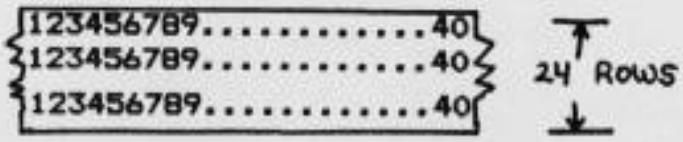


FIG. 1

Data interpreted 256 bytes per line

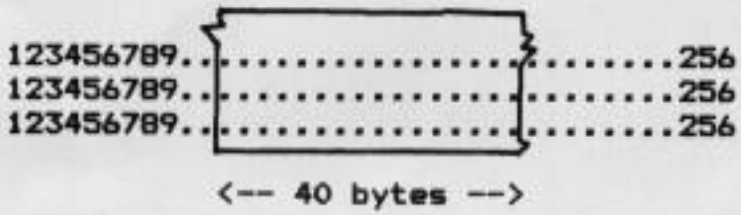


Fig.2

The idea in the above paragraph is not easy to grasp. Let's start with some basics. Memory is organized like a very long strip of numbered locations. When we say to peek at (look at) a certain memory location, what we are really doing is to go down this strip of locations until we get to the number in question, then we read what is written there. Got that? Good. When the computer starts to draw the screen... 60 times a second ....it goes out to the location we discussed earlier (DL+4 and DL+5). There it gets the location for the start of the screen data. In a 48 k machine in GR. 0, this would be at 40000. If an "A" is currently being displayed in the upper left corner, a PEEK at this location will show "33".

To draw the rest of the screen, the computer looks at the next 39 locations (for a total of 40) and draws them across the screen. Now it says to itself: "Since I'm in

GR.0, I must place the next value back at the start of the screen, but one row down. This means the 41st. value goes right under the 1st. This goes on until the screen is filled (after 40*24 values have been placed on the screen). What we do to make memory seem like FIG.2 is to fool the computer into taking every 256th value as the start of a new line. This allows us to move sideways over the data. The sideways comment is from the perspective of the screen... memory is still just that long strip of locations and we are still taking 40 memory values at a time for each row of the screen in GR.0, but after 40 values are used we skip 256-40 values before the next line is drawn. The other 216 values are being saved for when we scroll. In Fig.2, imagine that the initial screen shows us values from 8 to 47. To scroll left , just have the screen show values from 7 to 46, 6 to 45, etc. When we get to showing 0 to 39 we must stop or the values will jump up a line as we would now be showing 255 of the last row through 38 of the current row. This is what we saw in example 2.

Here then is example 3:

```
10 P106=PEEK(106)
20 POKE 106,P106-24
30 GR.0
40 DL= PEEK(560)+256*PEEK(561)
50 P560=PEEK(560):P561=PEEK(561)
60 DL4=PEEK(DL+4):DL5=PEEK(DL+5)
70 FOR I= 1 TO 24
80 POKE DL+3*I,66
90 POKE DL+1+3*I,DL4
100 POKE DL+2+3*I,DL5+I
110 NEXT I
120 POKE DL+78,65
130 POKE DL+79,P560
140 POKE DL+80,P561
150 K=0
160 ST=STICK(0)
170 IF ST=11 THEN K=K+1
180 IF ST=7 THEN K=K-1
190 FOR L=0 TO 23
200 POKE DL+4+3*L,DL4+K
210 NEXT K
220 GOTO 160
230 POKE 106,P106
```

PRETTY COMPLICATED, HAH?

WE'LL TAKE IT SLOW: Location 106 holds the number of pages that your memory size allows. In my 48k machine it is 106. In line 10 we store the original value to restore in line 230 when the program is done. Now in line 20 we fool the computer by subtracting 24 pages from 106 (a page is 256 bytes of memory). Then, in line 30 when we say graphics 0, the computer looks at location 106, thinks memory is 24*256 bytes less than it is, and places the Display List down by this much in memory. What we have so easily accomplished is to save that many extra bytes of memory for a lot of data for the screen. Lines 40 to 60 store the location of the DL and start of screen data as before.

The next lines, 70 to 110, do a nice trick. Now that we have such a large area to use, we poke the DL with a whole bunch of LMS,low,high instructions, with each address pointing to data 256 bytes further down in memory. So what? Well, now there are lines on the screen (run ex.3) that are 256 bytes long. With the joystick you can scroll and see NEW information appear....not from the row below as before. Remember that there has to be enough of these LMS pokes to fill the screen, which for GR.0 equals 24, thus the loop 1 to 24 on line 70.

The data we are scrolling over is again placed on the screen by the LIST command. Now you can see where the ending of each 40 bytes is, because that is where each new line of the program listing starts.

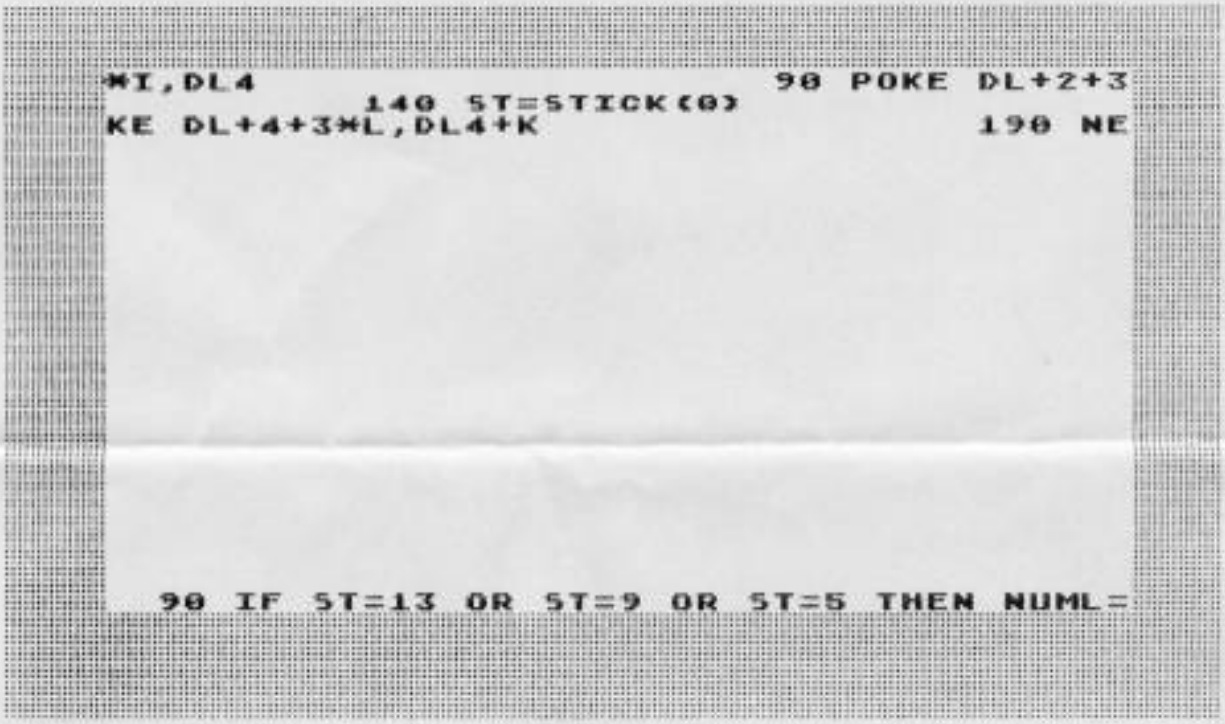
To better visualize this, look at Fig.2 again. The LMS addresses should point to the data for the left corner of each line. It sounds complicated, but by using 256 long lines we can just add 1 to the original DL5 value for each new LMS we set up. The new value will point down by one line of data...256 bytes!! Remember that values in the high byte of a number equal 256 times those in the low byte(just use the example if this is too hard...come back to it later) . Look at line 100 to see the above.

You could, with complicated mathamatics, set up lines of any length you want, but lines of 256 characters were easier and clearer for a demonstration like this. Lines 160 to 220 increase or decrease the DL4 number by one, as in example 1, but now the lines are 256 wide, so we can scroll without having characters jump up to the next line. Also note in

line 200 that we only POKE every third line. This is because the DL must now uses three lines of instructions for each line generated on the screen.

What if I'm lost

That is what is nice about the Tricky Tutorials - You don't have to understand all this. Just use the examples by copying them to your tape/disk and modify them with different Print and Plot commands. The technical stuff is included to make a complete package for your use when ready.



EXAMPLE 4

PLease run example four. If you wish, you may break the program and look at the program listing inorder to follow the discussion. Notice that we used Graphics 2 for this example. This was to show you that any mode will scroll. Also, we relocated the entire Display List to page six of memory (areserved area for us to use) so that it could be modified while leaving the original intact.

Different graphics modes!

What happends when you change graphics modes? Well look at that famous chart we keep promissing(next page) and read the discussions later.

HORIZONTAL & VERTICAL SCROLLING

LOOK AT GRAPHICS MODE(0-8) THEN GO DOWN UNTIL DESIRED OPTION IS FOUND

Desired Option	GRAPHICS MODE								
	0	1	2	3	4	5	6	7	8
DL #	2	6	7	8	9	10	11	13	15
HScroll	18	22	23	24	25	26	27	29	31
VScroll	34	38	39	40	41	42	43	45	47
HS&VS	50	54	55	56	57	58	59	61	63
LMS	66	70	71	72	73	74	75	77	79
HS & LMS	82	86	87	88	89	90	91	93	95
VS & LMS	98	102	103	104	105	106	107	109	111
HS,VS,LMS	114	118	119	120	121	122	123	125	127
# Pixel lines per mode line	8	8	16	8	4	4	2	2	1
# Rows to fill screen	24	24	12	24	48	48	96	96	192
Bytes of memory per line	40	20	20	10	10	20	20	40	40

65 = Start again at top of Display list
112= 8 Blank lines

EXAMPLE 5

This example generates some random characters and pokes them right into the screen data area. To some of you, this will look like a useless example, but please think a moment. Inorder to get the text YOU want onto the screen, just replace the random statements in lines 120 to 170 with print statements. If your text is not 256 characters wide, then fill in with blanks. The text could be stored in Data statements and then read in as needed. Think what a nice business spread sheet could be done!

Did you notice that this example scrolls faster. The reason for this is that in ATARI basic, for-next loops (which we use) go back to the first line and count forward on every loop. This means that they run faster if near the front. Also, if several lines are placed on one line speed increases. Finally, any math calculation should be done only once if possible. This example has these changes for you to study:

```
50 X=DL+4:FOR L=0 TO 69 STEP 3:POKE X+L,Y:NEXT L:RETURN

270 ST=STICK(0):K=K+(ST=11)-(ST=7) :Y=DL4+K

310 GOSUB 50
```

LINE 50 does the exact same thing as lines 190 to 220 in example 3. It is a subroutine to allow it to be moved to the top. Line 270 is much faster than the two IF statements. I better explain it. If ST = 11 then the ST=11 part will be TRUE, so (1) will be added to K. Likewise with ST=7. Since both can't be true at once, K will either go up or down by one each pass if the stick is moved. Atari Basic has it's nice points too!

EXAMPLE 6

What about a practical use? Well, how about a word processor that would allow you to input the width and then not wrap around, but rather scroll right or left as you want. To use example 6, just type in a bunch of lines of text(almost filling the page is best). When ready, press return. Now use the good old joystick to scroll. If you look at the code, we added only a few lines. The poke 559 at line 140 turns off the screen until we want it on at line 220. This speeds up the ATARI and looks more professional.

EXAMPLE 7

This looks just like the last example doesen't it. Well, when you press return and try to scroll something funny will happen. The screen will go "crazy" because we POKE the wrong numbers into the DL. Look at the code and find the error. This error is typical of the type that occurs when modifing displays.

Now, just like the contests that a certain computer company holds, we will give $50. to the LAST person to send us the correct answer.(Can you believe some people actually wrote us hoping they would be the last one and thus win a prize! It's a joke, folks...don't write in.)

EXAMPLE 8

This time we point out that you don't have to scroll all the lines on the screen. You can enable scrolling for only the lines you want. Look at lines 170 to 190 in this example. We POKE only the 3rd to 5th lines on the screen. You can scroll from one line to a screen full. Any mode line in the DL can be set from regular (#'s 2 to 15) to (see the chart) scrolling (ex...GR.6) horizontally (27), vertically (43), or BOTH directions for diagonal scrolling (59).

The other possibilites for a mode line are LMS (LOAD MEMORY SCAN) mentioned above; HS & LMS, which is when you want tell the computer where to get its screen data and also have that line (remember LMS sets up a line on the screen also) be able to scroll. The same applies to vertical scrolling & LMS, and Hs, Vs, & LMS.

Yes, I am using a lot of terms and abbrieviations. You may feel better knowing that even I, as a programmer, was confused at first. The concepts we are teaching here are NOT Basic programming, but rather Graphics programming, which is a new and complex area. After this lesson, some of you non-programmers will be way ahead of many professionals unfamiliar with graphics.

EXAMPLE 9

This example is coarse scrolling, but in the vertical direction with GRAPHICS. Just substitute your own PLOT commands if you like. The method is the same, only the numbers have been changed using the chart.

Fine Scrolling—Vert.
EXAMPLE 10

If you look on the chart, you'll see that for GR. 0 the regular number for the DL is 2, but to scroll(fine) vertically use 34. In the code for this example we POKE 34 into the DL in lines 25 to 40. This tells these lines to fine scroll, but how much? Fine scrolling means instead of moving a row (column) of letters or graphics one character or graphics pixel (these come in different sizes - see your Basic manual) at a time, we move these characters one TV pixel at a time. TV pixels are the tiny dots you see when you look CLOSELY at your TV, or Monitor.

To fine scroll, we change the value in a new location: 54277. This location may contain 0 (normal) to 15. These numbers are the number of TV pixels that the line will be moved. This example is in a loop to scroll up 7 then reset to 0, then loop again. If we were in another graphics mode, we would look at the chart to see how many pixels per mode line are being used. You then can scroll from 0 to one less than this number (for example, 0 to 7 is 8, the number of pixels in a mode 0 line). For GR.2, we scroll 0 to 15; for GR.8 you can't fine scroll(a coarse scroll here is the same as fine, one pixel line up).

A Message from our sponsor...Me

The chart doesn't give values for every mode that the Atari has. If you look at the row marked DL#, you will notice missing #'s 3,4,5,12,& 14. Until we do a special tutorial on these modes, you can use the Operating System manuals that Atari sells to explore these modes. Also missing are Graphics modes 9,10 & 11 which were not in U.S. machines when this was written. These modes use #15 (GR. 8) with a few Pokes into the operating system to change the way the data is interpreted. A Tutorial on these with many examples will be out in mid 1982.

EXAMPLE 11

Same thing as Ex.10, but using the joystick so you can practice moving up or down a pixel at a time. This will make sure everyone understands the diference between fine and coarse scrolling.

EXAMPLE 12
HORZ. FINE SCROLLING

There is also a horzizontal fine scroll register similar to the one for vertical fine scrolling. It is at 54276, right next to the other. To use it you again look at the chart, POKE the value into each line we want to scroll in the DL. For example, the correct value for a GR. 0 line that allows a horizontal scroll is 18). Now POKE the amount of clock cycles to scroll, 0 to 15 into 54276. No, I am not going to explain "clock cycles". It's exact meaning isn't important here and besides, like so many of the numbers we have been discussing, you will learn more if you try it yourself. It would be the exceptional person who could write a program to scroll without first trying it. Paper descriptions are not easy to read. Practice.....

EXAMPLE 13
Both Types Combined

We left a few goofs in this example. The LMS byte was not changed per the chart from 66 to 98. Also, a jump occurs on the screen. The method is just a combination of the two previous vert. scroll examples. After the great instruction you have been getting, you can figure this one out in a few minutes...CAN'T YOU ???

EXAMPLE 14

This is the best we could do to combine fine and coarse vertical scrolling. There are no new tricks to learn, just look at the sequence of how we put two previous examples together. Especially notice lines 190 and 300. This is pretty nice for Basic! This example could be renumbered into a subroutine for you to use in programs or..... Since there are so many possible combinations of scrolling, we don't show you each combination. It is easy to combine whichever types of scrolling you need. Really!

EXAMPLE 15

In the Basic language only example above, when vert. scrolling, the screen seems to blink as it scrolls. This is because we are moving a line up 1,2,...7 pixels. Then we move it back to 0 and as quickly as possible coarse scroll it up 8 pixels. This gets the text to move up or down but the "jump" shows on the screen. To hide the jump, the screen is Poked off(559,0) before the jump and Poked on(559,34) after. Thus, the blink.

NOW WE GIVE YOU...ASSEMBLY LANGUAGE.

Geoff Caras, of our little group, wrote a small assembly subroutine to do the pokes and you can see how quick and smooth it works. I won't feel bad if you use this example instead of my # 14 in your programs.

EXAMPLE 16

This is an example of changing graphics modes by the use of the chart. The titles could be brought in from data statements or disk if you wanted. These examples now POKE the border color to match the page (you really should have one of our Master Memory Maps to learn these pokes). The examples are really starting to be professional looking.

EXAMPLE 17

Another example of scrolling a graphics line up and down, but this time with the assembly subroutine. Think of the games you can write, or great educational programs! Don't fall into the trap of thinking that this is just a simple example, and a "real" example will be harder. All you have to change is the graphics statements!

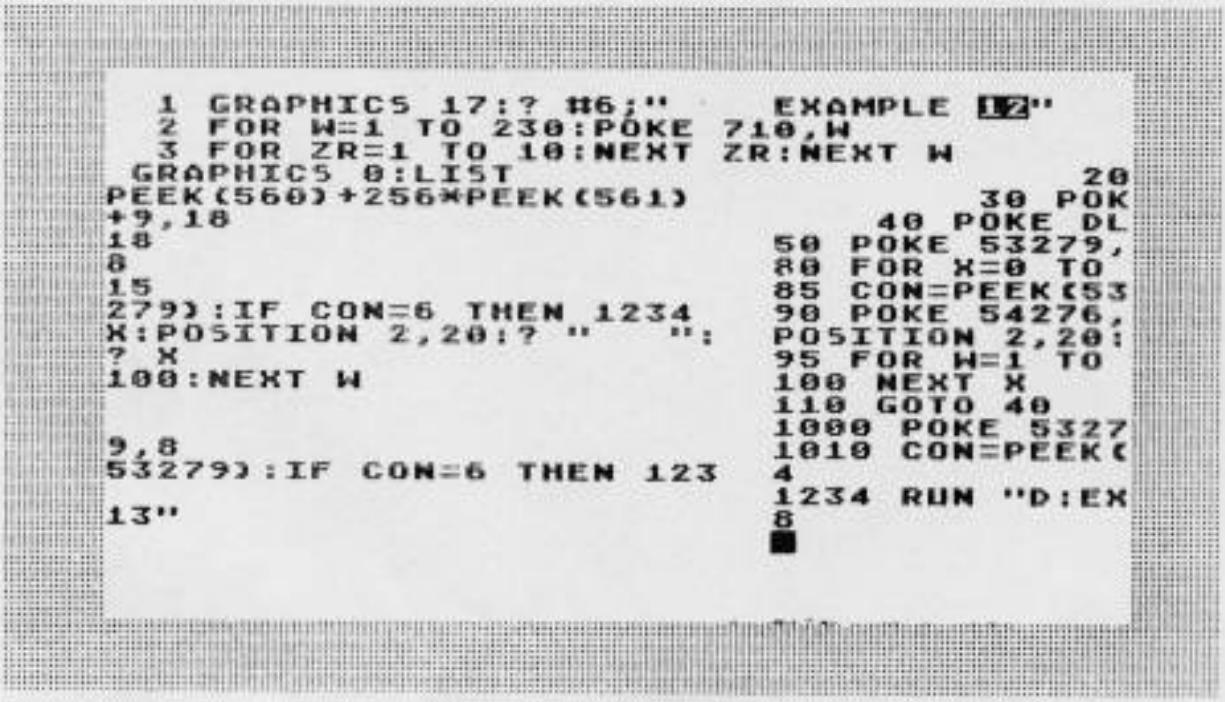
EXAMPLE 18

No, we aren't going to offer an assembly program for horz. scrolling. This is a tutorial to learn the principals of scrolling.

This example, our last, is again combining two previous examples to allow continuous scrolling horizontally. The poke to turn off the screen is still needed, so that is the blink you see. However, the junk on the screen is because of a point we didn't mention yet. When ever you poke a Display List, you should wait for the time when the TV is blank and waiting to do the next screen. Well, this happends 30 times a second! Basic can't go that fast, but if you program in assembly use WSYNC to prevent the flashing junk that this Basic example shows. For the rest of us, we'll wait for some more subroutines like example 15.

There are many details left out of this discussion. For example try poking 559,33 and 559,35. This gives you wide or small playfields of 48 or 32 characters width automatically, without the trouble we went through earlier for the 256 width. These playfields can be of use with scrolling. Plus.....well, see you in the next TRICKY TUTORIAL....boy I'm tired.. I think I'll go to bed.....

bye.




```
1 GRAPHICS 17:? #6;"    EXAMPLE 12"
2 FOR W=1 TO 230:POKE 710,W
3 FOR ZR=1 TO 10:NEXT ZR:NEXT W
10 GRAPHICS 0:LIST
20 DL=PEEK(560)+256*PEEK(561)
30 POKE DL+9,18
40 POKE DL+11,18
50 POKE 53279,8
80 FOR X=0 TO 15
85 CON=PEEK(53279):IF CON=6 THEN 1234
90 POKE 54276,X:POSITION 2,20:? "   ":POSITION 2,20:? X
95 FOR W=1 TO 100:NEXT W
100 NEXT X
110 GOTO 40
1000 POKE 53279,8
1010 CON=PEEK(53279):IF CON=6 THEN 1234
1234 RUN "D:EX13"
```

```
1 GRAPHICS 17:? #6;"    EXAMPLE 2A"
2 FOR W=1 TO 230:POKE 710,W
3 FOR ZR=1 TO 10:NEXT ZR:NEXT W
10 GRAPHICS 0
20 LIST
30 DL=PEEK(560)+256*PEEK(561)
40 DL4=DL+4:DL5=DL+5
50 NUML=PEEK(DL+4)
60 NUMH=PEEK(DL+5)
65 POKE 53279,8
70 ST=STICK(0)
80 IF ST=14 OR ST=10 OR ST=6 THEN NUML=NUML+40
90 IF ST=13 OR ST=9 OR ST=5 THEN NUML=NUML-40
92 IF ST=6 OR ST=7 OR ST=5 THEN NUML=NUML-1
94 IF ST=10 OR ST=11 OR ST=9 THEN NUML=NUML+1
95 CON=PEEK(53279):IF CON=6 THEN 1234
100 IF NUML<0 THEN 160
110 IF NUML<256 THEN 175
120 NUML=NUML-256
130 NUMH=NUMH+1
150 GOTO 175
160 NUML=NUML+256
170 NUMH=NUMH-1
175 IF NUMH<0 THEN 70
177 IF NUMH>255 THEN 70
180 POKE DL4,NUML:POKE DL5,NUMH
190 GOTO 70
1000 POKE 53279,8
1010 CON=PEEK(53279):IF CON=6 THEN 1234
1234 POKE 764,12:RUN "D:EX3"
```

```
1 GRAPHICS 17:? #6;"    EXAMPLE 10"
2 FOR W=1 TO 230:POKE 710,W
3 FOR ZR=1 TO 10:NEXT ZR:NEXT W
10 GRAPHICS 0:LIST
20 DL=PEEK(560)+256*PEEK(561)
25 FOR S=9 TO 13
30 POKE DL+S,34
40 NEXT S
45 POKE 53279,8
50 FOR Y=0 TO 7
60 POKE 54277,Y:POSITION 2,20:? "THE NUMBER OF PIXELS SCROLLED IS ";Y:FOR W=1 TO 50:NEXT W
70 NEXT Y
100 CON=PEEK(53279):IF CON=6 THEN 1234
110 GOTO 50
1234 RUN "D:EX11"
```