KEY COLOR SOFTWARE'S

KEY-264K

USER'S GUIDE

KEY COLOR SOFTWARE
P.O. BOX 360
HARVARD, MA. 01451

COPYRIGHT (C) 1983 BY NEIL CAPELESS
KEY COLOR SOFTWARE
P.O. BOX 360
HARVARD, MA. 01451

WARRANTY INFORMATION

THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT ANY WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED.

KEY COLOR SOFTWARE shall have no liability or responsibility to customer or any other person or entity with respect to any liability, damage, or loss caused by, or alleged to be caused, either directly or indirectly, by software sold by KEY COLOR SOFTWARE, including but not limited to any loss of business or profits, interruption of service, or consequential damages resulting from the use of such software.

SOFTWARE LICENSE

KEY COLOR SOFTWARE grants to CUSTOMER a license to use the software in accordance with the following terms:

1. CUSTOMER uses the software only on the single computer for which the software was purchased.

2. CUSTOMER shall not reproduce copies of the software.

3. CUSTOMER may transfer this license and the software to a third party provided that third party agrees to all terms of this agreement, and that CUSTOMER purchased one copy of this software for each one transferred.

THE KEY-264K

CONTENTS

THE KEY-264K

CONTENTS

INTRODUCTION

Congratulations on your purchase of the KEY-264K. The KEY-264K is a powerful software utility developed by KEY COLOR SOFTWARE that will unlock the hidden potential of your COLOR COMPUTER, allowing you to utilitize your computer in ways that no comparable computer can match.

The KEY-264K allows users of standard 32K TRS-80 COLOR COMPUTERS to utilize the full 64K ram memory from BASIC. AND ABSOLUTELY NO HARDWARE MODIFICATIONS ARE REQUIRED!!

The KEY-264K requires EXTENDED or DISK BASIC and GOOD 64K MEMORY CHIPS. Piggy back 32K or systems with half good 64K memory chips (rumored to be in earlier 32K versions) WILL NOT WORK!!! The KEY-264K does not require the Frank Hogg modification, necessary on earlier models to access 64K, and will operate with or without it. The KEY-264K will work on 32K systems with "E", "F", or even modified "D" boards.

The KEY-264K functions by switching two 32K memory banks (or sides) of the available 64K, in and out of the BASIC memory space. The software occupies the upper 3225 bytes of each bank and manages all bank switching and inter-bank communications. The KEY-264K also includes powerful new GRAPHICS VIEWING, BLOCK MEMORY MOVE, and FOREGROUND/BACKGROUND MULTI-TASKING commands thru extensions to the BASIC interpreter.

The KEY-264K expands the BASIC command set by implementing the following 15 additional statements and 1 additional function into either EXTENDED or DISK BASIC:

CMCOPY, CPOKE, CRUN, DUP, LCOPY,
MCOPY, MPAUSE, MTON, PULL, PUSH,
SCOLOR, SHIFT/CLEAR(BACKSLASH),
SWITCH, VIEW, WAIT, and CPEEK.

The KEY-264K also includes 8 keyboard commands, independent of BASIC, to allow switching sides, multi-tasking, break, reset, cold starts, and duplicating sides, all with simple keystrokes from the keyboard.

HOW IT WORKS

This section contains a brief technical overview of the COLOR COMPUTER'S addressing scheme, and the techniques utilized by the KEY-264K to access 64K. So if you are not technically inclined or simply don't care how it works, then skip this section, go on to the KEYBOARD LEARNING section and start using the KEY-264K.

The following is in no way intended to fully describe the inner workings of the COLOR COMPUTER. For a more complete description, please refer to the COLOR COMPUTER TECHNICAL REFERENCE MANUAL (#26-3193) available from Radio Shack, or the MOTOROLA MICROPROCESSORS DATA MANUAL available from Motorola.

The standard 32K EXTENDED BASIC COLOR COMPUTER contains 5 major functional blocks:

* The 6809E Central Processing Unit (CPU).
* 2 - 8K by 8 bit Read Only Memorys (ROM) containing both COLOR BASIC and EXTENDED BASIC.
* The 6847 Video Display Generator (VDG).
* 2 - 6821 Peripheral Interface Adapters (PIA).
* 64K of read/write memory in the form of 8 64K by 1 bit dynamic Random Access Memory (RAM) chips.

Tying these functional blocks together is the 6883 Synchronous Address Multiplexer, or SAM. The job of the 6883 SAM is to synchronize and control all timing among these functional blocks, provide transparent refresh to the dynamic RAM, and video display (VDG), and provide "device select" signals to select either the RAMs, ROMs, PIAs, or the external cartridge slot, depending on the 16 bit address presented to it by the 6809E CPU. It is important to note that the 6809E CPU presents the desired 16 bit memory address to the SAM, and the SAM determines what memory device to select and what address to actually use.

The 6883 SAM also provides other important functions such as controlling the location of the video page in RAM (VDG offset) and how it is presented to the 6847 VDG (MODE), CPU clock speed (vitamin E poke), and RAM memory size (4, 16, or 64K). But the two most important features to users of 64K are the "MAP TYPE" control and the "PAGE" switch.

HOW IT WORKS

The "MAP TYPE" control allows the SAM to reconfigure the memory map layout of the COLOR COMPUTER. Normally in "MAP TYPE" "0", the upper 32K memory is reserved for built in and cartridge ROMs, and the lower 32K addresses one half of the 64K read/write RAM memory. Setting the "MAP TYPE" control to "1" (writing to location FFDF hex) will cause the SAM to turn off, or disable, the COLOR COMPUTER ROMs and allow the 6809E CPU access to the full 64K read/write RAM memory. This is how the FLEX operating system and most other programs that access 64K work. Writing to location FFDE hex will reset the "MAP TYPE" to "0".

The "PAGE" switch can also be used to reconfigure the memory map but in a different way. Useful only in "MAP TYPE" "0", the "PAGE" switch allows the 64K RAM to be treated as two 32K banks (upper and lower) of read/write RAM. The "PAGE" switch is actually the high order bit of the 16 address bits used by the SAM to address the read/write RAM.

For example, if the 6809E CPU were to access address 0000 and the "PAGE" switch was set to "0" (write to location FFD4 hex), then the SAM would access the first location of the lower 32K bank (location 0000) of the read/write RAM. However if the "PAGE" switch were set to "1" (write to location FFD5 hex) then the SAM would access the first location of the upper 32K bank (location 8000 hex of the 64K) of the read/write RAM.

As far as the memory map is concerned, the upper 32K is still reserved for built in and cartridge ROMs, while the lower 32K is switched or "PAGED" between the upper and lower 32K banks of the 64K read/write RAM, depending on the setting of the "PAGE" switch. In effect this allows the COLOR COMPUTER to access a total of 80K with EXTENDED BASIC and 88K with DISK BASIC and a theoretical maximum of 96K!!!

The <u>KEY-264K</u> makes use of this "PAGE" switch in it's commands to both pass information and to switch between banks, or sides. It also makes use of the 1/60th second interrupts (generated by the VDG) which are utilized by BASIC for the TIMER and SOUND commands and to time out the disk drive motors in DISK BASIC.

Upon loading, the <u>KEY-264K</u> causes both 32K banks to be initalized independently by BASIC, determines whether EXTENDED or DISK BASIC is present, and patches itself into BASIC accordingly, in both banks. The <u>KEY-264K</u> commands become part of the BASIC command set and can be used like any other BASIC command. The 1/60th second interrupts are used to check for special keyboard commands, and to automatically switch banks, or sides when in the foreground/background multi-tasking mode.

ABOUT THIS MANUAL

The remainder of this manual is divided into four sections:

KEYBOARD LEARNING GUIDE

BASIC LEARNING GUIDE

KEYBOARD REFERENCE SECTION

BASIC REFERENCE SECTION

The LEARNING GUIDES are designed to be used as an aid in understanding the features of the KEY-264K and are meant to be used from beginning to end, in a "sit down" session at the computer.

The KEYBOARD LEARNING GUIDE will describe the overall concepts of the KEY-264K and then take you step by step through each keyboard command, explaining it's purpose, how it works, and how to use it.

Likewise, the BASIC LEARNING GUIDE will explain each new BASIC command, what it does, and how to use it. Examples are used throughout the guides.

The REFERENCE SECTIONS should be used as you become more familiar with the operation of the KEY-264K and need to occasionally refresh your memory about how a particular feature works.

The KEYBOARD REFERENCE SECTION will give a brief description of the function of each keyboard command.

The BASIC REFERENCE SECTION lists each BASIC command in alphabetical order on a separate page, showing the command syntax, purpose, and any applicable comments.

This manual is not intended to teach you BASIC or the operation of your COLOR COMPUTER. Previous knowledge of either EXTENDED or DISK BASIC and experience in the use of your computer is required.

KEYBOARD LEARNING GUIDE

1.0 Loading the KEY-264K cassette

Follow these 4 simple steps to load and start the KEY-264K:

1. If you have a DISK system, make sure the disk controller is plugged into the cartridge slot befor applying power. Turn on power to your COLOR COMPUTER.

2. IF you have a non Radio Shack printer connected to your system that does not provide an automatic line feed with each carriage return, then type "POKE 155,0". This will cause the KEY-264K to include an automatic line feed/carriage return routine for the printer. Note that this must be done before loading the KEY-264K.

3. Place the KEY-264K cassette in your cassette recorder and make sure it is fully rewound.

4. Type CLOADM and depress the play button on your cassette recorder. The screen will clear and the familiar "S" in the upper left hand corner will appear. In 5-10 seconds the "S" will change to "F" and the KEY-264K filename will appear, to indicate that the program is loading. After 10-20 seconds, the screen will begin to display the KEY-264K COPYRIGHT NOTICE by "painting" the display from top to bottom. Once the COPYRIGHT NOTICE screen is complete, it will take an additional 20-30 seconds for the KEY-264K to finish loading. When the KEY-264K has completed loading, the screen will clear and the normal BASIC COPYRIGHT NOTICE (either EXTENDED or DISK) sign-on message will appear just as it does when you first power up your computer. The KEY-264K is now ready to go.

If a read error should occur or if the "RESET" switch is pushed during the loading process, the screen will clear, and an I/O error message will appear. There are 3 copies of the KEY-264K on each side of the cassette separated by the musical scale, so if you have trouble loading the program then try the following:

Power down, wait 20 seconds and power up again. Type AUDIO ON:MOTOR ON and hit [ENTER], then type CLOADM but do not hit [ENTER] yet. Listen for the musical scale (do, re, mi....) that will follow each program. The scale will go up and then go down again (....mi, re, do). At the point after the "do" on the downward scale, you will be positioned about 10 seconds before the start of the next program copy, so hit [ENTER] and the loading process should begin.

The KEY-264K cassette is "auto loading". If it should become defective, it will be replaced if you send the original cassette along with your sales receipt and a check or money order for $5.95 ($3.95 + $2.00 shipping + handling) to:

KEY COLOR SOFTWARE
P.O. Box 360
HARVARD, MA. 01451

2.0 Getting started with keyboard commands

In order to familarize yourself with the operation of the KEY-264K you should first locate the following keys on the COLOR COMPUTER keyboard:

The DOWN ARROW key is the leftmost white key in the third row down from the top. From here on in the [↓] symbol will be used to represent the DOWN ARROW key.

The RIGHT ARROW key is the rightmost white key in the second row down from the top. From here on in the [→] symbol will be used to represent the RIGHT ARROW key.

The ENTER key is the long white key, second from the right in the third row down from the top. From here on in the [ENTER] symbol will be used to represent the ENTER key.

Also locate the gray [B], [C], [D], [E], [R], and [@] keys plus the "SPACE BAR".

3.0 Switching sides from the keyboard

At this point your screen should show the normal BASIC copyright notice sign-on screen for either EXTENDED or DISK BASIC, as if you had just turned the computer on. Type in the following:

```
PRINT 1/3
```

The computer responds:

```
.333333333
```

So what's the big deal? Read on!

Hold down the [ENTER] key and hit the [↓] key. You should have immediately noticed two things; first, the screen color changed from green to orange, and second, the command you just typed in has disappeared! This is because you have switched banks or sides. For ease of explanation, the two 32K memory banks will be referred to from now on as the "A" side (with the green display) and the "B" side (with the orange display). What you have just done by hitting the [ENTER] and [↓] keys was to switch from the "A" side to the "B" side.

Now type in the following:

PRINT 1/9

The computer responds:

.111111111

Hold down the [ENTER] key and hit the [→] key. You should now have a green screen, the PRINT 1/9 command is gone, and your original PRINT 1/3 command should have reappeared. You have just switched from the "B" side back to the "A" side. Try switching back and forth a few times to get the feel of it.

The KEY-264K uses the [↓] and [→] keys as control keys. They are used to initiate all keyboard commands. The [↓] is the control key for the "A" side, and will be used in conjunction with other keys to perform specific "A" side commands, such as [ENTER], for switching. Likewise, the [→] is the control key for the "B" side and must be used if you are performing keyboard commands in side "B". The [↓] and [→] keys were chosen because they normally do not echo on the screen, and usually have no effect on a BASIC program.

In normal operation, each 1/60th of a second the COLOR COMPUTER is interrupted by a signal from the Video Display Generator (VDG). For those of you unfamiliar with the term, an "interrupt" is an electrical signal that causes the normal processing of a computer to be interrupted, and control switched to a specific routine stored in memory to process a specialized function. At the conclusion of this specialized processing, control is returned to the point at which the computer was interrupted, and normal processing continues.

BASIC makes use of these 1/60th second interrupts to time the SOUND command and increment the TIMER, among other things. The KEY-264K also makes use of these interrupts to check the keyboard to see if a control key has been pressed. If in the "A" side when the interrupt occured, then a check is make for the [↓] key, and if in the "B" side then the [→] key is checked. If the proper control key for this side is being pressed then the KEY-264K will check the keyboard to see if any "command" keys (such as [ENTER]) are being used. If so, then the KEY-264K will execute the appropriate command, otherwise no action will be taken and normal processing will continue.

To demonstrate, try holding down the [ENTER], [↓], and [→] keys at the same time. The screen should flicker and show both the "A" (green) and "B" (orange) displays superimposed on each other. The computer is rapidly switching sides in time with the 1/60th second interrupts generated by the Video Display Generator (VDG).

Note that because of this dependency on the 1/60th second interrupts, there will be times when it appears that the control keys are not working. This is because whenever BASIC does any I/O functions such as reading or writing to cassette or disk, it first disables interrupts. This is necessary because of the exact timings required when interfacing with external devices, and the fact that these timings could be thrown off by interrupt processing. At the completion of the I/O processing, BASIC will re-enable the interrupts, and the control keys will again function normally.

Back to the keyboard switch command. So far you have learned how to switch back and forth between the "A" and "B" sides using the [↓], [→], and [ENTER] keys, but both sides have been at the BASIC command level. You can also switch sides even if the programs are running. Try the following:

Switch to the "B" side (orange screen, remember) if you are not already there, and type in the following program;

```
10 FOR X=1 TO 10000
20 PRINT "            SIDE B";X
30 NEXT X
```

Now type RUN and then quickly hit [→],[ENTER]. The program should have started printing to the screen, and then switched to the "A" side. Type the following program in the "A" side;

```
10 FOR Y=1 TO 10000
20 PRINT "SIDE A";Y
30 NEXT Y
```

Type "RUN" and then try switching back and forth between sides. Note that the programs are not both running at the same time. The "A" side program runs only when you switch to the "A" side, and the "B" side program runs only when you switch to the "B" side.

In summary then, to switch from side "A" (with the green screen) to side "B" (with the orange screen) you hold down both the [↓] and [ENTER] keys. To switch from side "B" to side "A" you hold down the [→] and [ENTER] keys. It makes no difference when switching whether a side is in the BASIC command mode, or in the process of running a BASIC program.

Note also that you could be displaying graphics in either or both sides and the KEY-264K will maintain the proper display as you switch back and forth between sides (There can be exceptions to this that will be discussed later in the BASIC LEARNING GUIDE).

4.0 Multi-tasking from the keyboard

The programs from above should still be running (if not, start them), so try holding down the [↓], [→], and [ENTER] keys at the same time. You should now see both program's displays superimposed on the screen, and both programs appear to be running at the same time, but at half speed. What is really happening is that the computer runs side "A" for 1/60th of a second, and then switches to side "B" when the 1/60th second interrupt occurs. It then runs side "B" for 1/60th of a second, switching back to side "A" at the next interrupt. This switching back and forth will continue for as long as the keys are held down, giving the appearance that both programs are running simultaneously, and for all practical purposes, they are.

This is basically how timesharing and multi-tasking are accomplished on larger machines (except with many more "sides" or users and a much more complex operating environment). This is also how Foreground/Background Multi-tasking works in the KEY-264K. (No, you're not going to have to sit there and hold down those keys in order to do multi-tasking! you will soon learn how to initiate it, so read on.)

While the KEY-264K may have succeeded in dividing the 64K RAM memory into two independent 32K sides, and found a way to run programs in both sides at the same time, it remains limited to only one video display (and only one keyboard, cassette, printer, disk etc.). Since allowing each side to use the video display during multi-tasking would result in a highly confusing and probably unreadable display (to say nothing of the headaches), the KEY-264K assigns the display to side "A".

The "B" side can continue to output text or graphics as it runs, but the output will not be displayed on the T.V. screen. Since you do not see any video output from the program operating in the "B" side, you can consider that program to be running in the "Background". If the "B" side is the "Background" side, then the "A" side which controls the video display, can be considered the "Foreground" side. Hence the term "Foreground/Background multi-tasking".

If the "A" and "B" side programs you entered earlier are still running, switch to the "B" side and hit the red [BREAK] key to "BREAK" the program, then switch to the "A" side and "BREAK" that program also. Now either EDIT or retype line 10 to read:

```
  10 FOR Y=1 TO 2000
```

Switch to side "B" and change it's line 10 to read:

```
  10 FOR X=1 TO 1500
```

Now type "RUN" to start the "B" side program running and then quickly hold down the SPACE BAR while at the same time pressing the [→] and [ENTER] keys.

You are now multi-tasking! Notice that control has switched over to the "A" or "Foreground" side with the green display screen. Type SOUND 100,10 and listen to the quaver in the tone. This is due to the rapid switching of sides that is occuring as the SOUND statement is being executed. This is a tell-tale sign that you are multi-tasking, so if you ever have doubts about whether you're multi-tasking or not, try the SOUND command.

Notice also that the duration of the generated sound is about twice as long as usual. This is due to the fact that the 1/60th second interrupt is used by BASIC to time the sound's duration, but instead of counting down on every interrupt as it normally does, BASIC will only count down on every other interrupt.

This is because if you are on the "A" side, and switch sides at each interrupt, then the next, and every other interrupt will occur on the "B" side, and will not be counted. This also causes the value of the TIMER on each side to be half of what it would normally be, if you are multi-tasking.

Try switching sides. As you can see, nothing happened! Since multi-tasking automatically switches sides every 1/60th of a second and since the "A" or "Foreground" side must keep control of the display, then manually switching sides cannot be allowed. The KEY-264K disables the ability to manually switch sides from the keyboard during multi-tasking. This is another way you can verify that multi-tasking is occuring.

Now type "RUN", sit back and watch the "A" side display go by. After a few minutes the display will suddenly switch from the "A" to the "B" side. Notice that the program in the "B" side has ended and that it's display that was hidden from view during multi-tasking is now on the screen. This is because the KEY-264K will automatically halt multi-tasking when the "B" or "Background" side program requests input from the keyboard.

This means that any INPUT or LINE INPUT statement executed in the "Background" program, or a return to BASIC'S command mode for any reason will cause an end to multi-tasking and trigger an automatic switch to the "B" or "Background" side. This is necessary because the "Background" side screen is not visible during multi-tasking and there would be no way to know that the program was either finished or requesting keyboard input. This could cause the "Background" program to "hang" waiting for input, or worse, accept keys typed in the "Foreground" side as either commands or data. Note that the INKEY$ command is an exception to the "stop multi-tasking" rule because it gets it's data from a look-ahead buffer and not directly from the keyboard. The INKEY$ command should be avoided if possible in programs that will be multi-tasked in the Background side.

If you now switch to the "A" side, you will see that the "A" side program is still in the "RUN" mode and has not been affected by the end of multi-tasking.

Let the "A" side program run to completion and then restore line 10 by typing:

```
 10 FOR Y=1 TO 10000
```

Switch to side "B" and change it's line 10 to read:

```
 10 FOR X=1 TO 10000
```

To summarize, To initiate multi-tasking from the keyboard, you must be in the "B" or "Background" side, and hold down the space bar, [→], and [ENTER] keys at the same time. Control will always switch to the "A" or foreground side while multi-tasking, and will automatically switch to the "B" side when the "B" side program is finished, or requests keyboard input, or returns to command mode for any reason.

Note that you cannot initiate multi-tasking from the keyboard if you are in the "A" or "Foreground" side. Also note that you cannot manually switch sides from the keyboard while you are multi-tasking.

You may note a delay in keyboard response and the occasional loss of a character or two while typing in the "Foreground" side if you are doing I/O in the "Background" side when multi-tasking. This is due to the fact that interrupts are shut off for the duration of the I/O command, causing a delay in switching from the "Background" to the "Foreground" side.

A note of caution is in order here. Since you only have one cassette, one printer, and one disk controller, any attempts to access the same device from both sides simultaneously during multi-tasking will cause unpredictable results.

5.0 Effect of the "BREAK" key while multi-tasking

You should still be in side "B", so type "RUN" and initiate multi-tasking. You should now be back in the "A" side. Try the SOUND command to insure that you are multi-tasking and then type RUN to get the "A" side program running. Now hold down the red [BREAK] key. You should have immediately stopped multi-tasking and switched to the "B" side.

What has happened is this; since you only have one keyboard, the programs on both sides sensed the BREAK key being down and executed a "BREAK", causing each to return to command mode, and as you learned earlier, a return to command mode in the "B" side will cause an end to multi-tasking and an automatic switch to the "B" side. Switch back to the "A" side and you will see that a "BREAK" has occured on that side also.

5.1 Special keyboard "BREAK" command

So what do you do if you are multi-tasking and want to "BREAK" one side without affecting the other side? Try this; you should still be in the "A" side if you were following the text, so type "RUN" and switch to the "B" side. Type "RUN" on this side, and then initiate multi-tasking. Back again on the "A" side, try switching sides (you shouldn't be able to) to make sure you are multi-tasking. Now hold down the [↓] and the gray [B] key.

The program should "BREAK". Notice that you did not switch to the "B" side, You have executed a "BREAK", but only in the "A" side. Verify that you are still multi-tasking with the SOUND command. Now try to continue the "A" side program with a "CONT" command. You more than likely got a "CN" (can't continue) or some other type of error (probably an "NF" next/for error in this case).

BASIC always executes it's "BREAK" (with the red [BREAK] key) between BASIC statements, making it possible to restart at the next statement with a "CONT" command from the keyboard.

With the KEY-264K, using the [↓] and [B] keys will cause a "BREAK" to occur at the next interrupt, and this will more than likely be in the middle of processing a BASIC statement, and not at the beginning.

Since the "CONT" statement will only work if you are restarting at the beginning of a statement, then unless you were lucky enough to "BREAK" between statements, you should get either the "CN" or some other type of error.

Although you can't always continue after this type of "BREAK", it will only "BREAK" one side and will also leave the display intact (as opposed to a "RESET"). Note that in many cases you could effect a restart by doing a "GOTO" the line number displayed at the "BREAK".

Type "RUN" to again start the "A" side program running, and then try switching sides to verify that you are multi-tasking. Now hold down the [→] and gray [B] key. Just as with the red [BREAK] key, you should have stopped multi-tasking and switched to the "B" side. If you now switch back to the "A" side you will see that unlike the "BREAK" with the red key, the "A" side is still in the "RUN" mode and did not execute a "BREAK". The "BREAK" occured on the "B" side only.

In summary, to execute a "BREAK" on both sides during multi-tasking, use the red [BREAK] key.

To execute a "BREAK" on only one side, use the corresponding control key for that side ([↓] for the "A" side, and [→] for the "B" side) in conjunction with the gray [B] key.

Note that you can still use this type of "BREAK" even if you are not multi-tasking (although use of the red [BREAK] key is preferred) as long as you use the proper control key for the side you are on.

Also remember that you can not always continue with a "CONT" command after a control key type "BREAK".

6.0 Effects of the "RESET" switch on the KEY-264K

Indiscriminate use of the "RESET" switch can have a disastrous effect on the operation of the KEY-264K. Pushing the "RESET" switch causes the COLOR COMPUTER to immediately stop what it is doing and transfer control to an initalization routine in BASIC. This initalization routine will always reset the computer to the "A" side regardless of what side it was in at the time of "RESET".

This means that if you are in side "A" a normal "RESET" will occur, but if you are in side "B" and you hit the "RESET" switch, you will be forced to side "A" and the KEY-264K will lose all information about where you were and what you were doing in side "B". A subsequent switch to side "B" under these circumstances will result in the COLOR COMPUTER going off on it's own, possibly wiping out part of the KEY-264K program itself, and probably causing the system to lock-up or "hang".

Also, remember that you are in side "B" one half of the time when you are multi-tasking, so avoid the temptation to reach for the "RESET" switch when something doesn't look quite right. In fact it might be a good idea to tape a thimble or a similar small plastic cap over the "RESET" switch, for as you will see, you will not be needing it.

A special note about "RESET". Some programs outside of BASIC will jump to the BASIC "RESET" vector (40999 dec. or A027 hex) when returning to BASIC. If this should occur from side "B", it will have the same disastrous effect as hitting the "RESET" switch while in side "B" as described earlier. If possible these programs should be modified to jump to 40974 dec. or A00E hex. This will still cause a BASIC "RESET" but it will remain in the current side and will not force a switch to side "A".

6.1 Special keyboard "RESET" command

Try the following; you should still be in side "A" so hold down the [↓] and the gray [R] key. You should have executed a "RESET" just as if you had hit the "RESET" switch. Do a "LIST" to insure your program is still there. Now switch to side "B" and you will see that the "RESET" has occured only on side "A".

While you are still in side "B" type "RUN" and then initiate multi-tasking. Back to side "A", verify that you are multi-tasking with a SOUND command. Type "RUN" to get the "A" side program running.

Now hold down the [→] and the gray [R] key. As with the "BREAK" command, you should have stopped multi-tasking and switched to the "B" side. Notice that a "RESET" has now occured on the "B" side, and the resultant return to command mode is what caused multi-tasking to end. Switch to the "A" side and you will see that the "A" side program is still in the "RUN" mode.

Switch back to the "B" side, type "RUN" and again initiate multi-tasking. Now do an "A" side keyboard "RESET". Notice that resetting the "A" side in this manner had no effect on multi-tasking. Try the SOUND command to verify that you are still multi-tasking.

In summary, NEVER USE THE "RESET" SWITCH AT THE BACK OF THE COMPUTER when using the KEY-264K. Instead use the control key corresponding to the side you are on (or the side you wish to "RESET" if you are multi-tasking), in conjunction with the gray [R] key. Note that resetting the "A" side has no effect on multi-tasking, while resetting the "B" side causes an end to multi-tasking and a switch to side "B".

7.0 Cold starts/Warm starts

For those of you who don't know, a "cold start" means to perform a complete initalization of a program or system (BASIC in our case) from the start, resetting all controls, pointers, etc. In the COLOR COMPUTER a "cold start" occurs when you first turn on power to the computer with the ON/OFF switch.

A "cold start" will also occur if memory location 113 dec. (71 hex) contains something other than a decimal 85 (55 hex), and either the "RESET" switch at the back of the computer is pushed, or an "EXEC 40999" (A027 hex) is executed. This is another reason why you should use the keyboard "RESET" instead of the "RESET" switch, since a "RESET" switch type of "cold start" will wipe out the KEY-264K program. Note that a keyboard "RESET" will not result in a "cold start" regardless of the contents of memory location 113 dec. (71 hex).

A "warm start" means to perform a partial initalization of a program or system, resetting some controls or pointers, but leaving others intact. Pushing the "RESET" switch or executing a keyboard "RESET" command results in a "warm start" of the COLOR COMPUTER.

7.1 Cold starts from the keyboard

As you may have guessed by now, you can perform a "cold start" from the keyboard. You should still be multi-tasking from the "keyboard RESET" exercises above (if not switch to side "B", type "RUN", then initiate multi-tasking) so try the following; hold down the [↓] and [C] keys. You have executed an "A" side "cold start", just like when you first turn on power to your system.

The BASIC copyright notice sign-on screen should have appeared for whichever level of BASIC you have, either EXTENDED or DISK. Now try the SOUND command and you will find that you are still multi-tasking. Cold starting the "A" side has had no effect on multi-tasking. If you do a list command, you will see that your "A" side program is no longer in memory, so type it back in as follows:

```
 10 FOR Y=1 TO 10000
 20 PRINT"SIDE A";Y
 30 NEXT Y
```

Now do a RUN command so both sides are running under multi-tasking. Hold down the [→] and [C] keys. You should have stopped multi-tasking and switched to side "B". You should see the BASIC copyright sign-on notice again. This time the "B" side has executed a "cold start", and the resultant return to command mode has caused multi-tasking to end, and a switch to side "B".

Now switch back to side "A" and you will see that the "A" side program is still in the "RUN" mode, and has been unaffected by the "B" side "cold start".

7.2 Special EXTENDED BASIC cold start for DISK users

Using the [↓],[C] or [→],[C] "colds starts" from section 7.1 above will always cause the corresponding side to "cold start" in whatever level of BASIC (EXTENDED or DISK) that you physically have in your computer. However, if you are lucky (or rich) enough to have DISK BASIC in your system, then it's possible for you to "cold start" either (or both) sides in EXTENDED BASIC without pulling your disk controller from the cartridge slot.

So for you DISK BASIC users (the rest of you can skip to section 7.3), you should still be in side "A" so try holding down the [↓] and [E] keys. You should have executed an "A" side "cold start" but this time in EXTENDED BASIC even though DISK BASIC is still physically present in your computer. Now switch to side "B" and notice the sign-on message. Side "B" is still in DISK BASIC.

Try a "DRIVE 0" command (used in DISK BASIC to set the default drive). It should process o.k. Now switch back to side "A" and try the "DRIVE 0" command on this side. Since side "A" is in EXTENDED BASIC and "DRIVE 0" is not a legal EXTENDED BASIC command, you should have gotten an "SN" (syntax) error.

Switch to side "B" again and hold down the [→] and [E] keys. You should have now "cold started" side "B" in EXTENDED BASIC. Now hold down the [→] and [C] keys. You just executed the regular "cold start" command which caused side "B" to "cold start" in DISK BASIC. As you can see, you can change a side from DISK BASIC to EXTENDED BASIC and back again by your choice of "cold start" commands.

Now "cold start" one side in DISK BASIC and the other side in EXTENDED BASIC, then type in the same "B" and "A" side programs from page 8 that you had typed in earlier:

```
SIDE "B" =  10 FOR X=1 TO 10000
            20 PRINT "           SIDE B";X
            30 NEXT X

SIDE "A" =  10 FOR Y=1 TO 10000
            20 PRINT "SIDE A";Y
            30 NEXT Y
```

Now switch to side "B", type RUN, and initiate multi-tasking. Back in side "A", type RUN to get the "A" side program running. You are now multi-tasking even though the program on one side is running in EXTENDED BASIC and the program on the other side is running in DISK BASIC.

7.3 Cold start the opposite side

This is a special keyboard command and is included mainly to try and recover when a side "hangs" or gets "locked up" for some reason. This command is disabled during multi-tasking, so hold down the red [BREAK] key to insure that multi-tasking is halted.

You should now be in side "B" (if you're not, switch there.), so hold down the [→] and [@] keys. You should have switched to side "A" and executed a "cold start". You have "cold started" the opposite side. Unlike the [←],[C] or [→],[C] "cold starts", a [←],[@] or [→],[@] opposite side "cold start" will "cold start" a side in the level of BASIC that it was already in at the time of the "cold start".

For example, if side "A" was in EXTENDED BASIC and you did an opposite side "cold start" from side "B", then you would switch to side "A" and do a "cold start" in EXTENDED BASIC. If in the above example, side "A" was instead in DISK BASIC, then it would have been "cold started" in DISK BASIC by the opposite side "cold start" command.

Try this; switch to side "B", uncover and press the "RESET" switch at the back of the computer (you did cover it earlier, didn't you?). You should have been forced into side "A", and now side "B" is no longer usable (see section 6.0 - page 13).

Now hold down the [←] and [@] keys. You have caused side "B" to be "cold started" from side "A" and you are no longer in trouble. Although you will lose whatever you were doing on side "B" due to the "cold start", at least side "B" is now usable and the <u>KEY-264K</u> can again function normally.

To summarize, you can "cold start" a side from the keyboard by holding down the control key for the side you wish to "cold start" ([←] for side "A", and [→] for side "B") along with the [C] key to "cold start" in whatever level of BASIC (EXTENDED or DISK) is physically in your computer, or with the [E] key to "cold start" in EXTENDED BASIC, even if your computer physically has DISK BASIC present. Note that these "cold starts" can be performed at any time, even if you are multi-tasking.

To "cold start" the opposite side, you hold down the control key corresponding to the side you are on, along with the [@] key, and the opposite side will be "cold started" in whatever level of BASIC it was in prior to it's "cold start". Note that this opposite side "cold start", unlike the others, is disabled during multi-tasking.

8.0 Keyboard duplicate command

This is a special command that can be used when debugging a BASIC program, or when you want to make a temporary backup of a BASIC program, or if you are playing a graphics game in BASIC and have to leave the computer for some reason. You would like to come back later and pick up where you left off, but don't want to hit "BREAK" because you could lose your graphics display.

Both sides should be recently "cold started" from the exercises above, so get into the "A" side and key in the following program:

```
 10 FOR Y=1 TO 500
 20 PRINT Y
 30 NEXT Y
```

Switch over to side "B" and do a "LIST" command to make sure there is no program on this side. Now switch back to side "A" and type "RUN". The program should start printing the increasing values of variable "Y" on the screen. When it gets to about 250, hold down the [↓] and [D] keys.

You should notice a pause as the KEY-264K copies or duplicates side "A" to side "B". Let the "A" side program run to completion. You should now be in command mode on side "A".

Switch to side "B" ([↓],[ENTER] remember?). You should have immediately picked up and started running at exactly the point you were at when you hit the [↓],[D] keys (around 250 on the screen).

As you can see, the duplicate command did more than simply copy your program from the "A" side to the "B" side. It also copied your BASIC variables and other status information such as where you were in the program. It also placed the "B" side in the RUN mode, exactly the same way the "A" side was.

You of course can also use the duplicate command to duplicate side "B" to side "A" by holding down the [→] and [D] keys while in the "B" side.

In summary, to make a duplicate, or copy of one side onto the other side, hold down the control key for the side you are on ([↓] for side "A", [→] for side "B") along with the [D] key. The duplicated side will be an exact copy of the original (except for the side it is on) in all respects, as it was at the time the duplicate command was executed.

DISK BASIC users note, the duplicate command will even duplicate the level of BASIC you are in. That means that if you have one side in EXTENDED BASIC and the other side in DISK BASIC, and execute the duplicate command, then the duplicated side will end up in whichever level of BASIC the original side was in.

Note that since the duplicate command also copies memory, any machine language program or other data residing in memory on the original side will automatically be copied to the duplicate side.

Also note that the duplicate command is disabled during multi-tasking.

9.0 Summing-up

You should now be somewhat familiar with the operation of the KEY-264K and what each of the keyboard commands do. Try different things. Load different programs in each side and switch back and forth as they run. Multi-task them. Load a program that uses graphics on one side, and one that uses text on the other. Switch back and forth. Use the duplicate command. Multi-task programs that create sounds or music. Try a PLAY command on both sides as you are multi-tasking. Try LLISTing a long program listing in the "Background" side during multi-tasking (don't forget to POKE the proper baud rate in the side you are printing from) as you type from the keyboard, or run another program in the "Foreground" side. Try the "BREAKs", "RESETs" and "COLD STARTS". Use you imagination, experiment!!! If you're not completly sure about how a command works, go back and review that section of the LEARNING GUIDE. Go through the whole LEARNING GUIDE again if necessary. Become familiar with the keyboard commands and the BASIC commands that follow in the BASIC LEARNING GUIDE and you will have computing power that no comparable computer can come close to matching!!

BASIC LEARNING GUIDE

1.0 Getting started with BASIC commandss

By now you should have completed the "Keyboard learning Guide" section of this manual and are familiar with some of the concepts and capabilities of the KEY-264K such as the "A" and "B" sides, switching sides, multi-tasking, and "cold starting" (if you're not, you better go back and re-read it). So now you're going to learn how to use these concepts and capabilities from within BASIC.

Note that in the following examples whatever you type into the computer will be in UPPER CASE or caps, and the computer's response, also in caps, wil be underlined. For brevity, the normal basic "OK" prompt will not be shown.

2.0 The PUSH and PULL commands

First, "cold start" both sides from the keyboard, then get in side "A" and type the following:

```
X=12345678
```

then type:

```
PRINT X
 12345678
```

You have loaded the variable X on side "A" with the value 12345678. Now switch to side "B" and type:

```
PRINT X
 0
```

As you can see, the variable X on side "B" has never been loaded and is thus equal to zero. Type the following:

```
PULL X
```

and now try:

```
PRINT X
 12345678
```

The contents of the variable X on the opposite, or "A" side has been "PULLed" or copied to the variable X on the "B" side. Switch to side "A" and print the variable X and you will see it has not been affected by the PULL command.

Now while you're still in side "A" type the following:

```
X=11223344

PRINT X
 11223344

PUSH X
```

What you have done is to change the value of the variable X on the "A" side to 11223344 and "PUSHed" or copied it to the variable X on the "B", or opposite side. Switch to side "B" and type:

```
PRINT X
 11223344
```

As you can see, the value of variable X has been changed and is now equal to the value "PUSHed" to it from the variable X on side "A".

Still in side "B", type in the following program:

```
 10 FOR X=1 TO 10000
 20 PRINT X
 30 NEXT X
```

Type RUN to start the program running, then quickly hold down the space bar, [→], and [ENTER] keys to initiate multi-tasking. Back on side "A" try the SOUND command to make sure you are multi-tasking.

Now try the following:

```
PULL X
PRINT X
```

At this point, you will get a printout of what the "B" side's value of X was, at the instant you executed the PULL X command.

Remember that because you are multi-tasking, the "B" side program is continuing to run in the "Background" and increasing the value of X as it runs.

Try the PULL X command again, followed by a PRINT X command. You should have displayed a larger number this time because the "B" side value of X has increased since you typed the first PULL X command. Try the PULL X, PRINT X commands a few more times and watch the value of variable X increase as the "B" side or "Background" program continues to run.

Note that "PULLing" the variable from the "B" side has had no effect on the operation of the "B" side program.

Now type the following:

```
X=10001
PUSH X
```

You should have immediatly stopped multi-tasking and switched to the "B" side (if you didn't, try the PUSH X command again) What you have done is to "PUSH" the value 10001 from the variable X on the "A" side to the variable X on the "B" side, causing the "B" side FOR-NEXT loop, and the program, to end. The resultant return to command mode caused multi-tasking to end, with control switched to the "B" side.

Note that is some instances, when running programs on both sides (whether you're multi-tasking or not), it may appear that the PUSH command does not work. This can only occur if you are "PUSHing" a value into a variable at the same time the other side's program is also placing a value into that same variable. What happens is that the "PUSH" command actually works, but it's value is immediately replaced by whatever value the other side's program was loading into that same variable.

In summary, to copy, or PUSH a variable from the current side to the opposite side, you use the "PUSH" command. To copy, or PULL a variable from the opposite side to the current side, you use the "PULL" command. The "PUSHing" or "PULLing" of variables has no effect on the operation of the program in the opposite side (other than the effect that changing the variable can have), and can be used even when multi-tasking.

The "PUSH" and "PULL" command works with all types of variables (numeric, numeric subscripted, string, and string subscripted). Note however, that when "PUSHing" a string variable, the BASIC "garbage collect" routine (this is the routine BASIC uses to delete unused strings when it is low on string space) will never be called. This is because it would most likely disrupt the opposite side's processing.

Because "garbage collect" cannot be called, the KEY-264K attempts, when doing a string variable "PUSH", to re-use the opposite side variable's string space.

If the new length of the "PUSHed" string variable is less than or equal to it's present length, then the string variable space will be re-used. Otherwise, the usual BASIC method of discarding the current string variable (resulting in an unused "garbage" string) and setting up a new string variable will be used.

Note that "PULLing" a string variable does not have this restriction, so normal string processing will occur when using the "PULL" command. Because "garbage collect" may be called when "PULLing" a string variable, this command is preferred over the "PUSH" command when working with strings.

Variable types can be intermixed, and more than one variable may be used in a "PUSH" or "PULL" command line as long as they are separated by a comma (,). The only restriction on a "PUSH" or "PULL" command is that the total length of the command line cannot exceed 128 characters. If the "PUSH" or "PULL" command line does exceed 128 characters you will get a "SL" (Statement too Long) error. The following are examples of legal "PUSH" and "PULL" commands:

```
  PUSH X,A$,G1(1),TR$(3,2),Y,HI$,NU(1,1,1)

  PULL NAME$(3),T1,LAST,L$,W(2,4)

  PUSH X1,Y$,A(1,2):PULL Z,B$,HI$(3,4),L1
```

Note that you can also use the "PUSH" and "PULL" commands to allow variables to be passed between programs running sequentially on the same side. First "PUSH" to the opposite side any variables you want passed on at the completion of one program. Then load the next program and "PULL" those same variables at the beginning of that program.

3.0 The CPEEK function and CPOKE command

The "CPEEK" or cross-peek function is exactly like the normal BASIC "PEEK" function, except that the memory location "PEEKed" will be on the opposite side.

Likewise, the "CPOKE" or cross-poke command is exactly like the normal BASIC "POKE" command, except that the memory location "POKEd" will be on the opposite side.

You should still be in the "B" side if you were following the text, so try the following:

```
POKE 29700,88
CPOKE 29700,44
```

What you have done is to "POKE" the value 88 into memory location 29700 (7404 hex) on the current, or "B" side, followed by a "CPOKE" or cross-poke to "POKE" the value 44 into memory location 29700 (7404 hex) on the opposite, or "A" side. To verify that it worked, type the following:

```
PRINT PEEK(29700)
 88 (should be the "B" side value)

PRINT CPEEK(29700)
 44 (should be the "A" side value)
```

Now switch to the "A" side, and type:

```
PRINT PEEK(29700)
 44
```

As you can see the "A" side value of memory location 29700 (7404 hex) is indeed 44. Now type:

```
PRINT CPEEK(29700)
 88
```

A look at memory location 29700 (7404 hex) on the opposite side (which is now side "B") shows the correct value of 88.

In summary, you can PEEK at memory locations on the opposite side by using the "CPEEK" function, and POKE values into memory locations on the opposite side through the use of the "CPOKE" command. In all other respects these statements will function exactly the same as their BASIC counterparts "PEEK" and "POKE".

Note that "CPEEK" and "CPOKE" can be used at any time, even when you are multi-tasking.

Also note that since the KEY-264K functions only with the lower two 32K memory banks, or sides, any "CPEEK" or "CPOKE" references to memory locations above 32767 (32K) will give the same results as if a regular BASIC "PEEK" or "POKE" command had been used (Remember there are no banks, or sides, in the upper 32K of memory).

4.0 The SWITCH command.

Now that the KEY-264K has opened up an additional 32K of memory for your use, and provided commands which allow you to communicate between sides, it only makes sense to also include a command to let you switch sides from within a BASIC program.

You should presently be in side "A" (green screen) so type:

```
SWITCH
```

You should have immediately switched to side "B" (orange screen) just as if you had executed the keyboard switch command ([↓],[ENTER]).

While you are in side "B", key in the following short program:

```
 10 FOR X=1 TO 400
 20 PRINT X
 30 NEXT X
```

Now type RUN and then initiate multi-tasking from the keyboard by holding down the space bar, [→], and [ENTER] keys.

Back in side "A", try a SOUND command to make sure you are multi-tasking, then type:

```
SWITCH
?MU ERROR
```

Just as with the keyboard switch command, switching sides cannot be allowed during multi-tasking. The KEY-264K actually disables the keyboard switch feature during multi-tasking, but since it cannot simply ignore a BASIC statement, it forces an "MU" (MUlti-tasking) error condition to occur.

"MU" (MUlti-tasking) errors occur whenever you attempt to use a BASIC command that would have a detrimental effect on multi-tasking .

Now either hit the red [BREAK] key, or let the "B" side program run to completion, to end multi-tasking and switch to side "B". Add the following line to your "B" side program:

```
 25 IF INT(X/8)=X/8 THEN SWITCH
```

This will cause the SWITCH statement to be executed for values of X that are multiples of 8 (8,16,24, etc.).

Now LIST your "B" side program and it should look as follows:

```
 10 FOR X=1 TO 400
 20 PRINT X
 25 IF INT(X/8)=X/8 THEN SWITCH
 30 NEXT X
```

Type RUN and the screen should quickly display the numbers 1 to 8 and you should then automatically switch to the opposite, or "A" side.

Now in the "A" side, type in the following program:

```
 10 FOR Y=1 TO 400
 20 PRINT Y
 30 IF INT(Y/8)=Y/8 THEN SWITCH
 40 NEXT Y
```

Type RUN and watch as the programs switch back and forth between side as they run (note that they are not multi-tasking), resulting in a confusing display. The "B" side program should end first, so when it does, type SWITCH to get to side "A" and let the "A" side program finish.

4.1 The SWITCHL command

Although you are now able to switch sides with a BASIC statement, each time you do, the display changes both color and content. This will usually cause the screen to flash when changing color, and will require a carefully coordinated display on both sides to prevent the text (or graphics) from being unreadable.

Still in side "A", type the following:

```
SWITCHL
```

Notice that the display has not changed, but the flashing cursor has disappeared. What has happened is that the switch to side "B" has occurred, and you are actually in side "B", but the display for side "A" has been left on the screen. The "L" at the end of the SWITCH command means to "Leave" the current display on the screen when you switch sides. Carefully type:

```
SOUND 100,10
```

You will not see the SOUND command appear on the screen as you type it because you are actually in side "B", even though the side "A"' display is on the screen. You should however, hear the tone if you typed the SOUND command correctly (if you don't hear the tone, hit [ENTER] a couple of times and then carefully type the SOUND command again).

Now switch back to side "A" using the "B" side keyboard switch command ([→];[ENTER]). The cursor should have reappeared. Switch to side "B" again, also using the keyboard switch command ([↓],[ENTER]).

Notice that when you use the keyboard switch commands after a "SWITCHL" that the display will remain unchanged as you switch sides. This is beacuse the "B" side (in our example) display was disabled by the "A" side SWITCHL command and can be re-enabled only be a SWITCH command without an "L", a CRUN command (cross run - to be discussed later) without an "L", a view reset command (VIEWR - to be covered later) or when multi-tasking ends (note that although this command canot be used while multi-tasking, it is possible to execute a SWITCHL command and then initiate multi-tasking).

Once again switch to side "A" using the keyboard switch command and type SWITCH.

As you can see, the regular SWITCH command (without the "L") has caused the "B" side display to be restored and a normal switch has occurred. Using the keyboard switch commands, switch sides a few times to insure that both displays are now working properly.

Now switch to side "B" if you are not already there and change line 25 to include an "L" at the end of the SWITCH command as follow:

```
 25 IF INT(X/8)=X/8 THEN SWITCHL
```

Your program should now look like this:

```
 10 FOR X=1 TO 400
 20 PRINT X
 25 IF INT(X/8)=X/8 THEN SWITCHL
 30 NEXT X
```

Now type RUN and the screen should display the numbers 1 to 8 and then appear to "freeze". The flashing cursor will be missing from the screen.

What has happened is that the "B" side program started to run, displayed the numbers 1 through 8 and then executed the SWITCHL command. The SWITCHL command caused a switch to side "A" but left the side "B" display on the screen. You are actually in side "A" even though the side "B" display is on the screen, and since side "A" is in the BASIC command mode, no activity will take place until you type a command.

Carefully type RUN (note that the RUN command will not appear on the screen), and the "A" side program will begin, resulting in the programs on both sides switching back and forth as they run.

Notice that the switching of sides will not be visible to you as only the "B" side display will appear on the screen, repeatedly displaying the next 8 numbers, then pausing while the "A" side program runs.

Note that the regular SWITCH command (without the "L") in the "A" side program will restore both displays, but since you are switching to side "B" when it is executed, the display is unaffected.

Let the programs run to completion. The "B" side program which controls the display will finish first, so when it does, type SWITCH to let the "A" side program finish. Notice that both displays are now restored.

4.2 The SWITCH command with a line number

The SWITCH or SWITCHL command followed by a line number can be used to switch sides, place the opposite side in the "RUN" mode (but without clearing any variables as a RUN command would do), and start execution at the specified line number.

Switch back to side "B", type NEW, then key in the following program:

```
 10 PRINT 10
 20 PRINT 20
 30 PRINT 30
 40 PRINT 40
 50 PRINT 50
 60 SWITCH
```

Now switch back to side "A" and type SWITCH 10.

The "SWITCH 10" command has caused you to switch to side "B", placed the "B" side in the "RUN" mode (it was previously in command mode) and started execution at line number 10.

The program should have printed the numbers 10 through 50, corresponding to the line number of each line, and then switched back to side "A" at line 60. This probably happened much to fast for you to see, so switch back to side "B" with a regular SWITCH command and look at the display. You should see the numbers 10 through 50 displayed there.

Now while still in the "B" side, type:

```
A=87654321
```

switch back to side "A" and type "SWITCH 30".

You should have again switched to side "B" and placed it in the "RUN" mode, but this time you should have started execution at line number 30. Switch to side "B" and verify that only the numbers 30 through 50 are displayed there. Print out the variable A that you set to 87654321 and you will see that it has not been affected by the SWITCH command.

Again switch to side "A" and type "SWITCH 25".

This time you should have switched to side "B", placed it in the "RUN" mode, and attempted to execute line 25. Since there is no line 25 in your "B" side program, you should have gotten a "?UL ERROR" (undefined line number). Note that since you did not successfully start the "B" side program, line 60 with the SWITCH command was never executed, so you remain in side "B" after the "UL" error.

4.3 SWITCH command summary

In summary, the BASIC SWITCH command without a line number will cause you to switch to the opposite side and pick up processing at precisely the point at which you last executed a SWITCH command in that (opposite) side (This is equivalent to the keyboard SWITCH command, but without the dependency on the 1/60th second interrupts). This means that if you were in the "command" mode before switching sides, that you will remain in the "command" mode when you switch back. Likewise, if you were in the "run" mode prior to switching sides, then you will be in the "run" mode when you switch back to this side.

The BASIC SWITCH command followed by a line number will cause you to switch to the opposite side, put the opposite side in the "run" mode, and attempt to begin execution starting at the specified line number. If the program on the opposite side is already in the "run" mode, but at a different line number, processing will be immediately re-directed to the line number specified in the SWITCH command.

If the line number specified in the SWITCH command does not exist in the opposite side program, a switch to the opposite side will still take place, but a "UL" (undefined line number) error will then occur as you attempt to execute a non-existent line number.

Note that you can use the SWITCH command followed by a line number to "call" subroutines (similar in operation to a GOSUB/RETURN) that reside on the opposite side. Just switch to the first line number of the opposite side subroutine, and make sure that the subroutine ends with a regular SWITCH command (without a line number) to return control to the "calling" side. Variables can be passed between the main program and the subroutine by using the PUSH or PULL commands.

The SWITCHL command can be used in place of a regular SWITCH command whenever you want to switch sides and leave the current display on the screen. The SWITCHL command will disable the opposite side's display until restored by a regular SWITCH command with no "L", a CRUN command (cross run - to be covered later) with no "L", a VIEWR command (view reset - to be discussed later), or an end to multi-tasking. Note that switching sides from the keyboard will not have any effect on the display after the execution of a SWITCHL command.

On the subject of switching sides from the keyboard, a word of caution is in order. <u>You should never manually switch sides from the keyboard if you are running BASIC programs that contains SWITCH commands.</u> This is because you could change the processing sequence of the programs, or in some cases change the expected processing results of one or both side's programs.

For example, if you were running a program in side "A" that switched to a subroutine in side "B" and you were to manually switch back to side "A" before the "B" side's subroutine was completed, then the "A" side program could continue with incomplete or erroneous data, thinking that the "B" side subroutine had run to completion.

5.0 The CRUN (Cross RUN) command

The CRUN (Cross RUN) command functions like a combination of the SWITCH and RUN commands.

You should still be in the "B" side if you were following the text, so type LIST to make sure that the "B" side program is still there (if not, refer to section 4.2 and type it in again). Switch to side "A" and type CRUN.

You should have switched to the "B" side, started execution at the first line of the program, printed the numbers 10 through 50 and then switched back to side "A" at line 60. Manually switch to side "B" and verify that the numbers 10 through 50 are displayed there.

The CRUN command has functioned just like the SWITCH 10 command you tried earlier with this same program, but with two important differences. First, with no line number specified, it placed the opposite side in the "run" mode and began execution at the first line number in the program. Second, it initialized all variables on the opposite side before starting execution, just as a regular RUN command would have done.

Now while still in side "B" type:

```
A=12345678
```

switch back to side "A" and type CRUN 30.

You should have again switched to side "B" and initialized all variables, but this time started execution at line number 30. Switch to side "B" and verify that this has indeed happened. Print out the variable "A" that you set to 12345678 and you will see that it has now been initialized to zero.

Switch again to side "A" and this time type CRUN 35.

Like with the SWITCH command, you switched to the opposite side, put it in the "run" mode, and attempted execution at line 35. Since there is no line 35, you got a "UL" error (undefined line number).

In summary, the CRUN command functions much like the SWITCH command. A CRUN with no line number will switch to the opposite side, initialize variables, and begin execution at the first line number in the opposite side program. A CRUN followed by a line number will function as above, but will start execution at the line number specified. If the specified line number does not exist, the switch to the opposite side will still take place but a "UL" error (undefined line number) will then occur.

Also note that like the SWITCH command, you can also use an "L" at the end of the CRUN command to leave the current display on the screen. Also like the SWITCHL command, the CRUNL command will disable the opposite side's display until restored by a regular SWITCH, CRUN, VIEWR (view reset - to be discussed later) or an end to multi-tasking.

Attempting to execute a CRUN or CRUNL command while multi-tasking will result in an "MU" (MUlti-tasking) error.

6.0 The SCOLOR (Switch COLOR) command

As you learned in section 4.1 (The SWITCHL command), switching sides with the regular SWITCH command can cause flashing in the display as the screen changes colors. However, there may be times when you wish to use the text display on both sides as you switch back and forth between two programs, but you would like to do so without any distracting screen flashes.

Still in side "B" from the exercises above (if you're not, switch there!), type SCOLOR.

The screen color should have changed from ORANGE to GREEN! You are still in side "B" but you now have a green display just like side "A". Now manually switch sides from the keyboard a few times. Notice that there is no screen flashing as you switch sides. In fact, except for the content of the screen changing, you would never know that you were switching sides. There is one problem however, you can no longer tell which side you're on by looking at the screen.

Switch to side "B" if you're not already there. If you're not sure where you are, hold down the [↓] and [ENTER] keys. Now again type SCOLOR. This time the screen went from GREEN back to ORANGE.

Switch to side "A" and type SCOLOR. the "A" side display should now be orange also. Manually switch sides from the keyboard a few times. Both screens are now orange, and as before, there is no screen flashing as you switch sides. Now make sure that you are in side "A" by holding down the [→] and [ENTER] keys, and type SCOLOR to restore the normal "A" side green display.

In summary, the SCOLOR (Screen COLOR) command toggles the screen color from ORANGE to GREEN, or from GREEN to ORANGE. So if you want to switch sides and use the text displays on both sides but don't want the screen flashes as you switch sides, then do an SCOLOR command on one (either) side so that both side's text displays are the same color (GREEN or ORANGE, it doesn't matter). Note that this command can be used while multi-tasking.

7.0 The MTON (Multi-Tasking ON) command

You should still be in the "A" side so do a keyboard "cold start" and switch to side "B". Do a "cold start" on this side also, and then key in the following program:

```
 10 FOR X=1 TO 10000
 20 PRINT X
 30 NEXT X
```

Type RUN to get the "B" side program running and then manually switch to side "A". Then type "MTON" (Multi-Tasking ON). You are now multi-tasking. Now try manually switching sides. Nothing happened because as you learned earlier, the keyboard switch commands are disabled during multi-tasking. Try a SWITCH command and You should get a "MU" (MUlti-tasking) error. Try a SOUND command and notice the tell-tale quaver of multi-tasking in the generated tone. The "MTON" command works just like the keyboard multi-tasking command, but can be executed from either side.

As you have learned so far, when you are multi-tasking from the keyboard and the "B" or "background" side either requests input from the keyboard or returns to command mode, then multi-tasking will end and you will automatically switch to the "B" or "Background" side.

7.1 MTON command options

With the "MTON" command you have a total of 3 options on how to handle the end of multi-tasking:

1. A regular "MTON" command by itself will automatically switch to side "B" at the end of multi-tasking. (just like the keyboard multi-tasking command.)

2. An "MTON" command followed by an "@" ("MTON@") will, at the end of multi-tasking, give 5 short tones as an audio indicator that multi-tasking has ended, and remain in side "A" (no switching sides).

3. An "MTON" command followed by a period ("MTON.") will simply remain in side "A" at the end of multi-tasking. (no audio indicator and no switching sides)

Hit the red [BREAK] key to stop multi-tasking. Since you had typed a regular "MTON" command, you should have automatically switched to side "B". Type RUN to get the "B" side program running, and then switch back to side "A".

This time type "MTON@", and verify that you are multi-tasking with a SOUND command. Now again hit the red [BREAK] key to end multi-tasking. You should have heard 5 short tones as multi-tasking ended but this time you remained in side "A" instead of switching to side "B". Now switch to side "B" and you will see that a "BREAK" has occured and you are now in command mode. Type RUN again and then switch back to side "A".

Now try the "MTON." command, and again verify that you are multi-tasking. Once again hit the red [BREAK] key to end multi-tasking. This time, although multi-tasking has indeed ended, you remained in side "A" and received no indication that it had ended. Switch to side "B" and you will see that you are now in command mode and that a "BREAK" has occured.

In summary, you can initiate multi-tasking from either side by issuing an "MTON" command. Just as with the keyboard command, multi-tasking will begin, and control will switch to (or remain in) the "A" or "Foreground" side. You have 3 options on how to handle the end of multi-tasking; Switch to side "B" (with a regular "MTON"), or give an audio indicator of 5 short tones and remain in side "A" (with a "MTON@"), or simply remain in side "A" with no indication of any kind that multi-tasking has ended (with an "MTON."). Note that if you attempt to execute an "MTON" command when you are already multi-tasking, the command will be ignored, and will not change the previously set "end of multi-tasking" option.

8.0 The MTPAUSE command

This command was included so that you could perform commands that would otherwise be illegal or would just not work properly while multi-tasking, such as doing disk I/O from both sides, or using the "PLAY" or "SOUND" commands. The MTPAUSE command is used to provide only a temporary halt to multi-tasking and will normally be followed at some point by another MTON command. The MTPAUSE command will cause multi-tasking to stop, with control remaining in the same side that executed the MTPAUSE command. It will not affect the current screen display. Also, it will not automatically cause one of the "end of multi-tasking" options to occur unless the statement following the MTPAUSE requires keyboard input (remember that the INKEY$ function is not in this category), or causes a return to command mode.

You should still be in side "B" at this point so add the following line to your program:

```
 25 IF X=100 THEN MTPAUSE:PLAY "ABCDEFG":MTON
```

Your "B" side program should now look like this:

```
 10 FOR X=1 TO 10000
 20 PRINT X
 25 IF X=100 THEN MTPAUSE:PLAY "ABCDEFG":MTON
 30 NEXT X
```

Now type RUN to get your "B" side program started and quickly switch to side "A". Type MTON to start multi-tasking and sit back and wait a few moments and you will eventually hear the seven notes from the PLAY command. Notice that the tell-tale quaver of multi-tasking is missing from the tones. Also notice that although multi-tasking was stopped during the PLAY command, that the screen display was not affected. Hit the red [BREAK] key to end multi-tasking and switch you to side "B".

In summary, if you want to generate music, or access disk from programs in both sides, or need to speed up processing for a particular routine while you are multi-tasking, then temporarily suspend multi-tasking with an MTPAUSE command, and then turn multi-tasking back on when you are through (with an MTON command). Note that control remains in the side that executed the MTPAUSE command, and that the screen display will not be affected. Also note that the MTON command used to turn multi-tasking back on after the MTPAUSE must contain the desired "end of multi-tasking" option. If you execute a MTPAUSE command when you are not multi-tasking, the command will be ignored.

9.0 The VIEW command

The VIEW command can be used to control what appears on your T.V. screen. Normally a BASIC program can be in one of two display types, either TEXT or GRAPHICS. There is only one kind of TEXT display but a GRAPHICS display can be in one of five modes. The modes are selected through the use of the PMODE command (PMODE 0 through PMODE 4) and select varying degrees of resolution and color.

BASIC is usually in the TEXT mode and can be changed to GRAPHICS mode by using the SCREEN 1,N command where the first digit (1) indicates GRAPHICS mode and the second digit (N, where N=1 to 8) indicates which GRAPHICS page to display and is dependent on what PMODE you are in. (see GOING AHEAD WITH EXTENDED COLOR BASIC for more information on graphics modes) A SCREEN 0 command will return you to TEXT mode. You will also switch back to TEXT mode whenever you try to print something on the text screen, which is always the case when you return to command mode.

Now that you have established that a BASIC program can have either a TEXT or GRAPHICS display, and you know that the KEY-264K has two BASIC sides ("A" and "B"), let's try a few examples using the VIEW command.

You should still be in command mode in side "B" if you were following the text, so first type 25 to delete line 25 with the PLAY command, then type the following commands:

```
 PMODE 4,1
 PCLS
 CIRCLE (128,96),50
```

Although you did not see it take place, the above commands set side "B"'s graphics mode to the highest resolution (black and green) and set the starting graphics page to 1. Cleared the hi-res graphic pages to all black, and drew a large green circle right in the center of the screen.

Now switch to side "A" and type these commands:

```
 PMODE 4,1
 PCLS
 LINE (78,46)-(178,146),PSET,B
```

These commands performed the same functions as those on the "B" side except that instead of a green circle, a large green box was drawn. Now, still in side "A" type VIEWBT (VIEW the "B" side Text display). The text display from side "B" should now be on the screen. Notice however, that the flashing cursor is missing from the screen. This is because you are still in side "A" even though the side "B" text display is on the screen.

Carefully type (because it will not appear on the screen) VIEWBG (VIEW the "B" side Graphic display). You should now see the side "B" hi-res graphic screen with the green circle on a black background. When displaying graphics with the VIEW command, the then current PMODE and GRAPHICS PAGE settings for the side being displayed will be used.

Now carefully type VIEWAG (VIEW the "A" side Graphics display). The "A" side hi-res graphic display with the green box on the black background should now be on the screen.

By now you have the idea, so let's get back to the "A" side text display by typing VIEWAT (VIEW the "A" side Text display). You should now see all those VIEW commands that you have been typing, but not seeing.

But what if you want to be able to run your program on either side without having to know if you're on side "A" or side "B" ? Try this, type VIEWCG (VIEW the Current side Graphics display) and notice that the hi-res display with the green box on the black background is now on the screen. Since you are in side "A", you displayed the side "A" hi-res screen. No matter which side you are in, using the VIEWCG or VIEWCT (VIEW the Current side Text display) commands will display the graphics or text screen from that side.

Try a VIEWOG (VIEW the Opposite side Graphics display). You should now see the hi-res display with the green circle on the black background. Since you are still in side "A", you displayed the "B" or opposite side hi-res screen. Again, no matter which side you are in, the VIEWOG or VIEWOT (VIEW the Opposite side Text display) commands will display the graphics or text screens from the opposite side. Now type VIEWAT to get the text display back.

Now type SWITCH to get to side "B", then type RUN to get the "B" side program running. Switch to side "A" and type MTON to initiate multi-tasking and verify with a SOUND command. Type VIEWOT or VIEWBT to view the side "B" text screen. Now for the first time, you are able to see the "B" or "background" side as it is running during multi-tasking.

Try a VIEWOG or VIEWBG command and you should again see the side "B" hi-res graphic display with the green circle on the black background. As you can see, the view command is fully functional whether you are multi-tasking or not.

Now hit the red [BREAK] key to force an end to multi-tasking and an automatic switch to side "B" (you started multi-tasking with a regular MTON command, remember?). Type VIEWBT. Since you were already displaying the "B" side text screen, it appears as if nothing happened. Now manually switch to side "A". Notice that the "B" side display remained on the screen. The VIEW command will disable both side's displays prior to putting the selected display on the screen.

Disabling a display in this case means to prevent it from being put on the screen from a keyboard switch command. It also has the effect of not automatically returning from graphics mode to text mode when you attempt to print something to the text screen. You can now put whatever display you want on the screen and if you then switch sides with an SWITCHL, CRUNL or keyboard switch command, the display will remain on the screen. The displays will remain disabled after a VIEW command until restored by a regular SWITCH or CRUN command (without the "L"), an end to multi-tasking, or by the command we will discuss next, the VIEWR or view reset command.

9.1 The VIEWR command

The VIEWR (VIEW Reset) command will restore both displays and the screen will then display either the TEXT or GRAPHICS display (depending on what mode you are in) of the side that executed the VIEWR command.

Manually switch back to side "B" and type VIEWR to reset the displays. Now type RUN to get the "B" side program running and manually switch to side "A". Type MTON to get multi-tasking started and then try a VIEWR command. You should now see both the "A" and "B" side displays superimposed on the screen. This is because multi-tasking, when initiated will always disable the "B" or "Background" side's display, leaving the "A" or "Foreground" side's display intact so that you will only see the "A" side's display during multi-tasking. Since the VIEWR command restores both the "A" and "B" side displays, they will each be displayed for 1/60th of a second as multi-tasking switches sides at each interrupt. Type VIEWAT to get the "A" side text display back.

Now hit the red [BREAK] key to force multi-tasking to end with an automatic switch to side "B". Notice that the "B" side display is now on the screen. Try manually switching sides a few times. Notice that both side's displays appear as you switch sides. This is because the end of multi-tasking has restored both displays just as if you had executed a VIEWR command.

Using the VIEW (as well as the SWITCHL or CRUNL) commands can at times be very complicated and you may find yourself confused as to where you are and what display is on the screen. Whenever this occurs hit the red [BREAK] key and then do a VIEWR command. This should restore the KEY-264K displays to the normal operating mode.

9.2 The special user mode VIEW command

This is a special version of the VIEW command that will allow you to set up and use displays modes (and graphics pages) not normally available from BASIC such as the various semi-graphics modes. It requires a fairly high level technical understanding of how the different display modes work and how the 6883 (SAM) and 6847 (VDG) function together. So if you're interested, read on, otherwise skip to section 9.4 (VIEW command summary) and go on from there.

The KEY-264K has four memory locations set aside on both the "A" and "B" sides for special display modes. These memory locations should be used as follows:

Location 32592 (7F50 hex) - should contain the high order seven bits of the sixteen bit video display address in the form XXXXXXX0 where the low order bit is always zero. These seven bits will be set into 6883 (SAM) registers F6-F0 and will set the video page to any desired 1/2K boundary in the 64K memory.

Location 32593 (7F51 hex) - should contain the three mode control bits for determining how the video memory is addressed by the 6883 (SAM). These are in the form 00000XXX and will be set into 6883 (SAM) registers V2-V0.

Location 32594 (7F52 hex) - should contain the five control line bits that determine the various modes and color sets of the 6847 (VDG) in the form XXXXX000. This location (all eight bits) will be written to the I/O port at memory location 65314 (FF22 hex).

location 32595 (7F53 hex) - This location is used as a flag to indicate that the special display mode setup locations are to be used in determining the screen display. If this location contains something other than zeros, then the special display mode will be used. Note that the special VIEW command will set and reset this flag location automatically, so DO NOT SET this location unless you know what you are doing. (see discussion below in section 9.3 on using these memory locations with the SWITCH command)

Once you have set the desired values into locations 32592, 32593, and 32594 (7F50, 7F51, and 7F52 hex) you can then use the VIEW command by typing VIEWAU (VIEW the "A" side User display), VIEWBU (VIEW the "B" side User display), or use the other variations of the VIEW command, VIEWCU (VIEW the Current side User display) or VIEWOU (VIEW the Opposite side User display). Note that these commands can be used while multi-tasking.

Let's try an example, get into side "A" if you are not there already and type the following:

```
 PMODE 4,1
 POKE 32592,PEEK(186)+12
 POKE 32593,6
 POKE 32594,240
```

The PMODE command sets side "A" to the highest resolution mode with a starting graphics page of 1. The POKE commands set the 6883 (SAM) video page to the current page (which is in location 186) plus 3072 (which is 1/2 a hi-res page), set the 6883 (SAM) video addressing mode to 6 (hi-res graphics) and sets the 6847 control bits to hex F0 (240 decimal) for graphics mode G6R, or the highest resolution black and green graphics mode.

Type VIEWAG and you should see the green box on the black background that we set up earlier. Notice that the green box is in the center of the screen. Now type VIEWAU. You should see only the lower half of the green box and it is now at the top of the screen and there may be "garbage" displayed from the center of the screen down. What you have done is to set up the special VIEW display mode to equal the BASIC display mode except that the starting graphics page has been set to page 3 (that is why the garbage is there, pages 5-8 have not been PCLSed). You could not have done this through BASIC unless you had done a PCLEAR 8 first. Type VIEWR to get the text screen back and then type:

```
 POKE 32592,PEEK(186)+2
 VIEWAU
```

You should again see the green box on the black background, but this time the box is only slightly above the center of the screen and there is only a little "garbage" displayed on the very bottom of the screen. What you have done is to keep the graphics display in the same mode as before, but this time the starting page has been moved only 512 bytes (1/2K) or 1/12 of a hi-res graphics page (which is 6144 bytes). There is no way to duplicate this using BASIC's graphics commands. It is beyond the scope of this manual to explore all the modes possible using the special VIEW display mode, but as you can see, many different modes can be set, plus you have the ability to vary the location of the graphics video page to anywhere in memory in increments of 512 bytes. Now type VIEWR to get text back.

9.3 Notes on the special display mode

Note that although it was not mentioned before, this special display mode can also be used to control the screen display when you are switching sides. By poking the desired values into the special setup locations and then poking something other than zeros into the flag location 32595 (7F53 hex) you will cause the special display mode to be used whenever you switch into that side. The special display mode can be used on either, or both sides and each side can be set up independently of the other side. To exit from the special display mode and get back into the current BASIC display mode, simply poke a zero into the flag location 32595 (7F53 hex) on that particular side, and it will take effect the next time you switch into that side. Note that like a screen command, you will return to text mode the next time you try to print something to the text screen.

9.4 VIEW command summary

In summary, the VIEW command can be used to control which display is on the screen and to keep that display on the screen when you switch sides. It can be used to view text or graphics displays from the current or opposite side, even if you are multi-tasking. The VIEWed command will stay on the screen until reset by a VIEWR (VIEW Reset), SWITCH or CRUN without an "L", or an end to multi-tasking. The VIEW command also has a special user display mode that allows you to use graphics modes and graphics pages that are not available in BASIC. A listing of the VIEW command options follows:

VIEWR	- Restore both side's displays.
VIEWAT	- View the "A" side text screen.
VIEWAG	- View the "A" side graphics screen - uses current BASIC settings to determine mode and page.
VIEWAU	- View special "A" side user display mode - mode and graphics page data is taken from user locations.
VIEWBT	- View the "B" side text screen.
VIEWBG	- View the "B" side graphics screen - uses current BASIC setting to determine mode and page.
VIEWBU	- View special "B" side user display mode - mode and graphics page data is taken from user locations.
VIEWCT	- View current side text screen.
VIEWCG	- View current side graphics screen (see VIEWAG above).
VIEWCU	- View current side special user display mode (see VIEWAU above).

VIEWOT - View opposite side text screen.

VIEWOG - View opposite side graphics screen
(see VIEWAG above).

VIEWOU - View opposite side special user display mode
(see VIEWAU above).

10.0 The "\" (BACKSLASH) command

This may well be the most powerful of the KEY-264K commands and is probably the one you will use the most. Legal only in the "A" side, a BASIC command preceded by a "\" (Backslash) will switch to the "B" or "Background" side, copy the remainder of the BASIC command line following the backslash from the "A" side to the "B" side, Cause the "B" side to begin executing the just copied BASIC line, initiate multi-tasking and switch back to the "A" or "Foreground" side. Multiple statement lines separated by a colon (":") are allowed, as the BACKSLASH command will copy everything on the line up to the ending carriage return or [ENTER].

Note that the backslash ("\") character is created in BASIC by holding down the [SHIFT] and [CLEAR] keys.

Type the following command:

```
\FOR X=1 TO 400:PRINT X:NEXT X
```

It appears as if nothing happened, but the command you just typed in side "A" is now executing in the "B" or "Background" side. Try the SOUND command to verify that you are multi-tasking. Wait a few moments for the "Background" program to end and switch you to side "B".

Just like with the MTON command, the BACKSLASH command has 3 "end of multi-tasking" options. You have just used the first option. IF the backslash character is not immediately followed by either a "@" character or a "." (period), then an end to multi-tasking will cause an automatic switch to side "B".

Option two is when you immediately follow the backslash character with an "@" character. An end to multi-tasking with this option will result in an audio indicator of 5 short tones, with control remaining in the "A" or "Foreground" side.

Switch back to side "A" and type this command:

```
\@FOR X=1 TO 400:PRINT X:NEXT X
```

Again, wait for the "Background" program to finish. You should have heard 5 short tones as multi-tasking ended, and as you can see, you remained in side "A". Manually switch to side "B" and verify that the copied BASIC command has finished.

The third option is when you immediately follow the backslash with a "." (period). This option will also cause control to remain in the "A" or "Foreground" side when multi-tasking ends, but will give you no indication that multi-tasking has ended.

Switch back to side "A" and type the following command:

```
\.FOR X=1 TO 400:PRINT X:NEXT X
```

Immediately try a SOUND command and you should hear the tell-tale quaver in the tone that results from multi-tasking. Wait a while and try the SOUND command again. Keep trying the SOUND command until the generated tone is back to normal (clean tone with no quaver). Switch to side "B" and you will see that the copied BASIC command has finished. Multi-tasking has ended but unlike the other two options, you had no indication that it had ended.

In summary, the BACKSLASH ("\") command can be used to initiate tasks (BASIC command lines) in the "B" or "Background" side while you continue to operate in the "A" or "Foreground" side. It does this by switching to the "B" or "Background" side and copying the command line from side "A" to side "B" starting at the first character following the "\", "\@", or "\.", depending on which "end of multi-tasking" option you are using, and ending with it encounters the carriage return or [ENTER]. Once the line is copied, it begings execution of the copied command line in side "B", initiates multi-tasking, and switches back to side "A".

The BACKSLASH ("\") command gives you complete "Foreground" side control over what is to be run in the "Background" side without ever having to switch there. For instance, if you had a disk system you could type \@RUN "PROGRAM/BAS" and the "Background" side would then load and run the program "PROGRAM/BAS" from disk and signal you with 5 short tones when it either needed keyboard input, encountered an error, or finished running (end of multi-tasking). If you wanted to occasionally check the status of the "Background" program as it was running, you could use the VIEW command.

Note that the BACKSLASH ("\") command can only be executed from the "A" or "Foreground" side. Attempting to execute a BACKSLASH ("\") command in the "B" or "Background" side will result in an "SD" (SiDe) error being generated to indicate that you are in the wrong side. Also note that an attempt to execute this command when you are already multi-tasking will result in an "MU" (MUlti-tasking) error.

11.0 The WAIT command

Since you now know how to use the BACKSLASH ("\") command to start the "Background" side running under multi-tasking, it might have occured to you that you could write a program to run in the "A" of "Foreground" side that will cause the "Background" side to execute a series of sequential commands. This is similar to the "COMMAND" file, or "BATCH" techniques used on other computers to execute a series of command without sitting there and typing them in from the keyboard.

For instance if you had a disk system and wanted to run two programs in sequence in the "Background" side and give an audio indication when the second program finished, you could type (in the "A" side, of course) the following two commands:

```
 \.RUN "PROG-ONE"
 \@RUN "PROG-TWO"
```

There is one major problem with this. Although the first command would work properly, the second command would attempt to execute while you were still multi-tasking from the first command. This would result in an "MU" (MUlti-tasking) error and the second command would not be executed.

By now you have probably figured out the function of the WAIT command. The WAIT command will cause execution of the "A" side program to pause or wait until multi-tasking ends, or is temporarily turned off with the MTPAUSE command. Now you can set up an operating command file in the "A" side by separating the BACKSLASH commands with WAIT commands to insure that the previous command is finished before attempting to start the next command. The example from above would now look like this:

```
 \RUN. "PROG-ONE"
 WAIT
 \RUN@ "PROG-TWO"
 WAIT
```

Note that the WAIT comand is not on the same line as the "\" (Backslash), as this would cause it to be executed on the opposite or "B" side where it is illegal.

Get into side "A" and try this example:

```
 \@FOR X=1 TO 400:PRINT X:NEXT
 WAIT:PRINT 1/3
```

Notice that the PRINT 1/3 command did not execute until the 5 short tones indicated that multi-tasking had ended.

In summary, the WAIT command should be used whenever the "A" side program needs to know that the "B" side program has completed some particular processing, or requires keyboard input. Note that the WAIT command can only be used in the "A" or "Foreground" side. Any attempts to use it in the "B" or "Background" side will result in a "SD" (SiDe) error. If you execute a WAIT command and you are not multi-tasking at the time, the command will be ignored.

12.0 The DUP (Duplicate) command

The DUP or duplicate command functions exactly the same as the keyboard duplicate command, allowing you to make a temporary backup of a BASIC program. Unlike the keyboard duplicate command however, the BASIC DUP command lets you determine exactly at what point the duplicate copy will be made as a program is running (remember, with the keyboard command you determined when to duplicate by manually hitting the keys).

Get into side "A", manually "cold start" the side from the keyboard, and type in the following program:

```
 10 FOR X=1 TO 500
 20 PRINT X
 30 IF X 300 THEN DUP
 40 NEXT X
```

Type RUN and you should see the program display the numbers from 1 to 300 and then pause as it duplicates the program on the other side. After the DUP, the program should display numbers starting at 301 and then run to completion.

Now switch to side "B" and you should immediately start processing at exactly the point where the DUP command occured. It should start displaying numbers beginning at 301 and run to completion. Now LIST the program. It should be the same as the one you just typed in on the "A" side. In fact, the only difference in the programs is the side they are on.

In summary, the DUP command will make an exact duplicate of one side onto the other side including status information as to where you were and whether you were in "run" mode, etc. Any BASIC or machine language programs residing there will also be duplicated. Note that the DUP command will also duplicate the level of BASIC you are in. If one side is in DISK BASIC and the other side is in EXTENDED BASIC before the DUP command, then after the DUP command, the duplicated side will end up in the same level of BASIC as the side that executed the DUP command.

Note that attempting the DUP command while multi-tasking will result in an "MU" (MUlti-tasking) error.

13.0 The LCOPY (Line COPY) command

The easiest way of describing the LCOPY or line copy command is to say that it functions just like the BASIC LIST command, except that instead of listing program lines on the screen, it will copy the program lines to the opposite side.

You should still be in side "B" so type NEW to clear the existing program and then type in the following:

```
 10 PRINT 10
 20 PRINT 20
 30 PRINT 30
 40 PRINT 40
 50 PRINT 50
```

Now switch to side "A" and do a LIST. The DUP example program should still be there so type NEW to clear it. Switch back to side "B" and type LCOPY. Again switch to side "A" and do a LIST command. This time a copy of the "B" side program is there. The LCOPY command copied all 5 lines just as a LIST command would have listed all 5 lines.

Type NEW to clear the "A" side program and switch to side "B". Type LCOPY 40, then switch to side "A" and do a LIST command. This time only line 40 was copied, again similar to a LIST 40 command.

Once more type NEW and then switch to side "B". This time type LCOPY 20-40. Now switch to side "A" and do a LIST command. Just like a LIST 20-40 command, only lines 20 through 40 were copied from side "B" to side "A". Note that also like a LIST command, if you specify non-existent line numbers when copying a group of lines, it will copy the line numbers between the non-existent lines without giving you an error. For instance in the example above if you had typed LCOPY 15-45, it still would have copied lines 20 through 40.

Now type NEW to clear the "A" side program and key in the following lines:

```
 17 PRINT 17
 20 REM THIS IS SIDE A LINE 20
 24 PRINT 24
 30 REM THIS IS SIDE A LINE 30
 52 PRINT 52
```

Switch to side "B" and type LCOPY, then switch back to side "A" and do a LIST command. The "A" side program should now look like this:

```
 10 PRINT 10
 17 PRINT 17
 20 PRINT 20
 24 PRINT 24
 30 PRINT 30
 40 PRINT 40
 50 PRINT 50
 52 PRINT 52
```

What has happened is that the LCOPY command copied each line from the "B" to the "A" side just as if you had typed it in from the keyboard. In other words it performed a "MERGE". If the same line number already exists in the "A" side, then it will be replaced by the line from the "B" side. If a given line number does not already exist on the "A" side, then the "B" side line will be inserted in it's proper numerical place. Note that the lines in side "A" that did not have a corresponding line number in side "B" were left undisturbed.

Now switch back to side "B" and type LCOPYN. Switch back to side "A" again and do a LIST command. This time only the 5 lines, 10 through 50, from side "B" appeared. But what happened to lines 17, 24, and 52 ?

Following the LCOPY command with an "N" will cause a NEW command to be executed in the opposite before any lines are copied.

Note that the LCOPY command can be used to "MERGE" two BASIC programs. Just load the two BASIC programs into the two sides ("A" and "B") and perform an LCOPY command from either side. The opposite side will then contain the "MERGED" program.

The LCOPY command can also be used to copy a group of BASIC lines (routines) from one part of a program to another. Just switch to the side that the main program resides in, and do an LCOPYN XXX-YYY, where XXX is the starting line number and YYY is the ending line number of the group of lines you want copied. Now switch to the opposite side and do a RENUM ZZZ (renumber a program), where ZZZ is the starting line number of where you want to copy this group of lines. Then do an LCOPY command from this side, and switch back to the main program side. A LIST command should show that the renumbered group of lines has now been copied and inserted in their proper numerical order.

To move a group of BASIC lines, follow the same procedure used to copy a group of lines, and when finished, delete the original lines.

In summary, The LCOPY command can be used to copy one line, a group of lines, or a whole program from one side to the other side. The form and syntax of the LCOPY command is similar to the BASIC LIST command. Non-existent lines in an LCOPY command will be treated the same as non-existent lines in a LIST command. The LCOPY command (without an "N") will "MERGE" the copied program lines with any program lines already existing on the opposite side, replacing any opposite side lines that have the same line number as a copied line, with the copied line. If the LCOPY command is followed by an "N" (LCOPYN), then a NEW command will be performed in the opposite side before copying any lines. Note that this command copies only BASIC lines, unlike the DUP command which duplicates the whole side. Attempting an LCOPY command while multi-tasking will result in an "MU" (multi-tasking) error.

Please note that copying BASIC lines when one side is in EXTENDED BASIC and the other side is in DISK BASIC can result in illegal commands being generated due to conflicts in BASIC's "tokens" (see Notes section 16.0, #7 for more information).

14.0 The MCOPY (Memory COPY) command

The MCOPY or memory copy command can be used to copy the contents of a group of specified memory locations from one part of memory to another, on the same side. A word of caution before using this command. You should first be thoroughly familiar with the memory layout (memory map) of the COLOR COMPUTER since the MCOPY command has the capability of overlaying or changing some of BASIC's control locations, or the KEY-264K program itself.

If you are familiar with the concepts and locations of the graphics video memory pages, then the MCOPY command can be very useful to you. If you are not, then you may want to skip to the MCOPY summary, as the following discussion will be fairly technical.

In BASIC, graphics page 1 starts at memory location 1536 (0600 hex) for EXTENDED or location 3584 (0E00 hex) for DISK. Graphics page 5 starts at memory location 7680 (1E00 hex) for EXTENDED or location 9728 (2600 hex) for DISK. For hi-res (PMODE 3 or 4) displays you require 4 graphics pages to display a full graphics screen (1 hi-res page = 4 graphics pages). If you are doing any serious animation, then you will need 2 hi-res pages (8 graphics pages) and will switch between them with the PMODE and VIEW or SCREEN commands (don't forget to PCLEAR 8 first). If order to copy graphics information from one hi-res page to another, you would normally use the PCOPY command. But the PCOPY command only lets you copy one graphics page at a time, requiring you to do 4 PCOPY commands in order to copy one full hi-res (4 graphics pages) page to the other. With the MCOPY command you can quickly copy one full hi-res page (4 graphics pages) to the other with one command.

Just in case your hi-res display from an earlier example is gone, lets create another. You should still be in side "A" (if not, switch there) so type the following:

```
 NEW
 PCLEAR 8
 PMODE 4,1
 PCLS
 LINE (78,46)-(178,146),PSET,B
 VIEWAG
```

At this point you should see the hi-res display with the green box on the black background. Type VIEWR to get back to the text screen, and type:

```
 PMODE 4,5
 VIEWAG
```

At this point you should see a hi-res display of "garbage" since you never PCLSed this hi-res page. Type VIEWR to get the text screen back, then do the following:

In EXTENDED BASIC type:

```
 MCOPY 1536,7679 TO 7680   or   MCOPY &H0600,&H1DFF TO &H1E00
```

In DISK BASIC type:

```
 MCOPY 3584,9727 TO 9728   or   MCOPY &H0E00,&H25FF TO &H2600
```

Now again type VIEWAG, and this time you should see the familiar green box on a black background. The MCOPY command has copied the full first hi-res page to the second.

Note that the MCOPY command can also be used to directly copy portions of a graphics page to either another graphics page, or to a different location on the same graphics page, without using PUT and GET commands.

Type VIEWR to again return to the text screen.

In summary, the MCOPY command can be used to copy the contents of a block of memory from one location to another in the same side. The MCOPY command uses the format:

```
MCOPY ssss,eeee TO dddd
```

Where ssss is the starting or source location of a block of memory, eeee is the ending location of the block of memory, and dddd is the destination starting location to which you want the block of memory copied.

Note that from as little as 1 byte up to the limits of memory (be careful!!) can be copied in a single command. The copied locations can overlap each other. Try this, type:

```
 CLS
 MCOPY 1024,1533 TO 1026   or   MCOPY &H0400,&H5FD TO &H0402
```

Because of the overlapping memory locations (1024 and 1026), the pattern of the first 2 characters is repeated and the screen is filled with the BASIC prompt "OK" (except for the cursor location).

The MCOPY command can also be used to move machine language programs around in memory if desired (although they will not necessarily run in another location). Note that attempting to use this command during multi-tasking will result in an "MU" (MUlti-tasking) error. Also note that the specified starting location must be less than or equal to the ending location or an "SN" (SyNtax) error will occur.

15.0 The CMCOPY (Cross Memory COPY) commands

The CMCOPY or cross memory copy can be used to copy a block of memory either to or from the opposite side. The CMCOPYT (Cross Memory COPY To) command performs exactly the same function as the MCOPY command except that the source locations are on the current side, and the destination locations will be on the opposite side. Likewise, the CMCOPYF (Cross Memory COPY From) command functions like the MCOPY command except that the source locations are on the opposite side and the destination locations are on the current side.

As with the MCOPY command, caution should be used when using the CMCOPY commands as they can wipe out BASIC control locations or parts of the KEY-264K program. In fact, all the same comments about being familiar with the COLOR COMPUTER's memory map, etc. when using the MCOPY command, also apply to the CMCOPY command.

Like the MCOPY command, the CMCOPY commands have the following format:

```
 CMCOPYT ssss,eeee TO dddd
             or
 CMCOPYF ssss,eeee TO dddd
```

Where ssss is the starting or source location of a block of memory, eeee is the ending location of the block of memory, and dddd is the destination starting location (remember that the source and destination locations will not be on the same side).

Still in side "A", type the following:

```
 CMCOPYT 1024,1535 TO 1024   or   CMCOPYT &H0400,&H5FF TO &H0400
```

Now switch to side "B" and you will see that the "A" or current side's text screen has been copied to the "B" or opposite side's text screen. Note that the flashing cursor may not be in the same position on the "B" side's screen. This is because all you did was to change the contents of the "B" side's text screen, you did not alter the "B" side's processing (such as cursor location) in any way. While you are still in side "B", type:

```
 FOR X=1 TO 200:PRINT X;:NEXT X
```

This will fill the "B" side screen with numbers. Switch to side "A" and type:

```
 CMCOPYF 1024,1535 to 1024   or   CMCOPYF &H0400,&H5FF TO &H0400
```

The screen should immediately fill with the numbers from the "B" side's screen (with the exception of the cursor location). You have just copied memory from the "B" or opposite side's text screen to the "A" or current side's text screen.

In summary, the CMCOPY (Cross Memory COPY) commands can be used to copy the contents of a specified block of memory either to or from a specified block of memory locations on the opposite side.

Note that you could create hi-res graphic displays in memory on the "A" side and then CMCOPYT them to 1 of 4 hi-res graphics pages (16 graphics pages) on the "B" side and display any of the 4 by setting up and using the VIEWBU command (VIEW the "B" side User display). Or you could simply CPOKE the desired PMODE (0-4) into memory location 182 (B6 hex), then CPOKE the high order 8 bits of the "B" side graphics page you wanted to display into memory location 186 (BA hex) and then use the VIEWBG (VIEW the "B" side Graphics display) Never attempt to switch to the "B" side, initiate multi-tasking, etc. when using these techniques, as you will be using most of the available memory on that side as graphic pages. Also do not attempt to get 5 hi-res graphics pages on the "B" side or you will wipe out the KEY-264K program on that side. Again, in order to use extra graphics pages, you must have knowledge of where the graphics pages are located for the level of BASIC you are in, etc. as described under the VIEW command.

The CMCOPY commands can also be used to move machine language programs to either the same or a different locatiom in memory on the opposite side. Although they may not run when moved to a different location in memory, they should function properly if moved to the same location on the opposite side. Note that attempting to use the CMCOPY commands while multi-tasking will result in an "MU" (MUlti-tasking) error. Also note that the specified starting location must be equal to or less than the ending location or a "SN" (SyNtax) error will occur.

16.0 General notes, comments and cautions

This section contains some random notes about the operation of the KEY-264K.

1 - Note that both side's BASICs have had the equivalent of a CLEAR 200,29543 performed at start-up to protect the KEY-264K program. any attempt to set "high memory" higher than this number with a CLEAR command will cause an error.

2 - Unlike programs that use map type 1 to access 64K, you can use the vitamin "E" poke (POKE 65495,0 if your system can handle it) with the KEY-264K to speed up processing. Note that both sides will be affected by the speed-up. Don't forget to reset it (POKE 65494,0) before doing any I/O.

3 - If your system "hangs" or "locks up" and you can't seem to get out of it with keyboard commands, try the following:

> Push the RESET switch at the back of the computer. If you don't end up in side "A" with the cursor on the screen, then try holding down the [↓], and [C] keys while you hit the RESET switch. Many times this will get you back to the KEY-264K with side "A" "cold started". Always hold down the [↓] and [@] keys to "cold start" the opposite ("B") side after pushing the RESET switch. If the opposite side "cold start" doesn't clear side "B", then follow the above procedures, but this time try to DUP (or push the [↓] and [D] keys) the "A" side to side "B". If this should fail to clear side "B", you will have to power down and reload the KEY-264K.

4 - Note that in DISK BASIC the 1/60th second interrupts are used to "timeout" (delay a few seconds after the last disk access) the disk drive motors before turning them off. If you are not multi-tasking and you initiate a disk command in one side and then switch to the other side before the disk drive motor has "timed out", then the motor will remain on until you switch back to the original side and stay there long enough for the "timeout" to occur, which will then shut the motors off. You can however shut off the disk drive motors from either side by POKEing a zero into location 65344 (FF40 hex).

5 - When using the VIEW commands, note that there is only one Video Display Generator (6847 VDG) and it doesn't know about the "PAGE" switch. This affects the KEY-264K in two ways; First, a SCREEN command will override a VIEW command, in that it can change what is displayed on the screen. all other aspects of the VIEW command will remain in effect however, until reset. Second, a screen command issued on side "B" will cause the corresponding "A" side display to appear on the screen. This is because the SCREEN command is executed by BASIC and not by the KEY-264K.

6 - Machine language programs that maintain the integrity of BASIC (leave the interrupts on, use the BASIC I/O routines, and don't modify any BASIC hooks) can usually be run with the KEY-264K as long as they are loaded and run below memory location 29542 (7366 hex) and don't use "MAP TYPE" "1". The keyboard switch, duplicate, and multi-tasking commands should work, but you will still have to try each program on an individual basis to make sure that it functions properly.

7 - If you have a DISK system you may run into a problem when loading programs from cassette that were not saved in ASCII mode. The problem is this; BASIC replaces each command keyword in a BASIC line with a single byte "token" or number so that each line takes less memory, and to speed up processed when the line is executed. As you upgrade from EXTENDED to DISK BASIC, additional (higher number) tokens are added for the DISK basic commands. You have the capability with the KEY-264K to be in either EXTENDED or DISK BASIC and the KEY-264K will automatically add it's "tokens" to the highest existing "token" for either EXTENDED or DISK BASIC. Now if you CSAVE a program to cassette in EXTENDED BASIC that contains some KEY-264K commands, and subsequently CLOAD it into DISK BASIC, the KEY-264K commands will appear as DISK commands. This is due to the fact that the "token" numbers added to EXTENDED BASIC by the KEY-264K are numerically the same as those added to EXTENDED BASIC by DISK BASIC. This is unavoidable and can be remedied by either CSAVEing the program in ASCII format or making sure that you always CLOAD in the same level of BASIC that you CSAVed from. Note that the reverse of this can be true also, CSAVEing in DISK BASIC and CLOADing in EXTENDED BASIC will cause DISK commands (which by the way are illegal in EXTENDED BASIC anyway) to appear as KEY-264K commands, and the KEY-264K commands will be invalid. Note that this problem also occurs when LCOPYing BASIC lines and one side is in EXTENDED BASIC and the other side is in DISK BASIC.

8 - Here's a way for you COLOR COMPUTER owners who are not able to write to the upper 32K memory in the 64K "MAP TYPE" "1" mode because of a ROM enable conflict, to copy BASIC from ROM to RAM, and modify it if you want. Or you can load a machine language program into the upper 6K (10K minus 4K for the KEY-264K) of unused space from D800 to F000 hex. The following example will give you an idea of what to do:

<u>BASIC LEARNING GUIDE</u>

A - "Cold start" both sides, then get into side "A" and type the following:

```
CMCOPYT &H8000,&HD7FF TO 0000
```

This will copy the BASIC ROMS to the page 1 (side "B") RAM. (Note that at this point, if you load a machine language program into memory locations between &H5800 to &H7000 in side "B", it will later appear in locations between &HD800 to &HF000)

```
CPOKE &H2A7B,&H4F
```

This will change the "I" in BASIC's "PRINT" command word located in ROM location AA7B hex, to an "O". Notice that we CPOKED the "O" (4F hex) into location 2A7B hex, because we have moved the ROMs to RAM and offset the locations by a minus 8000 hex.

B - Now POKE &H71,0 and hit the "RESET" switch at the back of the computer to create a "cold start" and get you out of the <u>KEY-264K</u> (Do not power down your computer or you will lose everything that you have been doing).

C - POKE &HFFDF,0 to get into "MAP TYPE" "1". This will turn off your BASIC ROMS and turn on the upper 32K bank of ram (side "B") in it's place. Since we have copied the BASIC ROMs to their corresponding locations in side "B" (ROM location, minus 32K or 8000 hex), then when we go into "MAP TYPE" "1", side "B" becomes the upper 32k bank and the copied BASIC code is in it's correct place. Note that you still cannot write to the upper 32K in the " MAP TYPE" "1" mode, so if you want to make changes or load maching language programs in the upper part of memory, you must first do so in side "B" under control of the <u>KEY-264K</u>.

D - Now try a PRINT 1/3 command. You should have gotten a "SN" or syntax error. Try a PRONT 1/3 command (remember, you changed the "I" in PRINT to an "O") and it should work OK. This proves that you are using the modified copy of BASIC in ram.

E - To get back to ROM BASIC, simply POKE &HFFDE,0 and this will switch you back to "MAP TYPE" "0". Now try a PRINT 1/3 command and it should work properly. You can now switch between "MAP TYPEs" "0" and "1" (ROM and RAM) as you wish, with the proper POKE commands.

9 - Note that you should be able to run programs that use disk on both sides during multi-tasking if you temporarily turn off multi-tasking with an MTPAUSE, do your disk command, and then turn multi-tasking back on with an MTON command. Don't forget to set the "end of multi-tasking" option with the MTON command.

10 - If you find you need additional space for your program on one side, there is a way to keep most of you variables on the opposite side until needed. This is accomplished by first "PUSHing" any necessary variables to the opposide side and the executing a "CLEAR" statement to reclaim the variable space.

Any variables necessary for program control such as counters, flags, constants, etc. would be "PULLed" back immediately, but the remainder of the variables could remain safely tucked away on the opposite side until needed, and then retrieved with a "PULL" command.

The process of "PUSH", "CLEAR", and "PULL when needed" can be repeated any number of times during the execution of a program. Note that you could use the "DUP" command instead of pushing each variable, but only the first time. After that you would have to "PUSH" whichever variables you wanted to save.

This technique would be most effective when processing large arrays of data. The following example should give you an idea of how it would work:

```
 10 DIM A(255)
 20 FOR X=0 TO 255:REM FILL ARRAY WITH NUMBERS 0-255
 30 A(X)=X
 40 NEXT X
 50 PRINT MEM:REM PRINT SPACE LEFT
 60 \.DIM A(255):REM DIM ARRAY ON THE OPPOSITE SIDE
 70 WAIT
 80 FOR X=0 TO 255
 90 PUSH X:REM VALUE OF X MUST BE THE SAME ON BOTH SIDES
100 PUSH A(X):REM PUSH ARRAY TO THE OPPOSITE SIDE
110 NEXT X
120 CLEAR
130 PRINT MEM:REM SHOULD HAVE RECLAIMED ARRAY + VARIABLE SPACE
140 REM AT THIS POINT NEW ARRAYS COULD BE DECLARED
150 REM OR NEW VARIABLES USED, AND THEN "CLEARed" IF NECESSARY
160 DIM A(255):REM DIM ARRAY BEFORE PULLING IT BACK
170 FOR X=0TO 255
180 PUSH X:REM VALUE OF X MUST BE THE SAME ON BOTH SIDES
190 PULL A(X):REM PULL ARRAY BACK TO ORIGINAL SIDE
200 NEXT X
210 FOR X=0 TO 255
220 PRINT A(X):REM PRINT OUT ARRAY TO SHOW IT IS THERE
230 NEXT X
240 END
```

Note the "PUSH X" statements in lines 90 and 180. These are necessary because when "PUSHing" or "PULLing" a subscripted variable that has a variable as a subscript ("X" in our case), then the value of the variable used as a subscript will be taken from each side as the location of the "PUSHed" or "PULLed" subscripted variable is determined for that side. (whew!)

In other words, if variable "X" was equal to 1 on side "A" and the same variable "X" was equal to 4 on side "B", then the "A" side statement "PUSH A(X)" would copy or "PUSH" the subscripted variable "A(1)" on side "A" to the subscripted variable "A(4)" on side "B".

There are many times that we can use this fact to our advantage, such as re-ordering an array as we "PUSH" or "PULL" it from one side to the other. However, failing to take this into account when "PUSHing" or "PULLing" subscripted variables can cause unexpected results.

Note that you can freely "PUSH" or "PULL" subscripted variables that have a numeric constant as the subscript (ie A(3)) with no problems.

11 - AS you probably know, you can gain extra memory space by doing a "PCLEAR 1" command. This frees up 3 of the 4 graphics pages set aside by BASIC.

There is also a technique known as "PCLEAR 0" (POKE 25,6:NEW) That will free up the 4th or last graphics page, but this dosen't appear to work with the KEY-264K.

The problem stems from the fact that BASIC expects to find a zero in the byte immediately preceeding the "start of BASIC" location, in this case 0600 hex. This space is usually cleared on cold start by the COLOR BASIC (not EXTENDED) ROM since this is where the "start of BASIC" is for non-extended BASIC. The KEY-264K does not clear this area since it requires at least EXTENDED BASIC to operate.

The solution is simple, just poke a zero into location 0600 hex before doing the "PCLEAR 0":

```
POKE &H0600,0:POKE25,6:NEW
```

If you do a "PRINT MEM" you will see that there is no space reserved for graphics. (make sure you don't use graphics commands or you may clobber your program.)

KEYBOARD REFERENCE SECTIOM

KEYSTROKES	FUNCTION
[↓], [ENTER]	Switch from side "A" to side "B". Either side can be in command or run mode. This command is disabled during multi-tasking.
[→], [ENTER]	Switch from side "B" to side "A". Either side can be in command or run mode. This command is disabled during multi-tasking.
SPACE BAR, [→], [ENTER]	Initiate multi-tasking and switch to side "A". Multi-tasking will remain in effect until ended by a "B" side request for keyboard input (with the exception of the INKEY$ function) or a return to command mode for any reason. This command must be initiated from side "B".
[↓], [B]	"BREAK" the "A" side. Note that you cannot usually CONT from this type of "BREAK". Does not affect side "B" and will not end multi-tasking.
[→], [B]	"BREAK" the "B" side. Note that you cannot usually CONT from this type of "BREAK". Does not affect side "A" but the return to command mode in side "B" will end multi-tasking.
[↓], [R]	"RESET" the "A" side. Does not affect the "B" side and will not halt multi-tasking.
[→], [R]	"RESET" the "B" side. Does not affect the "A" side, but the return to command mode on side "B" will end multi-tasking.
[↓], [C]	"COLD START" side "A" in whatever level of BASIC is physically in your computer. Does not affect side "B" and will not end multi-tasking.
[→], [C]	"COLD START" side "B" in whatever level of BASIC is physically in your computer. Does not affect side "A" but the return to command mode in side "B" will end multi-tasking.

KEYBOARD REFERENCE SECTION

KEYSTROKES	FUNCTION
[↓], [E]	"COLD START" side "A" in EXTENDED BASIC. Does not affect side "B" and will not end multi-tasking.
[→], [E]	"COLD START" side "B" in EXTENDED BASIC. Does not affect side "A" but the return to command mode in side "B" will end multi-tasking.
[↓], [@]	SWITCH to, and "COLD START" side "B" from side "A". You must be in side "A" when you execute this command. This command is disabled during multi-tasking.
[→]. [@]	SWITCH to, and "COLD START" side "A" from side "B". You must be in side "B" when you execute this command. This command is disabled during multi-tasking.
[↓], [D]	Duplicate side "A" to side "B". The level of BASIC in side "A" will be also be copied to side "B". This command is disabled during multi-tasking.
[→], [D]	Duplicate side "B" to side "A". The level of BASIC in side "B" will also be copied to side "A". This command is disabled during multi-tasking.

BASIC REFERENCE SECTION

BASIC COMMAND FORMAT NOTATIONS

The following format rules apply to the BASIC commands described in this reference section.

1. Items in capital letters must by typed as they appear.

2. The user will supply items show in lower case letters within the angle brackets < and >.

3. Any items appearing between the square brackets [and] are optional.

4. All commas, colons, and parentheses must be entered as shown.

5. Items that end with an ellipsis (......) can be repeated to the maximum length of the line.

1.0 BACKSLASH "\"

Format: \[<option>] <basic command> [:<basic command>......]
where <option> = The character "@" or "." (period) that is used to control how the end of multi-tasking will be handled.

Purpose: Used to initiate tasks (BASIC command lines) in the "Background" or "B" side in the multi-tasking mode from the "Foreground" or "A" side.

Comments: This command will switch to the "B" or "Background" side, copy the remainder of the command line following the backslash "\" (or option character if present) to side "B", initiate multi-tasking, and start the "B" side executing the copied command line. Multiple statements/line are allowed. Multi-tasking will end when the "B" side requires keyboard input (with the exception of the INKEY$ function) or returns to command mode for any reason.

There are 3 "end of multi-tasking" options:

1. If no special character follows the backslash "\" then multi-tasking will end with a switch to the "B" or "Background" side.

2. If an "@" character follows the backslash "\" then multi-tasking will end with an audio indicator of 5 short tones and control will remain in the "A" or "foreground" side.

3. If a "." (period) follows the backslash "\" then multi-tasking will give no indication that it has ended and control will remain in the "A" side.

Issusing a Backslash command during multi-tasking will cause a multi-tasking error "MU" to occur.

This command is legal only in the "Foreground" or "A" side. Attempting to execute a backslash "\" command in the "Background" or "B" side will cause a side error "SD" to occur.

Example:

```
\PRINT 1/3
\.CLOAD "PROGRAM"
\@FOR X=1 TO 1000:PRINT X:NEXT X
```

2.0 CMCOPY (Cross Memory COPY)

Format: CMCOPY<option> <ssss>,<eeee> TO <dddd>

where <option> = T (copy To) to copy from the current side to the opposite side.

<option> = F (copy From) to copy from the opposite side to the current side.

<ssss> = Starting or source location of a block of memory.

<eeee> = Ending location of the block of memory.

<dddd> = Destination's starting memory location.

Purpose: To copy the contents of a block of specified memory locations either to or from specified memory locations in the opposite side.

Comments: Attempting to use the CMCOPY commands while you are multi-tasking will result in a "MU" multi-tasking error.

The specified source starting location must be less then or equal to the ending location or a "SN" syntax error will occur.

As few as 1 byte up to the limits of memory can be copied in a single command.

Caution should be used with these commands since they have the capability of overlaying or changing some of BASIC's control locations, or the KEY-264K program itself.

Example:

```
CMCOPYF &H0400,&H05FF TO &H0400
CMCOPYT &H8000,&HD7FF TO 0000
```

3.0 CPEEK (Cross PEEK)

Format: CPEEK(<nnnn>)
where <nnnn> = A numeric constant, numeric variable, or a BASIC formula in the range 0-65535 representing the desired memory address.

Purpose: To return the single byte integer value contents (0-255) of a memory location on the opposite side.

Comments: Operates exactly like the normal BASIC PEEK function except that the memory location PEEKed will be on the opposite side.

Since there are no banks or sides in the upper 32K of memory, a CPEEK of a memory location greater than 32767 will give the same results as a regular BASIC PEEK function.

Note that this is a function and not a command.

Example:

```
A=CPEEK(X)
IF B$=STR$(CPEEK(&H4000)) THEN 100
PRINT CHR$(CPEEK(B*7))
```

4.0 CPOKE (Cross POKE)

```
Format:     CPOKE <aaaa>,<nnn>
            where <aaaa> = a numeric constant, numeric variable
                           or BASIC formula in the range
                           0-65535 representing the desired
                           memory address.
                  <nnn>  = a numeric constant, numeric variable
                           or BASIC formula representing the
                           desired integer value (0-255) you
                           wish to CPOKE into memory.
```

Purpose: To change the contents of a memory location on the opposite side.

Comments: Functions exactly like the normal BASIC POKE command except that the memory location POKEd will be on the opposite side.

Since there are no banks or sides in the upper 32K of memory, a CPOKE of a memory location greater than 32767 will give the same results as a regular BASIC POKE command.

Example:

```
CPOKE &H71,0
CPOKE X*7+Y,ASC(A$)
CPOKE M,INT(R*9/S)
```

5.0 CRUN (Cross RUN)

```
Format:      CRUN[L] [<line number>]
             where       L  = If present, leaves the current
                              display on the screen when switching
                              sides.
            <line number> = A numeric constant representing an
                              opposite side BASIC line number.
```

Purpose: To switch to the opposite side, and put the opposite side in the "run" mode.

Comments: If the CRUN command is followed by an "L", the current screen display will remain on the screen when you switch sides. This option disables the opposite side's display until restored by a SWITCH or CRUN command without an "L", a VIEWR command or an end to multi-tasking.

If the CRUN command has an optional line number then after switching to the opposite side, execution will be attempted at that line number, otherwise execution will begin at the first line number in the BASIC program.

Note that like the BASIC RUN command, all opposite side variables will be cleared before beginning execution of the opposite side program.

If an optional line number is specified but does not exist in the opposite side's BASIC program, then a switch to the opposite side will still take place but a "UL" undefined line number error will then occur.

Attempting to execute a CRUN command while multi-tasking will result in a "MU" multi-tasking error.

Example:

```
CRUN
IF X=7 THEN CRUNL
CRUN 100:CRUN 200
CRUNL 750
```

6.0 DUP (DUPlicate)

Format:	DUP
Purpose:	To make an exact duplicate of the current side onto the opposite side.
comments:	Functions the same as the keyboard duplicate command except under program control.

Duplicates all memory areas plus information such as BASIC programs, variables, status, whether you were in "command" or "run" mode , and all BASIC control locations. A subsequent switch to the opposite side will cause you to begin execution of the duplicated program at exactly the point at which the DUP command took place.

Any machine language programs residing in the current side will also be copied to the opposite side.

Note that the level of BASIC (EXTENDED or DISK) on the current side will also be copied to the opposite side.

Attempting to execute a DUP command while multi-tasking will result in a "MU" multi-tasking error.

Example:

```
DUP
IF X=100 THEN DUP
```

7.0 LCOPY (Line COPY)

Format: LCOPY[N] [<line number> [- [<line number>]]]

where N = If present, a BASIC NEW command will be executed on the opposite side before copying any lines.

<line number> = line numbers in the current side's BASIC program.

Purpose: To copy one or more BASIC program lines from the current side to the opposite side.

comments: Works like the BASIC LIST command except instead of listing BASIC program lines on the screen, They will be copied to the opposite side.

Non-existent line numbers are also handled in the same manner as the BASIC LIST command.

If the LCOPY command is not followed by an "N", then the program lines on the current side will be merged with any program lines that already exist on the opposite side. Note that if a current side line number is the same as an opposite side line number, then the current side line will overlay the opposite side program line. Also, any program lines that exist in the opposite side which do not have corresponding line numbers in the current side, will be left undisturbed. all other program lines will be copied to the opposite side.

An LCOPY or LCOPYN command without a line number will copy all BASIC program lines in the current side's program to the opposite side.

Attempting to execute an LCOPY command while multi-tasking will result in a "MU" multi-tasking error.

Example:

```
LCOPY 100
LCOPYN 50-75
LCOPY
LCOPYN 3000-
LCOPY -210
```

8.0 MCOPY (Memory COPY)

Format:	MCOPY <ssss>,<eeee> TO <dddd> where <ssss> = Starting or source location of a block of memory. <eeee> = Ending location of the block of memory. <dddd> = Destination's starting memory location.
Purpose:	To copy the contents of a block of memory from one set of memory locations to another set of memory locations on the same side.
Comments:	Attempting to use the MCOPY commands while you are multi-tasking will result in a "MU" multi-tasking error. The specified source starting location must be less then or equal to the ending location or a "SN" syntax error will occur. As few as 1 byte up to the limits of memory can be copied in a single command. Caution should be used with this command since it has the capability of overlaying or changing some of BASIC's control locations, or the KEY-264K program itself.
Example:	MCOPY &H0600,&H1DFF TO &H1E00 MCOPY 1024,1534 TO 1025

9.0 MTON (Multi-Tasking ON)

Format: MTON[<option>]
where <option> = The character "@" or "." (period) that is used to control how the end of multi-tasking will be handled.

Purpose: To turn on multi-tasking and set an "end of multi-tasking" option from either the "A" or "B" side.

Comments: There are 3 "end of multi-tasking" options:

1. If no special character follows the MTON command then multi-tasking will end with a switch to the "B" or "Background" side.

2. If an "@" character follows the MTON command then multi-tasking will end with an audio indicator of 5 short tones and control will remain in the "A" or "foreground" side.

3. If a "." (period) follows the MTON command then multi-tasking will give no indication that it has ended and control will remain in the "A" side.

Note that attempting to execute an MTON command while multi-tasking will cause the command to be ignored and will not change the previously set "end of multi-tasking" option.

Example:

```
MTON
IF X=250 THEN MTON@
MTON.
```

10.0 MTPAUSE (Multi-Tasking PAUSE)

Format: MTPAUSE

Purpose: To provide a temporary halt to multi-tasking.

comments: The MTPAUSE command will cause multi-tasking to halt with control remaining in the side that executed the command. The current screen display will not be affected.

The MTPAUSE command will not automatically cause the "end of multi-tasking" options to occur. This means you can continue to process normally without multi-tasking, until the "B" or "Background" side requests input from the keyboard or returns to command mode for any reason.

The MTPAUSE command can be used when you need to perform a command that would be illegal if you were multi-tasking, or you want to access the same I/O device from programs in both sides, or you need to speed up processing of a routine for some reason (like a PLAY or SOUND command).

You will usually follow an MTPAUSE command at some point with an MTON command to again turn on multi-tasking (don't forget the "end multi-tasking optiom).

Executing an MTPAUSE command when you are not multi-tasking will cause the command to be ignored.

Example:

```
MTPAUSE
IF X=50 THEN MTPAUSE
MTPAUSE:PLAY "O3CDEFGABO4C":MTON@
```

11.0 PULL (a variable)

Format: PULL <variable>[,<variable>,<variable>......]

Purpose: To copy the contents of a variable from the opposite side to the current side. More than one variable can be PULLed with the same command line as long as they are separated by a coma (","). Variable types (string and numeric) may be intermixed on the same command line.

Comments: The total length of the command line must not exceed 128 characters or an "SL" statement too long error will occur.

If the variable does not exist on the current side it will be created before the copy takes place.

If the variable does not exist on the opposite side then it will be created and assigned a value of zero before the copy takes place.

When PULLing a dimensioned variable and the variable does not exist (on either or both sides - see above) then a default dimensioning of size 10 takes place before the copy occurs. Note that if the dimensioned variable is greater than the default size, then the variable must be dimensioned to it's proper size on both sides before attempting the PULL command or a "BS" bad subscript error will occur.

Normal BASIC "garbage" collect routines will be called as required when PULLing string variables.

Example:

```
PULL A,X$,Z(2,3),H$(9)
IF X=99 THEN PULL X
PULL Z(1,2,2),NAME$,Y
```

12.0 PUSH (a variable)

Format: PUSH <variable>[,<variable>,<variable>......]

Purpose: To copy the contents of a variable from the current side to the opposite side. More than one variable can be PUSHed with the same command line as long as they are separated by a coma (","). Variable types (string and numeric) may be intermixed on the same command line.

Comments: The total length of the command line must not exceed 128 characters or an "SL" statement too long error will occur.

If the variable does not exist on the opposite side it will be created before the copy takes place.

If the variable does not exist on the current side then it will be created and assigned a value of zero before the copy takes place.

When PUSHing a dimensioned variable and the variable does not exist (on either or both sides - see above) then a default dimensioning of size 10 takes place before the copy occurs. Note that if the dimensioned variable is greater than the default size, then the variable must be dimensioned to it's proper size on both sides before attempting the PULL command or a "BS" bad subscript error will occur.

When PUSHing a string variable and the same named string variable already exists on the opposite side, then an attempt will be made to reuse that variable's string space on the opposite side as follows:

A - If the length of the current side's string variable is less than or equal to the size of the opposite side's string variable, then the opposite side string variable is reused.

B - If the length of the current side's string variable is greater than the length of the opposite side's string variable, then the string variable will not be reused and the normal BASIC string processing will occur. Note that when this occurs, BASIC's "garbage collect" routines will not be called, and previously used string spaces will not be reclaimed.

Example:

```
PUSH A,B$,C(1),D$(2,3)
IF X=101 THEN PUSH Y(2,3,4),TEST$,Z,U$,P$(1)
```

13.0 SCOLOR (Switch text screen COLOR)

Format: SCOLOR

Purpose: To switch the text screen color on either side from GREEN to ORANGE or from ORANGE to GREEN.

Comments: Can be used to make the text screen color the same on both sides. This eliminates the flashing that can occur when you are switching sides (and displays) and using the text displays on both sides.

Example:

```
SCOLOR
IF X=35 THEN SCOLOR
```

14.0 SWITCH

Format: SWITCH[L] [<line number>]

```
where         L  = If present, leaves the current
                   display on the screen when switching
                   sides.
<line number> = A numeric constant representing an
                   opposite side line number.
```

Purpose: To switch to the opposite side.

comments: If the SWITCH command is followed by an "L", the current screen display will remain on the screen when you switch sides. This option disables the opposite side's display until restored by a SWITCH or CRUN command without an "L", a VIEWR command, or an end to multi-tasking.

If the SWITCH command is followed by an optional line number, then after switching sides, the opposite side will be placed in the "run" mode and execution will start at the specified line number(but unlike the CRUN command, variables will not be cleared), otherwise the SWITCH command will switch to the opposite side and continue processing at the point where the last SWITCH command on that (opposite) side occured (can be in either "run" or "command" mode).

If an optional line number is specified but does not exist in the opposite side's BASIC program, then a switch to the opposite side will still take place, but a "UL" undefined line number error will then occur.

An attempt to execute a SWITCH command while multi-tasking will result in a "MU" multi-tasking error.

Example:

```
SWITCH
IF X=1 THEN SWITCHL 100
SWITCH 10:SWITCH 20:SWITCH 30
SWITCHL
```

15.0 VIEW (a display)

Format: VIEW<side><type>

```
where <side> = A - View the "A" side.
               B - View the "B" side.
               C - View the current side.
               O - View the opposite side.
               R - Restore both side's displays.
       <type> = T - Text screen display mode.
               G - Graphics screen display mode.
               U - Special User display mode.
```

Purpose: To provide new display capabilities in BASIC to allow maximum use of the KEY-264K's additional memory and "A" and "B" side concepts. Allows you to display either text, graphics, or a special user defined mode of either side, from either side.

Comments: Once a display is put up on the screen by a VIEW command (both side's displays will be disabled), it will remain on the screen until restored by a VIEWR command, a SWITCH or CRUN command without an "L", or an end to multi-tasking.

When displaying graphics with the VIEW command, the current BASIC PMODE and graphics page settings will be used from whichever side's graphics you are displaying.

The special USER display mode makes use of the following memory locations on both sides, to determine what should be displayed on the screen (see BASIC learning guide, page 37 for more information):

32592 (7F50 hex) -	Should contain the high order 7 bits of the 16 bit video display address (page) in the form XXXXXXX0.
32593 (7F51 hex) -	Should contain the 3 mode control bits that determine how the video memory is addressed in the form 00000XXX.
32594 (7F52 hex) -	Should contain the 5 control line bits that determine the various display modes and color sets in the form XXXXX000.

15.0 VIEW (a display) continued

Comments: The following is a list of the various view command options and their functions.

VIEWR	- Restore both side's displays.
VIEWAT	- View the "A" side text screen.
VIEWAG	- View the "A" side graphics screen - uses current BASIC settings to determine mode, color set, and page.
VIEWAU	- View special "A" side user display mode - mode, color set, and page data is taken from special user memory locations.
VIEWBT	- View the "B" side text screen.
VIEWBG	- View the "B" side graphics screen - uses current BASIC settings to determine mode, color set, and page.
VIEWBU	- View special "B" side user display mode - mode, color set, and page data is taken from special user memory locations.
VIEWCT	- View the current side text screen.
VIEWCG	- View the current side graphics screen (see VIEWAG above).
VIEWCU	- View the current side special user display mode (see VIEWAU above).
VIEWOT	- View the opposite side text screen.
VIEWOG	- View the opposite side graphics screen (see VIEWAG above).
VIEWOU	- View the opposite side special user display mode (see VIEWAU above).

Example:

```
IF X=5 THEN VIEWAT ELSE VIEWBT
VIEWAG:VIEWBG
VIEWAU:VIEWBU
```

16.0 WAIT (for multi-tasking to end)

Format:	WAIT
Purpose:	To provide a pause in "Foreground" or "A" side processing until the "Background" or "B" side program either ends or requests keyboard input (with the exception of the INKEY$ command) which will cause multi-tasking to end.
Comments:	This command is used whenever the "A" side program needs to know, before proceeding, that the "B" side program has paused, completed processing or is awaiting keyboard input. Note that this command is legal only in side "A". If you try to execute a WAIT command in side "B" you will get a "SD" illegal side error. Attempting to execute a WAIT command if you are not multi-tasking will cause the command to be ignored.
Example:	\.FOR X=1 to 1000:PRINT X:NEXT X WAIT IF X=6 THEN WAIT