COMMODORE®

1764 RAM EXPANSION MODULE™

Increases your RAM by 256K Bytes

User's Guide

Commodore® 1764 RAM Expansion Module User's Guide

Contents

Section 1: Getting Started 1
Section 2: 1764 Test/Demo Software. 5
Section 3: RAMDOS 11
Section 4: Programmer's Reference Notes. 17
Appendix A: 1764 Diagnostic Test Program. 25

How to Use This Guide

Section 1 introduces you to the 1764 RAM Expansion Module, and tells you a little bit about what you can do with it. There are installation instructions, and a few warnings you'll need to be aware of.

If you are a beginning to intermediate level programmer, you will find the programs on the 1764 Test/Demo disk indispensible in utilizing the additional memory. Section 2 concentrates on the software programs included that were specifically designed for use with your 1764, and should be helpful for all users, regardless of level of programming experience.

The software also enables you to run the RAMDOS, an operating system based in the RAM Expander that lets you perform disk operations quickly and easily. Section 3 provides technical and programming details on the RAMDOS, for both experts and novices.

If you intend to write your own programs to use the 1764, you will also find the programmer's reference notes useful. In addition to the memory map and details about the 1764 registers, this section has a sample program to load and recall a text screen using the 1764. Section 4 also includes information about the RAM Expansion Controller and other technical features of the 1764.

Section 1: Getting Started

Introduction

A RAM Expansion Module is a device that increases the amount of memory of your computer, giving you more room for storing programs and data. Your computer has two kinds of memory, Random Access Memory (RAM) and Read Only Memory (ROM). RAM is used for temporary storage of data. Programs stored in RAM only stay there as long as the computer stays on. ROM stores the operating system of your computer, and cannot be changed. The 1764 increases the amount of RAM of your Commodore 64 or Commodore 64C.

The RAM Expansion Module contains **indirectly** accessible RAM, meaning that programs stored in the 1764 cannot be executed directly by the C64. You must first transfer the program into the computer's main memory, which takes only the time to type in the command.

Keep in mind that RAM is not permanent storage; the information held in RAM is gone when you turn off the computer. **If you want to save something permanently, make sure you store it on disk.**

The 1764 RAM Expander increases the memory capacity of your C64 from 64 kilobytes (64K) by 256K, for a total of 320K. To give you an idea of how much space your 1764 gives you, an entire word processor takes up under 30K. A complex game with over 50 levels takes up about the same amount of memory. So you see how much extra memory your 1764 provides for your use.

But the 1764 gives you more than memory—it gives you speed and convenience. You won't have to wait for the computer to access the disk drive for parts of programs, or to use utilities. You can store these in the extra RAM, for immediate access. You can also use this added speed to perform animation by cycling sequences of pictures or graphics that you wouldn't ordinarily be able to manage on a C64 because of memory limitations. In short, you'll find your 1764 RAM Expansion Module a useful, convenient addition to your computer system.

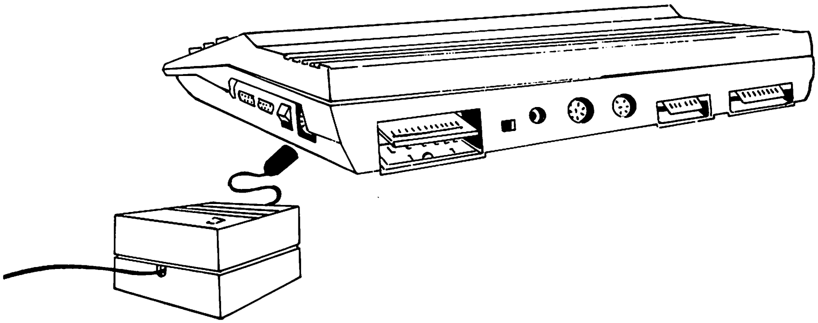
Included with your 1764, you'll find a software disk and a new power supply. The software disk includes utilities and demonstration programs designed to help you use the capabilities of the RAM Expander and demonstrate some ways to use the 1764 in programming. Section 2 tells more about the software on this disk.

The power supply is more powerful than the one that originally came with your computer. Since the RAM Expansion Module hooks directly into the 64, it draws its power from the computer. The new deluxe power supply is needed to provide that extra power.

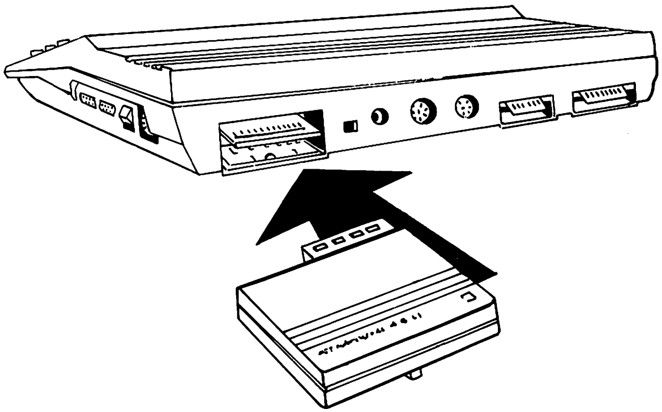
Before you use the 1764, change the power supply you have hooked to the computer. Plug in the deluxe power supply that came with the 1764 in place of the original power supply that came with your computer. The new power supply will provide the extra power your computer needs for the 1764. You can use this power supply all the time; you won't need to hook up the old one again.

Connecting the 1764 RAM Expander

1) Turn off your computer and all peripherals.
2) Replace old power supply with the new one if you have not done so already.



3) Insert the 1764 into the Expansion port on the back of your Commodore 64 or 64C.



Remember the 1764 was designed only for the 64, and can NOT be used with the Commodore 128.

4) Turn the computer and all peripherals back on.

Your expansion module is now ready to use.

Section 2: 1764 Test/Demo Software

Even if you know nothing about writing programs or memory maps, you can do a lot with your RAM Expander by running the programs on the Test/Demo software disk. You can install the RAM Expander, the RAM operating system, transfer information from disk into RAM, run animations, use your 1764 with the GEOS software package, and more.

The software disk that came with your RAM expander contains programs for the following applications:

- RAMDOS
- DOS wedge
- File copying
- Change unit numbers
- Animated sequences (pound and globe)
- Diagnostic testing
- GEOS configuration (on the flip side of the disk)

All of these programs are explained in this section except RAMDOS and GEOS configuration.

The RAMDOS is an elaborate 1764-based disk operating system. It is explained in detail in Section 3 of this manual.

The GEOS configure program is not part of the demo software. It is located on the flip side of the disk. This is used with the GEOS operating system, and creates an icon for the RAM expander so it can be easily used with GEOS. There is a separate set of instructions (included in this box).

The rest of the programs can be loaded from the Startup Menu.

Loading the Startup Menu

The best way to get started with your RAM expander is to load the Startup Menu from the disk, which lets you select any of the programs by typing:

```
LOAD"*",8 <RETURN>
RUN       <RETURN>
```

The program to load the Startup Menu is the first program on the disk. The Startup Menu lets you automatically load the specific program you select when you enter a number. The Startup Menu options:

1 INSTALL RAM DISK
2 INSTALL DOS WEDGE
3 FILE COPY UTILITY
4 CHANGE UNIT NUMBERS
5 RAM-PAK TEST PROGRAM
6 RAM-PAK POUND DEMO
7 RAM-PAK GLOBE DEMO
8 QUIT

Installing the DOS Wedge

The DOS wedge adds commands to C64 BASIC 2.0. These commands make it easier to display directories, send instructions to disk drives, and manipulate files. All commands in the DOS wedge may be used with the 1541/1571 series of disk drives, as well as the 1764 RAMDOS.

You can load the DOS wedge by selecting Option 2 from the Startup Menu. After you select option 2, the wedge boot message and help screen appear.

The help screen lists syntax for commands you can use. This typical form for a DOS command is:

Character <Unit #,> <Filename>

The unit number is optional. If there is no unit number specified in the command, the last unit number referenced is assumed by the DOS.

The wedge symbols you can use and examples of syntax are displayed on the help screen.

DOS Wedge Technical Information

The DOS wedge replaces DOS 5.1, which was located at $CC00-$CFFF in the C64 memory. The 1764 DOS wedge is located at $CC00-$CEFF. It interfaces via IERROR vector in BASIC 2.0. Note that DOS 5.1 is incompatible with 1764 DOS.

File Copy Utility

Option 3 is a very handy utility, which you can use to copy files from your disk drive to RAM, from RAM to disk drive, or from one disk drive to another.

The file copy reads a disk drive file (or files), and then stores the block of information in the RAM expander. That information can now be used immediately when you recall the file. The file still remains on disk, but is now accessible within the computer's memory. Remember that the file is available, but not automatically loaded up when it sits in RAM. You still have to load and run the file as you would from a disk drive. The difference is the speed at which the program is loaded.

Note: This program can copy PRG, SEQ, REL and USR files.

Load the file copy program by selecting option 3 from the Startup Menu.

At the first two prompts, specify your SOURCE unit number and DESTINATION unit number. If you are copying a file from a disk in the drive to the RAM expander memory, your source is unit 8 and destination is unit 9 (assuming you installed the 1764 as unit 9).

Then you are prompted:
ENTER TEMPLATE OR <RETURN> ?

If you press the <RETURN> key, the entire directory is read, and you are asked, one file at a time, whether you want to copy that particular file, Answer Y or N for each. After the last file, you are asked CONTINUE? (Y/N): If you still wish to copy the files, answer Y. If you've changed your mind, answer N.

Instead of pressing <RETURN>, you could enter a template, either a wild card name (like BAS.*) or an asterisk. Typing * loads everything from the directory and skips to the CONTINUE? prompt. Answer N to abort, or answer Y to copy the entire contents of the disk into RAM.

After the copying is completed, you are prompted MORE? (Y/N) for additional files to copy.

Change Unit Numbers

Sometimes you'll need to change the unit numbers of the 1764 or disk drive. Even though you can choose the unit number when you install the 1764, you may find it necessary to change to a different number. You may want the RAM expander to be unit number 8 when you no longer need to use a disk drive.

Option 4 of the Startup Menu is used to change unit numbers. It's pretty straight-forward: you are asked for the old number of the unit you want to change, then the new number. The change is almost instant, and a line describing the unit type and DOS. An initialization message (DOS error #73) lets you know the change is complete.

You are prompted for additional changes. If you answer N, you exit the program.

Tutorial Example

The file copy program provides you with one way to store data in the RAM expander for your convenience. For example, you can load a game into the 1764 by copying the files. Some games utilize the unit number 8 in their code, and you must re-number the 1764 as unit 8 after you copy the file. Here's the procedure to follow to load such a game:

Install the RAM expander as unit 9 using option 1 of the Startup Menu. Answer Y to initialize the 1764.

Load the File Copy Utility by selecting option 3 from the Startup Menu.

When the prompt ENTER TEMPLATE OR PRESS <RETURN> is displayed, change the disk in the drive to the one you wish to load. Then type * <RETURN> to copy every program on the disk.

There is a brief pause while the directory of the disk is read from the drive into the RAM. At the prompt CONTINUE? (Y/N), answer Y, which begins the copying process. Had you answered N, no files would have been copied.

After the programs have been stored in the RAM expander, answer N to the MORE? prompt, unless there are additional files you want to copy.

Now do a directory for unit 9 (to make sure the programs are all in RAM)

Change disks back to the 1764 Test/Demo Disk.

If the unit number is referenced in the program, there's one more change you have to make. Since the 1764 is installed as unit 9, that is where the program is now. However, many programs reference unit number 8 in the code. In this instance, change the unit number of the 1764 from 9 to 8, using option 4 of the Startup Menu.

Perform a directory for unit 8 (which is now the 1764). You should get the same listing as you did when you ran a directory for unit 9.

Load and RUN the program from the RAM expander, and it will begin almost instantly.

1764 Diagnostic Test

Option 5 is a test to check if the 1764 is operating properly. The test wipes out everything in RAM storage, and you are given a chance to abort the test. It takes about one minute for the test to be completed. When the test is done and the 1764 passes, the screen looks like this:

```
                256K EXPANSION RAM TEST
 4        PASS       55AA                         0
 4        PASS       55AA                         1
 4        PASS       55AA                         2
 4        PASS       55AA                         3

 7       PASS      0000      0000      1000      00
 8       PASS      0000      3000      0400      00
 9       PASS      0000      3000      0400      00
 A       PASS      0100      0100      0001      00
```

After the test is completed, you may re-run it by pressing the space bar, or exit by pressing RUN/STOP. Be careful not to press RUN/STOP while test is running. Full explanations of each of these diagnostic tests can be found in Appendix B of this manual.

The Animation Demos

The disk contains two animation demos, which you can select from options 6 and 7 of the Startup Menu. Both programs are documented, so when you list them, you can see for yourself how the cycling of images is done.

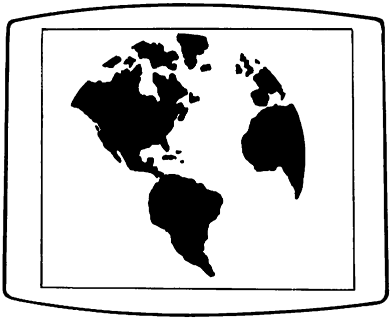
Option 6 is the POUND demo, consisting of 32 separate images which are stored and cycled from RAM. GLOBE, option 7, features 36 images. Both demos work similarly.

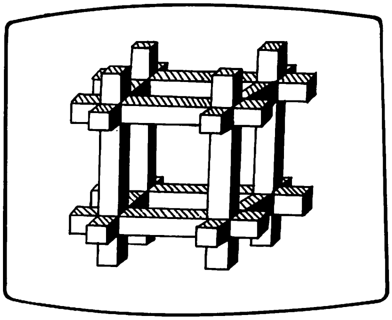
DO NOT use option 1 to install the RAMdisk before running either option 6 or 7. Prompts to install the RAMdisk are included within these programs. If you install the RAMdisk using option 1, there will not be enough space left in RAM to load all the images in POUND and GLOBE.

So instead, select option 6 or 7 from the Startup Menu, and when you are prompted LOAD RAM? (Y/N), so answer Y.

A message then appears on the screen, telling you which image is being loaded. When all the images are loaded, the animation starts.

Here are what each of the demos should look like on your monitor screen:





If you halt the demo by pressing RUN/STOP RESTORE, you can start it again by typing 'RUN' <RETURN>. Answer 'N' to the LOAD RAM? (Y/N) prompt.

Quitting the Startup Menu

Select option 8 to exit from the Startup Menu and back into BASIC.

Section 3: RAMDOS

The 1764 RAMDOS is software which allows the 1764 to emulate a disk drive. The software is installed into the C64 kernel, causing the C64 to treat the RAM Expander as an additional disk drive. This dramatically increases the speed of disk operations.

RAMDOS is a compatible subset of the 1541/1571 disk drive operating system. In effect, RAMDOS is a DOS without disks. The RAMDOS consists of two programs, which in effect allow you to use the 1764 as a disk drive. It is much faster, however, because once the program is held in the extra RAM made available by the 1764, it's no longer necessary to access the disk drive; every part of the program is instantly available.

Some applications (especially games) may not work with the RAMDOS installed. This is because the games depend on the physical aspects of the disk drive, sometimes executing proprietary code within the disk drive itself.

Those applications that directly read and write tracks and sectors or attempt to execute proprietary code within the 1541/1571 disk drive cannot be run properly with 1764 RAMDOS.

RAM Expander Installation

You initiate RAM DOS by selecting option 1 from the Startup Menu, installing the RAM Expander.

After the program loads, you are asked for a unit number to give the RAM expander. Normally, you will install the RAMdisk as unit 9, which is the default. This allows you to use the filecopy utility on the Test/Demo disk to copy files into the RAMdisk from a normal disk drive (which would be unit 8). You can always change the unit number of the RAMdisk to 8 later, using the change unit number option of the Startup Menu program.

While it is possible to make the RAM expander unit 8 (assuming you are currently using a 'real' unit 8 drive), you should realize that unless you have previously placed programs on the 1764 or have a second 'real' drive, you will have nothing to load.

Next, you can specify where the RAMDISK INTERFACE PAGE is to be installed. Unless you have a good reason for putting it elsewhere in memory, accept the default, which is 207 ($CF00). See the technical information section for further details.

You are then prompted to initialize (clear) the RAM expander. Normally, you will answer yes (the default, which is Y) to initialize the 1764. If you wish to retain the information in the RAM expander, answer N.

Always initialize the RAMdisk after power-up. If your computer has crashed or been reset via external hardware, you may use this option to reinstall the RAM driver without losing any information in the RAMdisk. Once the RAMDOS is installed, you can press any key to return to the Startup Menu, unless the 1764 is installed as unit 8 (in which case the program simply ends).

Using the RAMDOS

The RAMDOS behaves like any normal disk drive. It has a directory file, and supports PRG, SEQ, REL and USR files. It has a command channel and an error channel, just like a 1541 or 1571 drive. However, not all 1541/1571 commands are supported, due to a different track and sector organization.

The following DOS commands are supported:

Rename	
Copy	(only one file; no concatenation)
Scratch	
New	
Init	(closes all files)
Validate	(returns OK)
Pointer	(used for REL files)
UJ	(same as init)
UJO	(full init)
UI	(same as init)
U0>chr$(n)	(set unit number, where n=unit number)

The following DOS commands are NOT supported:

B-R	Block-Read
B-W	Block-Write
B-A	Block-Allocate
B-F	Block-Free
B-P	Buffer-Pointer
M-R	Memory-Read
M-W	Memory-Write (see note below)
M-E	Memory-Execute

Note: The M-W command may be used to change the unit number of the disk drive by using it to write to locations 119 or 120. The RAMDOS designates this single area to write to. Attempts to write to different areas in RAMDOS memory are trapped as syntax errors. Note that the colon is mandatory.

1541/1571 DOS Emulation

The following 1541/1571 operations are emulated by 1764 DOS.

- The DOS supports fifteen concurrent channels, specified by the secondary address. Channels 0 and 1 are treated like channels 2-14. Channel 15 is the command/error channel.
- All accesses to the directory file are treated as the interpreted version, which has a memory format similar to a BASIC program (i.e., line numbers and links are included).
- Wild cards are supported.
- Save with replace is supported.
- DOS supports file level activity. Direct access to tracks and sectors is not supported.
- PRG, SEQ, REL, and USR file types are supported.
- The DOS interface to the kernel is through the standard indirection vectors for the following standard kernel calls:

OPEN, CLOSE, CHKIN, BASIN, BSOUT, GETIN, LOAD SAVE, INMI

RAMDOS Software Technical Details

The RAMDOS software comes in two parts:

- RAMDOSxxxxxx.BAS
- RAMDOSxxxxxx.BIN

where xxxxxx represents the date the software was generated at Commodore.

RAMDOSxxxxxx.BAS is a small BASIC program that guides the user through the process of installing the RAMDOS into the C64 kernel. It provides workarounds for more advanced users should software incompatibility problems arise.

RAMDOSxxxxxx.BIN is the actual RAMDOS binary code which is loaded into the C64 RAM by the RAMDOSxxxxxx.BAS program. After loading, it is transferred to the REC (RAM Expansion Controller; see Section 4 for detail), thereby freeing up C64 memory for other uses.

DOS Operation

DOS installation overwrites system memory. Once the DOS is installed, that system memory is again free for usage, with two exceptions:
1) Indirection vectors—These vectors now point to the DOS interface block, and the DOS interface block contains the vectors' former values.

2) DOS interface block—This page of code may be installed on almost any page in the system provided it is not under any ROMs. The default is the page starting at $CF00 (HEX).

Execution of System I/O Calls

All system I/O calls are made through the indirection vectors which point to the DOS interface block. Code in the interface block determines whether the call is for the 1764 or another system device. If the call is not for the 1764, control is passed on according to the values of the indirection vectors prior to DOS installation.

If the call is for the 1764, code in the interface block swaps the DOS with some normal system RAM and passes control to the DOS. The DOS then processes your call (which can involve several swaps of DOS and user RAM), which upon completion returns control to the interface block.

The interface block swaps out the DOS, restoring the user's memory map, and returns control to the caller.

Interrupt Operation

The DOS never creates interrupts. During the time the DOS is swapped in, IRQ's are disabled, to prevent DOS operation from interfering with your programs.

NMI's cannot be disabled. Any NMI code used by your programs should never access or reference data within the area under DOS control. To counter this, use a DOS image which does not use memory accessed by your NMI routine.

Note that the only NMI's caused by the system are for RS-232 and RUN/STOP RESTORE operation.

Concurrent DOS and RS-232 operation is not possible because of the time required by the swapping process, which is not interruptable. Do not attempt to use the RS-232 to receive or transmit while accessing the RAM expander.

If you wish to use the RS-232, be careful not to install the 1764 in memory used by the RS-232.

RUN/STOP RESTORE

RAMDOS intercepts this operation, therefore is unaffected when you press RUN/STOP RESTORE. During DOS calls, RUN/STOP RESTORE is disabled. You may have to press these keys several times to get it to work during intense disk activity.

Section 4: Programmer's Reference Notes

Memory transfers involving the 1764 are accomplished through the RAM Expansion Controller (the REC), which is an I/O device. You cannot directly access the RAM in the 1764. Instead, you instruct the REC to transfer data between the 1764 memory and C64 memory. This may be accomplished using POKEs or via machine language.

The demo programs and the ones you'll write set the REC registers directly. This type of memory transfer is known as Direct Memory Access (DMA). To see examples of this process, look at the programs on the demo disk.

Remember that you can use either DMA or RAMDOS—they are NOT compatible.

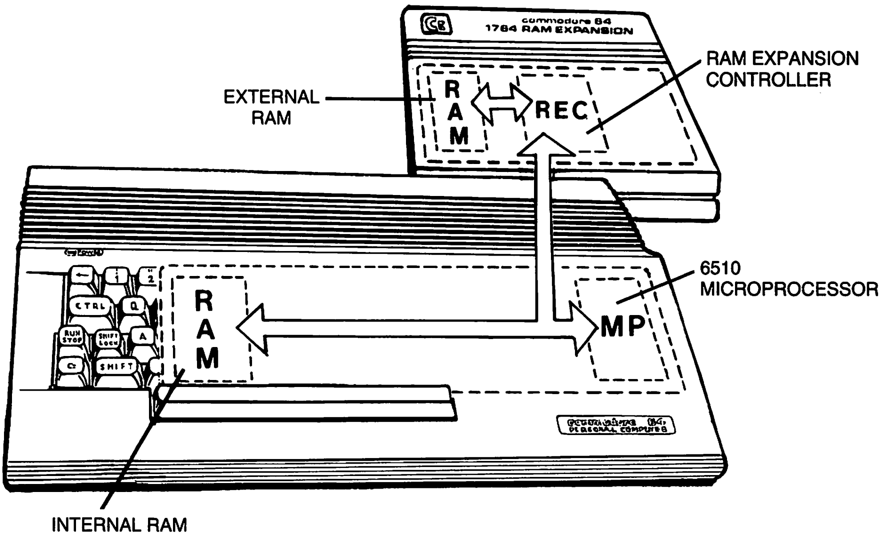


Figure 4.1. Illustration of REC function

Sample Program: Storing and Retrieving a Text Screen

Here's a functional example of how to change the REC register values in a program to perform memory manipulations using the 1764. This program lets you save and retrieve a screen of text to and from the external memory.

```
10 REM SAVE TEXT SCREEN                        (NOTES)
20 BA=13*4096+15*256            ($DF00 = REC IN I/O SPACE)
30 POKE BA+2,0:POKE BA+         ($400 = C64 TEXT SCREEN)
   3,4:REM CPU ADDRESS
40 POKE BA+4,0:POKE BA+         ($000 = EXPANSION RAM ADDRESS)
   5,0:REM DISK ADDRESS
50 POKE BA+6,0:REM DISK         ($000 = EXPANSION RAM BANK)
   BANK
60 POKE BA+7,0:POKE BA+         (1024 BYTES = SIZE OF SCREEN)
   8,4:REM NUMBER OF BYTES
70 POKE BA+9,0:REM RESET INTERRUPT CONTROL BITS
80 POKE BA+10,0:REM INCREMENT BOTH REGISTERS
90 POKE BA+1,252:REM SAVE SCREEN
100 PRINT 'THE SCREEN IS SAVED'
110 PRINT 'PRESS A KEY TO RETRIEVE'
120 GET A$:IF LEN(A$)=0 THEN GOTO 120
130 POKE BA+2,0:POKE BA+3,4:REM CPU ADDRESS
140 POKE BA+4,0:POKE BA+5,0:REM DISK ADDRESS
150 POKE BA+6,0:REM DISK BANK
160 POKE BA+7,0:POKE BA+8,4:REM NUMBER OF BYTES
170 POKE BA+1,253:REM READ SCREEN
```

Line 20 specifies the BASIC address for the beginning of the REC registers. The rest of the program utilizes this value in the POKE statements. Line 30 POKEs the address of the text screen. Lines 40 and 50 set the RAM expander address and bank, respectively. Line 60 POKEs the number of bytes to transfer, which is 1024 for the entire screen of text (only 1000 are actually used for the text screen). Line 70 resets the interrupt control bits. Line 80 specifies both C64 and REC address registers are to increment, and line 90 actually performs the 'save,' storing the text screen in the 1764 memory area.

Lines 130 to 160 are preparation for recalling the screen from RAM, and line 170 'loads' the screen.

Memory Locations

In order for a safe transfer, you must be aware of the C64 memory map. You are moving blocks of data in and out of main and expansion memory, so you must know the correct addresses to retrieve the blocks of data you intend to and store them in the proper places. You'll have to be careful not to put things in memory where they don't belong, e.g. over another program or memory allocated for a hardware function. If you transfer data into an area that is already occupied by a program or other data, you will overwrite the original data, which may affect the success of the transfer.

All the standard rules for C64 banking and I/O apply for the 1764 as well. For example, you may have to swap out the ROM to use the RAM, to

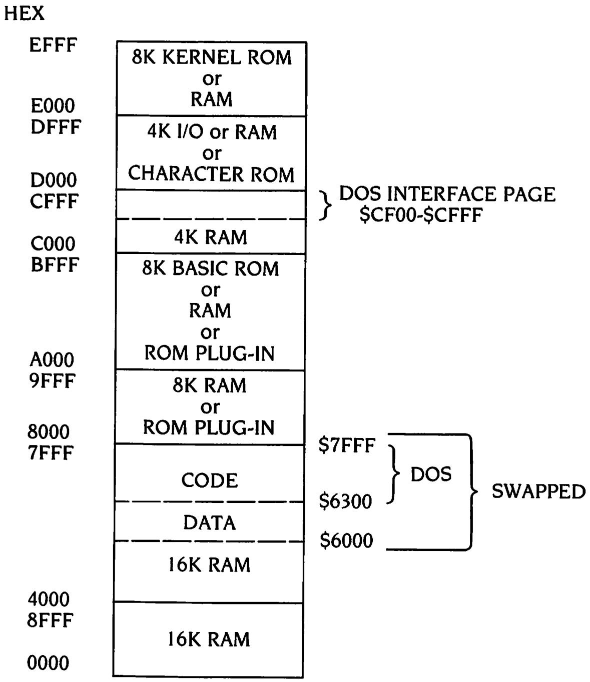


Figure 4.2 C64 Memory Map

avoid a bleed-through if you try to write to ROM. The Commodore 64 Programmer's Reference Guide thoroughly explains these topics.

REC Operation

The REC has four primary operating modes:
1) Transfer a block of data from main memory to expansion memory
2) Transfer a block of data from expansion memory to main memory
3) Exchange a block of main memory with a block of expansion memory
4) Verify a block of main memory with a block of expansion memory

The REC has several internal registers which can be used to set up a particular operating mode. Modes can be selected by setting the appropriate bits in the command register (see Figure 4.3). The starting C64 address, expansion RAM address, expansion RAM bank and number of bytes are all programmable values.

Direct Memory Access Operation

Direct Memory Access (DMA) is the process the REC uses to transfer data to and from the computer's memory. During DMA, the C64's computer processor is temporarily halted so the REC may access the computer's memory.

There are two different events which cause the REC to begin a transfer sequence.

1) If the FF00 option in the command register ($01) is disabled, the DMA begins right after the execute bit is set in the command register.

2) If the FF00 option is enabled, the DMA begins immediately after a write to address FF00 is detected - providing the execute bit was previously set in the command register.

The prupose of the FF00 option is to allow the REC to access the RAM beneath I/O space. The REC registers are I/O mapped on the C64 expansion bus, which means that, without the FF00 option, I/O space would be enabled when DMA was initiated. This option, therefore, allows the user to bank out the the C64 I/O space, replacing it with RAM, before the DMA takes place. The FF00 option is cleared each time it is used.

REC Command Register

ADDRESS	BITS	FUNCTION
$00	7-0	Status Register — Read Only 7 - Interrupt Pending 1 = Interrupt waiting to be serviced 6 - End of Block 1 = Transfer complete 5 - Fault 1 = Block verify error 4 - Size 1 = 256K 3-0 - Version Note: Bits 7-5 are cleared when this register is read
$01	7-0	Command Register — Read/Write 7 - Execute 1 = Transfer per current config. 6 - Reserved 5 - Load 1 = Enable AUTOLOAD option 4 - FF00 1 = Disable FF00 decode 3 - Reserved 2 - Reserved 1 - Transfer type 00 = Transfer C64 > RAM module 01 = Transfer C24 < RAM module 10 = Swap C64 <> RAM module 11 = Verify C64 — RAM module
$02	7-0	C64 Base Address, LSB — Read/Write Lower 8 bits of base address, C64
$03	7-0	C64 Base Address, MSB — Read/Write Upper 8 bits of base address, C64
$04	7-0	Expansion RAM address, LSB — Read/Write Lower 8 bits of base address, expansion RAM
$05	7-0	Expansion RAM address, MSB — Read/Write Upper 8 bits of base address, expansion RAM
$06	2-0	Expansion RAM bank — Read/Write Expansion RAM bank pointer. Bits 2 (MSB) to 0 (LSB) are significant
$07	7-0	Transfer Length, LSB — Read/Write Lower 8 bits of the byte counter
$08	7-0	Transfer Length, MSB — Read/Write Upper 8 bits of the byte counter
$09	7-5	Interrupt Mask Register — Read/Write 7 - Interrupt enable 1 = Interrupts enabled 6 - End of Block mask 1 = Interrupt on end of block 5 - Verify error 1 = Interrupt on verify error
$0A	7-6	Address Control Register — Read/Write 0,0 = Increment both addresses (default) 0,1 = Fix expansion address 1,0 = Fix C64 address 1,1 = Fix both addresses

Figure 4-3

Note: The base address of the REC registers in the C64 memory map is DF00 in I/O space. The addresses of the REC registers can be obtained by adding the value in the Address column in the above figure to the base address of DF00.

Special Features

AUTOLOAD—When you select this option in the command register, the C64 base address registers, the expansion memory base address registers, the expansion memory bank and the byte counter registers at the end of a transfer are automatically reloaded.

This is useful if one operation is to be executed repeatedly on one particular block of data. Note that if AUTOLOAD is selected in verify mode, the address where the verfiy error occurred is lost. Ordinarily, upon finding a verify error, the REC halts the DMA cycle and both address registers and the bank register point to one location above the address that failed.

Address Control—Another special feature of the REC is the ability to hold a source and/or destination fixed in any mode. This is accomplished by setting the appropriate bits in the address control register ($0A). The default is that addresses will increment for both source and destination. It is possible, with a fixed C64 address, to DMA to any I/O device.

Interrupts—An interrupt can be generated on an end of block condition or verify error by enabling interrupts and setting the corresponding mask bits in the interrupt mask register ($09). In either case, the corresponding flag is set in the status register. Interrupt flags are cleared upon reading the status register. The interrupt mask register bits stay as they were before the interrupt/clear sequence. For proper operation, the status register ($00) must be read at least once before a DMA is initiated with interrupts enabled. It is important to clear out the previous status before any mode is used with the interrupt option.

Operation Details

If a specified address range exceeds the address range in the selected expansion bank, a 'wrap' occurs into the next bank. This happens whenever the given byte counter exceeds the number of bytes left in the selected bank. The C64 side always wraps to the beginning of the same bank.

Under normal operation (no AUTOLOAD, addresss increment), both address registers point to the next sequential memory location outside the selected transfer range at the end of the transfer. This is true for any mode and applies to both base address pointers except one that is held fixed. The bank pointer is also updated (if necessary) at the end of the transfer. Also note that under normal operation, the byte counter decrements to the value 1. Care should be taken, therefore, to check the transfer complete bit in the status register and not rely on the byte counter value to indicate an end to the transfer condition. A byte counter of 0 results in a transfer of a full 64 K bytes. Again, wrapping occurs in all modes of operation.

Appendix A: 1764 Diagnostic Programs

The diagnostic test program runs a series of tests, then passes or fails the 1764 based on its performance.

TEST 1 Checks all memory on the expansion RAM. Four data patterns, $00, $AA, $55 and $FF, are written in every memory location. The verify mode of operation checks the data transfer.

TEST 2 To check the RAM expander memory, a checker pattern, $AA and $55, is written into alternating memory locations. All locations of every bank are checked.

TESTS 3 and 4 This checks the capabilities for data transfer from the RAM expander to the C64. Data patterns of $FF's are stored in C64 memory in the following areas:

```
$0002—$004F
$0080—$0136
$0200—$03FF
$0800—$7FFF
$C000—$CFFF
$E000—$FEFF
```

Data transfers from the RAM expander to the C64 memory fill locations $1000-$2FFF and $3000-$5FFF with the alternating $55 $AA data patterns. A thorough check of all filled memory in the C64 checks for RAM corruption in the computer memory. Test 4 repeats this procedure, using $00 pattern for background fills before the data transfer occurs.

TEST 5 Checks the AUTOLOAD feature of the RAM expander. After a data transfer, the RAM expander registers should reset themselves to the setting prior to the transfer. This test checks the Base Address Register, the RAM Expansion Base Register, the Byte Transfer Length Register and the RAM Bank Register.

TEST 6 The previous test checks 3 of the 4 transfer modes; this test checks the remaining transfer mode, RAM expander to C64 transfer with fixed expander address. A 1K byte transfer occurs to C64 locations

$3000-$33FF. The data should be the same in all these locations, indicating the RAM expander read from the proper locations.

TEST 7 This checks the $FF00 decode option selectable in the command register. The data transfer should not occur until a WRITE TO $FF00. After a time out subroutine executes, the program checks to see that the transfer did not occur before the WRITE TO $FF00.

TEST 8 This checks the interrupt generating capability of the RAM expander. If an interrupt is generated at the end of the block transfer, the status register interrupt pending bit goes high.

COMMODORE ™

Commodore Business Machines, Inc.
1200 Wilson Drive • West Chester, PA 19380

Commodore Business Machines, Limited
3470 Pharmacy Avenue • Agincourt, Ontario, M1W 3G3

312528-01

PRINTED IN U.S.A.