Kernal Routines

A section of the Kernal from FF81/FF8A through FFF5 contains JMP instructions for the 39/36 routines that have been defined by Commodore as the user-callable Kernal routines. Note that there are three more for the 64 than for the VIC.

These JMP instructions are intended to allow you to write programs that use any of these Kernal functions without having to wonder if they will still work on later Commodore computers or if the Kernal ROM is modified on the 64 or VIC. Thus, if a routine moves to a different memory location, you need not be concerned if you just JSR to the Kernal entry which is in the JMP instruction table.

Another reason to use the standard jump instruction table is that you don't have to be concerned with the internal workings of the routines. Rather, you just provide the information that is needed by the routine. The routines execute the function you request, returning information if that is part of the function and providing a means of error detection, either through the carry flag and the accumulator or the status byte, location 90.

Since these jump instructions are a standard feature of Commodore Kernal ROMs (at least on the VIC and 64), you may wonder why anyone would not use the jump instructions. Some reasons follow. If the jump vector does an absolute JMP (such as at FFB1, where there is a JMP ED0C/EE17 for the send-listen-with-attention-to-serial-device function), you cannot modify this serial function if you use the standard jump entry.

If you want to provide an additional feature that precedes or follows the standard routine, you may find using the Kernal jump instructions awkward.

For the jump table entries that jump indirectly, such as CHRIN, which does a JMP (0324), you can just change the vector at 0324 to point to your own routine. However, not all the jump instructions use indirect JMPs. See the table below for details of which use indirect jumps and which use absolute jumps. Perhaps the reason Commodore doesn't use indirect jumps for all the vectors is the lack of free space in pages 0, 2, and 3.

A jump vector that appears unused is the SETTMO routine. Also, the RAM vector at (032E) is not called.

The following list compares the Kernal RAM vector table starting at 031A to the way this function is called from the Kernal jump instruction table.

Kernal RAM Vectors and Kernal Jump Table Usage

Vector	Function	Jump Instruction
(031A)	OPEN	JMP (031A)
(031C)	CLOSE	JMP (031C)
(031E)	CHKIN	JMP (031E)
(0320)	CHKOUT	JMP (0320)
(0322)	CLRCHN	JMP (0322)
(0324)	CHRIN	JMP (0324)
(0326)	CHROUT	JMP (0326)
(0328)	STOP	JMP (0328)
(032A)	GETIN	JMP (032A)
(032C)	CLALL	JMP (032C)
(032E)	USRCMD	Not called
(0330)	LOAD	JMP F49E/F542
(0332)	SAVE	JMP F5DD/F675

This chart shows that only LOAD and SAVE do not execute a JMP indirect through a vector upon entering the Kernal table. Both LOAD and SAVE jump to routines which perform some setup operations before doing the JMP indirect through their vectors. LOAD sets the vector at (C3) as the starting address for the LOAD from the X and Y registers before doing a JMP (0330). SAVE sets (AE) to the end of the save area +1 from the X and Y registers, and (C1) to the start of the save area from a page 0 location whose address is in the accumulator, then SAVE does a JMP (0332). If you modify LOAD or SAVE, be aware of this preparation. There seems to be little reason why Commodore doesn't just do a jump indirect for SAVE and LOAD as they do for the other functions that have a RAM vector. If you are writing your own LOAD or SAVE, and you're still doing the JMP FFD5 for LOAD and JMP FFD8 for SAVE, you may not want (C3), (AE), or (C1) modified before they get control.

Another reason for not using the Kernal jump instruction is if it points to a routine that functions incorrectly. For example, the Kernal jump instructions for CHROUT and CHRIN that can be used for RS-232 operations do not function correctly for RS-232 x-line handshaking on the VIC.

A third reason not to use the jump instruction tables is if you want to use a section of Kernal ROM that doesn't have a Kernal table entry. For example, you can create autoload/autostart machine language tapes, but only by directly calling the Kernal ROM routines, since there are no JMP vectors to the necessary routines. Another example is if you want to use the screen editor routines from your machine language program, which are not available through the Kernal jump vectors.

In summary, the Kernal jump instructions do have a purpose, and you should use them when appropriate. However, the jump instructions cannot or should not always be used, and after reading the rest of this book you may find Kernal ROM routines that you want to use directly without using the jump instructions.

Commodore intends the jump instructions to provide you with a bridge to a different version of a Kernal ROM without your having to rewrite your machine language program. While this jump instruction compatibility is true for VIC-to-64 translations, it is not true for 64-to-VIC translations because there are three new 64 jump instructions that are not provided in the VIC.

The alphabetical table of jump instructions below should be useful when you are programming. The *Operand* column lists the place to which the JMP transfers control, either an absolute location or a location pointed to by an indirect vector. *Default* gives the address found in the RAM indirect vector if it has not been modified.

Kernal Jump Table

Name	Address	Operand	Default	Description
ACPTR	FFA5	EE13/EF19		Get a byte from serial bus
CHKIN	FFC6	(031E)	F203/F2C7	Open channel for input
CHKOUT	FFC9	(0320)	F250/F309	Open channel for output
CHRIN	FFCF	(0324)	F157/F20E	Get a byte from input channel
CHROUT	FFD2	(0326)	F1CA/F27A	Send a byte to output channel
CIOUT	FFA8	EDDD/EEE4		Send a byte to serial bus
CINT	FF81	FF5B (64 only)		Initialize screen editor
CLALL	FFE7	(032C)	F32F/F3EF	Close all channels and files
CLOSE	FFC3	(031C)	F291/F34A	Close logical file
CLRCH	FFCC	(0322)	F333/F3F3	Reset I/O channels
GETIN	FFE4	(032A)	F13E/F1F5	Retrieve character from channel
IOBASE	FFF3	E500		Return base address of I/O registers
IOINIT	FF84	FDA3 (64 only)		Initialize I/O devices
LISTEN	FFB1	ED0C/EE17		Send listen with attention to serial devices

Name	Address	Operand	Default	Description
LOAD	FFD5	F49E/F542		LOAD/VERIFY to RAM
MEMBOT	FF9C	FE34/FE82		Read or set the start-of-memory pointer
MEMTOP	FF99	FE25/FE73		Read or set the end-of-memory pointer
OPEN	FFC0	(031A)	F34A/F40A	Open logical file
PLOT	FFF0	E50A		Read or set cursor location
RAMTAS	FF87	FD50 (64 only)		Memory initialization
RDTIM	FFDE	F6DD/F760		Read jiffy clock into registers
READST	FFB7	FE07/FE57		Read or reset status
RESTOR	FF8A	FD15/FD52		Reset RAM vectors to default
SAVE	FFD8	F5DD/F675		Save contents of memory to device
SCNKEY	FF9F	EA87/EB1E		Detect keyboard entry
SCREEN	FFED	E505		Return number of columns and rows
SECOND	FF93	EDB9/EEC0		Send secondary address after listen to serial
SETLFS	FFBA	FED0/FE50		Set logical file number, device number, and secondary address
SETMSG	FF90	FE18/FE66		Set message control
SETNAM	FFBD	FDF9/FE49		Establish filename
SETTIM	FFDB	F6E4/F767		Set jiffy clock from registers
SETTMO	FFA2	FE21/FE6F		Set IEEE time-out
STOP	FFE1	(0328)	F6ED/F770	Test for STOP key
TALK	FFB4	ED09/EE14		Send talk with attention to serial devices
TKSA	FF96	EDC7/EECE		Send secondary address after talk to serial
UDTIM	FFEA	F69B/F734		Increment jiffy clock
UNLSN	FFAE	EDFE/EF04		Send unlisten to serial
UNTLK	FFAB	EDEF/EEF6		Send untalk to serial
VECTOR	FF8D	FD1A/FD57		Read or set RAM vectors

The following sections discuss the above jump instructions. Some routines used by these jump instructions are included here, if they are not discussed elsewhere. For example, the LOAD and SAVE routines that are jumped to are discussed here, since these routines can be used for both serial devices and tape. The specific parts of LOAD and SAVE that apply to serial or to tape are discussed in those sections. Routines, such as those jumped to by TALK, SECOND, or TKSA, that are specific to one device or topic are discussed in those sections.

ACPTR
FFA5

Called by:
None.

Setup routines:
TALK, TKSA

The vector is JMP EE13/EF19. At EE13/EF19 the computer goes through a handshake sequence with the serial bus. During this sequence, the EOI handshake is performed if the serial clock input line does not go low within 250 microseconds as expected. If the EOI handshake sequence is performed, the routine sets the EOI status in the I/O status word, location 90. It can set the time-out status in the status word if serial clock in fails to go low within a certain time range.

If the preparation to receive handshaking signals detects no problems, and if the eight bits are received without handshaking error, the routine returns the byte received in the accumulator.

Exit conditions:
The accumulator contains the byte received from the serial bus.

CHKIN
FFC6

Called by:
JSR at E11E/E11B in BASIC's Set Input Device.

Setup routines:
OPEN

Entry Requirements:
The X register should contain the logical file number.

JMP (031E) with a default of F20E/F2C7. If the logical file is in the logical file number table, the routine obtains the device number and secondary address for this logical file from the corresponding entries in the device number and secondary address tables. If the logical file is not in the logical file number table, it displays FILE NOT OPEN, and returns with carry set and accumulator set to 3.

If the current device is the screen or the keyboard, the routine stores 0 for the keyboard or 3 for the screen in 99, the location holding the device number of the current input device. You don't have to use OPEN and CHRIN to input from the keyboard.

If the current device is the tape, the routine also checks the secondary address. If the current secondary address is not

$60, the routine displays the NOT INPUT FILE message, and returns with carry set and accumulator set to 6. If the current secondary address is $60, then location 99 is set to 1 to make tape the current input device. OPEN does an ORA $60 of the secondary address.

If the current device is a serial device, it opens the input channel by sending a TALK command to the device, and sending the secondary address if the value for secondary address held in B9 is < 128 (decimal). If the serial device does not respond, it displays the DEVICE NOT PRESENT error message and returns with carry set and accumulator set to 5. Otherwise, it stores the serial device number in 99.

If the current device is RS-232, the routine opens an RS-232 input channel. This RS-232 routine sets the current input device, location 99, to 2 for RS-232, then handles either the 3-line handshaking or the x-line handshaking opening sequence.

CHKIN Execution
F20E/F2C7-F236/F2EF

Called by:
Indirect JMP through (031E) from Kernal CHKIN vector at FFC6.

If the current logical file passed in the X register is in the logical file number table, obtain its corresponding device number and secondary address from the device number and secondary address tables. If it is not in the logical file number table, exit with FILE NOT OPEN error message.

If the device is the screen or the keyboard, set location 99, the current input device number, from BA, the current device number, and exit.

If the current device is an RS-232 device, JMP to the Open RS-232 Device routine.

If the current device is a serial device, JMP to the Open Serial Input Channel routine.

If the current device is tape, see if the secondary address indicates reading from tape. If not, JMP to display the NOT INPUT FILE message.

Store the current device number in the input device number, 99, CLC, and exit.

Operation:

1. JSR F30F/F3CF to see if the logical file number in the X register exists. If the logical file passed in the X register is not in the logical file number table, JMP F701/F784 to FILE NOT OPEN error message, set accumulator to 3, set the carry, and exit.
2. JSR F31F/F3DF to set the current logical file number in B8, the current device number in BA, and the current secondary address in B9 from the tables for the logical file, device number, and secondary address.
3. If the current device, BA, is the keyboard (0), or the screen (3), branch to step 8.
4. If the current device number is > 3, the current device is a serial device; branch to F237/F2F0 to open a logical file for a serial device.
5. If the current device is an RS-232 device, JMP F04D/F116 to open an RS-232 logical file as an input channel.
6. If the current device is tape, see if the secondary address is $60. If the secondary address is $60, branch to step 8. The secondary address of $60 is set during the OPEN Execution routine when the secondary address is ORed with $60.
7. If the secondary address is not $60, JMP F70A/F78D to display the NOT INPUT FILE message and exit with the accumulator set to 6 and the carry set.
8. STA (the current device number is in the accumulator) into 99, the input device number.
9. CLC and RTS.

CHKOUT
FFC9

Called by:
JSR at E4AE/E115 in BASIC's Set Output Device.

Entry requirements:
Set X register to logical file number.

JMP (0320) with default of F250/F309. If the logical file is in the logical file number table, obtain the device number and secondary address for this logical file from the corresponding entries in the device number and secondary address tables. If the logical file is not in the logical file number table, display the FILE NOT OPEN message, and return with carry set and accumulator set to 3.

If the current device is the keyboard, display the NOT OUTPUT FILE message, and return with carry set and accumulator set to 7.

If the current device is the screen, just set 9A, the current output device, to 3, and exit. You do not have to call OPEN and CHROUT to display on the screen.

If the current device is tape, also check the secondary address. If the secondary address is not $61, display the NOT OUTPUT FILE message, and return with carry set and accumulator set to 7. If the current secondary address is $61, set 9A to 1 for tape. Note: OPEN does an ORA $60 of the secondary address.

If the current device is a serial device, open the output channel for a serial device. Do this by commanding the current device to listen. Then for secondary addresses < 128, set the serial attention output line high. If the serial device does not handshake as expected, display DEVICE NOT PRESENT, and return with carry set and accumulator set to 5. Otherwise, set 9A to the serial device number.

If the current device is RS-232, then open an RS-232 output channel. This routine sets 9A to 2, and then it handles the 3-line or x-line handshaking sequence.

CHKOUT Execution
F250/F309-F278/F331

Called by:
Indirect JMP through (0320) from Kernal CHKOUT vector at FFC9.

If the logical file number passed in the accumulator at entry is not in the logical file table, display the FILE NOT OPEN error message.

If the logical file is in the file number table, obtain the current device number and secondary address for this logical file.

If the device is the keyboard, display the NOT OUTPUT FILE error message.

If the device is the screen, store the device number in the output device number, 9A, and exit.

If the device is a serial device, branch to Open Serial Output Channel.

If the device number is 2 (RS-232), jump to Open RS-232 Output Channel.

If the device is tape, the secondary address must not be $60 because this indicates read from tape. If $60 is found, display the NOT OUTPUT FILE error message. If the secondary address is legal, set the output device number, 9A, to the value 1.

Operation:

1. JSR F30F/F3CF to see if the logical file number passed in the X register is in the logical file number table. If not, JMP F701/F784 to display the FILE NOT OPEN error message and return with 3 in accumulator and carry set, then exit.
2. JSR F31F/F3DF to obtain the current device number and the current secondary address from their respective tables.
3. If the current device is the keyboard, JMP F70D/F790 to display the NOT OUTPUT FILE error message, set accumulator to 7, set carry, and exit.
4. If the current device is the screen, store the device number in 9A, the output device number, then CLC, and exit.
5. If the current device is a serial device, branch to F279/F332 to Open Serial Output Channel.
6. If the current device is an RS-232 device, JMP EFE1/F0BC to Open RS-232 Output Channel.
7. If the current device is tape, the secondary address must not be $60 (read tape). If the secondary address is $60, JMP to NOT OUTPUT FILE error, set accumulator to 7, set carry, and exit. If the secondary address is legal, set the output device number, 9A, to 1 (tape).
8. CLC and RTS.

CHRIN
FFCF

Called by:

JSR at E112/E10F in BASIC's Input a Character.

Setup routines:

OPEN, CHKIN (not required in retrieving from keyboard).

JMP (0324) with default of F157/F20E.

If the current input device, 99, is tape, then return the next byte from the tape buffer. Also, read one byte ahead to

see if the next byte is zero, indicating end of file, and if true, set end-of-file status in 90.

If the current input device, 99, is a serial device, the accumulator returns the byte received over the serial bus. However, if there are any I/O status errors, return with accumulator set to $0D

If the current input device, 99, is RS-232, return with the next character from the RS-232 receive buffer. However, if the receive buffer is empty, the RS-232 routine on the VIC just loops until the receive buffer contains a character. The VIC can hang in an infinite loop if the RS-232 receive buffer never gets another character. If the receive buffer is empty on the 64, the routine returns with $0D in the accumulator.

If the current input device is the keyboard, each character typed (except for control characters such as the cursor keys) is displayed on the screen until the unshifted RETURN is entered. Once an unshifted RETURN is typed, reset the input routine to retrieve a character from this screen line. After each character is retrieved from the screen line, increment the pointer to the character being retrieved in this logical line. The screen POKE code is converted to the equivalent ASCII code, which is returned in the accumulator. If the end of the screen line has been reached, then return $0D, the ASCII code for a carriage return. The screen editor routines limit the size of a logical line to 80/88 characters. The way this CHRIN from the keyboard is typically used is to fill a buffer as BASIC does. BASIC calls the CHRIN routine to fill the BASIC input buffer at 0200. The BASIC routine keeps putting characters in the buffer until CHRIN retrieves a carriage return (ASCII $0D).

If the current input device, 99, is the screen, then return the ASCII code for the screen character in the current logical line pointed to by D3, the column the cursor is on. D3 is then incremented to point to the next character in the line. If D3 has reached the end of the line, return $0D signifying carriage return, and set D0 to 0 to force the next CHRIN to come from the keyboard.

When doing CHRIN from the keyboard, the keyboard routine uses this CHRIN from the screen once the carriage return has been entered. After processing the screen characters, the screen CHRIN then resets a flag at D0 to 0 to force input from the keyboard for the next CHRIN.

Exit conditions:
Accumulator holds byte returned from channel.

Determine Input Device
F157/F20E-F178/F22F

Called by:
Indirect JMP through (0324) from Kernal CHRIN vector at FFCF.

Call the appropriate character input routine based on the input device number, 99.

Operation:
1. If 99 is set to 0 (keyboard), save D3 in CA, save D6 in C9, and JMP E632/E64F (see chapter 7) to receive a character from the keyboard.
2. If 99 is set to 3 (screen), store 3 in D0, save D5 in C8, and JMP E632/E64F (see chapter 7) to receive a character from the screen.
3. If 99 is set to 2 (RS-232), branch to F1B8/F26F (see chapter 9) to receive a character from the RS-232 device.
4. If the value in 99 > 3 (serial), branch to F1AD/F264 (see chapter 8) to receive a character from the serial device.
5. If 99 is set to 1 (tape), fall through to F179/F230 to receive a character from tape.

CHROUT
FFD2

Called by:
JSR at E10C/E109 in BASIC's Output a Character, JSR at F135/F1EC in Display Kernal Message, JSR at F5C9/F661 in Display Filename, JSR at F726/F7A9 in Error Message Handler, JSR at F759/F7DC in Find Next Tape Header.

Setup routines:
OPEN, CHKOUT (not required if output device is the screen).

Entry requirements:
Accumulator should contain the character to be output, in CBM ASCII. JMP (0326) with a default of F1CA/F27A.

If 9A, the current output device, is the screen (3), the ASCII code is displayed on the screen unless the ASCII code is a screen control function (cursor key, DELete, INSerT, and so

on). If the character is a control code, the routine performs the action. If the ASCII code is a valid screen display code, the code is displayed on the screen at the current cursor position and then the cursor is advanced to the next position on the screen.

If the current output device, 9A, is a serial device, (> 3), then JMP EDDD/EEE4 to send the character to all open serial devices. When sending a character to a serial device, a one-byte buffer, 95, is maintained. If this buffer is empty, the character to be output is simply stored in the buffer. If the buffer already contains a character, the routine sends the character from the buffer onto the serial bus and stores the character to be output in the buffer. When the serial file is closed or the serial device is commanded to unlisten, the final byte in the buffer is sent.

If the current output device, 9A, is RS-232 (2), the character to be output is stored in the RS-232 transmit buffer, and transmission is started if this is the first byte to be sent.

If the current output device, 9A, is tape (1), store the character in the currently available position in the tape buffer and increment the index to the available position in the tape buffer. Once the index is set to 192, write the tape buffer to tape. Then set the first byte of the tape buffer to 2 (identification for a data buffer) and reset the index to point to the second byte of the tape buffer.

Although the character to be output is in ASCII code for output to the screen, this is not the case for RS-232, serial, or tape. For example, if you are storing bytes to tape containing a code other than ASCII, CHROUT will send them to the tape buffer. For the screen, though, the 64/VIC screen editor is set up to convert ASCII codes to screen codes or screen functions, and would not function well if you did not use ASCII.

Determine Output Device F1CA/F27A-F1E4/F28E

Called by:
Indirect JMP through (0326) from Kernal CHROUT vector at FFD2.

Call the appropriate character output routine based on the output device number, 9A.

Operation:
1. If the output device is the screen, JMP E716/E742 (see chapter 7) to output a character to the screen.
2. If the output device is a serial device, JMP EDDD/EEE4 (see chapter 8) to output a character to the serial device.
3. If the output device is an RS-232 device, branch to F208/F2B9 (see chapter 9) to output a character to an RS-232 device.
4. If the output device is tape, fall through to F1E5/F28F (see chapter 10) to handle CHROUT to tape.

CINT (64 only)
FF81

Called by:
None.

JMP FF5B to initialize the VIC-II (6567) chip registers, clear the screen, set the cursor pointer, initialize the screen line link table, set the PAL/NTSC flag, set value for CIA #1 timer A, enable interrupts for CIA #1 timer A, and start timer A.

This Kernal jump instruction is only available on the 64. The nearest equivalent on the VIC is to JSR E518 to set the VIC (6560–6561) chip registers, clear the screen, set the cursor pointers, and initialize the screen line link table. The 6560 and 6561 chips are, respectively, the NTSC and PAL versions of the VIC's video chip. The VIC equivalent of CINT for enabling the IRQ timer interrupt is to JMP FE39 to enable VIA #2 timer 1 interrupts and to set a timer 1 value.

Thus, a VIC version might be something like this:

```
....  JMP 02A1
02A1  JSR E518
02A4  JSR FE39
02A7  RTS
```

CINT, or its VIC equivalent, is only needed if you write an autostart cartridge program and need to use the screen editor or IRQ timer A/timer 1 interrupts. If no autostart cartridge exists, the 64/VIC performs the actions in CINT during system reset.

CIOUT
FFA8

Called by:
None.

Setup routines:
LISTEN, SECOND (if serial device requires a secondary address)

Entry requirements:
Accumulator should contain character to output. JMP EDDD/EEE4 to execute the Send Serial Byte Deferred routine.

When sending a character to a serial device, the routine maintains a one byte buffer at 95. If this buffer is empty, the character to be output is simply stored in the buffer. If the buffer already contains a character, the character from the buffer is sent onto the serial bus and the character to be output is stored in the buffer. When the serial file is closed or the serial device commanded to unlisten, the final byte in the buffer is sent. The character is sent to all open devices on the serial bus.

CLALL
FFE7

Called by:
JSR at A660/C660 in BASIC's CLR.

JMP (0322) with a default of F32F/F3EF.

Set 98, the number of currently open files, to 0.

If the current output device is a serial device, send an UNLISTEN command on the serial bus.

If the current input device is a serial device, send an UNTALK command on the serial bus.

Set 99, the current input device, to be the keyboard.

Set 9A, the current output device, to be the screen.

Reset to No Open Files
F32F/F3EF-F332/F3F2

Called by:
Indirect JMP through (032C) from Kernal CLALL vector at FFE7.

Reset location 98, the number of open files, to zero and fall through to F333/F3F3 to reset any open serial channels and reset the default device numbers.

Operation:
1. Set 98, the number of open files, to 0.
2. Fall through to F333/F3F3, Clear Serial Channels and Reset Default Devices routine.

CLOSE
FFC3

Called by:
JSR at E1CC/E1C9 in BASIC's CLOSE

Entry requirements:
Accumulator should contain the number of the logical file to be closed.

JMP(031C) with a default of F291/F34A.

If the logical file number in the accumulator is found in the logical file number table, also retrieve the current device number from the device number table and the secondary address from the secondary address table.

Then, execute the appropriate CLOSE routine for this current device.

If accessing a serial device, for secondary addresses < 128 (decimal), command the current device to LISTEN, send a CLOSE secondary address, and command the serial device to UNLISTEN. For secondary addresses > 128, this close sequence is omitted.

For an RS-232 device, bring the transmitted data line high, which is the idle state for RS-232 communications. Also, reset the pointers to the end of memory by reclaiming the space used for the RS-232 transmit and receive buffers.

When closing a logical tape file, determine whether writing to or reading from tape. If writing to tape then store a final byte of 0 in the tape buffer and write the buffer to tape.

For all types of devices, a common CLOSE exit is used. The number of open files, 98, is decremented, and the entry for this logical file is deleted from the logical file number table, the device number table, and the secondary address table.

Determine Device for CLOSE
F291/F34A-F2AA/F363

Called by:
Indirect JMP through (031C) from Kernal CLOSE vector at FFC3.

Upon entry, the accumulator contains the logical file number to be closed. First, it calls a routine to determine if the logical file number is in the logical file number table. If the number is not in the table, then it will exit with carry clear. If the number is in the table, then it will retrieve the current device number and secondary address corresponding to this file.

Push the current index into the tables corresponding to the logical file number onto the stack. Determine the type of device the logical file is using and branch to the appropriate routine for closing screen, keyboard, serial, or tape devices, or fall through to following code for closing RS-232 devices.

Entry requirements:
Accumulator should contain the number of the logical file to be closed.

Operation:
1. JSR F314/F3D4 to see if the logical file number is in the logical file number table. Return with X as the index into table corresponding to this logical file if it exists; return with Z = 1 (detected with BEQ) if the logical file is found.
2. If the file is not found CLC and RTS.
3. If the file is found, then JSR F31F/F3DF to retrieve the current device number, BA, the current secondary address, B9, and the current logical file number, B8, from the tables for these with entries corresponding to the location of current logical file number in the logical file number table.
4. Transfer index into tables to accumulator and push on stack for later retrieval by the individual CLOSE routines for RS-232 and serial devices.
5. If current device is the keyboard or screen, branch to F2F1/F3B1 to decrement number of open files and remove current file entry from the three tables.
6. If current device is a serial device, branch to F2EE/F3AE for a JMP to the routine to close a serial device.
7. If current device is tape, branch to F2C8/F38D to close tape files.

8. If current device is RS-232, fall through to the routine at F2AB/F364 to close an RS-232 device.

Common Exit for Close Logical File Routines F2F1/F3B1-F30E/F3CE

Called by:
Falls through after JSR to Close Logical File for Serial Device at F2EE/FEAE, BEQ at F29F/F358 and F2A3/F35C in Determine Device to Close, BEQ at in F2CC/F391 Close Logical File for Tape, BNE at F2E4/F3A4 in Close Logical File for Tape , JMP at F2EB/F3AB in Close Logical File for Tape; alternate entry at F2F2/F3B2 by JSR at F2AC/F365 in Close Logical File for RS-232 Device.

The index into the file tables for the current logical file is retrieved from the stack (except for the alternate entry from RS-232 which has already pulled it from the stack).

The number of open files, 98, is decremented and compared to the index into the file tables. If equal, the current logical file is the last entry in the file table. In this case, there is no need to delete the actual entries in the tables since the pointer to the tables will now cause the next OPEN to overwrite these entries.

If the current logical file index is not equal to the number of open files (after the decrement), replace the current entries of the logical file number, device number, and secondary address tables with the last entries in the table. As the order within a particular table is unimportant, this rearrangement effectively deletes the current entries for the logical file, device, and secondary address.

Entry conditions:
Index into file tables for current logical file is pulled from the stack at entry.

Operation:
1. Pull index into file tables for current logical file from stack.
2. Transfer the index into the tables to X register.
3. Decrement the number of open files (plus one), location 98.
4. If X register equals the value in 98, the logical file being closed is the last entry in the tables. Since 98 points to the next available space in the tables, the next OPEN will overwrite the entries for this current logical file. Thus, just CLC and RTS.

5. If the number of open files (plus one) after decrement is not equal to the value in the X register, the current logical file being closed is not the last entry in the table. In this case, move the last entries in the three tables (device number, secondary address, logical file number) to the current logical file entries. Before this move,the last entry is pointed to by 98 and the current entry by the X register.
6. CLC and RTS.

CLRCHN
FFCC

Called by:
JSR at A447/C447 in BASIC's Error Message Handler, JSR at ABB7/CBB7 in BASIC's INPUT#, JSR at E37B/E467 in BASIC's Warm Start, JSR at F6F4/F777 in Test for STOP Key, JSR at F716/F799 in Error Message Handler.

JMP(0322) with a default of F333/F3F3.

If the current output device is a serial device, send an UNLISTEN command on the serial bus. If the current input device is a serial device, send an UNTALK command on the serial bus.

Set 99, the current input device, to be the keyboard.

Set 9A, the current output device, to be the screen.

Clear Serial Channels and Reset Default Devices
F333/F3F3-F349/F409

Called by:
Indirect JMP through (0322) from Kernal CLRCHN vector at FFCC, fall through from F331/F3F1 in Reset to No Open Files.

Operation:
1. If the current output device is a serial device, JSR EDFE/EF04 to command the serial device to unlisten.
2. If the current input device is a serial device, JSR EDEF/EEF6 to command the serial device to untalk.
3. Reset 9A, the current output device, to the screen (3).
4. Reset 99, the current input device, to the keyboard (0).

GETIN
FFE4

Called by:
JSR at E121 in BASIC's Get a Character.

Setup routines:
OPEN, CHKIN

JMP(032A) with a default of F13E/F1F5.

When retrieving characters from the keyboard, if any characters are in the keyboard buffer, the first character (an ASCII value) in the buffer is returned in the accumulator, and the rest of the characters are moved up one position in the buffer. If no characters are in the keyboard buffer, return with accumulator cleared to 0.

You would use GETIN to retrieve the first character in the keyboard buffer. Contrast this to CHRIN, which does not retrieve anything until RETURN is entered, then returns a character from the logical screen line.

If retrieving from device 2, RS-232, see if the RS-232 receive buffer contains any characters. If it is empty, return with accumulator set to 0. If it contains characters, return with accumulator containing next character in the receive buffer and increment the pointer into the receive buffer.

If retrieving from channel 3 (the screen), channels >= 4 (serial devices), or channel 1 (tape), do the same routines for GETIN that CHRIN does for these devices.

For screen GETIN, return the ASCII code for the screen character in the current logical line pointed to by D3, the column the cursor is on. D3 is then incremented to point to the next character in the line. If D3 is on the end of the line, return the ASCII code $0D for return.

For serial GETIN, the accumulator returns the byte received over the serial bus. However, if any I/O status errors occur, return with accumulator containing $0D.

For tape GETIN, return the next byte from the tape buffer. Also, read one byte ahead to see if the next byte is zero, indicating end of file, and if true, set end-of-file status in 90.

GETIN Preparation
F13E/F1F5-F14D/F204

Called by:
Indirect JMP through (032A) from Kernal GETIN vector at FFE4.

This routine first determines if the current input device is the keyboard. If not, GETIN falls through to F14E/F205 for an RS-232 device or branches to F166/F21D for other devices using the same routines as are used by CHRIN.

If the current input device is the keyboard and if the keyboard buffer contains characters, JMP E5B4/E5CF to retrieve the first character from the keyboard buffer.

Operation:
1. If 99, the current input device, is not 0 (the keyboard), branch to step 5.
2. If 99 is 0 for the keyboard, see if any characters are in the keyboard buffer as indicated by C6, the number of characters in the keyboard buffer.
3. If no characters are in the keyboard buffer, just CLC and RTS, thus returning with the accumulator set to 0.
4. If characters do exist in the keyboard buffer, disable IRQ interrupts and JMP E5B4/E5CF to retrieve the first character from the keyboard buffer and exit.
5. If the current input device, 99, is 2 for RS-232, fall through to F14E/F205 for the routine to get characters from RS-232.
6. If the current input device is neither 0 nor 2, branch to F166/F21D to get a character from other devices. F166/F21D is located in the Determine Input Device routine used by CHRIN; thus, other devices (tape, screen, serial) perform the same routines for both GETIN and CHRIN.

IOBASE
FFF3

Called by:
JSR at E09E/E09B BASIC's in RND
JMP E500.

This routine returns (to the X and Y registers) the address of the start of the I/O registers that control the 6526 CIA/6522 VIA chips. You can write programs that refer to the I/O registers without knowing the exact address of the I/O

register. To write a program in this manner, you would call IOBASE to get the starting address of the I/O registers and add an index to the particular I/O register to which you are referring.

IOBASE for the VIC returns with X register set to $10 and Y register set to $91 (the address of the first register of VIA #1 is 9110). IOBASE for the 64 returns with X register set to $00 and Y register set to $DC (the address of the first register of CIA #1 is DC00). By calling IOBASE, you can determine both the starting address of the I/O registers and which computer the program is running on. Thus, your program could test which computer it is running on and read or write to the appropriate I/O register for this computer, giving you the ability to write one program that works on both the 64 and the VIC. You still will have to know what the CIA/VIA registers do and how to modify them, since what works for a VIA register does not necessarily work the same way on the corresponding CIA register.

BASIC's RND uses this routine to access the CIA/VIA timer registers in generating a random number.

IOINIT (64 only)
FF84

Called by:
None.

JMP FDA3 to initialize the CIA registers, the 6510 I/O data registers, the SID chip registers, start CIA #1 timer A, enable CIA #1 timer A interrupts, and set the serial clock output line high.

The closest equivalent to this routine on the VIC is JMP FDF9 to initialize the 6522 registers, set VIA #2 timer 1 value, start timer 1, and enable timer 1 interrupts.

Since these routines are called during system reset, the main use for IOINIT is for an autostart cartridge that wants to use the same I/O settings that the Kernal normally uses.

LISTEN
FFB1

Called by:
None.

JMP ED0C/EE17.

At ED0C/EE17 the accumulator, which contains the device number, is ORed with $20, turning on bit 5 to prepare to send a LISTEN command on the serial data output line. The device number should be 0–31 (decimal). (If you specify a value > 31, you mess up the high nybble which is used for sending commands on the serial bus.)

RS-232 interrupts are disabled.

Finally, the LISTEN command for this device is sent on the serial bus. To send the LISTEN, the 64/VIC brings the serial attention output line low to cause all devices on the serial bus to listen for a command coming on the bus.

LOAD
FFD5

Called by:
JSR at E175/E172 in BASIC's LOAD/VERIFY.

Setup routines:
SETLFS, SETNAM

Entry requirements:
Accumulator should be set to 0 for LOAD; accumulator set to 1 for VERIFY.

If relocatable load desired: Set X register to the low byte of load starting address, and Y register to the high byte of load starting address.

JMP F49E/F542 to store the X register and Y register in (C3), the starting address of the load, and then JMP(0330) with a default of F4A5/F549.

At F4A5/F549, determine the device. The keyboard, screen, and RS-232 are illegal devices.

For a serial device you must specify a filename. If you don't, the MISSING FILE NAME error message is displayed. With a valid filename, the computer commands the current serial device to listen and sends the secondary address of $60, indicating a load, followed by the filename. Then it tells the device to unlisten. Next, it tells the current serial device to talk, sends the current secondary address of $60, and receives a byte from the serial bus. If the I/O status word indicates the

byte was not returned fast enough, a read time-out has occurred and the FILE NOT FOUND error message is displayed. The first two bytes received from the serial device are used as a pointer to the start of the load area (AE). However, if a secondary address of 0 is specified at entry to load, the X and Y registers stored in (C3) at entry are used as the starting address of the load—thus providing for a relocatable load. Then it receives bytes from the serial bus and stores or verifies them until the EOI status is received. Once the EOI status is received, the serial device is commanded to untalk, and the serial device sends the last buffered character. The serial device then is sent a CLOSE and told to untalk.

For tape LOAD/VERIFY, the LOAD routine first checks if the tape buffer is located in memory >= 0200. If so, it loads the tape buffer with a header retrieved from the tape. If a filename has been specified, a specific header with this filename is loaded; if there is no filename, it loads the next header on the tape. Only tape headers with tape identifiers of 1 or 3 are acceptable for LOAD/VERIFY. A tape identifier of 5 indicates an end-of-tape header, and in this case the routine will exit with carry set and accumulator set to 5. Tape identifiers of 2 or 4 are for sequential files.

A tape identifier of 3 causes a nonrelocatable load even if you have specified values in the X and Y registers at entry and a secondary address of 0. That is, you can't override a tape identifier of 3—it forces a nonrelocatable load.

A tape identifier of 1 allows a relocatable load. If the tape identifier is 1 and the secondary address is 0, the X and Y register values at entry are used to determine the starting address for the load.

For a nonrelocatable load, the starting address for the load is taken from the tape header. The ending address for the load (in both relocatable and nonrelocatable loads) is determined by adding the length of the program to the starting address. After determining whether to do a relocatable or nonrelocatable load, it loads RAM from the next two program blocks on tape (two blocks are used for error correcting purposes; they should be identical copies of each other).

Jump to LOAD Vector F49E/F542-F4A4/F548

Called by:
JMP from Kernal LOAD vector at FFD5.

The starting address for a possible relocatable load, specified in the X and Y registers at entry, is stored in (C3), followed by JMP (0330) with the default vector of F4A5/F549.

Operation:
1. STX C3.
2. STY C4.
3. JMP(0330).

Determine Device for LOAD F4A5/F549-F4B7/F55B and F533/F5CA-F538/F5D0

Called by:
Indirect JMP through (0330) at F4A2/F546 in Jump to LOAD Vector.

This routine determines which device is being used for the load. Invalid devices are the screen, keyboard, or RS-232. Valid devices are serial devices or tape, and for these the routine passes control to the appropriate serial or tape load routine.

Operation:
1. STA 93, thus setting the LOAD/VERIFY flag to 0 for LOAD or to 1 for VERIFY.
2. Reset 90, the I/O status, to 0.
3. LDA from BA, the current device number.
4. If the current device is the keyboard (0) or the screen (3), JMP F713/F796 to display the ILLEGAL DEVICE NUMBER error message.
5. If the current device number is less than 3, branch to step 7.
6. If the current device number is greater than 3, it's a serial device. Fall through to F4B8/F55C to load from a serial device.
7. If the current device is RS-232, JMP F713/F0B9 to display ILLEGAL DEVICE NUMBER.
8. If the current device number is not 2 (RS-232), the only number left is 1 (tape). Fall through to F539/F5D1 to load from tape.

MEMBOT
FF9C

Called by:
JSR at E403/E3E5 in BASIC's Cold Start.

Entry requirements:
Carry should be set or clear, depending on function desired:

Set carry to read bottom of memory.

Clear carry to set bottom of memory. The X register is the low byte of the address of the bottom of memory, and the Y register is the high byte of the address of the bottom of memory.

JMP FE34/FE82.

If the carry is clear at entry, set the pointer to bottom of memory (0281) from X and Y registers.

If the carry is set at entry, load X and Y registers from (0281), the pointer to the bottom of memory.

The initial values of (0281) are 1000 for an unexpanded VIC, 0400 for a VIC with 3K expansion, 1200 for a VIC with 8K or more expanded, and 0800 for the 64.

MEMBOT Execution
FE34/FE82-FE42/FE90

Called by:
JMP from Kernal MEMBOT vector at FF9C.

If the carry is clear at entry, set (0281), the pointer to the bottom of memory, from the X and Y registers. If carry is set at entry, load X and Y registers from (0281).

Operation:
1. If carry is clear, branch to step 3.
2. Load X and Y registers from pointer to bottom of memory (0281), and fall through to step 3.
3. Set (0281) from values in X and Y registers.
4. RTS.

MEMTOP
FF99

Called by:
JSR at E40B/E3ED in BASIC's Cold Start.

Entry requirements:
Carry should be set or clear, depending on function desired:

Set carry to read end of memory.

Clear carry to set end of memory. The X register contains the low byte of the address of the start of memory, and the Y register contains the high byte of the address of the start of memory.

JMP FE25/FE73.

If carry is clear at entry, set pointer to top of memory (0283) from X and Y registers.

If carry is set at entry, load X and Y registers from (0283), the pointer to the top of memory.

MEMTOP Execution
FE25/FE73-FE33/FE81

Called by:
JMP from Kernal MEMTOP vector at FF99; alternate entry at FE27/FE75 by JSR at F2B2/F377 in Close Logical File for RS-232, JSR at F468/F527 in Open RS-232 Device; alternate entry at FE2D/FE7B by JMP at F480/F53F in Open RS-232 Device, JMP at FDCF in Initialize Memory Pointers (VIC only).

If entering at FE25/FE73, the carry flag determines whether the top of memory is being set or read. If the carry bit is clear, or if the routine is entered at FE2D/FE7B, the top of memory pointer (0283) is set from the X and Y register values. If the carry is set, or if the routine is entered at FE27/FE75, the X and Y registers are set from the top of memory pointer (0283).

Operation:
1. FE25/FE73: If carry is set, branch to step 3.
2. FE27/FE75: Load X and Y registers from pointer to top of memory (0283), and fall through to step 3.
3. FE2D/FE7B: Set (0283) from values in X and Y registers.
4. RTS.

OPEN
FFC0

Called by:
JSR at E1C1/E1BE in BASIC's OPEN.

Setup routines:
SETLFS, SETNAM

JMP(031A) with a default of F34A/F40A.

OPEN checks whether another logical file can be opened. Another logical file can be opened if the logical file number is not 0 and if fewer than ten logical files are already open. OPEN exits if trying to open to the screen or keyboard, as these devices do not use files.

For a serial device, OPEN commands the serial device to listen and then sends a secondary address for OPEN to this serial device.

For tape, OPEN checks for a tape header of a sequential file if reading, or writes a tape header for a sequential file if writing.

The RS-232 OPEN initializes various RS-232 lines and creates two 256-byte buffers at the top of memory. RS-232 OPEN handles the x-line handshaking opening sequence incorrectly on the VIC.

OPEN Execution
F34A/F40A–F3D4/F494

Called by:
Indirect JMP through (031A) from Kernal OPEN vector at FFC0.

OPEN creates a logical file that can be used by Kernal I/O routines.

OPEN first checks if the logical file number specified is 0. If it is 0, jump to the display the NOT INPUT FILE message and exit, as a logical file number of 0 is not permitted.

If the number of files already open is less than ten and the logical file specified is not yet open, create entries in the logical file number table, device number table, and secondary address tables for this file. If a file already exists that uses the specified file number, the FILE OPEN error message is displayed. If there are already ten open files, the TOO MANY FILES message is displayed.

If the device is the screen or keyboard, exit as these devices do not use files. For RS-232-C, serial, or tape devices, jump or branch to their respective OPEN routines.

Operation:

1. If the logical file number is 0, JMP F70A/F78D to set error message number of 6 (NOT INPUT FILE) and exit.
2. JSR F30F/F3CF to see if the logical file number in B8, the current logical file, is already present in the logical file number table.
3. If the logical file number is in the logical file number table, JMP F6FE/F781 to set the error message number of 2 (FILE OPEN) and exit.
4. LDX 98 to get the number of open files.
5. If less than ten open files exist, continue with step 6. If ten files are already open, JMP F6FB/F77E to set error message number of 1 (TOO MANY FILES).
6. Increment the number of open files, 98.
7. Store the current logical file number in the logical file number table, using the X register value from step 4 as an index into the table at 0259.
8. LDA B9, the secondary address, then ORA $60 in case a secondary address is to be sent to a serial device. Store this value in the secondary address table entry that corresponds to the entry for this file in the logical file number table.
9. Store the current device number, BA, in the device number table entry that corresponds to the entry for this file in the logical file number table.
10. If the current device is the keyboard or the screen, CLC and RTS as there is no need to open files to these devices.
11. If the current device is 1 or 2, branch to step 14. For devices > 3, fall through to step 12.
12. JSR to F3D5/F495 to send secondary address in B9 to the current device on the serial bus. The secondary address is ORed with $F0 before sending to provide the secondary address for OPEN. Return with carry clear if the serial device was present and the secondary address was sent without errors. No error routine is called from OPEN if this sequence fails.
13. If the carry is clear on return from the JSR in step 12, exit with carry clear. However, if the routine does not return the carry clear to indicate the device was present and responded in time, OPEN does not branch to any error routine. This appears to be why you can open nonexistent

devices without getting an error until you actually try to send or receive data for the device.

14. If the device number is 2 (RS-232), JMP F409/F4C7 to open an RS-232 device and RTS upon completion of the RS-232 OPEN.
15. If the device number is none of the above, it must be 1 (tape). Branch to F38B/F44B to open a logical file to tape.

PLOT
FFF0

Called by:
JSR at AAE9/CAE9 in BASIC's Tab to Column for PRINT, JSR at AAFA/CAFA in BASIC'S TAB and SPC, JSR at B39F/D39F in BASIC'S POS.

Entry requirements:
Carry bit should be set or clear, depending on function desired:

Set carry to read cursor location (X register = row, and Y register = column).

Clear carry to set cursor location (X register = row, and Y register = column).

JMP E50A (see screen routines in chapter 7).

If the carry bit is clear at entry, move the cursor to the specified location. The contents of the the X register determine the new cursor row andthe contents of the Y register determine the new cursor column.

If the carry bit is set at entry, read the cursor location and place the row value for the current cursor location into the X register and column value for the current cursor location into the Y register.

The row number indicates the physical line, while the column number indicates the column within a logical line. Valid physical line numbers in decimal are 0–24 (64) and 0–22 (VIC). Valid logical column numbers in decimal are 0–79 (64) and 0–87 (VIC).

RAMTAS (64 only)
FF87

Called by:
None.

JMP FD50 to the Initialize Memory Pointers routine on the 64.

At FD50 the routine stores $00 in locations 02–0101 and 0200–03FF; sets the pointer to the tape buffer, (B2), to 033C; sets the pointer to the end of RAM + 1 in (0283), sets the pointer to the start of RAM in (0281), sets the screen memory page to $04.

Although the VIC does not have a RAMTAS Kernal vector, the corresponding operation on the VIC is done by JMP FD8D. At FD8D the routine stores $00 in 00–FF and 0200–03FF; sets pointer to tape buffer, (B2), to 033C; sets the pointer to the end of RAM + 1 in (0283); sets the pointer to the start of RAM in (0281); sets the screen memory page to $1E or $10 depending on where RAM ends.

The RAMTAS routine would mainly be used by an autostart cartridge since the RAMTAS functions are normally executed during system reset.

RDTIM
FFDE

Called by:
JSR at AF84/CF84 in BASIC's TI and TI$.

JMP F6DD/F760.

This routine reads the jiffy clock (A2–A0) into the accumulator, X register, and Y register.

A0 is updated every 1/60 second. When the jiffy clock reaches a value equal to 24 hours, it is reset to 0.

Exit conditions:
Accumulator holds high byte of jiffy clock. X register holds middle byte of jiffy clock. Y register holds low byte of jiffy clock.

RDTIM/SETTIM Execution
F6DD/F760-F6EC/F76F

Called by:
JMP from Kernal RDTIM vector at FFDE; alternate entry at F6E4/F767 by JMP from Kernal SETTIM vector at FFDB.

From the RDTIM entry point, this routine reads the jiffy clock at A2–A0 into the accumulator, X register, and Y register, and then falls through to the following routine at F6E4/F767.

If entering at the SETTIM entry point at F6E4/F767, set the jiffy clock at A2, A1, and A0 from the accumulator, X register, and Y register.

Operation:
1. F6DD/F760: SEI to disable interrupts.
2. LDA from A2, LDX from A1, and LDY from A0.
3. F6E4/F767: SEI to disable interrupts (which has no effect if interrupts were already disabled in step 1).
4. STA at A2, STX at A1, and STY at A0.
5. CLI to enable interrupts, then RTS.

READST
FFB7

Called by:
JSR at ABDD/CBDD in BASIC's INPUT, JSR at AF9A/CF9A in BASIC's STATUS, JSR at E180/E17D E195 in BASIC's LOAD/VERIFY.

JMP FE07/FE57 to read the I/O status word, 90, returning the value in the accumulator. This value reflects certain conditions during serial or tape I/O.

The 64/VIC *Programmer's Reference Guides* contain some errors:

First, when VERIFYing for a serial device, a VERIFY error can occur.

Second, for the VIC you cannot read the RS-232 status register, 0297, by calling this routine. READST for RS-232 always returns zero on the VIC. If you want to read the RS-232 status on the VIC, read 0297 directly; don't call this routine. This error in READST is corrected in the 64.

Third, detecting an end-of-tape header allows BASIC to display the DEVICE NOT PRESENT error message, but the Kernal routines for OPEN or LOAD/VERIFY do not set location 90. Thus, READST will not return the end-of-tape status condition following OPEN or LOAD/VERIFY. You can check end-of-tape status upon return from OPEN or LOAD/VERIFY by checking for the carry bit set and the accumulator set to 5, which are the conditions that indicate end-of-tape.

The table below shows the possible values returned by READST:

READST Values

Hex Value	Bit	Serial I/O	Tape Read/ LOAD/VERIFY	RS-232 (64 only)
$80	7	Device not present		Break detected
$40	6	EOI status	End of file	DSR signal missing
$20	5		Checksum error	
$10	4	VERIFY error	Unrecoverable read error	CTS signal missing
$08	3		Long block	Receiver buffer empty
$04	2		Short block	Receiver buffer overrun
$02	1	Read timeout		Framing error
$01	0	Write timeout		Parity error

Status Terms

Long Block: Tape read is trying to read data bytes after the first block has already completed.

Short Block: Tape read is reading leader bits between blocks while the byte action routine is still expecting to be reading bytes from the block.

Unrecoverable Read Error: During tape read and LOAD/VERIFY, more than 31 errors were detected in block 1. This is also set if read or VERIFY errors for the same byte occurred in both blocks 1 and 2.

Checksum Error: Computed parity for the loaded area is not the same as the final byte of tape block 2 (the parity computed during the SAVE of the second block).

End of File: This status is set when doing CHRIN from tape for a sequential file and the read-ahead byte in the tape buffer is 0.

VERIFY Error: The byte retrieved from the serial device does not match the byte in memory.

EOI (End or Identify): This is set during the Receive Byte from Serial Device routine when the EOI handshake is performed. Set during serial read to indicate the last byte has been sent from the serial device. The unusual term EOI is a holdover from the IEEE-488 bus definitions used on older PET/CBM computers; you may find it simpler just to remember this as End of File for disk.

Device Not Present: Device does not respond with the proper handshake sequence during OPEN, LOAD, VERIFY, or SAVE operations.

Read Timeout, Write Timeout: Read or write timeouts are set when the serial device doesn't handshake within the allocated time.

Break Detected: This is set if the check for a stop bit finds a 0 rather than a 1, and the data bits received so far are all 0's.
Framing Error: This is set if the check for a stop bit finds a 0 and the data bits received so far included some bits set to 1.
DSR Signal Missing: The 64 can't detect the Data Set Ready signal from the RS-232 device during x-line handshaking.
CTS Signal Missing: The 64 can't detect the Clear To Send signal from the RS-232 device during x-line handshaking.
Parity Error: The parity bit indicates an error in transmission of this byte.
Receiver Buffer Empty: Nothing is in the RS-232 input buffer. This allows routines to test the status so they don't loop waiting for data.
Receiver Buffer Overun: The RS-232 input buffer is full and another byte has been received.

Read/Set Status and Set Message Control FE07/FE57-FE20/FE6E

Called by:
JMP from Kernal READST vector at FFB7; alternate entry at FE18/FE66 by JMP from Kernal SETMSG vector at FF90; alternate entry at FE1C/FE6A by JSR at EDB2/EEB9 in Set Status Word, JSR at EE4F/EF4D in Receive Byte from Serial Device, JSR at F18A/F241 in CHRIN from Tape, JSR at F518/F5AF in Load/Verify from Serial Device, JSR at FA81/FACE in Determine Action for Byte, JSR at FAC6/FB13 in Test for Short Block, JSR at FB35/FB82 in Flag Unrecoverable Read Error, JSR at FB8B/FBCC in Reset Vectors and Compute Checksum.

If entering at FE07/FE57, determine the device number. For an RS-232 device, the VIC always stores 0 in the RS-232 status word, 0297, and returns with accumulator set to 0, while the 64 correctly returns with the accumulator holding the value from 0297, and also stores 0 in 0297. For other devices, return with the value from 90, the I/O status, in the accumulator.

If entering at FE18/FE66, store the contents of the accumulator in 9D, the Kernal message control flag. $80 sets Kernal control messages on, $40 sets Kernal error messages on, $C0 sets both Kernal control and error messages on, and

$00 turns off all Kernal messages. Exit with the accumulator containing the value from 90, the I/O status word.

If entry at FE1A/FE6A, then set 90, the I/O status, by ORing the accumulator with the current value of 90. The routines that call this entry point set the accumulator to a value for an I/O status.

Operation:
1. FE07/FE57: If the device number, BA, is not 2, branch to step 4.
2. If device number is 2 (RS-232): LDA 0297, PHA (64 only), LDA $00, STA 0297, PLA (64 only), RTS.
3. FE18/FE66: Store accumulator in 9D, the Kernal message control flag.
4. LDA 90, the I/O status.
5. FE1C/FE6A: ORA 90. Thus, if the routine continues from the instructions above, 90 is not changed. However, if the alternate entry point of FE1C/FE6A is used, a new value will result from this ORA.
6. STA 90, updating the I/O status word if the alternate entry point of FE1C/FE6A was used.

RESTOR
FF8A

Called by:
None.

JMP FD15/FD52 to execute the routine to initialize the Kernal RAM vectors. This RAM vector initialization is also done during system reset.

Calling this routine restores the vectors at (0314)–(0332) to their default values from the table at FD30/FD6D.

SAVE
FFD8

Called by:
JSR at E15F/E15C in BASIC's SAVE.

Setup routines:
SETLFS, SETNAM (not required for saving to tape)

Entry requirements:
The accumulator should contain the offset within zero page to

a two-byte pointer to the start of the area to be saved. The X register should hold the low byte of the address of the end of the area to be saved + 1. Y register should hold the high byte of the address of the end of the area to be saved + 1.

JMP F5DD/F675 to save memory to a serial device or to tape. Saves to the screen, keyboard, or RS-232 are not permitted.

If saving to tape from the VIC, only the contents of memory locations 0–7FFF may be saved. This restriction does not apply when saving to tape from the 64.

A filename is required (through SETNAM) when saving to serial devices; a filename is optional when saving to tape.

At F5DD/F675 , the routine loads the pointer to the end of the save area + 1, (AE), from the X and Y registers. (End + 1 denotes the fact that you must load X and Y to point to the location just past the end of the save area, since the save routines consider the save complete when the pointer to the save area equals the value of the pointer (AE).) It also sets (C1), the pointer to the start of the save area, from the zero page pointer indexed by the accumulator, and then performs an indirect JMP through the vector at (0332), which defaults to F5ED/F675.

For a serial save, the routine commands the current serial device to listen with attention, then sends a secondary address of $61 to indicate a SAVE operation. If the device is present, the filename and the starting address are sent to the serial device. Next, the routine sends all the bytes from the save area over the serial bus. When the save is complete, it sends a secondary address of $E1 to indicate the CLOSE command and commands the serial device to unlisten.

For tape save, it is important that you specify the secondary address correctly. For an even secondary address, the header for the saved program will have a identifier byte of 1, indicating a relocatable program. An odd secondary address produces a header identifier byte of 3, indicating a non-relocatable program. Also, if you have bit 1 on in the secondary address ($02 or $03 would set bit 1), then an end-of-tape header with a identifier byte of 5 is written following the saved program.

The tape save operation first writes a header to tape. This tape header contains the identifier byte, the starting address and ending address + 1 of the save area, and the filename (if a filename is used). Then data from the save area is written to

tape. If bit 1 of the secondary address is 1, an end-of-tape header is also written following the data from the save area. Two identical copies of the tape header(s) and the program are written to tape to allow for error checking and correction during tape loading.

Jump to SAVE Vector F5DD/F675-F5EC/F684

Called by:
JMP from Kernal SAVE vector at FFD8.

Set the pointer to the end of the save area + 1, (AE), from the X and Y registers.

The accumulator value at entry is transferred to the X register and is used as an index into page zero for the location of two bytes that specify the starting address for the save. Set the pointer to the start address of the save area, (C1), from these two page-zero bytes.

Jump to the address in the vector at (0322), normally F5ED/F685.

Operation:
1. STX AE, the low byte of the address of the end of the save area + 1.
2. STY AF, the high byte of the address of the end of the save area + 1.
3. TAX and LDA 00,X to get the low byte of the address of the start of the save area, and STA in C1.
4. LDA 01,X to get the high byte of the address of the start of the save area, and STA in C2.
5. JMP (0322) to the save routine. The default address in the vector is F5ED/F685.

Determine Device for SAVE F5ED/F685-F5F9/F691

Called by:
Indirect JMP through (0322) at F5EA/F682 in Jump to SAVE Vector.

If the current device is the keyboard or the screen, load the accumulator with 9 and set the carry bit to display the ILLEGAL DEVICE NUMBER message, then exit.

If the current device is either tape (1) or RS-232 (2), branch to Control Routine for Tape Save (which treats RS-232 as an illegal device).

If the current device is a serial device, fall through to the Save to Serial Device routine.

Operation:

1. If the current device is the keyboard or the screen, JMP F713/F796 to display the ILLEGAL DEVICE NUMBER error message, set accumulator to 9, set carry, and exit. The keyboard or the screen is not a valid device for saves.
2. If the current device is RS-232 or tape, branch to F659/F6F1, a routine that determines whether the save is to RS-232 or tape. If RS-232 is specified, the ILLEGAL DEVICE NUMBER message is displayed. If the device is a tape device, the tape save routines are executed.
3. If the device is none of the above, its device number must be >= 4; thus, it's a serial device. Fall through to Save to Serial Device routine at F5FA/F692.

SCNKEY
FF9F

Called by:

None.

JMP EA87/EB1E to the Keyboard Scan routine (see chapter 4) to check for a keypress. If a valid key is found down and the keyboard buffer is not full, the ASCII code value for the key is placed in the buffer.

SCNKEY is useful if you have written a machine language program that runs with IRQ interrupts disabled, but you still want to scan the keyboard.

SCREEN
FFED

Called by:

None.

JMP E505 to return the number of columns on the standard display screen in the X register and the number of rows in the Y register. On the 64, the routine returns 40 in X and 25 in Y. The VIC routine returns 22 in X and 23 in Y.

A definitive way to let a program know whether it's running on the VIC or the 64 is to JSR to SCREEN and test the values returned.

SECOND
FF93

Called by:
None.

Setup routines:
LISTEN

JMP EDB9/EEC0 to send the byte in the accumulator on the serial bus as a secondary address command with the serial attention output line set low. After this command is sent, the serial attention output line is brought high, the setting for transmitting normal data bytes.

You must ORA the secondary address with $60 before calling this routine to convert the secondary address to a recognized IEEE secondary address command. See page 378 of Raeto Collin West's *Programming the PET/CBM* for a detailed chart of the IEEE command groups.

SETLFS
FFBA

Called by:
JSRs at E1DD/E1DA, E1F0/E1ED, and E1FD/E1FA in BASIC's Set LOAD/VERIFY/SAVE Parameters; JSRs at E22B/E228, E23F/E23C, and E24E/E24B in BASIC's Handle Parameters for OPEN and CLOSE.

Entry requirements:
The accumulator should hold the logical file number, the X register should hold the device number, and the Y register should hold the secondary address.

JMP FE00/FE50 to set the logical file number, device number, and secondary address for a subsequent open, load, or save.

The logical file number can be 1–255.

The device numbers can be 0–31. Assigned device numbers include 0 for the keyboard, 1 for tape, 2 for RS-232, 3 for the screen, and 4–31 for serial bus devices. By convention, se-

rial device numbers 4 and 5 are usually used for printers and 8–11 for disk drives.

See the comments in the paragraphs on SAVE and LOAD routines about secondary addresses. An even secondary address gives a identifier byte of 1 for a relocatable program tape header. An odd secondary address gives a tape identifier of 3 for a nonrelocatable program tape header. A secondary address that has bit 1 on (e.g., $02 or $03) produces an end-of-tape header with an identifier byte of 5.

Secondary addresses >= 128 (decimal) will not be sent on the serial bus. For reading from serial, use an even secondary address. For writing to serial, use an odd secondary address. Valid secondary addresses for serial devices are 0–31 (decimal). If you specify a higher value, you may be sending a command other than what you intended, since secondary addresses greater than 31 are used to represent commands to serial devices.

Set Logical File Number, Device Number, Secondary Address
FE00/FE50–FE06/FE56

Called by:
JMP from Kernal SETLFS vector at FFBA.

In preparation for other Kernal I/O routines, this sets the logical file number, device number, and secondary address.

Device numbers should range only from 0–31 (decimal). Serial commands such as OPEN, TALK, LISTEN, CLOSE, and others use bits 5–7 of the secondary address to specify the type of command. The commands do this ORA of the command high nybble with the device number. Thus, a device number greater than 31 might result in an invalid command after the ORA.

Also, if you specify a device number > 31, you do not get a device error until you actually try to send the device data (or retrieve data from it). This quirk is due to OPEN not checking for an invalid carry status after opening for a serial device.

Normally, a secondary address >= 128 (decimal) indicates no secondary address is desired. However, for tape operations, 255 (or any odd value > 128) is the same as a secondary address of 3. See " Block SAVE and LOAD" by Sheila Thornton in *COMPUTE!'s Second Book of VIC* from COMPUTE! Books.

Operation:
1. STA B8, the logical file number.
2. STX BA, the device number.
3. STY B9, the secondary address.

SETMSG
FF90

Called by:
JSR at A47D/C47D in BASIC's Enable Kernal Control Messages, JSR at A874/C874 in BASIC'S Disable Kernal Control Messages.

Entry requirements:
Accumulator should contain the value used to set message control: $80 allows Kernal control messages; $40 allows Kernal error messages; $C0 allows both Kernal control and error messages; $00 disallows all Kernal messages.

JMP FE18/FE66. This routine is called to determine which messages will be displayed in response to control or error conditions. The accumulator value at entry determines the setting of the message control status.

Thanks to Russ Davies for pointing out that bits 6 and 7 are reversed in describing how to set message control in the 64 and VIC *Programmer's Reference Guides.*

SETNAM
FFBD

Called by:
JSR at E1D6/E1D3 in BASIC's Set LOAD/VERIFY/SAVE Parameters, JSRs at E21B/E218 and E261/E25E in BASIC's Handle Parameters for OPEN and CLOSE.

Entry requirements:
Accumulator should contain the length of the filename. The X register should hold the low byte of the starting address of the filename. The Y register should hold the high byte of the filename address. The filename may be stored at any addressable memory location.

JMP FDF9/FE49 to prepare a filename for subsequent OPEN, LOAD/VERIFY, or SAVE processing. The accumulator value, the length of the filename, is stored in B7. The pointer to the filename from the X and Y registers is stored in (BB).

Although you could create a filename that is 255 (decimal) characters long (the accumulator can hold a maximum value of $FF or decimal 255), not all of this maximum filename size can be used.

For tape, the filename is stored in the tape buffer, which is 192 bytes long. However, 5 bytes are taken for the identifier and the starting and ending addresses, which leaves 187 bytes that can be used for the filename.

One quirk with the serial devices is that if the secondary address you specify in SETLFS is larger than 128, the filename is not sent for OPEN, LOAD, or SAVE.

Set Filename Location and Number of Characters FDF9/FE49–FDFF/FE4F

Called by:
JMP from Kernal SETNAM vector at FFBD.

The location of the filename is placed in a pointer at (BB), and the number of characters in the filename is placed in B7.

This routine sets filename information for later use of the Kernal routines OPEN, SAVE, and LOAD. If no filename is needed for these routines, load the accumulator with zero before calling this routine. However, loading or saving to a serial device requires that a filename be present.

Operation:
1. STA B7, the number of characters in the filename.
2. STX BB, the low byte of the address of the filename.
3. STY BC, the high byte of the address of the filename.

SETTIM
FFDB

Called by:
JMP at AA1A/CA1A in BASIC's TI$.

Entry requirements:
The accumulator should hold the high byte to be stored in the jiffy clock. The X register should hold the middle byte to be stored in the jiffy clock. The Y register should hold the low byte to be stored in the jiffy clock.

JMP F6E4/F767 to set the three-byte jiffy clock at A2–A0 from the values in the accumulator, X register, and Y register.

SETTMO
FFA2

Called by:
None.

JMP FE21/FE6F to store accumulator in 0285. The VIC-20 *Programmer's Reference Guide* refers to this routine as setting a serial timeout flag and the Commodore 64 *Programmer's Reference Guide* refers to it as setting a flag for IEEE timeout. However, neither BASIC nor the Kernal refers to this vector. Since 0285 is not a register for an I/O chip and it is never referred to, it's hard to see how it can be used to enable or disable timeouts.

STOP
FFE1

Called by:
JSR at A82C/C82C in BASIC's Test for STOP Key, JSR at F4F9/F590 in Load/Verify from Serial Device, JSR at F62E/F6C6 in Save to Serial Device; JSR at F8D0/F94B Test for STOP Key During Tape I/O; JSR at FE61/FECD in NMI Interrupt Handler (to find STOP and RESTORE).

JMP (0328) with a default of F6ED/F770. At F6ED/F770, test 91 for the value $7F/$FE. Location 91 contains the key switch value of the STOP key column (column seven/three) of the keyboard scan. If $7E/$FE is found, set the Z flag of the status register to 1, call FFCC to reset I/O channels, and set C6, the number of characters in the keyboard buffer, to 0.

If $7E/$FE is not found, the Z flag will be 0 on exit (BNE condition). In this case, the accumulator can still be tested for the keys shown below using the value shown following it.

STOP Routine Return Values

Commodore 64 Key	Accumulator	VIC-20 Key	Accumulator
1	$FE	Cursor down	$7F
Left arrow	$FD	/	$BF
CTRL	$FB	,	$DF
2	$F7	N	$EF
Space	$EF	V	$F7
Commodore	$DF	X	$FB
Q	$BF	Left SHIFT	$FD

If no key is down in the STOP column, the routine returns $FF in the accumulator (64 and VIC).

Test for STOP Key
F6ED/F770-F6FA/F77D

Called by:
Indirect JMP through (0328) from Kernal STOP vector at FFE1.

This routine is called to test whether the STOP key is being held down. When the STOP key is found down, this routine exits with the Z status flag set to 1, allowing the calling routine to test for this result with BEQ.

Location 91 has the value of the keyboard scan for the STOP key column during the last IRQ or NMI interrupt.

Operation:
1. LDA 91.
2. Check for the value that indicates the STOP key is pressed, $7F/$FE.
3. If STOP key is not pressed, then branch (BNE) to step 7 to RTS with the accumulator containing last value in $91.
4. If STOP key is pressed, then JSR FFCC (the Kernal CLRCHN vector) to clear serial I/O and reset default input and output devices, returning with accumulator cleared to 0.
5. STA C6, the number of characters in the keyboard buffer, thus clearing the buffer.
6. Restore the status of the comparison from step 2, thus restoring Z to 1 (BEQ condition).
7. RTS.

TALK
FFB4

Called by:
None.

Entry requirements:
Accumulator should hold the serial device number (4–31 decimal).

JMP ED09/EE14 where the accumulator is ORed with $40 to set the value for a talk command to the device. Send this command over the serial data bus while the serial attention output line is held low.

TKSA
FF96

Called by:
None.

Setup routines:
TALK.

Entry requirements:
Accumulator should hold the secondary address 0–31 (decimal) ORed with $60.

JMP EDC7/EECE to send the secondary address after the TALK command and to do the TALK-LISTEN turnaround where the serial device becomes the talker and the 64/VIC (and other devices on the serial bus) become the listener.

The IEEE convention is that $6X and $7X represent secondary addresses. The 0–31(decimal) that you ORA with $60 before calling TKSA results in $6X if the accumulator holds 0–15 (decimal) and $7X if the accumulator holds 16–31 (decimal). The VIC-1541 *User's Manual* states that the secondary addresses can be 2–15 with 15 the command channel and 0 and 1 reserved for load and save.

UDTIM
FFEA

Called by:
JSR at EA31/EABF in IRQ Interrupt Handler.

JMP F69B/F734 to update the jiffy clock at A2–A0 and store a value from the keyboard row for column number seven/three (which contains the STOP key) in 91 if a key in that row is detected.

Normally, this routine is called by the IRQ interrupt handler (64 and VIC) or by the NMI interrupt handler (VIC only). However, if you run a program with IRQ interrupts disabled, you should call this routine if you want the jiffy clock incremented and the STOP key column value saved in 91.

UNLSN
FFAE

Called by:
None.

JMP EDFE/EF04 to send $3F, the command for UNLISTEN, over the serial bus. Serial devices that are listening should recognize the command and terminate their connection to the serial bus.

UNTALK
FFAB

Called by:
None.

JMP EDEF/EEF6 to send $5F, the command for UNTALK, over the serial bus. Serial devices that are talking should quit talking and terminate their connection to the serial bus.

VECTOR
FF8D

Called by:
None.

Entry requirements:
Carry should be set or clear, depending on the function desired:

Set the carry bit to store the RAM vectors at (0314)–(0332) at the location pointed to by the X and Y registers.

Clear the carry bit to load the RAM vectors at (0314)–(0332) from the location pointed to by X and Y.

JMP FD1A/FD57 (see chapter 2).

Store X and Y at (C3), the base address of where the vector table will be read from or stored to.

If the carry is set, store the RAM vectors at (0314)–(0332) to the location pointed to by the X and Y registers.

If the carry is clear, load the RAM vectors at (0314)–(0332) from the location pointed to by X and Y.