```
;---------------
Commodore 64 ROM Tape Routines

Disassembly and Documentation by Fungus/Nostalgia 8/2023

Revision 3

Acknowledgements for resources and references used:

Nick Hampshire (C64 Rom Disassembly)
Luigi Defraia (TCE) (CBM Tape Loader Format)
Tom Roger Skauen (SLC) (Bouncing thoughts, confirmation of data)
Martin Piper (Timing Code verification)
Michael Steil (mist64) (CBM Source Code, final verifications)
Pontus Berg (Bacchus) Inspiration

This document was created in order to gain a complete understanding of the tape routines, to correct errors
in
other documents, and for fun.

;---------------

$90 = STATUS          ;Status byte
$91 = STOPKEY         ;Stop key pressed flag
$92 = TAPTIMCON       ;Tape timing constant adjustment
$93 = LODVER          ;Load/Verify flag
$96 = TAPLDRCNT       ;Tape Leader Bit Pair Count (pilot)
$97 = TAPFSTBIT       ;Tape First Bit in Pair
$9B = TAPPRTY         ;Tape Byte Parity
$9C = TAPBYTREC       ;Tape Byte Received Flag or Cassette Dipole ?
$9D = MSGFLG          ;Kernal Messages Flag
$9E = TAPFILTYP       ;Tape File Type / Tape Get Index / Tape Search Filename Index / Tape Pass 1 Error Log
Index
$9F = TAPBUFIDX       ;Tape Buffer Index / Tape Buffer Filename Index / Tape Pass 2 Error Log Index
$A3 = TAPBITSCNT      ;Tape Byte Bits Count
$A4 = TAPBITPR        ;Tape Bit Pair (bit 0 or 1)
$A5 = TAPCNTDN        ;Tape Sync Count Down
$A6 = TAPBUFPTR       ;Tape Buffer Byte Pointer
$A7 = TAPREADPASS     ;Tape Read Pass Count
$A8 = TAPREADERR1     ;Tape Read Error
$A9 = TAPSPCNT        ;Tape Short Pulse Count
$AA = TAPBYTEFLG      ;Tape Input Byte Flag (Data/Sync Read Mode)
$AB = TAPLEDCNTH      ;Tape Short Count MSB (Tape Leader Counter) / Calculated Checksum
$AC = LOADADDRLO      ;File Load Address Low (Read) / File Save Address Low (Write)
$AD = LOADADDRHI      ;File Load Address High (Read) / File Save Address High (Write)
$AE = ENDADDRLO       ;File End Address Low
$AF = ENDADDRHI       ;File End Address High
$B0 = TAPPLSLO        ;Tape Pulse Length Low
$B1 = TAPPLSHI        ;Tape Pulse Length High
$B2 = TAPBUFSLO       ;Tape Buffer Start Low
$B3 = TAPBUFSHI       ;Tape Buffer Start High
$B4 = TAPBITIO        ;Tape Bit In/Out Flag
$B5 = TAPEEOT         ;Tape Block Sync
$B6 = TAPREADERR2     ;Tape Read Error (Read) / Tape Bit Count Out (Write)
$B7 = FNAMLEN         ;Filename Length
$B8 = LFN             ;Logical File Number
$B9 = SA              ;Secondary Address
$BA = DVN             ;Current Device Number
$BB = FNAMHI          ;Filename Address Low
$BC = FNAMLO          ;Filename Address High
$BD = TAPBYTEIO       ;Tape Byte Input/Output / Checksum
$BE = TAPBLKCNT       ;Tape Block Copy Count
$BF = TAPBITIO        ;Tape Bits In/Out
$C0 = TAPBUTST        ;Tape Play Button Status (Tape Motor Interlock)
$C1 = TAPBUFLO        ;Tape Buffer Address Low
$C2 = TAPBUFHI        ;Tape Buffer Address High
$C3 = TAPELSLO        ;Tape Memory Load/Save Low
$C4 = TAPELSHI        ;Tape Memory Load/Save High
$D7 = TAPTEMP         ;Tape Temporary Storage (Read) / Tape File Checksum (Write)

$029F = IRQTMPLO      ;IRQ Vector Low Temporary Storage
$02A0 = IRQTMPHI      ;IRQ Vector High Temporary Storage
$02A1 = RS232ENAB     ;RS232 NMI Enable
$02A2 = TAPCTRLA      ;CIA 1 Control Register A Activity
$02A3 = TAPICRFLG     ;CIA 1 Interrupt Control Register Flags
$02A4 = TAPCTRLB      ;CIA 1 Control Register B Activity

;---------------
```

;STATUS Bits for Tape End Or Identify (EOI)

Bit 2 ($04) = Short Block Error
Bit 3 ($08) = Long Block Error
Bit 4 ($10) = Read Error
Bit 5 ($20) = Checksum Error
Bit 6 ($40) = End Of File (EOF)
Bit 7 ($80) = End Of Tape (EOT)

;---------------
;Tape Header Format

1 Byte : File type

$01 = Relocatable Program
$02 = Data block for Data file
$03 = Non Relocatable Program
$04 = Data file
$05 = EOT

2 Bytes : Start Address Low/High
2 Bytes : End Address Low/High
16 Bytes : File Name

;---------------
;Pulse format

3 different pulse types are used

Measured on PAL

(S)hort : $30 = $180 = ~384 uS
(M)edium : $42 = $210 = ~528 uS
(L)ong   : $56 = $2B0 = ~688 us

According to code

(S)hort : $2C = $160 = ~352 uS
(M)edium : $40 = $200 = ~512 uS
(L)ong   : $54 = $2A0 = ~672 uS

Pulses are interpreted as a pair, except for the pilot/leader/trailer

(S,S,...) pilot / trailing tone
(S,M) = 0 bit
(S,L) = error
(M,S) = 1 bit
(M,M) = error
(M,L) = error
(L,S) = end-of-data marker
(L,M) = new-data marker
(L,L) = inter-block marker

;---------------

Each byte is encoded as a stream of 20 bits.

```
(L,M) (?,?) (?,?) (?,?) (?,?) (?,?) (?,?) (?,?) (?,?) (?,?)
  |     |     |     |     |     |     |     |     |     |
  |    bit0  bit1  bit2  bit3  bit4  bit5  bit6  bit7   |
  |                                                     |
new-data marker                                      parity
```

The Endianess is LSbF.

Parity for the data bits is 1 xored will all other data bits in the byte.

The data stream is terminated when a (L,S) pair is received after a data byte.

Each block also has a checksum which is the last byte in the block.

The data block checksum is calculated 0 XOR all bytes.

The first block in a pair has a Pilot/Leader of $6A00 short pulses.

The first header block in a pair has the sync sequence $89,$88,$87,$86,$85,$84,$83,$82,$81

The second block in a pair has a Pilot/Leader of $1500 short pulses.

The second header block in a pair has the sync sequence $09,$08,$07,$06,$05,$04,$03,$02,$01

Between each block pair is a (L,L) inter-block marker

Between each block set is a (L,L) inter-block marker

```
;---------------
;Notes
```

The error checking/correction code can only log and fix up to $3f bytes, this could be improved upon. More complex code could provide better error correction across both copies of the files, and a larger number of errors.
Pilot/Leader is not assumed to be bit pairs, syncing occurs normally with a new data marker.
This disassembly is believed to be correct, there are possible errors. Most other documentation of the code has naming errors, or is flat out wrong. Newer versions with any corrections will be forthcoming. Please contact Fungus if you spot an error(s).

```
;---------------
;Disassembly

TAPPAUSE                            ;Tape Pause after header found

.C:e4e0  69 02       ADC #$02
.C:e4e2  A4 91       LDY $91
.C:e4e4  C8          INY
.C:e4e5  D0 04       BNE $E4EB
.C:e4e7  C5 A1       CMP $A1
.C:e4e9  D0 F7       BNE $E4E2
.C:e4eb  60          RTS

.C:ee8e  AD 00 DD    LDA $DD00      ;Set Vic Bank and Serial Outputs
.C:ee91  09 10       ORA #$10
.C:ee93  8D 00 DD    STA $DD00
.C:ee96  60          RTS

DISRS232                            ;Check and Disable RS-232 NMI

.C:f0a4  48          PHA
.C:f0a5  AD A1 02    LDA $02A1
.C:f0a8  F0 11       BEQ $F0BB
.C:f0aa  AD A1 02    LDA $02A1
.C:f0ad  29 03       AND #$03
.C:f0af  D0 F9       BNE $F0AA
.C:f0b1  A9 10       LDA #$10
.C:f0b3  8D 0D DD    STA $DD0D
.C:f0b6  A9 00       LDA #$00
.C:f0b8  8D A1 02    STA $02A1
.C:f0bb  68          PLA
.C:f0bc  60          RTS

>C:f0bd  0d 49 2f 4f  20 45 52 52  4f 52 20 a3  0d 53 45 41   .I/O ERROR ..SEA
>C:f0cd  52 43 48 49  4e 47 a0 46  4f 52 a0 0d  50 52 45 53   RCHING.FOR..PRES
>C:f0dd  53 20 50 4c  41 59 20 4f  4e 20 54 41  50 c5 50 52   S PLAY ON TAP.PR
>C:f0ed  45 53 53 20  52 45 43 4f  52 44 20 26  20 50 4c 41   ESS RECORD & PLA
>C:f0fd  59 20 4f 4e  20 54 41 50  c5 0d 4c 4f  41 44 49 4e   Y ON TAP..LOADIN
>C:f10d  c7 0d 53 41  56 49 4e 47  a0 0d 56 45  52 49 46 59   ..SAVING..VERIFY
>C:f11d  49 4e c7 0d  46 4f 55 4e  44 a0 0d 4f  4b 8d         IN..FOUND..OK.

CHRIN

.C:f157  A5 99       LDA $99        ;Get Input Device
.C:f159  D0 0B       BNE $F166      ;Branch if not the Keyboard
.C:f15b  A5 D3       LDA $D3
.C:f15d  85 CA       STA $CA
.C:f15f  A5 D6       LDA $D6
.C:f161  85 C9       STA $C9
.C:f163  4C 32 E6    JMP $E632      ;Go to Keyboard/Screen handler

not keyboard

.C:f166  C9 03       CMP #$03       ;Check if Screen
.C:f168  D0 09       BNE $F173      ;Branch if not the Screen
.C:f16a  85 D0       STA $D0
.C:f16c  A5 D5       LDA $D5
.C:f16e  85 C8       STA $C8
.C:f170  4C 32 E6    JMP $E632      ;Go to Keyboard/Screen handler
```

```
not screen

.C:f173  B0 38       BCS $F1AD      ;Higher than 3? Branch to Serial Handler
.C:f175  C9 02       CMP #$02       ;Check if RS-232
.C:f177  F0 3F       BEQ $F1B8      ;Branch if RS-232

must be tape

.C:f179  86 97       STX $97        ;Temp Save X
.C:f17b  20 99 F1    JSR $F199      ;GETTCHR
.C:f17e  B0 16       BCS $F196      ;Check EOI
.C:f180  48          PHA            ;Temp Save A
.C:f181  20 99 F1    JSR $F199      ;GETTCHR
.C:f184  B0 0D       BCS $F193      ;Check EOI
.C:f186  D0 05       BNE $F18D      ;Check EOF
.C:f188  A9 40       LDA #$40       ;Set ST EOF
.C:f18a  20 1C FE    JSR $FE1C      ;Print EOF

.C:f18d  C6 A6       DEC $A6        ;Decrement TAPBUFPTR
.C:f18f  A6 97       LDX $97        ;Temp Restore X
.C:f191  68          PLA            ;Temp Restore A
.C:f192  60          RTS

.C:f193  AA          TAX            ;Save Error Info
.C:f194  68          PLA            ;Clean up stack
.C:f195  8A          TXA            ;Restore Error Info

CHRIN Exit

.C:f196  A6 97       LDX $97        ;Temp Restore X
.C:f198  60          RTS            ;Return

GETTCHR

.C:f199  20 0D F8    JSR $F80D      ;INCTPTR
.C:f19c  D0 0B       BNE $F1A9      ;branch if buffer end not reached
.C:f19e  20 41 F8    JSR $F841      ;TAPREAD
.C:f1a1  B0 11       BCS $F1B4      ;Check EOI

.C:f1a3  A9 00       LDA #$00       ;Reset TAPBUFPTR
.C:f1a5  85 A6       STA $A6
.C:f1a7  F0 F0       BEQ $F199      ;Loop back

.C:f1a9  B1 B2       LDA ($B2),Y    ;Get Byte From Tape Buffer
.C:f1ab  18          CLC            ;Clear EOI
.C:f1ac  60          RTS            ;Return

.C:f1b4  60          RTS

CHKSTOP

.C:f6bc  AD 01 DC    LDA $DC01
.C:f6bf  CD 01 DC    CMP $DC01
.C:f6c2  D0 F8       BNE $F6BC
.C:f6c4  AA          TAX
.C:f6c5  30 13       BMI $F6DA
.C:f6c7  A2 BD       LDX #$BD
.C:f6c9  8E 00 DC    STX $DC00
.C:f6cc  AE 01 DC    LDX $DC01
.C:f6cf  EC 01 DC    CPX $DC01
.C:f6d2  D0 F8       BNE $F6CC
.C:f6d4  8D 00 DC    STA $DC00
.C:f6d7  E8          INX
.C:f6d8  D0 02       BNE $F6DC
.C:f6da  85 91       STA $91        ;STKEY
.C:f6dc  60          RTS

GETHEADER

.C:f72c  A5 93       LDA $93        ;Get Load Verify Flag
.C:f72e  48          PHA            ;Save Flag
.C:f72f  20 41 F8    JSR $F841      ;TAPREAD
.C:f732  68          PLA            ;Restore Flag
.C:f733  85 93       STA $93
.C:f735  B0 32       BCS $F769      ;EOI
.C:f737  A0 00       LDY #$00       ;Index 0
.C:f739  B1 B2       LDA ($B2),Y    ;Get File Type
.C:f73b  C9 05       CMP #$05       ;EOT?
```

```
.C:f73d  F0 2A       BEQ $F769
.C:f73f  C9 01       CMP #$01       ;Relocatable Program?
.C:f741  F0 08       BEQ $F74B
.C:f743  C9 03       CMP #$03       ;Non-Relocatable Program?
.C:f745  F0 04       BEQ $F74B
.C:f747  C9 04       CMP #$04       ;Sequential File?
.C:f749  D0 E1       BNE $F72C      ;Loop if invalid filetype

VALIDFILE

.C:f74b  AA          TAX            ;File Type to X
.C:f74c  24 9D       BIT $9D        ;Check MSG Flag
.C:f74e  10 17       BPL $F767      ;Skip MSG

.C:f750  A0 63       LDY #$63
.C:f752  20 2F F1    JSR $F12F      ;Print "FOUND"
.C:f755  A0 05       LDY #$05       ;Print Filename
.C:f757  B1 B2       LDA ($B2),Y
.C:f759  20 D2 FF    JSR $FFD2
.C:f75c  C8          INY
.C:f75d  C0 15       CPY #$15
.C:f75f  D0 F6       BNE $F757
.C:f761  A5 A1       LDA $A1        ;Get Jiffy Clock 256 Jiffies
.C:f763  20 E0 E4    JSR $E4E0      ;TAPPAUSE
.C:f766  EA          NOP
.C:f767  18          CLC
.C:f768  88          DEY
.C:f769  60          RTS            ;Return

CREATEHDR                           ;Create Tape Header

.C:f76a  85 9E       STA $9E        ;Save Block Type
.C:f76c  20 D0 F7    JSR $F7D0      ;GETBUFSTRT
.C:f76f  90 5E       BCC $F7CF      ;Exit if not Allocated
.C:f771  A5 C2       LDA $C2        ;Save Tape Buff Start High
.C:f773  48          PHA
.C:f774  A5 C1       LDA $C1        ;Save Tape Buff Start Low
.C:f776  48          PHA
.C:f777  A5 AF       LDA $AF        ;Save Tape End High
.C:f779  48          PHA
.C:f77a  A5 AE       LDA $AE        ;Save Tape End Low
.C:f77c  48          PHA
.C:f77d  A0 BF       LDY #$BF       ;Buffer length
.C:f77f  A9 20       LDA #$20       ;Clear Buffer
.C:f781  91 B2       STA ($B2),Y
.C:f783  88          DEY
.C:f784  D0 FB       BNE $F781
.C:f786  A5 9E       LDA $9E        ;File Type
.C:f788  91 B2       STA ($B2),Y    ;Store It
.C:f78a  C8          INY
.C:f78b  A5 C1       LDA $C1        ;Get Start Low
.C:f78d  91 B2       STA ($B2),Y    ;Store It
.C:f78f  C8          INY
.C:f790  A5 C2       LDA $C2        ;Get Start High
.C:f792  91 B2       STA ($B2),Y    ;Store It
.C:f794  C8          INY
.C:f795  A5 AE       LDA $AE        ;Get End Low
.C:f797  91 B2       STA ($B2),Y    ;Store It
.C:f799  C8          INY
.C:f79a  A5 AF       LDA $AF        ;Get End High
.C:f79c  91 B2       STA ($B2),Y    ;Store It
.C:f79e  C8          INY
.C:f79f  84 9F       STY $9F        ;Save Buffer Index
.C:f7a1  A0 00       LDY #$00
.C:f7a3  84 9E       STY $9E        ;Data Index
.C:f7a5  A4 9E       LDY $9E
.C:f7a7  C4 B7       CPY $B7        ;Filename Length
.C:f7a9  F0 0C       BEQ $F7B7      ;Branch if No Filename or Filename Length reached
.C:f7ab  B1 BB       LDA ($BB),Y    ;Get Filename
.C:f7ad  A4 9F       LDY $9F        ;Buffer Index
.C:f7af  91 B2       STA ($B2),Y    ;Put Filename in Buffer
.C:f7b1  E6 9E       INC $9E        ;Increment Data Index
.C:f7b3  E6 9F       INC $9F        ;Increment Buffer Index
.C:f7b5  D0 EE       BNE $F7A5      ;Loop
.C:f7b7  20 D7 F7    JSR $F7D7      ;Set Buffer End
.C:f7ba  A9 69       LDA #$69       ;Time for header (lol)
.C:f7bc  85 AB       STA $AB        ;Set Tape Leader Counter MSB
.C:f7be  20 6B F8    JSR $F86B      ;Write Header
```

```
.C:f7c1  A8          TAY
.C:f7c2  68          PLA
.C:f7c3  85 AE       STA $AE         ;Restore Tape End Low
.C:f7c5  68          PLA
.C:f7c6  85 AF       STA $AF         ;Restore Tape End High
.C:f7c8  68          PLA
.C:f7c9  85 C1       STA $C1         ;Restore Tape Buff Start Low
.C:f7cb  68          PLA
.C:f7cc  85 C2       STA $C2         ;Restore Tape Buff Start High
.C:f7ce  98          TYA
.C:f7cf  60          RTS             ;Return

GETBUFSTRT                           ;Get Tape Buffer Start

.C:f7d0  A6 B2       LDX $B2         ;Tape Buffer Start Low
.C:f7d2  A4 B3       LDY $B3         ;Tape Buffer Start High
.C:f7d4  C0 02       CPY #$02
.C:f7d6  60          RTS

SETBUFEND                            ;Set Tape Buffer End

.C:f7d7  20 D0 F7    JSR $F7D0       ;GETBUFSTRT
.C:f7da  8A          TXA
.C:f7db  85 C1       STA $C1         ;Tape Buffer Low
.C:f7dd  18          CLC
.C:f7de  69 C0       ADC #$C0        ;End Address Low
.C:f7e0  85 AE       STA $AE
.C:f7e2  98          TYA
.C:f7e3  85 C2       STA $C2         ;Tape Buffer High
.C:f7e5  69 00       ADC #$00
.C:f7e7  85 AF       STA $AF         ;End Address High
.C:f7e9  60          RTS

TAPSRCHHDR                           ;Search For Tape Header

.C:f7ea  20 2C F7    JSR $F72C       ;GETHEADER
.C:f7ed  B0 1D       BCS $F80C       ;EOI
.C:f7ef  A0 05       LDY #$05        ;Initialize Filename Index in Tape Buffer
.C:f7f1  84 9F       STY $9F
.C:f7f3  A0 00       LDY #$00        ;Initialize Filename Index
.C:f7f5  84 9E       STY $9E
.C:f7f7  C4 B7       CPY $B7         ;Check Filename length
.C:f7f9  F0 10       BEQ $F80B       ;Branch on max length
.C:f7fb  B1 BB       LDA ($BB),Y     ;Check if desired filename
.C:f7fd  A4 9F       LDY $9F
.C:f7ff  D1 B2       CMP ($B2),Y
.C:f801  D0 E7       BNE $F7EA       ;Search next header if not desired filename
.C:f803  E6 9E       INC $9E
.C:f805  E6 9F       INC $9F
.C:f807  A4 9E       LDY $9E
.C:f809  D0 EC       BNE $F7F7
.C:f80b  18          CLC             ;Filename Found
.C:f80c  60          RTS             ;Return

INCTPTR

.C:f80d  20 D0 F7    JSR $F7D0       ;GETBUFA
.C:f810  E6 A6       INC $A6         ;Increment pointer
.C:f812  A4 A6       LDY $A6         ;Get pointer
.C:f814  C0 C0       CPY #$C0        ;Check top of buffer
.C:f816  60          RTS

PLAYWAIT

.C:f817  20 2E F8    JSR $F82E       ;Test Tape Switch
.C:f81a  F0 1A       BEQ $F836       ;Return if Pressed
.C:f81c  A0 1B       LDY #$1B
.C:f81e  20 2F F1    JSR $F12F       ;Print "PRESS PLAY ON TAPE"
.C:f821  20 D0 F8    JSR $F8D0       ;Check STOP Key
.C:f824  20 2E F8    JSR $F82E       ;Test Tape Switch
.C:f827  D0 F8       BNE $F821       ;Loop while Play not pressed
.C:f829  A0 6A       LDY #$6A
.C:f82b  4C 2F F1    JMP $F12F       ;Print "OK" and Return

PLAYTEST

.C:f82e  A9 10       LDA #$10        ;Test Tape Switch
.C:f830  24 01       BIT $01
```

```
.C:f832  D0 02       BNE $F836      ;Loop if Not Pressed
.C:f834  24 01       BIT $01        ;Check Again to Debounce
.C:f836  18          CLC
.C:f837  60          RTS

RECORDWAIT

.C:f838  20 2E F8    JSR $F82E      ;Test Tape Switch
.C:f83b  F0 F9       BEQ $F836
.C:f83d  A0 2E       LDY #$2E
.C:f83f  D0 DD       BNE $F81E      ;Print "PRESS RECORD AND PLAY ON TAPE" and return

TAPREAD

.C:f841  A9 00       LDA #$00
.C:f843  85 90       STA $90        ;Initialize Status
.C:f845  85 93       STA $93        ;Set Load/Verify flag to Load
.C:f847  20 D7 F7    JSR $F7D7      ;Get Buffer A
.C:f84a  20 17 F8    JSR $F817      ;PLAYWAIT
.C:f84d  B0 1F       BCS $F86E      ;EOI

.C:f84f  78          SEI            ;Disable interrupts
.C:f850  A9 00       LDA #$00
.C:f852  85 AA       STA $AA        ;Initialize Tape Byte Flag
.C:f854  85 B4       STA $B4        ;Initialize Tape Bit In/Out Flag
.C:f856  85 B0       STA $B0        ;Initialize Tape Pulse Length Low
.C:f858  85 9E       STA $9E        ;Initialize Tape Pass 1 Error Log Index
.C:f85a  85 9F       STA $9F        ;Initialize Tape Pass 2 Error Log Index
.C:f85c  85 9C       STA $9C        ;Initialize Tape Byte Received Flag
.C:f85e  A9 90       LDA #$90       ;CIA 1 ICR Value
.C:f860  A2 0E       LDX #$0E       ;Tape Read IRQ Vector Table Offset
.C:f862  D0 11       BNE $F875      ;Branch always

TAPWRITE

.C:f864  20 D7 F7    JSR $F7D7      ;Get Buffer A
.C:f867  A9 14       LDA #$14
.C:f869  85 AB       STA $AB        ;Initialize Tape Leader Count MSB

TAPHDRWRITE

.C:f86b  20 38 F8    JSR $F838      ;RECORDWAIT
.C:f86e  B0 6C       BCS $F8DC      ;EOI
.C:f870  78          SEI
.C:f871  A9 82       LDA #$82       ;Enable Timer IRQ (CIA1 ICR Value %10000010)
.C:f873  A2 08       LDX #$08       ;Tape Write IRQ Vector Table Offset

TAPIRQINIT

.C:f875  A0 7F       LDY #$7F
.C:f877  8C 0D DC    STY $DC0D      ;Disable IRQs and clear flags
.C:f87a  8D 0D DC    STA $DC0D      ;Enable Timber B IRQ
.C:f87d  AD 0E DC    LDA $DC0E
.C:f880  09 19       ORA #$19
.C:f882  8D 0F DC    STA $DC0F      ;Start Timer B, Set one shot mode, Force load strobe
.C:f885  29 91       AND #$91
.C:f887  8D A2 02    STA $02A2      ;TAPCTRLB
.C:f88a  20 A4 F0    JSR $F0A4      ;Check and Disable RS232 NMI
.C:f88d  AD 11 D0    LDA $D011
.C:f890  29 EF       AND #$EF
.C:f892  8D 11 D0    STA $D011      ;Turn Screen off During IO
.C:f895  AD 14 03    LDA $0314      ;Save IRQ Vector
.C:f898  8D 9F 02    STA $029F
.C:f89b  AD 15 03    LDA $0315
.C:f89e  8D A0 02    STA $02A0
.C:f8a1  20 BD FC    JSR $FCBD      ;Set IRQ Vector
.C:f8a4  A9 02       LDA #$02
.C:f8a6  85 BE       STA $BE        ;Initialize Tape Block Copy Count
.C:f8a8  20 97 FB    JSR $FB97      ;TAPINITZP (initialize local counters)
.C:f8ab  A5 01       LDA $01
.C:f8ad  29 1F       AND #$1F
.C:f8af  85 01       STA $01        ;Turn Motor On
.C:f8b1  85 C0       STA $C0        ;Set Button Status
.C:f8b3  A2 FF       LDX #$FF       ;Wait ~325 mS (delay between blocks)
.C:f8b5  A0 FF       LDY #$FF
.C:f8b7  88          DEY
.C:f8b8  D0 FD       BNE $F8B7
.C:f8ba  CA          DEX
```

```
.C:f8bb  D0 F8       BNE $F8B5
.C:f8bd  58          CLI
.C:f8be  AD A0 02    LDA $02A0      ;Get Temp IRQ Vector High / Wait for Tape
.C:f8c1  CD 15 03    CMP $0315      ;Compare to Current IRQ Vector High
.C:f8c4  18          CLC
.C:f8c5  F0 15       BEQ $F8DC      ;Branch if Equal
.C:f8c7  20 D0 F8    JSR $F8D0      ;Check Stop
.C:f8ca  20 BC F6    JSR $F6BC      ;Update Time
.C:f8cd  4C BE F8    JMP $F8BE      ;Loop

.C:f8d0  20 E1 FF    JSR $FFE1      ;Check STOP Key
.C:f8d3  18          CLC
.C:f8d4  D0 0B       BNE $F8E1      ;Return if not pressed
.C:f8d6  20 93 FC    JSR $FC93      ;Restore default IRQ
.C:f8d9  38          SEC            ;Set EOI
.C:f8da  68          PLA
.C:f8db  68          PLA

.C:f8dc  A9 00       LDA #$00
.C:f8de  8D A0 02    STA $02A0      ;Clear IRQ Vector Temp Save High
.C:f8e1  60          RTS            ;Return

TAPSETRDT                           ;Set Read Timing

.C:f8e2  86 B1       STX $B1        ;Set Tape Pulse Length High (constant for timeout)
.C:f8e4  A5 B0       LDA $B0        ;Get Tape Pulse Length Low
.C:f8e6  0A          ASL A          ;Multiply by 4
.C:f8e7  0A          ASL A
.C:f8e8  18          CLC
.C:f8e9  65 B0       ADC $B0        ;Add Tape Pulse Length Low, Total Multiplication by 5
.C:f8eb  18          CLC
.C:f8ec  65 B1       ADC $B1        ;Add Tape Pulse Length High (constant for timeout)
.C:f8ee  85 B1       STA $B1        ;Set Tape Pulse Length High (constant for timeout)
.C:f8f0  A9 00       LDA #$00
.C:f8f2  24 B0       BIT $B0        ;Check High Bit of Tape Pulse Length Low
.C:f8f4  30 01       BMI $F8F7      ;Branch if set (no adjustment)

.C:f8f6  2A          ROL A          ;Rotate Carry into A Low Bit and High Bit into Carry (adjustment plus)
.C:f8f7  06 B1       ASL $B1        ;Shift Carry bit into Tape Pulse Length High Low Bit and High Bit Into
Carry (multiply correction by 4)
.C:f8f9  2A          ROL A          ;Rotate Carry into A Low Bit and High Bit into Carry
.C:f8fa  06 B1       ASL $B1        ;Shift Carry bit into Tape Pulse Length High Low Bit and High Bit Into
Carry
.C:f8fc  2A          ROL A          ;Rotate Carry into A Low Bit and High Bit into Carry
.C:f8fd  AA          TAX            ;Temp Save Adjustment in X
.C:f8fe  AD 06 DC    LDA $DC06      ;Get Timer B Low
.C:f901  C9 16       CMP #$16       ;Check if less than 22 uS (make sure there's enough cycles left before
rollover)
.C:f903  90 F9       BCC $F8FE      ;Loop if too short

.C:f905  65 B1       ADC $B1        ;Add Tape Pulse Length High (calculate and store adjusted time)
.C:f907  8D 04 DC    STA $DC04      ;Set Timer A Low
.C:f90a  8A          TXA            ;Restore Adjustment from X
.C:f90b  6D 07 DC    ADC $DC07      ;Add with Timer B High (Adjust for high time count)
.C:f90e  8D 05 DC    STA $DC05      ;Set Timer A High Adjusted
.C:f911  AD A2 02    LDA $02A2      ;Get TAPCTRLA
.C:f914  8D 0E DC    STA $DC0E      ;Set CIA Timer Control A
.C:f917  8D A4 02    STA $02A4      ;Set TAPCTRLB (stupid in source code hah), Non Zero means a /flag irq
has not occured
.C:f91a  AD 0D DC    LDA $DC0D      ;Read CIA Interrupt Control Register
.C:f91d  29 10       AND #$10       ;Check if an IRQ was triggered
.C:f91f  F0 09       BEQ $F92A      ;Branch not triggered

                                    ;IRQ occured while adjusting, so fake the IRQ
.C:f921  A9 F9       LDA #$F9       ;Push TAPREADIRQ (fake IRQ return address set to $f92b, return address -
$01)
.C:f923  48          PHA
.C:f924  A9 2A       LDA #$2A
.C:f926  48          PHA
.C:f927  4C 43 FF    JMP $FF43      ;JMP to Tape Fake IRQ

.C:f92a  58          CLI            ;Clear Interrupt Disable
.C:f92b  60          RTS            ;Return

TAPREADIRQ

.C:f92c  AE 07 DC    LDX $DC07      ;Read Timer B High (time since last irq)
.C:f92f  A0 FF       LDY #$FF       ;Set Y to $ff = Max Timer Value High
```

```
.C:f931  98          TYA            ;Set A to $ff = Max Timer Value Low
.C:f932  ED 06 DC    SBC $DC06      ;Subtract with Timer B Low (Pulse Length Low)
.C:f935  EC 07 DC    CPX $DC07      ;Compare X with Timer B High (Pulse Length High)
.C:f938  D0 F2       BNE $F92C      ;Loop if not equal (Stabilize the pulse width if timer low rolled over
while reading values, a few
                                    ;cycles won't matter)

.C:f93a  86 B1       STX $B1        ;Set Tape Pulse Length High
.C:f93c  AA          TAX            ;Transfer A to X
.C:f93d  8C 06 DC    STY $DC06      ;Set Timer B Low to $ff
.C:f940  8C 07 DC    STY $DC07      ;Set Timer B High to $ff
.C:f943  A9 19       LDA #$19
.C:f945  8D 0F DC    STA $DC0F      ;Start Timer B, One Shot, Force Load Strobe
.C:f948  AD 0D DC    LDA $DC0D      ;Get Interrupt Control Register Flags / Acknowledge IRQ ($90 %10010000 =
flag line caused IRQ, cia
                                    ;caused IRQ)
.C:f94b  8D A3 02    STA $02A3      ;Save Interrupt Control Register Flags

.C:f94e  98          TYA            ;Max Timer Value Constant ($ff)
.C:f94f  E5 B1       SBC $B1        ;Subtract Tape Pulse Length High
.C:f951  86 B1       STX $B1        ;Set Tape Pulse Length High
.C:f953  4A          LSR A          ;Rotate Low bit into Carry
.C:f954  66 B1       ROR $B1        ;Rotate Tape Pulse Length High
.C:f956  4A          LSR A          ;Rotate Low bit into Carry
.C:f957  66 B1       ROR $B1        ;Rotate Tape Pulse Length High
.C:f959  A5 B0       LDA $B0        ;Get Tape Pulse Length Low
.C:f95b  18          CLC
.C:f95c  69 3C       ADC #$3C       ;Add 60 cycles
.C:f95e  C5 B1       CMP $B1        ;Compare with Tape Pulse Length High
.C:f960  B0 4A       BCS $F9AC      ;Branch if pulse is less than Min (noise)

.C:f962  A6 9C       LDX $9C        ;Get Tape Byte Received Flag
.C:f964  F0 03       BEQ $F969      ;Branch if Byte Received
.C:f966  4C 60 FA    JMP $FA60      ;Last bit, so finish byte

.C:f969  A6 A3       LDX $A3        ;Get Tape Byte Bits Count
.C:f96b  30 1B       BMI $F988      ;Branch if Bit 7 set (rolled over to $ff)

                                    ;Check Short Pulse
.C:f96d  A2 00       LDX #$00       ;Short Pulse = 0
.C:f96f  69 30       ADC #$30       ;Add 48 cycles
.C:f971  65 B0       ADC $B0        :Add Tape Pulse Length Low
.C:f973  C5 B1       CMP $B1        ;Compare with Tape Pulse Length High
.C:f975  B0 1C       BCS $F993      ;Branch to Short Pulse Found

                                    ;Check Medium Pulse
.C:f977  E8          INX            ;Medium Pulse = 1
.C:f978  69 26       ADC #$26       ;Add 38 cycles
.C:f97a  65 B0       ADC $B0        ;Add Tape Pulse Length Low
.C:f97c  C5 B1       CMP $B1        ;Compare with Tape Pulse Length High
.C:f97e  B0 17       BCS $F997      ;Branch to Medium Pulse Found

                                    ;Check Long Pulse
.C:f980  69 2C       ADC #$2C       ;Add 44 cycles
.C:f982  65 B0       ADC $B0        ;Add Tape Pulse Length Low
.C:f984  C5 B1       CMP $B1        ;Compare with Tape Pulse Length High
.C:f986  90 03       BCC $F98B      ;Branch on Error/Very Long Pulse
.C:f988  4C 10 FA    JMP $FA10      ;JMP to Long Pulse Found

.C:f98b  A5 B4       LDA $B4        ;Get Tape Bit In/Out Flag (0 = sync, 1 = byte start)
.C:f98d  F0 1D       BEQ $F9AC      ;Branch if Not Syncing

.C:f98f  85 A8       STA $A8        ;Set Read Error
.C:f991  D0 19       BNE $F9AC      ;Branch always

                                    ;Short Pulse Found
.C:f993  E6 A9       INC $A9        ;Increment Tape Short Pulse Count (increment on 0 bit)
.C:f995  B0 02       BCS $F999      ;Branch on Carry Set

                                    ;Medium Pulse Found
.C:f997  C6 A9       DEC $A9        ;Decrement Tape Short Pulse Count (decrement on 1 bit)

.C:f999  38          SEC
.C:f99a  E9 13       SBC #$13       ;Subtract 19 cycles
.C:f99c  E5 B1       SBC $B1        ;Subtract Tape Pulse Length High
.C:f99e  65 92       ADC $92        ;Add Tape Timing Constant
.C:f9a0  85 92       STA $92        ;Set Tape Timing Constant, calculate if constant needs changed (software
servo)
```

```
.C:f9a2  A5 A4       LDA $A4        ;Get Tape Bit Pair (dipole)
.C:f9a4  49 01       EOR #$01       ;Flip Bit 1
.C:f9a6  85 A4       STA $A4        ;Set Tape Bit Pair (dipole)
.C:f9a8  F0 2B       BEQ $F9D5      ;Branch if 0 (check if 1st or 2nd bit of pair)

.C:f9aa  86 D7       STX $D7        ;Save First Pulse Length

.C:f9ac  A5 B4       LDA $B4        ;Get Tape Bit In/Out Flag
.C:f9ae  F0 22       BEQ $F9D2      ;Branch if no Byte Start (Exit IRQ)

.C:f9b0  AD A3 02    LDA $02A3      ;Get Interrupt Control Register Flags
.C:f9b3  29 01       AND #$01       ;Check if Timer A timed out
.C:f9b5  D0 05       BNE $F9BC      ;Branch if Timer A timed out

.C:f9b7  AD A4 02    LDA $02A4      ;Load TAPCTRLB (Timer A didn't time out) (stupid in source code hah)
.C:f9ba  D0 16       BNE $F9D2      ;Branch if not start

.C:f9bc  A9 00       LDA #$00
.C:f9be  85 A4       STA $A4        ;Set Tape Bit Pair (dipole) to Start
.C:f9c0  8D A4 02    STA $02A4      ;Set TAPCTRLB (stupid in source code hah)
.C:f9c3  A5 A3       LDA $A3        ;Get Tape Byte Bits Count
.C:f9c5  10 30       BPL $F9F7      ;Branch on bit 7 clear (still getting bits)
.C:f9c7  30 BF       BMI $F988      ;Branch on bit 7 set (check parity)

                                    ;Check for Parity Error
.C:f9c9  A2 A6       LDX #$A6       ;Setup for Long Timeout
.C:f9cb  20 E2 F8    JSR $F8E2      ;Set Tape Read Timing
.C:f9ce  A5 9B       LDA $9B        ;Get Parity
.C:f9d0  D0 B9       BNE $F98B      ;Branch if not 0 (set error)
.C:f9d2  4C BC FE    JMP $FEBC      ;Exit IRQ

.C:f9d5  A5 92       LDA $92        ;Get TAPTIMCON adjustment (adjust software servo)
.C:f9d7  F0 07       BEQ $F9E0      ;Branch if 0 (no pulse length adjustment)
.C:f9d9  30 03       BMI $F9DE      ;Branch if Bit 7 set (longer pulse length adjustment)
.C:f9db  C6 B0       DEC $B0        ;Decrement Tape Pulse Length Low, shorter pulse length (shorter pulse
length adjustment)
.C:f9dd  2C          BIT $XXXX      ;Skip next instruction
.C:f9de  E6 B0       INC $B0        ;Increment Tape Pulse Length Low, longer pulse length (longer pulse
length adjustment)

.C:f9e0  A9 00       LDA #$00       ;Clear timing constant adjustement value
.C:f9e2  85 92       STA $92        ;Set TAPTIMCON to no adjustment
.C:f9e4  E4 D7       CPX $D7        ;Compare with First Bit Pulse Length
.C:f9e6  D0 0F       BNE $F9F7      ;Branch if not Equal, Process Data

.C:f9e8  8A          TXA            ;Move Pulse Length to A
.C:f9e9  D0 A0       BNE $F98B      ;Branch if not 0, Error (two medium pulses occured)

.C:f9eb  A5 A9       LDA $A9        ;Get Tape Short Pulse Count
.C:f9ed  30 BD       BMI $F9AC      ;Branch if Bit 7 set (we're in leader/pilot)

.C:f9ef  C9 10       CMP #$10       ;Compare to 16
.C:f9f1  90 B9       BCC $F9AC      ;Branch if less (continue)

.C:f9f3  85 96       STA $96        ;Set Tape Bit Pair Count Flag (between blocks)
.C:f9f5  B0 B5       BCS $F9AC      ;Branch always (Exit IRQ)

.C:f9f7  8A          TXA            ;Move X to A (input bit)
.C:f9f8  45 9B       EOR $9B        ;EOR Tape Parity
.C:f9fa  85 9B       STA $9B        ;Store Tape Parity
.C:f9fc  A5 B4       LDA $B4        ;Get Tape Bit In/Out Flag (data)
.C:f9fe  F0 D2       BEQ $F9D2      ;Branch if 0 (no data)

.C:fa00  C6 A3       DEC $A3        ;Decrement Tape Byte Bits Count
.C:fa02  30 C5       BMI $F9C9      ;Branch if Rollover (Check Parity)

.C:fa04  46 D7       LSR $D7        ;Shift Right Pulse Length (Short = 0, Medium = 1)
.C:fa06  66 BF       ROR $BF        ;Rotate Right Tape Bits In/Out
.C:fa08  A2 DA       LDX #$DA       ;Setup for next dipole
.C:fa0a  20 E2 F8    JSR $F8E2      ;Set Tape Read Timing
.C:fa0d  4C BC FE    JMP $FEBC      ;Exit IRQ

.C:fa10  A5 96       LDA $96        ;Get Tape Leader Bit Pair Count Flag (sync)
.C:fa12  F0 04       BEQ $FA18      ;Branch if 0 (no sync)

.C:fa14  A5 B4       LDA $B4        ;Get Tape Bit In/Out Flag (data)
.C:fa16  F0 07       BEQ $FA1F      ;Branch if 0 (no data)
```

```
.C:fa18  A5 A3       LDA $A3        ;Get Tape Byte Bits Count (end of data byte)
.C:fa1a  30 03       BMI $FA1F      ;Branch if Rollover (Byte Received, adjust for long)

.C:fa1c  4C 97 F9    JMP $F997      ;Treat as Medium Length Pulse

.C:fa1f  46 B1       LSR $B1        ;Shift Right Pulse Length High
.C:fa21  A9 93       LDA #$93       ;Load A with 147 uS (medium pulse length)
.C:fa23  38          SEC
.C:fa24  E5 B1       SBC $B1        ;Subtract with Tape Pulse Length High
.C:fa26  65 B0       ADC $B0        ;Add with Tape Pulse Length Low
.C:fa28  0A          ASL A          ;Shift result Left
.C:fa29  AA          TAX            ;Save A
.C:fa2a  20 E2 F8    JSR $F8E2      ;Call TAPSETRDT (set timeout for last bit)
.C:fa2d  E6 9C       INC $9C        ;Increment Tape Byte Received Flag
.C:fa2f  A5 B4       LDA $B4        ;Load Tape Bit In/Out Flag (sync)
.C:fa31  D0 11       BNE $FA44      ;Branch if not 0 (Exit IRQ)

.C:fa33  A5 96       LDA $96        ;Get Tape Leader Bit Pair Count Flag (throw away data until sync)
.C:fa35  F0 26       BEQ $FA5D      ;Branch if 0 (No Sync)

.C:fa37  85 A8       STA $A8        ;Set Tape Read Error
.C:fa39  A9 00       LDA #$00
.C:fa3b  85 96       STA $96        ;Set Tape Leader Bit Pair Count (16 sync flag)
.C:fa3d  A9 81       LDA #$81
.C:fa3f  8D 0D DC    STA $DC0D      ;Enable Timer IRQ
.C:fa42  85 B4       STA $B4        ;Set Tape Bit In/Out Flag
.C:fa44  A5 96       LDA $96        ;Get Tape Leader Bit Pair Count
.C:fa46  85 B5       STA $B5        ;Set Block Sync
.C:fa48  F0 09       BEQ $FA53      ;Branch if 0 (No Sync)

.C:fa4a  A9 00       LDA #$00       ;Turn off Byte Sync
.C:fa4c  85 B4       STA $B4        ;Set Tape Bit In/Out Flag (sync)
.C:fa4e  A9 01       LDA #$01
.C:fa50  8D 0D DC    STA $DC0D      ;Turn Off Timer IRQ

.C:fa53  A5 BF       LDA $BF        ;Get Tape Bits In/Out
.C:fa55  85 BD       STA $BD        ;Set Tape Byte In/Out
.C:fa57  A5 A8       LDA $A8        ;Get Tape Read Error
.C:fa59  05 A9       ORA $A9        ;OR with Tape Short Pulse Count
.C:fa5b  85 B6       STA $B6        ;Set Tape Read Error 2

.C:fa5d  4C BC FE    JMP $FEBC      ;Exit IRQ

                                    ;Reset for Next Byte
.C:fa60  20 97 FB    JSR $FB97      ;Init Tape ZP
.C:fa63  85 9C       STA $9C        ;Set Tape Byte Received Flag
.C:fa65  A2 DA       LDX #$DA       ;Initialize for Next Dipole
.C:fa67  20 E2 F8    JSR $F8E2      ;Set Read Timing
.C:fa6a  A5 BE       LDA $BE        ;Get Tape Block Counter
.C:fa6c  F0 02       BEQ $FA70      ;Branch if 0

.C:fa6e  85 A7       STA $A7        ;Set Tape Read Pass Count
.C:fa70  A9 0F       LDA #$0F       ;Test Mode %00001111
.C:fa72  24 AA       BIT $AA        ;test TAPBYTFLG (Data/Sync)
.C:fa74  10 17       BPL $FA8D      ;Branch if Bit 7 not set (Not waiting for Short Pulses = 0s)

.C:fa76  A5 B5       LDA $B5        ;Block Sync?
.C:fa78  D0 0C       BNE $FA86      ;Branch if not 0 (Yes, Wait for Sync)

.C:fa7a  A6 BE       LDX $BE        ;Get Tape Block Counter
.C:fa7c  CA          DEX            ;Decrement Counter
.C:fa7d  D0 0B       BNE $FA8A      ;Exit IRQ if not 0 (More Blocks to Load)

.C:fa7f  A9 08       LDA #$08       ;Long Block Error
.C:fa81  20 1C FE    JSR $FE1C      ;Set Tape Status to Long Block Error
.C:fa84  D0 04       BNE $FA8A      ;Exit IRQ

.C:fa86  A9 00       LDA #$00       ;Wait for Sync Mode
.C:fa88  85 AA       STA $AA        ;Set TAPBYTFLG (Data/Sync , Sync = 0)
.C:fa8a  4C BC FE    JMP $FEBC      ;Exit IRQ

.C:fa8d  70 31       BVS $FAC0      ;Branch on Bit 6 set (Loading)

.C:fa8f  D0 18       BNE $FAA9      ;Branch if not 0 (Syncing)

.C:fa91  A5 B5       LDA $B5        ;Get Block Sync
.C:fa93  D0 F5       BNE $FA8A      ;Branch if not 0 (Sync)
```

```
.C:fa95  A5 B6       LDA $B6        ;Get Tape Read Error 2
.C:fa97  D0 F1       BNE $FA8A      ;Branch if not 0 (Error Occured)

.C:fa99  A5 A7       LDA $A7        ;Get Tape Read Pass Count
.C:fa9b  4A          LSR A          ;Shift low bit into Carry
.C:fa9c  A5 BD       LDA $BD        ;Get Tape Byte
.C:fa9e  30 03       BMI $FAA3      ;Branch on Bit 7 set (First Header/Block Sync?)

.C:faa0  90 18       BCC $FABA      ;Branch if Carry clear (Expecting First Header/Block)

.C:faa2  18          CLC            ;Clear Carry flag
.C:faa3  B0 15       BCS $FABA      ;Branch if Carry Set (Expecting Second Header/Block)

.C:faa5  29 0F       AND #$0F       ;Mask off Hi Nybbel
.C:faa7  85 AA       STA $AA        ;Set Tape Input Byte Flag (Have Correct Block)

.C:faa9  C6 AA       DEC $AA        ;Decrement Tape Input Byte Flag
.C:faab  D0 DD       BNE $FA8A      ;Exit IRQ if not 0 (Real Data Yet?)

.C:faad  A9 40       LDA #$40       ;%01000000
.C:faaf  85 AA       STA $AA        ;Set Tape Input Byte Flag to Data Mode
.C:fab1  20 8E FB    JSR $FB8E      ;Set Load Address to Tape Buffer
.C:fab4  A9 00       LDA #$00
.C:fab6  85 AB       STA $AB        ;Set Tape Leader Counter to 0
.C:fab8  F0 D0       BEQ $FA8A      ;Branch always

.C:faba  A9 80       LDA #$80       ;%10000000
.C:fabc  85 AA       STA $AA        ;Set Tape Input Byte Flag to Ignore Mode
.C:fabe  D0 CA       BNE $FA8A      ;Branch always

.C:fac0  A5 B5       LDA $B5        ;Get End of Block ?
.C:fac2  F0 0A       BEQ $FACE      ;Branch if 0 (End of Block)

.C:fac4  A9 04       LDA #$04       ;Short Block Error
.C:fac6  20 1C FE    JSR $FE1C      ;Set Tape Status to Short Block Error
.C:fac9  A9 00       LDA #$00
.C:facb  4C 4A FB    JMP $FB4A      ;Set Tape Input Byte Flag

.C:face  20 D1 FC    JSR $FCD1      ;Call Tape Check EOF
.C:fad1  90 03       BCC $FAD6      ;Branch if not EOF

.C:fad3  4C 48 FB    JMP $FB48      ;Jump EOF

.C:fad6  A6 A7       LDX $A7        ;Get Tape Read Pass Count
.C:fad8  CA          DEX            ;Decrement it
.C:fad9  F0 2D       BEQ $FB08      ;Branch if 0 (second pass)

.C:fadb  A5 93       LDA $93        ;Get Load/Verify Flag
.C:fadd  F0 0C       BEQ $FAEB      ;Branch if Load
                                    ;Verify
.C:fadf  A0 00       LDY #$00       ;Initialize Buffer Index
.C:fae1  A5 BD       LDA $BD        ;Get Tape Byte Input/Output
.C:fae3  D1 AC       CMP ($AC),Y    ;Compare with byte saved
.C:fae5  F0 04       BEQ $FAEB      ;Branch if Equal

.C:fae7  A9 01       LDA #$01
.C:fae9  85 B6       STA $B6        ;Set Error Flag

                                    ;Load
.C:faeb  A5 B6       LDA $B6        ;Get Error Flag
.C:faed  F0 4B       BEQ $FB3A      ;Branch if 0

.C:faef  A2 3D       LDX #$3D       ;Max Error Log Size
.C:faf1  E4 9E       CPX $9E        ;Compare with Tape Error Log Index 1
.C:faf3  90 3E       BCC $FB33      ;Branch to Signify there was an Error because the log end was reached

.C:faf5  A6 9E       LDX $9E        ;Get Tape Error Log Index 1
.C:faf7  A5 AD       LDA $AD        ;Get Load Address High
.C:faf9  9D 01 01    STA $0101,X    ;Save Error Address High
.C:fafc  A5 AC       LDA $AC        ;Get Load Address Low
.C:fafe  9D 00 01    STA $0100,X    ;Save Error Address Low
.C:fb01  E8          INX            ;Adjust Error Log Index 1
.C:fb02  E8          INX
.C:fb03  86 9E       STX $9E        ;Save Error Log Index 1
.C:fb05  4C 3A FB    JMP $FB3A      ;Jump to No Error

.C:fb08  A6 9F       LDX $9F        ;Get Tape Error Log Index 2
.C:fb0a  E4 9E       CPX $9E        ;Compare to Tape Error Log Index 1
```

```
.C:fb0c  F0 35       BEQ $FB43      ;Branch if End of Error Log

.C:fb0e  A5 AC       LDA $AC        ;Get Load Address Low
.C:fb10  DD 00 01    CMP $0100,X    ;Compare to Error Log Address Low
.C:fb13  D0 2E       BNE $FB43      ;Branch if not Equal

.C:fb15  A5 AD       LDA $AD        ;Get Load Address High
.C:fb17  DD 01 01    CMP $0101,X    ;Compare to Error Log Address High
.C:fb1a  D0 27       BNE $FB43      ;Branch if not Equal

.C:fb1c  E6 9F       INC $9F        ;Adjust Error Log Index 2
.C:fb1e  E6 9F       INC $9F
.C:fb22  F0 0B       BEQ $FB2F      ;Branch always

.C:fb24  A5 BD       LDA $BD        ;Get Tape Byte Input/Output
.C:fb26  A0 00       LDY #$00
.C:fb28  D1 AC       CMP ($AC),Y    ;Compare with Byte Saved
.C:fb2a  F0 17       BEQ $FB43      ;Branch if equal

.C:fb2c  C8          INY            ;Increment Y
.C:fb2d  84 B6       STY $B6        ;Set Error Flag

                                    ;Check if Error Occured
.C:fb2f  A5 B6       LDA $B6        ;Get Error Flag
.C:fb31  F0 07       BEQ $FB3A      ;Branch if no Error

                                    ;An Error Occured
.C:fb33  A9 10       LDA #$10
.C:fb35  20 1C FE    JSR $FE1C      ;Set Tape Status to Read Error
.C:fb38  D0 09       BNE $FB43      ;Branch always

                                    ;No Error Branch
.C:fb3a  A5 93       LDA $93        ;Get Load/Verify Flag
.C:fb3c  D0 05       BNE $FB43      ;Branch if Verify

.C:fb3e  A8          TAY            ;Y = $00
.C:fb3f  A5 BD       LDA $BD        ;Get Tape Byte Input/Output
.C:fb41  91 AC       STA ($AC),Y    ;Store Byte in Memory

.C:fb43  20 DB FC    JSR $FCDB      ;Increment Load Address
.C:fb46  D0 43       BNE $FB8B      ;Branch always

.C:fb48  A9 80       LDA #$80       ;Set Mode Skip Data
.C:fb4a  85 AA       STA $AA        ;Set Tape Input Byte Flag
.C:fb4c  78          SEI            ;Disable IRQs
.C:fb4d  A2 01       LDX #$01
.C:fb4f  8E 0D DC    STX $DC0D      ;Turn timer IRQ off
.C:fb52  AE 0D DC    LDX $DC0D      ;Acknowledge IRQ
.C:fb55  A6 BE       LDX $BE        ;Get Tape Block Count
.C:fb57  CA          DEX            ;Decrement it
.C:fb58  30 02       BMI $FB5C      ;Branch if -1 (EOF)

.C:fb5a  86 BE       STX $BE        ;Save Tape Block Count
.C:fb5c  C6 A7       DEC $A7        ;Decrement Tape Pass Count
.C:fb5e  F0 08       BEQ $FB68      ;Branch if 0 (Done)
.C:fb60  A5 9E       LDA $9E        ;Get Tape Error Index
.C:fb62  D0 27       BNE $FB8B      ;Branch if not 0

.C:fb64  85 BE       STA $BE        ;Set Tape Block Copy Count $00 if no Errors
.C:fb66  F0 23       BEQ $FB8B      ;Branch always

.C:fb68  20 93 FC    JSR $FC93      ;Disable IRQs and turn Screen back on
.C:fb6b  20 8E FB    JSR $FB8E      ;Set Load Address Low/High to Tape Buffer
.C:fb6e  A0 00       LDY #$00       ;Set Y index to $00
.C:fb70  84 AB       STY $AB        ;Set Checksum to $00

.C:fb72  B1 AC       LDA ($AC),Y    ;Get Byte from Buffer
.C:fb74  45 AB       EOR $AB        ;EOR Checksum
.C:fb76  85 AB       STA $AB        ;Set Checksum
.C:fb78  20 DB FC    JSR $FCDB      ;Check EOF
.C:fb7b  20 D1 FC    JSR $FCD1      ;Increment Address
.C:fb7e  90 F2       BCC $FB72      ;Loop until done

.C:fb80  A5 AB       LDA $AB        ;Get Calculated Checksum
.C:fb82  45 BD       EOR $BD        ;Compare to Checksum
.C:fb84  F0 05       BEQ $FB8B      ;Branch no Error

.C:fb86  A9 20       LDA #$20
```

```
.C:fb88  20 1C FE    JSR $FE1C      ;Set Tape Status Checksum Error
.C:fb8b  4C BC FE    JMP $FEBC      ;Exit IRQ

SETTAPLODBUF                        ;Set the Load Address Low/High to the Tape Buffer

.C:fb8e  A5 C2       LDA $C2
.C:fb90  85 AD       STA $AD
.C:fb92  A5 C1       LDA $C1
.C:fb94  85 AC       STA $AC
.C:fb96  60          RTS

TAPINITZP

.C:fb97  A9 08       LDA #$08
.C:fb99  85 A3       STA $A3        ;Set Tape Byte Bits Count
.C:fb9b  A9 00       LDA #$00
.C:fb9d  85 A4       STA $A4        ;Set Tape Bit Pair (Dipole)
.C:fb9f  85 A8       STA $A8        ;Set Tape Read Error
.C:fba1  85 9B       STA $9B        ;Set Tape Parity
.C:fba3  85 A9       STA $A9        ;Set Tape Short Pulse Count
.C:fba5  60          RTS

TAPOUTPUTBIT

.C:fba6  A5 BD       LDA $BD        ;Tape Byte Input/Output
.C:fba8  4A          LSR A          ;Shift Low Bit Into Carry
.C:fba9  A9 60       LDA #$60       ;Short Pulse Length = 96 + 25
.C:fbab  90 02       BCC $FBAF      ;Branch if Short Pulse (0)
.C:fbad  A9 B0       LDA #$B0       ;Medium Pulse Length = 176 cycles/~uS

                                    ;Send Pulse
.C:fbaf  A2 00       LDX #$00       ;High Byte of Pulse Length

                                    ;Send Tape IO Pulse
.C:fbb1  8D 06 DC    STA $DC06      ;Set Pulse Length Lo
.C:fbb4  8E 07 DC    STX $DC07      ;Set Pulse Length Hi
.C:fbb7  AD 0D DC    LDA $DC0D      ;Acknowledge IRQ
.C:fbba  A9 19       LDA #$19       ;%00011001 (Start Timer B, One Shot Mode, Force Load Strobe)
.C:fbbc  8D 0F DC    STA $DC0F      ;ICR
.C:fbbf  A5 01       LDA $01        ;Get 6510 IO Register
.C:fbc1  49 08       EOR #$08       ;Invert Tape Output Line
.C:fbc3  85 01       STA $01        ;Set 6510 IO Register
.C:fbc5  29 08       AND #$08       ;Mask Tape Output Line
.C:fbc7  60          RTS            ;Return

.C:fbc8  38          SEC            ;Set Carry
.C:fbc9  66 B6       ROR $B6        ;Rotate Carry into Tape Bit Counter
.C:fbcb  30 3C       BMI $FC09      ;Branch if Byte Written

TAPWRITEIRQ2

.C:fbcd  A5 A8       LDA $A8        ;Check Send Long
.C:fbcf  D0 12       BNE $FBE3      ;Branch if no Send Long
.C:fbd1  A9 10       LDA #$10       ;Long Pulse Length 272 cycles/~uS
.C:fbd3  A2 01       LDX #$01
.C:fbd5  20 B1 FB    JSR $FBB1      ;Send Long Pulse
.C:fbd8  D0 2F       BNE $FC09      ;Branch if Output Line High (Rising Edge)
.C:fbda  E6 A8       INC $A8        ;Increment Send Long (Falling Edge)
.C:fbdc  A5 B6       LDA $B6        ;Get Tape Bits Count
.C:fbde  10 29       BPL $FC09      ;Branch if Positive
.C:fbe0  4C 57 FC    JMP $FC57      ;Check EOF

.C:fbe3  A5 A9       LDA $A9        ;Get Tape Short Pulse Count
.C:fbe5  D0 09       BNE $FBF0      ;Branch to Send Bit
.C:fbe7  20 AD FB    JSR $FBAD      ;Send Medium Pulse
.C:fbea  D0 1D       BNE $FC09      ;Branch if Output Line High (Rising Edge)
.C:fbec  E6 A9       INC $A9        ;Increment Tape Short Pulse Count (Falling Edge)
.C:fbee  D0 19       BNE $FC09      ;Branch on not 0 (Exit IRQ)
.C:fbf0  20 A6 FB    JSR $FBA6      ;Send Bit
.C:fbf3  D0 14       BNE $FC09      ;Branch if Output Line High (Rising Edge)
.C:fbf5  A5 A4       LDA $A4        ;Get Tape Bit Pair Dipole (Falling Edge)
.C:fbf7  49 01       EOR #$01
.C:fbf9  85 A4       STA $A4        ;Set Tape Bit Pair Dipole
.C:fbfb  F0 0F       BEQ $FC0C      ;Branch if 0 (Bit Pair Sent?)
.C:fbfd  A5 BD       LDA $BD        ;Get Tape Byte Out
.C:fbff  49 01       EOR #$01       ;EOR Tape Byte Out bit 0
.C:fc01  85 BD       STA $BD        ;Store Tape Byte Out
.C:fc03  29 01       AND #$01       ;Mask bit 0
```

```
.C:fc05  45 9B       EOR $9B        ;EOR Tape Byte Parity
.C:fc07  85 9B       STA $9B        ;Store Tape Byte Parity
.C:fc09  4C BC FE    JMP $FEBC      ;Exit IRQ

.C:fc0c  46 BD       LSR $BD        ;Shift Tape Byte Out into Carry
.C:fc0e  C6 A3       DEC $A3        ;Decrement Tape Byte Bits Count
.C:fc10  A5 A3       LDA $A3        ;Get Tape Bytes Bits Count
.C:fc12  F0 3A       BEQ $FC4E

.C:fc14  10 F3       BPL $FC09      ;If Positive Exit IRQ

.C:fc16  20 97 FB    JSR $FB97      ;Tape Init ZP (Set Tape Bits Count to 8, Init Tape Bit Pair to 0, Set
Tape Error to 0, Set Tape
                                    ;Parity to 0, Set Tape Bit Pair Count to 0)
.C:fc19  58          CLI            ;Enable Interrupts
.C:fc1a  A5 A5       LDA $A5        ;Get Tape Sync Countdown
.C:fc1c  F0 12       BEQ $FC30      ;Branch if Countdown is Completed, Write File
.C:fc1e  A2 00       LDX #$00
.C:fc20  86 D7       STX $D7        ;Set Tape File Checksum to 0
.C:fc22  C6 A5       DEC $A5        ;Decrement Tape Sync Countdown
.C:fc24  A6 BE       LDX $BE        ;Get Tape Block Copy Count
.C:fc26  E0 02       CPX #$02       ;Check if First Block Copy
.C:fc28  D0 02       BNE $FC2C      ;Branch if First Block
.C:fc2a  09 80       ORA #$80       ;Flip Bit 7 on for Second Block
.C:fc2c  85 BD       STA $BD        ;Set Tape Byte Out
.C:fc2e  D0 D9       BNE $FC09      ;Branch Always

                                    ;Write File
.C:fc30  20 D1 FC    JSR $FCD1      ;Check Tape EOF
.C:fc33  90 0A       BCC $FC3F      ;Branch if not EOF
.C:fc35  D0 91       BNE $FBC8      ;Branch if not Rollover
.C:fc37  E6 AD       INC $AD        ;Increment Tape Load/Save Address High
.C:fc39  A5 D7       LDA $D7        ;Get Tape File Checksum
.C:fc3b  85 BD       STA $BD        ;Set Tape Byte Out
.C:fc3d  B0 CA       BCS $FC09      ;Branch if EOF (Exit IRQ)
.C:fc3f  A0 00       LDY #$00       ;Set Y Index to 0
.C:fc41  B1 AC       LDA ($AC),Y    ;Get Tape File Byte from Memory
.C:fc43  85 BD       STA $BD        ;Set Tape Byte Out
.C:fc45  45 D7       EOR $D7        ;Calculate Tape File Checksum
.C:fc47  85 D7       STA $D7        ;Store Tape File Checksum
.C:fc49  20 DB FC    JSR $FCDB      ;Set IRQ Vector
.C:fc4c  D0 BB       BNE $FC09      ;Branch Always (Exit IRQ)

.C:fc4e  A5 9B       LDA $9B        ;Get Tape Byte Parity
.C:fc50  49 01       EOR #$01       ;Invert Bit 0
.C:fc52  85 BD       STA $BD        ;Set Tape Byte Out
.C:fc54  4C BC FE    JMP $FEBC      ;Exit IRQ

.C:fc57  C6 BE       DEC $BE        ;Decrement Tape Block Copy Count
.C:fc59  D0 03       BNE $FC5E      ;Branch if not EOF
.C:fc5b  20 CA FC    JSR $FCCA      ;Turn Tape Motor Off

.C:fc5e  A9 50       LDA #$50
.C:fc60  85 A7       STA $A7        ;Set Tape Read Pass Count
.C:fc62  A2 08       LDX #$08
.C:fc64  78          SEI
.C:fc65  20 BD FC    JSR $FCBD      ;Set Tape IRQ 1
.C:fc68  D0 EA       BNE $FC54      ;Exit IRQ

TAPRWRITEIRQ1
                                    ;Send Leader/Pilot
.C:fc6a  A9 78       LDA #$78       ;Very Short Pulse Length + $14 cycles for setup time + 36 cycles IRQ
overhead = $B0 * 2 for
                                    ;negative edge on FLAG = $160 (352 cycles/uS)
.C:fc6c  20 AF FB    JSR $FBAF      ;Send Pulse
.C:fc6f  D0 E3       BNE $FC54      ;Branch if Tape IO state not 0 (used as dipole) Exit IRQ
.C:fc71  C6 A7       DEC $A7        ;Decrement Tape Read Pass Count (leader/pilot)
.C:fc73  D0 DF       BNE $FC54      ;Branch if not 0, Exit IRQ
.C:fc75  20 97 FB    JSR $FB97      ;Tape Init ZP (Set Tape Bits Count to 8, Init Tape Bit Pair to 0, Set
Tape Error to 0, Set Tape
                                    ;Parity to 0, Set Tape Bit Pair Count to 0)
.C:fc78  C6 AB       DEC $AB        ;Decrement Tape Short Count MSB (leader/pilot)
.C:fc7a  10 D8       BPL $FC54      ;Branch if Positive, Exit IRQ

.C:fc7c  A2 0A       LDX #$0A
.C:fc7e  20 BD FC    JSR $FCBD      ;Set IRQ 2
.C:fc81  58          CLI            ;Clear Interrupt Disable
.C:fc82  E6 AB       INC $AB        ;Increment Tape Short Count MSB (leader/pilot)
```

```
.C:fc84  A5 BE       LDA $BE        ;Get Tape Block Copy Count
.C:fc86  F0 30       BEQ $FCB8      ;Branch if Tape Block Copy Count is 0
.C:fc88  20 8E FB    JSR $FB8E      ;Set Tape Buffer
.C:fc8b  A2 09       LDX #$09       ;Tape Sync Countdown Count
.C:fc8d  86 A5       STX $A5        ;Set Tape Sync Countdown
.C:fc8f  86 B6       STX $B6        ;Set Tape Bits Count
.C:fc91  D0 83       BNE $FC16      ;Branch Always

.C:fc93  08          PHP            ;Push Flags
.C:fc94  78          SEI            ;Disable IRQs
.C:fc95  AD 11 D0    LDA $D011      ;Turn Screen On
.C:fc98  09 10       ORA #$10
.C:fc9a  8D 11 D0    STA $D011
.C:fc9d  20 CA FC    JSR $FCCA      ;Turn Tape Motor Off
.C:fca0  A9 7F       LDA #$7F
.C:fca2  8D 0D DC    STA $DC0D      ;Disable IRQ
.C:fca5  20 DD FD    JSR $FDDD      ;Set CIA Irq Timing
.C:fca8  AD A0 02    LDA $02A0      ;Get IRQ Vector High Temporary
.C:fcab  F0 09       BEQ $FCB6      ;Branch if 0
.C:fcad  8D 15 03    STA $0315      ;Set IRQ Vector High
.C:fcb0  AD 9F 02    LDA $029F      ;Get CIA Vector Low Temporary
.C:fcb3  8D 14 03    STA $0314      ;Set IRQ Vector Low
.C:fcb6  28          PLP            ;Restore Flags
.C:fcb7  60          RTS            ;Return

.C:fcb8  20 93 FC    JSR $FC93      ;Disable Tape IRQ and Restore Normal IRQ
.C:fcbb  F0 97       BEQ $FC54      ;Exit IRQ

SETIRQVEC                           ;Set IRQ Vector

.C:fcbd  BD 93 FD    LDA $FD93,X
.C:fcc0  8D 14 03    STA $0314
.C:fcc3  BD 94 FD    LDA $FD94,X
.C:fcc6  8D 15 03    STA $0315
.C:fcc9  60          RTS

.C:fcca  A5 01       LDA $01        ;Turn Tape Motor Off
.C:fccc  09 20       ORA #$20
.C:fcce  85 01       STA $01
.C:fcd0  60          RTS

TAPCHKEOF                           ;Check Tape End of File

.C:fcd1  38          SEC
.C:fcd2  A5 AC       LDA $AC
.C:fcd4  E5 AE       SBC $AE
.C:fcd6  A5 AD       LDA $AD
.C:fcd8  E5 AF       SBC $AF
.C:fcda  60          RTS

TAPINCLOAD                          ;Increment Tape Load/Save Address

.C:fcdb  E6 AC       INC $AC
.C:fcdd  D0 02       BNE $FCE1
.C:fcdf  E6 AD       INC $AD
.C:fce1  60          RTS

>C:fd9b  6a fc                      ;$fc6a IRQ Tape Write 1
>C:fd9d  cd fb                      ;$fbcd IRQ Tape Write 2
>C:fd9f  31 ea                      ;$ea31 IRQ Normal IRQ
>C:fda1  2c f9                      ;$f92c IRQ Tape Read

.C:fddd  AD A6 02    LDA $02A6      ;Get PAL/NTSC Flag
.C:fde0  F0 0A       BEQ $FDEC      ;Branch NTSC
.C:fde2  A9 25       LDA #$25       ;Get PAL Timing Low
.C:fde4  8D 04 DC    STA $DC04      ;Set PAL Timing Low
.C:fde7  A9 40       LDA #$40       ;Get PAL Timing High
.C:fde9  4C F3 FD    JMP $FDF3      ;Jump to Set Timing High
.C:fdec  A9 95       LDA #$95       ;Get NTSC Timing Low
.C:fdee  8D 04 DC    STA $DC04      ;Set NTSC Timing Low
.C:fdf1  A9 42       LDA #$42       ;Get NTSC TIming High
.C:fdf3  8D 05 DC    STA $DC05      ;Set Timing High
.C:fdf6  4C 6E FF    JMP $FF6E      ;Enable Standard IRQ

SETST                               ;Set Status

.C:fe1c  05 90       ORA $90
.C:fe1e  85 90       STA $90
```

```
.C:fe20  60          RTS

TAPFAKIRQ                           ;Tape Fake an IRQ

.C:ff43  08          PHP
.C:ff44  68          PLA
.C:ff45  29 EF       AND #$EF
.C:ff47  48          PHA

HWIRQ
                                    ;7 for HW IRQ
.C:ff48  48          PHA            ;3
.C:ff49  8A          TXA            ;2
.C:ff4a  48          PHA            ;3
.C:ff4b  98          TYA            ;2
.C:ff4c  48          PHA            ;3
.C:ff4d  BA          TSX            ;2
.C:ff4e  BD 04 01    LDA $0104,X    ;Get BRK Flag ;4
.C:ff51  29 10       AND #$10       ;2
.C:ff53  F0 03       BEQ $FF58      ;3
.C:ff55  6C 16 03    JMP ($0316)    ;JMP through BRK vector
.C:ff58  6C 14 03    JMP ($0314)    ;JMP through IRQ vector ;5
                                    ;IRQ overhead = 36 cycles

.C:ff6e  A9 81       LDA #$81       ;%10000001 Enable Timer A IRQ
.C:ff70  8D 0D DC    STA $DC0D      ;Set ICR
.C:ff73  AD 0E DC    LDA $DC0E      ;Get CIA Control Register A
.C:ff76  29 80       AND #$80       ;Mask TOD 50/60hz
.C:ff78  09 11       ORA #$11       ;%00010001 Start Timer A, Force Load Strobe
.C:ff7a  8D 0E DC    STA $DC0E      ;Set CIA Control Register A
.C:ff7d  4C 8E EE    JMP $EE8E
```