```
;=================================================================
====================================
;
SCROLLING ROUTINES
;
;
Peter 'Sig' Hewett aka Retroromicon
;=================================================================
====================================
;  Routines for scrolling the screen. The way we'll be doing
scrolling is by letting the
; player 'roam' within a boundry then 'pushing' the screen when
they hit a boundry.
; This should make things a little easier as there will never be
a point when scrolling is
; instantly reversed.
; Also given that the only way to exceed the top threshold
should be by ladder/elevator, and
; the bottom by an excessive fall (death), we can give the feel
of a 4/8 way scroller while
; only implementing 4 seperate one way scrollers - which will be
a LOT simpler.
;
; Ideally I'd like to implement a 'resetting' system at least on
the Y axis where once the
; character stops moving, scrolling continues till he is back in
the 'home position' where
; a start is possible with a 3 frame lead up. This shouldn't be
that noticable from a player
; standpoint
;
; NOTE : Like Sprite movement, these routines perform a ONE
pixel scroll, to scroll more than one
;        pixel, they must be called multiple times.
;
;       Direction scrolled is referring to the direction the
character is moving, not the actual
;       direction of the screen, so things might seem a little
'bass-ackwards'
;
; WARNING - this is a LONG file full of special case routines
and unwound loops
;-----------------------------------------------------------------
-----------------------------------
; Scroll Method:
; Each pixel scroll represents a 'frame', a countdown to 'the
jump frame' where the entire screen
```

```
; must be shifted one character across. in the frames leading up
to that we perform whatever tasks
; we can in the background. We do map lookups, prepare buffers
for 'new characters', copy the
; backbuffer one character across - then on the 'jump frame' we
copy color ram - in one pass,
; and swap the screens, and finally reset for the next round.
;
; During testing I'm pretty sure there are 4 viable speeds for
scrolling that can be accomplished
; (at my level of coding anyways) - 1/2 pixel , 1 pixel, 2
pixels and 3 pixels.
; that should make for decent variable speed scrolling and
player acceleration at a later date
;
; Notes :
;               Right scrolling MUST start on frame 4 for speed
1
;-------------------------------------------------------------
-----------------------------------
;
SCROLL_STOP     = 0
SCROLL_RIGHT    = 1
SCROLL_LEFT     = 2
SCROLL_UP       = 3
SCROLL_DOWN     = 4

;-------------------------------------------------------------
-----------------------------------
;
UPDATE SCROLLING
;-------------------------------------------------------------
-----------------------------------
; Entry point for all scrolling. Here we work out what direction
(if any) we are scrolling in
; and branch off to the directional scroll routines - there we
divide the work up along the
; frames and perform the actual scrolling.
;-------------------------------------------------------------
-----------------------------------
#region "UpdateScroll"
UpdateScroll
        lda SCROLL_DIRECTION            ; Check for direction
of scroll
        bne @start
```

```
                                        ; We have stopped - but
we need to know we are on
                                        ; the correct 'start
frames' so if we scroll again
                                        ; (especially in the
opposite direction) then everything
                                        ; is setup properly -
otherwise we could miss copying
                                        ; the buffers, updating
the back screen, or glitching the
                                        ; the color shift. The
extra shifts will be barely noticable

        lda SCROLL_MOVING               ; If we are stoppped
(SCROLL_STOP = 0) then we can return
        bne @fix_frame
        rts

@fix_frame
@fix_right
        cmp #SCROLL_RIGHT               ; Check if we're
scrolling right
        bne @fix_left
                                        ; fix right frame
        lda SCROLL_COUNT_X              ; get the scroll counter
        cmp #4                          ; if we're at frame 4 we
stop scrolling
        beq @fix_done

        ;-------------------------------------------------------
------------------
        ;                               ; Do our extra pixel
scroll adjustment
        ldx #0                          ; Move player sprite
left one pixel
        jsr MoveSpriteLeft              ; Eventually we'll
scroll all the active sprites
        ldx #1
        jsr MoveSpriteLeft

        lda SCROLL_MOVING
        jmp @start
        rts
        ;------------------------------------------------ FIX
SCROLL LEFT
@fix_left
        cmp #SCROLL_LEFT
```

```
        bne @fix_up
                                        ; Abort if needed
        lda SCROLL_FIX_SKIP
        bne @fix_done

@check_left
        lda SCROLL_COUNT_X
        cmp #4
        beq @fix_done

        ;-------------------------------------------------------
-----------------
        ;                               Do our extra pixel
scroll adjustment
        ldx #0
        jsr MoveSpriteRight
        ldx #1
        jsr MoveSpriteRight

        lda SCROLL_MOVING
        jmp @start

        rts

@fix_up
        cmp #SCROLL_UP
        bne @fix_down

        lda SCROLL_COUNT_Y
        cmp #3
        beq @fix_done

        ;-------------------------------------------------------
----------------
        ;                               Do extra pixel scroll
adjustment for up
        ldx #0
        jsr MoveSpriteDown
        ldx #1
        jsr MoveSpriteDown

        lda SCROLL_MOVING
        jmp @start


@fix_down
        cmp #SCROLL_DOWN
```

```
        bne @fix_done

        lda SCROLL_COUNT_Y
        cmp #3
        beq @fix_done

        ldx #0
        jsr MoveSpriteUp
        ldx #1
        jsr MoveSpriteUp

        lda SCROLL_MOVING
        jmp @start


@fix_done
        lda #SCROLL_STOP
        sta SCROLL_MOVING

@start
        ;-------------------------------- Simple testing at this
point
;        lda TIMER                       ; solid timed scroll
;        and #%1                        ; slow timer to test the
screen

 ;       bne @end

;        lda SCROLL_DIRECTION            ; Fetch scroll
direction

        ;-------------------------------- A = SCROLL_DIRECTION

        cmp #SCROLL_RIGHT               ; SCROLL RIGHT
        bne @left
        jmp ScrollRight
@left
        cmp #SCROLL_LEFT                ; SCROLL LEFT
        bne @down
        jmp ScrollLeft

@down   cmp #SCROLL_DOWN                ; SCROLL DOWN
        bne @up
        jmp ScrollDown

@up
        cmp #SCROLL_UP
```

```
        bne @end
        jmp ScrollUp

@end

        rts
#endregion
;-------------------------------------------------------------------
-----------------------------------
;
SCROLL UP
;-------------------------------------------------------------------
-----------------------------------
; Scroll one pixel up and take care of the screen/color wrap if
needed
;-------------------------------------------------------------------
-----------------------------------
#region "ScrollUp"
ScrollUp
        inc SCROLL_COUNT_Y
        lda SCROLL_COUNT_Y
        and #%00000111
        sta SCROLL_COUNT_Y
        ;---------------------------------------- FRAME 4
@frame4
        cmp #4
        bne @frame5
        rts
        ;---------------------------------------- FRAME 5
@frame5
        cmp #5
        bne @frame6
        jsr CopyVerticalBuffer
        rts
        ;---------------------------------------- FRAME 6
@frame6
        cmp #6
        bne @frame7
        jsr ShiftCharsUp
        jsr DrawUpBuffer
        rts
        ;---------------------------------------- FRAME 7 -
PREJUMP
@frame7
        cmp #7
        bne @frame0
        rts
```

```
        ;-------------------------------------- FRAME 0 - JUMP
FRAME
@frame0
        cmp #0
        bne @done

        jsr SwapScreens
        jsr ColorShiftUp
        jsr DrawUpColor

        lda MAP_Y_DELTA
        sec
        sbc #1
        and #%0011
        sta MAP_Y_DELTA

        cmp #3
        beq @newtile
        rts

@newtile

        dec MAP_Y_POS
@done
        rts

#endregion
;-------------------------------------------------------------------
-----------------------------------
;
SCROLL DOWN
;-------------------------------------------------------------------
-----------------------------------
; Scroll one pixel down and take care of screen/color wrap if
needed
;-------------------------------------------------------------------
-----------------------------------
#region "ScrollDown"
ScrollDown
        dec SCROLL_COUNT_Y              ; increemnt the scroll y
value
        lda SCROLL_COUNT_Y
        and #%00000111                  ; Mask it to a 0-7 count
        sta SCROLL_COUNT_Y              ; store it for the next
raster IRQ to update
        ;-------------------------------------------- FRAME 4
@frame4
```

```
        ;---------------------------------------------- FRAME 3
@frame3
        ;---------------------------------------------- FRAME 2
@frame2
        cmp #2
        bne @frame1
        jsr CopyVerticalBuffer
        rts
        ;---------------------------------------------- FRAME 1
@frame1
        cmp #1
        bne @frame0
        jsr ShiftCharsDown
        jsr DrawDownBuffer

        rts
        ;---------------------------------------------- FRAME 0
@frame0
        cmp #0
        bne @frame7

        rts
        ;---------------------------------------------- FRAME 7
- JUMP FRAME
@frame7
        cmp #7
        bne @done
        jsr SwapScreens                 ; Swap Back / Front
screens
        jsr ColorShiftDown              ; Shift color down one
character
        jsr DrawDownColor               ; Draw new colors in
bottom line

        lda MAP_Y_DELTA                 ; increment the
MAP_Y_DELTA
        clc
        adc #1
        and #%0011                      ; Mask to a value
between 0-3
        sta MAP_Y_DELTA

        cmp #0                          ; check for crossover to
a new tile
        beq @newtile
        rts
@newtile
```

```
        inc MAP_Y_POS                   ; increment MAP Y POS on
a new tile

        ; TODO - increment MAP_ADDRESS to next map line

@done
        rts
#endRegion
;-------------------------------------------------------------------
-----------------------------------
;
SCROLL LEFT
;-------------------------------------------------------------------
-----------------------------------
; Scroll one pixel left and take care of screen/color wrap if
needed
;-------------------------------------------------------------------
-----------------------------------
#region "ScrollLeft"
ScrollLeft
        inc SCROLL_COUNT_X              ; increment the scroll x
value
        lda SCROLL_COUNT_X
        and #%00000111                  ; Mask it to a 0-7 count
        sta SCROLL_COUNT_X              ; store it for the
raster IRQ to update
        ;------------------------------------------------ FRAME 4
@frame4
        ;------------------------------------------------ FRAME 5
@frame5
        cmp #5
        bne @frame6
;        jsr CopyHorizontalBuffer        ; copy new characters
from the map to the buffers
        jsr CopyLeftBuffer
        rts
        ;------------------------------------------------ FRAME 6
@frame6
        cmp #6
        bne @frame7
        jsr ShiftCharsLeft              ; shift the backscreen
over by one character
        jsr DrawLeftBuffer              ; draw the new
characters to the screen
        rts
        ;------------------------------------------------ FRAME 7
@frame7
```

```
        cmp #7
        bne @frame0


        rts
        ;---------------------------------------------- FRAME 0
- JUMP FRAME
@frame0
        cmp #0
        bne @done
        jsr SwapScreens                         ; Swap the
buffer screen and displayed screen
        jsr ColorShiftLeft                      ; Shift color
ram one character to the left
        jsr DrawLeftColor                       ; Draw in the
leftmost column in color ram

        sec
        lda MAP_X_DELTA                         ; decrement map
X delta by 3 (character within tile)
        sbc #1
        and #%00000011                          ; mask it's
value to 0-7
        sta MAP_X_DELTA                         ; save the new
delta
        cmp #3                                  ; if the delta
is 3, we hit a new tile
        beq @new_tile
        rts
@new_tile
        dec MAP_X_POS                           ; so decrement
the map X position
        sec
        lda MAP_ADDRESS                         ; subtract 1
from Map address
        sbc #1                                  ; (address of
the top - right corner of the map)
        sta MAP_ADDRESS
        lda MAP_ADDRESS + 1
        sbc #0
        sta MAP_ADDRESS + 1


@done
        rts
#endregion
```

```
;-------------------------------------------------------------
-----------------------------------
;
SCROLL RIGHT
;-------------------------------------------------------------
-----------------------------------
; Scroll one pixel right and take care of screen/color wrap if
needed
; Tasks are broken down and performed on each 'frame'
;-------------------------------------------------------------
-----------------------------------
#region "ScrollRight"
ScrollRight
        dec SCROLL_COUNT_X      ; decrement to Scroll Right
        lda SCROLL_COUNT_X      ; load into A
        and #%00000111          ; Mask lower 3 bits to make a 0
- 7 count
        sta SCROLL_COUNT_X      ; store the new count - Raster
IRQ does the ScrollRight

                                ; A holds the count, from here
we can test against it
                                ; to split the workload
depending on what 'frame' we are at

        ;-----------------------------------------------------
FRAME 2
@frame2
        cmp #2
        bne @frame1
        jsr CopyHorizontalBuffer     ; fetch the next column to
draw
        rts
        ;-----------------------------------------------------
FRAME 1
@frame1
        cmp #1
        bne @frame0
        jsr ShiftCharsRight     ; shift characters to the buffer
screen
        jsr DrawRightBuffer     ; draw in the new characters
from the buffer
        rts
        ;-----------------------------------------------------
FRAME 0 - PREJUMP
@frame0
        cmp #0
```

```
        bne @frame7
        rts
        ;-------------------------------------------------
FRAME 7 - JUMP FRAME
@frame7
        cmp #7
        bne @done
        jsr SwapScreens         ; bring the buffer to the
foreground
        jsr ColorShiftRight     ; shift the color ram one
character
        jsr DrawRightColor      ; draw in the new colors from
the buffer

        clc
        lda MAP_X_DELTA         ; Add one to delta
        adc #1
        and #%00000011          ; if delta wraps we go up a tile
        sta MAP_X_DELTA
        cmp #0
        beq @newTile
        rts
@newTile
        inc MAP_X_POS           ; inc map pos to next tile
        clc
        lda MAP_ADDRESS         ; inc map address to next
position
        adc #1
        sta MAP_ADDRESS
        lda MAP_ADDRESS + 1
        adc #0
        sta MAP_ADDRESS + 1

        rts
        ;-------------------------------------------------
------------------
@done
        rts
#endRegion
;-----------------------------------------------------------
-----------------------------------
;
SHIFT CHARS UP
;-----------------------------------------------------------
-----------------------------------
; Shift the characters for a scroll going up (Characters are
moving down)
```

```
;-----------------------------------------------------------------
-----------------------------------
#region "ShiftCharsUp"
ShiftCharsUp
        lda CURRENT_SCREEN + 1
        cmp #>SCREEN2_MEM
        beq @screen2
        jmp @copyfm1
@screen2
        jmp @copyfm2
        rts


@copyfm1
        ldx #0
@copyloop1

        lda SCREEN1_MEM,x
        sta SCREEN2_MEM + 40,x

        lda SCREEN1_MEM + 40,x
        sta SCREEN2_MEM + 80,x

        lda SCREEN1_MEM + 80,x
        sta SCREEN2_MEM + 120,x

        lda SCREEN1_MEM + 120,x
        sta SCREEN2_MEM + 160,x


        lda SCREEN1_MEM + 160,x
        sta SCREEN2_MEM + 200,x

        lda SCREEN1_MEM + 200,x
        sta SCREEN2_MEM + 240,x

        lda SCREEN1_MEM + 240,x
        sta SCREEN2_MEM + 280,x

        lda SCREEN1_MEM + 280,x
        sta SCREEN2_MEM + 320,x


        lda SCREEN1_MEM + 320,x
        sta SCREEN2_MEM + 360,x

        lda SCREEN1_MEM + 360,x
```

```
        sta SCREEN2_MEM + 400,x

        lda SCREEN1_MEM + 400,x
        sta SCREEN2_MEM + 440,x

        lda SCREEN1_MEM + 440,x
        sta SCREEN2_MEM + 480,x


        lda SCREEN1_MEM + 480,x
        sta SCREEN2_MEM + 520,x

        lda SCREEN1_MEM + 520,x
        sta SCREEN2_MEM + 560,x

        lda SCREEN1_MEM + 560,x
        sta SCREEN2_MEM + 600,x

        lda SCREEN1_MEM + 600,x
        sta SCREEN2_MEM + 640,x


        lda SCREEN1_MEM + 640,x
        sta SCREEN2_MEM + 680,x

        lda SCREEN1_MEM + 680,x
        sta SCREEN2_MEM + 720,x

        lda SCREEN1_MEM + 720,x
        sta SCREEN2_MEM + 760,x

        lda SCREEN1_MEM + 760,x
        sta SCREEN2_MEM + 800,x

        inx
        cpx #40
        bne @copyloop1
        rts

@copyfm2
        ldx #0
@copyloop2

        lda SCREEN2_MEM,x
        sta SCREEN1_MEM + 40,x

        lda SCREEN2_MEM + 40,x
```

```
sta SCREEN1_MEM + 80,x

lda SCREEN2_MEM + 80,x
sta SCREEN1_MEM + 120,x

lda SCREEN2_MEM + 120,x
sta SCREEN1_MEM + 160,x


lda SCREEN2_MEM + 160,x
sta SCREEN1_MEM + 200,x

lda SCREEN2_MEM + 200,x
sta SCREEN1_MEM + 240,x

lda SCREEN2_MEM + 240,x
sta SCREEN1_MEM + 280,x

lda SCREEN2_MEM + 280,x
sta SCREEN1_MEM + 320,x


lda SCREEN2_MEM + 320,x
sta SCREEN1_MEM + 360,x

lda SCREEN2_MEM + 360,x
sta SCREEN1_MEM + 400,x

lda SCREEN2_MEM + 400,x
sta SCREEN1_MEM + 440,x

lda SCREEN2_MEM + 440,x
sta SCREEN1_MEM + 480,x


lda SCREEN2_MEM + 480,x
sta SCREEN1_MEM + 520,x

lda SCREEN2_MEM + 520,x
sta SCREEN1_MEM + 560,x

lda SCREEN2_MEM + 560,x
sta SCREEN1_MEM + 600,x

lda SCREEN2_MEM + 600,x
sta SCREEN1_MEM + 640,x
```

```
        lda SCREEN2_MEM + 640,x
        sta SCREEN1_MEM + 680,x

        lda SCREEN2_MEM + 680,x
        sta SCREEN1_MEM + 720,x

        lda SCREEN2_MEM + 720,x
        sta SCREEN1_MEM + 760,x

        lda SCREEN2_MEM + 760,x
        sta SCREEN1_MEM + 800,x

        inx
        cpx #40
        bne @copyloop2
        rts

#endregion
;-------------------------------------------------------------------
-----------------------------------
;
SHIFT CHARS DOWN
;-------------------------------------------------------------------
-----------------------------------
; Shift the characters for a scroll going down (characters are
moving up)
;-------------------------------------------------------------------
-----------------------------------
#region "ShiftCharsDown"
ShiftCharsDown
        lda CURRENT_SCREEN + 1
        cmp #>SCREEN2_MEM
        beq @screen2
        jmp @copyfm1
@screen2
        jmp @copyfm2
        rts

@copyFm1
        ldx #0
@copyloop1
        lda SCREEN1_MEM + 40,x          ; Tile 1
        sta SCREEN2_MEM,x

        lda SCREEN1_MEM + 80,x
        sta SCREEN2_MEM + 40,x
```

```
lda SCREEN1_MEM + 120,x
sta SCREEN2_MEM + 80,x

lda SCREEN1_MEM + 160,x
sta SCREEN2_MEM + 120,x


lda SCREEN1_MEM + 200,x          ; Tile 2
sta SCREEN2_MEM + 160,x

lda SCREEN1_MEM + 240,x
sta SCREEN2_MEM + 200,x

lda SCREEN1_MEM + 280,x
sta SCREEN2_MEM + 240,x

lda SCREEN1_MEM + 320,x
sta SCREEN2_MEM + 280,x


lda SCREEN1_MEM + 360,x          ; Tile 3
sta SCREEN2_MEM + 320,x

lda SCREEN1_MEM + 400,x
sta SCREEN2_MEM + 360,x

lda SCREEN1_MEM + 440,x
sta SCREEN2_MEM + 400,x

lda SCREEN1_MEM + 480,x
sta SCREEN2_MEM + 440,x


lda SCREEN1_MEM + 520,x          ; Tile 4
sta SCREEN2_MEM + 480,x

lda SCREEN1_MEM + 560,x
sta SCREEN2_MEM + 520,x

lda SCREEN1_MEM + 600,x
sta SCREEN2_MEM + 560,x

lda SCREEN1_MEM + 640,x
sta SCREEN2_MEM + 600,x
```

```
        lda SCREEN1_MEM + 680,x
        sta SCREEN2_MEM + 640,x

        lda SCREEN1_MEM + 720,x
        sta SCREEN2_MEM + 680,x

        lda SCREEN1_MEM + 760,x
        sta SCREEN2_MEM + 720,x

        lda SCREEN1_MEM + 800,x
        sta SCREEN2_MEM + 760,x

        inx
        cpx #40
        bne @copyloop1

        rts


@copyFm2
        ldx #0
@copyloop2
        lda SCREEN2_MEM + 40,x          ; Tile 1
        sta SCREEN1_MEM,x

        lda SCREEN2_MEM + 80,x
        sta SCREEN1_MEM + 40,x

        lda SCREEN2_MEM + 120,x
        sta SCREEN1_MEM + 80,x

        lda SCREEN2_MEM + 160,x
        sta SCREEN1_MEM + 120,x


        lda SCREEN2_MEM + 200,x       ; Tile 2
        sta SCREEN1_MEM + 160,x

        lda SCREEN2_MEM + 240,x
        sta SCREEN1_MEM + 200,x

        lda SCREEN2_MEM + 280,x
        sta SCREEN1_MEM + 240,x

        lda SCREEN2_MEM + 320,x
        sta SCREEN1_MEM + 280,x
```

```
        lda SCREEN2_MEM + 360,x         ; Tile 3
        sta SCREEN1_MEM + 320,x

        lda SCREEN2_MEM + 400,x
        sta SCREEN1_MEM + 360,x

        lda SCREEN2_MEM + 440,x
        sta SCREEN1_MEM + 400,x

        lda SCREEN2_MEM + 480,x
        sta SCREEN1_MEM + 440,x


        lda SCREEN2_MEM + 520,x         ; Tile 4
        sta SCREEN1_MEM + 480,x

        lda SCREEN2_MEM + 560,x
        sta SCREEN1_MEM + 520,x

        lda SCREEN2_MEM + 600,x
        sta SCREEN1_MEM + 560,x

        lda SCREEN2_MEM + 640,x
        sta SCREEN1_MEM + 600,x


        lda SCREEN2_MEM + 680,x         ; Tile 5
        sta SCREEN1_MEM + 640,x

        lda SCREEN2_MEM + 720,x
        sta SCREEN1_MEM + 680,x

        lda SCREEN2_MEM + 760,x
        sta SCREEN1_MEM + 720,x

        lda SCREEN2_MEM + 800,x
        sta SCREEN1_MEM + 760,x

        inx
        cpx #40
        bne @copyloop2

        rts
#endregion
```

```
;-------------------------------------------------------------------
-----------------------------------
;
SHIFT CHARS LEFT
;-------------------------------------------------------------------
-----------------------------------
; Shift the characters for a scroll in the left direction
(characters are moving right)
; routines are unrolled for speed
;-------------------------------------------------------------------
-----------------------------------
#region "ShiftCharsLeft"
ShiftCharsLeft
        lda CURRENT_SCREEN + 1
        cmp #>SCREEN2_MEM
        beq @screen2
        jmp @copyFm1
        rts
@screen2
        jmp @copyFm2
        rts
@copyFm1
        ldx #0
@copyloop1
        lda SCREEN1_MEM,x
        sta SCREEN2_MEM + 1,x

        lda SCREEN1_MEM + 40,x
        sta SCREEN2_MEM + 41,x

        lda SCREEN1_MEM + 80,x
        sta SCREEN2_MEM + 81,x

        lda SCREEN1_MEM + 120,x
        sta SCREEN2_MEM + 121,x

        lda SCREEN1_MEM + 160,x
        sta SCREEN2_MEM + 161,x

        lda SCREEN1_MEM + 200,x
        sta SCREEN2_MEM + 201,x

        lda SCREEN1_MEM + 240,x
        sta SCREEN2_MEM + 241,x

        lda SCREEN1_MEM + 280,x
        sta SCREEN2_MEM + 281,x
```

```
        lda SCREEN1_MEM + 320,x
        sta SCREEN2_MEM + 321,x

        lda SCREEN1_MEM + 360,x
        sta SCREEN2_MEM + 361,x

        lda SCREEN1_MEM + 400,x
        sta SCREEN2_MEM + 401,x

        lda SCREEN1_MEM + 440,x
        sta SCREEN2_MEM + 441,x

        lda SCREEN1_MEM + 480,x
        sta SCREEN2_MEM + 481,x

        lda SCREEN1_MEM + 520,x
        sta SCREEN2_MEM + 521,x

        lda SCREEN1_MEM + 560,x
        sta SCREEN2_MEM + 561,x

        lda SCREEN1_MEM + 600,x
        sta SCREEN2_MEM + 601,x

        lda SCREEN1_MEM + 640,x
        sta SCREEN2_MEM + 641,x

        lda SCREEN1_MEM + 680,x
        sta SCREEN2_MEM + 681,x

        lda SCREEN1_MEM + 720,x
        sta SCREEN2_MEM + 721,x

        inx
        cpx #39
        bne @copyloop1
        rts

@copyFm2
        ldx #0
@copyloop2
        lda SCREEN2_MEM,x
        sta SCREEN1_MEM + 1,x

        lda SCREEN2_MEM + 40,x
        sta SCREEN1_MEM + 41,x
```

```
lda SCREEN2_MEM + 80,x
sta SCREEN1_MEM + 81,x

lda SCREEN2_MEM + 120,x
sta SCREEN1_MEM + 121,x

lda SCREEN2_MEM + 160,x
sta SCREEN1_MEM + 161,x

lda SCREEN2_MEM + 200,x
sta SCREEN1_MEM + 201,x

lda SCREEN2_MEM + 240,x
sta SCREEN1_MEM + 241,x

lda SCREEN2_MEM + 280,x
sta SCREEN1_MEM + 281,x

lda SCREEN2_MEM + 320,x
sta SCREEN1_MEM + 321,x

lda SCREEN2_MEM + 360,x
sta SCREEN1_MEM + 361,x

lda SCREEN2_MEM + 400,x
sta SCREEN1_MEM + 401,x

lda SCREEN2_MEM + 440,x
sta SCREEN1_MEM + 441,x

lda SCREEN2_MEM + 480,x
sta SCREEN1_MEM + 481,x

lda SCREEN2_MEM + 520,x
sta SCREEN1_MEM + 521,x

lda SCREEN2_MEM + 560,x
sta SCREEN1_MEM + 561,x

lda SCREEN2_MEM + 600,x
sta SCREEN1_MEM + 601,x

lda SCREEN2_MEM + 640,x
sta SCREEN1_MEM + 641,x

lda SCREEN2_MEM + 680,x
```

```
        sta SCREEN1_MEM + 681,x

        lda SCREEN2_MEM + 720,x
        sta SCREEN1_MEM + 721,x

        inx
        cpx #39
        bne @copyloop2
        rts

#endregion

;-------------------------------------------------------------------
-----------------------------------
;
SHIFT COLOR LEFT
;-------------------------------------------------------------------
-----------------------------------
#region "ShiftColorLeft"
ColorShiftLeft
        ldx #38
@copyloop
        lda COLOR_MEM,x
        sta COLOR_MEM + 1,x

        lda COLOR_MEM + 40,x
        sta COLOR_MEM + 41,x

        lda COLOR_MEM + 80,x
        sta COLOR_MEM + 81,x

        lda COLOR_MEM + 120,x
        sta COLOR_MEM + 121,x

        lda COLOR_MEM + 160,x
        sta COLOR_MEM + 161,x

        lda COLOR_MEM + 200,x
        sta COLOR_MEM + 201,x

        lda COLOR_MEM + 240,x
        sta COLOR_MEM + 241,x

        lda COLOR_MEM + 280,x
        sta COLOR_MEM + 281,x

        lda COLOR_MEM + 320,x
```

```
        sta COLOR_MEM + 321,x

        lda COLOR_MEM + 360,x
        sta COLOR_MEM + 361,x

        lda COLOR_MEM + 400,x
        sta COLOR_MEM + 401,x

        lda COLOR_MEM + 440,x
        sta COLOR_MEM + 441,x

        lda COLOR_MEM + 480,x
        sta COLOR_MEM + 481,x

        lda COLOR_MEM + 520,x
        sta COLOR_MEM + 521,x

        lda COLOR_MEM + 560,x
        sta COLOR_MEM + 561,X

        lda COLOR_MEM + 600,x
        sta COLOR_MEM + 601,X

        lda COLOR_MEM + 640,x
        sta COLOR_MEM + 641,x

        lda COLOR_MEM + 680,x
        sta COLOR_MEM + 681,x

        lda COLOR_MEM + 720,x
        sta COLOR_MEM + 721,x

        dex
        bpl @copyloop
        rts

#endregion
;-------------------------------------------------------------------
-----------------------------------
;
SHIFT CHARS RIGHT
;-------------------------------------------------------------------
-----------------------------------
; Shift the characters for a scroll in the right direction (the
characters are actually moving left)
; These routines are unrolled for speed
```

```
;-----------------------------------------------------------------
-----------------------------------
#region "ShiftCharsRight"
ShiftCharsRight

        lda CURRENT_SCREEN + 1          ; Detect our front
screen / backscreen
        cmp #>SCREEN2_MEM               ; check for screen2
        beq @screen2
        jmp @copyFm1                    ; we use jmp because it
WILL be > 256 bytes
        rts
@screen2
        jmp @copyFm2
        rts

        ;-------------------------------- Copy from screen1 to
screen2
@copyFm1
        ldx #0
@copyloop1
        lda SCREEN1_MEM + 1,x
        sta SCREEN2_MEM,x

        lda SCREEN1_MEM + 41,x
        sta SCREEN2_MEM + 40,x

        lda SCREEN1_MEM + 81,x
        sta SCREEN2_MEM + 80,x

        lda SCREEN1_MEM + 121,x
        sta SCREEN2_MEM + 120,x

        lda SCREEN1_MEM + 161,x
        sta SCREEN2_MEM + 160,x

        lda SCREEN1_MEM + 201,x
        sta SCREEN2_MEM + 200,x

        lda SCREEN1_MEM + 241,x
        sta SCREEN2_MEM + 240,x

        lda SCREEN1_MEM + 281,x
        sta SCREEN2_MEM + 280,x

        lda SCREEN1_MEM + 321,x
        sta SCREEN2_MEM + 320,x
```

```
        lda SCREEN1_MEM + 361,x
        sta SCREEN2_MEM + 360,x

        lda SCREEN1_MEM + 401,x
        sta SCREEN2_MEM + 400,x

        lda SCREEN1_MEM + 441,x
        sta SCREEN2_MEM + 440,x

        lda SCREEN1_MEM + 481,x
        sta SCREEN2_MEM + 480,x

        lda SCREEN1_MEM + 521,x
        sta SCREEN2_MEM + 520,x

        lda SCREEN1_MEM + 561,x
        sta SCREEN2_MEM + 560,x

        lda SCREEN1_MEM + 601,x
        sta SCREEN2_MEM + 600,x

        lda SCREEN1_MEM + 641,x
        sta SCREEN2_MEM + 640,x

        lda SCREEN1_MEM + 681,x
        sta SCREEN2_MEM + 680,x

        lda SCREEN1_MEM + 721,x
        sta SCREEN2_MEM + 720,x

        inx
        cpx #39
        bne @copyloop1
        rts

        ;---------------------------------- Copy from screen2 to
screen1
@copyFm2
        ldx #0
@copyloop2
        lda SCREEN2_MEM + 1,x
        sta SCREEN1_MEM,x

        lda SCREEN2_MEM + 41,x
        sta SCREEN1_MEM + 40,x
```

```
lda SCREEN2_MEM + 81,x
sta SCREEN1_MEM + 80,x

lda SCREEN2_MEM + 121,x
sta SCREEN1_MEM + 120,x

lda SCREEN2_MEM + 161,x
sta SCREEN1_MEM + 160,x

lda SCREEN2_MEM + 201,x
sta SCREEN1_MEM + 200,x

lda SCREEN2_MEM + 241,x
sta SCREEN1_MEM + 240,x

lda SCREEN2_MEM + 281,x
sta SCREEN1_MEM + 280,x

lda SCREEN2_MEM + 321,x
sta SCREEN1_MEM + 320,x

lda SCREEN2_MEM + 361,x
sta SCREEN1_MEM + 360,x

lda SCREEN2_MEM + 401,x
sta SCREEN1_MEM + 400,x

lda SCREEN2_MEM + 441,x
sta SCREEN1_MEM + 440,x

lda SCREEN2_MEM + 481,x
sta SCREEN1_MEM + 480,x

lda SCREEN2_MEM + 521,x
sta SCREEN1_MEM + 520,x

lda SCREEN2_MEM + 561,x
sta SCREEN1_MEM + 560,x

lda SCREEN2_MEM + 601,x
sta SCREEN1_MEM + 600,x

lda SCREEN2_MEM + 641,x
sta SCREEN1_MEM + 640,x

lda SCREEN2_MEM + 681,x
sta SCREEN1_MEM + 680,x
```

```
        lda SCREEN2_MEM + 721,x
        sta SCREEN1_MEM + 720,x

        inx
        cpx #39
        bne @copyloop2
        rts
#endregion

;-------------------------------------------------------------------
-----------------------------------
;
COLOR SHIFT RIGHT
;-------------------------------------------------------------------
-----------------------------------
; Shift the color for right scrolling by one character
;-------------------------------------------------------------------
-----------------------------------
#region "ColorShiftRight"
ColorShiftRight
        ldx #0
@copyloop
        lda COLOR_MEM + 1,x
        sta COLOR_MEM,x

        lda COLOR_MEM + 41,x
        sta COLOR_MEM + 40,x

        lda COLOR_MEM + 81,x
        sta COLOR_MEM + 80,x

        lda COLOR_MEM + 121,x
        sta COLOR_MEM + 120,x

        lda COLOR_MEM + 161,x
        sta COLOR_MEM + 160,x

        lda COLOR_MEM + 201,x
        sta COLOR_MEM + 200,x

        lda COLOR_MEM + 241,x
        sta COLOR_MEM + 240,x

        lda COLOR_MEM + 281,x
        sta COLOR_MEM + 280,x
```

```
        lda COLOR_MEM + 321,x
        sta COLOR_MEM + 320,x

        lda COLOR_MEM + 361,x
        sta COLOR_MEM + 360,x

        lda COLOR_MEM + 401,x
        sta COLOR_MEM + 400,x

        lda COLOR_MEM + 441,x
        sta COLOR_MEM + 440,x

        lda COLOR_MEM + 481,x
        sta COLOR_MEM + 480,x

        lda COLOR_MEM + 521,x
        sta COLOR_MEM + 520,x

        lda COLOR_MEM + 561,x
        sta COLOR_MEM + 560,X

        lda COLOR_MEM + 601,x
        sta COLOR_MEM + 600,X

        lda COLOR_MEM + 641,x
        sta COLOR_MEM + 640,x

        lda COLOR_MEM + 681,x
        sta COLOR_MEM + 680,x

        lda COLOR_MEM + 721,x
        sta COLOR_MEM + 720,x

        inx
        cpx #39
        bne @copyloop
        rts

#endregion
;-------------------------------------------------------------------
-----------------------------------
;
SHIFT COLORS UP
;-------------------------------------------------------------------
-----------------------------------
#region ColorShiftUp
ColorShiftUp
```

```
        ldx #0

@copyloop
        lda COLOR_MEM + 680,x
        sta COLOR_MEM + 720,x

        lda COLOR_MEM + 640,x
        sta COLOR_MEM + 680,x

        lda COLOR_MEM + 600,x
        sta COLOR_MEM + 640,x

        lda COLOR_MEM + 560,x
        sta COLOR_MEM + 600,x


        lda COLOR_MEM + 520,x
        sta COLOR_MEM + 560,x

        lda COLOR_MEM + 480,x
        sta COLOR_MEM + 520,x

        lda COLOR_MEM + 440,x
        sta COLOR_MEM + 480,x

        lda COLOR_MEM + 400,x
        sta COLOR_MEM + 440,x


        lda COLOR_MEM + 360,x
        sta COLOR_MEM + 400,x

        lda COLOR_MEM + 320,x
        sta COLOR_MEM + 360,x

        lda COLOR_MEM + 280,x
        sta COLOR_MEM + 320,x

        lda COLOR_MEM + 240,x
        sta COLOR_MEM + 280,x

        lda COLOR_MEM + 200,x
        sta COLOR_MEM + 240,x


        lda COLOR_MEM + 160,x
        sta COLOR_MEM + 200,x
```

```
        lda COLOR_MEM + 120,x
        sta COLOR_MEM + 160,x

        lda COLOR_MEM + 80,x
        sta COLOR_MEM + 120,x

        lda COLOR_MEM + 40,x
        sta COLOR_MEM + 80,x

        lda COLOR_MEM,x
        sta COLOR_MEM + 40,x


        inx
        cpx #40
        bne @copyloop



        rts
#endregion
;-------------------------------------------------------------------
-----------------------------------
;
SHIFT COLORS DOWN
;-------------------------------------------------------------------
-----------------------------------
#region ColorShiftDown
ColorShiftDown
        ldx #0
@copyloop
        lda COLOR_MEM + 40,x          ; Tile 1
        sta COLOR_MEM,x

        lda COLOR_MEM + 80,x
        sta COLOR_MEM + 40,x

        lda COLOR_MEM + 120,x
        sta COLOR_MEM + 80,x

        lda COLOR_MEM + 160,x
        sta COLOR_MEM + 120,x


        lda COLOR_MEM + 200,x       ; Tile 2
        sta COLOR_MEM + 160,x
```

```
            lda COLOR_MEM + 240,x
            sta COLOR_MEM + 200,x

            lda COLOR_MEM + 280,x
            sta COLOR_MEM + 240,x

            lda COLOR_MEM + 320,x
            sta COLOR_MEM + 280,x


            lda COLOR_MEM + 360,x          ; Tile 3
            sta COLOR_MEM + 320,x

            lda COLOR_MEM + 400,x
            sta COLOR_MEM + 360,x

            lda COLOR_MEM + 440,x
            sta COLOR_MEM + 400,x

            lda COLOR_MEM + 480,x
            sta COLOR_MEM + 440,x


            lda COLOR_MEM + 520,x          ; Tile 4
            sta COLOR_MEM + 480,x

            lda COLOR_MEM + 560,x
            sta COLOR_MEM + 520,x

            lda COLOR_MEM + 600,x
            sta COLOR_MEM + 560,x

            lda COLOR_MEM + 640,x
            sta COLOR_MEM + 600,x


            lda COLOR_MEM + 680,x          ; Tile 5
            sta COLOR_MEM + 640,x

            lda COLOR_MEM + 720,x
            sta COLOR_MEM + 680,x

;           lda COLOR_MEM + 760,x
;           sta COLOR_MEM + 720,x

;           lda COLOR_MEM + 800,x
```

```
;         sta COLOR_MEM + 760,x

        inx
        cpx #40
        bne @copyloop

        rts
#endregion
;--------------------------------------------------------------------
-----------------------------------
;
DRAW UP BUFFER
;--------------------------------------------------------------------
-----------------------------------
; Draw the characters in the vertical buffer to the top line of
the scrolling screen
;--------------------------------------------------------------------
-----------------------------------
#region "DrawUpBuffer"
DrawUpBuffer

        loadPointer ZEROPAGE_POINTER_1, VERTICAL_BUFFER

        lda CURRENT_BUFFER + 1
        cmp #>SCREEN2_MEM
        beq @screen2

        lda #<SCREEN1_MEM
        sta ZEROPAGE_POINTER_2
        lda #>SCREEN1_MEM
        sta ZEROPAGE_POINTER_2 + 1

        ldy #0
@copyloop1
        lda (ZEROPAGE_POINTER_1),y
        sta (ZEROPAGE_POINTER_2),y

        iny
        cpy #40
        bne @copyloop1

        rts

@screen2

        lda #<SCREEN2_MEM
        sta ZEROPAGE_POINTER_2
```

```
        lda #>SCREEN2_MEM
        sta ZEROPAGE_POINTER_2 + 1

        ldy #0
@copyloop2
        lda (ZEROPAGE_POINTER_1),y
        sta (ZEROPAGE_POINTER_2),y

        iny
        cpy #40
        bne @copyloop2

        rts



        rts
#endregion
;-------------------------------------------------------------------
-----------------------------------
;
DRAW DOWN BUFFER
;-------------------------------------------------------------------
-----------------------------------
; Draw the characters in the vertical buffer to the bottom line
of the scrolling screen
;-------------------------------------------------------------------
-----------------------------------

BOTTOM_DRAW_LINE = 17

#region "DrawDownBuffer"
DrawDownBuffer

        loadPointer ZEROPAGE_POINTER_1, VERTICAL_BUFFER

        ldx #BOTTOM_DRAW_LINE           ; line to draw to


        lda CURRENT_BUFFER + 1
        cmp #>SCREEN2_MEM
        beq @screen2

        ; TO DO - UNWRAP IF REQUIRED

        lda SCREEN1_LINE_OFFSET_TABLE_LO,x
        sta ZEROPAGE_POINTER_2
```

```
        lda SCREEN1_LINE_OFFSET_TABLE_HI,x
        sta ZEROPAGE_POINTER_2 + 1

        ldy #0
@copyloop
        lda (ZEROPAGE_POINTER_1),y
        sta (ZEROPAGE_POINTER_2),y

        iny
        cpy #40
        bne @copyloop

        rts

@screen2
        lda SCREEN2_LINE_OFFSET_TABLE_LO,x
        sta ZEROPAGE_POINTER_2
        lda SCREEN2_LINE_OFFSET_TABLE_HI,x
        sta ZEROPAGE_POINTER_2 + 1

        ldy #0
@copyloop2
        lda (ZEROPAGE_POINTER_1),y
        sta (ZEROPAGE_POINTER_2),y

        iny
        cpy #40
        bne @copyloop2

        rts
#endregion
;-------------------------------------------------------------------
-----------------------------------
;
DRAW UP COLOR
;-------------------------------------------------------------------
-----------------------------------
; Write the contents of the VERTICAL_COLOR_BUFFER to the top
'off screen' line in color ram
;-------------------------------------------------------------------
-----------------------------------
#region "DrawUpColor"
DrawUpColor

        loadPointer ZEROPAGE_POINTER_1, VERTICAL_COLOR_BUFFER

        lda #<COLOR_MEM
```

```
        sta ZEROPAGE_POINTER_2
        lda #>COLOR_MEM
        sta ZEROPAGE_POINTER_2 + 1

        ldy #0
@copyloop
        lda (ZEROPAGE_POINTER_1),y
        sta (ZEROPAGE_POINTER_2),y

        iny
        cpy #40
        bne @copyloop

        rts

#endregion
;-------------------------------------------------------------------
-----------------------------------
;
DRAW DOWN COLOR
;-------------------------------------------------------------------
-----------------------------------
; Write the contents of the VERTICAL_COLOR_BUFFER to the bottom
'off screen' line in color ram
;-------------------------------------------------------------------
-----------------------------------
#region "DrawDownColor"
DrawDownColor

        ldx #BOTTOM_DRAW_LINE

        loadPointer ZEROPAGE_POINTER_1, VERTICAL_COLOR_BUFFER

        lda COLOR_LINE_OFFSET_TABLE_LO,x
        sta ZEROPAGE_POINTER_2
        lda COLOR_LINE_OFFSET_TABLE_HI,x
        sta ZEROPAGE_POINTER_2 + 1

        ldy #0

@copyloop
        lda (ZEROPAGE_POINTER_1),y
        sta (ZEROPAGE_POINTER_2),y

        iny
        cpy #40
        bne @copyloop
```

```
        rts
#endregion
;-------------------------------------------------------------------
-----------------------------------
;
DRAW LEFT BUFFER
;-------------------------------------------------------------------
-----------------------------------
; Draw the contents of the horizontal buffer to the backbuffer -
unrolled for speed
;-------------------------------------------------------------------
-----------------------------------
#region "DrawLeftBuffer"
DrawLeftBuffer
        lda CURRENT_BUFFER + 1
        cmp #>SCREEN2_MEM
        beq @screen2

        lda HORIZONTAL_BUFFER
        sta SCREEN1_MEM
        lda HORIZONTAL_BUFFER + 1
        sta SCREEN1_MEM + 40
        lda HORIZONTAL_BUFFER + 2
        sta SCREEN1_MEM + 80
        lda HORIZONTAL_BUFFER + 3
        sta SCREEN1_MEM + 120

        lda HORIZONTAL_BUFFER + 4
        sta SCREEN1_MEM + 160
        lda HORIZONTAL_BUFFER + 5
        sta SCREEN1_MEM + 200
        lda HORIZONTAL_BUFFER + 6
        sta SCREEN1_MEM + 240
        lda HORIZONTAL_BUFFER + 7
        sta SCREEN1_MEM + 280

        lda HORIZONTAL_BUFFER + 8
        sta SCREEN1_MEM + 320
        lda HORIZONTAL_BUFFER + 9
        sta SCREEN1_MEM + 360
        lda HORIZONTAL_BUFFER + 10
        sta SCREEN1_MEM + 400
        lda HORIZONTAL_BUFFER + 11
        sta SCREEN1_MEM + 440

        lda HORIZONTAL_BUFFER + 12
```

```
        sta SCREEN1_MEM + 480
        lda HORIZONTAL_BUFFER + 13
        sta SCREEN1_MEM + 520
        lda HORIZONTAL_BUFFER + 14
        sta SCREEN1_MEM + 560
        lda HORIZONTAL_BUFFER + 15
        sta SCREEN1_MEM + 600

        lda HORIZONTAL_BUFFER + 16
        sta SCREEN1_MEM + 640
        lda HORIZONTAL_BUFFER + 17
        sta SCREEN1_MEM + 680
        lda HORIZONTAL_BUFFER + 18
        sta SCREEN1_MEM + 720
        lda HORIZONTAL_BUFFER + 19
        sta SCREEN1_MEM + 760

        rts
@screen2
        lda HORIZONTAL_BUFFER
        sta SCREEN2_MEM
        lda HORIZONTAL_BUFFER + 1
        sta SCREEN2_MEM + 40
        lda HORIZONTAL_BUFFER + 2
        sta SCREEN2_MEM + 80
        lda HORIZONTAL_BUFFER + 3
        sta SCREEN2_MEM + 120

        lda HORIZONTAL_BUFFER + 4
        sta SCREEN2_MEM + 160
        lda HORIZONTAL_BUFFER + 5
        sta SCREEN2_MEM + 200
        lda HORIZONTAL_BUFFER + 6
        sta SCREEN2_MEM + 240
        lda HORIZONTAL_BUFFER + 7
        sta SCREEN2_MEM + 280

        lda HORIZONTAL_BUFFER + 8
        sta SCREEN2_MEM + 320
        lda HORIZONTAL_BUFFER + 9
        sta SCREEN2_MEM + 360
        lda HORIZONTAL_BUFFER + 10
        sta SCREEN2_MEM + 400
        lda HORIZONTAL_BUFFER + 11
        sta SCREEN2_MEM + 440

        lda HORIZONTAL_BUFFER + 12
```

```
        sta SCREEN2_MEM + 480
        lda HORIZONTAL_BUFFER + 13
        sta SCREEN2_MEM + 520
        lda HORIZONTAL_BUFFER + 14
        sta SCREEN2_MEM + 560
        lda HORIZONTAL_BUFFER + 15
        sta SCREEN2_MEM + 600

        lda HORIZONTAL_BUFFER + 16
        sta SCREEN2_MEM + 640
        lda HORIZONTAL_BUFFER + 17
        sta SCREEN2_MEM + 680
        lda HORIZONTAL_BUFFER + 18
        sta SCREEN2_MEM + 720
        lda HORIZONTAL_BUFFER + 19
        sta SCREEN2_MEM + 760

        rts
#endregion
;-------------------------------------------------------------------
-----------------------------------
;
DRAW RIGHT COLOR
;-------------------------------------------------------------------
-----------------------------------
#region "DrawLeftColor"
DrawLeftColor
        lda HORIZONTAL_COLOR_BUFFER
        sta COLOR_MEM
        lda HORIZONTAL_COLOR_BUFFER + 1
        sta COLOR_MEM + 40
        lda HORIZONTAL_COLOR_BUFFER + 2
        sta COLOR_MEM + 80
        lda HORIZONTAL_COLOR_BUFFER + 3
        sta COLOR_MEM + 120

        lda HORIZONTAL_COLOR_BUFFER + 4
        sta COLOR_MEM + 160
        lda HORIZONTAL_COLOR_BUFFER + 5
        sta COLOR_MEM + 200
        lda HORIZONTAL_COLOR_BUFFER + 6
        sta COLOR_MEM + 240
        lda HORIZONTAL_COLOR_BUFFER + 7
        sta COLOR_MEM + 280

        lda HORIZONTAL_COLOR_BUFFER + 8
        sta COLOR_MEM + 320
```

```
        lda HORIZONTAL_COLOR_BUFFER + 9
        sta COLOR_MEM + 360
        lda HORIZONTAL_COLOR_BUFFER + 10
        sta COLOR_MEM + 400
        lda HORIZONTAL_COLOR_BUFFER + 11
        sta COLOR_MEM + 440

        lda HORIZONTAL_COLOR_BUFFER + 12
        sta COLOR_MEM + 480
        lda HORIZONTAL_COLOR_BUFFER + 13
        sta COLOR_MEM + 520
        lda HORIZONTAL_COLOR_BUFFER + 14
        sta COLOR_MEM + 560
        lda HORIZONTAL_COLOR_BUFFER + 15
        sta COLOR_MEM + 600

        lda HORIZONTAL_COLOR_BUFFER + 16
        sta COLOR_MEM + 640
        lda HORIZONTAL_COLOR_BUFFER + 17
        sta COLOR_MEM + 680
        lda HORIZONTAL_COLOR_BUFFER + 18
        sta COLOR_MEM + 720
        lda HORIZONTAL_COLOR_BUFFER + 19
        sta COLOR_MEM + 760

        rts
#endregion
;-------------------------------------------------------------------
-----------------------------------
;
DRAW RIGHT BUFFER
;-------------------------------------------------------------------
-----------------------------------
; Draw the contents of the right buffer to the backbuffer -
unrolled for speed
;-------------------------------------------------------------------
-----------------------------------
#region "DrawRightBuffer"
DrawRightBuffer
        lda CURRENT_BUFFER + 1
        cmp #>SCREEN2_MEM
        beq @screen2

        lda HORIZONTAL_BUFFER
        sta SCREEN1_MEM + 39
        lda HORIZONTAL_BUFFER + 1
        sta SCREEN1_MEM + 79
```

```
        lda HORIZONTAL_BUFFER + 2
        sta SCREEN1_MEM + 119
        lda HORIZONTAL_BUFFER + 3
        sta SCREEN1_MEM + 159

        lda HORIZONTAL_BUFFER + 4
        sta SCREEN1_MEM + 199
        lda HORIZONTAL_BUFFER + 5
        sta SCREEN1_MEM + 239
        lda HORIZONTAL_BUFFER + 6
        sta SCREEN1_MEM + 279
        lda HORIZONTAL_BUFFER + 7
        sta SCREEN1_MEM + 319

        lda HORIZONTAL_BUFFER + 8
        sta SCREEN1_MEM + 359
        lda HORIZONTAL_BUFFER + 9
        sta SCREEN1_MEM + 399
        lda HORIZONTAL_BUFFER + 10
        sta SCREEN1_MEM + 439
        lda HORIZONTAL_BUFFER + 11
        sta SCREEN1_MEM + 479

        lda HORIZONTAL_BUFFER + 12
        sta SCREEN1_MEM + 519
        lda HORIZONTAL_BUFFER + 13
        sta SCREEN1_MEM + 559
        lda HORIZONTAL_BUFFER + 14
        sta SCREEN1_MEM + 599
        lda HORIZONTAL_BUFFER + 15
        sta SCREEN1_MEM + 639

        lda HORIZONTAL_BUFFER + 16
        sta SCREEN1_MEM + 679
        lda HORIZONTAL_BUFFER + 17
        sta SCREEN1_MEM + 719
        lda HORIZONTAL_BUFFER + 18
        sta SCREEN1_MEM + 759
        lda HORIZONTAL_BUFFER + 19
        sta SCREEN1_MEM + 799

        rts
@screen2

        lda HORIZONTAL_BUFFER
        sta SCREEN2_MEM + 39
        lda HORIZONTAL_BUFFER + 1
```

```
        sta SCREEN2_MEM + 79
        lda HORIZONTAL_BUFFER + 2
        sta SCREEN2_MEM + 119
        lda HORIZONTAL_BUFFER + 3
        sta SCREEN2_MEM + 159

        lda HORIZONTAL_BUFFER + 4
        sta SCREEN2_MEM + 199
        lda HORIZONTAL_BUFFER + 5
        sta SCREEN2_MEM + 239
        lda HORIZONTAL_BUFFER + 6
        sta SCREEN2_MEM + 279
        lda HORIZONTAL_BUFFER + 7
        sta SCREEN2_MEM + 319

        lda HORIZONTAL_BUFFER + 8
        sta SCREEN2_MEM + 359
        lda HORIZONTAL_BUFFER + 9
        sta SCREEN2_MEM + 399
        lda HORIZONTAL_BUFFER + 10
        sta SCREEN2_MEM + 439
        lda HORIZONTAL_BUFFER + 11
        sta SCREEN2_MEM + 479

        lda HORIZONTAL_BUFFER + 12
        sta SCREEN2_MEM + 519
        lda HORIZONTAL_BUFFER + 13
        sta SCREEN2_MEM + 559
        lda HORIZONTAL_BUFFER + 14
        sta SCREEN2_MEM + 599
        lda HORIZONTAL_BUFFER + 15
        sta SCREEN2_MEM + 639

        lda HORIZONTAL_BUFFER + 16
        sta SCREEN2_MEM + 679
        lda HORIZONTAL_BUFFER + 17
        sta SCREEN2_MEM + 719
        lda HORIZONTAL_BUFFER + 18
        sta SCREEN2_MEM + 759
        lda HORIZONTAL_BUFFER + 19
        sta SCREEN2_MEM + 799

        rts
#endregion
;-------------------------------------------------------------
---------------------------------
```

```
;
DRAW RIGHT COLOR
;-------------------------------------------------------------------
-----------------------------------
#region "DrawRightColor"
DrawRightColor
        lda HORIZONTAL_COLOR_BUFFER
        sta COLOR_MEM + 39
        lda HORIZONTAL_COLOR_BUFFER + 1
        sta COLOR_MEM + 79
        lda HORIZONTAL_COLOR_BUFFER + 2
        sta COLOR_MEM + 119
        lda HORIZONTAL_COLOR_BUFFER + 3
        sta COLOR_MEM + 159

        lda HORIZONTAL_COLOR_BUFFER + 4
        sta COLOR_MEM + 199
        lda HORIZONTAL_COLOR_BUFFER + 5
        sta COLOR_MEM + 239
        lda HORIZONTAL_COLOR_BUFFER + 6
        sta COLOR_MEM + 279
        lda HORIZONTAL_COLOR_BUFFER + 7
        sta COLOR_MEM + 319

        lda HORIZONTAL_COLOR_BUFFER + 8
        sta COLOR_MEM + 359
        lda HORIZONTAL_COLOR_BUFFER + 9
        sta COLOR_MEM + 399
        lda HORIZONTAL_COLOR_BUFFER + 10
        sta COLOR_MEM + 439
        lda HORIZONTAL_COLOR_BUFFER + 11
        sta COLOR_MEM + 479

        lda HORIZONTAL_COLOR_BUFFER + 12
        sta COLOR_MEM + 519
        lda HORIZONTAL_COLOR_BUFFER + 13
        sta COLOR_MEM + 559
        lda HORIZONTAL_COLOR_BUFFER + 14
        sta COLOR_MEM + 599
        lda HORIZONTAL_COLOR_BUFFER + 15
        sta COLOR_MEM + 639

        lda HORIZONTAL_COLOR_BUFFER + 16
        sta COLOR_MEM + 679
        lda HORIZONTAL_COLOR_BUFFER + 17
        sta COLOR_MEM + 719
        lda HORIZONTAL_COLOR_BUFFER + 18
```

```
        sta COLOR_MEM + 759
        lda HORIZONTAL_COLOR_BUFFER + 19
        sta COLOR_MEM + 799
        rts
#endRegion

;-------------------------------------------------------------------
-----------------------------------
;
COPY TO VERTICAL BUFFER
;-------------------------------------------------------------------
-----------------------------------
; Copy the data needed to the VERTICAL_BUFFER and
VERTICAL_COLOR_BUFFER for edge drawing
; of new characters on the 'jump frame'
;
; Note : this will have to be rewritten to combine vert and
horiz scrolling and delta values
;-------------------------------------------------------------------
-----------------------------------
#region "CopyVertBuffer"
CopyVerticalBuffer

        ; VARIABLES:
        ; PARAM1 = Map X position (for the tile to be read from)
        ; PARAM2 = Adjusted Map Y position for the tile to be
read from
        ; PARAM3 = Adjusted Map X Delta
        ; PARAM4 = Adjusted Map Y Delta

        lda MAP_X_POS           ; load the MAP X position
        sta PARAM1
        lda MAP_X_DELTA         ; load the MAP X Delta (position
within tile)
        sta PARAM3

        lda MAP_Y_POS

                                ; What direction are we
scrolling? Up or Down?
                                ; The direction will dictate how
we calculate what tile
                                ; we need to read from
        ldx SCROLL_DIRECTION
        cpx #SCROLL_DOWN
        beq @scrollingDown
```

```
        ;-------------------------------------------------------
-----------------------------------
                                ; DIRECTION UP
                                ; Scrolling up.  The tile we
need is the same if MAP_Y_DELTA
                                ; is > 0.   If it's equal to 0,
we need MAP_Y - 1.
                                ; The delta is obviously current
delta - 1
@scrollingUp

                                ; A currently holds MAP_Y_POS
        sta PARAM2              ; store it in PARAM2

        ldx MAP_Y_DELTA
        dex
        txa
        and #%0011
        sta PARAM4              ; store adjusted Y delta
        cmp #3                  ; original delta was 0
        bne @fetchTile          ; if != 3 tile doesn't decrease

        dec PARAM2              ; decrement MAP_Y (in PARAM2)
        jmp @fetchTile

        ;-------------------------------------------------------
-----------------------------------
                                ; DIRECTION DOWN
                                ; Scrolling down, the tile we
need is MAP_Y + 4
                                ; and the character line within
the tile is MAP_Y_DELTA + 2
                                ; This means we may be reading
from tile MAP_Y + 5 if the
                                ; MAP_Y_DELTA >= 2
                                ; So if we add 2 then and %0011
(3) to it, to mask it
                                ; back to a 0-3 value.. if our
value > MAP_Y_DELTA we are in the
                                ; same tile. Else we advance to
the next tile down
@scrollingDown

        clc                     ; A still contains MAP_POS_Y
        adc #4                  ; add 4 to get the correct tile
        sta PARAM2              ; PARAM 2 contains the the
'adjusted MAP Y POSITION'
```

```
        ldx MAP_Y_DELTA         ; Fetch Delta Y in X
        inx
        inx                     ; increment by 2
        txa                     ; transfer to A
        and #%0011               ; Mask to 0-3 value
        sta PARAM4              ; Save the adjusted Delta Value

        clc
        cmp MAP_Y_DELTA         ; compare - if carry is set,
value is >= than delta (same tile)
        bcs @fetchTile          ; so we take PARAM2 and continue
to fetch the tile

        inc PARAM2              ; increment our adjusted MAP_Y
to the next tile down
        ;------------------------ TO DO insert a bounds check /
wrap here

        ;-----------------------------------------------------
------------------------------
        ;                               Next we have to fetch
the correct tile from the map
        ;                               OPTIMIZATION NOTE :
store the map Y line address so
        ;                               we don't have to look it
up every loop iteration

@fetchTile
        lda #0
        sta buffer_index                ; reset to the start of
the buffer
;        sta tile_counter                ; and reset the tile
counter for the loop


                                        ; Later we need to use
the adjusted Y delta value
                                        ; to get the correct
line in the tile. Each line is
                                        ; 4 tiles, so we need to
multiply the value by 4
                                        ; Once added to the tile
address, we can then pull
                                        ; values from the
correct line
        asl PARAM4
        asl PARAM4
```

```
        ldx PARAM2                      ; get our adjusted
MAP_Y_POS to get the map line
        lda MAP_LINE_LOOKUP_LO,x        ; and store the address
for that line in
        sta ZEROPAGE_POINTER_2          ; ZEROPAGE_POINTER_1
        lda MAP_LINE_LOOKUP_HI,x
        sta ZEROPAGE_POINTER_2 + 1


@tileloop                               ; We then use the MAP_X
pos (PARAM1) to get the tile
        ldy PARAM1
        lda (ZEROPAGE_POINTER_2),y      ; Fetch the tile number
in A

        tax                             ; using the tile number
we can lookup the address
        lda TILE_NUMBER_LOOKUP_LO,x     ; of the tile itself and
store it in ZEROPAGE_POINTER_1
        sta ZEROPAGE_POINTER_1
        lda TILE_NUMBER_LOOKUP_HI,x
        sta ZEROPAGE_POINTER_1 + 1
                                        ; Next we have to adjust
the address by delta value
                                        ; to get the correct
line
        clc
        lda ZEROPAGE_POINTER_1
        adc PARAM4
        sta ZEROPAGE_POINTER_1
        lda ZEROPAGE_POINTER_1 + 1
        adc #0
        sta ZEROPAGE_POINTER_1 + 1

        ;-------------------------------------------------------
-----------------------
        ;                               ; From here we loop
through the tile and copy
        ;                               ; the character data and
the color data to the
        ;                               ; VERTICAL_BUFFER and
VERTICAL_COLOR_BUFFER

        ;                               Use the adjusted X delta
as our start point
```

```
        ;                                   because we won't always
be on a tile boundry
        ;                                   NOTE: this is only for
the FIRST tile - afterwards
        ;                                         we will reset it
to 0 for a full tile

;        ldy #0

@copyloop
        ldy PARAM3

        ldx buffer_index                    ; load the buffer index
in x
        lda (ZEROPAGE_POINTER_1),y          ; Copy character from
tile data
        sta VERTICAL_BUFFER,x               ; to buffer + x

        tax                                 ; copy tile number to x
        lda ATTRIBUTE_MEM,x                 ; copy attribute for
tile x
        ldx buffer_index                    ; reload the
buffer_index in x
        sta VERTICAL_COLOR_BUFFER,x         ; store attribute data
in color buffer + x

        inc buffer_index                    ; increment the
buffer_index
        lda #40                             ; check for the end of
the buffer
        cmp buffer_index
        beq @done

;        iny                                 ; increment position in
tile data
;        cpy #4                              ; test for edge of tile

        inc PARAM3                          ; increment X delta
(adjusted)
        lda #4
        cmp PARAM3
;        lda PARAM3                         ; mask it to 0-3 value
;        and #%0011
        bne @copyloop                       ; branch if not 0

        lda #0
        sta PARAM3
```

```
        inc PARAM1                          ; increment PARAM1 to
the next tile
        jmp @tileloop
@done
        rts


#endregion
;-------------------------------------------------------------------
-----------------------------------
;
COPY TO HORIZONTAL BUFFER
;-------------------------------------------------------------------
-----------------------------------
; Copy the data needed to the HORIZONTAL_BUFFER and
HORIZONTAL_COLOR_BUFFER for edge drawing
; of new chars on the 'jump frame'
;
; V2 of this routine - rewritten and split from
CopyHorizontalBuffer to stay consistant with the
; new CopyLeftBuffer routine
;-------------------------------------------------------------------
-----------------------------------
#region "CopyHorizBuffer"
CopyHorizontalBuffer
CopyRightBuffer

        ; VARIABLES
        ; PARAM1 = Adjusted Map X Position
        ; PARAM2 = Adjusted Map Y Position
        ; PARAM3 = Adjusted Map X Delta
        ; PARAM4 = Adjusted Map Y Delta

        ;-------------------------------------------------------
-------------------------
        ; First we need to know what tile to fetch. Scrolling
right this will be
        ; MAP_POS_X + 10 (the map is 10 tiles wide)

        lda MAP_Y_POS                       ; Setup the variables we
don't need to adjust
        sta PARAM2
        lda MAP_X_DELTA
        sta PARAM3
        lda MAP_Y_DELTA
        sta PARAM4
```

```
        lda MAP_X_POS                   ; add 10 to MAP_X_POS
and store in PARAM1
        clc
        adc #10
        sta PARAM1


        ;-------------------------------------------------------
------------------------
        ; Variables are setup - now we need to fetch the tile to
read

@fetchtile
        lda #0
        sta buffer_index                ; reset the buffer index

@tileloop
        ldx PARAM2                      ; fetch adjusted
MAP_Y_POSITION
        lda MAP_LINE_LOOKUP_LO,x
        sta ZEROPAGE_POINTER_2          ; store the map line
address in ZEROPAGE_POINTER_2
        lda MAP_LINE_LOOKUP_HI,x
        sta ZEROPAGE_POINTER_2 + 1

                                        ; Next use the adjusted
MAP_X_POS to get the tile
        ldy PARAM1
        lda (ZEROPAGE_POINTER_2),y      ; Fetch the tile number
in A

        tax
        lda TILE_NUMBER_LOOKUP_LO,x     ; use the tile number to
lookup the address of the tile data
        sta ZEROPAGE_POINTER_1
        lda TILE_NUMBER_LOOKUP_HI,x
        sta ZEROPAGE_POINTER_1 + 1

        ;-------------------------------------------------------
------------------------
        ; We now have the address to the tile in
ZEROPAGE_POINTER_1, so now we have to
        ; loop through the tile at our delta X value (PARAM3)
for all 4 lines and copy
        ; the character and color attribute data to their
buffers.
```

```
                                        ; Our first block lookup
depends on the map Y delta
                                        ; so test to see if it's
0 - which is should unless it's
                                        ; the first lookup

        ldy PARAM3                      ; load the adjusted X
Delta in Y
        lda PARAM4                      ; load the adjusted Y
Delta
        beq @copyloop                   ; if it's 0, continue

        asl                             ; else multiply it by 4
(tile line)
        asl
        clc
        adc PARAM3                      ; add PARAM3 (adjusted X
delta)
        tay                             ; transfer it to Y as
our new start index

@copyloop
        ldx buffer_index                ; load the buffer index
        lda (ZEROPAGE_POINTER_1),y      ; fetch the character #
        sta HORIZONTAL_BUFFER,x         ; store it in the
horizontal buffer

        tax                             ; save the character
number in X
        lda ATTRIBUTE_MEM,x             ; use it to fetch that
characters attribute data
        ldx buffer_index                ; restore buffer_index
to x
        sta HORIZONTAL_COLOR_BUFFER,x   ; save the attribute
data in the color buffer

        iny                             ; Add 4 to Y to get to
the next tile line
        iny
        iny
        iny

        inc buffer_index                ; increment the buffer
index
        lda buffer_index                ; if the buffer is full,
we're done
        cmp #19
```

```
        beq @done

        inc PARAM4                      ; use delta Y as a
counter to the tiles end
        lda PARAM4
        cmp #4
        bne @copyloop
                                        ; Setup for the next
tile
        lda #0
        sta PARAM4                      ; reset delta Y to 0
(PARAM4)
        inc PARAM2                      ; increment MAP Y
position to the next tile line
        jmp @tileloop


@done

        rts


buffer_index
        byte 0

#endregion
;=================================================================
==================================
;
COPY LEFT BUFFER
;=================================================================
==================================
; Split from CopyHorizontal Buffer due to a bug that I just
can't seem to track down.
; Hopefully a rewrite of the routine will yeild a cleaner way of
doing things
; Copy Horizontal Buffer was a much cleaner routine, so we'll
look at that as well
;
; Actually it turned out so well that I decided to rewrite the
original CopyHorizontalBuffer
; routine, as this version will be much easier to integrate
handling MAP_Y_DELTA when I
; include vertical scrolling
;-----------------------------------------------------------------
----------------------------------
#region "CopyLeftBuffer"
```

```
CopyLeftBuffer

        ; VARIABLES
        ; PARAM1 = Adjusted Map X Position
        ; PARAM2 = Adjusted Map Y Position
        ; PARAM3 = Adjusted Map X Delta
        ; PARAM4 = Adjusted Map Y Delta

        ;-------------------------------------------------------
-----------------------
        ; First we need to know what map tile we are using -
moving left that is
        ; MAP_X_POS or MAP_X_POS - 1 depending on the
MAP_X_DELTA.
        ; On a delta value of 1 - 3 we will be on the same tile.
        ; On a delta value of 0 we will need one tile over.

        lda MAP_Y_POS                   ; store map Y pos, this
won't change
        sta PARAM2
        lda MAP_Y_DELTA                 ; store map Y delta
        sta PARAM4

        lda MAP_X_POS                   ; store map X pos, now
work on the adjusted value
        sta PARAM1                      ; based on the
MAP_X_DELTA

        ldx MAP_X_DELTA                 ; Fetch the delta value
        dex                             ; decrement it by 1
        txa                             ; transfer it to A
        and #%0011                      ; mask it to a value of
0 - 3
        sta PARAM3                      ; store this as the
adjusted x delta (current - 1)

        cmp #3                          ; if our new delta is 3,
we are on a new tile
        bne @fetchtile                  ; if not, we fetch the
current tile

        dec PARAM1                      ; adjust our Map X
position by -1

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

```
        ; PARAM1 should hold the correct MAP_X_POS we need to
look up
        ; PARAM2 should hold the (unchanged) MAP_Y_POS we need
to look up
        ; PARAM3 should hold the correct MAP_X_DELTA of the tile
to look up
        ; PARAM4 should hold the (unchanged) MAP_Y_DELTA for the
tile

        ; Now we need to get the fetch the tile

@fetchtile

        lda #0                          ; reset to the start of
the buffer
        sta buffer_index

@tileloop
        ldx PARAM2                      ; fetch MAP_Y_POS to
lookup the Map line address
        lda MAP_LINE_LOOKUP_LO,x
        sta ZEROPAGE_POINTER_2          ; Store the map line in
ZEROPAGE_POINTER_2
        lda MAP_LINE_LOOKUP_HI,x
        sta ZEROPAGE_POINTER_2 + 1

                                        ; Next use the adjusted
MAP_X_POS to get the tile
        ldy PARAM1
        lda (ZEROPAGE_POINTER_2),y      ; Fetch the tile number
in A

        tax                             ; Use the tile number to
lookup the address of the tile data
        lda TILE_NUMBER_LOOKUP_LO,x
        sta ZEROPAGE_POINTER_1          ; and store it in
ZEROPAGE_POINTER_1
        lda TILE_NUMBER_LOOKUP_HI,x
        sta ZEROPAGE_POINTER_1 + 1

        ;-------------------------------------------------------
-----------------------
        ; We now have the address to the tile in
ZEROPAGE_POINTER_1 - so now we have
        ; to loop through it and copy the data at our delta X
value (PARAM3) for all
```

```
        ; 4 lines of the tile into the buffer. We also need the
color attribute data
        ; for that character (as we've packed info into the
upper half of the color byte)

                                        ; Our first block lookup
depends on the map Y delta
                                        ; so test to see if it's
0 - which is should unless it's
                                        ; the first lookup

        ldy PARAM3                      ; load the adjusted X
Delta in Y
        lda PARAM4                      ; load the adjusted Y
Delta
        beq @copyloop                   ; if it's 0, continue

        asl                             ; else multiply it by 4
(tile line)
        asl
        clc
        adc PARAM3                      ; add PARAM3 (adjusted X
delta)
        tay                             ; transfer it to Y as
our new start index


@copyloop
        ldx buffer_index                ; load the buffer index
in X

        lda (ZEROPAGE_POINTER_1),y      ; fetch the character #
        sta HORIZONTAL_BUFFER,x         ; store it in the buffer

        tax                             ; save character number
in X
        lda ATTRIBUTE_MEM,x             ; lookup the attribute
for that character
        ldx buffer_index                ; reload the buffer
index in X
        sta HORIZONTAL_COLOR_BUFFER,x   ; so we can save the
attribute in the color buffer

        iny                             ; Add 4 to Y to get to
the next tile line
        iny
        iny
```

```
        iny

        inc buffer_index                ; increment the buffer
index
        lda buffer_index                ; if the buffer is full,
we're done
        cmp #19                         ; length of buffer_index
        beq @done

        inc PARAM4                      ; use the delta Y as a
counter to the tiles end
        lda PARAM4
        cmp #4
        bne @copyloop
                                        ; Setup for the next
tile
        lda #0
        sta PARAM4                      ; Set our Delta Y to 0 -
because it's a new tile
        inc PARAM2                      ; increment our MAP Y
position to the next tile line
        jmp @tileloop



@done
        rts

#endregion
;=================================================================
====================================
;
SCROLL DATA + TABLES
;=================================================================
====================================


SCROLL_FIX_SKIP                         ; Check to see if we can
'skip' scroll fixing on stop
        byte $0
;-----------------------------------------------------------------
--------- EDGE BUFFERS FOR MAP DRAW
HORIZONTAL_BUFFER
        byte 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
HORIZONTAL_COLOR_BUFFER
        byte 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
VERTICAL_BUFFER
        byte
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0
VERTICAL_COLOR_BUFFER
        byte
3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3,3,3
```