```
; ********************************************************************************
put_dot:
   ; ***** BUFFER16 *****
   ; +0: X
   ; +1: remainder X
   ; +2: Y
   ; +3: remainder Y
   ; +4: palette
   stx BUFFER16
   sty BUFFER16+2
   sta BUFFER16+4

   ; ************************ dealing with X
   lda BUFFER16
   sta BUFFER16+1
   and #3                    ; 0000.0011
   sta BUFFER16+1            ; store remainder
   lsr BUFFER16              ;; X divided
   lsr BUFFER16              ;; by 4

   ; ************************ dealing with Y
   lda BUFFER16+2
   sta BUFFER16+3
   and #7                    ; 0000.0111
   sta BUFFER16+3            ; store remainder
   lsr BUFFER16+2            ;; Y divided
   lsr BUFFER16+2            ;;
   lsr BUFFER16+2            ;; by 8

   ; ************************************************ (Y * 40) + X = CELL
   ; ************************ Y * 40
   ; ************************ 40 = 0010.1000
   ; ******************** Y << 5; Y * 32
   lda BUFFER16+2
   sta BUFFER16+6            ; using BUFFER16+6
   lda #0                    ; for
   sta BUFFER16+7            ; 16-bit integer

   clc
   ldx #5
:
   asl BUFFER16+7
   asl BUFFER16+6
   bcc :+
   inc BUFFER16+7
:
   dex
   bne :--

   ; ******************** Y << 3; Y * 8
   lda BUFFER16+2
   sta BUFFER16+8            ; using BUFFER16+8
   lda #0                    ; for
   sta BUFFER16+9            ; 16-bit integer

   clc
   ldx #3
:
   asl BUFFER16+8
   bcc :+
   inc BUFFER16+9
:
   dex
   bne :--

   ; ******************** BUFFER16+6 plus BUFFER16+8 = BUFFER16+10
   ; ******************** Note: Y * 40
   clc
   lda BUFFER16+6
   adc BUFFER16+8
   sta BUFFER16+10
   lda BUFFER16+7
   adc BUFFER16+9
   sta BUFFER16+11

   ; ******************** Add X to BUFFER16+10 to get CELL
   ; 6,7
   lda BUFFER16
```

```
   sta BUFFER16+6
   lda #0
   sta BUFFER16+7
   ; 8,9
   lda BUFFER16+10
   sta BUFFER16+8
   lda BUFFER16+11
   sta BUFFER16+9
   ; ***** add the X
   clc
   lda BUFFER16+6
   adc BUFFER16+8
   sta BUFFER16+10
   lda BUFFER16+7
   adc BUFFER16+9
   sta BUFFER16+11

   ; ******************** CELL * 8
   clc
   ldx #3
:
   asl BUFFER16+11
   asl BUFFER16+10
   bcc :+
   inc BUFFER16+11
:
   dex
   bne :--

   ; ********** add remainder Y
   ; 6,7
   lda BUFFER16+3
   sta BUFFER16+6
   lda #0
   sta BUFFER16+7
   ; 8,9
   lda BUFFER16+10
   sta BUFFER16+8
   lda BUFFER16+11
   sta BUFFER16+9
   clc
   lda BUFFER16+6
   adc BUFFER16+8
   sta BUFFER16+10
   lda BUFFER16+7
   adc BUFFER16+9
   sta BUFFER16+11

   ; ********** add SCREEN_MEM
   ; 6,7
   lda #<SCREEN_MEM
   sta BUFFER16+6
   lda #>SCREEN_MEM
   sta BUFFER16+7
   ; 8,9
   lda BUFFER16+10
   sta BUFFER16+8
   lda BUFFER16+11
   sta BUFFER16+9
   clc
   lda BUFFER16+6
   adc BUFFER16+8
   sta BUFFER16+10
   lda BUFFER16+7
   adc BUFFER16+9
   sta BUFFER16+11

   ; ********************************************** define palette-pixel
   lda BUFFER16+10
   sta 2
   lda BUFFER16+11
   sta 3

   ; ***** store original value of screen-byte in memory 4
   ldy #0
   lda (2),y
   sta 4
```

```
   ; ***** store palette in memory 5
   lda BUFFER16+4
   sta 5
   ; *** duplicate palette; EG, from 0000.0010 to 1010.1010
   asl
   asl
   ora 5
   sta 5
   asl
   asl
   ora 5
   sta 5
   asl
   asl
   ora 5
   sta 5

   ; ***** convert remainder X to pixel pattern and store value in memory 6; mask
   lda BUFFER16+1
   cmp #3
   beq three
   jmp :+
three:
   lda #3                   ; 0000.0011
   jmp continue
:
   cmp #2
   beq two
   jmp :+
two:
   lda #12                  ; 0000.1100
   jmp continue
:
   cmp #1
   beq one
   jmp :+
one:
   lda #48                  ; 0011.0000
   jmp continue
:
   lda #192                 ; 1100.0000
continue:
   sta 6                    ; mask
   ; ***** create ~mask and store it in memory 7
   eor #$ff
   sta 7

   ; ***** palette AND mask and store it in memory 8; aaa
   lda 5
   and 6
   sta 8

   ; ***** screen-byte AND ~mask and store it in memory 9; bbb
   lda 4
   and 7
   sta 9

   ; ***** aaa OR bbb and store it in screen
   lda 8
   ora 9
   ldy #0
   sta (2),y

   rts
```

```
; ******************************************************** clear background with a color
fill_background:
   stx BACKGROUND
   sty BORDER

   lda #<SCREEN_MEM
   sta MEM_TWO
   lda #>SCREEN_MEM
   sta MEM_TWO+1

   ldy #0
   lda #0
   ldx #32
:
   sta (MEM_TWO),y
   iny
   bne :-
   inc MEM_TWO+1
   dex
   bne :-

   rts

; ********************************************************************************************
set_color_ram:
   sta TMP

   lda #<COLOR_RAM
   sta MEM_TWO
   lda #>COLOR_RAM
   sta MEM_TWO+1

   lda TMP
   ldy #0
   ldx #4
:
   sta (MEM_TWO),y
   iny
   bne :-
   inc MEM_TWO+1
   dex
   bne :-

   rts

; ********************************************************************************************
set_color_cells:
   sta TMP

   lda #<BANK
   sta MEM_TWO
   lda #>BANK
   sta MEM_TWO+1

   lda TMP
   ldy #0
   ldx #4
:
   sta (MEM_TWO),y
   iny
   bne :-
   inc MEM_TWO+1
   dex
   bne :-
   rts

; ********************************************************************************************
set_multi_color_mode:
   ; ***** disable I/O & error messages
   lda MESSAGE
   and #$3f
   sta MESSAGE

   ; ***** turn off BASIC
   lda $1
   and #$fc
   ora #2
   sta $1
```

```
; ***** turn on bitmap
lda SCREEN_CONTROL_1
ora #32
sta SCREEN_CONTROL_1

; ***** turn on multi-color
lda SCREEN_CONTROL_2
ora #16
sta SCREEN_CONTROL_2

; ***** Bank
lda VIC_BANK
and #$fc
ora #1            ; Bank #2, $8000-$BFFF, 32768-49151.
sta VIC_BANK

; ***** (high nibble; 0) $0
; ***** (low nibble; 8)  $8
lda #$8
sta MEM_SETUP

rts
```