```
;  4 Ways Scroll
;  by malcolm bamber
;  http://www.dark-well.pwp.blueyonder.co.uk/
;  Assembler Used C64ASM.EXE
word1      word          $0801; Starting address for loader
*         = $0801
word2      word          nextLine; Line link
word3      word          $0; Line number
        byte          158; SYS
        byte          '1','4','5','0','0'
        byte          0
nextLine
        byte          0,0      ; end of basic
*         = 14500
asemcode
Ptrhiddenscreen = $2b                  ; hidden screen address
two byte address
Ptrscreen = $2d                    ; screen address you can see
two byte address
Ptrmap    = $2f                    ; map address
two byte address
Ptrmapcolour = $31                    ; map colour address
two byte address
Ptrcolour = $33                    ; colour address
two byte address
PtrSparecolour = $35
raster    = 245
free      = $5                     ; THESE ARE FREE AT THE
MONENT
free1     = $6                     ; THESE ARE FREE AT THE
MOMENT
scrollstop = $7                     ; if set then no stop
scrolling screen
sync      = $8
xscroll   = $9                     ; screen is all the way to
the left
yscroll   = $c                     ; screen is all the way to
the up
whichscreen = $d                     ; which screen we are
showing 1024 or 3072
mapwidth  = $e
maphight  = $f
mapx      = $12                    ; this will tell how far left
or right we are on map
mapy      = $14                    ; this will tell how far up
or down we are on the map
udflag    = $15                    ; if udflag = 0
```

```
lrflag    = $16                    ; what side of
key       = $C5                    ; WHAT KEY IS PRESSED
temp0     = $37
temp1     = $38
temp2     = $39
temp3     = $3a
temp4     = $3b
temp5     = $3c
temp6     = $3d
temp7     = $3e
temp8     = $3f
temp9     = $40
temp10    = $41
temp11    = $42
temp12    = $43
temp13    = $44
temp14    = $45
temp15    = $46
temp16    = $fb
temp17    = $fc
temp18    = $fd
temp19    = $fe
temp20    = $ff
temp21    = $20
maxwidth  = 39                     ; You must take 20 from
mapwidth has 20 tiles are show across
maxheight = 23                     ; you must take 13 from
maphight has 13 tiles arw show down
;*** setup varibles  ****
        lda             #0         ; MAKE IRQ JUMP OUT BEFORE
DOING ANY SCROLL WORK
        sta             scrollstop; STORE VALUE HERE
        lda             #0         ; SET SCREEN XSCROLL POSITION
NEAR THE MIDDLE 7=left 0=right
        sta             xscroll    ;
        lda             #0         ; SET SCREEN YSCROLL POSITION
NEAR THE MIDDLE 7=up 0=down
        sta             yscroll    ;
        lda             #59        ; TILES ACROSS
        sta             mapwidth   ; MAP WIDTH
        lda             #37        ; ROW DOWN
        sta             maphight   ; MAP HEIGHT
        lda             #$0f
        sta             $d418      ; SELECT FILTER MODE AND VOLUME
        lda             #0
        sta             mapx       ; THIS WILL TELL HOW FAR LEFT
OR RIGHT WE ARE ON MAP
```

```
        lda             #0
        sta             mapy        ; THIS WILL TELL HOW FAR UP OR
DOWN WE ARE ON THE MAP
        sta             sync        ; WAIT FOR RASTER TO START AT
THE TOP
        lda             #0
        sta             free
        sta             free1
        lda             #0

        sta       udflag                    ; WHICH SIDE TO DRAW
NEXT TILE STRIP TOP OR BOTTOM OF SCREEN
        sta       lrflag                    ; WHICH SIDE TO DRAW
NEXT TILE STRIP LEFT OR RIGHT OF SCREEN
        lda       #<1024                    ; SET ADDRESS OF
SCREEN YOU CAN SEE
        sta       Ptrscreen                 ; SET CURRENT SCREEN
BITMAP
        lda       #>1024
        sta       Ptrscreen+1               ; SET CURRENT SCREEN
BITMAP
        lda       #<3072                    ; SET ADDRESS OF
SCREEN THAT IS HIDDEN
        sta       Ptrhiddenscreen           ; SET HIDDEN SCREEN
BITMAP
        lda       #>3072
        sta       Ptrhiddenscreen+1           ; SET HIDDEN SCREEN
BITMAP
        lda       #1
        sta       whichscreen               ; WHICH SCREEN WE ARE
SHOWING 1024 OR 3072
        lda       #1                        ; TEXT COLOUR
        jsr       $ffd2                     ; SET COLOUR FOR
KERNAL SCREEN PRINTING
        jsr       setup                     ; SET UP MEMORY FOR
GAME
        jsr       $e544                     ; CLEAR SCREEN
        ;inc $d020
        jsr       setcursor
        jsr       makemapynumbers           ; WORK OUT ALL 16 BIT
ADD VALUES FOR MAPY
        ;***************
        ;** MAIN LOOP **
        ;***************
main
        jsr       setupirq                  ; START IRQ
```

```
        ldy       #0                        ; FILL SCREEN WITH
VALUE IN Y AND FILL COLOUR WITH VALUE IN X
        ldx       #0
        lda       Ptrscreen
        sta       temp0
        lda       Ptrscreen+1
        sta       temp1
        lda       Ptrcolour
        sta       temp2
        lda       Ptrcolour+1
        sta       temp3
        jsr       fillscreen
        ldy       #0                        ; FILL SCREEN WITH
VALUE IN Y AND FILL COLOUR WITH VALUE IN X
        ldx       #0
        lda       Ptrhiddenscreen
        sta       temp0
        lda       Ptrhiddenscreen+1
        sta       temp1
        lda       PtrSparecolour
        sta       temp2
        lda       PtrSparecolour+1
        sta       temp3
        jsr       fillscreen
        lda       Ptrscreen                 ; SCREEN ADDRESS
        sta       temp0
        lda       Ptrscreen+1
        sta       temp1
        lda       Ptrcolour                 ; COLOUR ADDRESS
        sta       temp2
        lda       Ptrcolour+1
        sta       temp3
        jsr       drawscreen                ; DRAW MAP ON SCREEN
        lda       Ptrscreen                 ; SCREEN ADDRESS
        sta       temp1
        lda       Ptrscreen+1
        sta       temp2
        lda       Ptrcolour                 ; COLOUR ADDRESS
        sta       temp3
        lda       Ptrcolour+1
        sta       temp4
        jsr       fillrightside             ; DRAW RIGHT SIDE OF
SCREEN
        clc
        lda       Ptrscreen                 ; SCREEN ADDRESS
        adc       #<960
        sta       temp1
```

```
        lda     Ptrscreen+1
        adc     #>960
        sta     temp2
        clc
        lda     Ptrcolour               ; COLOUR ADDRESS
        adc     #<960
        sta     temp3
        lda     Ptrcolour+1
        adc     #>960
        sta     temp4
        jsr     fillbottom              ; DRAW BOTTOM ROW ON
SCREEN
mainloop
        lda     sync                    ; WAIT FOR SYNC TO =
ONE
        cmp     #1
        bne     mainloop
        lda     #0                      ; CLEAR OLD SYNC FLAG
        sta     sync
        lda     scrollstop              ; CHECK IRQ FLAG
        cmp     #0                      ; IS IT ZERO
        bne     mainloop                ; NO
        lda     197                     ; get key
        cmp     #1                      ; is it returnkey
        bne     next1                   ; no
        jsr     swapscreen
        jsr     setcursor               ; yes
        ldx     mapx
        ldy     #0
        jsr     printnum
        jsr     swapscreen
next1
joystick
        ;lda $DC01                      ; VALUE FOR JOYSTICK IN
PORT ONE
        ;cmp #$7F                       ; NEUTRAL
        ;bne _checkforup
        lda     $DC00                   ; VALUE FOR JOYSTICK
IN PORT TWO
        cmp     #$7F
        bne     checkforup
        jmp     mainloop
checkforup
        cmp     #$7E                    ; UP
        bne     checkfordown            ; NO NOT UP
        lda     mapy                    ; GET VALUE OF MAP
POINTER
```

```
        cmp     #0                      ; MAKE SURE WE CAN
STILL MOVE DOWN
        beq     quitup                  ; NO
        lda     #2                      ; VALUE TO USE
        sta     scrollstop              ; SET FLAG TO SCROLL
UP
        lda     #3                      ; YSCROLL VALUE
        sta     yscroll                 ; SET SCREEN STARTING
POSITION
quitup
        jmp     mainloop                ; END OF JOYSTICK
LEFT
checkfordown
        cmp     #$7D                    ; DOWN
        bne     checkforleft            ; NO NOT DOWN
        lda     mapy                    ; GET VALUE OF MAP
POINTER
        cmp     #maxheight              ; MAKE SURE WE CAN
STILL MOVE DOWN
        beq     quitdown                ; NO
        lda     #2                      ; VALUE TO USE
        sta     scrollstop              ; SET FLAG TO SCROLL
DOWN
        lda     #4                      ; YSCROLL VALUE
        sta     yscroll                 ; SET SCREEN STARTING
POSITION
quitdown
        jmp     mainloop                ; END OF JOYSTICK
RIGHT
checkforleft
        cmp     #$7B                    ; LEFT
        bne     checkforright           ; NO NOT LEFT
        lda     mapx                    ; GET VALUE OF MAP
POINTER
        cmp     #0                      ; MAKE SURE WE CAN
STILL MOVE LEFT
        beq     quitleft                ; NO
        lda     #1                      ; VALUE TO USE
        sta     scrollstop              ; SET FLAG TO SCROLL
LEFT
        lda     #3                      ; XSCROLL VALUE
        sta     xscroll                 ; SET SCREEN STARTING
POSITION
quitleft
        jmp     mainloop                ; END OF JOYSTICK
LEFT
checkforright
```

```
        cmp     #$77                    ; RIGHT
        bne     quitright               ; NO NOT RIGHT
        lda     mapx                    ; GET VALUE OF MAP
POINTER
        cmp     #maxwidth               ; MAKE SURE WE CAN
STILL MOVE LEFT
        beq     quitright               ; NO
        lda     #1                      ; VALUE TO USE
        sta     scrollstop              ; SET FLAG TO SCROLL
RIGHT
        lda     #4                      ; XSCROLL VALUE
        sta     xscroll
        sta     $d020
                ; SET SCREEN STARTING POSITIO
quitright
        jmp     mainloop                ; END OF JOYSTICK
RIGHT
        ; quit
quitout
        lda     $D016                   ; SELECT 38/40 COLUMN
TEXT DISPLAY: 1 = 40 COLS
        eor     #%00001000              ; BIT 3=1 40 COLS
MODE,BIT 4=1 MULTI-COLOR MODE
        sta     $D016
        lda     $D016                   ; END OF SCROLL PART
OF THE SCREEN
        and     #%11111000
        ora     #7
        sta     $D016                   ; MOVE THE SCREEN ALL
THE WAY LEFT
        rts
        ;*************************************
        ;* WORK OUT Y VALUES OF MAP ADDRESS *
        ;*************************************
makemapynumbers
        lda     #0                      ; mapy value to use
        sta     temp8
        lda     #0
        sta     temp9                   ; index to pointer
array
loop1
        lda     temp8                   ; number to times
        sta     temp0
        lda     #0
        sta     temp1
        lda     mapwidth                ; multiplicand
        sta     temp2
```

```
        lda     #0
        sta     temp3
        jsr     mult16                  ; scratch temp0 -
temp7
        ldy     temp9
        lda     temp4                   ; low byte value
return from mult16
        sta     mapyaddress,y           ; store low byte
        iny
        lda     temp5                   ; high byte value
return from mult16
        sta     mapyaddress,y           ; store high byte
        inc     temp9
        inc     temp9                   ; move y index
pointer
        inc     temp8                   ; next y value
        lda     maphight                ; get the map hight
        cmp     temp8                   ; is y value = to it
        bne     loop1                   ; no
        rts                             ; yes
        ;*****************************************
        ;* FILL TOP ROW ON SCREEN                *
        ;* using temp1 - temp13                  *
        ;* temp 1 & 2   = screen address         *
        ;* temp 3 & 4   = colour address         *
        ;* temp 5 & 6   = Ptrmap address         *
        ;* temp 7 & 8   = Ptrmapcolour           *
        ;* temp 10      = X Position On Screen   *
        ;* temp 11      = X Position On Map      *
        ;* temp 12      = Ascii Value From Map   *
        ;* temp 13      = Colour Value  From Map *
        ;*****************************************
filltop
        clc
        lda     mapy                    ; get map array index
        asl     a
        tay
        clc
        lda     Ptrmap                  ; map address
        adc     mapyaddress,y           ; RESULT FROM
mapy*mapwidth
        sta     temp5
        lda     Ptrmap+1
        adc     mapyaddress+1,y         ; RESULT FROM
mapy*mapwidth
        sta     temp6
        clc
```

```
        lda     Ptrmapcolour            ; map colour address
        adc     mapyaddress,y           ; RESULT FROM
        sta     temp7
        lda     Ptrmapcolour+1
        adc     mapyaddress+1,y         ; RESULT FROM
        sta     temp8
        lda     #0
        sta     temp10                  ; SET ACROSS POSITION
ON SCREEN
        lda     mapx                    ; HOW FAR ACROSS MAP
        sta     temp11
drawtile1
        ldy     temp11                  ; SET ACROSS ON MAP
        lda     (temp5),y               ; GET FIRST CHAR ON
TILE FROM MAP
        ;lda #0                        ; ***** DEBUG ONLY
*****
        sta     temp12                  ; STORE IT
        lda     (temp7),y               ; COLOUR MAP
        sta     temp13                  ; STORE COLOUR OF
TILE
        lda     udflag                  ; ARE WE DRAWING THE
TOP OF THE TILE OR BOTTOM
        cmp     #1                      ; ARE WE DRAWING THE
BOTTOM PART OF THE TILE
        bne     drawfirst1              ; KEEP TEMP12 HAS IT
IS FOR DRAWING
        inc     temp12                  ; YES THEN SET CHAR
VALUE TO SECOND LINE OF TILE
        inc     temp12
drawfirst1
        ldy     temp10                  ; SET X POSITION ON
SCREEN
        lda     temp12                  ; GET TILE CHAR VALUE
        sta     (temp1),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp13                  ; GET COLOUR VALUE
        sta     (temp3),y               ; WRITE OUT COLOUR TO
SCREEN
        inc     temp10                  ; MOVE TO NEXT
POSITION ON SCREEN
drawsecond1
        ldy     temp10                  ; SET X POSITION ON
SCREEN
        ldx     temp12                  ; GET TILE CHAR VALUE
        inx
        txa
```

```
        sta         (temp1),y                   ; WRITE OUT CHAR TO
SCREEN
        lda         temp13                      ; GET COLOUR VALUE
        sta         (temp3),y                   ; WRITE OUT COLOUR TO
SCREEN
        inc         temp10                      ; MOVE TO NEXT
POSITION ON SCREEN
        inc         temp11                      ; HOW FAR ACROSS MAP
        lda         temp10                      ; CHECK HOW MANY TILE
WE HAVE DRAWN
        cmp         #39                         ; HAVE WE DONE A FULL
ROW ACROSS
        beq         quit1
        cmp         #40                         ; HAVE WE DONE A FULL
ROW ACROSS
        beq         quit1
        jmp         drawtile1                   ; JUMP IF CMP >TEMP10
quit1
        rts
        ;****************************************
        ;* FILL BOTTOM ROW ON SCREEN            *
        ;* using temp1 - temp13                 *
        ;* temp 1 & 2   = screen address       *
        ;* temp 3 & 4   = colour address       *
        ;* temp 5 & 6   = Ptrmap address       *
        ;* temp 7 & 8   = Ptrmapcolour         *
        ;* temp 10      = X Position On Screen *
        ;* temp 11      = X Position On Map    *
        ;* temp 12      = Ascii Value From Map *
        ;* temp 13      = Colour Value  From Map *
        ;****************************************
fillbottom
        clc
        lda         mapy                        ; get map array index
        adc         #12                         ;
        asl         a
        tay
        clc
        lda         Ptrmap                      ; map address
        adc         mapyaddress,y               ; RESULT FROM
mapy*mapwidth
        sta         temp5
        lda         Ptrmap+1
        adc         mapyaddress+1,y             ; RESULT FROM
mapy*mapwidth
        sta         temp6
        clc
```

```
        lda     Ptrmapcolour            ; map colour address
        adc     mapyaddress,y           ; RESULT FROM
        sta     temp7
        lda     Ptrmapcolour+1
        adc     mapyaddress+1,y         ; RESULT FROM
        sta     temp8
        lda     #0
        sta     temp10                  ; SET ACROSS POSITION
ON SCREEN
        lda     mapx                    ; HOW FAR ACROSS MAP
        sta     temp11
drawtile2
        ldy     temp11                  ; SET ACROSS ON MAP
        lda     (temp5),y               ; GET FIRST CHAR ON
TILE FROM MAP
        ;lda #0                        ; ***** DEBUG ONLY
*****
        sta     temp12                  ; STORE IT
        lda     (temp7),y               ; COLOUR MAP
        sta     temp13                  ; STORE COLOUR OF
TILE
        lda     udflag                  ; ARE WE DRAWING THE
TOP OF THE TILE OR BOTTOM
        cmp     #1                      ; ARE WE DRAWING THE
BOTTOM PART OF THE TILE
        bne     drawfirst2              ; KEEP TEMP12 HAS IT
IS FOR DRAWING
        inc     temp12                  ; YES THEN SET CHAR
VALUE TO SECOND LINE OF TILE
        inc     temp12
drawfirst2
        ldy     temp10                  ; SET X POSITION ON
SCREEN
        lda     temp12                  ; GET TILE CHAR VALUE
        sta     (temp1),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp13                  ; GET COLOUR VALUE
        sta     (temp3),y               ; WRITE OUT COLOUR TO
SCREEN
        inc     temp10                  ; MOVE TO NEXT
POSITION ON SCREEN
drawsecond2
        ldy     temp10                  ; SET X POSITION ON
SCREEN
        ldx     temp12                  ; GET TILE CHAR VALUE
        inx
        txa
```

```
        sta     (temp1),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp13                  ; GET COLOUR VALUE
        sta     (temp3),y               ; WRITE OUT COLOUR TO
SCREEN
        inc     temp10                  ; MOVE TO NEXT
POSITION ON SCREEN
        inc     temp11                  ; HOW FAR ACROSS MAP
        lda     temp10                  ; CHECK HOW MANY TILE
WE HAVE DRAWN
        cmp     #39                     ; HAVE WE DONE A FULL
ROW ACROSS
        beq     quit2
        cmp     #40                     ; HAVE WE DONE A FULL
ROW ACROSS
        beq     quit2
        jmp     drawtile2               ; JUMP IF CMP >TEMP10
quit2
        rts
        ;*******************************************
        ;* FILL LEFT SIDE ON SCREEN                *
        ;* using temp1 - temp14                    *
        ;* temp 1 & 2   = screen address           *
        ;* temp 3 & 4   = colour address           *
        ;* temp 5 & 6   = Ptrmap address           *
        ;* temp 7 & 8   = Ptrmap colour address    *
        ;* temp 9       = Ascii Value From Map     *
        ;* temp 10      = Colour Value  From Map   *
        ;* temp 11      = Were line count is stored*
        ;* temp 12      = Y Value On Map           *
        ;* temp 13      = which side of tile to draw*
        ;* temp 14      = X Position On Map        *
        ;*******************************************
fillleftside
        lda     mapy
        sta     temp12                  ; set mapy
        lda     temp12                  ; get mapy value
        asl     a
        tay
        clc
        lda     Ptrmap                  ; map address
        adc     mapyaddress,y
        sta     temp5
        lda     Ptrmap+1
        adc     mapyaddress+1,y
        sta     temp6
        clc
```

```
        lda     Ptrmapcolour            ; map colour address
        adc     mapyaddress,y
        sta     temp7
        lda     Ptrmapcolour+1
        adc     mapyaddress+1,y
        sta     temp8
        lda     #0                      ; count line drawn
down
        sta     temp11                  ; SET ACROSS POSITION
ON SCREEN
        lda     #0                      ; WE ALLWAYS START ON
THE FIRST PART OF A tile
        sta     temp13
        lda     mapx                    ; set mapx to left
side of map
        sta     temp14
drawtile3
        ldy     temp14                  ; SET ACROSS ON MAP
        lda     (temp5),y               ; GET FIRST CHAR ON
TILE FROM MAP
        ;lda #0                        ; ***** DEBUG ONLY
*****
        sta     temp9                   ; STORE IT
        lda     (temp7),y               ; COLOUR MAP
        sta     temp10                  ; STORE COLOUR OF
TILE
        lda     temp13                  ; ARE WE DRAWING THE
TOP OF THE TILE OR BOTTOM
        cmp     #0                      ; ARE WE DRAWING THE
TOP PART OF THE TILE
        beq     topline1                ; YES
        inc     temp9                   ; NO THEN SET CHAR
VALUE TO SECOND LINE OF TILE
        inc     temp9
topline1
        lda     lrflag                  ; YES THEN SEE WHICH
SIDE OF TILE TO DRAW
        cmp     #1                      ; RIGHT SIDE
        beq     drawrightside1          ; NO
drawleftside1
        ldy     #0                      ; SET X POSITION ON
SCREEN
        lda     temp9                   ; GET TILE CHAR VALUE
        sta     (temp1),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp10                  ; GET COLOUR VALUE
```

```
        sta     (temp3),y               ; WRITE OUT COLOUR TO
SCREEN
        jmp     next2
drawrightside1
        ldy     #0                      ; SET X POSITION ON
SCREEN
        ldx     temp9                   ; GET TILE CHAR VALUE
        inx
        txa
        sta     (temp1),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp10                  ; GET COLOUR VALUE
        sta     (temp3),y               ; WRITE OUT COLOUR TO
SCREEN
next2
        lda     temp13                  ; ARE WE DRAWING THE
TOP OF THE TILE OR BOTTOM
        cmp     #1                      ; ARE WE DRAWING THE
BOTTOM PART OF THE TILE
        bne     continue1               ; NO
        inc     temp12                  ; move mapy value
down one line
        lda     temp12                  ; get mapy value
        asl     a
        tay
        clc
        lda     Ptrmap                  ; map address
        adc     mapyaddress,y
        sta     temp5
        lda     Ptrmap+1
        adc     mapyaddress+1,y
        sta     temp6
        clc
        lda     Ptrmapcolour            ; map colour address
        adc     mapyaddress,y
        sta     temp7
        lda     Ptrmapcolour+1
        adc     mapyaddress+1,y
        sta     temp8
continue1
        sec                             ; BOOL UDFLAG
        lda     #1
        sbc     temp13
        sta     temp13
        clc                             ; hidden screen
address
        lda     temp1
```

```
        adc     #40
        sta     temp1
        lda     temp2
        adc     #0
        sta     temp2
        clc                             ; hidden screen
colour address
        lda     temp3
        adc     #40
        sta     temp3
        lda     temp4
        adc     #0
        sta     temp4
        inc     temp11                  ; NEXT LINE DOWN
COUNT
        lda     temp11                  ; GET TILE DOWN COUNT
        cmp     #25                     ; HAVE WE DONE 12
TILES DOWN
        beq     quit3                   ; QUIT OUT
        jmp     drawtile3               ; NO THEN CONTINE
quit3
        rts
        ;*********************************************
        ;* FILL RIGHT SIDE ON SCREEN                 *
        ;* using temp1 - temp14                      *
        ;* temp 1 & 2   = screen address            *
        ;* temp 3 & 4   = colour address            *
        ;* temp 5 & 6   = Ptrmap address            *
        ;* temp 7 & 8   = Ptrmap colour address     *
        ;* temp 9       = Ascii Value From Map       *
        ;* temp 10      = Colour Value  From Map     *
        ;* temp 11      = Were line count is stored  *
        ;* temp 12      = Y Value On Map             *
        ;* temp 13      = which side of tile to draw *
        ;* temp 14      = X Position On Map          *
        ;*********************************************
fillrightside
        lda     mapy
        sta     temp12                  ; set mapy
        lda     #0                      ; clear memory
        sta     temp16
        sta     temp17
        lda     temp12                  ; get mapy value
        asl     a
        tay
        clc
        lda     Ptrmap                  ; map address
```

```
        adc     mapyaddress,y
        sta     temp5
        lda     Ptrmap+1
        adc     mapyaddress+1,y
        sta     temp6
        clc
        lda     Ptrmapcolour            ; map colour address
        adc     mapyaddress,y
        sta     temp7
        lda     Ptrmapcolour+1
        adc     mapyaddress+1,y
        sta     temp8
        lda     #0                      ; count line drawn
down
        sta     temp11                  ; SET ACROSS POSITION
ON SCREEN
        lda     #0                      ; WE ALLWAYS START ON
THE FIRST PART OF A TILE
        sta     temp13
        clc                             ; set mapx to right
side of map
        lda     mapx
        adc     #19                     ; NOT SURE WHAT
NUMBER
        sta     temp14
drawtile4
        ldy     temp14                  ; SET ACROSS ON MAP
        lda     (temp5),y               ; GET FIRST CHAR ON
TILE FROM MAP
        ;lda #0                        ; ***** DEBUG ONLY
*****
        sta     temp9                   ; STORE IT
        lda     (temp7),y               ; COLOUR MAP
        sta     temp10                  ; STORE COLOUR OF
TILE
        lda     temp13                  ; ARE WE DRAWING THE
TOP OF THE TILE OR BOTTOM
        cmp     #0                      ; ARE WE DRAWING THE
TOP PART OF THE TILE
        beq     topline2                ; YES
        inc     temp9                   ; NO THEN SET CHAR
VALUE TO SECOND LINE OF TILE
        inc     temp9
topline2
        lda     lrflag                  ; YES THEN SEE WHICH
SIDE OF TILE TO DRAW
        cmp     #1                      ; RIGHT SIDE
```

```
        beq     drawrightside2          ; NO
drawleftside2
        ldy     #38                     ; SET X POSITION ON
SCREEN
        lda     temp9                   ; GET TILE CHAR VALUE
        sta     (temp1),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp10                  ; GET COLOUR VALUE
        sta     (temp3),y               ; WRITE OUT COLOUR TO
SCREEN
        jmp     next3
drawrightside2
        ldy     #38                     ; SET X POSITION ON
SCREEN
        ldx     temp9                   ; GET TILE CHAR VALUE
        inx
        txa
        sta     (temp1),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp10                  ; GET COLOUR VALUE
        sta     (temp3),y               ; WRITE OUT COLOUR TO
SCREEN
next3
        lda     temp13                  ; ARE WE DRAWING THE
TOP OF THE TILE OR BOTTOM
        cmp     #1                      ; ARE WE DRAWING THE
BOTTOM PART OF THE TILE
        bne     continue2               ; NO
        inc     temp12                  ; move mapy value
down one line
        lda     temp12                  ; get mapy value
        asl     a
        tay
        clc
        lda     Ptrmap                  ; map address
        adc     mapyaddress,y
        sta     temp5
        lda     Ptrmap+1
        adc     mapyaddress+1,y
        sta     temp6
        clc
        lda     Ptrmapcolour            ; map colour address
        adc     mapyaddress,y
        sta     temp7
        lda     Ptrmapcolour+1
        adc     mapyaddress+1,y
        sta     temp8
```

```
continue2
        sec                                     ; BOOL UDFLAG
        lda     #1
        sbc     temp13
        sta     temp13
        clc                                     ; hidden screen
address
        lda     temp1
        adc     #40
        sta     temp1
        lda     temp2
        adc     #0
        sta     temp2
        clc                                     ; hidden screen
colour address
        lda     temp3
        adc     #40
        sta     temp3
        lda     temp4
        adc     #0
        sta     temp4
        inc     temp11                          ; next line down
count
        lda     temp11                          ; get tile down count
        cmp     #25                             ; have we done 12
tiles down
        beq     quit4                           ; quit out
        jmp     drawtile4                       ; no then contine
quit4
        rts
        ;****************************************
        ;* COPY SCREEN LEFT & UP                *
        ;* temp0 to temp9                       *
        ;* x = 0 copy left , 1 copy up          *
        ;* temp 0 & 1   = screen address        *
        ;* temp 2 & 3   = hidden screen address *
        ;* temp 4 & 5   = colour address        *
        ;* temp 6 & 7   = hidden colour address *
        ;* temp8        = how many items across *
        ;* temp9        = how many line         *
        ;****************************************
copyscreenlu
        ;inc $d020
        lda     Ptrscreen                       ; current screen
address
        sta     temp0
        lda     Ptrscreen+1
```

```
        sta     temp1
        lda     Ptrhiddenscreen         ; hidden screen
address
        sta     temp2
        lda     Ptrhiddenscreen+1
        sta     temp3
        lda     Ptrcolour               ; screen colour
        sta     temp4
        lda     Ptrcolour+1
        sta     temp5
        lda     PtrSparecolour          ; spare colour
        sta     temp6
        lda     PtrSparecolour+1
        sta     temp7
        lda     #39                     ; 40 39set how many
across to maximum
        sta     temp8
        lda     #25                     ; set how many row
down to maximum
        sta     temp9
                                        ; which way are we
scrolling it
        cpx     #0                      ; are we scrolling
left
        bne     up                     ; no
        sec                             ; yes
        lda     temp8                   ; how many char
across to copy
        sbc     #1                      ; minus offset
        sta     temp8                   ; store it
        clc
        lda     temp0                   ; position screen
pointer to copy from
        adc     #1
        sta     temp0
        lda     temp1
        adc     #0
        sta     temp1
        clc
        lda     temp4                   ; position screen
colour to copy from
        adc     #1
        sta     temp4
        lda     temp5
        adc     #0
        sta     temp5
        jmp     continue3
```

```
up
        sec                             ; yes
        lda     temp9                   ; how many char
across to copy
        sbc     #1                      ; minus offset
        sta     temp9                   ; store it
        clc
        lda     temp0                   ; position screen
pointer to copy from
        adc     #40
        sta     temp0
        lda     temp1
        adc     #0
        sta     temp1
        clc
        lda     temp4                   ; position screen
colour to copy from
        adc     #40
        sta     temp4
        lda     temp5
        adc     #0
        sta     temp5
continue3
        ldy     #0                      ; how many char
across thew line have we copyed
        ldx     #0                      ; how many lines down
have we done
continue4
        lda     (temp0),y               ; load byte from
screen you can see
        sta     (temp2),y               ; copy it to the
screen you can not see
        lda     (temp4),y               ; load byte from
colour screen you can see
        sta     (temp6),y               ; copy it to the
colour screen you can not see
        iny
        cpy     temp8                   ; have we done 40
char across
        bne     continue4               ; no
        inx                             ; yes then inc row
counter
        ldy     #0                      ; set y to first char
on line
        clc                             ; current screen
address
        lda     temp0
```

```
        adc       #40                       ; add 40 to address
to get to next line
        sta       temp0
        lda       temp1
        adc       #0
        sta       temp1
        clc                                 ; hidden screen
address
        lda       temp2
        adc       #40                       ; add 40 to address
to get to next line
        sta       temp2
        lda       temp3
        adc       #0
        sta       temp3
        clc                                 ; screen colour
        lda       temp4
        adc       #40                       ; add 40 to address
to get to next line
        sta       temp4
        lda       temp5
        adc       #0
        sta       temp5
        clc                                 ; spare colour
        lda       temp6
        adc       #40                       ; add 40 to address
to get to next line
        sta       temp6
        lda       temp7
        adc       #0
        sta       temp7
        cpx       temp9                     ; have we done 24
rows yes
        bne       continue4                 ; no
quit5
        ;dec $d020
        rts                                 ; yes then quit
        ;*************************************
        ;* COPY SCREEN RIGHT ONE OR DOWN ONE    *
        ;* temp0 - temp9                       *
        ;* x = 0 copy right , 1 copy down      *
        ;* temp 0 & 1   = hidden screen address *
        ;* temp 2 & 3   = screen address       *
        ;* temp 4 & 5   = colour address       *
        ;* temp 6 & 7   = hidden colour address *
        ;* temp8        = how many items across *
        ;* temp9        = how many line        *
```

```
        ;**************************************
copyscreenrd
        ;inc $d020
        lda         Ptrhiddenscreen           ; hidden screen
address
        sta         temp0
        lda         Ptrhiddenscreen+1
        sta         temp1
        lda         Ptrscreen                 ; current screen
address
        sta         temp2
        lda         Ptrscreen+1
        sta         temp3
        lda         Ptrcolour                 ; screen colour
        sta         temp4
        lda         Ptrcolour+1
        sta         temp5
        lda         PtrSparecolour            ; spare colour
        sta         temp6
        lda         PtrSparecolour+1
        sta         temp7
        lda         #39                       ; set how many across
to maximum
        sta         temp8
        lda         #25                       ; set how many row up
to maximum
        sta         temp9
        txa                                   ; which way  are we
scrolling it
        cmp         #0                        ; are we scrolling
right
        bne         down                     ; no
        lda         #38                       ; set how many across
to maximum
        sta         temp8
        clc                                   ; yes
        lda         temp0                     ; position screen
pointer to copy from
        adc         #1
        sta         temp0
        lda         temp1
        adc         #0
        sta         temp1
        clc
        lda         temp6                     ; position colour
pointer to copy from
        adc         #1
```

```
        sta     temp6
        lda     temp7
        adc     #0
        sta     temp7
        jmp     continue5
down
        lda     #24                     ; set how many row up
to maximum
        sta     temp9
        clc                             ; yes
        lda     temp0                   ; position screen
pointer to copy from
        adc     #40
        sta     temp0
        lda     temp1
        adc     #0
        sta     temp1
        clc
        lda     temp6                   ; position colour
pointer to copy from
        adc     #40
        sta     temp6
        lda     temp7
        adc     #0
        sta     temp7
continue5
        ldy     #0                      ; how many char
across thew line have we copyed
        ldx     #0                      ; how many lines down
have we done
continue6
        lda     (temp2),y               ; load byte from
screen you can see
        sta     (temp0),y               ; copy it to the
screen you can not see
        lda     (temp4),y               ; load byte from
colour screen you can see
        sta     (temp6),y               ; copy it to the
colour screen you can not see
        iny                             ; next char on line
        cpy     temp8                   ; are we at the left
side of the screen
        bne     continue6               ; no
        inx                             ; yes then dec row
counter
        ldy     #0                      ; set y to first char
on line
```

```
        clc                                     ; add 40 to address
to get to next line
        lda       temp0
        adc       #40                           ; add 40 to address
to get to next line
        sta       temp0
        lda       temp1
        adc       #0
        sta       temp1
        clc                                     ; add 40 to address
to get to next line
        lda       temp2
        adc       #40                           ; add 40 to address
to get to next line
        sta       temp2
        lda       temp3
        adc       #0
        sta       temp3
        clc                                     ; add 40 to address
to get to next line
        lda       temp4
        adc       #40                           ; add 40 to address
to get to next line
        sta       temp4
        lda       temp5
        adc       #0
        sta       temp5
        clc                                     ; add 40 to address
to get to next line
        lda       temp6
        adc       #40                           ; add 40 to address
to get to next line
        sta       temp6
        lda       temp7
        adc       #0
        sta       temp7
        cpx       temp9                         ; have we done 24
rows yes
        bne       continue6                     ; no
        ;dec $d020
        rts                                     ; yes then quit
        ;************************************** HAS THE MAP IS
19 TILES ACROSS PLUS HALF A TILE
        ;* DRAW MAP ON SCREEN WITH COLOUR       * AND 12 TILES
DOWN PLUS HALF A TILE
        ;* temp 0 & 1   = screen address        * I DID NOT WANT
TO MESS A ROUND WITH DRAWING HALF A TILE
```

```
        ;* temp 2 & 3   = colour address        * WHEN I COULD
CALL A SOME CODE I HAVE ALREADY
        ;* temp 4 & 5   = map address           * WROTE TO DRAW
ONE SIDE OF A TILE
        ;* temp 6 & 7   = map colour address    *
        ;* temp 8       = tiles y screen        *
        ;* temp 9       = tiles down screen     *
        ;* temp10       = tile value            *
        ;* temp11       = tile colour           *
        ;* temp12       = ymap                  *
        ;****************************************
drawscreen
        lda       mapy                      ; GET CURRENT MAPY
VALUE
        asl       a
        tay
        clc
        lda       Ptrmap                    ; MAP ADDRESS
        adc       mapyaddress,y
        sta       temp4
        lda       Ptrmap+1
        adc       mapyaddress+1,y
        sta       temp5
        clc
        lda       Ptrmapcolour              ; COLOUR MAP ADDRESS
        adc       mapyaddress,y
        sta       temp6
        lda       Ptrmapcolour+1
        adc       mapyaddress+1,y
        sta       temp7
        lda       #0
        sta       temp8                     ; TILES DRAWN ACROSS
        sta       temp9                     ; TILES DRAWN DOWN
        sta       temp10                    ; TILE VALUE
        sta       temp11                    ; COLOUR VALUE
        sta       temp12                    ; Y FOR MAP
        tay                                 ; CLEAR
getdata
        ldy       temp12
        lda       (temp4),y                 ; GET FIRST CHAR ON
TILE FROM MAP
        sta       temp10                    ; STORE IT
        lda       (temp6),y                 ; GET COLOUR FROM
COLOUR MAP
        sta       temp11                    ; STORE IT
drawstart
        ldy       temp8                     ; POSITION ON LINE
```

```
        lda     temp10                  ; USE VALUE STORED
TEMP
        sta     (temp0),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp11                  ; USE VALUE STORED
TEMP
        sta     (temp2),y               ; WRITE OUT COLOUR TO
SCREEN
        iny                             ; MOVE RIGHT
        inc     temp10                  ; NEXT ASCII VALUE
        lda     temp10                  ; USE VALUE STORED
TEMP
        sta     (temp0),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp11                  ; USE VALUE STORED
TEMP
        sta     (temp2),y               ; WRITE OUT COLOUR TO
SCREEN
        tya                             ; SET Y TO NEXT LINE
POSITION
        clc
        adc     #39
        tay
        inc     temp10                  ; NEXT ASCII VALUE
        lda     temp10                  ; USE VALUE STORED
TEMP
        sta     (temp0),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp11                  ; USE VALUE STORED
TEMP
        sta     (temp2),y               ; WRITE OUT COLOUR TO
SCREEN
        iny                             ; MOVE RIGHT
        inc     temp10                  ; NEXT ASCII VALUE
        lda     temp10                  ; USE VALUE STORED
TEMP
        sta     (temp0),y               ; WRITE OUT CHAR TO
SCREEN
        lda     temp11                  ; USE VALUE STORED
TEMP
        sta     (temp2),y               ; WRITE OUT COLOUR TO
SCREEN
        tya                             ; SET Y BACK TO LAST
LINE
        clc
        sbc     #39
        tay
```

```
        inc     temp12
        inc     temp8                   ; MOVE TO NEXT TILE
        inc     temp8                   ; MOVE TO NEXT TILE
        lda     temp8                   ; CHECK WHAT Y IS
        cmp     #38                     ; IT Y SET TO THE
LAST TILE ACROSS SCREEN
        bne     getdata                 ; NO
        lda     #0                      ; YES
        sta     temp8                   ; CLEAR VALUES
        sta     temp12
        inc     temp9                   ; SET TO NEXT LINE
        lda     temp9                   ; GET LINES DOWN
NUMNBER
        asl     a                       ; TIMES IT BY 2
        tay                             ; COPY IT TO Y
        clc
        lda     Ptrmap                  ; MAP ADDRESS
        adc     mapyaddress,y           ; PLUS NEXT LINE
WIDTH
        sta     temp4
        lda     Ptrmap+1
        adc     mapyaddress+1,y         ; PLUS NEXT LINE
WIDTH
        sta     temp5
        clc
        lda     Ptrmapcolour            ; MAP COLOUR ADDRESS
        adc     mapyaddress,y           ; PLUS NEXT LINE
WIDTH
        sta     temp6
        lda     Ptrmapcolour+1
        adc     mapyaddress+1,y         ; PLUS NEXT LINE
WIDTH
        sta     temp7
        clc
        lda     temp0                   ; SCREEN ADDRESS
        adc     #80
        sta     temp0
        lda     temp1
        adc     #0
        sta     temp1
        clc
        lda     temp2                   ; COLOUR ADDRESS
        adc     #80
        sta     temp2
        lda     temp3
        adc     #0
        sta     temp3
```

```
        lda     temp9                   ; GET HOW MANY LINES
DOWN WE HAVE DONE
        cmp     #12                     ; HAVE WE DONE ALL
THE LINE DOWN WE CAN
        beq     quit6                   ; YES
        jmp     getdata                 ; NO
quit6
        rts
        ;****************************************
        ;* FILL SCREEN WITH COLOUR              *
        ;* temp0 - temp5                        *
        ;* x = what colour to use               *
        ;* y = what char to use                 *
        ;* temp 0 & 1   = Screen Address        *
        ;* temp 2 & 3   = Colour Address        *
        ;* temp 4       = Colour To Use         *
        ;* temp 5       = Ascii To Use          *
        ;****************************************
fillscreen
        stx     temp4                   ; what colour to use
        sty     temp5                   ; what char to fill
screen with
        ldx     #0                      ; clear how many
times to change high byte pointer
        ldy     #0                      ; clear how many
bytes we have written
conloop
        lda     temp5                   ; do char
        sta     (temp0),y               ; poke char value to
screen memory
        lda     temp4                   ; do colour
        sta     (temp2),y               ; poke colour value
to colour memory
        iny                             ; increase byte
counter
        cpx     #3                      ; are we on the last
block of byte to write
        beq     less                    ; yes then only 231
bytes are needed to be written
        cpy     #0                      ; if zero then y
counter has rap around from 255
                                        ; so we have done 256
bytes
        bne     conloop                 ; no
        jmp     contine
less
```

```
        cpy       #232                    ; have we done 231
bytes
        bne       conloop                 ; no
        rts
contine
        inx                               ; yes so increase
highe byte counter
        inc       temp1                   ; increase high byte
of memeory address
        inc       temp3                   ; increase high byte
of memeory address
        cpx       #4                      ; if x = 4 then we
have done 4 blocks of bytes
                                          ; 3*(0 to 255=256)
and 1 (0 to 231=232)
        bne       conloop                 ; no
        rts                               ; yes then we are
done
        ;*********************************
        ;* copy colour to on screen bitmap *
        ;*********************************
copycolour
        ldy       #0
loop2
        lda       sparecolour+(520),y          ; copy memory
        sta       55296+(520),y
        lda       sparecolour+(560),y          ; copy memory
        sta       55296+(560),y
        lda       sparecolour+(600),y          ; copy memory
        sta       55296+(600),y
        lda       sparecolour+(640),y          ; copy memory
        sta       55296+(640),y
        lda       sparecolour+(680),y          ; copy memory
        sta       55296+(680),y
        lda       sparecolour+(720),y          ; copy memory
        sta       55296+(720),y
        lda       sparecolour+(760),y          ; copy memory
        sta       55296+(760),y
        lda       sparecolour+(800),y          ; copy memory
        sta       55296+(800),y
        lda       sparecolour+(840),y          ; copy memory
        sta       55296+(840),y
        lda       sparecolour+(880),y          ; copy memory
        sta       55296+(880),y
        lda       sparecolour+(920),y          ; copy memory
        sta       55296+(920),y
        lda       sparecolour+(960),y          ; copy memory
```

```
        sta     55296+(960),y
        lda     sparecolour+(0),y          ; copy memory
        sta     55296+(0),y
        lda     sparecolour+(40),y          ; copy memory
        sta     55296+(40),y
        lda     sparecolour+(80),y          ; copy memory
        sta     55296+(80),y
        lda     sparecolour+(120),y          ; copy memory
        sta     55296+(120),y
        lda     sparecolour+(160),y          ; copy memory
        sta     55296+(160),y
        lda     sparecolour+(200),y          ; copy memory
        sta     55296+(200),y
        lda     sparecolour+(240),y          ; copy memory
        sta     55296+(240),y
        lda     sparecolour+(280),y          ; copy memory
        sta     55296+(280),y
        lda     sparecolour+(320),y          ; copy memory
        sta     55296+(320),y
        lda     sparecolour+(360),y          ; copy memory
        sta     55296+(360),y
        lda     sparecolour+(400),y          ; copy memory
        sta     55296+(400),y
        lda     sparecolour+(440),y          ; copy memory
        sta     55296+(440),y
        lda     sparecolour+(480),y          ; copy memory
        sta     55296+(480),y
        iny                                 ; move to next line
        cpy     #39                         ; 40 across count
        beq     quit7
        jmp     loop2
quit7
        rts

setup
        lda #<55296                     ; store colour map
address
        sta Ptrcolour
        lda #>55296
        sta Ptrcolour+1

        lda #<sparecolour               ; store spare colour map
address
        sta PtrSparecolour
        lda #>sparecolour
        sta PtrSparecolour+1
```

```
        lda #<level                     ; store map address
        sta Ptrmap
        lda #>level
        sta Ptrmap+1

        lda #<levelcolour+0            ; store colour map address
        sta Ptrmapcolour
        lda #>levelcolour+0
        sta Ptrmapcolour+1

        lda #255                        ; turn cursor off
        sta 204
        lda #1
        sta 649                         ; POKE 649,1 disable
keyboard buffering
        lda $D018                       ; set the computer to
were the new chars set are and use them
        and #240                        ; 11110000
        ora #12                         ; 00001100
        sta $D018                       ; set it at bank 1

        ;+-------+------+-------+----------+--------------------
-----------------+
        ;| VALUE | BITS |  BANK | STARTING |  VIC-II CHIP RANGE
|
        ;|  OF A |      |       | LOCATION |
|
        ;+-------+------+-------+----------+--------------------
-----------------+
        ;|   0   |  00  |   3   |   49152  | ($C000-$FFFF)*
|
        ;|   1   |  01  |   2   |   32768  | ($8000-$BFFF)
|
        ;|   2   |  10  |   1   |   16384  | ($4000-$7FFF)*
|
        ;|   3   |  11  |   0   |       0  | ($0000-$3FFF)
(DEFAULT VALUE)       |
        ;+-------+------+-------+----------+--------------------
-----------------+

        LDA $01                         ; switch off basic
        AND #$FE
        STA $01

        lda #7
        sta $D020                       ; border colour
        lda #0
```

```
        sta $D021                       ; screen background
colour

        lda #11                         ; Brown
        sta $D022                       ; background colour 1

        lda #15                         ; Lt Red
        sta $D023                       ; background colour 1

        ; D011 VIC Control Register
        ; 7     Raster Compare: (Bit 8) See 53266
        ; 6     Extended Color Text Mode 1 = Enable
        ; 5     Bit Map Mode. 1 = Enable
        ; 4     Blank Screen to Border Color: 0 = Blank
        ; 3     Select 24/25 Row Text Display: 1 = 25 Rows
        ; 2     Smooth Scroll to Y Dot-Position (0-7)
        ; 1     Smooth Scroll to Y Dot-Position (0-7)
        ; 0     Smooth Scroll to Y Dot-Position (0-7)

        lda #%00010111                  ; Select 24/25 Row Text
Display: 1 = 25 Rows
        sta $d011
        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora yscroll                     ; Smooth Scroll to Y
Dot-Position (0-7)
        sta $d011


        ; bit 0 1 2 of $d016 scroll screen left or right
        ; 0 = all the way to the right
        ; 7 = all the way to the left
        ; 3 = middle
        ; bit 4 of $d016 Select 38/40 Column Text Display: 1 =
40 Cols
        ; bit 5 of $d016 switch on mult colour

        ;+----------+-----------------------------------------
--------+
        ;| Bits 7-6 |    Unused
|
        ;| Bit  5   |    Reset-Bit: 1 = Stop VIC (no Video Out,
no RAM  |
        ;|          |    refresh, no bus access)
|
```

```
        ;| Bit  4   |    Multi-Color Mode: 1 = Enable (Text or
Bitmap)  |
        ;| Bit  3   |    Select 38/40 Column Text Display: 1 =
40 Cols  |
        ;| Bits 2-0 |    Smooth Scroll to X Dot-Position (0-7)
|
        ;+----------+-----------------------------------------
--------+

        lda #%00010111                  ; Select 38/40 Column
Text Display: 1 = 40 Cols
        sta $d016
        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora xscroll                     ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d016
        rts

        ;******************
        ;* set up the irq *
        ;******************
setupirq
        SEI
        LDA #$01
        STA $D01A                       ; VIC Interrupt Mask
Register (IMR)
        LDA #<vblank
        LDX #>vblank
        STA $0314                       ; irq address
        STX $0315                       ; irq address
        LDY #raster                     ; 251 raster position
        STY $D012                       ; Raster Position
        LDA #$7F
        STA $DC0D                       ; CIA Interrupt Control
Register
        LDA $DC0D                       ; CIA Interrupt Control
Register
        CLI
        rts


        ;************************************************
        ;* SWAP THE HIDDEN SCREEN FOR THE CURRENT SCREEN *
        ;************************************************
swapscreen
```

```
        lda whichscreen                 ; which screen is
beening shown
        cmp #0                          ; screen address 3072 is
not beening shown
        bne buf1
        lda $D018                       ; current screen
        and #%00001111
        ora #16                 ; set current screen that you can
see to 1024
        sta $D018
        lda #3072/256                   ; not need on pcKERNAL'S
screen editor
        sta 648
        lda #<1024                      ; set address of screen
you can see
        sta Ptrscreen                   ; set current screen
bitmap
        lda #>1024
        sta Ptrscreen+1                 ; set current screen
bitmap
        lda #<3072                      ; set address of screen
that is hidden
        sta Ptrhiddenscreen             ; set hidden screen
bitmap
        lda #>3072
        sta Ptrhiddenscreen+1           ; set hidden screen
bitmap
        lda #1
        sta whichscreen
        ;inc $d020
        rts
buf1
        lda $D018                       ; set default screeh
        and #%00001111
        ora #48                         ; set current screen
that you can see to 3072
        sta $D018
        lda #1024/256                   ; not need on pc
KERNAL'S screen editor
        sta 648
        lda #<3072                      ; set address of screen
you can see
        sta Ptrscreen                   ; set current screen
bitmap
        lda #>3072
        sta Ptrscreen+1                 ; set current screen
bitmap
```

```
        lda #<1024                      ; set address of screen
that is hidden
        sta Ptrhiddenscreen             ; set hidden screen
bitmap
        lda #>1024
        sta Ptrhiddenscreen+1           ; set hidden screen
bitmap
        lda #0
        sta whichscreen
        ;inc $d020
swapquit
        rts

        ;**************
        ;* SET CURSOR *
        ;**************
setcursor
        clc
        ldy xcursor                     ; across horizontal
column number in the .Y register
        ldx ycursor                     ; down the vertical row
number in the .X register
        jsr 65520
        rts
        ;**************
        ;* GET CURSOR *
        ;**************
getcursor
        sec
        jsr 65520
        sty xcursor                     ; across horizontal
column number in the .Y register
        stx ycursor                     ; down the vertical row
number in the .X register
        rts

xcursor
        byte 8                         ; cursor position were
any printing will be done on screen
ycursor
        byte 2                         ; ditto

        ;***************************************
        ;* PRINT A 16 BIT NUMBER TO THE SCREEN *
        ;* X = low byte = temp0                *
        ;* Y = high byte =temp1                *
        ;***************************************
```

```
printnum
        stx temp20
        sty temp21
        jsr clearbuffer
        ldy #5                          ; were in buffer to
store number image
LOOP3
        lda #00                         ; **** DO 16 bit divide
****
        ldx #16                         ; 16-bit number (in
temp16..temp16+1 count
                                        ; how many number we
have done
loop4
        asl temp20                      ; shift one bit position
towards the "left"
                                        ; Shift least
significant byte
        rol temp21                      ; Shift next-to-least-
significant byte with carry
        rol                             ; Shift next-to-least-
significant byte with carry
        cmp #10                         ; 8-bit number must be
10 to show a 16 bit number
        bcc loop5                       ; 10>a
        sbc #10                         ; 8-bit number must be
10 to show a 16 bit number
        inc temp20
loop5
        dex                             ;
        bne loop4                       ; IF NOT ZERO **** STOP
16 bit divide ****

        clc                             ; move left one position
for next number to be save
        adc #48                         ; 0 plus 48 = zero to
nine ancii number
        sta stringbuffer,y              ; store it
        dey                             ; next memory address in
buffer
        cpy #0
        bne LOOP3                       ; no more number to
convert

                                        ; from here the number
is in the stringbuffer
```

```
                                        ; go past any leading
zeros in number buffer
        ldy #1                          ; first number position
in buffer
donext
        lda stringbuffer,y              ; get number
        cmp #48                         ; look for zero
        bne print                      ; yes
        iny                             ; move to next number
        cpy #5                          ; are we on the last
number position
        bne donext                     ; jump out and print what
ever is there

print
        lda #28                         ; text colour red
        jsr $ffd2

getnextchar
        lda stringbuffer,y              ; address of string
        jsr $ffd2                       ; call CHROUT
        iny                             ; move to next letter
        cpy #6                          ; 5 numbers in 16 bit
address last letter to print
        bne getnextchar                ;
        rts

stringbuffer
        byte 0,0,0,0,0,0                ; maximum 65535


;*************************************************************
        ; SET THE PRINTNUM TO MOVE TO THE NEXT LINE AFTER
PRINTING *

;*************************************************************
carryagereturn
        lda #13
        jsr $ffd2
        lda #10
        jsr $ffd2
        lda #0
        jsr $ffd2
        rts
clearbuffer
        ldy #0
clearbuffer0
```

```
        lda #48                         ; this clear the buffer
we use to print a 16 bit number
        sta stringbuffer,y
        iny
        cpy #5
        bne clearbuffer0
        rts

        ;*************************************
        ;* MULTIPLIY                         *
        ;* temp0 - temp7                     *
        ;* temp0 = low byte of number        *
        ;* temp1 = high byte of number       *
        ;* temp2 = low byte of multiplicand  *
        ;* temp3 = high byte of multiplicand *
        ;* temp4 = low byte of result        *
        ;* temp5 = high byte of result       *
        ;*************************************
mult16
        lda #0                          ; product
        sta temp4
        lda #0
        sta temp5
        lda #$00
        sta temp6                       ; clear upper bits of
product
        sta temp7
        ldx #$10                        ; set binary count to 16
shift_r
        lsr temp1                       ; divide multiplier by 2
        ror temp0
        bcc rotate_r
        lda temp6                       ; get upper half of
product and add multiplicand
        clc
        adc temp2
        sta temp6
        lda temp7
        adc temp3
rotate_r
        ror                             ; rotate partial product
        sta temp7
        ror temp6
        ror temp5
        ror temp4
        dex
        bne shift_r
```

```
        rts

        ;*********************
        ;* DO SCREEN SCROLL *
        ;*********************
vblank
        lda #1
        sta sync

        lda scrollstop                  ; FLAG FOR WAITING FOR
JOYSTICK TO SET XSCROLL FOR SCROLLING
                                        ; THE SCREEN
        cmp #0                          ; STILL WAITING FOR
JOYSTICK
        beq vblankquit
        lda scrollstop
        cmp #2
        beq updownscroll                ; DO UP OR DOWN SCREEN
SCROLL
        jmp leftrightscroll             ; DO LEFT OR RIGHT
SCREEN SCROLL

vblankquit
        lda #$ff                        ; QUIT OUT AND WAIT
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31                       ; quit out


;*********************************************************
        ;** SCROLL THE SCREEN UP OR DOWN USING YSCROLL AND MAPY
**

;*********************************************************

updownscroll
        lda yscroll                     ; CURRENT YSCROLL VALUE
        ;dec $d020
ck7_1
        cmp #7                          ; DOING NOUT
        bne ck6                        ; NO MATCH SO CHECK NEXT
VALUE

        lda #1                          ; SET NEW FLAG VALUE
        sta yscroll                     ; SET YSCROLL FOR NEXT
IRQ CALL
```

```
        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora #7                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d011

        lda udflag                      ; WE SCROLL EACH WAY TWO
TIMES
        cmp #1
        bne not7next1
        LDA #<IRQUPDOWN2                ; COPY COLOURS TO
CURRENT SCREEN FROM HIDDEN COLOUR MAP
        LDX #>IRQUPDOWN2
        STA $0314                       ; irq address
        STX $0315                       ; irq address
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

not7next1
        LDA #<IRQUPDOWN                 ; COPY COLOURS TO
CURRENT SCREEN FROM HIDDEN COLOUR MAP
        LDX #>IRQUPDOWN
        STA $0314                       ; irq address
        STX $0315                       ; irq address
ck7quit1
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck6
        cmp #6                          ; CARRY ON TO FOUR
        bne ck5                        ; NO MATCH SO CHECK NEXT
VALUE
        lda #4                          ; SET NEW FLAG VALUE
        sta yscroll                     ; SET YSCROLL FOR NEXT
IRQ CALL

ck6quit1
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31
```

```
ck5
        cmp #5                          ; CONTINUE SCROLL UP
        bne ck4                        ; NO MATCH SO CHECK NEXT
VALUE

        lda Ptrhiddenscreen             ; hidden screen address
        sta temp1
        lda Ptrhiddenscreen+1
        sta temp2
        lda PtrSparecolour              ; colour address
        sta temp3
        lda PtrSparecolour+1
        sta temp4
        jsr filltop                     ; CALL FILLTOP

        lda #7                          ; SET SCREEN SCROLL
POSITION
        sta yscroll                     ; SAVE NEW SCROLL
POSITION

        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora #5                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d011

ck5quit1
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck4
        cmp #4                          ; START SCROLL DOWN
        bne ck3                        ; NO MATCH SO CHECK NEXT
VALUE

        inc udflag                      ; WE SCROLL EACH WAY TWO
TIMES
        lda udflag
        cmp #1                          ; DO WE NEED TO SCROLL
AGAIN
        beq ckcontinuey4b              ; NO
        cmp #2                          ; DO WE NEED TO SCROLL
AGAIN
        beq ckcontinuey4a              ; NO
```

```
        lda #0                          ; STOP SCROLLING
        sta scrollstop                  ; SET IRQ TO STOP UNTIL
JOYTSTICK IS MOVED AGAIN
        sta udflag
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ckcontinuey4a
        lda mapy                        ; GET VALUE OF MAP
POINTER
        cmp #maxheight                  ; MAKE SURE WE CAN STILL
MOVE DOWN
        beq ckcontinuey4b               ; NO
        inc mapy                        ; MOVE MAP POINTER DOWN
ONE LINE
ckcontinuey4b
        ldx #1                          ; COPY SCREEN UP ONE
POSITION
        jsr copyscreenlu                ; CALL copyscreenlu

        lda #2                          ; SET NEW FLAG VALUE
        sta yscroll                     ; SET YSCROLL FOR NEXT
IRQ CALL

        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora #4                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d011

ck4quit1
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck3
        cmp #3                          ; START SCROLL UP
        bne ck2                         ; NO MATCH SO CHECK NEXT
VALUE

        inc udflag
```

```
        lda udflag                      ; WE SCROLL EACH WAY TWO
TIMES
        cmp #1                          ; DO WE NEED TO SCROLL
AGAIN
        beq ckcontinuey3a               ; NO
        cmp #2                          ; DO WE NEED TO SCROLL
AGAIN
        beq ckcontinuey3b               ; NO
        lda #0                          ; STOP SCROLLIMG
        sta scrollstop                  ; SET IRQ TO STOP UNTIL
JOYTSTICK IS MOVED AGAIN
        sta udflag
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ckcontinuey3a
        lda mapy                        ; GET VALUE OF MAP
POINTER
        cmp #0                          ; MAKE SURE WE CAN STILL
MOVE DOWN
        beq ckcontinuey3b               ; NO
        dec mapy                        ; MOVE MAP POINTER DOWN
ONE LINE
ckcontinuey3b
        ldx #1                          ; SET COPY SCREEN DOWN
        jsr copyscreenrd                ; CALL COPYSCREENRD

        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora #3                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d011

        lda #5                          ; NEW VALUE FOR YSCROLL
        sta yscroll                     ; SET NEW YSCROLL VALUE

ck3quit1
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck2
        cmp #2                          ; CONTINUE SCROLL DOWN
```

```
        bne ck1                         ; NO MATCH SO CHECK NEXT
VALUE

                                        ; SET UP POINTER FOR
BOTTOM ROW
        clc                             ; POSITION ROW BOTTOM OF
SCREEN
        lda Ptrhiddenscreen             ; hidden screen address
        adc #<960
        sta temp1
        lda Ptrhiddenscreen+1
        adc #>960
        sta temp2

        clc
        lda PtrSparecolour              ; colour address
        adc #<960
        sta temp3
        lda PtrSparecolour+1
        adc #>960
        sta temp4
        jsr fillbottom                  ; CALL FILLTOPBOTTOM

        lda #0                          ; SET SCREEN SCROLL
POSITION FLAG
        sta yscroll                     ; STORE YSCROLL POSITION

ck2quit1
        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora #2                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d011

        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck1
        cmp #1                          ; DOING NOUT
        bne ck0                         ; NO MATCH SO CHECK NEXT
VALUE
        lda #3                          ; SET NEW FLAG VALUE
        sta yscroll                     ; SET YSCROLL FOR NEXT
IRQ CALL
```

```
ck1quit1
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck0
        cmp #0                          ; DOING NOUT
        bne ckyquit                     ; NO MATCH SO GO TO
RESET

        lda #6                          ; SET NEW FLAG VALUE
        sta yscroll                     ; SET YSCROLL FOR NEXT
IRQ CALL

        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora #0                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d011

        lda udflag                      ; WE SCROLL EACH WAY TWO
TIMES
        cmp #1
        bne not0next1
        LDA #<IRQUPDOWN                 ; COPY COLOURS FROM
HIDDEN COLOUR MAP TO CURRENT COLOUR MAP
        LDX #>IRQUPDOWN
        STA $0314                       ; irq address
        STX $0315                       ; irq address
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

not0next1
        LDA #<IRQUPDOWN2                ; COPY COLOURS FROM
HIDDEN COLOUR MAP TO CURRENT COLOUR MAP
        LDX #>IRQUPDOWN2
        STA $0314                       ; irq address
        STX $0315                       ; irq address
ckyquit
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
```

```
        jmp $ea31


;************************************************************
        ;** SCROLL THE SCREEN LEFT OR RIGHT USING XSCROLL AND
MAPX **

;************************************************************
leftrightscroll
        lda xscroll                     ; CURRENT XSCROLL VALUE

ck7
        cmp #7                          ; DOING NOUT
        bne ck6_1                        ; NO MATCH SO CHECK
NEXT VALUE

        lda #1                          ; SET SCREEN SCROLL
POSITION
        sta xscroll                     ; SAVE NEW SCROLL
POSITION

        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora #7                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d016

        lda lrflag                      ; WE SCROLL EACH WAY TWO
TIMES
        cmp #1
        bne not7next2
        LDA #<irqleftright2             ; COPY COLOURS TO
CURRENT SCREEN FROM HIDDEN COLOUR MAP
        LDX #>irqleftright2
        STA $0314                       ; irq address
        STX $0315                       ; irq address
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

not7next2
        LDA #<irqleftright              ; COPY COLOURS TO
CURRENT SCREEN FROM HIDDEN COLOUR MAP
        LDX #>irqleftright
        STA $0314                       ; irq address
```

```
        STX $0315                       ; irq address
ck7quit2
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck6_1
        cmp #6                          ; DOING NOUT
        bne ck5_1                       ; NO MATCH SO CHECK NEXT
VALUE

        lda #4                          ; SET SCREEN SCROLL
POSITION
        sta xscroll                     ; SAVE NEW SCROLL
POSITION

ck6quit2
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck5_1
        cmp #5                          ; CONTINUE SCROLL LEFT
        bne ck4_1                        ; NO MATCH SO CHECK
NEXT VALUE
        lda Ptrhiddenscreen             ; hidden screen address
        sta temp1
        lda Ptrhiddenscreen+1
        sta temp2

        lda PtrSparecolour              ; colour address
        sta temp3
        lda PtrSparecolour+1
        sta temp4
        jsr fillleftside                ; CALL FILLLEFTSIDE

        lda #7                          ; SET SCREEN SCROLL
POSITION
        sta xscroll                     ; SAVE NEW SCROLL
POSITION
        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora #5                          ; Smooth Scroll to x
Dot-Position (0-7)
```

```
        sta $d016

ck5quit2
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31
ck4_1
        cmp #4                          ; START SCROLL RIGHT
        bne ck3_1                        ; NO MATCH SO CHECK
NEXT VALUE

        inc lrflag

        lda lrflag                      ; WE SCROLL EACH WAY TWO
TIMES
        cmp #1                          ; DO WE NEED TO SCROLL
AGAIN
        beq ckcontinue4b                ; NO
        cmp #2                          ; DO WE NEED TO SCROLL
AGAIN
        beq ckcontinue4a                ; NO
        lda #0                          ; STOP SCROLLIMG
        sta scrollstop                  ; SET IRQ TO STOP UNTIL
JOYTSTICK IS MOVED AGAIN
        sta lrflag
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ckcontinue4a
        lda mapx                        ; GET VALUE OF MAP
POINTER
        cmp #maxwidth                   ; MAKE SURE WE CAN STILL
MOVE RIGHT
        beq ckcontinue4b                ; NO
        inc mapx                        ; MOVE MAP POINTER RIGHT
ONE TILE

ckcontinue4b
        ldx #0                          ; COPY SCREEN LEFT ONE
POSITION
        jsr copyscreenlu                ; CALL COPYSCREENlU

        lda #2                          ; SET NEW FLAG VALUE
```

```
        sta xscroll                     ; SET XSCROLL FOR NEXT
IRQ CALL
        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora #4                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d016
ck4quit2
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck3_1
        cmp #3                          ; START SCROLL LEFT
        bne ck2_1                        ; NO MATCH SO CHECK
NEXT VALUE

        inc lrflag

        lda lrflag                      ; WE SCROLL EACH WAY TWO
TIMES
        cmp #1                          ; DO WE NEED TO SCROLL
AGAIN
        beq ckcontinuex3a              ; NO
        cmp #2                          ; DO WE NEED TO SCROLL
AGAIN
        beq ckcontinuex3b              ; NO
        lda #0                          ; STOP SCROLLIMG
        sta scrollstop                  ; SET IRQ TO STOP UNTIL
JOYTSTICK IS MOVED AGAIN
        sta lrflag
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ckcontinuex3a
        lda mapx                        ; GET VALUE OF MAP
POINTER
        cmp #0                          ; MAKE SURE WE CAN STILL
MOVE LEFT
        beq ckcontinuex3b              ; NO
        dec mapx                        ; MOVE MAP POINTER LEFT
ONE LINE
```

```
ckcontinuex3b
        ldx #0                          ; SET COPY SCREEN LEFT
        jsr copyscreenrd                ; CALL COPYSCREENRD

        lda #5                          ; NEW VALUE FOR XSCROLL
        sta xscroll                     ; SET NEW XSCROLL VALUE

        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora #3                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d016

ck3quit2
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck2_1
        cmp #2                          ; CONTINUE SCROLL RIGHT
        bne ck1_1                        ; NO MATCH SO CHECK
NEXT VALUE

        lda Ptrhiddenscreen             ; hidden screen address
        sta temp1
        lda Ptrhiddenscreen+1
        sta temp2
        lda PtrSparecolour              ; colour address
        sta temp3
        lda PtrSparecolour+1
        sta temp4
        jsr fillrightside               ; CALL FILLRIGHTSIDE

        lda #0                          ; SET SCREEN SCROLL
POSITION
        sta xscroll                     ; SAVE NEW SCROLL
POSITION

        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora #2                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d016
```

```
ck2quit2
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck1_1
        cmp #1                          ; DOING NOUT
        bne ck0_1                        ; NO MATCH SO CHECK
NEXT VALUE

        lda #3                          ; SET SCREEN SCROLL
POSITION
        sta xscroll                     ; SAVE NEW SCROLL
POSITION

ck1quit2
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

ck0_1
        cmp #0                          ; DOING NOUT
        bne ckxquit                     ; NO MATCH SO QUIT OUT

        lda #6                          ; SET SCREEN SCROLL
POSITION
        sta xscroll                     ; SAVE NEW SCROLL
POSITION
        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora #0                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d016

        lda lrflag                      ; WE SCROLL EACH WAY TWO
TIMES
        cmp #1
        bne not0next2
        LDA #<irqleftright              ; COPY COLOURS FROM
HIDDEN COLOUR MAP TO CURRENT COLOUR MAP
        LDX #>irqleftright
        STA $0314                       ; irq address
        STX $0315                       ; irq address
        lda #$ff
```

```
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

not0next2
        LDA #<irqleftright2             ; COPY COLOURS FROM
HIDDEN COLOUR MAP TO CURRENT COLOUR MAP
        LDX #>irqleftright2
        STA $0314                       ; irq address
        STX $0315                       ; irq address
ckxquit
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

IRQUPDOWN
        ;inc $d020
        ldy #0                          ; COPY COLOUR FOR SCROLL
DOWN OR UP
loop6
        lda sparecolour,y               ; LINE 0 IF GOING UP
CHANGE THIS ONE
        sta 55296,y                     ; 0
        sta 55297,y
        lda sparecolour+40,y            ; 1
        sta 55296+40,y
        sta 55296+41,y
        lda sparecolour+120,y           ; 3
        sta 55296+120,y
        sta 55296+121,y
        lda sparecolour+200,y           ; 5
        sta 55296+200,y
        sta 55296+201,y
        lda sparecolour+280,y           ; 7
        sta 55296+280,y
        sta 55296+281,y
        lda sparecolour+360,y           ; 9
        sta 55296+360,y
        sta 55296+361,y
        lda sparecolour+440,y           ; 11
        sta 55296+440,y
        sta 55296+441,y
        lda sparecolour+520,y           ; 13
        sta 55296+520,y
        sta 55296+521,y
        lda sparecolour+600,y           ; 15
```

```
        sta 55296+600,y
        sta 55296+601,y
        lda sparecolour+680,y           ; 17
        sta 55296+680,y
        sta 55296+681,y
        lda sparecolour+760,y           ; 19
        sta 55296+760,y
        sta 55296+761,y
        lda sparecolour+840,y           ; 21
        sta 55296+840,y
        sta 55296+841,y
        lda sparecolour+920,y           ; 23
        sta 55296+920,y
        sta 55296+921,y
        lda sparecolour+960,y           ; YES LINE 24 IF GOING
DOWN CHANGE THIS ONE
        sta 55296+960,y
        sta 55296+961,y
        iny
        iny                             ; move to next line
        cpy #40                         ; 40 across count
        beq irq1quit1
        jmp loop6
irq1quit1
        ;dec $d020
        jmp IRQSWAPSCREEN


;*****************************************************************
******************
IRQUPDOWN2
        ;inc $d020
        ldy #0
loop7
        lda sparecolour,y               ; 0
        sta 55296,y
        sta 55297,y
        lda sparecolour+80,y            ; 2
        sta 55296+80,y
        sta 55296+81,y
        lda sparecolour+160,y           ; 4
        sta 55296+160,y
        sta 55296+161,y
        lda sparecolour+240,y           ; 6
        sta 55296+240,y
        sta 55296+241,y
        lda sparecolour+320,y           ; 8
```

```
        sta 55296+320,y
        sta 55296+321,y
        lda sparecolour+400,y           ; 10
        sta 55296+400,y
        sta 55296+401,y
        lda sparecolour+480,y           ; 12
        sta 55296+480,y
        sta 55296+481,y
        lda sparecolour+560,y           ; 14
        sta 55296+560,y
        sta 55296+561,y
        lda sparecolour+640,y           ; 16
        sta 55296+640,y
        sta 55296+641,y
        lda sparecolour+720,y           ; 18
        sta 55296+720,y
        sta 55296+721,y
        lda sparecolour+800,y           ; 20
        sta 55296+800,y
        sta 55296+801,y
        lda sparecolour+880,y           ; 22
        sta 55296+880,y
        sta 55296+881,y
        lda sparecolour+960,y           ; 24
        sta 55296+960,y
        sta 55296+961,y
        iny                             ; move to next line
        iny
        cpy #40                         ; 40 across count
        beq irq2quit
        jmp loop7

irq2quit
IRQSWAPSCREEN
        lda whichscreen                 ; which screen is
beening shown
        cmp #0                          ; screen address 3072 is
not beening shown
        bne buf2
        lda $D018                       ; current screen
        and #%00001111
        ora #16                         ; set current screen
that you can see to 1024
        sta $D018
        lda #3072/256                   ; not need on pcKERNAL'S
screen editor
        sta 648
```

```
        lda #<1024                      ; set address of screen
you can see
        sta Ptrscreen                   ; set current screen
bitmap
        lda #>1024
        sta Ptrscreen+1                 ; set current screen
bitmap
        lda #<3072                      ; set address of screen
that is hidden
        sta Ptrhiddenscreen             ; set hidden screen
bitmap
        lda #>3072
        sta Ptrhiddenscreen+1           ; set hidden screen
bitmap
        lda #1
        sta whichscreen
        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora #3                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d011

        LDA #<vblank                    ; wait for start of the
scroll screen
        LDX #>vblank
        STA $0314                       ; irq address
        STX $0315                       ; irq address

        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

buf2
        lda $D018                       ; set default screen
        and #%00001111
        ora #48                         ; set current screen
that you can see to 3072
        sta $D018
        lda #1024/256                   ; not need on pc
KERNAL'S screen editor
        sta 648
        lda #<3072                      ; set address of screen
you can see
        sta Ptrscreen                   ; set current screen
bitmap
```

```
        lda #>3072
        sta Ptrscreen+1                 ; set current screen
bitmap
        lda #<1024                      ; set address of screen
that is hidden
        sta Ptrhiddenscreen             ; set hidden screen
bitmap
        lda #>1024
        sta Ptrhiddenscreen+1           ; set hidden screen
bitmap
        lda #0
        sta whichscreen

        lda $d011                       ; set screen scroll
position
        and #%01111000
        ora #3                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d011

        LDA #<vblank                    ; wait for start of the
scroll screen
        LDX #>vblank
        STA $0314                       ; irq address
        STX $0315                       ; irq address

        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

;****************************************************************
******************

irqleftright
        ;inc $d020
        ldy #1                          ; COPY ODD NUMBERS
loop8
        lda sparecolour+0,y             ; 0
        sta 55296,y
        lda sparecolour+40,y            ; 1
        sta 55296+40,y
        lda sparecolour+80,y            ; 2
        sta 55296+80,y
        lda sparecolour+120,y           ; 3
        sta 55296+120,y
        lda sparecolour+160,y           ; 4
```

```
sta 55296+160,y
lda sparecolour+200,y           ; 5
sta 55296+200,y
lda sparecolour+240,y           ; 6
sta 55296+240,y
lda sparecolour+280,y           ; 7
sta 55296+280,y
lda sparecolour+320,y           ; 8
sta 55296+320,y
lda sparecolour+360,y           ; 9
sta 55296+360,y
lda sparecolour+400,y           ; 10
sta 55296+400,y
lda sparecolour+440,y           ; 11
sta 55296+440,y
lda sparecolour+480,y           ; 12
sta 55296+480,y
lda sparecolour+520,y           ; 13
sta 55296+520,y
lda sparecolour+560,y           ; 14
sta 55296+560,y
lda sparecolour+600,y           ; 15
sta 55296+600,y
lda sparecolour+640,y           ; 16
sta 55296+640,y
lda sparecolour+680,y           ; 17
sta 55296+680,y
lda sparecolour+720,y           ; 18
sta 55296+720,y
lda sparecolour+760,y           ; 19
sta 55296+760,y
lda sparecolour+800,y           ; 20
sta 55296+800,y
lda sparecolour+840,y           ; 21
sta 55296+840,y
lda sparecolour+880,y           ; 22
sta 55296+880,y
lda sparecolour+920,y           ; 23
sta 55296+920,y
lda sparecolour+960,y           ; 24
sta 55296+960,y
iny
iny                             ; move to next line
;dec $d020
cpy #39                         ; 40 across count
beq irq1quit2
jmp loop8
```

```
irq1quit2
        ;dec $d020
        jmp IRQSWAPSCREEN2


;*****************************************************************
******************
irqleftright2
        ;inc $d020
        ldy #0                          ; COPY EVEN NUMBERS
loop9

        lda sparecolour+0,y             ; 0
        sta 55296,y
        lda sparecolour+40,y            ; 1
        sta 55296+40,y
        lda sparecolour+80,y            ; 2
        sta 55296+80,y
        lda sparecolour+120,y           ; 3
        sta 55296+120,y
        lda sparecolour+160,y           ; 4
        sta 55296+160,y
        lda sparecolour+200,y           ; 5
        sta 55296+200,y
        lda sparecolour+240,y           ; 6
        sta 55296+240,y
        lda sparecolour+280,y           ; 7
        sta 55296+280,y
        lda sparecolour+320,y           ; 8
        sta 55296+320,y
        lda sparecolour+360,y           ; 9
        sta 55296+360,y
        lda sparecolour+400,y           ; 10
        sta 55296+400,y
        lda sparecolour+440,y           ; 11
        sta 55296+440,y
        lda sparecolour+480,y           ; 12
        sta 55296+480,y
        lda sparecolour+520,y           ; 13
        sta 55296+520,y
        lda sparecolour+560,y           ; 14
        sta 55296+560,y
        lda sparecolour+600,y           ; 15
        sta 55296+600,y
        lda sparecolour+640,y           ; 16
        sta 55296+640,y
```

```
        lda sparecolour+680,y           ; 17
        sta 55296+680,y
        lda sparecolour+720,y           ; 18
        sta 55296+720,y
        lda sparecolour+760,y           ; 19
        sta 55296+760,y
        lda sparecolour+800,y           ; 20
        sta 55296+800,y
        lda sparecolour+840,y           ; 21
        sta 55296+840,y
        lda sparecolour+880,y           ; 22
        sta 55296+880,y
        lda sparecolour+920,y           ; 23
        sta 55296+920,y
        lda sparecolour+960,y           ; 24
        sta 55296+960,y
        ;inc $d020
        iny
        iny                             ; move to next line
        cpy #40                         ; 40 across count
        beq irq1quit3
        jmp loop9

irq1quit3
        ;dec $d020

IRQSWAPSCREEN2

        lda whichscreen                 ; which screen is
beening shown
        cmp #0                          ; screen address 3072 is
not beening shown
        bne buf3
        lda $D018                       ; current screen
        and #%00001111
        ora #16                         ; set current screen
that you can see to 1024
        sta $D018
        lda #3072/256                   ; not need on pcKERNAL'S
screen editor
        sta 648
        lda #<1024                      ; set address of screen
you can see
        sta Ptrscreen                   ; set current screen
bitmap
        lda #>1024
```

```
        sta Ptrscreen+1                 ; set current screen
bitmap
        lda #<3072                      ; set address of screen
that is hidden
        sta Ptrhiddenscreen             ; set hidden screen
bitmap
        lda #>3072
        sta Ptrhiddenscreen+1           ; set hidden screen
bitmap
        lda #1
        sta whichscreen
        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora #3                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d016
        LDA #<vblank                    ; wait for start of the
scroll screen
        LDX #>vblank
        STA $0314                       ; irq address
        STX $0315                       ; irq address
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

buf3
        lda $D018                       ; set default screeh
        and #%00001111
        ora #48                         ; set current screen
that you can see to 3072
        sta $D018
        lda #1024/256                   ; not need on pc
KERNAL'S screen editor
        sta 648
        lda #<3072                      ; set address of screen
you can see
        sta Ptrscreen                   ; set current screen
bitmap
        lda #>3072
        sta Ptrscreen+1                 ; set current screen
bitmap
        lda #<1024                      ; set address of screen
that is hidden
        sta Ptrhiddenscreen             ; set hidden screen
bitmap
```

```
        lda #>1024
        sta Ptrhiddenscreen+1           ; set hidden screen
bitmap
        lda #0
        sta whichscreen
        lda $d016                       ; set screen scroll
position
        and #%11111000
        ora #3                          ; Smooth Scroll to x
Dot-Position (0-7)
        sta $d016
        LDA #<vblank                    ; wait for start of the
scroll screen
        LDX #>vblank
        STA $0314                       ; irq address
        STX $0315                       ; irq address
        lda #$ff
        sta $D019                       ; VIC Interrupt Request
Register (IRR)
        jmp $ea31

        ;************************
        ;* RANDOM NUMBER MAKER *
        ;************************
random
        lda $DC04                       ; CIA#1  Timer A  Lo
byte
        eor $DC05                       ; CIA#1  Timer A  Hi
byte
        eor $DD04                       ; CIA#2  Timer A  Lo
byte
        adc $DD05                       ; CIA#2  Timer A  Hi
byte
        eor $DD06                       ; CIA#2  Timer B  Lo
byte
        eor $DD07                       ; CIA#2  Timer B  Hi
byte
        rts


;       level char and colour data loaded here
;       Map Size X          59
;       Map Size Y          36
;       Mult Colour Flag    1
;       Back Ground Colour  0
;       Mult Colour 1       11
;       Mult Colour 2       15
```

```
;       Tile Size           2
;       Clear Value         27   tile used to clear a space

level

        byte
031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,031,031,031,031,031,031
        byte
031,031,000,000,031,000,000,000,000,000,000,000,031,074,000,000,
000,000,000,000,000,094,098,094,031,000,082,082,082,082,031,000,
000,132,000,136,031,031,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,031,000,058,031
        byte
031,031,000,000,031,000,031,031,000,031,031,000,031,000,078,094,
000,128,000,094,128,000,094,098,031,000,082,082,082,082,031,136,
000,000,000,000,031,031,066,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,031,030,070,030,082,031
        byte
031,031,000,031,031,000,031,000,000,000,031,000,031,000,094,098,
094,000,094,098,094,000,000,094,031,000,082,082,082,082,031,128,
000,132,000,136,031,074,000,074,031,074,074,074,074,031,082,078,
086,031,136,090,136,031,030,031,030,058,031
        byte
031,000,000,000,000,000,031,000,000,000,031,000,031,000,000,094,
000,000,000,094,000,000,000,000,031,066,031,031,031,031,031,000,
000,000,000,000,031,082,000,082,031,136,136,136,136,031,082,000,
086,031,136,136,136,031,030,031,031,031,031
        byte
031,000,031,031,000,031,031,031,031,031,031,000,031,128,000,000,
000,128,000,000,000,000,000,000,031,000,000,000,000,000,000,000,
000,000,000,000,031,074,000,074,031,136,136,136,136,031,082,000,
086,031,136,136,136,031,030,031,030,058,031
        byte
031,000,031,000,000,000,031,078,000,082,031,000,031,000,094,000,
000,000,074,094,000,000,128,000,031,000,031,031,066,031,031,031,
066,031,031,000,031,082,000,082,031,136,136,136,136,031,031,066,
031,031,136,136,136,031,030,070,030,082,031
        byte
031,000,031,000,000,000,031,000,136,082,031,000,031,000,098,094,
000,000,094,098,094,000,000,000,031,000,031,128,128,128,031,128,
128,128,031,000,031,074,000,074,031,136,136,136,136,031,140,140,
140,031,000,000,000,031,030,031,030,058,031
        byte
031,000,031,000,078,128,031,136,000,082,031,000,031,078,094,098,
```

```
094,000,000,094,000,000,000,094,031,066,031,128,128,128,031,128,
128,128,031,000,031,086,078,086,031,000,000,000,062,000,140,140,
140,031,000,000,000,031,030,031,031,031,031
        byte
031,000,031,031,031,031,031,031,066,031,031,000,031,031,031,031,
031,000,000,128,000,000,094,098,094,128,031,128,128,128,031,128,
128,128,031,000,031,031,031,031,031,000,000,062,062,031,140,140,
140,031,000,000,000,031,000,031,000,058,031
        byte
031,000,000,000,000,031,031,000,000,136,031,000,031,078,000,140,
031,000,082,094,000,000,000,094,000,000,031,031,031,031,031,031,
031,031,031,000,031,082,074,082,031,000,062,062,000,031,031,031,
031,031,000,000,000,031,000,070,000,082,031
        byte
031,000,031,031,066,031,031,136,000,000,031,000,070,140,000,000,
031,000,094,098,094,000,000,000,000,000,031,078,000,000,074,000,
000,078,031,000,070,000,000,074,031,062,062,000,000,070,136,136,
136,031,000,000,000,031,000,031,000,058,031
        byte
031,000,031,058,000,058,031,000,000,000,031,000,031,000,000,140,
031,000,000,094,082,000,128,000,000,094,031,132,132,132,000,140,
140,140,031,000,031,082,074,082,031,062,000,000,000,031,136,136,
136,031,031,066,031,031,000,031,031,031,031
        byte
031,000,031,000,140,000,000,000,000,140,031,000,031,031,031,031,
031,031,000,031,000,000,000,000,094,098,031,132,132,132,000,140,
140,140,031,000,031,031,031,031,031,031,031,031,031,031,031,000,
031,031,031,000,000,000,000,000,000,000,031
        byte
031,000,031,058,000,058,031,000,000,000,031,000,000,000,000,031,
000,000,128,031,078,000,082,094,098,094,031,132,132,132,000,140,
140,140,031,000,031,058,132,132,132,000,000,000,000,000,000,000,
082,082,031,000,000,000,000,000,000,000,031
        byte
031,000,031,031,031,031,031,031,031,031,031,031,031,066,031,031,
000,031,031,031,031,031,031,031,031,031,031,031,031,066,031,031,
031,031,031,000,031,090,132,132,132,000,000,000,000,000,000,000,
000,082,031,000,000,000,000,000,000,000,031
        byte
031,000,031,000,000,140,062,000,000,098,000,000,062,128,000,031,
000,000,000,132,000,000,031,132,132,000,140,140,140,031,000,000,
000,000,000,000,031,058,132,132,132,000,000,000,000,000,000,000,
000,000,031,000,000,000,000,000,000,000,031
        byte
031,000,031,094,000,000,062,062,098,094,098,062,062,000,000,031,
031,000,094,098,094,000,031,031,031,000,031,031,031,031,031,031,
```

```
031,031,031,031,031,031,031,031,031,000,031,000,031,000,031,128,
128,128,031,000,000,000,000,000,000,000,031
        byte
031,000,031,098,094,062,062,062,000,098,000,062,062,062,000,098,
031,000,098,094,098,000,031,136,031,136,031,136,031,136,031,136,
031,136,031,074,082,082,082,074,031,136,031,136,031,136,031,128,
128,128,031,000,000,000,000,000,000,000,031
        byte
031,000,031,094,000,140,062,000,000,000,000,000,062,000,098,094,
031,000,094,098,094,000,031,136,031,136,031,136,031,136,031,136,
031,136,031,082,000,000,000,082,031,062,031,062,031,062,031,128,
128,128,031,031,031,031,031,031,031,031,031
        byte
031,000,031,140,000,000,094,000,132,000,132,000,098,128,000,098,
031,000,000,000,000,136,000,000,000,000,000,000,000,000,000,000,
000,000,031,082,000,000,000,082,031,136,031,136,031,136,031,128,
128,128,031,000,136,136,000,000,136,000,031
        byte
031,000,031,062,062,094,098,094,000,094,000,098,094,098,062,062,
031,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,031,082,000,000,000,082,031,062,031,062,031,062,031,128,
128,128,031,136,000,000,136,136,000,136,031
        byte
031,000,031,000,140,000,094,000,000,000,132,000,098,000,000,062,
031,000,098,094,098,000,031,128,031,128,031,128,031,128,031,128,
031,000,031,074,000,000,000,074,031,136,031,136,031,136,031,074,
074,074,031,000,136,000,000,136,000,000,031
        byte
031,000,031,094,000,000,062,062,132,000,000,062,062,000,132,094,
031,128,094,098,094,000,031,128,031,128,031,128,031,128,031,128,
031,066,031,031,031,066,031,031,031,062,031,062,031,062,031,031,
031,031,031,031,031,031,000,031,031,031,031
        byte
031,000,031,098,094,000,062,062,000,000,000,062,062,000,094,098,
031,000,098,094,098,000,031,031,031,031,031,031,031,031,031,031,
031,000,000,000,000,000,000,000,000,000,000,000,000,000,031,000,
000,000,000,000,070,000,000,000,000,000,031
        byte
031,000,031,031,031,031,031,031,031,066,031,031,031,031,031,031,
031,000,000,140,000,000,031,000,000,000,094,098,062,062,062,062,
000,000,000,036,099,036,000,000,031,031,031,031,031,031,031,000,
031,000,031,000,031,031,031,031,031,000,031
        byte
031,000,031,058,000,058,000,058,000,000,000,000,000,000,000,000,
031,031,031,031,031,031,031,000,000,000,098,094,000,000,062,062,
000,000,000,099,095,099,000,000,031,000,000,000,000,140,000,000,
000,094,000,000,031,058,000,058,031,000,031
```

```
        byte
031,000,031,000,000,000,000,000,000,000,000,000,000,000,000,
000,140,140,140,140,140,031,000,000,000,062,062,000,000,094,098,
000,000,000,036,099,036,000,000,070,000,031,000,031,000,031,000,
031,000,031,000,031,000,000,000,070,000,031
        byte
031,000,031,058,000,058,000,058,000,000,000,000,000,000,000,000,
000,140,140,140,140,140,031,000,000,000,062,062,062,062,098,094,
000,000,000,000,000,000,000,000,031,000,000,094,000,000,094,000,
000,128,000,000,031,058,000,058,031,000,031
        byte
031,000,031,031,031,031,031,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,031,031,128,128,128,031,031,031,031,031,031,
031,031,031,031,031,031,031,031,031,000,031,000,031,000,031,000,
031,000,031,000,031,031,031,031,031,035,031
        byte
031,000,000,000,000,000,000,000,000,000,000,000,070,000,000,000,
000,000,000,140,140,140,031,082,128,082,031,082,000,082,031,058,
000,000,000,058,031,058,000,058,031,000,000,128,000,000,000,094,
000,000,000,000,031,000,000,000,128,000,031
        byte
031,031,066,031,031,031,066,031,031,031,066,031,031,000,000,000,
000,000,000,140,140,140,031,078,128,078,031,082,000,082,031,074,
000,000,000,074,031,000,000,000,031,000,031,000,031,000,031,000,
031,000,031,000,031,000,128,000,000,000,031
        byte
031,082,128,082,031,082,128,082,031,082,128,082,031,128,128,000,
000,000,000,000,000,000,031,086,074,086,031,082,000,082,031,058,
000,000,000,058,031,058,000,058,031,000,000,000,000,000,000,140,
000,000,000,000,031,000,000,000,000,132,031
        byte
031,082,128,082,031,082,128,082,031,082,128,082,031,128,128,000,
000,000,000,000,000,000,031,031,031,031,031,031,066,031,031,031,
031,066,031,031,031,031,066,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,000,000,136,000,000,031
        byte
031,082,074,082,031,082,074,082,031,082,078,082,031,128,128,000,
000,000,000,000,000,000,070,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,070,000,000,000,000,000,031
        byte
031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,031,
031,031,031,031,031,031,031,031,031,031,031
```

```
levelcolour
        byte
6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6
        byte
6,14,14,14,6,14,14,14,14,14,14,14,6,10,14,14,14,14,14,14,14,15,1
5,15,6,14,15,15,15,15,6,14,14,10,14,10,6,6,15,15,15,15,15,15,15,
15,15,15,15,15,15,15,15,15,15,6,15,10,6
        byte
6,14,14,14,6,14,6,6,14,6,6,14,6,14,12,15,14,10,14,15,10,14,15,15
,6,14,15,15,15,15,6,10,14,14,14,14,6,6,15,6,6,6,6,6,6,6,6,6,6,
6,6,6,6,15,10,15,15,6
        byte
6,6,14,6,6,14,6,14,14,14,6,14,6,14,15,15,15,14,15,15,15,14,14,15
,6,14,15,15,15,15,6,10,14,10,14,10,6,10,15,10,6,10,10,10,10,6,15
,15,15,6,10,10,10,6,15,6,15,10,6
        byte
6,14,14,14,14,14,6,14,14,14,6,14,6,14,14,15,14,14,14,15,14,14,14
,14,6,13,6,6,6,6,6,14,14,14,14,14,6,15,15,15,6,10,10,10,10,6,15,
13,15,6,10,10,10,6,15,6,6,6,6
        byte
6,14,6,6,12,6,6,6,6,6,6,14,6,10,14,14,14,10,14,14,14,14,14,14,6,
14,14,14,14,14,14,14,14,14,14,14,6,10,15,10,6,10,10,10,10,6,15,1
3,15,6,10,10,10,6,15,6,13,10,6
        byte
6,14,6,14,10,14,6,15,14,15,6,14,6,14,15,14,14,14,10,15,14,14,10,
14,6,14,6,6,13,6,6,6,13,6,6,14,6,15,15,15,6,10,10,10,10,6,6,15,6
,6,10,10,10,6,15,10,13,15,6
        byte
6,14,6,14,10,14,6,14,10,15,6,14,6,14,15,15,14,14,15,15,15,14,14,
14,6,14,6,10,10,10,6,10,10,10,6,14,6,10,15,10,6,10,10,10,10,6,10
,10,10,6,13,13,13,6,15,6,13,10,6
        byte
6,14,6,10,10,10,6,10,14,15,6,14,6,12,15,15,15,14,14,15,14,14,14,
15,6,13,6,10,10,10,6,10,10,10,6,14,6,15,10,15,6,13,13,13,14,13,1
0,10,10,6,13,13,13,6,15,6,6,6,6
        byte
6,14,6,6,6,6,6,6,12,6,6,14,6,6,6,6,6,14,14,10,14,14,15,15,15,10,
6,10,10,10,6,10,10,10,6,14,6,6,6,6,6,13,13,14,14,6,10,10,10,6,13
,13,13,6,15,6,15,10,6
        byte
6,14,14,14,14,6,6,14,14,10,6,14,6,12,14,10,6,14,15,15,14,14,14,1
5,14,14,6,6,6,6,6,6,6,6,6,14,6,15,10,15,6,13,14,14,13,6,6,6,6,6,
13,13,13,6,15,10,15,15,6
        byte
6,14,6,6,12,6,6,10,14,14,6,14,12,10,14,14,6,14,15,15,15,14,14,14
```

```
,14,14,6,10,14,14,10,14,14,15,6,14,12,15,15,10,6,14,14,13,13,15,
10,10,10,6,13,13,13,6,15,6,15,10,6
        byte
6,14,6,10,14,10,6,14,14,14,6,14,6,14,14,10,6,14,14,15,15,14,10,1
4,14,15,6,10,10,10,14,10,10,10,6,14,6,15,10,15,6,14,13,13,13,6,1
0,10,10,6,6,15,6,6,15,6,6,6,6
        byte
6,14,6,13,10,13,14,14,14,10,6,14,6,6,6,6,6,6,14,6,14,14,14,14,15
,15,6,10,10,10,14,10,10,10,6,14,6,6,6,6,6,6,6,6,6,6,6,13,6,6,6,1
5,15,15,15,15,15,15,6
        byte
6,14,6,10,13,10,6,14,14,14,6,14,14,14,14,6,14,14,10,6,12,14,15,1
5,15,15,6,10,10,10,14,10,10,10,6,14,6,10,10,10,10,13,13,13,13,13
,13,13,15,15,6,15,15,15,15,15,15,15,6
        byte
6,14,6,6,6,6,6,6,6,6,6,6,6,12,6,6,14,6,6,6,6,6,6,6,6,6,6,6,6,6,1
3,6,6,6,6,14,6,10,10,10,10,13,13,13,13,13,13,13,13,15,6,15,15,15
,15,15,15,15,6
        byte
6,14,6,14,14,10,14,14,14,15,14,14,14,10,14,6,14,14,14,10,14,14,6
,10,10,14,10,10,10,6,14,14,14,14,14,14,6,10,10,10,10,13,13,13,13
,13,13,13,13,13,6,15,15,15,15,15,15,15,6
        byte
6,14,6,15,14,14,14,14,15,15,15,14,14,14,14,6,6,14,15,15,15,14,6,
6,6,14,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,13,6,13,6,13,6,10,10,10,6,1
5,15,15,15,15,15,15,6
        byte
6,14,6,15,15,14,14,14,14,15,14,14,14,14,14,15,6,14,15,15,15,14,6
,10,6,10,6,10,6,10,6,10,6,10,6,10,15,15,15,10,6,10,6,10,6,10,6,1
0,10,10,6,15,15,15,15,15,15,15,6
        byte
6,14,6,15,14,10,14,14,14,14,14,14,14,14,15,15,6,14,15,15,15,14,6
,10,6,10,6,10,6,10,6,10,6,10,6,15,13,13,13,15,6,14,6,14,6,14,6,1
0,10,10,6,6,6,6,6,6,6,6,6
        byte
6,14,6,10,14,14,15,14,10,14,10,14,15,10,14,15,6,14,14,14,14,10,1
4,14,14,14,14,14,14,14,14,14,14,14,14,6,15,13,13,13,15,6,10,6,10,6,
10,6,10,10,10,6,13,10,10,13,15,10,13,6
        byte
6,14,6,14,14,15,15,15,14,15,14,15,15,15,14,14,6,14,14,14,14,14,1
4,14,14,14,14,14,14,14,14,14,14,14,14,6,15,13,13,13,15,6,14,6,14,6,
14,6,10,10,10,6,10,13,13,10,10,13,10,6
        byte
6,14,6,14,10,14,15,14,14,14,10,14,15,14,14,14,6,14,15,15,15,14,6
,10,6,10,6,10,6,10,6,10,6,14,6,10,13,13,13,10,6,10,6,10,6,10,6,1
0,10,10,6,13,10,15,13,10,13,13,6
```

```
        byte
6,14,6,15,14,14,14,14,10,14,14,14,14,14,10,15,6,10,15,15,15,14,6
,10,6,10,6,10,6,10,6,10,6,15,6,6,6,15,6,6,6,14,6,14,6,14,6,6,6,6
,6,6,6,6,13,6,6,6,6
        byte
6,14,6,15,15,14,14,14,14,14,14,14,14,14,15,15,6,14,15,15,15,14,6
,6,6,6,6,6,6,6,6,6,6,14,14,13,13,13,13,13,13,13,13,13,13,6,13
,13,13,13,13,15,13,13,13,13,13,6
        byte
6,14,6,6,6,6,6,6,6,10,6,6,6,6,6,6,6,14,14,10,14,14,6,14,14,14,15
,15,14,14,14,14,14,14,14,14,15,14,13,13,6,6,6,6,6,6,6,13,6,13,6,
13,6,6,6,6,6,13,6
        byte
6,14,6,10,14,10,14,10,14,14,14,14,14,14,14,14,6,6,6,6,6,6,6,13,1
3,13,15,15,14,14,14,14,14,14,14,15,15,15,13,13,6,13,13,13,13,10,
13,13,13,15,13,13,6,10,13,10,6,13,6
        byte
6,14,6,13,13,13,13,14,13,14,14,14,14,14,14,14,13,10,10,10,10,10,
6,13,13,13,14,14,14,14,15,15,14,14,14,14,15,14,13,13,15,13,6,13,
6,13,6,13,6,13,6,13,6,13,13,13,10,13,6
        byte
6,14,6,10,14,10,14,10,13,14,14,14,14,14,14,14,13,10,10,10,10,10,
6,13,13,13,14,14,14,14,15,15,14,14,14,13,13,13,13,13,6,13,13,15,
13,13,15,13,13,10,13,13,6,10,13,10,6,13,6
        byte
6,14,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,10,10,10,6,6,6,6,
6,6,6,6,6,6,6,6,6,6,6,13,6,13,6,13,6,13,6,13,6,13,6,6,6,6,6,13,6
        byte
6,14,14,14,14,14,14,14,14,14,14,14,10,14,14,14,14,14,14,10,10,10
,6,15,10,15,6,15,13,15,6,10,13,13,13,10,6,10,13,10,6,13,13,10,13
,13,13,15,13,13,13,13,6,13,13,13,10,13,6
        byte
6,6,12,6,6,6,12,6,6,6,12,6,6,13,13,14,14,14,14,10,10,10,6,10,10,
15,6,15,13,15,6,10,13,13,13,10,6,13,13,13,6,13,6,13,6,13,6,13,6,
13,6,13,6,13,10,13,13,13,6
        byte
6,15,10,15,6,15,10,15,6,15,10,15,6,10,10,14,14,14,13,13,13,13,6,
15,10,15,6,15,13,15,6,10,13,13,13,10,6,10,13,10,6,13,13,13,13,13
,13,10,13,13,13,13,6,13,13,13,13,10,6
        byte
6,15,10,15,6,15,10,15,6,15,10,15,6,10,10,14,14,14,13,13,13,13,6,
6,6,6,6,6,10,6,6,6,6,10,6,6,6,6,10,6,6,6,6,6,6,6,6,6,6,6,6,6,1
3,13,10,13,13,6
        byte
6,15,10,15,6,15,10,15,6,15,10,15,6,10,10,14,14,14,13,13,13,13,10
,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,
13,13,13,13,13,13,13,13,10,13,13,13,13,13,6
```

```
        byte
6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6



sparecolour
        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    ;0
        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    ;1
        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    ;2
        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    ;3
        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    ;4
        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    ;5
        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    ;6
        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    ;7
        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    ;8
        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    ;9
        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    ;10
        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    ;11
        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    ;12
```

```
        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   ;13
        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   ;14
        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   ;15
        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   ;16
        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   ;17
        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   ;18
        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   ;19
        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   ;20
        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   ;21
        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   ;22
        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   ;23
        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   ;24

        ; where the begining address of the map lines is stored
mapyaddress
        word 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        word 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        word 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        word 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        word 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

        * = 12288
; $3000 12288  Address were char image are being loaded
```

```
charrom
;          byte 60,102,110,110,96,98,60,0
;          byte 24,60,102,126,102,102,102,0
;          byte 124,102,102,124,102,102,124,0
;          byte 60,102,96,96,96,102,60,0
          byte 0,0,0,0,0,0,0,0
          byte 0,0,0,0,0,0,0,0
          byte 0,0,0,0,0,0,0,0
          byte 0,0,0,0,0,0,0,0
          byte 120,108,102,102,102,108,120,0
          byte 126,96,96,120,96,96,126,0
          byte 126,96,96,120,96,96,96,0
          byte 60,102,96,110,102,102,60,0
          byte 102,102,102,126,102,102,102,0
          byte 60,24,24,24,24,24,60,0
          byte 30,12,12,12,12,108,56,0
          byte 102,108,120,112,120,108,102,0
          byte 96,96,96,96,96,96,126,0
          byte 99,119,127,107,99,99,99,0
          byte 102,118,126,126,110,102,102,0
          byte 60,102,102,102,102,102,60,0
          byte 124,102,102,124,96,96,96,0
          byte 60,102,102,102,102,60,14,0
          byte 124,102,102,124,120,108,102,0
          byte 60,102,96,60,6,102,60,0
          byte 126,24,24,24,24,24,24,0
          byte 102,102,102,102,102,102,60,0
          byte 102,102,102,102,102,60,24,0
          byte 99,99,99,107,127,119,99,0
          byte 102,102,60,24,60,102,102,0
          byte 102,102,102,60,24,24,24,0
          byte 126,6,12,24,48,96,126,0
          byte 0,0,0,0,0,0,0,0
          byte 0,0,0,0,0,0,0,0
          byte 0,0,0,0,0,0,0,0
          byte 0,0,0,0,0,0,0,0
          byte 126,127,94,55,94,42,84,0
          byte 62,94,46,92,46,84,42,0
          byte 63,95,47,93,46,85,42,0
          byte 252,254,188,110,188,212,168,0
          byte 20,20,20,65,65,20,20,20
          byte 20,20,20,65,65,20,20,20
          byte 20,20,20,65,65,20,20,20
          byte 20,20,20,65,65,20,20,20
          byte 0,0,0,0,0,0,0,0
          byte 0,0,0,0,0,0,0,0
          byte 0,0,0,0,0,0,0,0
```

```
byte 0,0,0,0,0,0,0,0
byte 170,126,126,86,126,126,126,126
byte 170,126,126,86,126,126,126,126
byte 126,126,126,126,86,126,126,85
byte 126,126,126,126,86,126,126,85
byte 0,3,6,12,24,48,96,0
byte 60,102,110,118,102,102,60,0
byte 24,24,56,24,24,24,126,0
byte 60,102,6,12,48,96,126,0
byte 60,102,6,28,6,102,60,0
byte 6,14,30,102,127,6,6,0
byte 126,96,124,6,6,102,60,0
byte 60,102,96,124,102,102,60,0
byte 126,102,12,24,24,24,24,0
byte 60,102,102,60,102,102,60,0
byte 60,102,102,62,6,102,60,0
byte 20,20,20,106,111,47,47,47
byte 20,20,20,169,249,248,248,216
byte 47,47,47,111,106,20,20,20
byte 216,248,248,249,169,20,20,20
byte 20,20,20,66,75,47,47,47
byte 20,20,84,129,225,248,248,248
byte 47,47,47,75,66,20,20,20
byte 248,248,248,225,129,20,20,20
byte 20,20,20,65,65,20,255,255
byte 20,20,20,65,65,20,255,255
byte 255,255,20,65,65,20,20,20
byte 255,255,20,65,65,20,20,20
byte 23,23,23,67,67,23,23,23
byte 212,212,212,193,193,212,212,212
byte 23,23,23,67,67,23,23,23
byte 212,212,212,193,193,212,212,212
byte 20,20,20,65,106,43,43,47
byte 20,20,20,65,169,232,232,248
byte 47,43,43,106,65,20,20,20
byte 248,232,232,169,65,20,20,20
byte 20,20,20,65,65,20,20,63
byte 20,20,20,65,65,20,52,204
byte 52,20,20,65,65,20,20,20
byte 204,52,20,65,65,20,20,20
byte 20,20,20,65,170,149,157,157
byte 20,20,20,65,169,88,216,216
byte 157,157,149,170,65,20,20,20
byte 216,216,88,169,65,20,20,20
byte 20,20,20,65,106,37,39,39
byte 20,20,20,65,169,88,216,88
byte 37,39,37,101,106,20,20,20
```

```
byte 88,216,216,89,169,20,20,20
byte 20,20,20,65,65,23,23,23
byte 20,20,20,65,65,212,212,212
byte 23,23,23,65,65,20,20,20
byte 212,212,212,65,65,20,20,20
byte 20,55,55,102,119,55,38,55
byte 20,220,220,153,221,220,152,220
byte 55,38,55,119,102,55,55,20
byte 220,152,220,221,153,220,220,20
byte 20,238,238,238,85,85,238,238
byte 20,236,236,237,85,84,236,236
byte 238,85,85,238,238,238,20,20
byte 236,84,84,237,237,236,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 20,20,20,65,65,20,20,20
byte 240,240,240,240,0,0,0,0
byte 240,240,240,240,15,15,15,15
byte 20,20,22,66,77,28,55,63
byte 20,20,170,170,77,28,220,252
byte 237,221,221,221,221,29,63,23
byte 187,183,119,119,119,116,252,212
byte 63,255,214,213,127,63,214,214
byte 20,20,212,241,113,52,216,216
byte 213,213,63,127,214,213,255,63
byte 216,216,24,73,249,248,24,24
```

```
byte 3,63,237,221,221,221,221,221
byte 212,252,187,183,119,119,119,119
byte 55,55,28,77,66,22,20,20
byte 220,220,28,77,170,170,20,20
byte 36,36,47,111,97,36,39,39
byte 252,252,91,87,253,252,91,91
byte 39,39,28,77,67,23,20,20
byte 87,87,252,253,91,87,252,252
byte 0,0,0,0,0,0,0,0
byte 0,0,0,0,0,0,0,0
byte 0,0,0,0,0,0,0,0
byte 0,0,0,0,0,0,0,0
byte 129,231,231,231,231,231,231,255
byte 153,153,153,153,153,153,195,255
byte 153,153,153,153,153,195,231,255
byte 156,156,156,148,128,136,156,255
byte 153,153,195,231,195,153,153,255
byte 153,153,153,195,231,231,231,255
byte 129,249,243,231,207,159,129,255
byte 195,207,207,207,207,207,195,255
byte 243,237,207,131,207,157,3,255
byte 195,243,243,243,243,243,195,255
byte 255,231,195,129,231,231,231,231
byte 255,239,207,128,128,207,239,255
byte 255,255,255,255,255,255,255,255
byte 231,231,231,231,255,255,231,255
byte 153,153,153,255,255,255,255,255
byte 153,153,0,153,0,153,153,255
byte 231,193,159,195,249,131,231,255
byte 157,153,243,231,207,153,185,255
byte 195,153,195,199,152,153,192,255
byte 249,243,231,255,255,255,255,255
byte 243,231,207,207,207,231,243,255
byte 207,231,243,243,243,231,207,255
byte 255,153,195,0,195,153,255,255
byte 255,231,231,129,231,231,255,255
byte 255,255,255,255,255,231,231,207
byte 255,255,255,129,255,255,255,255
byte 255,255,255,255,255,231,231,255
byte 255,252,249,243,231,207,159,255
byte 195,153,145,137,153,153,195,255
byte 231,231,199,231,231,231,129,255
byte 195,153,249,243,207,159,129,255
byte 195,153,249,227,249,153,195,255
byte 249,241,225,153,128,249,249,255
byte 129,159,131,249,249,153,195,255
byte 195,153,159,131,153,153,195,255
```

```
byte 129,153,243,231,231,231,231,255
byte 195,153,153,195,153,153,195,255
byte 195,153,153,193,249,153,195,255
byte 255,255,231,255,255,231,255,255
byte 255,255,231,255,255,231,231,207
byte 241,231,207,159,207,231,241,255
byte 255,255,129,255,129,255,255,255
byte 143,231,243,249,243,231,143,255
byte 195,153,249,243,231,255,231,255
byte 255,255,255,0,0,255,255,255
byte 247,227,193,128,128,227,193,255
byte 231,231,231,231,231,231,231,231
byte 255,255,255,0,0,255,255,255
byte 255,255,0,0,255,255,255,255
byte 255,0,0,255,255,255,255,255
byte 255,255,255,255,0,0,255,255
byte 207,207,207,207,207,207,207,207
byte 243,243,243,243,243,243,243,243
byte 255,255,255,31,15,199,231,231
byte 231,231,227,240,248,255,255,255
byte 231,231,199,15,31,255,255,255
byte 63,63,63,63,63,63,0,0
byte 63,31,143,199,227,241,248,252
byte 252,248,241,227,199,143,31,63
byte 0,0,63,63,63,63,63,63
byte 0,0,252,252,252,252,252,252
byte 255,195,129,129,129,129,195,255
byte 255,255,255,255,255,0,0,255
byte 201,128,128,128,193,227,247,255
byte 159,159,159,159,159,159,159,159
byte 255,255,255,248,240,227,231,231
byte 60,24,129,195,195,129,24,60
byte 255,195,129,153,153,129,195,255
byte 231,231,153,153,231,231,195,255
byte 249,249,249,249,249,249,249,249
byte 247,227,193,128,193,227,247,255
byte 231,231,231,0,0,231,231,231
byte 63,63,207,207,63,63,207,207
byte 231,231,231,231,231,231,231,231
byte 255,255,252,193,137,201,201,255
byte 0,128,192,224,240,248,252,254
byte 255,255,255,255,255,255,255,255
byte 15,15,15,15,15,15,15,15
byte 255,255,255,255,0,0,0,0
byte 0,255,255,255,255,255,255,255
byte 255,255,255,255,255,255,255,0
byte 63,63,63,63,63,63,63,63
```

```
          byte 51,51,204,204,51,51,204,204
          byte 252,252,252,252,252,252,252,252
          byte 255,255,255,255,51,51,204,204
          byte 0,1,3,7,15,31,63,127
          byte 252,252,252,252,252,252,252,252
          byte 231,231,231,224,224,231,231,231
          byte 255,255,255,255,240,240,240,240
          byte 231,231,231,224,224,255,255,255
          byte 255,255,255,7,7,231,231,231
          byte 255,255,255,255,255,255,0,0
          byte 255,255,255,224,224,231,231,231
          byte 231,231,231,0,0,255,255,255
          byte 255,255,255,0,0,231,231,231
          byte 231,231,231,7,7,231,231,231
          byte 63,63,63,63,63,63,63,63
          byte 31,31,31,31,31,31,31,31
          byte 248,248,248,248,248,248,248,248
          byte 0,0,255,255,255,255,255,255
          byte 0,0,0,255,255,255,255,255
          byte 255,255,255,255,255,0,0,0
          byte 252,252,252,252,252,252,0,0
          byte 255,255,255,255,15,15,15,15
          byte 240,240,240,240,255,255,255,255
          byte 231,231,231,7,7,255,255,255
          byte 15,15,15,15,255,255,255,255

charromend

*=3072
; 3072
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
```

```
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
```

```
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
        ; sprite pointer part of the screen
        byte
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32
```