```
initializing printer for ibm tabs

                initialized!
c1      c2      c3      c4      c5      c6      c7
done!
;howdy.src

        instxt  /a:zpage.src/   ;zero page equ's

objnum  equ     T7
offset  equ     T8
temp1   equ     T9
temp    equ     T10
temps   equ     T11
        public  BARF,OBJX,OBJY,HOWDY,LBASE,HBASE
        PUBLIC  COLOR,XNXTLIN,TOGGLE,STOGGLE
        PUBLIC  CHARFLG,CHARPO,CHARG
        public  PLOTPO,PLOTIT,PLOTFLG

;maxobj equ     8       ;number of sprites

$len    equ     16      ;sprite length in lines high
$len4   equ     64      ;slen *4
$len2   equ     32      ;slen *2

mski    fcb     $ff,$fc,$fc,$fc,$f3,$f0,$f0,$f0 ;0    ;used to get 'and' mask
        fcb     $f3,$f0,$f0,$f0,$f3,$f0,$f0,$f0
        fcb     $cf,$cc,$cc,$cc,$c3,$c0,$c0,$c0 ;1
        fcb     $c3,$c0,$c0,$c0,$c3,$c0,$c0,$c0
        fcb     $cf,$cc,$cc,$cc,$c3,$c0,$c0,$c0 ;2
        fcb     $c3,$c0,$c0,$c0,$c3,$c0,$c0,$c0
        fcb     $cf,$cc,$cc,$cc,$c3,$c0,$c0,$c0 ;3
        fcb     $c3,$c0,$c0,$c0,$c3,$c0,$c0,$c0
        fcb     $3f,$3c,$3c,$3c,$33,$30,$30,$30 ;4
        fcb     $33,$30,$30,$30,$33,$30,$30,$30
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;5
        fcb     $03,$00,$00,$00,$03,$00,$00,$00
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;6
        fcb     $03,$00,$00,$00,$03,$00,$00,$00
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;7
        fcb     $03,$00,$00,$00,$03,$00,$00,$00
        fcb     $3f,$3c,$3c,$3c,$33,$30,$30,$30 ;8
        fcb     $33,$30,$30,$30,$33,$30,$30,$30
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;9
        fcb     $03,$00,$00,$00,$03,$00,$00,$00
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;a
        fcb     $03,$00,$00,$00,$03,$00,$00,$00
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;b
        fcb     $03,$00,$00,$00,$03,$00,$00,$00
        fcb     $3f,$3c,$3c,$3c,$33,$30,$30,$30 ;c
        fcb     $33,$30,$30,$30,$33,$30,$30,$30
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;d
        fcb     $03,$00,$00,$00,$03,$00,$00,$00
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;e
        fcb     $03,$00,$00,$00,$03,$00,$00,$00
        fcb     $0f,$0c,$0c,$0c,$03,$00,$00,$00 ;f
        fcb     $03,$00,$00,$00,$03,$00,$00,$00

COLOR   fcb     $80,$80,$80,$80 ;,$80,$80,$80,$80      ;color mode bytes for all
;                                       ;0= green/purple    $80= red/blue

remain  fcb     00,00,00,00     ;,00,00,00,00   ;remainders for all sprites wh
```

```
oldx    fcb     00,00,00,00     ;,00,00,00,00
        fcb     00,00,00,00     ;,00,00,00,00   ;old x value by screen

oldy    fcb     00,00,00,00     ;,00,00,00,00   ;old Y value by screen
        fcb     00,00,00,00     ;,00,00,00,00


OBJX    fcb     00,00,00,00     ;,00,00,00,00

OBJY    fcb     00,00,00,00     ;,00,00,00,00


work1
        rmb     64
work2
        rmb     64
work3
        rmb     64
work4
        rmb     64
;work5
;       rmb     64
;work6
;       rmb     64
;work7
;       rmb     64
;work8
;       rmb     64

andmsk1
        rmb     64
andmsk2
        rmb     64
andmsk3
        rmb     64
andmsk4
        rmb     64
;andmsk5
;       rmb     64
;andmsk6
;       rmb     64
;andmsk7
;       rmb     64
;andmsk8
;       rmb     64


wrkl
        fcb     <work1
        fcb     <work2
        fcb     <work3
        fcb     <work4
;       fcb     <work5
;       fcb     <work6
;       fcb     <work7
;       fcb     <work8
wrkh
        fcb     >work1
        fcb     >work2
        fcb     >work3
        fcb     >work4
```

```
;       fcb     >work5
;       fcb     >work6
;       fcb     >work7
;       fcb     >work8
andl
        fcb     <andmsk1
        fcb     <andmsk2
        fcb     <andmsk3
        fcb     <andmsk4
;       fcb     <andmsk5
;       fcb     <andmsk6
;       fcb     <andmsk7
;       fcb     <andmsk8
andh
        fcb     >andmsk1
        fcb     >andmsk2
        fcb     >andmsk3
        fcb     >andmsk4
;       fcb     >andmsk5
;       fcb     >andmsk6
;       fcb     >andmsk7
;       fcb     >andmsk8

buf
b0      rmb     64      ;sprite #0 buffer for screen one
b1      rmb     64      ;sprite #0 buffer for screen two
b2      rmb     64      ;sprite #1 buffers
b3      rmb     64
b4      rmb     64      ;sprite #2 buffers
b5      rmb     64
b6      rmb     64      ;sprite #3
b7      rmb     64
;b8     rmb     64      ;sprite #4
;b9     rmb     64
;bA     rmb     64      ;sprite #5 buffers
;bB     rmb     64
;bC     rmb     64      ;sprite #6
;bD     rmb     64
;bE     rmb     64      ;sprite #7
;bF     rmb     64

bufl
        fcb     <b0
        fcb     <b1
        fcb     <b2
        fcb     <b3
        fcb     <b4
        fcb     <b5
        fcb     <b6
        fcb     <b7
;       fcb     <b8
;       fcb     <b9
;       fcb     <bA
;       fcb     <bB
;       fcb     <bC
;       fcb     <bD
;       fcb     <bE
;       fcb     <bF

bufh
        fcb     >b0
        fcb     >b1
```

```
        fcb     >b2
        fcb     >b3
        fcb     >b4
        fcb     >b5
        fcb     >b6
        fcb     >b7
;       fcb     >b8
;       fcb     >b9
;       fcb     >bA
;       fcb     >bB
;       fcb     >bC
;       fcb     >bD
;       fcb     >bE
;       fcb     >bF

HOWDY
; this is the way it is and the way it will be in the future.
; glory to be oh lord in the highest... peace and all that good stuff.
;
;
; 1. replace old background.. ( basically erase sprite)
; 2. get new sprite mask.
; 3. save new sprites background.
; 4. shift mask for positioning.
; 5. generate shadow for new shifted sprite.
; 6. "and" background out of where sprite goes.
; 7. "or" in sprite mask
; 8. done.....
;
;1)
;a. get old parameter for sprite on this background
;b. get position of where old background mask is
;c. copy background mask into background screen
;
;2)
;a. find new sprite mask
;b. copy new sprite mask into work buffer
;
;3)
;a. find position of new mask
;b. copy background screen into save area.
;
;4)
;a. using the position data determined above.
;   shift mask accordingly.
;
;5)
;a. copy shifted mask into and mask work area
;b. generate shadow mask for sprite
;
;6)
;a. UNPACK 'AND' AND 'OR' MASKS
;
;7)
;a. PUT UP new masks.
;
;
;
;1)
;a. get old parameter for sprite on this background
;b. get position of where old background mask is
```

```
;c. copy background mask into background screen
;
        lda     #0              ; sprite counter
        sta     objnum
;
restore
        jsr     getoff

        tax
        lda     bufl,x          ;modify code for address of this sprite's...
        sta     copys+1         ;...background buffer
        lda     bufh,x
        sta     copys+2
;
        lda     calptl,x        ;TO+2,3 <= screen address of where to ...
        sta     TO+2            ;...restore background
        lda     calpth,x
        sta     TO+3

;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
; RESTORE BACKGROUND
;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

        ldx     #0
dcpy
        ldy     #3
copys
        lda     copys,x         ;modified code
        sta     (TO+2),y        ;copy line to screen
        inx
        dey
        bpl     copys           ;continue
scpy
        cpx     #<slen4
        beq     larf            ;all done copying
        jsr     XNXTLIN
        jmp     dcpy

larf
        inc     objnum
        lda     objnum
        cmp     MAXOBJS ;#maxobj
        bne     restore

;=========================================================
;=========================================================
;==========    character to screen    ====================
;=========================================================
;=========================================================

;code for in HOWDY.src
        lda     CHARFLG         ;is a character ready for output
        beq     nochar          ;no,jmp
        dec     CHARFLG
;output the character in CHARG to screen,the screen offset is in CHARPO
        lda     CHARPO
        sta     TO+2
        lda     #$20
        ldx     TOGGLE
        beq     usethis
        lda     #$40
usethis
        ora     CHARPO+1
```

```
        sta     TO+3
        ldx     #14
newlin
        ldy     #0
        lda     CHARG,x
        sta     (TO+2),y
        lda     CHARG+1,x
        iny
        sta     (TO+2),y
        jsr     XNXTLIN
        dex
        dex
        bpl     newlin
nochar

;=========================================================
;=========================================================
;==========   Plot a pixel to screen   ===================
;=========================================================
;=========================================================

;code for in HOWDY.src
        lda     PLOTFLG         ;is a pixel plot ready for output?
        beq     noplot          ;no,jmp
        dec     PLOTFLG
;output the pixel of color PLOTPO+2 to address in PLOTPO+0,+1
        lda     PLOTPO          ;TO,T1 = screen offset with top3 bits...
        sta     TO              ;...set to bit number
        lda     PLOTPO+1
        sta     T1
        ldx     PLOTPO+2        ;x=clr # (0-5)
        ldy     #$20            ;y=hi screen start address
        lda     TOGGLE
        beq     gotpage
        ldy     #$40
gotpage
        lda     #$ff            ;accum="and" mask for offsets>$fff
        jsr     PLOTIT          ;call plot routine in INST1.src
noplot

;=========================================
BARF

;       enter here first time for each screen to initialize the...
;       ... background buffers

;2)
;b. copy new sprite mask into work buffer

        JSR     LDMAXL1
;       lda     #maxobj-1       ;do all 5 sprites
        sta     objnum

gmask
        jsr     getoff
        ldx     objnum          ;set sprite number
        lda     LBASE,x         ;ADDRESS OF SPRITE IN TO+0,1
        sta     TO
        lda     HBASE,x
        sta     TO+1

        ldx     objnum          ;or' mask buffer in TO+2,3
        lda     wrkl,x
```

```
        sta     TO+2
        lda     wrkh,x
        sta     TO+3

        ldx     offset
        ldy     objnum
        lda     OBJX,y          ;copy current coordinates to old values for thi
        sta     oldx,x
        lda     OBJY,y
        sta     oldy,x
;

        sec
        sbc     #176
        sta     botoff

        lda     oldy,x
        sec
        sbc     #16
        sta     topoff

;TO  POINTS TO SOURCE
;TO+2 points to DESTINATION

;WE MUST DO A LITTLE MANIPULATION ON THE MASK AS IT IS COPIED:
;source:         DEST:
;  TO+0,1        TO+2,3
;$AA,$BB         $00,$AA,$BB,$00
;$CC,$DD         $00,$CC,$DD,$00

        ldx     #<slen4-1       ;63
nxtl
        txa
        tay             ;63,59, ...31,27,23
        lda     #0
        sta     (TO+2),y

        dex
        txa
        lsr     a
        tay             ;31,29,...15,13,11
        lda     (TO),y
        pha
        txa
        tay             ;62,58,...30,26
        pla
        sta     (TO+2),y

        dex
        txa
        lsr     a
        tay             ;30,28,...14,12
        lda     (TO),y
        pha
        txa
        tay             ;61,57,...29,25
        pla
        sta     (TO+2),y

        dex
        txa
        tay             ;60,56,...28,24
```

```
        lda     #0
        sta     (T0+2),y

        dex                     ;59,55,...27,23,19,15,11,7,3,-1
        bpl     nxt1

        dec     objnum          ;check if all 5 sprites done
        bmi     cysz
        jmp     gmask

cysz

;3)
;b. copy background screen into save area.

        JSR     LDMAXL1
;       lda     #maxobj-1
        sta     objnum
forcry
        jsr     getoff
        tax
        lda     bufl,x          ;MODIFY CODE TO POINT TO 'OR' BUFFER
        sta     cay+1
        lda     bufh,x
        sta     cay+2
;
        lda     oldy,x          ;T0+2,3 <= screen address
        lsr     a               ;divide by 8
        lsr     a
        lsr     a
        sta     T0+3            ;save high byte
        lda     #$00
        ror     a
        sta     T0+2            ;roll into low byte
;
dasn
        lda     T0+3
        sec
        sbc     #$04
        bcc     ok3
        sta     T0+3
        lda     #$27            ; carry is set!
        adc     T0+2
        sta     T0+2
        jmp     dasn
ok3
        lda     oldy,x          ;set offset into character cell
        and     #$07    ;get remainer
        asl     a       ;mult by 4
        asl     a
        adc     T0+3    ;add to top nibble
        sta     T0+3    ;store it
        lda     oldx,x
        ldx     objnum
;divide routine
;enter with accum:  will perform accum/7
;div
        ldy     #00             ;whole number result
        sec
gts
        sbc     #07             ;x/a
        bcc     gots
```

```
        iny
        bne     sts             ;unc
sots
        adc     #7
;exit with remainder in accum and integer answer in register Y.

        sta     remain,x
        tya                     ;whole part
        asl     a
        adc     TO+2
        sta     TO+2
        bcc     nicx
        inc     TO+3
nicx
        lda     #$20            ;toggle to $2000 or $4000
        ldx     TOGGLE
        beq     nchb
        lda     #$40
nchb
        ora     TO+3
        sta     TO+3            ;TO+2,3 now has screen address
        ldx     offset
        sta     calpth,x        ;save screen address for later restore...
        lda     TO+2            ;...of background
        sta     calptl,x
;
;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
; SAVE BACKGROUND copys from screen to memory location
; TO+2 -> screen memory
;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

        ldx     #0
fcpy    ldy     #3              ;set size value
copyf   lda     (TO+2),Y        ;copy a line from SCREEN TO buffer
cay     sta     cay,x           ;copy line to buffer
        inx
        dey                     ;next byte
        bpl     copyf           ;continue
fdpy
        cpx     #<slen4
        beq     dcoy            ;all done copying
        jsr     XNXTLIN
        jmp     fcpy
dcoy
        dec     objnum
        bmi     frcryn
        jmp     forcry
frcryn


;4) do shift of mask for positioning

        JSR     LDMAXL1
;       lda     #maxobj-1
        sta     objnum
fponm
        jsr     getoff          ;find where on screen
        tax
        lda     #<slen          ;length
        sta     temp1
;
        ldx     objnum
```

```
        lda     remain,x
        asl     a                       ;mult the remainder by 2 ( number of pixels...
;                                       ;       ... off in character cell)
        sec
        sbc     #06                     ;determine if already @ center position?
        sta     temp                    ;save offset value
        beq     hover                   ;dont have to shift lines up perfecto
        bcc     shiftr                  ;shift the mother right
;ok shift the mother left a number of times

        ldx     objnum
        lda     wrkl,x                  ;modify code with 'or' mask address
        sta     loopa+1
        lda     wrkh,x
        sta     loopa+2

loopc
        ldy     temp                    ;set shift value
loopb
        ldx     #0                      ;set x to zero
        clc                             ;clear carry
loopa
        rol     loopa,x                 ;modified code  ;shift left <-
        inx                             ;inx
        txa
        eor     #$04
        bne     loopa                   ;dec temp length value
        dey                             ;dec shift value
        bne     loopb                   ;not do it again. ( max should be 3 times)/line
        dec     templ
        beq     hover
        lda     #$04
        clc
        adc     loopa+1
        sta     loopa+1
        bcc     loopc
        inc     loopa+2
        bne     loopc   ;unc            ;not done yet do another line

hover
        jmp     overi                   ;all done

shiftr
        lda     temp                    ;comp shift value ( remember carry is clear alr
        eor     #$ff
        adc     #$01
        sta     temp                    ;save new shift value

        ldx     objnum                  ;modify code for address of 'or' mask
        lda     wrkl,x
        sta     loopd+1
        lda     wrkh,x
        sta     loopd+2

loopf
        ldy     temp                    ;set shift value
loope
        ldx     #3                      ;set size data
        clc
loopd
        ror     loopd,x                 ;shift right -> ( move left)
        dex
        bpl     loopd                   ;continue to shift entire line
```

```
        dey                     ;dec number of times to shift
        bne     loope           ;continue to shift entire line again
        dec     templ           ;dec the number of lines
        beq     overi
        lda     #$04
        clc
        adc     loopd+1
        sta     loopd+1
        bcc     loopf
        inc     loopd+2
        bne     loopf   ;unc    ;continue till all are done
overi
        dec     objnum
        bmi     pomn
        jmp     fponm
pomn
;5)
;a. generate 'and' mask from shifted mask

        JSR     LDMAXL1
;       lda     #maxobj-1
        sta     objnum
csmmwa
        jsr     setoff

        ldx     objnum          ;TO+0,1 <= 'or' mask address
        lda     wrkl,x
        sta     TO
        lda     wrkh,x
        sta     TO+1

        lda     andl,x          ;TO+2,3 <= 'and' mask address
        sta     TO+2
        lda     andh,x
        sta     TO+3

        ldy     #<slen4-1       ;length
linee
        lda     (TO),y          ;get shifted mask
        tax                     ;use mask to index into table for "and" mask
        lda     mski,x          ;
        sta     (TO+2),y        ;save AS 'and' mask
        dey
        bpl     linee
;
        dec     objnum
        bmi     sntsk
        jmp     csmmwa
sntsk
;6)
;       UNPACK ALL SPRITES
;
;       JSR     LDMAXL1
;       lda     #maxobj-1
        sta     objnum
unpck
        jsr     setoff

        ldx     objnum          ;TO+0,1 <= 'and' mask address
        lda     andl,x
        sta     TO
        lda     andh,x
```

```
        sta     TO+1

        jsr     unpack

        ldx     objnum          ;TO+0,1 <= 'or' mask address
        lda     wrkl,x
        sta     TO
        lda     wrkh,x
        sta     TO+1

        jsr     unpack

        dec     objnum
        bpl     unpck


;7)
;       now actually put sprite on screen
;
        JSR     LDMAXL1
;       lda     #maxobj-1
        sta     objnum
final
        jsr     setoff
        tax
        lda     calptl,x        ;TO+2,3 <= screen address
        sta     TO+2
        lda     calpth,x
        sta     TO+3

        ldx     objnum          ;T6 <= top bit for color mode select
        lda     COLOR,x
        sta     T6

        lda     wrkl,x          ;TO+4,5 <= work buffer address ('or' mask)
        sta     TO+4
        lda     wrkh,x
        sta     TO+5

        lda     andh,x          ;TO+0,1 <= 'and' mask address
        sta     TO+1
        lda     andl,x
        sta     TO

        lda     #<slen          ;sprite length
        sta     templ

        ldy     #0              ;index into 'or'/'and' masks
        sty     temp
loopo
        ldx     #0              ;index into screen line
        stx     temps

loopin
        ldy     temp
        lda     (TO+4),y        ;get 'or' mask
        beq     noreason        ;if =0 then no reason to change background
;       beq     there

;THIS PROCESSING IF 'OR' MASK <> 0
        ldy     temps
        lda     (TO+2),y        ;get background
        ldy     temp
```

```
        and     (TO),y          ;'and' with 'and' mask
        ora     (TO+4),y        ;'or' in 'or' mask
        ora     T6              ;'or' in COLOR

        ldy     temps
;       jmp     entry

;there
;;THIS PROCESSING IF 'OR' MASK = 0
;       lda     (TO),y          ;get 'and' mask
;       ora     #$80            ;mask off color
;       ldy     temps
;       and     (TO+2),y        ;and in background

;entry
        sta     (TO+2),y        ;save on screen
noreason
        inc     temp    ;POINT TO NEXT BYTE OF MASK
        inc     temps   ;POINT TO NEXT BYTE IN SCREEN LINE
        lda     temps
        cmp     #4
        bne     loopin          ;continue IF STILL ON LINE
        dec     templ
        beq     fpaint
        jsr     XNXTLIN
        jmp     loopo
;
fpaint
        dec     objnum
        bmi     complet
        jmp     final

complet
;rts

STOGGLE
        lda     TOGGLE
        eor     MAXOBJS         ;#maxobj
        sta     TOGGLE
        beq     ntog
        lda     $c054   ;secondary page active now
        rts
ntog
        lda     $c055   ;primary page active now
        rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;   subroutines   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;
calptl  fcb     00,00,00,00     ;,00,00,00,00   ;screen address storage
        fcb     00,00,00,00     ;,00,00,00,00
```

```
calpth  fcb     00,00,00,00     ;,00,00,00,00
        fcb     00,00,00,00     ;,00,00,00,00

unpack
        ldy     #<slen4-3       ;sprite length
loopn
        lda     (T0),y  ;61,57
        lsr     a
        sta     (T0),y
        dey             ;60,56
        lda     (T0),y
        ror     a
        lsr     a
        sta     (T0),y
        iny
        iny             ;62,58
        lda     (T0),y
        tax
        and     #$7f
        sta     (T0),y
        txa
        asl     a
        iny             ;63,59
        lda     (T0),y
        rol     a
        and     #$7f
        sta     (T0),y

        tya
        sec
        sbc     #6
        tay             ;57,53,49,45,41,37,33,29,25,21,17,13,09,05,01,-3
        bcs     loopn
        rts

; subroutine to move T0 2&3 to next line on screen
XNXTLIN
        lda     T0+3
        tay                     ;save last value of high byte of screen pointe
        clc
        adc     #$04            ;add 400h to screen pointer
        sta     T0+3            ;save it
        tya                     ;determine if over screen boundary
        eor     T0+3
        and     #$e0            ;20h or 30h - > 40h or 40h or 50h -> 60h
        beq     loopz           ;continue not over
        lda     T0+2            ;adjust to next line byte
;       sec
        sbc     #$7f            ;80 subtracted 2000h
        sta     T0+2            ;and add 80h to goto next line
        lda     T0+3
        sbc     #$1f
        sta     T0+3
; the only problem is when you hit the roll over point
        and     #$0f
        cmp     #$04
        bcc     loopz           ;its ok roll over
        lda     T0+3
        sbc     #$04
        sta     T0+3
        lda     #$27            ;28
;       clc
        adc     T0+2
```

```
        sta     TO+2
loopz   rts                     ;continue till done
; adds objects wide ( size) to pointer


getoff
        lda     objnum
        clc
        adc     TOGGLE
        sta     offset
        rts



LBASE
        FCB     <BLNK
        FCB     <BLNK
        FCB     <BLNK
        FCB     <BLNK
;       FCB     <BLNK
;       FCB     <BLNK
;       FCB     <BLNK
;       FCB     <BLNK
HBASE
        FCB     >BLNK
        FCB     >BLNK
        FCB     >BLNK
        FCB     >BLNK
;FCB    >BLNK
;FCB    >BLNK
;FCB    >BLNK
;FCB    >BLNK
TOGGLE  fcb     0               ;goes back and forth from 0 to maxobj
                                ;0=primary graphics page ($2000-$3fff)
                                ;non-0=secondary graphics page ($4000-$5fff)
MAXOBJS FCB     04

LDMAXL1
        LDX     MAXOBJS
        DEX
        TXA
        RTS                     ;NO SPRITES.... ON.
botoff  fcb     0
topoff  fcb     0
lins    fcb     0
BLNK
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
        FCB     00,00
```

```
FCB     00,00
```