```
;   Program to demonstrate simple use of the Nick chip.
;
;   This program sets up a simple pixel mapped display and then runs a
; moving line demonstration program on it.  Note that this program is
; designed to be started from CP/M and so is origined at 100h.  Once
; started it makes no further use of CP/M.
;
;     Copyright 1984 Intelligent Software Ltd.

RAM0    EQU     0FCH            ; First 16K of video RAM
RAM1    EQU     0FDH            ; Second  ''
RAM2    EQU     0FEH            ; Third   ''
RAM3    EQU     0FFH            ; Fourth  ''


;                       Z80 SPACE ADDRESSES
;                       -------------------
screen  equ     08000h                  ;Start address for pixel screen
lpt     equ     0C000h                  ;Start address of line parameter
                                        ;   table in RAM.
stack   equ     0D000h                  ;Z-80 stack.
;
;
;
;-------------------------------------------------
;
;   Data areas used for the line drawing demo program
;
;
offset  equ     0D000h
colour  equ     0D002h
count   equ     0D003h
;
BLUE_TABLE      equ     0D100h
RED_TABLE       equ     0D200h
;
x1              equ     0               ;Offsets into RED and BLUE tables
y1              equ     1
x2              equ     2
y2              equ     3
delta_x1        equ     4
delta_y1        equ     5
delta_x2        equ     6
delta_y2        equ     7
table           equ     8               ;16*4 byte table
;
;
;
;
```

```
;                               MISC PARAMETERS
;                               ---------------
NULL    EQU     00                      ; NO USE VALUE

BLACK   equ     00000000b       ;Some useful colours to put in the palette.
RED     equ     01001001b
GREEN   equ     10010010b
YELLOW  equ     11011011b
BLUE    equ     00100100b
MAGENTA equ     01101101b
CYAN    equ     10110110b
WHITE   equ     11111111b


;                               LINE PARAMETER MASK BITS
;                               ------------------------

; MODE BYTE  (byte 1)
; ---------

RELOAD  EQU     001H            ; Reload line parameter counters
VBLANK  EQU     000H            ; Use margin for VSYNC
PIXEL   EQU     002H            ; Normal PIXEL mode
ATTR    EQU     004H            ; ATTRIBUTE mode
CH256   EQU     006H            ; 256 font CHAR mode
CH128   EQU     008H            ; 128 font CHAR mode
CH64    EQU     00AH            ; 64 font CHAR mode
TPIXEL  EQU     00CH            ; --------future expansion--------
LPIXEL  EQU     00EH            ; half resolution pixel mode

VRES    EQU     010H            ; low vertical resolution

C2      EQU     000H            ; 2-colour mode
C4      EQU     020H            ; 4-colour mode
C16     EQU     040H            ; 16-colour mode
C256    EQU     060H            ; 256-colour mode

VINT    EQU     080H            ; Take *INT low

; LEFT MARGIN  (byte 2)
; -----------

MSBALT  EQU     080H            ; Alt colours on D7 (PIXEL mode) 0->2
LSBALT  EQU     040H            ; Alt colours on D0 (PIXEL mode) 0->4

; RIGHT MARGIN  (byte 3)
; ------------

ALTIND1 EQU     080H            ; Alt colours on MSB of character index 0->4
ALTIND0 EQU     040H            ; Alt colours on next MSB of char index 0->2
```

```
;               Nick Chip I/O Ports
;               -------------------

FIXBIAS EQU     080H            ; Fixed colour bias
BORDER  EQU     081H            ; Border colour
LPL     EQU     082H            ; Line parameter pointer (A4-A11)
LPH     EQU     083H            ; Line parameter pointer (A12-A15) (see below)

;               LPH MASK BITS
;               ------------
LPLOAD  EQU     080H            ; Force *LOAD of LP counter low
LPCLOCK EQU     040H            ; Force *CLK of LP counter high


PAGE0   EQU     0B0H            ; Page in 4M of 0000H-3FFFH in Z80 space
PAGE1   EQU     0B1H            ; Page in 4M of 4000H-7FFFH in Z80 space
PAGE2   EQU     0B2H            ; Page in 4M of 8000H-BFFFH in Z80 space
PAGE3   EQU     0B3H            ; Page in 4M of C000H-FFFFH in Z80 space

        ASEG
        ORG     100h

        DI
        LD      A,RAM2          ;Put VIDEO-RAM in pages 2 and 3
        OUT     (PAGE2),A       ; (Page 1 not used, Page 0 already has this
        INC     A               ;  code in.)
        OUT     (PAGE3),A

        LD      A,0             ;Set BORDER colour
        OUT     (BORDER),A
        LD      A,067H          ;Any old value for fixed colour bias.
        OUT     (FIXBIAS),A

        LD      SP,STACK        ;Set up stack pointer

        LD      BC,ENTAB-LPTAB  ;Copy line parameter table from ROM
        LD      HL,LPTAB        ;  into video RAM.
        LD      DE,LPT
        LDIR

        LD      A,low(LPT/16)
        OUT     (LPL),A         ;Set up LINE PARAMETER POINTER
        LD      A,high(LPT/16)
        OUT     (LPH),A         ;Ensure LP counter is loaded (clk=0, load=0)
        OR      LPCLOCK         ;(clk=1)
        OUT     (LPH),A
        OR      LPLOAD          ;(load=1)
        OUT     (LPH),A         ;Then let it run
        ld      hl,screen       ;Clear the screen
        ld      de,screen+1
        ld      bc,256*64-1
        ld      (hl),0
        ldir
```

```
;-------------------------------------------------------------------------
;
;   The code from here on is just a moving line demo program which uses
; the RAM from SCREEN upwards as a 4 colour 256*256 pixel bit mapped display.
;
;
                ld      de,RED_TABLE
                ld      hl,red_initial
                call    setup_table
;
                ld      de,BLUE_TABLE
                ld      hl,blue_initial
                call    setup_table
;
;
;
LOOP:           ld      a,11110000b
                ld      (colour),a
                ld      hl,BLUE_TABLE
                call    ITERATE
;
                ld      a,00001111b
                ld      (colour),a
                ld      hl,RED_TABLE
                call    ITERATE
;
                ld      a,(count)
                inc     a
                and     0Fh
                ld      (count),a
;
                jr      LOOP
```

```
;
;-----------------------------------------
;
ITERATE:        push    hl
                call    NEW_XY
                inc     hl
                inc     hl
                call    NEW_XY
;
                pop     hl
                push    hl
                ld      a,(count)
                add     a,a
                add     a,a
                add     a,table
                ld      e,a
                ld      d,0
                add     hl,de
;
                push    hl
                call    LINE
;
                pop     de
                pop     hl
                push    de
                ld      bc,4
                ldir
;
                pop     hl
;
                jp      LINE
;
;
;
;-----------------------------
;
;
setup_table:    ld      bc,9
                ldir
                ld      h,d
                ld      l,e
                dec     hl
                ld      bc,16*4-1
                ldir
                ret
;
;
red_initial:    db      0,0,0,0,7,5,5,3,0
blue_initial:   db      255,0,255,0,256-5,5,256-9,3,0
;
```

```
;
;-------------------------------
;
NEW_XY:         push    hl
                pop     iy
                call    NEW_X_OR_Y
                inc     iy
;
;
NEW_X_OR_Y:     ld      a,r
                and     0Fh
                ld      c,a
                or      0F0h
                ld      d,a
;
                ld      a,(iy+0)
                ld      b,(iy+4)
                bit     7,b
                jr      nz,delta_minus
;
                add     a,b
                jr      c,overflow_plus
                cp      d
                jr      c,no_overflow
                cpl
overflow_plus:  cpl
                jr      overflow
;
delta_minus:    add     a,b
                jr      nc,overflow_minus
                cp      c
                jr      nc,no_overflow
                jr      overflow
overflow_minus: neg
overflow:
                ld      (iy+0),a
                xor     a
                sub     b
                ld      (iy+4),a
                ret
;
no_overflow:    ld      (iy+0),a
                ret
;
;
;
;--------------------
```

```
;
;
LINE:           ld      e,(hl)                  ;Get two co-ordinate pairs
                inc     hl                      ; into DE and HL
                ld      d,(hl)
                inc     hl
                ld      a,(hl)
                inc     hl
                ld      h,(hl)
                ld      l,a
;
                ld      a,h
                cp      d                       ;Start from lowest Y co-ord
                jr      nc,no_reverse           ; (in DE)
                ex      de,hl
no_reverse:
;
;
                ld      ix,normal_x
                ld      a,l                     ;If starting X is higher than
                sub     e                       ; finishing X then flag
                jr      nc,no_reflect_x         ; reflection is required and
                ld      ix,reflect_x            ; negate both X co-ordinates.
                sub     l
                ld      e,a
                neg                             ;E  = Starting X
no_reflect_x:   exx
                ld      c,a
                exx                             ;C' = DELTA_X
;
                ld      a,h
                sub     d
                exx                             ;D = Starting Y
                ld      b,a                     ;B' = DELTA_Y
;
                ld      iy,normal_xy            ;If DELTA_Y is more than
                cp      c                       ; DELTA_X then flag XY reflec-
                jr      c,no_reflect_xy         ; tion and swap DELTA_X and
                ld      iy,reflect_xy           ; DELTA_Y.
                ld      b,c
                ld      c,a
                exx
                ld      a,e                     ;Also swap starting X and Y
                ld      e,d                     ; co-ordinates over.
                ld      d,a
                exx
no_reflect_xy:
                ld      l,c                     ;Set HL' = ERROR = DELTA_X/2
                srl     l
                ld      h,0
                ld      a,c
                exx
                add     a,e
                ex      af,af'                  ;A' = Finishing X-coordinate
```

```
;
;
                ld      b,d             ;BC := initial co-ordinates
                ld      c,e
;
line_loop:      jp      (iy)            ;Transform from virtual co-ords
normal_x:                               ; in BC to screen co-ords in DE
                xor     a
                srl     d               ;Calculate address of pixel
                rr      e               ; = Y*64 + X/4
                rla
                srl     d
                rr      e
                rla
                set     7,d             ;Screen starts at 8000h
;
                ld      hl,mask_tab     ;Find correct mask for this
                add     a,l             ; pixel in the byte.
                ld      l,a
                ld      a,(colour)      ;Find correct mask for this
                and     (hl)            ; colour.
                ex      de,hl
                xor     (hl)            ;Exclusive-OR plot the pixel
                ld      (hl),a
;
;
                ex      af,af'          ;If reached final X co-ord then
                cp      c               ; exit.
                ret     z
                ex      af,af'
;
                inc     c               ;Step X-coordinate on by one.
                exx
                xor     a
                ld      d,a             ;ERROR -:= DELTA_Y
                ld      e,b
                sbc     hl,de
                exx
                jp      p,line_loop     ;If still positive then do
                exx                     ; not need to adjust Y.
                ld      e,c             ;Must go diagonally so
                add     hl,de           ; ERROR +:= DELTA_X
                exx
                inc     b               ;Step on Y-coordinate
                jp      line_loop
;
```

```
;
reflect_x:      xor     a
                sub     e
                ld      e,a
                jp      normal_x
;
normal_xy:      ld      d,b
                ld      e,c
                jp      (ix)
;
reflect_xy:     ld      d,c
                ld      e,b
                jp      (ix)



        IF      LOW($) GT 256-4
                ds      256-low($)
        ENDIF

mask_tab:       db      10001000b
                db      00100010b
                db      01000100b
                db      00010001b
```

```
;
;-------------------------------------------------------
;
;   The line parameter table.  This defines the screen format.
;
LPTAB:

; PIXEL for 256 lines (4 colour)
        db      256-256,pixel+c4+vres,14,14+32          ;This is the actual
        dw      screen,0                                ; display area.
        db      black,red,blue,magenta,0,0,0,0

; BLANK for 5 lines                                     ;Bottom margin.
        db      256-5,1pixel,63,0
        db      0,0,0,0,0,0,0,0,0,0,0,0

; BLACK for 10 lines                                    ;Start of vertical
        db      256-10,1pixel,4,63                      ; blanking period.
        db      0,0,0,0,0,0,0,0,0,0,0,0

; Start vertical sync (4 lines)                         ;Vertical sync pulse
        db      256-4,vblank,4,63
        db      0,0,0,0,0,0,0,0,0,0,0,0

; End vertical sync (1 line)                            ;Finish vertical sync
        db      256-1,vblank,63,4                       ; pulse.
        db      0,0,0,0,0,0,0,0,0,0,0,0

; BLACK for 30 lines
        db      256-30,1pixel,4,63                      ;Rest of vertical
        db      0,0,0,0,0,0,0,0,0,0,0,0                 ; blanking period.

; BLANK for 6 lines
        db      256-6,reload+1pixel,63,0                ;Top marigin with
        db      0,0,0,0,0,0,0,0,0,0,0,0                 ; RELOAD to start again
ENTAB   db      0
```