

## PROGRAM LISTING 6: MPASM VIDEO INTERFACE PROGRAM

By Malcolm Stagg

```

;
;|-----|
;| Video Interface Controller (VIC) |
;|       with Mean Filter           |#
;|       [Source Code]              |#
;|-----|
;|                               |#
;|               By Malcolm Stagg  |#
;|                               |#
;| -Designed for the Science Fairs- |#
;|                               |#
;| Used to interface with a 320x240 |#
;| standard video signal. Please   |#
;| see circuit diagram for more    |#
;| connection information.         |#
;|-----|
;| #####
;
; <----- [PIN OUTS] ----->
;
; PORTA.0 - RAM RE           -O
; PORTA.1 - RESTART FRAME   -I
; PORTA.2 - RAM WE           -O
; PORTA.3 - RAM W RESET     -O
; PORTA.4 - ODD              -I
; PORTA.5 - RAM R RESET     -O
;
; PORTB.0-7 - RAM DATA OUT -I
;
; PORTC.0 - HANDSHAKE OUT   -O
; PORTC.1 - HANDSHAKE IN   -I
; PORTC.2 - RAM R CLOCK    -O
;
; PORTD.0-7 - LAYER 2 OUT  -O
;
; <----- [PROGRAM] ----->
; LIST P=18F452, R=DEC
; INCLUDE "p18f452.inc"
;
; CBLOCK 0x020
;     var, times, times2, times3, times4, times5, temp, times6, times7
;     pointh, pointl
;     a, b, c
;     d, e, f
;     g, h, i
;     getxl, getxh, gety
;     d1h, d1l, d2l
;
; ENDC
; line equ 0x80
;
; _CONFIG 0x300001, 0xFA ;HS-PLL, No Osc Switch
; _CONFIG 0x300002, 0xFC ;Power Up Timer, No Brown Out, 2V
; _CONFIG 0x300003, 0xFE ;No Watchdog Timer, 1:128
; _CONFIG 0x300005, 0xFF ;CCP2 Mux RC1

```

```
__CONFIG 0x300006, 0x7B ;Background Debug (+80 for no), Stack  
Overflow, No Low Voltage
```

```
__CONFIG 0x300008, 0xFF ;Code Protect Stuff...
```

```
__CONFIG 0x300009, 0xFF
```

```
__CONFIG 0x30000A, 0xFF
```

```
__CONFIG 0x30000B, 0xFF
```

```
__CONFIG 0x30000C, 0xFF
```

```
__CONFIG 0x30000D, 0xFF
```

```
org 0
```

```
nop
```

```
clrf pointh
```

```
clrf pointl
```

```
clrf getxl
```

```
clrf getxh
```

```
clrf gety
```

```
;I/O setup
```

```
movlw 7
```

```
movwf ADCON1 ; set all pins to digital
```

```
;bsf STATUS, 5
```

```
movlw b'00010010'
```

```
movwf TRISA
```

```
clrf PORTA
```

```
movlw b'11111111'
```

```
movwf TRISB
```

```
movlw b'00000010'
```

```
movwf TRISC
```

```
clrf PORTC
```

```
movlw b'00000000'
```

```
movwf TRISD
```

```
clrf PORTD
```

```
movlw b'00000001'
```

```
movwf TRISE
```

```
clrf PORTE
```

```
;bcf STATUS, 5
```

```
bsf PORTA, 2 ; write enabled for 80+ dummy read/write cycles
```

```
bsf PORTA, 0
```

```
movlw 100
```

```
movwf temp
```

```
bsf PORTC, 2 ; Advance RAM Read Clock
```

```
nop
```

```
bcf PORTC, 2
```

```
decfsz temp
```

```
goto $ - 8
```

```
;get a frame first:
```

```
getfram
```

```
bcf PORTA, 2
```

```
bcf PORTA, 0
```

```

    btfsc PORTA, 4 ; wait for even field
    goto $ - 2

    bsf PORTA, 2 ; write enabled
    bsf PORTA, 3 ; reset write
    nop
    nop
    nop
    nop
    bcf PORTA, 3

    btfss PORTA, 4 ; wait for odd field
    goto $ - 2
    btfsc PORTA, 4 ; wait for even field
    goto $ - 2
    bcf PORTA, 2; write disabled

;delay 1000 clocks (only 960 needed)

    movlw 250 ; *4 in loop
    movwf times

    bsf PORTA, 0

loop   nop
       decfsz times
       goto loop

;get pixels
    bsf PORTC, 3 ; I'm starting a new frame now
    btfss PORTC, 1 ; ok, continue?
    goto $ - 2
    bcf PORTC, 3 ; end pulse
    btfsc PORTC, 1 ; end pulse?
    goto $ - 2

frame bcf PORTA, 0
      nop
      nop
      nop
      nop
      nop
      bsf PORTA, 0 ; enable read
      bsf PORTA, 5 ; reset read
      nop
      bsf PORTC, 2 ; Advance RAM Read Clock
      nop
      bcf PORTC, 2
      nop
      bcf PORTA, 5

      call linest2 ;[* 8]

```

```

    movlw 240 ; lines = 240
    movwf times2

repeat2
    call linest
    movlw 160 ; times = times to repeat/2 (140)
    movwf times
    ;bcf PORTA, 5 ; can delete if other uncommented

repeat call get1
    movwf temp
    btfsc temp, 0
    goto frame ; restart frame

    bsf PORTC, 2 ; Advance RAM Read Clock (skip 2)
    nop
    bcf PORTC, 2
    nop

    call get1
    movwf temp
    btfsc temp, 0
    goto frame ; restart frame

    bsf PORTC, 2 ; Advance RAM Read Clock (skip 2)
    nop
    bcf PORTC, 2
    nop

    decfsz times ; repeat pixel capture
    goto repeat
    decfsz times2 ; repeat for each line
    goto repeat2

    goto getfram ; get another frame

;subs

linest bsf PORTC, 2 ; Advance RAM Read Clock
    nop
    bcf PORTC, 2
    nop
    movf PORTB, w ; w = pixel(Clock)
    movwf var ; var = pixel
    btfss var, 0
    goto linest ; if no sync, goto linest

    movlw 112 ; skip 112 clock for front porch of video signal
    movwf times5
    bsf PORTC, 2 ; Advance RAM Read Clock
    nop
    bcf PORTC, 2

```

```

        decfsz times5
        goto $ - 8

        return

linest2
        movlw 128
        movwf times6
linest2a
        movlw 25
        movwf times5
linest2b

        bsf PORTC, 2 ; Advance RAM Read Clock
        nop
        bcf PORTC, 2
        nop
        bsf PORTC, 2 ; Advance RAM Read Clock
        nop
        bcf PORTC, 2

        decfsz times5
        goto linest2b
        decfsz times6
        goto linest2a

        return

get1
back

;mean filter
        movf pointh, w
        xorlw HIGH 960
        btfss STATUS, Z
        goto array

        movf pointl, w
        xorlw LOW 960
        btfsc STATUS, Z
        goto compute ; done capture of 3 lines

array
        movlw HIGH line
        addwf pointh, w
        movwf FSR0H

        movlw LOW line
        addwf pointl, w
        movwf FSR0L
        btfsc STATUS, C
        incf FSR0H

```

```

bsf PORTC, 2 ; Advance RAM Read Clock
nop
bcf PORTC, 2
nop
movf PORTB, w ; w = pixel(Clock)
movwf INDF0

incf pointl
btfsc STATUS, Z
incf pointh

; don't use the following code with the mean filter
; movwf PORTD ; send pixel through "parallel" port D
; bsf PORTC, 0 ; ok, I sent it
; btfss PORTC, 1 ; thanks, I got it?
; goto $ - 2
; bcf PORTC, 0 ; end pulse
; btfsc PORTA, 1 ; restart frame?
; retlw 1
; btfsc PORTC, 1 ; pulse has ended?
; goto $ - 2

retlw 0

```

compute

```

; line [getx]

movlw HIGH line
addwf getxh, w
movwf FSR0H

movlw LOW line
addwf getxl, w
btfsc STATUS, C
incf FSR0H
movwf FSR0L

movf INDF0, w
movwf a

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line [getx + 1]

movf INDF0, w
movwf b

incf FSR0L
btfsc STATUS, Z
incf FSR0H

```

```

; line [getx + 2]

movf INDF0, w
movwf c

incf FSR0H
movlw 62
addwf FSR0L
btfsc STATUS, C
incf FSR0H

; line [getx + 320]

movf INDF0, w
movwf d

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line [getx + 321]

movf INDF0, w
movwf e

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line [getx + 322]

movf INDF0, w
movwf f

incf FSR0H
movlw 62
addwf FSR0L
btfsc STATUS, C
incf FSR0H

; line [getx + 640]

movf INDF0, w
movwf g

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line [getx + 641]

movf INDF0, w

```

```

movwf h

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line [getx + 642]

movf INDF0, w
movwf i

movf b, w
clrf b
addwf a
btfsc STATUS, C
incf b

movf c, w
addwf a
btfsc STATUS, C
incf b

movf d, w
addwf a
btfsc STATUS, C
incf b

movf f, w
addwf a
btfsc STATUS, C
incf b

movf g, w
addwf a
btfsc STATUS, C
incf b

movf h, w
addwf a
btfsc STATUS, C
incf b

movf i, w
addwf a
btfsc STATUS, C
incf b

; b high byte, a low byte
; divide by 16

;divide by 2 first
bcf STATUS, C
rrcf b, f

```

```

    rrcf a, f
    ;divide by 2 (4)
    bcf STATUS, C
    rrcf b, f
    rrcf a, f
    ;divide by 2 (8)
    bcf STATUS, C
    rrcf b, f
    rrcf a, f
    ;divide by 2 (16)
    bcf STATUS, C
    rrcf b, f
    rrcf a, f

    ;add e / 2

    bcf STATUS, C
    rrcf e, w
    addwf a

    ; output a

    call fdone

    movf a, w

    movwf PORTD ; send pixel through "parallel" port D
    bsf PORTC, 0 ; ok, I sent it
    btfss PORTC, 1 ; thanks, I got it?
    goto $ - 2
    bcf PORTC, 0 ; end pulse
    btfsc PORTA, 1 ; restart frame?
    retlw 1
    btfsc PORTC, 1 ; pulse has ended?
    goto $ - 2

;    retlw 0
    goto get1

fdone  incf getxl
       btfsc STATUS, Z
       incf getxh

       btfss getxh, 0
       return
       movf getxl, w
       xorlw 0x40
       btfsc STATUS, Z
       call shiftln

       return

shiftln  movlw LOW 640

```

```

movwf pointl
movlw HIGH 640
movwf pointh
clrf getxl
clrf getxh

movlw 1
movwf d1h
movlw 0
movwf d1l
goto for2
for2b clrf d1l ;same as =256 to decfsz
; line [d1]

for2 movlw HIGH line
addwf d1h, w
;addlw 1
movwf FSR0H

movlw LOW line
addwf d1l, w
btfsc STATUS, C
incf FSR0H
addlw 64 ; what if > 255?
btfsc STATUS, C
incf FSR0H
movwf FSR0L

movf INDF0, w
movwf d2l

movlw HIGH line
addlw 1
subwf d1h, w ; w = f - w -> w = d1h - 1
movwf FSR0H

movlw LOW line
addwf d1l, w
btfsc STATUS, C
incf FSR0H
movwf FSR0L

movf d2l, w
movwf INDF0

;decfsz d1l
;goto for2
;decfsz d1h
;goto for2b

incf d1l
btfss STATUS, Z

```

```
        goto sline1
        incf d1h
        goto for2b

sline1  movf d1h, w
        xorlw 3
        btfss STATUS, Z
        goto sline2
        movf d1l, w
        xorlw 128
        btfsc STATUS, Z
        goto finish
sline2  goto for2

finish  return

        end
```