

**PROGRAM LISTING 7:
MPASM EDGE DETECTION PROGRAM**

By Malcolm Stagg

```
;
;-----|
;| Edges Controller (EC) |
;| With Non-Maximal Suppression |#
;| [Source Code] |#
;|-----|#
;| By Malcolm Stagg |#
;| |#
;| -Designed for the Science Fair- |#
;| |#
;| Used to change a signal from a |#
;| VIC to simple edges. Please see |#
;| circuit diagram for connection |#
;| information. |#
;|-----|#
;| #####|
;
; <-----[PIN OUTS]----->
;
; PORTA.0 - HANDSHAKE1 OUT -O
; PORTA.1 - HANDSHAKE1 IN -I
; PORTA.2 - RESTART FRAME -O
; PORTA.3 - NEW FRAME -I
;
; PORTB.0-7 - LAYER 1 IN -I
;
; PORTC.0 - HANDSHAKE2 OUT -O
; PORTC.1 - HANDSHAKE2 IN -I
; PORTC.2 - EDGE OUT -O
; PORTC.3 - NEW FRAME2 -O
; PORTC.4 - RESTART FRAME2 -I
;
; PORTD.0-7 - THRESHOLD IN -I
;
; <-----[PROGRAM]----->
; LIST P=18F452, R=DEC
; INCLUDE "p18f452.inc"
;
; CBLOCK 0x020
; getxl, getxh, t
; n1l, n1h, n2l, n2h
; d1l, d1h, d2l, d2h
; p2, p4, p6, p8
; k1a, k1b, k1c, k1d, k1e, k1f
; k2a, k2b, k2c, k2d, k2e, k2f
; pointl, pointh
; ktemp
; tmp1, tmp2
; bit, bit2, byte
; pass
; hmx, vmx, hgv
; y
; ENDC
```

```
line equ 0x80 ; 320 byte
line2 equ 0x1C0 ;320 byte
line3 equ 0x300 ;320 byte
line4 equ 0x440 ;320 byte
hgrv equ 0x580 ; 320 bit
ymax equ 0x5A8 ; 320 bit
hgrv2 equ 0x5D0 ;320 bit
```

```
;0x70F bytes of memory used
;=1807 bytes = 1.76kb
```

```
__CONFIG 0x300001, 0xFA ;FE 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, 0xFB ;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
```

```
;I/O setup
```

```
movlw 6
movwf ADCON1 ; set all pins to digital
movlw b'00001010'
movwf TRISA
;clrf PORTA
movlw b'11111111'
movwf TRISB
movlw b'11001010'
movwf TRISC
movlw b'11111111'
movwf TRISD
movlw b'00000000'
movwf TRISE
clrf pinth
clrf pointl
```

```
clrf PORTA
clrf PORTC
```

```
chkfrm movf PORTD, w ; set threshold
movwf t
```

```
btfs PORTA, 3 ; there is going to be a new frame?
call nf ; yes
```

```

    btfsc PORTA, 1 ; VIC's going to send a pixel?
    call chkpx ; yes

    goto chkfrm

chkpx
    movf pointh, w
    xorlw HIGH 1280
    btfss STATUS, Z
    goto array

    movf pointl, w
    xorlw LOW 1280
    btfsc STATUS, Z
    goto compute ; done capture of 4 lines

    ; line [point] = PORTB

array

    movlw HIGH line
    addwf pointh, w
    movwf FSR0H

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

    bcf STATUS, C ; 7-bit
    rrcf PORTB, w
    movwf INDF0

    bcf INDF0, 5 ;allow all data to be recieved when debugger plugged in
    bcf INDF0, 6
    btfsc PORTC, 6
    bsf INDF0, 5
    btfsc PORTC, 7
    bsf INDF0, 6

    incf pointl
    btfsc STATUS, Z
    incf pointh

    bsf PORTA, 0
    btfsc PORTA, 1
    goto $ - 2
    bcf PORTA, 0

    return

rf    bsf PORTA, 2 ; please restart this frame

```

```

    btfss PORTA, 1 ; wait for ok
    goto $ - 2
    bsf PORTA, 0 ; good pulse
    btfsc PORTA, 1 ; wait for ok to end
    goto $ - 2
    bcf PORTA, 0 ; end good pulse
    bcf PORTA, 2

    clrf pointl ; reset all variables for the next video frame
    clrf pointh

    retlw 0

nf
    clrf getxl
    clrf getxh
    clrf pointl
    clrf pointh

    bsf PORTA, 0 ; ok - i'm ready for you to send it
    btfsc PORTA, 3 ; wait for VIC to finish pulse
    goto $ - 2
    bcf PORTA, 0

    bsf PORTC, 4 ; layer 3: You'll be getting a new frame
    btfss PORTC, 1 ; do you read me?
    goto $ - 2
    bcf PORTC, 4 ; end new frame pulse
    btfsc PORTC, 1 ; good - wait for pulse to end
    goto $ - 2

    return

compute
    clrf getxh
    clrf getxl

    btfss pass, 0
    call sobel123

    clrf getxh
    clrf getxl

    call sobel234

    clrf getxh
    clrf getxl

    btfsc pass, 0
    call ymaxupdt ; ymax(n) is 0 anyways for the first pass

    clrf getxh
    clrf getxl

```

```
    btfsc pass, 0
    call suppression

    clrf getxh
    clrf getxl

    call ymaximum

    call shiftln ; remember to move hgrv2 to hgrv as well

    bsf pass, 0

    return

sobel123
```

```
    ; 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 k1a
    movwf k2a

    incf FSR0L
    btfsc STATUS, Z
    incf FSR0H

    ; line [getx + 1]

    movf INDF0, w
    movwf p2

    incf FSR0L
    btfsc STATUS, Z
    incf FSR0H

    ; line [getx + 2]

    movf INDF0, w
    movwf k1d
    movwf k2c

    incf FSR0H
```

```

movlw 62
addwf FSR0L
btfsc STATUS, C
incf FSR0H

; line [getx + 320]

movf INDF0, w
movwf p4

incf FSR0L
btfsc STATUS, Z
incf FSR0H
incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line [getx + 322]

movf INDF0, w
movwf p6

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

; line [getx + 640]

movf INDF0, w
movwf k1c
movwf k2d

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line [getx + 641]

movf INDF0, w
movwf p8

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line [getx + 642]

movf INDF0, w
movwf k1f
movwf k2f

```

```
;p1 p2 p3  
;p4 p5 p6  
;p7 p8 p9
```

```
;k1a k1d  
;k1b k1e  
;k1c k1f
```

```
;k2a k2b k2c
```

```
;k2d k2e k2f
```

```
bcf STATUS, C  
rlcf p4, w  
movwf k1b
```

```
bcf STATUS, C  
rlcf p6, w  
movwf k1e
```

```
bcf STATUS, C  
rlcf p2, w  
movwf k2b
```

```
bcf STATUS, C  
rlcf p8, w  
movwf k2e
```

```
clrf d1h  
clrf d2h
```

```
clrf n1h  
movf k1a, w  
movwf n1l  
movf k1b, w  
addwf n1l  
btfsc STATUS, C  
incf n1h  
movf k1c, w  
addwf n1l  
btfsc STATUS, C  
incf n1h
```

```
clrf n2h  
movf k1d, w  
movwf n2l  
movf k1e, w  
addwf n2l  
btfsc STATUS, C  
incf n2h  
movf k1f, w  
addwf n2l  
btfsc STATUS, C
```

```

incf n2h

movf n1h, w
subwf n2h, w ; w = n2h - n1h
btfss STATUS, C
goto n1mn2
btfss STATUS, Z
goto n2mn1

movf n1l, w
subwf n2l, w; w = n2h - n1h
movwf d1l
btfsc STATUS, C
goto nxt

movf n2l, w
subwf n1l, w
movwf d1l
goto nxt

n2mn1 movwf d1h
movf n1l, w
subwf n2l, w ; w = n2l - n1l
movwf d1l
btfsc STATUS, C
goto nxt ; positive
decf d1h
goto nxt

n1mn2 movf n2h, w
subwf n1h, w ; w = n1h - n2h
movwf d1h
movf n2l, w
subwf n1l, w ; w = n1l - n2l
movwf d1l
btfsc STATUS, C
goto nxt ; positive
decf d1h

nxt  clrf n1h
movf k2a, w
movwf n1l
movf k2b, w
addwf n1l
btfsc STATUS, C
incf n1h
movf k2c, w
addwf n1l
btfsc STATUS, C
incf n1h

clrf n2h
movf k2d, w

```

```

movwf n2l
movf k2e, w
addwf n2l
btfsc STATUS, C
incf n2h
movf k2f, w
addwf n2l
btfsc STATUS, C
incf n2h

movf n1h, w
subwf n2h, w ; w = n2h - n1h
btfss STATUS, C
goto n1mn2b
btfss STATUS, Z
goto n2mn1b

movf n1l, w
subwf n2l, w; w = n2h - n1h
movwf d2l
btfsc STATUS, C
goto nextb

movf n2l, w
subwf n1l, w
movwf d2l
goto nextb

n2mn1b      movwf d2h
            movf n1l, w
            subwf n2l, w ; w = n2l - n1l
            movwf d2l
            btfsc STATUS, C
            goto nextb ; positive
            decf d2h
            goto nextb

n1mn2b      movf n2h, w
            subwf n1h, w ; w = n1h - n2h
            movwf d2h
            movf n2l, w
            subwf n1l, w ; w = n1l - n2l
            movwf d2l
            btfsc STATUS, C
            goto nextb ; positive
            decf d2h
            incf d2l

nextb      movf d1h, w
            btfss STATUS, Z
            setf d1l

            movf d2h, w

```

```

    btfss STATUS, Z
    setf d2l

    clrf d2h
    clrf d1h

    movf d1l, w
    mulwf d1l

    movf PRODH, w
    movwf d1h
    movf PRODL, w
    movwf d1l

    movf d2l, w
    mulwf d2l

    movf PRODH, w
    movwf d2h
    movf PRODL, w
    movwf d2l

    movf d1l, w
    addwf d2l, w
    movwf d1l
    btfsc STATUS, C
    incf d1h

    movf d1h, w
    addwf d2h, w
    movwf d1h

    btfss STATUS, C
    goto $ + 8
    movlw 0xFD
    movwf d1h

;FD00 is max for sqr

    sublw 0xFC ;0xFC - d1h
    btfsc STATUS, C
    goto $ + 8
    movlw 0xFD
    movwf d1h

    clrf n1l
for1  incf n1l

    movf n1l, w
    mulwf n1l

    movf d1h, w

```

```

subwf PRODH, w
btfss STATUS, C
goto for1 ; d1h < result

btfss STATUS, Z
goto save1

movf d1l, w
subwf PRODL, w
btfsc STATUS, C ; changed!!
goto save1 ; d1l < result

goto for1

; n1l is square root of kernel result

save1
movlw HIGH line
addwf getxh, w
movwf FSR0H

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

movf n1l, w
movwf INDF0 ; store in line 1

incf getxl
btfsc STATUS, Z
incf getxh

btfss getxh, 0
goto sobel123

movf getxl, w
xorlw 0x40 ;3E maybe

btfss STATUS, Z
goto sobel123

;repeat until done line

return

sobel234

; line2 [getx]

movlw HIGH line2
addwf getxh, w

```

```

movwf FSR0H

movlw LOW line2
addwf getx1, w
btfsc STATUS, C
incf FSR0H
movwf FSR0L

movf INDF0, w
movwf k1a
movwf k2a

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line2 [getx + 1]

movf INDF0, w
movwf p2

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line2 [getx + 2]

movf INDF0, w
movwf k1d
movwf k2c

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

; line2 [getx + 320]

movf INDF0, w
movwf p4

incf FSR0L
btfsc STATUS, Z
incf FSR0H
incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line2 [getx + 322]

movf INDF0, w
movwf p6

```

```

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

; line2 [getx + 640]

movf INDF0, w
movwf k1c
movwf k2d

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line2 [getx + 641]

movf INDF0, w
movwf p8

incf FSR0L
btfsc STATUS, Z
incf FSR0H

; line2 [getx + 642]

movf INDF0, w
movwf k1f
movwf k2f

;p1 p2 p3
;p4 p5 p6
;p7 p8 p9

;k1a k1d
;k1b k1e
;k1c k1f

;k2a k2b k2c

;k2d k2e k2f

bcf STATUS, C
rlcf p4, w
movwf k1b

bcf STATUS, C
rlcf p6, w
movwf k1e

bcf STATUS, C

```

```
rlcf p2, w
movwf k2b
```

```
bcf STATUS, C
rlcf p8, w
movwf k2e
```

```
clrf d1h
clrf d2h
```

```
clrf n1h
movf k1a, w
movwf n1l
movf k1b, w
addwf n1l
btfsc STATUS, C
incf n1h
movf k1c, w
addwf n1l
btfsc STATUS, C
incf n1h
```

```
clrf n2h
movf k1d, w
movwf n2l
movf k1e, w
addwf n2l
btfsc STATUS, C
incf n2h
movf k1f, w
addwf n2l
btfsc STATUS, C
incf n2h
```

```
movf n1h, w
subwf n2h, w ; w = n2h - n1h
btfss STATUS, C
goto n1mn22
btfss STATUS, Z
goto n2mn12
```

```
movf n1l, w
subwf n2l, w; w = n2h - n1h
movwf d1l
btfsc STATUS, C
goto nxt2
```

```
movf n2l, w
subwf n1l, w
movwf d1l
goto nxt2
```

```
n2mn12      movwf d1h
```

```

    movf n1l, w
    subwf n2l, w ; w = n2l - n1l
    movwf d1l
    btfsc STATUS, C
    goto nxt2 ; positive
    decf d1h
    goto nxt2

n1mn22      movf n2h, w
            subwf n1h, w ; w = n1h - n2h
            movwf d1h
            movf n2l, w
            subwf n1l, w ; w = n1l - n2l
            movwf d1l
            btfsc STATUS, C
            goto nxt2 ; positive
            decf d1h

nxt2      clrf n1h
            movf k2a, w
            movwf n1l
            movf k2b, w
            addwf n1l
            btfsc STATUS, C
            incf n1h
            movf k2c, w
            addwf n1l
            btfsc STATUS, C
            incf n1h

            clrf n2h
            movf k2d, w
            movwf n2l
            movf k2e, w
            addwf n2l
            btfsc STATUS, C
            incf n2h
            movf k2f, w
            addwf n2l
            btfsc STATUS, C
            incf n2h

            movf n1h, w
            subwf n2h, w ; w = n2h - n1h
            btfss STATUS, C
            goto n1mn2b2
            btfss STATUS, Z
            goto n2mn1b2

            movf n1l, w
            subwf n2l, w; w = n2h - n1h
            movwf d2l
            btfsc STATUS, C

```

```

goto nextb2

movf n2l, w
subwf n1l, w
movwf d2l
goto nextb2

n2mn1b2    movwf d2h
            movf n1l, w
            subwf n2l, w ; w = n2l - n1l
            movwf d2l
            btfsc STATUS, C
            goto nextb2 ; positive
            decf d2h
            goto nextb2

n1mn2b2    movf n2h, w
            subwf n1h, w ; w = n1h - n2h
            movwf d2h
            movf n2l, w
            subwf n1l, w ; w = n1l - n2l
            movwf d2l
            btfsc STATUS, C
            goto nextb2 ; positive
            decf d2h

nextb2 ;movf d1h, w
        ;btfss STATUS, Z
        ;setf d1l

        ;movf d2h, w
        ;btfss STATUS, Z
        ;setf d2l

        call bits

        ; bit 0 - 7, byte 0 - 39

        movlw HIGH hgrv2
        movwf FSR0H
        movlw LOW hgrv2
        addwf byte, w
        movwf FSR0L
        btfsc STATUS, C
        incf FSR0H

        btfsc bit2, 0
        clrf INDF0 ; if on the first bit, clear variable

        ;d2l is vertical
        ;d1l is horizontal
        ;d1l - d2l >= 0 for hgrv = 1

```

```

    movf d2h, w
    subwf d1h, w ; w = d1l - d2l
    btfss STATUS, C
    goto hvskip

    btfss STATUS, Z
    goto hvdo

    movf d2l, w
    subwf d1l, w ; w = d1l - d2l
    btfss STATUS, C
    goto hvskip

;movf d2l, w
;subwf d1l, w ; w = d1l - d2l
;btfss STATUS, C
;goto hvskip

hvdo  movf bit2, w
      iorwf INDF0, f ; is >= 0, so set this bit to 1

hvskip ;clrf d2h
      ;clrf d1h

      ;movf d1l, w
      ;mulwf d1l

      ;movf PRODH, w
      ;movwf d1h
      ;movf PRODL, w
      ;movwf d1l

      ;movf d2l, w
      ;mulwf d2l

      ;movf PRODH, w
      ;movwf d2h
      ;movf PRODL, w
      ;movwf d2l

      movf d1l, w
      addwf d2l, w
      movwf d1l
      btfsc STATUS, C
      incf d1h

      movf d1h, w
      addwf d2h, w
      movwf d1h

      ;-----New Code-----
      bcf STATUS, C
      rrcf d1h, w

```

```

movwf n1h
rrcf d1l, w
movwf n1l ;/2
movf n1h, w
btfss STATUS, Z
setf n1l
;-----

; btfss STATUS, C
; goto $ + 8
; movlw 0xFD
; movwf d1h

; FD00 is max for sqr

; sublw 0xFC ; 0xFC - d1h
; btfsc STATUS, C
; goto $ + 8
; movlw 0xFD
; movwf d1h

; clrf n1l

; for12 incf n1l

;   movf n1l, w
;   mulwf n1l

;   movf d1h, w
;   subwf PRODH, w
;   btfss STATUS, C
;   goto for12 ; d1h < result

;   btfss STATUS, Z
;   goto save2

;   movf d1l, w
;   subwf PRODL, w
;   btfsc STATUS, C ; CHANGED TO btfsc!!!
;   goto save2 ; d1h < result

;   goto for12

; n1l is square root of kernel result

save2
movlw HIGH line2
addwf getxh, w
movwf FSR0H

movlw LOW line2
addwf getxl, w
btfsc STATUS, C

```

```

    incf FSR0H
    movwf FSR0L

    movf n1l, w
    movwf INDF0 ; store in line 2

    ;call thresh
    ;call outedge

    incf getxl
    btfsc STATUS, Z
    incf getxh

    btfss getxh, 0
    goto sobel234

    movf getxl, w
    xorlw 0x40 ;3E maybe

    btfss STATUS, Z
    goto sobel234

    ;repeat until done line

    return

bits
    clrf byte
    clrf bit
    btfsc getxh, 0
    bsf byte, 5
    btfsc getxl, 7
    bsf byte, 4
    btfsc getxl, 6
    bsf byte, 3
    btfsc getxl, 5
    bsf byte, 2
    btfsc getxl, 4
    bsf byte, 1
    btfsc getxl, 3
    bsf byte, 0
    btfsc getxl, 2
    bsf bit, 2
    btfsc getxl, 1
    bsf bit, 1
    btfsc getxl, 0
    bsf bit, 0

    incf bit
    movlw 1
bitloop
    movwf bit2
    rlnf bit2, w

```

```

    decfsz bit
    goto bitloop

;bit2 should now = 1, 2, 4, 8, 16, 32, 64, or 128

    return

ymaxupdt
    call bits

    movlw HIGH ymax
    movwf FSR0H
    movlw LOW ymax
    addwf byte, w
    movwf FSR0L
    btfsc STATUS, C
    incf FSR0H

    movf bit2, w
    andwf INDF0, w

    btfsc STATUS, Z
    goto ymxcnt ; if 0, don't continue

;ymax(n) = ymax(n) AND 1(n) >= 2(n)

    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 tmp1

    movlw HIGH line2
    addwf getxh, w
    movwf FSR0H

    movlw LOW line2
    addwf getxl, w
    btfsc STATUS, C
    incf FSR0H
    movwf FSR0L

    movf INDF0, w

    movwf tmp2

```

```

;check tmp2 < tmp1
;tmp1 - tmp2 >0
movf tmp2, w
subwf tmp1, w

btfsc STATUS, C
goto ymxcnt ;positive - keep it = 1

movlw HIGH ymax
movwf FSR0H
movlw LOW ymax
addwf byte, w
movwf FSR0L
btfsc STATUS, C
incf FSR0H

movf bit2, w
xorwf INDF0; this will set it to 0

ymxcnt      incf getxl
            btfsc STATUS, Z
            incf getxh

            btfss getxh, 0
            goto ymaxupdt

            movf getxl, w
            xorlw 0x40 ;3E maybe

            btfss STATUS, Z
            goto ymaxupdt

            return

suppression
            movf getxl, w
            movwf tmp1
            movf getxh, w
            movwf tmp2

            incf getxl
            btfsc STATUS, Z
            incf getxh

            call bits

            movf tmp1, w
            movwf getxl
            movf tmp2, w
            movwf getxh

            movlw HIGH ymax

```

```
movwf FSR0H
movlw LOW ymax
addwf byte, w
movwf FSR0L
btfsc STATUS, C
incf FSR0H
```

```
clrf vmx
movf bit2, w
andwf INDF0, w
btfss STATUS, Z
setf vmx
```

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

```
movf INDF0, w
movwf k1a
```

```
;movwf n1l
```

```
;call thresh
;call outedge
```

```
incf FSR0L
btfsc STATUS, Z
incf FSR0H
```

```
movf INDF0, w
movwf k1b
```

```
incf FSR0L
btfsc STATUS, Z
incf FSR0H
```

```
movf INDF0, w
movwf k1c
```

```
;FOR HORIZONTAL MAXIMA
; k1b > k1a
; k1b >= k1c
; k1b - k1a > 0
; k1c - k1b <= 0
```

```
clrf hmx
```

```
movf k1b, w
```

```

subwf k1a, w
btfss STATUS, C
setf hmx

clrf k1d
movf k1c, w
subwf k1b, w
btfsc STATUS, C
setf k1d

movf k1d, w
andwf hmx, f

movlw HIGH hgrv
movwf FSR0H
movlw LOW hgrv
addwf byte, w
movwf FSR0L
btfsc STATUS, C
incf FSR0H

setf hgv ; set? clear? confusing
movf bit2, w
andwf INDF0, w
btfss STATUS, Z
clrf hgv

clrf k1e

movf vmx, w
andwf hmx, w
btfss STATUS, Z
setf k1e ;edge pixel
nop

movf vmx, w
andwf hgv, w
btfss STATUS, Z
setf k1e ;horizontal edge pixel

movlw 255
xorwf hgv, f

movf hmx, w
andwf hgv, w
btfss STATUS, Z
setf k1e ;vertical edge pixel

btfss k1e, 0
clrf k1b ; suppress
nop

; movf hgv, w

```

```

;    movwf n1l
;
;    call thresh
;    call outedge

    movf k1b, w
    movwf n1l

    call thresh
    call outedge

    incf getxl
    btfsc STATUS, Z
    incf getxh

    btfss getxh, 0
    goto suppression

    movf getxl, w
    xorlw 0x40 ;3E maybe

    btfss STATUS, Z
    goto suppression

    incf y

    return

ymaximum
    call bits

;ymax(n) = 2(n) > 1(n)

    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 tmp1

    movlw HIGH line2
    addwf getxh, w
    movwf FSR0H

    movlw LOW line2
    addwf getxl, w

```

```

    btfsc STATUS, C
    incf FSR0H
    movwf FSR0L

    movf INDF0, w

    movwf tmp2

    ;tmp1 - tmp2 <= 0

    clrf k1a

    movf tmp2, w
    subwf tmp1, w

    btfss STATUS, C
    setf k1a ; >=0

    movlw HIGH ymax
    movwf FSR0H
    movlw LOW ymax
    addwf byte, w
    movwf FSR0L
    btfsc STATUS, C
    incf FSR0H

    btfsc bit2, 0
    clrf INDF0

    movf bit2, w
    andwf k1a, w
    iorwf INDF0

    incf getxl
    btfsc STATUS, Z
    incf getxh

    btfss getxh, 0
    goto ymaximum

    movf getxl, w
    xorlw 0x40 ;3E maybe

    btfss STATUS, Z
    goto ymaximum

    return

shiftln clrf getxh
        clrf getxl

        call shifthgrv

```

```

movlw LOW 960
movwf pointl
movlw HIGH 960
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

```

```

        btfsc STATUS, Z
        incf d1h

sline1 movf d1h, w
        xorlw 4
        btfss STATUS, Z
        goto sline2
        movf d1l, w
        xorlw 192
        btfsc STATUS, Z
        goto finish
sline2 goto for2
finish return

shifthgrv

        movlw HIGH hgrv2
        movwf FSR0H
        movlw LOW hgrv2
        addwf getxl, w
        movwf FSR0L
        btfsc STATUS, C
        incf FSR0H

        movf INDF0, w
        movwf k1a

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

        movf k1a, w
        movwf INDF0

        incf getxl

        movf getxl, w
        xorlw 40 ;3E maybe

        btfss STATUS, Z
        goto shifthgrv

        return

thresh movf n1l, w
        subwf t, w; if neg then w > t
        movlw 0
        btfss STATUS, C

```

```

movlw 1
movwf n1l ; it is above the threshold - make it 1, otherwise 0

retlw 0

outedge
bcf PORTC, 4
bcf PORTC, 2 ; set output to 1 or 0 depending on ktemp
btfsc n1l, 0
bsf PORTC, 2
bsf PORTC, 0 ; handshake pulse start
btfss PORTC, 1 ; wait for recieved pulse
goto $ - 2
btfsc PORTC, 3 ; restart frame?
call rf ; orders from a lower layer must be obeyed
bcf PORTC, 0
btfsc PORTC, 1 ; wait for end of recieve pulse
goto $ - 2

retlw 0

end

```