

PROGRAM LISTING 8: MPASM VECTOR FINDING PROGRAM

By Malcolm Stagg

[This program is currently untested]

```

;
;|-----|
;|      Vector Controller (VC)      |
;|      [Source Code]                |#
;|-----|#
;|      By Malcolm Stagg            |#
;|-----|#
;| -Designed for the Science Fair-  |#
;|-----|#
;| Used to change a signal from a   |#
;| EC to simple vector lines.       |#
;| Please see circuit diagram for   |#
;| connection information.          |#
;|-----|#
;|#####|
;
; <-----[PIN OUTS]----->
;
; PORTA.0 - HANDSHAKE1 OUT -O
; PORTA.1 - HANDSHAKE1 IN  -I
; PORTA.2 - EDGE IN       -I
; PORTA.3 - SHIFT REG A&B -O
; PORTA.4 - SHIFT REG CP  -O
; PORTA.5 - SHIFT REG /WR -O
;
; PORTB.0 - DOUT8          -O
; PORTB.1 - HANDSHAKE2 OUT -O
; PORTB.2 - HANDSHAKE2 IN  -I
; PORTB.3 - LINE POINT    -O
; PORTB.4 - RIGHT/LEFT    -I/O
; PORTB.5 - X/Y           -O
; PORTB.6 - NEW FRAME1    -I
; PORTB.7 - RESTART FRAME1 -O
;
; PORTC.0-7 - DQ0-7       -I/O
;
; PORTD.0-7 - DOUT0-7     -O
;
; PORTE.0 - RAM /CE       -O
; PORTE.1 - RAM /WE       -O
; PORTE.2 - RAM /OE       -O
;
; <-----[PROGRAM]----->
; LIST P=18F452, R=DEC
; INCLUDE "p18f452.inc"
;
; CBLOCK 0x020
;     ramout, rampx
;     rambit, bitval
;     addressl, addressh
;     _addressl, _addressh
;     shifttimes

```

```

again, co, cr
x, xh, y
_x, _xh, _y
x2, x2h, y2
x2old, x2oldh, y2old
x3, x3h, y3
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x_, y_, z
nota, notb, notc, notd, note, notf, notg, noth, noti, notj, notk, notl,
notm, notn, noto, notp, notq, notr, nots, nott, notu, notv, notw, notx_, noty_, notz
temp
lines, _lines, good, chordout
a0, a1, a2, negc, zeroc
t1neg, t1zero, t2neg, t2zero
b1, b2, c1, c2, pru, prh, prl
dxh, dxl, dxneg, dxz
dy, dyneg, dyz
npt, pts
dist
xp1, xp2, xpneg
yp1, yp2, ypneg
mina1, mina2
minb1, minb2
line1x, line1xh, line1y
line2x, line2xh, line2y
linex, linexh, liney
lastxst, lastxsth, lastyst
linexst, linexsth, lineyst
statush, dthresh
stuck, dirs
cornernum, clp, corners, nocorner
minimum, n2
dmax
t1, t2
ENDC

;   __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON

    __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

```

```

;
;| External RAM Organization |
;|-----| #
;|0000 to 37FF: Image 1    | #
;|3800 to 6FFF: Image 2    | #
;|7000 to 75FF: Corners   | #
;|7600 to 7FFF: Feats     | #
;|8000 to 85FF: Lines     | #
;|-----| #
;| ##### |

```

nop

;I/O setup

```

movlw 6
movwf ADCON1 ; set all pins to digital
movlw b'00000110'
movwf TRISA
;clrf PORTA
movlw b'01111101'
movwf TRISB
movlw b'00000000'
movwf TRISC
movlw b'11111111'
movwf TRISD
movlw b'00000000'
movwf TRISE

```

```

movlw 25
movwf dthresh

```

```

movlw 8
movwf rampx

```

```

movlw 1
movwf rambit

```

```

bsf PORTA, 5
bsf PORTA, 4

```

```

bsf PORTE, 0
bsf PORTE, 1
bsf PORTE, 2

```

;The following is a RAM test:

```

nop
movlw 0x9F
movwf addressl
movlw 0x1F
movwf addressh
movlw 0x0C3
movwf ramout
call writebyte

```

```
movlw 0x0D2
movwf ramout
call writebyte
movlw 0xEF
movwf addressl
movlw 5
movwf addressh
movlw 0x0C3
movwf ramout
call writebyte
```

```
movlw 0x9F
movwf addressl
movlw 0x1F
movwf addressh
call readbyte
nop ; breakpoint
;End of test
```

```
clrf addressl
clrf addressh
```

```
bcf PORTA, 0
clrf PORTB
```

```
chkfrm  btfsc PORTB, 6 ; there is going to be a new frame?
        nop ;call nf ; yes
        btfsc PORTA, 1 ; EC's going to send a pixel?
        call chkpx ; yes
        ;call loadram ; HACK
```

```
goto chkfrm
```

```
chkpx  bcf STATUS, C ; keep pixels in order - ramout.0 is lsb
        btfsc PORTA, 2
        bsf STATUS, C
        rrcf ramout, f ; left to right - MSB to LSB
```

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

```
decfsz rampx
return
```

```
incf x
```

```
movf x, w
xorlw 40
btfss STATUS, Z
goto $ + 8
clrf x
```

```

    incf y

    clrf temp

    movf x, w
    btfss STATUS, Z
    goto $ + 8
    movlw b'00000111'
    andwf ramout

    movf x, w
    xorlw 39
    btfss STATUS, Z
    goto $ + 8
    movlw b'11100000'
    andwf ramout

    movlw 5
    subwf y, w
    btfss STATUS, C
    clrf ramout

    movf y, w
    sublw 235
    btfss STATUS, C
    clrf ramout

    call loadram
    movlw 8
    movwf rampx
    return

```

```
loadram ;call testing1
```

```

    call writebyte

    call bkupadr

    movf addressh, w
    addlw 38 ; REMEMBER ONLY ADDRESSH IS INCREASED BY 38!
    movwf addressh

    call writebyte

    call bkupad2

    incf addressl
    btfsc STATUS, Z
    incf addressh

    movf addressh, w
    xorlw 35
    btfss STATUS, Z

```

```

    return
    movf addressl, w
    btfss STATUS, Z
    return

; done loading imageA (0) and imageB (9728)!
    clrf addressl
    clrf addressh
    clrf rambit

    call filter1
    call vector
    call outedge

    return

filter1
    clrf x
    clrf xh
    clrf y

    clrf addressl
    clrf addressh
    movlw 1
    movwf rambit

flp1b
    call gethm

    movf h, w
    iorwf m, w
    btfsc STATUS, Z
    goto flp1c
    call getbigkern
    movf m, w
    andwf nota, w
    andwf notb, w
    andwf notc, w
    andwf notd, w
    andwf note, w
    andwf notf, w
    andwf notj, w
    andwf notk, w
    andwf noto, w
    andwf notp, w
    andwf nott, w
    andwf notu, w
    andwf notv, w
    andwf notw, w
    andwf notx_, w
    andwf noty_, w
    btfss STATUS, Z
    call clearmid

```

```

movf notg, w
andwf noth, w
andwf noti, w
andwf notl, w
andwf notn, w
andwf notq, w
andwf notr, w
andwf nots, w
andwf m, w
btfss STATUS, Z
    call clearmid
movf g, w
andwf m, w
btfsc STATUS, Z
goto fif2
    movf notc, w
    iorwf notd, w
    iorwf noti, w
    movwf t1

    movf noti, w
    iorwf notn, w
    movwf t2

    movf notb, w
    iorwf notc, w

    andwf notl, w
    andwf t1, w
    andwf t2, w
    btfsc STATUS, Z
    goto $ + 12
    call setxym1
    goto flp1c

    movf notq, w
    iorwf notp, w
    iorwf notk, w
    movwf t1

    movf notk, w
    iorwf notf, w
    movwf t2

    movf notr, w
    iorwf notq, w

    andwf noth, w
    andwf t1, w
    andwf t2, w
    btfss STATUS, Z
    call setxm1y
    goto flp1c

```

```

fif2  movf l, w
      andwf h, w
      btfsc STATUS, Z
      goto flp1c
      movf notf, w
      iorwf nota, w
      iorwf notb, w
      movwf t1

      movf notb, w
      iorwf notc, w
      movwf t2

      movf notk, w
      iorwf notf, w

      andwf notm, w
      andwf t1, w
      andwf t2, w
      btfsc STATUS, Z
      goto $ + 12
      call setxm1ym1
      goto flp1c

      movf notn, w
      iorwf nots, w
      iorwf notr, w
      movwf t1

      movf notr, w
      iorwf notq, w
      movwf t2

      movf noti, w
      iorwf notn, w

      andwf notg, w
      andwf t1
      andwf t2
      btfss STATUS, Z
      call setxy

flp1c call incpx
      incf x
      btfsc STATUS, Z
      incf xh

      movf x, w
      xorlw LOW 315
      btfss STATUS, Z
      goto flp1b

```

```

    movf xh, w
    xorlw HIGH 315
    btfss STATUS, Z
    goto flp1b

    clrf xh
    clrf x

    call incpx
    call incpx
    call incpx
    call incpx
    call incpx

    incf y
    movf y, w
    xorlw 231
    btfss STATUS, Z
    goto flp1b

    return

vector clrf corners
       clrf lines
       clrf good

       movlw 1
       movwf x
       clrf xh
       movwf y

       clrf addressl
       clrf addressh
       movlw 0x01
       movwf rambit
       call incpx
       call inc320px
vloop1 call decpx
       call dec320px
       call getkern

       btfss e, 0
       goto vloop1b ; = 0, skip to vloop1b

       call connected
       ; skip if co = 0, 2

       movf co, w
       btfsc STATUS, Z
       goto vloop1b ; = 0, skip to vloop1b

       xorlw 2

```

```

    btfss STATUS, Z
    call addcorn

vloop1b
    call incpx
    call inc320px

    call incpx
    incf x
    btfsc STATUS, Z ; 2 byte x variable
    incf xh

    movf x, w
    xorlw LOW 318
    btfss STATUS, Z
    goto vloop1 ; return to loop start if not = 313

    movf xh, w
    xorlw HIGH 318 ; check high byte too
    btfss STATUS, Z
    goto vloop1

    clrf xh
    movlw 1
    movwf x

    call incpx
    call incpx
    call incpx

    incf y
    movf y, w
    xorlw 238
    btfss STATUS, Z
    goto vloop1 ; check y

    clrf lines

vecst
    clrf good

    movf corners
    btfsc STATUS, Z
    goto vclp2b
    movwf clp

vclp
    movf clp, w
    movwf cornernum
    call corner

    movf x2, w
    iorwf y2, w

```

```

    btfsc STATUS, Z
    goto vclp2
    call xy_to_adr
    call getkern2 ; use X2 and Y2 in the kernel
    call connected
    movf co, w
    btfsc STATUS, Z
    goto vclp2
    movf x2h, w
    movwf xh
    movf x2, w
    movwf x
    movf y2, w
    movwf y
    setf good
    goto vnxt

vclp2  decfsz clp
       goto vclp

       ; no corner found

vclp2b
    clrf x
    clrf xh
    clrf y

    clrf addressl ; reset variables
    clrf addressh
    movlw 0x01
    movwf rambit

vloop2
    call readbit

    btfss bitval, 0
    goto vloop2b
    call addcorn
    setf good
    goto vnxt

vloop2b call incpx
        incf x
        btfsc STATUS, Z ; 2 byte variable
        incf xh

        movf x, w
        xorlw LOW 318
        btfss STATUS, Z
        goto vloop2 ; not = 317, go back

        movf xh, w
        xorlw HIGH 318
        btfss STATUS, Z ; check high byte too

```

```

goto vloop2

clrf xh
clrf x

call incpx
call incpx

incf y
movf y, w
xorlw 238
btfss STATUS, Z
goto vloop2

vnxt
btfsc good, 0
goto vnxt2
call vector3
return ; if not good now, exit sub
vnxt2

clrf bitval
call writebit ; pnt(x,y)=0
clrf pts

call bkupadr

movlw 76 ; start of feats() array - 76*256=19456 - 3bytes per x/y pixel
movwf addressh ; feats(0,0)=x, feats(1,0)=y
clrf addressl

call feats

call bkupad2

ve    clrf good

xym1  call dec320px ; /\
      call readbit ; |
      call inc320px

      btfss bitval, 0
      goto xm1y
      movf 2, w
      movwf good

      decf y

      call chord

      incf y

      btfsc chordout, 0

```

```

goto xm1y
call dec320px
decf y
setf good
call bkupadr
movlw 76
movwf addressh
movlw 3
mulwf pts
incf pts
movf PRODH, w
addwf addressh
movf PRODL, w
movwf addressl
call feats
call bkupad2

xm1y  btfsc good, 0
      goto xp1y
      call decpx ; <--
      call readbit
      call incpx

      btfss bitval, 0
      goto xp1y

      movlw 2
      movwf good

      decf x ; CHECK LATER!!!
      btfss STATUS, C
      decf xh

      call chord

      incf x
      btfsc STATUS, C
      incf xh

      btfsc chordout, 0
      goto xp1y
      call decpx
      decf x ; CHECK LATER
      btfss STATUS, C
      decf xh
      setf good
      call bkupadr
      movlw 76
      movwf addressh
      movlw 3
      mulwf pts
      incf pts
      movf PRODH, w

```

```

        addwf addressh
        movf PRODL, w
        movwf addressl
        call feats
        call bkupad2

xp1y   btfsc good, 0
        goto xyp1
        call incpx ; -->
        call readbit
        call decpx

        btfss bitval, 0
        goto xyp1

        movlw 2
        movwf good

        incf x
        btfsc STATUS, C
        incf xh

        call chord

        decf x ; CHECK LATER
        btfss STATUS, C
        decf xh

        btfsc chordout, 0
        goto xyp1
        call incpx
        incf x
        btfsc STATUS, Z
        incf xh
        setf good
        call bkupadr
        movlw 76
        movwf addressh
        movlw 3
        mulwf pts
        incf pts
        movf PRODH, w
        addwf addressh
        movf PRODL, w
        movwf addressl
        call feats
        call bkupad2

xyp1   btfsc good, 0
        goto ndvec
        call inc320px ; |
        call readbit ; \|/
        call dec320px

```

```

    btfss bitval, 0
    goto ndvec

    movlw 2
    movwf good

    incf y

    call chord

    decf y

    btfsc chordout, 0
    goto ndvec

    call inc320px
    incf y
    setf good
    call bkupadr
    movlw 76
    movwf addressh
    movlw 3
    mulwf pts
    incf pts
    movf PRODH, w
    addwf addressh
    movf PRODL, w
    movwf addressl
    call feats
    call bkupad2

ndvec
    clrf bitval
    call writebit ; pnt(x,y)=0

    btfss good, 0
    goto ngoodbk

    movf corners
    btfsc STATUS, Z
    goto ngoodbk
    movwf clp

vclpb
    movf clp, w
    movwf cornernum
    call corner

    movf x2, w
    iorwf x2h, w
    btfsc STATUS, Z
    goto vclp3b

```

```
movf y2, w
btfsc STATUS, Z
goto vclp3b
```

```
movf xh, w
xorwf x2h, f
movf x, w
xorwf x2, f
movf y, w
xorwf y2, f
movf x2, w
iorwf x2h, w
iorwf y2, w
btfss STATUS, Z
goto vclp3b
movlw 2
movwf good
```

```
vclp3b decfsz clp
goto vclpb
```

```
ngoodbk btfsc good, 7
goto x2y2bk2
```

```
call readbit
btfss bitval, 0
goto x2y2bk2
call decpx
call dec320px
call getkern2
call inc320px
call incpx
call connected
movf co, w
xorlw 1
btfss STATUS, Z
goto x2y2bk2
movlw 5
subwf y, w
movwf y2
```

```
movlw 5
subwf x, w
movwf x2
btfsc STATUS, C
decf x2h
```

```
clrf n
clrf n2
```

```
movf x2h, w
movwf x2oldh
```

```
movf x2, w
movwf x2old
```

```
movf y2, w
movwf y2old
```

x2y2bk

```
call x2y2_to_adr
call readbit
btfss bitval, 0
goto x2y2bk2
movf x, w
xorwf x2, w
movwf x3
movf y, w
xorwf y2, w
iorwf x3, w ; ??? does this work? 0 if (x2 != x | y2 != y)
```

```
btfsc STATUS, Z
goto x2y2bk2
clrf good
call decpx
call dec320px
call getkern2
call inc320px
call incpx
call connected
movf co, w
xorlw 1
btfss STATUS, Z
goto x2y2bk2
call remcorn
call remcorn2
clrf bitval
call writebit
setf good
call x2y2_to_adr
clrf bitval
call writebit
goto ngoodbk
```

```
incf x2
btfsc STATUS, Z
incf x2h
```

```
incf n
```

```
movf n, w
xorlw 10
btfsc STATUS, Z
goto x2y2bk
```

```
clrf n
```

```
    movf x2oldh, w
    movwf x2h

    movf x2old, w
    movwf x2

    incf n2

    incf y2

    movf n2, w
    xorlw 10
    btfsc STATUS, Z
    goto x2y2bk
```

```
x2y2bk2  clrf bitval
         call writebit
         movf good, w
         btfsc STATUS, Z
         goto vecst
         clrf pts
```

```
         call bkupadr
```

```
         movlw 76 ; start of feats() array - 76*256=19456 - 3bytes per x/y pixel
         movwf addressh ; feats(0,0)=x, feats(1,0)=y
         clrf addressl
```

```
         call feats
```

```
         call bkupad2
```

```
         goto ve;endvec
```

```
         clrf bitval
         call writebit
         goto ve
```

```
         return
```

```
vector3
    call ramtrans ; pnt(x,y)=pnt2(x,y)
    movlw 1
    movwf lines

    call findstcorn
```

```
vec3_do
    call updtidir
    clrf bitval
    call writebit2
```

```
movf dirs, w
xorlw 1
btfsc STATUS, Z
decf y
```

```
movf dirs, w
xorlw 2
btfsc STATUS, Z
incf x
btfss STATUS, Z
incf xh
```

```
movf dirs, w
xorlw 3
btfsc STATUS, Z
incf y
```

```
movf dirs, w
xorlw 4
btfsc STATUS, Z
decf x
btfsc STATUS, C ; CHECK LATER
decf xh
```

```
clrf bitval
call writebit2
```

```
call cornerchk
```

```
clrf bitval
call writebit2
```

v3gobk

```
btfss stuck, 0
goto ndv3
call dec320px
call decpx
call getkern
call incpx
call inc320px
call connected
movf co, w
btfss STATUS, Z
goto ndv3
```

```
movlw 5
subwf y, w
movwf y2
```

```
movlw 5
subwf x, w
movwf x2
```

```
    btfsc STATUS, C
    decf x2h
```

```
    clrf n
    clrf n2
```

```
    movf x2h, w
    movwf x2oldh
```

```
    movf x2, w
    movwf x2old
```

```
    movf y2, w
    movwf y2old
```

v3x2y2bk

```
    call x2y2_to_adr
    call readbit2
    btfss bitval, 0
    goto ndv3
```

```
    call getkern2
    call connected
```

```
    movf co, w
    xorlw 1
    btfsc STATUS, Z
    goto ndv3
```

```
    clrf stuck
    movf x2, w
    movwf x
    movf x2h, w
    movwf xh
    movf y2, w
    movwf y
```

```
    goto v3gobk
```

```
    incf x2
    btfsc STATUS, Z
    incf x2h
```

```
    incf n
```

```
    movf n, w
    xorlw 10
    btfsc STATUS, Z
    goto v3x2y2bk
```

```
    clrf n
```

```
    movf x2oldh, w
```

```

    movwf x2h

    movf x2old, w
    movwf x2

    incf n2

    incf y2

    movf n2, w
    xorlw 10
    btfsc STATUS, Z
    goto v3x2y2bk

ndv3  btfsc stuck, 0
      call findstcorn

      btfss stuck, 0
      goto vec3_do

      return

bkupadr movf addressl, w
        movwf _addressl
        movf addressh, w
        movwf _addressh
        return

bkupad2 movf _addressl, w
        movwf addressl
        movf _addressh, w
        movwf addressh
        return

feats
    movf xh, w
    movwf ramout
    call writebyte

    incf addressl
    btfsc STATUS, Z
    incf addressh

    movf x, w
    movwf ramout
    call writebyte

    incf addressl
    btfsc STATUS, Z
    incf addressh

    movf y, w
    movwf ramout

```

```

    call writebyte

    incf addressl
    btfsc STATUS, Z
    incf addressh

    return

featsb
    movf _xh, w
    movwf ramout
    call writebyte

    incf addressl
    btfsc STATUS, Z
    incf addressh

    movf _x, w
    movwf ramout
    call writebyte

    incf addressl
    btfsc STATUS, Z
    incf addressh

    movf _y, w
    movwf ramout
    call writebyte

    incf addressl
    btfsc STATUS, Z
    incf addressh

    return

featsr
    call readbyte
    movf ramout, w
    movwf _xh

    incf addressl
    btfsc STATUS, Z
    incf addressh

    call readbyte
    movf ramout, w
    movwf _x

    incf addressl
    btfsc STATUS, Z
    incf addressh

    call readbyte

```

```

    movf ramout, w
    movwf _y

    incf addressl
    btfsc STATUS, Z
    incf addressh

    return

ramtrans
    ;image b to image a

    movlw 128
    movwf addressl
    movlw 37
    movwf addressh

rtloop
    call bkupadr
    movlw 38
    addwf addressh, f

    call readbyte

    call bkupad2

    call writebyte

    decf addressl
    btfsc STATUS, C
    goto rtloop

    decf addressh
    btfsc STATUS, C
    goto rtloop

    return

getkern
    call readbit
    movf bitval, w
    movwf a

    call incpx
    call readbit
    movf bitval, w
    movwf b

    call incpx
    call readbit
    movf bitval, w
    movwf c

```

```
call inc320px
call readbit
movf bitval, w
movwf f
```

```
call inc320px
call readbit
movf bitval, w
movwf i
```

```
call decpx
call readbit
movf bitval, w
movwf h
```

```
call decpx
call readbit
movf bitval, w
movwf g
```

```
call dec320px
call readbit
movf bitval, w
movwf d
```

```
call incpx
call readbit
movf bitval, w
movwf e
```

```
call dec320px
call decpx
```

```
return
```

```
getkern2
```

```
call readbit2
movf bitval, w
movwf a
```

```
call incpx
call readbit2
movf bitval, w
movwf b
```

```
call incpx
call readbit2
movf bitval, w
movwf c
```

```
call inc320px
call readbit2
movf bitval, w
```

```
movwf f
```

```
call inc320px  
call readbit2  
movf bitval, w  
movwf i
```

```
call decpx  
call readbit2  
movf bitval, w  
movwf h
```

```
call decpx  
call readbit2  
movf bitval, w  
movwf g
```

```
call dec320px  
call readbit2  
movf bitval, w  
movwf d
```

```
call incpx  
call readbit2  
movf bitval, w  
movwf e
```

```
call dec320px  
call decpx
```

```
return
```

```
connected
```

```
clrf co
```

```
btfsc b, 0  
incf co  
btfsc d, 0  
incf co  
btfsc f, 0  
incf co  
btfsc h, 0  
incf co
```

```
return
```

```
crossing
```

```
clrf cr
```

```
movf a, w  
xorwf b, w  
btfss STATUS, Z
```

```

    incf cr

    movf b, w
    xorwf c, w
    btfss STATUS, Z
    incf cr

    movf c, w
    xorwf f, w
    btfss STATUS, Z
    incf cr

    movf f, w
    xorwf i, w
    btfss STATUS, Z
    incf cr

    movf i, w
    xorwf h, w
    btfss STATUS, Z
    incf cr

    movf h, w
    xorwf g, w
    btfss STATUS, Z
    incf cr

    movf g, w
    xorwf d, w
    btfss STATUS, Z
    incf cr

    movf d, w
    xorwf a, w
    btfss STATUS, Z
    incf cr

    rrrncf cr

    return

inc320px
    movf addressl, w
    addlw 40 ; 320/8
    movwf addressl
    btfsc STATUS, C
    incf addressh
    return

dec320px
    movlw 40
    subwf addressl, f
    btfss STATUS, C

```

```

    decf addressh
    return

incpx  rlncf rambit
       btfss rambit, 0
       return
       incf addressl
       btfsc STATUS, Z
       incf addressh
       return

decpx  rrncf rambit
       btfss rambit, 7
       return
       decf addressl
       btfss STATUS, C
       decf addressh
       return

writebit
; A very inefficient but interesting way of loading a bit into RAM
call readbyte ; first find what is loaded at the current byte

    movf ramout, w
    xorlw 255
    iorwf rambit, w
    xorlw 255 ; set this bit to 0

    btfsc bitval, 0
    iorwf rambit, w ; set to 1 if bit is 1

    movwf ramout

    call writebyte

    return

writebit2
call bkupadr

    movf addressh, w
    addlw 38
    movwf addressh

; A very inefficient but interesting way of loading a bit into RAM
call readbyte ; first find what is loaded at the current byte

    movf ramout, w
    xorlw 255
    iorwf rambit, w
    xorlw 255 ; set this bit to 0

    btfsc bitval, 0

```

```

iorwf rambit, w ; set to 1 if bit is 1

movwf ramout

call writebyte

call bkupad2

return

readbit
call readbyte ; first find what is loaded at the current byte

clrf bitval

movf ramout, w
andwf rambit, w

btfss STATUS, Z
setf bitval ; if bit is 1, bitval = 255

return

readbit2
call bkupadr

movf addressh, w
addlw 38
movwf addressh

call readbyte ; first find what is loaded at the current byte

clrf bitval

movf ramout, w
andwf rambit, w

btfss STATUS, Z
setf bitval ; if bit is 1, bitval = 255

call bkupad2

return

writebyte
call loadaddr

movf ramout, w
movwf PORTC

; data is now loaded!

bcf PORTE, 0

```

```

    bcf PORTE, 1
    nop
    bsf PORTE, 1
    bsf PORTE, 0

    ; data is now in RAM!

    return

readbyte
    call loadaddr

    clrf PORTC

    movlw b'11111111'
    movwf TRISC

    ; data input is now ready!

    bcf PORTE, 0
    bcf PORTE, 2
    nop
    movf PORTC, w
    movwf ramout
    bsf PORTE, 2
    bsf PORTE, 0

    movlw b'00000000'
    movwf TRISC ; set to output again

    ; data is now in ramout!

    return

loadaddr
    bcf PORTA, 5
    bsf PORTA, 5 ; reset shift registers for address
    movlw 8
    movwf shifttimes
    bcf PORTA, 3
    bcf PORTA, 4
    bsf PORTA, 4

shfth  bcf PORTA, 4

    bcf PORTA, 3
    btfsc addressh, 7
    bsf PORTA, 3

    bsf PORTA, 4

    rlnf addressh, f

```

```

    decfsz shifttimes
    goto shfth

    movlw 8
    movwf shifttimes

shftl bcf PORTA, 4

    bcf PORTA, 3
    btfsc addressl, 7
    bsf PORTA, 3

    bsf PORTA, 4

    rlncl addressl, f

    decfsz shifttimes
    goto shftl

    ; address1 is now loaded!
    return

nf    bsf PORTA, 0 ; ok - i'm ready for you to send it

    btfsc PORTB, 6 ; wait for EC to finish pulse
    goto $ - 2

    bcf PORTA, 0

    movlw 8
    movwf rampx
    clrf addressl
    clrf addressh

    return

divide clrf c1
    clrf c2

d1    movf c1, w
    mulwf b1
    movf PRODL, w
    movwf pru

    movf c1, w
    mulwf b2
    movf PRODH, w
    addwf pru
    movf PRODL, w
    movwf prh

    movf c2, w
    mulwf b1

```

```

movf PRODH, w
addwf pru
movf PRODL, w
addwf prh

btfsc STATUS, C ; check
incf pru

movf b2, w
mulwf c2
movf PRODH, w
addwf prh
btfsc STATUS, C
incf pru
movf PRODL, w
movwf prl
btfsc STATUS, C
incf prh
btfsc STATUS, C
incf pru

movf pru, w
subwf a0, w; w = a0 - pru
btfsc STATUS, Z ; if correct, w <= 0 ; z=1 or c=0
goto term2
btfss STATUS, C
goto d2

movf c1, w
xorlw 255
btfss STATUS, Z
goto inc12
movf c2, w
xorlw 255
btfsc STATUS, Z
return

inc12  incfsz c2
      decf c1
      incf c1
      goto d1

term2
movf prh, w
subwf a1, w; w = a1 - prh
btfsc STATUS, Z ; if correct, w <= 0 ; z=1 or c=0
goto term3
btfss STATUS, C
goto d2

incfsz c2
decf c1
incf c1

```

```

        goto d1
term3
    movf prl, w
    subwf a2, w; w = a2 - prl
    btfsc STATUS, Z ; if correct, w <= 0 ; z=1 or c=0
        goto d2
    btfss STATUS, C
        goto d2

    incfsz c2
    decf c1
    incf c1
    goto d1

d2    return

multiply
m1    movf a1, w
        mulwf b1
        movf PRODL, w
        movwf pru

        movf a1, w
        mulwf b2
        movf PRODH, w
        addwf pru
        movf PRODL, w
        movwf prh

        movf a2, w
        mulwf b1
        movf PRODH, w
        addwf pru
        movf PRODL, w
        addwf prh

        btfsc STATUS, C
            incf pru

        movf a2, w
        mulwf b2
        movf PRODH, w
        addwf prh
        btfsc STATUS, C
            incf pru
        movf PRODL, w
        movwf prl
        btfsc STATUS, C
            incf prh
        btfsc STATUS, C
            incf pru

```

```

    return

subtract
    ; c1:c2 = a1:a2 - b1:b2
    clrf negc
    clrf zeroc
    clrf t1neg
    clrf t2neg
    clrf t1zero
    clrf t2zero

    movf b1, w
    subwf a1, w
    movwf c1

    btfss STATUS, C ; c set if +
    setf t1neg
    btfsc STATUS, Z
    setf t1zero

    movf b2, w
    subwf a2, w
    movwf c2

    btfss STATUS, C
    setf t2neg
    btfsc STATUS, Z
    setf t2zero

    btfsc t2neg, 0
    decf c1

    clrf t1zero
    movf c1, w
    btfsc STATUS, Z
    setf t1zero

    movf t1zero
    andwf t2neg
    iorwf t1neg
    movwf negc

    btfss negc, 0
    return
    movlw 255
    xorwf c1

    xorwf c2
    incf c2

    return

```

chord

```

call bkupadr

movlw 76 ; start of feats() array - 76*256=19456 - 3bytes per x/y pixel
movwf addressh
clrf addressl
call featsr

movf _xh, w
movwf a1
movf _x, w
movwf a2

movf xh, w
movwf b1
movf x, w
movwf b2

call subtract

movf c1, w
movwf dxh
movf c2, w
movwf dxl
movf negc, w
movwf dxneg

clrf dyneg
movf y, w
subwf _y, w ; w = _y - y
movwf dy
btfsc STATUS, C
goto $ + 14 ; !!![ CHECK $+14 ]!!!
setf dyneg
movf dy, w
xorlw 255
movwf dy
incf dy

clrf dmax

movf pts, w
btfss STATUS, Z
goto $ + 12
call bkupad2
clrf chordout
return

movwf npt

up  decf npt

movlw 76 ; start of feats() array - 76*256=19456 - 3bytes per x/y pixel
movwf addressh

```

```
movlw 3
mulwf npt
movf PRODH, w
addwf addressh
movf PRODL, w
movwf addressl
call featsr
```

```
movf dxh, w
btfss STATUS, Z
goto chkdy0
```

```
movf dxl, w
btfss STATUS, Z
goto chkdy0
clrf a0
movf _xh, w
movwf a1
movf _x, w
movwf a2
movf xh, w
movwf b1
movf x, w
movwf b2
call subtract
movf c1, w
btfss STATUS, Z
setf c2
movf c2, w
movwf dist
goto chkd
```

```
chkdy0 movf dy, w
btfss STATUS, Z
goto chkelse
movf y, w
subwf _y, w
movwf dist
btfsc STATUS, C
goto chkd
xorlw 255
movwf dist
incf dist
goto chkd
```

```
chkelse clrf negc
```

```
movf y, w
subwf _y, w
btfsc STATUS, C
goto $ + 10
xorlw 255
addlw 1
```

```

    setf negc

    clrf a1
    movwf a2
    movf dxh, w
    movwf b1
    movf dxl, w
    movwf b2
    call multiply
    movf pru, w
    movwf a0
    movf prh, w
    movwf a1
    movf prl, w
    movwf a2
    clrf b1
    movf dy, w
    movwf b2
    call divide

    movf negc, w
    xorwf dxneg, w
    xorwf dyneg, w
    movwf xpneg

    btfsc xpneg, 0
    goto xpn

    movf c2, w
    addwf x, w
    movwf c2
    btfsc STATUS, C
    incf c1
    goto xpin

xpn  clrf b1
     movf x, w
     movwf b2
     movf c1, w
     movwf a1
     movf c2, w
     movwf a2
     call subtract
     movf negc, w
     xorlw 255
     xorwf xpneg

xpin movf c1, w
     movwf xp1
     movf c2, w
     movwf xp2

     movf _xh, w

```

```
movwf a1
movf _x, w
movwf a2
movf xh, w
movwf b1
movf x, w
movwf b2
call subtract
```

```
movf negc, w
xorwf dxneg, w
xorwf dyneg, w
movwf ypneg
```

```
movf c1, w
movwf a1
movf c2, w
movwf a2
clrf b1
movf dy, w
movwf b2
call multiply
movf pru, w
movwf a0
movf prh, w
movwf a1
movf prl, w
movwf a2
movf dxh, w
movwf b1
movf dxl, w
movwf b2
call divide
btfsc ypneg, 0
goto ypn
```

```
movf c2, w
addwf y, w
movwf c2
btfsc STATUS, C
incf c1
goto ypfin
```

```
ypn  clrf b1
      movf y, w
      movwf b2
      movf c1, w
      movwf a1
      movf c2, w
      movwf a2
      call subtract
      movf negc, w
      xorwf ypneg
```

```

ypfin  movf c1, w
        movwf yp1
        movf c2, w
        movwf yp2

        btfsc xpneg, 0
        goto xpn2

        clrf a1
        movf _x, w
        movwf a2
        movf xp1, w
        movwf b1
        movf xp2, w
        movwf b2
        call subtract
        movf c1, w
        movwf mina1
        movf c2, w
        movwf mina2
        goto ypp

xpn2   movf xp1, w
        movwf mina1
        movf _x, w
        addwf xp2, w
        movwf mina2
        btfsc STATUS, C
        incf mina1

ypp    btfsc ypneg, 0
        goto ypn2
        clrf a1
        movf _y, w
        movwf a2
        movf yp1, w
        movwf b1
        movf yp2, w
        movwf b2
        call subtract
        movf c1, w
        movwf minb1
        movf c2, w
        movwf minb2
        goto endyp

ypn2   movf yp1, w
        movwf minb1
        movf _y, w
        addwf yp2, w
        movwf minb2
        btfsc STATUS, C

```

```

    incf minb1

endyp  movf mina1, w
      movwf a1
      movf mina2, w
      movwf a2
      movf minb1, w
      movwf b1
      movf minb2, w
      movwf b2
      call subtract

      movf mina1, w
      btfss STATUS, Z
      setf mina2

      movf minb1, w
      btfss STATUS, Z
      setf minb2

      movf minb2, w
      movwf dist

      movf mina2, w
      btfsc negc, 0
      movwf dist

chkd  movf dthresh, w
      subwf dist, w ; w = dist - 5
      btfss STATUS, C ; if negative, do stuff
      goto goup

      setf chordout
      call bkupad2
      return

goup  incf npt
      decfsz npt
      goto up

      clr f chordout
      call bkupad2
      return

outedge
      movf lines, w
      movwf _lines ; keep a backup of the line (may be used in future updates)
      ;decf lines

oe1  decf lines
      movlw 0x80
      movwf addressh
      movlw 6

```

```
mulwf lines
movf PRODH, w
addwf addressh
movf PRODL, w
movwf addressl
call featsr
call bkupad2
```

```
movf _xh, w
movwf line1xh
movf _x, w
movwf line1x
movf _y, w
movwf line1y
```

```
movlw 0x80
movwf addressh
movlw 6
mulwf lines
movf PRODH, w
addwf addressh
movf PRODL, w
addlw 3
movwf addressl
btfsc STATUS, C
incf addressh
call featsr
call bkupad2
```

```
movf _xh, w
movwf line2xh
movf _x, w
movwf line2x
movf _y, w
movwf line2y
```

```
call outvec
```

```
incf lines
```

```
decfsz lines ; carry, not 0
goto oe1
```

```
setf line1xh
setf line1x
setf line1y
setf line2xh
setf line2x
setf line2y
```

```
call outvec
```

```
return
```

```

outvec btfss PORTB, 4
      goto $ - 2
      bcf TRISB, 3
      bcf TRISB, 5
      clrf TRISD
      bcf TRISB, 0
      bcf TRISB, 1
      ;set to output when asked for this unit to send data

      bcf PORTB, 3
      bcf PORTB, 5

      movf line1x, w
      movwf PORTD
      bcf PORTB, 0
      btfsc line1xh, 0
      bsf PORTB, 0

      bsf PORTB, 1 ; handshake pulse start

      btfss PORTB, 2 ; wait for recieved pulse
      goto $ - 2

      bcf PORTB, 1

      btfsc PORTB, 2 ; wait for end of recieve pulse
      goto $ - 2

      bcf PORTB, 3
      bsf PORTB, 5

      movf line1y, w
      movwf PORTD
      bcf PORTB, 0

      bsf PORTB, 1 ; handshake pulse start

      btfss PORTB, 2 ; wait for recieved pulse
      goto $ - 2

      bcf PORTB, 1

      btfsc PORTB, 2 ; wait for end of recieve pulse
      goto $ - 2

      bsf PORTB, 3
      bcf PORTB, 5

      movf line2x, w
      movwf PORTD
      bcf PORTB, 0
      btfsc line2xh, 0

```

```

    bsf PORTB, 0

    bsf PORTB, 1 ; handshake pulse start

    btfss PORTB, 2 ; wait for recieved pulse
    goto $ - 2

    bcf PORTB, 1

    btfsc PORTB, 2 ; wait for end of recieve pulse
    goto $ - 2

    bsf PORTB, 3
    bsf PORTB, 5

    movf line2y, w
    movwf PORTD
    bcf PORTB, 0

    bsf PORTB, 1 ; handshake pulse start

    btfss PORTB, 2 ; wait for recieved pulse
    goto $ - 2

    bcf PORTB, 1

    btfsc PORTB, 2 ; wait for end of recieve pulse
    goto $ - 2

    bsf TRISB, 3
    bsf TRISB, 5
    setf TRISD
    bsf TRISB, 0
    bsf TRISB, 1
    ;reset to input
    return

```

findstcorn

```

    movf corners
    btfsc STATUS, Z
    return
    movwf clp

```

vclpc

```

    movf clp, w
    movwf cornernum
    call corner

```

```

    movf x2, w
    movwf x

```

```

    movf x2h, w
    movwf xh

```

```

movf y2, w
movwf y

setf stuck

call dec320px
call readbit2
btfss bitval, 0
goto $ + 10
  clrf stuck
  movlw 1
  movwf dirs
  call inc320px

call incpx
call readbit2
btfss bitval, 0
goto $ + 10
  clrf stuck
  movlw 2
  movwf dirs
  call decpx

call inc320px
call readbit2
btfss bitval, 0
goto $ + 10
  clrf stuck
  movlw 1
  movwf dirs
  call dec320px

call decpx
call readbit2
btfss bitval, 0
goto $ + 10
  clrf stuck
  movlw 1
  movwf dirs
  call incpx

movf xh, w
iorwf x, w
btfsc STATUS, Z
  setf stuck
iorwf y, w
btfsc STATUS, Z
  setf stuck

movf xh, w
movwf linexh
movf x, w

```

```

        movwf linex
        movf y, w
        movwf liney

        call setline_st

        btfss stuck, 0
        return

vclp2c  decfsz clp
        goto vclpc

        return

cornerchk
        setf nocorner

        movf corners
        btfsc STATUS, Z
        return
        movwf clp

        call corner

vclpe  movf clp, w
        movwf cornernum
        call corner

        movf x2h, w
        xorwf xh, w
        btfss STATUS, Z
        goto vclp2e
        movf x2, w
        xorwf x, w
        andlw 0xFD
        btfss STATUS, Z
        goto vclp2e
        movf y2, w
        xorwf y, w
        andlw 0xFD ; include everything but low 2 bits
        btfss STATUS, Z
        goto vclp2e
        ;If linex(0, lines) <> x2 Or liney(0, lines) <> y2 Then
        movf linexsth, w
        xorwf x2h, w
        movwf x2h
        btfss x2h, 0
        goto vclp2e
        movf linexst, w
        xorwf x2, w
        movwf x2
        movf lastyst, w
        xorwf y2, w

```

```
iorwf x2, w
btfsc STATUS, Z
goto vclp2e
movf x2h
movwf linexh
movf x2
movwf linex
movf y2
movwf liney
call setline_end
incf lines
call setline_st
```

```
clrf nocorner
setf stuck
```

```
vclp2e decfsz clp
goto vclpe
```

```
return
```

```
addcorn
clrf minimum
```

```
movf corners
btfsc STATUS, Z
goto vclp2ee
movwf clp
```

```
vclpd
movf clp, w
movwf cornernum
call corner
```

```
movf x2h, w
xorwf xh, w
btfss STATUS, Z
goto vclp2d
movf x2, w
xorwf x, w
andlw 0xFD
btfss STATUS, Z
goto vclp2d
movf y2, w
xorwf y, w
andlw 0xFD
btfsc STATUS, Z
setf minimum
```

```
vclp2d decfsz clp
goto vclpd
```

```
btfsc minimum, 0
return
```

```

vclp2ee incf corners
        call setcorner
        return

;      include "testimg.asm"

getbigkern
        call readbit
        movf bitval, w
        movwf a
        xorlw 255
        movwf nota
        call incpx
        call readbit
        movf bitval, w
        movwf b
        xorlw 255
        movwf notb
        call incpx
        call readbit
        movf bitval, w
        movwf c
        xorlw 255
        movwf notc
        call incpx
        call readbit
        movf bitval, w
        movwf d
        xorlw 255
        movwf notd
        call incpx
        call readbit
        movf bitval, w
        movwf e
        xorlw 255
        movwf note
        call inc320px
        call readbit
        movf bitval, w
        movwf j
        xorlw 255
        movwf notj
        call inc320px
        call readbit
        movf bitval, w
        movwf o
        xorlw 255
        movwf noto
        call inc320px
        call readbit
        movf bitval, w
        movwf t
        xorlw 255

```

```
movwf nott
call inc320px
call readbit
movf bitval, w
movwf y_
xorlw 255
movwf noty_
call decpx
call readbit
movf bitval, w
movwf x_
xorlw 255
movwf notx_
call decpx
call readbit
movf bitval, w
movwf w
xorlw 255
movwf notw
call decpx
call readbit
movf bitval, w
movwf v
xorlw 255
movwf notv
call decpx
call readbit
movf bitval, w
movwf u
xorlw 255
movwf notu
call dec320px
call readbit
movf bitval, w
movwf p
xorlw 255
movwf notp
call dec320px
call readbit
movf bitval, w
movwf k
xorlw 255
movwf notk
call dec320px
call readbit
movf bitval, w
movwf f
xorlw 255
movwf notf
call incpx
call readbit
movf bitval, w
movwf g
```

```
xorlw 255
movwf notg
call incpx
call readbit
movf bitval, w
movwf h
xorlw 255
movwf noth
call incpx
call readbit
movf bitval, w
movwf i
xorlw 255
movwf noti
call inc320px
call readbit
movf bitval, w
movwf n
xorlw 255
movwf notn
call inc320px
call readbit
movf bitval, w
movwf s
xorlw 255
movwf nots
call decpx
call readbit
movf bitval, w
movwf r
xorlw 255
movwf notr
call decpx
call readbit
movf bitval, w
movwf q
xorlw 255
movwf notq
call dec320px
call readbit
movf bitval, w
movwf l
xorlw 255
movwf notl
call incpx
call readbit
movf bitval, w
movwf m
xorlw 255
movwf notm
call dec320px
call dec320px
call decpx
```

```

    call decpx
    return

gethm  call incpx
       call incpx
       call inc320px
       call readbit
       movf bitval, w
       movwf h
       xorlw 255
       movwf noth
       call inc320px
       call readbit
       movf bitval, w
       movwf m
       xorlw 255
       movwf notm
       call dec320px
       call dec320px
       call decpx
       call decpx
       return

remcorn movf corners, w
        btfsc STATUS, Z
        return
        movwf clp

remcorn1
        movf clp, w
        movwf cornernum
        call corner
        movf x2h, w
        xorwf xh, w
        btfss STATUS, Z
        goto remcorn1b
        movf x2, w
        xorwf x
        btfss STATUS, Z
        goto remcorn1b
        movf y2, w
        xorwf y
        btfss STATUS, Z
        goto remcorn1b
        clrf x2h
        clrf x2
        clrf y2
        call setcorner2
remcorn1b
        decfsz clp
        goto remcorn1
        return

```

```

remcorn2
    movf x2h, w
    movwf x3h
    movf x2, w
    movwf x3
    movf y2, w
    movwf y3

    movf corners, w
    btfsc STATUS, Z
    return
    movwf clp

remcorn2a
    movf clp, w
    movwf cornernum
    call corner
    movf x2h, w
    xorwf x3h, w
    btfss STATUS, Z
    goto remcorn2b
    movf x2, w
    xorwf x3
    btfss STATUS, Z
    goto remcorn2b
    movf y2, w
    xorwf y3
    btfss STATUS, Z
    goto remcorn2b
    clrf x2h
    clrf x2
    clrf y2
    call setcorner2

remcorn2b
    decfsz clp
    goto remcorn2a

    return

corner
    call bkupadr
    movlw 0x70
    movwf addressh
    movlw 3
    mulwf cornernum
    movf PRODH, w
    addwf addressh
    movf PRODL, w
    movwf addressl
    call featsr
    movf _xh, w
    movwf x2h
    movf _x, w

```

```
movwf x2
movf _y, w
movwf y2
call bkupad2
```

```
return
```

```
setcorner
```

```
call bkupadr
movlw 0x70
movwf addressh
movlw 3
mulwf cornernum
movf PRODH, w
addwf addressh
movf PRODL, w
movwf addressl
call feats
call bkupad2
return
```

```
setcorner2
```

```
call bkupadr
movlw 0x70
movwf addressh
movlw 3
mulwf cornernum
movf PRODH, w
addwf addressh
movf PRODL, w
movwf addressl
movf x2h, w
movwf _xh
movf x2, w
movwf _x
movf y2, w
movwf _y
call featsb
call bkupad2
return
```

```
setline_st
```

```
call bkupadr
movlw 0x80
movwf addressh
movlw 6
mulwf lines
movf PRODL, w
movwf addressl
btfsc STATUS, C
incf addressh
movf PRODH, w
addwf addressh
```

```
movf linexh, w
movwf ramout
call writebyte
incf addressl
btfsc STATUS, Z
    incf addressh
movf linex, w
movwf ramout
call writebyte
incf addressl
btfsc STATUS, Z
    incf addressh
movf liney, w
movwf ramout
call writebyte

call bkupad2
return
```

```
setline_end
    call bkupadr
    movlw 0x80
    movwf addressh
    movlw 6
    mulwf lines
    movf PRODL, w
    addlw 3
    movwf addressl
    btfsc STATUS, C
        incf addressh
    movf PRODH, w
    addwf addressh
    movf linexh, w
    movwf ramout
    call writebyte
    incf addressl
    btfsc STATUS, Z
        incf addressh
    movf linex, w
    movwf ramout
    call writebyte
    incf addressl
    btfsc STATUS, Z
        incf addressh
    movf liney, w
    movwf ramout
    call writebyte

    call bkupad2
    return
```

```
setxm1y
```

```

    call incpx
    call inc320px
    call inc320px
    setf bitval
    call writebit
    call writebit2
    call decpx
    call dec320px
    call dec320px
    return

setxm1ym1
    call incpx
    call inc320px
    setf bitval
    call writebit
    call writebit2
    call decpx
    call dec320px
    return

setxy
    call incpx
    call incpx
    call inc320px
    call inc320px
    setf bitval
    call writebit
    call writebit2
    call decpx
    call decpx
    call dec320px
    call dec320px
    return

setxym1
    call incpx
    call incpx
    call inc320px
    setf bitval
    call writebit
    call writebit2
    call decpx
    call decpx
    call dec320px
    return

updtDir
    setf stuck
    call dec320px
    call readbit2
    btfss bitval, 0
    goto updtrt

```

```

        clrf stuck
        movlw 1
        movwf dirs
updtrt  call inc320px

        call incpx
        call readbit2
        btfss bitval, 0
        goto updttn
        clrf stuck
        movlw 2
        movwf dirs
updttn  call decpx

        call inc320px
        call readbit2
        btfss bitval, 0
        goto updtlf
        clrf stuck
        movlw 3
        movwf dirs
updtlf  call dec320px

        call decpx
        call readbit2
        btfss bitval, 0
        goto uptdone
        clrf stuck
        movlw 4
        movwf dirs
uptdone call incpx

        movf xh, w
        iorwf x, w
        btfsc STATUS, Z
        setf stuck
        movf y, w
        btfsc STATUS, Z
        setf stuck
        return

xy_to_adr
        movlw 160
        mulwf y
        bcf STATUS, C
        rlcw PRODL
        rlcw PRODH
        movf PRODH, w
        movwf addressh

```

```
movf PRODL, w
movwf addressl
movf xh, w
addwf addressh
movf x, w
addwf addressl
btfsc STATUS, C
incf addressh
return
```

x2y2_to_adr

```
movlw 160
mulwf y2
bcf STATUS, C
rlcf PRODL
rlcf PRODH
movf PRODH, w
movwf addressh
movf PRODL, w
movwf addressl
movf x2h, w
addwf addressh
movf x2, w
addwf addressl
btfsc STATUS, C
incf addressh
return
```

clearmid

```
call inc320px
call incpx
clrf bitval
call writebit
call writebit2
call incpx
clrf bitval
call writebit
call writebit2
call incpx
clrf bitval
call writebit
call writebit2
```

```
call inc320px
clrf bitval
call writebit
call writebit2
call inc320px
clrf bitval
call writebit
call writebit2
```

```
call decpx
```

```
clrf bitval  
call writebit  
call writebit2  
call decpx  
clrf bitval  
call writebit  
call writebit2
```

```
call dec320px  
clrf bitval  
call writebit  
call writebit2
```

```
call incpx  
clrf bitval  
call writebit  
call writebit2
```

```
call dec320px  
call dec320px  
call decpx  
call decpx
```

```
return
```

```
end
```