

PROGRAM LISTING 9: MPASM 3D CONVERSION PROGRAM

By Malcolm Stagg

[This program has not been updated to new calculations, and is currently untested]

```
;
;|-----|
;| Three Dimensional Controller (3DC) |#
;|           [Source Code]           |#
;|-----|
;|           By Malcolm Stagg        |#
;|           |                       |#
;| -Designed for the Science Fairs-  |#
;|           |                       |#
;| Used to interface convert 2D      |#
;| lines to 3D lines. Please see     |#
;| circuit diagram for connection    |#
;| information.                      |#
;|-----|
;| #####|
;
; <----- [PIN OUTS] ----->
;
; PORTA.0 - DIN8           -I
; PORTA.1 - DOUT8          -O/I
; PORTA.2 - X/Y2           -O
; PORTA.3 - XY/Z2          -O
; PORTA.4 - HSI2           -I
; PORTA.5 - HSO2           -O
;
; PORTB.0-7 - DOUT0-7      -O/I
;
; PORTC.0-7 - DIN0-7       -I
;
; PORTD.0 - LNPT           -I
; PORTD.1 - X/Y            -I
; PORTD.2 - HSI            -I
; PORTD.3 - HSO            -O
; PORTD.4 - RT/LF          -O
; PORTD.5 - LNPT2          -O
; PORTD.6 - S1             -I Inc Left Stepper
; PORTD.7 - S2             -I Inc Right Stepper
;
; PORTE.0 - BUSY           -I   Math Co-Processor
; PORTE.1 - SERIAL I/O     -I/O
; PORTE.2 - SERIAL CLOCK   -O
;
; <----- [PROGRAM] ----->
; LIST P=18F452, R=DEC
; INCLUDE "p18f452.inc"
;
; CBLOCK 0x020
;   s, o1, o2, x1, x2, y1, y2, x1b, x2b
;   xiii1, yiii1, ziii1
;   xiii1b, yiii1b, ziii1b
;   vax, va2x, va3x, va4x, va5x
```

```

    va6x, va7x, va8x, va9x, va10x, vaxs, vax2s, vays
    vax2, va2x2, va3x2, va4x2, va5x2
    va6x2, va7x2, va8x2, va9x2, va10x2
    vay, va2y, va3y, va4y, va5y
    va6y, va7y, va8y, va9y, va10y
    vbx, vb2x, vb3x, vb4x, vb5x
    vb6x, vb7x, vb8x, vb9x, vb10x, vbxs, vbx2s, vbys, vbzs
    vbx2, vb2x2, vb3x2, vb4x2, vb5x2
    vb6x2, vb7x2, vb8x2, vb9x2, vb10x2
    vby, vb2y, vb3y, vb4y, vb5y, done, var
    e, e2, sr, sr2, sr1o4, yc, d, mathout, for, times ;, for2
ENDC

;    __CONFIG_CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_ON

    org 0

    nop

;I/O setup
    movlw b'00010001'
    movwf TRISA
    movlw b'00000000'
    movwf TRISB
    movlw b'11111111'
    movwf TRISC
    movlw b'11000111'
    movwf TRISD
    movlw b'00000011'
    movwf TRISE

    movlw 100 ; distance between camera center and origin in mm
    movwf s
    movlw 85
    movwf yc ; camera height in mm
    movlw 20
    movwf d ; lens to axis distance in mm
    movlw 90
    movwf o1
    movwf o2 ; O1 and O2 are the stepper angles - start at 90, progressing
positively CCW (0-180)
    movlw 158
    movwf e
    movlw 1 ; 1 = 256, 2 = 512, 4 = 1024, 8 = 2048, 16 = 4096, 32 = 8192, 64
= 16384, 128 = 32768
    movwf e2 ; 1 in the high bit, and 158 in the low bit make 414
    movlw 144
    movwf sr
    movlw 1
    movwf sr2 ; 1 in the high bit, and 144 in the low bit make 400
steps/revolution
    movlw 100
    movwf sr1o4 ; steps / rev divided by 4

```

```

pakpaus macro  n
    movlw n
    movwf times
    decfsz times
    goto $ - 1
endm

```

```

getvert      btfsc PORTB, 2 ; have you started yet?
            goto $ - 1
            btfsc PORTB, 4 ; 0 for left, 1 for right
            call right
            btfss PORTB, 4
            call left
            btfss PORTB, 5 ; done transmitting vertices?
            goto $ + 5 ; yup
            incf done
            btfss done, 1 ; is it done left & right?
            goto $ + 2
            call calc ; yes - continue calculations
            goto getvert ; repeat

```

```

left  movf PORTC, w
      incf vaxs
      call lsto1
      movf PORTB, w
      incf vax2s
      call lsto2
      movf PORTD, w
      incf vays
      call lsto3
      return

```

```

lsto1  movwf var
      rlnsf vaxs ; *2
      movf vaxs, w
      rlnsf vaxs ; /2
      addwf PCL ; add vbxs to the current address to choose where to save
      nop
      movf var, w ; [1] - it doesn't matter that it is also saved in future addresses...
      movwf vax
      movf var, w ; [2]
      movwf vax + 1
      movf var, w ; [3]
      movwf vax + 2
      movf var, w ; [4]
      movwf vax + 3
      movf var, w ; [5]
      movwf vax + 4
      movf var, w ; [6]
      movwf vax + 5
      movf var, w ; [7]
      movwf vax + 6

```

```
movf var, w ; [8]
movwf vax + 7
movf var, w ; [9]
movwf vax + 8
movf var, w ; [10]
movwf vax + 9
return
```

```
lsto2  movwf var
      rlncl vax2s ; *2
      movf vaxs, w
      rrrncf vbx2s ; /2
      addwf PCL ; add vbx to the current address to choose where to save
      nop
      movf var, w ; [1] - it doesn't matter that it is also saved in future addresses...
      movwf vax2
      movf var, w ; [2]
      movwf vax2 + 1
      movf var, w ; [3]
      movwf vax2 + 2
      movf var, w ; [4]
      movwf vax2 + 3
      movf var, w ; [5]
      movwf vax2 + 4
      movf var, w ; [6]
      movwf vax2 + 5
      movf var, w ; [7]
      movwf vax2 + 6
      movf var, w ; [8]
      movwf vax2 + 7
      movf var, w ; [9]
      movwf vax2 + 8
      movf var, w ; [10]
      movwf vax2 + 9
      return
```

```
lsto3  movwf var
      rlncl vays ; *2
      movf vaxs, w
      rrrncf vbys ; /2
      addwf PCL ; add vbx to the current address to choose where to save
      nop
      movf var, w ; [1] - it doesn't matter that it is also saved in future addresses...
      movwf vay
      movf var, w ; [2]
      movwf vay + 1
      movf var, w ; [3]
      movwf vay + 2
      movf var, w ; [4]
      movwf vay + 3
      movf var, w ; [5]
      movwf vay + 4
      movf var, w ; [6]
```

```

movwf vay + 5
movf var, w ; [7]
movwf vay + 6
movf var, w ; [8]
movwf vay + 7
movf var, w ; [9]
movwf vay + 8
movf var, w ; [10]
movwf vay + 9
return

right  movf PORTC, w
       incf vbxs
       call rsto1
       movf PORTB, w
       incf vb2s
       call rsto2
       movf PORTD, w
       incf vbys
       call rsto3
       return

rsto1  movwf var
       rlnf vbxs ; *2
       movf vbxs, w
       rrcf vbxs ; /2
       addwf PCL ; add vbxs to the current address to choose where to save
       nop
       movf var, w ; [1] - it doesn't matter that it is also saved in future addresses...
       movwf vbx
       movf var, w ; [2]
       movwf vbx + 1
       movf var, w ; [3]
       movwf vbx + 2
       movf var, w ; [4]
       movwf vbx + 3
       movf var, w ; [5]
       movwf vbx + 4
       movf var, w ; [6]
       movwf vbx + 5
       movf var, w ; [7]
       movwf vbx + 6
       movf var, w ; [8]
       movwf vbx + 7
       movf var, w ; [9]
       movwf vbx + 8
       movf var, w ; [10]
       movwf vbx + 9
       return

rsto2  movwf var
       rlnf vb2s ; *2
       movf vbxs, w

```

```

rrncf vbx2s ; /2
addwf PCL ; add vbx to the current address to choose where to save
nop
movf var, w ; [1] - it doesn't matter that it is also saved in future addresses...
movwf vbx2
movf var, w ; [2]
movwf vbx2 + 1
movf var, w ; [3]
movwf vbx2 + 2
movf var, w ; [4]
movwf vbx2 + 3
movf var, w ; [5]
movwf vbx2 + 4
movf var, w ; [6]
movwf vbx2 + 5
movf var, w ; [7]
movwf vbx2 + 6
movf var, w ; [8]
movwf vbx2 + 7
movf var, w ; [9]
movwf vbx2 + 8
movf var, w ; [10]
movwf vbx2 + 9
return

```

```

rsto3 movwf var
rlncf vbys ; *2
movf vbx, w
rrncf vbys ; /2
addwf PCL ; add vbx to the current address to choose where to save
nop
movf var, w ; [1] - it doesn't matter that it is also saved in future addresses...
movwf vby
movf var, w ; [2]
movwf vby + 1
movf var, w ; [3]
movwf vby + 2
movf var, w ; [4]
movwf vby + 3
movf var, w ; [5]
movwf vby + 4
movf var, w ; [6]
movwf vby + 5
movf var, w ; [7]
movwf vby + 6
movf var, w ; [8]
movwf vby + 7
movf var, w ; [9]
movwf vby + 8
movf var, w ; [10]
movwf vby + 9
return

```

```

busy   btfsc PORTE, 0
        goto busy
        retlw 0

pakxmit movwf mathout
        movlw 8
        movwf for
for1    rlnsf mathout, f ; Most Significant Bit first
        btfsc mathout, 0 ; is it 1?
        bsf PORTE, 1 ; yes
        btfss mathout, 0 ; or 0?
        bcf PORTE, 1 ; yes
        bsf PORTE, 2 ; clock
        pakpaus 18
        bcf PORTE, 2
        pakpaus 18
        decfsz mathout
        goto for1
        pakpaus 16 ; delay before sending next byte
        retlw 0

pakrcv  movlw b'00000011'
        movwf TRISE ; set Data as input
        clrf mathout
        movlw 8
        movwf for
for2    rlnsf mathout, f ; MSB First
        btfsc PORTE, 1 ; is it 1?
        bsf mathout, 0 ; yes
        btfss PORTE, 1 ; or 0?
        bcf mathout, 0 ; yes
        bsf PORTE, 2 ; clock
        pakpaus 18
        bcf PORTE, 2
        pakpaus 18
        decfsz for
        goto for2
        movlw b'00000001'
        movf mathout, w ; return from sub with result in w
        return

pakres  bcf PORTE, 1 ; reset the PAKII
        bcf PORTE, 2
        pakpaus 18
        bsf PORTE, 2
        pakpaus 18
        bsf PORTE, 1
        pakpaus 18
        bcf PORTE, 2
        pakpaus 18
        return

calc    btfss PORTA, 2 ; is the math co being used?

```

```

goto $ - 1 ; yes, wait
bsf PORTB, 6 ; no - set it
nop
nop
nop
nop
nop
nop
bcf PORTB, 6 ; reset to 0
nop
nop
nop
nop
nop
nop
bfsc PORTC, 5 ; wait for pulse end
goto $ - 1
movlw b'00000001' ; Make Data & Clock Outputs
movwf TRISE
call pakres ; reset pak: always a good idea
call mcsetup

calc1 call dox
      call doy
      call doz

      call out

      decfsz vaxs
      goto calc1

      return

out   movlw b'11011100' ; set pins as outputs now
      movwf TRISB
      movlw b'00000000'
      movwf TRISC
      movlw b'00000000'
      movwf TRISD

      movf xiii1, w ; x, y, then z
      movwf PORTC
      bcf PORTB, 0
      bfsc xiii1b, 0
      bsf PORTB, 0
      movf yiii1b, w
      movwf PORTD

      bfsc PORTA, 5 ; step?
      call step

      bsf PORTA, 0 ; handshake

      btfs PORTA, 1
      goto $ - 1

```

```

    btfsc PORTA, 1
    goto $ - 1

    bcf PORTA, 0

    clrf PORTD
    movf ziii1, w ; now z
    movwf PORTC
    bcf PORTB, 0
    btfsc ziii1b, 0
    bsf PORTB, 0

    bsf PORTA, 0 ; handshake

    btfss PORTA, 1
    goto $ - 1
    btfsc PORTA, 1
    goto $ - 1

    bcf PORTA, 0

    return

mcsetup    movlw 0x01 ; load X with o1
           call pakxmit
           clrw
           call pakxmit
           call pakxmit
           call pakxmit
           movf o1
           call pakxmit
           call busy

           call stepper

           movlw 0x12 ; store radians in location 1
           call pakxmit
           movlw 0
           call pakxmit
           call busy

           movlw 0x01 ; load X with o2
           call pakxmit
           clrw
           call pakxmit
           call pakxmit
           call pakxmit
           movf o2
           call pakxmit
           call busy

           call stepper

```

```

movlw 0x12 ; store radians in location 2
call pakxmit
movlw 1
call pakxmit
call busy

call doa
call dob
call doc
call dof
call dog
call doh
call don

return

stepper      movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x04 ; swap x and y
call pakxmit
call busy

movlw 0x01 ; load X with 1/4 steps per rotation
call pakxmit
clrw
call pakxmit
call pakxmit
call pakxmit
movf sr104
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x04 ; swap x and y
call pakxmit
call busy

movlw 0x8E ; subtract - STEPS - 100
call pakxmit
call busy

movlw 0x04 ; swap x and y
call pakxmit
call busy

movlw 0x01 ; load X with sr
call pakxmit
clrw

```

```

call pakxmit
call pakxmit
movf sr2, w
call pakxmit
movf sr, w
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x04 ; swap x and y
call pakxmit
call busy

movlw 0x8D ; divide    - (STEPS - 100) / STEPS_PER_REVOLUTION
call pakxmit
call busy

movlw 0x04 ; swap x and y
call pakxmit
call busy

movlw 0x01 ; load X with 360
call pakxmit
clrw
call pakxmit
call pakxmit
movlw 2 ; + 256
call pakxmit
movlw 104
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x8C ; multiply    - (STEPS - 100) / STEPS_PER_REVOLUTION * 360
call pakxmit
call busy

movlw 0x04 ; swap x and y
call pakxmit
call busy

movlw 0x01 ; load X with pi    - ((STEPS - 100) / STEPS_PER_REVOLUTION
* 360) * PI
call pakxmit ; pi = 80490FDB
movlw 0x80
call pakxmit
movlw 0x49

```

```

call pakxmit
movlw 0x0F
call pakxmit
movlw 0xDB
call pakxmit
call busy

movlw 0x8C ; multiply
call pakxmit
call busy

movlw 0x04 ; swap x and y
call pakxmit
call busy

movlw 0x01 ; load X with 180
call pakxmit
clrw
call pakxmit
call pakxmit
call pakxmit
movlw 180
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x04 ; swap x and y
call pakxmit
call busy

movlw 0x0D ; divide - ((STEPS - 100) / STEPS_PER_REVOLUTION * 360)
* PI / 180
call pakxmit
call busy

return

doa movlw 0x01 ; do A - load X with X1
call pakxmit
clrw
call pakxmit
call pakxmit
call pakxmit
movf x1
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit

```

```

call busy

btfsc x1b, 0 ; subtract 160?
goto $ + 18

movlw 0x02 ; put 160 in Y
call pakxmit
clr
call pakxmit
call pakxmit
call pakxmit
movlw 160
call pakxmit

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x8E ; subtract - x1 = x1 - 160 for center of frame as origin
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x01 ; load X with e
call pakxmit
clr
call pakxmit
call pakxmit
movf e2
call pakxmit
movf e
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x8D ; divide - x1 / e
call pakxmit
call busy

```

```

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x13 ; load O1 radians into X
call pakxmit
movlw 0
call pakxmit
call busy

movlw 0xA1 ; take sine of O1
call pakxmit
call busy

movlw 0x8C ; multiply - (x1 / e) * sin(O1)
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x13 ; load O1 radians into X
call pakxmit
movlw 0
call pakxmit
call busy

movlw 0xA2 ; take cosine of O1
call pakxmit
call busy

movlw 0x8F ; add - ((x1 / e) * sin(O1)) + cos(O1)
call pakxmit
call busy

movlw 0x12 ; save A into 3
call pakxmit
movlw 2
call pakxmit
call busy

return

dob movlw 0x01 ; do B - load X with X2
call pakxmit
clr
call pakxmit
call pakxmit
call pakxmit
movf x2
call pakxmit
call busy

```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
btfscl x2b, 0 ; subtract 160?
goto $ + 18
```

```
movlw 0x02 ; put 160 in Y
call pakxmit
clrw
call pakxmit
call pakxmit
call pakxmit
movlw 160
call pakxmit
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
movlw 0x8E ; subtract - x2 = x2 - 160 for center of frame as origin
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x01 ; load X with e
call pakxmit
clrw
call pakxmit
call pakxmit
movf e2
call pakxmit
movf e
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x8D ; divide - x2 / e
```

```
call pakxmit  
call busy
```

```
movlw 0x04 ; swap  
call pakxmit  
call busy
```

```
movlw 0x13 ; load O2 radians into X  
call pakxmit  
movlw 1  
call pakxmit  
call busy
```

```
movlw 0xA1 ; take sine of O2  
call pakxmit  
call busy
```

```
movlw 0x8C ; multiply - (x2 / e) * sin(O2)  
call pakxmit  
call busy
```

```
movlw 0x04 ; swap X and Y  
call pakxmit  
call busy
```

```
movlw 0x13 ; load O2 radians into X  
call pakxmit  
movlw 1  
call pakxmit  
call busy
```

```
movlw 0xA2 ; take cosine of O2  
call pakxmit  
call busy
```

```
movlw 0x8F ; add - ((x2 / e) * sin(O2)) + cos(O2)  
call pakxmit  
call busy
```

```
movlw 0x12 ; save B into 4  
call pakxmit  
movlw 3  
call pakxmit  
call busy
```

```
return
```

```
doc movlw 0x13 ; load O2 radians into X  
call pakxmit  
movlw 1  
call pakxmit  
call busy
```

```

movlw 0xA2 ; take cosine
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x13 ; load O1 radians into X
call pakxmit
movlw 0
call pakxmit
call busy

movlw 0xA2 ; take cosine
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x8E ; subtract - cos(O2) - cos(O1)
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x01 ; load d into X
call pakxmit
clr
call pakxmit
call pakxmit
call pakxmit
movf d
call pakxmit
call busy

movlw 0x8C ; multiply - d * (cos(O2) - cos(O1))
call pakxmit
call busy

movlw 0x12 ; save C radians into 5
call pakxmit
movlw 4
call pakxmit
call busy

return

dof movlw 0x01 ; do F - load X with X1

```

```
call pakxmit
clr
call pakxmit
call pakxmit
call pakxmit
movf x1
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
btfscl x1b, 0 ; subtract 160?
goto $ + 18
```

```
movlw 0x02 ; put 160 in Y
call pakxmit
clr
call pakxmit
call pakxmit
call pakxmit
movlw 160
call pakxmit
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
movlw 0x8E ; subtract - x1 = x1 - 160 for center of frame as origin
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x01 ; load X with e
call pakxmit
clr
call pakxmit
call pakxmit
movf e2
call pakxmit
movf e
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
```

```

call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x8D ; divide - x1 / e
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x13 ; load O1 radians into X
call pakxmit
movlw 0
call pakxmit
call busy

movlw 0xA2 ; take cosine of O1
call pakxmit
call busy

movlw 0x8C ; multiply - (x1 / e) * cos(O1)
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x13 ; load O1 radians into X
call pakxmit
movlw 0
call pakxmit
call busy

movlw 0xA1 ; take sine of O1
call pakxmit
call busy

movlw 0x8E ; subtract - sin(O1) - ((x1 / e) * cos(O1))
call pakxmit
call busy

movlw 0x12 ; save A into 6
call pakxmit
movlw 5
call pakxmit
call busy

```

```

return

dog  movlw 0x01 ; do G - load X with X2
     call pakxmit
     clrw
     call pakxmit
     call pakxmit
     call pakxmit
     movf x2
     call pakxmit
     call busy

     movlw 0x87 ; floating point convert
     call pakxmit
     call busy

     btfsc x2b, 0 ; subtract 160?
     goto $ + 18

     movlw 0x02 ; put 160 in Y
     call pakxmit
     clrw
     call pakxmit
     call pakxmit
     call pakxmit
     movlw 160
     call pakxmit

     movlw 0x04 ; swap
     call pakxmit
     call busy

     movlw 0x87 ; floating point convert
     call pakxmit
     call busy

     movlw 0x8E ; subtract - x2 = x2 - 160 for center of frame as origin
     call pakxmit
     call busy

     movlw 0x04 ; swap
     call pakxmit
     call busy

     movlw 0x01 ; load X with e
     call pakxmit
     clrw
     call pakxmit
     call pakxmit
     movf e2
     call pakxmit
     movf e
     call pakxmit

```

```

call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x8D ; divide - x2 / e
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x13 ; load O2 radians into X
call pakxmit
movlw 1
call pakxmit
call busy

movlw 0xA2 ; take cosine of O2
call pakxmit
call busy

movlw 0x8C ; multiply - (x2 / e) * cos(O2)
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x13 ; load O2 radians into X
call pakxmit
movlw 1
call pakxmit
call busy

movlw 0xA1 ; take sine of O2
call pakxmit
call busy

movlw 0x8E ; subtract - sin(O2) - ((x2 / e) * cos(O2))
call pakxmit
call busy

movlw 0x12 ; save B into 7
call pakxmit
movlw 6

```

```

    call pakxmit
    call busy

    return

doh  movlw 0x13 ; load O2 radians into X
     call pakxmit
     movlw 1
     call pakxmit
     call busy

     movlw 0xA1 ; take sine
     call pakxmit
     call busy

     movlw 0x04 ; swap X and Y
     call pakxmit
     call busy

     movlw 0x13 ; load O1 radians into X
     call pakxmit
     movlw 0
     call pakxmit
     call busy

     movlw 0xA1 ; take sine
     call pakxmit
     call busy

     movlw 0x04 ; swap X and Y
     call pakxmit
     call busy

     movlw 0x8E ; subtract    - sin(O2) - sin(O1)
     call pakxmit
     call busy

     movlw 0x04 ; swap X and Y
     call pakxmit
     call busy

     movlw 0x01 ; load d into X
     call pakxmit
     clrw
     call pakxmit
     call pakxmit
     call pakxmit
     movf d
     call pakxmit
     call busy

     movlw 0x8C ; multiply    - d * (cos(O2) - cos(O1))
     call pakxmit

```

```

call busy

movlw 0x12 ; save C radians into 8
call pakxmit
movlw 7
call pakxmit
call busy

return

don  movlw 0x01 ; do Z - load X with s
call pakxmit
clr
call pakxmit
call pakxmit
call pakxmit
movf s
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x01 ; load X with 2
call pakxmit
clr
call pakxmit
call pakxmit
call pakxmit
movlw 2
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x8C ; multiply - s * 2
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x13 ; load H (8) into X
call pakxmit
movlw 7

```

call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x8E ; subtract - $s * 2 - h$
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x13 ; load A (3) into X
call pakxmit
movlw 2
call pakxmit
call busy

movlw 0x8C ; multiply - $a * (s * 2 - h)$
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x12 ; save in 11
call pakxmit
movlw 10
call pakxmit
call busy

movlw 0x13 ; load C (5)
call pakxmit
movlw 4
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x13 ; load F (6)
call pakxmit
movlw 5
call pakxmit
call busy

movlw 0x8C ; multiply - $c * f$
call pakxmit

```

call busy

movlw 0x04 ; swap X and Y
call pakxmit
call busy

movlw 0x13 ; load 11
call pakxmit
movlw 10
call pakxmit
call busy

movlw 0x8E ; subtract - (c * f) + (a * (s * 2 - h))
call pakxmit
call busy

movlw 0x12 ; save in 11
call pakxmit
movlw 10
call pakxmit
call busy

movlw 0x13 ; load B (4)
call pakxmit
movlw 3
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x13 ; load F (6)
call pakxmit
movlw 5
call pakxmit
call busy

movlw 0x8C ; multiply - B * F
call pakxmit
call busy

movlw 0x12 ; save memory 10
call pakxmit
movlw 9
call pakxmit
call busy

movlw 0x13 ; load A (3)
call pakxmit
movlw 2
call pakxmit
call busy

```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x13 ; load G (7)
call pakxmit
movlw 6
call pakxmit
call busy
```

```
movlw 0x8C ; multiply - A * G
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x13 ; load memory 10
call pakxmit
movlw 9
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x8C ; subtract - (A * G) - (B * F)
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x13 ; load 11
call pakxmit
movlw 10
call pakxmit
call busy
```

```
movlw 0x8D ; divide - ((c * f) + (a * (s * 2 - h))) / ((A * G) - (B * F))
call pakxmit
call busy
```

```
movlw 0x12 ; save memory 11
call pakxmit
movlw 10
call pakxmit
call busy
```

```

return

doz  movlw 0x13 ; load B (4) into X
     call pakxmit
     movlw 3
     call pakxmit
     call busy

     movlw 0x04 ; swap
     call pakxmit
     call busy

     movlw 0x13 ; load 11 - n
     call pakxmit
     movlw 10
     call pakxmit
     call busy

     movlw 0x8C ; multiply - B * N
     call pakxmit
     call busy

     movlw 0x12 ; save 9
     call pakxmit
     movlw 8
     call pakxmit
     call busy

     movlw 0x13 ; load O2 radians into X
     call pakxmit
     movlw 1
     call pakxmit
     call busy

     movlw 0xA2 ; take cosine
     call pakxmit
     call busy

     movlw 0x04 ; swap
     call pakxmit
     call busy

     movlw 0x01 ; put d in X
     call pakxmit
     clrw
     call pakxmit
     call pakxmit
     call pakxmit
     movf d, w
     call pakxmit
     call busy

```

```

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x8C ; multiply - d * (cos(O2))
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x13 ; load memory 9
call pakxmit
movlw 8
call pakxmit
call busy

movlw 0x8F ; add - (B * N) + (d * cos(O2))
call pakxmit
call busy

movlw 0x0B ; convert to integer
call pakxmit
call busy

movlw 0x03
call pakxmit ; get ziii result
call pakrcv ; ignore exponent
call pakrcv
call pakrcv
movf mathout, w
movwf ziii1b
call pakrcv
movf mathout, w
movwf ziii1
call busy

return

dox movlw 0x13 ; load G (7) into X
call pakxmit
movlw 6
call pakxmit
call busy

movlw 0x0A ; negate - -G
call pakxmit
call busy

movlw 0x04 ; swap X and Y
call pakxmit

```

```

call busy

movlw 0x13 ; load N (11)
call pakxmit
movlw 10
call pakxmit
call busy

movlw 0x8C ; multiply (-G * N)
call pakxmit
call busy

movlw 0x12 ; save 9
call pakxmit
movlw 8
call pakxmit
call busy

movlw 0x13 ; load O2 radians into X
call pakxmit
movlw 1
call pakxmit
call busy

movlw 0xA1 ; take sine
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x01 ; put d in X
call pakxmit
clrw
call pakxmit
call pakxmit
call pakxmit
movf d, w
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x8C ; multiply - d * (sin(O2))
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

```

```

movlw 0x13 ; load memory 9
call pakxmit
movlw 8
call pakxmit
call busy

movlw 0x8E ; subtract - (-G * N) - (d * sin(O2))
call pakxmit
call busy

movlw 0x04 ; swap
call pakxmit
call busy

movlw 0x01 ; load X with s
call pakxmit
clrw
call pakxmit
call pakxmit
call pakxmit
movf s
call pakxmit
call busy

movlw 0x87 ; floating point convert
call pakxmit
call busy

movlw 0x8F ; add - ((-G * N) - (d * sin(O2))) + s
call pakxmit
call busy

movlw 0x0B ; convert to integer
call pakxmit
call busy

movlw 0x03
call pakxmit ; get ziii result
call pakrcv ; ignore exponent
call pakrcv
call pakrcv
movf mathout, w
movwf xiii1b
call pakrcv
movf mathout, w
movwf xiii1
call busy

return

doy movlw 0x01 ; load X with y2
call pakxmit

```

```
clrw
call pakxmit
call pakxmit
call pakxmit
movf y2
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x01 ; load X with 120
call pakxmit
clrw
call pakxmit
call pakxmit
call pakxmit
movlw 120
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x8E ; subtract - y2 = y2 - 120
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x01 ; load X with e
call pakxmit
clrw
call pakxmit
call pakxmit
movf e2
call pakxmit
movf e
call pakxmit
call busy
```

```
movlw 0x87 ; convert to floating point
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x8D ; divide -  $y_2 / e$ 
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x13 ; load memory N (11)
call pakxmit
movlw 10
call pakxmit
call busy
```

```
movlw 0x8C ; multiply -  $(y_2 / e) * N$ 
call pakxmit
call busy
```

```
movlw 0x12 ; store in 9
call pakxmit
movlw 8
call pakxmit
call busy
```

```
movlw 0x01 ; load X with Yc
call pakxmit
clrwl
call pakxmit
call pakxmit
call pakxmit
movf yc
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x01 ; load X with 10
call pakxmit
clrwl
```

```
call pakxmit
call pakxmit
call pakxmit
movlw 10
call pakxmit
call busy
```

```
movlw 0x87 ; floating point convert
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x8D ; divide -  $yc = yc / 10$ 
call pakxmit
call busy
```

```
movlw 0x04 ; swap
call pakxmit
call busy
```

```
movlw 0x13 ; load 9
call pakxmit
movlw 8
call pakxmit
call busy
```

```
movlw 0x8F ; add -  $((y2 / e) * N) + yc$ 
call pakxmit
call busy
```

```
movlw 0x0B ; convert to integer
call pakxmit
call busy
```

```
movlw 0x03
call pakxmit ; get yiii result
call pakrcv ; ignore exponent
call pakrcv
call pakrcv
movf mathout, w
movwf yiii1b
call pakrcv
movf mathout, w
movwf yiii1
call busy
```

```
return
```

```
step bsf PORTA, 0 ; handshake
```

```

btfss PORTA, 1
goto $ - 1
btfsc PORTA, 1
goto $ - 1

bcf PORTA, 0

goto $ + 1 ; delay 10
goto $ + 1
goto $ + 1
goto $ + 1
goto $ + 1

btfsc PORTA, 5 ; still there
incf o1 ; increment first motor

bsf PORTA, 0 ; handshake

btfss PORTA, 1
goto $ - 1
btfsc PORTA, 1
goto $ - 1

bcf PORTA, 0

goto $ + 1 ; delay 10
goto $ + 1
goto $ + 1
goto $ + 1
goto $ + 1

btfsc PORTA, 5 ; still there
decf o1 ; decrement first motor

bsf PORTA, 0 ; handshake

btfss PORTA, 1
goto $ - 1
btfsc PORTA, 1
goto $ - 1

bcf PORTA, 0

goto $ + 1 ; delay 10
goto $ + 1
goto $ + 1
goto $ + 1
goto $ + 1

btfsc PORTA, 5 ; still there
incf o2 ; increment second motor

bsf PORTA, 0 ; handshake

```

```
    btfss PORTA, 1
    goto $ - 1
    btfsc PORTA, 1
    goto $ - 1

    bcf PORTA, 0

    goto $ + 1 ; delay 10
    goto $ + 1
    goto $ + 1
    goto $ + 1
    goto $ + 1

    btfsc PORTA, 5 ; still there
    decf o2 ; decrement second motor

    bsf PORTA, 0 ; handshake

    btfss PORTA, 1
    goto $ - 1
    btfsc PORTA, 1
    goto $ - 1

    bcf PORTA, 0

    goto $ + 1 ; delay 10
    goto $ + 1
    goto $ + 1
    goto $ + 1
    goto $ + 1

    btfsc PORTA, 5 ; still there
    goto step ; yup

    return

    end
```