```
; PICStep v1.01 Firmware - PIC based microstepping motor controller
; Copyright (C) 2004 Alan Garfield <alan@fromorbit.com>

; This program is free software; you can redistribute it and/or
; modify it under the terms of the GNU General Public License
; as published by the Free Software Foundation; either version 2
; of the License, or (at your option) any later version.

; This program is distributed in the hope that it will be useful,
; but WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
; GNU General Public License for more details.

; You should have received a copy of the GNU General Public License
; along with this program; if not, write to the Free Software
; Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.

  TITLE "PICStep V1.01"

  LIST R=DEC
  INCLUDE "p16f628a.inc"

  __CONFIG _CP_OFF & _WDT_ON & _HS_OSC & _PWRTE_ON & _LVP_OFF & _BOREN_ON & _MCLRE_OFF

; Registers
  CBLOCK 0x020
        step
        mode
        timeout_reg
        timeout:2
        temp

        _w
        _status
        _fsr
        _pclath
  ENDC

; Macros

  INC16 MACRO   DST                             ; 16 bit increment macro for use in the timeout
routines
        incfsz  (DST), w
        decf    (DST)+1, f
        incf    (DST)+1, f
        movwf   (DST)
        iorwf   (DST)+1, w
  ENDM

; Mainline Start
        org             0
        goto            Mainline

; Interupt Start
        org             4
Interupt

; Save Current Context
        movwf           _w
        movf            STATUS, w
        bcf             STATUS, RP1
        bcf             STATUS, RP0
        movwf           _status
        movf            FSR, w
        movwf           _fsr
        movf            PCLATH, w
        movwf           _pclath
        clrf            PCLATH

; Interrupt service routine
        btfsc           INTCON, INTF
          call INTB0                            ; Call INTBO interrupt handler
        btfsc           PIR1, TMR2IF
```

```
          call TIMEOUT                    ; Call TIMEOUT interrupt handler

; Reset Current Context
        movf            _pclath, w
        movwf           PCLATH
        movf            _fsr, w
        movwf           FSR
        movf            _status, w
        movwf           STATUS
        swapf           _w, f
        swapf           _w, w

        retfie

INTB0
; Handle interupt on RB0

        clrf            timeout         ; Reset timeout timer and register
        clrf            timeout+1
        clrf            timeout_reg

; Advance the index position

        movf            mode, w         ; Load the current mode
        call            MODE_TABLE      ; Get the advance value for this mode

        movwf           temp            ; Store for later

        btfss           PORTB, 1        ; Check on the direction pin (RB1)
          goto          $ + 4           ; If set jump to dec

        addwf           step, w         ; Add the current position to the current mode value
        movwf           step            ; Update step
        goto            $ + 3

        subwf           step, w         ; Subtract the current position to the current mode
                                        ; value
        movwf           step            ; Update step

; Process stepA
        movlw           32              ; Check if step has overflowed the edge of the table
        subwf           step, w
        btfsc           STATUS, Z
          clrf          step

        movf            temp, w         ; Check if step has underflowed the edge of the
                                        ; table
        sublw           32
        subwf           step, w
        btfsc           STATUS, C
          subwf         step, f

        movf            step, w         ; Reload step into w

        call            STEP_TABLE      ; Get the result from the table
        andlw           B'00001111'     ; Mask out the upper nibble
        movwf           PORTA           ; Output the lower nibble to PORTA

; Process stepB
        movf            step, w         ; Reload step into w
        sublw           7               ; Check if we're greater than position 7 where B
                        rolls over
        btfsc           STATUS, C
          goto          $ + 4

        movlw           8               ; step is > 8 so B has rolled over
        subwf           step, w         ; Subtract 8 from step to get stepB
        goto            $ + 3

        movf            step,w          ; step is < 8 so B hasn't rolled over
        addlw           24                      ; Add 24 to step to get stepB

        call            STEP_TABLE      ; Get the result from the table
```

```
        movwf   temp                    ; Store the result in a temp register ready for
                                        ; rotation
        rlf     temp, f
        rlf     temp, w
        andlw   B'11111100'             ; Mask out the un-needed bits
        movwf   PORTB                   ; Output to PORTB

        bcf     INTCON, INTF            ; Clear RB0 Interrupt flag

        return

TIMEOUT
; Handle motor timeout TMR2 interrupt and return ASAP

        bsf     timeout_reg, 7          ; Set the timeout bit so the count can increment
        bcf     PIR1, TMR2IF            ; Clear TMR2 Interrupt flag

        return

  PAGE

Mainline

; Initialize Variables

        clrf    step
        clrf    mode
        clrf    temp
        clrf    timeout
        clrf    timeout+1
        clrf    timeout_reg

; Setup I/O ports / Timers / Interupts

        clrf    PORTA                   ;Initialize PORTA
        clrf    PORTB                   ;Initialize PORTB

        movlw   (1 << CM0) | (1 << CM1) | (1 << CM2)
        movwf   CMCON                   ; Turn comparators off and enable pins for I/O

        bcf     STATUS, RP1
        bsf     STATUS, RP0             ;Select Bank1

        movlw   B'11110000'             ;Set RA<0:3> as outputs
        movwf   TRISA ^ 0x080

        movlw   B'00000011'             ;Set RB<2:7> as outputs
        movwf   TRISB ^ 0x080

        movlw   (1 << INTEDG)           ;Setup Interupt Edge
        movwf   OPTION_REG ^ 0x080

        movlw   (1 << TMR2IE)           ; Enable TMR2 Interupt
        movwf   PIE1 ^ 0x080

        bcf     STATUS, RP0             ;Select Bank0

        movlw   B'01111111'             ; Setup TMR2 1:16 pre and post scaler and enable
        movwf   T2CON

        movlw   (1 << GIE) | (1 << INTE) | (1 << PEIE) ; Enable global interupts, perph
                                        ; and RB0 Interupts
        movwf   INTCON

Loop

        clrwdt                          ; Clear the watchdog timer (maximum loop for entire
                                        ; code is ~0.18ms watchdog is 18ms plenty of time!)

  ; Monitor mode switches
        btfsc   PORTA, 4
          goto  $ + 3
```

```
        bsf             mode, 0
        goto            $ + 2
        bcf             mode, 0
        btfsc           PORTA, 5
          goto          $ + 3
        bsf             mode, 1
        goto            $ + 2
        bcf             mode, 1

  ; Motor timeout counter
        btfss           timeout_reg, 7          ; Check to see if a timeout interrupt has occured
          goto          Loop

  ; Timeout interrupt occured updated counter

        bcf             timeout_reg, 7          ; Reset the Interrupt flag

        INC16           timeout                 ; Increment timeout value

        movwf           timeout                 ; Test if the timeout value has overflowed
        btfss           STATUS, Z
          goto          Loop
        movwf           timeout+1
        btfss           STATUS, Z
          goto          Loop

        incf            timeout_reg             ; Increase the timeout reg

        btfss           timeout_reg, 2          ; Check we've been around the 4 times of the 16 bit
                                                ; counter (~5 minutes 45 seconds @ 20MHz)
          goto          Loop

  ; Timeout!

        clrf            PORTA                   ; Reset PORTA and PORTB to turn off the motors
        clrf            PORTB                   ; The next INTB0 will awaken them again

        goto            Loop

; 1/8 Step Table
STEP_TABLE
        addwf           PCL, 1
        retlw           B'00011111'             ; 1i    --- A Start     -- 1/4  -- 1/2          -- 1
        retlw           B'00011111'             ; 0.98
        retlw           B'00011110'             ; 0.92                  -- 1/4
        retlw           B'00011101'             ; 0.83
        retlw           B'00011011'             ; 0.70                  -- 1/4  -- 1/2
        retlw           B'00011001'             ; 0.55
        retlw           B'00010110'             ; 0.38                  -- 1/4
        retlw           B'00010011'             ; 0.19
        retlw           B'00000000'             ; 0                     -- 1/4  -- 1/2          -- 1
        retlw           B'00000011'             ; 0.19
        retlw           B'00000110'             ; 0.38                  -- 1/4
        retlw           B'00001001'             ; 0.55
        retlw           B'00001011'             ; 0.70                  -- 1/4  -- 1/2
        retlw           B'00001101'             ; 0.83
        retlw           B'00001110'             ; 0.92                  -- 1/4
        retlw           B'00001111'             ; 0.98
        retlw           B'00101111'             ; 1                     -- 1/4  -- 1/2          -- 1
        retlw           B'00101111'             ; 0.98
        retlw           B'00101110'             ; 0.92                  -- 1/4
        retlw           B'00101101'             ; 0.83
        retlw           B'00101011'             ; 0.70                  -- 1/4  -- 1/2
        retlw           B'00101001'             ; 0.55
        retlw           B'00100110'             ; 0.38                  -- 1/4
        retlw           B'00100011'             ; 0.19
        retlw           B'00110000'             ; 0     --- B Start     -- 1/4  -- 1/2          -- 1
        retlw           B'00110011'             ; 0.19
        retlw           B'00110110'             ; 0.38                  -- 1/4
        retlw           B'00111001'             ; 0.55
        retlw           B'00111011'             ; 0.70                  -- 1/4  -- 1/2
        retlw           B'00111101'             ; 0.83
        retlw           B'00111110'             ; 0.92                  -- 1/4
```

```
        retlw           B'00111111'             ; 0.98


MODE_TABLE
        addwf           PCL, 1
        retlw           0x001
        retlw           0x002
        retlw           0x004
        retlw           0x008

        end
```