

```

1  ;*****
2  ;   This file is a basic code template for assembly code generation   *
3  ;   on the PICmicro PIC16F84A. This file contains the basic code     *
4  ;   building blocks to build upon.                                     *
5  ;                                                                 *
6  ;   If interrupts are not used all code presented between the ORG     *
7  ;   0x004 directive and the label main can be removed. In addition   *
8  ;   the variable assignments for 'w_temp' and 'status_temp' can     *
9  ;   be removed.                                                       *
10 ;                                                                 *
11 ;   Refer to the MPASM User's Guide for additional information on     *
12 ;   features of the assembler (Document DS33014).                     *
13 ;                                                                 *
14 ;   Refer to the respective PICmicro data sheet for additional        *
15 ;   information on the instruction set.                                 *
16 ;                                                                 *
17 ;   Template file assembled with MPLAB V4.00.00 and MPASM V2.20.12.  *
18 ;                                                                 *
19 ;*****
20 ;                                                                 *
21 ;   Filename:      xxx.asm                                             *
22 ;   Date:                                                 *
23 ;   File Version:                                         *
24 ;                                                                 *
25 ;   Author:                                               *
26 ;   Company:                                             *
27 ;                                                                 *
28 ;                                                                 *
29 ;*****
30 ;                                                                 *
31 ;   Files required:                                           *
32 ;                                                                 *
33 ;                                                                 *
34 ;                                                                 *
35 ;*****
36 ;                                                                 *
37 ;   Notes:                                                 *
38 ;                                                                 *
39 ;                                                                 *
40 ;                                                                 *
41 ;                                                                 *
42 ;*****
43
44
45     list      p=16F84A          ; list directive to define processor
46     #include <p16F84A.inc>      ; processor specific variable definitions
47
48     __CONFIG    _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC
49
50 ; '__CONFIG' directive is used to embed configuration data within .asm file.
51 ; The labels following the directive are located in the respective .inc file.
52 ; See respective data sheet for additional information on configuration word.
53
54
55
56
57 ;***** VARIABLE DEFINITIONS
58 w_temp      EQU      0x0C      ; variable used for context saving
59 status_temp EQU      0x0D      ; variable used for context saving
60

```

```

61
62 DigDly      EQU      d'3'      ;Digit delay
63 PassNum     EQU      d'6'      ;Holes/Parts button looked at every
64            ;time main routine finishes this many times
65 Zonk        EQU      d'10'     ;Sets auto shut-off time
66
67
68 TEMP_VAR    UDATA      0x20    ; explicit address specified is not required
69
70 Count0      RES      1          ;Counter, digit 0
71 Count1      RES      1          ;Counter, digit 1
72 Count2      RES      1          ;Counter, digit 2
73 Count3      RES      1          ;Counter, digit 3
74 HoleCnt     RES      1          ;Hole counter
75 HoleSet     RES      1          ;Hole setpoint. Hole counter rolls at this
76 PartInc     RES      1          ;Amount to increment counter after hole cnt
77 CurDigt     RES      1          ;Current digit, 0 - 6
78 PassCnt     RES      1          ;Pass counter, used to control green button
79 Delay0      RES      1          ;Outer delay loop variable
80 Delay1      RES      1          ;Inner delay loop variable
81 SleepLo     RES      1          ;Auto power-off delay count low byte
82 SleepHi     RES      1          ;Auto power-off delay count high byte
83
84
85
86
87
88
89
90 ;*****
91      ORG      0x000            ; processor reset vector
92      goto    main             ; go to beginning of program
93
94
95      ORG      0x004            ; interrupt vector location
96      goto    ISR
97
98
99
100     CODE
101 main
102
103 ; remaining code goes here
104
105     banksel   TRISA
106     movlw    b'11110000'      ;RA2-0: Digit select
107     movwf    TRISA           ;RA3: Sensor power  RA4: Hole/part button
108     movlw    b'00000001'      ;RB7-1: Segment anodes
109
110     movwf    TRISB           ;RB0: Sensor input
111     movlw    0xC7            ;Set option register to disable RB pullups,
112     movwf    OPTION_REG      ;rising edge on INT pin, TMR0 internal
113     banksel   PORTA           ;prescaler to 1:256
114     clrf     PORTA           ;Select digit 0, turn off sensor power
115     clrf     Count0
116     clrf     Count1
117     clrf     Count2
118     clrf     Count3
119     clrf     HoleCnt
120     clrf     PassCnt

```

```

121      clrf      SleepLo
122      clrf      SleepHi
123
124      banksel   EEADR
125      clrf      EEADR      ;Part setpoint in byte 0 of EEPROM
126      banksel   EECON1
127      bsf      EECON1, RD      ;Order an EEPROM read operation
128      banksel   EEDATA
129      movf      EEDATA, w
130      movwf     PartInc
131
132      movlw     0x01      ;Next location has hole setpoint
133      movwf     EEADR
134      banksel   EECON1
135      bsf      EECON1, RD      ;Order an EEPROM read operation
136      banksel   EEDATA
137      movf      EEDATA, w
138      movwf     HoleSet
139
140      banksel   PORTA
141
142      movlw     d'10'      ;Must make sure numbers from EEPROM <10
143      bcf      STATUS, C      ;Clear carry before comparisons
144      subwf     PartInc, w
145      movlw     0x01      ;If bad, will set to a 1
146      btfsc    STATUS, C      ;Carry set means >=10
147      movwf     PartInc      ;Forced to a 1
148
149      movlw     d'10'      ;Must make sure numbers from EEPROM <10
150      bcf      STATUS, C      ;Clear carry before comparisons
151      subwf     HoleSet, w
152      movlw     0x01      ;If bad, will set to a 1
153      btfsc    STATUS, C      ;Carry set means >=10
154      movwf     HoleSet      ;Forced to a 1
155
156      clrf      CurDigt
157      bsf      PORTA, 3      ;Turn on opto-sensor
158      call     Delay      ;Wait a moment for sensor to wake up
159      bsf      INTCON, 7      ;Enable global interrupts
160      bsf      INTCON, 4      ;Enable RB0 external interupt
161
162
163      Loop1    movf      Count0, w
164              call     GoDigit
165
166              movf      Count1, w
167              call     GoDigit
168
169              movf      Count2, w
170              call     GoDigit
171
172              movf      Count3, w
173              call     GoDigit
174
175              movf      HoleSet, w
176              call     GoDigit
177
178              movlw     d'10'      ;Select the '/'
179              call     GoDigit
180
181              movf      PartInc, w

```

```

182      call      GoDigit
183
184      incf      PassCnt,f      ;See if we can look at green button yet
185      movlw    PassNum
186      xorwf    PassCnt,w      ;Z bit set if OK to look at button
187      btfss   STATUS,Z
188      goto     Loop1          ;Display is now updated, no button check
189
190      clrf     PassCnt
191
192      decfsz   SleepLo,f      ;Check for long periods of inactivity
193      goto     BtnChk
194      incf     SleepHi,f      ;Once this equals Zonk, bye-bye!
195      movlw    Zonk
196      xorwf    SleepHi,w      ;Z is set if Zonk time
197      btfss   STATUS,Z
198      goto     BtnChk
199      clrf     PORTB          ;Turn off all segments
200      bcf     INTCON,7        ;No interrupts please-I'm resting
201      bcf     PORTA,3        ;Turn off opto-sensor
202      sleep
203      nop
204      goto     main          ;Hopefully, like a reset
205
206  BtnChk  btfsc   PORTA,4      ;Goes low if green button pressed
207          goto   Loop1        ;Not pressed. Continue normal operation
208          bcf    INTCON,7      ;Mask all interrupts
209          bcf    PORTA,0      ;Zero in on digit 4
210          bcf    PORTA,1
211          bsf    PORTA,2
212
213  Loop2   movf    HoleSet,w    ;Prepare to display hole setpoint
214          call   Segment      ;w now has segment codes
215          movwf  PORTB
216          movlw  0xFF          ;Delay a bit
217          call   Delay
218          call   Delay
219          btfss  PORTB,0      ;Check sensor. If beam broken, up count
220          goto   ChkBtn
221          incf   HoleSet,f
222          movlw  d'10'
223          xorwf  HoleSet,w    ;Z set if hit 10
224          movlw  0x01          ;If roll, roll to 1, not 0
225          btfsc  STATUS,Z
226          movwf  HoleSet
227          goto   Loop2
228  ChkBtn  btfsc   PORTA,4      ;Check green button
229          goto   Loop2        ;Not pressed. Keep checking sensor
230
231
232
233  Loop3   bsf     PORTA,1      ;Move to digit 6 (Part setpoint)
234          movf   PartInc,w    ;Prepare to display part setpoint
235          call   Segment      ;w now has segment codes
236          movwf  PORTB
237          movlw  0xFF          ;Delay a bit
238          call   Delay
239          call   Delay
240          btfss  PORTB,0      ;Check sensor. If beam broken, up count
241          goto   ChkBtn2
242          incf   PartInc,f
243          movlw  d'10'

```

```

243      xorwf      PartInc,w      ;Z set if hit 10
244      movlw     0x01            ;If roll, roll to 1, not 0
245      btfsc     STATUS,Z
246      movwf     PartInc
247      goto      Loop3
248 ChkBtn2 btfsc     PORTA,4      ;Check green button
249      goto      Loop3        ;Not pressed. Keep checking sensor
250
251      clrf      Count0        ;Always clear count when
252      clrf      Count1        ;changing counting parameters
253      clrf      Count2
254      clrf      Count3
255
256
257
257      movf      PartInc,w      ;Get the current part increment value
258      banksel   EEADR
259      clrf      EEADR        ;Always write to location 0
260      movwf     EEDATA        ;We're set. Now trigger write sequence.
261      banksel   EECON1
262      bsf       EECON1,WREN    ;Enable the write
263      movlw     0x55          ;Following required sequence
264      movwf     EECON2
265      movlw     0xAA
266      movwf     EECON2
267      bsf       EECON1,WR     ;Write bit set. Write operation begins.
268      movlw     0x30          ;We'll delay a while
269      call      Delay
270
271      movlw     0x01          ;Hole setpoint in location 1
272      banksel   EEADR
273      movwf     EEADR
274      movf      HoleSet,w     ;Get hole setpoint
275      movwf     EEDATA        ;We're set. Now trigger write sequence.
276      banksel   EECON1
277      bsf       EECON1,WREN    ;Enable the write
278      movlw     0x55          ;Following required sequence
279      movwf     EECON2
280      movlw     0xAA
281      movwf     EECON2
282      bsf       EECON1,WR     ;Write bit set. Write operation begins.
283      movlw     0xF0          ;We'll delay a goodly while
284      call      Delay
285
286      banksel   PORTA
287
288      bsf       INTCON,7      ;Enable interrupts again
289      goto      Loop1        ;Resume normal operation
290
291
292 ;*****End of Main routine*****
293
294
295 Delay  movwf     Delay0        ;Initialize outer delay loop
296      clrf      Delay1        ;Initialize middle delay loop
297
298 Dloop  decfsz    Delay1,f      ;
299      goto      Dloop
300      decfsz    Delay0,f      ;
301      goto      Dloop
302      return
303
;All done. Back to the program

```

```

304 GoDigit call      Segment      ;Convert count to LED segment codes
305         movwf     PORTB        ;Drive segments
306         movlw    DigDly       ;Hold a bit
307         call     Delay        ;
308         call     NextDig      ;Select next digit
309         return
310
311 NextDig  incf      CurDigt,f    ;Point to next digit
312         movlw    d'7'         ;See if at last digit yet
313         xorwf    CurDigt,w     ;Z set if so
314         btfsc   STATUS,Z      ;
315         clrf     CurDigt      ;Z set. Point back to rightmost digit
316
317         btfss   CurDigt,0     ;Must do this bit-by-bit
318         bcf     PORTA,0       ;First, clear as necessary
319         btfss   CurDigt,1
320         bcf     PORTA,1
321         btfss   CurDigt,2
322         bcf     PORTA,2
323
324         btfsc   CurDigt,0     ;Now, set if necessary
325         bsf     PORTA,0
326         btfsc   CurDigt,1
327         bsf     PORTA,1
328         btfsc   CurDigt,2
329         bsf     PORTA,2
330         return
331
332
333 ISR
334         movwf   w_temp        ; save off current W register contents
335         movf    STATUS,w      ; move status register into W register
336         movwf   status_temp   ; save off contents of STATUS register
337
338
339 ; isr code can go here or be located as a call subroutine elsewhere
340
341
342         incf    HoleCnt,f     ;Another hole
343         movf    HoleSet,w     ;See if reached hole setpoint yet
344         xorwf   HoleCnt,w     ;Z will be set if we have
345         btfss  STATUS,Z      ;
346         goto    DoneAdd      ;Nope. Need more holes!
347         clrf   HoleCnt      ;Yep. Now gotta up part count
348
349         movf    PartInc,w     ;Get amount to add to part count
350         addwf   Count0,f     ;Bump up the count
351         movlw   d'10'        ;See if 10 or more
352         bcf     STATUS,C      ;Clear the carry bit before comparisons
353         subwf   Count0,w     ;If 10 or more, Carry will be set
354         btfss  STATUS,C      ;
355         goto    DoneAdd      ;Not 10. Continue the count
356         movlw   d'10'        ;Must take 10 away now
357         subwf   Count0,f
358
359         incf    Count1,f     ;Increment next digit
360         movlw   d'10'
361         xorwf   Count1,w     ;Z set at 10
362         btfss  STATUS,Z      ;
363         goto    DoneAdd
364         clrf   Count1      ;Clear this digit

```

```

365
366      incf      Count2,f      ;Bump next digit
367      movlw    d'10'
368      xorwf    Count2,w      ;Z set at 10
369      btfss   STATUS,Z
370      goto     DoneAdd
371      clrf     Count2        ;Clear this digit
372
373      incf     Count3,f      ;Bump next digit
374      movlw    d'10'
375      xorwf    Count3,w      ;Z set at 10
376      btfss   STATUS,Z
377      goto     DoneAdd
378      clrf     Count3        ;Clear this digit
379 DoneAdd bcf     INTCON,1     ;Clear RB0 interupt flag like a good ISR
380      clrf     SleepLo      ;Parts actively being counted,
381      clrf     SleepHi      ;so stay awake!
382
383      movf     status_temp,w  ; retrieve copy of STATUS register
384      movwf    STATUS        ; restore pre-isr STATUS register contents
385      swapf   w_temp,f      ;
386      swapf   w_temp,w      ; restore pre-isr W register contents
387      retfie   ; return from interrupt
388
389
390 Segment andlw    0x0F      ;Drop the upper nibble
391      addwf   PCL,f        ;Change PC to appropriate table entry
392      ;Format:   gfedcbax , 1 in position activates segment
393      dt      b'01111110'  ;0
394      dt      b'00001100'  ;1
395      dt      b'10110110'  ;2
396      dt      b'10011110'  ;3
397      dt      b'11001100'  ;4
398      dt      b'11011010'  ;5
399      dt      b'11111010'  ;6
400      dt      b'01001110'  ;7
401      dt      b'11111110'  ;8
402      dt      b'11011110'  ;9
403      dt      b'10100100'  ;/
404
405
406
407
408
409
410
411
412
413      END                ; directive 'end of program'
414
415

```