```
#include<p16f873.inc> ; define o pic a utilizar
;#define    DEBUG
;#define    DEBUG_DELAY
;#define    USE_MINILOOP

    __config _XT_OSC & _WDT_OFF & _PWRTE_OFF & _CP_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _DEBUG_OFF & _WRT_OFF

    ; 0X20 é o inicio da memoria usavel nos dados (MD)
    CBLOCK 0x20
        ; variavies
        aux
        portb_mirror
        ; fucoes de serial
        byte_recebido_serial
        byte_enviar_serial
        ; funcao delay
        count_l
        count_h
        count_uh
        ; funcao pulso
        th
        tl
        th_thresh_hold
        mini_loop
        mini_loop_lenth
        ; troca entre th e tl
        counter
        port_b_high_value
        port_b_low_value
        ; setup do timer
        TMR0_mirror
    ENDC

    ORG 0
    GOTO    start                   ; go to beginning of program
    ORG  4
        ; INTERRUPCOES :
        ; timer
        BANKSEL INTCON
        BTFSS   INTCON, T0IF
        GOTO    end_int

        ; prepara o timer para a proxima chamada
        CALL    inicia_timer

        ;BANKSEL    0
        BCF     STATUS,     RP0
        BCF     STATUS,     RP1

        #ifdef DEBUG_DELAY
            MOVLW   0x02
            CALL    delay
        #endif

        #ifdef  USE_MINILOOP
            ; se o mini_loop não é zero, finaliza a interrupção
```

```
57              DECFSZ  mini_loop,  1
58                  GOTO    end_int
59              ; reinicia o mini_loop
60              MOVF    mini_loop_lenth, 0
61              MOVWF   mini_loop
62          #endif
63
64          MOVF    counter,    0
65          #ifdef DEBUG
66              CALL    envia_w_serial
67          #endif
68          DECFSZ  counter,    1
69              GOTO    end_int
70
71          BANKSEL PORTB
72          MOVF    PORTB               ,0
73
74          ;CALL   espera_valor ; espera valor não é preeptivo
75  espera_valor_int:
76          ; espera os dados da serial
77          MOVF    th, 0                       ; leitura_serial_asinc faz "MOVWF
            byte_recebido" caso nao receba nada
78          CALL    leitura_serial_asinc
79          MOVF    byte_recebido_serial, 0
80
81  calcula_valor_int:
82          ; pega valor de w e testa ele para ser o novo th
83          ; menor que 78, espera outro
84          SUBLW   D'77'
85          BTFSC   STATUS, C
86              GOTO    aritimetica
87
88          ; maior que 124, espera outro
89          MOVF    aux, w
90          SUBLW   D'124'
91          BTFSS   STATUS, C
92              GOTO    aritimetica
93
94          ; TH = valorRecebido
95          ; aux = valorRecebido
96          MOVF    byte_recebido_serial, 0
97          MOVWF   aux
98          MOVWF   th
99  aritimetica_int:
100         ; faz aritimética
101         ; 4) Então calcule TL =125-TH
102
103         MOVF    th , 0
104         SUBWF   th_thresh_hold, 0   ; W= 125-TH
105         MOVWF   tl                  ; TL = W (125-TH)
106
107         ; envia th e tl calculados
108         #ifdef DEBUG
109             MOVLW   0x54    ; ascii T
110             CALL envia_w_serial
111             MOVLW   0x48    ; ascii H
112             CALL envia_w_serial
```

```
113             MOVLW   0x00
114             CALL envia_w_serial
115             MOVF    th, 0
116             CALL envia_w_serial
117 
118             MOVLW   0x54    ; ascii T
119             CALL envia_w_serial
120             MOVLW   0x4C    ; ascii L
121             CALL envia_w_serial
122             MOVLW   0x00
123             CALL envia_w_serial
124             MOVF    tl, 0
125             CALL envia_w_serial
126 
127             MOVLW   0x00
128             CALL envia_w_serial
129         #endif
130 
131         ; encotra a fase atual
132         MOVF    port_b_low_value,   0   ; W = low_value
133         SUBWF   PORTB,              0   ; W = W - PORTB
134         BTFSC   STATUS,             Z   ; if PORTB == low_value (Z is SET ?)
135             GOTO    fase_alta           ; TRUE era BAIXO e agora deve ser ALTO
136         GOTO    fase_baixa              ; FALSE era ALTO e agora deve ser BAIXO
137 
138 fase_alta:
139         ; reinicia o contador de acordo com a fase
140         MOVF    th, W
141         MOVWF   counter
142 
143         ; configura a saida de acordo com a fase
144         MOVF    port_b_high_value, 0
145         MOVWF   PORTB
146 
147         GOTO    end_int
148 fase_baixa:
149         ; reinicia o contador de acordo com a fase
150         MOVF    tl, W
151         MOVWF   counter
152 
153         ; configura a saida de acordo com a fase
154         MOVF    port_b_low_value, 0
155         MOVWF   PORTB
156 
157 end_int:
158     RETFIE
159 
160 start:
161     ; inicializa as variaveis
162 
163     ; counter não deve ser iniciado com zero
164     MOVLW   0x01
165     MOVWF   counter
166 
167     ; configura todas as portas RB para output para incicializar PORTB
168     MOVLW   0x00
169     BANKSEL TRISB
```

```
170       MOVWF   TRISB
171
172       MOVLW   0xFF
173       BANKSEL PORTB
174       MOVWF   PORTB
175
176       ; inicia th e tl com 100
177       ; apenas th é nessesário mas iniciar todas as variaveis é bom
178       MOVLW   d'100'
179       MOVWF   th
180       MOVWF   tl
181
182       MOVLW   0xFF
183       MOVWF   port_b_high_value
184       MOVLW   0x00
185       MOVWF   port_b_low_value
186
187       MOVLW   d'125'
188       MOVWF   th_thresh_hold
189
190       MOVLW   D'100'
191       MOVWF   mini_loop_lenth
192       MOVWF   mini_loop
193
194       ; outras configuracoes
195       CALL    configura_serial
196       CALL    configura_timer
197
198       #ifdef  DEBUG
199           MOVLW   0xFF
200           CALL    envia_w_serial
201           MOVLW   0x00
202           CALL    envia_w_serial
203       #endif
204
205       ; o primeiro valor de th deve ser recebido pela serial sincronamente
206       ; ou seja, a execução é parada até receber algo pela serial
207       ; os demais são assincronos
208       CALL    leitura_serial
209       MOVF    byte_recebido_serial, 0
210       CALL    calcula_valor
211
212       ; inicia o timer
213       CALL    inicia_timer
214
215       ; fim da rotina principal
216       GOTO    $ ;security hold state loop
217
218   espera_valor:
219           ; espera os dados da serial
220           MOVF    th, 0                       ; leitura_serial_asinc faz "MOVWF
          byte_recebido" caso nao receba nada
221           CALL    leitura_serial_asinc
222           MOVF    byte_recebido_serial, 0
223
224   calcula_valor:
225           ; pega valor de w e testa ele para ser o novo th
```

```
226             ; menor que 78, espera outro
227             SUBLW   D'77'
228             BTFSC   STATUS, C
229                 GOTO    aritimetica
230
231             ; maior que 124, espera outro
232             MOVF    aux, w
233             SUBLW   D'124'
234             BTFSS   STATUS, C
235                 GOTO    aritimetica
236
237             ; TH = valorRecebido
238             ; aux = valorRecebido
239             MOVF    byte_recebido_serial, 0
240             MOVWF   aux
241             MOVWF   th
242     aritimetica:
243             ; faz aritimética
244             ; 4) Então calcule TL =125-TH
245
246             MOVF    th , 0
247             SUBWF   th_thresh_hold, 0   ; W= 125-TH
248             MOVWF   tl                  ; TL = W (125-TH)
249
250             ; envia th e tl calculados
251             #ifdef DEBUG
252                 MOVLW   0x54    ; ascii T
253                 CALL envia_w_serial
254                 MOVLW   0x48    ; ascii H
255                 CALL envia_w_serial
256                 MOVLW   0x00
257                 CALL envia_w_serial
258                 MOVF    th, 0
259                 CALL envia_w_serial
260
261                 MOVLW   0x54    ; ascii T
262                 CALL envia_w_serial
263                 MOVLW   0x4C    ; ascii L
264                 CALL envia_w_serial
265                 MOVLW   0x00
266                 CALL envia_w_serial
267                 MOVF    tl, 0
268                 CALL envia_w_serial
269
270                 MOVLW   0x00
271                 CALL envia_w_serial
272             #endif
273         RETURN
274
275     ; ------------------------------ SECAO SERIAL -----------------------------
276     configura_serial:
277             ; TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)
278             ; CSRC: Clock Source Select bit     = 0 (assincrono)
279             ; TX9: 9-bit Transmit Enable bit    = 0 (sem 9º bit)
280             ; TXEN: Transmit Enable bit         = 1 (liga o tx)
281             ; SYNC: USART Mode Select bit       = 0 (assincrono)
282             ; U-0 :                            = 0 (n/a)
```

```
283         ; BRGH: High Baud Rate Select bit   = 1 (High speed)
284         ; TRMT: Transmit Status bit         = 0 (read-only)
285         ; TX9D: 9th bit                     = 0 (n/a)
286         ; TXSTA = 0010 0100
287         BANKSEL TXSTA
288         MOVLW   B'00100100'
289         MOVWF   TXSTA
290
291         ; RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)
292         ; SPEN: Serial Port Enable bit          = 1 (enable)
293         ; RX9: 9-bit Receive Enable bit         = 0 (sem 9º bit)
294         ; SREN: Single Receive Enable bit       = 0 (N/A)
295         ; CREN: Continuous Receive Enable bit   = 1 (continuous receive)
296         ; ADDEN: Address Detect Enable bit      = 0 (N/A)
297         ; FERR: Framing Error bit               = 0 (read-only)
298         ; OERR: Overrun Error bit               = 0 (read-only)
299         ; RX9D: 9th bit of Received Data        = 0 (read-only)
300         ; RCSTA = 1001 0000
301         BANKSEL RCSTA
302         MOVLW   B'10010000'
303         MOVWF   RCSTA
304
305         ; SPBRG = 25
306         ; calculado para 9600 bps
307         BANKSEL SPBRG
308         MOVLW   D'25'
309         MOVWF   SPBRG
310
311         ;BANKSEL    0
312         BCF     STATUS,     RP0
313         BCF     STATUS,     RP1
314     RETURN
315 leitura_serial:
316         BANKSEL PIR1
317         ; registrador PIR1 contem as flags individuais de cada uma das
318         ; interrupcoes perifericas
319         ; bit RCIF indica se o buffer de entrada esta cheio
320         ; (entrada serial)
321 espera_leitura_serial:
322         BTFSS   PIR1,   RCIF            ; se bit RCIF do registrador PIR1
323                                         ; ou seja, se tem coisa a ler
324         GOTO    espera_leitura_serial   ; nao chegou byte
325         BANKSEL RCREG
326         MOVF    RCREG, W                ; chegou byte
327         ;BANKSEL    0
328         BCF     STATUS,     RP0
329         BCF     STATUS,     RP1
330         MOVWF   byte_recebido_serial
331     RETURN
332
333 leitura_serial_asinc:
334         BANKSEL PIR1
335         ; registrador PIR1 contem as flags individuais de cada uma das
336         ; interrupcoes perifericas
337         ; bit RCIF indica se o buffer de entrada esta cheio
338         ; (entrada serial)
339         BTFSS   PIR1,   RCIF            ; se bit RCIF do registrador PIR1
```

```
340                                         ; ou seja, se tem coisa a ler
341             GOTO    end_leitura_serial_asinc
342         BANKSEL RCREG
343         MOVF    RCREG, W                ; chegou byte
344         ;BANKSEL    byte_recebido_serial
345 end_leitura_serial_asinc:
346         BCF     STATUS,     RP0
347         BCF     STATUS,     RP1
348         MOVWF   byte_recebido_serial
349     RETURN
350
351 envia_w_serial:
352         BCF     STATUS, RP0
353         BCF     STATUS, RP1
354         MOVWF   byte_enviar_serial  ; guarda w em byte_enviar_serial
355 escrita_serial:
356         ; TXSTA:    transmit status and control register
357         ; TRMT:  bit de buffer de escrita cheio
358         BANKSEL TXSTA
359         ; caso ainda nao tenha enviado o que esta no buffer
360         ; espera o envio (anterior)
361 espera_escrita_serial:
362         ;TRMT: Transmit Shift Register Status bit
363         ;1 = TSR empty
364         ;0 = TSR full
365         BTFSS   TXSTA, TRMT             ; TRMT == 1? ;
366             GOTO    espera_escrita_serial   ; buffer cheio, espere
367         ;BANKSEL    byte_enviar_serial
368         BCF     STATUS, RP0
369         BCF     STATUS, RP1
370         MOVF    byte_enviar_serial, W   ; pega a variavel
371         BANKSEL TXREG
372         MOVWF   TXREG                   ; escreve a variavel para o buffer
373     RETURN
374 ; ------------------------------ SECAO TIMER -----------------------------
375 configura_timer:
376         ; T = 1ms
377
378         ; Fint = 1000 Hz
379         ; Prescaler = 2:1 (000)
380         ; TRM0 = 131157
381
382         ; Fint = 100 Hz
383         ; Prescaler = 16:1 (011)
384         ; TRM0 = 100
385
386         ; Fint = 10 Hz
387         ; Prescaler = 256:1 (111)
388         ; TRM0 = 158
389
390         ; ATIVA O TIMER
391         ; OPTION_REG: control bits to configure
392         ;   the TMR0 prescaler/WDT postscaler (single assign-
393         ;   able register known also as the prescaler), the External
394         ;   INT Interrupt, TMR0 and the weak pull-ups on PORTB.
395         ;   (ADDRESS 81h, 181h)
396         ;
```

```
397         ; RBPU: PORTB Pull-up Enable bit        = 0 (PORTB pull-ups are enabled by
            individual port latch values)
398         ; INTEDG: Interrupt Edge Select bit     = 0 (Interrupt on falling edge of
            RB0/INT pin)
399         ; T0CS: TMR0 Clock Source Select bit    = 0 (Internal instruction cycle
            clock CLKOUT)
400         ; T0SE: TMR0 Source Edge Select bit     = 0 (assincrono)
401         ; PSA: Prescaler Assignment bit         = 0 (Prescaler is assigned to the
            Timer0 module)
402         ; PS2:PS0: Prescaler Rate Select bits   = (1:2)
403         ; PS2                                   = 0
404         ; PS1                                   = 0
405         ; PS0                                   = 0
406         BANKSEL OPTION_REG
407         MOVLW   B'00000010'
408         MOVWF   OPTION_REG
409
410         ; FREQUENCIA DE INTERRUPÇAO DO TIMER
411         ; (1MHz/4)/64[prescaler]
412         ; ------------------------
413         ;       (256 - TMR0)
414         BANKSEL TMR0
415         MOVLW   D'131'
416         MOVWF   TMR0
417
418         ; BANKSEL 0
419         BCF STATUS, RP0
420         BCF STATUS, RP1
421         MOVWF   TMR0_mirror
422
423         ; CONFIGURA A INTERRUPÇÃO
424         ; INTCON: (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)
425         ;   enable and flag bits for the
426         ;   TMR0 register overflow, RB Port change and External
427         ;   RB0/INT pin interrupts.
428         ;   NÃO ATIVO A INTERRUPCAO AINDA
429         ; GIE: Global Interrupt Enable bit             = 1 (Enables all unmasked
            interrupts)
430         ; PEIE: Peripheral Interrupt Enable bit        = x
431         ; T0IE: TMR0 Overflow Interrupt Enable bit     = 1 (Enables the TMR0
            interrupt)
432         ; INTE: RB0/INT External Interrupt Enable bit  = x
433         ; RBIE: RB Port Change Interrupt Enable bit    = x
434         ; T0IF: TMR0 Overflow Interrupt Flag bit       = 0
435         ; INTF: RB0/INT External Interrupt Flag bit    = x (The RB0/INT external
            interrupt did not occur)
436         ; RBIF: RB Port Change Interrupt Flag bit      = x (None of the RB7:RB4
            pins have changed state)
437         ; INTCON = 1010 0000
438         BANKSEL INTCON
439         MOVLW   B'10000000'
440         MOVWF   INTCON
441
442         ; BANKSEL 0
443         BCF STATUS, RP0
444         BCF STATUS, RP1
445     RETURN
```

```
446  inicia_timer:
447          ; RECONFIGURA O TMR0
448          MOVF    TMR0_mirror, 0
449          BANKSEL TMR0
450          MOVWF   TMR0
451
452          ; DISPARA O TIMER
453          BANKSEL INTCON
454          BSF INTCON, T0IE
455          ; LIMPA AS INTERRUPÇOES
456          BCF INTCON, T0IF
457
458          ; BANKSEL 0
459          BCF STATUS, RP0
460          BCF STATUS, RP1
461      RETURN
462
463  ; ------------------------------- delay ---------------------------------------
464  delay:
465          MOVWF   count_uh
466  delay_loop_uh:
467          MOVLW   0xff
468          MOVWF   count_h
469  delay_loop_h:
470          MOVLW   0xff
471          MOVWF   count_l
472  delay_loop_l:
473          DECFSZ  count_l
474          GOTO    delay_loop_l
475          DECFSZ  count_h
476          GOTO    delay_loop_h
477          DECFSZ  count_uh
478          GOTO    delay_loop_uh
479      RETURN
480      GOTO    $   ; security loop forever
481      END
482
```