```
/*
Programa que utiliza o PIC18F4520, para funcionamento como gestor de parâmetros.

Parâmetros:
   P0...Mostra o valor definido em P1
   P1...Escolhe o valor a ser mostrado em P0
           P1=0: Erro (AN0)
           P1=1: Vin  (AN1)
           P1=2: isw  (AN2)
           P1=3: ibat (AN3)
   P2...Iref=8.0A
   P3...Umax=9.9A
   P4...Imin=1.0A
   P5...Vmax=70V
   P6...Vmin=30V
   P7...Dinc=0.05
   P8...Ts=5s
   P9...Te=30s
   PA...Dinicial=0.1
   PB...Fonte do PWM1
           PB=0: Auto
           PB=1: Teste - AN0
           PB=2: Teste - AN1
           PB=3: Teste - AN2
           PB=4: Teste - AN3
           PB=5: Valor definido em PC
   PC...Valor de PWM1 em caso de teste, se escolhido em PB
   PD...Fonte do PWM2
           PB=0: Auto
           PB=1: Teste - AN0
           PB=2: Teste - AN1
           PB=3: Teste - AN2
           PB=4: Teste - AN3
           PB=5: Valor definido em PE
   PE...Valor de PWM2 em caso de teste, se escolhido em PD
   PF...Leitura / Gravação de parametros
           PF=0: Não altera os parâmetros
           PF=1: Repor os parâmetros de fábrica
           PF=2: Gravar parâmetros na EEPROM
           PF=3: Ler parâmetros na EEPROM


                  +-- PIC18F4520 --+
    (VPP)   ---- |  1          40 | ---- D1_B (PGD)
    AN0     ---- |  2          39 | ---- D1_A (PGC)
    AN1     ---- |  3          38 | ---- D1_F
    AN2     ---- |  4          37 | ---- D1_G
    AN3     ---- |  5          36 | ---- D2_B
    w1      ---- |  6          35 | ---- D2_A
            ---- |  7          34 | ---- D2_G
    Botão_P ---- |  8          33 | ---- D2_F
    Botão_- ---- |  9          32 | ---- VDD
    Botão_+ ---- | 10          31 | ---- VSS
    VDD     ---- | 11          30 | ---- D2_D
    VSS     ---- | 12          29 | ---- D2_E
            ---- | 13          28 | ---- D2_C
            ---- | 14          27 | ---- D2_DP
            ---- | 15          26 | ---- D1_E
    PWM2    ---- | 16          25 | ---- D1_D
    PWM1    ---- | 17          24 | ---- D1_C
            ---- | 18          23 | ---- D1_DP
            ---- | 19          22 | ----
```

```
 63:             ---- | 20          21 | ----
 64:                  +-----------------+
 65:
 66: f(pwm)max = f(osc)/256
 67:
 68: O relogio já está a funcionar à custa do Timer0. Mas dado que "apenas" funciona
 69: a 8 bit's não tem tempo de respoasta, ie, o tempo de que demora um scan é maior
 70: que o tempo que o timer0 demora a "estourar". De momento o valor minimo Prescale
 71: é 1:2
 72: */
 73:
 74: bit DP1;
 75: bit DP2;
 76: char I0=0, I0_tmp=0, I0_old=0;
 77: char I0_timer[3] = {0, 0, 0};
 78: char N1, N2;
 79: char QB, QC, QD;
 80: char V0=0, V1=0, V2=0, V3=0, V4=0;
 81: int AIW0=0, AIW2=0, AIW4=0, AIW6=0, AIW10=0;
 82: char aiw_num=0;
 83: char Display=0;
 84:
 85: char erro, dt=50*2.56;
 86: float Vin, Isw, Ibat, dto, dti, gi, tb;
 87:
 88: char parametro[16]; //       1  2  3  4  5  6  7  8  9  A  B  C  D  E F
 89: char parametro_def[16] = {0, 1,40,50,10,70,30, 5, 5,30,10, 1,10, 2,50,0};
 90: char parametro_max[16] = {0, 3,99,99,50,99,50,99,15,50,99, 5,99, 5,99,2};
 91: char parametro_n=0;      // Nº do parametro selecionado
 92: char parametro_n_max=14; // No máximo pode valer 14 (E em HEX)
 93: bit parametro_mostrar;   /* Significado
 94:                              0 - mostra o nº do parametro
 95:                              1 - mostra o valor do parametro */
 96:
 97: long relogio_imp=0, relogio_ms=0;
 98: float relogio_s;
 99: long relogio_i0, relogio_i1, relogio_i2;
100:
101: void registos() {
102:    //Controlo do Oscilador - OSCCON
103:    OSCCON=0b11111100;
104:    /*      >||  ||| <
105:             ||  ||+--bit 1-0 SCS<1:0>: System Clock Select bits
106:             ||  ||      1x = Internal oscillator block
107:             ||  ||      01 = Secondary (Timer1) oscillator
108:             ||  ||      00 = Primary oscillator
109:             ||  |+---bit 2 IOFS: INTOSC Frequency Stable bit
110:             ||  |       1 = INTOSC frequency is stable
111:             ||  |       0 = INTOSC frequency is not stable
112:             ||  +----bit 3 OSTS: Oscillator Start-up Timer Time-out Status bit(1
113:             ||          1 = Oscillator Start-up Timer (OST) time-out has expired
     primary oscillator is running
114:             ||          0 = Oscillator Start-up Timer (OST) time-out is running;
    primary oscillator is not ready
115:             |+-------bit 6-4 IRCF<2:0>: Internal Oscillator Frequency Select bit
116:             |           111 = 8 MHz (INTOSC drives clock directly)
117:             |           110 = 4 MHz
118:             |           101 = 2 MHz
119:             |           100 = 1 MHz(3)
120:             |           011 = 500 kHz
```

```
121:             |           010 = 250 kHz
122:             |           001 = 125 kHz
123:             |           000 = 31 kHz (from either INTOSC/256 or INTRC directly)(
     )
124:             +--------bit 7 IDLEN: Idle Enable bit
125:                        1 = Device enters an Idle mode on SLEEP instruction
126:                        0 = Device enters Sleep mode on SLEEP instruction
127:       Note 1: Reset state depends on state of the IESO Configuration bit.
128:       Note 2: Source selected by the INTSRC bit (OSCTUNE<7>), see text.
129:       Note 3: Default output frequency of INTOSC on Reset. */
130:
131:    // Ajuste do Oscilador - OSCTUNE
132:    OSCTUNE=0b01000000;
133:    /*       >||||    <
134:              |||+-----bit 4-0 TUN<4:0>: Frequency Tuning bits
135:              |||         011111 = Maximum frequency
136:              |||         •              •
137:              |||         000001
138:              |||         000000 = Center frequency. Oscillator module is running
     at the calibrated frequency.
139:              |||         111111
140:              |||         •              •
141:              |||         100000 = Minimum frequency
142:              ||+------bit 5 Unimplemented: Read as '0'
143:              |+-------bit 6 PLLEN: Frequency Multiplier PLL for INTOSC Enable bi
     (1)
144:              |           1 = PLL enabled for INTOSC (4 MHz and 8 MHz only)
145:              |           0 = PLL disabled
146:              +--------bit 7 INTSRC: Internal Oscillator Low-Frequency Source Sel
     ct bit
147:                        1 = 31.25 kHz device clock derived from 8 MHz INTOSC so
     rce (divide-by-256 enabled)
148:                        0 = 31 kHz device clock derived directly from INTRC int
     rnal oscillator
149:       Note 1: Available only in certain oscillator configurations; otherwise,
150:          this bit is unavailable and reads as '0'. See Section 2.6.4 "PLL in
151:          INTOSC Modes" for details. */
152:
153:    //Configuração do PORTA (Entradas / Saídas) - TRISA
154:    TRISA=0b11111111;
155:    /*      1 = PORTA pin configured as an input
156:            0 = PORTA pin configured as an output
157:       Note 1: bit 7-6 RA<7:6>: and their associated latch and data direction
158:          bits are enabled as I/O pins based on oscillator configuration;
159:          otherwise, they are read as '0'. */
160:
161:    //Configuração do PORTB (Entradas / Saídas) - TRISB
162:    TRISB=0b00000000;
163:    /*      1 = PORTB pin configured as an input
164:            0 = PORTB pin configured as an output */
165:
166:    //Configuração do PORTC (Entradas / Saídas) - TRISC
167:    TRISC=0b00001001;
168:    /*      1 = PORTC pin configured as an input
169:            0 = PORTC pin configured as an output */
170:
171:    //Configuração do PORTD (Entradas / Saídas) - TRISD
172:    TRISD=0b00001111;
173:    /*      1 = PORTD pin configured as an input
174:            0 = PORTD pin configured as an output */
175:
176:    //Configuração do PORTE (Entradas / Saídas) - TRISE
```

```
177:    TRISE=0b00000111;
178:    /*     >||| ||  <
179:             ||| |+----bit 2-0 TRISE2-TRISE0: RE2-RE0 Direction Control bit
180:             |||||        1 = Input
181:             |||||        0 = Output
182:             ||||+-----bit 3 Unimplemented: Read as '0'
183:             |||+------bit 4 PSPMODE: Parallel Slave Port Mode Select bit
184:             |||          1 = Parallel Slave Port mode
185:             |||          0 = General purpose I/O mode
186:             ||+-------bit 5 IBOV: Input Buffer Overflow Detect bit (in Microproce
     ssor mode)
187:             ||           1 = A write occurred when a previously input word has no
      been read (must be cleared in software)
188:             ||           0 = No overflow occurred
189:             |+--------bit 6 OBF: Output Buffer Full Status bit
190:             |            1 = The output buffer still holds a previously written w
     rd
191:             |            0 = The output buffer has been read
192:             +---------bit 7 IBF: Input Buffer Full Status bit
193:                          1 = A word has been received and waiting to be read by t
     e CPU
194:                          0 = No word has been received */
195:
196:
197:    //Configuração das entradas (Analógico / Digital) - ADCON1
198:    ADCON1=0b00001101;
199:    /*      >| |||   <
200:              | ||+----bit 3-0 PCFG<3:0>: A/D Port Configuration Control bits:
201:              | ||         +--------+----+----+----+----+----+----+----+----+----+-
     --+----+----+----+
202:              | ||         | PCFG3: | AN | AN | AN | AN | AN | AN | AN | AN | AN |
     N | AN | AN | AN |
203:              | ||         | PCFG0  | 12 | 11 | 10 |  9 |  8 |7(2)|6(2)|5(2)|  4 |
     3 |  2 |  1 |  0 |
204:              | ||         +--------+----+----+----+----+----+----+----+----+----+-
     --+----+----+----+
205:              | ||         | 0000(1)|  A |  A | A  |  A |  A |  A |  A |  A |  A |
     A |  A |  A |  A |
206:              | ||         | 0001   |  A |  A | A  |  A |  A |  A |  A |  A |  A |
     A |  A |  A |  A |
207:              | ||         | 0010   |  A |  A | A  |  A |  A |  A |  A |  A |  A |
     A |  A |  A |  A |
208:              | ||         | 0011   |  D |  A | A  |  A |  A |  A |  A |  A |  A |
     A |  A |  A |  A |
209:              | ||         | 0100   |  D |  D | A  |  A |  A |  A |  A |  A |  A |
     A |  A |  A |  A |
210:              | ||         | 0101   |  D |  D | D  |  A |  A |  A |  A |  A |  A |
     A |  A |  A |  A |
211:              | ||         | 0110   |  D |  D | D  |  D |  A |  A |  A |  A |  A |
     A |  A |  A |  A |
212:              | ||         | 0111(1)|  D |  D | D  |  D |  D |  A |  A |  A |  A |
     A |  A |  A |  A |
213:              | ||         | 1000   |  D |  D | D  |  D |  D |  D |  A |  A |  A |
     A |  A |  A |  A |
214:              | ||         | 1001   |  D |  D | D  |  D |  D |  D |  D |  A |  A |
     A |  A |  A |  A |
215:              | ||         | 1010   |  D |  D | D  |  D |  D |  D |  D |  D |  A |
     A |  A |  A |  A |
216:              | ||         | 1011   |  D |  D | D  |  D |  D |  D |  D |  D |  D |
     A |  A |  A |  A |
217:              | ||         | 1100   |  D |  D | D  |  D |  D |  D |  D |  D |  D |
     D |  A |  A |  A |
```

```
218:              | ||        | 1101   |  D |  D | D  |  D |  D |  D |  D |  D |  D |
    D |  D |  A |  A |
219:              | ||        | 1110   |  D |  D | D  |  D |  D |  D |  D |  D |  D |
    D |  D |  D |  A |
220:              | ||        | 1111   |  D |  D | D  |  D |  D |  D |  D |  D |  D |
    D |  D |  D |  D |
221:              | ||        +--------+----+----+----+----+----+----+----+----+----+-
    --+----+----+----+
222:              | ||                                 A = Analog input D = Digital I/O
223:              | |+-----bit 4 VCFG0: Voltage Reference Configuration bit (VREF+ sou
    ce)
224:              | |         1 = VREF+ (AN3)
225:              | |         0 = VDD
226:              | +-----bit 5 VCFG1: Voltage Reference Configuration bit (VREF- sour
    e)
227:              |           1 = VREF- (AN2)
228:              |           0 = VSS
229:              +-------bit 7-6 Unimplemented: Read as '0'
230:       Note 1: The POR value of the PCFG bits depends on the value of the PBADEN
231:          Configuration bit. When PBADEN = 1, PCFG<2:0> = 000; when PBADEN = 0,
232:          PCFG<2:0> = 111.
233:       Note 2: AN5 through AN7 are available only on 40/44-pin devices.*/
234:
235:    //Configuração do Timer0 - T0CON
236:    T0CON=0b11000010;
237:    /*     >||||||  <
238:             |||||+----bit 2-0 T0PS<2:0>: Timer0 Prescaler Select bits
239:             |||||        111 = 1:256 Prescale value
240:             |||||        110 = 1:128 Prescale value
241:             |||||        101 = 1:64 Prescale value
242:             |||||        100 = 1:32 Prescale value
243:             |||||        011 = 1:16 Prescale value
244:             |||||        010 = 1:8 Prescale value
245:             |||||        001 = 1:4 Prescale value
246:             |||||        000 = 1:2 Prescale value
247:             ||||+-----bit 3 PSA: Timer0 Prescaler Assignment bit
248:             ||||         1 = Timer0 prescaler is not assigned. Timer0 clock input
    bypasses prescaler.
249:             ||||         0 = Timer0 prescaler is assigned. Timer0 clock input com
    s from prescaler output.
250:             |||+------bit 4 T0SE: Timer0 Source Edge Select bit
251:             |||           1 = Increment on high-to-low transition on T0CKI pin
252:             |||           0 = Increment on low-to-high transition on T0CKI pin
253:             ||+-------bit 5 T0CS: Timer0 Clock Source Select bit
254:             ||            1 = Transition on T0CKI pin
255:             ||            0 = Internal instruction cycle clock (CLKO)
256:             |+--------bit 6 T08BIT: Timer0 8-Bit/16-Bit Control bit
257:             |             1 = Timer0 is configured as an 8-bit timer/counter
258:             |             0 = Timer0 is configured as a 16-bit timer/counter
259:             +---------bit 7 TMR0ON: Timer0 On/Off Control bit
260:                           1 = Enables Timer0
261:                           0 = Stops Timer0 */
262:
263:    //Configuração do Timer1 - T1CON
264:    T1CON=0b01000000;
265:    /*     >||| ||||<
266:             ||| |||+-bit 0 TMR1ON: Timer1 On bit
267:             ||| |||     1 = Enables Timer1
268:             ||| |||     0 = Stops Timer1
269:             ||| ||+--bit 1 TMR1CS: Timer1 Clock Source Select bit
270:             ||| ||      1 = External clock from pin RC0/T1OSO/T13CKI (on the risi
    g edge)
```

```
271:            ||| ||      0 = Internal clock (FOSC/4)
272:            ||| |+---bit 2 -T1SYNC: Timer1 External Clock Input Synchronization S
    lect bit
273:            ||| |       When TMR1CS = 1:
274:            ||| |          1 = Do not synchronize external clock input
275:            ||| |          0 = Synchronize external clock input
276:            ||| |       When TMR1CS = 0:
277:            ||| |          This bit is ignored. Timer1 uses the internal clock wh
    n TMR1CS = 0.
278:            ||| +----bit 3 T1OSCEN: Timer1 Oscillator Enable bit
279:            |||         1 = Timer1 oscillator is enabled
280:            |||         0 = Timer1 oscillator is shut off
281:            |||         The oscillator inverter and feedback resistor are turned
    ff to eliminate power drain.
282:            ||+------bit 5-4 T1CKPS<1:0>: Timer1 Input Clock Prescale Select bits
283:            ||          11 = 1:8 Prescale value
284:            ||          10 = 1:4 Prescale value
285:            ||          01 = 1:2 Prescale value
286:            ||          00 = 1:1 Prescale value
287:            |+-------bit 6 T1RUN: Timer1 System Clock Status bit
288:            |           1 = Device clock is derived from Timer1 oscillator
289:            |           0 = Device clock is derived from another source
290:            +--------bit 7 RD16: 16-Bit Read/Write Mode Enable bit
291:                        1 = Enables register read/write of Timer1 in one 16-bit o
    eration
292:                        0 = Enables register read/write of Timer1 in two 8-bit op
    rations */
293:
294:    //Configuração do Timer2 - T2CON
295:    T2CON=0b00100010;
296:    /*     >||   || <
297:            ||   |+--bit 1-0 T2CKPS<1:0>: Timer2 Clock Prescale Select bits
298:            ||   |      00 = Prescaler is 1
299:            ||   |      01 = Prescaler is 4
300:            ||   |      1x = Prescaler is 16
301:            ||   +---bit 2 TMR2ON: Timer2 On bit
302:            ||          1 = Timer2 is on
303:            ||          0 = Timer2 is off
304:            |+-------bit 6-3 T2OUTPS<3:0>: Timer2 Output Postscale Select bits
305:            |           0000 = 1:1 Postscale
306:            |           0001 = 1:2 Postscale
307:            |           0010 = 1:3 Postscale
308:            |           0011 = 1:4 Postscale
309:            |           0100 = 1:5 Postscale
310:            |           0101 = 1:6 Postscale
311:            |           0110 = 1:7 Postscale
312:            |           0111 = 1:8 Postscale
313:            |           1000 = 1:9 Postscale
314:            |           1001 = 1:10 Postscale
315:            |           1010 = 1:11 Postscale
316:            |           1011 = 1:12 Postscale
317:            |           1100 = 1:13 Postscale
318:            |           1101 = 1:14 Postscale
319:            |           1110 = 1:15 Postscale
320:            |           1111 = 1:16 Postscale
321:            +--------bit 7 Unimplemented: Read as '0' */
322:             wdtcon=0b00000000;
323:    }
324:
325: void Entradas_D() {
326:    I0_old=I0;
327:    I0_tmp.b0=porte.b0;
```

```
328:    I0_tmp.b1=porte.b1;
329:    I0_tmp.b2=porte.b2;
330:    I0_tmp.b3=porta.b4;
331:
332:    if (!I0_tmp.b0) I0_timer[0]=0;
333:    if (!I0_tmp.b1) I0_timer[1]=0;
334:    if (!I0_tmp.b2) I0_timer[2]=0;
335:
336:    if (I0_tmp.b0) I0_timer[0]++;
337:    if (I0_tmp.b1) I0_timer[1]++;
338:    if (I0_tmp.b2) I0_timer[2]++;
339:
340:    for(v4=0; v4<3; v4++) if (I0_timer[V4]==255) I0_timer[V4]=254;
341:
342:    I0.B0=(I0_timer[0]>20);
343:    I0.B1=(I0_timer[1]>20);
344:    I0.B2=(I0_timer[2]>20);
345:    I0.b3=I0_tmp.b3;
346:    }
347:
348: void Entradas_A() {
349:    aiw_num++;
350:    if (aiw_num>3) aiw_num=0;
351:    switch (aiw_num) {
352:       case 0: {AIW0=adc_read(0); erro=aiw0/10.24;} break;
353:       case 1: {AIW2=adc_read(1); Vin=aiw2/10.24;} break;
354:       case 2: {AIW4=adc_read(2); Isw=aiw4/102.4;} break;
355:       case 3: {AIW6=adc_read(3); Ibat=aiw6/102.4;} break;
356:       }
357:    }
358:
359: void Saidas_D() {
360:    if (Display<100) {
361:       N2=Display/10;
362:       N1=Display-N2*10;
363:       }
364:
365:    if ((Display>=100) && (Display<200)) N2=20;
366:
367:    if ((Display>=200) && (Display<216)) {
368:       N2=11;
369:       N1=Display-200;
370:       }
371:
372:    if (Display>=216) N2=20;
373:
374:    switch (N1) {
375:       case  0: {QB=224; QC=224; QD=  0;}; break;
376:       case  1: {QB=128; QC= 32; QD=  0;}; break;
377:       case  2: {QB=208; QC=192; QD=  0;}; break;
378:       case  3: {QB=208; QC= 96; QD=  0;}; break;
379:       case  4: {QB=176; QC= 32; QD=  0;}; break;
380:       case  5: {QB=112; QC= 96; QD=  0;}; break;
381:       case  6: {QB=112; QC=224; QD=  0;}; break;
382:       case  7: {QB=192; QC= 32; QD=  0;}; break;
383:       case  8: {QB=240; QC=224; QD=  0;}; break;
384:       case  9: {QB=240; QC= 96; QD=  0;}; break;
385:       case 10: {QB=240; QC=160; QD=  0;}; break;
386:       case 11: {QB= 48; QC=224; QD=  0;}; break;
387:       case 12: {QB= 96; QC=192; QD=  0;}; break;
388:       case 13: {QB=144; QC=224; QD=  0;}; break;
389:       case 14: {QB=112; QC=192; QD=  0;}; break;
```

```
390:       case 15: {QB=112; QC=128; QD=  0;}; break;
391:       //default: {QB=XXX; QC=XXX; QD=XXX;};
392:    }
393:
394:    switch (N2) {
395:       case  0: {QB=QB+ 13; QC=QC+  0; QD=QD+224;}; break;
396:       case  1: {QB=QB+  8; QC=QC+  0; QD=QD+ 32;}; break;
397:       case  2: {QB=QB+ 14; QC=QC+  0; QD=QD+192;}; break;
398:       case  3: {QB=QB+ 14; QC=QC+  0; QD=QD+160;}; break;
399:       case  4: {QB=QB+ 11; QC=QC+  0; QD=QD+ 32;}; break;
400:       case  5: {QB=QB+  7; QC=QC+  0; QD=QD+160;}; break;
401:       case  6: {QB=QB+  7; QC=QC+  0; QD=QD+224;}; break;
402:       case  7: {QB=QB+ 12; QC=QC+  0; QD=QD+ 32;}; break;
403:       case  8: {QB=QB+ 15; QC=QC+  0; QD=QD+224;}; break;
404:       case  9: {QB=QB+ 15; QC=QC+  0; QD=QD+160;}; break;
405:       case 10: {QB=QB+ 15; QC=QC+  0; QD=QD+ 96;}; break;
406:       case 11: {QB=QB+ 15; QC=QC+  0; QD=QD+ 64;}; break;
407:       default: {QB=18; QC=0; QD=0;};
408:    }
409:
410:    if (dp1) QC.b4=1;
411:    if (dp2) QD.b4=1;
412:
413:    portb=QB;
414:    portc=QC;
415:    portd=QD;
416:    }
417:
418: void ler_defeito() {
419:    for (v0=1; V0<16; V0++) {
420:       parametro[V0]=parametro_def[V0];
421:       if (parametro[V0]>parametro_max[V0]) parametro[V0]=parametro_max[V0];
422:       }
423:    }
424:
425: void ler_eeprom() {
426:    for (v0=1; V0<16; V0++) {
427:       parametro[V0]=eeprom_read(V0);
428:       if (parametro[V0]>parametro_max[V0]) parametro[V0]=parametro_max[V0];
429:       }
430:    }
431:
432: void gravar_eeprom() {
433:    eeprom_write(0,1);
434:    for (v0=1; V0<16; V0++) {
435:       eeprom_write(v0,parametro[V0]);
436:       }
437:    }
438:
439: void main() {
440:    // * * * Inicialização do PIC (Oscilador, Entradas / Saídas, etc) * * *
441:    registos();
442:    pwm1_init(125000);
443:    PWM1_Stop();
444:    PWM1_Set_Duty(100);
445:    PWM1_start();
446:
447:    pwm2_init(125000);
448:    PWM2_Stop();
449:    PWM2_Set_Duty(154);
450:    PWM2_start();
451:
```

```
452:    porta=0;
453:    portb=0;
454:    portc=0;
455:    portd=0;
456:
457:    DP1=0;
458:    DP2=0;
459:
460:    parametro_mostrar=1;
461:
462:    V0=eeprom_read(0);
463:    if (V0==1) {
464:       ler_eeprom();
465:       parametro_max[15]=3;
466:       }
467:       else {
468:          ler_defeito();
469:          parametro_max[15]=2;
470:          }
471:    parametro[15]=0;
472:
473:    while(1) { // * * * Inicio do ciclo While principal * * *
474:       if (INTCON.B2==1) { //incremento do relogio
475:          relogio_imp++;
476:          INTCON.B2=0;
477:          relogio_s=relogio_imp*256E-6;
478:          relogio_ms=relogio_s*1000;
479:          //relogio_ms=relogio_imp/4;
480:          }
481:       Entradas_D();   // Leitura das entradas digitais
482:       Entradas_A(); // Leitura das entradas analógicas
483:
484:       // * * * Processamento * * *
485:
486:       /*if (PIR1.b0) {
487:          relogio_imp++;
488:          PIR1.b0=0;
489:          relogio_s=relogio_imp*8.192E-3; //32e-6;
490:          }*/
491:
492:       if ((i0.b0) && (!i0_old.b0)) { // Botão Parametros
493:          if ((parametro_mostrar) && (parametro_n==15)) {
494:             if (parametro[15]==1) ler_defeito();
495:             if (parametro[15]==2) {
496:                gravar_eeprom();
497:                parametro_max[15]=3;
498:                }
499:             if (parametro[15]==3) ler_eeprom();
500:             parametro[15]=0;
501:             }
502:          parametro_mostrar=!parametro_mostrar;
503:          }
504:
505:       if ((i0.b1) && (!i0_old.b1)) { // Botão -
506:          if (parametro_mostrar) {
507:             parametro[parametro_n]--;
508:             if (parametro[parametro_n]==255) parametro[parametro_n]=parametro_max
     [parametro_n];
509:             }
510:          if (!parametro_mostrar) {
511:             parametro_n--;
512:             if (parametro_n==255) parametro_n=15;
```

```
513:             if ((parametro_n>parametro_n_max) && (parametro_n <15)) parametro_n=pa
     arametro_n_max;
514:             }
515:          }
516:
517:       if ((i0.b2) && (!i0_old.b2)) { // Botão +
518:          if (parametro_mostrar) {
519:             parametro[parametro_n]++;
520:             if (parametro[parametro_n]>parametro_max[parametro_n]) parametro[param
     metro_n]=0;
521:             }
522:          if (!parametro_mostrar) {
523:             parametro_n++;
524:             if (parametro_n==16) parametro_n=0;
525:             if (parametro_n>parametro_n_max) parametro_n=15;
526:             }
527:          }
528:
529:       //Determinação do conteudo a escrever nos displays
530:       if (parametro_mostrar) {
531:          if (parametro_n==0) {
532:             if (parametro[1]==0) Display=erro;
533:             if (parametro[1]==1) Display=Vin;
534:             if (parametro[1]==2) Display=(isw*10);
535:             if (parametro[1]==3) Display=(ibat*10);
536:             if (parametro[1]==4) Display=floor(relogio_ms/1000);
537:             }
538:          if (parametro_n>0) Display=parametro[parametro_n];
539:          }
540:          else {
541:          Display=200+parametro_n;
542:          }
543:
544:       DP2=(!parametro_mostrar);
545:       if (parametro_mostrar) {
546:          switch (parametro_n) {
547:             case 0: {
548:                switch (parametro[1]) {
549:                   case 2: DP2=1; break;
550:                   case 3: DP2=1; break;
551:                   }
552:                }; break;
553:             case 2: DP2=1; break;
554:             case 3: DP2=1; break;
555:             case 4: DP2=1; break;
556:             }
557:          }
558:
559:
560:
561:         delay_us(200);
562:
563:
564:         if(parametro[11]==0){
565:         if((AIW2/(4*2.56)>parametro[6]) && (AIW4/(4*2.56)<parametro[3])){
566:         if(dti==0){
567:         dti=50*2.56;
568:         }
569:         if((AIW0/(4*2.56))>parametro[2]){
570:         if(tb==0){
571:         dto=dt;
572:         dti=dto+((AIW0/4)-parametro[2]*2.56)*0.1;
```

```
573:          /*dt=dti;*/
574:          delay_ms(100);
575:          tb=0;
576:          }
577:          else {
578:          dto=dti;
579:          dti=dto+((AIW0/4)-parametro[2]*2.56)*0.1;
580:          delay_ms(100);
581:          tb=0;
582:          }
583:          }
584:          if((AIW0/(4*2.56))<parametro[2]){
585:          if(tb==0){
586:          dto=dt;
587:          dti=dto+((AIW0/4)-parametro[2]*2.56)*0.1;
588:          /*dt=dti;*/
589:          delay_ms(100);
590:          tb=0;
591:          }
592:          else {
593:          dto=dti;
594:          dti=dto+((AIW0/4)-parametro[2]*2.56)*0.1;
595:          delay_ms(100);
596:          tb=0;
597:          }
598:          }
599:          }
600:           else dti=0;
601:           tb=1;
602:           delay_ms(100);
603:          }
604:
605:
606:
607:       //pwm1_set_duty(64);
608:       //pwm2_set_duty(0);
609:       switch (parametro[11]) {
610:          case 0: pwm1_set_duty(parametro[2]*2.56); break;
611:          case 1: pwm1_set_duty(AIW0/4); break;
612:          case 2: pwm1_set_duty(AIW2/4); break;
613:          case 3: pwm1_set_duty(AIW4/4); break;
614:          case 4: pwm1_set_duty(AIW6/4); break;
615:          case 5: pwm1_set_duty(parametro[12]*2.56); break;
616:          }
617:
618:       switch (parametro[13]) {
619:          case 0: pwm2_set_duty(dti); break;
620:          case 1: pwm2_set_duty(AIW0/4); break;
621:          case 2: pwm2_set_duty(AIW2/4); break;
622:          case 3: pwm2_set_duty(AIW4/4); break;
623:          case 4: pwm2_set_duty(AIW6/4); break;
624:          case 5: pwm2_set_duty(parametro[14]*2.56); break;
625:          }
626:
627:
628:       // * * * Activação das Saídas * * *
629:       Saidas_D();
630:       } // * * * Fim do ciclo While principal * * *
631:    }
```