```
1    /*
2    contiene tutte le routine per la gestione della tastierina e del display LCD
3    */
4
5
6    /*UserInterface ***************************************************************
7    Raggruppa tutte le routine gestite manualmente: Keypad, LCD, EEPROM.
8    Dal momento che sono eseguite molto raramente sono tutte ragruppate sotto un unico
9    semaforo.
10   Questo permette di risparmiare diversi cicli di controllo nel funzionamento normale.
11   */
12
13   void UserInterface (void)
14   {
15   //EEPROM=========================================
16   if (EepromFlag)      // se = 0 e' disabilitata la gestione dell'eeprom
17   {
18      if (!EECON1bits.WR)   // se e' finita la scrittura della variabile
19      {                     // scrive la successiva
20         switch (EepromStatus)
21         {
22            case 0:
23               WriteEeprom(Light1,EepromStatus);
24               EepromStatus++; // variabile successiva
25            break;
26
27            case 1:
28               WriteEeprom(Light2,EepromStatus);
29               EepromStatus++; // variabile successiva
30            break;
31
32            case 2:
33               WriteEeprom(GasOn,EepromStatus);
34               EepromStatus++; // variabile successiva
35            break;
36
37            case 3:
38               WriteEeprom(kp,EepromStatus);
39               EepromStatus++; // variabile successiva
40            break;
41
42            case 4:
43               WriteEeprom(ki,EepromStatus);
44               EepromStatus++; // variabile successiva
45            break;
46
47            case 5:
48               WriteEeprom(kd,EepromStatus);
49               EepromStatus++; // variabile successiva
50            break;
```

```
51
52              case 6:
53                 WriteEeprom(En,EepromStatus);
54                 EepromStatus++; // variabile successiva
55              break;
56
57              default:
58                 EepromFlag=0;// una volta salvate tutte le variabili disabilita la routine
59                 EECON1bits.WREN=0;  // e disabilita la scrittura su EEPROM
60                 if (En)             // abilitazione motori scritta in EEPROM
61                 {                   // per effettuare le misure senza far girare le ruote
62                    MotEnable=1;     // avvia motori, per debug=0, altrimenti = 1
63                 }
64                 else
65                 {
66                    MotEnable=0;
67                 }
68              break;
69           }  // end switch
70        }
71     }
72
73
74     //Init e Lcd Display============================
75     if (DisplayStatus)   // se = 0 sono disabilitate tutte le routine relative al display
76     {
77        if (InitFlag)
78        /* Inizializza l'LCD e le routine di movimento.
79           Il flag e' settato solo prima di entrare nel main,in questo modo l'inizializzazione
80           dell'LCD avviene una sola volta.
81           Rimane in loop ciclando i led colorati fino a quando non si tocca un pulsante
82        */
83
84        {
85           if  (BumperDx && BumperSx) // resta fermo finche' non si tocca  un baffo
86           {
87              if (TimerLcd <= 0)      // Per le temporizzazioni necessarie all'init dell'LCD
88              {
89                 if (!LcdPutCharFlag) // e' terminata la trasmissione di un byte
90                 {
91                    if (LcdStringPtr== 0xFF) // e' terminato l'invio della stringa
92                    {
93                       LcdInit();
94                    }
95                 }
96              }
97           }
98           else
99           {
100             if (FlagCmpCal)    // e' stata richiesta la calibrazione della bussola
```

```
101             {
102                CmpCal();
103             }
104             else                // avvio normale
105             {
106                SequenzaStart();// esce dallo stato init avviando le sequenze di movimento
107             }
108             if (En)             // abilitazione motori scritta in EEPROM
109             {                   // per effettuare le misure senza far girare le ruote
110                MotEnable=1;     // avvia motori, per debug=0, altrimenti = 1
111             }
112          }
113       }
114       else  // finita la fase di init, l'LCD viene usato per mostrare i parametri voluti
115       {
116         if (TimerLcd <= 0)     // e' tempo di aggiornare il display
117         {
118             TimerLcd=LcdCycle;// reset timer
119
120             if (DisplayStatus==99)
121             {
122                DisplayStatus=0;  // ha visualizzato la scritta fissa e disabilita LCD
123             }
124             else
125             {
126                if (DisplayStatus> MaxLcdStatusItem || DisplayStatus <= 0)
127                {
128                   LcdBL=0;          // LCD Backlight off
129                                     //   "0123456789012345  0123456789012345"
130                   LcdPutStringInitRom(0, "----- RUN ------\r----------------");
131                   DisplayStatus=99;
132                }
133                else
134                {
135                LcdBL=1;          // LCD Backlight on
136                Display();
137                /* Visualizza valori diversi in funzione della variabile DisplayStatus
138
139                   1: visualizza valori sensori di prossimita'
140                   2: visualizza angolo bussola
141                   3: visualizza valori gas e fotoresistenze
142                   4: visualizza soglie Light1, Light2 e GasOn
143                   5: visualizza costanti Kp, Ki, Kd, En
144
145                  99: disabilita la routine dopo la scritta fissa
146
147                      impostando un qualsiasi valore diverso da quelli elencati
148                      si scrive prima una stringa fissa e poi (al giro successivo)
149                      si disabilita la scrittura su display.
150                */
```

```
151                 }
152             }
153         }
154     }
155
156
157 //LcdPutChar ----------------------------------
158     if (LcdPutCharFlag)
159     {
160     /*
161        La routine a livello superiore ha abilitato la scrittura di un nuovo carattere
162        verso l'LCD tramite LcdPutChar()
163     */
164
165        if (!I2c[LcdPtr].Flag.Tx && !I2c[LcdPtr].Flag.Rx)
166        /*
167           se il buffer di trasmissione I2C e' libero si può passare un nuovo byte
168           all'I/O expander.
169           Per scrivere un carattere sull'LCD occorre inviare una sequenza di byte
170           tramite l'I/O expander
171        */
172        {
173           if (Lcd4_8BitFlag)   // modalita' 4 o 8 bit
174           {
175              Lcd4Bit();
176           }
177           else
178           {
179              Lcd8Bit();
180           }
181        }
182     }
183
184
185 //LcdPutString -----------------------------------------------
186     if (LcdStringPtr!=0xFF)    // c'e' una stringa da scrivere sull'LCD
187     {
188     /*
189        La routine a livello superiore ha abilitato la scrittura di una nuova stringa
190        verso l'LCD tramite LcdPutStringInit()
191     */
192        if (!LcdPutCharFlag) // e' terminata la trasmissione del byte  precedente
193        {
194           LcdPutString();
195        }
196     }
197
198 }//Init e Lcd Display
199 //===============================================
200
```

```
201  //Keypad----------------------------------------
202  if (FlagKbdIntr)                    // E' arrivato un'interrupt dall'I/O expander
203  {                                   // al quale e' collegata la tastiera
204     if (!FlagKbdStartRead)           // non e' in corso una lettura
205     {
206         FlagKbdStartRead=1;          // avvia una lettura della tastiera
207         TimerKeypad=0;               // avvia timer debounce
208         FlagKbdStartI2c=0;           // per attesa lettura I2c
209     }
210     else
211     {
212        if (TimerKeypad >= 20)         // debounce, aspetta 20ms
213        {
214           if (!FlagKbdStartI2c)       // non e' in corso una lettura I2c
215           {
216              I2c[KeypadPtr].Flag.Rx=1;// avvia sequenza lettura da I/O expander 2
217              FlagKbdStartI2c=1;
218           }
219           else
220           {
221              if (!I2c[KeypadPtr].Flag.Rx)  // il contatore di ricezione e' vuoto
222              {                             // e' finita la lettura I2c
223                 FlagKbdStartRead=0;        // e' finito il ciclo debounce,
224                                            // rimane in attesa del prossimo evento
225                 FlagKbdIntr=0;             // disabilita la routine, sara' riattivata dal
226                                            // prossimo interrupt, resetta anche eventuali
227                                            // interrupt arrivati nel frattempo
228
229                 switch (I2c[KeypadPtr].RxBuff[0])   // tasto premuto
230                 {
231                    case 0b11111110:                 // ROSSO
232                       if (FlagMenuMod)
233                       {  // se in modalita' modifica incrementa variabile
234                          LcdVar[DisplayStatus][CursorStatus]++;
235                       }
236                       else
237                       {  // altrimenti passa a menu successivo
238                          DisplayStatus++;
239                          LcdStatus=0;// ripassa per il ciclo clear del display
240                       }
241                    break;
242
243                    case 0b11111101:                 // VERDE
244                       if (FlagMenuMod)
245                       {  // se in modalita' modifica decrementa variabile
246                          LcdVar[DisplayStatus][CursorStatus]--;
247                       }
248                       else
249                       {  // altrimenti passa a menu precedente
250                          DisplayStatus--;
```

```
251                     LcdStatus=0;// ripassa per il ciclo clear del display
252                     if (DisplayStatus >=  254)
253                     {     // indietro dal menu 0 ricomincia dall'ultimo menu
254                        DisplayStatus  = MaxLcdStatusItem;
255                     }
256                     else
257                     {
258                        if (DisplayStatus < 1)
259                        {  // arrivato al primo menu disabilita terminale
260                           DisplayStatus = 90;
261                        }
262                     }
263                  }
264               break;
265
266               case 0b11111011:                 // BIANCO
267                  if (FlagMenuMod)
268                  {  // sposta il cursore dalla posizione 0 alla 3 ciclicamente
269                     CursorStatus++;
270                     if (CursorStatus>3)
271                     {
272                        CursorStatus=0;
273                     }
274                  }
275               break;
276
277               case 0b11110111:                 // BLU
278                  if (FlagMenuMod)
279                  {  // Toggle: Flag modifica menu e Lampeggio cursore LCD
280                     FlagMenuMod=0;
281                     EepromFlag=1;   // uscendo dalle modifiche abilita
282                     EepromStatus=0; // la scrittura delle variabili su EEPROM
283                  }
284                  else
285                  {
286                     if (LcdVar[DisplayStatus][4])
287                     {  // Solo se il menu e' abilitato alla modifica
288                        FlagMenuMod=1;
289                        CursorStatus=0;
290                     }
291                  }
292               break;
293
294               case 0b11110000:     // tutti i tasti insieme (solo in fase di init)
295                  FlagCmpCal=1;     // avvia procedura calibrazione bussola
296               break;
297
298               default:       // se si premono due tasti contemporanemente non fa niente
299               break;
300
```

```
301                 } // end switch
302               }
303             }
304           }
305       }
306   }
307   if (!DisplayStatus && !FlagKbdIntr && !EepromFlag)
308   {
309      UserInterfaceFlag=0; // se tutte le routine sono disabilitate, disabilita il flag
310                           // generale per non entrare piu' in questo ciclo
311                           // verra' riabilitato alla pressione di un tasto
312   }
313
314   } // UserInterface
315   /*****************************************************************************/
316
317
318   /*Display ********************************************************************
319   Visualizza valori sul display in funzione della variabile DisplayStatus
320   */
321
322   void Display (void)
323   {
324         switch (LcdStatus)
325         {
326         case 0:
327            TimerLcd=LcdCycle;                          // pausa all'avvio del ciclo
328            LcdPutChar (0x01, 0, 4);                    // clear display
329            LcdStatus ++;                               // passa allo stato successivo
330            break;
331
332         case 1:
333            LcdPutStringInitRom(0, &(LcdTable1[DisplayStatus][0]));// invia stringa all'LCD
334            TimerLcd=LcdCycle;                          // reset timer
335            LcdStatus ++;                               // passa allo stato successivo
336            break;
337
338         case 2:
339            LcdPutStringInitC2A(0x40,LcdVar[DisplayStatus][0]);// visualizza valore
340            TimerLcd=LcdCycle;                       // reset timer
341            LcdStatus ++;                            // passa allo stato successivo
342            break;
343
344         case 3:
345            LcdPutStringInitC2A(0x44,LcdVar[DisplayStatus][1]);// visualizza valore
346            TimerLcd=LcdCycle;                       // reset timer
347            LcdStatus ++;                            // passa allo stato successivo
348            break;
349
350         case 4:
```

```
351            if (FlagMenuMod)
352            {// se in modalita' modifica lampeggia il nome della variabile da modificare
353               LcdPutStringInitRom(0, &(LcdTable2[DisplayStatus+6+(CursorStatus*6)][0]));
354            }
355            else
356            {
357               LcdPutStringInitRom(0, &(LcdTable2[DisplayStatus][0]));// invia stringa all'LCD
358            }
359            TimerLcd=LcdCycle;                        // reset timer
360            LcdStatus ++;                             // passa allo stato successivo
361            break;
362
363         case 5:
364            LcdPutStringInitC2A(0x49,LcdVar[DisplayStatus][2]);// visualizza valore
365            TimerLcd=LcdCycle;                        // reset timer
366            LcdStatus ++;                             // passa allo stato successivo
367            break;
368
369         default:
370            LcdPutStringInitC2A(0x4D,LcdVar[DisplayStatus][3]);// visualizza valore
371            TimerLcd=LcdCycle;                        // reset timer
372            LcdStatus = 1;                            // ricomincia dal secondo passo
373            break;
374
375         } // end switch
376
377
378   } // Display
379   /*****************************************************************************/
380
381
382   /*LcdPutString ***************************************************************
383   Scrive sull'LCD, la stringa posta nel buffer
384   Un carattere "\r" all'interno della stringa significa "a capo" (Cursor = 0x40)
385   */
386
387   void LcdPutString(void)
388   {
389      if (LcdString[LcdStringPtr])
390      {
391         if (LcdString[LcdStringPtr] == '\r') // a capo
392         {
393            LcdPutChar(0x80+0x40, 0, 4);  // inizio seconda riga
394            LcdStringPtr++;               // al prossimo giro scrive il carattere successivo
395         }
396         else
397         {
398            LcdPutChar(LcdString[LcdStringPtr], 1, 4);   // scrive nesimo carattere
399            LcdStringPtr ++;              // al prossimo giro scrive il carattere successivo
400         }
```

```
401     }
402     else
403     {
404        LcdStringPtr=0xFF;   // ultimo passaggio, disabilita questa  routine
405     }
406
407  } // LcdPutString
408  /*****************************************************************************/
409
410
411  /*LcdPutStringInitC2A ***********************************************************
412  Inizializza la scrittura di una stringa di caratteri sull'LCD alla posizione "Cursor".
413  Converte una variabile unsigned char in tre caratteri ASCII
414  grazie a RI
415  */
416
417  void LcdPutStringInitC2A(unsigned char Cursor,unsigned char Val)
418  {
419     unsigned char u,d,c,tmp;
420
421     LcdPutChar(0x80+Cursor, 0, 4);   // posizione di partenza del cursore
422
423     LcdStringPtr=0;       // Puntatore al carattere della stringa in stampa
424                           // se diverso da 0xFF avvia la stampa della stringa
425     c=Val/100;
426     tmp=(Val-100*c);
427     d=tmp/10;
428     u=tmp-10*d;
429
430     LcdString[3]=0;                    // EOL
431     LcdString[2]="0123456789"[u];      // unita'
432     LcdString[1]="0123456789"[d];      // decine
433     LcdString[0]=" 123456789"[c];      // centinaia
434
435  } // LcdPutStringInitC2A
436  /*****************************************************************************/
437
438
439  /*LcdPutStringInitRom ***********************************************************
440  Inizializza la scrittura di una stringa di caratteri sull'LCD alla posizione "Cursor".
441  Dal momento che l'allocazione della memoria e' diversa tra memoria programma e memoria
442  ram, questa routine e' valida solo per la visualizzazione di stringhe fisse.
443  */
444
445  void LcdPutStringInitRom(unsigned char Cursor,const char rom *LcdStr)
446  {
447     char count=0;         // contatore
448     LcdStringPtr=0;       // Puntatore al carattere della stringa in stampa
449                           // se diverso da 0xFF avvia la stampa della stringa
450
```

```
451     while (*(LcdStr+count))
452     {
453        LcdString[count] = *(LcdStr+count);
454        count++;
455     }
456     LcdString[count]=0;     // nul character
457
458
459     LcdPutChar(0x80+Cursor, 0, 4);   // posizione di partenza del cursore
460
461  } // LcdPutStringInitRom
462  /*****************************************************************************/
463
464
465  /*LcdPutChar ***************************************************************
466  inizializza flag e variabili per inviare un nuovo carattere verso l'LCD.
467  Parametri:
468  LChar    = carattere da scrivere (dato o comando)
469  LRS      =1 se dato, =0 se comando
470  L4_8Bit  =4 se modalita' 4 bit, 8 se modalita' 8 bit
471  */
472
473  void LcdPutChar (char LChar, unsigned char LRS, unsigned char L4_8Bit)
474  {
475     LcdChar = LChar;  // carattere (dato o comando)
476
477     if (LRS)          // = 1 se Dato, =0 se Comando
478     {
479        LcdRS=1;
480     }
481     else
482     {
483        LcdRS=0;
484     }
485
486     if (L4_8Bit==4)
487     {
488        Lcd4_8BitFlag=1;  // = 1 se modalita' 4 bit, =0 se 8 bit
489     }
490     else
491     {
492        Lcd4_8BitFlag=0;
493     }
494
495     LcdRW=0;          // scrittura
496     LcdPutCharFlag=1; // abilita la scrittura del byte
497     LcdByteStatus=0;  // azzera il contatore di stato della routine LcdXBit
498
499  } // LcdPutChar
500  /*****************************************************************************/
```

```
501
502
503 /*LcdByteSet*****************************************************************************
504 ricostruisce il byte da inviare all'LCD tramite I2C dai singoli bit
505 */
506
507 unsigned char LcdByteSet (void)
508 {
509
510 return (((((((0x00|LcdData)<<1)|LcdNA)<<1)|LcdRW)<<1)|LcdRS)<<1)|LcdBL;
511
512 } // LcdByteSet
513 /*****************************************************************************************/
514
515
516 /*Lcd4Bit ********************************************************************************
517 Invia un byte all'LCD nella modalita' a 4 bit tramite I2C
518 L'I/O expander 1 usato per pilotare l'LCD e' il device I2c numero 3
519 Il driver LCD HD44780 con il clock interno a 270KHz richiede una pausa di 37uSec
520 tra un carattere e l'altro. Per inviare un byte tramite l'I/O expander PCF8574 con il
521 clock a 100KHz ci vogliono almeno 90uSec (8 bit + ACK * 10uSec a bit), piu' che
522 sufficienti quindi per le temporizzazioni richieste.
523 */
524
525 void Lcd4Bit (void)
526 {
527    switch (LcdByteStatus)
528    {
529       case (0):
530          LcdEN=0;
531          LcdData=LcdChar >> 4;                  // upper nibble
532          I2c[LcdPtr].Flag.Tx = 1;               // un byte da trasmettere
533          I2c[LcdPtr].TxBuff[0] = LcdByteSet() ; // compone il byte da inviare e lo passa
534                                                 // alle routine I2C. Il buffer e' stato
535                                                 // controllato prima e quindi e'
536                                                 // sicuramente vuoto
537          LcdByteStatus ++;                      // passa allo stato successivo
538       break;
539
540       case (1):
541          LcdEN=1;                               // strobe
542          LcdByteStatus ++;                      // passa allo stato successivo
543          LcdEN=0;
544       break;
545
546       case (2):
547          LcdData=LcdChar;                       // lower nibble
548          I2c[LcdPtr].Flag.Tx = 1;               // in modalita' 4 bit trasmette  un
549          I2c[LcdPtr].TxBuff[0] = LcdByteSet() ; // nibble per volta
550          LcdByteStatus ++;                      // passa allo stato successivo
```

```
551         break;
552
553         case (3):
554            LcdEN=1;                                    // strobe
555            LcdPutCharFlag = 0;  // l'invio del carattere e' finito, se la routine a
556                                 // livello superiore ha altri byte da inviare
557                                 // inizia una nuova sequenza con LcdPutChar
558            LcdEN=0;
559         break;
560
561      } // end switch
562
563   } // Lcd4Bit
564   /*****************************************************************************/
565
566
567   /*Lcd8Bit ********************************************************************
568   Invia un byte all'LCD nella modalita' a 8 bit tramite I2C, usata solo per l'init
569   L'I/O expander 1 usato per pilotare l'LCD e' il device I2c numero 3
570   Il driver LCD HD44780 con il clock interno a 270KHz richiede una pausa di 37uSec
571   tra un carattere e l'altro. Per inviare un byte tramite l'I/O expander PCF8574 con il
572   clock a 100KHz ci vogliono almeno 90uSec (8 bit + ACK * 10uSec a bit), piu' che
573   sufficienti quindi per le temporizzazioni richieste.
574   */
575
576   void Lcd8Bit (void)
577   {
578      switch (LcdByteStatus)
579      {
580         case (0):
581            LcdEN=0;
582            LcdData=LcdChar;
583            I2c[LcdPtr].Flag.Tx = 1;                    // un byte da trasmettere
584            I2c[LcdPtr].TxBuff[0] = LcdByteSet() ; // compone il byte da inviare e lo passa
585                                                        // alle routine I2C. Il buffer e' stato
                                                  // controllato prima e quindi e'
586                                                        // sicuramente vuoto
587            LcdByteStatus ++;                           // passa allo stato successivo
588         break;
589
590         case (1):
591            LcdEN=1;                                    // strobe
592            LcdPutCharFlag = 0;  // l'invio del carattere e' finito, se la routine a
593                                 // livello superiore ha altri byte da inviare
594                                 // inizia una nuova sequenza con LcdPutChar
595            LcdEN=0;;
596         break;
597
598      } // end switch
599
```

```
600  } // Lcd8Bit
601  /*****************************************************************************/
602
603
604  /*LcdInit ********************************************************************
605  Per l'inizializzazione del display LCD
606  Interfaccia a 4 bit, 2 linee di visualizzazione,e caratteri di 5 * 7 punti.
607  */
608  void LcdInit (void)
609  {
610
611     switch (LcdInitStatus)
612     {
613        case 0:
614           LcdPutChar (0x03, 0, 8);               // invia carattere 0x30 all'LCD
615           LcdInitStatus ++;                      // passa allo stato successivo
616           break;
617
618        case 1:
619           TimerLcd=5;                            // inizializza timer 5mS
620           LcdInitStatus ++;                      // passa allo stato successivo
621           break;
622
623        case 2:
624
625           LcdPutChar (0x03, 0, 8);               // invia carattere 0x30 all'LCD
626           LcdInitStatus ++;                      // passa allo stato successivo
627           break;
628
629        case 3:
630           TimerLcd=5;                            // inizializza timer 5mS
631           LcdInitStatus ++;                      // passa allo stato successivo
632           break;
633
634        case 4:
635           LcdPutChar (0x03, 0, 8);               // invia carattere 0x30 all'LCD
636           LcdInitStatus ++;                      // passa allo stato successivo
637           break;
638
639        case 5:
640           LcdPutChar (0x02, 0, 8);                  // Modalita' 4 bit
641           LcdInitStatus ++;                      // passa allo stato successivo
642           break;
643
644        case 6:
645           TimerLcd=5;                            // inizializza timer 5mS
646           LcdInitStatus ++;                      // passa allo stato successivo
647           break;
648
649        case 7:
```

```
650            LcdPutChar (0x28, 0, 4);                // 1/16 duty, 5x7 font
651            LcdInitStatus ++;                       // passa allo stato successivo
652            break;
653
654         case 8:
655            LcdPutChar (0x0C, 0, 4);                // display on, blink cursor off
656            LcdInitStatus ++;                       // passa allo stato successivo
657            break;
658
659         case 9:
660            LcdPutChar (0x06, 0, 4);                // entry mode=increment cursor position
661            LcdInitStatus ++;                       // passa allo stato successivo
662            break;
663
664         case 10:
665            LcdPutChar (0x01, 0, 4);                // clear display
666            LcdInitStatus ++;                       // passa allo stato successivo
667            break;
668
669         case 11:
670            TimerLcd=5;                             // inizializza timer 5mS
671            LcdInitStatus ++;                       // passa allo stato successivo
672            break;
673
674         case 12:
675            LcdBL=1;                                // Backlight ON
676            LcdPutStringInitRom(0,Ver);             // scrive versione
677            LcdInitStatus ++;                       // passa allo stato successivo
678            break;
679
680         case 13:
681            LcdBL=0;                                // Backlight OFF
682            LedRossoON;                             // rimane in loop ciclando il led
683            LedGialloOFF;                           // ogni 250mS
684            LedVerdeOFF;
685            LcdInitStatus ++;                       // passa allo stato successivo
686            LcdPutStringInitC2A(0x40,GasVal);       // visualizza valore sensore gas
687            break;
688
689         case 14:
690            TimerLcd=250;                           // inizializza timer 250mS
691            LcdInitStatus ++;                       // passa allo stato successivo
692            break;
693
694         case 15:
695            LedRossoOFF;
696            LedGialloON;
697            LcdInitStatus ++;                       // passa allo stato successivo
698            break;
699
```

```
700         case 16:
701            TimerLcd=250;                        // inizializza timer 250mS
702            LcdInitStatus ++;                    // passa allo stato successivo
703            break;
704
705         case 17:
706            LedGialloOFF;
707            LedVerdeON;
708            LcdInitStatus ++;                    // passa allo stato successivo
709            break;
710
711         case 18:
712            TimerLcd=250;                        // inizializza timer 250mS
713            LcdInitStatus = 13;                  // ricomincia il ciclo dei led
714            break;
715
716
717         default:
718            break;
719
720      } // end switch
721
722   } // LcdInit
723   /*****************************************************************************/
724
725
```