```
1    /*
2    contiene tutte le routine per la gestione a basso ed alto livello dell'I2C
3    */
4
5    /*I2cLowService ******************************************************************
6    Per la gestione a basso livello della comunicazione I2c
7    */
8
9    void I2cLowService (void)
10   {
11      if (I2cBusCollFlag || SSPCON2bits.ACKSTAT)
12      {  // c'e' stata una collisione sul bus o un NACK -> annnulla sequenza
13         I2cBusCollFlag = 0;          // reset stato errore
14         I2cStat = FINE;              // stato attuale = FINE sequenza
15         StopI2C();                   // invia stop
16         // non azzera il contatore byte Tx e Rx per ritentare scambio al prossimo giro
17      }
18
19      else
20      {
21      // esegue la sequenza I2c stabilita per ogni byte nel record puntato
22
23         switch (I2cStat)
24         {
25            case (START):  // e' stata eseguita una sequenza di start
26               if (I2c[I2cDevPtr].Flag.Tx > 0)// c'e' qualcosa da trasmettere ?
27               {
28                 I2cStat = WRITE;             // aggiorna stato I2c attuale
29                 SSPBUF = I2cAdr[I2cDevPtr];  // scrive indirizzo con flag R/W = write
30               }
31               else     // se non ha dati da trasmettere ne ha sicuramente da ricevere
32               {
33                 I2cStat = READ;              // aggiorna stato I2c attuale
34                 SSPBUF = I2cAdr[I2cDevPtr]+1;// scrive indirizzo con flag R/W = read
35               }
36               break;
37
38
39            case (WRITE):
40               /* invia il byte Nesimo
41                  controlla se sono stati inviati tutti i byte
42                  controlla se ci sono byte da ricevere
43                  altrimenti chiude la sequenza
44               */
45               SSPBUF = I2c[I2cDevPtr].TxBuff[Ptr.Tx];    // invio primo byte
46               Ptr.Tx ++;                                 // byte Tx successivo
47               if (Ptr.Tx >= I2c[I2cDevPtr].Flag.Tx)      // ha finito Tx
48               {
49                  if (I2c[I2cDevPtr].Flag.Rx > 0)  // ha qualcosa da ricevere ?
50                  {
```

```
51                   I2cStat = RSTART;              // inizia sequenza di ricezione
52                }
53                else                              // non ha niente da ricevere
54                {
55                   I2cStat = STOP;                // invia stop regolare
56                }
57             }
58             else                                 // ha ancora byte da trasmettere
59             {
60                I2cStat = WRITE;                  // inizia sequenza invio byte(s) successivo(i)
61             }
62             break;
63
64
65          case (READ):
66             SSPCON2bits.RCEN = 1;                        // avvia ricezione di un byte
67             Ptr.Rx ++;                                   // byte Rx successivo
68             if (Ptr.Rx >= I2c[I2cDevPtr].Flag.Rx)        // ha finito Rx ?
69             {
70                I2cStat = NACK;                           // SI, invia NACK (fine Rx)
71             }
72             else
73             {
74                I2cStat = ACK;                            // NO. invia ACK (prosegue Rx)
75             }
76             break;
77
78
79          case (ACK):
80             I2c[I2cDevPtr].RxBuff[Ptr.Rx-1] = SSPBUF; // memorizza byte Nesimo ricevuto
81                                                       // il Ptr era stato gia' incrementato
82             I2cStat = READ;                           // altri byte da ricevere
83             AckI2C();
84             break;
85
86
87          case (NACK):
88             I2c[I2cDevPtr].RxBuff[Ptr.Rx-1] = SSPBUF; // memorizza byte Nesimo ricevuto
89                                                       // il Ptr era stato gia' incrementato
90             I2cStat = STOP;                           // ricevuto ultimo byte
91             NotAckI2C();
92             break;
93
94
95          case (RSTART):
96             // reinizializza bus senza rilasciarlo
97             I2cStat = ADRR;              // al prossimo giro invia inizio ricezione
98             RestartI2C();
99             break;
100
```

```
101
102           case (ADRR):
103              I2cStat = READ;               // al prossimo giro inizia la ricezione
104              SSPBUF = I2cAdr[I2cDevPtr]+1;// scrive indirizzo con flag R/W = read
105              break;
106
107
108           case (STOP):
109              I2cStat = FINE;               // aggiorna stato I2c attuale
110              I2c[I2cDevPtr].Flag.Rx = 0; // non ci sono altri byte da Tx o Rx le routine a
111              I2c[I2cDevPtr].Flag.Tx = 0; // livello piu' alto possono scambiare altri dati
112              StopI2C();                    // invia stop
113              break;
114
115
116           case (FINE):
117              I2cBusyFlag = 0;              // La comunicazione I2c e' finita
118              break;
119
120
121           default:
122              break;
123
124        }  // end switch
125     }  // end else
126
127  } // I2cLowService
128  /*****************************************************************************/
129
130
131  /*I2cHighService ***************************************************************
132  Per la gestione ad alto livello della comunicazione I2c
133  */
134
135  void I2cHighService (void)
136  {
137     /*
138      inizializza sequenza I2c azzerando flag, contatori e buffer
139      lancia il primo start del protocollo
140      il resto dello scambio del pacchetto sara' eseguito tramite interrupt
141     */
142     Ptr.Tx = 0;           // Azzera puntatore buffer dati da trasmettere
143     Ptr.Rx = 0;           // Azzera puntatore buffer dati da ricevere
144     I2c[I2cDevPtr].RxBuff[0] = 0; // Azzera buffer ricezione
145     I2c[I2cDevPtr].RxBuff[1] = 0; // Azzera buffer ricezione
146     I2cEventFlag = 0;     // Evento I2c sulla porta SSP
147     I2cBusCollFlag = 0;   // Collisione su bus I2c
148     I2cBusyFlag = 1;      // E' in corso una comunicazione I2c
149     I2cStat = START;      // Contatore stati I2c
150
```

```
151     StartI2C();          // inizia scambio I2c con il primo stato
152
153  } // I2cHighService
154  /*****************************************************************************/
155
```