

```

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

```

```

47         I2cStat = RSTART;           // inizia sequenza di ricezione
48     }
49     else                               // non ha niente da ricevere
50     {
51         I2cStat = STOP;             // invia stop regolare
52     }
53 }
54 else                                   // ha ancora byte da trasmettere
55 {
56     I2cStat = WRITE;               // inizia sequenza invio byte(s) successivo(i)
57 }
58 break;
59
60
61 case (READ):
62     SSPCON2bits.RCEN = 1;           // avvia ricezione di un byte
63     Ptr.Rx ++;                       // byte Rx successivo
64     if (Ptr.Rx >= I2c[I2cDevPtr].Flag.Rx) // ha finito Rx ?
65     {
66         I2cStat = NACK;             // SI, invia NACK (fine Rx)
67     }
68     else
69     {
70         I2cStat = ACK;             // NO. invia ACK (prosegue Rx)
71     }
72     break;
73
74
75 case (ACK):
76     SSPBUF = I2c[I2cDevPtr].RxBuff[Ptr.Rx-1]; // memorizza byte Nesimo ricevuto
77                                             // il Ptr era stato già incrementato
78     I2cStat = READ;                 // altri byte da ricevere
79     AckI2C();
80     break;
81
82
83 case (NACK):
84     SSPBUF = I2c[I2cDevPtr].RxBuff[Ptr.Rx-1]; // memorizza byte Nesimo ricevuto
85                                             // il Ptr era stato già incrementato
86     I2cStat = STOP;                 // ricevuto ultimo byte
87     NotAckI2C();
88     break;
89
90
91 case (RSTART):
92     // reinizializza bus senza rilasciarlo

```

```
93         I2cStat = ADDR;           // al prossimo giro invia inizio ricezione
94         RestartI2C();
95         break;
96
97
98     case (ADDR):
99         I2cStat = READ;           // al prossimo giro inizia la ricezione
100        SSPBUF = I2cAdr[I2cDevPtr]+1; // scrive indirizzo con flag R/W = read
101        break;
102
103
104     case (STOP):
105         I2cStat = FINE;           // aggiorna stato I2c attuale
106         I2c[I2cDevPtr].Flag.Rx = 0; // non ci sono altri byte da Tx o Rx le routine a
107         I2c[I2cDevPtr].Flag.Tx = 0; // livello pi~ alto possono scambiare altri dati
108         StopI2C();               // invia stop
109         break;
110
111
112     case (FINE):
113         I2cBusyFlag = 0;         // La comunicazione I2c è finita
114         break;
115
116
117     default:
118         break;
119
120     } // end switch
121 } // end else
122
123
124 } // I2cLowService
125 /*****/
126
```