```
/*I2cLowService ******************************************************************
Per la gestione a basso livello della comunicazione I2c
*/

void I2cLowService (void)
{
    if (I2cBusCollFlag || SSPCON2bits.ACKSTAT)
    {   // c'è stata una collisione sul bus o un NACK -> annnulla sequenza
        I2cBusCollFlag = 0;          // reset stato errore
        I2cStat = FINE;              // stato attuale = FINE sequenza
       StopI2C();                        // invia stop
       // non azzera il contatore byte Tx e Rx per ritentare scambio al prossimo giro
    }

    else
    {
    // esegue la sequenza I2c stabilita per ogni byte nel record puntato

        switch (I2cStat)
        {
            case (START):   // è stata eseguita una sequenza di start
               if (I2c[I2cDevPtr].Flag.Tx > 0)// c'è qualcosa da trasmettere ?
               {
               I2cStat = WRITE;            // aggiorna stato I2c attuale
               SSPBUF = I2cAdr[I2cDevPtr]; // scrive indirizzo con flag R/W = write
                }
                else        // se non ha dati da trasmettere ne ha sicuramente da ricevere
                {
            I2cStat = READ;              // aggiorna stato I2c attuale
            SSPBUF = I2cAdr[I2cDevPtr]+1;// scrive indirizzo con flag R/W = read
            }
                break;


            case (WRITE):
                /* invia il byte Nesimo
                    controlla se sono stati inviati tutti i byte
                    controlla se ci sono byte da ricevere
                    altrimenti chiude la sequenza
                */
                SSPBUF = I2c[I2cDevPtr].TxBuff[Ptr.Tx];    // invio primo byte
                Ptr.Tx ++;                                 // byte Tx successivo
                if (Ptr.Tx >= I2c[I2cDevPtr].Flag.Tx)      // ha finito Tx
                {
                if (I2c[I2cDevPtr].Flag.Rx > 0)  // ha qualcosa da ricevere ?
                {
```

```
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
```

```
                I2cStat = ADRR;              // al prossimo giro invia inizio ricezione
                RestartI2C();
                break;


            case (ADRR):
                I2cStat = READ;              // al prossimo giro inizia la ricezione
                SSPBUF = I2cAdr[I2cDevPtr]+1;// scrive indirizzo con flag R/W = read
                break;


            case (STOP):
                I2cStat = FINE;             // aggiorna stato I2c attuale
            I2c[I2cDevPtr].Flag.Rx = 0; // non ci sono altri byte da Tx o Rx le routine a
            I2c[I2cDevPtr].Flag.Tx = 0; // livello pi˘ alto possono scambiare altri dati
            StopI2C();                      // invia stop
                break;


            case (FINE):
                I2cBusyFlag = 0;               // La comunicazione I2c è finita
                break;


            default:
                break;

        }   // end switch
    }   // end else


} // I2cLowService
/*****************************************************************************/

```