```
1
2       ...
3
4
5   /*=========================================================================*/
6
7   // Low priority interrupt vector
8
9   #pragma code LowVector = 0x18
10  void InterruptVectorLow (void)
11  {
12    _asm
13      goto InterruptHandlerLow //jump to interrupt routine
14    _endasm
15  }
16
17  //---------------------------------------------------------------------------
18  // Low priority interrupt routine
19
20  #pragma code
21  #pragma interruptlow InterruptHandlerLow
22
23  /*=========================================================================*
24
25  /*IntServiceRoutine*************************************************************/
26  void InterruptHandlerLow (void)
27
28  {
29       if (!EncoderRint && !EncoderLint && !PIR1bits.SSPIF && !PIR2bits.BCLIF)
30       //se non è interrupt previsto c'è un errore
31      {
32              while (1)       // blocca l'esecuzione del programma
33              {
34              MotEnable = 0;  // ferma i motori
35              Buzzer=1;       // suona il beep
36              LedRossoOFF;
37              LedGialloON;
38              LedVerdeON;
39              }
40       }
41
42      if (EncoderLint) // impulso dall'encoder SX?
43      {
44       /* encoder in quadratura, se al cambiamento di stato i due segnali non sono in fase
45          la direzione è = FWD, altrimenti è REW
46      */
47        if (EncoderLpulse != EncoderLdir)
48        {
49            EncoderLcount ++;   // allora incrementa contatore
50        }
51        else
52        {
53            EncoderLcount --;   // altrimenti decrementa contatore
54        }
55        EncoderLint = 0;
56      }
57
58
59      if (EncoderRint) // ripete per l'altro encoder
60      {
61            if (EncoderRpulse != EncoderRdir)
62        {
63            EncoderRcount ++;   // allora incrementa contatore
64        }
65        else
```

```
      {
          EncoderRcount --;   // altrimenti decrementa contatore
      }
      EncoderRint = 0;   // reset interrupt flag
    }


    if  (PIR1bits.SSPIF)                // un evento I2c è stato completato
    {
            PIR1bits.SSPIF = 0;         // reset dell'interrupt I2c
            I2cEventFlag = 1;               // sarà eseguita la I2cLowService
     }


     if  (PIR2bits.BCLIF)               // c'è stata una collisione sul bus I2c
    {
            PIR2bits.BCLIF = 0;         // reset dell'interrupt I2c
            I2cEventFlag = 1;               // sarà eseguita la I2cLowService
            I2cBusCollFlag = 1;         // avverte della collisione
     }


}   // Low Priority IntServiceRoutine
/*******************************************************************************/

    ...
```