```
+-----------------------------------------------------------+
| ME_I2CSW  - Bus I2C software                              |
+-----------------------------------------------------------+
| Versione: 0.10B                                           |
| Revisione Documento: 2014-11-04 (2014-04-07)              |
| Autore: Roberto Gatti                                     |
| Tipo: Componente PSoC                                     |
+-----------------------------------------------------------+
|                             (c) 2014 Movie Engineering Srl |
+-----------------------------------------------------------+
```

INTRODUZIONE

Componente che implementa un master I2C software su due pin qualsiasi.

La necessità di questo componente è dovuta al fatto che ci sono dei malfunzionamenti sia nel componente Cypress che nell'hardware (a quanto pare).

La velocità delle transazioni I2C è impostabile mediante parametro.

Vengono fornite, per le primitive I2C, anche delle funzioni "alias" definite come quelle della libreria standard PSoC. Questo rende il componente perfettamente compatibile con quello Cypress.

PARAMETRI DEL COMPONENTE PSoC

_PERIOD : tempo di una singola transazione I2C (ad es. Start, Stop, ecc.)
I valori standard sono:
PERIOD = 5 per 400 kBps
PERIOD = 20 per 100 kBps

MECCANISMIO PRINCIPI GENERALI DI FUNZIONAMENTO

Vedere le specifiche I2C sui datasheet.

Per scrivere NEI registri della periferica slave:

1) Spedire una start-sequence
2) Spedire l'indirizzo della periferica (pari)
3) Spedire il numero del registro
4) Spedire i dati (quelli che si aspetta)
5) Spedire la stop-sequence

Per leggere DAI registri della periferica:

1) Spedire una start-sequence
2) Spedire l'indirizzo della periferica (pari)
3) Spedire il numero del registro
4) Spedire un'altra start-sequence
5) Spedire l'indirizzo della periferrica (dispari)
6) Ricevere i dati (quelli che manda)
7) Spedire la stop-sequence

Queste sono solo indicazioni di massima, l'effettivo funzionamento dello specifico periferico potrebbe (e di solito lo fa) variare (vedere datasheet specifico).

Esempi di transazioni, legenda: L = low, H = high, . = L oppure H

Sequenza di start :

```
      SDA :  LLLL
      SCL :  ..LL

  Sequenza di restart :
     SDA :  HHLLLL
     SCL :  .HHHLL

  Sequenza di stop :
     SDA :  ..LLLLHHH
     SCL :  LLLLHHHHH

  Sequenza di send (x, y, z = bits) :
     SDA : xxxxxx yyyyyy zzzzzz etc.
     SCL : .HHHLL LHHHLL LHHHLL etc.

  Lettura (n)ack:
     SDA : HH????
     SCL : .HHHLL

  Sequenza di recv ( ?? = lettura )
     SDA : HH?? HH?? etc..
     SCL : .HHL LHHL etc...

  Spedizione (n)ack :
     SDA : .....
     SCL : .HHLL



DESCRIZIONE API
===============

  Considerazioni generali sull'interfaccia di programmazione (es. librerie richiamate)

  Variabili esportate
  -------------------

    _PERIOD : periodo del singolo evento I2C in microsecondi (i ritardi tra la variazione
              dei segnali sulle linee sono un quarto e metà di questo tempo)
              Nota: le singole transazioni potrebbero durare complessivamente più o meno questo
                    tempo.
              Nota: questa variabile viene inizializzata col valore dell'unico parametro del
                    componente.


  Funzioni Esportate
  ------------------

    _InitI2C() : inizializza il bus mettendo entrambe le linee alte.

    _StartI2C() : spedisce una start-sequence

    _RestartI2C() : spedisce una re-start-sequence

    _StopI2C : manda una stop sequence sul bus

    _SendI2C( byte ) : manda un byte sul bus (MSBit prima)
                          Restituisce il valore di (n)ack

    _RecvI2C( ack ): legge un byte dal bus (MSBit prima)
                     Spedisce il valore di (n)ack

  Alias per compatibilità componente originale PSoC:

    void `$INSTANCE_NAME`_Start(void);
    BOOL `$INSTANCE_NAME`_MasterSendStart(BYTE address, BYTE read);
    BOOL `$INSTANCE_NAME`_MasterSendRestart(BYTE address, BYTE read);
    void `$INSTANCE_NAME`_MasterSendStop(void);
    void `$INSTANCE_NAME`_MasterWriteByte(BYTE b);
    BYTE `$INSTANCE_NAME`_MasterReadByte(BOOL ack);
```

ESEMPIO DI UTILIZZO
==========

```
    void MASTERINO_IOPort_Write(BYTE ioport_number, WORD w)
    {
        MASTERINO_I2C_PERIOD = MASTERINO_ioports_i2c_period ;           // Imposta periodo bus
        i2c

        MASTERINO_ioports[ioport_number].output_latch = w ;         // Aggiorna il latch di
        uscita

      // Seleziona il registro da usare

        MASTERINO_I2C_MasterSendStart(MASTERINO_ioports[ioport_number].address, I2C_WRITE);
        MASTERINO_I2C_MasterWriteByte(MASTERINO_IOPORT_REG_OUT_0);

      // Scrive i byte componenti i 16 bit da scrivere

        MASTERINO_I2C_MasterWriteByte(L(w));
        MASTERINO_I2C_MasterWriteByte(H(w));
        MASTERINO_I2C_MasterSendStop();
    }
```