

# Compte rendu : Feature #13387

## WifiBot - Développement d'un soft I<sup>2</sup>C "LED" pour carte synthèse PIC18F45X50

### Recherches aux préalables :

Val maximal de transmission du bus I<sup>2</sup>C : 400 kHz avec le pic1845k50

Dans le cadre de la maquette, la valeur de transmission est mise à 100 kHz pour avoir un fonctionnement standard du bus I<sup>2</sup>C.

Bus I<sup>2</sup>C : sur le PIC1845K50 :

- broche **RB0** : **SDA** (data)
- broche **RB1** : **SCL** (clock)

Fonctionnement du projet :

Le code est réalisé uniquement en interruption, la fonction "while" du main ne fait donc rien.

```
73 |         while(1)
74 |         {
75 |             |
76 |         }
77 |
78 |     }
79 |
```

Pour réaliser le code du projet, je suis passé par le MPLAB Code Configurator (ou MCC), qui permet de configurer le port I<sup>2</sup>C plus efficacement qu'à la main.

Tous les fichiers du projet sont donc répertoriés dans les "MCC Generated Files" dans les fichiers headers et dans les fichiers sources.

**Toutes les fonctions principales sont répertoriées dans le fichier i2c\_slave.c.**

Pour lire la valeur envoyé par l'outillage, on regarde dans la fonction I2C1\_SlaveDefRdInterruptHandler(), les valeurs lues par le buffer sont enregistrés dans la variable **i2c1RdData**.

```

246 static void I2C1_SlaveDefRdInterruptHandler() {
247     i2c1RdData = I2C1_SlaveGetRxData();
248     if (((i2c1RdData)=='a') || ((i2c1RdData)==1))
249     {
250         LATDbits.LATD0=1;
251         LATDbits.LATD1=0;
252     }
253     else
254     {
255         LATDbits.LATD1=1;
256         LATDbits.LATD0=0;
257     }
258 }

```

Pour allumer les LEDs conformément au cahier des charges, j'ai rajouté les lignes 248 à 257.

La broche RD1 pilotée par le **LATD1 allume une LED rouge**, et la broche RD0 pilotée par le **LATD0 allume une LED verte**.

Ici le programme teste la réception d'un caractère 'a' ou bien d'un entier '1'.

Pour changer les valeurs testées, il faut donc venir modifier cette condition.

A noter que tant qu'aucune communication I2C n'est effectuée, aucune LED ne s'allumera.

Pour savoir et changer l'adresse du PIC18F45K50 il faut regarder la ligne 50 du fichier i2c1\_slave.c :

```

50 #define I2C1_SLAVE_ADDRESS 80

```

Pour l'instant l'adresse est 80, ou 0x50 en hexadécimal. (Fichier i2c1\_slave.c)

Pour tester le programme, il a fallu développer un outillage, qui sera fourni avec le projet.

### Outillage :

Il consiste en deux projets supportés par une arduino. En premier lieu, le projet nommé "Arduino\_scanner\_code" a pour but de vérifier qu'une connexion I<sup>2</sup>C est bien établie, et écrit l'adresse de l'esclave sur le port série (outil → moniteur série). On pourra vérifier que le code détecte bien le PIC, et à l'adresse 0x50.

Le deuxième projet, nommé "wifi\_bot\_master", a pour but de communiquer via I<sup>2</sup>C avec le PIC, grâce à la fonction "Wire.write". La fonction Serial.print(x) permet de surveiller les valeurs effectivement envoyées en I<sup>2</sup>C en les réécrivant sur le port série. Si l'adresse de l'esclave testée n'est plus 0x50, il faut modifier les fonctions suivantes `Wire.begin(0x50);` et `Wire.beginTransmission(0x50)`

La variable x est incrémenté pour voir effectivement que la LED verte s'allume pour une seule valeur sur 10 (x va de 0 à 9 dans le code fourni).

Les fonctions digitalWrite(12,HIGH) et digitalWrite(12,LOW) permettent de faire clignoter une LED externe ou bien la LED nommée L sur la arduino, permettant de suivre le début et la fin de chaque itération de la fonction principale grâce au delay(200).

Les serial.prints servent à suivre la valeur envoyée en l'écrivant sur le port série.

Pour modifier la vitesse des itérations de la arduino (et donc la durée d'allumage des LEDs), il suffit de modifier les delay. Pour modifier le type de donnée, il suffit de modifier la description de x. Par exemple pour envoyer des caractères (ici de 'a' à 'g') :

```
byte x='a';

void loop()
{
  digitalWrite(12,HIGH); //begining program visual identification

  Wire.beginTransmission(0x50); // transmit to device #0x50
  Wire.write(x);                // sends one byte
  Wire.endTransmission();       // stop transmitting

  Serial.println(x); // allows to see the value of x on serial port
  Serial.println("\n");

  x++;

  if (x=='h') // resets x
  {
    x='a';
  }

  delay(200);
  digitalWrite(12,LOW); //program ends
  delay (200);
}
```

A noter : ici le code envoie le caractère sous forme de code ASCII, pour envoyer vraiment un caractère, on note : char x = 'a'.

Ou pour envoyer des entiers (de 0 à 9) :

```
byte x=0;

void loop()
{
  digitalWrite(12,HIGH); //begining program visual identification

  Wire.beginTransmission(0x50); // transmit to device #0x50
  Wire.write(x);                // sends one byte
  Wire.endTransmission();       // stop transmitting

  Serial.println(x); // allows to see the value of x on serial port
  Serial.println("\n");

  x++;

  if (x==10) // resets x
  {
    x=0;
  }

  delay(200);
  digitalWrite(12,LOW); //program ends
  delay (200);
}
```