ECAM
Brussels Engineering School

vinci
HAUTE ÉCOLE LÉONARD DE VINCI

E3020 Systèmes embarqués

CM4-I2C APPLICATIONS

Cédric Marchand – mar@ecam.be

06/02/2018

Surprise…

Objectifs

1. Module I2C dans un microcontrôleur (PIC18F46K22)
2. I2C et Arduino
3. Capteur : accéléromètre (MMA7455L)

Bus : I2C – Ressources

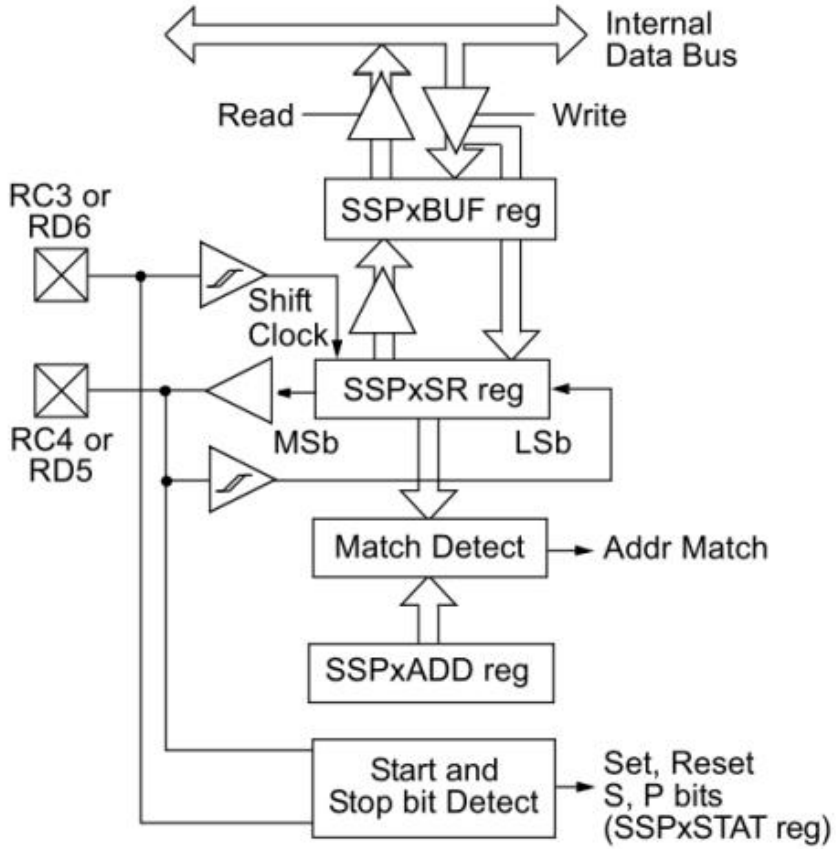
- Dans le microcontroleur, comment feriez-vous le périphérique qui est compatible avec le protocole I2C ?
- Quelles composants numériques sont nécessaires ?

CFR – datasheet PIC18

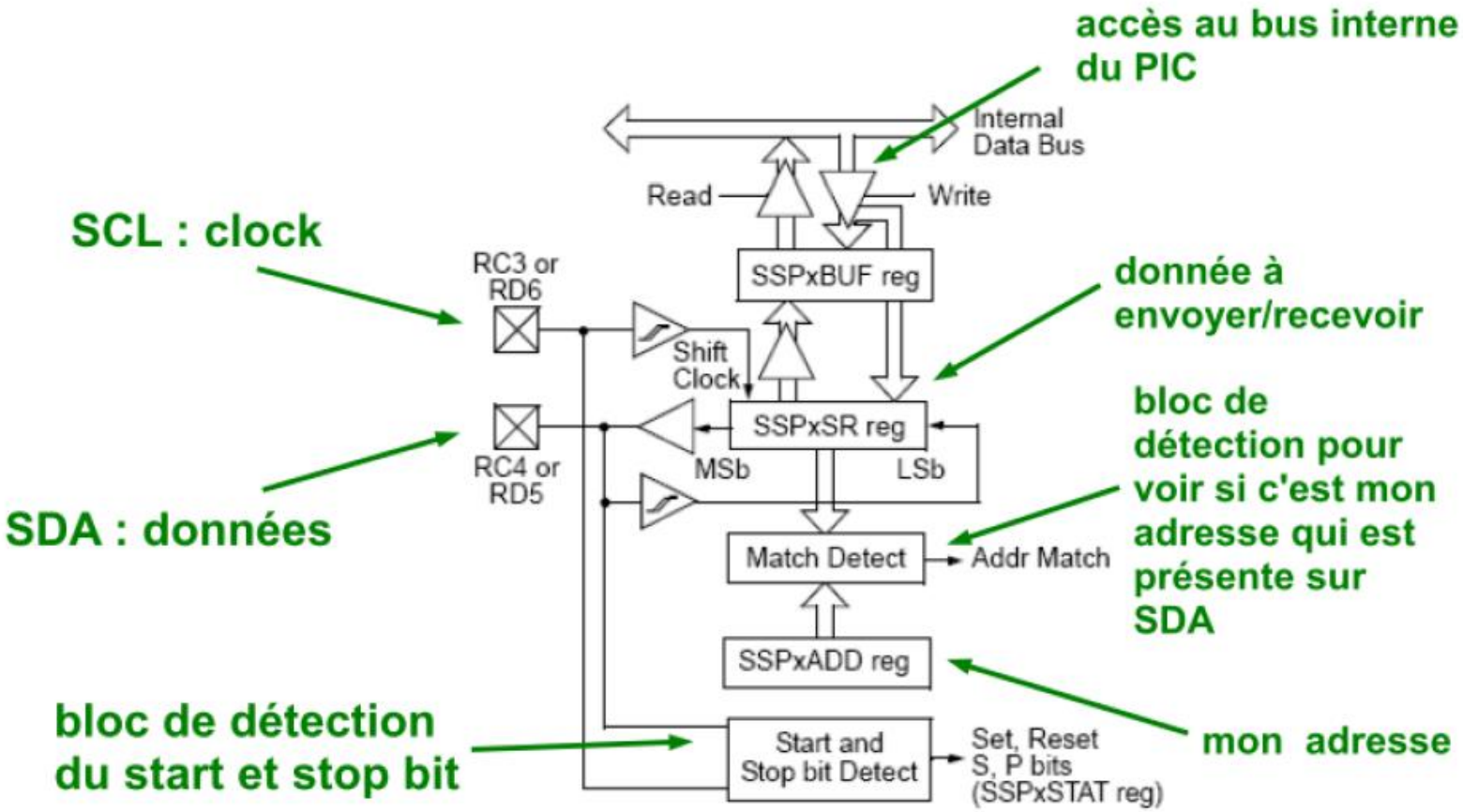
Curiosity:PIC18F46K22 (esclave)

Vue schématique :

- Adresse du µcont :
  `SSPxADD`
- Registre à décalage :
  `SSPxSR`
- Registre buffer :
  `SSPxBUF`
- Détection d'un stop et start bit
- Peut être maître ou esclave

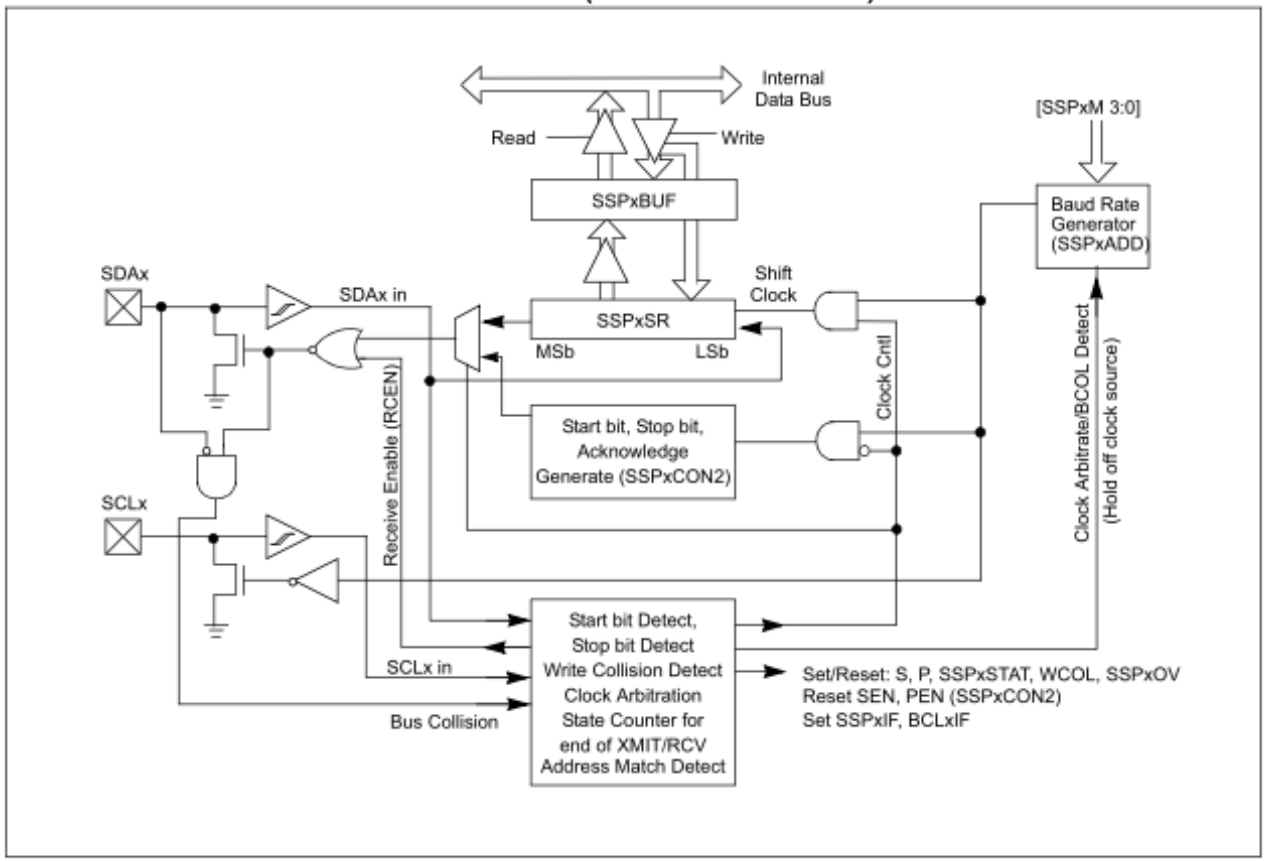


Bus : I2C – Ressources



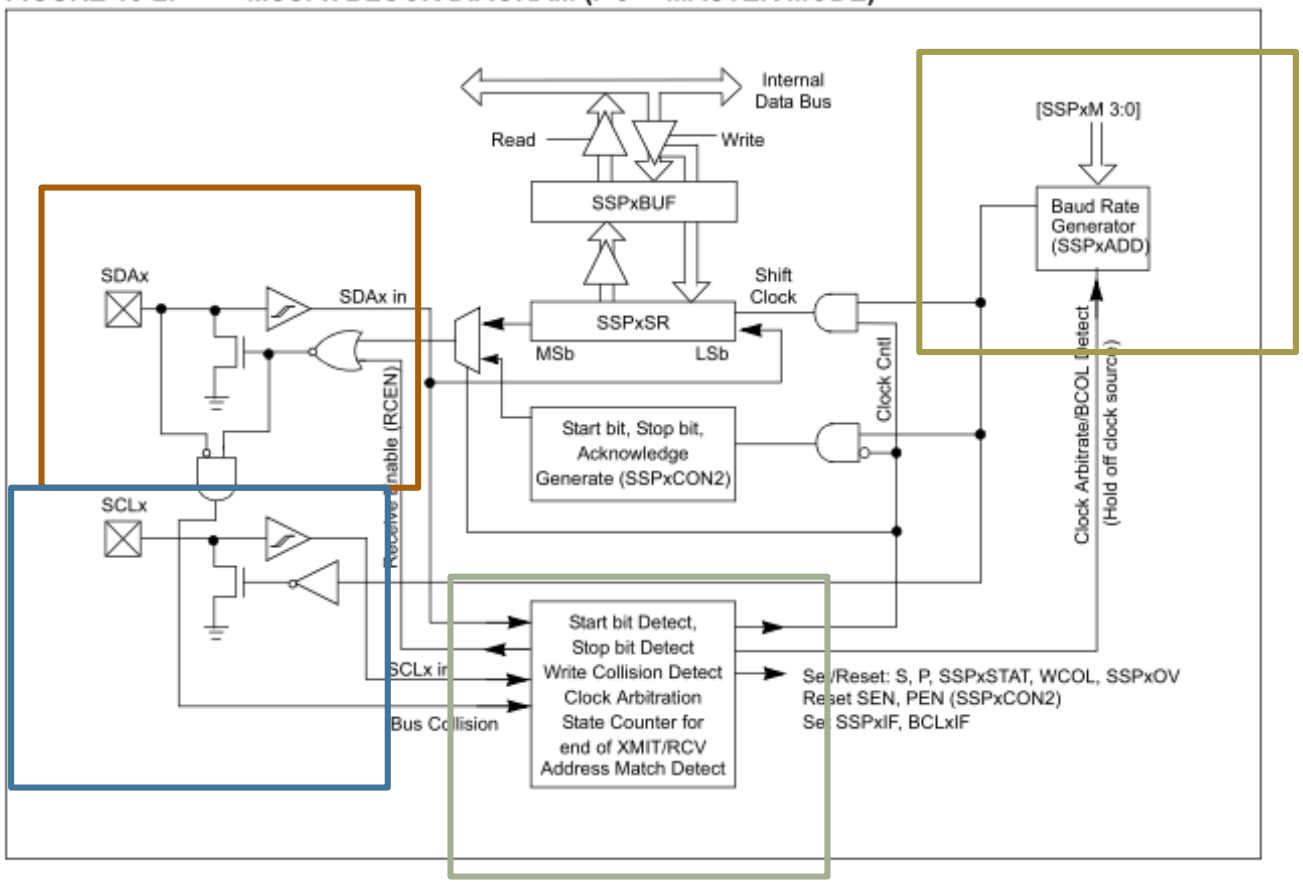
Curiosity:PIC18F46K22

FIGURE 15-2: MSSPx BLOCK DIAGRAM (I^2C™ MASTER MODE)



Curiosity:PIC18F46K22 (master)

FIGURE 15-2: MSSPx BLOCK DIAGRAM (I^2C™ MASTER MODE)



Bus : I2C – Ressources

- Dans le PIC18F, le périphérique compatible s'appelle le MSSP (Master Synchronous Serial Port). Il y en a deux MSSP1 et MSSP2.
- Il y a trois registres de configuration :
  - SSPxSTAT : qui donne le status
    - Stopbit/startbit/adresse détectée/bufferfull/....
  - SSPxCON1 et SSPxCON2 :
    - configure la communication.

Bus : I2C – Ressources

- Séquence pour configurer la communication en **esclave** :
  - 1. Mettre les pins en entrées :
    - MSSP1 : RC3/SCL1 et RC4/SDA1
    - MSSP2 : RD6/SCL2 et RD5/SDA2
    - Et connecter des résistances de pull-up
  - 2. Génèrer une interruption s'il y a matching de l'adresse.
  - 3. A la fin d'une transmission, ack automatique et transfert de SSPxSR dans SSPxBUF. SSPxIF est activé.
  - 4. Il faut alors lire dans le registre SSPxBUF doit être effacé avant le nouveau mot et SSPxIF doit être mis à zéro.

Bus : I2C – Ressources

- Séquence pour configurer la communication en **maître**:
  - Trop complexe pour le cours.
  - Il faut gérer :
    - La fréquence de clock
    - La détection de collisions
    - La détection d'acknowledge
    - Générer le start et le stop bit
    - ...

Librairie .h ou dans le code

Faites un set de fonctions qui permettent de :

En mode maître

Initialiser la puce en mode maître

Tester si une transmission est en cours

Lancer une strat condition

Lancer une restart condition

Lancer une stop condition

Permettre une écriture

Permettre une lecture

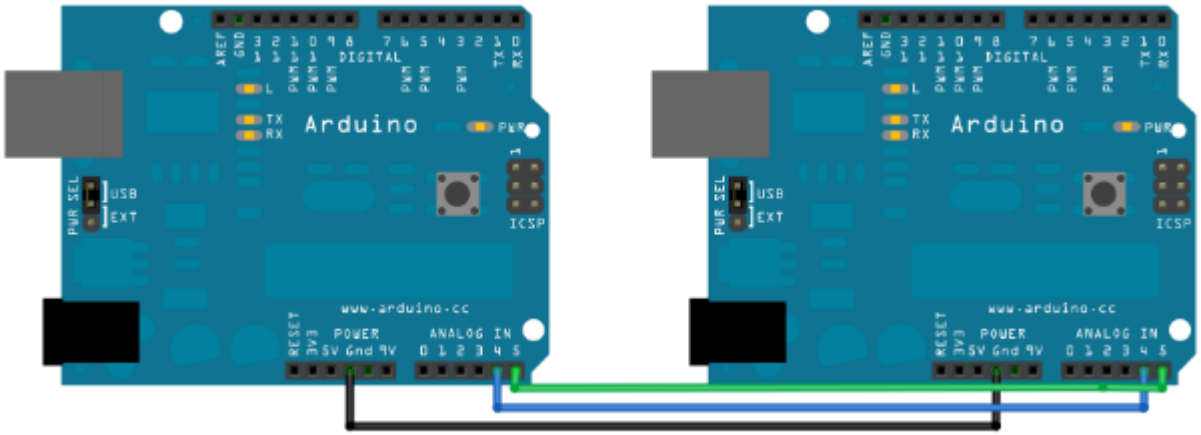
But du laboratoire : bibliothèque I2C

- Faire le travail de comprendre le fonctionnement d'un capteur
- Créer une suite de fonctions pour réaliser un échange d'information via I2C entre le microcontrôleur et le capteur.

I2C et Arduino

Bibliothèque Wire

Board	I2C / TWI pins
Uno, Ethernet	A4 (SDA), A5 (SCL)



I2C et Arduino

Comment cela fonctionne-t-il ?

I2C et Arduino

MASTER READER

```
#include <Wire.h>

void setup() {
  Wire.begin();        // join i2c bus (address optional for master)
  Serial.begin(9600);  // start serial for output
}

void loop() {
  Wire.requestFrom(8, 6);    // request 6 bytes from slave device #8

  while (Wire.available()) { // slave may send less than requested
    char c = Wire.read(); // receive a byte as character
    Serial.print(c);         // print the character
  }

  delay(500);
}
```

SLAVE SENDER

```
#include <Wire.h>

void setup() {
  Wire.begin(8);                // join i2c bus with address #8
  Wire.onRequest(requestEvent); // register event
}

void loop() {
  delay(100);
}

// function that executes whenever data is requested by master
// this function is registered as an event, see setup()
void requestEvent() {
  Wire.write("hello "); // respond with message of 6 bytes
  // as expected by master
}
```

I2C et Arduino

MASTER WRITER

```
#include <Wire.h>

void setup() {
  Wire.begin(); // join i2c bus (address optional for master)
}

byte x = 0;

void loop() {
  Wire.beginTransmission(8); // transmit to device #8
  Wire.write("x is ");        // sends five bytes
  Wire.write(x);              // sends one byte
  Wire.endTransmission();    // stop transmitting

  x++;
  delay(500);
}
```

SLAVE RECEIVER

```
#include <Wire.h>

void setup() {
  Wire.begin(8);                // join i2c bus with address #8
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);           // start serial for output
}

void loop() {
  delay(100);
}

// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany) {
  while (1 < Wire.available()) { // loop through all but the last
    char c = Wire.read(); // receive byte as a character
    Serial.print(c);         // print the character
  }
  int x = Wire.read();    // receive byte as an integer
  Serial.println(x);         // print the integer
}
```

I2C et capteur

±2g/±4g/±8g Three Axis Low-g Digital Output Accelerometer

The MMA7455L is a Digital Output (I^2C/SPI), low power, low profile capacitive micromachined accelerometer featuring signal conditioning, a low pass filter, temperature compensation, self-test, configurable to detect 0g through interrupt pins (INT1 or INT2), and pulse detect for quick motion detection. 0g offset and sensitivity are factory set and require no external devices. The 0g offset can be customer calibrated using assigned 0g registers and g-Select which allows for command selection for 3 acceleration ranges (2g/4g/8g). The MMA7455L includes a Standby Mode that makes it ideal for handheld battery powered electronics.

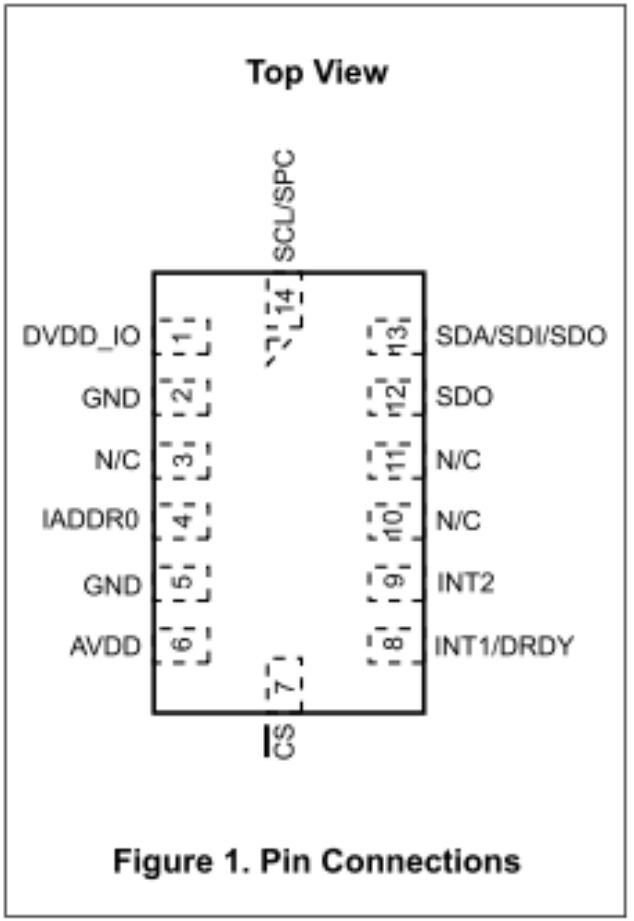


Figure 1. Pin Connections