

protocole I2C

I2C caractéristiques principales

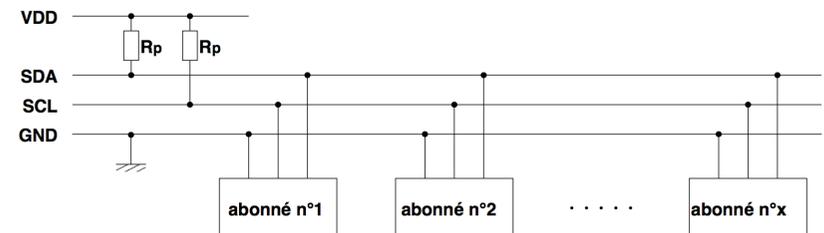
- I2C = IIC = Inter Integrated Circuit
- Protocole défini dans les années 80 par Philips.
- Protocole simple et très diffusé.
- Jusqu'à 128 abonnés (version de base) communiquent sur 3 fils :
 - SCL (horloge),
 - SDA (data),
 - GND (tension de référence).
- Abonnés Plug and Play acceptant le Hot Plug (branchement à chaud).
- Bus multi-maitres, tout abonné peut devenir maître du bus.
- Arbitrage décentralisé.
- débit de 100Kbauds à 3.4Mbauds.
- Adaptation du débit en fonction de l'abonné.
- Permet la communication entre différentes technologies (5 et 3.3V).

I2C Glossaire

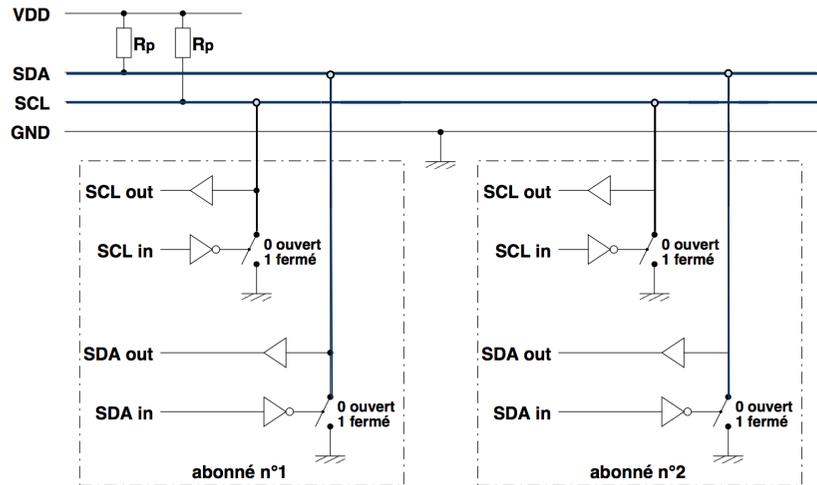
- abonné tout élément connecté sur le bus.
- émetteur tout abonné qui envoie des données sur SDA.
- récepteur tout abonné qui reçoit des données de SDA.
- maître tout abonné qui démarre et termine un échange.
Le maître place l'horloge sur SCL.
- esclave tout abonné adressé par un maître.
Un esclave à la possibilité de ralentir l'horloge du maitre.
- adresse numéro attribué à un esclave.
Sur le bus tous les esclaves ont une adresse unique.
- échange dialogue entre un maitre et un esclave.
Il commence par une adresse émise par le maitre, suivie d'une ou plusieurs données émises par le maitre ou l'esclave.
Un maitre peut chainer plusieurs échanges d'affilé.
- arbitrage résolution du conflit d'un accès simultané par 2 maîtres.

I2C cablage

- Les lignes SCL et SDA sont à VDD si aucun abonné ne parle.
- Pour mettre 1 sur SCL ou SDA, un abonné programme le port en entrée, la résistance Rp se charge de tirer la ligne à 1
- Pour mettre 0 sur SCL ou SDA, un abonné doit écrire un 0, c.-à-d. relier la ligne à la masse.
- Il ne peut jamais y avoir de conflit électrique (court-circuit VDD-GND).



I2C schéma de principe



I2C principe d'un échange

Le maître :

- émet une condition de démarrage
- envoie une adresse sur 7 bits
- envoie la commande r/w
- lit l'accués et stoppe si NACK
- pour une écriture, il boucle sur
 - envoie les 8 bits de donnée
 - lit l'accusé et stoppe si NACK
- pour une lecture, il boucle sur
 - lit les 8 bits de donnée
 - émet ACK, ou NACK pour stopper
- émet une condition de stop

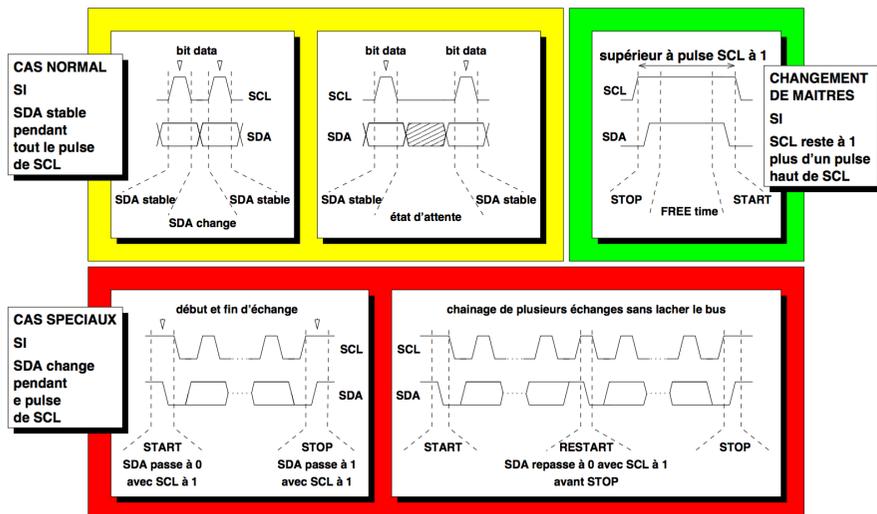
L'esclave :

- attend une condition de démarrage
- lit l'adresse sur 7 bits
- lit la commande r/w
- émet ACK si concerné
- pour une écriture, il boucle sur
 - lit les 8 bits de donnée
 - met ACK ou NACK pour arrêter
- pour une lecture, il boucle sur
 - écrit les 8 bits de donnée
 - lit l'accusé et stoppe si NACK

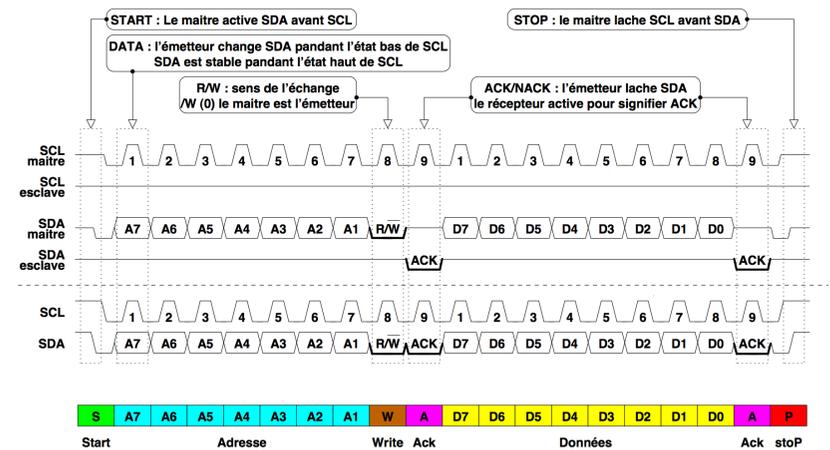
le maître et l'esclave peuvent ralentir l'échange en jouant sur SCL

- C'est le maître qui pilote l'horloge SCL en la mettant à 0 et à 1
- Si l'esclave force l'horloge à 0, le maître ne peut plus la mettre à 1
Le maître comprend qu'il doit attendre que l'esclave "relâche" l'horloge

Etat des lignes SDA et SCL



I2C trame de base : écriture d'un octet

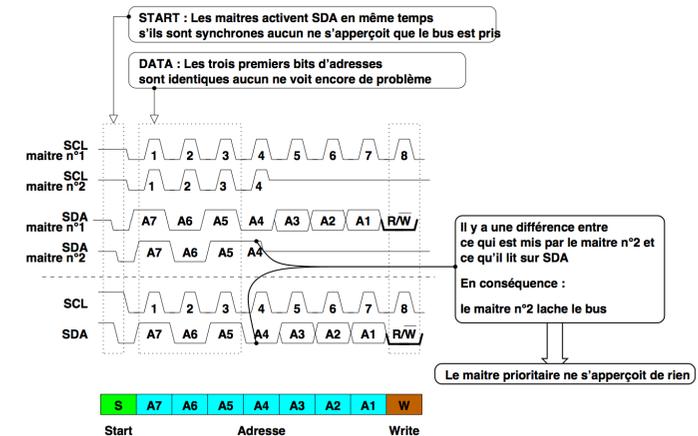


Arbitrage entre maîtres

- Quand un maître adresse un esclave il place l'adresse de celui-ci sur SDA.
 - Pour mettre un 0 sur SDA, donc il active le transistor de pull-down
 - Pour mettre un 1 sur SDA, donc il utilise le pull-up de la ligne.
- Quand un maître écrit sur SDA, il vérifie en relisant SDA
- Si SDA vaut 0 alors que le maître n'a pas activé son pull-down alors c'est qu'un autre maître communique aussi
- Le perdant se retire aussitôt
- Comme les adresses sont données avec les bits de poids fort d'abord, les adresses d'esclaves les plus petites sont prioritaires par rapport aux grandes

Arbitrage entre maîtres

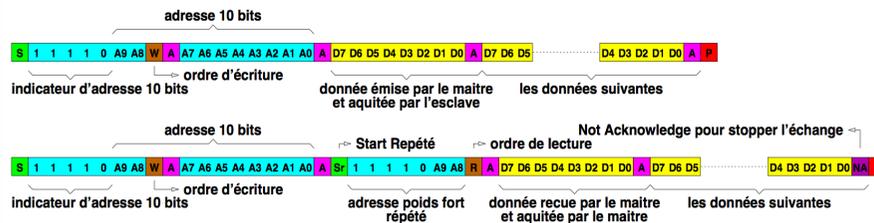
Si deux maîtres tentent de démarrer un échange simultanément : le premier qui dit 1 sur SDA a perdu.



I2C adressage 10 bits

Passage à 10 bits d'adresse, il y a deux cas de figure :

1. Le maître veut faire une écriture : il envoie l'adresse de 10 bits dans les deux premiers octets puis il envoie les données normalement.
2. Le maître veut faire une lecture : il envoie l'adresse de 10 bits en indiquant qu'il s'agit d'une écriture puis il envoie à nouveau le poids fort de l'adresse en indiquant qu'il s'agit d'une lecture, seul l'esclave qui s'était reconnu répond.



I2C Arduino : Wire Library

Pins

- UNO A4 (SDA), A5 (SCL)
- Mega2560 20 (SDA), 21 (SCL)

Le choix est imposé par le microcontrôleur

API

INITIALISATION

- begin()
- begin(adresse)

initialise communication avec Arduino "maître"
initialise communication avec Arduino "esclave"

MODE MAÎTRE

- beginTransmission(adresse)
- endTransmission()
- write(<valeur|chaîne|pointeur,taille>)
- requestFrom(adresse, taille)
- available()
- read()

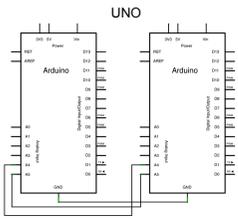
début communication avec un esclave
envoi des données vers esclave, rend 0 si succès
écrit les données à envoyer vers esclave
demande de données à un esclave
rend le nombre d'octets disponibles après un requestFrom
lit l'octet reçu de l'esclave après un requestFrom

MODE ESCLAVE

- onReceive(fonction)
- onRequest(fonction)
- write()
- available()
- onReceive()
- read()

définit la fonction à appeler sur réception de données du maître
définit la fonction à appeler sur requête du maître
envoie les données vers le maître après requête
test si données disponibles en provenance du maître (cf
lit données en provenance maître

Exemple 1 : <http://arduino.cc/en/Tutorial/MasterWriter>



```
#include <Wire.h>

void setup()
{
  Wire.begin(4); // join i2c bus with address #4
  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
  }
}
```

Master Writer Code - Program for Arduino 1

```
#include <Wire.h>

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

byte x = 0;

void loop()
{
  Wire.beginTransmission(4); // transmit to device #4
  Wire.write("x is "); // sends five bytes
  Wire.write(x); // sends one byte
  Wire.endTransmission(); // stop transmitting

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

Slave Receiver Code - Program for Arduino 2

// by Nicholas Zambetti <<http://www.zambetti.com>>

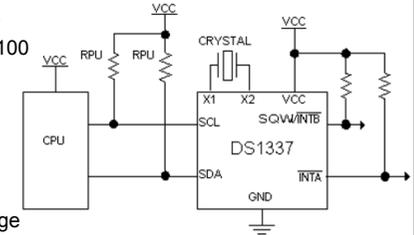
Exemple 2 : Horloge temps réel [ds1337](http://www.dz1337.com)

The DS1337 serial real-time clock is a low-power clock/calendar with two programmable time-of-day alarms and a programmable square-wave output. Address and data are transferred serially through an I²C bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator.

The device is fully accessible through the serial interface while VCC is between 1.8V and 5.5V. Timekeeping operation is maintained with VCC as low as 1.3V.

Key Features

- Real-Time Clock (RTC) Counts Seconds, Minutes, Hours, Day, Date, Month, and Year with Leap-Year Compensation Valid Up to 2100
- I²C Serial Interface
- Two Time-of-Day Alarms
- Oscillator Stop Flag
- Programmable Square-Wave Output Defaults to 32kHz on Power-Up
- Available in 8-Pin DIP, SO, or μSOP
- -40°C to +85°C Operating Temperature Range



Exemple 2 : Horloge temps réel [ds1337](http://www.dz1337.com)

Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00H	0	10 Seconds			Seconds			Seconds	00-59	
01H	0	10 Minutes			Minutes			Minutes	00-59	
02H	0	12/24	AM/PM	10 Hour	Hour			Hours	1-12 +AM/PM 00-23	
03H	0	0	0	0	Day			Day	1-7	
04H	0	0	10 Date			Date			Date	01-31
05H	Century	0	0	10 Month	Month			Month/ Century	01-12 + Century	
06H	10 Year			Year			Year	Year	00-99	
07H	A1M1	10 Seconds			Seconds			Alarm 1 Seconds	00-59	
08H	A1M2	10 Minutes			Minutes			Alarm 1 Minutes	00-59	
09H	A1M3	12/24	AM/PM	10 Hour	Hour			Alarm 1 Hours	1-12 + AM/PM 00-23	
0AH	A1M4	DY/DT	10 Date			Day			Alarm 1 Day	1-7
0BH	A2M2	10 Minutes			Minutes			Alarm 2 Minutes	00-59	
0CH	A2M3	12/24	AM/PM	10 Hour	Hour			Alarm 2 Hours	1-12 + AM/PM 00-23	
0DH	A2M4	DY/DT	10 Date			Day			Alarm 2 Day	1-7
0EH	ESOSC	0	0	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0FH	OSF	0	0	0	0	0	A2F	A1F	Status	—

Note: Unless otherwise specified, the state of the registers is not defined when power is first applied or V_{CC} falls below the V_{OS0}.

Exemple 2 : Horloge temps réel [ds1337](http://www.dz1337.com)

Figure 3. Data Write—Slave Receiver Mode

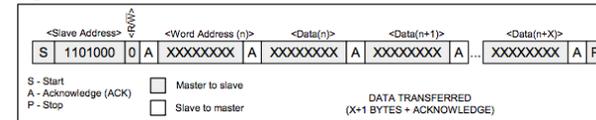


Figure 4. Data Read (from Current Pointer Location)—Slave Transmitter Mode

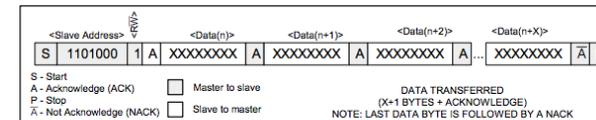


Figure 5. Data Read (Write Pointer, Then Read)—Slave Receive and Transmit

