LYCÉE de l'ÉLORN
le choix de votre réussite

Le bus de liaison synchrone I²C

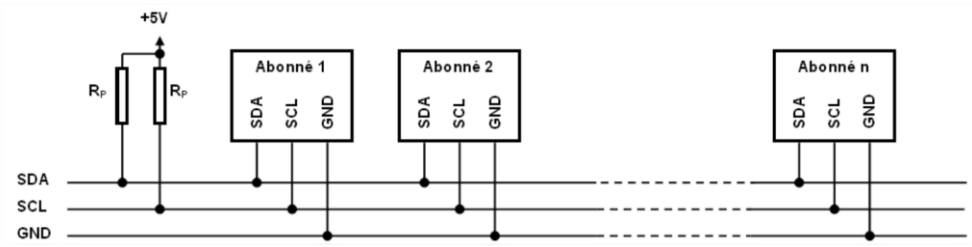
1. Historique

Le bus I2C (Inter Integrated Circuit Bus) est une liaison série synchrone développée par Philips pour les applications de domotique et d'électronique domestique. Il est présent sur les systèmes embarqués comme le Raspberry ou l'Arduino.

2. Caractéristiques

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement trois fils :

- un signal de donnée (SDA),
- un signal d'horloge (SCL) destiné à valider la présence des bits de données,
- un signal de référence électrique (Masse).

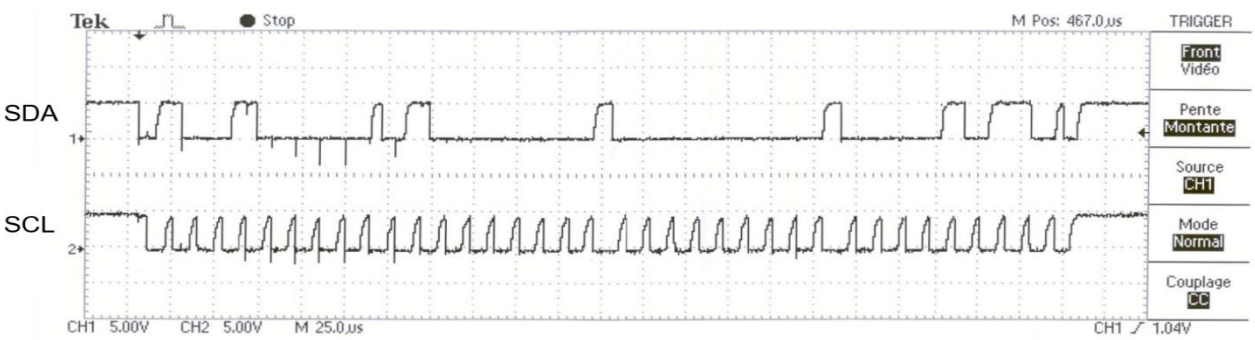


Les données sont transmises en série à une vitesse de 100Kbits/s en mode standard et jusqu'à 400Kbits/s en mode rapide.

3. Principe

Tous les abonnées peuvent définir un niveau électrique sur le bus sans qu'il y ait un risque de destruction de composant. Dès que le bus est libre, le premier abonné qui prend la parole devient le « maître » d'un échange de données.

Exemple de chronogramme :

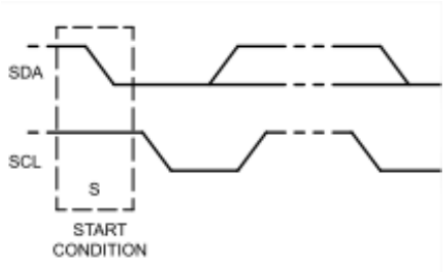


4. Le protocole de communication I²C

La communication sur le bus I²C ne peut se faire qu'entre 2 abonnés. Lorsqu'un abonné prend le contrôle du bus, il devient le maître de la communication. Il génère le signal d'horloge SCL et communique avec un esclave. Selon le sens de la communication, il sera l'émetteur ou le récepteur.

a) La condition de départ

Un abonné prend le contrôle du bus I²C en émettant une condition de départ : Niveau haut sur SCL et front descendant sur SDA.
Cet abonné devient le maître.

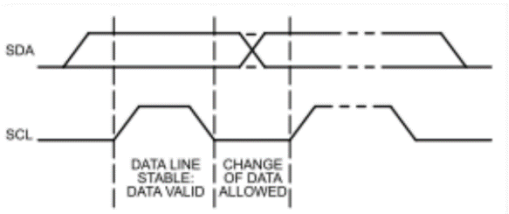


b) Transmission de l'adresse

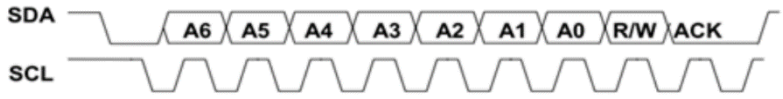
Après avoir pris le contrôle, le maître transmet un octet contenant l'adresse de l'esclave (sur 7 bits) ainsi que l'opération effectuée (écriture ou lecture). « 1 » pour lecture, « 0 » pour écriture.

La validation d'un bit se fait lorsque le signal SCL est au niveau haut.

Philips Semiconductors

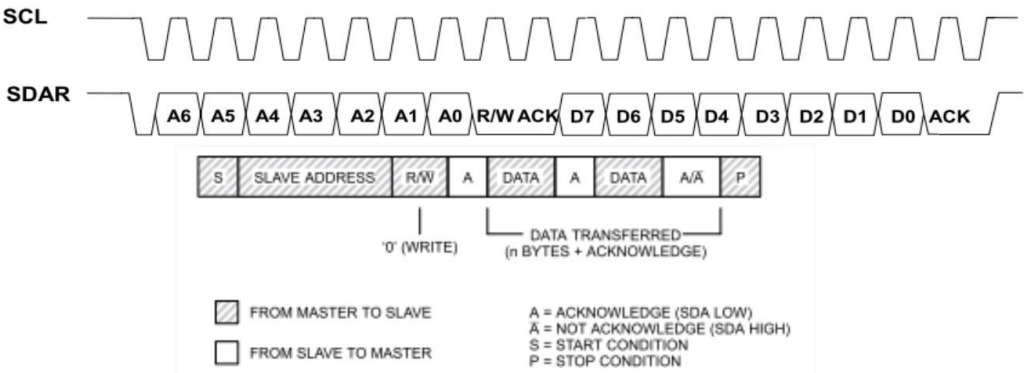


Lorsque l'esclave a détecté son adresse, il émet un bit d'acquittement (ACK) au niveau logique bas.

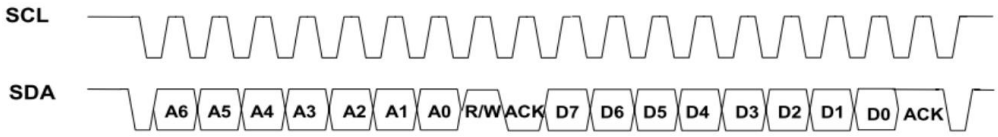


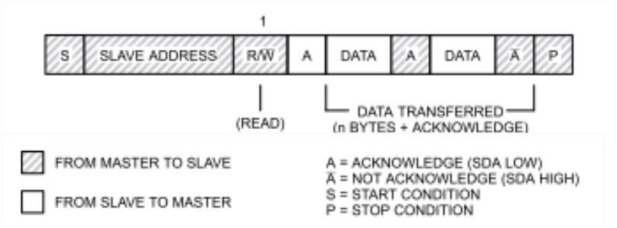
c) Transmission des données

Deux situation sont envisageables :
1er cas : Le maître envoie des données à l'esclave. A la fin de la transmission de chaque octet, l'esclave émet un acquittement.



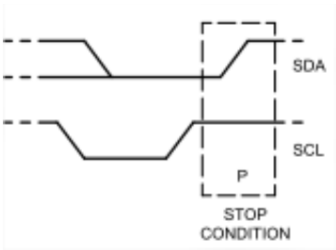
2nd cas : L'esclave envoie des données au maître. A la fin de la transmission d'un octet, le maître émet un acquittement (niveau « 0 ») s'il veut recevoir encore un octet ou bien un non acquittement (niveau « 1 ») s'il a terminé de recevoir.





d) Fin de la communication

Pour terminer la communication, le maître émet une condition d'arrêt : Un niveau haut sur SCL et un front montant sur SDA

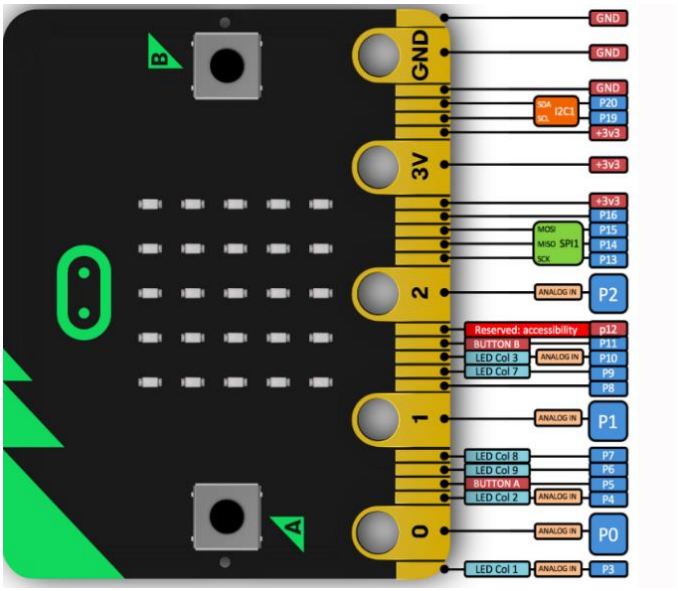


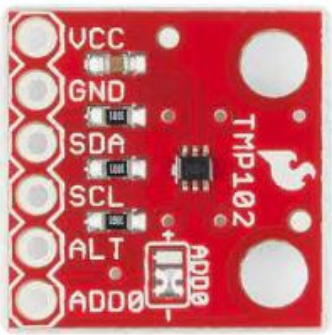
5. Application : Câblage d'un capteur de température sur une carte Micro:Bit

Caractéristiques du capteur ***TMP102***

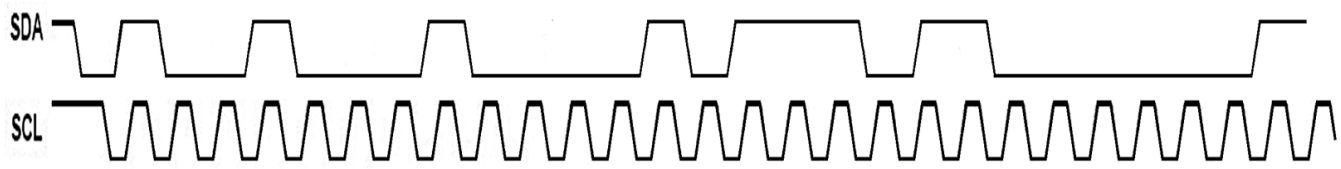
- Conversion sur 12 bits, résolution de 0,0625 °C,
- Plage de mesure nominale : -55°C à 128°C,
- Tension d'alimentation de 1,4 à 3,6 V,
- Le capteur envoi 2 octets. Le 1er octet contient les bits D11 à D4 de la donnée, Les 4 autres bits sont envoyés sur le second octet dont les 4 bits de poids faible sont toujours à 0 (voir doc technique).
- Adresse I2C par défaut : 0x48

a) Réaliser le câblage du composant sur la carte Micro:bit
La broche ADD0 sert à changer l'adresse I2C, la broche ALT à définir une alerte. Leur usage est facultatif.





On relève la trame I^2C suivante :



b) Décodage de la trame :

1. Entourez sur la trame le bit de START.
2. Relevez l'adresse du capteur. La mettre en hexadécimal.
3. Entourez sur la trame le bit de R/W. Quel est son état logique et que cela signifie-t-il ?
4. Entourez sur la trame les bits d'acquittement (ACK).
5. Entourez sur la trame les bits de données transmis par le capteur.
6. Entourez sur la trame le bit de non-acquittement (NACK)

c) Analyse des données :

1. Donnez la valeur des 12 bits de mesure que le capteur a envoyé (lue sur la trame).
2. En déduire la température mesurée par le capteur.

Code Python possible sur une carte Micro:bit

```
from microbit import *
import time

while True:
    lecture = i2c.read(0x48, 2) # lecture des 2 octets à l'adresse 0x48
    MSB = lecture[0]
    LSB = lecture[1]
    data = ((MSB << 8) + LSB)>> 4
    temperature = data * 0.0625
    print(temperature)
    time.sleep(1)
    print("------------------")
```

TEMPERATURE REGISTER

The Temperature Register of the TMP102 is configured as a 12-bit, read-only register (Configuration Register EM bit = '0', see the *Extended Mode* section), or as a 13-bit, read-only register (Configuration Register EM bit = '1') that stores the output of the most recent conversion. Two bytes must be read to obtain data, and are described in Table 3 and Table 4. Note that byte 1 is the most significant byte, followed by byte 2, the least significant byte. The first 12 bits (13 bits in Extended mode) are used to indicate temperature. The least significant byte does not have to be read if that information is not needed. The data format for temperature is summarized in Table 5 and Table 6. One LSB equals 0.0625°C. Negative numbers are represented in binary twos complement format. Following power-up or reset, the Temperature Register will read 0°C until the first conversion is complete. Bit D0 of byte 2 indicates Normal mode (EM bit = '0') or Extended mode (EM bit = '1') and can be used to distinguish between the two temperature register data formats. The unused bits in the Temperature Register always read '0'.

Table 3. Byte 1 of Temperature Register(1)

D7	D6	D5	D4	D3	D2	D1	D0
T11 (T12)	T10 (T11)	T9 (T10)	T8 (T9)	T7 (T8)	T6 (T7)	T5 (T6)	T4 (T5)

(1) Extended mode 13-bit configuration shown in parenthesis.

Table 4. Byte 2 of Temperature Register(1)

D7	D6	D5	D4	D3	D2	D1	D0
T3 (T4)	T2 (T3)	T1 (T2)	T0 (T1)	0 (T0)	0 (0)	0 (0)	0 (1)

(1) Extended mode 13-bit configuration shown in parenthesis.

Table 5. 12-Bit Temperature Data Format(1)

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	HEX
128	0111 1111 1111	7FF
127.9375	0111 1111 1111	7FF
100	0110 0100 0000	640
80	0101 0000 0000	500
75	0100 1011 0000	4B0
50	0011 0010 0000	320
25	0001 1001 0000	190
0.25	0000 0000 0100	004
0	0000 0000 0000	000
–0.25	1111 1111 1100	FFC
–25	1110 0111 0000	E70
–55	1100 1001 0000	C90

(1) The resolution for the Temp ADC in Internal Temperature mode is 0.0625°C/count.

For positive temperatures (for example, +50°C):

Twos complement is not performed on positive numbers. Therefore, simply convert the number to binary code with the 12-bit, left-justified format, and MSB = 0 to denote a positive sign.

Example: (+50°C)/(0.0625°C/count) = 800 = 320h = 0011 0010 0000

For negative temperatures (for example, –25°C):

Generate the twos complement of a negative number by complementing the absolute value binary number and adding 1. Denote a negative number with MSB = 1.

Example: (|–25°C|)/(0.0625°C/count) = 400 = 190h = 0001 1001 0000

Twos complement format: 1110 0110 1111 + 1 = 1110 0111 0000