Thème n°4: Interfaces de communication (2. Bus TWI (I2C, IIC))

CELECT7 : Applications des microcontrôleurs

M. Zwingelstein

rev. 2020

Organisation de la séance

— Cours 1 pour comprendre le bus TWI + exercice d'analyse de signaux TWI
— Lab1 de mise en oeuvre avec la couche d'abstraction arduino (bibliothèque *Wire*)
— Cours 2 pour connaître les registres de l'ATmega328p associés au module hardware TWI
— Lab2 pour mettre en oeuvre une liaision TWI en agissant directement sur les registres, sans utiliser la couche d'abstraction arduino

---

Travail préparatoire

Lectures de la partie péparatoire

1. Page d'accueil du site https://www.i2c-bus.org/
2. Paragraphes 1.3, 1.4, 2.1, 2.2, 2.4, 4.10, 7.2 de la datasheet du capteur MPU9250 : [lien] (cliquer sur le rectangle bleu « DOCUMENTATION »)
3. Paragraphes 21.1 (Features), 21.5 (Fig. 21-9 : Overview of the TWI module), 21.5.2 (Bit-rate generator unit), 21.9.3 (registre TWSR) de la datasheet de l'ATmega 328p : [lien]

Questionnaire de la partie préparatoire

Questions sur le capteur MPU9250

1. Quelle est l'adresse esclave du capteur MPU9250, en notation hexadécimale, lorsque :
   — la broche AD0 est reliée à la masse ?
   — la broche AD0 est reliée à VDD ?
2. Quelle fréquence d'horloge maximale peut-on utiliser pour communiquer avec le capteur MPU9250 via l'interface TWI/I2C ?

Questions sur l'interface TWI du microcontrôleur ATmega 328p

3. Quelle valeur faut-il mettre dans le registre **TWBR** pour avoir une fréquence horloge de l'interface TWI/I2C de 100 kHz lorsque :

   - la fréquence d'horloge de l'ATmega328p est de 16 MHz
   - la *prescaler value* est fixée par les bits TWPS1 et TWPS0 à zéro

4. Il est écrit dans la datasheet de l'ATmega328p (paragraphe 21.9.3) : « *The application designer should mask the prescaler bits to zero when checking the Status bits* ».
   - expliquer le fonctionnement de l'opération de masquage dont il est fait question
   - compléter la ligne de code permettant de récupérer le statut de l'interface TWI/I2C (bits 7:3 du registre TWSR) par cette opération de masquage : `statut = TWSR & ...`

Cours 1 : Le bus TWI (I2C, IIC)

Généralités

- TWI : **T**wo **W**ires **I**nterface
- I2C (IIC) :
  - **I**nter **I**ntegrated **C**ircuit
  - Marque déposée par par Philips Semiconductor en 1982 (maintenant NXP)
  - Spécification complète disponible [ici]
- Transfert série, octet par octet
- Liaison bidirectionnelle et half-duplex
- Bus synchrone (une pulsation d'horloge est générée pour chaque bit transmis)
- Courte portée (quelques mètres)
- Débits : max 100 kbits/s en mode standard, max 400 kbits/s en mode fast, max 5 Mbits/s en mode ultra fast

Interface électrique

3 lignes en plus de la masse :

- ligne **SDA** (**S**erial **DA**ta)
- ligne **SCL** (**S**erial **CL**ock)

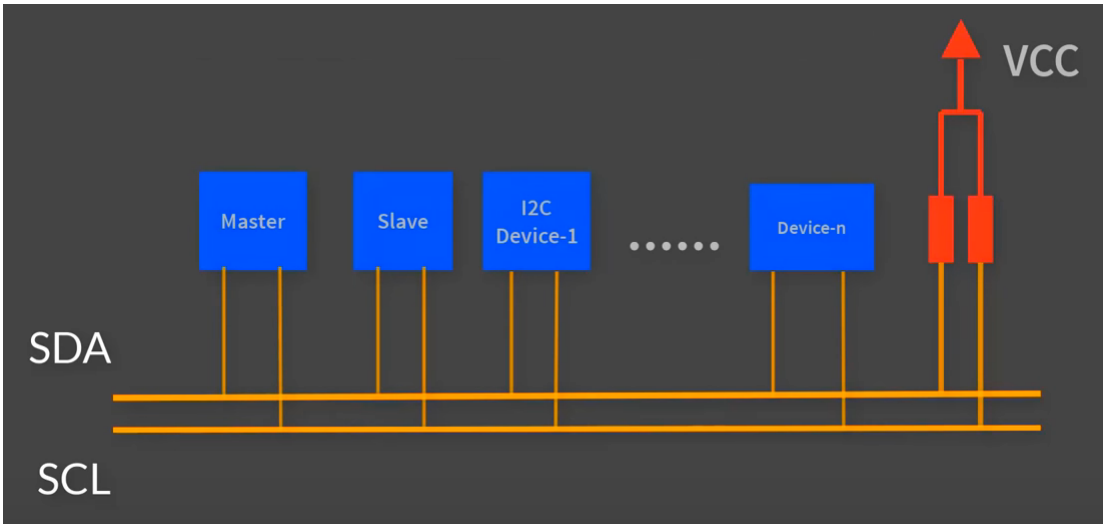


FIGURE 1 – Principe de câblage d'un bus TWI.

Lignes en mode *drain ouvert* (*open drain*)

Le bus TWI utilise les lignes SDA et SCL en mode *drain ouvert* en association avec des résitances de tirage vers le haut (pull-up resistors, R_p)

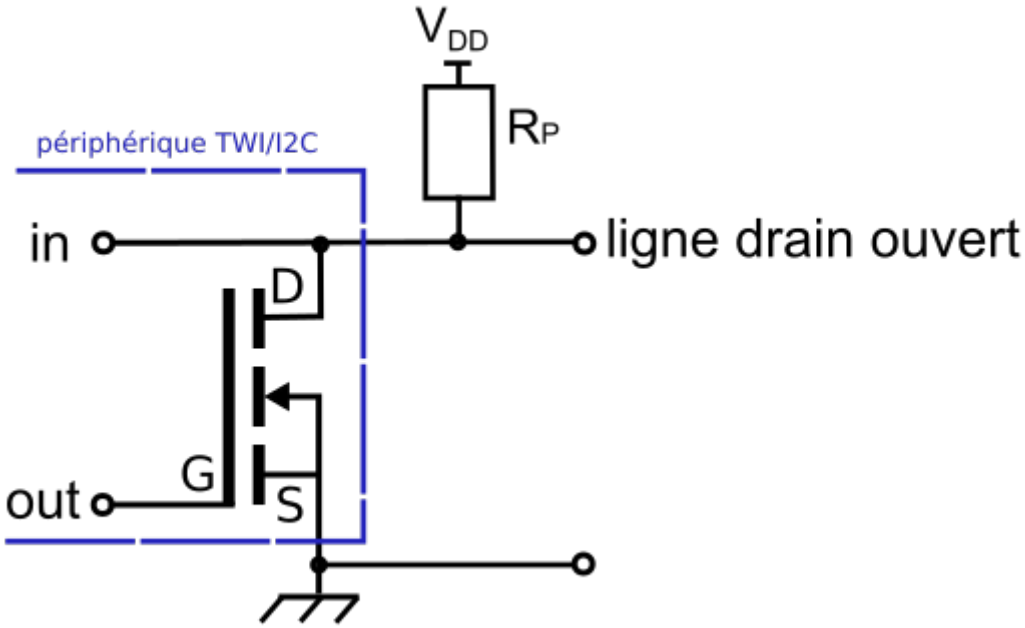


FIGURE 2 – Principe d'une ligne en mode *drain ouvert*.

Propriétés :

- N'importe quel périphérique du bus est capable de forcer la ligne à l'état bas (en imposant un état bas sur sa sortie),
- Aucun périphérique ne peut forcer la ligne à l'état haut
- Grâce à la résistance de tirage, la ligne est à l'état haut par défaut, si aucun périphérique ne l'a forcée à l'état bas

$\Rightarrow$ « *Wired AND function* », dont tire parti le protocole TWI

- Lorsque le bus n'est pas utilisé (état *idle*): SCL = SDA = 1

Remarques :

- *Open-drain* : par opposition à *push-pull*, où chaque périphérique est capable de forcer la ligne à l'état haut ou à l'état bas (cas du bus de communication SPI)
- Si des transistors bipolaires sont utilisés à la place des transistor MOSFET, on parle de ligne en mode *collecteur ouvert* (*open collector*)

- Les valeurs des résistances de tirage doivent être choisies avec attention :
  - $R_{p,max}$ est en lien avec la contrainte du temps de montée du signal (circuit RC passe-bas (R_p, C_b))
    Ex : Temps de montée < 300 ns si f_{SCL} = 100 kHz
  - $R_{p,min}$ est en lien avec la tension VDD
    Ex : $R_{p,min}$ = 1,7 kOhms pour VDD = 5V

Fonctionnement en mode maître-esclave :

Un périphérique du bus aura soit la fonction de maître, soit la fonction d'esclave. Chaque bus contient au minimum un maître.

- un microcontrôleur peut fonctionner en maître ou en esclave
- un capteur, une mémoire… fonctionnera en esclave
- l'horloge est générée par le maître
- plusieurs maîtres sont possibles sur un même bus TWI. Un mécanisme d'arbitrage est prévu dans le protocole pour ce cas.

Term	Description
Transmitter	the device which sends data to the bus
Receiver	the device which receives data from the bus
Master	the device which initiates a transfer, generates clock signals and terminates a transfer
Slave	the device addressed by a master

FIGURE 3 – Terminologie.

Modes de transmission

L'interface TWI peut se trouver dans l'un des quatre états suivants :

- maître qui émet, **MT**
- maître qui reçoit, **MR**
- esclave qui émet, **ST**
- esclave qui reçoit, **SR**

On distingue ainsi trois modes de communication possibles :

- mode 1 : **MT-SR**
- mode 2 : **MR-ST**

— mode 3 : mélange des modes 1 et 2 au sein d'une même trame TWI (ex : mode 1 (**MT-SR**) suivi de mode 2 (**MR-ST**))

Format de trame

Généralités

— Une trame commence par une *condition de start* (**S**) et finit par une *condition de stop* (**P**)
  — **S** : SCL = 1, SDA 1 $\rightarrow$ 0
  — **P** : SCL = 1, SDA 0 $\rightarrow$ 1
— En dehors des conditions de *start* et de *stop*, la ligne SDA ne peut **jamais** changer d'état lorsque SCL est au niveau haut
— L'émetteur envoie les données octet par octet et le recepteur acquite chaque octet reçu par un bit d'acquittement (0 : ack, 1 : nack)
— Les types possibles d'octets envoyés sont :
  — 1er octet de la trame :
    — **SLA+W** : **SL**ave **A**dress (7 bits) + **W**rite (1 bit=0)
    — **SLA+R** : **SL**ave **A**dress (7 bits) + **R**ead (1 bit=1)
  — $\geq$ 2ème octet de la trame : **DATA** (8 bits) :
    — adresse d'un registre ou d'une mémoire de l'esclave
    — valeur à écrire dans un registre ou une mémoire de l'esclave
    — valeur (contenu d'un registre de l'esclave) à envoyer au maître suite à sa requête
— A chaque début de trame, le bus est en mode 1 (**MT-SR**). Le premier octet d'une trame est donc toujours à l'initiative du maître, quelque soit le mode de transmission (mode 1, mode 2 ou mode 3)

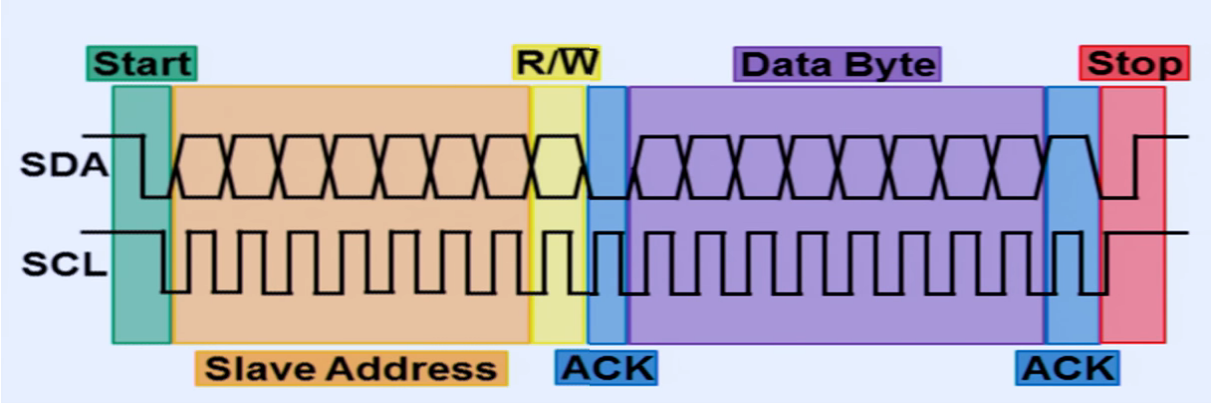


Figure 4 – Illustration d'une trame I2C avec 1 octet de données.

Format de trame en mode 1 (MT-SR)

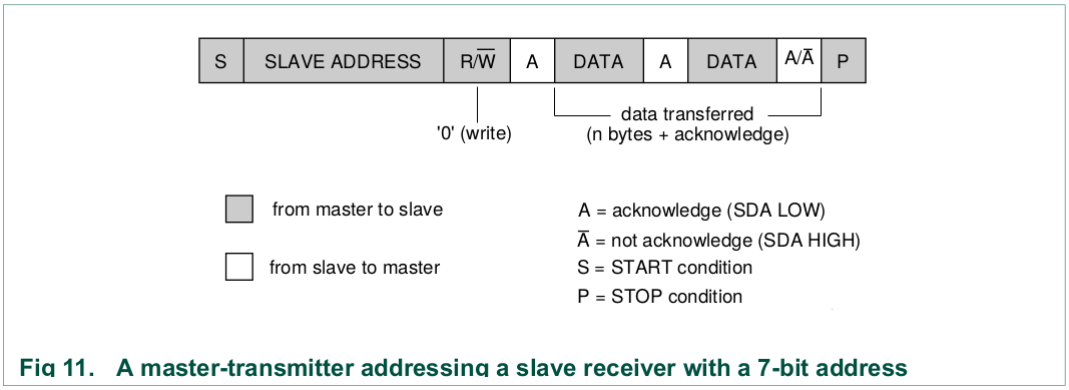


Fig 11. A master-transmitter addressing a slave receiver with a 7-bit address

FIGURE 5 – Mode 1 : MT-SR.

- Premier octet : octet **SLA+W** (bit $R/\bar{W} = 0$)
- L'esclave acquite chaque octet reçu du maître
- Le nombre d'octets transmis entre *start* (**S**) et *stop* (**P**) est illimité et fixé par le maître

Format de trame en mode 2 (MR-ST)

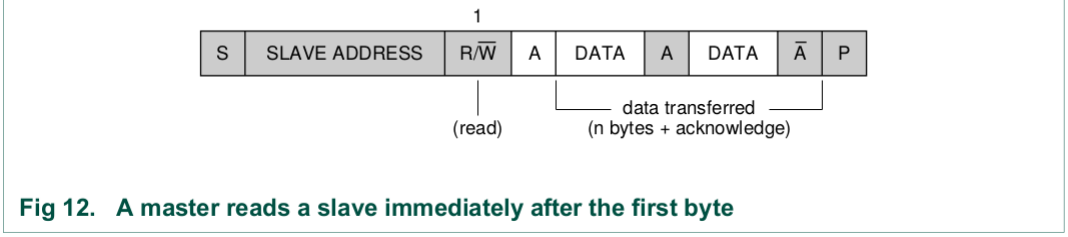


Fig 12. A master reads a slave immediately after the first byte

FIGURE 6 – Mode 2 : MR-ST.

- Premier octet : octet **SLA+R** (bit $R/\bar{W} = 1$) acquité par le maître
- Le maître acquite chaque octet **DATA** reçu de l'esclave
  - cas particulier du dernier octet reçu : **n**ack pour indiquer à l'esclave qu'il doit cesser d'émettre
- Le nombre d'octets transmis entre *start* (**S**) et *stop* (**P**) est illimité et fixé par le maître

Format de trame en mode 3 (mixte)

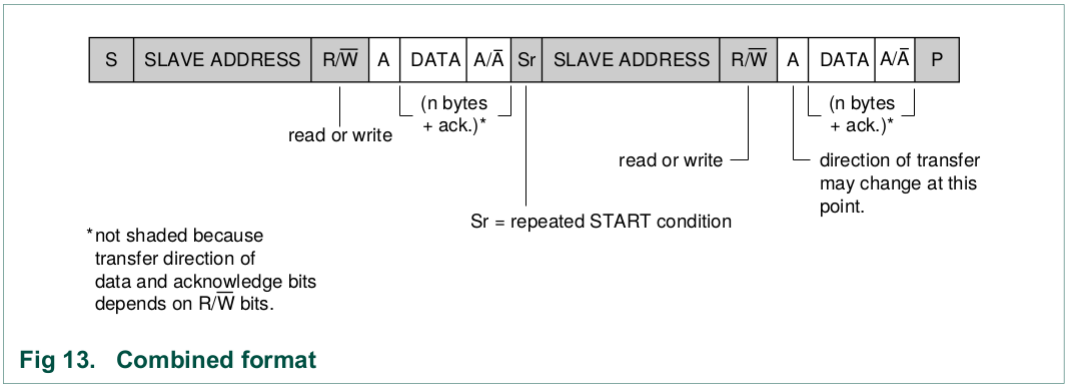


FIGURE 7 – Mode 3 : sens modifié au cours du transfert.

— « *Repeated start* » (**Sr**) au moment du changement de sens,
  — le **Sr** est identique à un *start* (**S**), mais a lieu alors que la ligne n'est pas dans l'état IDLE. Ceci permet de changer le sens de la communication sans libérer la ligne,
  — un octet d'adressage (**SLA+R** ou **SLA+W**) doit suivre le **Sr**.

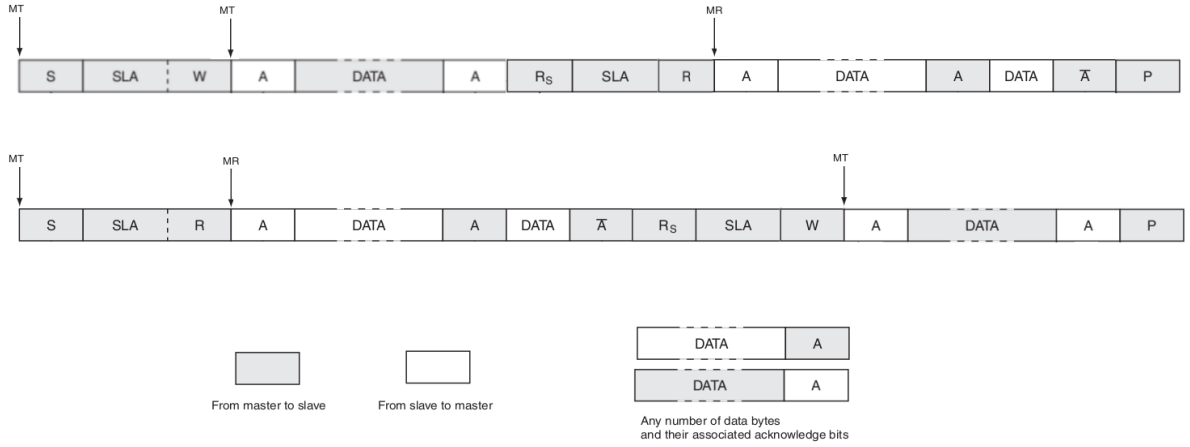


FIGURE 8 – Exemples de trames en mode 3.

— le dernier octet reçu par un MR (avant un P ou un Sr) est suivi d'un **n**ack (et non d'un ack)

Périphériques ayant des tensions d'alimentation différentes

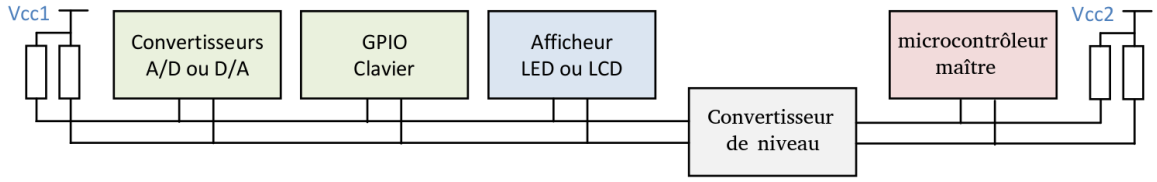


FIGURE 9 – Exemple d'application TWI avec plusieurs VDD différentes.

Il est généralement nécessaire d'utiliser un circuit de conversion de niveau bi-directionnel.

— voir à ce sujet la note d'application NXP AN10441: [lien]

Circuit de conversion de niveau bi-directionnel (convertisseur de niveau)

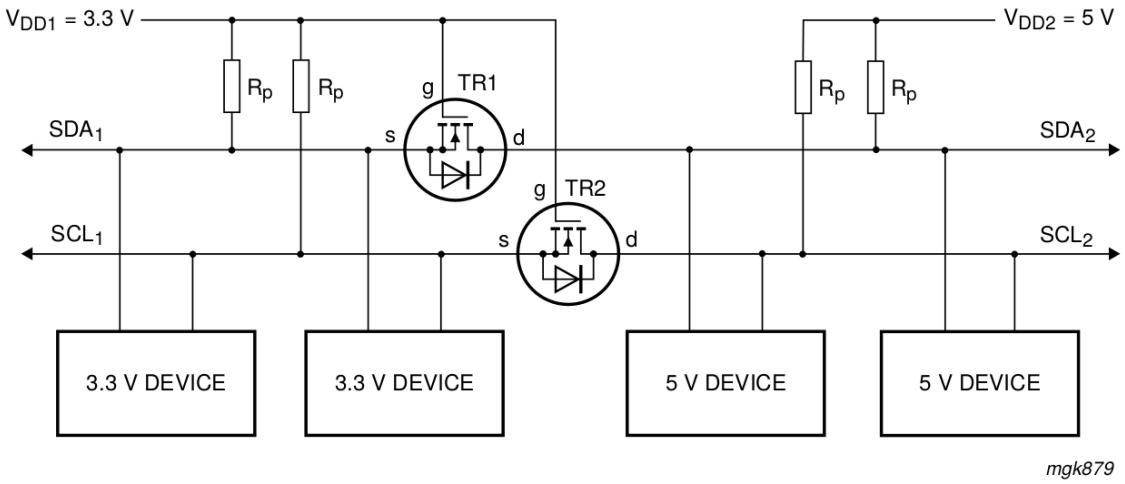


Fig 1. Bidirectional level shifter circuit connecting two different voltage sections in an I²C-bus system

FIGURE 10 – Circuit de conversion de niveau bi-directionnel.

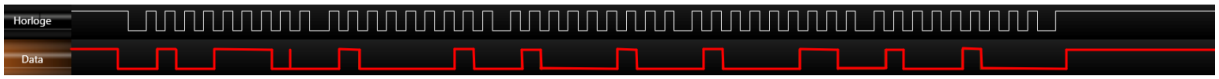
Fonctions réalisées :

— conversion de niveau lorsqu'aucun côté ne force l'état bas
— maintient de la fonction « *wire AND* » : si un côté force l'état bas, l'autre côté se trouve également à l'état bas

Exercice : Analyse de signaux

relevé n°1

relevé n°2



relevé n°3

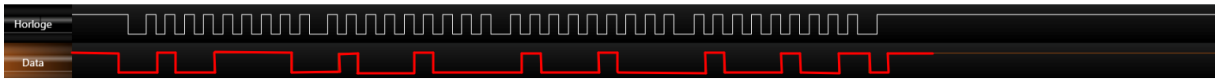


FIGURE 11 – Relevés de signaux TWI.

Retrouver tous les éléments des trames correpondant à ces trois signaux :

- Modes de transmission (qui émet, qui reçoit ?),
- Champs de la trame (**S**, **SLA+R**, **SLA+W**, **DATA**, **Sr**, **P**),
- Valeurs numériques des champs de la trame (en hexadécimal pour les octets),

Lab 1 - Mise en oeuvre avec la bibliothèque *Wire* d'arduino

Présentation

Dans ce lab, vous allez apprendre à utiliser la bibliothèque `Wire` d'arduino pour mettre en place une communication TWI entre :

- le uC ATmega328p de la carte UNO
- un capteur inertiel Invensense MPU9250 à 9 degrés de libeeté (DoF)

Matériel

- Carte arduino UNO
- Platine capteur MPU9250 (comprend les résitances de pull-up)
- Adaptateur de niveau (le MPU9250 est alimenté en 3,3V et l'ATmega328p en 5V)
- Fils de câblage
- Oscilloscope

Documentation

- Bibliothèque `Wire` d'arduino :
  - officielle : [ici]
  - détails sur le playground de arduino.cc [ici]
- Datsheet du microcontrôleur ATmega328p : [ici]
- Datasheet et register maps du capteur MPU9250 : [ici]
- Adaptateur de niveau : [ici] et [là]

Travail à réaliser

Câblage :

1. Câbler la liaison TWI entre le capteur MPU9250 et le uC ATmega328p via l'adaptateur de niveau.
   Indications :
   - les broches SDA et SCL sont respectivement les broches A4 et A5 de la carte UNO,
   - la broche AD0 du capteur sera reliée à la masse,
   - les résistances de pull-up du bus TWI sont déjà présentes sur la platine MPU9250,

Programme préliminaire :

2. Ouvrir dans l'IDE arduino le programme `seance4_I2C_1.ino` fourni.

Analyser le code en vous référant à la documentation de la bibliothèque `Wire` d'arduino, et répondre aux questions suivantes :

- à quoi voit-on que le uC est configuré en *maître* et pas en *esclave* ?
- le mode de transmission est-il **MT-SR** ou **MR-ST** ? Pourquoi ?
- à quoi correspondent les valeurs (à chercher dans la datasheet du MPU9250) :
  - 0x68
  - 0x3B

3. Compiler et téléverser le programme puis observer à l'oscilloscope les signaux SCL et SDA
   - prendre en photo (ou enregister sur clé usb depuis l'oscilloscope) les signaux visualisés (à mettre dans le compte-rendu)
   - représenter les séquences de bits véhiculés sur la liaison
   - interpréter ces séquences en lien avec le mode de transmission utilisé et les format de trames TWI décrits en cours :
     - **S**
     - **P**
     - **SLA+W**
     - **SLA+R**
     - **Rs**
     - **DATA** (indiquer la valeur en hexadécimal)

Second programme :

4. Ouvrir dans l'IDE arduino le programme `seance4_I2C_2.ino` fourni

   4.1 Analyse de code :

   - Analyser le code en vous référant à la documentation de la bibliothèque `Wire` d'arduino
     - rechercher dans le document *Register Maps* quel registre du capteur MPU9250 possède l'adresse 0x75. Quelle valeur par défaut contient ce registre ?
     - repérer dans le code à quels endroits les modes de transmission *master transmitter* (**MT**) et *master receiver* (**MR**) sont utilisés
     - expliquer en quelques mots ce que fait le programme d'un point de vue applicatif
     - dessiner la structure de la trame TWI qui sera générée par le programme

   4.2 Programmation :

- Compiler et téléverser le programme puis observer à l'oscilloscope les signaux SCL et SDA
  - prendre en photo les signaux visualisés (à mettre dans le compte-rendu)
  - représenter les séquences de bits véhiculés sur la liaison
  - interpréter ces séquences en lien avec la structure de trame que vous avez proposée à la question précédente

Configuration et exploitation du capteur MPU9250 :

En guise de synthèse, l'objectif de cette dernière partie du Lab 1 est de configurer le capteur MPU9250 pour avoir une plage de mesure des accéléromètres de plus ou moins 4g, et de récupérer à intervalles de temps réguliers (100 ms) les données des trois accéléromètres selon les axes x, y et z du capteur.

5. Rechercher dans la datasheet du MPU9250 les éléments suivants :
   - adresse du registre **PWR_MGMT_1** (sert à activer le MPU9250)
   - adresse du registre **ACCEL_CONFIG**
   - configuration des bits ACCEL_FS_SEL[1:0] du registre **ACCEL_CONFIG** pour avoir une plage de plus ou moins 4g (voir register maps p.14, paragraphe 4.7)
   - adresses des registres des trois accéléromètres. Attention, chaque accéleration est stockée sur 16 bits, et donc dans deux registres 8 bits. Par exemple, pour l'accélération selon l'axe x : registres **ACCEL_XOUT_H** (8 bits de poids fort) et **ACCEL_XOUT_L** (8 bits de poids faible)

   Consigner toutes ces informations dans un tableau synthétique pour le compte-rendu.
6. Programmation arduino
   - Ouvrir dans l'IDE arduino le programme `seance4_I2C_3.ino` fourni
   - Compléter le programme afin qu'il affiche sur le terminal série les valeurs des accélérations selon les trois axes x, y et z à une cadence d'environ 100 ms.
     - prendre le temps de bien comprendre les trames qui doivent être échangées avant de commencer à programmer
     - aidez-vous des commentaires du code
     - aidez-vous également des formats de trame des pages 34 et 35 de la datasheet du capteur

Cours 2 : interface TWI du uC ATmega328p

Vue d'ensemble

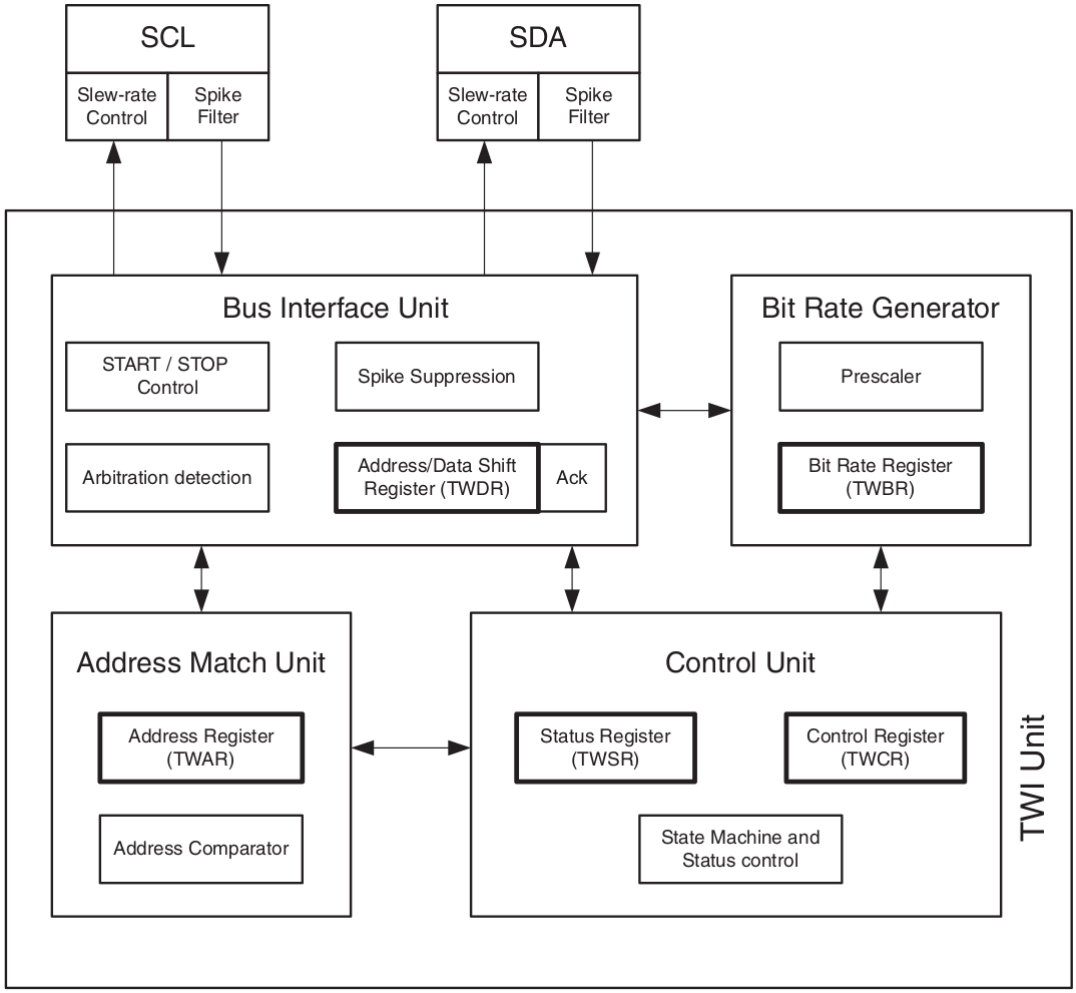


Figure 12 – Schéma bloc du module I2C de l'ATmega328p.

4 unités qui interragissent, et qui incluent 5 registres principaux : **TWAR**, **TWBR**, **TWDR**, **TWSR**, **TWCR**

- unité *Adress Match Unit* (utile si le uC est configuré en esclave)
  - **TWAR** : **T**wo **W**ire **A**dress **R**egister
- unité *Bit Rate Generator*
  - **TWBR** : **T**wo **W**ire **B**it-rate **R**egister $\rightarrow f_{scl} = \frac{F_{CPU}}{(16+2*\textbf{TWBR}*(\text{prescaler value}))}$
- unité *Bus Interface Unit*
  - **TWDR** : **T**wo **W**ire **D**ata **R**egister

- unité *Control Unit*
  - **TWSR** : **T**wo **W**ire **S**tatus **R**egister
  - **TWCR** : **T**wo **W**ire **C**ontrol **Register**

Unité *Bus Interface Unit*

- En mode *émission*, gère l'envoi :
  - du *start* (**S**), du *stop* (**P**),
  - de l'octet présent dans le registre **TWDR**
- En mode *réception*, place le dernier octet reçu dans le registre **TWDR**
- Gère l'arbitrage dans une configuration à plusieurs maîtres
- « Nettoie » le signal (spikes)

Registre **TWDR** (**T**wo **W**ire **D**ata **R**egister) :

Unité *Control Unit*

- Registre de statut : **TWSR** :

  - bits TWSR[7:3] : code du statut de l'interface après chaque opération (ex: TWSR & 0b1111100 = 0x08 après la transmission d'un *Start* - voir table 21-3)
  - bits TWPS[1:0] : *prescaler value* de l'unité *Bit Rate Generator* (voir table 21-8)
- Registre de contrôle : **TWCR** :

  - *bit TWINT : TWI Interrupt Flag (the Flag is cleared when this bit is set to 1 !)*
  - bit TWEA: TWI Enable Acknowledge Bit
  - *bit TWSTA : TWI STArt condition bit*
  - *bit TWSTO : TWI STOp condition bit*
  - bit TWWC: TWI Write Collision Flag
  - *bit TWEN : TWI ENable bit*
  - bit TWIE : TWI Interrupt Enable

Fonctionnement de l'interface TWI

Principe :

Le fonctionnement de l'interface est décrit précisemment aux paragraphes 21.6 *Using the TWI* et 21.7 *Transmission Modes* de la datasheet.

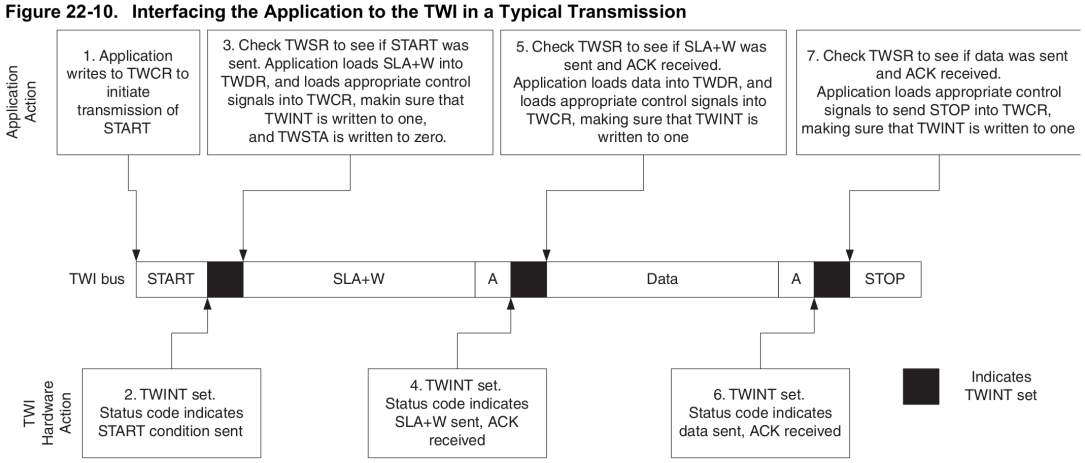


Figure 13 – Principe de dialogue entre l'interface et l'application.

Voir le tableau 21-2 de la datasheet pour le code en langage C de manipulation des registres associé à cet exemple.

Détail du fonctionnement pour l'envoi d'un START :

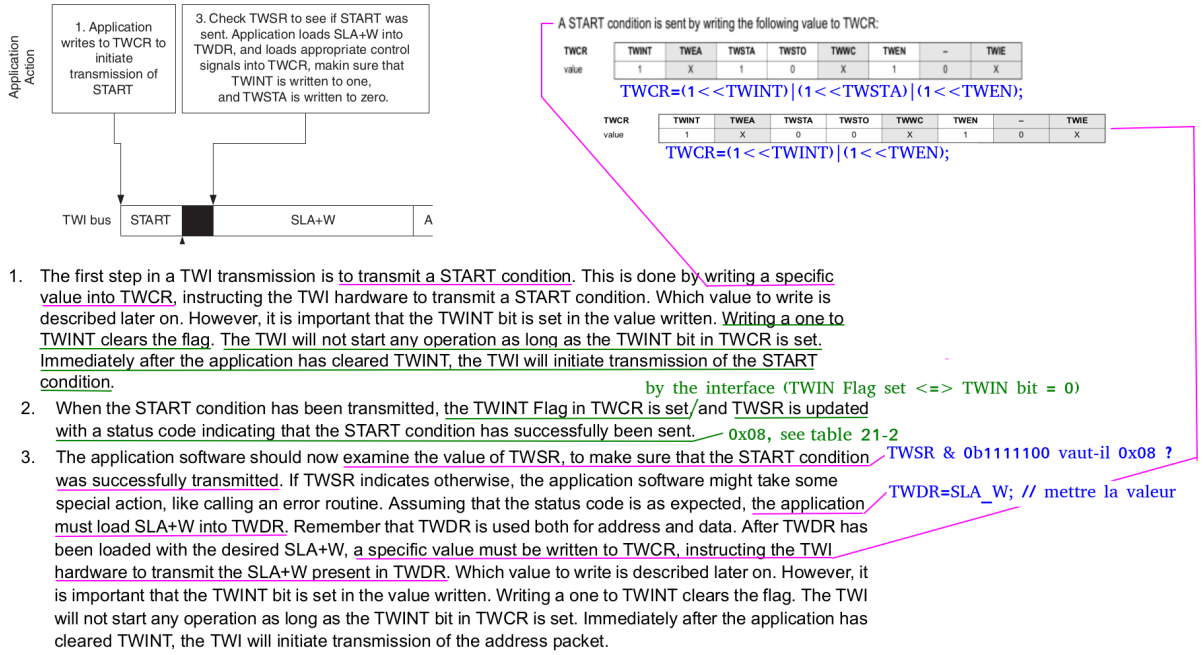


FIGURE 14 – Détail du fonctionnement pour l'envoi d'un START et d'un SLA+W.

En résumé :

- Quand l'interface TWI a réalisé une action et attends quelque chose de l'application, elle met à 0 le bit TWINT du registre **TWSR** (TWINT Flag set)
- Avant de programmer une nouvelle action (ex : **SLA+W**, **DATA**, **P**, ...), l'application doit :
  - attendre que le bit TWINT a bien été mis à zéro par l'interface : `while (TWCR & (1<<TWINT)== 0);`
  - vérifier que le code du statut de l'action réalisée est le bon. Exemple pour une action réalisée *Start*: `while (TWSR & 0B11111000 != 0x08);`
- Le déclenchement d'une nouvelle action par l'interface se fait lors de la mise à 1 du bit TWINT du registre **TWSR** (TWINT Flag cleared)
- Les actions **SLA+R**, **SLA+W** et **DATA** (en émission) nécessitent une écriture dans le registre **TWDR**
- L'action **DATA** (en réception) nécessite une lecture du registre **TWDR**

Lab 2 - Mise en oeuvre sans *Wire* d'arduino

Présentation

Dans ce lab, vous allez étudier puis réaliser la mise en oeuvre le l'interface TWI depuis l'IDE arduino, mais sans utiliser sa couche d'abstraction à travers la bibliothèque `Wire`.

— Deux étapes :

1. analyse de code mettant en oeuvre le mode 1 (**MT-SR**)
2. écriture de code mettant en oeuvre le mode 3, afin de lire la valeur du registre **WHOAMI** du capteur MPU9250

Documentation et matériel

Idem Lab 1

Travail à réaliser

Analyse de code

1. Ouvrir le programme `seance4_I2C_4.ino` fourni dans l'IDE arduino et l'analyser. Dessiner sur papier la trame TWI que génère ce programme.
2. Compiler et téléverser le code dans l'ATmega328p de la carte arduino uno puis observer les signaux SCL et SDA à l'oscilloscope (prendre une photo). Décoder la trame observée (bits et nom/valeur des champs associés).

Ecriture de code

On souhaite écrire un code qui permette de lire via le bus TWI le contenu du registre **WHOAMI** du capteur MPU9250. L'adresse de ce registre ainsi que sa valeur par défaut ont été traités dans la partie préparatoire à cette séance.

3. Dessiner sur papier la trame TWI correspondante
4. Rechercher dans la datasheet (section 22.7) la façon de programmer les champs (notamment le **SLA+R** et le **Sr**).

5. Ouvrir dans l'IDE arduino et compléter le programme `seance4_I2C_5.ino` fourni. Compiler et téléverser le code dans l'ATmega328p de la carte arduino uno. Observer les signaux SCL et SDA à l'oscilloscope (prendre une photo). Décoder la trame observée (bits et nom/valeur des champs associés).