Lab. #3 : Protocole de communication I2C

I. But et matériel requis :

Se familiariser avec le protocole I2C pour la communication locale entre périphériques dans un système embarqué.

Matériel requis:

- Carte d'extension MSP-EXP430FR6989 sans le microcontrôleur **ou** kit de composants équivalent, plaquette de montage, et fils de raccordement.
- Kit de développement CY8CKIT-059 PSoC 5LP

II. Protocole I2C

Le standard Inter Integrated Circuit (IIC ou I2C) offre un moyen simple de communication entre un ou des microcontrôleurs et des périphériques situés à courte distance, typiquement sur la même carte de circuit imprimé (PCB). Il utilise pour cela une couche physique avec seulement deux lignes de communication sérielle, et les périphériques ont chacun une adresse unique sur le bus I2C. Ce mécanisme permet l'ajout de périphérique sans avoir besoin de lignes de sélection des périphériques dédiées comme pour le protocole concurrent SPI.

Bases de l'I2C

a) La couche physique

La figure 1 montre un exemple de réseau I2C avec ses deux lignes de communication physiques, une pour les données échangées (SDA) et l'autre pour l'horloge de synchronisation des transferts (SCL). Tous les périphériques sur le bus doivent se connecter à ces deux lignes, et le seul matériel externe requis consiste en deux résistances de tirage vers VDD (rail haut) connectées à SDA et SCL.

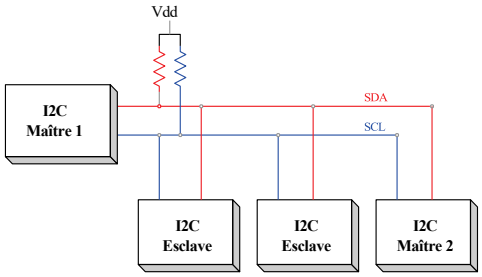


Figure 1. Bus I2C typique

Les composants reliés à un bus I2C ont tous une relation maître-esclave, avec plusieurs maîtres et esclaves pouvant coexister, et la fréquence de la ligne SCL peut être programmée jusqu'à 5 MHz. En pratique, des valeurs plus basses sont utilisées, dont 100 kHz et 400 kHz.

Les lignes SDA et SCL sont bidirectionnelles et permettent d'utiliser la ligne SDA en semi-duplex pour l'envoi et la réception de données. Les deux lignes sont aussi à drain ouvert et actives basses (voir la Figure 2), faisant qu'un périphérique raccordé au bus I2C peut seulement les tirer vers le bas ou être en haute impédance; cela évite de créer un court-circuit alimentation-masse lorsque deux périphériques reliés au bus ont des états logiques opposés.

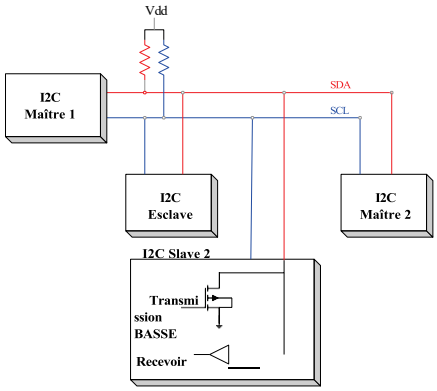


Figure 2. Pilotage de ligne à drain ouvert

b) La couche protocolaire

Seul un maître peut gérer un transfert sur le bus et générer le signal d'horloge. Une transaction comprend quatre phases : départ (start), génération d'adresse et de direction des données, transfert des données, et arrêt (stop).

Démarrage/démarrage répété

Pour démarrer une transaction, le maître envoie d'abord un signal de départ comme défini à la figure 3. Il signale ainsi aux autres composant sur le bus qu'il en prend le contrôle et va envoyer une adresse. Tout autre maître doit attendre alors un signal d'arrêt sur le bus avant de pouvoir le commander à son tour.

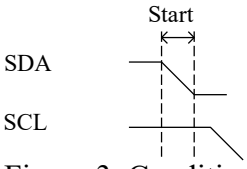


Figure 3. Condition START

Adresse de destination

Cette donnée suit la condition de démarrage, avec la plupart des adresses I2C comprenant sept bits (une option de 10 bits existe aussi), avec un bit de lecture/écriture (R/W) ajouté à la fin pour indiquer la direction du transfert de données à venir (voir la Figure 4). Tous les octets sont transmis en commençant par le bit le plus significatif (MSB).

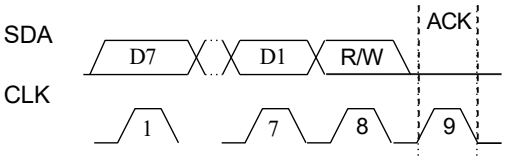


Figure 4. Adresse 7 bits

ACK/NAK

Après chaque octet transmis, le maître attend un accusé de réception de l'esclave sous forme d'un bit sur la ligne SDA (ACK), faisant que chaque transaction comprend neuf bits. L'esclave concerné doit émettre ce bit après avoir reconnu son adresse sur le bus, en tirant la ligne SDA vers le bas. Si aucun esclave ne reconnaît son adresse, la ligne SDA demeure à l'état haut, et le maître interprète cet état comme un acquiescement négatif (NAK).

Données

Une fois l'adresse émise par le maître et acquiescée par l'esclave, les données sont transmises un octet à la fois, avec le bit ACK à chaque fois signalant d'autres octets à venir, et un bit NAK la fin de la transaction. La Figure 5 donne un exemple. Noter que dans le cas d'une opération de lecture, l'esclave n'a pas le moyen d'avertir le maître qu'il n'a plus d'octet à envoyer.

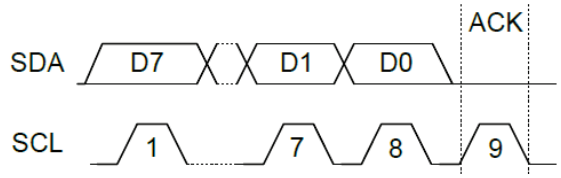


Figure 5. Exemple d'un octet transmis avec un acquiescement ACK retourné

Stop

Une fois tous les octets transmis, le maître émet une condition d'arrêt comme défini à la figure 6, indiquant ainsi la fin de la transaction en cours et la libération du bus I2C.

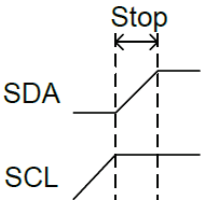


Figure 6. Condition Stop

c) Sommaire des types de transactions

La figure 7 donne l'exemple de deux transactions complètes pour lire ou écrire deux octets par un maître. Elle donne également l'exemple d'une transaction à démarrage répété qui écrit d'abord deux octets, puis lit deux octets.

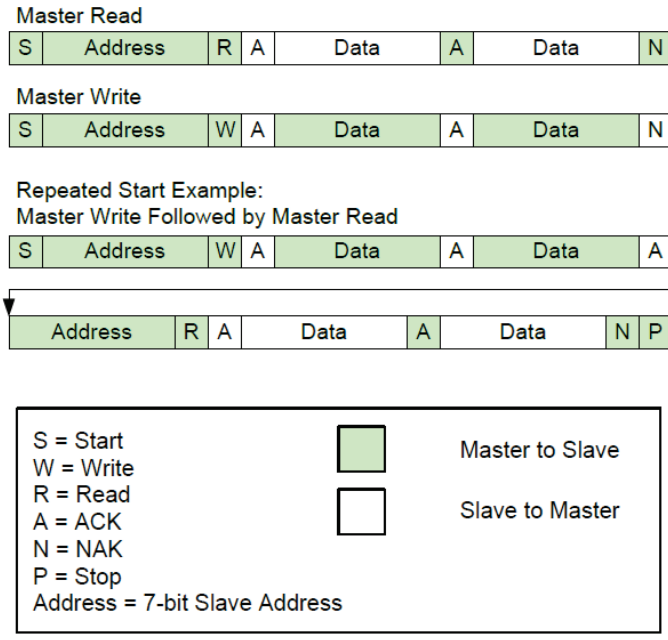
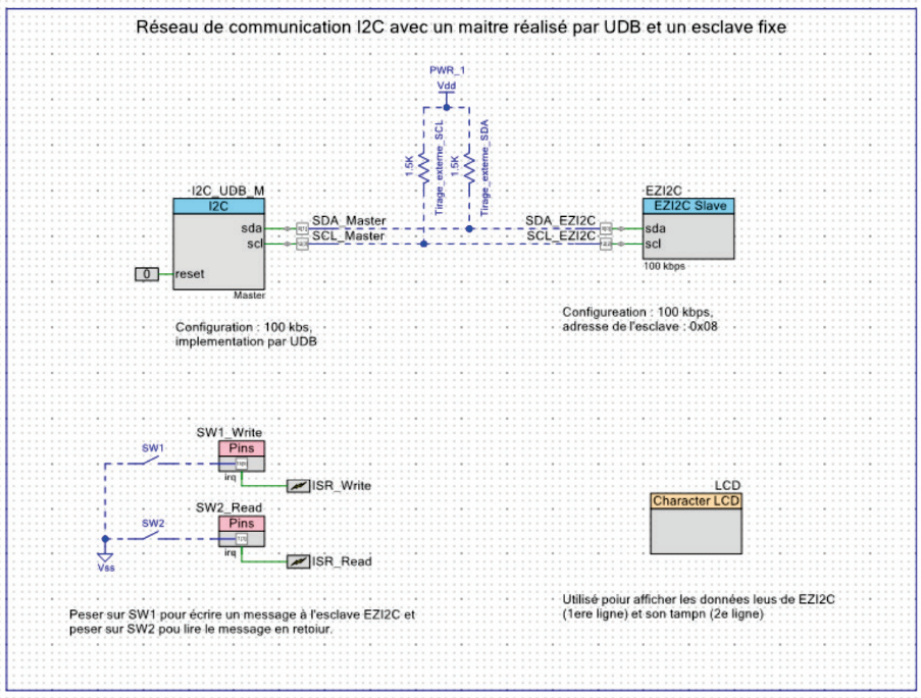


Figure 7. Exemples de transactions I2C

Plus d'information est disponible dans la note d'application AN50987 de Infineon.

III. Procédure à effectuer



Cliquer sur *Find Code Exemple* dans la fenêtre du milieu de la page d'accueil de PSOC Creator et installer le projet `Ezi2cDesign`, après avoir sélectionné `PSoC 5LP` dans *Device family* et `I2C` dans *Filter by*. Noter qu'un projet similaire est décrit dans le document CE95314 de Cypress

Le projet inclut les composants `I2C`, `EZI2C` et `Character LCD`, ce dernier étant utilisé pour l'affichage ds données transmises par `I2C` qui agit comme maître et retournées par `EZI2C` qui agit comme esclave (bus I2C de 100 kbps, adresse d'esclave 0x08 en version 7 bits). Dans l'exemple, le maître est synthétisé par des blocs UDB et l'esclave est déjà présent dans la puce PSoC 5LP.

Lorsque l'interrupteur SW1 est pressé, le maître envoie un message à l'esclave, et lorsque l'interrupteur SW2 est pressé, le maître I2C lit des données de l'esclave. Dans les deux cas, les données transmises sont affichées sur l'écran LCD. Noter que le composant `EZI2C` contient un seul tampon de données, avec seulement la première moitié pouvant être modifiée par le maître, la moitié suivante étant en lecture seulement pour lui.

Le fonctionnement du système dans le fichier *main.c* est comme suit :

1. Initialisation du système avec les interruptions autorisées, suivie de l'affichage d'un message d'accueil sur l'écran LCD.
2. Boucle itérative avec une action spécifiée par une variable *actionSelect* réglée par les commutateurs SW1 et SW2 dans leurs routines d'interruption. Trois cas sont possibles :
    a. SW1 et SW2 inactifs : aucune action;
    b. Cas SW1 pressé : envoi d'un message du maître vers l'esclave avec affichage des données et mise à jour du contenu du tampon;
    c. Cas SW2 pressé : lecture d'un message de l'esclave avec affichage des données.

Pour exécuter le projet sur le kit de développement CY8CKIT-059, il faut d'abord modifier le processeur cible de l'exemple pour `CY5888LTI 5LP` dans *Device selector* sous l'item de menu *Project* de PsoC Creator. Ensuite, la connexion des différentes lignes logique aux broches physiques peut se faire comme dans la figure ci-contre.

Name	Port	Pin	Lock
\LCD:LCDPort[6:0]\	P2[6:0]	1,68,66…62	☑
SCL_EZI2C	P12[2]	46	☑
SCL_Master	P12[3]	47	☑
SDA_EZI2C	P0[0]	48	☑
SDA_Master	P0[1]	49	☑
SW1_Write	P15[5]	61	☑
SW2_Read	P1[7]	19	☑

Expériences :

1. Étudier le code du projet, notamment les fonctions du maître générées par PSoC Creator, et vérifier son bon fonctionnement en le compilant et exécutant.
2. Reproduire l'affichage ACL avec un deuxième afficheur à contrôleur Hitachi piloté par bus I2C, à travers un extenseur de lignes d'e/s de type PCF8574 (noter qu'il existe deux versions de cette puce avec des adresses différentes : 0x20 à 0x27 pour PCF8574 et 0x38 à 0x3f pour PCF8574).

++++++++++++++