Gestion du port I2C (mode maître)

Objectif

Nous avons vu lors d'une précédente séance les échanges rs232. Ce mode de communication est très utile pour l'échange de données avec un terminal (un PC ou un palm). Il est aussi parfois utilisé pour accéder à des périphériques intelligents comme un afficheur LCD ou même une caméra. Mais ce n'est pas idéal car le rs232 est un protocole point-à-point et il faut multiplier les ports et les cables si on veux connecter plusieurs capteurs.

Aujourd'hui nous allons expérimenter l'I2C, un bus d'entrée-sortie qui permet de faire des échanges de données entre plusieurs interlocuteurs (on parle d'abonnés). Ce document contient une présentation succinte du protocole I2C limitée (ou peu s'en faut) aux seules spécifications offertes par le PIC réduites encore à ce dont nous avons besoin pour ce TME. De plus, sur l'I2C on distingue les abonnés maîtres et les abonnés esclaves, on ne programmera le PIC qu'en mode maître. Les étudiants intéressés pourront se reporter à la spécification officielle du protocole I2C présente sur le site du module microcontrôleur.

Au niveau des expériences, nous allons tenter de communiquer avec deux composants I2C : un télémètre à ultra-sons et un convertisseur numérique-analogique à 8 sorties. Nous allons aussi en profiter pour voir une méthode de programmation très utile en assembleur: les automates d'états finis.

Protocole I2C

Cette présentation de l'I2C reprend des dessins et du texte glanés sur le web (en particulier le site ?atmicroprog), et dans le livre de Dominique Paret «Le bus I2C» chez Dunod, merci à eux.

caractéristiques principales du bus I2C

Ce protocole a été défini dans les années 80 par Philips. C'est un protocole simple à mettre en oeuvre, tant d'un point de vue matériel que logiciel. Il est très diffusé et il existe un très grand nombre de composants proposant directement une interface I2C.

I2C ou plutôt IIC signifie Inter-Integrated-Circuit. Comme son nom l'indique, il est destiné à faire communiquer des circuits intégrés sur des distances courtes (50 cm) jusqu'à 3.4 Mbits/s. Il est toutefois possible grâce à des répéteurs, qui regénèrent les signaux et pas mal de précautions, d'atteindre des distances de 100 mètres avec un

débit de 100 kbits/s.

La version de base permet à 128\footnote{Ce nombre de 128 est théorique, car la charge maximale de chaque ligne (SCL ou SDA) ne peut dépasser 400 pF (picoFarad), et les changements d'états des signaux SDA et SCL ne peuvent dépasser 1 µs.} abonnés de communiquer sur 2 fils: un fil pour l'horloge (SCL), un fil pour les données (SDA). En pratique, les cables sont composés de 4 fils: SCL, SDA, GND, VDD. Les deux supplémentaires sont respectivement la masse (GND) et la tension d'alimentation (VDD). Cette dernière n'est pas indispensable mais elle permet généralement d'alimenter le périphérique distant (un peu comme pour l'USB).

Glossaire I2C

abonné
: tout élément connecté sur le bus se nomme un abonné.

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 d'arrêter temporairement l'horloge du maitre.

émetteur
: tout abonné, qu'il soit maître ou esclave, qui envoie des données sur SDA.

récepteur
: tout abonné, qu'il soit maître ou esclave, qui reçoit des données de SDA.

adresse
: numéro attribué à un esclave. Sur le bus tous les esclaves ont une adresse unique. Les adresses petites sont prioritaires vis-à-vis des adresses grandes.

échange
: dialogue entre un maitre et un esclave. Un échange 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'affilée. La fin normale d'un échange est décidée par le maître. L'esclave peut cependant demander l'abandon de l'échange par le maître si l'esclave est récepteur.

arbitrage
: résolution d'un conflit d'accès simultané par 2 maitres.

Principaux avantages de l'I2C

Le protocole du bus I2C est particulièrement rusé, au sens où il offre beaucoup d'avantages tout en restant simple à mettre en oeuvre, c'est dans ses avantages que réside la raison de son succès. Parmi les principaux avantages:

- Un abonné peut se connecter au bus sans perturber le fonctionnement du bus. Il faut quand même respecter un protocole et veiller à ne pas dépasser la charge maximale du bus (400 pF). Ce branchement à chaud permet par exemple de faire de la maintenance sans stopper le système.
- C'est un bus multi-maitres, tout abonné peut devenir maître du bus\footnote{Le module I2C intégré au pic16f877 gère les deux modes mais pas en même temps.}.
- L'arbitrage entre les maîtres est décentralisé. C'est un point important car c'est ce qui permet d'aller loin. Il existe un ordre total des esclaves du point de vue de la priorité. Cette priorité est statique. À un instant donné, la priorité entre deux maîtres dépend des esclaves accédés.

- Le protocole gère les collisions d'accès. Les échanges prioritaires passent sans être perturbés ni même retardés par les échanges non prioritaires. Un échange est priortaire par rapport à un autre s'il est fait avec un esclave plus prioritaire.
- Un maître peut garder la possession du bus entre deux échanges avec un ou plusieurs esclaves.
- Le débit du bus va de 100 kbauds à 3.4 Mbauds dans la version étendue du protocole (HighSpeed?). Le pic16f877 peut aller jusqu'à 400 kbauds. Il offre aussi une version non standard à 1Mbauds.
- Un maître peut adapter son débit en fonction de l'esclave accédé.
- On peut communiquer entre des circuits de différentes technologies (5V et 3.3V) moyennant des adaptateurs très simples.

Cablage I2C et comportement électrique

Le bus I2C est un vrai bus au sens où les fils électriques sont physiquement partagés par tous les abonnés présents. Sur la figure ci-après, on voit que les différents abonnés se branchent directement sur les 3 fils (4 si compte VDD). Il n'y a pas de différences électriques entre un maître et un esclave. Les résistances de rappel (une par ligne) ne sont pas dupliquées. La ligne VDD est en pointillés pour signifier qu'elle est optionnelle. Il existe d'autres connexions plus complexes utilisées s'il y a beaucoup d'abonnés, si on veut aller loin ou si VDD n'est pas 5V. Cette connectique convient sur quelques mètres et pour 10 à 20 abonnés.

Pour permettre le partage des lignes, le bus I2C réalise un ET-cablé entre tous les abonnés. En d'autres termes:

- Les lignes SCL et SDA sont à VDD si personne ne parle grâce aux résistances de rappel à VDD (pullup).
- Pour mettre 1 sur SCL ou SDA, un abonné programme le port en entrée, la résitance 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).

Résumé des caractéristiques électriques standards

- **Tension de fonctionnement** : 5 Volts (fonctionne aussi en 3.3 V).
- **Fréquence de fonctionnement** : jusqu'à 100 kiloHertz en mode standard et 400kHz en mode Fast.
- **Etat logique haut** : de 3 à 5 Volts.
- **Etat logique bas** : de 0 à 1.5 Volts.
- **Capacité maximum du bus** : 400 picoFarad au total. c'est-à-dire 10 pF par abonné auquels s'ajoute la capacité de la ligne elle-même.
- **Temps de montée maximal** : 1 µs à à 100 kHz, 0.3 µs à 400 kHz.
- **Temps de descente maximal** : 0.3 µs à 100 et 400 kHz.
- **Courrant maximal entrant par un abonné** : 3 milliAmpères.
- **Résistance de rappel** : de 1.5 à 3.5 kiloOhms (dépend de la charge du bus).
- **Durée de l'état haut d'un pulse d'horloge** : 4 µs minimum à 100 kHz, de 0.6 µs minimum à 400 kHz.

Le protocole d'échange

Le bus ne dispose que de deux lignes SCL et SDA. Les différents états que vont pouvoir prendre ces lignes vont permettre d'exprimer les étapes d'un échange. On parle de conditions car il s'agit en fait de changements d'état. Pour prendre un exemple, un maître veut faire un échange avec un esclave.

1. Le bus étant libre (**free time**), le maitre prend le bus en imposant une condition de démarrage (**start condition**).

2. Le maitre envoie l'**adresse de l'abonné** esclave visé et le sens du transfert **read/write**. Pour cela, il génère le signal d'horloge SCL périodique et transmet l'adresse bit après bits (**bit transfer**) sur SDA.
3. L'esclave qui se reconnait renvoie un acquitement (**acknowledge** ou **ACK**) sur SDA.
4. Le transfert se poursuit par l'envoi de donnée, nous allons voir comment, et après chaque envoi par l'émetteur, le récepteur envoie un acquitement.
5. L'échange se termine normalement à l'initiative du maître (sauf exception) et s'achève par une condition d'arrêt (**stop condition**).
6. Éventuellement, le maître peut remplacer la condition d'arrêt par une condition de redémarrage (**restart condition**) dans le cas où il veut changer d'abonné ou de sens d'échange.
7. Les bits sont transférés par paquet de 8, bit de poids fort en premier (à l'inverse du rs232), qu'il s'agissse des données ou des adresses. Le recepteur envoie un acquitement après chaque envoi de 8 bits par l'émetteur. l'acquitement est le 9ième bit.

les différentes conditions

Les 4 figures ci-dessous illustrent les 4 conditions du bus. Il ne manque que l'acquitement, mais en fait l'acquitement est un bit comme les autres si on se contente de regarder l'état des lignes. Ce qui change c'est celui qui parle sur SDA, en l'occurence c'est celui qui a reçu le mot qui l'acquite en imposant SDA à 0. S'il laisse SDA à 1 c'est un non-acquitement (**NACK**).

L'adressage

Pour pouvoir communiquer avec les différents composants raccordés au bus, nous allons détailler le protocole de communication qui permet d'orchestrer les échanges. Dans un premier temps, et après avoir émis une condition de départ, le maitre indique avec quel esclave il souhaite se connecter, pour cela il envoie sur le bus l'adresse de l'esclave composé de 7 bits (donc 128 adresses différentes), puis un dernier bit indiquant le sens du transfert : Lecture (1) ou Écriture (0). Tous les esclaves scrutent le bus et voient la condition de départ, celui qui reconnait son adresse envoie un acquitement (SDA à 0).

Ecriture d'un octet

La figure ci-dessus montre l'écriture d'octet. Le maître décide d'arrêter l'échange en faisant une condition d'arrêt. L'esclave se contente d'envoyer des acquitements après chaque réception.

Lecture d'un octet

La figure ci-dessus montre la lecture d'un octet. Il faut noter deux choses. La première est qu'après avoir envoyé l'adresse, le maître lâche SDA pour lire l'octet de donnée émit par l'esclave. La seconde est que pour signifier à l'esclave que la transaction s'achève il envoie un NACK (Not ACKnowledge) à l'esclave, puis il place une condition d'arrêt. Ce NACK ne signifie pas que la donnée n'a pas été correctement reçue mais que le maître veut stopper l'échange et que donc l'esclave doit lâcher SDA. Vous pouvez vous demander comment ferait le maître pour indiquer qu'il n'a pas bien reçu la donnée, et bien il ferait pareil, il imposerait un NACK. La différence est qu'il recommencerait la transaction, tout de suite ou plus tard en reprenant là ou il faut.

Espaces d'adressage

La version standard du protocole utilise des adresses sur 7 bits. L'usage de ces adresses est réglementé. Les esclaves doivent se reconnaitre et il n'est pas possible d'avoir deux esclaves à la même adresse. C'est pour cette raison que certains composants permettent de choisir en partie ou totalement l'adresse. Le bus I2C permet le branchement à chaud (hotplug) mais pas l'autoconfiguration (plug-n-play).

Adresses réservées par le protocole I2C

- `0000000` General Call, adresse reconnue par tous les esclaves
- `0000001` réservé aux composant CBUS (ancètre)
- `0000010` réservé aux autres systèmes
- `0000011` réservé au futur
- `00001--` réservé aux composants haute vitesse (3.4Mbauds)
- `11111--` réservé au futur
- `11110xy` adressage 10 bits

112 Adresses réservées par les fournisseurs ou réquisitionables

- `0100xyz` sextuble CNA
- `1010xyz` 1024 x 8 bits eeprom
- `1110000` Télémetre à ultra-sons
- `.......` toutes les autres

En plus de ces adresses I2C, il y a souvent un second système d'adressage propre au composant accédé. Par exemple si le composant accédé est une ram. Ce composant ram a une adresse I2C et des adresses de cases internes. Du point de vue I2C, ces adresses de cases sont des données. Il y a donc en général un protocole spécifique propre à chaque composant.

Nous allons pouvoir expérimenter ces deux niveaux de protocole dans les expériences de ce TME. Nous verrons en effet que chaque composant dispose d'une adresse unique I2C chacun et de plusieurs adresses internes dont le mode d'accès spécifique est présenté dans la spécification du composant.

Pour finir

Nous venons de voir l'écriture et la lecture d'un octet unique. En fait, le nombre d'octets échangés n'est pas limité. Un maître peut décider d'écrire autant d'octets qu'il veut. L'esclave peut refuser une donnée en envoyant un NACK et demander ansi au maître de stopper l'échange. Pour la lecture, l'esclave ne dispose pas de moyen pour indiquer au maître qu'il n'a plus de données à fournir. C'est un protocole de plus haut niveau qui doit régler ce problème.

C'est le maître qui envoie les pulses d'horloge, mais l'esclave peut ralentir la transaction en maintenant la ligne SCL à 0. Le maître est capable de s'apercevoir de ce ralentissement car il lit la valeur de la ligne SCL en permamence et il peut se rendre compte que la valeur lue n'est pas celle attendue. Le maitre peut aussi de lui-même decider de faire disparaitre certains pulses pour, par exemple, se laisser le temps de réagir quand il est récepteur d'envoyer des acquitements. Cette caractéristique permet donc au maître et à l'esclave de prendre le temps nécessaire à chaque étape d'un échange, c'est particulièrement utile si c'est un programme en assembleur qui fait avancer l'échange.

Pour les deux figures précédentes, nous avons ajouté une notation des échanges qui simplifie leur description. On ne fait plus apparaitre les chronogrammes, mais juste les conditions et les mots échangés. Cette notation se retouve souvent dans les documentations des circuits à interface I2C, en particulier ceux que vous allez voir aujourd'hui. On n'a pas besoin de faire apparaitre le temps d'attente.

Conclusion hative

Nous allons arrêter là pour la description de l'I2C car nous n'avons pas besoin de plus pour ce TME. Encore une fois, nous vous invitons à lire la spécification officielle de Philips sur l'I2C. Le pic16f877 offre plus de choses que la simple écriture et lecture d'un octet mais nous ne le verrons pas aujourd'hui.

Ce que vous n'avez pas vu, c'est:

- Le mode d'adressage 10 bits.
- La méthode de résolution des conflits

Les circuits I2C de ce TME

Nous allons communiquer avec 2 circuits: un convertisseur numérique analogique et un télémètre à ultra-son. Nous allons commencer par le convertisseur car il peut être commandé en faisant seulement des écritures i2c. Le télémetre nécessite écritures et lectures et nous allons voir que c'est un peu plus compliqué.

Le module I2C du pic16f877

Le PIC peut agir comme un maitre ou un esclave. Ici nous le faisons fonctionner en maitre. Vous trouverez les informations dans la documentation technique fournie à partir de la page 24. Le bus I2C permet à plusieurs maitre de se partager les ressources. Ceci entraine des collisions et cela complexifie le contrôle par le PIC. Dans notre cas nous n'avons qu'un seul maitre, en conséquence il y a pas mal d'information inutiles dans la documentation. Pour comprendre la gestion du bus par le PIC, c'est-à-dire comprendre quels sont les registres à consulter et modifier pour réaliser une transaction, vous devez vous reporter aux chronogramme de transaction I2C maitre. Vous pouvez voir qu'une transaction se fait en controlant les bits SEN, PEN et SSPIF.

Travaux pratiques

Experience n°1

- A la lecture de la documentation commentez chaque ligne de l'initialisation du module I2C et des envois au DAC. Donnez un estimation de la durée de chaque étape et donc de la durée d'envoi d'un caractère.

Experience n°2

- Quelle est la fréquence du signal ?
- Augmenter la fréquence en faisant des trames i2c plus longue.
- Envoyer une dent de scie sur 2 sorties.

Experience n°3

- Commander le télémetre avec affichage sur le port LCD de la distance ou sur RS232.

Experience n°4

- Pour les plus avancés: usage des interruptions ! Expliquez le principe de fonctionnement de l'I2C avec les interruptions, je ne vous demande pas de le programmer vous n'aurez pas le temps.

Le modèle de programme fourni

Nous donnons un programme qui fait une rampe sur le DAC. Il n'a pas les commentaires puisque vous devez les ajouter. Notez aussi l'usage des macros pour la réservation des adresses de la mémoire, et surtout l'usage des macros pour les différentes étapes du transfert I2C. Je vous demande (suggère) de creer 2 fichiers i2c_macro.inc et i2c_fun.asm, que vous pourrez inclure respectivement au début et à la fin de votre programme i2c (c'est comme pour le lcd).

```
        list       p=16f877
        errorlevel -302
        list       n=0
        include    "p16f877.inc"
;-----------------------------------
        VARIABLE bank0_break=0x20
        VARIABLE bank1_break=0xA0
        VARIABLE bank2_break=0x110
        VARIABLE bank3_break=0x190
        VARIABLE bank4_break=0x70
bank0   macro
            VARIABLE bank_current=bank0_break
        endm
bank1   macro
            VARIABLE bank_current=bank1_break
        endm
bank2   macro
            VARIABLE bank_current=bank2_break
        endm
bank3   macro
            VARIABLE bank_current=bank3_break
        endm
bank4   macro
            VARIABLE bank_current=bank4_break
        endm
byte    macro register_name, size
            CBLOCK bank_current
                register_name : size
            ENDC
            VARIABLE bank_current=bank_current+size
        endm
;-----------------------------------
        CONSTANT DAC_ADDR = 0x40
        CONSTANT READ     = 1
        CONSTANT WRITE    = 0
i2c_init macro
            call i2c_init_fun
        endm
i2c_start macro
            call i2c_start_fun
        endm
i2c_stop macro
            call i2c_stop_fun
        endm
i2c_lwrite macro val
            movlw val
            call i2c_wwrite_fun
        endm
i2c_fwrite macro reg
            movf reg,w
            call i2c_wwrite_fun
        endm
;-----------------------------------
        BANK0
        byte  val, 1
```

```
;----------------------------------
        org     0
        PAGESEL reset
        goto    reset
        org     4
        return

reset
        i2c_init

main
LOOP
        i2c_start
        i2c_lwrite DAC_ADDR|WRITE
        i2c_lwrite 0
        i2c_fwrite VAL
        i2c_stop
        decf       VAL,f
        goto       LOOP
;----------------------------------
i2c_init_fun
        BANKSEL TRISC
        bsf     TRISC,3
        bsf     TRISC,4
        bcf     PIE1,SSPIE
        bsf     SSPSTAT,SMP
        bcf     SSPSTAT,CKE
        movlw   D'49'
        movwf   SSPADD
        clrf    SSPCON2
        BANKSEL SSPCON
        movlw   0x28
        movwf   SSPCON
        bcf     PIR1,SSPIF
        return
i2c_start_fun
        BANKSEL SSPCON2
        bsf     SSPCON2, SEN
        clrf    STATUS
        btfss   PIR1, SSPIF
        goto    $-1
        bcf     PIR1, SSPIF
        return
i2c_wwrite_fun
        movwf   SSPBUF
        btfss   PIR1, SSPIF
        goto    $-1
        bcf     PIR1, SSPIF
        return
i2c_stop_fun
        BANKSEL SSPCON2
        bsf     SSPCON2, PEN
        clrf    STATUS
        btfss   PIR1, SSPIF
        goto    $-1
        bcf     PIR1, SSPIF
        return
;----------------------------------
        END
```