Lycée technique Louis Armand
321, rue Lecourbe 75015 PARIS tel : 01.45.57.51.50

Architecture des microprocesseurs

Le microcontrôleur 80C552

Le bus I^2C

Description des spécifications du bus I^2C et applications simples.

J. FRANCOMME
Version 1.1 - 22/03/2002

J. FRANCOMME - francomme@univ-paris12.fr
Lycée Technique Louis Armand
321, rue Lecourbe 75015 PARIS tel : 01.45.57.51.50

Sommaire

J. FRANCOMME - francomme@univ-paris12.fr
Lycée Technique Louis Armand
321, rue Lecourbe 75015 PARIS tel : 01.45.57.51.50

Le principe du bus I^2C

Le bus I^2C *ou Inter-IC-Communication* a été conçu pour réaliser la liaison entre les circuits intégrés d'une même platine. II se charge de la communication entre les périphériques qui est assurée d'ordinaire par un bus parallèle**.** En développant ce bus, Philips a équipé la plupart de ses appareils électroniques destinés au grand public (appareils TV et radio, systèmes audio et radio, postes téléphoniques, systèmes électrique automobile, appareils électroménager…).

> *Pour ce type d'appareils, la vitesse de transfert de données entre périphériques n'est pas très importante (100 kbits / seconde max). De plus, l'utilisation d'un bus parallèle pour assurer la liaison entre chaque composant demanderait une surface sur la platine relativement importante. Comme le bus I^2C ne comporte que deux fils, le nombre de liaisons entre les composants est réduit ; la surface utilisée et par conséquent les coûts de fabrication sont ainsi moins importants. Pour Philips, le bus I^2C est devenu un* ***standard*** *pour les liaisons lentes sur une même platine.*

Les informations sont échangées au moyen de deux lignes **bidirectionnelles** : **SDA** (Serial DAta) et **SCL** (Serial CLock). Chaque circuit intégré possède une adresse unique qui le distingue des autres. Chaque composant peut émettre ou recevoir des informations suivant sa fonction. Le bus est piloté par un circuit appelé **maître** (Pour les Travaux Pratiques : la carte XEVA de Raisonance : http://www.raisonance.com) qui prend l'initiative du transfert des informations, décide du sens de ce transfert et gère la ligne SCL. Les autres composants sont alors désignés par le terme **esclave**.

Un bus peut prendre une configuration multi-maître si plusieurs composants peuvent être à tour de rôle maître ou esclave. A tout moment, un seul maître actif doit être présent sur le bus. Une procédure d'arbitrage est prévue pour éviter la perte ou la détérioration des informations lorsque plusieurs maîtres essaient de prendre la commande du bus simultanément. Mais sur la plupart des circuits, seul un seul composant réalise la fonction de maître étant donné la complexité de cette fonction.

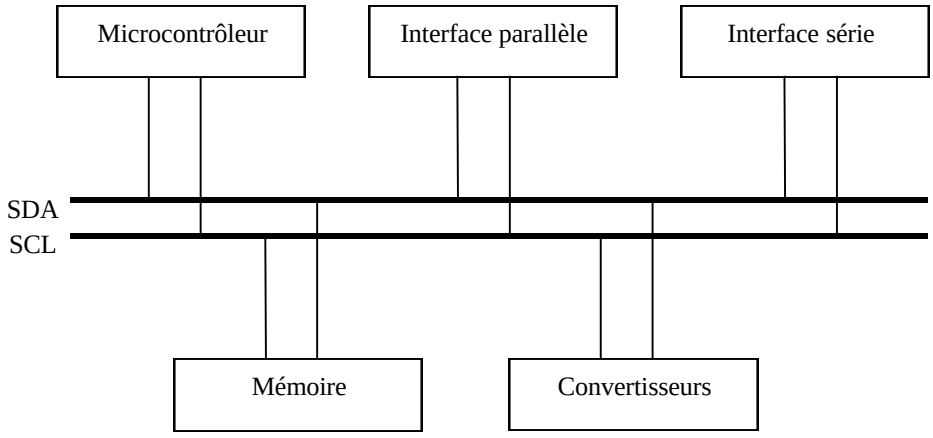


Figure 1 – ***Connexions d'unités au bus I^2C.***

Pour le bus I^2C, le niveau dominant est l'état bas (le niveau complémentaire est dit récessif). Les deux lignes SDA et SCL sont donc maintenues au niveau haut tant que le bus est libre. Sur les deux lignes, le niveau est reconnu bas pour toute tension inférieure à 1,5V et haut pour toute tension supérieure à 3V. Tous les niveaux intermédiaires ne sont pas pris en considération.

Les étages de sorties des circuits connectés au bus doivent avoir un drain ou un collecteur ouvert pour remplir la fonction « **ET câblé** ».

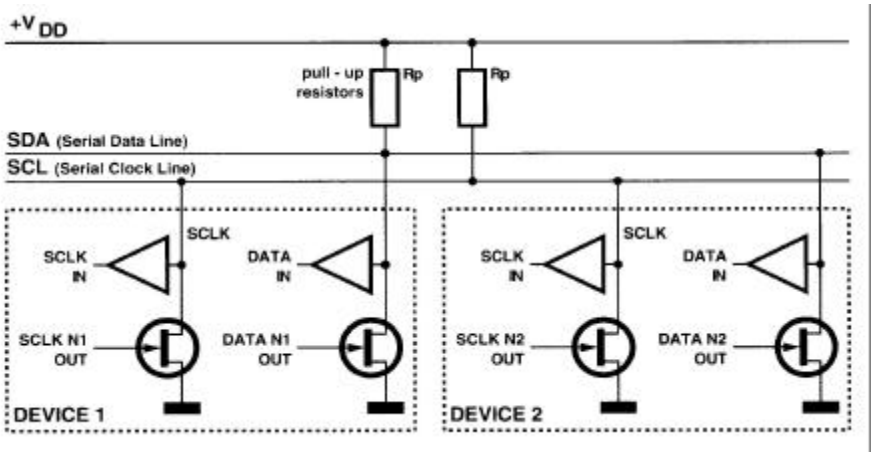


Figure 2 - ***Connexions d'interfaces I^2C sur le bus I^2C.***

Le protocole I^2C

Fonctionnement standard

Le bus I^2C appartient à la catégorie des bus série. Par opposition aux bus parallèles où les données sont transmises par bloc, les données sont ici envoyées bit par bit par groupe d'octet sur la ligne SDA. La ligne SCL fonctionne comme une horloge sérielle d'un registre à décalage. Tant que la ligne SCL est à l'état haut, les données de la ligne SDA doivent être stables. Lorsque la ligne SCL est à l'état bas, le circuit qui émet les données peut modifier l'état de la ligne SDA.

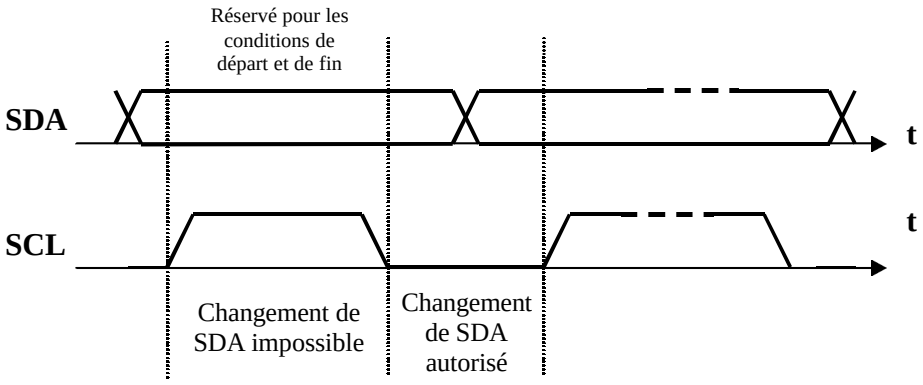


Figure 3 – ***Conditions de validité des données sur le bus I^2C.***

Certaines combinaisons particulières de niveaux et de fronts des deux lignes déterminent la **condition de départ ou d'arrêt** de la transmission des données.

- ✓ *Condition de départ* : un front descendant sur SDA quand SCL est à l'état haut.
- ✓ *Condition d'arrêt* : un front montant sur SDA quand SCL est à l'état haut.

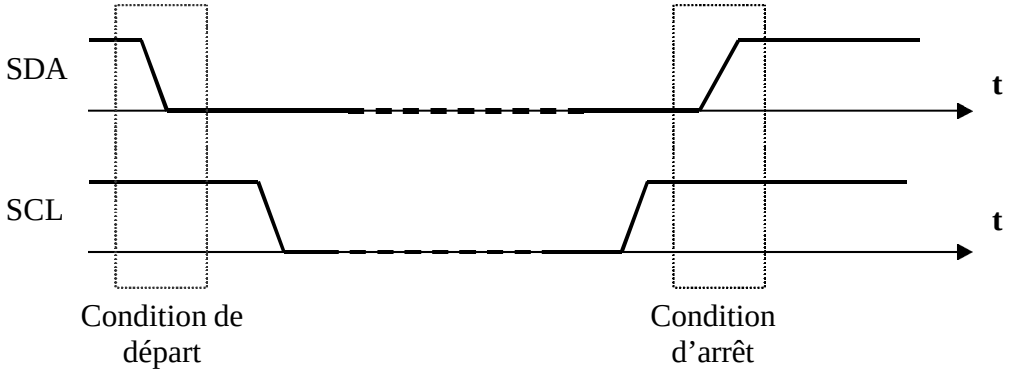


Figure 4 – **Les conditions de départ et d'arrêt**.

***Remarque** : les conditions de départ et d'arrêt sont toujours générées par le maître du bus.*

La condition de départ est immédiatement suivie par l'adresse du composant appelé par le maître. Chaque composant possède une adresse unique codée sur sept bits. Le huitième bit émis indique la direction du transfert (0 = écriture, 1 = lecture).

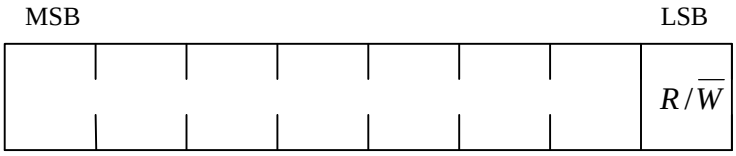


Figure 5 – ***Le format de l'octet d'adresse***.

***Remarque** : les circuits connectés sur le bus doivent avoir une adresse binaire unique, puisqu'il n'y a pas de ligne de validation dans un système bifilaire.*

Le transfert des huit bits de données est suivi par l'émission d'un bit acquiescement (ou **ACK**). Le circuit (maître ou esclave) qui reçoit les données accuse réception de l'octet envoyé en maintenant la ligne SDA (ligne de données) au niveau bas.

Une impulsion d'horloge est générée pour chaque bit de données transféré.

Transfert de données

Format des données

- ✓ Chaque mot transféré sur la ligne SDA doit contenir 8 bits. Le nombre d'octets qui peut être transmis par transfert est illimité.
- ✓ Chaque octet est suivi d'un bit d'acquittement fourni par le récepteur sur la ligne SDA.

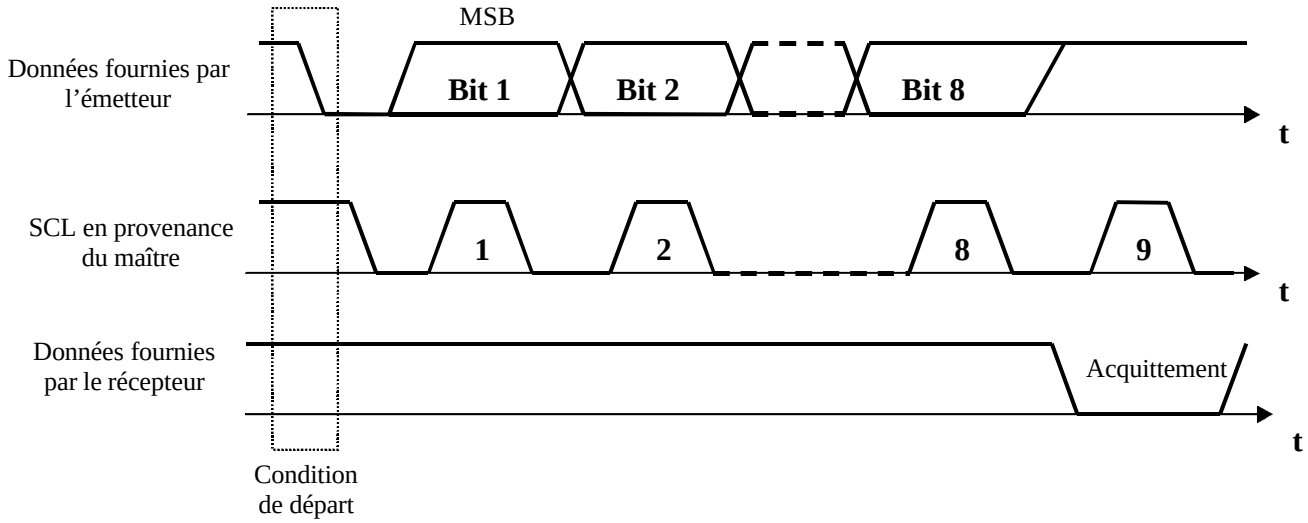


Figure 6 – ***Acquittement sur le bus I^2C***

Si le récepteur ne peut pas recevoir un autre octet de données complet tant qu'il n'a pas rempli une certaine fonction – par exemple, le traitement d'une interruption interne -, il peut maintenir la ligne d'horloge SCL à l'état bas pour mettre l'émetteur en attente. Le transfert reprend dès que le récepteur est prêt à recevoir un autre octet et libère la ligne SCL.

Acquittement

L'impulsion d'horloge correspondant à l'acquittement est générée par le maître qui a le contrôle du bus. Pendant cette impulsion, l'émetteur libère la ligne SDA (retour au niveau haut : niveau récessif), tandis que le récepteur remet la ligne SDA au niveau bas où elle reste stable pendant la durée de l'impulsion (figure 6).

Un récepteur qui a été adressé est normalement obligé de générer un acquittement après chaque octet reçu. Lorsqu'un récepteur n'accuse pas réception de son adresse, par exemple, lorsqu'il ne peut pas recevoir parce qu'il est occupé à exécuter une fonction temps réel, il doit maintenir la ligne de données au niveau haut (état de repos : récessif). Le maître peut alors générer une condition d'arrêt pour abandonner le transfert.

@ ***Question*** *: dessiner les chronogrammes des signaux sur les deux lignes SDA et SCL répondant au comportement énuméré ci-dessus.*

Si un récepteur esclave accuse réception de son adresse d'esclave mais qu'ultérieurement pendant le transfert il ne peut plus recevoir d'autres octets de données, le maître doit à nouveau arrêter le transfert. Dans ce cas, l'esclave n'ayant pas accusé réception d'un octet de données, la ligne de données reste au niveau haut (récessif) et le maître génère une condition d'arrêt.

Si un récepteur maître reçoit des données, il doit signaler une fin de données à l'émetteur esclave en n'acquittant pas le dernier octet de l'émetteur esclave (ça n'est pas poli, mais ils se comprennent). Ce dernier doit alors libérer la ligne de données pour permettre récepteur maître de générer la condition d'arrêt.

Fonctionnement en mode multi-maître

Etat du bus

- ✓ Le bus est considéré comme occupé après l'apparition de la condition de départ ;
- ✓ Il est considéré comme libéré au bout d'un certain laps de temps suivant l'apparition de la condition d'arrêt.

Remarque : les unités connectées au bus peuvent détecter facilement les conditions de départ et d'arrêt si elles possèdent l'interface nécessaire. Cependant, lorsque l'arbitre de bus est un micro-ordinateur, la ligne SDA doit être échantillonnée au moins deux fois par période d'horloge pour détecter à coup sûr les changements d'état.

Génération d'horloge et arbitrage

Synchronisation des horloges

Chaque maître génère sa propre horloge sur la ligne SCL pendant le transfert des données.

Les données ne sont valides que lorsque l'horloge est au niveau logique haut. Il est donc nécessaire de définir une horloge pour permettre l'exécution d'une procédure bit par bit. La synchronisation de l'horloge est réalisée par la connexion « **ET câblé** » des dispositifs de la ligne SCL. Lorsqu'un front descendant est présent sur la ligne SCL, tous les dispositifs candidats à la prise du bus (tous les maîtres en attente du bus) décomptent leur niveau bas sur la ligne SCL :

Un maître qui émet un niveau bas sur la ligne SCL, relâche la ligne à un moment donné pour le remettre au niveau haut. A partir de ce moment, il ne continue pas sa procédure mais il lit le niveau de la ligne SCL et attend qu'il soit effectivement repassé à un. Cela signifie que, aussi longtemps qu'un autre circuit intégré maintient la ligne au niveau bas, le maître qui demande un niveau haut est bloqué. Cet autre circuit peut être aussi bien un maître qui émet avec un débit plus lent qu'un esclave qui a besoin d'un temps d'attente pour exécuter des opérations internes. Le résultat est que le bus se synchronise la durée de l'état bas le plus long, donc le circuit intégré le plus lent.

Arbitrage

L'arbitrage permet l'attribution du bus aux différents maîtres. Le bus I2C comporte une procédure particulière dénommée CSMA/CD (Carrier Sense, Multiple Access with Collision Avoidance). Carrier sense est la procédure exécutée par un maître qui veut transmettre pour savoir si le bus est occupé ou non. L bus est considéré comme occupé si l'une des lignes est au niveau électrique bas. L'intervalle de temps pendant lequel les deux lignes doivent être au niveau haut pour déterminer un bus libre est désigné par $t_{HD;STA}$ dans les spécifications de Philips, sa valeur est de 4ìs. Une fois que le maître a reconnu le bus libre, il émet une condition de départ puis un premier octet qui représente l'adresse de l'esclave qu'il veut appeler. Si un autre maitre veut en faire autant en même temps, ce sont d'abord les impulsions de la ligne SCL qui sont comparées (synchronisation des horloges). Pendant les états hauts de la ligne d'horloge, chaque maitre compare l'état réel de la ligne de donnée SDA à la valeur qu'il a lui-même demandée. Le mécanisme est le même que celui de la synchronisation des horloges. Un maître qui émet un état haut (à cause de la fonction **ET câblé** reconnaît qu'un autre maître utilise le bus et il cesse sa transmission.

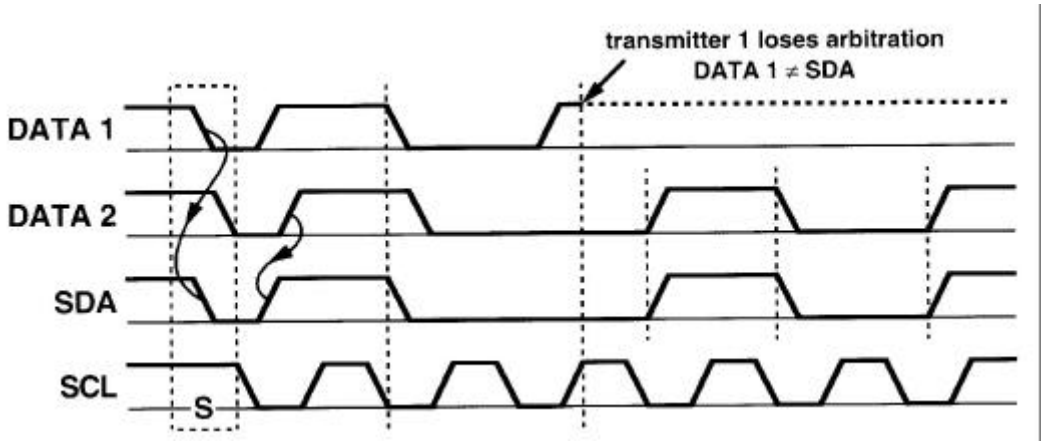


Figure 7 ***– Le mécanisme anti-collision***

Ce mécanisme est désigné par *Collision Avoidance*, puisque le maître gagnant n'entre pas en collision avec son concurrent. Il vient en complément d'un autre mécanisme, dit *Collision Detection*, qui oblige les deux maîtres à libérer le bus si un maitre imparfaitement conforme au standard I2C parvient à provoquer une collision.

Les composants I²C les plus courant :

Le bus I^2C est supporté par une vaste gamme de microcontrôleurs et de circuits périphériques réalisés dans différentes technologies de semi-conducteur (CMOS, NMOS, Bipolaire).

Pour l'ensemble de l'étude les étudiants utiliseront une carte sur laquelle sont implantés

- ✓ Deux CAN/CNA (PCF 8591A de Philips)
- ✓ Une interface d'entrée/sortie numérique 8 bits (PCF 8574A de Philips)
- ✓ Un driver d'afficheur 7 segments et ses afficheurs (SAA 1064 de Philips)

Description de chacun des composants

1) Le convertisseur A/N, N/A : PCF 8591A de Philips

Le PCF 8591A est muni de :

- quatre entrées analogiques AIN0→AIN3,
- d'une sortie analogique AOUT
- d'une interface I^2C.

L'adressage du composant se fait sur 3 broches A0→A2.

La commande se fait par :

- l'émission de l'octet d'adresse « **1 0 0 1 A2 A1 A0 R/**$\overline{W}$ »
  (R/$\overline{W}$ détermine le sens de la conversion),
- suivi d'un octet de commande qui détermine le mode de conversion.

Remarque *: La vitesse de conversion est entièrement déterminée par la vitesse du bus I^2C.*

✍ Imprimer à l'aide de l'oscilloscope numérique une trame I2C. La sonde est à connecté sur l'outil de développement XEVA de 'Raisonance'.

✍ Retrouver sur cette trame l'ensemble du message envoyé par le maitre à un esclave et la réponse de celui-ci.

2) L'interface d'entrée/sortie parallèle : le PCF 8574A de Philips

Le PCF 8574A permet le pilotage de huit entrées ou sorties par le bus I^2C.
C'est ici le composant le plus facile à utiliser et le plus fiable.
Il dispose

- de huit lignes numériques bidirectionnelles,
- d'une interface I^2C.
- d'une sortie $\overline{INT}$ qui génère une demande d'interruption lorsque l'une des entrées change d'état,

Utilisation de la sortie interruption : Le maître du bus I^2C averti par l'arrivée d'un évènement peut ensuite aller lire le port du PCF 8574A.

L'adressage du composant se fait sur trois broches A0→A2.
Le contrôle se fait par

- l'émission de l'octet d'adresse « **0 1 1 1 A2 A1 A0 R/**$\overline{W}$ »
- suivi des octets de données.

3) L'interface pour la commande des afficheurs à leds: SAA1064 de Philips

Ce module permet de gérer par multiplexage 2 groupes de 2 afficheurs à 7 segments à anode commune et point décimal.

Le SAA 1064 dispose :

- de 2 fois 8 sorties numériques
- d'une interface I^2C,
- d'une entrée d'adressage analogique ADR,
- et de 2 sorties MX1 et MX2 ;

Ces deux dernières sont connectées à la base de transistors NPN.
Le niveau de ces deux sorties détermine l'envoi des données vers chaque groupe d'afficheurs.
La fréquence du multiplexage étant de l'ordre de 150 hertz, l'œil humain ne peut discerner que les afficheurs s'allument et s'éteignent alternativement.

L'adressage du composant se fait de manière analogique par une broche unique ADR (le nombre de broches étant limité).

Le rapport entre la tension d'alimentation du SAA 1064 et la tension sur ADR détermine après une conversion A/N le niveau des deux bits d'adresse A1 et A0.

La tension ADR peut ainsi être réglée à l'aide d'un simple diviseur résistif.

Le contrôle se fait par l'émission

- de l'octet d'adresse « **0 1 1 1 0 A1 A0 R/**$\overline{W}$ ».
- Il est suivi par un octet d'instruction qui règle l'utilisation des registres du SAA 1064
- puis par un octet de commande qui détermine le mode d'utilisation de l'afficheur et enfin par les octets destinés à chaque afficheur.

Remarque *: Il est à noter que le SAA 1064 et PCF 8574A ont le premier groupe de l'octet d'adresse (les quatre premiers bits) identiques ; le choix des résistances dans le pont diviseur de tension est donc important afin d'éviter tout conflit avec les autres unités*

Valeur de la tension sur ADR	*Bits A0/A1*	*Valeur de l'adresse en écriture (binaire)*	*Valeur de l'adresse en écriture (hexa)*	*Valeur de l'adresse en lecture (binaire)*	*Valeur de l'adresse en lecture (hexa)*
V_{EE}	0 / 0	01110**00**0b	70h	01110**00**1b	71h
3/8 de V_{CC}	0 / 1	01110**01**0b	72h	01110**01**1b	73h
5/8 de V_{CC}	1 / 0	01110**10**0b	74h	01110**10**1b	75h
V_{CC}	1 / 1	01110**11**0b	76h	01110**11**1b	77h

table 1 ***– Adressage du SAA1064***

4) L'EEPROM : le PCA8581C de Philips

Le PCA8581C est une EEPROM série pour laquelle on accède aux données par un bus I^2C.
L'adressage se fait par les trois broches A0→A2.
Le contrôle se fait par l'émission de l'octet d'adresse 1 0 1 0 A2 A1 A0 R/$\overline{W}$

5) L'horloge/calendrier temps réel : le PCF 8583 de Philips

Cf. sujet de travaux pratiques.

J. FRANCOMME - francomme@univ-paris12.fr
Lycée Technique Louis Armand
321, rue Lecourbe 75015 PARIS tel : 01.45.57.51.50

Mise en oeuvre d'une communication I²C avec le 80C552 de Philips Semiconductor

1) Présentation générale de l'interface série I²C (SIO1)

Les lignes pour la communication I²C sont accessibles sur le port P1 par les lignes **P1.6** et **P1.7** respectivement **SCL** (Serial Clock Line) et **SDA** (Serial DAta line) ;

Le 80C552 respecte les spécifications du bus I²C (cf. document pdf accessible sur se site http://jackece.fr.fm – en anglais) et supporte tous les modes de transfert.

Afin de pouvoir utiliser convenablement l'interface I²C, il est nécessaire d'initialiser les deux lignes **SDA** et **SCL** à l'état haut, (qui correspond à l'état récessif : l'état non absorbant) ce qui permet à tous les composants connectés sur le bus de pouvoir envoyer des 'informations' (commencer une communication si le composant en question est un maître, renvoyer un accusé de réception ou des données à la suite d'une requête s'il s'agit d'un esclave).

Les caractéristiques essentielles du 80c552 pour la communication série avec le protocole I²C sont les suivantes :

- ✓ Transfert d'information bidirectionnel entre maître et esclaves
- ✓ Fonctionnement multi maître possible
- ✓ Procédure d'arbitrage pour la prise du bus par deux maîtres
- ✓ Procédures de synchronisation des horloges permettant la communication sur le bus de différents composants ayant des vitesses de transfert différentes.

Le port série du 80c552 prenant en charge la communication série avec le protocole I²C est **SIO1.** Cette interface I²C est gérée à l'aide 4 registres de 'fonction spéciale' (dans la zone des SFR).

S1CON : registre de contrôle de SIO1
S1STA : registre d'état de SIO1
S1DAT : registre de données de SIO1
S1ADR : registre d'adresse de SIO1 (pour le fonctionnement en esclave du 80C552)

Le port série peut fonctionner avec 4 modes différents :

Mode émetteur maître : les lignes SCL et SDA du bus I²C sont contrôlées par le microcontrôleur ; le premier octet transmis contient l'adresse de l'esclave (7 bits) ainsi qu'un bit de direction (**R/**$\overline{W}$) ;

Dans ce mode de fonctionnement :

- ✓ le bit **R/***W* est à l'état logique 0 (écriture)
- ✓ 8 bits sont transmis en série à chaque fois
- ✓ Après chaque octet est attendu un bit d'accusé de réception (ACK)

- ✓ La condition de départ et la condition de fin indiquent respectivement le début puis la fin d'un transfert série.

Mode Récepteur maître :

La description est identique à celle vue précédemment à l'instar du bit **R/**$\overline{W}$ qui prend la valeur 1.

Mode émetteur esclave : Sera vu ultérieurement

Mode récepteur esclave : Sera vu ultérieurement

2) Description des registres de l'interface série I^2C (SIO1)

a. Le registre **S1ADR** sera chargé avec une adresse sur 7 bits (les 7 bits de poids forts) utilisée lorsque le composant sera dans les modes de fonctionnement en esclave (émetteur ou récepteur).
Le bit de poids faible (LSB) est utilisé pour valider la reconnaissance d'un appel général (GC : General Call).

b. Le registre **S1DAT** (registre à décalage) est un registre 8 bits qui contient l'octet à transmettre sur la ligne série ou l'octet reçu.
Ce registre est toujours décalé de la droite vers la gauche, ce qui fait que le bit 7 (bit de poids fort de l'adresse) est toujours en première position lors d'une émission et de même lors de la réception c'est le premier bit reçu qui sera stocké à la fin de la réception dans le 7 du registre S1DAT.

Sens du décalage

S1DAT (DAh)	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0

Figure8 ***– Contenu du registre S1DAT***

c. Le registre de contrôle **S1CON** est utilisé par le microcontrôleur pour contrôler les différentes fonctions de SIO1 :
   - ➢ Condition de 'départ' et 're-départ' de la transmission série
   - ➢ Condition de 'fin' d'une transmission série
   - ➢ Vitesse de transfert
   - ➢ Gestion du bit d'accusé de réception ACK

S1CON (D8h)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Figure9 ***– Contenu du registre S1CON***

ENS1 (ENable Bit on serial Line) :

- ✓ A l'état bas '0' force les lignes à l'état haute impédance ; Les signaux SDA et SCL entrants sont ignorés.
- ✓ A l'état haut '1' SIO1 est validé ; les latchs des lignes P1.6 et P1.7 doivent être mis à 1_L afin que la communication soit possible.

STA (STArt flag) :

- ✓ A l'état haut permet d'entrer dans la mode de fonctionnement 'maître' : SIO1 consulte le contenu du registre d'état (pour vérifier le mode de fonctionnement programmé) et génère une condition de départ si le bus est libre. Si le bus n'est pas libre, SIO1 attend une condition d'arrêt lui permettant de prendre le bus.
  Si le bit STA est mis à l'état haut alors que le composant est déjà en mode maître et que plusieurs octets ont été reçus ou émis, une nouvelle condition de départ est générée sur le bus I^2C (re-départ : repeated start condition) ; STA peut être mise à 1 à n'importe quel moment.
- ✓ A l'état bas aucune condition de départ ne peut être générée.

STO (STOp flag) :

- ✓ A l'état haut permet de générer une condition de STOP sur le bus I^2C lorsque le composant est en mode de fonctionnement 'maître'.
  Quand une condition d'arrêt est détectée sur le bus I^2C, le bit STO est mis à 0_L.
- ✓ A l'état bas aucune condition d'arrêt ne sera générée sur le bus I^2C.

SI (Serial Interrupt flag) :

- ✓ Quand le bit SI est l'état haut et que les bit EA et ES1 (interrupt Enable Register) sont à l'état logique haut, une interruption pour la ligne série est requise.
  SI est mis à l'état logique haut par hardware lorsque 25 des 26 états possibles de SIO1 sont activés.
  Le seul état qui ne force pas le bit SI à 1 est l'état 0F8h qui indique qu'aucune information d'état pertinente n'est disponible.
- ✓ Quand le bit SI est mis à l'état bas, aucune interruption de la ligne série ne sera prise en compte

AA (Assert Acknowledge flag) :

- ✓ Si le bit AA est mis à l'état haut, un bit d'accusé de réception devra être retourné sur la ligne de donnée SDA lors de $9^{ème}$ pulse d'horloge sur la ligne SCL dans les cas suivants :
  - o L'esclave adressé a bien reçu la donnée
  - o L'adresse d'appel général a bien été reçue tant que le bit GC de S1ADR est à l'état haut.
  - o Le maître a bien reçu un octet en mode maître récepteur
  - o Le microcontrôleur a bien reçu un octet en mode esclave récepteur

- ✓ Si le bit AA est à l'état bas, un NACK (no Acknowledge) sera retourné sur la ligne SDA lors 9ème pulse d'horloge sur la ligne SCL dans les cas suivants :
  - o Un octet a été reçu en le mode maître récepteur
  - o Un octet a été reçu en mode esclave récepteur

CR0, CR1, CR2 (Clock Rate bits):

Ces trois bits fixent la fréquence de l'horloge qui sera transmise sur la ligne SCL en mode maître.

Pour un quartz de 12 MHz, on obtient les fréquences suivantes pour les combinaisons des bits CRi :

CR2	CR1	CR0	Fréquence de l'horloge de transmission (kHz)	F_{osc} divisé par
			12MHz	
0	0	0	47	256
0	0	1	54	224
0	1	0	63	192
0	1	1	75	160
1	0	0	12.5	960
1	0	1	100	120
1	1	0	-	60
1	1	1	0.5<fréquence<62.5	96*(256-valeur de chargement du timer1)

table 2 ***– paramètrage de la vitesse de transmission sur la ligne série***

Remarque *: La valeur de ces bits n'est pas importante quand le microcontrôleur fonctionne en mode esclave car le port série SIO1 se synchronise avec n'importe quelle horloge ayant une fréquence maximum de 100 kHz.*

Le registre **S1STA** est un registre 8 bits accessible en lecture seule.

Les trois bits de poids faibles sont toujours à l'état bas.

Les 5 bits de poids forts contiennent le code d'état.

Il y a 26 codes d'états possibles dont on retrouve dans les tables 3 et 4 ci-dessous uniquement ceux concernant le mode maître émetteur et le mode maître récepteur.

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
08H	A START condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted; ACK bit will be received
10H	A repeated START condition has been transmitted	Load SLA+W or Load SLA+R	X X	0 0	0 0	X X	As above SLA+W will be transmitted; SIO1 will be switched to MST/REC mode
18H	SLA+W has been transmitted; ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
20H	SLA+W has been transmitted; NOT ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
28H	Data byte in S1DAT has been transmitted; ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
30H	Data byte in S1DAT has been transmitted; NOT ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
38H	Arbitration lost in SLA+R/$\overline{W}$ or Data bytes	No S1DAT action or No S1DAT action	0 1	0 0	0 0	X X	I²C bus will be released; not addressed slave will be entered A START condition will be transmitted when the bus becomes free

table 3 **– *Mode maître émetteur***

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
08H	A START condition has been transmitted	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received
10H	A repeated START condition has been transmitted	Load SLA+R or Load SLA+W	X X	0 0	0 0	X X	As above SLA+W will be transmitted; SIO1 will be switched to MST/TRX mode
38H	Arbitration lost in NOT ACK bit	No S1DAT action or No S1DAT action	0 1	0 0	0 0	X X	I²C bus will be released; SIO1 will enter a slave mode A START condition will be transmitted when the bus becomes free
40H	SLA+R has been transmitted; ACK has been received	No S1DAT action or no S1DAT action	0 0	0 0	0 0	0 1	Data byte will be received; NOT ACK bit will be returned Data byte will be received; ACK bit will be returned
48H	SLA+R has been transmitted; NOT ACK has been received	No S1DAT action or no S1DAT action or no S1DAT action	1 0 1	0 1 1	0 0 0	X X X	Repeated START condition will be transmitted STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
50H	Data byte has been received; ACK has been returned	Read data byte or read data byte	0 0	0 0	0 0	0 1	Data byte will be received; NOT ACK bit will be returned Data byte will be received; ACK bit will be returned
58H	Data byte has been received; NOT ACK has been returned	Read data byte or read data byte or read data byte	1 0 1	0 1 1	0 0 0	X X X	Repeated START condition will be transmitted STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset

*table 4 – **Mode maître récepteur***

3) Description des modes opératoires

Je ne décrirais dans un premier temps que les modes de fonctionnement abordés par mes étudiants en l'occurrence :

- ✓ le mode maître émetteur
- ✓ le mode maître récepteur

Le transfert dans chacun des modes opératoires est décrit sur les figures 10 et 11. Ces figures contiennent les abréviations suivantes :

Abréviations	Correspondances
S	Start Condition – condition de départ
SLA	7-bit slave address – adresse esclave sur 7 bits
R	Read bit (high level at SDA) – bit **R**/$\overline{W}$ à 1
W	Write bit (low level at SDA) – bit **R**/W à 0
$\overline{A}$	Acknowledge bit (low level at SDA)
A	Not Acknowledge bit (high level at SDA)
Data	8-bit Data byte
P	Stop condition – condition d'arrêt

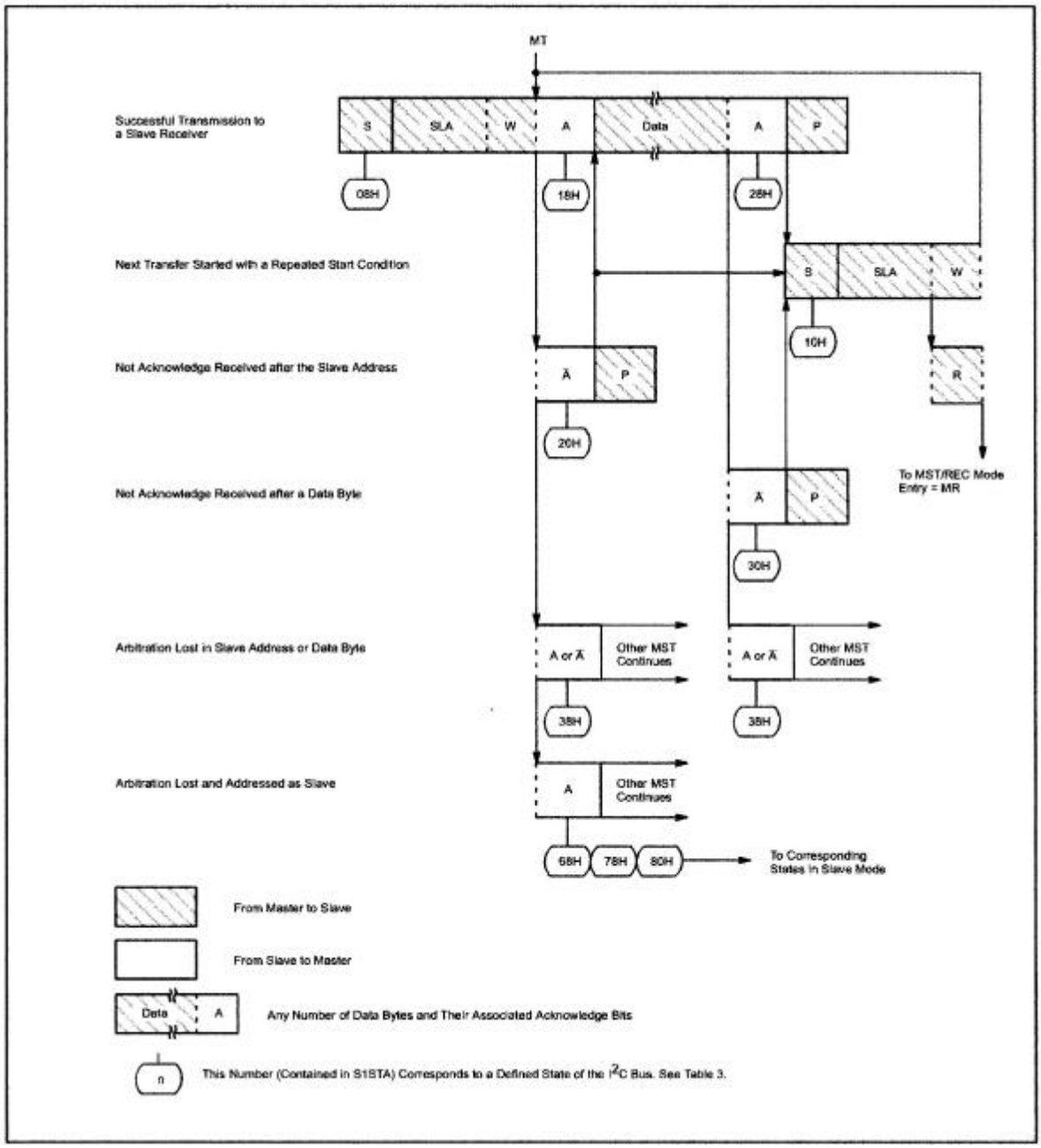


Figure10– Format et état d'une trame de données sur le bus I^2C en mode maître émetteur

J. FRANCOMME - francomme@univ-paris12.fr
Lycée Technique Louis Armand
321, rue Lecourbe 75015 PARIS tel : 01.45.57.51.50

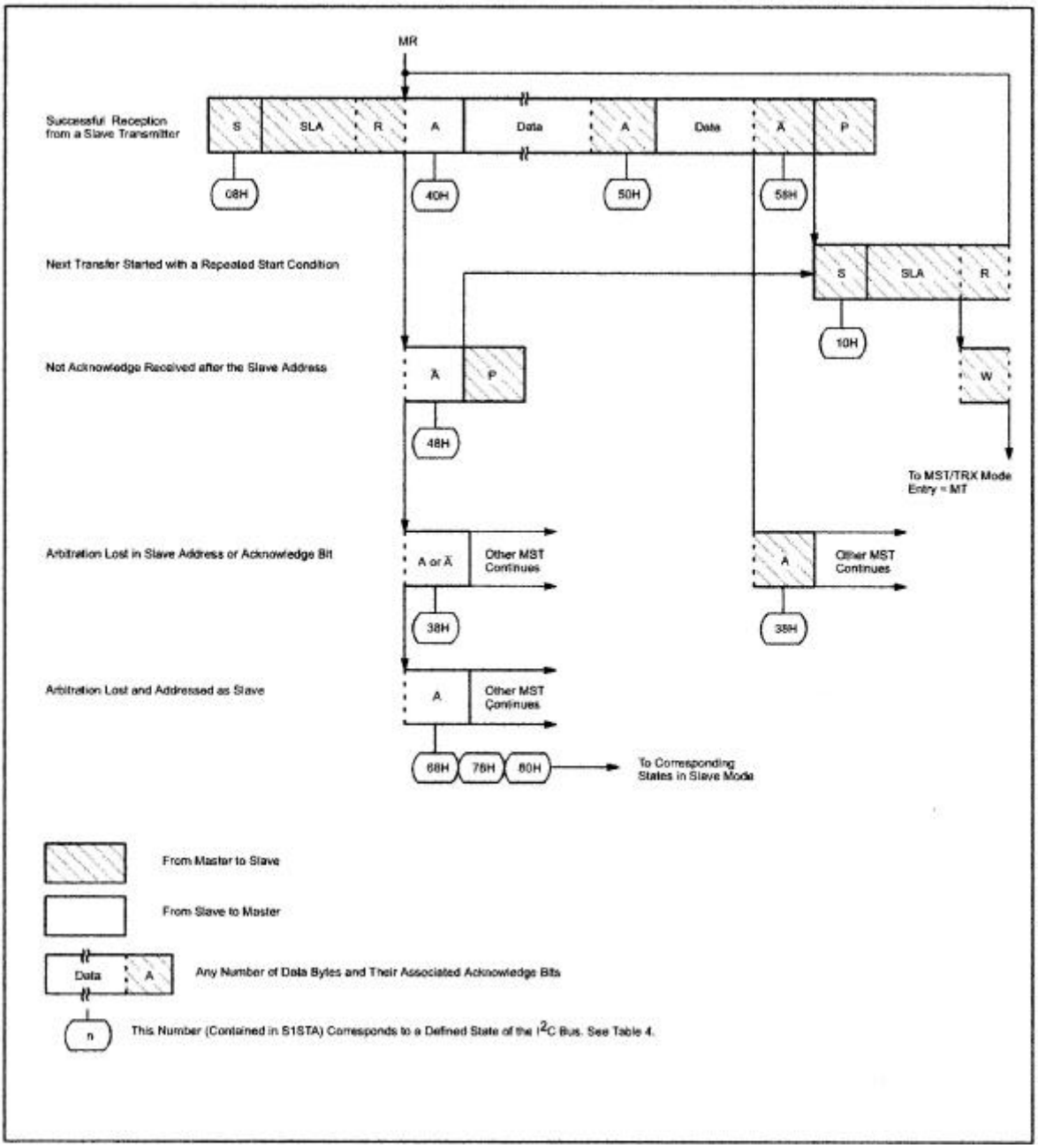


Figure11– ***Format et état d'une trame de données sur le bus I^2C en mode maître récepteur***

4) Elements pour la programmation en assembleur 80C552

Cette séquence de programmation utilise une ressource propre au 80C552 : ce n'est pas une ressource commune à tous les microcontrôleurs de la famille 80C51 ; il ne faudra donc pas oublier d'inclure le fichier mettant en relation les appellations des registres particuliers SFR du 80C552 au début de votre fichier source avec ma syntaxe suivante :

$include (reg_552.pdf)

Pour la compréhension générale de la valeur à enregistrer dans le registre de contrôle en fonction des différentes étapes de la communication il vous est proposé le modèle suivant (utilisé par Philips dans les spécification du bus I^2C) :

Génération de la condition d'arrêt

ENS1_NOSTA_STO_NOSI_AA_CRx pour indiquer que cet octet de contrôle :

- ✓ Valide le fonctionnement de l'interface série SIO1
- ✓ Ne génère pas de condition de départ
- ✓ **Génère une condition d'arrêt**
- ✓ Ne valide pas la prise en charge des interruptions
- ✓ **Attend un bit d'accusé de réception après l'envoi d'un octet**
  Ou envoi un bit d'accusé de réception après la réception d'un octet
- ✓ Sélectionne la vitesse de transmission sur la ligne série

Remarque : On considèrera que les seuls bits qu'il faut modifier dans notre cas de fonctionnement ne sont que : STA, STO et AA
Les autres sont fixés lors de l'initialisation et ne sont pas modifiés lors du déroulement de l'application

Soit : CRx = CR2 CR1 CR0 = 101_2 pour une vitesse de transmission =100KHz
ENS1 = 1 Interface série SIO1 toujours validée
SI = 0 prise ne compte des interruptions non validée

Libération du bus et ACK

ENS1_NOSTA_NOSTO_NOSI_AA_CRx pour indiquer que cet octet de contrôle :

- ✓ Ne génère pas de condition de départ
- ✓ Ne génère pas de condition d'arrêt
- ✓ **Attend un bit d'accusé de réception après l'envoi d'un octet**
  Ou envoi un bit d'accusé de réception après la réception d'un octet

Libération du bus et NACK

ENS1_NOSTA_NOSTO_NOSI_NAA_CRx pour indiquer que cet octet de contrôle :

- ✓ Ne génère pas de condition de départ
- ✓ Ne génère pas de condition d'arrêt
- ✓ **N'attend pas un bit d'accusé de réception après l'envoi d'un octet**
  N'envoi pas un bit d'accusé de réception après la réception d'un octet

Libération du bus et activation d'une condition de départ

ENS1_STA_NOSTO_NOSI_AA_CRx pour indiquer que cet octet de contrôle :

- ✓ Génère de condition de départ
- ✓ Ne génère pas de condition d'arrêt
- ✓ **Attend un bit d'accusé de réception après l'envoi d'un octet**
  Envoi un bit d'accusé de réception après la réception d'un octet

Remarque importante*: la mise à un ou à zéro d'un bit dans le registre de contrôle active l'élément en question et lance sur le bus l'ensemble des informations ;*
Exemple : pour envoyer un octet sur la ligne série I^2C*, il faudra mettre au préalable la donnée à transmettre dans le registre S1DAT puis écrire dans le registre S1CON pour activer la transmission et attendre le bit ACK ou NACK.*
Ce qui signifie aussi que l'écriture dans le registre de contrôle S1CON provoque la transmission de la donnée.

J. FRANCOMME - francomme@univ-paris12.fr
Lycée Technique Louis Armand
321, rue Lecourbe 75015 PARIS tel : 01.45.57.51.50

Questionnaire :

A. Définition de la terminologie du bus I^2C.

Donner la définition des termes suivants :

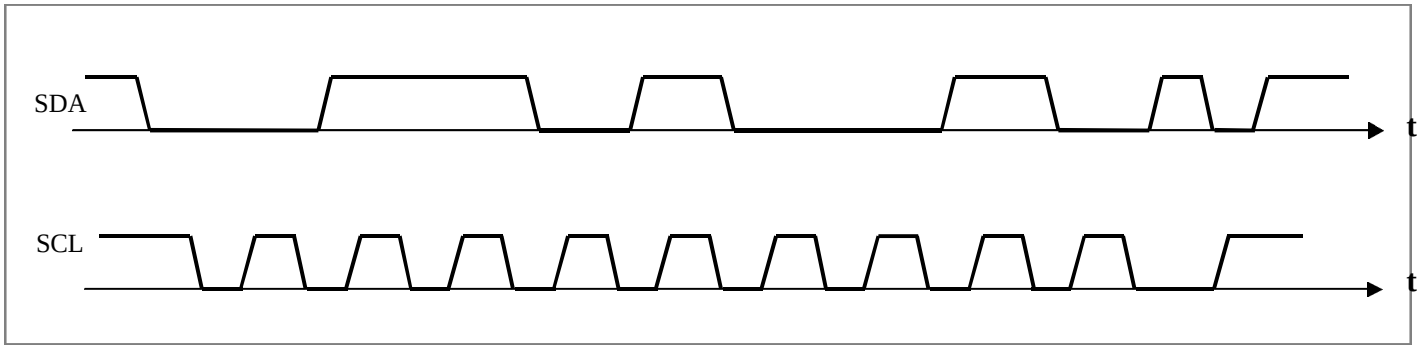
a. Emetteur
b. Récepteur
c. Maître
d. Esclave
e. Multi maître
f. Arbitrage

B. Description des conditions de fonctionnement du bus I^2C

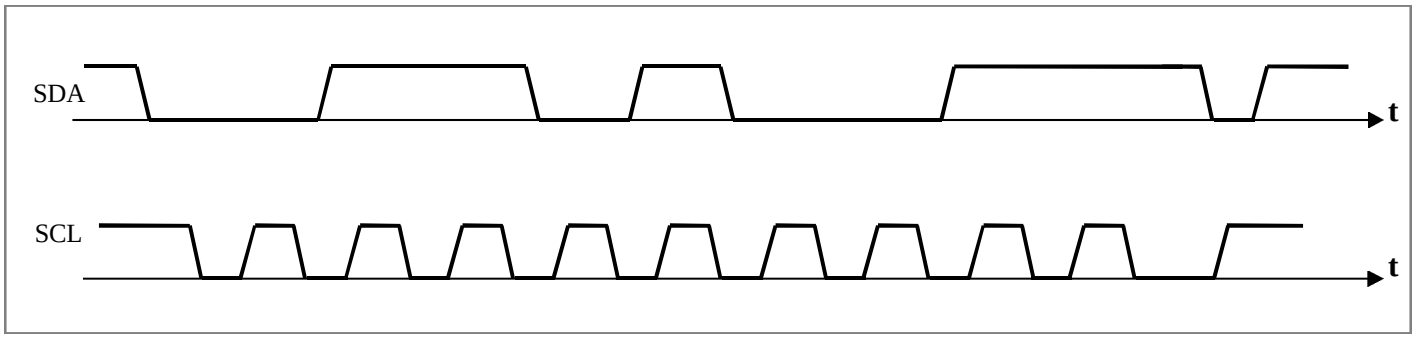
1) Combien de lignes sont utilisées par ce bus série ?
2) Donner le nom de chacune des lignes et leur mode de fonctionnement.
3) Donner le nom d'un système dans lequel plusieurs unités peuvent prendre le contrôle du bus de communication.
4) Qu'est ce qui permet de distinguer les différents circuits connectés sur le bus.
5) Que doit-il exister dans le système pour que plusieurs unités puissent prendre le contrôle du bus ?
6) Que contient le premier octet transféré sur le bus au début d'une communication entre deux éléments ?
7) Quel est le nom du bit qui suit l'octet d'adresse transmis sur le bus ?
8) quelle est la vitesse de transmission maximum sur le bus I^2C ?
9) De quoi dépend le nombre maximum d'éléments que l'on peut connecter sur un même bus I^2C ?

C. Description des signaux sur le bus I^2C

1) Repérez les différents éléments sur les chronogrammes ci-dessous sachant qu'il s'agit de l'envoi d'un octet d'adresse.
2) Retrouvez le message émis par le maître sur le bus I^2C à partir des chronogrammes suivants :



- Chronogramme 1 -



- Chronogramme 2 -

3) Décrivez et justifiez la différence essentielle que vous avez repéré sur ces deux premiers chronogrammes.
4) C'est a vous maintenant de compléter les chronogrammes en utilisant le cahier des charges exprimé ci-dessous :

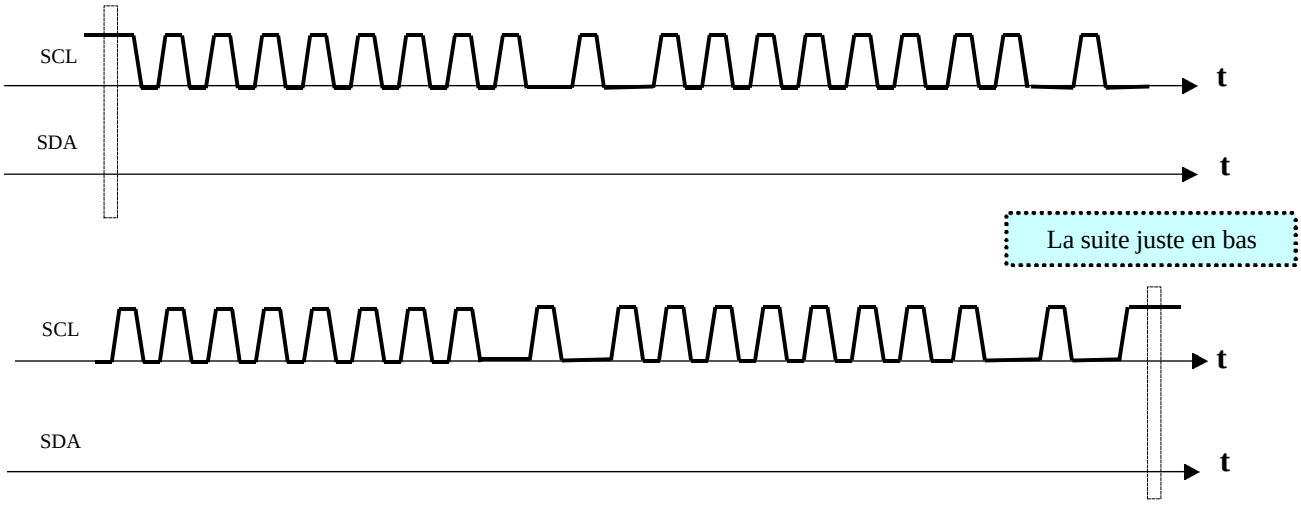
Dans cet exercice vous avez un maître et deux esclaves ;

- Le premier esclave est une interface d'entrée/sortie numérique de référence PCF8574A ; ses broches A0→A2 sont connectées à la masse.
- Le deuxième esclave est aussi une interface de même référence ; ses broches A0→A2 sont connectées à $+V_{CC}$.

Problème :

On veut envoyer les trois octets 01h, 02h, 04h vers le deuxième esclave

a) Décrivez le contenu des différents octets à envoyer sur le bus I^2C.
b) Décrivez les différentes étapes dans la communication entre le maître et l'esclave.
c) Compléter le chronogramme suivant :



5) Dans ce nouvel exercice, les éléments connectés sur le bus sont les mêmes que ceux de l'exercice précédent ;

Cette fois-ci l'émetteur veut lire une valeur provenant de l'esclave 1 et l'envoyer telle quelle à l'esclave 2.

a) Décrivez le contenu des différents octets à envoyer sur le bus I^2C.
b) Décrivez les différentes étapes dans la communication entre le maître et les esclaves.
c) Compléter le chronogramme suivant :