LE BUS I^2C

1. Bref historique du bus I^2C.

Ce bus a été élaboré au début des années 80 et fait partie de la grande famille des L.A.N. (Local Aera Networks - réseaux locaux) avec pour cible privilégiée le marché grand public.

Depuis des millions de téléviseurs, récepteurs de radio, autoradios utilisent ce moyen de communication interne à leurs propres systèmes. Après être entré par les synthèses de fréquences, les commandes clavier, les étages petits signaux l'I^2C est maintenant utilisé dans la totalité des fonctions présentes dans les autoradio, les Compacts Discs,… et les téléviseurs (bases de temps comprises).

Évidemment, lorsqu'un tel concept est utilisé dans de tels marchés, c'est que l'intérêt économique est fondé et de ce fait, des industries produisant de mêmes volumes, et fonctionnant par conséquent elles aussi à coût serré, se déclarent intéressées.

Ce fut notamment le cas de l'industrie automobile qui depuis quelques années utilise aussi le bus I^2C pour gérer la plupart des équipements de bord des véhicules.

Malgré de nombreuses similitudes, les fonctionnalités à remplir par les circuits n'étaient pas les mêmes, et de nouvelles gammes de produits (citons par exemple les convertisseurs AID-DIA, les RAM...) furent ajoutées aux catalogues des nombreux fabricants, qui avaient adhéré au bus I^2C grâce à l'achat de licences d'exploitation garantissant de ce fait une vraie compatibilité au bus indépendamment des marques.

Un autre marché demandeur de nombreuses fonctions à faible coût est celui de la téléphonie moderne, c'est-à-dire celle qui possède des mémoires, des générateurs de fréquences vocales, des afficheurs LCD...

Dans l'intervalle de temps, le marché professionnel a été un peu "chahuté" par des soucis économiques et, lors de l'optimisation des coûts, de nombreuses PME/PMI et de Grands Groupes ont redécouvert ce fameux bus I^2C. D'autres questions ont donc surgi et des compléments de famille sont apparus (interface $I^2C/\mu C$…).

2. Le protocole du bus I^2C en mode standard.

Comme dans tout protocole structuré, il comprend :

- des définitions des niveaux électriques haut et bas,
- des conditions de fonctionnement,

- des conditions de changement d'états,
- des conditions de validité des données,
- des conditions de départ START et d'arrêt STOP,
- des formats de mots,
- des formats de transmissions,
- des procédures d'acquittement,
- des procédures de synchronisation,
- des procédures d'arbitrage.

2.1. Terminologie du bus I^2C.

On appelle :
Emetteur : le composant qui envoie des données sur le bus.
Récepteur : le composant qui reçoit les données présentes sur le bus.
Maître : le composant qui initialise un transfert, génère le signal d'horloge et termine le transfert. Un maître peut être soit récepteur soit émetteur.
Esclave : le composant adressé par un maître. Un esclave peut être soit récepteur soit émetteur.
Multimaître : plus d'un maître peut tenter de commander le bus en même temps sans en altérer le message.
Arbitrage : procédure réalisée afin de s'assurer que si plus d'un maître essaie simultanément de prendre la commande du bus, un seul sera autorisé à le faire.
Synchronisation : procédure réalisée afin de synchroniser les signaux d'horloge fournis par deux ou plusieurs maîtres.
SDA : ligne des signaux de données (Serial DAta).
SCL : ligne des signaux d'horloge (Serial CLock).

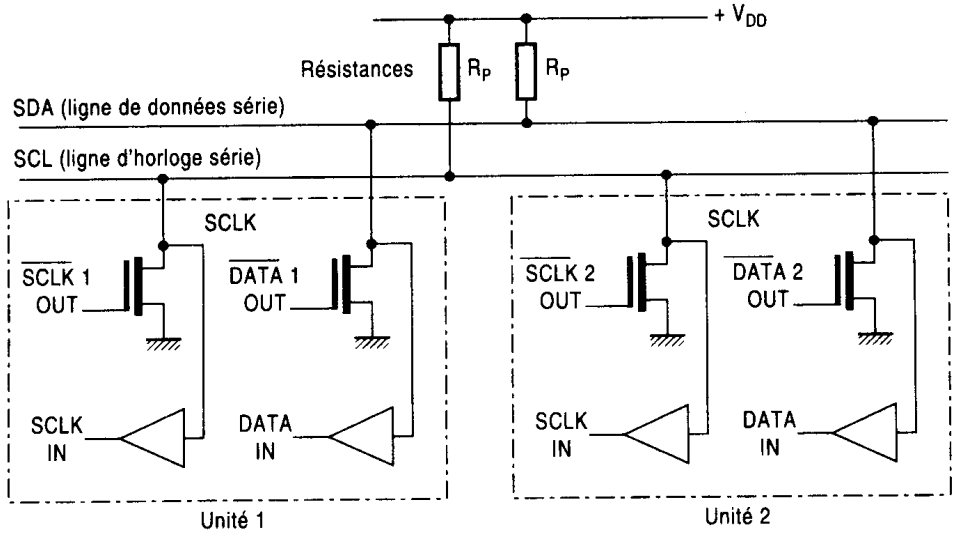
2.2. Terminologie d'un transfert sur le bus I^2C.

F (FREE) LIBRE : le bus est libre ; la ligne de données SDA et la ligne d'horloge SCL sont toutes deux à l'état haut.
S (START) DÉPART : le transfert de données commence avec une condition de start (ne pas confondre avec un bit de start) qui fait que le niveau de la ligne de données SDA change de haut à bas tandis que la ligne d'horloge SCL reste à l'état haut.
Le bus est alors dit **occupé**.
C (CHANGE) CHANGEMENT : pendant que la ligne d'horloge SCL est à l'état bas, le bit de donnée à transmettre peut être appliqué à la ligne de donnée SDA et il peut changer de niveau.
D (DATA) DONNÉE : un bit d'information (haut ou bas sur la ligne de données SDA) est considéré comme émis pendant le niveau haut de la ligne d'horloge SCL. Ce niveau doit être maintenu stable pendant toute la durée de la période haute de la ligne d'horloge afin d'éviter d'être interprété comme une condition de START ou de STOP.
P (STOP) ARRET : un transfert de données se conclut par une condition d'arrêt stop (ne pas confondre avec un bit de stop). Ceci se produit

lorsque le niveau sur la ligne de données SDA passe de l'état bas à l'état haut tandis que la ligne d'horloge SCL reste à l'état haut.
Le bus est à nouveau dit **libre**.

2.3. Configuration matérielle des étages reliés au bus.

Des exemples de schémas électriques de configuration des étages d'entrées / sorties réunis physiquement au bus sont donnés sur la figure suivante :



Aussi surprenant que cela puisse paraître une très grande partie de la force de ce type de bus réside dans le choix de cette configuration. Elle est très simple et pourtant...

Les deux lignes SDA, SCL sont chacune bidirectionnelles (entrante et sortante) et sont connectées à la ligne positive d'alimentation au travers de résistances ayant pour mission d'assurer une double fonction de charge et de rappel (dites de pull-up).
Au repos, lorsque les transistors ne sont pas conducteurs le bus est dit libre et ces deux lignes sont relâchées à l'état haut via les résistances de rappel.

Le choix de cette configuration de sortie (collecteur ou drain ouvert selon les technologies) permet (et tous autres composants reliés au bus doivent le permettre) de réaliser la fonction **ET câblé** pour les broches de même nom.

Le fait que l'on réalise simultanément sur une broche unique l'entrée et la sortie d'une information (SDA ou SCL) permet au circuit lui-même de s'auto-espionner, c'est-à-dire de pouvoir vérifier si sur sa propre sortie il y a bien l'information électrique que lui-même voulait y voir afficher.
C'est grâce à ce type de subterfuge qu'il est possible de réussir la synchronisation et l'arbitrage entre différents maîtres.

2.4. Conséquences de la configuration.

En ce qui concerne la **chargeabilité** ou encore le nombre de circuits que l'on peut relier simultanément au bus de nombreuses questions peuvent se poser.

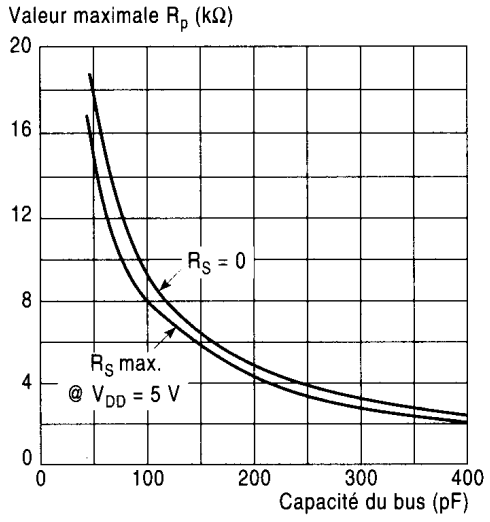
Chargeabilité statique.
Ne sachant combien, comment, quand… seront électriquement opérationnels les différents circuits reliés au bus, il faut tenir compte de l'éventualité du cas le plus défavorable du système.

Celui-ci correspond en fait à la chargeabilité maximale de courant que peut supporter le transistor de sortie toute technologie confondue bipolaire, CMOS... (valeur maximale de la spécification : 3 mA). Ceci permet de définir, dans le cas d'une tension d'alimentation de 5 V, la valeur minimale des résistances de rappel (se trouvant toutes en parallèle) à environ 1,5 kÙ (la valeur maximale n'étant pas critique).

Chargeabilité dynamique.
Différents paramètres sont en effet à prendre en considération pour définir la chargeabilité dynamique :

- la valeur maximale du temps de montée (1 µs) que peuvent prendre les signaux sur SDA et SCL (cette valeur est principalement liée à la valeur maximale du débit - 100 kbits/s, mais elle est totalement indépendante du débit utilisé) ;
- le fait qu'il pourrait y avoir, connecté sur le bus, un (des) maître(s) fonctionnant au débit le plus élevé;
- la valeur maximale de la capacité de sortie d'un circuit élémentaire (20 pF).

Si l'on prend uniquement en compte la valeur maximale du temps de montée, il est aisé de définir quelle serait la valeur maximale de la capacité de charge équivalente du bus.
Le résultat de ces calculs est donné par la courbe de la figure suivante.



On peut donc conclure que 400 pF représentent une chargeabilité maximale de 19 (+ 1, le maître) circuits à capacité maximale de 20 pF ou que l'on peut déporter un circuit (de 20 pF max) à 3,6 m en le reliant avec du câble à100pF/m.
Toutefois, les performances sont meilleurs car :

- tous les circuits ne sont pas à la valeur maximale de 20 pF ;
- il est facile de réduire les temps de montée quand on en a envie en bufférisant les sorties, diminuant ainsi les valeurs apparentes des résistances de sortie.

2.5. Les niveaux haut et bas.

Du fait qu'une grande variété de composants de différentes technologies (NMOS, CMOS, bipolaire...) peuvent être connectés sur le bus, les niveaux logiques O (bas) et 1 (haut) ont dû être déterminés de façon à ne pénaliser aucune possibilité d'emploi. C'est pour cette raison que les valeurs suivantes ont été retenues.

Sous 5 V d'alimentation :

$$V_{il\ max} = 1,5\ V \text{ et } V_{ih\ min} = 3,0\ V$$

et pour des circuits de type CMOS supportant de grandes plages de tension d'alimentation :

$$V_{il\ max} = 0,3\ V_{DD} \text{ et } V_{ih\ min} = 0,7\ V_{DD}$$

et quelle que soit la valeur de l'alimentation :

$$V_{ol\ max} = 0,4\ V \text{ à } 3\ mA \text{ de courant extrait}$$

Toutefois, il n'est pas recommandé de connecter sur un même bus des composants ayant différentes tensions d'alimentation.

2.6. Transfert des données sur le bus.

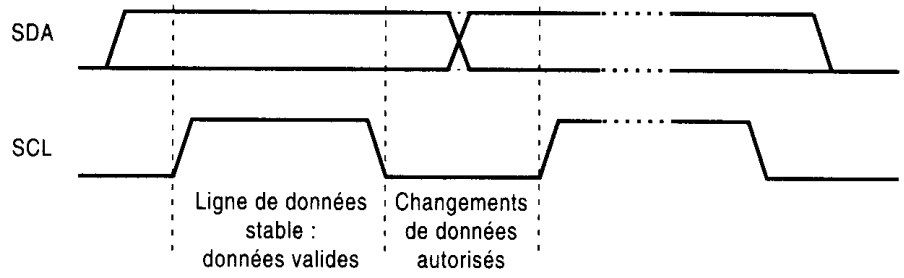
Le transfert des informations est très spécifique au protocole du bus I^2C.

Conditions de fonctionnement des transferts.
Principe de base : une impulsion d'horloge est générée à chaque fois qu'un bit est transféré.

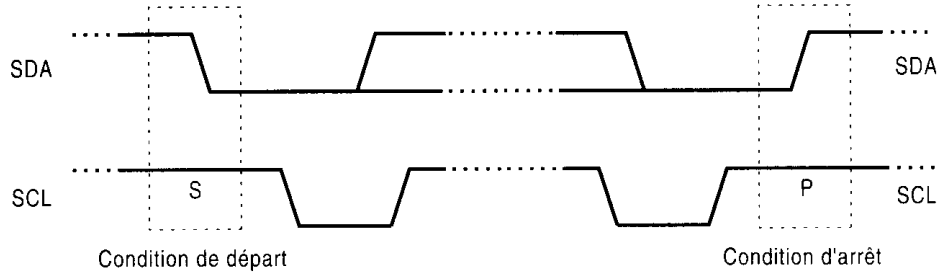
Conditions de changement d'état et de validité des données.
Ces deux conditions sont intimement liées et il est raisonnable de prendre la ligne d'horloge SCL pour référence.
Dans ce cas on définit :

- La validité d'une donnée: Une donnée est considérée comme valable lorsque la ligne de donnée SDA est stable (haute ou basse) pendant que la ligne d'horloge SCL est à son état haut.

- Le changement d'état: Quel que soit le niveau (haut ou bas) de la ligne de données SDA sa valeur ne pourra changer que lorsque la ligne d'horloge SCL sera à l'état BAS.



Conditions de départ start et d'arrêt stop.
À l'intérieur du protocole de définition de l'I^2C les situations qui régissent ces conditions sont uniques. Il est à noter que ce sont des conditions de départ et de stop et non des bits de départ et de stop.



Condition de start : Cette situation a lieu et uniquement lieu lorsque la ligne de données SDA passe de l'état haut à l'état bas tandis que la ligne d'horloge reste à l'état haut.
Condition de stop : Cette situation a lieu et uniquement lieu lorsque la ligne de données SDA passe de l'état bas à l'état haut tandis que la ligne d'horloge reste à l'état haut.

À tout cela il faut ajouter les compléments suivants :
- les conditions de start et de stop sont toujours créées par le maître,
- le bus est dit occupé après la condition de départ,
- le bus sera considéré comme libre après un certain temps (bien défini) après la condition de stop.

2.7. Les formats de transmission.

2.7.1. Format des mots.

Examinons maintenant les formats des mots et des transferts des données.
Chaque mot transmis sur la ligne de donnée SDA doit avoir une longueur de 8 bits. Jusque là rien de particulier et ceci permettra de

traiter directement les mots reçus à l'aide d'un microcontrôleur standard.
Les données contenues dans le mot sont transférées avec le bit de poids fort (MSB) en tête.
La signification de chacun des mots transmis sera définie lors de la définition du format de transmission.

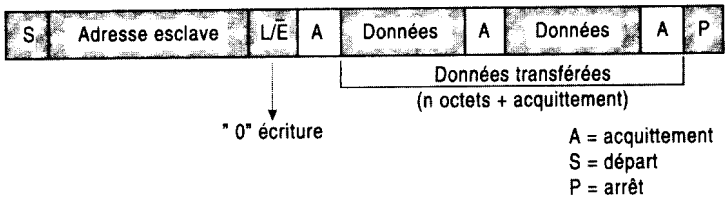
2.7.2. Formats des transferts.

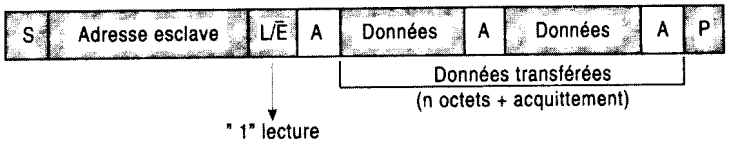
Principe de base d'un transfert.
Chaque transfert commence par une condition de START.
Chaque mot transmis doit être suivi d'un bit d'acquittement. Le système récepteur peut, s'il le désire, forcer l'émetteur à interrompre momentanément son émission entre deux mots successifs afin d'effectuer une tâche jugée par lui instantanément plus prioritaire. Pour indiquer cela à l'émetteur le récepteur doit forcer (maintenir) au niveau BAS la ligne d'horloge SCL. Dès qu'il relâchera la ligne d'horloge celle-ci remontera et indiquera à l'émetteur qu'il peut continuer son transfert, s'il le désire.
Le nombre de mots transmis lors d'un transfert est illimité.
Chaque transfert se termine par une condition de STOP générée par le maître.

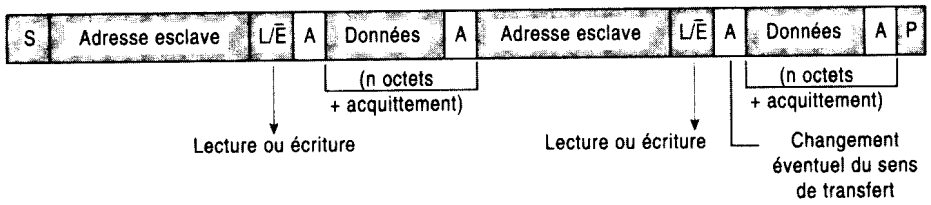
Le premier mot transmis est l'adresse de l'esclave que le maître souhaite sélectionner.
Cette adresse a une longueur de 7 bits.
Le huitième bit (LSB) de ce premier mot a une signification particulière. Il est baptisé **R/W bit** (bit lecture / écriture). Il sert à indiquer la direction que vont prendre les données circulant sur le bus :

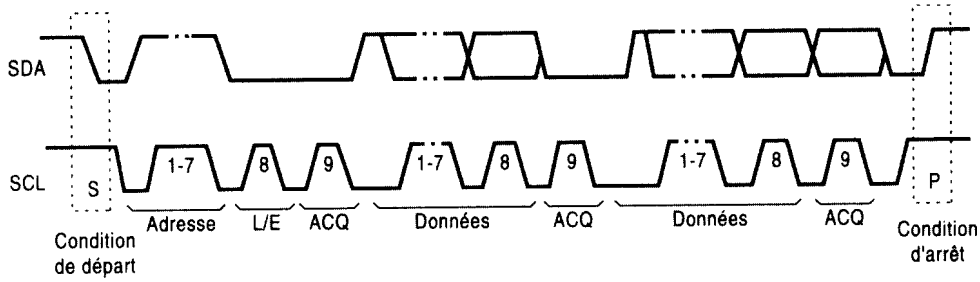
- **zéro** indique une transmission d'écriture du maître vers l'esclave,
- **un** indique une demande de données (ce qui revient en fait à vouloir effectuer une lecture des données que détient l'esclave).

Lorsqu'une adresse est envoyée, tous les composants présents physiquement sur le bus comparent les sept premiers bits qui suivent la condition de start à leur propre adresse. Si celle-ci correspond exactement à la sienne, il se considère comme adressé. Il attend alors la transmission du huitième bit (R/W) pour savoir s'il doit se considérer comme un esclave récepteur (où l'on va écrire), ou comme un esclave émetteur (à qui l'on va demander de lire et d'envoyer son contenu).









Examinons maintenant avec détails comment sont constitués les bits d'adresse.

Évidemment n'importe qui pourrait faire n'importe quoi (tel constructeur, telle fonction, tel circuit, telle adresse !) et l'on se retrouverait dans le commerce avec un répertoire d'adresses totalement anarchique. Afin d'éviter cela un **I²C Committee** coordonne les allocations aux différents fabricants (voir un exemple sur le tableau suivant).

TYPE	DESCRIPTION	I²C ADRESSE ESCLAVE						
		A6	A5	A4	A3	A2	A1	A0
-	General call address	0	0	0	0	0	0	0
-	Reserved addresses	0	0	0	0	X	X	X
-	Reserved addresses	1	1	1	1	X	X	X
CCR921	RDS/RBDS decoder	0	0	1	0	0	A	A
NE5751	Audio processor for RF communication	1	0	0	0	0	0	A
PCA1070	Programmable speech transmission IC	0	1	0	0	0	0	A
PCA8516	Stand-alone 0SD IC	1	0	1	1	1	0	1
PCA8581/C	128 x 8-bit EEPROM	1	0	1	0	A	A	A
PCB2421	1K dual mode serial EEPROM	1	0	1	0	0	0	0
PCD3311C	DTMF/modem/musical tone generator	0	1	0	0	1	0	A
PCD3312C	DTMF/modern/musicai tone generator	0	1	0	0	1	0	A
PCD444O	Voice scrambler/descrambler for mobile telephones	1	1	0	1	1	1	A
PCD5OO2	Pager decoder	0	1	0	0	1	1	1
PCD5096	Universal codec	0	0	1	1	0	A	A
PCE84C467/8	8-bit CMOS auto-sync monitor controller	0	1	1	0	0	1	1
PCE84C882	8-bit microcontroller for monitor applications	0	1	1	0	0	1	1
PCE84C886	8-bit microconfroller for monitor applications	0	1	1	0	0	1	1
PCF2116	LCD controller/driver	0	1	1	1	0	1	A
PCF8522/4	512 x 8-bit CMOS EEPROM	1	0	1	0	A	A	A
PCF8566	96-segment LCD driver 1:1 - 1:4 Mux rates	0	1	1	1	1	1	A
PCF8568	LCD rom driver for dot matrix displays	0	1	1	1	1	0	A
PCF8569	LCD column driver for dot matrix displays	0	1	1	1	1	0	A
PCF8570	256 x 8-bit static RAM	1	0	1	0	A	A	A
PCF8573	Clock/calendar	1	1	0	1	0	A	A
PCF8574	8-bit remote I/O port (I²C-bus to parallel converter)	0	1	0	0	A	A	A
PCF8574A	8-bit remote I/O port (I²C-bus to parallel converter)	0	1	1	1	A	A	A
PCF8576C	16-segment LCD driver 1:1 - 1:4 Mux rates	0	1	1	1	0	0	A
PCF8577C	32/64-segment LCD display driver	0	1	1	1	0	1	0
PCF8578/9	Row/column LCD dot matrix driver/display	0	1	1	1	1	0	A
PCF8582/A	256 x 8-bit EEPROM	1	0	1	0	A	A	A
PCF8583	256 x 8-bit RAM/clock/calendar	1	0	1	0	0	0	A
PCF8584	I²C-bus controller	x	x	x	x	x	x	x
PCF8591	4-channel, 8-bit Mux ADC and one DAC	1	0	0	1	A	A	A
PCF8593	Low-power clock calender	1	0	1	0	0	0	1
PCX8594	512x8-bit CMOS EEPROM	1	0	1	0	A	A	P
PCX8598	1024 x 8-bit CMOS EEPROM	1	0	1	0	A	P	P
PDIUSB11	Universal serial bus	0	0	1	1	0	1	1
SAA1O64	4-digit LED driver	0	1	1	1	0	A	A
SAA1300	Tuner switch circuit	0	1	0	0	0	A	A
SAA1770	D2MAC decoder for satellite and cable TV	0	0	1	1	1	1	A
SAA2502	MPEG audio source decoder	0	0	1	1	1	0	1
SAA2510	Video-CD MPEG-audio/video decoder	0	0	1	1	0	1	A
SAA2530	ADR/DMX digital receiver	0	0	0	1	1	A	A
SAA4700/T	VPS dataline processor	0	0	1	0	0	0	A
SAA5233	Dual standard PDC decoder	0	0	1	0	0	0	A
SAA5243	Computer controlled teletext circuit	0	0	1	0	0	0	1
SAA5244	Integrated VIP and teletext	0	0	1	0	0	0	1
SAA5245	525-fine teletext decoder/controller	0	0	1	0	0	0	1
SAA5246A	Integrated VIP and teletext	0	0	1	0	0	0	1
SAA5249	VIP and teletext controller	0	0	1	0	0	0	1
SAA5252	Line 21 decoder	0	0	1	0	1	0	0
SAA5301	MOJI processor for Japan/China	0	1	1	0	0	0	0
SAA6750	MPEG2 encoder for Desk Top Video (=SAA7137>	0	1	0	0	0	0	0
SAA7110A	Digital multistandard decoder	1	0	0	1	1	1	A
SAA7140B	High performance video scaler	0	1	1	1	0	0	A
SAA7151B	8-bit digital multistandard TV decoder	1	0	0	0	1	A	1
SAA7165	Video enhancement D/A processor	1	0	1	1	1	1	1
SAA7186	Digital video scaler	1	0	1	1	1	A	0
SAA7l 91B	Digital multistandard TV decoder	1	0	0	0	1	A	1
SAA7192	Digital colour space-converter	1	1	1	0	0	0	A
SAA7199B	Digital multistandard encoder	1	0	1	1	0	0	A
SAA7370	CD-decoder plus digital servo processor	0	0	1	1	0	0	A

SAA9056	Digital SCAM colour decoder	1	0	0	0	1	A	1
SAA9065	Video enhancement and D/A processor	1	0	1	1	1	1	1
SAB9075H	PIP controller for NTSC	0	0	1	0	1	1	A
SAF1135	Dataline 16 decoder for VPS (call array)	0	0	1	0	0	A	A
TDA1551Q	2 x 22W BTL audio power amplifier	1	1	0	1	1	0	0
TDA4670/1/2	Picture signal improvement (PSI) circuit	1	0	0	0	1	0	0
TDA4680/5/7/8	Video processor	1	0	0	0	1	0	0
TDA4780	Video control with gamma control	1	0	0	0	1	0	0
TDA4845	Vector processor for TV-pictures tubes	1	1	0	1	1	A	A
TDA4885	150 MHz video controller	1	0	0	0	1	0	0
TDA8043	QPSK demodulator and decoder	1	1	0	1	0	0	A
TDA8045	QAM-64 demodulator	0	0	0	1	1	A	A
TDA4853/4	{Autosync detlection processor	1	0	0	0	1	1	0
TDA8366	Multistandard one-chip video processor	1	0	0	0	1	0	1
TDA8373	NTSC one-chip video processor	1	0	0	0	1	0	1
TDA8374	Multistandard one-chip vîdeo processor	1	0	0	0	1	0	1
TDA8375/A	Multistandard one-chip video processor	1	0	0	0	1	0	1
TDA8376/A	Multistandard one-chip video processor	1	0	0	0	1	0	1
TDA8415	TV/VCR stereo/dual sound processor	1	0	0	0	0	1	0
TDA8416	TV/VCR stereo/dual sound processor	1	0	1	1	0	1	0
TDA8417	TV/VCR atereo/dual aound processor	1	0	0	0	0	1	0
TDA8421	Audio processor	1	0	0	0	0	0	A
TDA8424/5/6	Audio processor	1	0	0	0	0	0	1
TDA8433	TV deflection processor	1	0	0	0	1	1	A
TDA8440	Video/audio switch	1	0	0	1	A	A	A
TDA8442	Interface for colour decoder	1	0	0	0	1	0	0
TDA8443A	YUV/RGB matrix switch	1	1	0	1	A	A	A
TDA8444	Octuple 6-bit DAC	0	1	0	0	A	A	A
TDA848OT	RGB gamma-correction processor	1	0	0	0	0	1	A
TDA8540	4 x 4 video switch matrîx	1	0	0	1	A	A	A
TDA8722	Negative video modulator with FM sound	1	1	0	0	1	0	0
TDA8735	150 MHz PLL frequency synthesiser	1	1	0	0	0	1	A
TDA8745	Satellite sound decoder	1	1	0	1	0	1	A
TDA8752	Triple fast ADC for LCD	1	0	0	1	1	A	A
TDA9141/3/4	Alignment-free multistandard decoder	1	0	0	0	1	A	1
TDA9150B	Deflection processor	1	0	0	0	1	1	0
TDA9151B	Programmable deflection processor	1	0	0	0	1	1	0
TDA9160	Multistandard decoder/sync. processor	1	0	0	0	1	A	1
TDA9161A	Bus-controlled decoder/sync. Processor	1	0	0	0	1	0	1
TDA9162	Multistandard decoder/sync. processor	1	0	0	0	1	A	1
TDA917O	YUV processor with picture improvement	1	1	0	1	0	0	A
TDA9177	2nd address for LTI (1st is '40')	1	1	1	0	0	0	0
TDA9178	YUV transient improvement processor	0	1	0	0	0	0	0
TDA9610	Audio FM processor for VHS	1	0	1	1	1	0	0
TDA9614H	Audio processor for VHS	1	0	1	1	1	0	0
TDA9840	TV stereo/dual sound processor	1	0	0	0	0	1	0
TDA9850	BTSC stereo/SAP decoder	1	0	1	1	0	1	A
TDA9852	BTSC stereo/SAP decoder	1	0	1	1	0	1	1
TDA9855	BTSC stereo/SAP decoder	1	0	1	1	0	1	A
TDA9860	Hi-fi audio processor	1	0	0	0	0	0	A
TEA6100	FM/IF for computer-controlled radio	1	1	0	0	0	0	1
TEA6300	Sound fader control and preamplifier/source selector	1	0	0	0	0	0	0
TEA6320/1/2/3	Sound fader control circuit	1	0	0	0	0	0	0
TEA6330	Tone/volume controller	1	0	0	0	0	0	0
TEA6360	5-band equalizer	1	0	0	0	1	1	A
TEA6821/2	Car radio AM	1	1	0	0	0	1	0
TEA6824T	Car radio IF IC	1	1	0	0	0	1	0
TSA5511/2/4	1.3GHz PLL frequency synthesizer for TV	1	1	0	0	0	A	A
TSA5522/3M	1.4GHz PLL trequency synthesizer for TV	1	1	0	0	0	A	A
TSA6O57	Radio tuning PLL trequency synthesizer	1	1	0	0	0	1	A
TSA6O60	Radio tuning PLL trequency synthesizer	1	1	0	0	0	1	A
UMA1000T	Data processor for mobile telephones	1	1	0	1	1	A	A
UMA1014	Frequency synthesizer for mobile telephones	1	1	0	0	0	1	A

X = Don't care, A = Programmable address bit, P = Page selection bit.

Partant du principe qu'il est toujours bon de laisser des portes de secours à un système, on se rend vite compte que souvent dans un même système on est appelé à utiliser plusieurs fois le même type de circuit (exemple : RAM, ...). Il devient alors nécessaire d'avoir des adresses différentes pour des circuits de même type afin de pouvoir accéder particulièrement à l'un d'entre eux. C'est notamment dans ce but qu'il a été décidé qu'une partie des sept bits d'adresses serait fixe (les 4, 5,... bits de poids forts) et que les autres seraient modifiables.

Remarquez bien que rien n'est dit sur la manière de les modifier. Ces bits peuvent être modifiables, par exemple, soit par hardware (logique ou analogique) soit par logiciel soit... par n'importe quoi. Le nombre de ces bits variables dépend en fait plus du nombre de broches disponibles sur le boîtier du circuit que d'autre chose.
Par exemple, un circuit qui aurait quatre bits fixes d'adresses et trois bits configurables permettrait à un concepteur de disposer sur un même bus sept petits frères d'un même circuit.

Remarques :

- Avec sept bits d'adresse, on peut adresser 128 circuits dont les adresses sont différentes avec ce protocole.
- Avec trois bits programmables on peut disposer de huit circuits d'un même type sur le bus. Mais il existe de nombreuse astuces pour en mettre bien plus, par exemple en multiplexant temporellement ces fameux bits configurables. Certains schémas simples permettent d'adresser plus de 300 circuits d'un même type.

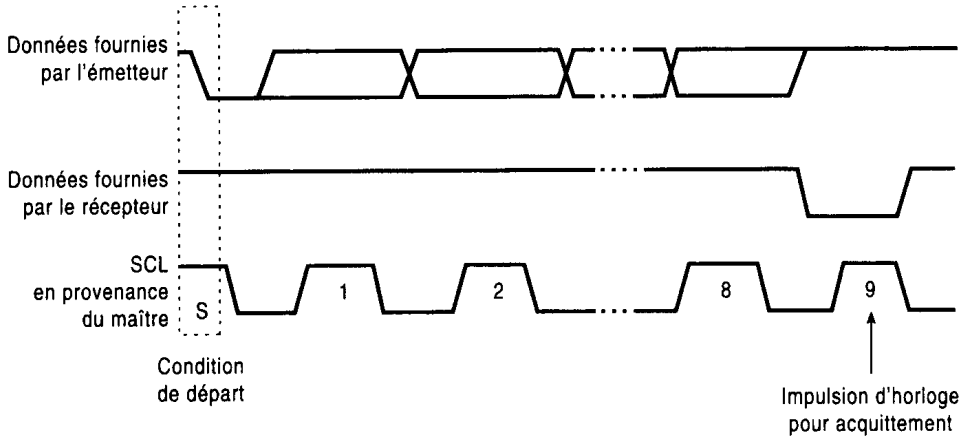
Il existe aussi dans ce premier octet transmis des valeurs bien définies correspondant à des fonctionnalités particulières ayant trait aux protocoles du multimaître, à des adresses concernant l'appel général, au mot de start.

Les mots suivant l'adresse n'ont pas de signification particulière. Ils sont codés sur 8 bits.
On peut résumer globalement leurs fonctions en disant qu'ils transportent des données.
Très souvent les données qui sont présentes dans les mots qui suivent immédiatement l'adresse ont un sens plus orienté vers l'organisation interne du circuit commandé (mot de sous-adresse, de statuts, de commande ...), mais il n'y a pas de règles générales et ces agréments sont définis types à types de circuits intégrés.
Puis viennent ensuite les mots qui contiennent des données, des valeurs au sens strict.

Le nombre de mots transmis n'est pas (sur son principe) limité. Il dépend, d'une part, du type de circuit auquel on s'adresse et, d'autre part, du fait qu'en pratique on est bien obligé de se limiter car si l'on ne faisait que transmettre des données à un récepteur particulier on ne ferait rien d'autre...

2.8. Acquittement.

Comme vous pouvez le remarquer sur la figure suivante, entre chaque mot transmis il y a une procédure d'acquittement.
Pour bien comprendre cette fonction et la puissance qu'elle possède, il est nécessaire de bien analyser son principe.



Conditions de fonctionnement :

- Le transfert de données avec acquittement est obligatoire.
- Un acquittement doit avoir lieu après chaque mot transmis.
- L'impulsion d'horloge (la neuvième) pendant laquelle le composant récepteur devra renvoyer à l'émetteur son acquittement est générée par le maître qui a lui-même le contrôle du bus.
- Pendant le neuvième coup d'horloge le composant émetteur devra laisser libre sa ligne de données SDA préalablement relâchée à l'état haut.

Condition d'acquittement :
Ceci étant, pour qu'il y ait acquittement, il faut que le composant récepteur mette à l'état bas la ligne de données SDA pendant que la ligne d'horloge SCL est à l'état haut et dite stable.

Non-acquittement :
Dans tous les autres cas de figures de non-acquittement il faut définir des procédures d'actions bien répertoriées.

Voici deux exemples précis de non-acquittement :

- Le récepteur étant en train de terminer une tâche réalisée en temps réel (écriture d'une EEPROM par exemple) ne peut ou ne veut accepter la donnée qui se présente et, pour le signifier à l'émetteur, fait exprès de laisser la ligne de données SDA telle que l'on vient de la lui présenter, c'est-à-dire à l'état haut. Le maître, comprenant ce refus (non-acquittement), a tout loisir de générer une condition de stop pour abandonner ce récepteur récalcitrant et en profiter pour s'occuper d'un autre à l'accueil plus sympathique. Évidemment ces petites fâcheries n'étant que de courtes durées, il pourra essayer à nouveau de temps en temps de savoir dans quelles dispositions se trouve ce récepteur pour continuer ou reprendre le transfert.

- Deuxième cas de figure plus délicat, mais d'usage fréquent. Le maître (celui qui génère l'horloge) est récepteur (il vient de demander par exemple à une mémoire de lui faire parvenir son contenu). De part son travail spécifique (gestion d'une interruption...) il se peut que le maître veuille faire comprendre à l'esclave émetteur qu'il désire arrêter le transfert. Pour cela, volontairement, en n'acquittant pas le dernier mot transmis par l'esclave il signifie à ce dernier de bien vouloir relâcher la ligne de données SDA afin que le maître puisse générer une condition de stop.

2.9. Exemple : circuit d'affichage à LED.

Le circuit SAA1064 commande quatre digits (sept segments + point décimal) à LED. Il est simple d'emploi, sa fonction étant principalement passive car sa fonction réside, dans 99% des cas, en un esclave récepteur où l'on ne fera qu'écrire.
Pour le 1% restant, un circuit d'affichage peut avoir, sous certaines conditions, un soupçon d'intelligence. C'est peut-être beaucoup dire, mais il peut avoir été conçu de façon prévenir lorsque par hasard quelqu'un a sournoisement coupé son alimentation et indiquer de ce fait (en devenant un esclave émetteur) qu'il ne détient plus aucune information valable à afficher.

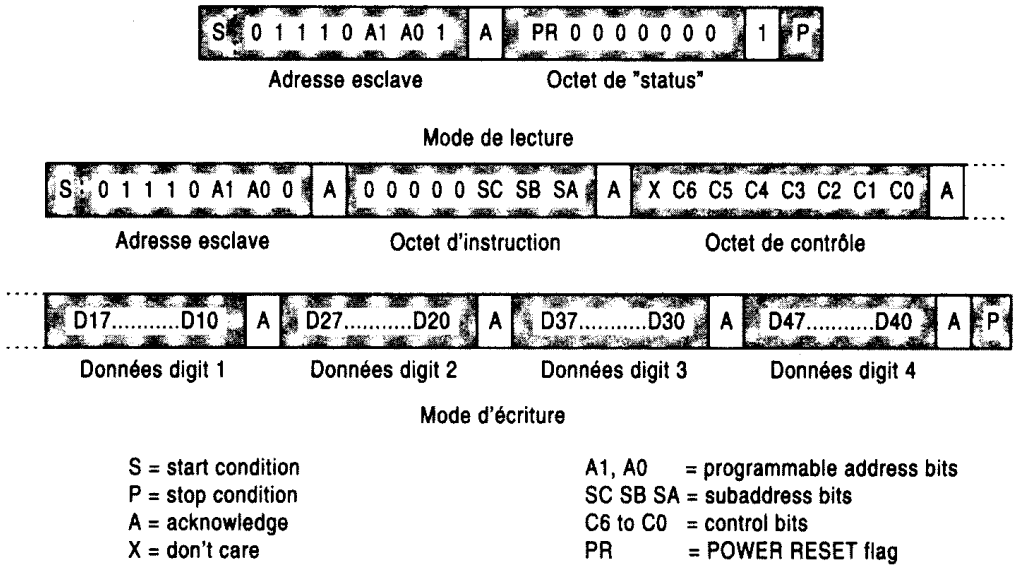
La famille des SAA1O64 (car ils peuvent être plusieurs dans un montage) répond au nom charmant de : 01110 XX (.).
Son adresse comporte trois parties :

- 01110 partie fixe de la famille générique SÀA 1064
- XX partie configurable choisi par exemple égale à 10
- (.) bit représentant R/W (R=1; W=0). Dans ce cas d'écriture, ce bit sera 0.

on a donc:

adresse spécifique :	0111 010.	
+ condition d'écriture :	+ 0	
octet à transmettre :	0111 0100	soit 76 (en hexadécimal)

La spécification du circuit SAA1064 (voir figure suivante) indique que les octets suivants représentent d'abord un mot d'instruction puis un mot de commande.



Le mot d'instruction correspond, pour ce circuit, à une sous-adresse. Pour simplifier, ce mot est égal à 0.
Le mot de commande décrit l'aspect visuel souhaité par l'utilisateur. Par exemple, dans le cas d'un affichage dynamique (multiplexé) et au courant maximal circulant dans les afficheurs, ce mot doit avoir pour valeur X1110111 soit 77 en hexadécimal.
Les quatre mots suivants représentent le contenu des valeurs que chaque digit doit afficher.
Le contenu de la séquence de transmission sera donc le suivant :