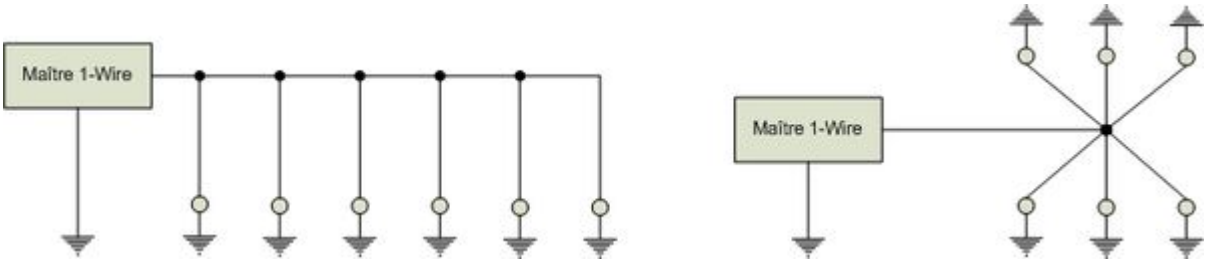
Le BUS 1WIRE.

Présentation

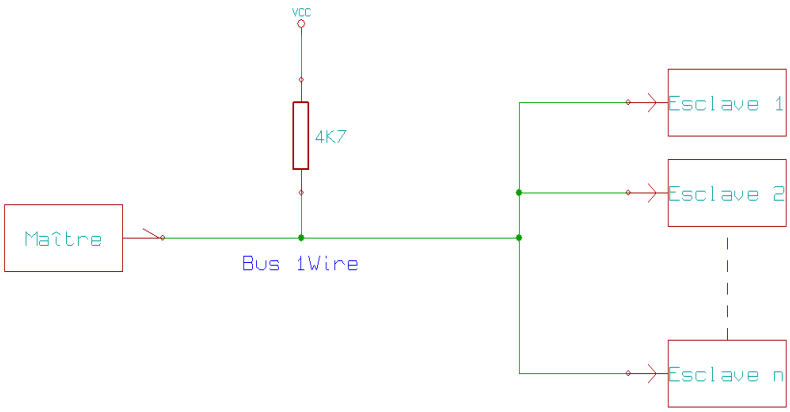
Le bus 1-Wire (ou OneWire) est un bus conçu par Dallas Semiconductor, permet de connecter et de faire dialoguer entre eux des circuits sur un seul fil.
Ce système de bus utilise un seul maître, qui pourra dialoguer avec un ou plusieurs esclaves.
Ce bus supporte une topologie série, parallèle ou en étoile, et fonctionne suivant le principe maître / esclave.



Lors de la conception de grands réseaux, il est important de bien réfléchir à la topologie de celui-çi, sinon vous risquez d'avoir des problèmes entre les temporisations et les réflexions et d'avoir alors des pertes de données.

Pour les très petits réseaux, il est possible de choisir une topologie en étoile. Chaque périphérique esclave est alors relié à un point central avec son propre câble. Ce point central est alors relié au maître avec un câble séparé.

Pour les réseaux un peu plus étoffés, il est recommandé d'utiliser un bus cablé avec du câble de bonne section. Chaque périphérique esclave est alors raccordé à ce bus par un câble de maximum 5cm. Cette méthode de raccordement minimise les erreurs dues aux réflexions. Malgré cette méthode, si vous utilisez plus de 10-15 périphériques esclaves, il est possible que des problèmes apparaissent à cause d'une certaine surcharge du bus de données. Il est alors possible d'insérer une résistance de 100 ou 120 ohms en série dans la connection de chaque esclave sur le bus de données.



Le niveau de tension utilisé sur ce bus est +5V (niveau TTL).

Toutes les commandes et données sont envoyées avec le bit LSB en tête.

Le fil unique du bus doit être tiré au +Vcc par une résistance de 4,7KΩ.

L'état repos du bus est donc un état haut.

L'avantage de ce bus est qu'il peut être utilisé en mode "parasite" (alimentation à partir du fil de données). Cela permet d'utiliser seulement 2 fils (et non un seul comme le nom le laisse supposer), un fil de données et un fil de masse.

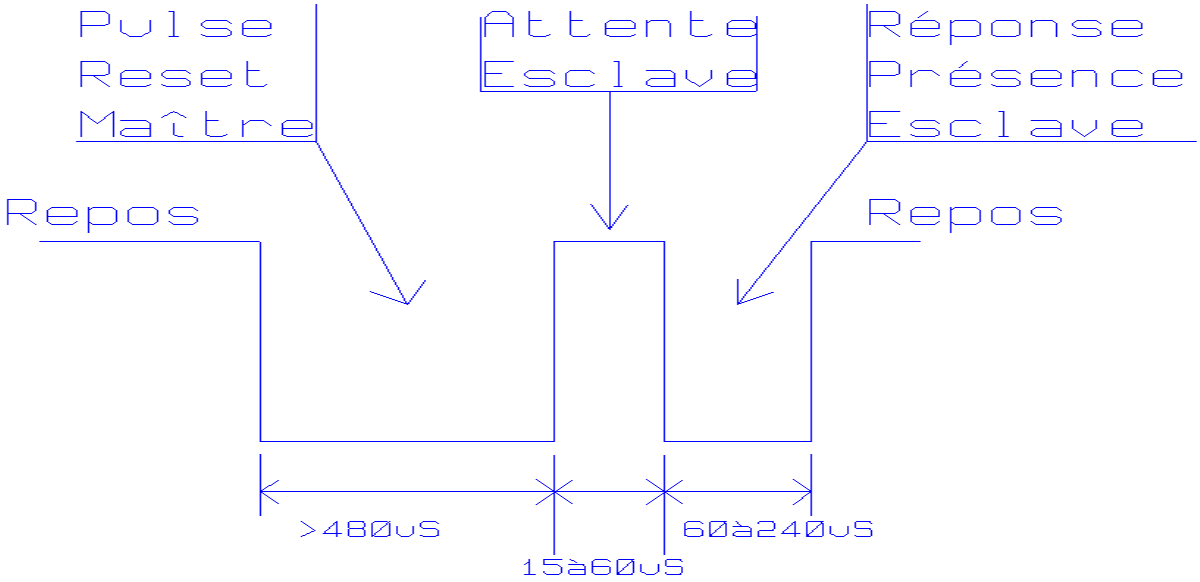
Généralement utilisé pour des mesures de températures, il existe une gamme complète de composants compatibles.
Chaque circuit possède une adresse physique unique, gravée dans la puce à la fabrication. C'est une des raisons expliquant la réticence de Dallas à permettre la création d'esclave 1-wire autres que ceux qu'il produit : avoir la maîtrise de l'identification des esclaves du bus.

Protocole

La communication sur le bus 1-wire est caractérisée par un ensemble de pulse "changement d'état du bus".

Sachant que l'état par défaut de la ligne data est +5V, ce qui permet d'alimenter les différents composants à partir de la ligne data en mode parasite.

Avant toute communication, le maître met le bus à 0 pendant 480µs. C'est le pulse d'initialisation ou de Reset. Après un délai de 15 à 60 µs, le ou les esclaves raccordés, forcent le bus à l'état bas pendant 60 à 240 µs pour signaler leur présence.



Le maître reçoit alors la liste des esclaves connectés sur le bus. Il pourra utiliser une commande particulière pour sélectionner l'esclave avec lequel il souhaite communiquer (commande ROM)

Pour émettre un bit sur le bus le maître force le bus à "0" pendant 1 à 15us, pour indiquer qu'il souhaite envoyer un bit de données, puis il doit positionner le bus suivant le bit qui doit être émis (0 ou 1) l'esclave "actif" va lire le bus entre 15us et 45us après la détection du front descendant initial.

La durée total d'un bit est donc de 60us, ce qui donne une vitesse de communication maximale de 16kbits/s.

Si le bus est maintenu à l'état bas plus de 480 µs par le maître, tous les composants sur le bus sont remis à zéro.

Chaque circuit possède une adresse physique unique, gravée dans la puce à la fabrication.
Cette adresse est constituée de 64 bits soit 8 octets. Le premier octet détermine le type de famille auquel appartient le circuit. Les 6 octets suivants, constituent le code propre du circuit. Le dernier octet est le CRC. C'est un octet de contrôle calculé à partir des 56 bits précédents.

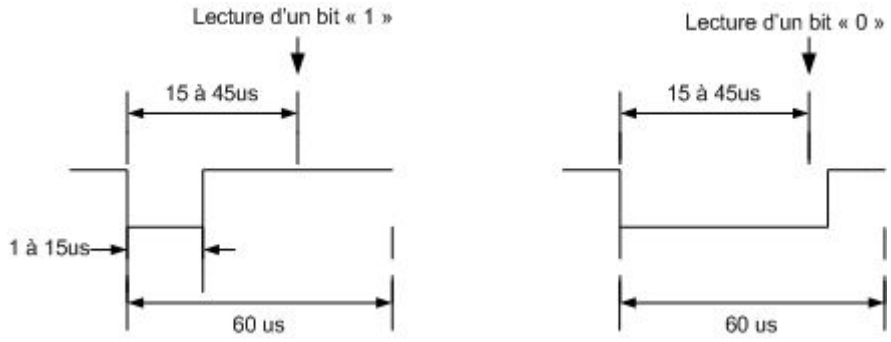
Toute transaction entre un maître et un ou plusieurs esclaves, débute par une initialisation, constituée par l'envoi du pulse de Reset par le maître.

Le maître doit ensuite envoyer une commande de type ROM qui est propre au protocole 1Wire, et que tous les circuits de ce type vont reconnaître. Cela va permettre entre autre de sélectionner un circuit parmi les différents esclaves qui ont répondu présents au pulse de Reset.
Le dialogue et l'échange de données pourra ensuite commencer, entre le maître et l'esclave sélectionné.

Emission d'un bit du maître vers l'esclave:

Le maître force le bus à "0" pendant 1 à 15 µs. L'esclave va lire le bus entre 15 et 45 µs après le front descendant (valeur typique 30 µs).

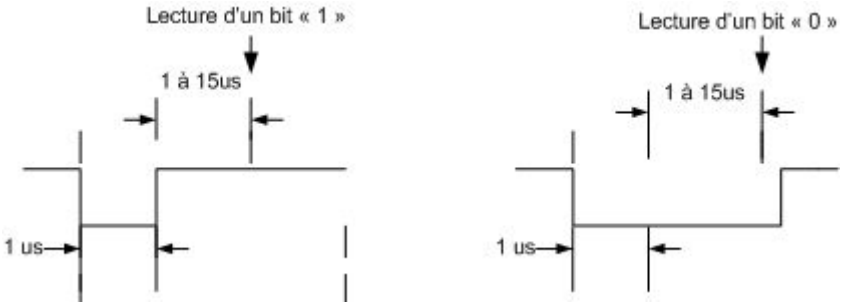
Si on veut émettre un "1", il faut repasser le bus à "1" immédiatement, et ne plus rien faire jusqu'à t = 60 µs.
Pour émettre un "0" il faut laisser le bus à "0" jusqu'à t = 60 µs, puis repasser le bus à "1".
La durée du bit est donc de 60 µs, ce qui donne un débit de 16 kbits/sec.



Réception d'un bit par le maître:

Le maître force le bus à "0" pendant au moins 1 µs. Si l'esclave veut émettre un "1", il laisse le bus libre donc tiré à "1". Pour émettre un "0", l'esclave doit tirer le bus à "0" pendant 15 µs au minimum.

Le maître devra donc dans tous les cas lire le bus 15 µs maximum après avoir tiré le bus à "0" pendant 1 µs.
L'état du bus donnera alors le bit transmis par l'esclave.



COMMANDES ROM:

Ces commandes sont constituées d'un octet que le maître devra envoyer après avoir fait un reset.

$33h = READ ROM

Cette commande ne peut être utilisée que s'il n'y a qu'un seul esclave sur le bus. Celui-ci répond alors ces 64 bits de code.

$55h = MATCH ROM

Cette commande suivi de 64 bits de code, va permettre au maître de sélectionner un esclave particulier.

$CCh = SKIP ROM

Commande d'appel général, pour adresser tous les esclaves. Cette fonction est utile pour adresser un esclave qui est seul sur le bus, sans avoir à envoyer les 64 bits de son code.

$F0h = SEARCH ROM

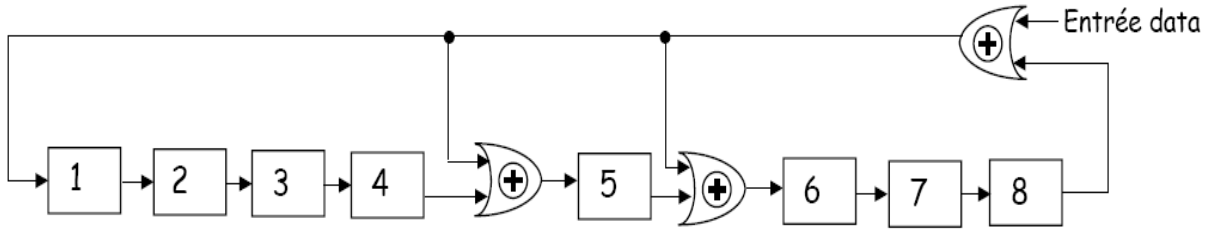
Cette commande va permettre de rechercher bit à bit, les codes de tous les esclaves étant raccordés au bus 1Wire.
En réponse à cette commande, les esclaves envoient leur premier bit, puis ce même bit inversé. Le maître émet à son tour ce premier bit. Les esclaves qui reconnaissent leur 1er bit restent à l'écoute, et les autres s'éliminent et ne répondront plus. Les esclaves toujours présents vont maintenant envoyer leur 2eme bit, puis ce même 2eme bit mais inversé. Le maître comme précédemment va émettre ce 2eme bit. Les esclaves qui ne reconnaissent pas leur 2eme bit vont s'éliminer.
Quand le maître reçoit le bit et son inverse à : 1 1 c'est qu'il n'y a pas de circuit sur le bus 1Wire.
Quand le maître reçoit le bit et son inverse à : 0 0 c'est qu'il y a conflit, car des esclaves ont un "1" et des autres un "0" à cette position. Dans ce cas, il enverra en réponse un bit à "0" pour ne garder que les circuits ayant un "0" à cette position et éliminer ceux qui ont un "1".
Quand le maître reçoit le bit et son inverse à : 0 1 c'est qu'il n'y a que des circuits ayant un bit à "0" à cette position. Il enverra un "0" pour garder tous ces circuits. Et s'il a reçu 1 0 il enverra "1", car le bit de cette position est à "1" et on gardera les circuits.
Le principe général de la recherche est de désélectionner les uns après les autres les circuits à chaque conflit sur les différentes positions des bits.
A la fin de chaque étape de recherche, le maître connaît un nouveau code de 64 bits complet d'un circuit. L'étape suivante est identique jusqu'au niveau de la dernière décision après le conflit. Le maître part alors dans la direction opposée, il enverra un "1" alors qu'il n'avait gardé que les circuits ayant un "0" à cette position. Ainsi, bit par bit, on va arriver à lire les 64 bits de tous les esclaves. Le maître va ainsi savoir combien il y a d'esclaves sur le bus et quelles sont leurs codes propres.

$ECh = CONDITIONAL SEARCH

Cette commande fonctionne comme la commande SEARCH ROM, à la différence que seul les circuits ayant une condition bien spécifiée participent à la recherche. Par exemple les circuits de mesure de la température qui ont le Flag d'alarme actif ou les port E/S qui ont leur sortie à "1".

Le CRC:

Le polynôme générateur du CRC est : X^8 + X^5 + X^4 +1
Sa représentation est la suivante:



Calcul pratique du CRC:

Quand on aura reçu les 56 premiers bits du circuit, soit 7 octets, on devra calculer l'octet de CRC pour le comparer à celui que l'on va recevoir avec les 8 bits restants. Pour éviter les calculs complexes du polynôme, on va utiliser une table indexée de 256 valeurs décimales.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16
0	0	94	188	226	97	63	221	131	194	156	126	32	163	253	31	65
16	157	195	33	127	252	162	64	30	95	1	227	189	62	96	130	220
32	35	125	159	193	66	28	254	160	225	191	93	3	128	222	60	98
48	190	224	4	92	223	129	99	61	124	34	192	158	29	67	161	255
64	70	24	250	164	39	121	155	197	132	218	56	102	229	187	89	7
80	219	133	103	57	16	228	6	88	25	71	165	251	120	38	196	154
96	101	59	217	135	4	90	184	230	167	249	27	69	198	152	122	36
112	248	166	68	26	153	199	37	123	58	100	134	216	91	5	231	185
128	140	210	48	110	237	179	81	15	78	16	242	172	47	113	147	205
144	17	79	173	243	112	46	204	146	211	141	111	49	178	236	14	80
160	175	241	19	77	206	144	114	44	109	51	209	143	12	82	176	238
176	50	108	142	208	83	13	239	177	240	174	76	18	145	207	45	115
192	202	148	118	40	171	245	23	73	8	86	180	234	105	55	213	139
208	87	9	235	181	54	104	138	212	149	203	41	119	244	170	72	22
224	233	183	85	11	136	214	52	106	43	117	151	201	74	20	246	168
240	116	42	200	150	21	75	169	247	182	232	10	84	215	137	107	53

Exemple: On vient de recevoir les 7 octets suivants : $ **00 00 04 0C 38 F0 01** h

1. Le CRC que l'on va calculer est initialisé au départ à la valeur $00h.
2. Nous partons du premier octet : $01h. Le XOR est réalisé entre le CRC et l'octet soit : $00h XOR $01h. Le résultat est $01h soit 1. Cette valeur est l'index de la table qui va donner le CRC . A l'index 01 de la table la valeur lue est 94 soit $5Eh. Donc CRC = $5Eh.
3. Maintenant un XOR est réalisé entre le nouveau CRC et le 2eme octet. $5Eh XOR $F0h donne $AEh soit 174 en décimal. Cette valeur d'index nous donne dans la table 176 soit $B0h.
4. Le XOR avec le 3eme octet donne : $B0h XOR $38h = $88h soit 136 qui nous donne 78 ou $4Eh.
5. Le XOR avec le 4eme octet donne : $4Eh XOR $0Ch = $42h soit 66 qui donne 250 ou $FAh.
6. Le XOR avec le 5eme octet donne : $FAh XOR $04h = $FEh soit 254 qui donne 107 ou $6Bh.
7. Le XOR avec le 6eme octet donne : $6Bh XOR $00h = $6Bh soit 107 qui donne 69 ou $45h.
8. Le XOR avec le 7eme octet donne : $45h XOR $00h = $45h soit 69 121 ou $79h.
9. Le CRC des 7 octets reçu est donc $79h qu'il faudra comparer au 8ème octet que l'on va recevoir.

CAPTEUR de TEMPERATURE DS18B20

Ces circuits possèdent un code unique sur 64 bits comme tous les circuits 1Wire.
Le code famille est $28h, suivi de 6 octets propre au circuit et d'un octet de CRC.

La détection de présence de ce circuit se fait en envoyant le pulse de Reset, qui est un état bas pendant au moins 480 µs. Quand un circuit DS 18B20 est présent sur le bus, il le signale en maintenant le bus à l'état bas pendant 60 à 240 µs.

Toute transaction avec un tel circuit doit démarrer par un pulse de Reset suivi de l'envoi d'une commande ROM. On pourra après envoyer une commande de fonction propre à ce type de circuit.
Si le circuit est seul sur le bus 1Wire, la commande **ROM** peut être l'appel général **SKIP ROM = $CCh**.
Si ce n'est pas le cas, il faudra connaître les 64 bits propre au circuit que l'on veut atteindre et utiliser la commande **MATCH ROM = $55h** suivi des 8 octets du code.
Une recherche préalable des 8 octets de code sera faite par la commande **READ ROM = $33h** si le circuit est seul ou bien par **SEARCH ROM = $F0h** s'il y a plusieurs circuits sur le bus.

MEMOIRE INTERNE:
Elle est constituée d'une zone RAM de 9 octets et d'une zone EEPROM non volatile de 3 octets.

	RAM	EEPROM
Octet 1	TEMPERATURE LSB	
Octet 2	TEMPERATURE MSB	
Octet 3	Alarme seuil HAUT ou octet 1	Alarme seuil HAUT ou octet 1
Octet 4	Alarme seuil BAS ou octet 2	Alarme seuil BAS ou octet 2
Octet 5	Registre de CONFIGURATION	Registre de CONFIGURATION
Octet 6	Réservé	
Octet 7	Réservé	
Octet 8	Réservé	
Octet 9	CRC	

OCTET 1 et 2 : TEMPERATURE LSB et MSB
La température est donnée sur 16 bits en complément à 2 entre –55 °C et + 125 °C.
S = Signe de la température. Ce bit est à "1" si elle est négative et à "0" si elle est positive.
Si le signe est positif (bit S=0) la valeur absolue de la température sera donnée par les bits significatifs de LSB et MSB .
Par contre si la température est négative (bit S=1), la valeur absolue sera obtenue en complémentant la valeur des bits significatifs de LSB et MSB et en ajoutant 1 au résultat.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LSB	2^3=8	2^2=4	2^1=2	2^0=1	2^-1=0,5	2^-2=0,25	2^-3=0,125	2^-4=0,0625
	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
MSB	S	S	S	S	S	2^6=64	2^5=32	2^4=16

OCTET 5 : Registre de CONFIGURATION
Seuls deux bits sont significatifs dans ce registre: X1 et X0. Ces bits permettent de choisir la résolution.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Config	0	X1	X0	1	1	1	1	1

X1	X0	Résolution	Temps Conversion	Remarques
0	0	9 Bits	94 ms	Bits 0,1 et 2 de LSB non définis. Résolution = 0,5°C
0	1	10 Bits	188 ms	Bits 0 et 1 de LSB non définis. Résolution = 0,25°C
1	0	11 Bits	375 ms	Bit 0 non défini. Résolution = 0,125°C
1	1	12 Bits	750 ms	Résolution = 0,0625°C

CODES de COMMANDES :

Après avoir envoyé une commande ROM pour adresser un DS18B20 esclave, le maître doit envoyer un code de commande.

Début de conversion : $44h
Cette commande lance la conversion de température. Le résultat est rangé dans les 2 octets LSB et MSB .
Le temps de conversion dépend de la résolution choisie.
Le maître doit interroger le DS18B20 qui répond par un bit à "0" tant que la conversion n'est pas terminée.
Quand l'opération est terminée, l'esclave répond par un bit à "1".

Ecriture en RAM : $4Eh
Seuls les octets 3, 4 et 5 de la zone RAM peuvent être écrits. Il s'agit des octets: Alarme seuil haut, Alarme seuil bas et Configuration.
Le maître doit commencer par envoyer en premier le LSB de l'octet 3.
Tous les octets seront ensuite envoyés avec le LSB en tête.
Il doit impérativement envoyer les 3 octets, avant de faire un reset, pour que l'écriture soit éffective.

Lecture de la RAM : $BEh
Les 9 octets de la RAM sont envoyés vers le maître. L'esclave commence par le bit 0 du premier octet et transmet ainsi les 9 octets de sa RAM.
Le maître peut interrompre à tout moment la lecture en faisant un Reset.

Copie de la RAM en EEPROM : $48h
Copie des octets 3, 4 et 5 de la zone RAM dans la zone EEPROM pour sauvegarde en cas de coupure d'alimentation.

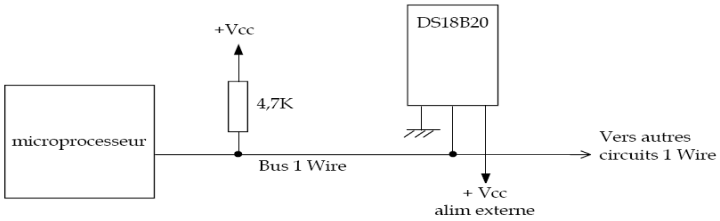
Recopie EEPROM en RAM : $B8h
Cette commande récupère en EEPROM les octets Alarme seuil haut, Alarme seuil bas et Configuration pour les placer en RAM dans les octets 3, 4 et 5.

Alimentation Parasite : $B4h
L'esclave répond à cette commande par un bit à "1" s'il fonctionne avec une alimentation extérieure, c'est-à-dire sur 3 fils. Et par le bit à "0" s'il est en mode d'alimentation parasite, sur 2 fils.

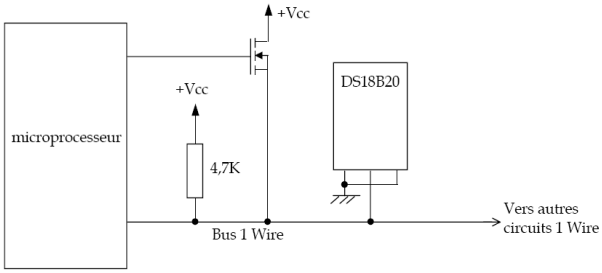
Alimentation du DS18B20 :

Il y a 2 méthodes pour alimenter ce circuit.

- Soit par une alimentation extérieure entre 3V et 5V :



- Soit par une alimentation parasite :



Quand le bus 1Wire est à l'état haut, une capacité interne se charge et sert de réservoir pour alimenter le circuit quand le bus est à l'état bas. Dans ce cas la broche Vcc du circuit est raccordée à la masse.

Seulement, quand le circuit est en mode conversion ou en mode écriture en EEPROM, le courant d'alimentation doit être d'au moins 1,5mA. Dans ce cas, l'alimentation parasite ne suffit plus.
Il faut pour cela mettre le bus 1Wire directement au +Vcc. C'est ce qui sera fait par la mise en conduction du transistor MOSFET.
Cette opération devra intervenir dans les 10 µs qui suivent l'envoi de la commande "Conversion Température" ou "Copie RAM en EEPROM".
Ensuite le bus devra rester au +Vcc pendant toute la durée de la conversion de température (750 ms) ou de l'écriture en EEPROM (10ms).

Aucune autre activité sur le bus 1Wire ne peut avoir lieu pendant ce temps là, à la différence du mode d'alimentation extérieure, où ce temps peut être mis à profit pour envoyer d'autres données sur le bus.

Exemple de dialogue:

- Reset du maître
- Maître envoie $CCh (Appel général) *
- Maître envoie $4Eh (Ecriture RAM)
- Maître envoie 3 octets (Seuils haut et bas + résolution)
- Reset du maître
- Maître envoie $CCh (Appel général) *
- Maître envoie $44h (Début Conversion)
- Le maître attend la réception d'un "1"
- Reset du maître
- Maître envoie $CCh (Appel général)
- Maître envoie $BEh (Lecture RAM)
- Esclave envoi 9 octets dont la Température

* La commande ROM **SKIP ROM = $CCh** doit être remplacée par la commande **MATCH ROM = $55h** suivie des 64 bits d'adresse, dans le cas ou plusieurs circuits sont sur le bus 1wire.

PORT 8 ENTREES/SORTIES DS 2408

Ces circuits possèdent un code unique sur 64 bits comme tous les circuits 1Wire. Le code famille est $29h, suivi de 6 octets propre au circuit et d'un octet de CRC.
La détection de présence de ce circuit se fait en envoyant le Pulse de Reset, qui est un état bas pendant au moins 480 µs. Quand un circuit DS 2405 est présent sur le bus, il le signale en maintenant le bus à l'état bas pendant 60 à 240 µs comme tout circuit répondant au protocole 1Wire.
Tout dialogue doit débuter par ce pulse de détection de présence ou Reset envoyé par le maître.
Le maître doit ensuite envoyer une commande ROM, qui pourra être suivie d'une commande de fonction.

Commandes de fonction

Lecture registres : $F0h
Cette commande doit être suivie de l'adresse du registre interne à atteindre. L'octet LSB de l'adresse est envoyé en premier suivi de l'octet MSB de cette adresse. Le maître lit alors l'octet pointé par l'adresse, et à chaque lecture, il lit l'octet suivant jusqu'à l'adresse $008Fh. Le maître peut envoyer à tout moment un Reset 1Wire pour sortir de cette commande.

Adresse	Type	REGISTRE
$0000 à $0087h	x	Non défini
$0088h	Read	Etat logique du PIO
$0089h	Read	Registre d'état de Latch du PIO
$008Ah	Read	Registre d'état d'activité du PIO
$008Bh	Read/Write	Masque de sélection pour recherche conditionnelle
$008Ch	Read/Write	Sélection de polarité pour recherche conditionnelle
$008Dh	Read/Write	Registre CONTROLE/STATUS
$008Eh à $008Fh	Read	Non définis. Toujours à $FFh

Lecture PIO : $F5h
A la différence de la commande de lecture des registres suivie de l'adresse $0088h, cette commande renvoie pendant 32 fois l'état du PIO suivi d'un octet de CRC, qui va permettre au maître de vérifier que la transmission s'est effectuée sans erreur. Cette commande peut être terminée à tout moment par l'envoi d'un Reset 1Wire.

Ecriture PIO : $5Ah
C'est la seule possibilité que l'on a, pour écrire dans le Latch de sortie du PIO. Le maître doit envoyer la DATA après cette commande, puis il doit envoyer la DATA complémentée. Si la transmission est bonne, le DS2408 répond par un bit à "0". Dans le cas contraire, ce bit est à "1" et un Reset 1Wire est nécessaire pour sortir de cette commande.
Si la transmission a été bonne, le DS2408 écrit la DATA reçue sur le PIO. Il envoie ensuite l'octet $AAh pour confirmer qu'il n'y a pas eu d'erreur et tout de suite après, il lit le PIO et renvoie la valeur, pour permettre au maître de vérifier ce qu'il a écrit. Cette commande peut être terminée à tout moment par un Reset 1Wire.

LECTURE du PIO :
- Reset 1WIRE
- $CCh : Envoi "appel général"
- $F0h : Envoi "Lecture PIO"
- $88h : Envoi LSB adresse PIO
- $00h : Envoi MSB adresse PIO
- Réception octet état PIO : $'data'h
- Reset 1WIRE

ECRITURE dans le PIO :
- Reset 1WIRE
- $CCh : Envoi "appel général"
- $5Ah : Envoi "Ecriture PIO"
- Envoi valeur à mettre dans PIO $'data'h
- Envoi valeur complémentée $'data'h
- Reset 1WIRE