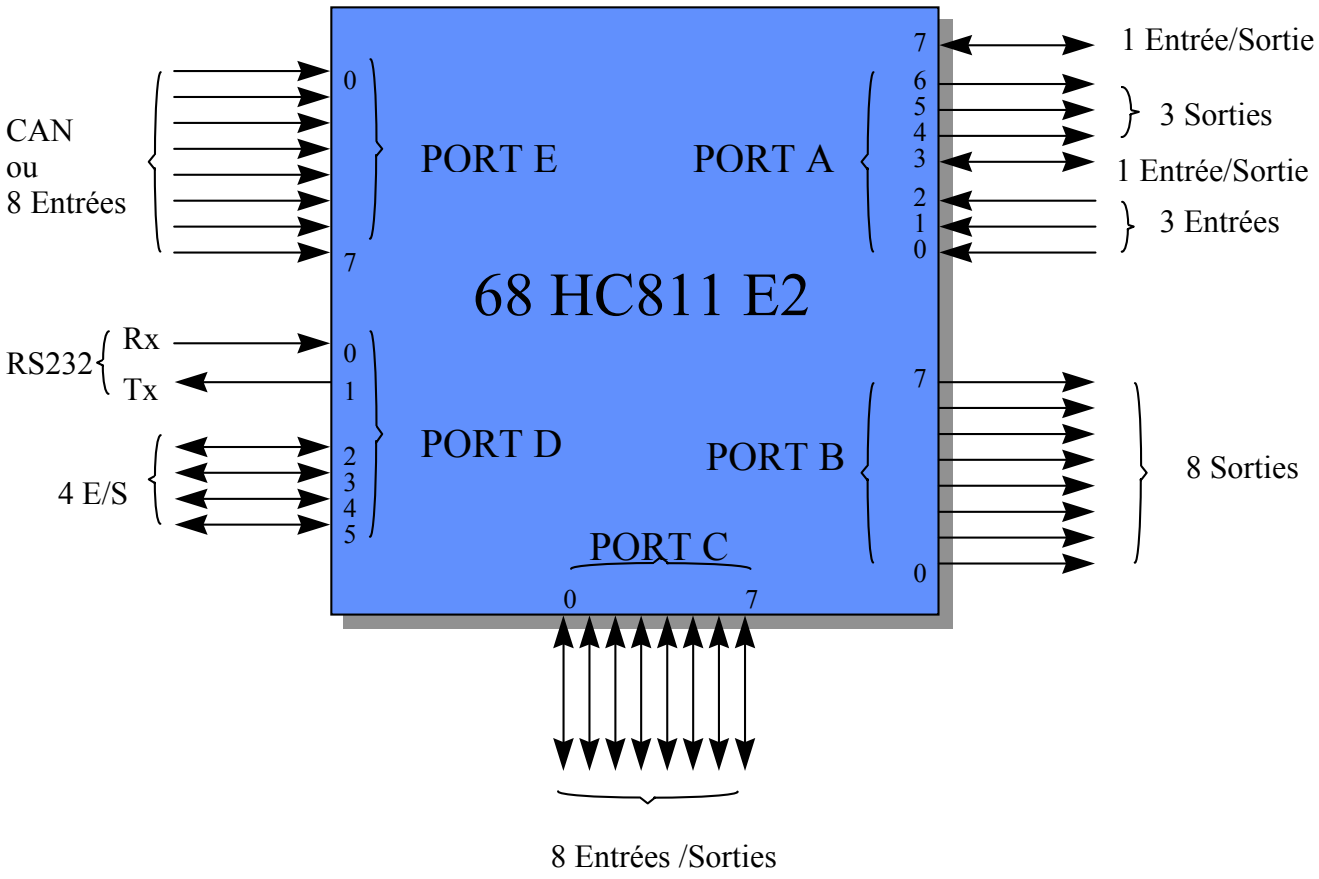
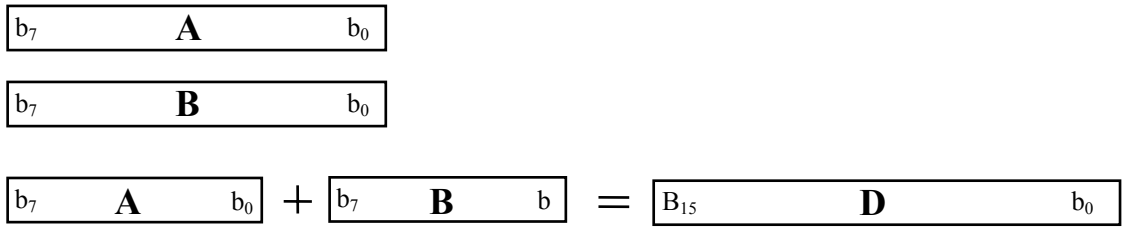
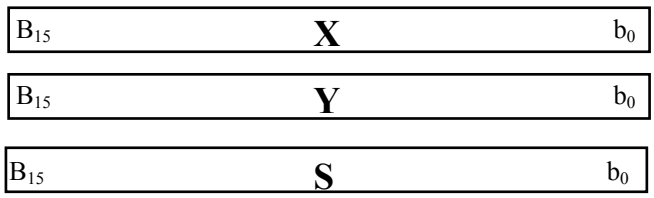
MICRO CONTROLEUR 68HC811

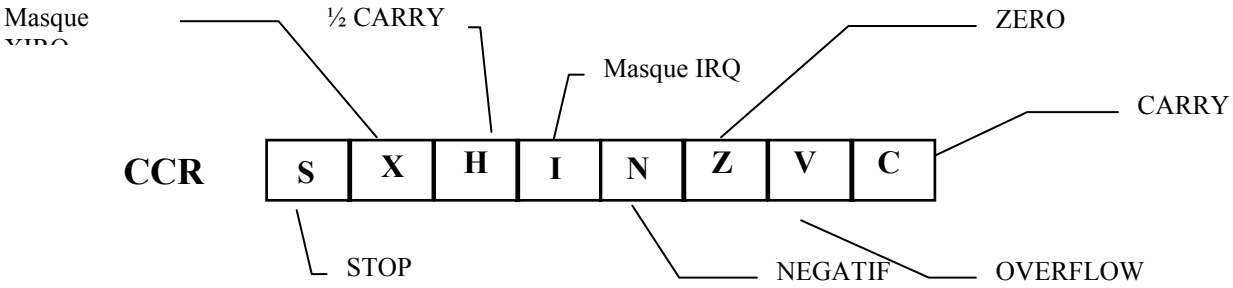


1. INTRODUCTION:

a. *Accumulateurs et Instructions*







INSTRUCTIONS SPECIALES MICROCONTROLEUR :
Ces instructions ne marchent qu'avec un adressage direct ou indexé.

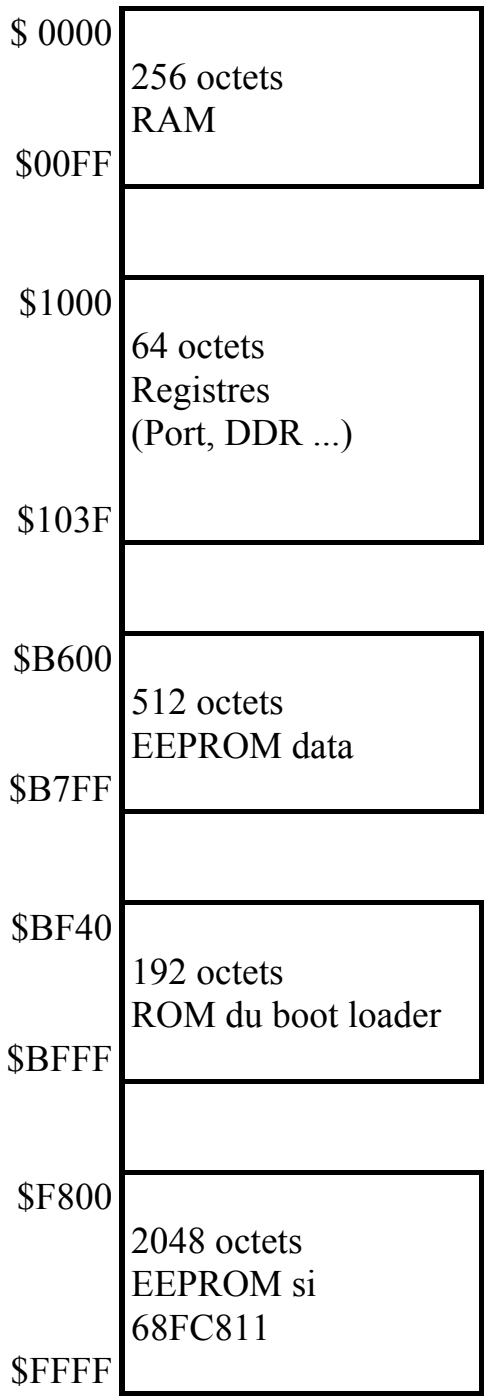
ETIQ BSET PORTD,X,#$40 = Mise à « 1 » du bit b_6 du de l'adresse X+PORTD

ETIQ BCLR MEM,X,#$08 = Mise à « 0 » du bit b_3 de l'adresse X+MEM

ETIQ BRSET PORTB,X,#$01 ETIQ = Saut à ETIQ si le bit b_0 de PORTB+X est à « 1 »

ETIQ BRCLR MEM,#$02 ETIQ = Saut à ETIQ si le bit b_1 de MEM est à « 0 »

b. MAPPING 68HC11



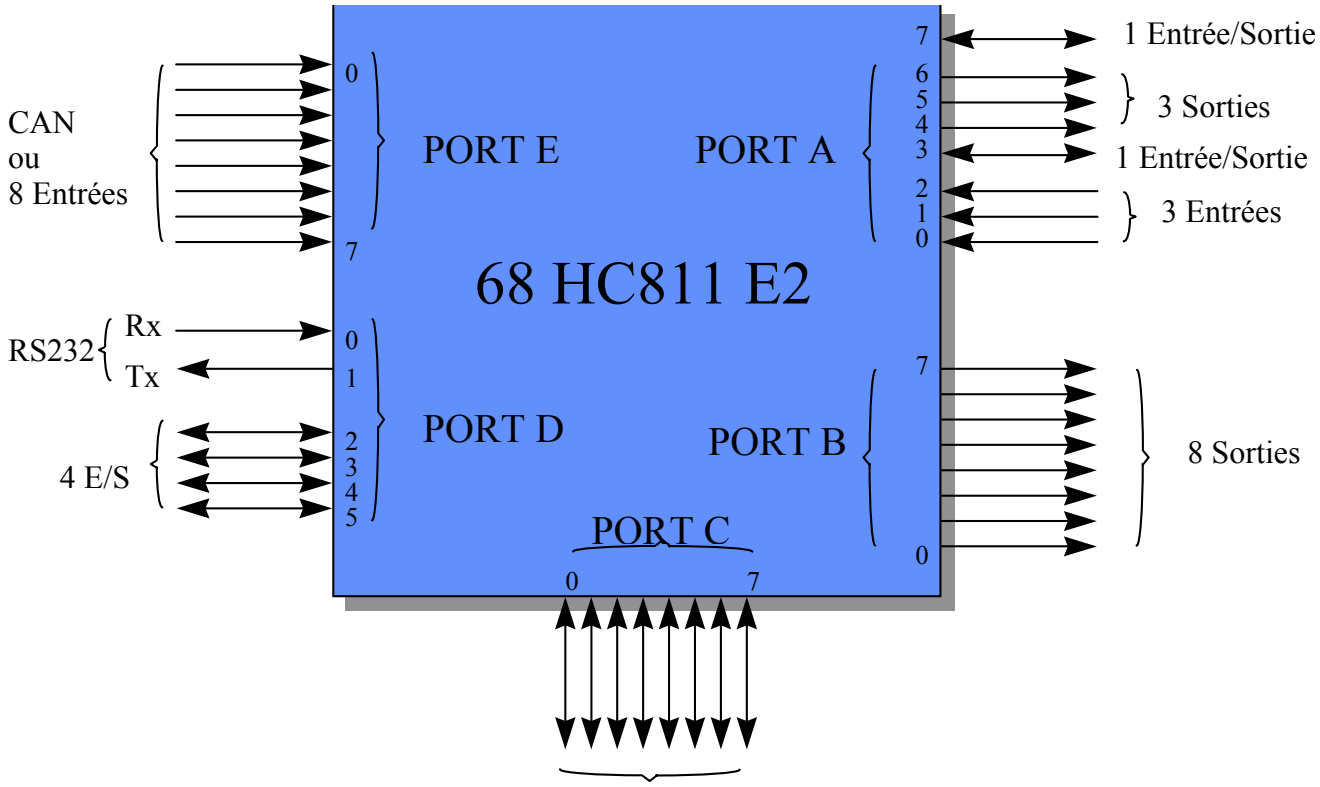
MODE NORMAL: Après un Reset le micro démarre à l'adresse qui est en $FFFE - $FFFF.

MODE BOOT STRAP:

Si aprés un Reset le micro reçoit un $FF sur le PORT série ($T^x$ et R^x non bouclés), il démarre en $BF40 donc il exécute le programme Boot Loader. Celui ci charge en RAM les 256 octets qui arrivent par la RS232.

Si après le Reset, le micro reçoit un $00 sur le PORT série ($T^x$ et R^x bouclés), il démarre en EEPROM en $B600.

c. PORTS du 68HC11



TOTAL : ➔ 12 Entrées.
➔ 12 Sorties.
➔ 14 Entrées / Sorties.

NB: Au reset toutes les lignes E/S sont configurées en Entrées.
DDR = 0 ➔ Ligne configurée en Entrée.
DDR = 1 ➔ Ligne configurée en Sortie.

PORT A : 2 Entrées/Sorties, 3 Entrées, 4 Sorties. Partagé avec les E/S du TIMER.

PORT B : 8 Sorties. Partagé avec MSB du bus adresse en mode étendu.

PORT C : 8 Entrées/Sortie. Partagé avec Bus Data et LSB adresse en mode étendu.

PORT D : 6 Entrées/Sorties. Partagé avec PORT SYN (I^2C) ou ASYN (RS232).

PORT E : 8 Entrées. Partagé avec les entrées du Convertisseur Analogique Numérique.

NOTE: Les Ports en Entrée doivent être tirés au + Vcc par des résistances de 10 K.
Les 8 du port C : **PC_0 à PC_7**
Les 6 du port D: **PD_0 / PD_1 / PD_2 / PD_3 / PD_4** et **PD_5**
Les 4 du port A : **PA_0 / PA_1 / PA_2 / PA_7**

d. 68 HC 811 E2

Il est compatible pin à pin avec le 68HC11A1 mais il possède 2K d'EEPROM implantée de $F800 à $FFFF.

- 1 - ECRITURE du PROGRAMME :

Pour que le micro puisse tourner en mode « SEUL » :

- Il faut initialiser la pile au début de la RAM en $00FF par :

```
LDS   #$00FF
```

- Il faut initialiser le vecteur de Reset en $FFFE et $FFFF . Si DEBPROG est l'étiquette du début de programme :

```
ORG   $FFFE
FDB   DEBPROG
```

2 - PROGRAMMATION DE L'EEPROM :

En mode « BOOT » sous PCBUG :

- Changer la valeur du registre BPROT de protection en écriture EEPROM à l'adresse $1035. Au Reset il y a $1F et il faut mettre $10.

```
MM $1035
$1F ⟶ $10   ↵
```

- Définir la zone EEPROM :

EEPROM $F800 $FFFF : « Erase before write » est autorisé.

EVENTUELLEMENT : Si on fait **EEPROM ERASE BULK $F800** après avoir définit la zone EEPROM la fonction « Erase before write » est inhibée. La programmation sera plus rapide

- Charger le programme :

LOADS NOMFICH :

Durée de programmation 20ms : 10ms pour erase + 10 ms pour write par octet.

- Lancer le programme :

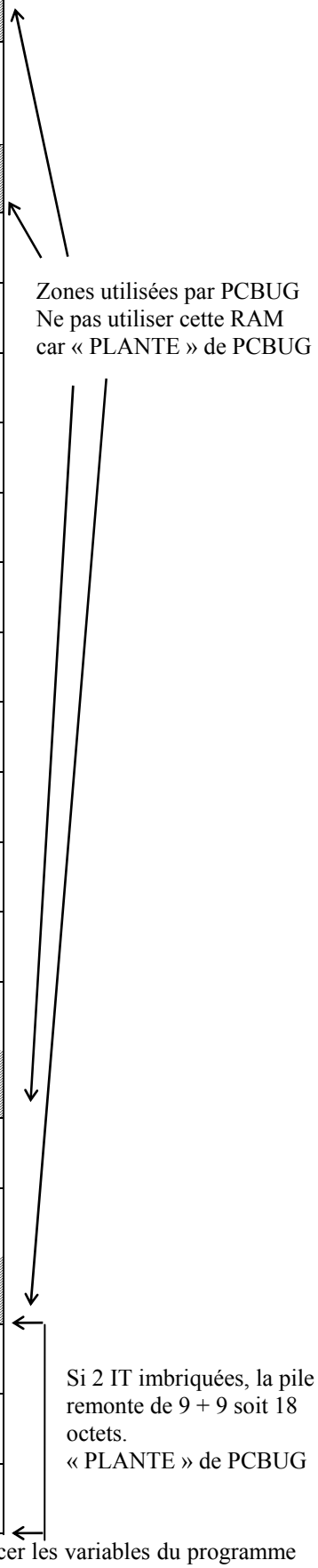
Sous PCBUG en mode BOOT : **G $F800**

En mode seul : **Reset**

.

RAM en MODE BOOTSTRAP

$00 $AF	175 Octets Programme Talker avec PCBUG 11
$B0 $C3	20 Octets DISPONIBLE
$C4 $C6	3 Octets Interruption SCI (Série ASYN)
$C7 $C9	3 Octets Interruption SPI (Série SYN)
$CA $CC	3 Octets Interruption : ENTREE DU COMPTEUR
$CD $CF	3 Octets Interruption : DEBORDEMENT DU COMPTEUR
$D0 $D2	3 Octets Interruption : DEBORDEMENT DU TIMER
$D3 $D5	3 Octets Interruption : OC5 : COMPARATEUR 5
$D6 $D8	3 Octets Interruption : OC4 : COMPARATEUR 4
$D9 $DB	3 Octets Interruption : OC3 : COMPARATEUR 3
$DC $DE	3 Octets Interruption : OC2 : COMPARATEUR 2
$DF $E1	3 Octets Interruption : OC1 : COMPARATEUR 1
$E2 $E4	3 Octets Interruption : IC3 : ENTREE de CAPTURE 3
$E5 $E7	3 Octets Interruption : IC2 : ENTREE de CAPTURE 2
$E8 $EA	3 Octets Interruption : IC1 : ENTREE de CAPTURE 1
$EB $ED	3 Octets Interruption temps reel : RTI
$EE $F0	3 Octets Interruption IRQ
$F1 $F3	3 Octets Interruption de PCBUG : XIRQ
$F4 $F6	3 Octets Interruption de PCBUG : SWI
$F7 $F9	3 Octets Interruption CODE ILLEGAL
$FA $FC	3 Octets Interruption COP : CHIEN DE GARDE
$FD $FF	3 Octets Interruption DEFAUT d'HORLOGE



Pour exécuter un programme en mode BOOTSTRAP sous contrôle de PCBUG11 il faut placer les variables du programme et la PILE dans la zone de 20 octets :**$B0 - $C3**. On peut également exploiter la zone des interruptions qui ne sont pas validées : **$C4 - $FF**. La pile utilise **9 octets** pour sauvegarder le contexte l'ors d'une interruption et **2 octets** pour un SP.

e. **LES INTERRUPTIONS**

Exemple :

Initialisation de IRQ en mode BOOT : **ORG $00EE**
FCB $7E
FDB ETIQ-SPIRQ

Initialisation de IRQ en mode seul : **ORG $FFF2**
FDB ETIQ-SPIRQ

VECTEURS des INTERRUPTIONS

ADRESSE	INTERRUPTION
$FFFE - $FFFF	RESET.
$FFFC - $FFFD	Surveillance horloge.
$FFFA - $FFFB	COP (Chien de garde).
$FFF8 - $FFF9	Code opération illégal.
$FFF6 - $FFF7	SWI.
$FFF4 - $FFF5	XIRQ .
$FFF2 - $FFF3	IRQ.
$FFF0 - $FFF1	Interruption temps réel.
$FFEE - $FFEF	TIC1.
$FFEC - $FFED	TIC2.
$FFEA - $FFEB	TIC3.
$FFE8 - $FFE9	TOC1.
$FFE6 - $FFE7	TOC2.
$FFE4 - $FFE5	TOC3.
$FFE2 - $FFE3	TOC4.
$FFE0 - $FFE1	TOC5.
$FFDE - $FFDF	Débordement TIMER.
$FFDC - $FFDD	Accu impulsions débordement.
$FFDA - $FFDB	Accu impulsions entrée.
$FFD8 - $FFD9	SPI.
$FFD6 - $FFD7	SCI.

f. DIRECTIVES ASSEMBLEUR

◆ **ORG :** Fixe l'origine des adresses pour l'assemblage.

Ex: ORG $B600

◆ **FCB :** Réserve une adresse mémoire et l'initialise à une valeur.

Ex : ORG $xxxx
FCB $7E

La valeur $7E sera écrite à l'adresse $xxxx

◆ **FDB :** Réserve deux adresses (un mot) et les initialise.

Ex : ORG $xxxx
FDB ETIQ

La valeur de ETIQ est écrite à l'adresse $xxxx. Comme ETIQ est une adresse sur 16 bits son MSB sera mis en $xxxx et son LSB en $xxxx + 1

◆ **FCC :** Réserve plusieurs adresses et les initialise avec le code ASCII des caractéres du texte.

Ex : ORG $xxxx
FCC "TEXTE"

Le code ASCII de T sera mis à l'adresse $xxxx puis celle de E en $xxxx +1

◆ **RMB :** Réserve des adresses memoire mais sans les initialiser à une valeur particulière.

Ex: ORG $xxxx
RMB $06

Les 6 cases memoire à partir de l'adresse $xxxx sont réservées.

◆ **EQU :** Affecte une valeur permanente à une étiquette.

Ex: PORTB EQU $04

g. REGISTRES INTERNES du 68HC11

$1000 = PORT A (DDRA des Bit3 et Bit7 en $1026 = PACTL)
$1001 = RESERVE MOTOROLA
$1002 = PIOC
$1003 = PORT C
$1004 = PORT B
$1005 = PORT CL
$1006 = RESERVE MOTOROLA
$1007 = DDRC
$1008 = PORT D
$1009 = DDRD
$100A = PORT E
$100B = CFORC
$100C = OC1M
$100D = OC1D
$100E et $100F = TCNT
$1010 et $1011 = TIC1
$1012 et $1013 = TIC2
$1014 et $1015 = TIC3
$1016 et $1017 = TOC1
$1018 et $1019 = TOC2
$101A et $101B = TOC3
$101C et $101D = TOC4
$101E et $101F = TOC5
$1020 = TCTL1
$1021 = TCTL2
$1022 = TMSK1
$1023 = TFLG1
$1024 = TMSK2
$1025 = TFLG2
$1026 = PACTL
$1027 = PACNT
$1028 = SPCR
$1029 = SPSR
$102A = SPDR
$102B = BAUD
$102C = SCCR1
$102D = SCCR2
$102E = SCSR
$102F = SCDR
$1030 = ADCTL
$1031 = ADR1
$1032 = ADR2
$1033 = ADR3
$1034 = ADR4
$1035 à $1038 = RESERVE MOTOROLA
$1039 = OPTION
$103A = COPRST
$103B = PPROG
$103C = HPRIO
$103D = INIT
$103E = TEST1
$103F = CONFIG

h. SORTIES « OU CABLE »

Les Ports C et D peuvent être configurés en sorties à « Drain Ouvert » donc en « OU CABLE ». Dans ce cas le transistor MOS canal P vers V_{DD} est désactivé. Il faut alors tirer la sortie vers le + V_{DD} avec une résistance. Plusieurs sorties peuvent alors être reliées entre elles.
Attention : Ne pas tirer la sortie par la résistance de rappel à une tension supérieure à V_{DD} car le transistor MOS interne qui est désactivé, reste physiquement en place et se comporte comme une diode entre V_{DD} et la sortie.

PORTC : Le bit 5 **CWOM** du registre **PIOC en $1002** mis à « 1 » passe les 8 sorties du PORT C en « OU CABLE ». Au reset ce bit est à « 0 », donc les sorties sont configurées normalement.

PORT D : Le bit 5 **DWOM** du registre **SPCR en $ 1028** mis à « 1 » passe les 6 sorties du PORT D en « OU CABLE ». Au reset ce bit est à « 1 », donc les sorties sont en « DRAIN OUVERT ».

i. 68HC11 et ADRESSAGE

***IMMEDIAT*:**
La donnée fait partie de l'instruction. Par exemple si on veut initialiser un registre avec une valeur donnée, on utilisera ce mode d'adressage.
Exemple: **LDAA #$4F** Le registre A est chargé par la valeur en hexadécimal : 4F.
Ce mode est spécifié à l'assembleur par l'utilisation du préfixe # avant la donnée.

***DIRECT*:**
La donnée est à une adresse codée sur 8 bits, c'est à dire comprise entre $00 et $FF donc dans la zone RAM. Un seul octet est nécessaire pour spécifier l'adresse.
Exemple: **STAA $0F** Le contenu du registre A est rangé à l'adresse $0F.

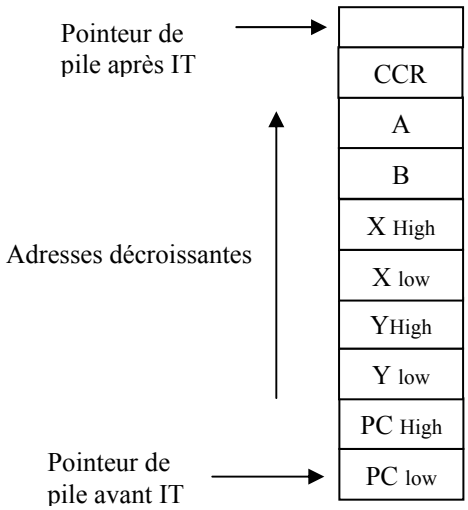
***ETENDU*:**
La donnée est à une adresse codée sur deux octets, c'est à dire n'importe où dans l'espace adressable par le 68HC11 qui a 16 bits d'adresse.
Exemple: **STAA $1000** Le contenu du registre A est rangé à l'adresse $1000

***INDEXE*:**
La donnée est à une adresse spécifiée par le contenu un registre d'index: X ou Y auquel on ajoute une valeur d'offset contenue dans l'instruction.
Exemple: **LDX #$1000**
LDAA 5,X Le registre A est chargé par l'octet de l'adresse $1005.

***INHERENT*:**
La donnée n'est pas nécessaire au micro car l'instruction est implicite.
Exemple: **TAB** Transfert du contenu du registre A vers le registre B.

j. 68HC11 et INTERRUPTIONS

Il existe de nombreuses sources d'interruption pour le 68HC11: 2 entrées externes sensibles à un état bas : IRQ et XIRQ, une interruption par logiciel SWI et un certain nombre de sources internes liées aux différents périphériques qui équipent le circuit (timer, SCI, SPI).
Quand une interruption apparaît, le micro termine d'abord l'instruction en cours, tous les registres ainsi que le compteur ordinal sont sauvegardés dans la pile, et le bit I du registre de condition est mis à "1", ce qui interdit toute nouvelle interruption. Seule XIRQ qui est NON MASQUABLE sera traitée si elle intervient pendant une phase d'interruption.
Ensuite le vecteur correspondant à l'interruption de plus haute priorité est recherché, et sa valeur chargée dans le compteur ordinal. Le sous programme lié à l'interruption va donc s'exécuter.
Ce programme se termine par l'instruction **RTI** qui a pour effet de faire exécuter les opérations inverses, à savoir restitution des registres, remise à "0" du bit I du CCR autorisant la prise en compte de nouvelles interruptions et chargement du compteur ordinal avec la valeur qu'il avait avant l'interruption, ce qui fait reprendre le programme à l'endroit où il avait été interrompu.
Au reset du système le bit I du registre CCR est à "1", ce qui masque les interruptions. Si on désire que les interruptions soient actives, il faudra les autoriser en passant le bit I à "0". On utilisera l'instruction: **CLI**. Au contraire si on a besoin de les masquer au cours du programme on utilisera l'instruction : **SEI**.



A chaque interruption il y aura 9 octets de sauvegardés dans la RAM.

Il ne faut pas oublier d'initialiser le pointeur de pile SP sur une adresse en RAM ou le micro pourra sauvegarder ses registres.
Exemple: SP pointe le bas de la zone des 256 octets de RAM interne: **LDS #$00FF**
Il faudra initialiser le vecteur d'interruption en écrivant à l'adresse du vecteur l'adresse du programme correspondant.
Exemple: SPIRQ est l'étiquette du sous programme à exécuter si un niveau bas apparaît sur l'entrée IRQ.

```
ORG   $FFF2
FDB   SPIRQ
```

Il faudra au début du programme autoriser les interruptions.
Exemple: **CLI**

2. Les ports d'entrées / sorties parallèles

Le **68HC811E2** a un total d'une quarantaine de broches d'entrées sorties d'usage général. Ces broches sont partagées avec au moins une autre fonction périphérique non étudiée dans ce chapitre.

I) Le port A :

a) description de l'interface :

Le port A est un port parallèle à usage général partagé avec le **TIMER** et l'accumulateur d'impulsions. De ce fait, **PA0** à **PA2** sont figées en entrée, **PA4** à **PA6** sont figées en sortie et **PA7 et PA3** est bidirectionnel.

Les broches configurées en entrée sont triggerisées, permettant d'éliminer les déclenchements multiples lorsqu'elles sont utilisées par le **TIMER** ou l'accumulateur d'impulsions.

Le sens de fonctionnement de **PA7** est programmé par le bit **DDRA7** du registre de contrôle de l'accumulateur d'impulsions **PACTL**.

Le registre **PORTA** permet l'écriture ou la lecture de **PA0** à **PA7**.

Remarque :
Si certaines lignes de sortie sont configurées pour être utilisées par le **TIMER**, il est impossible de modifier leur état par une écriture dans le registre **PORTA**.

Le résultat de cette écriture est cependant mémorisé dans un latch et, lorsque les broches sont libérées par le **TIMER**, les données mémorisées sont transférées en sortie.

Même si certaines lignes sont configurées pour être utilisées par le **TIMER**, leur état peut toujours être lu dans le **PORTA**.

b) les registres associés :

- **le registre PORTA :**

Ce registre 8 bits est accessible en lecture écriture.

Reset	-	0	0	0	0	-	-	-	
	PA7	**PA6**	**PA5**	**PA4**	**PA3**	**PA2**	**PA1**	**PA0**	PORTA $1000

Une lecture du contenu de ce registre retourne l'état réel des broches en entrée mais donne l'état précédemment écrit dans le registre pour les bits correspondants à des broches de sortie.

Pour envoyer une donnée vers le port A, il suffit donc d'écrire :

```
PORTA   EQU   $1000
        STAA  PORTA ;  la donnée est dans l'accumulateur A
```

Pour recevoir une donnée qui arrive sur le port A :

```
PORTA   EQU   $1000
        LDAA  PORTA ; la donnée se trouve maintenant dans l'accumulateur A
```

- **Le registre PACTL :**

Ce registre 8 bits est accessible en lecture écriture. Seul le bit **DDRA7** nous intéresse dans ce paragraphe.

Reset	0	0	0	0	0	0	0	0	
	DDRA7	PAEN	PAMOD	PEDGE	-	-	RTR1	RTR0	PACTL $1005

	Sens de fonctionnement de PA7
0	Entrée commune avec l'entrée PAI de l'accumulateur d'impulsions
1	Sortie commune avec l'entrée PAI de l'accumulateur d'impulsions

Si vous voulez que **PA7** soit, par exemple en sortie, il faut écrire :

```
PACTL   EQU   $1005
        LDAA  #%10000000  ; on veut mettre DDRA7 en sortie
        ANDA  #%10000000  ; on masque les autres bits pour ne changer que DDRA7
        STAA  PACTL       ; on envoie dans PACTL
```

II) les ports B et C, les broches STRA et STRB :

a) Description des interfaces :

Ces deux ports sont traités ici dans la même rubrique car ils partagent un certain nombre de mécanismes de fonctionnement communs.

Le port B est un port de sortie à usage général dont le sens est figé.
Le positionnement des lignes du port B se fait en écrivant dans le registre **PORTB**.

Une écriture de ce registre renvoie les données qui ont été précédemment écrits mais est sans rapport avec l'état réel des broches du port.

Le port C est un port bidirectionnel. Le sens de chaque ligne est programmé grâce aux bits correspondants au registre **DDRC**.
La lecture ou écriture des données de ou vers le port C fait par contre appel à deux registres : **PORTC** et **PORTCL**.

Les ports B et C supportent divers modes de fonctionnement en « dialogue » qui font intervenir les lignes **STRA** et **STRB**.
Ces modes de fonctionnement se programment grâce à divers bits du registre **PIOC**.

b) les registres associés :

- **le registre PORTB** :

Le port B est un port de sortie 8 bits à usage général accessible en lecture écriture.
Il ne dispose pas de registre **DDR** associé puisque son sens de fonctionnement est figé.

Reset	0	0	0	0	0	0	0	0	
	PB7	**PB6**	**PB5**	**PB4**	**PB3**	**PB2**	**PB1**	**PB0**	PORTB $1004

Pour envoyer une donnée vers le port A, il suffit donc d'écrire :

```
PORTB   EQU   $1004
        STAA  PORTB ;  la donnée est dans l'accumulateur A
```

- **le registre DDRC :**

Ce registre 8 bits de direction de données est accessibles en lecture écriture.

Reset	0	0	0	0	0	0	0	0
	DDRC7	**DDRC6**	**DDRC5**	**DDRC4**	**DDRC3**	**DDRC2**	**DDRC1**	**DDRC0**

DDRC $1007

	Sens
0	Entrée
1	Sortie

Si vous voulez par exemple que les broches **PC0** à **PC5** soient en sortie et les broches **PC6** et **PC7** en entrée, il faut écrire :

```
DDRC    EQU     $1007
        LDAA    #%00111111
        STAA    DDRC
```

- **le registre PORTC et PORTCL :**

Ces deux registres 8 bits sont accessibles en lecture écriture.

Reset	-	-	-	-	-	-	-	-
	P C7	**PC6**	**PC5**	**PC4**	**PC3**	**PC2**	**PC1**	**PC0**

PORTC $1003

Reset	-	-	-	-	-	-	-	-
	PCL7	**PCL6**	**PCL5**	**PCL4**	**PCL3**	**PCL2**	**PCL1**	**PCL0**

PORTCL $1005

- **le registre PIOC :**

Ce registre huit bits, accessible en lecture écriture, sert à sélectionner le mode de dialogue utilisé mais aussi les états des divers signaux ou transitions.

	Etat du détecteur de front de STRA
0	RAZ : lecture de PIOC (avec STAF = 1) et accès à PORTCL (lecture ou écriture suivant mode de fct)
1	Transition active détectée sur STRA

	Masque d'interruption de STRA
0	Demande inhibée
1	Interruptions (générées par passage à 1 de STAF) autorisées

	Configuration matérielle du port C
0	Sortie CMOS
1	Sorties à drain ouvert nécessitant une résistance de pull up

		Mode de fonctionnement
0	x	Dialogue simple
1	0	Dialogue complet en entrée
1	1	Dialogue complet en sortie

Reset	0	0	0	0	0	-	1	1	
	STAF	**STAI**	**CWOM**	**HDNS**	**OIN**	**PLS**	**EGA**	**INVB**	PIOC $1002

Fonctionnement de STRB	
Mode verrouillé	0
Mode impulsionnel	1

Choix du front actif de STRA	
Front descendant	0
Front montant	1

Niveau actif de STRB	
Niveau bas	0
Niveau haut	1

III) Description des modes de fonctionnement :

a) Dialogue simple :

Une écriture dans le registre **PORTB** place les données correspondantes en sortie sur le port de même nom et génère une impulsion d'une durée de 2 cycles d'horloge E sur la ligne **STRB**.

En entrée : la lecture du registre **PORTC** est le reflet de l'état courant des lignes de port. La lecture de **PORTCL**, par contre, est le reflet de l'état de ces lignes au moment de la transition valide de la ligne **STRA**.

En sortie : le fait d'écrire dans **PORTC** ou dans **PORTCL** produit les mêmes effets au niveau des broches configurées en sortie.
Par contre l'écriture dans un de ces registres est évidemment sans aucun effet sur les broches configurées en entrée.
Ces données sont cependant mémorisées localement et sont présentées en sortie si les broches correspondantes changent de sens de fonctionnement.

b) Dialogue complet en entrée :

Lorsqu'une condition « prêt » est reconnue, le système externe place les données sur les lignes du port C, puis délivre un front actif sur la ligne **STRA**.
Ce front verrouille les données du port C dans le registre **PORTCL** et positionne l'indicateur **STAF** (provoquant éventuellement une interruption).

En mode « verrouillé » (**PLS** = 0), la ligne **STRB** est mise à son état actif pour inhiber le système externe en ce qui concerne la validation de nouvelles données sur le port C.
La lecture du registre **PORTCL** efface l'indicateur **STAF** et repositionne **STRB** à l'état « prêt » pour permettre un nouveau cycle.
En mode « impulsionnel » (**PLS** = 1), la lecture du registre **PORTCL** efface l'indicateur **STAF** et génère une impulsion d'une durée de 2 cycles sur la ligne **STRB** pour indiquer que le microcontrôleur est « prêt » pour un nouveau cycle.

c) Dialogue complet en sortie :

L'écriture d'une donnée dans le registre **PORTCL** positionne les lignes de sortie du port C, active la ligne **STRB** pour indiquer au système externe que les informations sont valides et remet éventuellement à zéro le bit **STAF**.

En mode impulsionnel (**PLS** = 0), la ligne **STRB** reste positionnée seulement pendant 2 cycles d'horloge.
En mode verrouillé (**PLS** = 1), la ligne **STRB** restera jusqu'à l'apparition d'un front valide sur **STRA**.

Le système externe, après avoir traité la donnée, active la ligne **STRA** pour indiquer qu'un nouveau cycle peut débuter.

Le front actif reconnu par le microcontrôleur sur la ligne **STRA** positionne le bit **STAF** à « 1 » et génère éventuellement une interruption.
Une écriture dans **PORTCL** initialise un nouveau cycle.

IV) Le port D :

a) Description de l'interface :

Le port D est un port d'entrées sorties 8 bits à usage général partagé avec les interfaces séries asynchrones (**SCI**) et synchrones (**SPI**).
Tant que ces interfaces ne sont utilisées, le sens de travail des lignes du port est programmé par les bits correspondants du registre **DDRD** .
La sortie de données se fait par écriture de registre **PORTD** et c'est dans ce même registre que l'on vient lire l'état des broches qui sont positionnées en entrées.
Il est possible de programmer le port D pour fonctionner avec des sorties a drain ouvert en mettant à « 1 » le bit **DWON** du registre du contrôle **SPCR** de l'interface série synchrone.
Avec ce type de sortie, il est nécessaire de câbler des résistances de pull up.
Si l'une ou l'autre des interfaces séries est utilisée, le sens des lignes du port D est fixé en priorité par ces fonctions quelque soit l'état du bit correspondant du registre **DDRD**.
Dés que la fonction est sélectionnée, le bit de **DDRD** redevient significatif automatiquement.
Toutefois, dans certaines situations, le fonctionnement des lignes de la SPI est influencé par les bits correspondants du **DDRD**.

b) Les registres associés :

- **Le registre DDRD**

Ce registre 8 bits de direction de données est accessibles en lecture écriture.

Reset	0	0	0	0	0	0	0	0	
	0	**0**	**DDRD5**	**DDRD4**	**DDRD3**	**DDRD2**	**DDRD1**	**DDRD0**	DDRD $1009

	Sens
0	Entrée
1	Sortie

- **Le registre PORTD :**

Ce registre 8 bits de direction de données est accessibles en lecture écriture.

Reset	0	0	0	0	0	0	0	0	
	0	**0**	**PD5**	**PD4**	**PD3**	**PD2**	**PD1**	**PD0**	PORTD $1008

V) le port E :

Les 8 broches du port E sont les entrées du convertisseur analogique numérique.
Elles peuvent être aussi utilisées comme port d'entrée à usage général.
L'information présente sur le port est transmise seulement pendant la lecture du port.

Reset	0	0	0	0	0	0	0	0	
	PE7	**PE6**	**PE5**	**PE4**	**PE3**	**PE2**	**PE1**	**PE0**	PORTE $100A

3. L'INTERFACE SERIE ASYNCHRONE

a. Description de l'interface :

Cette interface permet au microcontrôleur d'échanger des données avec des circuits périphériques utilisant une liaison série asynchrone (ordinateurs, tables traçantes, modem, instruments de musique…)

Les circuits internes de la SCI permettent :

- Des échanges asynchrones bidirectionnels en full duplex
- De choisir la vitesse de transmission
- De choisir le format de transmission
- De choisir le mode d'activation de la partie réception (« classique » ou « réveil »)
- De gérer les erreurs de transmission
- D'indiquer la fin de transmission

b. Les broches utilisées :

La liaison est assurée par deux broches RXD (PD0) et TXD (PD1) du port D. après un RESET, la SCI est inhibée et le port D est configuré comme un port d'usage général.

La SCI peut être activée en réception et/ou en émission (half ou full duplex).

Lorsque la SCI est activée en réception, PD0 devient l'entrée de réception des données RXD.

Lorsque la SCI est activée en émission, PD1 devient la sortie de transmission des données TXD.

c. Les registres associés :

La liaison série asynchrone est entièrement gérée par 5 registres et le bit **DWON** commun à la SCI et à la SPI.

Le registre des données (**SCDR**) est le registre dans lequel on vient écrire la donnée à transmettre ou dans lequel on vient lire la donnée reçue.

Il faut au préalable avoir configuré correctement la **SCI** en programmant ses registres de contrôle **SCCR1** et **SCCR2** ainsi que le bit **DWON** du registre **SPCR**.

Le registre **BAUD** permet, en fonction du quartz cadençant le microcontrôleur, de sélectionner la vitesse de transmission des données.

Les informations sur l'état de la liaison sont fournies dans le registre d'état **SCSR**.

- **Le registre SCDR :**

Ce registre, accessible en lecture - écriture, est un registre 8 bits. Lors d'une transmission, c'est le bit de poids faible D0 qui apparaît en premier sur la ligne après le bit de start.

Reset : - - - - - - - -

D7	D6	D5	D4	D3	D2	D1	D0

SCDR $102F

- **Le registre BAUD :**

Le récepteur et l'émetteur travaillent à la même vitesse, dérivée de l'horloge interne du microcontrôleur. Le registre 8 bits **BAUD**, accessible en lecture – écriture permet de choisir cette vitesse.

Reset : 0 0 0 0 0 0 0 0 0

0	**0**	**SCP1**	**SCP0**	**0**	**SCR2**	**SCR1**	**SCR0**

BAUD $102B

Débit de transmission Maxi pour Q = 8 MHz		
D_M = 125 000 bauds	0	0
D_M = 41 666 bauds	0	1
D_M = 31 250 bauds	1	0
D_M= 9 600 bauds	1	1

Débit de transmission en bauds				**D_M = 9600 bauds**
$D=D_M/1$	0	0	0	9600 bauds
$D=D_M/2$	0	0	1	4800 bauds
$D=D_M/4$	0	1	0	2400 bauds
$D=D_M/8$	0	1	1	1200 bauds
$D=D_M/16$	1	0	0	600 bauds
$D=D_M/32$	1	0	1	300 bauds
$D=D_M/64$	1	1	0	150 bauds
$D=D_M/128$	1	1	1	75 bauds

- **Le registre SCSR :**

Ce registre 8 bits, accessible uniquement en lecture, permet de surveiller le bon déroulement des opérations de la SCI.

	Registre d'émission vide
0	Registre occupé
1	Registre vide (RAZ par lecture de SCSR puis lecture de SCDR)

	Transmission complète
0	Emission complète
1	Emission terminée (RAZ par lecture de SCSR puis lecture de SCDR)

	Registre de réception plein
0	Registre de réception incomplet
1	Réception terminée (RAZ par lecture de SCSR puis lecture de SCDR)

	Détection ligne en attente
0	Ligne RXD en activité
1	Ligne RXD en attente au niveau « 1 ».(RAZ par lecture de SCSR puis lecture de SCDR)

Reset	1	1	0	0	0	0	0	0
	TDRE	**TC**	**RDRF**	**IDLE**	**OR**	**NF**	**FE**	**0**

SCSR $102E

Erreur de surcharge	
Pas de surcharge	0
Surcharge du registre de réception. (RAZ par lecture de SCSR puis lecture de SCDR)	1

	Erreur de format
0	Format correct
1	Format incorrect. (RAZ par lecture de SCSR puis lecture de SCDR)

Bruit de transmission	
Réception non bruitée	0
Réception bruitée (RAZ par lecture de SCSR puis lecture de SCDR)	1

- **le registre SCCR1 :**

ce registre huit bits, accessible en lecture – écriture, permet de définir le format de transmission et la procédure de réveil de réception.

R8	En réception sur 9 bits : valeur de D8
T8	En émission sur 9 bits : valeur de D8

Reset : - - 0 0 0 0 0 0

R8	**T8**	**0**	**M**	**WAKE**	**0**	**0**	**0**

SCCR1 $102C

	Mode de réveil du récepteur
0	RXD à « 1 » pendant au moins 1 trame de 10 ou 11 bits
1	MSB du mot à « 1 »⇔adresse

Format de trame	
1 bit de start + 8 bits de données + 1 bit de stop	0
1 bit de start + 9 bits de données + 1 bit de stop	1

- **le registre SPCR :**

La liaison série asynchrone, tout comme la liaison série synchrone, peut être configurée pour disposer de sorties de type « **drain ouvert** ». Le choix se fait en positionnant le bit **DWOM** du registre **SPCR** et influence le comportement de l'ensemble du port D.

Reset : 0 0 0 0 0 0 - -

SPIE	**SPE**	**DWOM**	**MSTR**	**CPOL**	**CPH**	**SPR1**	**SPR0**

SCPR $1028

	Configuration matérielle de sortie du port D
0	Sorties de type CMOS
1	Sorties à « drain ouvert » nécessitant des résistances de pull up

- **le registre SCCR2 :**

ce registre 8 bits, accessible en lecture – écriture, permet de valider ou inhiber certaines fonctions de la SPI.

	Autorisation des interruptions en transmission
0	Interruption inhibée
1	Interruption si TDRE = 1 (registre de transmission vide)

	Autorisation de l'interruption « transmission complète »
0	Interruption inhibée
1	Interruption si TC = 1

	Autorisation des interruptions en réception
0	Interruption inhibée
1	Interruption si RDRF =1 (registre de réception plein) ou si OR = 1

	Autorisation d'interruption « ligne en attente »
0	Interruption inhibée
1	Interruption dés que la ligne est à « MARK » (10 ou 11 bits à 1

TIE	**TCIE**	**RIE**	**ILIE**	**TE**	**RE**	**RWU**	**SBK**

SCCR2 $102D

Activation de la SCI		
SCI désactivée (PD0 et PD1 sont des E/S usage général	0	0
Half duplex en réception (PD1 usage général)	0	1
Half duplex en émission (PD0 usage général)	1	0
Full duplex	1	1

Mode d'activation du récepteur	
UART classique	0
Mode réveil	1

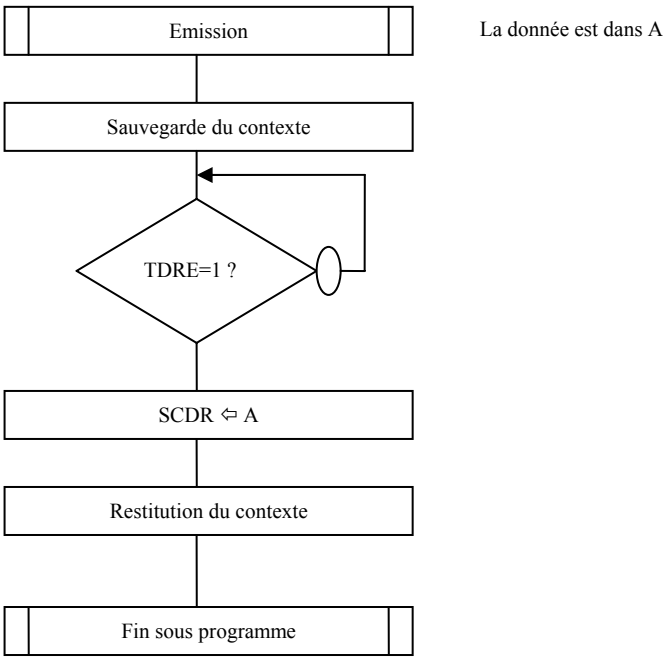
Emission d'un BREAK	
Opération normale en réception	0
Emission nen continu d'un BREAK (10 ou 11 bits au niveau 0)	1

d. fonctionnement de l'interface :

- fonctionnement en émission :

Par programme, le microcontrôleur vient tester l'indicateur d'état **TDRE** (registre de transmission vide).
S'il est à 1, le calculateur sait qu'il peut écrire une nouvelle donnée dans **SCDR** (l'interface peut également envoyer une demande d'interruption au microcontrôleur lorsque le registre de transmission est vide).
L'écriture d'un caractère dans **SCDR** désactive les indicateurs. Ce caractère est alors envoyé sur la ligne de transmission de données TX au rythme de l'horloge d'émission.

Algorithme :



Programme :

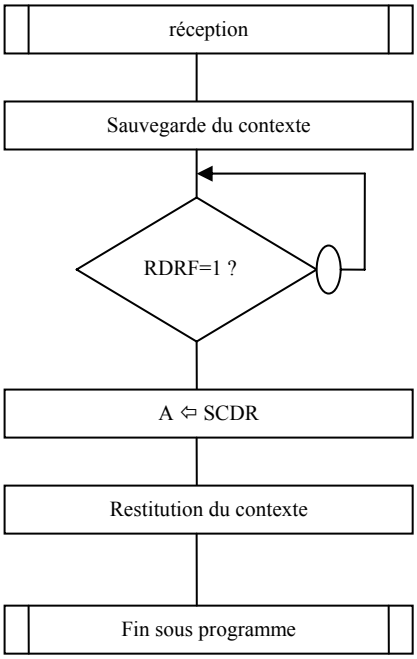
```
Envoi   PSHX
        LDX     #SCSR
        BRCLR   0,X,%10000000,envoi
        STAA    SCDR
        PULX
        RTS
```

- fonctionnement en réception :

le périphérique envoie un caractère à l'interface SCI sur la ligne de réception de donnée (RX). La transition sur la ligne RX du niveau 1 au niveau 0 (signal start) est détectée par l'interface série. S'il reconnaît le signal start, le microcontrôleur effectue sur les bits de données les traitements nécessaires, puis il positionne à « 1 » l'indicateur d'état **RDRF** (registre de réception plein).

Par programme, le microcontrôleur vient tester l'indicateur d'état RDRF (l'interface peut également envoyer une demande d'interruption au microcontrôleur lorsque le registre de réception est plein). S'il est à 1, le microcontrôleur sait qu'une donnée lui est destinée dans l'interface. Eventuellement, il vient lire le registre d'état et tester les indicateurs d'erreur pour vérifier la validité des caractères transmis.
Le microcontrôleur effectue ensuite une lecture du registre de réception, ce qui désactive les indicateurs.

Algorithme :



Programme :

```
recep   PSHX
        LDX     #SCSR
        BRCLR   0,X,%00100000,recep
        LDAA    SCDR
        PULX
        RTS
```

4. Le convertisseur analogique numerique

a. Description de l'interface :

Le convertisseur intégré dans le 68HC811E2 est un CAN à approximation successive. Il dispose de 8 entrées multiplexées vers un échantillonneur bloqueur permettant de diminuer les erreurs de conversion dues à la variation du signal d'entrée.
Les 8 entrées sont communes avec le port E.
Deux lignes spécifiques Vrefh et Vrefl sont utilisées comme référence de tension.
La précision de ce convertisseur est de ± ½ LSB sur toute la gamme de température, ces références de tension devront être stables par rapport aux variations de l'alimentation ou de la température pour conserver au convertisseur toute sa précision.
Il faut toujours garder Vrefh – Vrefl > 2,5 Volts.

b. Description de la conversion :

Une tension d'entrée égale à Vrefl est convertie en $00, une tension d'entrée égale ou supérieure à Vrefh est convertie en $FF.
Lors d'une demande de conversion, une séquence de 4 conversions est lancée.
Le convertisseur utilise le principe des approximations successives :
La tension est comparée à la différence Vi = Vrefh – Vrefl divisée par 2.
si la tension à convertir est supérieure à cette valeur, le bit 7 du registre de comparaison est mis à 1, dans le cas contraire, il est mis à 0 ainsi que la tension Vi.
Cette nouvelle valeur est comparée à la tension à mesurer et le bit 6 est positionné en fonction de ce résultat.
Finalement, la tension à mesurer est comparée à Vi avec cette équation :

$$V_i = \left(V_{refh} - V_{refl}\right)\left(\frac{b_7}{2} + \frac{b_6}{4} + \frac{b_5}{8} + \frac{b_4}{16} + \frac{b_3}{32} + \frac{b_2}{64} + \frac{b_1}{128} + \frac{b_0}{256}\right)$$

b_n=0 ou 1 suivant le résultat des comparaisons internes.

Chaque opération est effectuée en 32 cycles d'horloge E. (si E > 750 kHz).
Le convertisseur possède 4 registres de résultat séparés qui permettent de mémoriser les résultats de la séquence de 4 conversions.
Cette séquence débute un cycle d'horloge interne après une écriture dans le registre **ADCTL**.

c. Les entrées de conversion :

Un multiplexeur permet de sélectionner une entrée analogique parmi 16 voies analogiques :
8 de ces voies correspondent aux broches d'entrée du port E, 4 autres à des références internes, le reste étant réservé pour un usage futur.

Le choix de la voie grâce au registre **ADCTL**

d. Les registres du convertisseur :

- **le registre ADCTL :**

	Indicateur de conversion terminée
1	Lorsqu'une séquence de conversion est terminée et que les 4 registres de résultats contiennent les résultats de la conversion.
0	Par écriture dans le registre ADCTL, ce qui à pour effet de relancer une nouvelle séquence de conversion.

	Commande de balayage continu
1	Conversion continue, les registres résultats sont en permanence mis a jour
0	La séquence (4 conversions) s'arrete lorsque les 4 registres résultats sont pleins

	Commande 1 voie / plusieurs voies
1	La conversion s'effectue sur 4 voies successives
0	La conversion est effectuée 4 fois sur la voie choisie grâce à CD à CA

Reset	0	0	-	-	-	-	-	-	
	CCF	**0**	**SCAN**	**MULT**	**CD**	**CC**	**CB**	**CA**	ADCTL $1030

Sélection de voie					
Canal	**Résultat**				
AN0	ADR1	**0**	**0**	**0**	**0**
AN1	ADR2	**0**	**0**	**0**	**1**
AN2	ADR3	**0**	**0**	**1**	**0**
AN3	ADR4	**0**	**0**	**1**	**1**
AN4	ADR1	**0**	**1**	**0**	**0**
AN5	ADR2	**0**	**1**	**0**	**1**
AN6	ADR3	**0**	**1**	**1**	**0**
AN7	ADR4	**0**	**1**	**1**	**1**
Réservé	ADR1	**1**	**0**	**0**	**0**
Réservé	ADR2	**1**	**0**	**0**	**1**
Réservé	ADR3	**1**	**0**	**1**	**0**
Réservé	ADR4	**1**	**0**	**1**	**1**
Vrh	ADR1	**1**	**1**	**0**	**0**
Vrl	ADR2	**1**	**1**	**0**	**1**
Vrh/2	ADR3	**1**	**1**	**1**	**0**
Réservé	ADR4	**1**	**1**	**1**	**1**

On peut donc convertir sur une seule entrée analogique avec **MULT** = 0 et on aura deux cas :
SCAN = 0 : la voie convertie est 4 fois et les résultats se retrouvent dans **ADR1** à **ADR4** et la conversion est stoppée. On peut relancer une conversion par écriture dans **ADCTL**.
SCAN = 1 : la conversion est effectuée sans arrêt sur le canal choisi. Le résultat de la 5 éme conversion est stocké dans **ADR1** écrasant la première conversion.

On peut aussi convertir sur plusieurs entrées avec **MULT** = 1 avec toujours 2 possibilités :

SCAN = 0 : une conversion est effectuée successivement sur chaque voie, les résultats sont placés respectivement dans **ADR1**, **ADR2**, **ADR3** et **ADR4** puis la conversion s'arrête.

SCAN = 1 : la conversion est effectuée sans arrêt selon le principe vu précédemment.

- **le registre OPTION :**

Deux bits du registre **OPTION** sont utilisés pour contrôler le convertisseur analogique numérique.

Reset	0	0	0	1	0	0	0	0	
	ADPU	**CSE**	**IRQE**	**DLY**	**CME**	**-**	**CR1**	**CR0**	OPTION $1039

	Choix de l'horloge de commande CAN
0	Le CAN utilise l'horloge interne E
1	Sélectionne l'horloge locale à cellule RC

	Choix de l'horloge de commande CAN
0	A 0 après un reset
1	Pour utiliser le CAN, il faut mettre ce bit à 1

e. Exemple d'utilisation :

On relie sur les broches PE0 et PE1 deux LDR. On veut convertir la tension fournie qui est le reflet de la luminosité reçue par les LDR, et sauvegarder ces valeurs en mémoire.

```
CONV    LDX     #OPTION
        BSET    0,X,%10000000 ;  validation CAN
        LDX     #ADCTL
        BSET    0,X,%00010000 ; début de conversion
Bcl     LDX     #ADCTL
        BRCLR   0,X,%10000000,Bcl
        LDAA    ADR1
        STAA    $0000
        LDAA    ADR2
        STAA    $0001
        RTS
```

5. Le TIMER

a. Description de l'interface :

Le TIMER est divisé en quatre blocs fonctionnels distincts :

a) Le TIMER en usage général
b) Le watchdog (appelé aussi COP : computer operating properly) c'est-à-dire le chien de garde
c) Le générateur d'impulsion temps réel
d) L'accumulateur d'impulsions

Ces quatre blocs utilisent une source d'horloge et une chaîne de diviseurs en partie commune mais sont indépendantes les uns des autres.

b. Le TIMER à usage général :

Il utilise un compteur 16 bits qui s'incrémente en permanence (free running) dés la remise à zéro du microcontrôleur.

Il est précédé par un pré diviseur programmable cadencé par l'horloge interne E. On peut donc programmé des fréquences d'entrée multiples de E.

Il dispose aussi de trois registres dits de *capture* (**TIC1**, **TIC2** et **TIC3**) sont prévus pour mémoriser l'instant d'arrivée de front actif sur les trois entrées correspondantes (**IC1**, **IC2** et **IC3**), un bit d'état est positionné (**IC1F**, **IC2F** et I**C3F**) et une interruption peut être générée.
Ces trois entrées vont permettre la mesure de longueur d'impulsions, de fréquence ou d'écart entre événements différent.

La deuxième fonction importante du TIMER est la comparaison en sortie et fait appel à 5 registres de comparaison en sortie (**TOC1** à **TOC5**).
On peut les programmer à tout instant et ils sont comparés en permanence avec le contenu du compteur 16 bits.
Quand il y a égalité, un bit d'état est positionné (**OC1F** à **OC5F**), une interruption peut être générée et la patte de sortie correspondante (**OC1** à **OC5**) peut changer d'état comme on le désir.

- **Pré diviseur et compteur :**

La base du TIMER est un compteur 16 bits qui compte en permanence de $0000 à $FFFF sans aucun moyen de l'arrêter et qui positionne à 1 un bit de dépassement lors du passage de $FFFF à $0000.

Par contre, on peut venir lire en permanence la valeur du compteur grâce au registre **TCNT**.
Le pré diviseur permet de définir le signal d'horloge qui pilote ce compteur (en fait on divise E).

- **le registre TCNT :**

Bit15							Bit8	$100E	TCNT
Bit7							Bit0	$100F	

C'est un registre 16 bits, pour le lire on ne peut donc pas utiliser les accumulateurs A et B.

```
LDX   TCNT
LDY   TCNT
LDD   TCNT
```

- **Le registre TMSK2 :**

Le prédiviseur est programmable grâce aux bits **PR0** et **PR1**.
Le tableau suivant est donné pour une fréquence de quartz de 8 MHz.

Ce registre est partagé avec certaines parties du TIMER :

- Seul PR0 et PR1 nous serons utiles pour la programmation du prédiviseur.
- Validation des interruptions avec PAII, TOI, RTII et PAOVI.

		Rapport de pré division	
		E Divisé par	Période/ temps maximal
0	0	1	500ns/32 ,77ms
0	1	4	2µs/131,1ms
1	0	8	4µs/262,1ms
1	1	16	8µs/524,3ms

Reset	0	0	0	0	0	0	0	0	
	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0	TMSK2 $1024

	Bit de validation des interruptions sur PAI
0	Interruptions inhibées
1	Interruptions autorisées et effectives à chaque front actif sur l'entrée

	Validation d'interruption lors de dépassement
0	Interruption inhibée
1	Les interruptions sont autorisées et effectives a chaque passage à 0 du compteur

	Interruption temps réel
0	Interruption inhibée, mais la RTI continue de fonctionner et de positionner le bit RTIF
1	Interruptions autorisées

	Validation des demandes d'interruption par l'indicateur de débordement
0	Interruptions inhibées mais le bit TOF continue de fonctionner normalement
1	Interruptions autorisées (par passage à 1 de TOF lorsqu'il est mis à 1)

- **Interruption temps réel :**

La fonction interruption temps réel peut être utilisée pour produire des interruptions à intervalle de temps régulier.
On peut, grâce à cela, savoir le temps d'exécution d'un ou d'une partie de programme.
Pour cela, on valide l'interruption à un point particulier du programme et on l'arrête en un autre point.
Elle peut aussi être utilisée pour produire des interruptions à intervalle régulier servant de base de temps. On peut choisir entre 4 rapports de division (registre **PACTL**).
Il ne faut pas oublier, si on utilise la fonction **RTI** de remettre l'indicateur **RTIF** à 0 sinon, des la fin de l'exécution du programme d'interruption, il y a retour dans ce même programme.
4 bits de différents registres seront utilisés dans ce but :
RTII de **TMSK2**
RTIF de **TFLG2**
RTR0 et **RTR1** de **PACTL.**

- **Le registre TFLG2 :**

Reset	0	0	0	0	0	0	0	0	
	TOF	RTIF	PAOVF	PAIF	0	0	0	0	TFLG2 $1025

	Indicateur d'interruption sur PAI
0	Raz par écriture de 1 sur ce bit
1	A 1 lors de la détection d'une transition valide sur PAI

	Indicateur de dépassement
0	Raz par écriture de 1 sur ce bit
1	A 1 lors du débordement du compteur de $FF à $00

	Indicateur d'interruption temps réel
0	Raz par écriture de 1 sur ce bit
1	A 1 lorsqu 'une interruption temps réelle à lieu.

	Indicateur de débordement
0	Raz par écriture de 1 sur ce bit
1	Il passe à 1 à chaque débordement du compteur.

- **Le registre PACTL :**

Dans ce registre, seul les bits **RTR0** et **RTR1** sont utilisés pour les interruptions en temps réel.

Ces deux bits déterminent le rapport de sélection de la fonction **RTI**.

Les autres bits sont utilisés pour la programmation du compteur d'impulsion qui sera vu un peu plus tard dans le cours.

Rythme de l'horloge temps réel	E/2^{13} Divisé par		
4,10ms	1	0	0
8,19ms	2	0	1
16,38ms	4	1	0
32,77ms	8	1	1

Reset		0	0	0	0	0	0	0	0
	DDR7	PAEN	PAMOD	PEDGE	0	0	RTR1	RTR0	PACTL $1026

	Choix du front de comptage et du niveau de commande
0	PAI réagit sur front descendant ou ouvre la porte de comptage avec un niveau bas
1	PAI réagit sur front montant ou ouvre la porte de comptage sur un niveau haut

	Choix du mode
0	L'accumulateur fonctionne en mode comptage sur PAI
1	L'accumulateur fonctionne en mode comptage de l'horloge E/64 sous contrôle du niveau appliqué sur PAI

	Bit de commande de comptage
0	Le comptage est inhibé
1	Le comptage est autorisé

	Bit de direction associé à PA7
0	Entrée
1	Sortie

c. La fonction « chien de garde »:

Le watchdog fonctionne ou non suivant l'état du bit **NOCOP** du registre **CONFIG** .
Sa période est positionnée à son temps le plus court.
On peut venir changer sa période en programmant les bits **CR0** et **CR1** du registre **OPTION**.
Ce dispositif permet de protéger le système contre les défaillances logicielles.
Quand on utilise le watchdog, si une boucle infinie se présente dans le programme, et que le watchdog n'est pas relancé alors une séquence de réinitialisation interne est lancée.

- **Le registre OPTION :**

Reset	0	0	0	1	0	0	0	0
	ADPU	**CSE**	**IRQE**	**DLY**	**CME**	**-**	**CR1**	**CR0**

OPTION $1039

Période	E/2^{15} Divisé par		
16,384ms	1	0	0
65,536ms	4	0	1
262,14ms	16	1	0
1,049s	64	1	1

- **Le registre CONFIG :**

Validation de l'EEPROM	
Les 512 octets d'EEPROM sont inhibés	0
Les 512 octets d'EEPROM sont validés de $B600 à $B7FF	1

Validation de la ROM interne	
Les 8 koctets sont inhibés	0
Les 8 koctets sont validés	1

Reset	0	0	0	0	0	1	1	1	
	0	0	0	0	NOSEC	NOCOP	ROMON	EEON	CONFIG $103F

	Arrêt du watchdog
0	Il est arrêté et ne produit pas de reset
1	Il est validé dés la fin du reset

	Verrouillage de l'EEPROM
0	Cette possibilité n'est offerte que si la ROM à été programmée par usine pour cette usage.
1	

d. Entrées de captures :

On peut choisir la polarité du front qui déclenche la capture et valider ou non les demandes d'interruption.

Si les interruptions sont inhibées, il est nécessaire de lire périodiquement les indicateurs de capture.

Trois registres correspondent à ces trois entrées de capture : **TIC1**, **TIC2** et **TIC3**. Ce sont des registres 16 bits.

Bit15							Bit8	$1010	TIC1
Bit7							Bit0	$1011	

Bit15							Bit8	$1012	TIC2
Bit7							Bit0	$1013	

Bit15							Bit8	$1014	TIC3
Bit7							Bit0	$1015	

L'information lue avec ces registres est l'information correspondante au front qui s'est produit immédiatement avant la lecture et pas forcement celle qui a déclenché l'indicateur de capture.

Les entrées de capture étant indépendantes, il est possible de saisir simultanément trois fronts de déclenchement et d'effectuer des mesures simultanées de délais sur trois signaux différents.

- **<u>les registres TMSK1 et TFLG1 :</u>**

Les indicateurs de capture **ICxF** sont automatiquement mis à 1 à chaque fois que le front sélectionné est détecté sur l'entrée de capture correspondante.

Reset	0	0	0	0	0	0	0	0	
	OC1F	OC2F	OC3F	OC4F	OC5F	IC1F	IC2F	IC3F	TFLG1 $1023

Indicateurs de capture			
Raz en écrivant 1 sur le bit correspondant	0	0	0
A chaque front sélectionné détecté sur l'entrée correspondante	1	1	1

les bits de contrôle **ICxI** permettent à l'utilisateur de configurer chaque entrée en mode d'interruption ou en mode de lecture périodique.

Reset	0	0	0	0	0	0	0	0	
	OC1I	OC2I	OC3I	OC4I	OC5I	IC1I	IC2I	IC3I	TMSK1 $1022

Choix du mode			
Mode périodique	0	0	0
Mode interruption	1	1	1

- **<u>Le registre TCTL2 :</u>**

L'utilisateur peut choisir pour chaque entrée de capture le sens du front de déclenchement.

Reset	0	0	0	0	0	0	0	0	
	0	0	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2 $1021

EDGxB	EDGxA	**Configuration**
0	0	Capture inhibée
0	1	Capture sur fronts montants
1	0	Capture sur fronts descendants
1	1	Capture sur chaque fronts

e. Exemples :

- Mesure d'une période arrivant sur PA0 (IC3):

```
        LDAA    #%00000001
        STAA    TCTL2
        STAA    TFLG1
BCL     LDX     #TFLG1
        BRCLR   0,X,%00000001,BCL
        LDD     TIC3
        STD     $0000
        LDAA    #%00000001
        STAA    TFLG1
BCL1    LDX     TFLG1
        BRCLR   0,X,%00000001,BCL1
        LDD     TIC3
        SUBD    $0000
```

- Mesure de largeur d'impulsions :

On veut mesurer la largeur d'impulsion (temps de niveau haut d'un signal) entre un front montant et un front descendant en examinant l'état de l'indicateur **IC3F** (entrée PA0).

```
        LDAA    #%00000001
        STAA    TCTL2
        STAA    TFLG1
BCL     LDX     #TFLG1
        BRCLR   0,X,%00000001,BCL
        LDD     TIC3
        STD     $0000
        LDAA    #%00000010
        STAA    TCTL2
        LDAA    #%00000001
        STAA    TFLG1
BCL1    LDX     #TFLG1
        BRCLR   0,X,%00000001,BCL1
        LDD     TIC3
        SUBD    $0000
```

f. Sorties de comparaison :

Grâce à ces sorties, on peut programmer des sorties pour la modulation en largeurs d'impulsion par exemple, ou la génération d'impulsions programmables.
Les sorties associées à cette fonction sont au nombre de 5 et peuvent contenir une valeur différente (entrée par le programmeur).

Un drapeau (**OCxF**) est positionné quand la valeur du ou des registres est égale à celle du compteur et on peut ou non associer une interruption à chacun des indicateurs.
Les 5 sorties sont **OC1** à **OC5**.

Pour **OC2** à **OC5** on peut choisir quatre actions au moment de la comparaison (état haut, état bas, aucune action, changement d'état à chaque comparaison) en programmant les bits **OMx** et **OLx**.

OC1 est particulier : il peut commander directement les 5 sorties de comparaison. Les registres de contrôle et de données associées à cette fonction sont **OC1M** et **OC1D**.
Il est possible de commander les actions sur une seule sortie à partir du comparateur **OC1** et du comparateur associé à cette sortie.

- **Les registres de comparaisons TOC1 à TOC5 :**

Bit15							Bit8	$1016	TOC1
Bit7							Bit0	$1017	
Bit15							Bit8	$1018	TOC2
Bit7							Bit0	$1019	
Bit15							Bit8	$101A	TOC3
Bit7							Bit0	$101B	
Bit15							Bit8	$101C	TOC4
Bit7							Bit0	$101D	
Bit15							Bit8	$101E	TOC5
Bit7							Bit0	$101F	

Ces registres 16 bits peuvent être lu ou écrit.
On utilisera donc l'instruction LDD pour lire ces registres.

- **le registre TMSK1 :**

Reset	0	0	0	0	0	0	0	0	
	OC1I	OC2I	OC3I	OC4I	OC5I	IC1I	IC2I	IC3I	TMSK1 $1022

					Validation des interruptions
0	0	0	0	0	Les interruptions sont masquées
1	1	1	1	1	Les interruptions sont validées

- **le registre TFLG1 :**

Reset	0	0	0	0	0	0	0	0	
	OC1F	OC2F	OC3F	OC4F	OC5F	IC1F	IC2F	IC3F	TFLG1 $1023

					Indicateur de comparaison
0	0	0	0	0	Raz en écrivant 1 sur le bit correspondant
1	1	1	1	1	A 1 quand le contenu du registre identique au compteur

- **le registre TCTL1 :**

On peut programmer indépendamment l'action qui apparaît sur chacune des sorties **PA3** à **PA6**.

Reset	0	0	0	0	0	0	0	0	
	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1 $1020

OMx	OLx	**Action**
0	0	Aucune action
0	1	La sortie change d'état à chaque comparaison
1	0	La sortie passe à 0 à chaque comparaison
1	1	La sortie passe à 1 à chaque comparaison

Exemple :

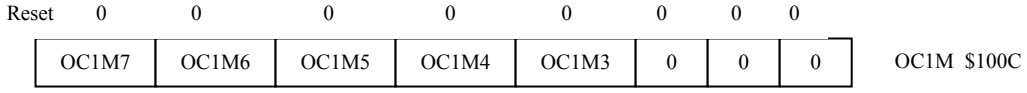
Fabrication d'un générateur de signaux carrés à fréquence fixe :

```
TCTL1   EQU   $1020
debut   LDAA  #%00000001  ;  OM5 = 0 et OL5 = 1
        STAA  TCTL1
        BRA   debut
        END
```

La sortie PA3 est programmée pour changer d'état à chaque comparaison.
Le registre de comparaison est à $FFFF après le reset.
Donc on aura un changement d'état à chaque comparaison donc toutes les 32,77 ms pour un quartz à 8 MHz.

- **Le registre OC1M :**

Le comparateur **OC1** occupe une place à part.
Il permet de contrôler une ou plusieurs autres sorties de comparaison.
Les sorties affectées par la fonction de comparaison **OC1** sont repérées en mettant à 1 les bits **OC1Mx** correspondant.



OC1Mx	**Sélection des sorties**
0	Sortie non sélectionnée
1	Sortie sélectionnée

- **Le registre OC1D :**

Reset	0	0	0	0	0	0	0	0
	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	0	0	0

OC1D $100D

OC1Dx	**Etat des sorties**
0	Sortie à 0
1	Sortie à 1

L'état pris par la sortie au moment de la comparaison avec **OC1** est indiqué par **OC1Dx**.

Exemple :

On veut réaliser une horloge multiphase sur les sorties **PA3** à **PA7**.

```
            LDAA    #$80
            LDAA    PACTL
COMP :
            LDY     #DEBTAB
            LDX     #TFLG1
            LDAA    #$F8
            STAA    OC1M
RAZ :
            LDAA    #OC1F
            STAA    TFLG1
            LDAA    0,X
            INY
            CPY     #FINTAB
            BNE     SUITE
            LDY     #DEBTAB
SUITE :
            STAA    OC1D
            LDD     TOC1
            ADDD    #20000
            STD     TOC1
BASCUL :
            BRCLR   0,X,OC1F,BASCUL
            BRA     RAZ
DEBTAB :
            FCB     00,$80,00,$10,00,$20,00,$40,00,$80
FINTAB      EQU     *
            END
```

- **Le registre CFORC :**

Il est possible de forcer, par programme, l'état d'une sortie sans avoir besoin d'attendre le résultat d'une comparaison.

Reset	0	0	0	0	0	0	0	0	
	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	CFORC $100B

FOCx	**Forçage des sorties**
0	Sortie normale
1	Comparaison sortie