



1.1. STRUCTURE INTERNE  
1.2 STRUCTURE INTERNE HC11EX  
1.2. STRUCTURE INTERNE HC11F1

### 3. CÂBLAGE DE BASE EN MODE ÉTENDU

- 4.1. GÉNÉRALITÉS
- 4.2. REPRÉSENTATIONS DES CHIFFRES:
- 4.3. LES REGISTRES
- 4.4. MODES D'ADRESSAGE
- 4.5. JEU D'INSTRUCTIONS
- 4.6. INSTRUCTIONS PARTICULIÈRES DU MICROCONTROLEUR :
- 4.7. LA PILE SYSTÈME S
- 4.8. RESET ET INTERRUPTIONS
- 4.9. LES ASSEMBLEURS, SYNTAXE :
- 4.10. EXEMPLE DES DIRECTIVES AS11
- 4.11. REGISTRES INTERNES 68HC11EX
- 4.12. REGISTRES INTERNES 68HC11

- 5.1. PORTS PARALLÈLES DIGITAUX
- 5.2. PORTS ANALOGIQUES
- 5.3. TIMER
- 5.4. GÉNÉRATEUR D'INTERRUPTION PÉRIODIQUE
- 5.5. TIMERS : MESURES ET PRODUCTION DE TEMPS
- 5.5.1. COMPARAISON EN ENTRÉE :
- 5.5.2. COMPARAISON EN SORTIE :
- 5.6. CHIEN DE GARDE ET SURVEILLANCE DE L'HORLOGE
- 5.7. ACCUMULATEUR D'IMPULSIONS
- 5.8. PORTS DE COMMUNICATION DE TYPE SPI
- 5.9. EEPROM
- 5.10. PORTS DE COMMUNICATION SÉRIE ASYNCHRONE(UART)

- INITIALISATION
- 6.1. PROGRAMMATION DES PORTS PARALLÈLES
- 6.2. UTILISATION DE RTI
- 6.3. UTILISATION DU CHIEN DE GARDE
- 6.4. PROGRAMMATION DU TIMER EN ENTRÉE, MESURE DE PÉRIODES
- 6.5. PROGRAMMATION DU TIMER EN ENTRÉE, MESURE DE DURÉE
- 6.6. PROGRAMMATION DU TIMER EN SORTIE -MODE ASTABLE
- 6.7. PROGRAMMATION DE L'ACCUMULATEUR D'IMPULSION
- 6.8. PROGRAMMATION DU SPI
- 6.9. PROGRAMMATION DU SCI
- 6.10. PROGRAMMATION DES PORTS ANALOGIQUES
- 6.11. PROGRAMMATION DE L'EEPROM INTERNE
- EXERCICES

Schémas à Motorola MC68HC11F1, MC68HC11E9, USER MANUAL 68HC11

# 1. Description de la gamme

## 1.1. Structure interne

Le micro-contrôleur Motorola 68HC11 peut fonctionner avec des horloges allant jusqu'à 12MHz. Tous les registres étant statiques, une coupure d'horloge n'entraîne pas de perte de donnée. Le 68HC11 intègre de puissants périphériques :



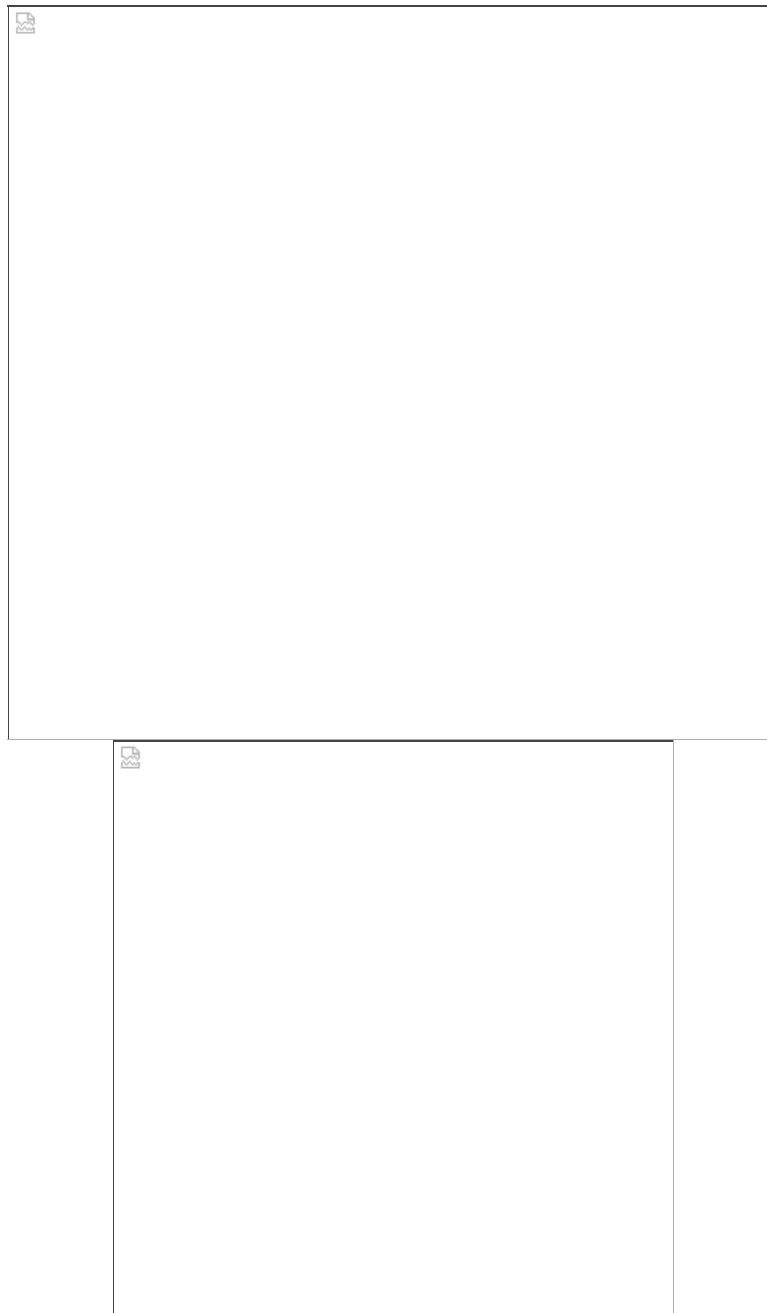
- Ports parallèles
- Port de communication série asynchrone
- Port de communication série synchrone
- Ports analogiques
- Timers très complets
- Chien de garde
- Génération d'interruptions temps réel
- Jusqu'à 12KO de ROM ou d'EPROM
- Jusqu'à 1KOde RAM
- Jusqu'à 8KO d'EEPROM

Le 68HC11 est disponible suivant les version en boîtier DIP ou PLCC. Le modèle fonctionnel du 68HC11Ex est donné ci-dessous. Des blocs fonctionnels peuvent être différents ou absents dans certaines versions



## 1.2 Structure interne HC11Ex

## 1.2. Structure interne HC11F1



## 2. Description des broches

- Ü **VDD/VSS** : Alimentation 5v/0v, consommation de 85 mW en mode single ship avec quartz 4MHz à 195 mW en mode étendu-multipléxé avec quartz 12MHz
- Ü **MODB/VSTBY et MODA/LIR** : Durant la phase de RESET ces broches définissent le mode de fonctionnement suivant le tableau ci-dessous. les quatre bits de poids fort du registre HPRIO sont alors initialisés comme suit :

Niveaux lors du RESET		Mode	Bits de control du registre HPRIO		
MODB	MODA		RBOOT	SDMOD	MDA
1	0	UC seul	0	0	0
1	1	Etendu	0	0	1
0	0	Bootstrap	1	1	0
0	1	Special test	0	1	1

- Ü Configuration mémoire pour un 68HC11E9.



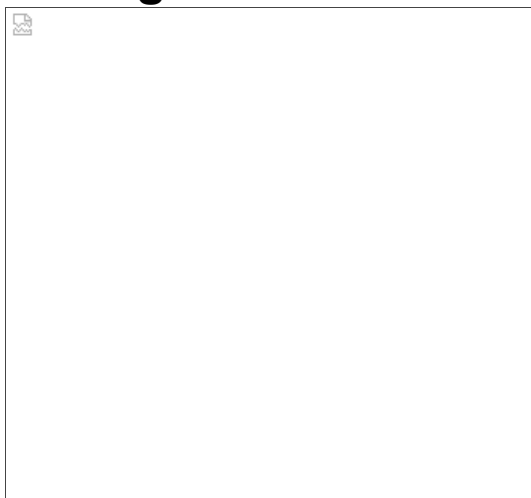
## Ü Configuration mémoire pour un 68HC11F1



- ü **Single-Ship** : Fonctionnement autonome, tous les ports du micro-contrôleur sont disponibles, par contre la mémoire est limitée à la capacité interne
- ü **Expended multiplexed** : Ce mode permet d'étendre la capacité mémoire ainsi que les périphériques par l'adjonction de boîtiers externes. L'ensemble devient plus puissant mais le matériel est plus complexe et deux ports 8 bits sont perdus sur le micro-contrôleur
- ü **Spécial Bootstap** : Lors du RESET un logiciel interne appelé BOOTLOADER télécharge automatiquement en RAM un programme provenant de la liaison série asynchrone (SCI) et exécute celui-ci. Ce mode est utilisé pour stocker à distance des valeurs de consigne (pour une régulation par exemple) ou pour le développement (utilisé par CBOY)
- ü **Spécial Test** : Destiné au départ aux tests de production de Motorola, ce mode peut être utilisé pour le développement en particulier pour l'émulation, il permet entre autre de modifier le registre CONFIG après le RESET

- ⌚ **EXTAL/XTAL** : Entrées pour oscillateur à quartz Pour des fréquences de quartz supérieures à 1MHz on utilisera le câblage suivant. Pour des valeurs de fréquence inférieures les valeurs des composants passifs devront être déterminés expérimentalement (Reference Manual page 2.11 à 2.15)  
**Rf** = 1M à 20M ( Les fortes valeurs sont sensibles à l'humidité et les faibles valeurs réduisent le gain de l'oscillateur )  
**C1** = 5pF à 25pF (Valeur fixe)  
**C2** = 5pF à 25pF (Valeur pouvant être variable pour ajuster la fréquence de l'oscillateur)
- ⌚ **RESET** : Cette broche à deux fonctions.
  - L'application d'un 0 logique provoque l'initialisation du micro-contrôleur ,**l'information est en entrée**
  - Lors d'un problème de fonctionnement (Détection d'un défaut d'horloge, activation du chien de garde) la broche RESET transmet un niveau logique 0 (pour initialiser des périphériques par exemple ou un micro-contrôleur esclave) **L'information est en sortie.***Un simple circuit RC ne suffit pas.(voir manuel de référence)*
- ⌚ **E** : Sortie de l'horloge interne de fréquence  $FE = F_{Quartz}/4$
- ⌚ **IRQ** : Entrée d'interruption, peut être configurée par le registre OPTION lors de l'initialisation pour être active sur front descendant ou sur niveau bas . (Défaut : niveau bas) .
- ⌚ **XIRQ**: Entrée d'interruption active sur niveau bas. Cette interruption ne peut être démasquée qu'une fois (elle devient ensuite non-masquable). Elle sert également pour la tension  $V_{pp}$  lors de la programmation de l'EPROM interne.
- ⌚ **VREFL, VREFH** : (*uniquement sur les modèle équipé de ports analogiques*) Ces entrées donnent les valeurs de référence du convertisseur analogique numérique.

### 3. Câblage de base en mode étendu



**Note:** En mode Single Ship AS et RW suivant le type de micro contrôleur sont les lignes STRC et STRB (STROBE) des ports parallèles ou les bits D7/D6 du port D

- ⌚ **AS** : Adresse Strobe. Actif au niveau haut ce signal indique que les adresses basses sont valident et doivent être mémorisées.
- ⌚ **RW** : A l'état bas indique une opération d'écriture, à l'état haut une opération de lecture

## 4. Programmation de l'unité centrale

### 4.1. Généralités

Le 68HC11 possède une unité centrale 8/16Bits. Le 68HC11 reconnaît des pour certaines instructions des mots de 16 bits, c'est un uC 8/16 bits.

### 4.2. Représentations des chiffres:

- ü Nombres entiers non signés (Unsigned Integer): Un octet peut avoir des valeurs entre 0 et 255 (\$FF), un mot entre 0 et 65535 (\$FFFF).
- ü Nombres entiers signés complément à 2 (Two's complement). Un octet peut avoir des valeurs entre -128 (\$80) et +127 (\$7F), un mot entre -32768 (\$8000) et 32767 (\$7FFF). Le bit de poids fort indique toujours le signe.
- ü Décimal codé binaire (BCD Binary Coded Decimal): Un octet contient deux chiffres décimaux. Les bits 7,6,5,4 contiennent le chiffre de poids fort, les bits 3,2,1,0 contiennent le chiffre de poids faible. Un octet peut donc avoir des valeurs entre 0 (\$00) et 99 (\$99). Cette présentation est peu utilisée.
- ü Les adresses ont 16 bits et adressent des octets. L'espace d'adressage de l'unité centrale comprend 65536 octets (\$0000..\$FFFF).

### 4.3. Les registres

#### Unité centrale 68HC11

7 0	<b>A</b>	7 0	<b>B</b>	Accumulateurs A et B 8bits
15 0		<b>D</b> 0		Accumulateur A+B=D 16 bits
		<b>IX</b>		Registre d'index X
		<b>IY</b>		Registre d'index Y
		<b>SP</b>		Pointeur pile système
		<b>PC</b>		Compteur de programme
		<b>CCR</b>		Registre code condition

ü ACCA et ACCB sont les deux accumulateurs ou registres centraux A et B. Chaque registre comprends 8 bits. le registre double D a 16 bits et contient les deux registres A et B.

ldaa #\$12 charge la valeur \$12 dans le registre A  
ldab #\$34 charge la valeur \$34 dans le registre B  
idd #\$1234 charge \$12 dans le registre A et \$34 dans B

ü Les registres IX et IY servent comme registres d'index pour adresser la mémoire par exemple

ldaa \$1033 Charge le registre d'E/S  
staa \$1034 Enregistre dans le registre d'E/S

Au lieu de cette séquence on peut écrire

ldx #\$1000 Le registre IX adresse maintenant l'espace d'E/S  
ldaa \$33,x Charge le registre d'E/S  
staa \$34,x Enregistre dans le registre d'E/S

ü Le pointeur de pile SP adresse toujours le premier octet, qui n'est pas encore réservé par la pile.

ü Le registre d'instructions PC adresse l'instruction à exécuter.

ü Registre CCR : Codes conditions

Le registre des codes conditions CCR contient huit bits. Beaucoup d'opérations changent ces bits. Quelques opérations changent leur comportement selon ces bits. Par exemple

decB diminue de 1 le registre B  
change le CCR bits N, Z et V  
beq branche si Z=1, ça veut dire si le registre B est égal à 0

Les 8 bits du registre CCR ont les significations suivantes:

- ü **C(0)** Carry, Retenue: Le résultat de la dernière opération a causé une retenue. Le bit permet des opérations sur des opérandes plus longues que 16 bits.
- ü **V(1)** Overflow, Débordement: Le résultat de la dernière opération a causé un débordement. Seulement pour des nombres entiers signés complément à 2.
- ü **Z(2)** Zéro: Le résultat de la dernière opération est zéro. Après une comparaison, zéro indique que les deux opérandes étaient égaux.
- ü **N(3)** Négatif: Le résultat de la dernière opération est négatif. Seulement pour des nombres entiers signés complément à 2.
- ü **I(4)** Masque d'interruption: **Quand le bit est mis à 1, les interruptions sont ignorées.**
- ü **H(5)** Half Carry, Demi-retenue: Ce bit permet des opérations sur des opérandes en présentation décimal codé binaire.
- ü **X(6)** Masque d'interruption: **Quand le bit est mis à 1, les interruptions XIRQ sont ignorées.**
- ü **S(7)** Ignorer Stop: **Quand le bit est mis à 1, l'instruction STOP est ignorée.**

### 4.4. Modes d'adressage

ü **Inhérent** : Il n'y a **pas d'opérande** ex : INCA, LSLA, RTI, RTS, RORA, etc...

ü **Immédiat**.. L'opérande se trouve directement dans le programme derrière le code de l'instruction. Au niveau d'assemblage, ce mode d'adressage s'exprime avec le symbole dièse (#).

86 7F ldaa #127 charge 127 dans ACCA  
8B 10 adda #\$10 additionne 16  
CE 10 00 idx #\$1000 charge l'adresse \$1000 dans le registre IX

ü **Etendu** 16 bits: l'adresse de l'opérande se trouve dans les deux octets suivants le code de l'instruction. L'opérande peut être un octet ou un mot de 16 bits. Ce mode d'adressage s'exprime avec le symbole supérieur (>). *Il est toujours pris par défaut par les assembleurs le symbole (>) est donc facultatif*

B6 10 33    ldaa    >\$1033    charge un octet de l'adresse \$1033  
 FF 11 00                    stx            >\$1100    enregistre le registre IX à l'adresse \$1100

Ü **Direct** 8 bits: l'adresse de l'opérande se trouve dans l'octet suivant le code de l'instruction. L'opérande peut être un octet ou un mot de 16 bits. L'adressage permet d'adresser les octets aux adresses \$0000..\$00FF, donc la mémoire vive. Ce mode d'adressage s'exprime avec le symbole inférieur (<). *(La plupart des assembleurs utilise automatiquement ce mode pour les adresses \$0000 à \$00FF)*

96 33            ldaa    <\$0033    charge un octet de l'adresse \$0033  
 DF 80            stx    <\$0080    enregistre le registre IX à l'adresse \$0080:0081

Ü **Relatif** PC: Cet adressage est réservé aux instructions de branchement. L'adresse du branchement est calculée par l'adresse de l'instruction diminué jusqu'à -128 ou augmenté jusqu'à +127 par l'octet qui se trouve derrière le code de l'instruction. *Les adresses sont données par des étiquettes, l'assembleur se chargeant de calculer les distances relatives.*

1F 08 20 FC    brclr    \$08,x \$20 \*    attend que le bit \$20 atteigne la valeur 1  
 Si la distance est trop grande (>128) il faut utiliser une instruction de saut jmp. *(déconseillée car le programme ne sera plus relogable)*  
 2C ??            bge    troploin    branche si plus grand ou égal


remplacé par  
 2D 03            blt    L            branche si plus petit  
 7E FA 00        jmp    troploin    saute si plus grand ou égal  
                  L    equ    \*


Ü **Indexé** (IX, IY) : L'adresse est calculée par le contenu du registre IX ou IY en ajoutant une valeur entre \$00 et \$FF qui se trouve dans l'octet suivant le code de l'instruction. Le résultat est sur 16 bits et adresse donc la totalité d'espace d'adressage. Cet adressage est variable et chaque sous-programme qui veut adresser n'importe quel adresse doit s'en servir. Le programme suivant efface la mémoire vive de l'adresse \$0040 à \$004F


CE 00 40        idx    #\$0040    Commence à l'adresse \$0040  
 86 10            ldaa    #16        ACCA compteur: 16 octets  
 6F 00        L:    clr    0,x        Efface l'octet à l'adresse dans IX  
 08            inx            Augmente de 1 l'adresse  
 4A            deca          Diminue de 1 le compteur  
 26 FA        bne    L            Continue si c'est pas encore fini

Ü **Différences entre les adressages : immédiat, direct, étendu, indexé :**  
 Exemple : charger la donnée \$4A dans l'accumulateur A (adresse de l'exemple : \$E000)  
**La donnée \$4A se trouve également à l'adresse \$0040**

**Immédiat :**  
 \$E000 86 4A            LDAA    #\$4A   

**Etendu :**  
 \$E000 B6 00 40        LDAA    \$0040    

**Direct :**  
 \$E000 96 40            LDAA    >\$40    

**Indexé : (IX=\$0040)**  
 \$E000 A6 00            LDAA    0,X    



## 4.5. Jeu d'instructions

### Charger et Enregistrer

<b>LDAA/B</b>	#, dir, ext, ind	NZV	ACCx = M		
<b>LDD/S/X/Y</b>	##, dir, ext, ind	NZV	ACCD/SP/IX/IY = M:M+1		
<b>PSHA/B/X/Y</b>		-	push A/B/IX/IY		
<b>PULA/B/X/Y</b>		-	pop A/B/IX/IY		
<b>STAA/B</b>	dir, ext, ind	NZV	M = ACCx		
<b>STD/S/X/Y</b>	dir, ext, ind	NZV	M:M+1 = ACCD/SP/IX/IY		
<b>TAB</b>		NZV	ACCB = ACCA		
<b>TAP</b>		tous	CCR = ACCA		
<b>TBA</b>				NZV	ACCA = ACCB
<b>TPA</b>	-		ACCA = CCR		
<b>TSX/Y</b>	-		IX/IY = SP+1		
<b>TXS/TYS</b>	-		SP = IX/IY - 1		
<b>XGDX/Y</b>	-		ACCD <=> IX/IY		

### Arithmétique

<b>ABA</b>		HNZVC	ACCA += ACCB		
<b>ADCA/B</b>	#, dir, ext, ind	HNZVC	ACCx += M + C		
<b>ADDA/B</b>	#, dir, ext, ind	HNZVC	ACCx += M		
<b>ADDD</b>	##, dir, ext, ind	NZVC	ACCD += M:M+1		
<b>ANDA/B</b>	#, dir, ext, ind	NZV	ACCx &= M		
<b>CBA</b>		NZVC	Comparer: ACCA-ACCB		
<b>CLRA/B/m</b>	ext, ind	NZVC	ACCx/M = 0		
<b>CMPA/B</b>	#, dir, ext, ind	NZVC	ACCx - M		
<b>COMA/B/m</b>	ext, ind	NZVC	ACCx/M = ~ ACCx/M		
<b>CPD</b>	##, dir, ext, ind	NZVC	ACCD - M:M+1		
<b>DAA</b>		NZVC	régler ACCA pour BCD		
<b>DECA/B/m</b>	ext, ind	NZV	ACCx/M --		
<b>EORA/B</b>	#, dir, ext, ind	NZV	ACCx ^= M ou exclusif		
<b>FDIV</b>		ZVC	IX,ACCD = ACCD/IX fractional		
<b>IDIV</b>		ZVC	IX,ACCD = ACCD/IX non signé		
<b>INCA/B/m</b>	ext, ind	NZV	ACCx/M ++		
<b>MUL</b>		C	ACCD = ACCA * ACCB		
<b>NEGA/B/m</b>	ext, ind	NZVC	ACCx/M = 0 - ACCx/M		
<b>ORAA/B</b>	#, dir, ext, ind	NZV	ACCx  = M		
<b>SBA</b>		NZVC	ACCA -= ACCB		
<b>SBCA/B</b>	#, dir, ext, ind	NZVC	ACCx -= M + C		
<b>SUBA/B</b>	#, dir, ext, ind	NZVC	ACCx -= M		
<b>SUBD</b>	##, dir, ext, ind	NZVD	ACCD -= M		
<b>TSTA/B/m</b>	ext, ind	NZVC	ACCx/M - 0		

### Décalage, Rotation

<b>ASLA/B/m</b>	ext, ind	NZVC	ACCx/M arithm. shift left 1 bit		
<b>ASLD</b>		NZVC	ACCD arithm. shift left double 1 bit		
<b>ASRA/B/m</b>	ext, ind	NZVC	ACCx/M arithm. shift right 1 bit		
<b>LSLA/B/m</b>	ext, ind	NZVC	ACCx/M shift left 1 bit		
<b>LSLD</b>		NZVC	ACCD shift left double 1 bit		
<b>LSRA/B/m</b>	ext, ind	NZVC	ACCx/M shift right 1 bit; bit7=0		
<b>LSRD</b>		NZVC	ACCD shift right 1 bit; bit15=0		
<b>ROLA/B/m</b>	ext, ind	NZVC	ACCx/M rotate left 1 bit thru Carry		
<b>RORA/B/m</b>	ext, ind	NZVC	ACCx/M rotate right 1 bit thru Carry		

### Opérations sur bits

<b>BCLR</b>	dir, ind mask	NZV	M &= ~mask		
<b>BSET</b>	dir, ind mask	NZV	M  = mask		
<b>BITA/B</b>	#, dir, ext, ind	NZV	Bit test: ACCx & M		

### Registres d'index

<b>ABX/Y</b>			IX/Y += ACCB		
<b>CPX/Y</b>	##, dir, ext, ind	NZVC	Comparer IX/Y - M:M+1		
<b>DES</b>		-	SP --		
<b>DEX/Y</b>		Z	IX/Y --		
<b>INS</b>		-	SP++		
<b>INX/Y</b>		Z	IX/Y++		

### Branchement, Contrôle

Branche selon la condition, CCR non affecté

20 xx	<b>BRA</b>	<-128..+127>	toujours
21 xx	<b>BRN</b>	<-128..+127>	jamais
24 xx	<b>BCC</b>	<-128..+127>	C du CCR à 0
25 xx	<b>BCS</b>	<-128..+127>	C du CCR à 1
27 xx	<b>BEQ</b>	<-128..+127>	Z du CCR à 1
2B xx	<b>BMI</b>	<-128..+127>	N du CCR à 1
26 xx	<b>BNE</b>	<-128..+127>	Z du CCR à 0
2A xx	<b>BPL</b>	<-128..+127>	N du CCR à 0

#### Branche si

#### Après une comparaison des nombres signés

2C xx	<b>BGE</b>	<-128..+127>	1.opérande >= 2.opérande
2E xx	<b>BGT</b>	<-128..+127>	1.opérande > 2.opérande
2F xx	<b>BLE</b>	<-128..+127>	1.opérande <= 2.opérande

	2D xx	<b>BLT</b>	<-128..+127>	1.opérande < 2.opérande
<b>Après une comparaison des nombres non signés</b>				
	22 xx	<b>BHI</b>	<-128..+127>	1.opérande > 2.opérande
	24 xx	<b>BHS</b>	<-128..+127>	1.opérande >= 2.opérande
	25 xx	<b>BLO</b>	<-128..+127>	1.opérande < 2.opérande
	23 xx	<b>BLS</b>	<-128..+127>	1.opérande <= 2.opérande
<b>Après des opérations avec des nombres signés</b>				
	28 xx	<b>BVC</b>	<-128..+127>	Débordement à 0
	29 xx	<b>BVS</b>	<-128..+127>	Débordement à 1
<b>BRA</b>	rel	-		Brancher
<b>BSR</b>	rel	-		Brancher au sous-programme
<b>BRCLR</b>	dir, ind mask	rel -		Brancher si les bits sont à 0
<b>BRSET</b>	dir, ind mask	rel -		Brancher si les bits sont à 1
<b>JMP</b>	ext, ind	-		Sauter
<b>JSR</b>	dir, ext, ind	-		Sauter au sous-programme
<b>RTI</b>		tous		Retour d'interruption
<b>RTS</b>		-		Retour de sous-programme
<b>STOP</b>		-		
<b>SWI</b>		I		Interruption logicielle
<b>WAI</b>		-		Attendre une interruption

### Opérations sur CCR

<b>CLC//V</b>	CIV	Mettre à 0 Carry / Int / Overflow
<b>SEC//V</b>	CIV	Mettre à 1 Carry / Int / Overflow

### Opérandes

<b>#</b>	<b>#&lt;\$00..\$FF&gt;</b>	imédiat 8 bits
<b>#</b>	<b>#&lt;\$00..\$FFFF&gt;</b>	imédiat 16 bits
<b>dir</b>	<b>&lt;\$00..\$FF&gt;</b>	adressage direct 8 bits
<b>ext</b>	<b>&lt;\$0000..\$FFFF&gt;</b>	adressage direct 16 bits
<b>ind</b>	<b>&lt;\$00..\$FF&gt;,X</b>	indexé par IX
	<b>&lt;\$00..\$FF&gt;,Y</b>	indexé par IY
<b>rel</b>	<b>&lt;-128..+127&gt;</b>	relatif PC
<b>mask</b>	<b>&lt;\$00..\$FF&gt;</b>	Masque, 2. Opérande

#### 4.6. Instructions particulières du microcontrôleur :

**BSET et BCLR, positionnent les bits d'une adresse qui peut être celle d'un port externe**

BSET PRC,X %0110000 met à un les bits 5 et 6 du port C  
BCLR PRC,X %0110000 met à zéro les bits 5 et 6 du port C

**BRSET, BRCLR, effectuer un branchement (-128,+127) si les bits spécifiés sont bien positionnés**

BRSET PRC,X %01100000 LABAS effectue un branchement vers l'adresse LABAS si les bits 4 et 5 du port C sont à 1.  
BRCLR PRC,X %01100000 LABAS effectue un branchement vers l'adresse LABAS si les bits 4 et 5 du port C sont à 0.

#### 4.7. La pile système S

Ce registre, qui doit être initialisé en RAM avant tout appel de sous programme ou de déclenchement d'interruption permet de récupérer le « contexte » lors d'un retour de sous programme.

**Exemple : S=\$00FF (sommet de la RAM de la plupart des 68HC11)**

Lors de l'instruction BSR LABAS le compteur de programme est chargé avec l'adresse LABAS et exécute le sous programme s'y trouvant. Lorsque le CPU rencontre l'instruction RTS. Le PC est rechargé avec l'adresse qui suit celle de l'instruction BSR LABAS

Le PC est en \$E000 et LABAS = \$E0A4

\$E000 8D A4 BSR LABAS  
\$E002 86 01 LDAA #\$01

Sous programme LABAS

**PILE S**



Zone de Texte: Données  
AdressesRetLSPRetH  
SP-11YLSP-2IYH  
SP-3IXLSP-4IXH  
SP-5AccASP-6AccB  
SP-7CCRSP-8?SP-9?  
SP-10

#### Le cas des interruptions :

Les interruptions sont asynchrones, le microcontrôleur doit donc sauvegarder dans la pile non seulement l'adresse de retour mais également tout le contexte comme suit :

Lors du déclenchement d'une interruption, le contexte est automatiquement sauvé dans la pile système S (SP), voir ci contre ,puis le PC est chargé avec le contenu du vecteur d'interruption (voir doc microcontrôleur pour connaître les adresses)

## 4.8. RESET et interruptions

La table des vecteurs d'interruption est donnée dans la documentation du modèle de 68HC11 utilisé.

- **RESETS :**
- Par une mise à 0 de la broche RESET
  - Par un dépassement de la durée du chien de garde
  - Par une détection de défaut d'horloge
  - Par une détection de code instruction illégal (utilisé lors du développement)

Le choix de l'interruption masquable la plus prioritaire peut être programmé grâce aux bits 0 à 3 (PSEL à PSEL3) de HPRIO.

Adresses des vecteurs	Sources des interruptions	Masque du CCR	Masque local
FFC0, C1–FFD4, D5	Reserved	—	—
FFD6, D7	SCI Serial System <div style="display: flex; flex-direction: column; align-items: flex-start;"> <div>0</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> </div> SCI Receive Data Register Full SCI Receiver Overrun SCI Transmit Data Register Empty SCI Transmit Complete SCI Idle Line Detect	I	RIE RIE TIE TCIE ILIE
FFD8, D9	SPI Serial Transfer Complete	I	SPIE
FFDA, DB	Pulse Accumulator Input Edge	I	PAI
FFDC, DD	Pulse Accumulator Overflow	I	PAOVI
FFDE, DF	Timer Overflow	I	TOI
FFE0, E1	Timer Input Capture 4 / Output Compare 5	I	I4/O5I
FFE2, E3	Timer Output Compare 4	I	OC4I
FFE4, E5	Timer Output Compare 3	I	OC3I
FFE6, E7	Timer Output Compare 2	I	OC2I
FFE8, E9	Timer Output Compare 1	I	OC1I
FFEA, EB	Timer Input Capture 3	I	IC3I
FFEC, ED	Timer Input Capture 2	I	IC2I
FFEE, EF	Timer Input Capture 1	I	IC1I
FFF0, F1	Real-Time Interrupt	I	RTI
FFF2, F3	IRQ (External Pin)	I	None
FFF4, F5	XIRQ Pin	X	None
FFF6, F7	Software Interrupt	None	None
FFF8, F9	Illegal Opcode Trap	None	None
FFFA, FB	COP Failure	None	NOCOP
FFFC, FD	Clock Monitor Fail	None	CME
FFFE, FF	RESET	None	None

S00B0000020202020202020F4  
S121F0008606973F8E00FF86039707860197034F97409740B7004018CE03E88D3A73B8  
S121F01E00037A004020F2860F860586FB860F86098606860086FF860F0C1241000CA5  
S121F03K00120041436563962065737420756E20746578744652041534349293686DB12  
S113F05A4A010126F9180926F73201010101393B4F  
S113FFF0F069F069F069F069F069F069F0009E  
S903F0000C

org	<valeur>	changer l'adresse d'assemblage
fcbl	<valeur>[, <valeur>]	stocker des octets en mémoire
fcbl	'<chaîne ASCII>'	stocker des octets en mémoire
fdb	<valeur>[, <valeur>]	stocker des mots de 16 bit
rmb	<valeur>	réserver octets en mémoire
<nom> equl	= <valeur>	déclarer un symbole
end		fin du programme

'	caractère ASCII	
\$	numéro hexadécimal ou 0x	numéro hexadécimal
%	numéro binaire	
	Sinon c'est une valeur décimale.	

**etiquette[:]**      **opération**      **[opérandes]**      **[\*][commentaire]**

#### 4.10. Exemple des directives AS11

```

; Démonstration des directives de l'assembleur AS11
; C.D 10/1997
; CLIGNOTEMENT DU PORTC
;
; EQUIVALENCES GLOBALES :
;0420 SCONF EQU $0420 ;REGISTRE DE CONFIG DU X68C75
;0408 PORTC EQU $0408 ;REGISTRE DE SORTIE DU PORT C DU X68C75
;0040 RAM EQU $40
;F000 ROM EQU $F000
;1000 REGBASE EQU $1000 ;Base des registres
;
; ZONE RAM DU 68HC11Ex
;0040 ORG RAM
;0040 MEM RMB 10
;004A MEM2 EQU *
;
; DEBUT DE LA ZONE EPROM
;F000 ORG ROM ;DEBUT PROG EN DEBUT EPROM
;
; INITIALISATION DU MICRO-CONTROLEUR ET DES PERIPHERIQUES
;F000 0E 2 RESET CLI ; le moniteur reste actif
;F001 CE1000 3 LDX #REGBASE
;F004 8644 2 LDAA #%01000100
;F006 B70420 4 STAA SCONF ; Port C en sortie
;F009 8655 2 LDAA #%01010101 ;
;F00B B70408 4 STAA PORTC
;F00E 4F 2 CLRA
;F00F 9740 3 STAA MEM
;
; PROGRAMME PRINCIPAL
;F011 PP EQU *
;F011 18CE03E8 4 LDY #1000
;F015 8D33 6 BSR TMP1MS ;ATTENDRE 1S
;F017 730408 6 COM PORTC ;LA LED CHANGE D'ETAT
;F01A 7A0040 6 DEC MEM
;F01D 20F2 3 BRA PP
;
; Démonstration des autres directives
;F01F 860F 2 LDAA #10+5 ;Addition
;F021 8605 2 LDAA #12-7 ;Soustraction
;F023 86FB 2 LDAA #7-12 ;Soustraction
;F025 860F 2 LDAA #5*3 ;Multiplication
;F027 8609 2 LDAA #27/3 ;Division entière
;F029 8606 2 LDAA #27/4 ;Division
;F02B 4365636920 FCB 'Ceci est un texte ASCII'
;F042 0C1241 FCB 12,$12,'A' ;Déclarations de caractères
;F045 000C001241 FDB 12,$12,'A'
;
; FIN DU PROGRAMME PRINCIPAL
;
; Lien avec la librairie TMP1MS.A11 qui contient le sous
; programme TMP1MS
;
#include TMP1MS.A11
;
; RTI - TRAITEMENT DES INTERRUPTIONS :
; AUCUNE INTERRUPTION UTILISEE
; RETOUR PAR RTI SUR INTERRUPTION ALEATOIRE
;
;F05C 3B 12 ADRTI RTI
;
; FIN DE ROUTINES
; VECTEURS D'INTERRUPTIONS GENERALES (16 OCTETS EPROM)
;
;FFF0 ORG $FFF0 ;
;FFF0 F05C TIMVECT FDB ADRTI ;INT PERIODIQUE
;FFF2 F05C IRQVECT FDB ADRTI ;INT BROCHE IRQ
;FFF4 F05C XRQVECT FDB ADRTI ;INT BROCHE XRQ
;
END

```

#### 4.11. Registres internes 68HC11Ex

Adress	Bit 7	6	5	4	3	2	1	Bit 0		
\$1000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PORTA	
\$1001	Reserved									
\$1002	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	PIOC	
\$1003	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PORTC	
\$1004	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PORTB	
\$1005	PCL7	PCL6	PCL5	PCL4	PCL3	PCL2	PCL1	PCL0	PORTCL	
\$1006	Reserved									
\$1007	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC	
\$1008	0	0	PD5	PD4	PD3	PD2	PD1	PD0	PORTD	
\$1009	0	0	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD	
\$100A	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	PORTE	
\$100B	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	CFORC	
\$100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	0	0	0	OC1M	
\$100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	0	0	0	OC1D	
\$100E	Bit 15	14	13	12	11	10	9	Bit 8	TCNT	(High)
\$100F	Bit 7	6	5	4	3	2	1	Bit 0	TCNT	(Low)
\$1010	Bit 15	14	13	12	11	10	9	Bit 8	TIC1	(High)
\$1011	Bit 7	6	5	4	3	2	1	Bit 0	TIC1	(Low)
\$1012	Bit 15	14	13	12	11	10	9	Bit 8	TIC2	(High)
\$1013	Bit 7	6	5	4	3	2	1	Bit 0	TIC2	(Low)

\$1014	Bit 15	14	13	12	11	10	9	Bit 8	TIC3	(High)
\$1015	Bit 7	6	5	4	3	2	1	Bit 0	TIC3	(Low)
\$1016	Bit 15	14	13	12	11	10	9	Bit 8	TOC1	(High)
\$1017	Bit 7	6	5	4	3	2	1	Bit 0	TOC1	(Low)
\$1018	Bit 15	14	13	12	11	10	9	Bit 8	TOC2	(High)
\$1019	Bit 7	6	5	4	3	2	1	Bit 0	TOC2	(Low)
\$101A	Bit 15	14	13	12	11	10	9	Bit 8	TOC3	(High)
\$101B	Bit 7	6	5	4	3	2	1	Bit 0	TOC3	(Low)
\$101C	Bit 15	14	13	12	11	10	9	Bit 8	TOC4	(High)
\$101D	Bit 7	6	5	4	3	2	1	Bit 0	TOC4	(Low)
\$101E	Bit 15	14	13	12	11	10	9	Bit 8	TI4/O5	(High)
\$101F	Bit 7	6	5	4	3	2	1	Bit 0	TI4/O5	(Low)
\$1020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1	
\$1021	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2	
\$1022	OC1I	OC2I	OC3I	OC4I	I4/O5I	IC1I	IC2I	IC3I	TMSK1	
\$1023	OC1F	OC2F	OC3F	OC4F	I4/O5F	IC1F	IC2F	IC3F	TFLG1	
\$1024	TOI	RTI	PAQVI	PAII	0	0	PR1	PR0	TMSK2	
\$1025	TOF	RTIF	PAOVF	PAIF	0	0	0	0	TFLG2	
\$1026	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0	PACTL	
\$1027	Bit 7	6	5	4	3	2	1	Bit 0	PACNT	
\$1028	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR	
\$1029	SPIF	WCOL	0	MODF	0	0	0	0	SPSR	
\$102A	Bit 7	6	5	4	3	2	1	Bit 0	SPDR	
\$102B	TCLR	SCP2 <sup>1</sup>	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0	BAUD	
\$102C	R8	T8	0	M	WAKE	0	0	0	SCCR1	
\$102D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2	
\$102E	TDRE	TC	RDRF	IDLE	OR	NF	FE	0	SCSR	
\$102F	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0	SCDR	
\$1030	CCF	0	SCAN	MULT	CD	CC	CB	CA	ADCTL	
\$1031	Bit 7	6	5	4	3	2	1	Bit 0	ADR1	
\$1032	Bit 7	6	5	4	3	2	1	Bit 0	ADR2	
\$1033	Bit 7	6	5	4	3	2	1	Bit 0	ADR3	
\$1034	Bit 7	6	5	4	3	2	1	Bit 0	ADR4	
\$1035	0	0	0	PTCON	BPRT3	BPRT2	BPRT1	BPRT0	BPROT	
\$1036	MBE	0	ELAT	EXCOL	EXROW	T1	T0	PGM	EPROG	2
\$1037	Reserved									
\$1038	Reserved									
\$1039	ADPU	CSEL	IRQE	DLY	CME	0	CR1	CR0	OPTION	
\$103A	Bit 7	6	5	4	3	2	1	Bit 0	COPRST	
\$103B	ODD	EVEN	ELAT <sup>3</sup>	BYTE	ROW	ERASE	EELAT	EPGM	PPROG	
\$103C	RBOOT	SMOD	MDA	IRVNE	PSEL3	PSEL2	PSEL1	PSEL0	HPRIO	
\$103D	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0	INIT	
\$103E	TILOP	0	OCCR	CBYP	DISR	FCM	FCOP	TCON	TEST1	
\$103F	EE3 <sup>4</sup>	EE2 <sup>4</sup>	EE1 <sup>4</sup>	EE0 <sup>4</sup>	NOSEC	NOCOP	ROMON	EEON	CONFIG	

## 4.12. Registres internes 68HCF1

Adresses	Bit 7	6	5	4	3	2	1	Bit 0	Registre	
\$1000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PORTA	
\$1001	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA	
\$1002	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0	PORTG	
\$1003	DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	DDRG	
\$1004	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PORTB	
\$1005	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	PORTF	
\$1006	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PORTC	
\$1007	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC	
\$1008	0	0	PD5	PD4	PD3	PD2	PD1	PD0	PORTD	
\$1009	0	0	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD	
\$100A	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	PORTE	
\$100B	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	CFORC	
\$100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	0	0	0	OC1M	
\$100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	0	0	0	OC1D	
\$100E	15	14	13	12	11	10	9	8	TCNT	(High)
\$100F	7	6	5	4	3	2	1	0	TCNT	(Low)
\$1010	15	14	13	12	11	10	9	8	TIC1	(High)
\$1011	7	6	5	4	3	2	1	0	TIC1	(Low)
\$1012	15	14	13	12	11	10	9	8	TIC2	(High)
\$1013	7	6	5	4	3	2	1	0	TIC2	(Low)
\$1014	15	14	13	12	11	10	9	8	TIC3	(High)
\$1015	7	6	5	4	3	2	1	0	TIC3	(Low)
\$1016	15	14	13	12	11	10	9	8	TOC1	(High)
\$1017	7	6	5	4	3	2	1	0	TOC1	(Low)
\$1018	15	14	13	12	11	10	9	8	TOC2	(High)
\$1019	7	6	5	4	3	2	1	0	TOC2	(Low)
\$101A	15	14	13	12	11	10	9	8	TOC3	(High)
\$101B	7	6	5	4	3	2	1	0	TOC3	(Low)
\$101C	15	14	13	12	11	10	9	8	TOC4	(High)
\$101D	7	6	5	4	3	2	1	0	TOC4	(Low)
\$101E	15	14	13	12	11	10	9	8	T4/O5	(High)
\$101F	7	6	5	4	3	2	1	0	T4/O5	(Low)
\$1020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1	
\$1021	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2	
\$1022	OC1I	OC2I	OC3I	OC4I	I4/O5I	IC1I	IC2I	IC3I	TMSK1	
\$1023	OC1F	OC2F	OC3F	OC4F	I4/O5F	IC1F	IC2F	IC3F	TFLG1	
\$1024	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0	TMSK2	
\$1025	TOF	RTIF	PAOVF	PAIF	0	0	0	0	TFLG2	
\$1026	0	PAEN	PAMOD	PEDGE	0	I4/O5	RTR1	RTR0	PACTL	
\$1027	7	6	5	4	3	2	1	0	PACNT	
\$1028	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR	
\$1029	SPIF	WCOL	0	MODF	0	0	0	0	SPSR	
\$102A	7	6	5	4	3	2	1	0	SPDR	
\$102B	TCLR	0	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0	BAUD	
\$102C	R8	T8	0	M	WAKE	0	0	0	SCCR1	
\$102D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2	
\$102E	TDRE	TC	RDRF	IDLE	OR	NF	FE	0	SCSR	
\$102F	7	6	5	4	3	2	1	0	SCDR	
\$1030	CCF	0	SCAN	MULT	CD	CC	CB	CA	ADCTL	
\$1031	7	6	5	4	3	2	1	0	ADR1	
\$1032	7	6	5	4	3	2	1	0	ADR2	
\$1033	7	6	5	4	3	2	1	0	ADR3	
\$1034	7	6	5	4	3	2	1	0	ADR4	
\$1035	0	0	0	PTCON	BPRT3	BPRT2	BPRT1	BPRT0	BPROT	
\$1036	et	\$1037	Reserved							
\$1038	GWOM	CWOM	CLK4X	0	0	0	0	0	OPT2	
\$1039	ADPU	CSEL	IRQE	DLY	CME	FCME	CR1	CR0	OPTION	
\$103A	7	6	5	4	3	2	1	0	COPRST	
\$103B	ODD	EVEN	0	BYTE	ROW	ERASE	EELAT	EEPGM	PPROG	
\$103C	RBOO	SMOD	MDA	IRV	PSEL3	PSEL2	PSEL1	PSEL0	HPRIO	
\$103D	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0	INIT	
\$103E	TILOP	0	OCCR	CBYP	DISR	FCM	FCOP	0	TEST1	
\$103F	EE3	EE2	EE1	EE0	1	NOCOP	1	EEON	CONFIG	
\$1040	to	\$105B	Reserved							
\$105C	IO1SA	IO1SB	IO2SA	IO2SB	GSTHA	GSTHB	PSTHA	PSTHB	CSSTRH	
\$105D	IO1EN	IO1PL	IO2EN	IO2PL	GCSPR	PCSEN	PSIZA	PSIZB	CSCTL	
\$105E	GA15	GA14	GA13	GA12	GA11	GA10	0	0	CSGADR	
\$105F	IO1AV	IO2AV	0	GNPOL	GAVLD	GSIZA	GSIZB	GSIZC	CSGSIZ	

## 5. Périphériques intégrés :

- ü Ports parallèles digitaux en sortie, en entrée, en entrée/sortie
- ü Ports analogiques (sur certains modèles)
- ü Ports de communication série (UART) suivant la norme NRZ
- ü Ports de communication de type SPI vers d'autres circuits (CAN, CNA, Afficheurs ...)
- ü Timer (Production de signaux, mesure de fréquences, de durées)
- ü Accumulateur d'impulsions
- ü Chien de garde
- ü Générateur d'interruption périodique
- ü EEPROM

Une broche peut être commune à plusieurs périphériques.

Ex: La broche du port A PA0 sert également d'entrée de comptage du Timer IC1.

### 5.1. Ports parallèles digitaux

Il existe cinq types de ports parallèles :

- ü En entrée seulement
- ü En sortie seulement (Bipolaire)
- ü En sortie avec Drain ouvert
- ü En entrée ou en sortie bipolaire (configuration

par un registre DDR)

- ü En entrée ou en sortie avec Drain ouvert (configuration par un registre DDR)



Le mode de fonctionnement des broches STROBE et le type de connexion du port C (Bipolaire ou drain ouvert) sont programmés dans le registre **PIOC**

Le sens de transfert des informations peut être programmé bit à bit dans les registres **DDR<sub>x</sub>**

Les données sont lues et écrites dans les registres **PORT<sub>x</sub>**

*Les entrées sont protégées par une diode MOS limitant la tension à 7v, il y a alors un effet d'avalanche, le courant ne doit pas dépasser 25mA*

**Les ports B et C permettent d'effectuer des communications paralleles avec protocole « Handshaking »**



## 5.2. Ports analogiques



**Type** : 8 entrées analogiques multiplexées et un convertisseur analogique numérique 8 bits à approximations successives, à capacités commutés. Précision  $\frac{1}{2}$  LSB. Durée d'une conversion 32 cycles soit 128 cycles pour quatre conversions. Le port E est un port analogique si la fonction CAN est activée, dans le cas contraire c'est un port numérique en entrée.

- Ü Le BIT ADPU du registre OPTION doit être mis à 1 pour activer la fonction CAN
- Ü Le bit CSEL doit être mis à 1 si la fréquence de l'horloge interne (E) est supérieure à 750 KHz
- Ü La conversion débute après une écriture dans le registre **ADCTL**, le micro contrôleur effectue alors quatre conversions successives et place les résultats dans les registres **ADR4-ADR1**.

### ADCTL (A/D Control)

D7	D6	D5	D4	D3	D2	D1	D0
CCF	x	SCAN	MULT	CD	CC	CB	CA

**Une conversion commence après l'écriture dans ADCTL et la mise à 0 de CCF (SOC)**

Le bit CCF est mis à 1 après 4 conversions... (EOC)

**SCAN = 1** : le CAN effectue une seule fois les quatre conversions

**SCAN = 0** : le CAN effectue en permanence fois les quatre conversions

**MULT=0** : Le CAN effectue quatre conversions sur le canal sélectionné par CD, CC, CB, CA et range les résultats dans ADR1, ADR2, ADR3, ADR4 (voir tableau

registres)

**MULT=1** : Le CAN effectue quatre conversions sur quatres canaux différents CB et CA sont ignorés

#### MULT=0

CD : CC : CB : CA	CANAL
0000	AN0 (PE0)
0001	AN1 (PE1)
0010	AN2 (PE2)
0011	AN3 (PE3)
0100	AN4 (PE4)
0101	AN5 (PE5)
0110	AN6 (PE6)
0111	AN7 (PE7)
10xx	Interdit
1100	VRH
1101	VRL
1110	VRH/2
1111	interdit

#### MULT=1

CD : CC : CB : CA	CANAL	Destination
00xx	AN0 (PE0)	ADR1
00xx	AN1 (PE1)	ADR2
00xx	AN2 (PE2)	ADR3
00xx	AN3 (PE3)	ADR4
01xx	AN4 (PE4)	ADR1
01xx	AN5 (PE5)	ADR2
01xx	AN6 (PE6)	ADR3
01xx	AN7 (PE7)	ADR4
10xx	Interdit	
11xx		ADR1
11xx		ADR2
11xx		ADR3
11xx	interdit	

### 5.3. Timer

Les bases de temps :



### 5.4. Générateur d'interruption périodique

**Real Time interrupt : RTI**

Cette fonction permet de générer une impulsion périodique. (Horloge temps réel, mesures automatiques et périodiques, gestion d'affichage multiplexé etc...)

Le bit 6 (RTI) de TMSK2 permet de valider l'interruption de RTI

Le bit 6 (RTIF) de TMSG2 est mis à 1 par la fonction RTI à la fin de chaque période (fonctionnement sans interruption)

Les bits 0 et 1 (RTR0 et RTR1) de PACTL permettent de programmer la période des interruptions temps réel en fonction du quartz

RTR1	RTR0	$E/2^{13}$ div	F(RTI) si Q=8MHz	F(RTI) si Q=12MHz	F(RTI) si Q=16MHz
0	0	1	4.096ms	2.7307ms	2.048ms
0	1	2	8.192ms	5.4613ms	4.096ms
1	0	4	16.386ms	10.9227ms	8.192ms
1	1	8	32.768ms	21.8453ms	16.386ms



## 5.5. TIMERS : Mesures et production de temps



### 5.5.1. Comparaison en entrée :

Un front sur une entrée transfère la valeur de TCNT dans le TIC correspondant. La détection peut être programmée (Registre TCL2) pour un front montant, descendant ou l'un ou l'autre. Un drapeau dans TFLG1 est positionné lors de la détection du front. Il provoque une interruption si le bit correspondant de TMSK1 est actif (=1).

- Ü La différence de comptage de TCNT entre deux fronts identiques permet de mesurer une période.
- Ü La différence de comptage de TCNT entre deux fronts différents permet de mesurer une durée.

EDGxB	EDBxA	Configuration
0	0	Pas de déclenchement
0	1	Déclenchement sur front montant
1	0	Déclenchement sur front descendant
1	1	Déclenchement sur n'importe quel front

**TFLG2** : Registre contenant les drapeaux indiquant un événement en entrée

**TMSK2** : Registre contenant les masques d'interruption

L'interruption TO (Timer Overflow permet de compter les débordements de TCNT et donc de mesurer ou de produire des temps très longs (en fait il n'y a pas de limite), en revanche la limite inférieure de mesure ou de production d'un temps dépend de la durée de traitement de l'interruption (au moins 20 uS sur E9)

**TOF** : détection de passage de TCNT de \$FFFF à \$0000

**TOI** validation de l'interruption de dépassement de capacité de TCNT



### 5.5.2. Comparaison en sortie :

Lorsqu'il y a égalité entre TCNT et TOC le bit OCF de TFLG1 est mis à 0. Une interruption est générée si le bit correspondant de OC de TMSK1 est à 1. La sortie peut être bloquée par le bit FOC de CFROC correspondant.

**OC1M** permet de connecter ou de débrancher les broches du port A du TIMER

**OC1D** contient les valeurs binaires à placer sur les ports PA3 à PA7 lors d'une activation du TIEMR OC1

**TCTL1** : Permet de définir une action périodique pour les bits 3 à 6 du port A

OMx	OLx	Configuration
0	0	PAX inchangé
0	1	Bascule PAX
1	0	PAX=0
1	1	PAX=1

**CFORC** : permet d'inhiber l'action sur les sorties

Pour produire un signal carré, il est nécessaire après chaque interruption de recharger TOCx avec D

**D**= durée                      **N**=rapport de  
**PR1-PR0**, E est exprimée en secondes  
**ExN**

### 5.6. Chien de garde et surveillance de l'horloge

Une fonction de surveillance permet de



s'assurer du bon fonctionnement de l'horloge  
*Les systèmes possédant une horloge E de fréquence inférieure à 200KHz ne doivent pas utiliser cette fonction.* La détection d'une fréquence inférieure à 10KHz provoque si elle est activée une réinitialisation sur le vecteur de RESET de défaut d'horloge (\$FFFC,\$FFFD) et un niveau 0 sur la broche RESET. Pour activer cette fonction, le bit CME du registre OPTION doit être positionné.

Une fonction chien de garde est intégrée dans le 68HC11. Celui-ci doit être réarmé périodiquement sans cela un RESET (vecteur en \$FFFA et \$FFFB) est activé automatiquement (broche RESET =0)  
 Le bit 2 (NOCOP) de CONFIG permet de valider le chien de garde  
 Les bits CRO et CR1 de OPTION permettent de programmer la durée du chien de garde en fonction du quartz

Le registre COPRST permet la réinitialisation du chien de garde

**L'écriture de \$55 puis de \$AA dans ce registre réinitialise le compteur du chien de garde**

CR1	CR0	E/2 <sup>15</sup> div	Durée WD si Q=8MHz
0	0	1	16.384 mS
0	1	4	65.536 mS
1	0	16	262.14 mS
1	1	64	1.049 S

## 5.7. Accumulateur d'impulsions

Ü Cette fonction permet de compter des événements et de générer une interruption lors du passage du compteur 8 bits de \$FF à \$00

Ü Suivant le bit PAMOD de PACTL le registre compteur PACNT compte des événements ou compte les impulsions provenant de fréquence E/64 lorsque la broche PA7 est active. (Conversion temps/nombre)

*Avec un quartz de 8MHz : la résolution est de 32uS et un débordement apparaît après 8.19mS*



Entre parenthèses est donné le nom du registre à qui appartient le bit

**PAEN (PACTL)** : Valide l'accumulateur d'impulsions si égal à 1

**DDRA7 (PACTL)** : Ce bit doit être à 0 (broche en entrée ) lors de l'utilisation de l'accumulateur d'impulsions

**PAOVI (TMSK2)** : Masque de l'interruption générée lors du dépassement de capacité

**PAOVF (TFLG2)** : indicateur de dépassement de capacité

**PAII (TMSK2)** : Masque de l'interruption générée lors de la détection d'un front actif sur PA7

**PAIF (TFLG2)** : indicateur de front actif sur PA7

**PAEN (PACTL)** : =0 le compteur PACNT est incrémenté à chaque front actif

=1 le compteur compte les impulsions de fréquence E/64 durant le niveau actif de PA7

**PEDGE (PACTL)** : =0 comptage sur front descendant ( Blocage de E/64 au niveau 0 de PA7)

=1 comptage sur front montant ( Blocage de E/64 au niveau 1 de PA7)

(L'initialisation de PEDGE provoque la validation de PAIF)

## 5.8. Ports de communication de type SPI

Cet interface permet de communiquer en série de manière synchrone avec de nombreux périphérique (ex: voir doc TLC1549C ou MCM2814P)

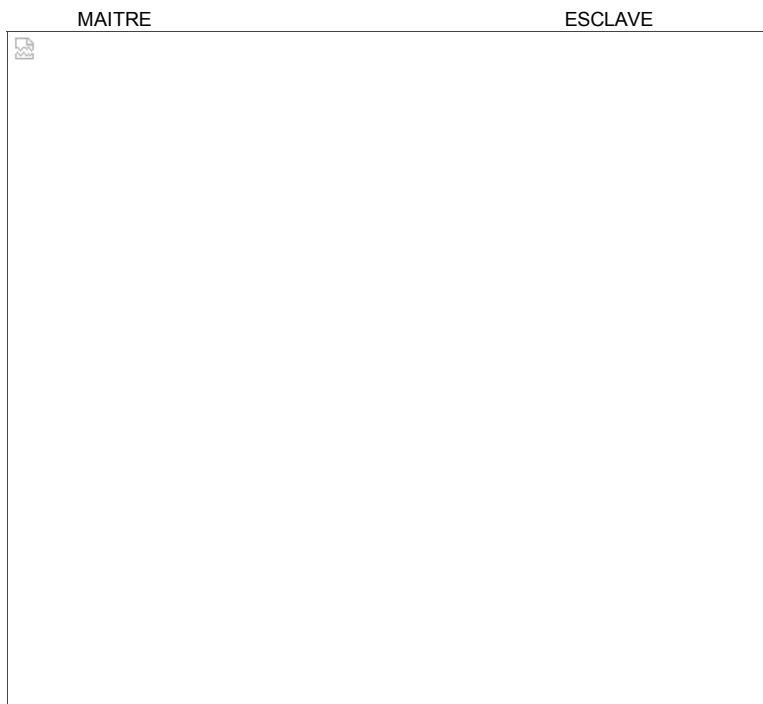
Le 68HC11 est généralement configuré en maître et les autres périphériques en esclave.

**Principe :**

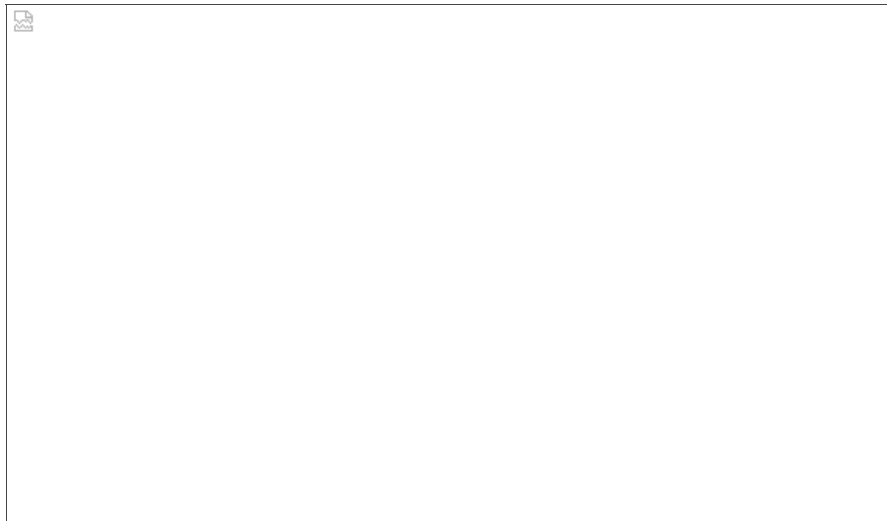


Zone





La transmission réception est simultanée (il n'y a qu'un registre à décalage dans l'émetteur et un dans le récepteur). Lors de la transmission le maître est transféré dans l'esclave et vice-versa. L'horloge est commune (transmission synchrone)



CPHA=0 : SS doit être désactivée entre chaque octet (certains périphériques nécessitent un front pour débiter leur émission)

CPHA=1 : SS peut rester à 0 (active) entre chaque octet, il n'y a pas de risque de collision

#### SPI status register (SPSR) :

SPIF=1 (SPI transfert complète flag) à la fin du transfert, mis à 0 par la lecture de SPSR suivie de celle de SPDR

WCOL=1 (Write collision) si une écriture dans SPDR est effectuée pendant un transfert. Mis à 0 par la lecture de SPSR suivie de celle de SPDR

MODF=1 (Mode fault error flag) si SS=0 en mode maître par la lecture de SPSR suivie de celle de SPCR

#### SPI control register (SPCR):

SPIE=1 : l'interruption SPI est générée si SPIF=1 ou MODF=1

SPE=1 : validation des communications

DWOM : 0 => port D push-pull 1=> port D drains ouverts

MSTR=1 mode maître MSTR=0 mode esclave

CPOL: polarité active de l'horloge (0 ou 1)

CPHA : phase (voir chronogrammes)

SPR1,SPR0: vitesse de transmission

SPR1	SPR0	E divisée par
0	0	2
0	1	4
1	0	16
1	1	32



## 5.9. EEPROM

Certaines version possèdent jusqu'à 2K0 d'EEPROM, un seul registre en assure le controle (PROG)

- Généralement les adresses de l'EEPROM vont de \$B600 à \$BFFF (512 octets)

**PROG** : 7 6 5 4 3 2 1 0  
0 0 0 BYTE ROW ERASE EELAT EEPGM

BYTE	ROW	Type d'effacement
0	0	Tout de l'EEPROM
0	1	16 Octets
1	0	1 Octet
1	1	1 Octet

**ERASE** : 0 : Mode lecture ou programmation

1 : Mode effacement

**EELAT** : 0 : Mode ROM

1 : Mode entrée de données

**EEPGM** : 1 : Activation de Vpp

- Vpp est généré par la même pompe de charge que celle de la fonction C.A.N. Si la fréquence de E est supérieure à 2MHz le bit CSEL du registre OPTION doit être mis à 1.

- Lors d'une opération faisant appel à **Vpp** (Ecriture, Effacement) cette tension doit être activée **durant 10 mS** pour une fréquence de **E de 2 MHz**

## 5.10. Ports de communication série asynchrone(UART)

Ce port permet au micro-contrôleur de communiquer avec d'autres équipements (Terminal alphanumérique, ordinateur ...) en ajoutant un interface suivant la norme RS232C.

Le format de transmission/réception est le code NRZ :

Niveau de repos à l'état haut : 1 bit de start, 8 ou 9 bits de données, 1 bit de stop.

### Lecture/Ecriture

Les données sont lues et écrites dans un registre : **SCDR**

Une opération d'écriture dans ce registre, correspond à 'une demande d'émission, le registre est alors transféré dans le registre à décalage interne ce qui libère le registre d'émission.

Lors d'une réception la donnée lue est placée dans le registre SCDR ce qui permet en attendant la lecture la réception de la donnée suivante.

**Attention** : Une écriture dans SCDR si une donnée a été reçue et non lue provoque une erreur de recouvrement et la perte de la donnée reçue.

### Réveil automatique

En cas de connexion avec plusieurs équipements, il est nécessaire de définir en réception à qui le message est adressé. Le 68HC11 possède une fonction de veille permettant d'ignorer un message si celui ci ne lui est pas adressé.

Réveil sur ligne libre: (IDLE Line Wakeup) Dès réception du premier caractère, le logiciel reconnaît ou non un code d'identification. S'il le reconnaît le message est lu puis l'UART est placée en mode **IDLE Line Wakeup**. S'il ne le reconnaît pas le logiciel met le bit RWU du registre SCCR2 à 1, ce qui a pour effet d'inhiber la réception jusqu'à la fin du message.

Réveil sur reconnaissance d'adresse: (Address-Mark wakeup) Le premier caractère d'un message doit avoir le bit de poids fort à 1, les autres bits composant le numéro d'identification du périphérique. Dans ce mode la réception est activée lorsqu'un caractère possède le bit de poids fort à 1. Si le code d'identification est reconnu le message est lu, sinon le bit RWU est mis à 1 et la réception est ignorée jusqu'à l'arrivée d'un caractère avec bit de poids fort à 1.

**Détection d'erreur** : Il y a trois types d'erreur reconnus et indiqués :

- **Erreur de recouvrement** : la donnée reçue dans SCDR a été écrasée par une écriture (bit OR du registre SCSR)
- **Détection de bruit** : Par exemple, un bit de start de durée inférieure à la période de transmission.(bit NF du SCSR)
- **Absence de bit de stop** : (Framing error) bit FE du SCSR

Des événements permettent de déclencher si elle est activée l'interruption SCI :



TDRE =1 indique que SCDR est vide

TIE =1 provoque une interruption lorsque TDRE = 1

TC=1 indique que le caractère est transmis (Registre à décalage vide)

TCIE= 1 provoque une interruption lorsque TC= 1

T8 : si M=1 le bit T8 est le neuvième de la transmission

M=1 9 bits de donnée, M=0 8 bits de donnée

TE:=1 validation du transmetteur (utilisé avec WAKEUP)

SBK =1 génération d'un break (TX=0)



RDRF=1 un caractère est arrivé et se trouve dans SCDR

RIE=1 : provoque une interruption si RDRF= 1

IDLE : une fois mis à 0 ce bit passe à 1 après une réception de n caractères.

ILIE=1 provoque une interruption si IDLE=1

OR=1 si un caractère est reçu avant la lecture du précédent

RIE=1 provoque une interruption si OR=1

RE=1 : validation du récepteur (utilisé avec WAKEUP)

RWU=1 WAKEUP validé et interruptions de réception inhibées

WAKE =0 détection d'absence de réception (ligne libre)

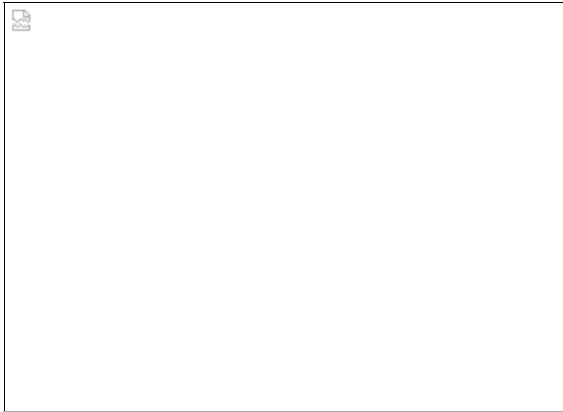
=1 Détection de bit de poids fort à 1

R8 : si M=1 R8 contient le neuvième bit de la réception


Le registre **BAUD** permet de sélectionner la vitesse de transmission réception en fonction du quartz.

*Les bits 7 (TCLR) et 3 (RCBK) sont utilisées lors des tests de production*

## 6.



## Exemples de logiciels

 Zone de Texte: Attention :  
Certains exemples sont écrits pour  
le 68HC11D3 celui ci possède ses  
registres à l'adresse \$0000 et non  
\$1000 comme le 68HC11Ex

### Initialisation

La configuration du micro-contrôleur n'étant pas fixe, certains registres doivent être configurés pendant les 64 premiers cycles d'horloge, il est ensuite impossible de les réécrire.



**CONFIG** : Indique la présence du chien de garde  
La présence de l'EPROM  
La présence de l'EEPROM (s'il y en a une)  
(CONFIG doit obligatoirement être initialisé)

**INIT** : Permet de définir le plan mémoire interne (RAM et registres)  
(Ce registre possède des valeurs par défaut, il n'est pas obligatoire de l'initialiser, sinon durant les 64 premiers cycles d'horloge)

**OPTION** : Valide la pompe de charge du C.A.N,  
Permet le choix de l'horloge de la pompe de charge de l'EEPROM  
Permet le choix du déclenchement d'IRQ (Niveau ou front)  
Permet d'attendre la stabilisation de l'horloge système lors d'un RESET  
Valide le testeur d'horloge  
Définit la durée du chien de garde

**BPROT** : Permet de protéger en écriture le registre CONFIG et une zone de l'EEPROM

**HPRIO** : Permet De définir la source interruption prioritaire  
De connaître le mode de fonctionnement lors du RESET  
D'activer le mode DEBUG (les données internes sont placées sur le bus externe)

## 6.1. Programmation des ports parallèles

```

;Clignotement de la LED rouge sur CBOYF1
; pour Controlboy F1
0002      portg equ 2
0003      ddrgr equ 3
8000      org $8000 ; debut EEPROM
8000 0E      2 start cli ; pour le debogueur !
8001 CE1000 3      ldx #$1000
8004 1C0301 7      bset ddrgr,x $01 ; pg0 = sortie
8007 6C02 6 led inc portg,x ; bascule led
8009 CC8000 3      ldd #$8000 ; delai
800C 830001 4 boucle subd #1
800F 26FB 3      bne boucle
8011 20F4 3      bra led

```

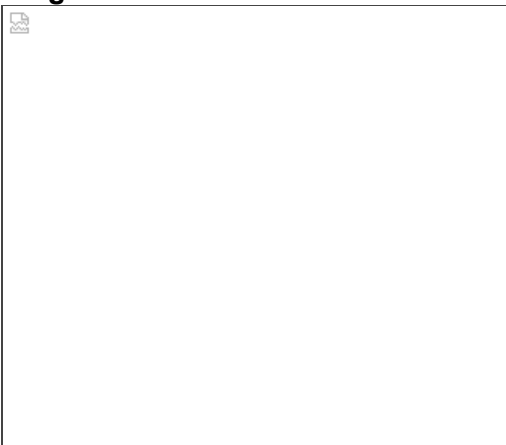
## 6.2. Utilisation de RTI

```

; Compteur BCD utilisant RTI
; programme pour CB2 avec afficheur 7 segments
; C.D 02/1998 Lycee M.M.Fourcade 13120 Gardanne
; La fonction RTI génère une interruption automatique de tempo = 4.096mS avec Q=8MHz
0420      SCONF EQU $420
0410      PRB EQU $410 ;REGISTRE DE SORTIE PORT C
0026      PACTL EQU $26 ;Bits 0 et 1 pour delay RTI
0025      TFLG2 EQU $25 ;Bit 6 : Indicateur d'IRQ RTI
0024      TMSK2 EQU $24 ;Bit 6 : Masque d'IRQ RTI
0040      COMPT EQU $40 ;Compteur de boucle pour tempo
0041      TIME EQU $41 ;Compteur de temps reel, pointeur de table
E000      ROM EQU $E000
E000      ORG ROM ;DEBUT PROG EN DEBUT EPROM
E000      RESET EQU *
E000 18CE1000 LDY #$1000 ; base des registres
;init PORTB
E004 8648 LDAA #$48 ;INITIALISATIONS PORT B
E006 B70420 STAA SCONF ;PORT B EN SORTIE
; init RTI
E009 7F0040 CLR COMPT
E00C CEE03E LDX #TABLE ;pointe le premier nombre 7 seg (0)
E00F DF41 STX TIME
E011 8603 LDAA #%00000011 ;INITIALISATION RTI
E013 18A726 STAA PACTL,Y ;Division RTI=4.096mS
E016 8640 LDAA #%01000000 ;Valide RTI
E018 18A724 STAA TMSK2,Y
; Noter que maintenant le programme principal ne fait RIEN
E01B 0E BOUCLE CLI
E01C 20FE DEBUT BRA DEBUT ;ATTENTE RTI
;TRAITEMENT DES INTERRUPTIONS :
E01E D640 ADRTC LDAB COMPT
E020 5C INCB
E021 C1C8 CMPB #200 ;pour obtenir 1S (environ)
E023 2612 BNE NON ;Si non pas de changement
E025 DE41 LDX TIME
E027 A600 LDAA 0,X
E029 2605 BNE AFFCPT
E02B CEE03E LDX #TABLE ;réinitialise le pointeur de 7 seg
E02E A600 LDAA 0,X
E030 B70410 AFFCPT STAA PRB
E033 08 INX ;pointe le chiffre suivant
E034 DF41 STX TIME
E036 5F CLR RB ;Reinit compteur
E037 D740 NON STAB COMPT
E039 181D2540 BCLR TFLG2,Y %01000000 ;efface drapeau d'IT RTC
E03D 3B ADRTI RTI
; Table de caractères 7SEG
E03E TABLE EQU *
E03E 0001020304 FCB 0,1,2,3,4,5,6,7,8,9 ;!!! à coder 7 seg par les 0 sur PRB de CBOY2
E048 00 FCB 0 ;le 0 permet de détecter la fin de la table
;
FFF0 ORG $FFF0 ;POSITION DU BLOC VECTEURS (EPROM)
FFF0 E01E ADRTC ;INTERRUPTION PERIODIQUE

```

## 6.3. Utilisation du chien de garde



```

0001      * INITIALISATION DU CHIEN DE GARDE
0002      *
0003      * Lycee M.M.Fourcade

```

```

0004      * 13120 Gardanne
0005      *
0006      * Ce programme ne fait rien
0007      * Le chien de garde est armé, (T=1.049s)
0008      * puis réarmé 10 fois
0009      * en fin de tempo (0.8s) un interruption COP est
0010      * générée, le S/P d'interruption effectue alors un RESET
0011      *
0012      *EQUIVALENCES GLOBALES :
0013      *
0014 003f    CONFIG    EQU    $3F    0 0 0 0 NOCOP EPON 0
0015 0002    EPON      EQU    2      CONFIG-1 : EPROM INTERNE ACTIVE A 1
0016 0006    NOCOP     EQU    6      CONFIG-2 : PAS DE WATCH DOG SIA 1
0017 0039    OPTION    EQU    $39    ADPU CSEL IRQE DLY CME 0 CR1 CR0
0018      *          0 0 0 0 0 0 x x
0019 003a    COPRST     EQU    $3A    Controle chien de garde
0020      *
0021      *DEBUT DE LA ZONE EPROM 4K DU 68711D3
0022      *
0023 f000      ORG      $F000    DEBUT PROG EN DEBUT EPROM
0024      *
0025      *DEBUT DU PROGRAMME PRINCIPAL
0026      *
0027      * Init 68HC11
0028      *
0029 f000 86 03 [ 2 ] RESET    LDA    #%00000011    COP Timeout=1,049s pour E=2MHz
0030 f002 97 39 [ 3 ]          STA    <OPTION
0031 f004 86 08 [ 2 ]          LDAA   #NOCOP+EPON
0032 f006 97 3f [ 3 ]          STAA   <CONFIG EPROM INTERNE ACTIVE
0033 f008 8e 00 ff [ 3 ]          LDS    #$00FF INITIALISATION STACK SYSTEME
0034      *
0035      *DEBUT DE LA BOUCLE PRINCIPALE :
0036      *
0037 f00b c6 0a [ 2 ]          LDB     #10      10 r,armements
0038 f00d 86 55 [ 2 ] BOUCLE   LDA     #$55      Initialise compteur
0039 f00f 97 3a [ 3 ]          STA     <COPRST chien de garde
0040 f011 43 [ 2 ]          COMA    A=$AA
0041 f012 97 3a [ 3 ]          STA     <COPRST
0042 f014 8d 05 [ 6 ]          BSR     TEMPO    Dur,e 800 mS
0043 f016 5a [ 2 ]          DECB    BNE
0044 f017 26 f4 [ 3 ]          BNE     BOUCLE
0045      *
0046 f019 20 fe [ 3 ] FIN      BRA     FIN      Attent INT COP
0047      *
0048      * S/P tempo 800 mS
0049      *
0050 f01b ce 06 40 [ 3 ] TEMPO   LDX     #1600    X DECOMPTE 1600 X 200 X BOUCLES
0051 f01e 86 c8 [ 2 ] DELAYL1  LDAA    #200     A DECOMPTE 200 X BOUCLES DE 5 CYCLES
0052 f020 4a [ 2 ] DELAYL2  DECA    DECOMPTE NOMBRE DE BOUCLES
0053 f021 26 fd [ 3 ]          BNE     DELAYL2 REPEND SINON 200 ( 0.5 MS )
0054 f023 09 [ 3 ]          DEX     DECOMPTE NOMBRE DE 200 BOUCLES
0055 f024 26 fb [ 3 ]          BNE     DELAYL1 REPEND SINON
0056 f026 39 [ 5 ]          RTS     RETOURNE APRES 1600 X 200 X 5 CYCLES
0057      *
0058      * TRAITEMENT DES INTERRUPTIONS :
0059      * INTERRUPTION CHIEN DE GARDE (COP) UTILISEE
0060      * RETOUR PAR RTI SUR INTERRUPTION ALEATOIRE
0061      *
0062 f027 20 d7 [ 3 ] ADCOP     BRA     RESET    Reinitialisation
0063      *
0064 f029 3b [ 12 ] ADRTI      RTI             Pour INT Aleatoires
0065      *
0066      *FIN DE ROUTINES
0067      *VECTEURS D'INTERRUPTIONS GENERALES (16 OCTETS EPROM)
0068      *
0069 fff0      ORG      $FFF0    POSITION DU BLOC VECTEURS (EPROM)
0070 fff0 f0 29 TIMVECT     FDB     ADRTI    INTERRUPTION PERIODIQUE
0071 fff2 f0 29 IRQVECT     FDB     ADRTI    INTERRUPTION BROCHE IRQ
0072 fff4 f0 29 XRQVECT     FDB     ADRTI    INTERRUPTION BROCHE XRQ
0073 fff6 f0 29 SWVECT      FDB     ADRTI    INTERRUPTION SOFT
0074 fff8 f0 29 CODVECT     FDB     ADRTI    CODE OPERATION ILLEGAL
0075 fffa f0 27 COPVECT     FDB     ADCOP    WATCH DOG
0076 fffc f0 29 CLKVECT     FDB     ADRTI    PROBLEME D'HORLOGE
0077 fffe f0 00 RSTVECT     FDB     RESET    RESET HARD PREMIERE INSTRUCTION
0078      END

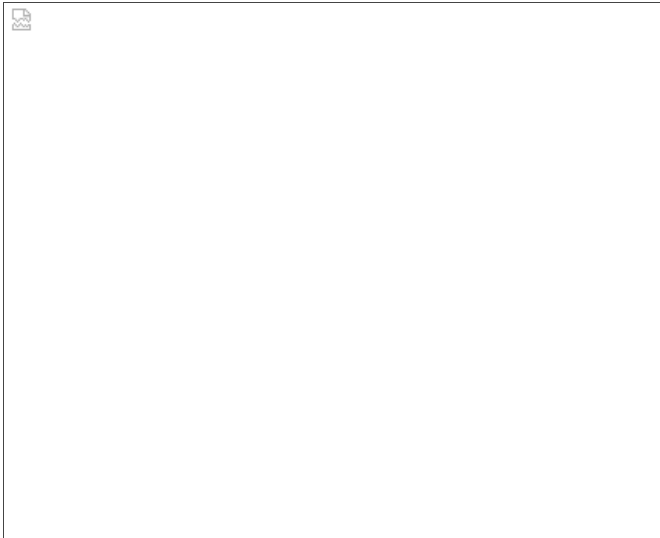
```

## **Equivalences générales communes à tous les exemples**

\*\* Equates - Registers will be addressed with Ind,X mode

REGBAS	EQU	\$1000	Starting address for register block
PORTB	EQU	\$04	Output port B
OC1M	EQU	\$0C	OC1M7,OC1M6,OC1M5,OC1M4;OC1M3,-,-,-
OC1D	EQU	\$0D	OC1D7,OC1D6,OC1D5,OC1D4;OC1D3,-,-,-
TCNT	EQU	\$0E	Free running counter (16-bit)
TIC1	EQU	\$10	IC1 register (16-bit)
TOC1	EQU	\$16	OC1 register (16-bit)
TOC2	EQU	\$18	OC2 register (16-bit)
TOC3	EQU	\$1A	OC3 register (16-bit)
TCTL1	EQU	\$20	OM2,OL2,OM3,OL3;OM4,OL4,OM5,OL5
TCTL2	EQU	\$21	-,EDG1B,EDG1A;EDG2B,EDG2A,EDG3B,EDG3A
TMSK1	EQU	\$22	OC1I,OC2I,OC3I,OC4I;OC5I,IC1I,IC2I,IC3I
TFLG1	EQU	\$23	OC1F,OC2F,OC3F,OC4F;OC5F,IC1F,IC2F,IC3F
TMSK2	EQU	\$24	TOI,RTII,PAOVI,PAIFI,-,-,PR1,PR0
TFLG2	EQU	\$25	TOF,RTIF,PAOVF,PAIF,-,-,-,-

### **6.4. Programmation du TIMER en entrée, mesure de périodes**



\* Measuring Period with Input Capture

\*

\* Uses polling rather than interrupts.

\* Measures period between two rising edges at IC1 pin.

\* Overflows not considered so max period is 65,535 cyc

\* Min period measurable with this program is about 27 cyc

\*

\*\*\*

```
PERTOP      LDS    #$0047      Top of User's stack area on EVB
             LDX    #REGBAS    Point to register block
             LDAA   #%00010000
             STAA   TCTL2,X     EDG1B:EDG1A = 0:1, rising edges
             LDAA   #$04
             STAA   TFLG1,X     Clear any old OC1 flag
```

\* Ready to detect first rising edge

```
BRCLR TFLG1,X    $04 *      Loop here until edge
```

\* First edge detected

```
LDD    TIC1,X     Read time of first edge
STD    FRSTE      Save first capture value
LDAA   #$04
STAA   TFLG1,X   Clear IC1F before next edge
```

\* Ready to capture time of second edge

```
BRCLR TFLG1,X    $04 *      Loop here until edge
```

\* Second edge detected

```
LDD    TIC1,X     Read time of second edge
SUBD   FRSTE      2nd - 1st -> D
STD    PERC       Save result (period in cycles)
```

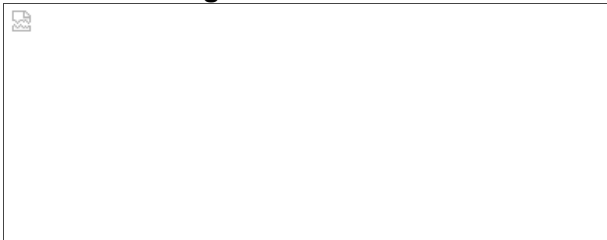
\* The period of the signal at the PA2/IC1 pin has been measured

\* and the time is stored at "PERC" as a 16-bit hex number

\* representing the number of CPU bus cycles that elapsed

\* between two rising edges.

### **6.5. Programmation du TIMER en entrée, mesure de durée**



```
PWTOP      LDX    #REGBAS    Point to register block
             LDAA   #%00010000 Top of Main for PW24 prog
             STAA   TCTL2,X     EDG1B:EDG1A = 0:1, IC1 rising edges
             LDAA   #$FF
             STAA   IC1MOD      FF-IC1 off; 0-1st edge; 1-last edge
             CLR    IC1DUN      Signal pulse not done
             BCLR   TFLG1,X $FB clear IC1F (if any)
             BSET   TMSK1,X $04 enable IC1 interrupts
             CLI                      Enable Interrupts

WAITL2      LDAA   IC1DUN      Sets after pulse done
             BEQ    WAITL2      Tight loop till pulse has been timed
             SEI                      Pulse done, disable interrupts
```

```

***
* SV2IC1 - Input Capture 1 service routine
*
* Called first when a rising edge is detected and again when
* a falling edge is detected.
***
SV2IC1          LDX  #REGBAS    point at top of register block
                INC  IC1MOD      $FF->0 at 1st edge; 0->1 at 2nd
                BNE  NO1ST2      if not 0, this is trailing edge

* Process leading edge of pulse
                LDD  TIC1,X      read time of first edge
                STD  FRSTE       save till next capture

* Reconfigure IC1 for trailing falling edge
                BCLR TCTL2,X $30  EDG1B:EDG1A->0:0
                BSET TCTL2,X $20  EDG1B:EDG1A->1:0
                BRA  OU2IC1      done processing first edge

* Process trailing edge of pulse
NO1ST2          LDD  TIC1,X      get time of trailing edge
                SUBD FRSTE       time of last minus time of first
                STD  HPW         update result
                BCLR TCTL2,X $30  disable IC1
                LDAA #1
                STAA IC1DUN      signal pulse measured
OU2IC1          BCLR TFLG1,X $FB clear IC1F
                RTI              ** Return from IC1 service **

```

## 6.6. Programmation du TIMER en sortie -mode astable

```

PER24T          LDX  #REGBAS    Point to register block
                LDAA #00010000  Top of Main for PER24 prog
                STAA TCTL2,X    EDG1B:EDG1A = 0:1, IC1 rising edges
                LDAA #0FF
                STAA IC1MOD      FF-IC1 off; 0-1st edge; 1-last edge
                CLR  IC1DUN      Signal period not done
                BCLR TFLG1,X $FB clear IC1F (if any)
                BCLR TFLG2,X $7F clear TOF (if any)
                BSET TMSK1,X $04 enable IC1 interrupts
                BSET TMSK2,X $80 enable TOF interrupts
                CLI              Enable Interrupts

***
* SV5OC2 - Output Compare 2 service routine
*
* Called at each OC2 interrupt.
***
SV5OC2          LDD  HDLY       Get delay time for 1/2 cycle
                ADDD TOC2,X     Add to last compare value
                STD  TOC2,X     Update OC2 (schedule next edge)
                BCLR TFLG1,X $BF Clear OC2F
                RTI              ** Return from OC2 service **

*

```

## 6.7. Programmation de l'accumulateur d'impulsion

```

0001      * Exemple :
0002      * programmation de l'accumulateur d'impulsions
0003      * La fonction est utilisée pour générer une interruption
0004      * après une minute de fonctionnement
0005      * Une horloge de fréquence 1Hz produit un signal sur la broche PA7
0006      * L'interruption PAOVI devra donc être générée après 60 impulsions
0007      * (PAOVI : Pulse Accumulator Overflow Interrupt)
0008      *
0009      * EQUIVALENCES GLOBALES :
0010      *
0011 003f    CONFIG    EQU    $3F 0 0 0 0 NOCOP EPON 0
0012 0007    DDRC     EQU    7      REGISTRE DE DIRECTION PORT C
0013 0002    EPON     EQU    2      CONFIG-1 : EPROM INTERNE ACTIVE A 1
0014 0006    NOCOP    EQU    6      CONFIG-2 : PAS DE WATCH DOG SIA 1
0015 0003    PRC      EQU    3      REGISTRE DE SORTIE PORT C
0016 0026    PACTL    EQU    $26    Controle Accumulateur d'impulsions
0017 0027    PACNT    EQU    $27    Compteur de l'accumulateur d'impulsions
0018 0024    TMSK2    EQU    $24    Timer interrupt mask 2
0019 ffff    TEMPO    EQU    $FFFF   Durée d'attente d'IT
0020      *
0021      * DEBUT DE LA ZONE EPROM 4K DU 68711D3
0022      *
0023 f000      ORG      $F000    DEBUT PROG EN DEBUT EPROM
0024      *
0025      * DEBUT DU PROGRAMME PRINCIPAL
0026      *
0027      * Init 68HC11
0028 f000 86 08  [2]    RESET    LDAA    #NOCOP+EPON  PAS DE WATCH DOG
0029 f002 97 3f  [3]          STAA    <CONFIG EPROM INTERNE ACTIVE
0030 f004 8e 00 ff [3]          LDS     #$00FF  INITIALISATION STACK SYSTEME
0031      *
0032 f007 86 20  [2]          LDAA    #00100000    DEMASQUE PAOVI
0033 f009 97 24  [3]          STAA    <TMSK2
0034 f00b 86 c3  [2]          LDAA    #255-60 Compte 60 impulsions avant débordement
0035 f00d 97 27  [3]          STAA    <PACNT
0036      *
0037      * PA7 en entr,e, Valide accumulateur, compte les impulsions
0038      * sur front descendant sur PA7
0039      *
0040 f00f 86 40  [2]          LDAA    #01000000    Init mode Accumulateur
0041 f011 97 26  [3]          STAA    <PACTL
0042      *
0043      * DEBUT DE LA BOUCLE PRINCIPALE :
0044      *
0045 f013 0e      [2]    BOUCLE    CLI      Autorise les INT
0046 f014 20 fe  [3]    DEBUT     BRA      ATTENTE INTERRUPTION
0047 f016 20 fb  [3]          BRA      BOUCLE

```



```

0048      *
0049      * TRAITEMENT DES INTERRUPTIONS :
0050      * PAOINT UTILISEE
0051      * RETOUR PAR RTI SUR INTERRUPTION ALEATOIRE
0053 f018 15 24 a0 [6] PAOINT BCLR TMSK2 #00100000 Masque PAOVI
0054 f01b 3b [12] RTI RTI
0056      * FIN DE ROUTINES
0058 ffdc      ORG $FFDC VECTEUR INT P.A.OVERFLOW
0059 ffdc f0 18 PAOJECT FDB PAOINT
0061      * VECTEURS D'INTERRUPTIONS GENERALES (16 OCTETS EPROM)
0062 fff0      ORG $FFF0 POSITION DU BLOC VECTEURS (EPROM)
0063 fff0 f0 1b TIMVECT FDB RTI INTERRUPTION PERIODIQUE
0064 fff2 f0 1b IRQVECT FDB RTI INTERRUPTION BROCHE IRQ
0065 fff4 f0 1b XRQVECT FDB RTI INTERRUPTION BROCHE XRQ
0066 fff6 f0 1b SWVECT FDB RTI INTERRUPTION SOFT
0067 fff8 f0 1b CODVECT FDB RTI CODE OPERATION ILLEGAL
0068 fffa f0 1b COPVECT FDB RTI WATCH DOG
0069 fffc f0 1b CLKVECT FDB RTI PROBLEME D'HORLOGE
0070 fffe f0 00 RSTVECT FDB RESET RESET HARD PREMIERE INSTRUCTION
0071      * FIN DE LA ZONE EPROM DU 68711D3
0072      END      *

```

## 6.8. Programmation du SPI

```

* S.T.S Electronique Lycee M.M.Fourcade 13120 Gardanne
* C.D 10/93
* TEST SPI AVEC TLC549
* Le programme effectue une mesure de la tension sur le TLC549 en permanence
* et range le resultat dans la memoire MESURE
*
* EQUIVALENCES GLOBALES :
*
CONFIG EQU $3F 0 0 0 0 NOCOP EPON 0
EPON EQU 2 CONFIG-1 : EPROM INTERNE ACTIVE A 1
NOCOP EQU 6 CONFIG-2 : PAS DE WATCH DOG SIA 1
PRD EQU
SPCR EQU
SPDR EQU
SPSR EQU
REGBAS EQU $1000
PROG EQU $E000 zone EPROM
*
* VARIABLES
*
MESURE ORG $40
RMB 1 Resultat de la mesure
*
* ORG PROG DEBUT PROG
* DEBUT DU PROGRAMME PRINCIPAL
*
* Init 68HC11
*
RESET LDX #REGBAS
LDAA #NOCOP+EPON PAS DE WATCH DOG
STAA <CONFIG,X EPROM INTERNE ACTIVE
LDS #$00FF INITIALISATION STACK sur XEMUL11
*
* Init SPI pour CAN TLC549
*
LDAA #????????? Pas d'IT, SPION
STAA <SPCR,X CMOS, Master, 125 KBAUD
*
* DEBUT DE LA BOUCLE PRINCIPALE :
*
DEBUT BSR CAN
BOUCLE BRA *
*
* Gestion CAN TLC549
*
CAN EQU *
BCLR PRD,X %?????????
LDAA #00000000 Par exemple...
STAA <SPDR,X
CAN1 BCLR SPSR,X %????????? CAN1 Test SPIF
LDAA <SPDR,X
STAA <MESURE Sauve le resultat
BSET PRD,X %?????????
RTS
*

```

## 6.9. Programmation du SCI

```

0001      * Exemple de routine de gestion
0002      * du port SCI
0003      *
0004      * Un message est transmis sur le port s,rie
0005      * La reception est traitee en interruption
0006      * les caractères recus sont places dans un buffer
0007      * en RAM
0008      *
0009      * INIT : initialisation du port : 1200 Bauds 8 bits 1 Stop
0010      * avec un quartz 8MHz
0011      *
0012      * TX : Routine d'emission d'un chaine
0013      * pointé par Y
0014      *
0015      * RX : Routine de reception placant le caractère
0016      * reçu a l'adresse donnée par le contenu de ADBUFF
0017      * (dans l'exemple ADBUFF=$40 et contient au d.part $50)
0018      *
0019 0028 SPCR EQU $28 Control du port D
0020 002b BAUD EQU $2B Baud Register
0021 002c SCCR1 EQU $2C SCI Control Register 1
0022 002d SCCR2 EQU $2D SCI Control Register 2
0023 002e TCSR EQU $2E SCI Status Register
0024 0080 TDRE EQU $80 BIT Transmit Data Register Empty
0025 002f SCDR EQU $2F SCI Data Register

```

```

0026 0007      EOT      EQU      $07      Fin de texte
0027      *
0028 f000      *      ORG      $F000      Zone EPROM
0029
0030 f000 8e 00 ff [3] RESET      LDS      #$00FF      Sommet de la RAM
0031 f003 ce 00 00 [3]      LDX      #$0000      Base des registres pour un 68HC711E9
0032 f006 8d 09 [6]      BSR      INIT
0033 f008 18 ce f0 46 [4]      LDY      #MESSAGE      Envoie un message
0034 f00c 8d 15 [6]      BSR      TX
0035 f00e 0e [2]      CLI      *      Demasque les IT
0036 f00f 20 fe [3]      BRA
0037      *
0038      *      Initialise le port SCI
0039      *
0040 f011 1c 28 20 [7] INIT      BSET      SPCR,X %00100000      Port D en DRAIN ouverts
0041      *
0042 f014 86 b3 [2]      LDAA      #10110011
0043 f016 a7 2b [4]      STAA      BAUD,X      RAZ Compteur de BAUD
0044      *      Divise horloge 4MHz par 13 (BaudMax=9600)
0045      *      Baud=1200
0046 f018 86 28 [2]      LDAA      #00101000
0047 f01a b4 00 42 [4]      ANDA      >VITESS
0048 f01d a7 2d [4]      STAA      SCCR2,X      IT transmissions inhibees
0049      *      IT reception validees
0050      *      IT IDLE LINE inhibee
0051      *      Validation des transmissions
0052      *      Validation des receptions
0053      *      Inhibition du reveil automatique
0054      *      Pas de BREAK
0055 f01f 7f 00 2c [6]      CLR      SCCR1      TX/RX sur 8 bits
0056 f022 39 [5]      RTS
0057      *
0058      *      TX: Envoie un message pointe par Y
0059      *      fin de message code par EOT
0060      *
0061 f023 18 a6 00 [5] TX      LDAA      0,Y
0062 f026 81 07 [2]      CMPA      #EOT      fin du message ?
0063 f028 26 01 [3]      BNE      OK
0064 f02a 39 [5]      RTS
0065 f02b 8d 02 [6] OK      BSR      ENVOIE
0066 f02d 20 f4 [3]      BRA      TX
0067      *
0068      *      ENVOIE : Routine emettant le caractere contenu dans ACCA
0069      *
0070 f02f 1f 2e 80 fc [7] ENVOIE      BRCLR      SCSR,X TDRE ENVOIE      Attend transmetteur pret
0071 f033 a7 2f [4]      STAA      SCDR,X
0072 f035 39 [5]      RTS
0073      *
0074      *      RX: reception d'un caractere
0075      *      le place dans le buffer de reception
0076      *
0077 f036 a6 2f [4] RX      LDAA      SCDR,X      Sauve le caractere
0078 f038 18 fe 00 40 [6]      LDY      >ADBUFF
0079 f03c 18 a7 00 [5]      STAA      0,Y
0080 f03f 18 08 [4]      INY
0081 f041 18 ff 00 40 [6]      STY      >ADBUFF      Incrmente le buffer
0082 f045 3b [12]      ADRTI      RTI      le sauvegarde
0083      *
0084 f046 54 45 53 54 20 54 MESSAGE      FCC      'TEST TRANSMISSION 68HC11'
0085      *      52 41 4e 53 4d 49
0086      *      53 53 49 4f 4e 20
0087      *      36 38 48 43 31 31
0088 f05e 07      *      FCB      EOT
0089      *
0090      *
0091      *      ORG      $0040
0092      *      ADBUFF      FCB      $00,$50      Initialise ADBUFF
0093      *      VITESS      FCB      %00000011      Vitesse TX/RX : 1200 BAUDS
0094      *
0095      *      Positionnement du vecteur d'interruption
0096      *      de reception
0097      *
0098      *      ORG      $FFD6      Vecteur SCI
0099      *      RX
0100      *
0101      *      FDB
0102      *
0103      *      *FIN DE ROUTINES
0104      *      *VECTEURS D'INTERRUPTIONS GENERALES (16 OCTETS EPROM)
0105      *
0106      *      ORG      $FFF0      POSITION DU BLOC VECTEURS (EPROM)
0107      *
0108      *      TIMVECT      FDB      ADRTI      INTERRUPTION PERIODIQUE
0109      *      IRQVECT      FDB      ADRTI      INTERRUPTION BROCHE IRQ
0110      *      XRQVECT      FDB      ADRTI      INTERRUPTION BROCHE XRQ
0111      *      SWIVECT      FDB      ADRTI      INTERRUPTION SOFT
0112      *      CODVECT      FDB      ADRTI      CODE OPERATION LLEGAL
0113      *      COPVECT      FDB      ADRTI      WATCH DOG
0114      *      CLKVECT      FDB      ADRTI      PROBLEME D'HORLOGE
0115      *      RSTVECT      FDB      RESET      RESET HARD PREMIERE INSTRUCTION
0116      *      END      FIN DU LISTING FLASHER

```

## 6.10. Programmation des ports analogiques

```

0001      * Exemple de programme de
0002      * controle du port analogique
0003      *
0004      * Ce sous/programme effectue
0005      * Quatres mesures successives
0006      * sur PE3 puis range les
0007      * résultats aux quatres adresses
0008      * pointés par X
0009      *
0010 1030      ADCTL      EQU      $1030      Contrôle CAN
0011 1031      ADR1      EQU      ADCTL+1 Adresse de base résultat mesure
0012      *
0013 0000 86 07 [2] CAN      LDAA      #00000111      Une série de conversions sur PE3
0014 0002 b7 10 30 [4]      STAA      ADCTL      La conversion commence
0015      *      après l'écriture dans ADCTL
0016 0005 13 30 47 fc [6] ATT      BRCLR      ADCTL %1000111      ATT Attend CCF=1 (fin de conversion)
0017      *

```

```

0018 0009 18 fe 10 31 [6] LDY ADR1 Adresse de base des r,sultats
0019 000d 18 a6 00 [5] LDAA 0,Y sauve mesure 1
0020 0010 a7 00 [4] STAA 0,X
0021 0012 18 a6 01 [5] LDAA 1,Y sauve mesure 2
0022 0015 a7 01 [4] STAA 1,X
0023 0017 18 a6 02 [5] LDAA 2,Y sauve mesure 3
0024 001a a7 02 [4] STAA 2,X
0025 001c 18 a6 03 [5] LDAA 3,Y sauve mesure 4
0026 001f 18 a7 03 [5] STAA 3,Y
0027 0022 39 [5] RTS
0028 END

```

## 6.11. Programmation de l'EEPROM interne

```

0001 * Exemples de controle de l'EEPROM
0002 * interne sur 68HC711E9
0003 *
0004 * Les routines suivantes permettent
0005 * l'ecriture et l'effacement de l'EEPROM
0006 * L'accumulateur A contient l'octet ... programmer
0007 * Le registre X l'adresse ... effacer dans l'EEPROM
0008 * ($B600 < X < $B7FF)
0009 *
0010 103b PROG EQU $103B Registre d'etat de l'EEPROM
0011 f000 EPROM EQU $F000 Adresse de base de l'EPROM
0012 *
0013 f000 ORG EPROM
0014 *
0015 * Ecriture d'un octet
0016 *
0017 f000 c6 02 [2] ECRIT LDAB #%00000010 EELAT=1 (mode LATCH)
0018 f002 f7 10 3b [4] STAB PROG
0019 f005 a7 00 [4] STAA 0,X Ecrit la donnée
0020 f007 86 03 [2] LDAA #%00000011 EEPGM=1 (activation de Vpp)
0021 f009 b7 10 3b [4] STAA PROG
0022 f00c 8d 37 [6] BSR TEMP10 Tempo de 10 mS
0023 f00e 7f 10 3b [6] CLR PROG Desactive Vpp et mode ROM
0024 *
0025 * Effacement d'un octet
0026 *
0027 f011 c6 16 [2] EFFT LDAB #%00010110 Mode LATCH Effacement octet
0028 f013 f7 10 3b [4] STAB PROG
0029 f016 e7 00 [4] STAB 0,X Ecrit n'importe quoi à l'adresse
0030 f018 c6 17 [2] LDAB #%00010111 Active Vpp
0031 f01a f7 10 3b [4] STAB PROG
0032 f01d 8d 26 [6] BSR TEMP10 Tempo de 10 mS
0033 f01f 7f 10 3b [6] CLR PROG Desactive Vpp et mode ROM
0034 *
0035 * Effacement d'une rang, de 16 octets
0036 * ex: $B600 ... $B60F ou $B7F0 ... $B7FF
0037 *
0038 f022 c6 0e [2] EFFRAN LDAB #%00001110 Mode LATCH effacement rang,e
0039 f024 f7 10 3b [4] STAB PROG
0040 f027 e7 00 [4] STAB 0,X Ecrit n'importe quoi ... une des 16 adresses
0041 f029 c6 0f [2] LDAB #%00001111 Active Vpp
0042 f02b f7 10 3b [4] STAB PROG
0043 f02e 8d 15 [6] BSR TEMP10 Tempo de 10 mS
0044 f030 7f 10 3b [6] CLR PROG Desactive Vpp et mode ROM
0045 *
0046 * Effacement de toute l'EEPROM
0047 *
0048 f033 c6 06 [2] EFFROM LDAB #%00000110 Mode LATCH effacement EEPROM
0049 f035 f7 10 3b [4] STAB PROG
0050 f038 f7 b6 00 [4] STAB $B600 Ecrit n'importe quoi ... une adresse EEPROM
0051 f03b c6 07 [2] LDAB #%00000111 Active Vpp
0052 f03d f7 10 3b [4] STAB PROG
0053 f040 8d 03 [6] BSR TEMP10 Tempo de 10 mS
0054 f042 7f 10 3b [6] CLR PROG Desactive Vpp et mode ROM
0055 *
0056 * Tempo de 10mS calculer pour E=2MHz
0057 *
0058 f045 18 ce 0b 29 [4] TEMP10 LDY #2857 2857*(4+3)*0.5e-6 = 10 mS
0059 f049 18 09 [4] BOUCLE DEY 4 cycles
0060 f04b 26 fc [3] BNE BOUCLE 3 cycles
0061 f04d 39 [5] RTS
0062 *
0063 END

```



## Exercices

- ⌚ L'extension des fichiers assembleurs 68HC11 doit être A11 ex : COMPTEUR.A11
- ⌚ Ces fichiers doivent être rangés obligatoirement dans le répertoire TRAVAIL
- ⌚ Les fichiers assemblés possèdent l'extension LST
- ⌚ Les fichiers objets sont au format S19
- ⌚ Le vecteur de RESET ainsi que le registre S ne doivent pas être positionné sur CBOY . Les sous programmes seront testés à l'aide de programmes appelant.
- ⌚ Toutes les valeurs numériques doivent être déclarées en équivalences
- ⌚ Les commentaires doivent être suffisants explicites et judicieux.

### Exercice 1 :

Addition de deux octets : N1 en \$41, N2 en \$42, résultat en \$43

### Exercice 2 :

Soustraction de deux octets N2-N1. Résultat en \$43 et signe en \$44 (\$44=0 pour positif)

### Exercice 3 :

Clignotant de la LED rouge, la temporisation sera ajustée de manière à obtenir  $f=1\text{Hz}$  (environ).

### Exercice 4 :

Modifier le programme précédent de manière à placer la temporisation dans un sous programme

### Exercice 5 :

Réaliser un compteur BCD (0 à 100 dans une mémoire)

### Exercice 6 :

Addition de deux nombres de 16 bits. N1 en \$40, N2 en \$42, résultat en \$44, \$45, \$46

### Exercice 7 :

Recopier le texte « MC68HC11 » situé à l'adresse \$40 vers l'adresse \$50

### Exercice 8 :

Recopie d'une zone mémoire. Adresse de début de zone en \$40,\$41. Destination en \$42,\$43. Longueur en \$44 (max 255)

### Exercice 9 :

Réaliser un sous programme de conversion Hexadécimal ASCII. Le nombre à convertir est placé dans accA, les deux caractères ASCII sont retournés dans accA et accB. Par exemple si accA = \$36, la fonction HEXASC retourne accA = \$33 et accB = \$36. Le sous programme sera appelé par BSR HEXASC

### Exercice 10 :

Réaliser un clignotant sur la LED rouge en utilisant la fonction RTI. (le programme principal ne fait rien. Ex bra \*)

### Exercice 11 :

Produire un signal rectangulaire de fréquence 1KHz en utilisant la fonction Timer OC3, visualiser le résultat à l'oscilloscope

### Exercice 12 :

Intégrer les programmes 10 et 11 afin de constater la puissance de la programmation avec interruptions

### Exercice 13 :

Réaliser un sous programme SINUS retournant le sinus d'un angle entier en degré compris entre 0 et 10 (dans accA). Le résultat sera donné avec une précision sur quatre chiffres ex :  $\sin(8)=0,0871$ . Le résultat serait 08 dans accA et 71 dans accB. (la partie entière est ignorée)

## CORRECTION

\* EXERCICE 1 : Addition de deux octets

\* N1 en \$40, N2 en \$41, résultat en \$43

\* Le résultat est tronqué, à 8bits

\*

```
N1 EQU $40
N2 EQU $41
RES EQU $42
ROM EQU $8000
ORG ROM
LDAA N1      *N1 est placé dans accA
ADDA N2      *N1 est ajouté, à N2, résultat dans accA
STAA RES     *résultat dans RES
BRA ROM
END
```

\* EXERCICE 2 : Soustraction de deux octets

\* N1 en \$40, N2 en \$41, résultat en \$42, signe en \$43 (\$00 si positif)

\* Le résultat est tronqué 8bits

\*

```
N1 EQU $40
N2 EQU $41
RES EQU $42
SIG EQU $43
ROM EQU $8000
ORG ROM
CLR SIG      *Positif par défaut
LDAA N1      *N1 est placé dans accA
SUBA N2      *N1 est ajouté à N2, résultat dans accA
STAA RES     *résultat dans RES
BGE POSI     *Si N1 > N2
COM SIG
POSI BRA ROM
END
```

\* EXERCICE 3 : Clignotement sur PA7

\*

```
PRA EQU $00      * les poids forts de l'adresse dans X
PA7 EQU %10000000 *bit 7
DDRA7 EQU $80     *idem
PACTL EQU $26
REGBASE EQU $1000
ROM EQU $8000
ORG ROM
LDX #REGBASE     *X pointe la base des registres
BSET PACTL,X DDRA7 *PA7 en sortie
BASCULE LDAA #%10000000 *basculer uniquement PA7
EORA PRA,X
STAA PRA,X
LDAB #$F         *Temporisation sur accB (environ 0.5s sur simul)
ENCORE DECB
BNE ENCORE
BRA BASCULE
END
```

\* EXERCICE 4 : Clignotement sur PA7 avec sous programme de tempo

\*

```
PRA EQU $00      * les poids forts de l'adresse dans X
PA7 EQU %10000000 *bit 7
DDRA7 EQU $80     *idem
PACTL EQU $26
REGBASE EQU $1000
ROM EQU $8000
ORG ROM
LDX #REGBASE     *X pointe la base des registres
BSET PACTL,X DDRA7 *PA7 en sortie
BASC LDAA #%10000000 *basculer uniquement PA7
EORA PRA,X
STAA PRA,X
BSR TEMPO
BRA BASC
TEMPO LDAB #$9     *Temporisation sur accB (environ 0.5s sur simul)
ENCORE DECB
BNE ENCORE
RTS
END
```

\* CD0298

\* EXERCICE 5 : Compteur decimal sur PRC

\*

```
ROM EQU $8000
DDRC EQU $1007
PRC EQU $1003
ORG ROM
CLR DDRC      *PRC en sortie
COM DDRC
CLR PRC
CLRA
CONTINU STAA PRC
INCA
TAB
ANDB #%00001111
CMPB #9
BLS CONT
ADDA #6
CONT CMPA #100
BLS CONTINU
CLRA
BRA CONTINU
END
```

\* EXERCICE 6 : Addition de deux mots de 16bits

\* N1 en \$40, N2 en \$42, résultat en \$44

\* Le résultat est tronqué, à 8bits

\*

```

N1 EQU $40
N2 EQU $42
RES EQU $44
ROM EQU $8000
ORG ROM
LDD N1 *N1 est placé dans accD
ADDD N2 *N1 est ajout, à N2, résultat dans accD
STD RES *résultat dans RES
BRA *
END

```

\* EXERCICE 7 : Recopie d'un message

\* Message en \$40, destination en \$50

\*

```

MESS ORG $40
      FCC 'MC68HC11'
      FCB 0 *fin du message
      ORG $50
DEST RMB 8 *r,serve 8 octets en RAM
FINMESS EQU *
ROM EQU $8000
ORG ROM
LDX #MESS *X contient l'adresse du caractère
LDY #DEST *Y contient l'adresse de destination
ENCORE LDAA 0,X
      BEQ FINI *si caractère =0 (fin du message)
      STAA 0,Y
      INX *pointe caractère suivant
      INY *pointe destination suivante
      BRA ENCORE
*
FINI BRA *
END

```

\* EXERCICE 8 : Recopie d'une zone mémoire

\* Début en \$40, destination en \$42 longueur en \$44

\*

```

ORG $40
SOURCE RMB 2
DEST RMB 2
LONG RMB 1
*
ROM EQU $8000
ORG ROM
LDX SOURCE *X contient l'adresse du caractère
LDY DEST *Y contient l'adresse de destination
LDAB LONG
ENCORE LDAA 0,X
      STAA 0,Y
      DECB
      BEQ FINI *si B=0 le transfert est fini
      INX *pointe caractère suivant
      INY *pointe destination suivante
      BRA ENCORE
*
FINI BRA *
END

```

\* EXERCICE 9 : Conversion HEXA ASCII

\* accA contient le nombre HEXA, le retour est dans D

\* dans cet exemple la conversion s'effectue en permanence...

\*

```

PILE EQU $FF *Pile s
NB EQU $40 *nombre à convertir en $40
RES EQU $50 *resultat
*
ROM EQU $8000
ORG ROM
LDS #PILE
CALC LDAA NB *accA contient NB
      BSR HEXASC
      STD RES
FINI BRA CALC
*
HEXASC PSHA *Garde pour les poids forts
      ANDA #%00001111 *Garde les poids faibles
      BSR CONVASC *Converti accA en ascii
      TAB *Les poids faibles vont dans accB
      PULA *On s'occupe des poids forts
      LSRA
      LSRA
      LSRA
      BSR CONVASC
      STD RES
      RTS
*
CONVASC EQU * *Converti accA (entre 0 et $F) en ascii
      ADDA #'0'
      CMPA #'9' *A,B,C,D,E,F ?
      BLS FINCNV
      ADDA #7 *différence entre la série 0..9 et A..F
FINCNV RTS
END

```

\* EXERCICE 10 : Clignotement sur PRC par RTI

\* Ne pas confondre Real Time Interrupt et Return Interrupt !!!!!

\*

```

ROM EQU $8000
PRC EQU 3 *le port C va "clignoter"
DDRC EQU 7
TMSK2 EQU $24 *contient RTI
TFLG2 EQU $25 *contient RTIF
PACTL EQU $26 *contient RTR1 et RTR0
RTI EQU %01000000 *bit de masque de l'IT RTI
RTIF EQU %01000000 *drapeau de l'IT RTI
DUREE EQU %00 *durée entre 2 RTI (pour RTR1 et RTR0)

```

```

REGBASE EQU $1000 *Base des registres
*
ORG ROM
LDX #REGBASE
LDAA #FF *PRC en sortie
STAA DDRC,X
STAA PRC,X *par exemple
BSET TMSK2,X RTI *autorise l'IT RTI
LDAA #DUREE *duree entre deux RTI
ORAA PACTL,X
STAA PACTL,X
CLI *
BRA * *autorise toutes les IT demasquee
*Ne fait plus rien
*
ITRTI COM PRC,X *complemente PRC
BSET TFLG2,X RTIF *efface le drapeau d'IT RTI
RTI
*
ORG $FFF0
FDB ITRTI *adresse du vecteur RTI
END

```

\* CD0298  
\* EXERCICE 11 : Clignotant avec l'IT OC3

```

*
ROM EQU $8000
TMSK1 EQU $22 *contient OC3I
TFLG1 EQU $23 *contient OC3F
PACTL EQU $26 *contient RTR1 et RTR0
TCTL1 EQU $20 *action sur PAX
OC3I EQU %00100000 *bit de masque de l'IT RTI
OC3F EQU %00100000 *drapeau de l'IT RTI
TOC3 EQU $1A *Registre TIMER 3
DUREE EQU $100 *duree entre 2 coincidences x0,5uS avec Q=8MHz
REGBASE EQU $1000 *Base des registres
*
ORG ROM
LDX #REGBASE
BSET TMSK1,X OC3I *autorise l'ITOC3
BSET TCTL1,X %00010000 *PA5 bascule à chaque IT
LDD #200 *pour avoir une simulation rapide
STD TOC3,X
CLI *
BRA * *autorise toutes les IT demasquees
*Ne fait plus rien
*
ITOC3 LDD #DUREE *ajoute la duree a TOC3
ADDD TOC3,X
STAA TOC3,X
BSET TFLG1,X OC3F *efface le drapeau d'IT
RTI
*
ORG $FFE4
FDB ITOC3 *adresse du vecteur OC3
END

```

\* CD0298  
\* EXERCICE 12 : Combine les exercices 10 et 11

```

*
ROM EQU $8000
TMSK1 EQU $22 *contient OC3I
TFLG1 EQU $23 *contient OC3F
PACTL EQU $26 *contient RTR1 et RTR0
TCTL1 EQU $20 *action sur PAX
OC3I EQU %00100000 *bit de masque de l'IT RTI
OC3F EQU %00100000 *drapeau de l'IT RTI
TOC3 EQU $1A *Registre TIMER 3
DUROC3 EQU $100 *duree entre 2 coincidences x0,5uS avec Q=8MHz
REGBASE EQU $1000 *Base des registres
PRC EQU 3 *le port C va "clignoter"
DDRC EQU 7
TMSK2 EQU $24 *contient RTI
TFLG2 EQU $25 *contient RTIF
RTI EQU %01000000 *bit de masque de l'IT RTI
RTIF EQU %01000000 *drapeau de l'IT RTI
DURRTI EQU %00 *duree entre 2 RTI (pour RTR1 et RTR0)
*
ORG ROM
LDX #REGBASE
*
* INITIALISE RTI
*
LDAA #FF *PRC en sortie
STAA DDRC,X
STAA PRC,X *par exemple
BSET TMSK2,X RTI *autorise l'IT RTI
LDAA #DURRTI *duree entre deux RTI
ORAA PACTL,X
STAA PACTL,X
*
* INITIALISE OC3
*
BSET TMSK1,X OC3I *autorise l'ITOC3
BSET TCTL1,X %00010000 *PA5 bascule à chaque IT
LDD #200 *pour avoir une simulation rapide
STD TOC3,X
*
* PROGRAMME PRINCIPAL
*
CLI *
BRA * *autorise toutes les IT demasquees
*Ne fait plus rien
*
* SOUS PROGRAMME IT TIMER OC3
*
ITOC3 LDD #DUROC3 *ajoute la duree a TOC3
ADDD TOC3,X
STAA TOC3,X
BSET TFLG1,X OC3F *efface le drapeau d'IT
RTI

```



```
*
* SOUS PROGRAMME IT RTI
*
ITRTI  COM  PRC,X      *complemente PRC
      BSET  TFLG2,X RTIF  *efface le drapeau d'IT RTI
      RTI
*
* POSITIONNEMENT DES VECTEURS D'IT
*
      ORG  $FFE4
      FDB  ITOC3      *adresse du vecteur OC3
*
      ORG  $FFF0
      FDB  ITRTI      *adresse du vecteur RTI
      END

* CD0298
* EXERCICE 13 : Calcul de sin(x) en degre (uniquement la partie decimale)
* pour les nombres de 0 à 9
* pour des raisons pratique les resultats sont donn,s
* en hexa (donc fauts)
*
ROM  EQU  $8000
N  EQU  $40      *nombre à calculer
RES  EQU  $50
*
      ORG  ROM
CALC  LDAA  N
      BSR  SINUS
      STD  RES
      BRA  CALC
*
SINUS  TAB
      ROLB
      LDX  #TABLE      *A dans B car seul ABX existe
      ABX      *B=Bx2
      LDD  0,X      *X va pointe le sinus de accA
      RTS      * qui va dans accD
*
TABLE  FDB  00,$0174,$0348      *table realisee à l'aide
      FDB  $0523,$0697,$0871      *d'une calculatrice
      FDB  $1045,$1218,$1391,$1564
      END
```