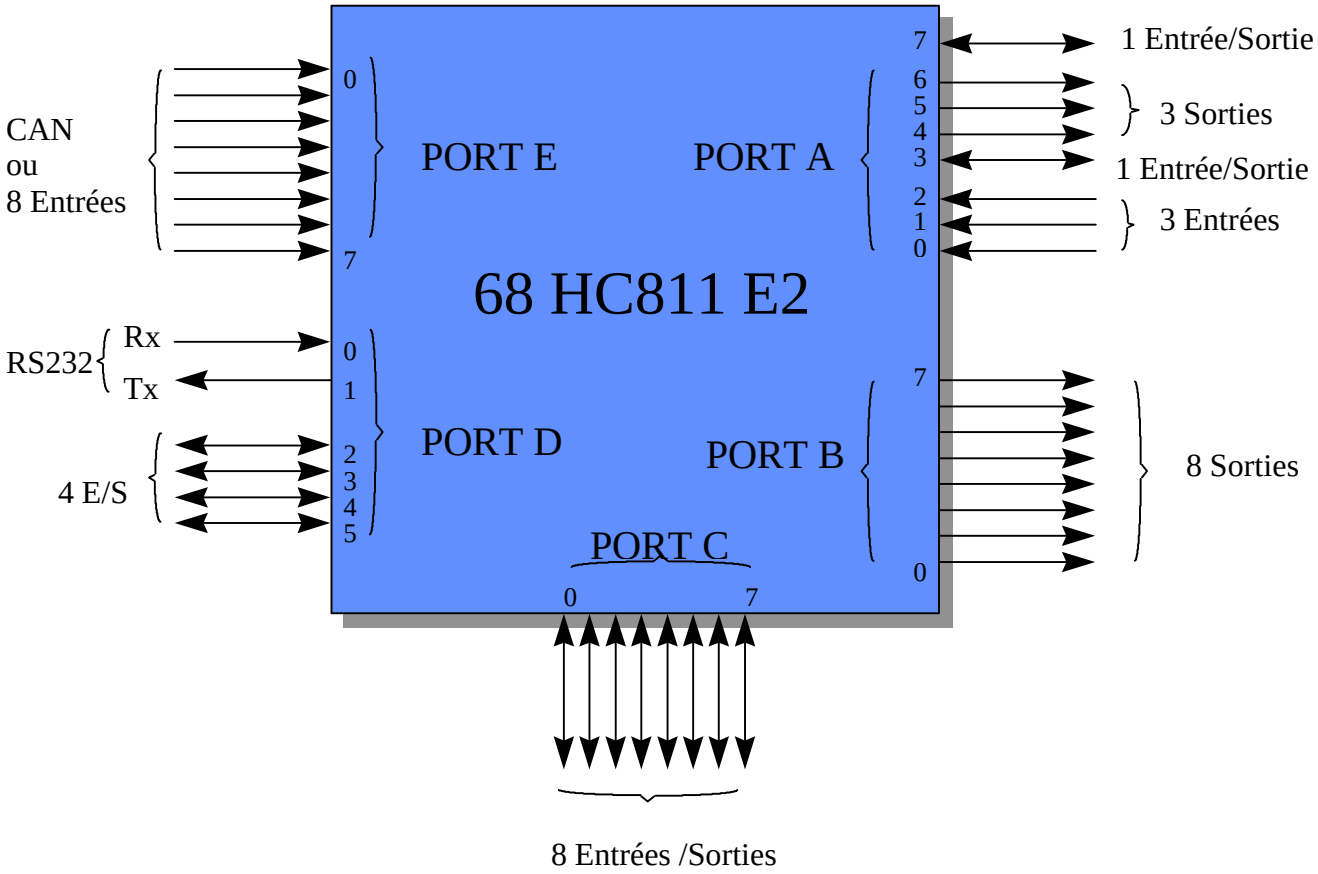
MICRO CONTROLEUR

68HC811

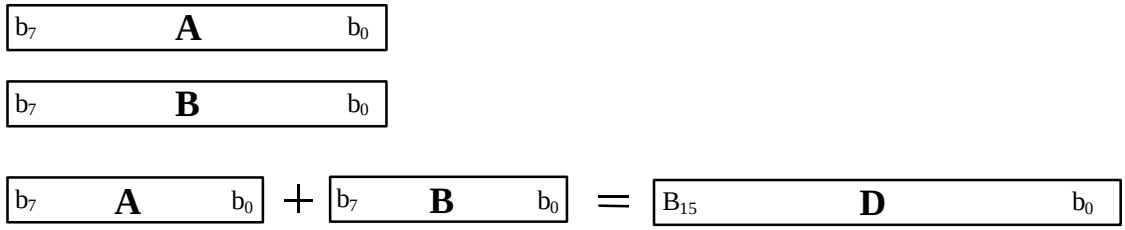


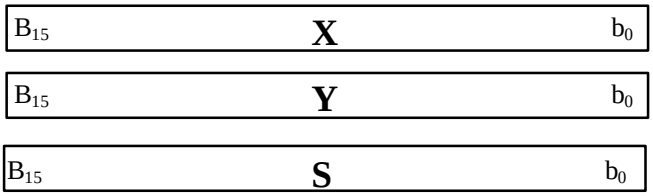
Documentation interne ELE

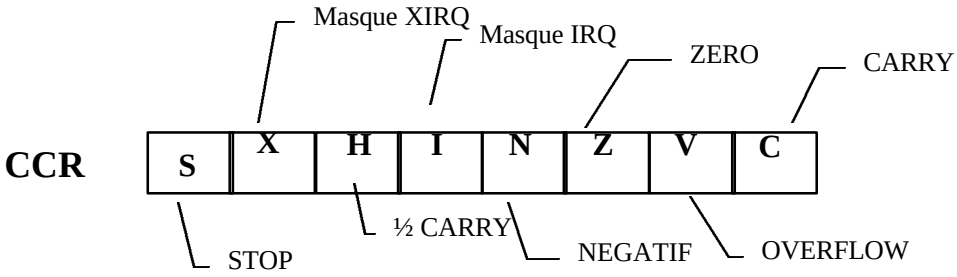
D. MENESPLIER 1998

68HC11

Accumulateurs et Instructions







<u>INSTRUCTIONS SPECIALES MICROCONTROLEUR</u> :
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 »

MICRO-CONTROLEUR 68HC11-A1

MODE BOOT: *mod A=0 et mod B = 0.*

■ Après le Reset, le micro démarre le programme "Boot loader" de 192 octets qui se trouve en ROM de l'adresse $BF40 à $BFFF. Ce programme charge les 256 octets qui arrivent sur la RS232 en RAM puis exécute ce programme à partir de l'adresse $0000.

Avec un PC et le logiciel de MONITORING : PCBUG11.exe on peut charger en RAM un programme de dialogue via la RS232 qui nous permet depuis le PC de faire diverses actions sur la mémoire du 68HC11. On écrit, on efface, on lit l'EEPROM interne de 512 octets de l'adresse $B600 à B7FF, et on peut lancer un programme à une adresse de cette EEPROM.

■ Si pendant le Reset on boucle T^x data et R^x data de la RS232, le micro démarre alors le programme en EEPROM . On peut donc faire tourner un programme écrit et testé avec PCBUG en autonome. Après 15 ms on peut oter le strapp Tx / Rx et récupérer ainsi la liaison RS232.

MODE NORMAL CIRCUIT SEUL : *mod A= 0 et mod B =1.*

Après le Reset, le micro démarre à l'adresse $FFFE / $FFFF comme tous les processeurs de la famille Motorola. Avec un 68HC811 il y à une EEPROM de 2K à ces adresses. Un programme écrit avec PCBUG dans cette EEPROM en mode BOOT peut maintenant tourner tout seul dans ce mode. Les autres types de 68HC11 n'ont soit pas de mémoire à cette adresse soit de la ROM programmable par masque soit de la PROM facilement programmable mais non effacable (OTPROM).

FONCTIONNEMENT de PCBUG11.EXE

Sous PCBUG:

Mode BOOT: ModA=0 et ModB=0.
Reset du micro.
Lancer PCBUG.EXE
Utliser les commandes pour écrire, effacer, lire l'EEPROM, lancer un prog.

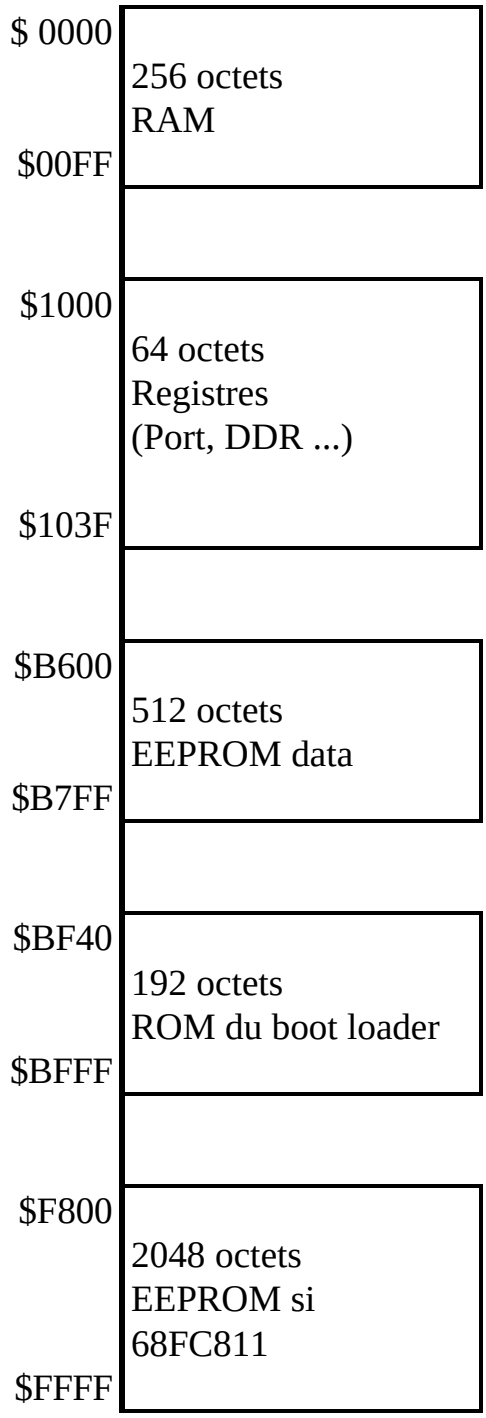
Micro seul:

68HC11-A1

Mode BOOT.
Reset du micro avec strapp pin 20 et 21.
Le programme de l'EEPROM s'exécute.
On peut éventuellement enlever le strapp Tx /Rx de la RS232.

68HC811

Mode NORMAL et SEUL : modA=0 et modB=1.
Reset du micro, le programme en EEPROM en $FFFE / $FFFF s'exécute.



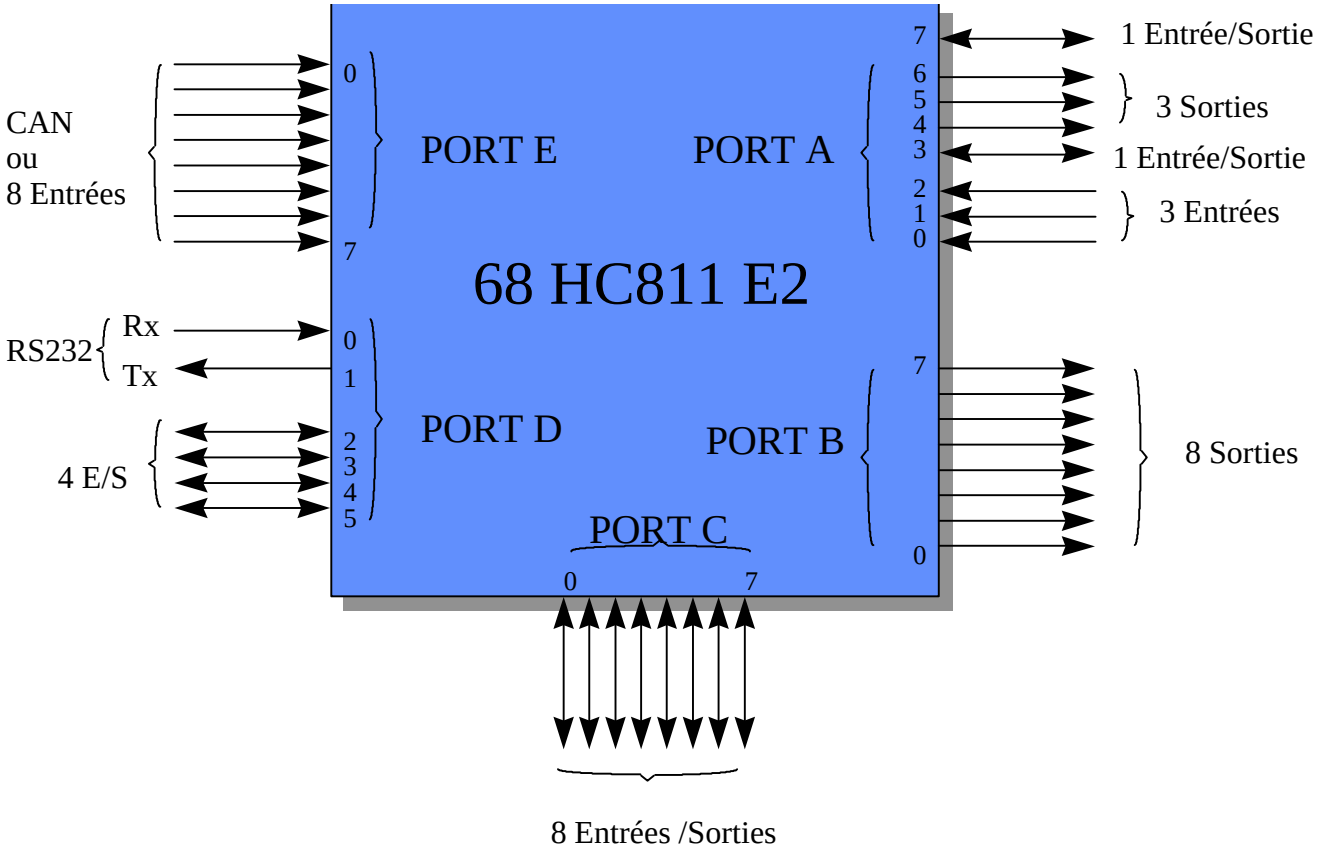
MODE NORMAL: Après un Reset le micro démarre à l'adresse qui est en $FFFE - $FFFF.

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.

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**

68HC11

CREATION D'UN PROGRAMME
ASSEMBLAGE
PROGRAMMATION DU CIRCUIT

◆ EDITION DU FICHIER SOURCE:

➔ ***EDIT NOMFICH.ASC***

◆ ASSEMBLAGE :

➔ ***ASMHC11 NOMFICH.ASC***

L'extension ASC est facultative
Création d'un fichier listing : NOMFICH.LST
Création d'un fichier objet au format Motorola: NOMFICH.S19

◆ MONITORING PAR PCBUG:

➔ ***PCBUG11.EXE -A*** Faire préalablement un RESET du microprocesseur en mode BOOT**.**

◆ EFFACEMENT DE TOUTE L'EEPROM :

➔ ***EEPROM ERASE BULK :*** efface tout l'EEPROM data de $B600 à$B7FF

➔ ***EEPROM ERASE BULK $F800 :*** efface toute l'EEPROM de $F800 à $FFF. Il faut auparavant avoir modifier le registre de protection en $1035 en mettant $10 à la place de $1F.

◆ CHARGEMENT DU PROGRAMME DANS L'EEPROM :

➔ ***LOADS NOMFICH.S19 :***L'extension S19 est facultative**.** Il faut auparavant avoir définit la zone EEPROM soit $B600 à $B7FF, soit $F800 à $FFFF.

◆ LANCEMENT DU PROGRAMME:

➔ G $B600

NB: Ce même programme peut tourner également sans le contrôle de PCBUG. Il suffit de faire un strapp pendant 15 ms entre T^x et R^x du microprocesseur avant le Reset .

68HC11 sous PCBUG

PCBUG11 -A : lance PCBUG avec le Talker du 68HC11 A1 sur COM 1.
PCBUG11 -A port=2 : lance PCBUG avec le Talker du 68HC11 A1 sur COM 2.

- La liaison RS232 vers le PC est branchée.
- On alimente la carte.
- Strapp entre T^x data et R^x data enlevé.
- Faire un Reset en mode BOOT (MODA = 0 et MODB = 0).
- Lancer PCBUG sur le PC par PCBUG11 -A ↵

SOUS PCBUG :

- " CTRL R " : permet de tester la communication.
- " RESTART" : Relance PCBUG. Faire avant un reset du micro.
- " CLS" : efface la fenêtre.

CHARGER UN PROGRAMME EN EEPROM :

- EEPROM ERASE BULK
- LOADS " Nomfichier.S19 " (extension S19 facultative)

MODIFIER UNE CASE MEMOIRE EN EEPROM :

- EEPROM ERASE ENABLE
- EEPROM $B600 $B6FF
- MM $B6xx

DESASSEMBLER UNE ZONE EN EEPROM :

- DASM $B600 $B6FF

LIRE ET AFFICHER EEPROM :

- MD $B600 $B6FF

LANCER UN PROGRAMME :

- G $B600

STOPPER UN PROGRAMME :

- S

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**

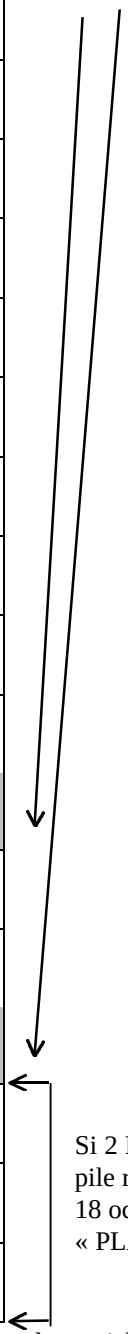
.

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)

Zones utilisées par PCBUG
Ne pas utiliser cette RAM
car « PLANTE » de PCBUG

$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



Si 2 IT imbriquées, la pile remonte de 9 + 9 soit 18 octets.
« PLANTE » de PCBUG

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.

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.

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

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

TIMER

$1028 = SPCR
$1029 = SPSR
$102A = SPDR

SPI

$102B = BAUD
$102C = SCCR1
$102D = SCCR2
$102E = SCSR
$102F = SCDR

UART

$1030 = ADCTL
$1031 = ADR1
$1032 = ADR2
$1033 = ADR3
$1034 = ADR4

CAN

$1035 à $1038 = RESERVE MOTOROLA
$1039 = OPTION
$103A = COPRST
$103B = PPROG
$103C = HPRIO
$103D = INIT
$103E = TEST1
$103F = CONFIG

LE TIMER du 68HC11

- 1 - <u>LE COMPTEUR</u> :

Il est constitué d'un compteur 16 bits, qui compte sans arrêt l'horloge du 68HC11. Il part de $0000 jusqu'à $FFFF puis déborde et revient à $0000 en positionnant un bit de débordement.
Il est précédé d'un prédiviseur programmable.

PR1	PR0	Div
0	0	1

TMSK2 : $1024

						PR1	PR0

0	1	4
1	0	8
1	1	16

Les bits PR0 et PR1 sont des bits protégés. Ils doivent impérativement être modifiés pendant les 64 cycles d'horloge qui suivent le Reset.
Avec un quartz de 8 Mhz on a E= 2 Mhz donc T=500ns. La capacité max du comptage est : 65536 x 16 x 500ns =524,288 ms si Prédiv = 16.

Le registre 16 bits du compteur n'est accessible qu'en lecture seulement.

TCNT : $100E

b_{15}							B_8

= MSB

$100F

B_7							B_0

= LSB

Le compteur possède un bit d'état et un bit d'autorisation d'interruption.

TFLG2 : $1025

TOF							

Le bit TOF se positionne à 1 quand il y a débordement du compteur. Il reste dans cet état tant qu'il n'est pas remis à zéro par l'utilisateur. Pour cela il faut écrire un "1" en position b7 dans TFLG2.
NE PAS UTILISER : BSET TFLG2,X,#$80 **qui risque de modifier d'autres bits.**
UTILISER: BCLR TFLG2,X,#$7F ou LDAA #$80 suivi de STAA TFLG2,X

TMSK2 : $1024

TOI							

Le bit TOI mis à 1 autorise les interruptions produites par le passage à 1 du bit TOF signalant le débordement du compteur.

- 2 - LES REGISTRES DE CAPTURE en ENTREE :

Il y a 3 entrées de capture : IC1 IC2 et IC3 qui sont sur le port A en PA2 PA1 et PA0.
Quand une transition, que l'utilisateur peut choisir, arrive sur l'entrée de capture, la valeur du timer est recopiée dans une registre 16 bits : TIC1 TIC2 ou TIC3. Une interruption est générée et un bit d'état IC1F IC2F ou IC3F est positionné.

TIC1 : $1010

b_{15}							B_8

B_7							B_0

\$1011

TIC2 : \$1012

b_{15}							B_8

\$1013

B_7							B_0

TIC3 : \$1014

b_{15}							B_8

\$1015

B_7							B_0

Les bits IC1F IC2F et IC3F passent à 1 quand une transition valide est générée sur l'entrée de capture correspondante. Pour les remettre à zéro il faut écrire dans TFLG1 un 1 à la place du bit correspondant. On peut remettre plusieurs bits à zéro en même temps. **NE PAS UTILISER L'INSTRUCTION BSET (voir remarque précédente pour TFLG2).**

TFLG1 : \$1023

					IC1F	IC2F	IC3F

Les bits IC1I IC2I et IC3I quand ils sont mis à 1 autorisent la génération d'une interruption quand l'entrée de capture reçoit une transition valide.

TMSK1 : \$1022

					IC1I	IC2I	IC3I

Le choix du type de transition qui sera active sur l'entrée de capture, se fait par le registre TCTL2.

TCTL2 : \$1021

		Edge1B	Edge1A	Edge2B	Edge2A	Edge3B	Edge3A

Edge B	Edge A	Transition active
0	0	Invalide
0	1	Front montant
1	0	Front descendant
1	1	Front montant et descendant

- 3 - LES REGISTRES DE COMPARAISON en SORTIE :

Il y a 5 sorties de comparaison OC1 OC2 OC3 OC4 et OC5 qui sont sur le PORT A en PA3 PA4 PA5 PA6 et PA7.

Il y a 5 registres 16 bits qui sont en permanence comparés avec le compteur 16 bits du timer :

TOC1 en \$1016 et \$1017
TOC2 en \$1018 et \$1019
TOC3 en \$101A et \$101B
TOC4 en \$101C et 101D
TOC5 en \$101E et 101F

b_{15}							B_8
B_7							B_0

OC5 agit sur PA3
OC4 agit sur PA4
OC3 agit sur PA5
OC2 agit sur PA6
OC1 agit sur PA7 ou sur PA6/PA5/PA4/PA3

Dés qu'il y a égalité, un bit d'état est positionné, une IT est générée et la sortie correspondante change d'état de façon programmable.

TFLG1 en $1023

OC1F	OC2F	OC3F	OC4F	OC5F			

Les bits d'état OCF passent à 1 quand une comparaison est valide (égalité entre Timer et Registre de comparaison).
Pour remettre ces bits à zéro il faut écrire dans TFLG1 avec le bit correspondant à 1. **NE PAS UTILISER L'INSTRUCTION BSET (voir remarque précédente pour TFLG2)**

TMSK1 en $1022

OC1I	OC2I	OC3I	OC4I	OC5I			

Quand ils sont positionnés à 1, ces bits OCI autorisent les IT produites par le passage à 1 des bits d'état OCF.

TCTL1 en $1020

OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5

Ces bits définissent le comportement de la sortie de comparaison correspondante (pour OC2 OC3 OC4 et OC5).

OM	OL	Action après comparaison valide
0	0	Pas d'action.
0	1	Changement d'état à chaque fois.
1	0	Mise à zéro.
1	1	Mise à 1.

Sortie OC1 : elle peut agir sur n'importe quelle pins du PORT A entre PA3 et PA7.

OC1M en $100C

OC1M7	OC1M6	OC1M5	OC1M4	OC1M3			

Si le bit OC1Mx est mis à 1 la sortie de comparaison agira sur le port correspondant PAx. On peut faire agir OC1 sur plusieurs pins de sortie.

OC1D en $100D

OC1D7	OC1D6	OC1D5	OC1D4	OC1D3			

Les bits OC1Dx définissent l'état de la pin de sortie quand la comparaison est valide (soit 1 soit zéro) à condition que OC1M l'autorise.

- 4 - <u>LES INTERRUPTIONS TEMPS REEL = RTI</u>:

La source d'horloge est la fréquence du micro E divisée par 2^{13} soit une période entre deux interruptions de : 0,5ms x 8192 = 4,1 ms.
Les bits RTR1 et RTR0 permettent de diviser par 2, 4 ou 8 cette fréquence.

PACTL en $1026

						RTR1	RTR0

RTR1	RTR0	DIV	Période RTI Quartz 8 Mhz
0	0	1	4,1 ms
0	1	2	8,2 ms

1	0	4	16,4 ms
1	1	8	32,8 ms

TFLG2 en $1025

	RTIF						

Le bit RTIF passe à 1 quand une interruption temps réel arrive. On doit impérativement le remettre à zéro avant de quitter le SP d'interruption. Pour cela on écrit un 1 en position b_6 dans TFLG2. **NE PAS UTILISER L'INSTRUCTION BSET (voir remarque précédente pour TFLG2)**

TMSK2 en $1024

	RTII						

Quand le bit RTII est mis à 1 les interruptions temps réel sont autorisées. Quand il est à zéro, les IT sont inhibées mais la RTI fonctionne et positionne toujours le bit RTIF du registre TFLG2.

- 5 - L'ACCUMULATEUR D'IMPULSIONS:

Il est composé d'un TIMER de 8 bits qui peut soit compter les impulsions qui lui sont appliquées sur son entrée : PA_7 soit ouvrir par commande sur PA_7 une porte qui autorise le comptage du signal à la fréquence E / 64.
Deux sources d'IT sont disponibles. Une quand le compteur déborde de $FF à $00. L'autre quand une transition valide est détectée sur PA_7.

PACTL en $1026

DDRA7	PAEN	PAMOD	PEDGE				

Bit 4 : PEDGE = si à 0, PA7 réagit à des fronts descendants ou ouvre la porte si PA7=0.
Bit 5 : PAMOD= si à 0 mode comptage d'impulsion. Si à 1 mode ouverture porte.
Bit 6 : PAEN = à 1 autorise l'accumulateur à fonctionner. A 0 le compteur est inhibé.
Bit 7 : DDRA7= si à 0 la pin est configurée en entrée. Si à 1 c'est une sortie (OC1 du timer).

TFLG2 en $1025

		PAOVF	PAIF				

Bit 4 : PAIF = ce bit est mis à 1 l'ors de la détection d'une transition valide sur PA7.
Bit 5 : PAOVF = ce bit est mis à 1 quand le compteur déborde de $FF à $00.
Ces deux bits doivent être remis à zéro, par écriture d'un 1 en position 4 ou 5 de TFLG2.

TMSK2 en $1024

		PAOVI	PAII				

Mis à 1 ces bits autorisent les IT quand les bits PAOVF et PAIF passent à 1.
Ces bits à zéro inhibent les IT provoquées par ces deux flags.

PWM avec le 68HC11

Il y a 5 Sorties de comparaison :

OC1: quand le registre TOC1 = TIMER alors action sur une ou plusieurs sorties parmi : PA7 PA6 PA5 PA4 ou PA3.
OC2: quand TOC2 = TIMER alors action sur PA6.
OC3: quand TOC3 = TIMER alors action sur PA5.
OC4: quand TOC4 = TIMER alors action sur PA4.
OC5: quand TOC5 = TIMER alors action sur PA3.

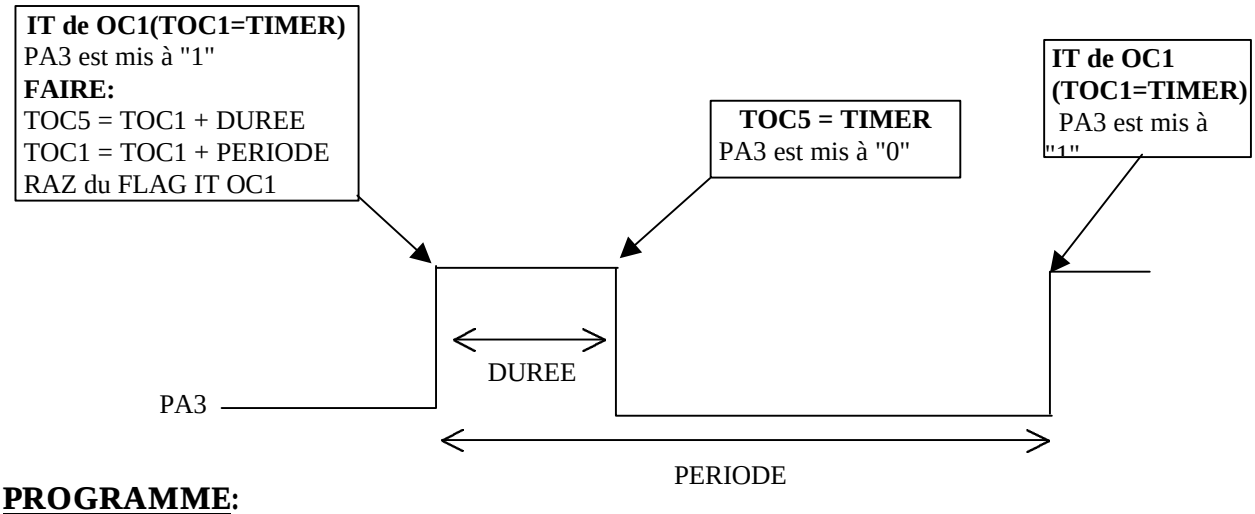
Le type d'action est : Mise à "1" ou Mise à "0" ou Changement à chaque comparaison.

EXEMPLE de PWM sur PA3:

OC1 agira sur PA3 et son action mettra cette sortie à "1".

OC5 agira sur PA3 et son action mettra cete sortie à "0".

On utilise les IT et OC1 déclenchera une IT quand TOC1 = TIMER



PROGRAMME:

```
INIT  BSET  TMSK2,X,#$02  ; prédiv=8 donc TIMER = 4 µs avec Xtal de 8 MHz.
      BSET  OC1M,X,#$08   ; bit 3 de OC1M mis à "1": OC1 agit sur PA3.
      BSET  OC1D,X,#$08   ; bit 3 de OC1D mis à "1": OC1 mettra PA3 à "1".
      BSET  TCTL1,X,#$02  ; OC5 mettra PA3 à "0".
      LDD   #$00FA        ; valeur sur 16 bits.
      STD   PERIODE       ; Période = 250 x = 4 µs =1 ms.
      STD   TOC1,X        ; initialisation du registre TOC1 avec la valeur de la période.
      LDD   #$valeur      ; valeur sur 16 bits de $000D (min) à $00FA (max=période).
      STD   DUREE         ; Durée = valeur x = 4 µs. Valeur variable pour PWM.
      BSET  TMSK1,X,#$80  ; autorisation de IT par OC1.
      CLI                 ; Autorisation des IT.
```

***** SOUS PROGRAMME IT par OC1 *****

```
OC1   LDD   TOC1,X
      ADDD DUREE
      STD   TOC5,X        ; TOC5 = TOC1 + DUREE.
      LDD   TOC1,X
      ADDD PERIODE
      STD   TOC1,X        ; TOC1 = TOC1 + PERIODE.
      BCLR TFLG1,X,#$7F   ; RAZ du flag de OC1 qui passe à "1" quand TOC1 = TIMER.
      RTI                 ; on ne peut pas utiliser BSET pour mettre b7 à "1"
```

Convertisseur Analogique Numérique

Le CAN est précédé d'un multiplexeur à 8 entrées. Il s'agit d'un convertisseur 8 bits et sa précision est meilleure que ± ½ bit le moins significatif sur toute la plage de température acceptable par le circuit.
Il est programmable à l'aide de deux registres (OPTION et ADCTL) et les résultats sont disponibles dans 4 registres (ADR1 ADR2 ADR3 ADR4).

ADR1 : $1031
ADR2 : $1032
ADR3 : $1033
ADR4 : $1034

OPTION : $1039

ADPU	CSEL						

Le bit **CSEL** est le Clock select. Pour une horloge E normale ce bit est mis à « o ». Par contre si la fréquence de E est trop lente (< 750 Khz) ce bit dit être mis à « 1 » pour que la cellule RC prévu sur la puce soit mise en service.

Le bit **ADPU** doit être mis à « 1 » pour valider le fonctionnement du convertisseur. S'il est à « 0 » le convertisseur est considéré comme inactif.

ADCTL : $1030

CCF		SCAN	MULT	CD	CC	CB	CA

Le bit **MULT** pour multiple channel.
Si ce bit est à « 0 » une seule entrée est convertie. Les bits CA/CB et CC définissent cette entrée.
Si ce bit est à « 1 » quatre entrées d'un même groupe défini par CC sont converties. Si CC = 0 il s'agit du 1er groupe : PE_0 PE_1 PE_2 et PE_3 si CC=1 alors le 2eme groupe est converti : PE_4 PE_5 PE_6 et PE_7.

Le bit **SCAN** s'il est à « 1 » permet la conversion en continu. Quand il est à « 0 » une seule conversion est effectuée.

Le bit **CCF** est un Flag qui doit être lu. Il est à « 1 » quand la conversion est terminé et que les registres de données contiennent des data valides. Pour remettre ce bit à « 0 » il faut lire une fois le registre ADCTL.

Bit CD = 0 :

CC	CB	CA	Entrée	Résultat
0	0	0	PE_0	ADR1
0	0	1	PE_1	ADR2
0	1	0	PE_2	ADR3
0	1	1	PE_3	ADR4
1	0	0	PE_4	ADR1
1	0	1	PE_5	ADR2
1	1	0	PE_6	ADR3
1	1	1	PE_7	ADR4

Bit CD = 1 : permet de mesurer Vref H et Vref L

EEPROM DATA

Les 512 octets d'EEPROM entre $B600 et $B7FF sont réservés pour sauvegarder des données. La gestion de cette zone mémoire se fait avec le registre PPROG.

PPROG : $103B

ODD	EVEN		BYTE	ROW	ERASE	EELAT	EEPGM

Le bit **EEPGM** s'il est à « 1 » permet le génération de la tension interne de programmation Vpp.

Le bit **EELAT** permet de latcher l'EEPROM. Quand il est à « 0 » la mémoire peut être lue. Quand il est à « 1 » la mémoire se comporte comme si elle avait été enlevée de l'espace adressable du 68HC11 et mise sur un programmateur. Dans ce cas l'accès en lecture est impossible.
Un programme qui écrit ou qui efface l'EEPROM ne peut pas être contenu dans cette l'EEPROM.

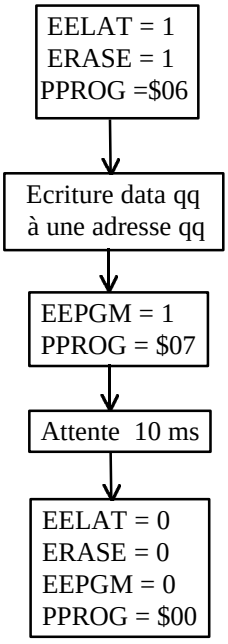
Le bit **ERASE** permet s'il est à « 1 » d'effacer l'EEPROM. A « 0 » il permet la lecture ou l'écriture.

Les bits **ROW** et **BYTE** sélectionnent le mode d'effacement quand le bit ERASE = 1.

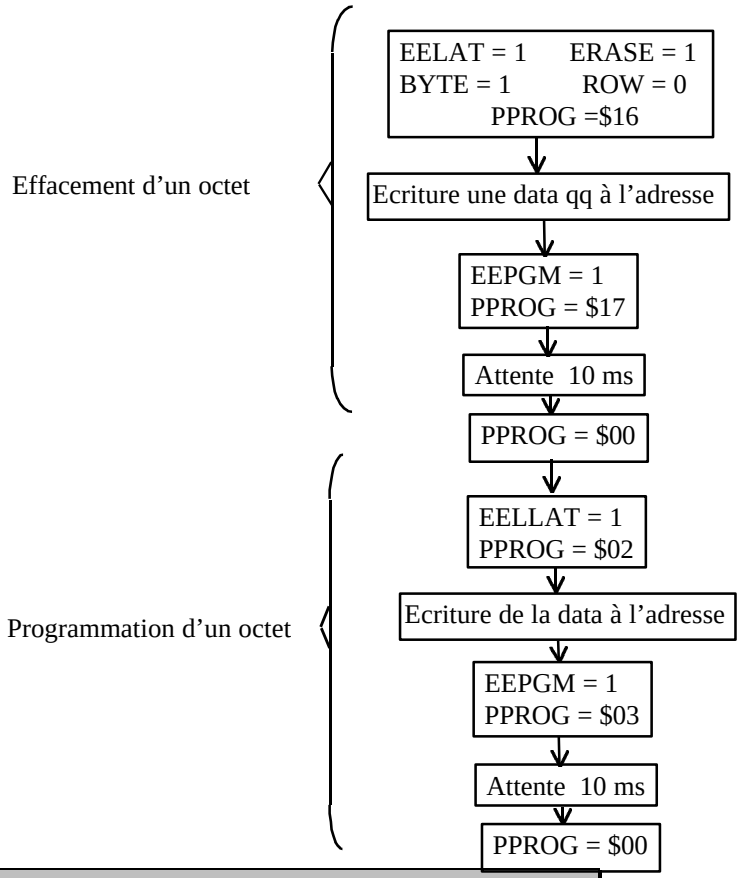
BYTE	ROW	Type d'effacement
0	0	Effacement complet.
0	1	Effacement 16 octets d'une ligne.
1	0	Effacement d'un octet.
1	1	Effacement d'un octet.

Les bits **ODD** et **EVEN** ne sont utilisés que par Motorola en phase de test.

EFFACEMENT TOTAL



EFFACEMENT OCTET AVANT PROGRAMMATION



SOUS PROGRAMMES GESTION EEPROM

```
*********************************
**       SP   ERASE BYTE       **
**   X pointe adresse à effacer **
*********************************
CLBYT       PSHA
            LDAB #$16
            STAB PPROG
            STAB 0,X
            LDAB #$17
            STAB PPROG
            JSR  DLY10
            CLR  PPROG
            PULA
            RTS
```

```
*********************************
**       SP   ERASE TOUT       **
**       512 octets EEPROM     **
**         $B600 à $B7FF       **
*********************************
CLTOU       LDAB #$06
            STAB PPROG
            STAB $B600
            LDAB #$07
            STAB PPROG
            JSR  DLY10
            CLR  PPROG
            RTS
```

```
*********************************
**       SP   ECRITURE         **
**   X pointe adresse à programmer **
**   A = DATA à programmer     **
*********************************
ECRIT       LDAB #$02
            STAB PPROG
            STAA 0,X
            LDAB #$03
            STAB PPROG
            JSR  DLY10
            CLR  PPROG
            RTS
```

```
*********************************
**          SP   DELAI         **
**     durée de 10 ms environ  **
*********************************
DLY10       PSHX
            LDX  #$1000
BOU         DEX
            BNE  BOU
            PULX
            RTS
```

SPI

La SPI : Serial Peripheral interface est une interface de type série synchrone. Elle permet la connexion sous forme série de plusieurs circuits du type Maître / Esclave comme par exemple les circuits I^2C.
Quatre pattes du 68HC11 sont réservées pour cette SPI sur le PORTD.
PD_2 = MISO : C'est une entrée si le 68HC11 est configuré en maître et une sortie si on est en esclave.

PD3 = MOSI : C'est une sortie si le 68HC11 est configuré en maître et une entrée si on est en esclave.

PD4 = SCK : C'est l'horloge fournie par le maître pour synchroniser les échanges de données. C'est une entrée pour un esclave.

PD5 = $\overline{\text{SS}}$: C'est une entrée de sélection du mode. Sur un esclave cette patte est à « 0 ». Sur un maître cette patte peut être soit une entrée soit une sortie en fonction du bit DDR5. Si DDR5 = 0 alors $\overline{\text{SS}}$ est une entrée de détection de défaut de transmission. Un passage de « 1 » à « 0 » signale au 68HC11 qu'un esclave veut devenir maître et qu'il doit cesser de transmettre. Si DDR5 = 1 alors $\overline{\text{SS}}$ est une sortie à usage général PD_5.

Avant d'utiliser la SPI on doit initialiser le registre de direction du port D.

DDRD : $1009

		DDR5	DDR4	DDR3	DDR2		

DDR2 : Ce bit donne le sens de la patte MISO. Il est mis à « 1 » pour un esclave, afin d'avoir la patte configurée en sortie. Pour un maître cette patte est une entrée quelque soit la valeur du bit DDR2.
DDR3 : Ce bit donne le sens de la patte MOSI. Il est mis à « 1 » pour un maître, afin que cette patte soit configurée en sortie. Pour un esclave cette patte est toujours une entrée quelque soit DDR3.
DDR4 : Ce bit détermine le sens de la patte SCK. Pour un maître on doit la mettre à « 1 » pour avoir une sortie. En mode esclave cette patte est une entrée quelque soit l'état du bit DDR4.
DDR5 : Ce bit donne le sens de la patte $\overline{\text{SS}}$. En mode esclave cette patte est toujours une entrée quelque soit l'état de ce bit. En mode maître si DDR5 = 1 la patte $\overline{\text{SS}}$ est une sortie d'usage général, et si DDR5 = 0 cette patte est une entrée de détection des défauts de transmission.

SPCR : $1028

SPIE	SPE	DWO	MSTR	CPOL	CPHA	SPR1	SPR0

C'est le registre de contrôle qui permet de choisir le mode de fonctionnement de l'interface.
Les bits **SPR0** et **SPR1** permettent de programmer la vitesse de transmission.

SPR1	SPR2	E Divisée par	Fréq. pour E = 2 Mhz
0	0	2	1 Mhz
0	1	4	500 Khz
1	0	16	125 Khz
1	1	32	62,5 Khz

Le bit **CPHA** permet de choisir la phase de l'horloge. En général CPHA = 1.

Le bit **CPOL** à « 1 » impose que le SCK est inactif à l'état haut. Si ce bit est à « 0 » alors SCK est inactif à l'état bas. C'est le choix pour le protocole I^2C (les data changent quand SCK =0)

Le bit **MSTR** doit être à « 1 » pour un maître et à « 0 » pour un esclave. Il est à « 0 » après un reset.

Le bit **DWOM** à « 0 » configure le sorties du port D en normal. Ce bit à « 1 » permet le OU cablé.

Le bit **SPE** permet de mettre en service la SPI s'il est à « 1 ». Avec ce bit à « 0 » la SPI est passive et le port D est disponible.

Le bit **SPIE** permet s'il est à « 1 » la génération d'interruption par la SPI du type WCOL ou bien SPIF (voir les bits correspondants dans le registre SPSR).

SPSR : $1029

SPIF	WCOL		MOD				

C'est le registre de status.
Bit **MOD** : si plusieurs maîtres en même temps.
Bit **WCOL** : Si tentative d'écriture dans le registre de DATA pendant une transmission.
Bit **SPIF** : Flag qui signale que le transfert est terminé entre le 68HC11 et le circuit externe. Si les IT sont autorisées par le bit SPIE du registre SPCR une interruption est générée.

SPDR : $ 102A

C'est le registre DATA dans lequel on vient lire les données reçues ou bien écrire les données à transmettre. C'est le registre buffer du registre à décalage 8 bits du sérialisateur/ désérialisateur.

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 ».

UART

L'interface série asynchrone peut être programmée par 5 registres :

SCDR : $102F

C'est le registre de DATA. Une écriture à cette adresse met la DATA dans le registre TDR, et une lecture de cette adresse permet de lire la données reçue qui est dans RDR.

BAUD : $102B

		SCP1	SCP0		SCR2	SCR1	SCR0

Les bits **SCP** fixent la vitesse max en fonction du quartz de l'oscillateur du CPU.

SCP1	SCP0	Div	X^{tal} 8 Mhz	X^{tal} : 4,9152 Mhz
0	0	1	125 K baud	**76,8 K baud**
0	1	3	41,667 K Baud	25,6 K baud
1	0	4	31,250 K baud	**19,2 K baud**
1	1	13	**9600 Baud**	5908 Baud

Les bits **SCR** permettent d'obtenir la vitesse qui est un sous multiple de la vitesse max déterminée par les bits SPR en fonction du quartz.

SCR2	SCR1	SCR0	DIV	Max : 9600	Max : 19200	Max : 76,8 K
0	0	0	1	9600 Baud	19200 Baud	76,8 K Baud
0	0	1	2	4800 Baud	9600 Baud	38,4 K Baud
0	1	0	4	2400 Baud	4800 Baud	19200 Baud
0	1	1	8	1200 Baud	2400 Baud	9600 Baud
1	0	0	16	600 Baud	1200 Baud	4800 Baud
1	0	1	32	300 Baud	600 Baud	2400 Baud
1	1	0	64	150 Baud	300 Baud	1200 Baud
1	1	1	128	75 Baud	150 Baud	600 Baud

Les deux registres de contrôle :

SCCR1 : $102C

R8	T8		M	WAKE			

SCCR2 : $102D

TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

Le bit **WAKE** : permet le choix de deux modes de réveil du R^x. Un « 0 » fait que le R^x se réveille si la ligne réception reste inactive pendant au moins un caractére. Un « 1 » permettra le réveil si le MSB reçu (8^{eme} ou 9^{eme} bit) est un « 1 ».
Le bit **M** : C'est le format de transmission. Un « 1 » fixe : 1 start + 9 bits + 1stop. Avec un « 0 » on aura : 1 start + 8 bits + 1 stop.
Le bit **R8** : Contient le 9^{eme} bit reçu quand ce format a été sélectionné par le bit M.
Le bit **T8 :** C'est ce bit que l'on écrit et qui est transmis comme 9^{eme} bit.
Le bit **SBK :** Si ce bit est à « 1 » le T^x envoie en permanence des *BREAK* sur la ligne, c'est à dire des blocs de 10 zéros logiques. Un récepteur qui reçoit un tel signal ne voit pas de bit STOP (niveau logique « 1 ») après les 8 bits de data et produit donc une « *FRAMING ERROR* « . Ce bit à « 0 » permet un fonctionnement normal du T^x.

Le bit **RWU :** si ce bit est à « 1 », le R^x est en sommeil. Il se réveillera en fonction du choix fait avec le bit *WAKE.* Si ce bit est à « 0 » l'interface fonctionne normalement.

Le bit **RE :** S'il est à « 0 », le récepteur est désactivé. Quand il est à « 1 », le récepteur fonctionne normalement.

Le bit **TE :** S'il est à « 0 », l'émetteur est désactivé et la ligne PD1 peut alors être récupérée et passe sous contrôle de DDRD1. Quand il est à « 1 » le T^x fonctionne normalement.

Le bit **ILIE :** S'il est à « 1 », il autorise les IT causées par le bit IDLE du registre d'état.
Le bit **RIE :** S'il est à « 1 », il autorise les IT causées par le récepteur, donc par les bits RDRF et OR du registre d'état.
Le bit **TCIE :** S'il est à « 1 », il autorise les IT si une transmission est complètement terminée, donc causées par le bit TC du registre d'état.
Le bit **TIE :** S'il est à « 1 » il autorise les IT si le registre de transmission est vide, donc causées par le bit TDRE du registre d'état.

SCSR : $102E

TDRE	TC	RDRF	IDLE	OR	NF	FE	

C'est le registre d'état ou STATUS.
Le bit **FE** : S'il est à « 1 » , il indique une erreur de format qui est causée par exemple par la non réception du STOP.

Le bit **NF** : Il est mis à « 1 » si du bruit est détecté sur la ligne.

Le bit **OR** : Il est mis à « 1 » si un débordement du R^X apparaît, c'est à dire si une data arrive et que la précédente n'a pas été lue.

Le bit **IDLE** : Il est mis à « 1 » si la ligne réception reste au repos, c'est à dire au niveau haut pendant au moins la durée d'un caractère (10 ou 11 bits suivant le bit M).

Le bit **RDRF** : Il est mis à « 1 » quand le registre de réception des données est plein et qu'il peut donc être lu. Ce bit est remis à « 0 » après une lecture du SCSR suivie d'une lecture de SCDR.

Le bit **TC** : Il est mis à « 1 » quand une transmission de données est complètement terminée (registre de sérialisation vidé).

Le bit **TDRE** : Il est mis à « 1 » pour indiquer que le registre de transmission SCDR est vide, et qu'une nouvelle data peut donc être chargée.

MACRO sous PCBUG

SOUS PCBUG :

Taper la commande : **DEFM NOM-MACRO** *ou bien* **DEFM AUTOSTART** *(si on veut que cette macro se lance automatiquement au démarrage de PCBUG).*

Une fenêtre s'ouvre et on peut taper directement les commandes. Attention les directives **BEGIN** et **END** sont ajoutées automatiquement par PCBUG.

***Exemple*: EEPROM $F800 $FFFF**
MM $1035 $10
EEPROM ERASE BULK $F800

Terminer par la touche **ESC** puis la touche ↵

RETOUR A PCBUG :
Sauver la macro dans une librairie par la commande : **SAVEM NOM-LIBRAIRIE**

LANCEMENT D'UNE MACRO : deux méthodes :

➔ Soit :

-1- *Taper la commande* : **LOADM NOM-LIBRAIRIE**

-2- *Taper ensuite :* **NOM-MACRO** *puis* ↵

➔ Soit :

-1- *Mettre dans PCBUG.bat la ligne suivante* : **MACRO=NOM-LIBRAIRIE**

-2- *Après le lancement de PCBUG taper* : **NOM-MACRO .** *Si la macro* **AUTOSTART** *fait partie de la librairie chargée, elle se lancera automatiquement.*

EXEMPLE :

La librairie **MENES.MCR** contient trois macro : AUTOSTART EFF et GO

```
DEFM AUTOSTART              DEFM EFF                          DEFM GO
BEGIN                       BEGIN                             BEGIN
      eeprom $F800 $ffff          EEPROM ERASE BULK $F800             G $F800
      mm $1035 $10          END                               END
END
```

Le fichier PCBUG.bat est le suivant : **PCBUG11.EXE-A PORT=2 MACRO=MENES**

Au lancement de PCBUG la librairie MENES est chargée et la macro AUTOSTART qu'elle contient est exécutée automatiquement (définit la zone EEPROM et ôte la protection écriture).
Si on veut effacer l'EEPROM il suffit de taper **EFF** qui lancera la macro correspondante.
Pour lancer le programme à l'adresse $F800 il suffit de taper **GO**.

68HC811 et PCBUG
PRISE EN MAIN

PCBUG:

Sous DOS créer un répertoire (ex: 68HC11) et y copier le fichier **ENAC6811.exe**. Ce fichier contient tous les fichiers compressés de **PCBUG11.exe** ainsi que le cross assembleur **ASMHC11.exe** de

MOTOROLA. L'exécution de ce fichier décompressera l'archive et créera tous les fichiers nécessaires ainsi que les talkers des différents 68HC11.

Pour lancer PCBUG11.exe il faut lancer un fichier *.bat (par exemple **PCBUG.bat**) qui charge le bon Talker , qui détermine le port série utilisé et qui charge éventuellement une bibliothèque de MACRO.

Exemple : PCBUG.bat = PCBUG11.exe -a MACRO=MENES PORT=1

PCBUG va démarrer sur le PORT COM1, utilisera le Talker du 68HC11E2 et la bibliothèque de MACRO nommée MENES.mcr sera chargée.

FAIRE un PROGRAMME:

Lancer l'éditeur du DOS en tapant "**EDIT nom-prog.asc**". L'extension du nom de programme est importante pour l'assembleur. Après avoir entré le programme, le sauver et quitter l'éditeur.

L'assemblage se fait en tapant "**ASMHC11 nom-prog**". Le cross assembleur attend un nom de programme ayant pour extension *.asc. Il va créer un fichier listing : **nom-prog.lst** dans lequel il signalera les éventuelles erreurs d'assemblage, et un fichier objet: **nom-prog.s19** qui sera directement chargeable par PCBUG dans l'EEPROM du 68HC11

68HC11:

La maquette comportant le 68HC11 doit être reliée au Port COM1 et le microcontrôleur doit fonctionner en mode BOOT (pin MOD B à "0").

Faire un reset du micro. Puis lancer PCBUG.bat en tapant "**PCBUG**".

- Sous PCBUG définir la zone EEPROM en tapant : "**EEPROM $F800 $FFFF**"
- Lever la protection de l'EEPROM en tapant "**MM $1035**" puis **$10** à la place de **$FF**.
- Ces commandes sont faites automatiquement si l'on a chargé la macro: **MENES.mcr.**

On peut alors charger le programme du 68HC11 en tapant "**LOADS nom-prog**".

Chaque octet se programme en 20 ms (10 ms pour d'abord effacer et 10ms pour programmer).

Pour gagner les 10 ms d'effacement à chaque octet il suffit d'effacer totalement l'EEPROM avant de commencer le LOADS. Pour cela il faut taper : "**EEPROM ERASE BULK $F800**".

Si la Macro MENES.mcr a été chargée il suffit de taper "**EFF**"

Le programme est en place dans l'EEPROM du 68HC11 on peut donc l'exécuter sous PCBUG en mode BOOT en tapant "**G $F800**". Si la macro MENES.mcr a été chargée il suffit de taper : "**GO**".

Le programme peut aussi être exécuté en mode SEUL. Pour cela passer le micro en mode SEUL (pin MOD B à "1") et faire un reset, le programme en EEPROM est exécuté.

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.

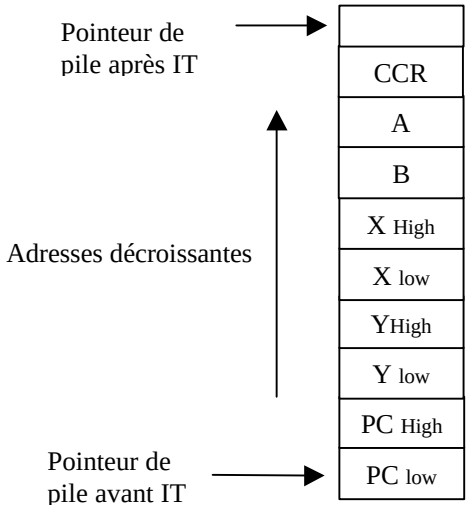
Exemple: **TAB** Transfert du contenu du registre A vers le registre B.

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**