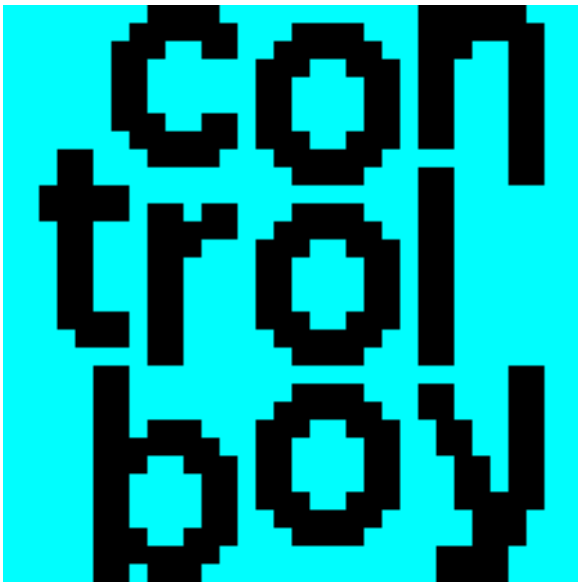
Outils de développement



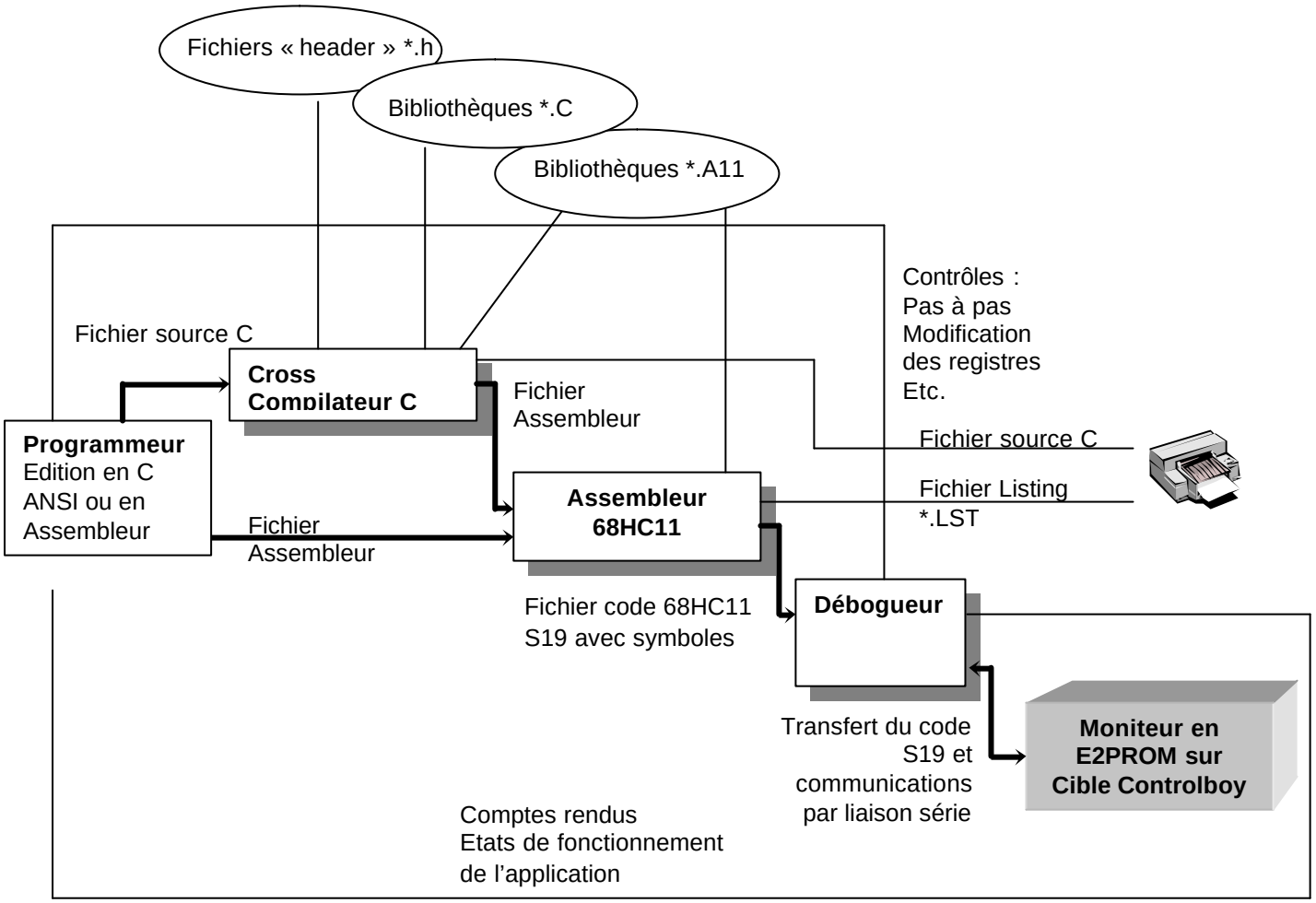
68HC11 assembleur

Cours et travaux pratiques

Christian Dupaty
Professeur d'Electronique
Lycée Fourcade
13120 Gardanne
V2.0

SOMMAIRE

1. Structure de l'outil de développement :

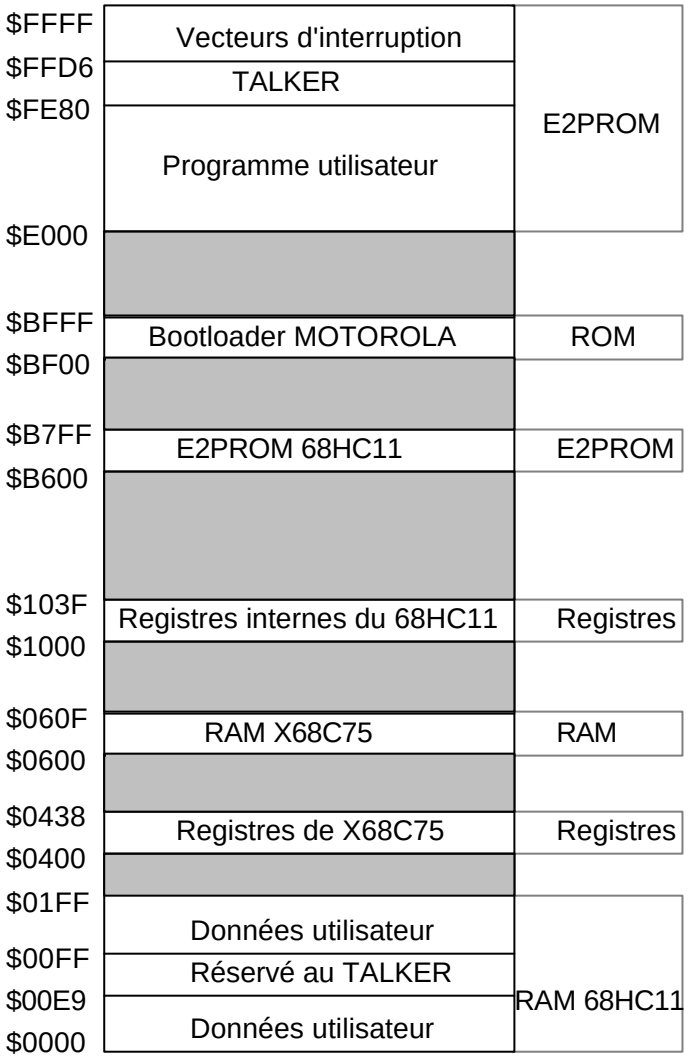


- Le programme peut être rédigé en assembleur 68HC11 ou en langage C.
- Controlboy permet également de développer une application en "prototypage rapide", ce procédé peu employé dans l'industrie n'est pas traité dans ce cours.
- Les échanges de fichiers sont automatiques.
- Controlboy fonctionne Windows 95/98, NT4 et 2000
- Controlboy dispose de la fonction débugage source en assembleur, en C, en BASIC

2. Caractéristiques des Controlboy

- **Controlboy 2 Kit** : Microcontroleur 68HC11E0, EEPROM X68C75, 8 entrées analogiques, 4 entrées digitales et 12 sorties digitales ou 12 entrées digitales et 4 sorties digitales, 2 touches, 1 compteur, 2 relais, 1 afficheur, 2 LEDs, RS232
- **Controlboy 3 Kit** : Microcontroleur 68HC11E0, EEPROM X68C75, 8 entrées analogiques, des entrées digitales et des sorties digitales, 1 touche, 1 compteur, 6 relais, 2 sorties Darlington 500mA, 1 afficheur, 2 LEDs, RS232. CNA 3 canaux 8 bits, Afficheur LCD 16 caractères, Clavier 12 touches, 8 entrées optocouplées.
- **Controlboy F1** : Micrcontroleur 68HC11F1 (12 ou 16 MHz), 32 KO EEPROM, 32 KO RAM, possibilité d'horloge temps réel. Afficheur LCD 16 caractères, Clavier 12 touches. Tous les ports sur connecteur DIL.

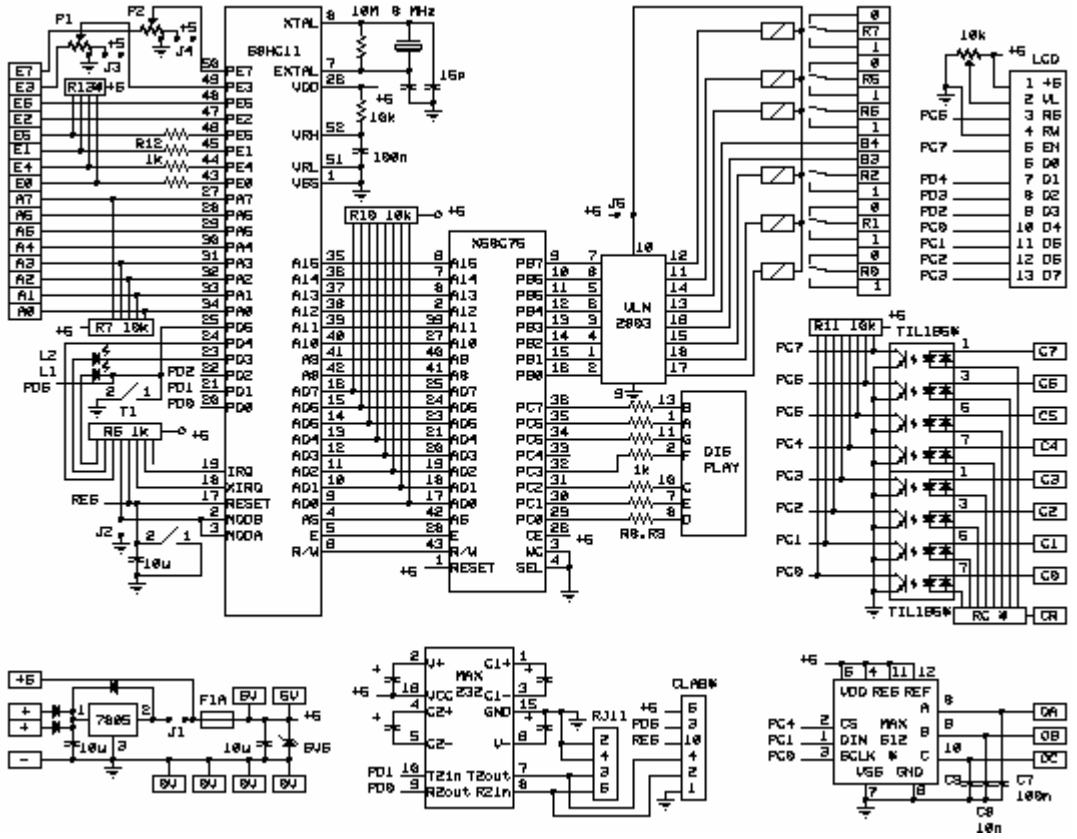
2.1. CBOY 2 et 3 Carte mémoire



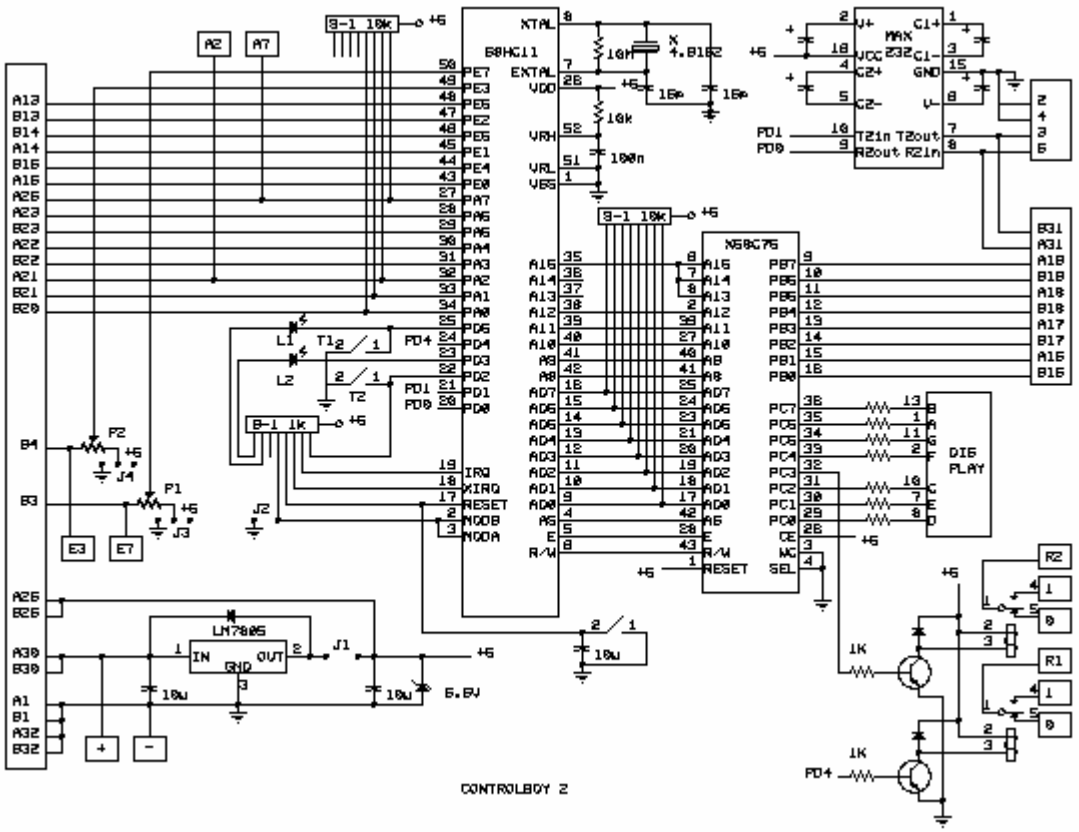
2.2. Cboy F1 Carte mémoire

Adresses	Type
FE00-FFFF	TALKER et vecteurs d'interruptions
8000-FDFF	EEPROM 28HC256 Programme utilisateur
2000-7FFF	RAM 62256 Données utilisateur
1800-180F	Ports externes sur connecteur X
1064-17FF	RIEN
1060-1063	Ports B,C,M,N
1000-105F	Registres du 68HC11F1
0400-0FFF	RIEN
0000-03FF	RAM interne du 68HC11F1

2.3. Schéma structurel Controlboy 3



2.4. Schéma structurel Controlboy 2



? La configuration Controlboy 2 et 3 permet de disposer en plus d'un port B **bidirectionnel**, le microcontroleur étant en mode EXPENDED, les ports B et C sont perdus mais reconstitués par le circuit X68C75 qui intègre 8KO d'EEPROM

Caractéristiques :

? CPU 68HC11, 8 KO E2PROM en mode EXPENDED dans le circuit X68C75 pour le programme
? 512 Octets d'E2PROM sur 68HC11 avec protection et 512 Octets de RAM

2.5. X68C75

? Le 68HC11 est placé dans le mode EXPANDED MODE et les ports B et C servent comme bus externe. Les deux ports du X68C75 remplacent les ports B et C internes. Le registre SCONF du X68C75 permet de configurer les nouveaux ports B et C.
? *Note : pour plus de commodité, les ports A et B du X68C75 sont renommés respectivement B et C*
? *Registre SCONF*

7	6	5	4	3	2	1	0
IRST	1	BWO	CWO	DIRB	DIRC	STPB	STPC

Interrupt Request Reset Mode
Les broches STRA et STRB ne sont pas connectés sur Controlboy, les interruptions ne sont donc pas utilisées IRST=0

Types de sorties :
0 : CMOS Push-Pull
1 : Drain ouvert

Direction :
0 : port en entrée
1 : port en sortie
(le sens n'est pas modifiable bit à bit)

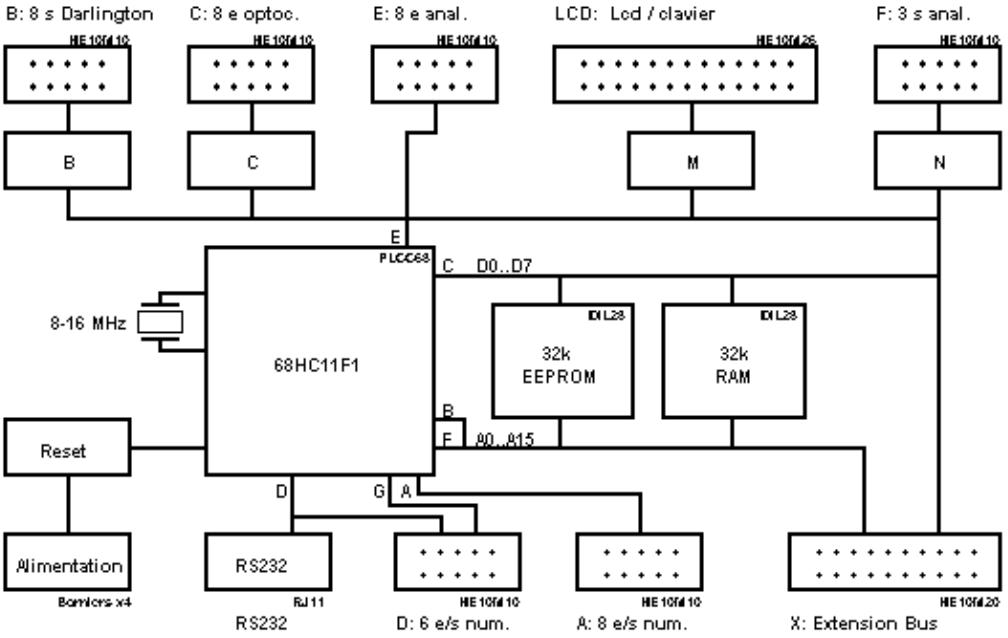
Strobe actif sur O ou 1
Inutilisé sur Controlboy, toujours à 0

Registres	Adresses
SCONF	$0420
PRBO, PORTB en SORTIE	$0410
PRBI, PORTB en ENTREE	$0430
PRCO, PORTC en SORTIE	$0408
PRCI, PORTC en ENTREE	$0428

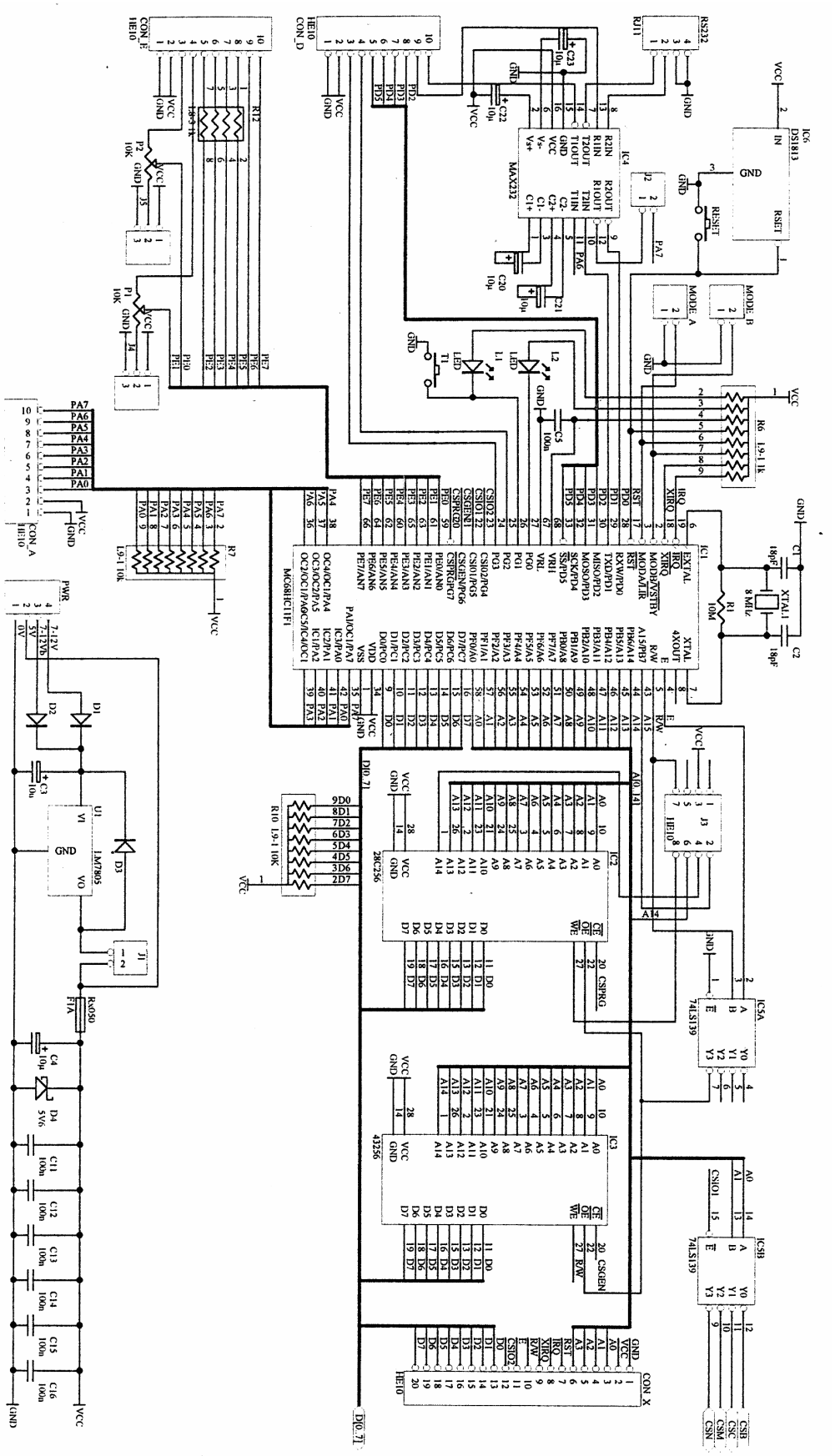
2.6. Controlboy F1

Schéma fonctionnel

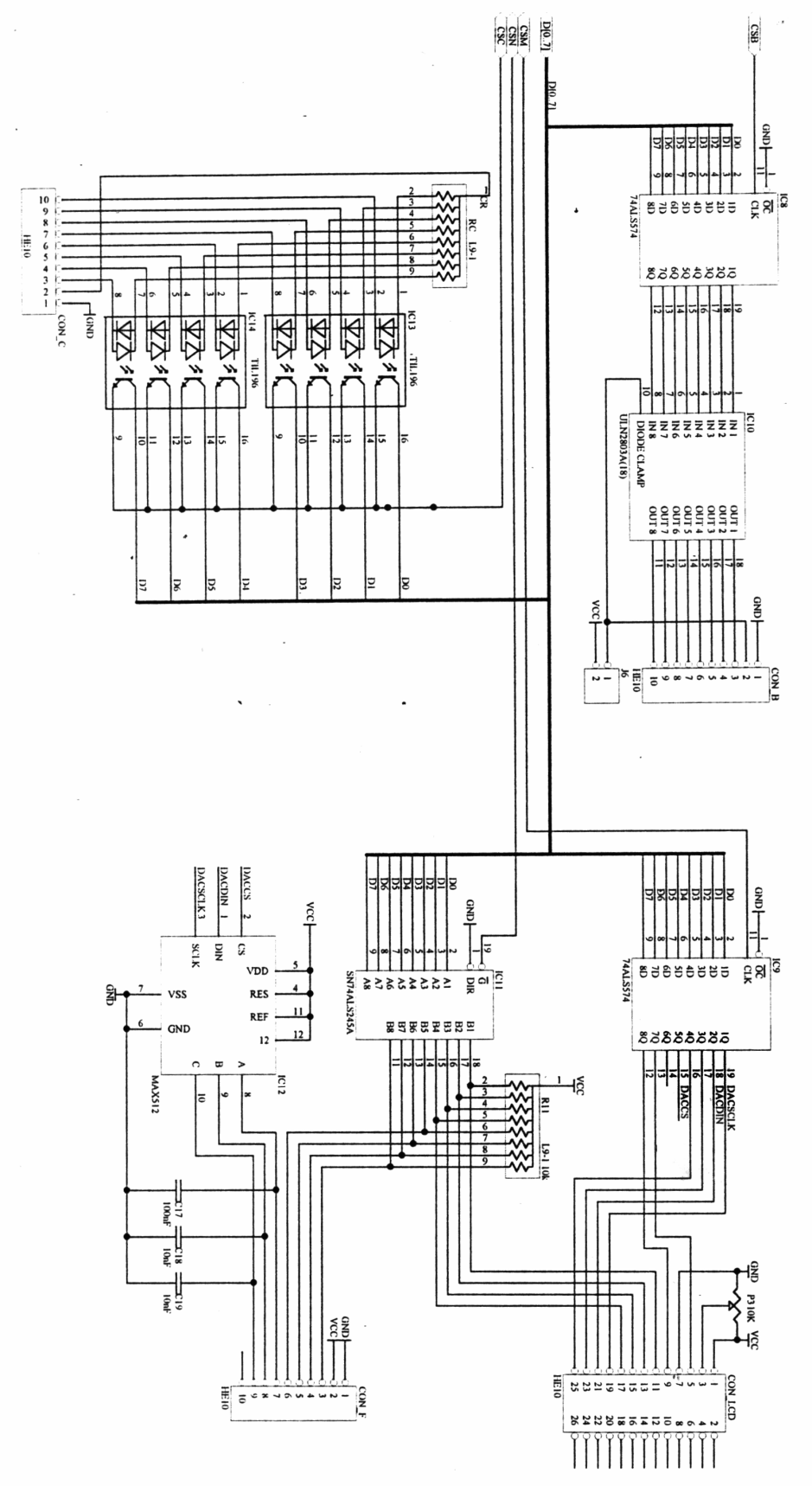
(consulter la doc CBOYF1)



2.6.1. Schéma structurel CBOYF1 1/2



2.6.2. Schéma structurel CBOYF1 2/2



Contraintes liées au moniteur(talker)

Le moniteur réside en EEPROM ($FE80 – vecteurs d'interruption). Ils permet les échanges entre le 68HC11 et le PC. Lors d'un appui sur RESET :
Si T1 est appuyé, le TALKER garde le contr^le en entre en mode débug. Dans le cas contraire le TALKER lance le programme à l'adresse $8000 pour CBF1 et $E000 pour CB2 et CB3. Pour activer le moniteur, il suffit donc d'enfoncer simultanément RESET et T1 et de relâcher RESET avant T1.
Il peut arriver que la programme utilisateur écrase le TALKER, dans ce cas clique « init talker » et suivre les instructions.

Le moniteur positionne le pointeur de pile système (S) . NE PAS LE MODIFIER.
La RAM comprise entre $E9 et $FF est réservée aux pseudos vecteurs du TALKER.

	Adresse	**Interruption**
*	FFD6	SCI Asynchronous Serial Interface (RS232)
	FFDA	Counter / Timer PA Input Edge
	FFDC	Counter / Timer PA Overflow
	FFF0	Real Time Interrupt
*	FFF6	SWI Software interrupt
*	FFF8	Illegal opcode
*	FFFE	Reset

- *Les interruptions marquées par* **** sont prises par le talker****.*
- Le talker utilise la ligne série pour la communication avec l'hôte.
- Néanmoins le programme peut aussi bien utiliser la ligne. Si une interruption de la ligne série arrive, l'unité centrale saute à l'adresse $00FA dans la mémoire vive. Un programme qui veut traiter toutes données de la ligne série écrit à l'adresse $00FB l'adresse de son sous-programme qui traite les interruptions. Par conséquent quand ce programme tourne, le talker est inutilisable. Autrement si le talker reçoit un caractère de la ligne série qu'il ne reconnaît pas comme commande du débogueur, il saute à l'adresse $00FD le caractère dans le registre A. Un programme qui écrit l'adresse d'un sous-programme à l'adresse $00FE peut donc traiter toutes les données qui ne sont pas dédiées au talker.

FFD6: 00FA **EEPROM: vecteur d'interruption SCI**
00FA jmp talkerInt RAM: saute au talker
00FE talkerint: lit caractère
si caractère >$06 *jmp 00FD sinon traité par le talker*
00FD jmp talkerend RAM: saute au talker
00FE takerend rti EEPROM talker

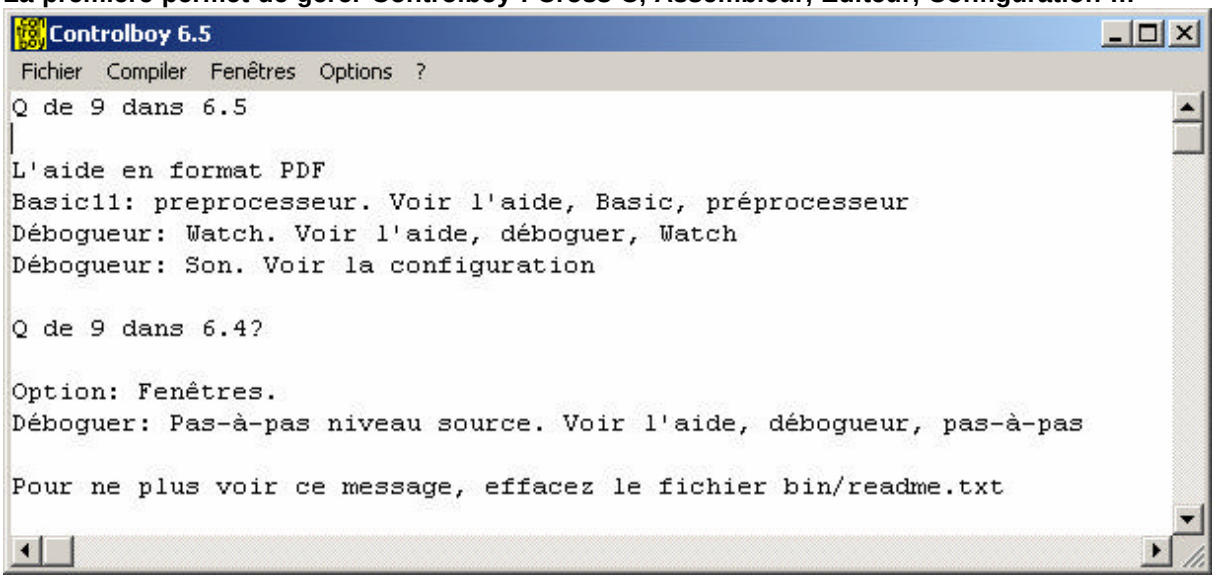
Voir TP sur SCI

3. TP n°1 : Prise en main de Controlboy et de l'ASSEMBLEUR

- ? Connecter Controlboy au port série COM2 du PC.
- ? Mettre le PC sous tension
- ? Alimenter Controlboy (8v < VCC < 10v)

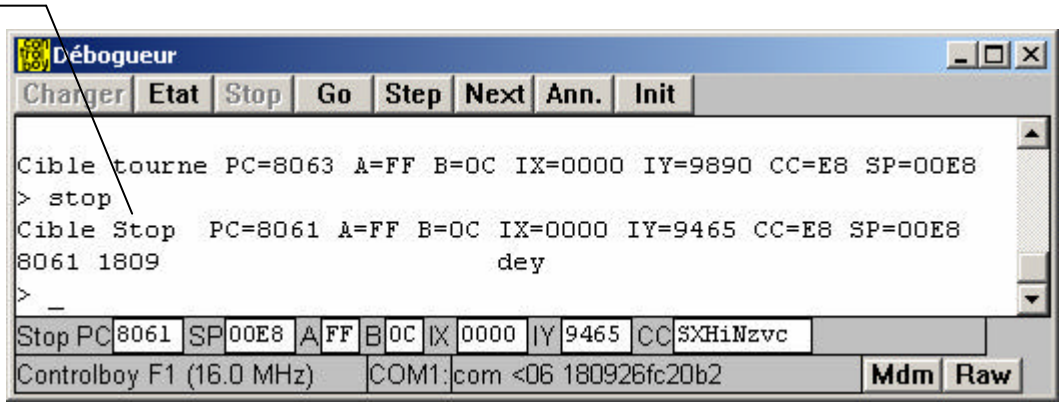
✍ sur l'icône , deux fenêtres apparaissent :

La première permet de gérer Controlboy : Cross C, Assembleur, Editeur, Configuration …



La seconde permet de controler la cible : téléchargement du fichier S19, exécution pas à pas , visualisation de la mémoire, etc.

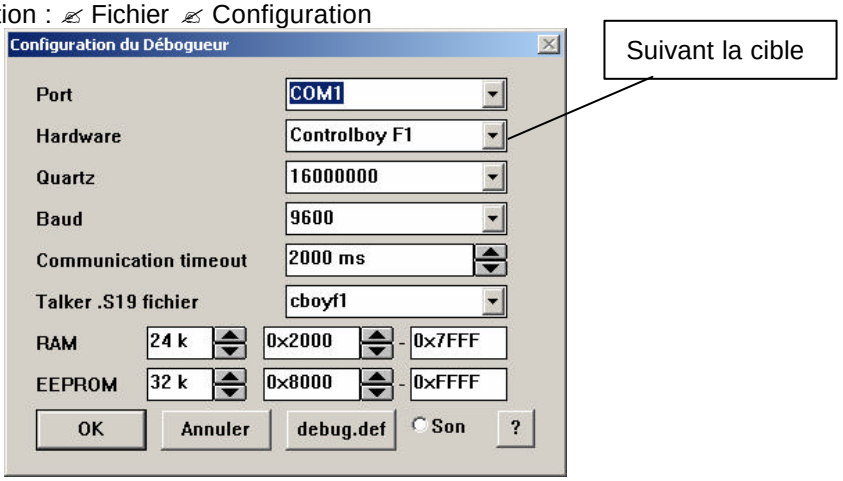
Cette ligne indique que les communications séries entre le PC et Controlboy fonctionnent



Cible Stop : le programme d'application n'est pas lancé, le moniteur gère Controlboy

PC : compteur de programme, **A, B, IX, IY, CC** : Registres internes, **SP** : pointeur de pile système

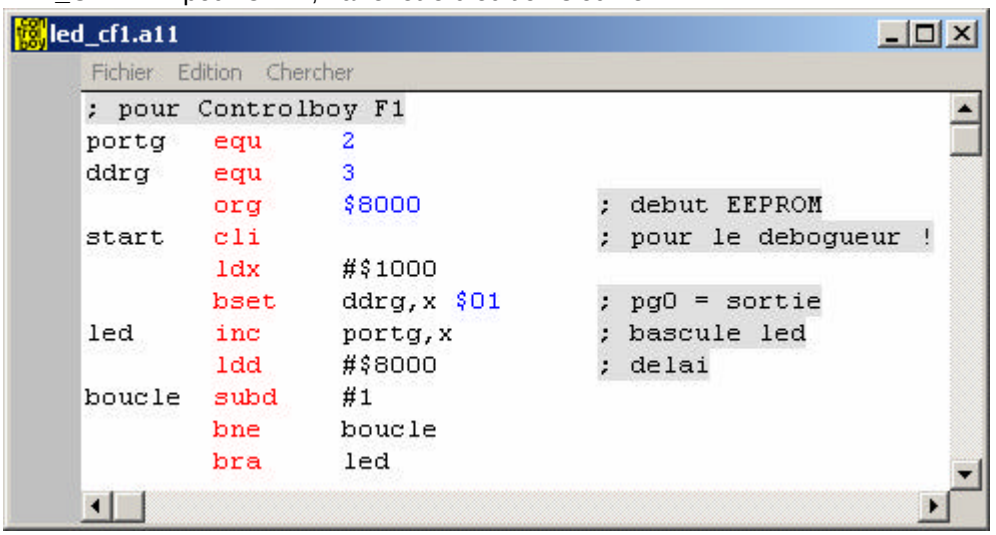
Vérification de la configuration : ✍ Fichier ✍ Configuration



Vérifier la configuration puis ✍ OK

? Chargement d'un exemple d'application

"Fichier" "Ouvrir", dans le répertoire \CBOY\exoasm ✍✍ **LED.A11** pour CB2 et 3, et ✍✍ **LED_CF1.A11** pour CBF1, La fenêtre d'édition s'ouvre

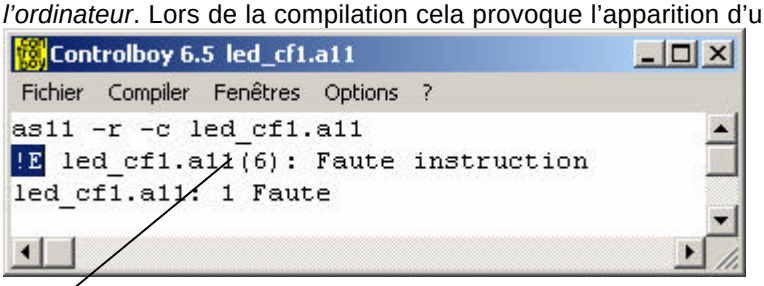


Après analyse de ce petit programme, ✍ **Option,** entrer l'option de compilation –c, les cycles machines apparaitront ainsi dans le fichier listing (*.LST).

Remarque: L'instruction CLI autorise les interruption et en particulier l'interruption SCI (port série asynchrone) qui est utilisée par le moniteur en E2PROM.

Compiler, le résultat doit donner 0 Errors, dans ce cas un fichier listing LED.LST et un fichier code LED.S19 ont été crée par l'assembleur.

Introduire une erreur dans le fichier LED.A11. Par exemple remplacer LDX par LDXX . *Enregistrer le fichier, le compilateur assemble les fichiers sur le disque dur et non dans la mémoire RAM de l'ordinateur*. Lors de la compilation cela provoque l'apparition d'un message d'erreur.



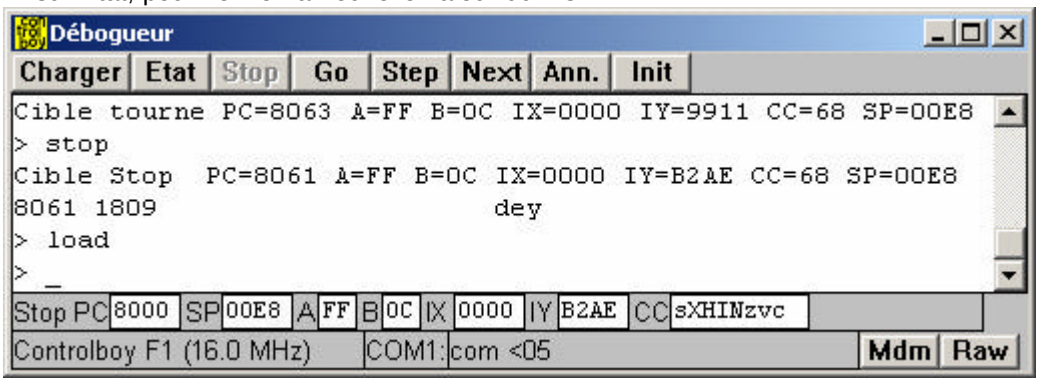
✍✍ sur la ligne d'erreur, l'éditeur se place automatiquement sur la ligne de l'erreur

Corriger l'erreur et recompiler le fichier LED.A11

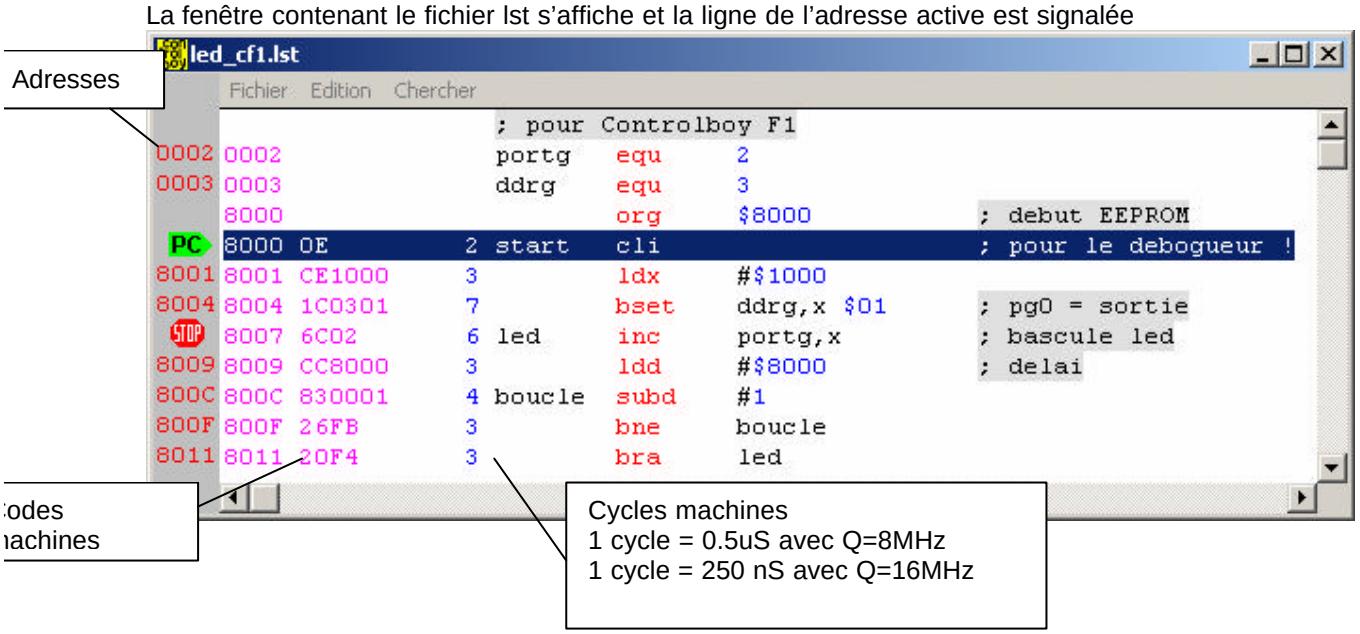
✍ sur la fenêtre du débogueur, puis ✍ sur Charger, le fichier LED.S19 est alors transféré dans l'E2PROM de Controlboy.

Pour modifier l'adresse du PC et la placer en début de programme tapez "r pc 8000" (pour registre PC=$8000) sur CBF1 et r pc E000 sur CB2 et CB3

✍ sur Etat, pour vérifier la nouvelle valeur du PC



La fenêtre contenant le fichier lst s'affiche et la ligne de l'adresse active est signalée



✍ sur Step pour tester le programme en pas à pas
Tapez à nouveau "r pc 8000" puis ✍ Go, le programme LED.A11 s'execute (la LED clignote).

Placer un point d'arrêt en $8007 en cliquant dans la colonne des adresses rouges. Faire « go » le programme s'arrête en $8007, « go » le relance. Un clique sur $8007 en rouge supprime le point d'arrêt.


Remarque : Une aide très détaillée peut être obtenue en ✍ sur ?

3.1. *Utilisation de bibliothèques*

```
                ; Programme de démonstration de l'utilisation     d'une bibliothèque (CD 10/00)
                ; Sur      CBOYF1 (LED sur  PRG)
                ; La définition des secteurs est obligatoire lors de l'utilisation de        bibliothèques
                ; Pour la démo une variable "var" est  incrémentée à chaque boucle et recopiée          sur PG0
                ; ce qui a pour effet              de faire clignoter le LED verte
                ; Le S/P de tempo se trouve dans une bibliothèque, la       durée de la tempo en ms       est
                ; rangée dans       une variable 16      bits "nbms"
                ;
1002            portg     equ     $1002           ; adresses absolues du portg et de ddr
1003            ddrg      equ     $1003
0000            q16meg    equ     *               ; définition du quartz pour la bibliothèque
                ;
0000                      sect    data            ;initialisation du secteur data en $2000 (CBF1)
2000                      org     $2000           ;début RAM utilisateur
0000                      sect    text            ;initialisation du secteur text en $8000 (CBF1)
8000                      org     $8000           ; debut EEPROM
                ;
8000 0E         2         cli                     ; pour activer les com avec le debogueur !
8001 8601       2         ldaa    #1
8003 B71003     4         staa    ddrg            ; pg0     = sortie
8006 B72000     4         staa    var             ; initialise var
8009 CC01F4     3         ldd     #500            ; durée de la tempo en ms (la          période    sera de    1S)
800C FD2001     5         std     nbms
                ;
800F 7C2000     6 led     inc     var             ; vara = var+1 -> portg
8012 B62000     4         ldaa    var
8015 B71002     4         staa    portg           ; bascule led
8018 8D02       6         bsr     tmp1ms  ; appelle S/P       de tempo dans la bibliothèque
801A 20F3       3         bra     led
                ;
2000                      sect    data            ; on repasse en assemblage sur la RAM
2000            var       rmb     1               ; var      occupe un octet      en RAM
                ;
                #include tmp1ms.a11                ; bibliothèque de temporisation
```

Le secteur data (RAM) est initialisé en $2000

Le secteur text (EEPROM) est initialisé en $8000

nbms est définie dans la bibliothèque

Directive pour inclure une bibliothèque

La librairie tmp1ms.a11 qui est liée au programme

```
                    ; Librairie 68HC11
                    ; Temporisation en mS        : TMP1MS
                    ; durée= nbms*(10+7*D)+41)*t . t = période base de temps E = 4/q
                    ;
                    ; exemple pour choisir la fréquence du quartz dans le programme appelant
                    ;q16meg equ       *
2001                        sect      data      ; nombre de ms à décompter
2001                nbms    rmb       2
                    ;
801C                        sect      text      ;la bibliothèque est placée en EEPROM
801C 36             3 tmp1ms psha               ; sauve les registres
801D 37             3       pshb
801E 183C           5       pshy
8020 18FE2001       6       ldy       nbms      ; compteur de ms
                    ;
8024                tmp2    equ       *

                    #ifdef q16meg   ; q=16 MHZ CBF1
8024 CC0226         3               ldd       #550
                    #endif
                    #ifdef q12meg   ; q=12 MHZ CBF1
                                    ldd       #409
                    #endif
                    #ifdef q8meg        ; q=8 MHZ CB3
                                    ldd       #265
                    #endif
                    #ifdef q4meg        ; c'est donc un quartz 4.194304 MHZ CB2 ou CB1
                                    ldd       #129
                    #endif

8027 830001         4 tmp1  subd      #1
802A 26FB           3       bne       tmp1
802C 1809           4       dey                 ;Decompte le nombre de mS demandé
802E 26F4           3       bne       tmp2
                    ;
8030 1838           6       puly
8032 33             4       pulb
8033 32             4       pula
8034 39             5       rts
```

Variable déclarée dans la bibliothèque

Les directives #ifdef et #endif permettent de tenir compte de la fréquence du quartz suivant les modèles de CB.

Tester le programme tstlib.a11 accompagné de sa bibliothèque tmp1ms.a11

3. Résumé des options de l'assembleur AS11 Controlboy

? **L'assembleur traduit le fichier source et génère plusieurs fichiers.**

<nom du fichier>.a11 fichier source assembleur
<nom du fichier>.bak fichier sauvegarde du fichier source
<nom du fichier>.s19 fichier objet. Motorola S-records
<nom du fichier>.lst fichier listage

? **Options :**

-r Ne pas changer le fichier source.
-n Fichier objet sans informations pour le débogueur.
-b Ne pas transformer les Branchs en Jumps.
-j Traiter les Jumps comme les Branchs.
-g Les symboles sont locaux par défaut.
-l Numéros de ligne dans le fichier listage.
-c Comptage de cycles du 68HC11 dans le fichier listage.
-2..-9 Passes (défaut: 3)

? **Directives d'assemblage.**

option	[r|n|b|j|g|l|c|2..9]*	mettre option
org|.org	<valeur>	changer l'adresse d'assemblage
fcb|.byte	<valeur>[,<valeur>]*	stocker des octets en mémoire
fcb|.ascii	'<chaîne ASCII>'	stocker des octets en mémoire
fdb|.word	<valeur>[,<valeur>]*	stocker des mots de 16 bits
rmb|.blkb	<valeur>	réserver octets en mémoire
<nom> equ|=	<valeur>	déclarer un symbole
sect|.area	text|data|bss|none	changement de la section
.globl	<nom>[,<nom>]*	déclarer symbole global
end	ignoré	

La commande sect permet de gérer plusieurs adresses d'assemblage dans le fichier, par exemple pour la RAM (sect data) et pour l'EEPROM (sect text).

Un symbole a jusqu'à 31 caractères. Les caractères suivants sont reconnus. (a..z A..Z 0..9 _ . $)
Un nom ne peut pas commencer avec un chiffre ni avec $.
Une expression arithmétique <expr> peut être composée par des symboles, des constantes et le *' ou le .', qui représentent l'adresse d'assemblage actuel. On peut les combiner par des opérations + - * / ( ).
Le premier caractère d'une constante décide son interprétation :

' caractère ASCII
$ numéro hexadécimal
0x numéro hexadécimal
% numéro binaire

Sinon c'est une valeur décimale.
Une ligne qui commence avec * ou avec ; est un commentaire.

? **Les commandes de préprocesseur commencent tout à gauche dans une ligne avec #.**

#if <ifexpr>	assembler lignes suivantes si <ifexpr> est vrai
#ifdef <nom>	assembler lignes suivantes si <nom> est défini
#ifndef <nom>	assembler lignes suivantes si <nom> n'est pas défini
#else	inverser #if #ifdef #ifndef
#endif	fin de #if #ifdef #ifndef
#assert <ifexpr>	signaler une erreur si <ifexpr> est faux
#include <nom de fichier>	inclure <nom de fichier>

La commande #include permet d'inclure un autre fichier dans la source. L'assembleur compile ce fichier et ensuite retourne au premier fichier et continue après la commande. Le fichier inclus peut inclure d'autres fichiers.
<ifexpr> est une expression arithmétique <expr> ou la comparaison des deux expressions.
<ifexpr> = <expr> [==|!=|>|>=|<|<= <expr>]

4. Instructions du débogueur

BRDEL **Effacement de point d'arrêt**

BRD[EL] Efface tous les points d'arrêt.
BRD[EL] <Adresse> Efface le point d'arrêt à l'adresse

BREAK **Points d'arrêt**

BR[EAK] affiche les points d'arrêt en vigueur
BR[EAK] <Adresse> met un point d'arrêt à l'adresse

D **Affichage des variables**

D <variable> [<variable>]*
Lit la mémoire à l'adresse et affiche les variables comme elles sont définies.

DIS **Désassemblage**

DIS
DIS <Adresse> Lit la mémoire à l'adresse et affiche les instructions
DIS <Adresse> <Longueur>

GO **Démarrage du programme**

G[O] Démarre le programme à l'adresse en cours
G[O] <Adresse>

INITTALKER **Initialisation du talker**

INIT
INITTALKER

LOAD **Chargement du programme**

L[OAD] [<nom de fichier>] Charge le programme dans l'EEPROM. le SP est mis à $00E8.
La commande charge sur les adresses de l'EEPROM jusqu' à $FE7F et $FFD6 à $FFFF. Il refuse le chargement sur des autres adresses.

LOADS **Chargement de la table de symboles**

LOADS [<nom de fichier>]

LOG **Journal**

LOG
LOG ON
LOG <Nom de fichier>
LOG ON <Nom de fichier>
LOG OFF
Active ou désactive le journal. Sans nom, le journal est écrit dans le fichier CLAB.LOG.

MEM **Affichage de la mémoire**

M[EM]
M[EM] <Adresse> Lit la mémoire à l'adresse
M[EM] <Adresse> <Longueur>

MFILL **Remplissage de la mémoire**

MF[ILL] <Adresse> <Longueur> [=] <Octet>

MSET **Ecriture mémoire**

MS[ET] <Adresse> [=] <Octet> [<Octet>]* Ecrit des octets ou des mots
MS[ET] -W <Adresse> [=] <Mot> [<Mot>]* W pour un pointeur

NEXT **Pas à Pas**

N[EXT] Exécute une instruction du programme

REG **Registres**

R[EG]
R[EG] A | B | CC | PC | SP | IX | IY [=] <Valeur>

STEP **Pas à Pas**

S[TEP] Exécute une instruction du programme

STOP **Arrêt du programme**

STOP Arrête le programme en exécution.

UPLOAD **Enregistrer le programme dans un fichier**

UPLOAD <Adresse> <Longueur> [<nom de fichier>]
Lit la mémoire de la cible et enregistre les données dans un fichier en format S19

VER **Vérification du programme**

V[ER] [<nom de fichier>] Compare le programme dans l'EEPROM avec celui dans le fichier

W **Watch d'une variable**

W toto affiche la variable toto, la valeur est réactualisé à chaque step ou stop

5. Exemple des directives de l'assembleur AS11

```
                    ; Démonstration      des directives de l'assembleur AS11
                    ; C.D       10/1997
                    ; CLIGNOTEMENT DU PORTC          sur CBOY3
                    ;
                    ; 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
E000                ROM     EQU     $E000
1000                REGBASE         EQU     $1000    ;Base    des regis tres
                    ;
                    ; ZONE RAM DU 68HC11Ex
0000                        sect    data
0040                        ORG     RAM
0040                MEM     RMB     10
004A                MEM2    EQU     *
                    ;
                    ;DEBUT DE LA ZONE EPROM
0000                        sect    text
E000                        ORG     ROM      ;DEBUT PROG EN DEBUT EPROM
                    ;
                    ; INITIALISATION DU MICRO-CONTROLEUR ET           DES PERIPHERIQUES
E000 0E             RESET   CLI             ; le moniteur       reste actif
E001 CE1000                 LDX     #REGBASE
E004 8644                   LDAA    #%01000100
E006 B70420                 STAA    SCONF           ; Port C en sortie
E009 8655                   LDAA    #%01010101      ;
E00B B70408                 STAA    PORTC
E00E 4F                     CLRA
E00F 9740                   STAA    MEM
                    ;
                    ; PROGRAMME PRINCIPAL
E011                PP      EQU     *
E011 18CE03E8               LDY     #1000
E015 8D33                   BSR     tmp1ms  ;ATTENDRE 1S
E017 730408                 COM     PORTC   ;LA LED CHANGE D'ETAT
E01A 7A0040                 DEC     MEM
E01D 20F2                   BRA     PP
                    ;
                    ; Démonstration      des autres directives
E01F 860F                   LDAA    #10+5           ;Addition
E021 8605                   LDAA    #12-7           ;Soustraction
E023 86FB                   LDAA    #7-12           ;Soustraction
E025 860F                   LDAA    #5*3            ;Multiplication
E027 8609                   LDAA    #27/3           ;Division entière
E029 8606                   LDAA    #27/4           ;Division
E02B 4365636909             FCB     'Ceci    est un texte ASCII'
E042 0C1241                 FCB     12,$12,'A'      ;Déclarations       de caractères
E045 000C001241             FDB     12,$12,'A'      ;Déclarations       de mot (2 octets)
                    ;
                    ; 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
                    ;
E05F 3B             ADRTI   RTI
                    ;
                    ;FIN DE   ROUTINES
                    ;VECTEURS D'INTERRUPTIONS GENERA LES (16          OCTETS EPROM)
                    ;
FFF0                        ORG     $FFF0                   ;
FFF0 E05F           TIMVECT FDB     ADRTI    ;INT PERIODIQUE
FFF2 E05F           IRQVECT         FDB     ADRTI    ;INT BROCHE IRQ
FFF4 E05F           XRQVECT         FDB     ADRTI    ;INT BROCHE XRQ
                            END
```

6. Première série d'exercices

- Les exercices suivants sont à tester sur Controlboy
- L'extension des fichiers assembleurs 68HC11 doit être A11, ex : COMPTEUR.A11
- Ces fichiers doivent être rangés obligatoirement dans le répertoire TRAVAIL
- Le vecteur **RESET n'a pas à être déclaré**
- La pile **S ne doit pas être modifiée** (elle est initialisée par le moniteur)
- Toutes les valeurs numériques doivent être déclarées en équivalences
- Les commentaires doivent êtres suffisants explicites et judicieux.
- Les programmes peuvent être systématiquement bouclés de manière à permettre une modification dynamique des opérandes.

Exercice 1 :

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

Exercice 2 :

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

Exercice 3 :

Modifier le programme led.a11 (led_cbf1.a11), de manière à obtenir une fréquence de clignotement de 1KHz.

Exercice 4 :

Modifier le programme led.a11 (led_cbf1.a11) de manière à placer la temporisation dans un sous programme

Exercice 5 :

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

Exercice 6 :

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

Exercice 7 :

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 8 :

Placer le SP hexasc précédent dans une bibliothèque et créer un petit programme de test, convertissant un octet en $40 en deux codes ascii placé en $50 et $51

Exercice 9 :

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)

Exercice 10 :

Réaliser un compteur BCD (0 à 99 dans une mémoire mem)

7. Deuxième série d'exercices

Ces exercices sont à tester à l'aide de l'outils Controlboy et de l'instrumentation adaptée (oscilloscope / GBF)

9.1. Ex n°1 : Gestion des périphériques d'E/S parallèles sans interruption

A partir de : En modifiant LED.A11, Réaliser un programme réalisant la fonction bistable, à chaque appui sur la touche T1, la LED L2 change d'état.

9.2. EX n°2 : Utilisation de l'interruption temps réel

A partir de : EXRTI.A11 . Réaliser un programme de comptage des minutes et secondes dans deux mémoires « min » et « sec »

9.3. EX n°3 : Production de signaux

Production de signaux périodiques :

A partir de : Programme de production de signaux carrés. EXCARRES.A11. Analyser EXCARRES.A11, vérifier son fonctionnement à l'aide d'un oscilloscope.
Produire sur port PA5 un signal rectangulaire de fréquence 100 Hz et de rapport cyclique ¼. La gestion sera faite en interruption (le programme principal ne fait rien)

Modulation de largeur d'impulsion

Ce type de modulation est utilisé pour la commande des hacheurs et donc dans les asservissements.
A partir de : Programme de gestion PWM : EXPWM.A11 , *une variable contient le rapport cyclique en % et une autre la période.*
Analyser PWM.A11. Produire des signaux de rapport cyclique , ¼ , ½ , ¾ et de fréquence 100 Hz et 1000 Hz en introduisant en RAM les paramètres adéquats.
Réaliser un programme produisant un double rampe PWM de fréquence 1Khz en utilisant l'interruption RTI.
Toutes les 100 mS, le rapport cyclique varie de 10%, la variation s'inverse lorsque le rapport cyclique égale 0% ou 100%

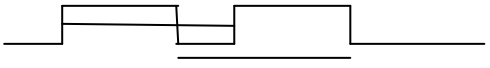
Diviseur de fréquence / compteur d'événements

Ces fonctions font appel à l'accumulateur d'impulsions du 68HC11
A partir de : EXACCU.A11. Ce programme incrémente un compteur toutes les 10 impulsions reçues.
Analyser et tester EXACCU.A11. Visualiser l'évolution du compteur en fonction des impulsions reçus (à basse fréquence de manière à pouvoir les compter)
Modifier le programme afin de produire un signal dont la fréquence est celle du signal d'entrée divisée par un entier compris entre 1 et 255. Le programme générera une interruption à chaque débordement de l'accumulateur d'impulsions

9.4. EX n°4 Mesure de temps

Mesure de périodes

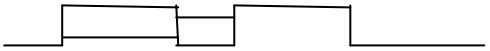
Il s'agit de mesurer la durée entre deux événements identiques (deux fronts montants ou deux fronts descendants)



A partir de : EXPERIOD.A11
Analyser le programme et le tester. calculer quel doit être le résultat de la mesure pour un signal de 100 Hz, 1khz, 10khz et vérifier expérimentalement.
Quelle est la durée la plus longue mesurable ?
Modifier le programme afin de le faire fonctionner en interruption. La mémoire mesure sera actualisée à chaque front montant du signal en entrée.

Mesure de durées

Il s'agit maintenant de mesurer la durée entre deux événements différents.



A partir de : EXDUREE.A11
Analyser le programme et le tester: calculer quel doit être le résultat de la mesure pour une durée à l'état haut de 1mS, 10mS, 100mS et vérifier expérimentalement
Modifier le programme afin de le faire fonctionner en interruption et en mesure de durées à l'état bas. La mémoire mesure sera actualisée après chaque front montant

9.5. EX n°5 : Utilisation d'une bibliothèque externe

A partir de : Documentation de l'afficheur LTN211 et bibliothèque HC11 stdio.a11 ou stdiof1.a11
Essayer et analyser le programme tststdio.a11
Réaliser une horloge affichant heures, minutes, secondes

9.6. Ex n°6 : voltmètre

Réaliser un voltmètre affichant sur l'afficheur CL la tension sur PE0 en mV

9.7. Ex n°7 : période mètre

Réaliser un période mètre affichant sur l'afficheur CL la période sur PA0 en nS

9.8. EX n°8 : Port de communication Asynchrone : SCI

A partir de it_sci.a11
Analyser ce programme
Réaliser un programme retournant le caractère ASCII reçu sur le port SCI (fonction ECHO)
Communications 9800bauds 8bits sans parité 1stop
Lors du test, utiliser la fonction « MODE BRUT ou RAW MODE » de ControlBoy
Utiliser ensuite un autre terminal, l'hyper terminal windows par exemple
Réaliser un compteur de secondes avec affichage sur SCI.

9.9. EX n°10 : sorties sur SCI

A partir du fichier « le code ascii.htm »
Réaliser un voltmètre 8 entrées, les tensions seront actualisées toutes les 500 ms et affichées en mV dans la fenêtre RAW comme suit
PE7 = 0125
PE6 = 5000
PE5 = 2561
…

9.10. EX n°11 : Gestion d'un C.A.N série (SPI) Texas Instrument TLC549

A l'aide de la documentation du TLC1549. Câbler un TLC549 sur le port SPI
A partir de EXTLC549.A11
Compléter et tester le programme. Celui-ci effectue une mesure de la tension

9.11. Ex n°12 : Gestion d'un C.N.A série

A partir de la documentation du MAX512 (✍ sur ?) réaliser un programme générant une fonction triangulaire sur la sortie A du C.N.A

Structure de l'outil de développement

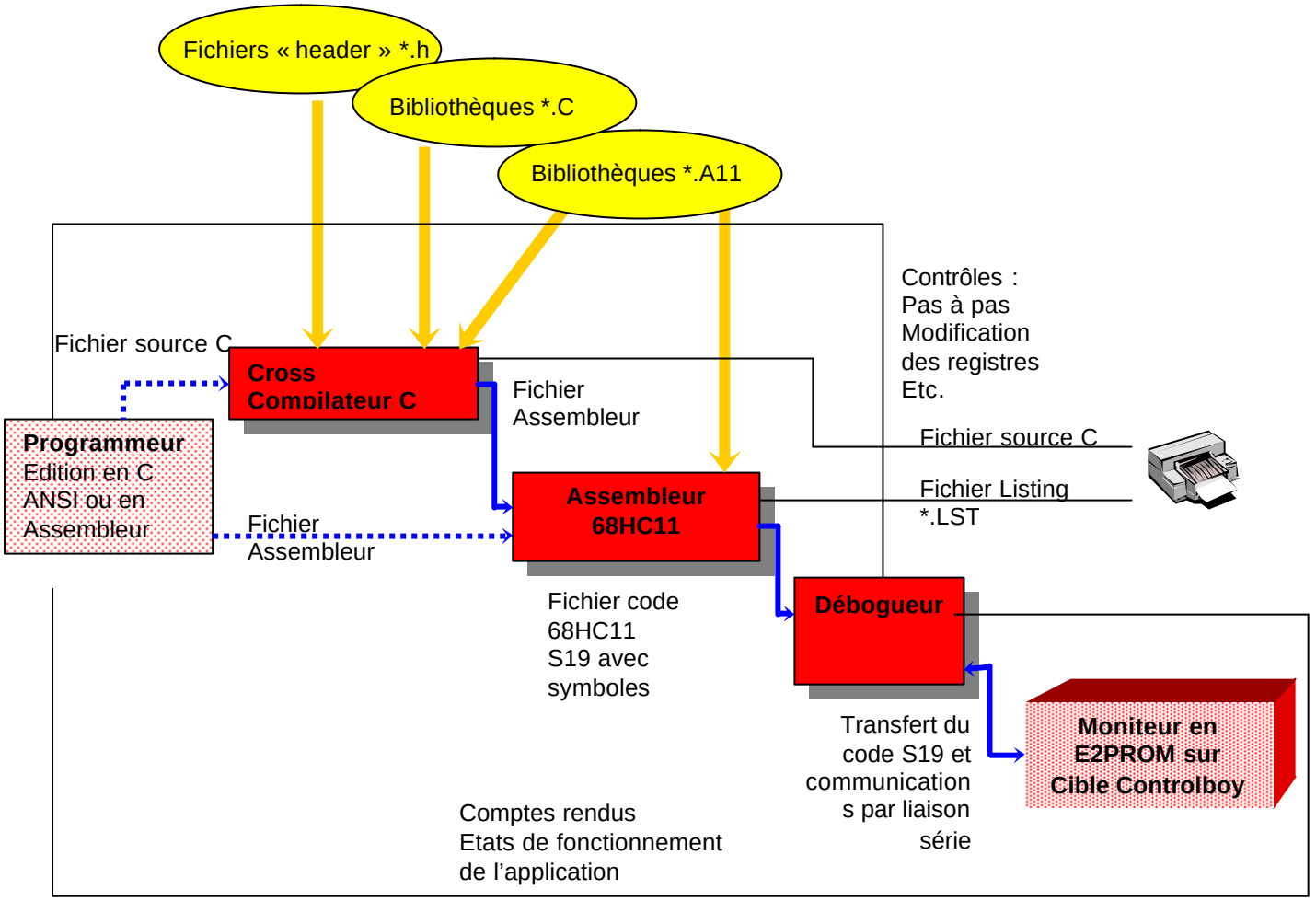
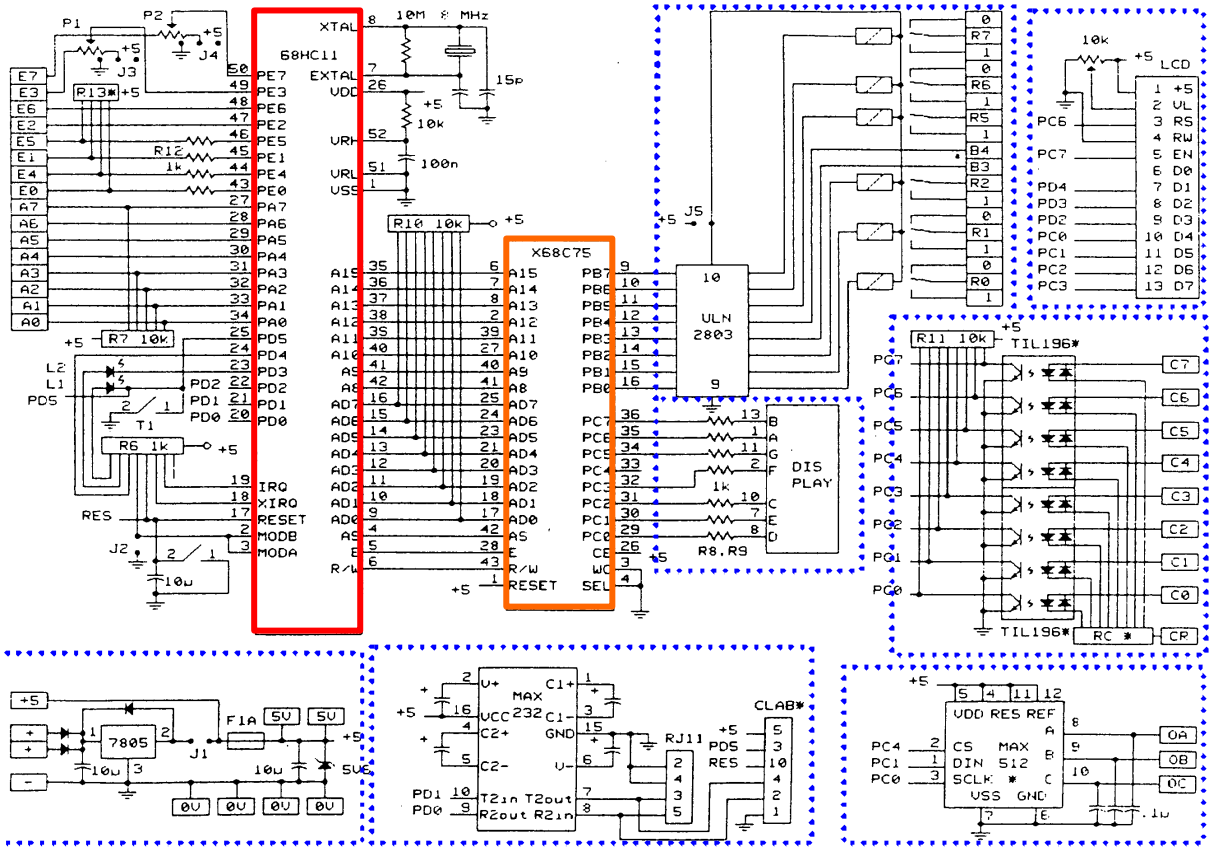
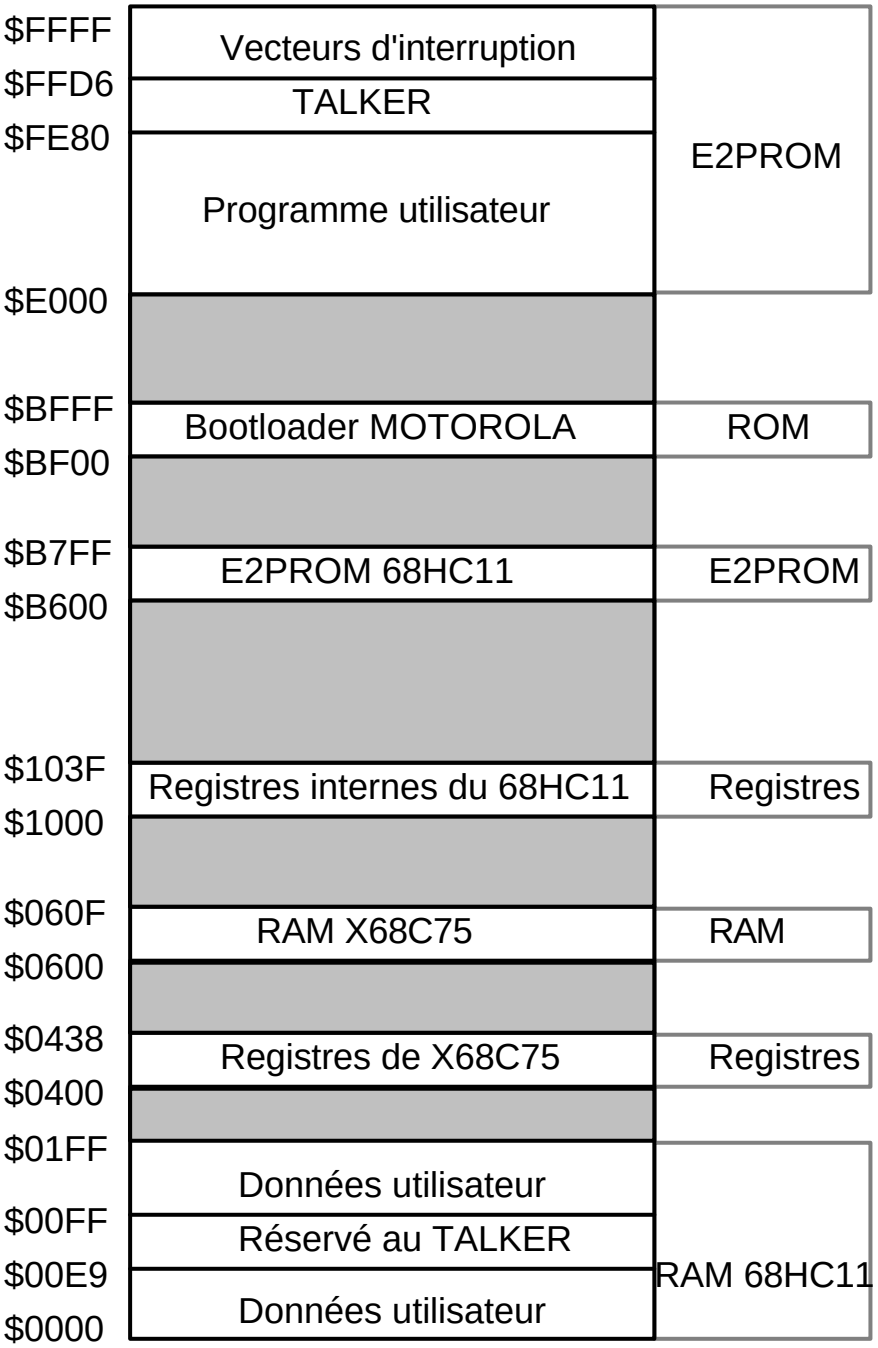


Schéma structurel Controlboy 3

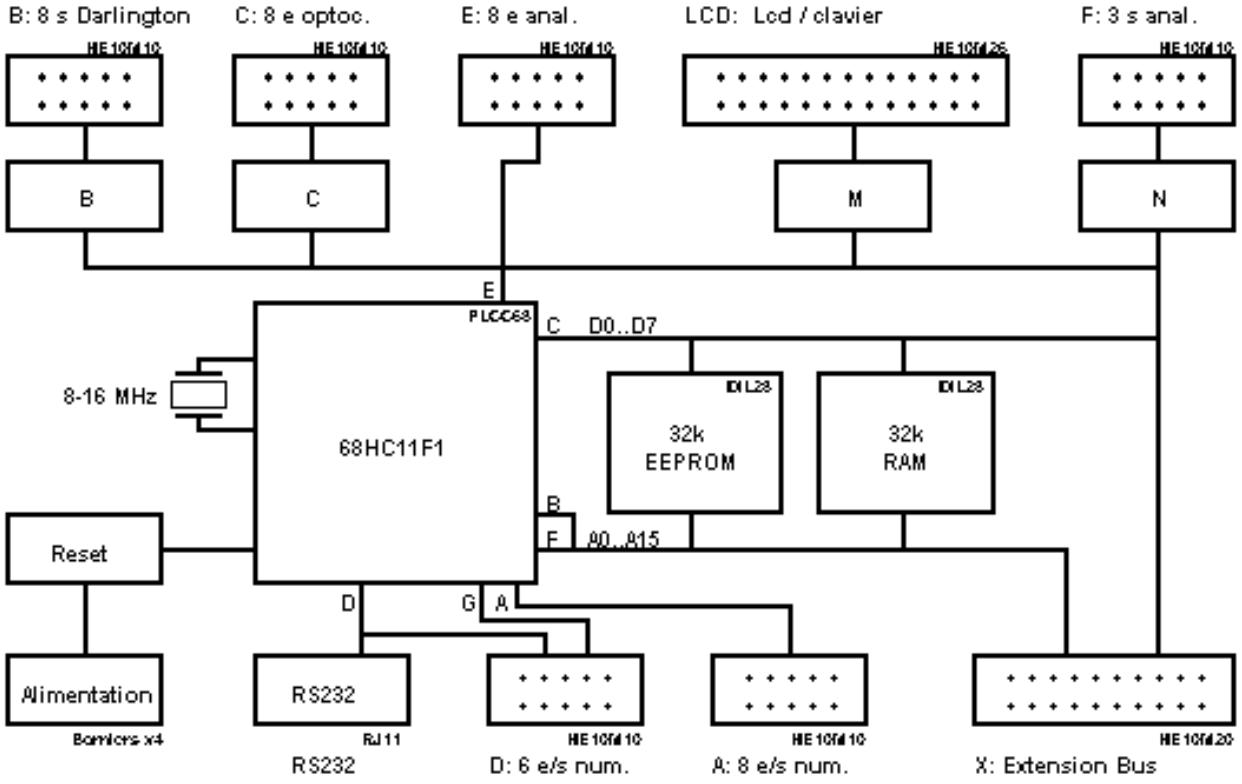


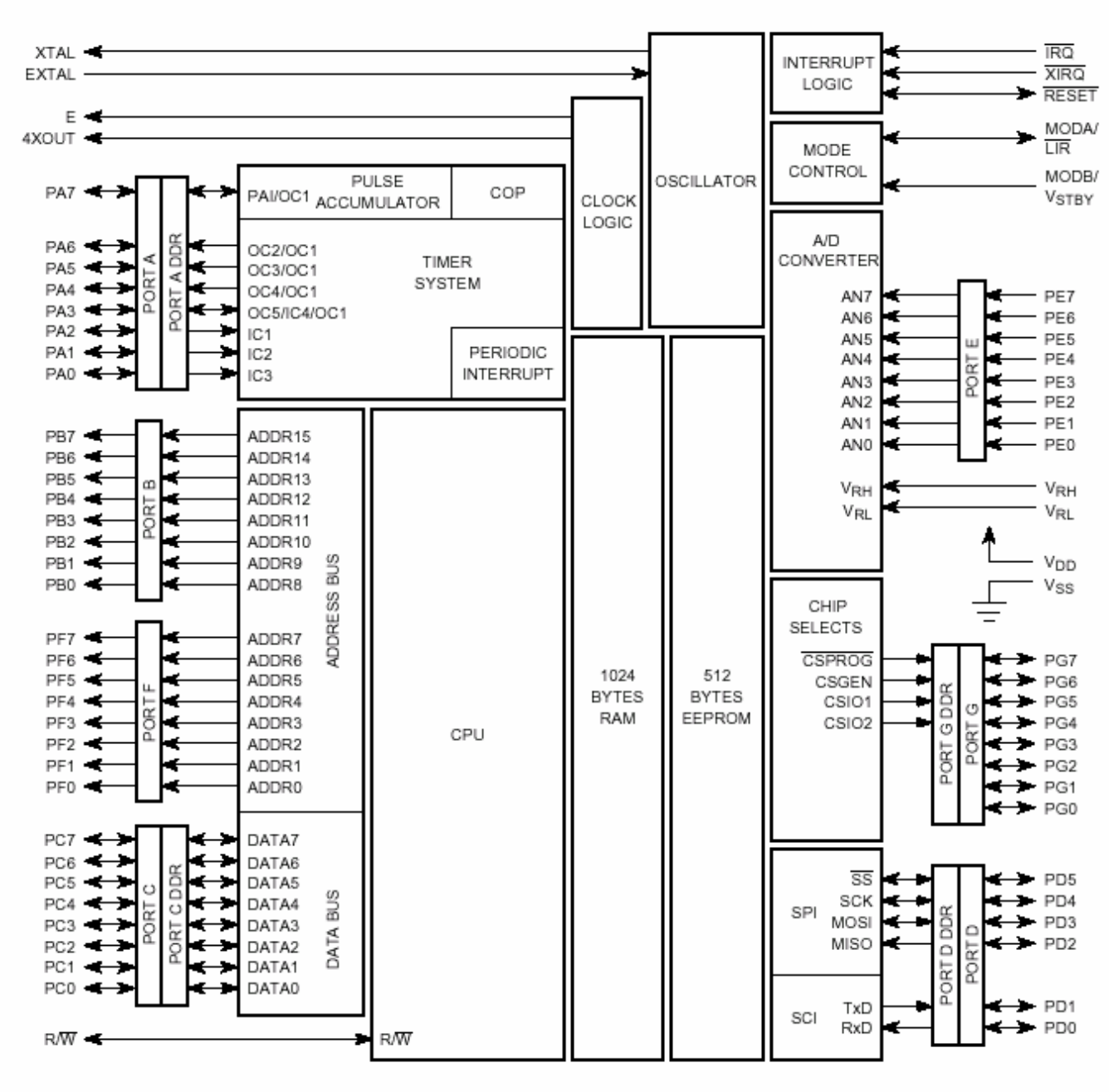
Plan mémoire Controlboy 2 et 3



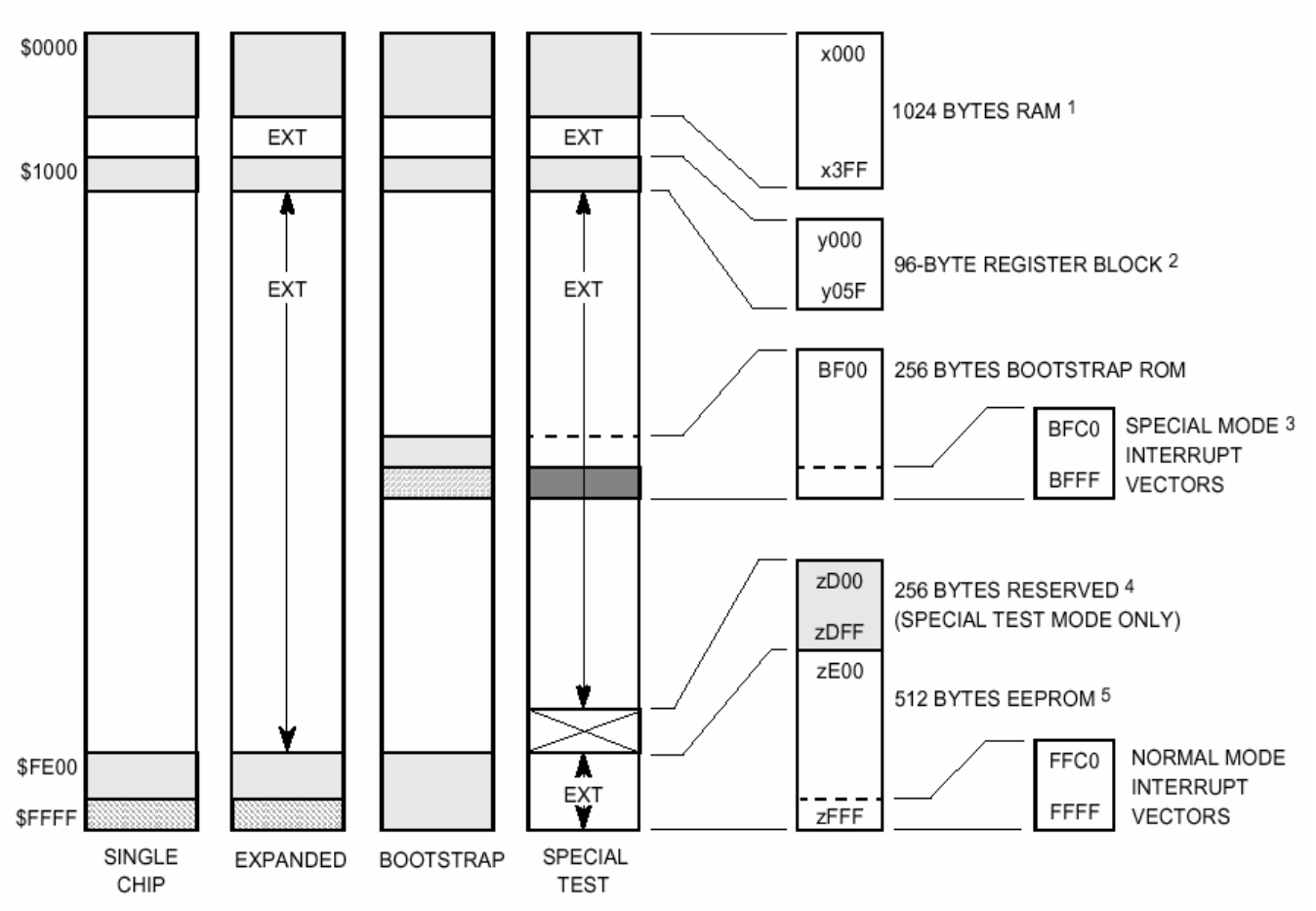
Cboy F1

Adresses	Type
FE00-FFFF	TALKER et vecteurs d'interruptions
8000-FDFF	EEPROM 28HC256 Programme utilisateur
2000-7FFF	RAM 62256 Données utilisateur
1800-180F	Ports externes sur connecteur X
1064-17FF	RIEN
1060-1063	Ports B,C,M,N
1000-105F	Registres du 68HC11F1
0400-0FFF	RIEN
0000-03FF	RAM interne du 68HC11F1



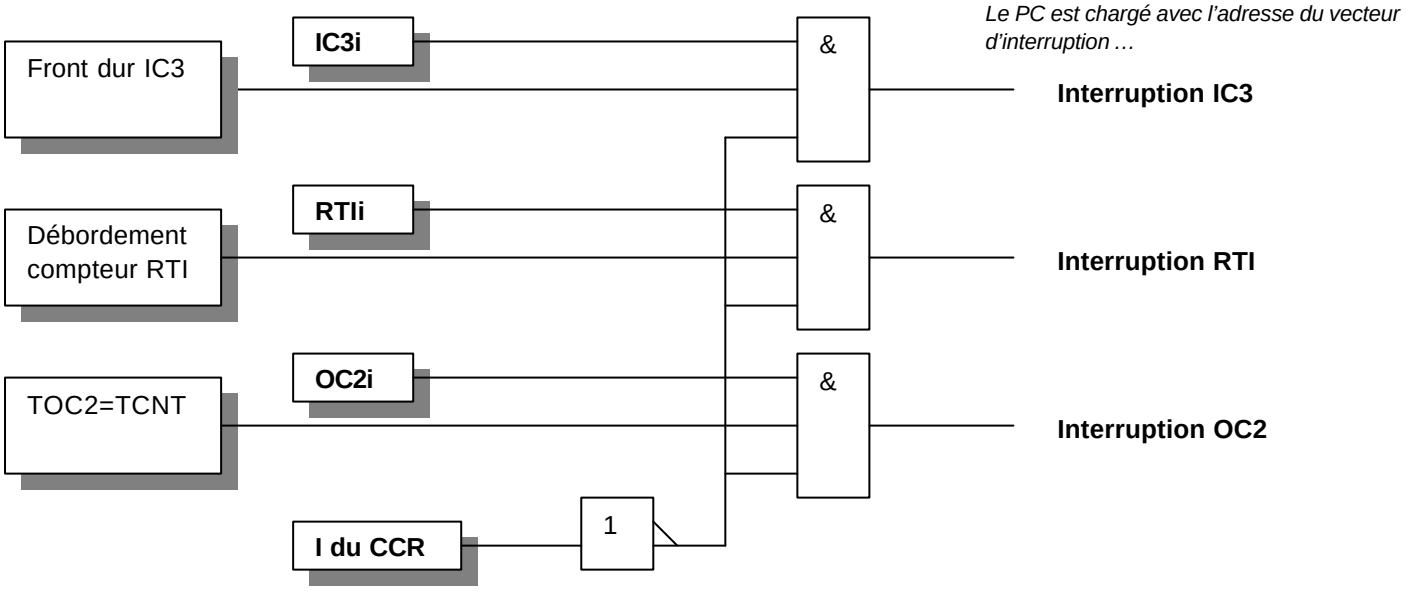

XTAL
EXTAL
E
4XOUT
OSCILLATOR
CLOCK LOGIC
INTERRUPT LOGIC
IRQ
XIRQ
RESET
MODE CONTROL
MODA/ LIR
MODB/ VSTBY
A/D CONVERTER
AN7
AN6
AN5
AN4
AN3
AN2
AN1
AN0
PORT E
PE7
PE6
PE5
PE4
PE3
PE2
PE1
PE0
VRH
VRL
VDD
VSS
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PORT A
PORT A DDR
PAI/OC1
PULSE ACCUMULATOR
COP
OC2/OC1
OC3/OC1
OC4/OC1
OC5/IC4/OC1
IC1
IC2
IC3
TIMER SYSTEM
PERIODIC INTERRUPT
PB7
PB6
PB5
PB4
PB3
PB2
PB1
PB0
PORT B
ADDR15
ADDR14
ADDR13
ADDR12
ADDR11
ADDR10
ADDR9
ADDR8
ADDRESS BUS
PF7
PF6
PF5
PF4
PF3
PF2
PF1
PF0
PORT F
ADDR7
ADDR6
ADDR5
ADDR4
ADDR3
ADDR2
ADDR1
ADDR0
CPU
1024 BYTES RAM
512 BYTES EEPROM
CHIP SELECTS
CSPROG
CSGEN
CSIO1
CSIO2
PORT G DDR
PORT G
PG7
PG6
PG5
PG4
PG3
PG2
PG1
PG0
PC7
PC6
PC5
PC4
PC3
PC2
PC1
PC0
PORT C
PORT C DDR
DATA7
DATA6
DATA5
DATA4
DATA3
DATA2
DATA1
DATA0
DATA BUS
SPI
SS
SCK
MOSI
MISO
SCI
TxD
RxD
PORT D DDR
PORT D
PD5
PD4
PD3
PD2
PD1
PD0
R/W
R/W

68HC11 F1


$0000
$1000
$FE00
$FFFF
EXT
EXT
EXT
EXT
EXT
SINGLE CHIP
EXPANDED
BOOTSTRAP
SPECIAL TEST
x000
x3FF
1024 BYTES RAM 1
y000
y05F
96-BYTE REGISTER BLOCK 2
BF00
256 BYTES BOOTSTRAP ROM
BFC0
BFFF
SPECIAL MODE 3 INTERRUPT VECTORS
zD00
zDFF
256 BYTES RESERVED 4 (SPECIAL TEST MODE ONLY)
zE00
zFFF
512 BYTES EEPROM 5
FFC0
FFFF
NORMAL MODE INTERRUPT VECTORS

68HC11 F1

68HC11 : Interruptions multiples



Le PC est chargé avec l'adresse du vecteur d'interruption …