SYSTÈME À MICROPROCESSEUR

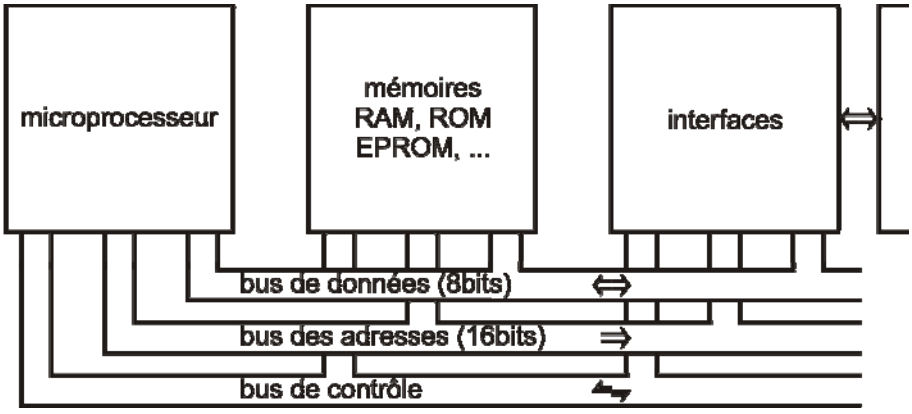
Famille 68xx

1. Généralités

Le microprocesseur est un composant intégré, il est en relation avec le monde extérieur par l'intermédiaire de trois ***bus*** et contient un certain nombre de ***registres***. La taille des bus, le nombre et la taille des registres déterminent, en général, les capacités du microprocesseur.

1.1. Les bus sont :

- Le **bus de données**, il véhicule en entrée et en sortie les informations traitées par le micro (8 bits D0-D7).
- Le **bus d'adresses**, il permet au micro de pointer un des éléments de son environnement (16bits A0-A15).
- Le **bus de contrôle**, permet au micro de gérer son environnement et de recevoir de celui-ci des informations. Le bus de contrôle est spécifique à chaque type de micro.



2. Les registres internes programmables accessibles par l'utilisateur sont :

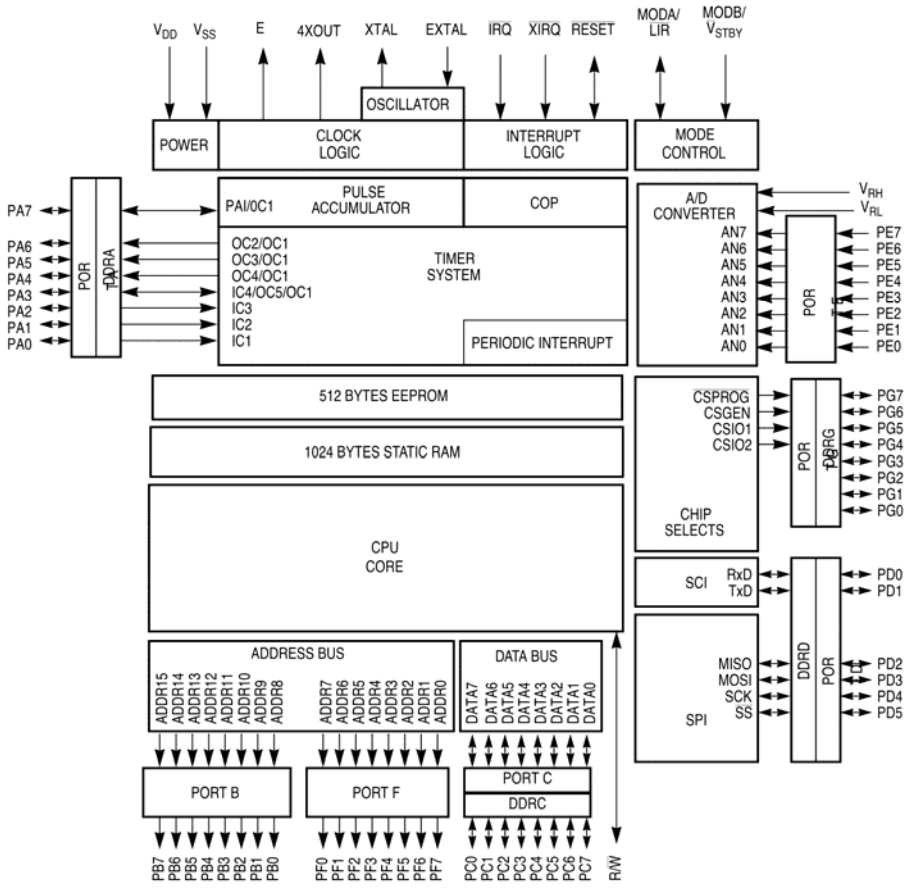
- Les **accumulateurs**, ils permettent les calculs, les manipulations de données, ils peuvent être concaténés (un registre A de 8bits et un registre B de 8bits peuvent être fusionnés pour former un registre D de 16 bits).
- Les **registres** d'**index**, le contenu de ces registres permet de pointer des données.
- Le **registre pointeur de pile**, ce registre permet la gestion des sauvegardes des registres internes pendant l'exécution de sous-programmes.
- Le **registre compteur programme**, ce registre contient l'adresse de la prochaine instruction à exécuter.
- Le **registre code condition**, ce registre définit en permanence l'état des indicateurs (flags) du microprocesseur. Ces indicateurs sont en général de deux types, certains bits sont positionnés par le résultat des opérations effectuées par le micro, d'autres sont liés au fonctionnement de l'environnement du micro (interruptions).

3. Cas particulier : microcontrôleur

Le microcontrôleur est un composant qui rassemble sur une puce un microprocesseur, de la mémoire (RAM et EPROM), des interfaces (série, parallèle, timer, communication, …).

Exemple :

68HC11



Le 68HC11 contient 7 registres :

7 **A** 0	7 **B** 0	**Accumulateurs A** & **B** 8 bits
15 **D** 0		**Ou Accumulateurs D** 16 bits

15 **IX** 0	**Registre d'index X**

15 **IY** 0	**Registre d'index Y**

15 **SP** 0	**Pointeur de Pile**

15 **PC** 0	**Compteur Programme**

S X H I N Z V C	**Registre Codes Condition**

Les drapeaux (flags) du registre Codes Condition ont la signification suivante :

C : **C**arry/**B**orrow from MSB – retenue sur bit de poids fort

V : O**v**erflow – dépassement de capacité

Z : **Z**éro

N : **N**égatif

I : Masque d'**i**nterruptions **IRQ**

H : **H**alf-carry (bit3)

X : Masque d'interruptions **FIRQ** (**f**ast **IRQ**)

S: **S**top disable – État de sauvegarde des registres dans la pile.

4. Les Entrées/Sorties :

Le 68HC11F1 possède 7 ports (54 lignes d'entrée/sortie)

- Port A (8 bits) : Le port A est à l'adresse $1000. Il a deux utilisations possibles
  - Le port A est utilisable pour la gestion du timer intégré au 68HC11. Dans ce mode de fonctionnement 4 broches de ce port sont prédéfinies comme des sorties de comparaison, 3 broches sont des entrées et une broche peut être configurée comme entrée ou comme sortie (bit3 de PACTL à l'adresse $1023).
  - Le port A est utilisable en entrées/sorties logiques, la configuration d'un bit en entrée est obtenue en forçant à 0 le même bit dans le registre DDRA, la configuration en sortie est obtenue par la valeur 1. Le registre DDRA est à l'adresse $1001.

- Port B (8 bits) : tous les bits sont en sortie, dans le mode de fonctionnement étendu ces bits représentent la partie haute de l'adresse. Adresse $1004

- Port C (8 bits) : ces bits sont configurables en entrée ou en sortie (comme le port A). Dans le mode de fonctionnement étendu ces bits représentent la donnée 8bits associée à l'adresse générée sur les ports B et F. Adresse $1006, le DDRC est à l'adresse $1007.

- Port D (6 bits) : configurables en entrée ou en sortie, ce port permet aussi la gestion des communications séries. Adresse $1008, le DDRD est à l'adresse $1009.

- Port E (8 bits) : tous les bits sont en entrée, ce port est utilisable en mode logique ou en mode analogique pour effectuer des conversions analogiques/numériques sur 8 bits. Il est à l'adresse $100A.

- Port F (8 bits) : tous les bits sont en sortie (comme le port B), dans le mode de fonctionnement étendu ces bits représentent la partie basse de l'adresse. Adresse $1005.

- Port G (8 bits) : configurable en entrée ou en sortie comme A et C le port G est en $1002, le DDRG est lui en $1003.

5. Programmation : Compilateur Basic

Le compilateur vous permet d'écrire des programmes en **Basic** sur le système P.C. hôte pour des cibles à base de MC68HC11 de Motorola. **Basic** (**B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) a été inventé en 1964 par John Kemeny et Tom Kurz au Dartmouth College. Le **Basic** a été conçu comme un langage de base, facile à apprendre et à maîtriser en peu de temps. Les premiers systèmes **Basic** étaient des **interpréteurs** qui tournaient sur des ordinateurs bien avant l'âge des microprocesseurs et des P.C. Sa deuxième vie a vu **Basic** en forme d'**interpréteur** qui tournait dans la partie **ROM** des microprocesseurs.
Basic n'est pas normalisé, même s'il existe un 'ANSI Standard for minimum Basic'. Chaque implémentation de langage **Basic** a ses propriétés, souvent dues aux caractéristiques du microprocesseur ciblé, mais aussi aux contraintes de la cible elle même.
On trouve sur les petits et les très petits microprocesseurs des dialectes **Basic** qui se contentent du strict minimum: l'octet comme seul type de donnée, **IF**, **FOR**, **GOTO**, **GOSUB**.
Le **BASIC11** utilise un dialecte qui est plutôt orienté vers un **langage structuré** comme le **Pascal** ou le **C**. Même si on retrouve les fameux **GOTO** et **GOSUB**, il existe bien d'autres éléments pour remplacer ces directives un peu trop **basic**.
Le **BASIC11** n'est pas un **interpréteur** mais un **compilateur** qui traduit le programme source en programme objet. Ce programme est prévu pour être brûlé dans une **PROM**. Il tournera sans l'aide d'un interpréteur ou d'un système d'exploitation sur le microprocesseur **68HC11**.

Quelle est la différence entre un interpréteur et un compilateur ?

- Un interpréteur assure la traduction du programme en langage machine instruction par instruction au moment de son exécution, il est donc très lent. Le programme compilé est lui traduit une fois pour toutes par le compilateur, c'est cette traduction qui est exécutée, le programme est largement plus rapide.
- Un interpréteur sur la cible occupe automatiquement de la place dans la mémoire et reste donc limité au niveau du confort de manipulation et de la richesse du langage.
- L'interpréteur tourne directement sur la carte cible. On n'a donc pas besoin d'un P.C. comme hôte pour compiler le programme.
- Le travail sur un système hôte permet de mieux gérer et de conserver la source et la documentation du programme.

Le compilateur **BASIC11** et l'assembleur **AS11** tournent sous **DOS**. Le compilateur **BASIC11** donne comme résultat un fichier en assembleur qui est traduit par **AS11** en fichier format **Motorola S-record**.
Le programme **WBASIC11** tourne sous **Windows**. Ce programme permet d'éditer les fichiers source, lancer le compilateur sous **DOS** et récupérer les messages du compilateur comme les messages d'erreur. Le **débogueur** permet de charger le programme dans la mémoire de la cible et de mettre en évidence les erreurs dans celui-ci.

Le programme **WBASIC11** vous présente après le lancement deux fenêtres.

- La fenêtre en haut est la fenêtre principale. Elle vous permet de lancer un éditeur pour visualiser le code source et compiler le programme.
- La fenêtre en bas contient le débogueur qui vous permet de communiquer avec la carte cible à base du 68HC11.

Dans le menu FICHIER vous trouvez les outils pour ouvrir et traiter le programme source principal. COMPILER lance le compilateur. Le curseur d'attente s'affiche pendant la compilation. Après on trouve la sortie des programmes de compilation.

!E <fichier>(<ligne>)

Une telle ligne affiche une erreur. Cliquer deux fois sur cette ligne ouvre une fenêtre pour éditer ce fichier et positionne la source sur la ligne erronée.

Le menu FENETRES affiche les fichiers source de l'option et tous les fichiers inclus dans ces fichiers par la directive **#include**.
Cliquer sur un fichier ouvre une fenêtre pour éditer ce fichier.
Le dialogue OPTIONS vous permet de spécifier les options pour compiler le programme.

- Commande de compilateur
- Options de compilation
- Fichiers à compiler
- Répertoires des fichiers sources

On peut donner plusieurs options, fichiers, et répertoires, séparés par espace ou virgule. Les options sont enregistrées dans un fichier <nom>.prj. Le nom vient du nom du fichier source principal.

Exemple de programme :
Le programme suivant permet de commander le clignotement d'une diode électroluminescente présente sur la carte d'étude ControlBoy.

- Associer le code source fourni à son algorigramme

Après avoir raccordé la carte ControlBoy au PC puis au secteur, démarrer le PC :

- Lancer ControlBoy
- Taper le programme suivant dans l'éditeur de texte

```
*Programme pour une carte ControlBoy F1
    ProgramPointer      $8000
    DataPointer         $0002
    StackPointer        $7FEF

    byte DDRG  at $1003
    byte PORTG at $1002

    int  i

    DDRG.0=1                        *  PG0 = sortie
    ASM   cli                       * enable debugger
    do                              * faire
          if PORTG.0=0 then
                        PORTG.0=1
                else
                        PORTG.0=0
          end if
          if PORTG.1=1 then
                        i=20000
                else
                        i=5000
          end if
          for i = i to 0 step -1    * tempo boucle for… next
          next i
    loop                            *fin faire
```

- Compiler le programme
- Transférer le programme sur la cible
- Run ….