

Chapitre I :

*Automate
programmable et
grafcet*

I*1* Introduction :

La réalisation et l'exploitation des automatismes industriels font appel à divers outils, matériels et logiciels, qui doivent être convenablement choisis et utilisés, en vue de remplir certaines fonctions.

Lors de la conception des équipements d'automatismes, plusieurs familles technologiques (relais électriques, séquenceurs pneumatiques, automates programmables industriels, micro et mini-ordinateurs industriels .etc.) Peuvent être envisagées et un choix sera fait au moment de la construction du système.

Le développement de ces outils matériels destinés à la réalisation de la partie commande des systèmes automatisés offre des commodités d'emploi et des performances toujours croissantes. L'un de ces outils matériels les plus répandu est l'automate programmable. L'automate programmable est un système de commande en pleine évolution, la demande sur le marché est de plus en plus grande. Les applications envisagées sont de plus en plus variées et des utilisateurs de tous les domaines s'y intéressent.

I*2* Caractéristiques générales d'un automate programmable :**I*2*1* Définition :**

Un automate est une machine que l'on programme comme on veut et qui limite les décisions systématiques d'un homme. Un Automate Programmable Industriel (API) est une commande conçue autour d'un microprocesseur. Sa conception et son langage de programmation sont spécialement adaptés aux contrôles des processus industriels.

Les caractéristiques d'un automate programmable fixent les limites de ses performances et sont :

- La capacité d'entrées / sorties,
- La capacité de programme,
- La capacité de mémorisation de données,
- La capacité de mémorisation de textes,
- Le temps de cycle et le temps de réaction,
- Les possibilités du jeu d'instructions,
- Les possibilités de communication.

I*2*2* Structure d'un automate programmable :

I*2*2*1* Structure générale d'un automate programmable :

Les éléments principaux que l'on rencontre toujours dans un API sont l'alimentation, l'unité centrale CPU (central processor unit), la mémoire, les modules d'entrées / sorties et la console de programmation (fig.I.1)

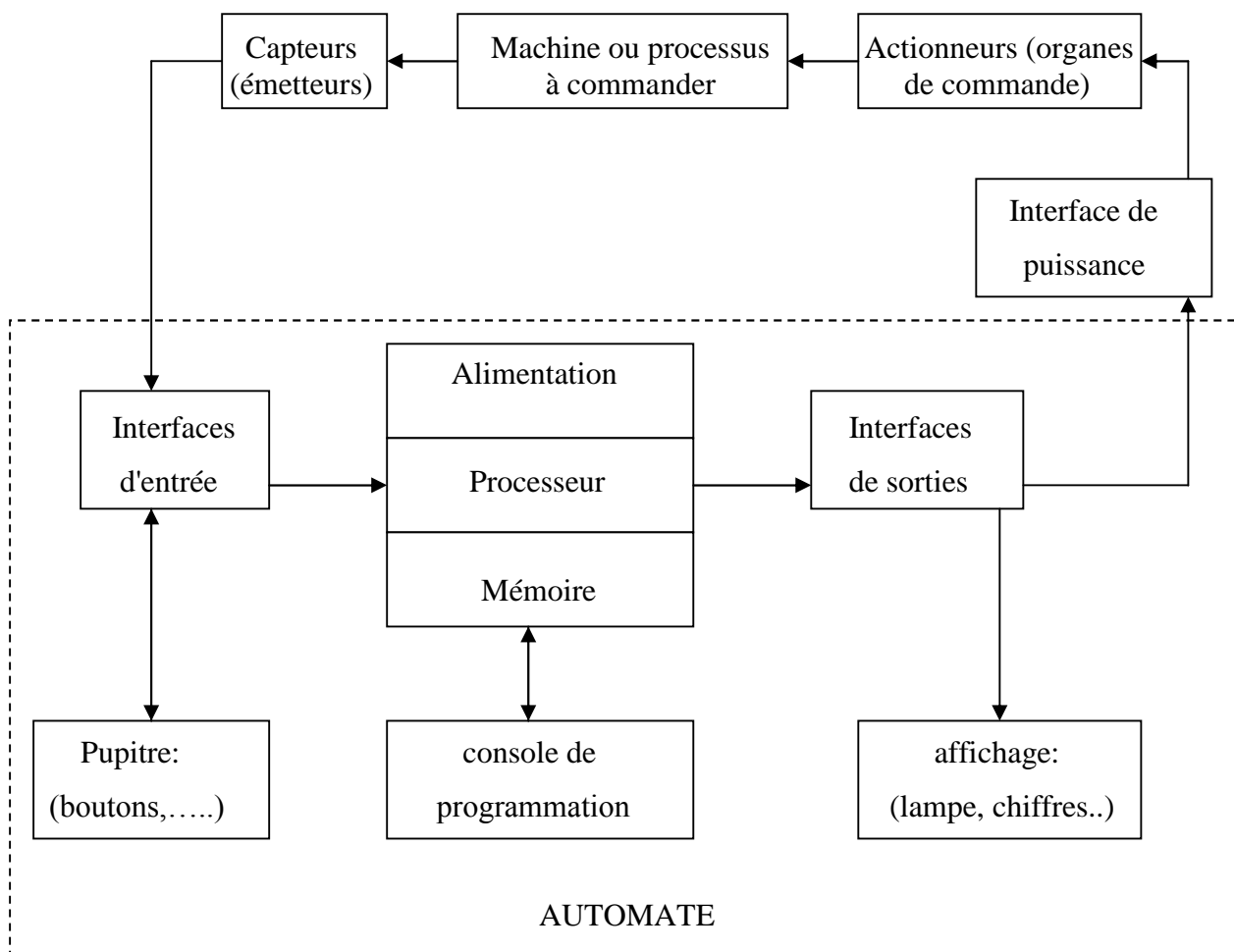


Fig.I.1: Structure Générale d'un API

I*2*2*2* Module d'alimentation :

C'est un module qui fournit une tension stable pour assurer le bon fonctionnement des autres modules.

I*2*2*3* L'unité centrale (CPU : central processor unit) :

L'unité centrale est le cerveau de l'automate, elle lit les états des signaux des entrées, exécute le programme d'utilisateur et commande les sorties.

I*2*2*4* Module d'entrées / sorties (E/S) :

Ce sont les interfaces qui permettent les échanges d'informations avec l'environnement extérieur de l'automate, il en existe deux types :

Le module d'E/S tout ou rien (TOR) exploité pour des applications ne faisant intervenir que les états 0 et 1 (binaire).

Le module d'E/S analogique qui traite les grandeurs variant de façon continue (courant, tension).

I*2*2*5* La mémoire :

Elle est disponible, pour la conservation du programme utilisateur et le transfert dans l'automate. La cartouche mémoire est de trois types :

EPROM (contient le système d'exploitation)

EEPROM (De type programmable par l'utilisateur et effaçable électriquement),

RAM (La mémoire vive, elle est réservée à la sauvegarde de données et à la pile).

I*2*2*6* La console de programmation :

C'est une unité qui nous permet le développement et le test du programme utilisateur ainsi que des opérations sur les éléments internes et les entrées / sorties. Pour cela on la branche à l'automate par une prise spéciale sur le module CPU.

I*2*3* Méthodes de programmation des automates programmables :

Pour la programmation, on distingue deux types de langages : les langages graphiques et les langage littéraux.

I*2*3*1* Langages littéraux :

Dans cette forme de programmation, les instructions s'écrivent sous la forme d'expressions littérales utilisant des parties textuelles ou mots réservés.

I*2*3*2* Le grafcet (description générale) :

La création d'une machine automatisée nécessite un dialogue entre le client qui définit le cahier des charges (qui contient les besoins et les conditions de fonctionnement de la machine) et le constructeur qui propose des solutions. Ce dialogue n'est pas toujours facile : le client ne possède peut-être pas la technique lui permettant de définir correctement son

problème. D'autre part, le langage courant ne permet pas de lever toutes les ambiguïtés dues au fonctionnement de la machine (surtout si des actions doivent se dérouler simultanément). C'est pourquoi l'ADEPA (Agence pour le développement de la productique appliquée à l'industrie) a créé le GRAFCET.

I*3* Le grafcet :

I*3*1* Définition :

Le GRAFCET (Graphe Fonctionnel de Commande des étapes et Transitions) est l'outil de représentation graphique d'un cahier des charges. Il a été proposé par l'ADEPA (en 1977 et normalisé en 1982 par la NF C03-190).

Le GRAFCET est une représentation alternée d'étapes et de transitions. Une seule transition doit séparer deux étapes (fig.I.2).

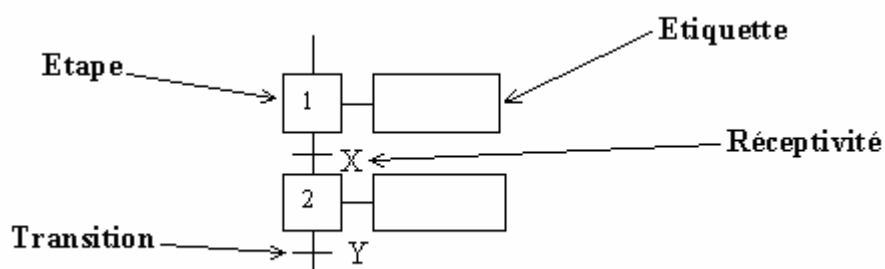


Fig.I.2: Les composants d'un grafcet

Une étape correspond à une situation dans laquelle les variables de sorties conservent leur état. Les actions associées aux étapes sont inscrites dans les étiquettes.

Une transition indique la possibilité d'évolution entre deux étapes successives. A chaque transition est associée une condition logique appelée réceptivité.

I*3*2* Règles d'évolution :

Il est plus que nécessaire de fixer un ensemble de règles d'évolution pour un GRAFCET.

Règle 1 :

L'initialisation précise les étapes actives au début du fonctionnement. Elle est activée inconditionnellement et repérées sur le GRAFCET en doublant les côtés des symboles correspondants.

La figure (I.3) représente la règle 1.



Fig. I.3: Etape initiale active

Règle 2 :

Une transition est soit non validée. Elle est validée lorsque toutes les étapes immédiatement précédentes sont activées et ne peut être franchie que si :

Elle est validée

Et que la réceptivité associée à la transition est vraie

La transition est obligatoirement franchie.

La figure (I.4) explique cette règle

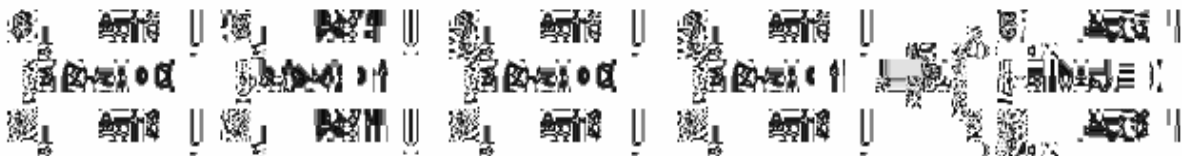


Fig.I.4:règle 2

Règle 3 :

Le franchissement d'une transition entraîne l'activation de toutes les étapes immédiatement suivantes et désactivation de toutes étapes immédiatement précédentes. Cette évolution du GRAFCET est donc synchrone lorsque le franchissement de la transition entraîne l'activation des étapes suivant et que c'est la vérification de cette activation qui autorise la désactivation des étapes précédentes.

La figure (I.5) explique cette règle.

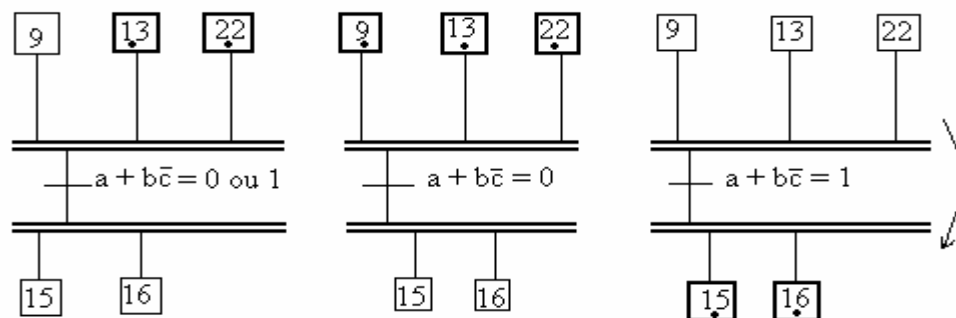


Fig.I.5: le franchissement d'une transition

Règle 4 :

Il est possible d'ouvrir plusieurs transitions simultanément franchies.

Règle 5 :

Si au cours du fonctionnement, une même étape doit être désactivée simultanément, elle reste activée. L'activation doit être prioritaire sur le niveau d'une même étape.

I*3*3* Structures de base :

I*3*3*1* Divergence et convergence en ET (séquences simultanées) :

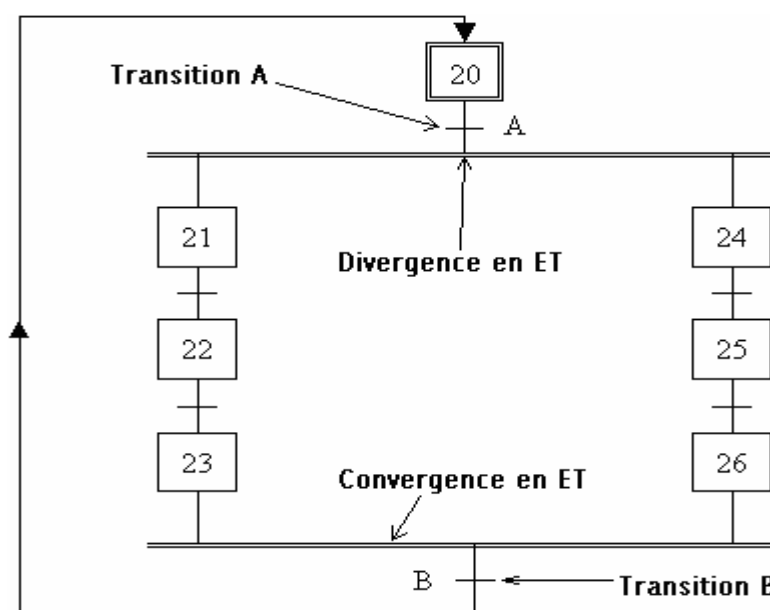


Fig.I.6: Divergence et convergence en ET (séquences simultanées)

Divergence en ET : lorsque la transition A est franchie, les étapes 21 et 24 sont actives.

Convergence en ET : la transition B sera validée lorsque les étapes 23 et 26 seront actives. Si la réceptivité associée à cette transition est vraie, alors celle-ci est franchie.

Remarques :

Après une divergence en ET, on trouve une convergence en ET.

Le nombre de branches parallèles peut-être supérieur à 2.

La réceptivité associée à la convergence peut-être de la forme $= 1$. Dans ce cas la transition est franchie dès qu'elle est active.

I*3*4*2* Divergence et convergence en OU (aiguillage) :

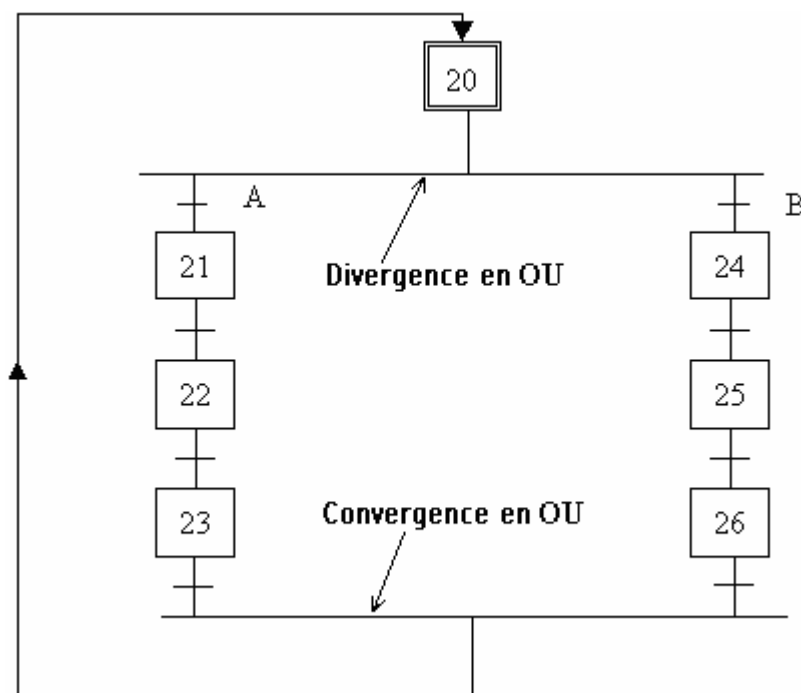


Fig.I.7: Divergence et convergence en OU (aiguillage)

Divergence en OU : l'évolution du système vers une branche dépend des réceptivités A et B associées aux transitions.

Convergence en OU : après l'évolution dans une branche, il y a convergence vers une étape commune.

Remarques :

A et B ne peuvent être vrais simultanément (conflit).

Après une divergence en OU, on trouve une convergence en OU.

Le nombre de branches peut-être supérieur à 2.

La convergence de toutes les branches ne se fait pas obligatoirement au même endroit.

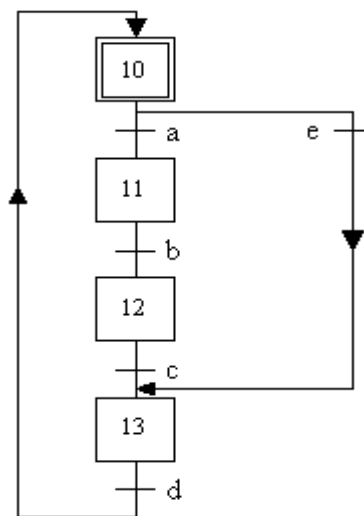
Saut en avant (saut de phase) :

fig.I.8: Saut en avant (saut de phase)

Le saut en avant permet de sauter une ou plusieurs étapes lorsque les actions à réaliser deviennent inutiles.

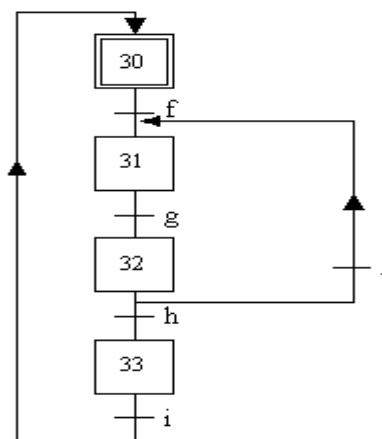
Saut en arrière (reprise de phase) :

Fig.I.9: Saut en arrière (reprise de phase)

Le saut en arrière permet de reprendre une séquence lorsque les actions à réaliser sont répétitives.

I*4* Conclusion :

Les A.P.I. utilisent des langages différents selon les constructeurs et les pays d'origine. Nous trouvons des automates utilisant le langage logique, basé sur les fonctions logiques, et des automates utilisant le langage des contacts basé sur les associations de contacte comme dans les schémas à relais, d'où leur succès auprès des électriciens. D'autres automates plus performants utilisent le langage grafcet, qui permet la transcription immédiate du grafcet en programme ou le langage organigramme basé sur des représentations inspirées de l'informatique industriel. En plus l'A.P.I. présente l'avantage par rapport aux solutions câblées, d'autoriser un travail très complet, et il est manipulable par un personnel non informaticien et de qualification moyenne.

Chapitre II :
Le microcontrôleur
68HC11

II*1* Introduction :

L'électronique actuelle repose de plus en plus sur l'utilisation des composants programmables, tel que les microprocesseurs, les microcontrôleurs et les DSP (digital signal processeur). Ceci explique d'une part que leur utilisation simplifie considérablement les structures électroniques et qu'il est généralement facile de modifier leurs comportements par modification de leur programme d'autre part.

De plus, les outils de développement associés à ces composants permettant aisément d'effectuer une simulation avant leur implantation et rendent possible la détection et la correction des éventuelles erreurs pendant la phase de développement.

Tous cela contribue à réduire le coût de fabrication de produit utilisant des composants programmables.

II*2* Microprocesseurs 8 bits (6809) :**II*2*1* Définition du microprocesseur :**

Un microprocesseur est un circuit intégré logique piloté par une série d'instructions programmables et capable d'organiser des transferts d'informations entre mémoires et périphériques d'un système ainsi que de réaliser des opérations arithmétiques et logiques.

Remarque : L'exécution des instructions est cadencée par un signal d'horloge.

II*2*2* Les Mémoires :

Cette série de commandes ou d'instructions constitue **le programme** qu'on stocke dans une **mémoire permanente** appelée **ROM** (Read-only Memory). Pendant l'exécution du programme, le microprocesseur dispose d'une **mémoire temporaire** nommée **RAM** (Random-access Memory) qui permet de stocker certaines informations utilisées lors de l'exécution du programme. On utilise le plus souvent l'**EPROM** (Erasable & Programmable ROM) qui présente l'avantage de pouvoir être effacée une fois écrite.

Remarque : L'effacement de l'EPROM se fait au moyen d'un tube à rayons ultraviolets. C'est pourquoi les EPROM se distinguent des autres mémoires par la présence sur leur face

supérieure d'une petite fenêtre de quartz, souvent obturée par un adhésif de manière à ne pas être exposée aux rayons ultraviolets naturels (soleil, néon...)

II*2*3* Les Bus :

Ce sont des ensembles de 8 lignes parallèles qui assurent les transferts de données.

Il existe trois grands types de bus : bus de données, bus d'adresses, bus de contrôle.

II*2*3*1* Data Bus (DB) :

Le bus de données (Data Bus), sert à transporter :

- Les instructions de la mémoire programme ROM vers l'unité centrale (CPU: Central Processing Unit).
- Les données lues ou écrites en mémoire temporaire RAM.
- Les données qui proviennent de l'extérieur du système, ou qui vont vers l'extérieur par les ports d'entrée et de sortie.

II*2*3*2* Adresse Bus (AB) :

Les circuits de mémoire RAM et ROM comportent un grand nombre d'informations, chaque instruction ou donnée se trouvant à une certaine adresse. C'est en pilotant le bus d'adresses (AB) que l'unité centrale sélectionne :

- L'instruction dont elle a besoin dans la ROM.
- La case de RAM à laquelle elle souhaite accéder pour lire ou écrire une donnée.
- Le périphérique auquel elle souhaite accéder pour lire une information (port d'entrée) ou donner une commande (port de sortie).

II*2*3*3* Control Bus (CB) :

Le bus de contrôle comporte :

- Des lignes qui permettent à l'unité centrale de spécifier à la RAM ou aux ports d'entrée et de sortie si elle veut faire une écriture ou une lecture.
- Des lignes utilisées par les périphériques pour transmettre des signaux à l'unité centrale (demande d'interruption, d'accès direct à la mémoire...)
- Des lignes utilisées par l'unité centrale pour répondre aux périphériques (acceptation d'une demande d'interruption ou d'accès direct à la mémoire...)

Remarque : Liaisons

Les fils reliant l'émetteur au récepteur sont exposés aux impulsions parasites de toutes natures. Pour améliorer l'immunité au bruit de la liaison, la différence de potentiel entre les fils de liaison et la masse est accrue.

II*2*4* Ports d'entrée et de sortie :

Le microprocesseur communique avec l'extérieur par l'intermédiaire des ports d'entrée et de sortie. Les différents périphériques (touches de commande, capteurs, moteurs...) sont raccordés au microprocesseur par ces circuits d'interfaces.

II*2*4*1* Port d'entrée :

La fonction essentielle d'un port d'entrée est **d'isoler le bus de données** du microprocesseur des circuits périphériques. Le circuit d'entrée doit être capable de décoder les adresses présentes sur le bus d'adresses afin de connecter les lignes provenant des circuits périphériques au bus de données au moment adéquat. Les signaux appliqués à l'entrée du port doivent être obligatoirement de type binaire.

II*2*4*2* Port de sortie :

Lorsque le microprocesseur décide d'envoyer une donnée vers un circuit périphérique il fait apparaître la donnée sur le bus de données pendant un instant très court : la fonction essentielle d'un port de sortie est de mémoriser cette donnée et de la maintenir à disposition du périphérique. Le circuit de sortie est capable de décoder les adresses présentes sur le bus d'adresses afin de verrouiller la donnée présente sur le bus de données.

II*2*5* Les interruptions :

Les interruptions permettent de faire des transferts de données entre le microprocesseur et les périphériques qui provoquent eux-mêmes la demande d'interruption par la ligne INT (Interrupt) du bus de contrôle qui est réservée à cet effet. L'unité centrale termine l'instruction en cours, sauvegarde le contenu du PC dans la pile et effectue un branchement vers le sous-programme approprié qui effectue la tâche requise par le périphérique. Lorsque le sous-programme d'interruption est terminé, il faut replacer dans le PC l'adresse sauvegardée dans la pile et l'exécution du programme principal peut alors continuer.

II*2*6* Traitement des données :**II*2*6*1* Accumulateur :**

Registre 8 bits jouant un rôle central dans la plupart des microprocesseurs 8 bits.

- Tous les transferts de données entre cellules de la mémoire ou entre cellule de mémoire et périphérique s'effectuent en deux temps

(1) : transfert du registre source (mémoire ou port d'Entrée/Sortie) vers l'accumulateur

(2) : transfert de l'accumulateur vers le registre de destination.

- Toutes les opérations arithmétiques et logiques exigent qu'un des deux nombres soit dans l'accumulateur. Le résultat de l'opération est ensuite remplacé dans l'accumulateur.

II*2*6*2* Registre de travail :

Mémoire temporaire (registre 8 bits) pour stocker les résultats intermédiaires lors des calculs.

II*6*3* Registres temporaires :

Registres utilisés lors des opérations arithmétiques et logiques.

Par exemple, pour additionner deux nombres, on place d'abord le 1^{er} nombre dans l'accumulateur au moyen d'une instruction de transfert, le 2^{ème} nombre est placé dans le registre temporaire 2 et simultanément le 1^{er} nombre est transféré dans le registre temporaire 1 de manière à ce que les deux nombres à additionner apparaissent aux entrées de l'unité arithmétique et logique qui exécute l'opération demandée. Enfin, le résultat de l'opération apparaît en sortie de l'unité arithmétique et logique est envoyée dans le bus de données interne et verrouillé dans l'accumulateur.

II*2*7* Unité arithmétique et logique :

Elle effectue les opérations arithmétiques et logiques portant sur deux nombres au maximum.

Cette unité est constituée d'un additionneur et de 8 circuits logiques (AND, OR, XOR) et 8 inverseurs pour les opérations logiques.

II*2*8* Gestion des adresses :

Pour pouvoir exécuter un programme, le microprocesseur doit pouvoir accéder à la mémoire pour :

- lire les instructions à exécuter (souvent en ROM)
- lire les données à traiter : celles-ci se trouvent soit dans la zone de mémoire réservée aux données, soit à l'extérieur du système –introduites par les ports d'entrée-
- écrire les résultats du traitement des données en mémoire ou les transmettre à des périphériques par l'intermédiaire des ports de sortie.

II*2*8*1* Pointeur de programme (Program Counter) :

Registre de 16 bits contenant à chaque instant l'adresse de la prochaine instruction à exécuter. La 1^{ère} instruction du programme doit se trouver à la cellule mémoire d'adresse 0000H ; le bloc mémoire commençant en 0000H doit donc être un bloc de mémoire programme.

Dans les microprocesseurs de la famille 6800 de Motorola, le pointeur de programme est chargé, au moment de l'initialisation, avec le contenu des cellules mémoire d'adresses FFFE_H et FFFF_H ; ceci donne une plus grande souplesse, puisque le programme ne doit pas démarrer obligatoirement en 0000H.

II*2*8*2* Pointeur de données (Data Pointer) :

Pour accéder à des données situées dans la mémoire externe, l'adresse de la donnée soit se trouver dans le pointeur de données ; au moment adéquat, l'unité de contrôle et de séquençement fait apparaître l'adresse contenue dans le DP sur le bus d'adresse, ce qui permet de lire ou d'écrire dans la cellule souhaitée.

II*2*8*3* Pointeur de piles (Stack pointer) :

Registre de 16 bits permettant de créer dans l'espace mémoire en plus des zones de programme et de données, une troisième zone.

II*2*9* Traitement des instructions :**II*2*9*1* Registre d'instructions (Instruction Register) :**

Registre où est stockée l'instruction (en provenance de la mémoire programme) pendant son exécution. Si l'instruction comporte plusieurs octets, seul le 1^{er} est chargé (code opératoire) dans le registre d'instructions.

II*2*9*2* Décodeur d'instructions :

Quand il reçoit à ses entrées le code opératoire il indique au circuit de contrôle et séquençement quelle instruction il doit exécuter.

II*2*9*3* Circuit de contrôle et séquençement :

Etablit les liaisons nécessaires

Commande les opérations d'écriture

II*2*10* Exécution des instructions par l'unité centrale :

L'exécution d'une instruction quelconque peut être décomposée en une succession d'opérations élémentaires de transfert de données entre registres du microprocesseur, cellules de la mémoire et ports d'E/S. Les données sont soit transférées sans modification soit modifiées par un passage à travers l'unité arithmétique et logique.

Ces opérations élémentaires sont nommées micro-instructions (ou cycles machines). Une instruction est un ensemble de micro-instructions qui elles-mêmes sont un ensemble de cycles horloge (ou microcycles). Le nombre de cycles machines nécessaires pour exécuter une instruction est variable. En général, on trouve :

- Un premier cycle consacré à l'appel de l'instruction.
- Un ou deux cycles consacrés à la lecture en mémoire d'une donnée ou d'une adresse.
- Un cycle d'exécution de l'opération demandée et d'écriture du résultat dans un registre.

Remarque : Les données, les adresses et les instructions sont toutes représentées par des octets et rien ne permet à priori de distinguer une instruction, d'une donnée par exemple. Tout octet lu au moment du cycle d'appel d'instruction est considéré comme une instruction.

Exemple : Pour transférer le contenu de l'accumulateur dans une cellule mémoire :

- L'adresse de la cellule mémoire est dans le pointeur de données ; elle apparaît sur le bus d'adresse au début du cycle machine.
- Le circuit de contrôle et de séquençement établit une liaison entre les sorties de l'accumulateur et les entrées de la mémoire, en ouvrant les tampons 3E. Entre l'accumulateur et le bus interne puis entre le bus de données interne et externe. Le contenu de l'accumulateur apparaît aux entrées de la mémoire. Il ne reste plus qu'à activer l'entrée WR de la mémoire pour stocker l'information.

II*2*11* Programmation des microprocesseurs :

II*2*11*1* Architecture des microprocesseurs :

Le microprocesseur apparaît comme un ensemble de registres sur lesquels on opère :

<i>8 bits</i>	<i>16 bits</i>
Accumulateur	Pointeur de programme
Indicateur d'état	Pointeur de données
Registres de travail	Pointeur de piles
	Registre d'index

Tab.II.1 architecture des microprocesseurs

Une instruction comporte plusieurs octets. Le 1^{er} octet (Code opératoire) spécifie le type d'instruction ; il peut être suivi d'un ou deux octets spécifiant une donnée ou une adresse.

II*2*11*2* Modes d'Adressage des données :

Il existe diverses façons de spécifier la source et la destination lors du transfert de données.

- Adressage implicite :

Lorsque la donnée se trouve dans un registre bien précis de l'unité centrale, il n'est pas nécessaire de spécifier une adresse.

- Adressage registre :

L'unité centrale ne contenant qu'un nombre limité de registres de travail, il suffit d'un petit nombre de bits pour spécifier l'adresse du registre. (3 bits suffisent lorsqu'il y a 8 registres). Cette adresse de registre est généralement placée dans l'octet de code opératoire.

Adressage direct

Adressage indirect à registre

Adressage immédiat

II*2*11*3* Principaux types d'instruction :**- Instructions de transfert :**

Instructions permettant le transfert d'octets entre la mémoire, les registres de l'unité centrale et les périphériques

- Instructions arithmétiques et logiques :

Les opérations arithmétiques directement exécutables par tous les microprocesseurs 8 bits sont les additions et soustractions portant sur les entiers compris entre 0 et 255 ou -128 à 127. Les opérations arithmétiques positionnent les indicateurs d'états en fonction du résultat de l'opération (ceux-ci indiquent si le contenu de l'accumulateur est nul, positif, négatif, si la parité est paire ou impaire, s'il y a eu dépassement de la capacité de l'accumulateur).

L'unité centrale est aussi capable d'exécuter des opérations logiques sur deux données de 8 bits. Les fonctions logiques disponibles sont **AND**, **OR**, **XOR**. Les opérations sont effectuées sur les bits de même poids dans les deux octets.

La comparaison de deux nombres A et B s'effectue en faisant la soustraction A-B mais le résultat de cette soustraction n'est pas placé dans l'accumulateur. C'est en testant l'état des indicateurs d'état que l'on peut déterminer si les deux nombres sont égaux (l'indicateur de zéro est positionné) ou différents (c'est l'indicateur de signe qui permet alors de savoir lequel est le plus grand). Les opérations logiques à l'exception de la comparaison ne modifient pas les indicateurs d'état.

- Instructions booléennes :

Instructions agissant sur des bits (et non pas sur les octets comme les instructions de transfert et instructions arithmétiques et logique)

- Instructions de branchement :

Ces instructions n'agissent pas sur des données mais provoquent des modifications dans le déroulement du programme

- Instructions de la gestion de la pile :

La pile est consacrée à la sauvegarde d'informations qui n'ont d'importance que pendant l'exécution d'une partie d'un programme. Son emplacement est déterminé par le programmeur qui doit prévoir, en début de programme, une instruction d'initialisation du pointeur de pile. Les instructions de gestion de la pile permettent de transférer des octets de différents registres internes de l'unité centrale vers la pile (PUSH) et vice-versa (POP). Le transfert est effectué vers ou à partir de la cellule mémoire dont l'adresse est située dans le pointeur de pile. Celui-ci pointe en permanence vers la première cellule libre de la pile. Le pointeur de pile est automatiquement incrémenté ou décrémenté à chaque accès. Lorsque plusieurs informations doivent être sauvegardées, elles sont placées dans les cellules consécutives par une série d'instructions PUSH et elles seront récupérées par des POP, la dernière donnée sauvée étant la première à ressortir de la pile.

II*3* Le microcontrôleur 68HC11 :**II*3*1* Introduction :**

Le 68HC11 est un microcontrôleur 8 bits, fabriqué par MOTOROLA qui peut adresser 64Ko de mémoires. Le jeu d'instruction est dérivé de celui de ces ancêtres : 6801, 6805 et autres 6809. Il est Décliné en différentes versions comportant chacune des tailles de mémoires différentes (RAM, EPROM, EEPROM). Les autres périphériques restent identiques : liaison série (SPI est SCI), 8 entrées ADC, timers (Input capture, Output Compare, Real Time Interrupt, Pulse Accumulator, Interface parallèle). La consommation du circuit est d'environ 15mA en fonctionnement normal (en mode SINGLE) ou 27mA (en mode EXPANDED), de 6mA en mode WAIT et de 50uA en mode STOP.

II*3*2* Architecture interne du 68HC11 :

Le microcontrôleur 68HC11 fait parti d'une vaste gamme de microcontrôleurs portant le même nom générique et compatibles entre eux sur le plan logiciel. Selon le type, les ressources internes diffèrent.

La figure (II.1) présente l'architecture interne d'un 68HC11F1 :

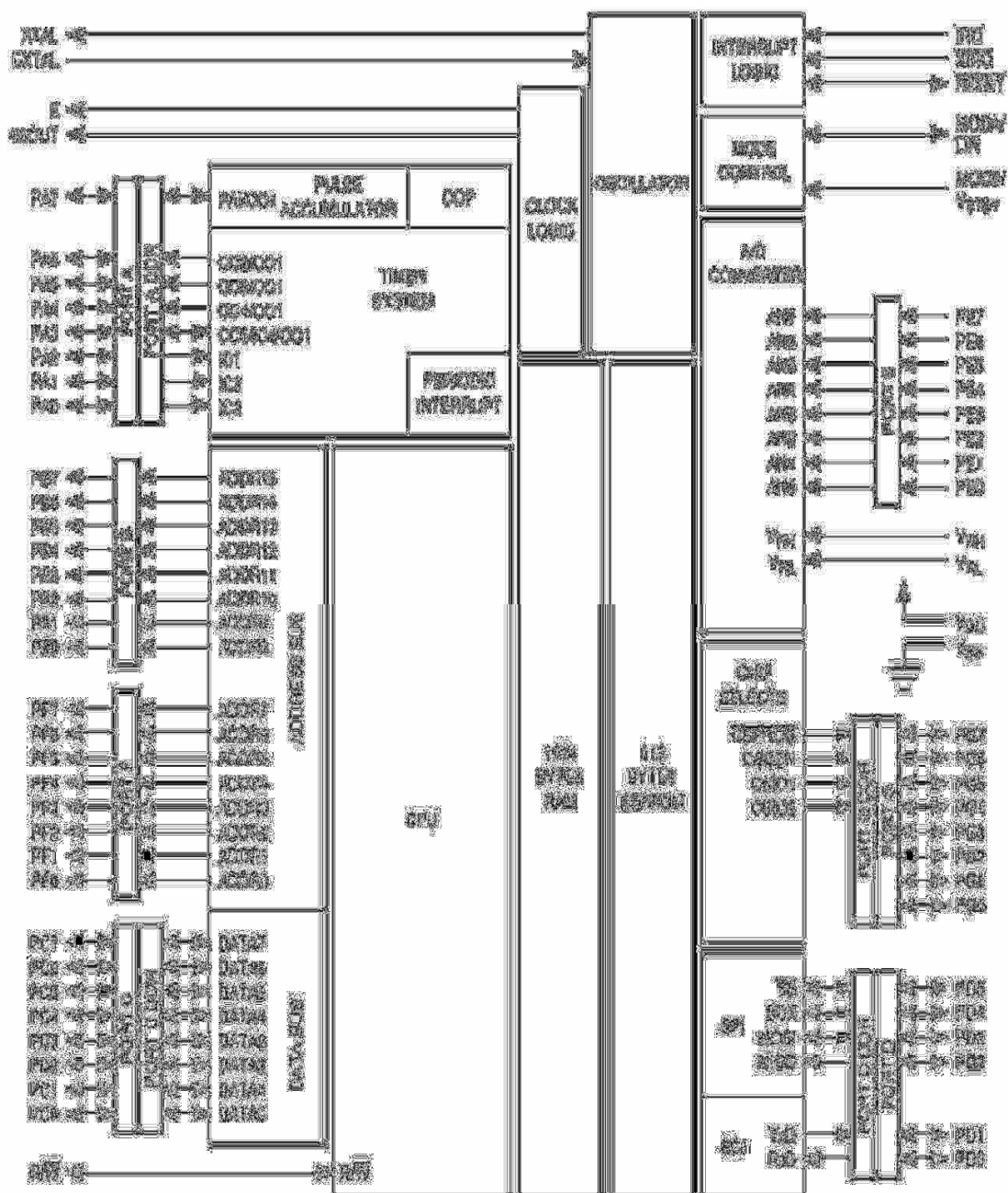
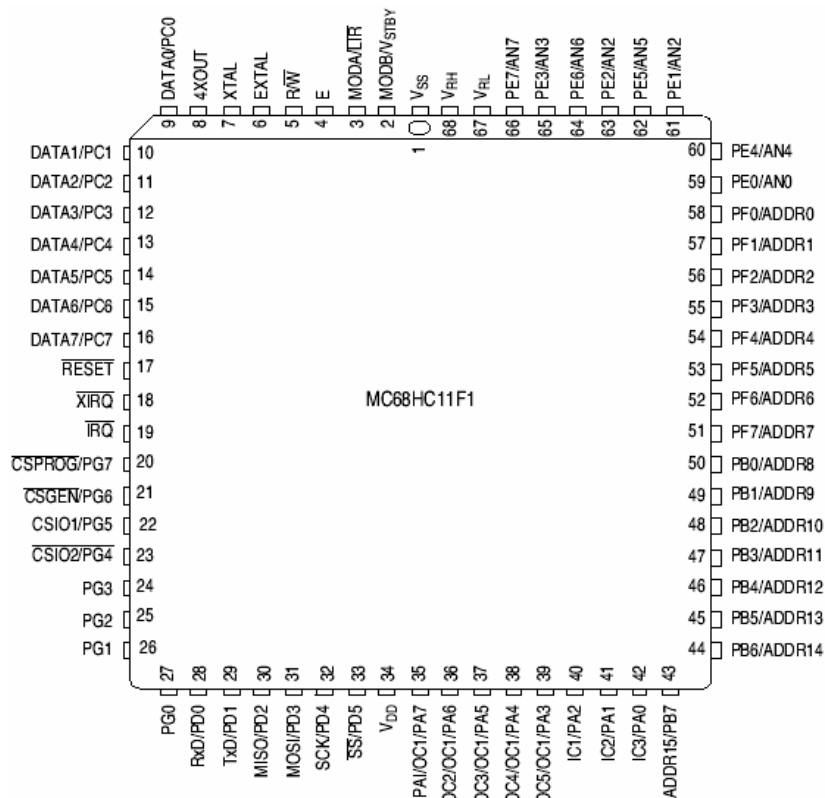


Fig.II.1 schéma de bloc de 68HC11 (logiciel moto6811)

3*2*1* Brochage :**Fig.II.2 : Brochage du MC68HC11F1**

- **Broche RESET :**

La broche RESET est une ligne bidirectionnelle : un niveau bas sur cette entrée durant un temps supérieur à un cycle provoque une réinitialisation du microprocesseur. Le programme sera ensuite exécuté à partir de l'adresse pointée par le vecteur de RESET (adresses \$FFFE et \$FFFF). A la mise sous tension, cette ligne doit être maintenue à l'état bas jusqu'à ce que l'oscillateur d'horloge ait atteint un régime de fonctionnement normal. Lorsque le chien de grade provoque une réinitialisation interne du microcontrôleur, RESET devient une sortie qui passe au niveau bas provoquant ainsi la réinitialisation de tous les composants périphériques reliés à cette broche.

- **Broche E, XTAL et EXTAL :**

Les broches E, XTAL et EXTAL sont les trois broches concernant l'horloge. On connecte le quartz sur les broches XTAL et EXTAL en parallèle avec une résistance de 2.2MΩ (entre 1MΩ et 10MΩ), chaque patte étant reliée à la masse par l'intermédiaire d'une capacité de

22pF. La broche E est une sortie d'horloge (fréquence du quartz / 4) qui permet de synchroniser les échanges avec les composants périphériques lorsque le microcontrôleur est en mode étendu (EXPANDED).

- **Broche RxD :**

Lorsqu'on utilise l'interface série asynchrone SCI (Serial Communication Interface), cette broche réceptionne les données. Le dialogue s'effectue selon le protocole RS232.

- **Broche TxD :**

Lorsqu'on utilise l'interface série asynchrone SCI (Serial Communication Interface), cette broche permet d'émettre les données. Le dialogue s'effectue selon le protocole RS232.

- **Broche MISO : (PD2)**

Lorsqu'on utilise l'interface série synchrone SPI (Serial Peripheral Interface), cette broche est soit une entrée en mode maître, soit comme une sortie en mode esclave. La broche MISO d'un esclave est placée en haute impédance quand ce dernier n'est pas sélectionné.

- **Broche MOSI : (PD3)**

Lorsqu'on utilise l'interface série synchrone SPI (Serial Peripheral Interface), cette broche est soit une sortie en mode maître, soit comme une entrée en mode esclave.

- **Broche SS : (PD5)**

Lorsqu'on utilise l'interface série synchrone SPI (Serial Peripheral Interface), la broche /SS (Slave Select) permet de sélectionner les différents esclaves. Lorsque ce signal est à l'état haut, l'esclave est inactif. Il ignore les signaux d'horloge et il maintient la broche de sortie MISO en haute impédance. Lorsque ce signal est à l'état bas, l'esclave est actif et envoie les données de MOSI à la cadence de l'horloge SCK tout en renvoyant le contenu de son registre SPDR sur la ligne MISO (échange de registre SPDR entre le maître et l'esclave).

- **Broche SCK : (PD4)**

Lorsqu'on utilise l'interface série synchrone SPI (Serial Peripheral Interface), cette broche en est le signal d'horloge fournie par le maître. Cette broche devient une entrée sur les périphériques SPI esclaves.

- **Broche VREFL et VREFH :**

VREFL et VREFH sont les entrées de référence de tension utilisées par le convertisseur analogique/numérique. VREFL est le niveau bas de Référence (typiquement 0 volts) et VREFH est le niveau haut de référence qui doit avoir un potentiel supérieur de 3volts à celui de VREFL. Les deux potentiels doivent être compris entre VSS et VDD.

- **broche STRB :**

Lorsque le microcontrôleur 68hc11 est utilisé en mode EXPANDED, la broche STRB et le signal R/W de lecture ou d'écriture sur le bus de données. Lorsque le 68HC11 est en mode SIGNAL, cette broche est utilisée comme une entrée permettant la synchronisation des échanges sur le port C (handshake) ou comme une entrée de détection d'événements, sensible aux fronts et pouvant générer une interruption.

- **Broche STRA :**

Lorsque le microcontrôleur 68HC11 est utilisé en mode EXPANDED, la broche STRA est le signal AS (Adresse Strobe) qui est utilisé pour démultiplexer les adresses basses des données sur le port C. Lorsque le 68HC11 est en mode SINGLE, cette broche est utilisée comme une entrée permettant la synchronisation des échanges sur le port C ou comme une entrée de détection d'événements, sensible aux fronts et pouvant générer une interruption.

- **Broches MODA et MODB :**

Ces broches sont utilisées lors de l'initialisation du microcontrôleur pour sélectionner le mode de fonctionnement de celui-ci. (Tab.1)

MODB	MODA	Mode sélectionné
1	0	Circuit seul Single chip (Mode 0)
1	1	Etendu Expanded multiplexed (Mode 1)
0	0	Special bootstrap
0	1	Special test

TabII.2 : Les modes de fonctionnement

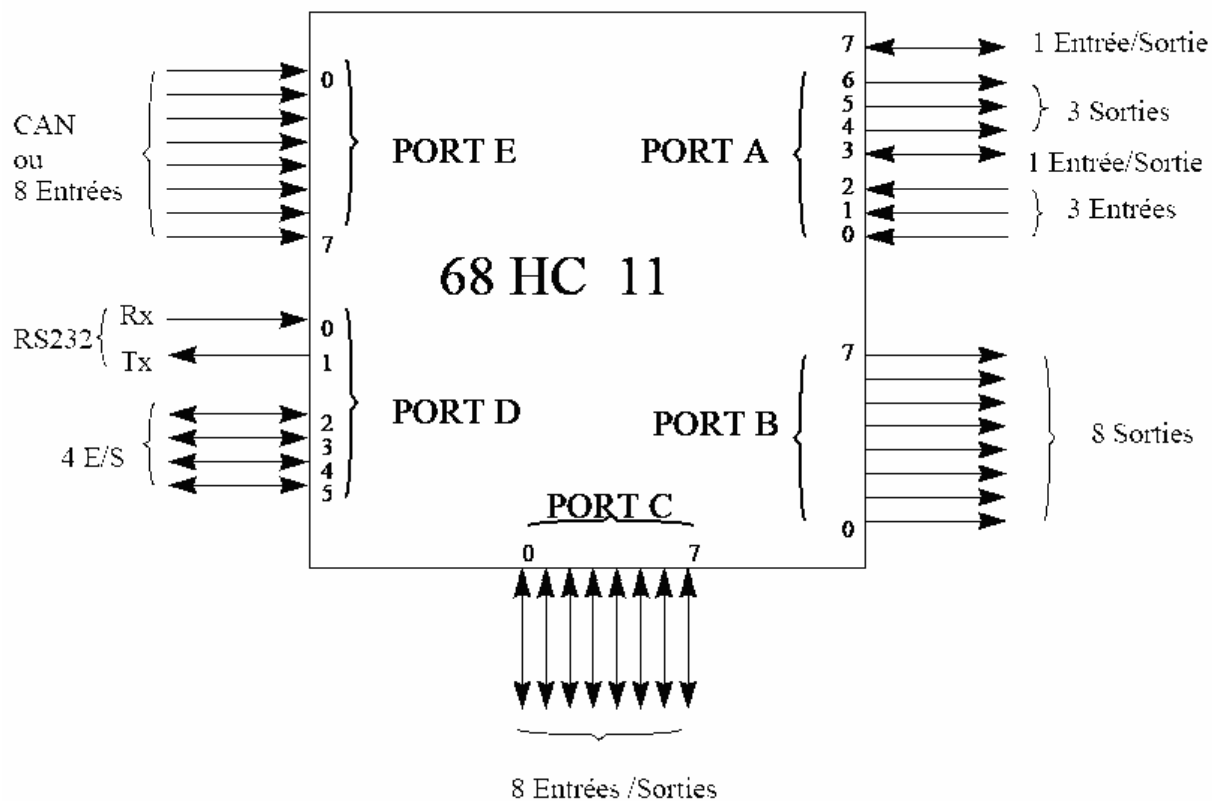
- **Les broches D'alimentation VDD, VSS :**

Les branches d'alimentation sont repérées par VDD pour le +5v et par VSS pour la masse. La tension d'alimentation normale est de 5v à 10% près. Cette tension est aussi utilisée par défaut pour alimenter les circuits du convertisseur analogique numérique. Le 68HC11 est un circuit CMOS dont les transitions internes sont très rapides, un bon découplage de son alimentation est nécessaire. Ce découplage est obtenu en plaçant un condensateur de 11µF entre VDD et VSS. Ce condensateur doit être de bon qualité en haut fréquence et placé le plus près possible de la ligne d'alimentation. Pour améliorer ce découpage il est conseillé de placer en parallèle sur le condensateur chimique de 1µF, un condensateur céramique de 0.01µF.

- **Les broches IRQ, XIRQ :**

Ce sont des entrées d'interruption externe. IRQ est une entrée d'interruption masquable alors que XIRQ est non masquable, une fois passés la phase d'initialisation du circuit. Les circuits générant un niveau bas sur ces entrées doivent être conçus de telle façon que le niveau soit maintenu jusqu'à ce que le 68HC11 leur ait effectivement signifié la prise en compte de l'interruption, sauf éventuelle pour l'entrée IRQ qu'il est possible de programmer pour être sensible à un front descendant. Chacune de ces entrées doit être ramenée à VDD par une résistance de 4.7KΩs

II*3*2*2* Ports du 68HC11 :



FigII.3: 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 (I2C) ou ASYN (RS232).

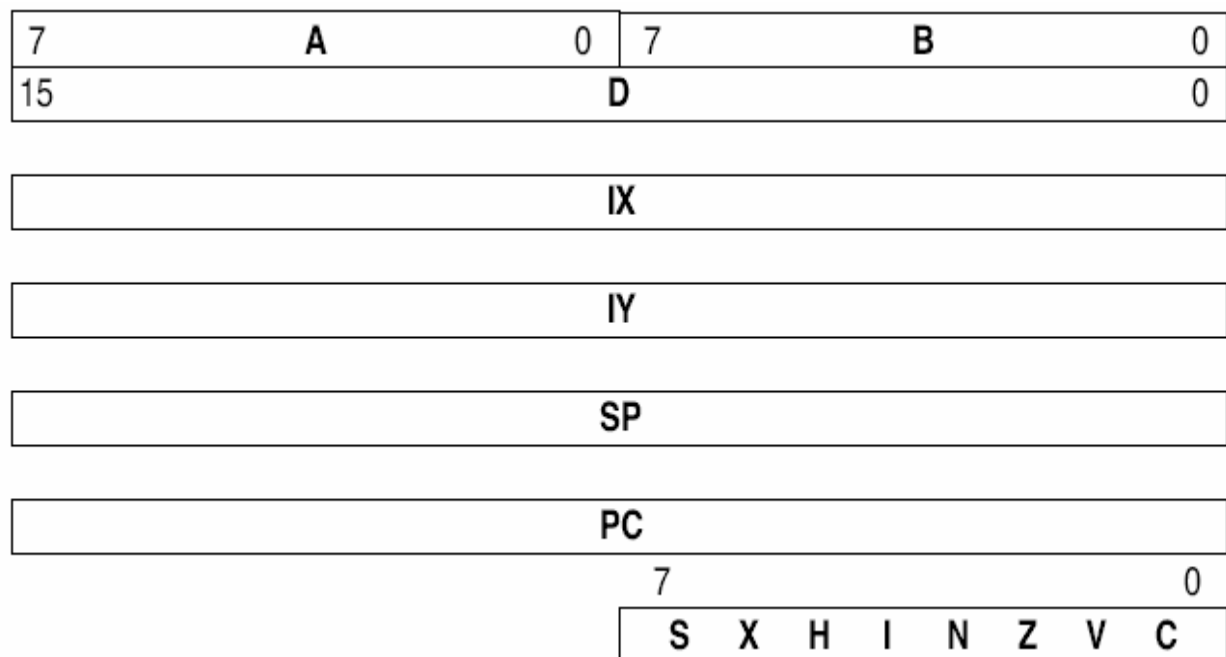
PORT E : 8 Entrées. Partagé avec les entrées du Convertisseur Analogique Numérique.

II*3*3* Les ressources du 68HC11 :

II*3*3*1* L'unité centrale (le noyau) :

Son unité centrale, coeur du circuit, est composé de 6 registres :

Les définitions registre :



FigII.4: Les registres

- Registers 8 Bits : A, B, CCR

. **A** et **B** : Ces 2 registres servent pratiquement à tout faire au format 8 Bits.

Leurs juxtapositions donnent le registre D (16 bits), de la manière suivante :

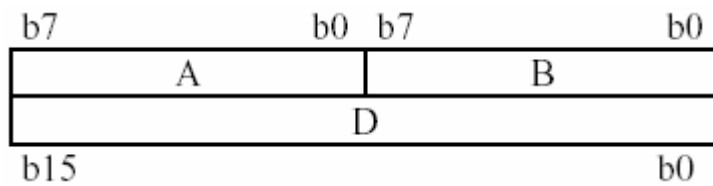


Fig.II.5 Registre A et B

· **CCR** : le registre d'état



Fig.II.6: registre CCR

Le registre CCR est positionné par l'**instruction en cours, à la fin** de son traitement par le processeur. Bien qu'on puisse regrouper différentes instructions entre elles de manière à connaître leur résultat vis à vis de ce registre, chaque instruction va influencer sur le CCR d'une manière qui lui est propre. Pour lever le moindre doute, le seul recours est le jeu d'instructions fourni par le constructeur qui fait état des bits suite à une instruction donnée.

Signification des différents bits :

C (Carry : Retenue) : est mis à 1 lorsque le résultat de l'instruction génère une retenue. Le terme retenue doit être pris au sens large du terme, elle concerne les instructions qui se rapportent à :

- l'arithmétique
- les décalages / rotations
- les comparaisons
- les calculs de compléments
- etc.

V (oVerflow = débordement) : positionné à 1 lorsque l'opération arithmétique a généré un débordement de l'accumulateur utilisé.

Z (Zéro) : passe à 1 lorsque le résultat de l'instruction est nul.

N (Négatif) : positionné à 1 si le résultat de la dernière opération arithmétique réalisée est négative, c'est à dire lorsque le bit de poids fort vaut 1 (cas des nombres signés).

H (Half Carry = demi retenue) : utilisé lors des calculs DCB (Décimal Codé Binaire).

I (Interrupt mask = masque d'interruption) : ce bit inhibe les interruptions lorsqu'il vaut 1.

X (XIRQ interrupt mask = masque d'interruption XIRQ) : de la même manière que le bit I, X inhibe l'interruption XIRQ lorsqu'il vaut 1.

S (Stop) : positionné à 1, l'exécution de l'instruction STOP est remplacée par un NOP.

Remarque : l'instruction STOP fait passer le processeur en mode « endormi ». Cela signifie que l'horloge interne est arrêtée. Lorsque le processeur est réveillé, cette horloge sera valide après un certains nombres de cycles dus à un état transitoire de mise en oscillation du quartz (Ce qui peut être à proscrire dans certaines applications de type temps réel).

- Registres 16 bits : D, X, Y, SP, PC.

· **D** : Juxtaposition des registres A et B (8 bits).

· **X** : Voici la version du « registre à tout faire » au format 16 bits.

Contrairement à ses confrères 8 bits (A et B), il assure aussi un rôle fondamental d'**index**.

· **Y** : Idem au registre X.

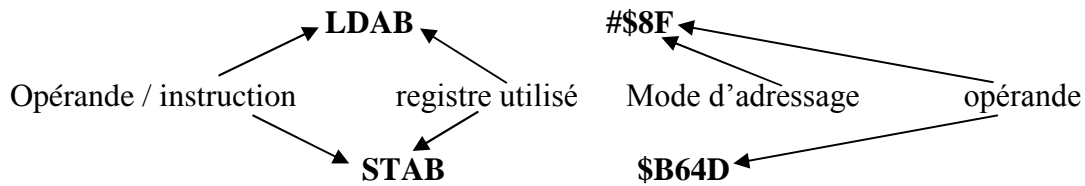
Cependant, les instructions concernant ce registre sont codées avec un octet supplémentaire par rapport au registre X, et donc, prennent un cycle machine en plus lors de leur exécution.

· **SP** (Stack Pointer = pointeur de pile) : indique en permanence la **prochaine adresse de l'octet libre** de la pile.

· **PC (Program Counter = Compteur ordinal)** : ce registre 16 bits contient en permanence l'adresse de la prochaine instruction à exécuter.

II*3*3*2* Mode d'adressage :

L'unité centrale du MC68HC11 utilise six modes d'adressage : immédiat, direct, étendu, indexée, relatif et inhérent. Ces modes d'adressages permettent d'effectuer des opérations avec la mémoire sur un ou deux octets suivant les cas mais aussi, pour un certain nombre d'instructions sur un ou plusieurs bits. Le CPU exécute à tout instant une instruction associée à un opérande. Figure (II.7)



FigII.7: une instruction associée à un opérateur.

Le mode d'adressage c'est le moyen au CPU pour trouver le lieu exact où se trouve l'opération. Les divers types de mode d'adressages sont les suivants :

- Immédiat :

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.

- Étendu :

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 d'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.

- Inhérent :

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.

II*3*4* Configuration et modes de fonctionnements :

Explication sur les différents modes de démarrage du 68HC11:

Le **68HC11** possède 2 broches **MODA** et **MODB** qui lui permettent de choisir sa configuration de démarrage lors du RESET:

MODE A	MODE B	MODE
1	0	MONO CHIP
1	1	ETENDU
0	0	BOOTSTRAP
0	1	TEST

TabII.3: configuration de démarrage lors du RESET du MODE A ET MODE B.

- **Le Mode Monochip** n'est possible que si le 68HC11 possède sa propre ROM (ou UVPRM ou EEPROM) interne. L'EEPROM de 512 octets à l'adresse \$B600 n'est pas considéré comme étant une ROM de programme dans ce mode.
- **Le Mode étendu** ne se fait que si vous utilisez le 68HC11 avec de la ROM externe. Pour palier son manque de ports utilisés pour adresser les mémoires, on utilise un 68HC24.
- **Le Mode Test** est utilisé par le fondeur pour vérifier l'intégrité de la puce. Sans intérêt pour nous.
- **Le mode bootstrap** permet à l'utilisateur de charger un programme à partir de liaison série.

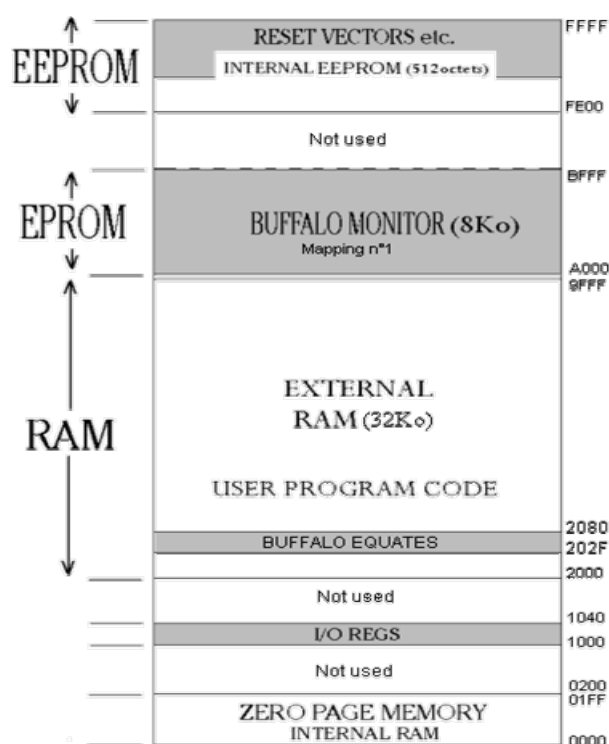
II*3*5* La mémoire :

Suivant les versions, la mémoire programme peut être interne (mode étendu). Le choix entre les deux modes, s'effectue à la mise sous tension en agissant sur les niveaux des deux broches du circuit de contrôle de mode. La mémoire interne peut être :

- De type mémoire morte à lecture seule (ROM), programmé par le constructeur.
- De type programmable par l'utilisateur et effaçable électriquement (EEPROM).
- De type programmable par l'utilisateur, effaçable par ultraviolet (UVPRM).

La mémoire vive (RAM) a une capacité de 1024 octets pour le 68HC11F1, elle est réservée à la sauvegarde de données et à la pile.

La figure (II.8) montre la structure de la mémoire pour la version 68HC11F1. La mémoire EEPROM de données a une capacité de 512 octets. Elle peut être lue ou effacée par le programme interne. Une pompe de charge interne permet de produire la tension de programmation qui est supérieure à 5V. Cette mémoire est très utile pour sauvegarder des données de programmation entre deux arrêts du microcontrôleur.



FigII.8 La mémoire

II*3*6* Reset :

L'opération RESET permet le démarrage d'un programme qui est implanté dans la mémoire. Elle consiste à placer un niveau logique 0 sur la connexion appelée RESET du boîtier du microcontrôleur.

Une fois cette tension est détectée par l'unité centrale une procédure d'initialisation est déclenchée. Celle-ci va aller chercher dans les cases mémoires \$FFFE \$FFFF, l'adresse de départ du logiciel et va la placer dans le compteur de programme.

Les sources de RESET :

Dans MC68HC11, il y a sources possibles de RESET. L'unité centrale permet de détecter des défauts de fonctionnement et de produire une information de reset sur la broche correspondante, pour faire une distinction entre ces sources, il y a plusieurs vecteurs de reset (TabII.4)

Source de reset	Vecteurs (mode normal)	Vecteur (mode test ou bootstrap)
POR ou Reset	\$FFFE, \$FFFF	\$BFFE, \$BFFF
Défaut d'horloge	\$FFFC, \$FFFD	\$BFFC, \$BFFD
COP watchdog	\$FFFA, \$FFFB	\$BFFC, \$BFFD

TabII.4 : les sources de reset

A* POR (Power On Reset) : à la mise sous tension

Ce type de Reset est destiné à initialiser le système à la mise sous tension. Quand VDD est appliquée au microcontrôleur, le POR déclenche une séquence de remise à zéro, la broche RESET est maintenue à l'état bas pendant 4064 cycles d'horloge.

B* RESET extérieur :

Une opération de remise à zéro peut être lancée en appliquant un niveau bas extérieur sur la broche RESET. Le résultat de cette séquence est la même que pour un reset interne. Pour que la demande de reset puisse être prise en compte, il est nécessaire que la broche RESET reste à l'état bas pendant au moins 4 cycles d'horloge.

C* surveillance de fonctionnement (chien de garde) le COP (Computer Operating Properly) :

Un compteur interne est décrémenté en permanence. Quand le logiciel fonctionne normalement, il recharge ce compteur avec un délai programmé bien précis l'empêchant ainsi d'arriver à 0.

Si le logiciel est perturbé, le rechargement n'a plus lieu en temps voulu et un niveau 0 est placé pendant 4 cycles d'horloge sur la connexion RESET.

Le bit NOCOP du registre CONFIG permet la mise en service du COP.

Le délai de rechargement est programmable avec les bits CR0 et CR1 du registre CPTION (adresse : \$1039).

D* surveillance d'horloge : Clock Moniteur (CM)

La surveillance d'horloge CM déclenche un RESET si la période du signal d'horloge devient supérieure à une déterminée par la constante de temps d'un circuit RC interne (5μs).

Pratiquement ceci interdit de faire descendre la fréquence d'horloge en dessous de 200KHz. Cette surveillance complète le COP car celui-ci ne peut détecter d'erreur d'horloge. Le bit CME du registre OPTION permet la mise en service de la surveillance d'horloge.

II*3*7* Les interruptions :

Une interruption est une procédure qui va interrompre le déroulement normal du programme

pour effectuer un traitement prioritaire.

Certaines d'entre elles sont masquables et d'autres non.

Une interruption peut avoir plusieurs origines :

- Matérielle :

- Reset
- XIRQ
- IRQ

Interne : générée par les périphériques internes au 68 HC 11.

Logiciel : instruction SWI

Voici comment se comporte le 68 HC 11 lors d'une interruption :

L'instruction en cours est complètement terminée.



Sauvegarde des registres de l' UC (Unité Centrale) dans la pile.



Le masque d'interruption passe à 1.



Chargement du vecteur dans le PC.



Traitement du sous programme d'interruption.



Le traitement se termine par l'instruction RTI.

Les Interruptions sont classées selon une hiérarchie, les voici rangées de la plus prioritaire à la moins prioritaire :

*** Dans les non masquable :**

- POR, Reset externe.
- CM.
- COP.
- XIRQ.
- Code illégal.
- SWI.

*** Dans les masquables :**

- Débordement Timer.

Débordement accumulateur d'impulsions.

- Détection de front actif de l'accumulateur d'impulsion.
- Fin de la transmission synchrone.
- Fin de la transmission asynchrone.
- IRQ.
- Horloge Temps Réel (RTI)
- IC1 \
- IC2 } => Entrées de capture du TIMER
- IC3 /
- OC1 \
- OC2 \
- OC3 } => Sorties de comparaisons du TIMER
- OC4 /

II*3*8* L'horloge :

Un circuit d'horloge interne permet d'obtenir les signaux de cadencement du microcontrôleur. Ce circuit est piloté par un quartz externe dont la fréquence peut atteindre 24Mhz. Les signaux de cadencement interne ont une fréquence égale à celle du quartz divisée par quatre 4.

II*3*9* Le convertisseur analogique numérique :

Le convertisseur A/N de la famille 68HC11 travaille selon le principe de l'approximation successive, à une résolution de 8 bits et dispose d'un circuit d'échantillonnage et de maintien (sample and hold). Grâce au multiplexeur analogique à l'entrée, il est possible de connecter jusqu'à 8 signaux d'entrée. Les entrées analogiques inutilisées peuvent servir d'entrées numériques. Le temps de conversion pour un signal d'entrée à une fréquence d'horloge de 8 MHz, temps d'échantillonnage et de maintien inclus, est de 16 μ s (soit 32 cycles machine). Le SFR ADCTL permet de régler divers modes opératoires. C'est ainsi, par exemple, que l'on peut choisir entre une conversion unique ou continue avec un ou quatre canaux d'entrée. Il faut prévoir une tension de référence externe pour le convertisseur A/N.

II*3*10* Le timer :

Le Timer du MC68HC11 est un bloc du microcontrôleur spécialisé dans la mesure et la génération de temporisation. Son usage n'est pas très simple lorsque l'on débute, mais grâce à des exemples, il est possible de comprendre son fonctionnement.

Pour le décrire, il est possible d'isoler 4 blocs qui sont :

- 1) Le Timer à usage général
- 2) Le générateur d'interruptions temps réel
- 3) Le Chien de garde (COP)
- 4) L'accumulateur d'impulsions

II*3*11* L'environnement de développement du microcontrôleur 68HC11 :

Pour le développement d'une application à base du microcontrôleur 68HC11, plusieurs étapes sont nécessaires pour l'accomplissement de la tâche. En effet, on doit d'abord définir un cahier de charges qui doit comporter :

- Les algorithmes pour le logiciel qu'il faut implanter dans la mémoire du μC ;
- Les schémas électriques.

On écrit par la suite le programme sous forme d'instructions assembleur du MOTOROLA, avec un éditeur de texte simple comme l'éditeur DOS ou un autre éditeur. Après avoir vérifié le bon déroulement du programme, on procédera ensuite à l'assemblage en utilisant un assembleur (A11) qui nous donne un fichier objet ou un fichier S19. Après l'assemblage il faut utiliser un logiciel qui nous permet de télécharger le programme dans le μC .

Pour l'édition du programme, on a utilisé le simulateur MOTO6811 (Fig.II.9) Qui permet de simuler le fonctionnement du microcontrôleur 68HC11F1 de Motorola (TM) et donc de plusieurs autres microprocesseurs de la même famille. Il permet aussi de mettre aux points des programmes simples faisant appel aux ressources de ce microcontrôleur.

Caractéristiques essentielles du moto6811 :

- Utilisation de l'environnement Windows 95.
- Visualisation simultanée du contenu de tous les registres internes et des cases mémoires.

Exécution complète ou pas-a-pas.

- Possibilité de simulation des interruptions.
- Editeur intégré.
- Assemblage au format S19.
- Modification du contenu des registres et des cases mémoires en cours d'exécution.
- Simulation de toutes les fonctionnalités des interfaces parallèles.
- Simulation du convertisseur analogique numérique.

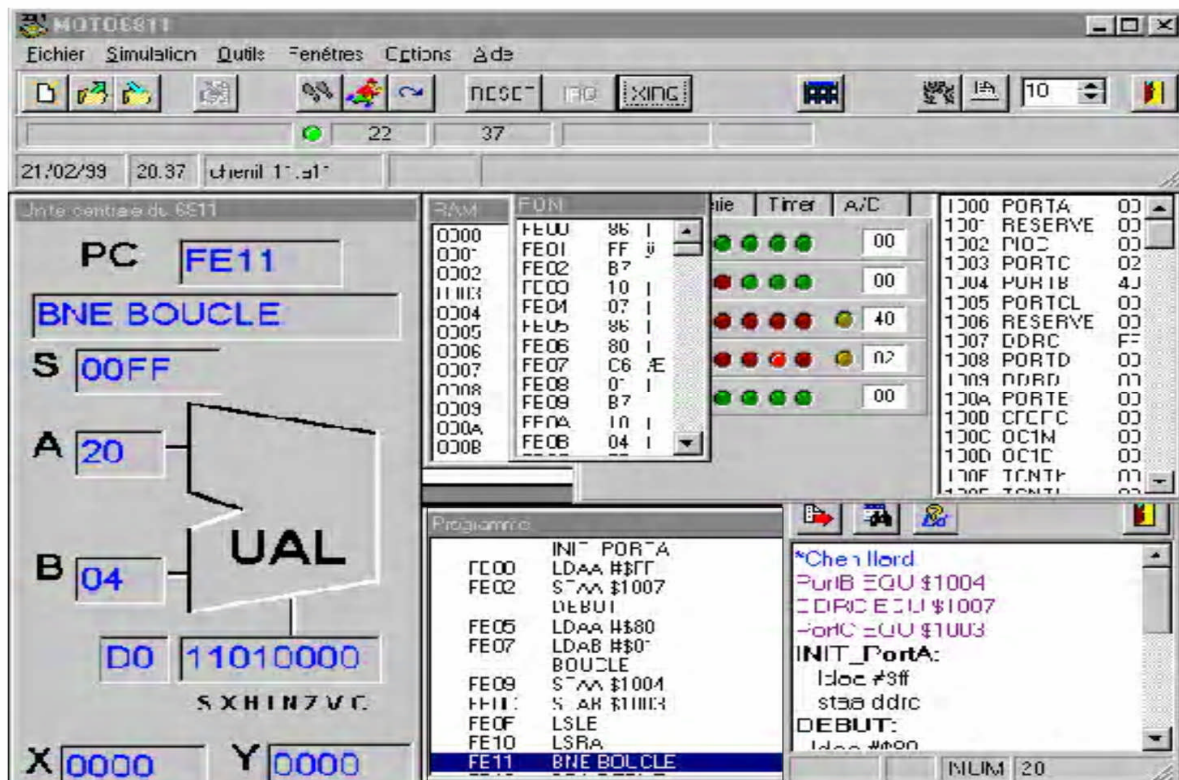


Fig.II.9: simulateur MOTO6811

Après le développement du logiciel avec MOTO6811 on fait appel à un logiciel de téléchargement de ce programme dans le μ C, qui est le PCBUG11.

Le fonctionnement de PCBUG11 :

Le logiciel PCBUG11 (fig.II.10) fonctionne uniquement en mode bootstrap. En effet, en mode bootstrap, il apparaît une « boot ROM » en \$BF40-\$BFFF, zone de mémoire morte à laquelle le processeur saute après le Reset. Cette mémoire de boot, présente dans la cartographie uniquement en mode bootstrap et spécial test, contient une petite routine qui

autorise le téléchargement des octets depuis le port série vers la RAM à partir de l'adresse \$0000.

Le logiciel PCBUG11 envoie un petit programme de 192 octets environ qui est appelé TALKER. Comme son nom l'indique, ce petit programme va permettre de communiquer avec l'ordinateur hôte et ainsi d'exécuter les commandes de PCBUG.

Le logiciel PCBUG11 se présente de la manière suivante :

La zone supérieure est la zone de dialogue où s'affiche le résultat des opérations lancées par l'utilisateur. La zone centrale au milieu montre l'état des registres du 68HC11 et à droite le type de processeur ainsi que le mode de fonctionnement de PCBUG11 (Running, Stopped, Trace). La partie inférieure est réservée à l'utilisateur pour rentrer les commandes de PCBUG11.

```

PCBUG342 Axx
Total bytes loaded: $0246
Total bytes written: $0246
Total bytes loaded: $0270
Total bytes written: $002A
C000 4F      > CLRA
C001 CEF000  > LDX  #$F000
C004 2003    > BRA  $C009
C006 A700    > STAA $00,X
C008 00      > INX
C009 8CF002  > CPX  #$F002
C00C 26F8    > BNE  $C006
C00E 8E00FF  > LDS  #$00FF
C011 BDC0A1  > JSR  >$C0A1
C014 20FE    > BRA  $C014
C016 3C      > PSHX
C017 3C      > PSHX

PC  ACCA  ACCB  X    Y    CCR (SXHNZUC) SP  MCU: 68HC11A8
$0000 $40  $2C  $1000 $0100 $40  %1..... $00EB RTS Level :ON
State: STOPPED
Base : HEX
User RST $XXXX
User SWI $XXXX
User XIRQ $XXXX

restart
loads testio2
asm c000
»_

```

Fig.II.10: Le logiciel PCBUG11

Chargement d'un programme avec PCBUG11 :

Lorsqu'on désire charger un programme en EEPROM on doit procéder comme suit :

- Effectuer un RESET sur la carte cible en mode bootstrap.
- Exécuter PCBUG11

Et puis on écrit les instructions suivantes :

- **MM \$1035 \$10** pour autoriser l'écriture sur la mémoire EEPROM.
- **EEPROM \$FE00 \$FFFF** : pour définir la zone où se trouve l'EEPROM du μ C, comme pour le 68HC11F1, l'EEPROM situé de l'adresse \$FE00 jusqu'à \$FFFF.
- taper **EEPROM ERASE BULK** pour effacer l'EEPROM.
- taper ensuite **LOADS [Nom de fichier]** (nom du fichier compilé), pour télécharger le fichier S19 dans l'EEPROM.

Et ensuite si on veut exécuter le programme on peut procéder de deux façons :

1) Exécution ONLINE c-à-d le test de programme se fait à partir du logiciel PCBUG11. On écrit comme suite :

G [l'adresse de début], et le programme commence à s'exécuter, puis

S pour terminer l'exécution

2) Exécution OFFLINE. Dans ce mode on quitte le PCBUG11, et on met la carte en mode circuit seul, suivant le tableau II.3 (dans le chapitre II), avec un reset.

II*3*12* Conclusion :

En conclusion on peut dire que le microcontrôleur peut jouer le rôle d'un automate programmable ceci est particulièrement justifié par les éléments suivants :

1. facilité de communication avec le monde extérieur (set ports d'entrées/sorties analogiques,...etc.)
2. facilité de communication avec le PC, ce qui permet sa programmation via le port série.

On retient en plus de ces caractéristiques. Ses avantages de disponibilité de faible coût.

Chapitre : III

Description de la carte automate

III*1* Introduction :

Dans le chapitre précédent nous avons analysé en détail le fonctionnement des microcontrôleurs de la famille HC11. Dans ce chapitre nous allons examiner les moyens à mettre en œuvre pour développer une application à base d'un MC.

Un automate programmable industriel ou API est un appareil électrique capable de piloter un système. Notre but est de réaliser un automate programmable à bas d'un microcontrôleur, où le microcontrôleur est le cerveau de automate à travers la CPU, qui communique avec le monde extérieur par le biais des interfaces d'entrées / sorties.

Le schéma synoptique d'un automate (fig.III.1)

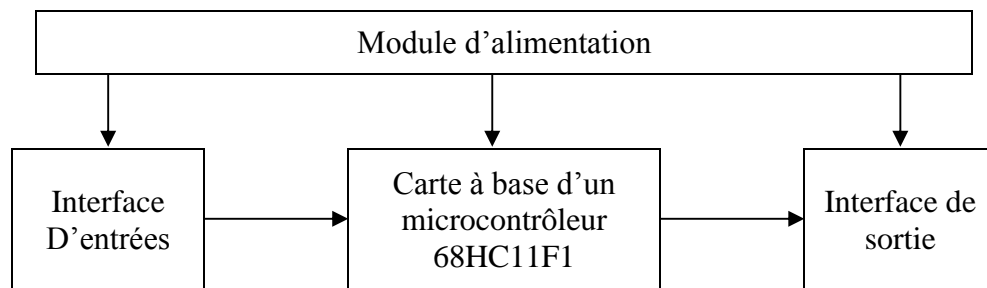


Fig.III.1 : Le schéma synoptique d'automate

III*2* Etude de la carte à base d'un MC68HC11F1 :

Schéma électrique de la carte :

La figure.III.2 montre le schéma électrique de la carte qui se compose de plusieurs circuits externes.

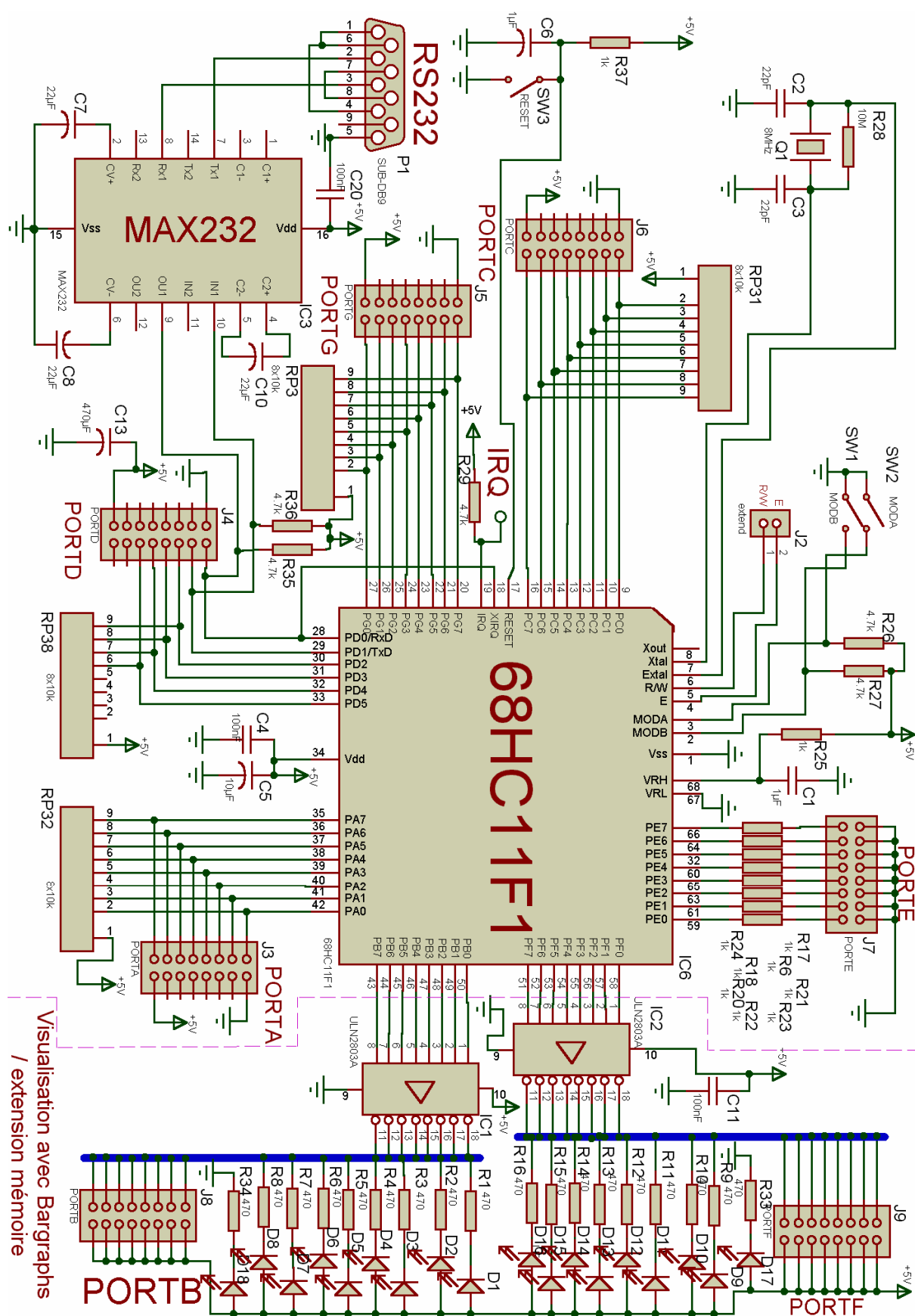


Fig.III.2 : schéma électrique (électronique pratique n°225)

III*2*1* Circuit d'alimentation :

Les broches d'alimentation sont repérées par VDD pour le +5v et par VSS pour la masse. La tension d'alimentation nominale est de 5V à 10% près.

Cette tension est utilisée par défaut pour alimenter les circuits du convertisseur analogique numérique.

Cette tension doit être parfaitement découplée pour éviter tout problème. La meilleure solution est de faire appel à un condensateur chimique de 1 μ F en parallèle avec un condensateur de 0.01 μ F.

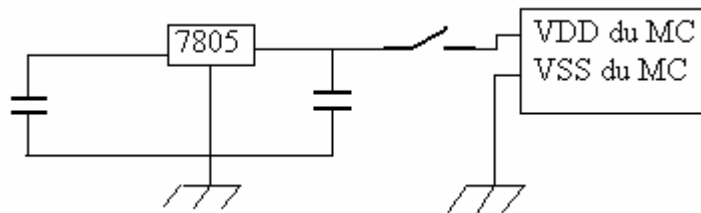


Fig.III.3 : Circuit d'alimentation

III*2*2* Circuit de Reset :

On utilise par fois un bouton poussoir pour réaliser un RESET. Il est fortement conseillé de piloter l'entrée RESET du MC68HC11 par un circuit détectant automatiquement toute baisse de VDD en dessous de la valeur limite autorisé (le circuit TL7705) et générant alors un RESET dans ce cas, dans notre cas nous avons utilisé le circuit de la figure III.4.



Fig.III.4 : Circuit de reset

III*2*3* Circuit d'horloge :

La figure.III.5, montre la connexion d'un quartz aux pattes EXTALL et XTAL du microcontrôleur. Les branches XTAL et EXTAL sont reliées à un quartz pour générer l'horloge interne de l'oscillateur, il doit être connecté par ses cotés avec des capacités qui sont reliées à la masse.

La valeur de résistance dépend de la fréquence du quartz. Elle varie habituellement de $1\text{M}\Omega$ à $20\text{M}\Omega$.

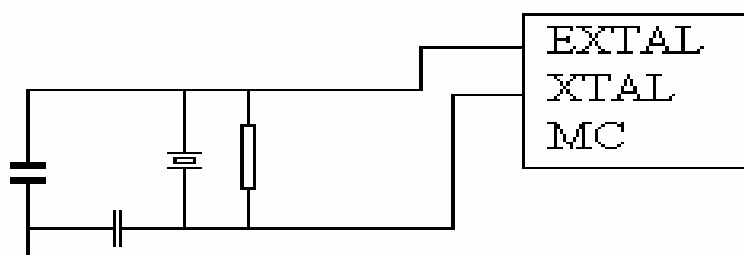


Fig.III.5 : Circuit d'horloge

III*2*4* Le Max 232 :

Le MC68HC11 dispose d'une interface série synchrone ou SCI lui permettant ainsi de communiquer très facilement avec nombreux équipement informatiques classique : micro-ordinateurs modems.. etc. cette interface utilise la norme RS 232 (+12 , -12) pour communiquer avec le PC, cette communication est assurée par le MAX 232, il fait la conversion des niveaux TTL(0 , 5v) fournis ou reçus par le MC 68HC11 en niveaux RS 232 (fig.III.6)

III*3* Etude des interfaces d'entrées/sorties :

Lors de réalisation des applications, ce pose le problème de la définition de la forme des signaux qui transportent les informations et le choix des modules d'entrées/sorties.

La forme des signaux d'entrée / sorties peut être très différente suivant les informations reçues ou fournies par les périphériques. Les signaux les plus courants sont :

- Signal digital,
- Signal analogique,
- Impulsions incrémentales,
- Signaux binaires parallèles,
- Signaux binaires séries.

III*3*1* Etude des interfaces d'entrées :

On utilise les entrées pour connaître l'état des divers éléments, soit celles des entrées numériques ou celles des entrées analogiques.

III*3*1*1* Les entrées numériques :

Ces entrées sont destinées à nous donner des informations sur la présence ou l'absence de tension, qui correspond à un niveau logique haut (1 logique) ou bas (0 logique). Ces niveaux peuvent être lus par le CPU du μC .

Pour réaliser une interface d'entrée numérique on a utilisé le circuit de la figure.III.7. Pour avoir un niveau haut (1 logique), on injecte 5V sur l'entrée correspondante à travers une résistance de quelque K Ω . Cette résistance pourra limiter le courant entrant dans le circuit en présence d'un bouton poussoir comme l'indique la figure ci-dessous.

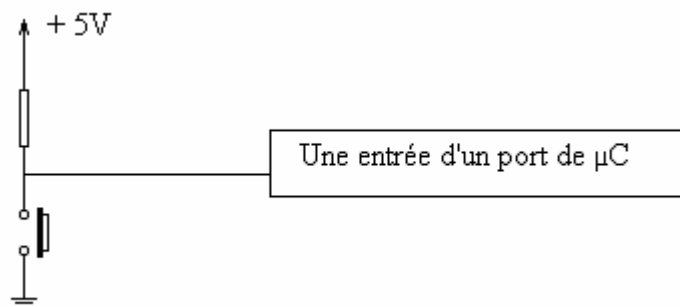


Fig.III.7 : Connexion d'un poussoir ou interrupteur

III*3*1*2* Les entrées analogiques :

Le μC dispose d'un convertisseur analogique numérique qui nous permet de convertir une tension analogique variant entre 0V et 5V en valeur numérique. Cette tension variable est obtenue grâce à un potentiomètre comme l'indique la figure.III.8.

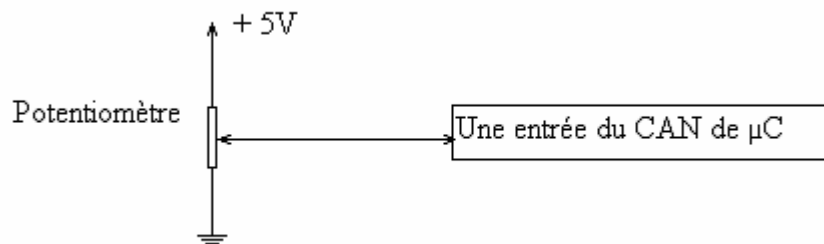


Fig.III.8 : Entrée analogique sur la CAN du μC

III*3*2* Etude des interfaces de sorties :

En général on utilise les sorties pour commander des actionneurs (moteurs et d'électrovannes ...etc.), donc tout dépend de l'application.

III*3*2*1 Les sorties numériques :

Ces sorties sont à fournir un niveau logique haut (5V) ou un niveau logique bas (0V) produisant respectivement la fermeture et l'ouverture d'un contact électrique.

Pour réaliser une interface de sortie on utilise certains éléments comme le transistor ou phototransistor, ULN2803.

Dans le cas où on utilise le transistor pour la commande, il ne faut surtout pas oublier la diode de protection montée en inverse sur la bobine comme indiquée sur la figure.III.9.

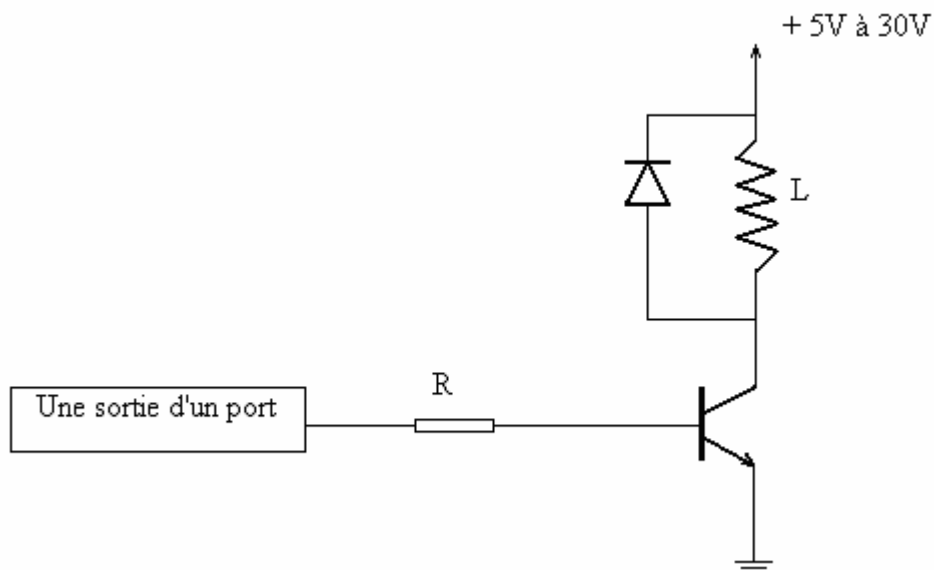


Fig.III.9 : Utilisation d'un transistor amplificateur

On peut utiliser un autre circuit qui est :

Circuit ULN2803 :

Dans le cas des ULN2803 où la diode est déjà intégrée dans le circuit. Les sorties sont directement commandées par les sorties du MC, on peut les utiliser directement mais à ce niveau la puissance est très faible. Ceci explique l'utilisation des réseaux de transistor darlington ULN2803. Un tel réseau est capable d'écouler un courant MAX de 500mA par transistor et supporte une tension MAX de 50V, ainsi nous pouvons connecter directement la majorité des relais.

La figure ci-dessous montre le câblage d'un ULN283 pour commander des relais.

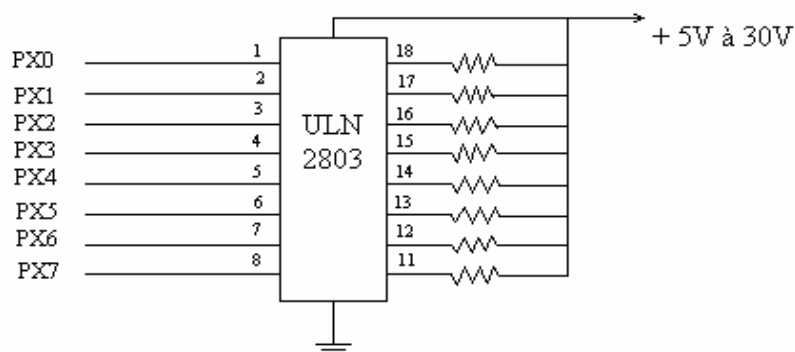


Fig.III.10 : Câblage d'un ULN2803

III*3*2*2* Les sorties analogiques :

Pour obtenir des sorties analogiques on fait appel à un convertisseur Numérique/Analogique qui convertie les signaux numériques sortants du MC.

III*4* Conclusion :

Grâce à cette carte, on peut réaliser plusieurs systèmes de commande capable d'automatiser des nombreuses applications. Le chapitre suivant montrera l'utilisation de cet automate dans notre application.

CHAPITRE IV :

Application

COMMANDE D'UN CLOSOIRE (CHAMBRE POUR ELEVAGE) :

Notre application consiste à commander un closoire (chambre pour élevage (incubateur petrsime modèle 576-œufs de poules)).

IV*1* Généralités :**Local et montage :**

Les machines seront montées de niveau sur un pavement bien endurci, lisse et de niveau, avec une portance de minimum 750 kos/m^2 . Une inclinaison légère vers le devant peut se faire pour faciliter le nettoyage.

Comme bonne isolation des parois extérieures d'un couvoir on accepte : $k \ 1$ ou mieux pour les murs extérieures et $k \ 0.7$ ou mieux pour les plafonds.

La température : dans le local est très importante. Elle doit être de $18-20$ jusqu'à 22°c maximum pour appareils par eau et de $22-25$ jusqu'à 30°c maximum pour appareils refroidis par air.

L'humidité : dans le local doit être de $55-65$.

Il faut laisser suffisamment de place devant les machines (de 2.30 M à 3 M) pour faciliter la manutention des chariots.

Il est recommandé de faire dans le pavement et devant les machines un canal d'écoulement avec grille. Ceci avantage des appareils et du local.

Une aération suffisante du local est à prévoir.

Evitez toutefois des courants d'air qui donnent directement sur les incubateurs.

Branchement sur conduite d'eau :

Le système d'humidification exige l'apport d'eau propre à 3 à 5 Kos/cm^2 . Les conduites d'arivée auront un diamètre suffisant pour ne pas gêner l'apport d'eau suffisant aux appareils.

Branchement sur gaine d'extraction :

Les appareils 576 sont équipés de 2 amenées et 4 sorties d'air. Ces sorties peuvent être branchées (indirectement) sur une gaine centrale avec ventilateur d'extraction, de sorte que l'air sortant des appareils est conduit à l'extérieur du local.

Le diamètre de cette gaine d'extraction ainsi que la capacité du ventilateur sont à calculer cas par cas.

En aucun cas faire un branchement direct ! De préférence nous contacter pour une telle installation.

IV*2* Cahier de charges :

Il s'agit de commander les différents organe d'un closoire (chambre pour élevage) qui sont:

La température, humidité, et la ventilation

Notre but consiste à réaliser les commandes de ces organes mais en utilisant la logique programmée basée sur les circuits intégrés programmés comme notre automate.

Principe de fonctionnement :

Le principe de fonctionnement de cette chambre est comme suivant :

A* commandes manuelles de la chambre :

On a commandés automatiquement les valeurs consigne de température (T_{max} , T_{min}), d'humidité (H_{max} , H_{min}) et de ventilation (V_{max} , V_{min}) par boutons

B* commande automatique de fonction :

En fonction si on a :

B*1* dépassement supérieur de température a celle des valeurs initiale (consigne) :

Notre programme faire comme premier action arrêt la résistance R et mouvementer le vérin (augmentation de la ventilation V) si pas suffisant après N minutes il doit déclencher le turbo en plus

B*2* dépassement inférieur de température a celle des valeurs initiale (consigne) :

Si on un dépassement inférieur de température notre programme travail comme suit : il doit déclencher la résistance R et mouvementer le vérin (diminué la ventilation V).

B*3* dépassement supérieur d'humidité a celle des valeurs initiale (consigne) :

Notre programme faire comme première action arrêt la résistance R et arrêt la humidité (le moteur Mh) et mouvementer le vérin (augmentation la ventilation V) si pas suffisant il doit déclencher le turbo en plus

B*3* dépassement inférieur de humidité a celle des valeurs initiale (consigne) :

Si on un dépassement inférieur de température notre programme travail comme suit : il doit déclencher le moteur de humidité (Mh) et la résistance R et mouvementer le vérin (diminué la ventilation V).

C* si on ouvre la porte notre system arrêt et si on la ferme le system va démarrer.

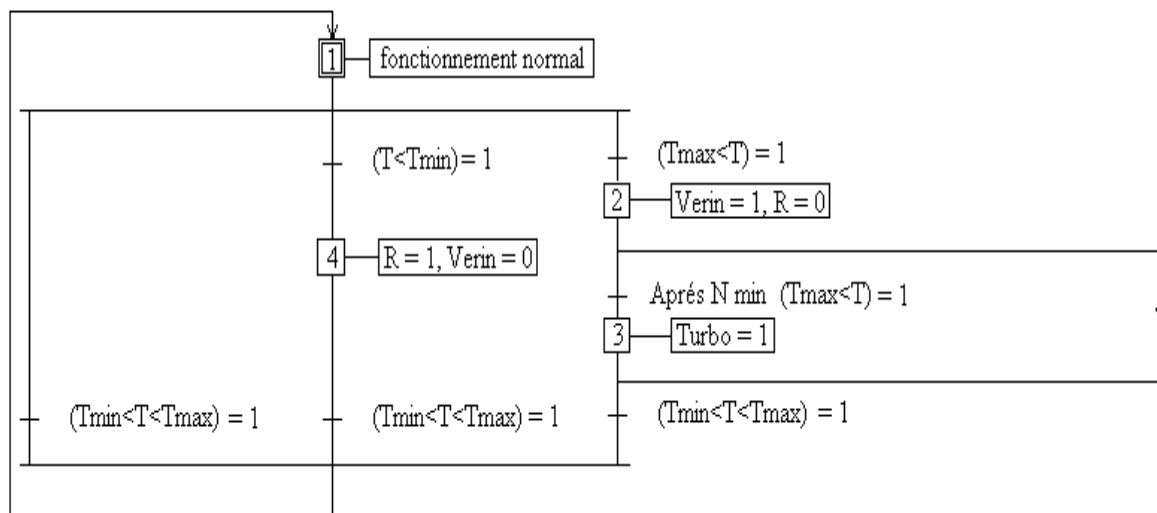
IV*4* Le grafcet et l'organigramme correspondant de chaque élément :**A* la température :**

Fig.IV.1 : grafcet de fonctionnement de température

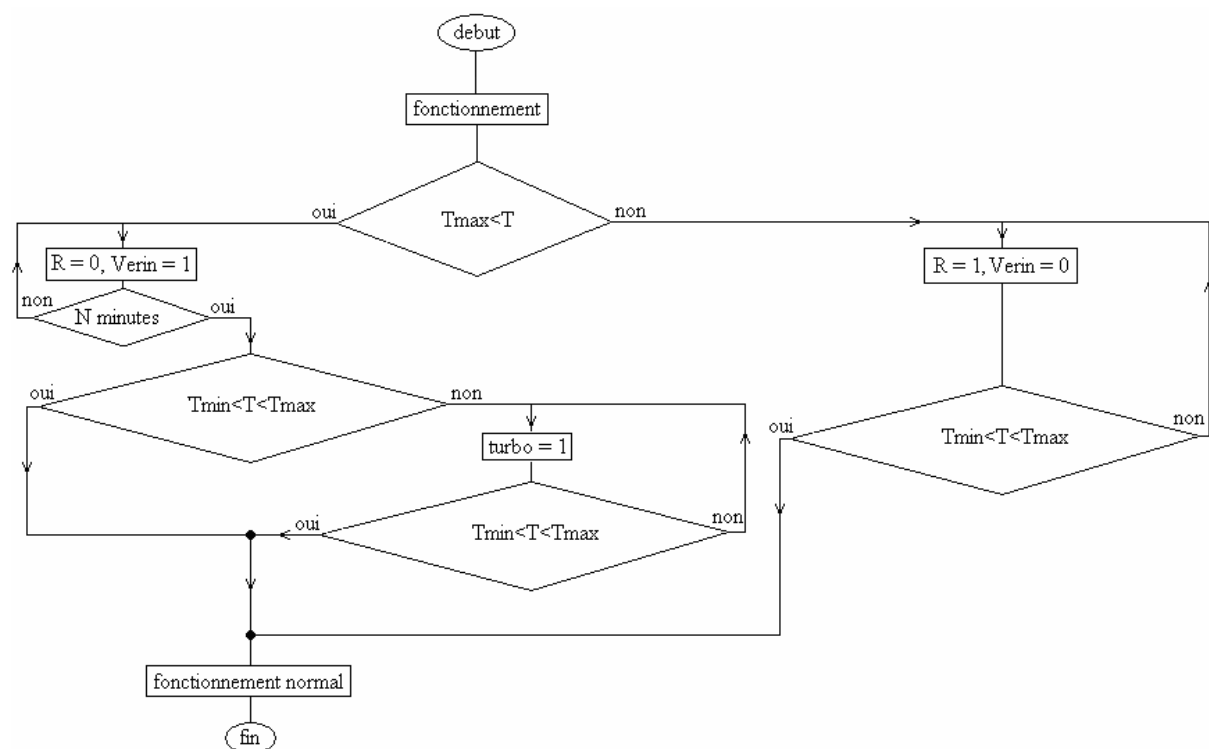


Fig.IV.2 : l'organigramme de fonctionnement du température

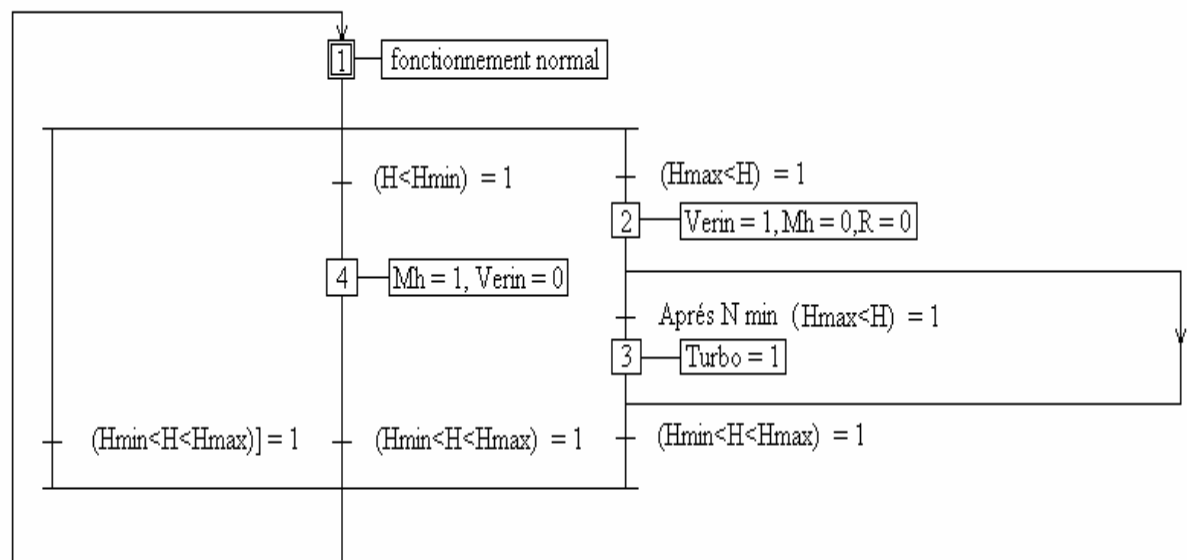
B* L'humidité :

Fig.IV.3 : grafcet de fonctionnement d'humidité

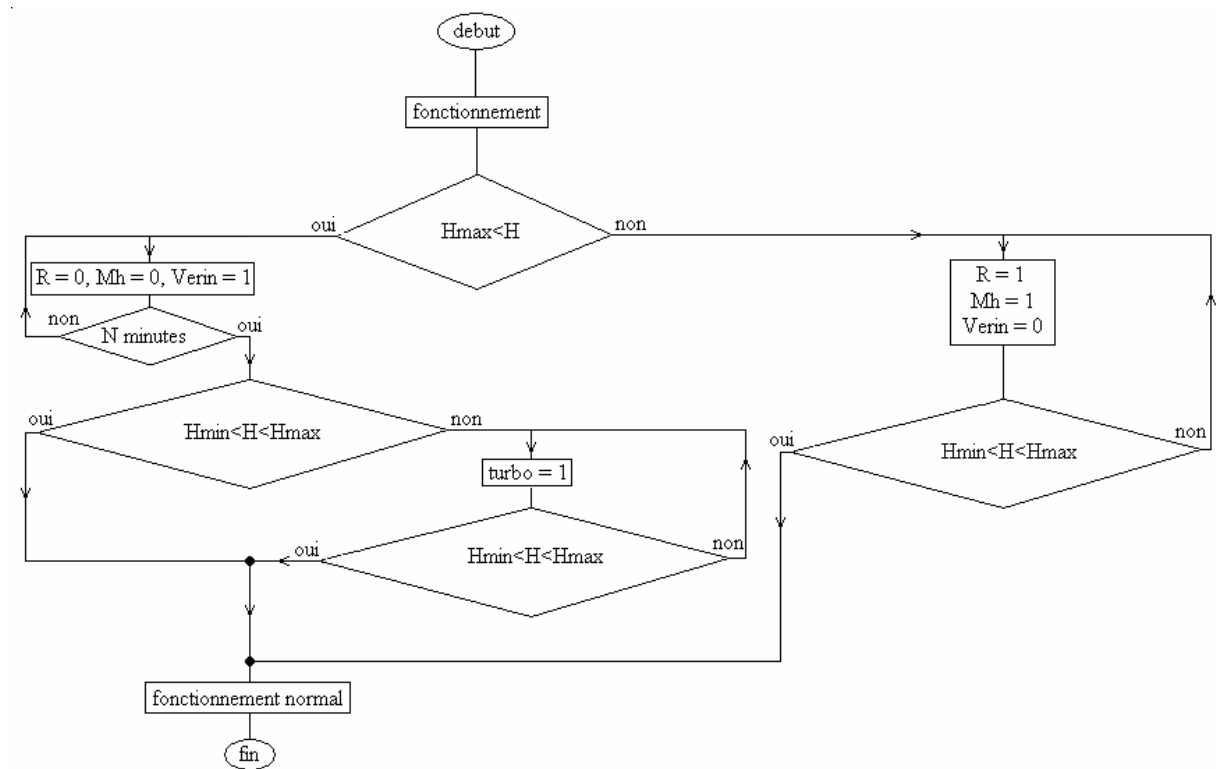


Fig.IV.4 : l'organigramme de fonctionnement d'humidité

IV*4* Le grafcet et l'organigramme total du programme :

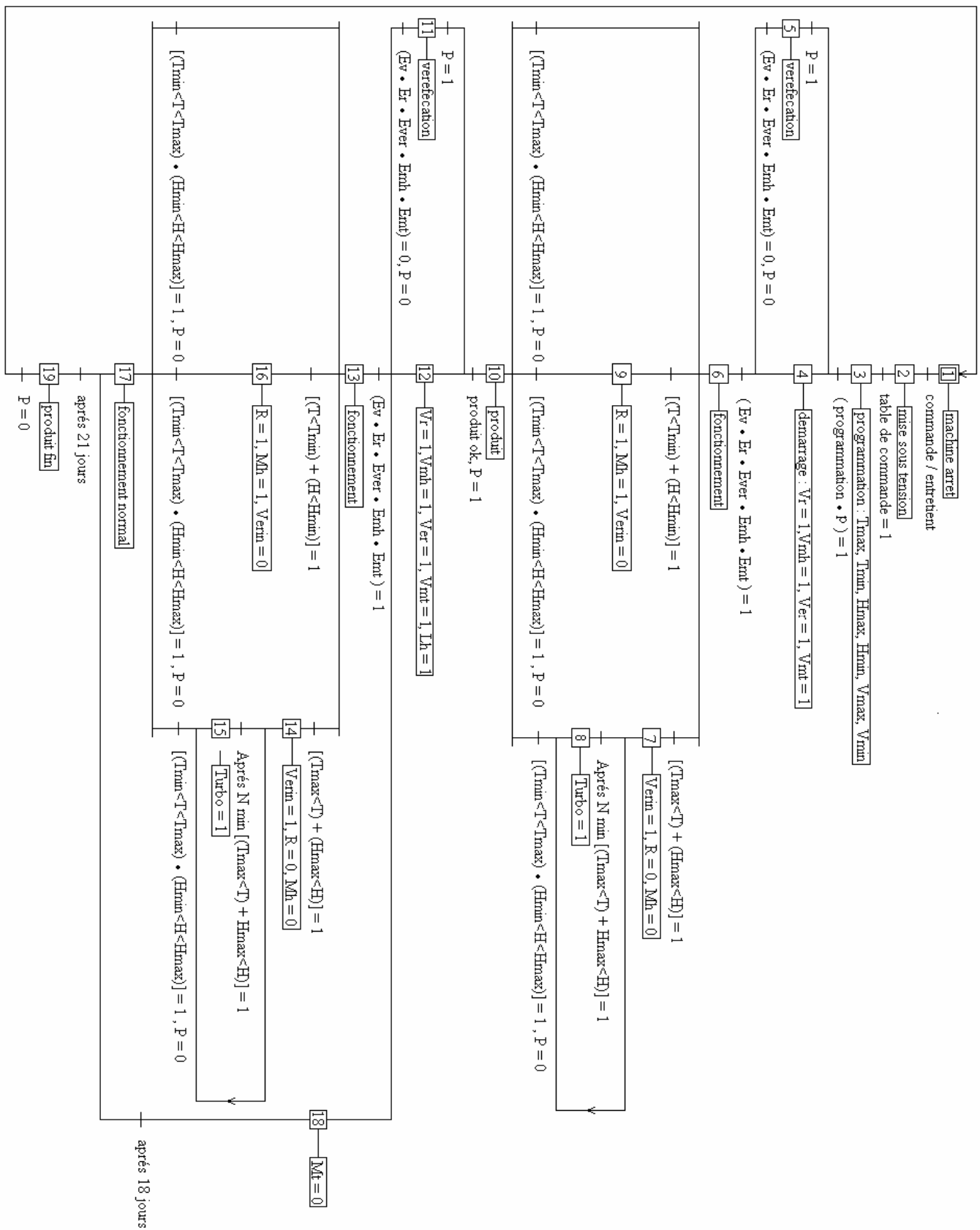


Fig.IV.5 : le grafcet de fonctionnement de chambre

IV*5*Programme générale:

```
portc equ $06
ddrc equ $07
ddra equ $01
portf equ $05
portg equ $02
ddrg equ $03
OPTION EQU $39
ADCTL EQU $30
ADR1 EQU $31
ADR2 EQU $32
portg equ $02
ddrg equ $03
porta equ $00
ddra equ $01
porte equ $0a
portb equ $04
ldx #$1000
ldaa #$f0
staa ddrg,x
int0:
brset $00,x,$01,inta
bclr portb,x,$01
bra test2
inta:
bset portb,x, $01
test2:
brset porta,x,$03,intb
bclr portb,x,$f0
bra erreur
intb:
bset portb,x,$70
```

```
brset porta,x,$03,erreur
bclr portb,x,$f0
erreur:
brset porta,x,$f3,int01
bra int0
int01:
bset portg,x,$10
ldab #$33
stab $b0
ldaa #$E5
staa $b1
ldab#$66
stab $b2
ldaa #$cc
staa $b3
ldab #$60
stab $b4
ldaa #$30
staa $b5
int1:
ldx #$1000
ldaa #$00
staa $0a
ldab #$00
stab $0b
ldaa #$80
staa OPTION,x
ldab #$b0
stab ADCTL,x
ldaa ADR1,x
ldab adr2,x
suba $b0
blt test20
```

```
subb $b2
blt test20
suba $b1
bge test21
relfin1:
decalage3:
bra int0
subb $b3
bge test21
etatfonct:
bset portf,x,$80
bra production
test20:
bset portb,x,$61
bra finall1
test21:
bset portb,x,$11
temporisation:
ldy #$05ff
réaltemp:
dey
bne réaltemp
suba $b1
bge test22
subb $b3
bge test22turbo
bset portf,x,$80
test22:
bratest22a
decalage2:
bra decalage3
test22a:
ldaa adr1,x
```

```
ldab adr2,x
suba $b1
bgetest22turbo
subb $b3
bge test22turbo
bra etatfonct
test22turbo:
bset portf,x,$40
finall1:
ldaa ADR1,x
ldab adr2,x
suba $b1
bge production
bclr portf,x,$80
bra fin
production:
brset porta,x,$f7int10
bra fin
int10:
bra int10a
decalage1:
bra decalage2
int10a:
bset portg,x,$30
brset portc,x,$01,int11
bset portg,x,$70
brset portc,x,$02,int12
brset portc,x,$04,int13
brset portc,x,$10,int14
bra fin
int11:
bset portg,x,$70
bra fin
```

```
int12:
bset portg,x,$b0
int13:
bset portc,x,$80
bra fin
int14:
bset portc,x,$c0
fin:
bra decalage1
end
```

CONCLUSION GENERALE:

L'objectif qu'on s'est fixé au départ était de concevoir et d'étude un automate programmable à base d'un microcontrôleur 68HC11.

En premier lieu nous avons fait une étude sur les automates programmables et l'un de ses langages de programmation graphiques, qui est le GRAFCET.

L'automate que nous étudions est constituée de six cartes (deux cartes mère, deux modules d'entrée, deux modules de sortie) et pour ce projet on a besoin de deux cartes à base microcontrôleur 68HC11. La première carte contient la temporisation (6 jours - 12 jours et 18 jours) la deuxième carte contient le programme général sans temporisation qui devient une sortie de la carte brset1 et comme entrée pour la carte 2.

- La carte mère conçue autour du microcontrôleur 68HC11, possède une RAM de 1 Koctets, une EPROM de 8 Koctets et une EEPROM de 512 octets, 64 lignes d'entrées/sorties qui lui permettent de dialoguer avec le module extérieur.
- Le module d'entrée contient 8 entrées numériques permettant d'avoir des niveaux logiques 1 et 0, et une entrée analogique fournissant des tensions qui varient entre 0 et 5V.
- Le module de sortie comporte 8 entrées provenant de l'automate, associées à une carte qui peut piloter 8 relais, par le biais du circuit ULN 2803.

Le schéma électrique de la carte automate, et leur circuit imprimé ont été trouvés dans l'électronique pratique N°225.

Nous avons étudié l'automate sur cette application, ce que nous permet d'appliquer cet automate sur cette application et d'autres, et surtout dans le domaine industriel.

Enfin nous souhaitons compléter à l'avenir dans un temps plus large et au moins de petits moyens. Nous espérons que ce projet qui nous a donné l'occasion d'élargir nos horizons de connaissance, sera complété à l'avenir par le développement d'un logiciel qui permet d'avoir un fichier assembleur MOTOROLA, associé avec un éditeur GRAFCET.

ANNEXE

Le programme de la 2^{ème} carte (la temporisation) :

```
porta equ $00
ddra equ $01
porte equ $0a
portb equ $04
portf equ $05
ldaa #$00
staa ddra
ldx #$1000
ldab #$00
ldab #$00
stab $0a
int0:
ldaa #$ff
int1:
ldy #$ffff
int2:
dey
bne int2
deca
bne int1
bset portb ,x,$01
incb
stab $0a
subb #$03
beq int4
ldab $0a
brset $0a,$02,int3
bra int0
int3:
bclr portb,x,$01
bset portb,x,$02
bra int0
int4:
bclr portb,x,$02
bset portb,x,$04
int01:
ldaa #$7f
int5:
ldy #$ffff
int6:
dey
bne int6
deca
bne int5
bclr portb,x,$04
bset portb,x,$08
end
```


1* Définitions des symboles utilisés:

T= température

V= ventilation

H= humidité

R= résistance

P= porte de la chambre (fin de course)

Vérin= le vérin

Lh= l'horloge

Mh= le moteur d'humidité

Mt= moteur d'incubateur

Turbo= le turbo

Vr= la tension d'alimentation de résistance

Vmt= la tension d'alimentation de moteur d'incubateur

Vmh = la tension d'alimentation de moteur d'humidité

Ver= la tension d'alimentation de vérin

Tmin= température minimal

Tmax= température maximal

Vmin= ventilation minimal

Vmax= ventilation maximal

Hmax= humidité maximale

Hmin= humidité minimale

R= 1 la résistance est déclancher

R= 0 la résistance est arrête

V=1 la ventilation est déclancher

V= 0 la ventilation est arrête

Verin= 1 le vérin est dans la position maximal

Verin=0 le vérin est la position minimal

Mt=1 le moteur d'incubateur est déclancher

Mt=0 le moteur d'incubateur est arrêt

Mh=1 le moteur d'humidité est déclancher

Mh=0 le moteur d'humidité

Lh= 1démarrage de l'horloge

P=1 la porte est ouvert

P=0 la porte est fermé

Er= erreur dans la résistance:

- Er=1 : aucun erreur dans la résistance
- Er=0 : il y a un erreur dans la résistance

Ev= erreur dans la ventilateur :

- Ev= 1 : aucun erreur dans la ventilateur
- Ev= 0 il y a un erreur dans la ventilateur

Ever= erreur dans le vérin :

- Ever=1 aucun erreur dans le vérin
- Ever=0 il y a un erreur dans le vérin

Emh= erreur dans le moteur d'humidité :

- Emh= 1 aucun erreur dans le moteur d'humidité
- Emh= 0 il y a une erreur dans le moteur d'humidité

Emt= erreur dans le moteur de l'incubateur :

- Emt= 1 aucun erreur dans le moteur de l'incubateur
- Emt= 0 il y a une erreur dans le moteur de l'incubateur

2* Le jeu d'instruction de MOTOROLA:

Liste alphabétique des instructions :

ABA : addition de A et de B

ABX : Addition de B à X

ABY : Addition de B à X

ADCA : Addition avec retenue du registre A

ADCB : Addition avec retenue du registre B

ADDA : Addition avec A

ADDB : Addition avec B

ADDD : addition de D avec un double octet

ANDA : ET logique avec A

ANDB : ET logique avec B

ASL : décalage à gauche d'une case mémoire

ASLA : Décalage arithmétique à gauche de A

ASLB : Décalage arithmétique à gauche de B

ASLD : Décalage arithmétique à gauche de D

ASR : Décalage arithmétique à droite d'une case mémoire

ASRA : Décalage arithmétique à droite de A

ASRB : Décalage arithmétique à droite de B

BCC : branchement si la retenue est nulle

BCLR : Mise à 0 d'un bit spécifique

BCS : branchement si la retenue est égale à 1

BEQ : branchement si égal (à 0)

BGE : branchement si supérieur ou égal

BGT : Branchement si supérieur à zéro

BHI : branchement si plus grand

BHS : branchement si supérieur ou égal

BITA/BITB: Test d'un octet

BLE: Branchement si inférieur ou égal

BLO : branchement si inférieur

BLS: branchement si plus petit ou égal

BLT : Branchement si inférieur

BMI: branchement si négatif

BNE: branchement si différent (de 0)

BPL: branchement si positif

BRA: branchement inconditionnel

BRCLR: branchement si le bit spécifié est à 0

BRN : Ne jamais brancher

BRSET: branchement si le bit spécifié est à 1

BSET : Mise à 1 d'un bit spécifique

BSR: branchement à un sou programme

BVC : branchement si le dépassement est nul

BVS : branchement si le dépassement est à 1

CBA : comparaison de A et B

CLC: Mise à 0 du bit C du registre de code condition

CLI : Mise à 0 du bit I

CLR / CLRA / CLRB: mise à zéro d'un octet

CLV : mis à 0 du bit V

CMPA / CMPB: comparaison d'un accumulateur avec la donnée spécifiée

CPD : Comparaison entre D et une donnée 16 bits

CPX/CPY: comparaison du registre d'index avec la donnée spécifiée

COM / COMA / COMB: complémentation logique

DAA : ajustement décimal

DEC/DECA//DECB: Diminue de 1 (décrémente)

DES : Décrémente le registre S

DEX : décrémenter X

DEY : Décrémenter Y

EQU: Equivalent

END : fin de programme

EORA/EORB: OU exclusif entre l'accumulateur et une donnée 8 bits

FDIV : division fractionnaire

IDIV : division entière

INC / INCA / INCB: incrémentation d'une donnée en mémoire, de A ou de B

INS : incrémentation du pointeur de pile

INX/INY : incrémentation de X ou Y

JMP: Saut inconditionnel

JSR: Saut à un sous programme

LDAA / LDAB: chargement des accumulateurs

LDD : Chargement du registre D

LDS : chargement dans S d'un double octet

LDX : Chargement du registre X

LDY : Chargement du registre Y
LSL : Décalage arithmétique à gauche
LSLA : Décalage arithmétique à gauche de A
LSLB : décalage arithmétique à gauche de B
LSLD : Décalage arithmétique à gauche de D
LSLX : Décalage arithmétique à gauche de X
LSR/LSRA/LSRB/LSRD: décalage logique à droite

MUL : multiplication de A par B
NEG/NEGA/NEGB: changement de signe ; complément à 2
NOP: instruction sans effet
ORAA: OU logique entre A et la donnée spécifiée
ORAB: OU logique entre B et la donnée spécifiée
ORG : Origine de programme
PSHA/ PSHB : sauvegarder l'accumulateur dans la pile
PSHX/PSHY : sauvegarder les registres X ou Y
PULA/PULB : restaurer le contenu de la pile dans A ou B

PULX/PULY : Restaurer le contenu de X ou Y
ROL/ROLA/ROLB: Rotation à gauche
ROR/RORA/RORB: Rotation à droite
RTI: retour d'interruption
RTS: Retour de sous-programme
SBA : Soustraction entre A et B
SBCA/SBCB: soustraction avec retenue
SEC: mise à 1 du bit C
SEI : Mise à 1 du bit I
SEV : Mise à 1 du bit V
STAA/STAB: rangement du contenu de A ou de B
STD : stockage du registre D
STOP : arrêt du processeur

STS : stockage du registre S

STX / STY : stockage des registres d'index

SUBA/SUBB: soustraction

SUBD : soustraction sur 16 bits

SWI: interruption logicielle

TAB : transfert de A dans B

TAP : transfert de A dans CC

TBA: transfert de B dans A

TPA: transfert de CC dans A

TST / TSTA / TSTB : Teste si négatif ou nul

TSX : transfert de S dans X

TSY : Transfert de S dans Y

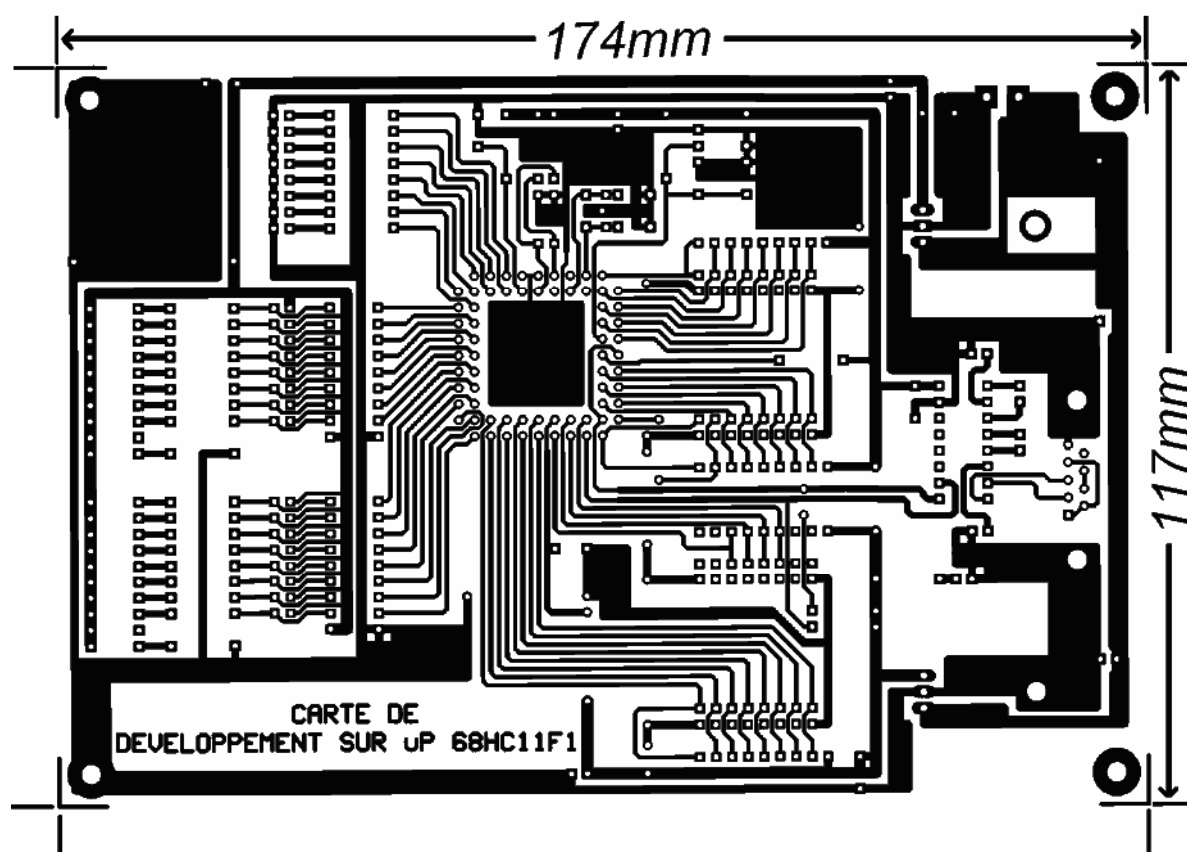
TXS : transfert de X dans S

TYS : transfert de Y dans S

WAI : attente d'interruption

XGDX : échange entre X et D

XGDY : échange entre Y et D



Circuit imprimé de la carte automate(électronique pratique n°225)

BIBLIOGRAPHIE

➤ André SIMON.

« **AUTOMATES PROGRAMMABLES INDUSTRIELS** »

Edition L'ELAN LIEGE EYROLLES PARIS 1991.

➤ G. Michel, C. Laurgeau, B. Esplau.

« **LES AUTOMATES PROGRAMABLES INDUSTRIELS** »

Edition DUNOD PARIS 1979.

➤ J.C.BOSSY, P.BRARD, P.FAUGERE, C.MERLAUD.

« **LE GRAFCET SA PRATIQUE ET SES APPLICATIONS** »

Editons Casteilla .

➤ Bernard Beghyn.

« **LE MICROCONTROLEUR 68HC11** »

Edition HERMES PARIS 1997.

➤ Christian Tavernier.

« **MICROCONTROLEUR 68HC11** »

Version UVPROM ET EEPROM APPLICATIONS.

Edition DUNOD PARIS 1997.

➤ Christian CAZAUBON.

« **LE MICROCONTROLEUR 68HC11 ET LEUR PROGRAMATION** »

Edition MASSONPARIS 1997.

➤ Satta. KH, Louassaa.M.

« **ETUDE ET REALISATION D'UN AUTOMATE PROGRAMABLE A
BASE D'UN MICROCONTROLEUR 68HC11F1** »

Thèse d'ingénieur d'état en automatique. 2002 .

➤ <http://legrafcet.html>.

➤ <http://perso.wanadoo.fr/ybnet/F1board.html>.

➤ <http://www.chez.com/68HC11/descrip.html>.

➤ <http://members.lycose.fr/superjp007/microprocesseur.html>.