République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de La Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de Projet de Fin d'Etudes en vue de l'Obtention du Diplôme d'Ingénieur d'Etat en Electronique

Thème

Intégration d'un API à base de PIC dans un processus

Proposé et dirigé par :
Mr M. HADDADI *Maître de conférences*
Mr M.S.AIT CHEICK *Maître de conférences*

Présenté par :
Mr TLEMSANI Hichem

Soutenu le : 09.07.2003

Devant le jury composé de :
- Mr R.IBTIOUEN Président de jury
- Mr Z.TERRA Examinateur
- Mr C.LARBES Examinateur

Promotion : Juin 2003

ENP, 10 Avenue Hassen Badi EL-Harrach Alger

A la mémoire de mon oncle Hebrri

A mes grands parents

A mes parents

A mes amis

Et à tous ceux qui m'ont aidé notamment mes professeurs et l'équipe de la laiterie TREFLE

Résumé

Ce projet consiste à intégrer un API à base de PIC dans un processus opérationnel comme celui de la laiterie TREFLE avec lequel on a travaillé. On a réalisé dans un premier temps l'automate à base d'un PIC 16F84 a partir d'une conception faite préalablement; en ajoutant quelques modifications aux circuits d'un coté pour qu'il présente une interface de communication série RS232. Et aux programmes de l'automate pour intégrer la routine RS232. Par la suite on a programmé la routine RS232 sur l'automate ACCOS de la laiterie TREFLE, en utilisant les cartes d'E/S. Cette routine est programmée sous forme d'une séquence qui est intégré dans le projet complet de l'usine.

Abstract

The purpose of this project is the integration of an API made of PIC 16F84 in a process. First, we realize the API with PIC 16F84 which conception is doing already. We add some modifications to circuit on the one hand in order that presents a RS232 serial communication interface and to programmes on the another hand for integration of RS232 routine. After, we programme the RS232 routine on ACCOS "API of TREFLE dairy". In this programming we used input/output cards of API, and this programme is including in the full project under sequence form.

Mots clefs

- API
- PIC 16F84
- RS232
- MPLAB
- MAX232
- driver
- Grafcet
- Interpréteur grafcet
- Communication
- Protocole
- Scrutation
- Séquence
- Interruption
- Mémoire image entrée
- Mémoire image sortie
- Profibus

Keys words

- API
- PIC 16F84
- RS232
- MPLAB
- MAX232
- driver
- Grafcet
- Grafcet interpreter
- Communication
- Protocol
- Scrutiny
- Sequence
- Interruption
- Input image memory
- Output image memory
- Profibus

Sommaire

Introduction

Le travail que nous nous sommes proposé d'effectuer dans le cadre de ce projet de fin d'études consiste à réaliser un automate à base du pic 16F84 et de le relier à un second automate industriel ACCOS fonctionnel et déjà installé dans l'usine de production laitière TREFLE.

L'automate à PIC ayant été développé [15], il nous reste à étudier la liaison et développer les programmes qui permettront le fonctionnement entre ces deux automates.

La communication entre les automates se fait habituellement à l'aide d'une liaison série RS232, ou via un réseau Ethernet. On a choisi pour notre réalisation la liaison RS232 pour sa simplicité de mise en œuvre que ce soit au niveau matériel ou logiciel.

Dans un premier temps on réalisera l'automate à base de PIC 16F84, et on intégrera la routine RS232 dans ses programmes. Après cette phase on abordera le problème de l'intégration de cet automate dans le processus de la laiterie TREFLE.

I. Les Automates programmables :

1. Historique : [3]

Les automates programmables sont apparus aux U.S.A en 1969 – 1970, et plus particulièrement dans le secteur de l'industrie automobile; ils furent utilisés en Europe environ deux ans plus tard. Leur date de création coïncide donc avec le début de l'ère du microprocesseur et avec la généralisation de la logique câblée modulaire.

L'automate est la première machine langage c'est-à-dire un calculateur logique dont le jeu d'instructions est orienté vers les problèmes de logique et vers les systèmes à évolution séquentielle. Il est à noter que de plus en plus l'universalité des calculateurs tend à disparaître et l'avenir semble s'ouvrir vers cette nouvelle classe de dispositifs : machines pour le traitement du signal, machines pour la gestion de base de données...

L'automate programmable est donc dans ce sens un précurseur et constitue une ébauche de la véritable machine pour les automaticiens.

L'extension croissante des applications de l'électronique, la diminution fantastique du prix des composants, la naissance et le développement des microprocesseurs et surtout la miniaturisation des circuits mémoires ont permis d'imaginer des automates programmables, dont les prix attractifs, pour des performances guère moindres, seraient tels que les champs d'application nouveaux leur seraient offerts.

L'automate programmable peut être employé seul ou inséré dans un système de contrôle de processus réparti et, cela, dans des installations aussi variées qu'une boulangerie, une fonderie, une raffinerie, un téléphérique ou une installation de chauffage.

Avec une combinaison de possibilités de régulation PID et de commande séquentielle, l'automate programmable satisfait aux exigences des processus à la fois continus et discontinus; il règle les pressions, températures, niveaux et débits, et assure toutes les fonctions annexes de temporisation, cadence, comptage et logique associée.

Assorti d'une carte de communication additionnelle, l'automate se transforme en un puissant satellite dans un réseau de contrôle distribué.

2. Définition :(projet de norme C63-850 de l'UTE et DIN 19237)

L'API pour automate programmable industriel est un appareil électronique programmable par un utilisateur automaticien et destiné à piloter en environnement industriel des machines ou des processus logiques séquentiels.

3. Système automatisé :

Un système automatisé se compose de deux parties qui coopèrent :

-une **partie opérative** constituée du processus à commander, des actionneurs qui agissent sur ce processus et des capteurs permettant de mesurer son état.

-une **partie commande** qui élabore les ordres pour les actionneurs en fonction des informations issues des capteurs et des consignes. Cette partie commande peut être réalisée par des circuits câblés, ou par des dispositifs programmables (automates, calculateurs)

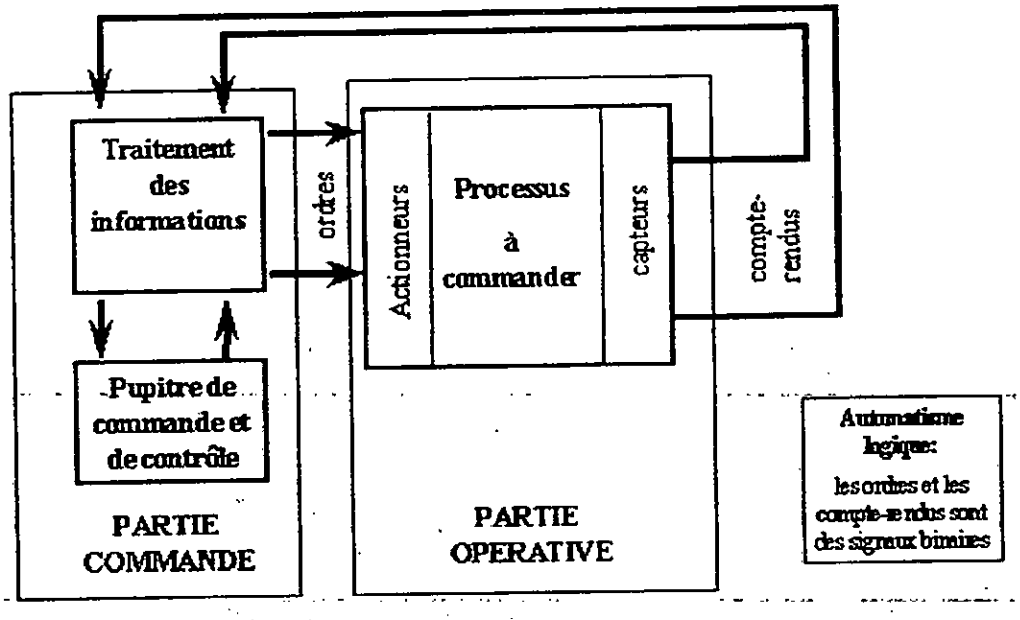


Fig.2.1. Un système automatisé

Cahier des charges d'un automatisme logique

Le cahier des charges décrit :

-les relations entre la partie commande et la partie opérative

-les conditions d'utilisation et de fonctionnement de l'automatisme

Le fonctionnement d'un automatisme séquentiel peut être décomposé en un certain nombre d'étapes. Le passage (ou transition) d'une étape à une autre étape se fait à l'arrivée d'un évènement particulier (réceptivité) auquel le système est réceptif.

4. Structure générale :

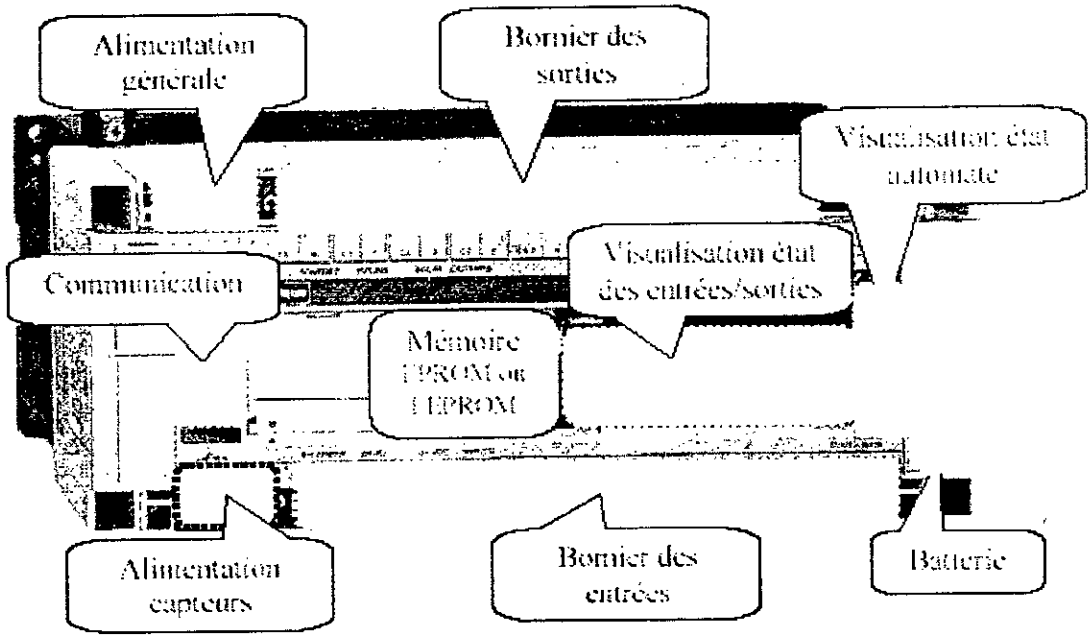


Fig.2.2. Structure générale d'un API

5. Principe de fonctionnement :

La plupart des automates utilisent le principe de scrutation afin de permettre des réactions en temps réel (au temps de réponse prés). Le déroulement du programme repose sur le principe suivant :

- Les instructions sont automatiquement exécutées les unes après les autres
- Lorsque toutes les instructions du programme ont été lues, le programme se déroule de nouveau depuis le début.
- Cette relecture cyclique est ininterrompue, on parle de cycle de scrutation. Le déroulement normal du programme peut être brièvement interrompu si des tâches prioritaires s'avérant nécessaires (cette interruption peut être générée soit par occurrence d'un événement soit périodiquement), après traitement prioritaire le programme est repris à l'endroit où il a été interrompu.

La puissance d'une UC est directement fonction de sa vitesse : on appelle période d'un A.P.I le temps d'exécution de 1 K-instructions logiques.

- **Différents types de cycles de scrutation :**

Trois éléments sont susceptibles de conditionner la structure du cycle, c'est-à-dire la répartition des diverses phases de travail au cours de son déroulement :

_ La politique d'acquisition des entrées/sorties;

_ L'utilisation d'instructions de saut;

_ La présence éventuelle de calculs numériques.

Dans l'hypothèse d'un traitement purement logique, sur un A.P.I monoprocesseur avec pour seul saut le retour à l'instruction n°1, on rencontre typiquement quatre types de cycles, cette organisation peut être figée, ou, dans certains cas dynamique

a) Toutes les entrées sont acquises en début de cycle et les sorties sont commandées après que toutes les équations aient été résolues.

b) Les entrées sont acquises comme en a) mais, par exemple, les sorties sont commandées après chaque résolution d'équation.

c) Le cycle correspond à une organisation équation par équation comprenant la lecture des variables et le traitement logique qui en résulte. C'est le cas le plus

favorable à l'introduction d'aléas puisqu'une variable peut être prise en compte avec deux valeurs différentes dans deux équations successives d'un même cycle.

d) Les entrées sont scrutées toutes les n millisecondes, quelque soit la durée du cycle; les sorties sont activées à la demande.

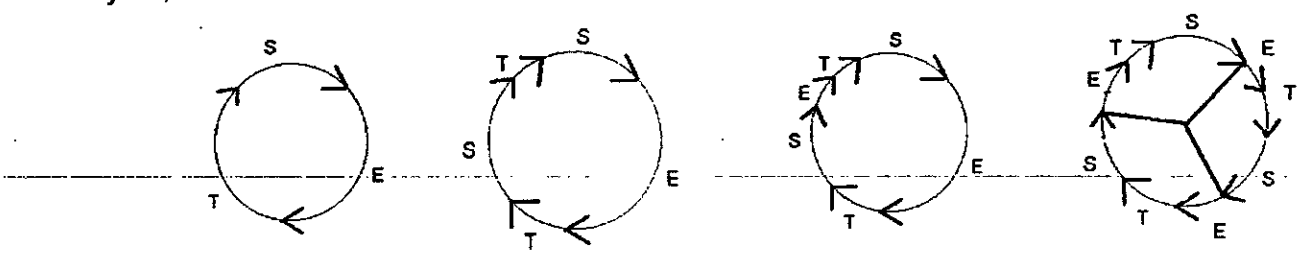


Fig.2.3. types de cycles de scrutation

Les structures multiprocesseurs engendrent des cycles apparemment encore plus complexes, par l'existence d'un cycle principal, de cycles secondaires (scrutation des E/S par les processus d'E/S; rafraîchissement périodique des valeurs de régulation PID;...) et de processeurs de calcul, éventuellement asynchrones. En fait, le degré de complexité est le même. Les différentes entités travaillent à leur propre rythme avec une puissance accrue grâce à leur spécialisation. La synchronisation interprocesseur est soit à la charge du processeur principal qui consulte, lorsqu'il en a besoin, les co-processeurs, soit effectuée par le jeu du système d'interruption. On retrouve schématiquement le s cas évoqués précédemment pour la gestion des E/S par le cycle de base.

- **Mémoires images des entées (MIE)**

L'apparition et la disparition des signaux délivrés par les capteurs ont évidemment lieu indépendamment de l'automate : les états des entrées basculent tout au long du déroulement du programme. Cette instabilité peut générer des aléas de fonctionnement car lorsqu'on développe une application, une entrée est considérée comme étant soit vraie, soit fausse à un moment donné et non les deux à la fois. Pour la durée d'exécution d'un seul cycle, les états des entrées sont rendus stables par les mémoires images des entrées. Le mécanisme en est le suivant :

- En début de scrutation, les états des entrées physiques telles qu'elles sont vues par les coupleurs d'entrées sont mémorisés dans les mémoires images correspondantes

_ a chaque entrée correspond une mémoire-image (un bit interne en fait)

_ Ce bit interne conserve son état jusqu'au prochain rafraîchissement, même si l'entrée physique correspondante change d'état pendant la scrutation.

- Pendant l'exécution du programme, toute instruction qui requiert l'état d'une entrée provoque en réalité la lecture de l'image de cette entrée

_ En général, le bit interne porte le même nom que l'entrée qu'il représente

_ De ce fait, le programmeur distrait peut penser traiter les entrées physiques alors que le programme qu'il écrit traite leurs images.

- Les mémoires images des entrées sont rafraîchies en début de chaque nouvelle scrutation

_ Si une entrée physique change d'état, son nouvel état ne sera pris en compte par le programme que lors de la scrutation suivante

_ Si l'état vrai d'une entrée physique a une durée si brève qu'il apparaît puis disparaît pendant une même scrutation, cette entrée est considérée par le programme comme étant restée fausse.

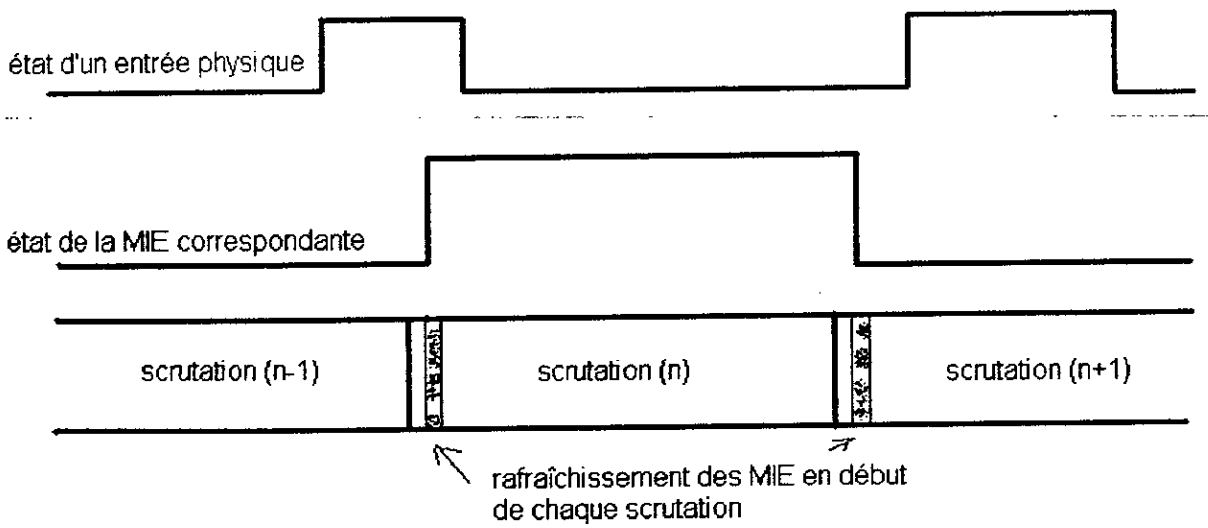


Fig.2.4. mémoires images des sorties (MIS)

- **Mémoires images des sorties (MIS)**

Un mécanisme similaire met en œuvre les mémoires images des sorties, interfaces logique avec les sorties physiques. Là aussi, le programmeur peut penser

assigner directement les sorties, alors qu'il assigne en fait des bits internes qui portent simplement les mêmes noms que les sorties. Le mécanisme est le suivant :

- Chaque instruction du programme qui positionne une sortie dans un état déterminé a en réalité comme effet de positionner le bit interne, image de sortie en question. La sortie n'est pas affectée par cette instruction, le programme se poursuit
- Chaque nouvelle instruction qui positionne une sortie met la mémoire image correspondante à jour. L'état de la mémoire-image change d'état au fur et à mesure du déroulement du programme.
- Après déroulement complet du programme, les états des mémoires images sont transférés globalement vers les coupleurs de sortie. L'une des fonctions des coupleurs de sortie est de mémoriser ces états pendant toute la durée du cycle de scrutation suivant, les sorties sont ainsi maintenues dans leur état jusqu'à la fin du prochain cycle.

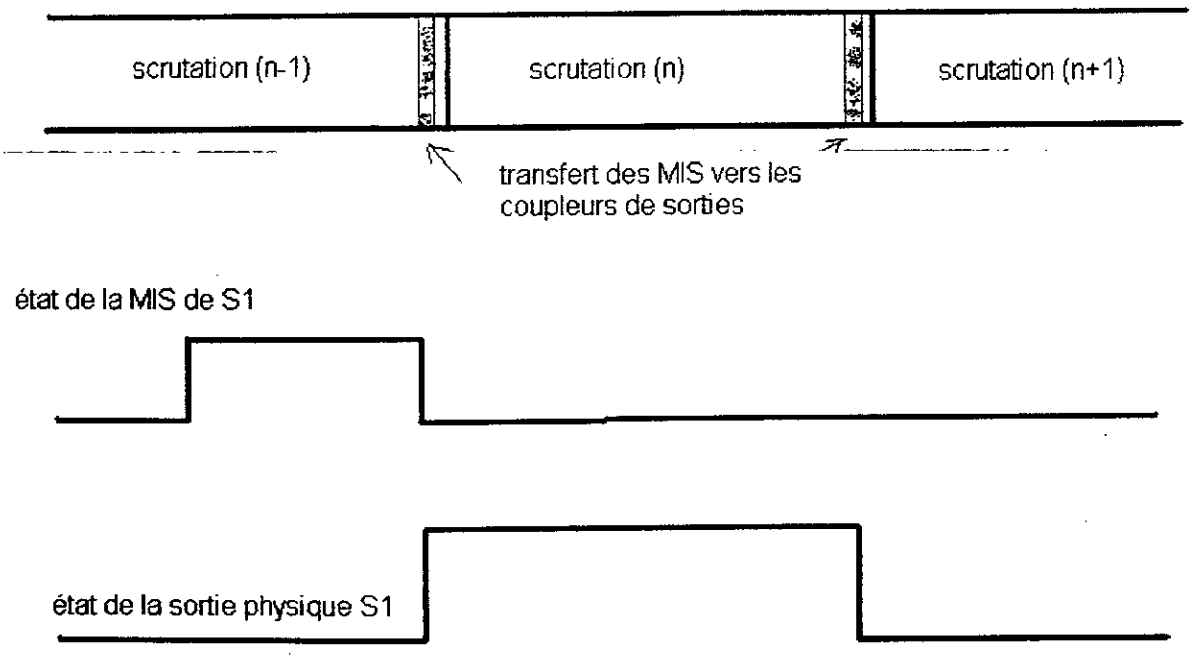


Fig.2.5. Mémoires images des sorties (MIS)

Le traitement à lieu en quatre phases :

•Phase 1 : Gestion du système

•Autocontrôle de l'automate

•**Phase 2** : **Acquisition des entrées**

Prise en compte des informations du module d'entrées et écriture de leur valeur dans RAM (zone DONNEE).

•**Phase 3** : **Traitement des données**

Lecture du programme (située dans la RAM programme) par l'unité de traitement.

Lecture des variables (RAM données),

Traitement et écriture des variables dans la RAM données.

•**Phase 4** : **Emissions des ordres**

Lecture des variables de sorties dans la RAM données et transfert vers le module de sorties.

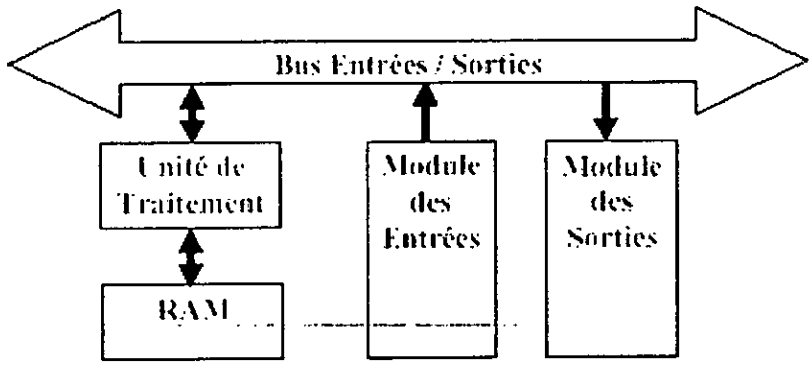


Fig.2.6. Mode de fonctionnement d'un automate

6. Caractéristiques générales :

Les caractéristiques principales d'un API sont :

•Compact ou modulaire

•Tension d'alimentation

•Taille mémoire

•Temps de scrutation

•Sauvegarde (EPROM, EEPROM, pile, ...)

•Nombre d'entrées / sorties

•Modules complémentaires (analogique, communication,...)

•Langage

a. Unité Centrale

L'unité centrale est le regroupement du processeur et de la mémoire centrale. Elle commande l'interprétation et l'exécution des instructions programmes. Les instructions sont effectuées les unes après les autres, séquencées par une Horloge. Exemple : Si deux actions doivent être simultanées, l'API les traite successivement.

Caractéristiques principales :

- Vitesses de traitement : C'est la vitesse de l'UC pour exécuter 1 K-instructions logiques. (10 à 20 ms/Kmots).
- Temps de réponse : scrutation des entrées, vitesse de traitement et affectation des sorties.

b. Mémoire

Deux types de mémoire cohabitent :

•***La mémoire Langage*** où est stocké le langage de programmation. Elle est en général figée, c'est à dire en lecture seulement. (ROM : mémoire morte)

•***La mémoire Travail*** utilisable en lecture-écriture pendant le fonctionnement c'est la RAM (mémoire vive).

.Attribution des zones mémoire travail en RAM :

Nature des Inform.	Désignations	Exploitation	Zones Mémoires
Etats des Capteurs	Variable d'entrée		
Ordres aux préactionneurs	Variable de sortie		
Résultats de fonctions comptage, temps	Variable Interne et ou Variable mot		
Résultats intermédiaires			
Instructions du cycle dans l'API	Programme		

•***Sauvegarde :***

Sauvegarde de la RAM (programmes, configuration, données)		Sauvegarde Externe (programme, configuration)
1 heure minimum par pile interne	1an par pile externe	permanente par EPROM (effaçable par ultraviolet), EEPROM (effaçable par courant électrique).

c. Les Modules Entrées - Sorties

Module d'extension d'Entrées/Sorties TOR

Module réseau : communication entre automate

Module d'extension d'Entrées Analogiques 0-10V Module d'extension de Sorties Analogiques 0-10V

d. Branchement des Entrées TOR

- Le principe de raccordement consiste à envoyer un signal électrique vers l'entrée choisie sur l'automate dés que l'information est présente.

L'alimentation électrique peut être fourni par l'automate (en général 24V continu) ou par une source extérieure.

Un automate programmable peut être à **logique positive** ou **négative**.

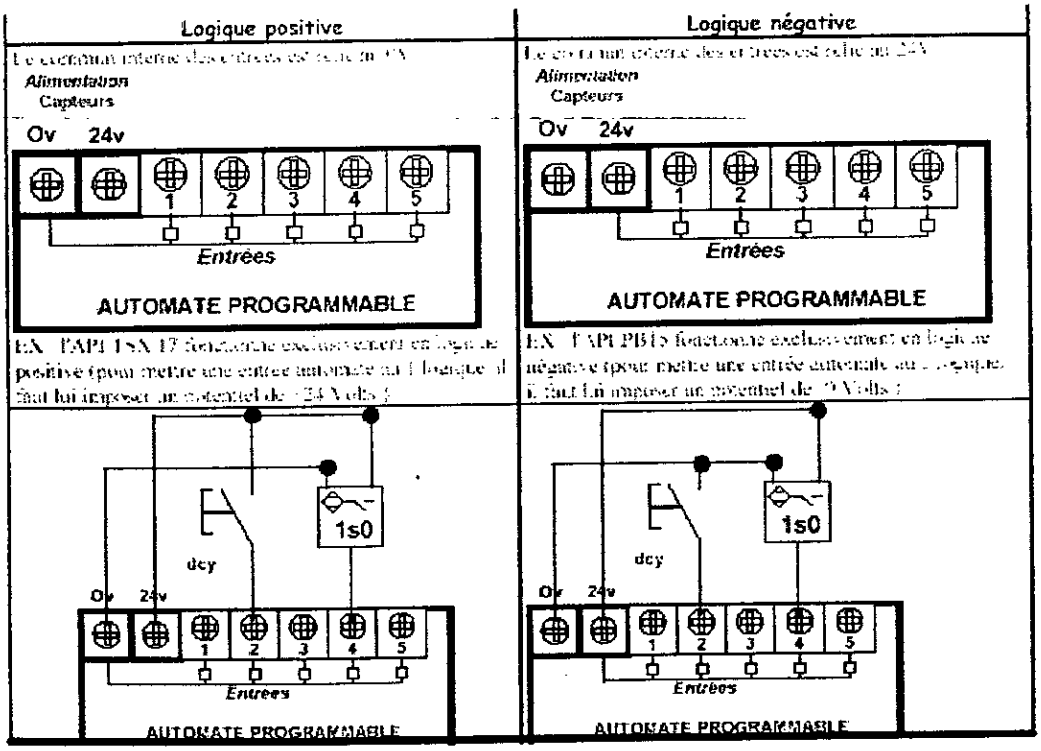


Fig.2.7. Branchement des entrées d'un automate

- **Branchement des sorties**

Le principe de raccordement consiste à envoyer un signal électrique vers le préactionneur connecté à la sortie choisie de l'automate dés que l'ordre est émis. L'alimentation électrique est fournie par une source extérieure à l'automate programmable.

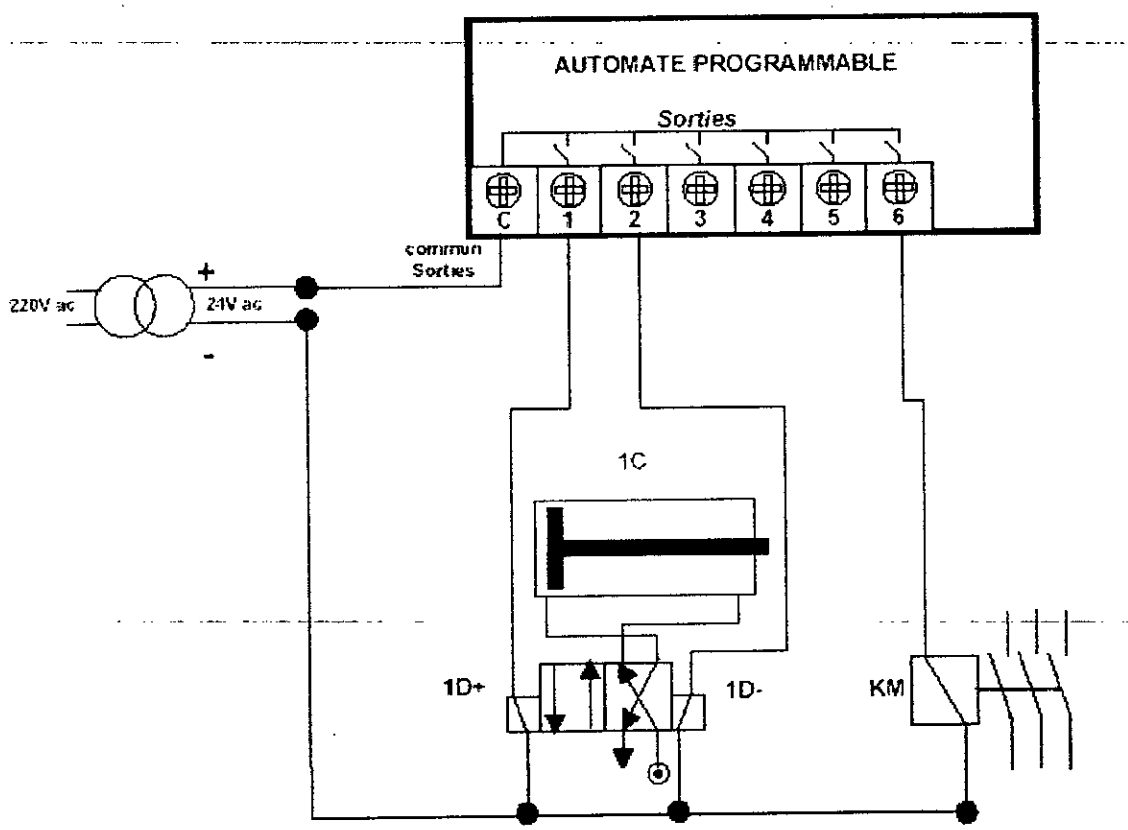


Fig.2.8. Branchement des sorties

e. Terminaux de programmation et de réglage

L'API doit permettre un dialogue avec :

•Le personnel d'étude et de réalisation pour réaliser la première mise en oeuvre (Edition programme, Transfert, Sauvegarde...)

•Le personnel de mise au point et de maintenance de réaliser des opérations sur le système (Forçage, Visualisation de l'état, Modification de paramètres temporisation, compteurs....)

Ce dialogue peut être réalisé par :

•Une Console : Elle sera utilisée sur site. Elle comporte un clavier, un écran de visualisation et le langage de programmation.

•Un Micro-ordinateur avec un logiciel d'assistance à la programmation : Il sera utilisé hors site. Il comprend plusieurs modules pour permettre l'édition, l'archivage, la mise au point des applications.

7. Préparation

La Partie Opérative du système, les grafcets de Production Normale, le Dialogue, le GEMMA (Modes de Marches et d'Arrêts), les GRAFCET de Sécurité et de Conduite étant définis, il reste à définir la Partie Commande.
Si le choix se porte sur un automate programmable, celui-ci étant relié aux préactionneurs (affectation Entrées/ Sorties) et ayant son propre langage de programmation, il faut traduire les GRAFCET précédents en un programme.
Tracer les GRAFCET adaptés à l'automate programmable.
_ Remplacer les réceptivités et les actions par les affectations des variables d'Entrées/Sorties.
_ Modifier les structures GRAFCET si nécessaire en fonction des possibilités du langage de programmation.
_ Préparer la programmation pour les temporisations, les compteurs, les mémorisations d'action etc... En respectant la syntaxe du langage de programmation.
Ecrire les équations de sorties
Recherche des conditions d'exécution des actions dans l'ensemble des grafcets et des équations logiques.
Noter l'état initial des variables Étapes actives au démarrage, mots de données pour tempo ou compteur.
Ecrire le programme. Il existe 2 possibilités d'édition de Programme :
_ Ecrire le programme directement dans le langage programmable sur feuille de programmation. (Ex : Langage littéral booléen ou GRAFCET PB15 ou Langage Graphique Schéma à contact ou GRAFCET PL7-2 pour console TSX).
Ecriture de l'ossature GRAFCET et des réceptivités, puis des équations de Sorties.
_ Utiliser un logiciel d'assistance à la Programmation (en général GRAPHIQUE) exemple AUTOMGEN

- ***Transfert du programme dans l'automate programmable***

Le transfert du programme peut être fait soit :

•manuellement en entrant le programme et l'état initial à l'aide d'une console de programmation

•automatiquement en transférant le programme à l'aide du logiciel d'assistance, et en réalisant la liaison série entre l'ordinateur et l'automate.

- ***Vérification du fonctionnement***

Lors de sa première mise en oeuvre il faut réaliser la mise au point du système.

_ *Prendre connaissance du système* (dossier technique, des grafcets et du GEMMA, affectation des entrées / sorties, les schémas de commande et de puissance des entrées et des sorties).

_ *Lancer l'exécution du programme* (RUN ou MARCHE)

_ *Visualiser l'état des GRAFCET, des variables...*

Il existe deux façons de vérifier le fonctionnement :

•En simulation (sans Partie Opérative).

•En condition réelle (avec Partie Opérative).

Simulation sans P.O	**Condition réel**
Le fonctionnement sera vérifié en simulant le comportement de la Partie Opérative, c'est à dire l'état des capteurs, en validant uniquement des entrées. _ Valider les entrées correspondant à l'état initial (position) de la Partie Opérative. _ Valider les entrées correspondant aux conditions de marche du cycle. _ Vérifier l'évolution des grafcets (étapes actives). _ Vérifier les ordres émis (Leds de sorties). _ Modifier l'état des entrées en fonction	Le fonctionnement sera vérifié en suivant le comportement de la P.O. _ Positionner la P.O. dans sa position initiale. _ Valider les conditions de marche du cycle. _ Vérifier l'évolution des grafcets et le comportement de la P.O. _ ... Toutes les évolutions du GEMMA et des

des ordres émis. (état transitoire de la P.O.). _ Modifier l'état des entrées en fonction des ordres émis (état final de la P.O.). _ ...	grafcets doivent être vérifiées.
Toutes les évolutions du GEMMA et des grafcets doivent être vérifiées.	

8. L'environnement des automates programmables :

On appelle périphériques d'un A.P.I les dispositifs, externes à L'A.P.I proprement dit, qui lui sont directement connectés.

Les auxiliaires d'un A.P.I désignent les périphériques de sa console de programmation.

La configuration d'un A.P.I est l'ensemble des dispositifs internes (processeurs spécialisés ou non, E/S industrielles) et externes (périphériques et auxiliaires) qui le complètent pour former un système apte à remplir les missions qui lui sont assignées.

Le choix d'une configuration est étroitement lié aux fonctions dévolues à l'A.P.I, aux solutions retenues pour les remplir et au catalogue du fournisseur de l' A.P.I ou de fournisseurs de matériels compatibles.

Les types de consoles les plus courantes qu'on trouve dans une configuration d'un A.P.I sont :

Une console de programmation

Une console d'exploitation

On trouve parmi les périphériques et les auxiliaires d'un automate :

Un simulateur : Pour valider le fonctionnement d'un programme sans pour autant perturber le fonctionnement de l'automate

L'unité de dialogue en ligne : (UDEL) Est un concept apparu peu avant 1980, et qui était issu de l'expérience d'utilisation des A.P.I. Permettant au personnel d'entretien d'accéder à la mémoire de l' A.P.I pour détecter d'éventuelles erreurs.

Les mémoires de masse : on trouve lecteur de disquette, disque dur

Imprimante :

Systèmes d'édition de dossiers d'automatismes :

Pour compléter la configuration d'un automate surtout dans un environnement multi-automate un tel système est indispensable. MULTIDOC de Citroën-Industries est à cet égard tout à fait significatif. Ses principales fonctionnalités sont :

- La création et la mise à jour des dossiers de programmation
- La création et le maintien d'une documentation facile d'accès aux différents programmes sur différentes machines.
- De joindre des commentaires automatiquement aux programmes
- La constitution et l'édition d'un dossier complet non seulement pour les codes, mais aussi concernant le nom, la nature et l'endroit ou interviennent les variables et aussi pour les schémas de raccordement. Tout ceci sous une forme propre et intelligible à l'automaticien

9. Communication entre les automates programmables :

Environné de sa configuration et connecté au procédé, l'A.P.I dépense une part importante de son énergie à communiquer. Cette communication présente les aspects suivants :

- La nature des informations traitées
- La finalité de leur traitement
- La topologie locale ou déportée des entrées-sorties.

On relève deux facteurs déterminant la forme de communication :

- Le caractère locale des échanges de haut niveau : entre console et A.P.I ou entre 2 A.P.I constituant un système dual.
- La nature centralisée des systèmes

On retrouve parmi les différentes formes de communication entre les A.P.I :

- La communication série que ce soit synchrone ou asynchrone comme la RS232C
- La communication parallèle
- Le réseau Ethernet

II. Présentation du PIC 16F84 [5],[2]

Le PIC n'est rien d'autre qu'un microcontrôleur, c'est-à-dire un system minimal comportant : un microprocesseur, une mémoire et des entrées sorties.

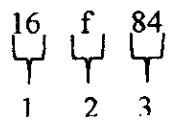
PIC est sous copyright de Microchip™, donc les autres fabricants ont été dans l'impossibilité d'utiliser ce terme pour leurs propres microcontrôleurs.

Microchip a lancé une gamme de PIC appelé PICmicro™ qui englobe trois familles :

- La famille Base-Line qui utilise des mots d'instruction de 12 bits
- La famille Mid-Range qui utilise des mots d'instruction de 14 bits (c'est le cas du 16f84
- La famille High-End qui utilise des mots instructions de 16 bits

Le PIC est conçu selon une architecture RISC (Reduce Instructions Set Computer) ceci induit l'utilisation d'un pipeline pour l'exécution des différents sous cycles de traitement d'une instruction. Donc toutes les instructions (sauf les sauts) s'exécutent en un cycle d'horloge.

Pour identifier un PIC on utilise la notification suivante :



Ou le premier le bloc indique la famille a laquelle appartient le PIC (16 pour les Mid-Range). Le deuxième bloc indique la nature de la mémoire programme, on peut rencontrer quatre types principalement :

- C pour une mémoire programme de type EPROM ou plus rarement de type EEPROM
- CR pour une mémoire de type ROM

- F pour une mémoire de type FLASH

En dernier le troisième bloc indique le PIC en type (ex : 84)

A présent on va s'intéresser particulièrement au PIC 16f84 en mettant en valeur ses caractéristiques les plus importantes et les avantages qu'on peut en tirer de son architecture et de son jeu d'instructions.

1. Caractéristiques :

Le PIC16f84 appartient à la famille Mid-Range donc elle possède 35 instructions chacune de longueur égale à 14 bits. Elle se présente sous la forme d'un boîtier de 18 broches dont 13 réservé pour les ports A et B, en outre il y a Vss et VDD pour l'alimentation, OSC1/CLKIN et $\overline{OSC2}$/CLKOUT pour l'horloge et une entrée reset MCLR comme le montre la figure.3.1.

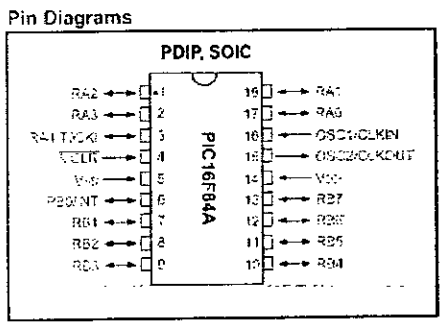


Fig.3.1. diagramme des pins du pic 16F84

Le PIC16f84 possède des fusibles programmables pour configurer l'utilisation ou non du :

_ CP "code protect" qui ne permet pas de relire le programme une fois écrit.

_ WDT "Watchdog" qui permet de surveiller si celui-ci s'exécute toujours dans l'espace que vous lui avez attribué

_ PWRT "Power-up Time" lors de son activation maintient le PIC en arrêt durant un temps typique de 72 ms dés la détection de la condition de reset.

_ L'horloge qu'on peut configurer par LP, XT ou HS pour une horloge externe ou par RC un réseau de capacité et résistance.

Les PIC 16f84 peuvent monter jusqu'à 10 MHz, donc une vitesse de traitement d'instruction de 2,5 MIPS. Il comporte entre autre :

- 15 registres spéciaux
- 8 niveaux d'imbrication de pile
- Adressage directe, indirect et immédiat
- Un bus de donnés de 8 bits
- Une mémoire programme de type Flash de 1K mot
- Une mémoire RAM de 68 Ko
- Une mémoire EEPROM de 64 Ko
- 4 sources d'interruptions :

  RB0/INT

  Débordement du TMR0

  Changement de niveau de l'un des pins du Port B <7:4>

  Fin d'écriture d'une donné en EEPROM
- 1000 cycles d'écriture et d'effacement de la zone programme
- 10.000.000 cycles d'écriture et d'effacement de la zone EEPROM
- Un timer/counter à 8 bits avec prédivision
- technologie CMOS :

  _ Niveaux de tensions de 2.0V à 6.0V

  _ Le courant : 25 mA en entrée max et 20 mA en sortie max

  _ Composant statique (abaissement de la fréquence de travail jusqu'à l'arrêt sans pertes de données et sans dysfonctionnement)
- Basse consommation
- Mode SLEEP

2. Les outils de programmation :

Pour notre cas on utilisera MPLAB de Microchip qui est un logiciel de compilation et de simulation. D'un autre coté on utilisera IPROG pour la programmation du PIC à travers un port de communication.

MPLAB :

C'est un logiciel fournit gratuitement par Microchip; il comporte plusieurs modes de développement pour notre application; on utilisera le mode édition et le mode simulation.

On a utilisé la version 6.10.0.0 de MPLAB qui diffère un peu de l'ancienne version. En premier lieu on devra choisir le PIC qu'on voudra programmer, par la suite créer un nouveau projet en mentionnant le nom du projet, les fichiers includes utilisés, le compilateur utilisé pour notre cas MPASM, le PIC utilisé "16F84" et le fichier .asm associé. On peut spécifier par la suite quelque paramètres de MPASM comme tenir compte de la majuscule ou pas et le format des nombres par défaut. Le fichier include utilisé est P16f84.INC et P16f84A.INC Une fois le fichier asm écrit et ajusté on peut passer à l'étape de compilation et création du fichier hexadécimal .HEX. Si il y a une erreur dans le programme le compilateur signalera Son type et son emplacement de cette erreur.

Simulation :

Si la compilation a été réalisée sans erreurs, on pourra à ce moment basculer en mode simulation en choisissant dans le menu debugger → Select Tool le simulateur qu'on veut utiliser. Pour notre cas on choisira MPLAB SIM.

ICPROG :

C'est un logiciel de programmation du PIC a partir d'un fichier hexadécimal, Son utilisation est simple il suffit de suivre les étapes suivantes :

- Choisir le programmateur adéquat parmi la liste proposée dans la fenêtre "Hardware Setting" en spécifiant les paramètres nécessaires : Delay, invert Vcc,.... Pour notre cas on a choisi un programmateur SCHAER PROGRAMMER.
- Spécifier le PIC qu'on veut programmer

- Charger le fichier hexadécimal du programme ou celui de la zone EEPROM

car le programmateur nous permet de les programmer séparément ou les deux en même temps.

- Effacer le contenu du PIC
- Lancer la programmation du PIC

Lors du chargement du fichier hexadécimal une modification de ce fichier à l'aide de ICPROG est possible. Ce logiciel permet aussi de lire le contenu d'un PIC.

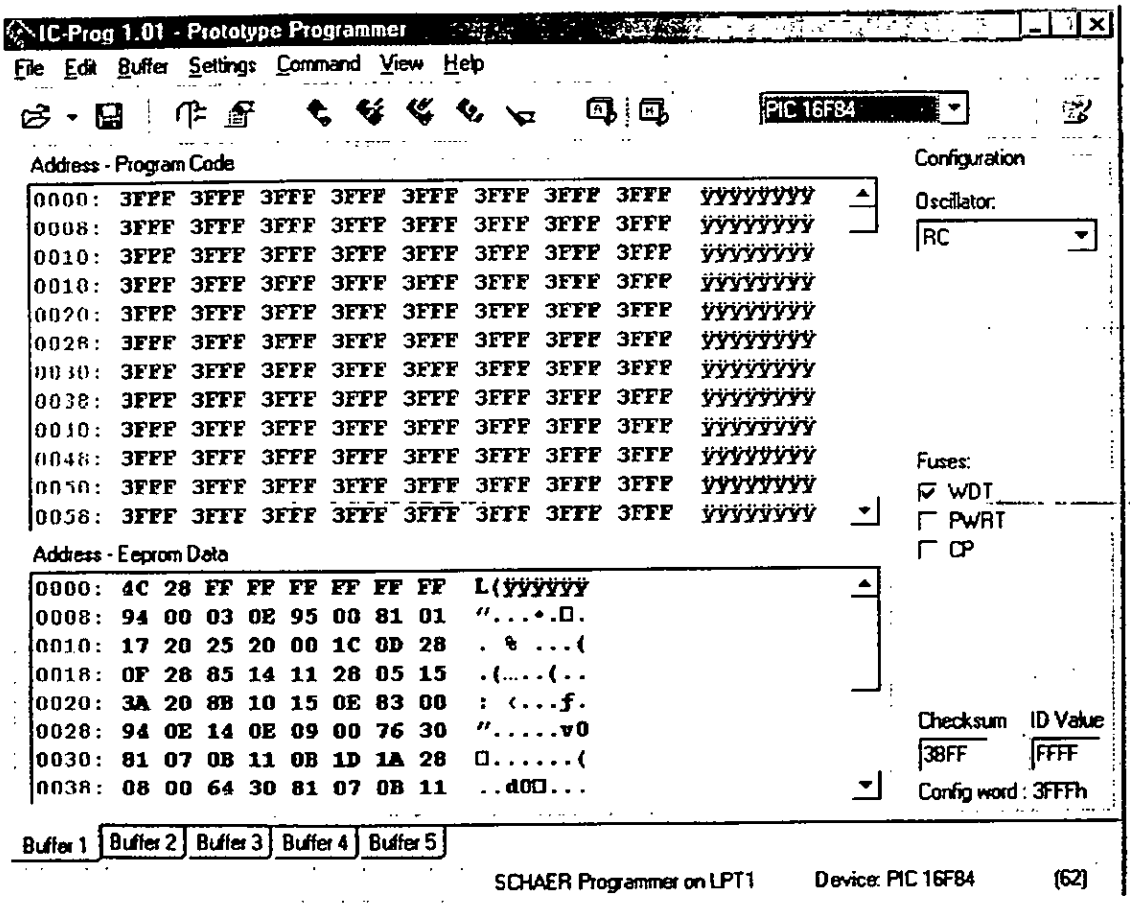


Fig.3.2. écran de ICPROG

III. Conception d'un automate à base d'un PIC16F84 : [15]

Le principe est le suivant, on construit le grafcet de commande sur le PC, puis on le transfère dans l'automate qui l'exécute. Donc cette réalisation comprend 3 grandes parties qui sont :

1. La partie électronique
2. La partie informatique au niveau du PC
3. La partie informatique au niveau du uC de l'automate

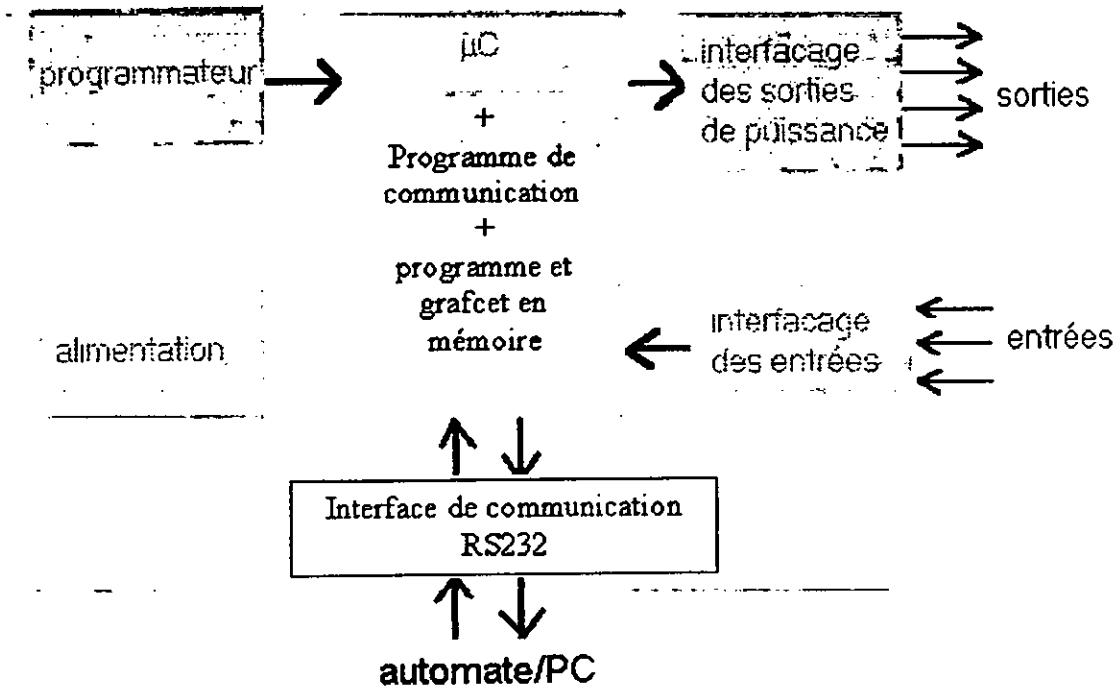


Fig.4.1. schéma fonctionnel de la partie électronique de l'automate

Du point de vue électronique l'automate se décompose en 5 sous-ensembles : figure.4.1.

- Le microcontrôleur coeur du montage qui grâce à un programme exécute le grafcet qui est dans sa mémoire.
- Le programmateur permettant de "rentrer" le grafcet dans l'automate et de programmer le uC.
- L'interface de communication qui permet de communiquer avec un autre automate.
- L'alimentation
- Les entrées (ici 7 entrées)
- Le système d'interfaçage de sortie (ici 4 sorties)

Schéma électrique de la partie automate

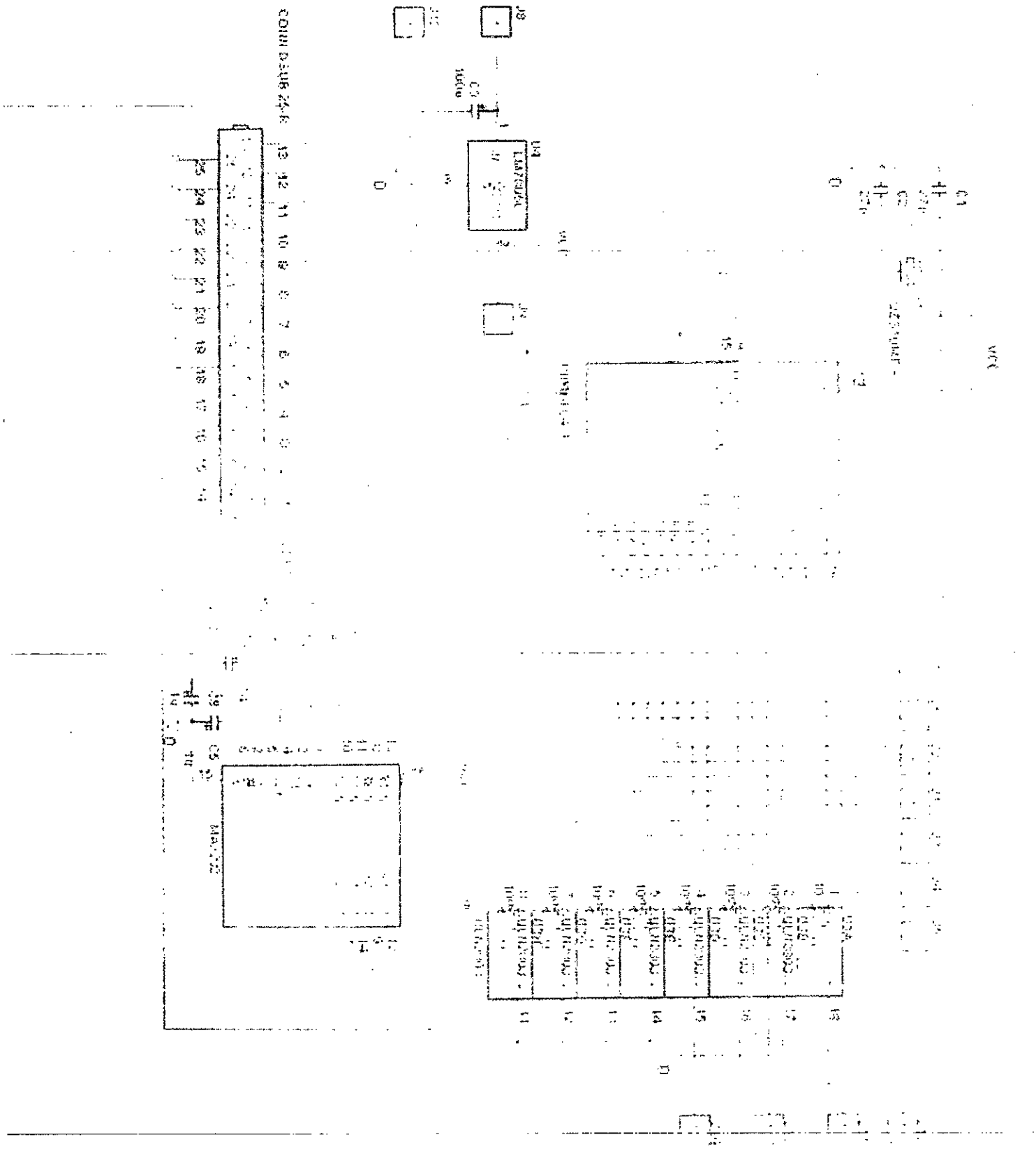


Fig.4.2. schéma électrique

Les composants utilisés pour ce circuit sont :

U1 : PIC16F84 (microcontrôleur)

U2 : MAX232 (convertisseur de niveaux tensions TTL/RS232)

U3A, U3G : ULN2803 (amplification de puissance (Darlington))

U4 : Régulateur 5 volts 7805

X1 : Quartz 4 MHz

C1 et C2 : 22 pF céramique

C3 :10 µF/25 V électrochimique sorties radiales

C4, C5, C6, C7 et C8 : 1µF/50 V électrochimique sorties radiales

6 borniers à vis pour circuit imprimé à 2 broches (J1, J2, J3, J4, J5, J6, J7, J11, J12, J13 et J14)

1 bornier à vis pour circuit imprimé à 3 broches (J8, J9 et J10)

2 supports tulipes à 18 broches

1 support de CI à 16 broches

JDR1 : connecteur SubD 25 points femelle coudé à implanter sur CI

La fonction de chaque bornier est comme suit :

Numéro du bornier	Fonction
J1	Entrée capteur (RB1)
J2	Entrée capteur (RB2)
J3	Entrée capteur (RB3)
J4	Entrée capteur (RB4)
J5	Entrée capteur (RB5)
J6	Entrée capteur (RB6)
J7	Entrée capteur (RB7)
J8	Entrée régulateur (5 à 18V)
J9	Entrée alimentation stabilisée 5V
J10	Sortie actionneur (RA0)
J11	Sortie actionneur (RA1)
J12	GND
J13	Sortie actionneur (RA3)
J14	Sortie actionneur (RA4)

Choix du µC :

Le choix du uC est primordial car de lui dépend en grande partie : les performances, la taille, la facilitée d'utilisation et le prix du montage.
Le µC doit interpréter le grafcet littéral, donc il faut une mémoire pour y mettre ce dernier .Pour que l'interprétation soit rapide, il faut une mémoire a accès parallèle
On a ici deux possibilités, soit utiliser un uC avec bus externe sur lequel on y met une mémoire, soit utiliser un µC qui possède sa propre mémoire .C'est cette dernière solution qu'on a choisi afin de rendre le montage simple.
Il faut maintenant choisir le type de mémoire en sachant qu'elle ne doit pas s'effacer lorsque l'on coupe le courant.
On peut donc utiliser une RAM sauvegardée , une EPROM , une EEPROM ou FLASH .On a choisi la solution EEPROM/FLASH car c'est la plus fiable et la plus flexible , en effet il n'y a ni besoin de pile pour la sauvegarder ,ni d'effaceur à ultraviolet pour l'effacer
A ce stade deux microprocesseurs font l'affaire : le PIC16C84 ou 16F84 et le 68HC811E2 .On a choisi le réaliser avec le PIC 16F84 car l'interpréteur et le grafcet logent dans la mémoire programme, les valeurs des temporisations sont dans l'EEPROM de données, tandis la RAM accueille les variables temporaires et la zone des bits qui indique l'activité du grafcet, des sorties ...
Le programmateur programme le PIC dans son ensemble, interpréteur + grafcet, donc un PIC même vierge peut y être mis sans programmation préalable, ceci simplifie grandement la réalisation.

ULN2803 :

ULN2803 est un réseau de transistors Darlington. Un tel réseau est capable d'écouler un courant max de 500mA par transistor et supporte une tension max de 50V. Ainsi on pourra connecter directement la majorité des relais sans problème.
Ces transistors se comportent comme des interrupteurs à la masse , ainsi en l'absence de commande ou une commande "0" bloque les transistors , l'interrupteur est ouvert , tandis qu'une commande "1" sature le transistor et donc ferme

l'interrupteur. Il faudra donc brancher la bobine du relais entre le plus et un transistor du réseau.

On a évité d'implanter des relais sur la carte car ceci dépend fortement de l'application. En effet pour allumer un voyant il n'y a pas besoin de relais avec les Darlington, de même on n'utilise pas les mêmes relais pour commander une machine à laver 220V ou un ventilateur 12V, on peut utiliser un triac pour une lampe 220V ... Ce choix permet de mettre ce qu'il faut, où il faut, c'est donc économique et réaliste.

La routine RS232 [10]

Introduction

La liaison RS232 a de multiples applications. Sa mise en oeuvre étant très facile, et permet un gain d'espace et une économie lors de la conception du circuit.

La liaison série RS232

Actuellement, pour faire dialoguer des équipements numériques entre eux, l'électronicien amateur dispose d'une variété importante de moyens de communication. D'une part, la connexion parallèle qui possède un nombre de bits variable en fonction du matériel (4, 8, 16, 32, ...). Ce type de connexion a l'avantage de satisfaire des débits très importants, mais aussi le fâcheux inconvénient de prendre beaucoup de place en circuit imprimé et de pins sur les circuits intégrés. D'autre part, il existe d'autres types de liaisons comme les liaisons séries synchrones qui offrent de bonnes performances en terme de rapidité, mais qui sont très délicates à mettre en oeuvre. Le troisième groupe est composé de liaisons séries asynchrones. Ce type de liaisons connaît un très grand succès grâce à sa simplicité de connexion et d'utilisation (les deux principales sont les liaisons série RS232 et RS485), car elles sont les plus intéressantes. Les liaisons RS232 et RS485 sont habituellement utilisées en asynchrone, mais il ne faut pas oublier que la norme prévoit également un fonctionnement en liaison synchrone.

La RS232 est utilisée depuis l'origine sur les PC en version asynchrone, elle présente les caractéristiques suivantes :

Niveaux de tension ±12V bipolaire

Débits bits/sec : 100-20 K

Distances utiles : 1m à 15 m

Nombre d'émetteurs par liaison : 1

Nombre de récepteurs par liaison : 1

La génération des trames série, peut se faire avec plusieurs manières. Une solution, courante, consiste à utiliser une UART (Universal Asynchrone Receive Transmit) qui grossièrement permet de convertir une donnée parallèle en donnée série et vice versa. Enfin la dernière solution, plus récente et liée à l'évolution des composants, est celle d'utiliser les entrées/sorties séries d'un microcontrôleur qui intègre une UART. Ou de programmer le fonctionnement d'une UART sur un microcontrôleur.

Description d'une trame

Transmettre une donnée en mode série impose l'utilisation d'une structure de trame. Pour notre cas, elle est définie par la norme. L'état initial est égal à "1", puis on émet le bit de start, un "0". Ensuite, on a la donnée en 4,6,7 ou 8 (7 pour l'ASCII), puis on a la possibilité d'avoir ou non un bit de parité permettant de vérifier la donnée. Enfin on a 1, 1.5 ou 2 bits de stop. La durée de maintien de chaque bit doit correspondre à une période d'horloge. Cette horloge n'est bien sûr pas la même des deux côtés de la liaison mais doit avoir la même fréquence. Une tolérance de 5% est acceptée sur cette horloge : au-delà, de nombreuses erreurs de réception sont à prévoir. Cette horloge n'est pas définie en Hertz par la norme, elle est indiquée en débit d'information, soit 75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200,2800, 56000, 128000 ou 256000 bauds

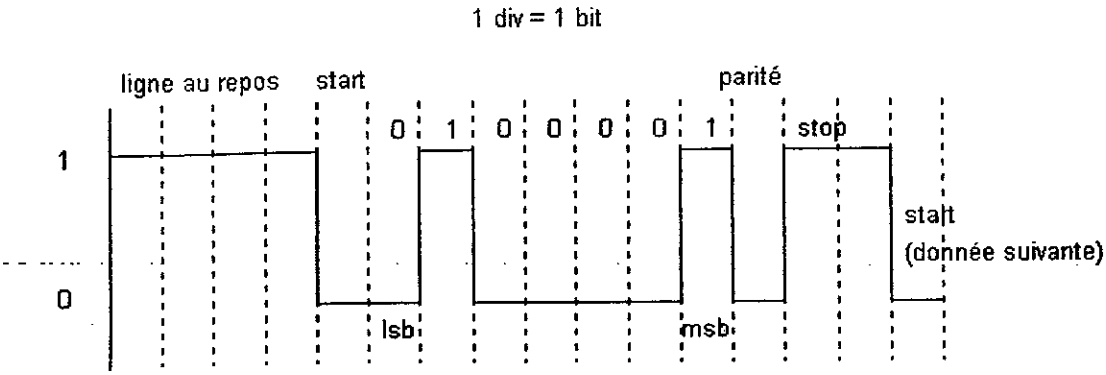


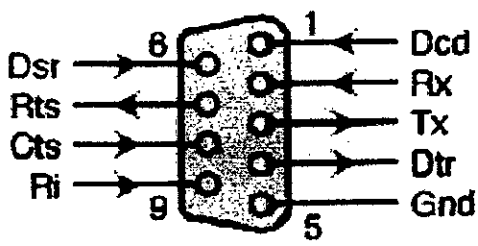
Fig.4.3. Protocole RS232

Description mécanique

De par la norme, la RS232-C avait pour connectique la Canon SUB-D 25 broches, mais les évolutions du matériel, notamment des PC, ont contribué à passer de la SUB-D 25 broches à la SUB-D 9 broches plus petite tout en conservant le nombre de broches nécessaire à une liaison complète. La Canon SUB-D 9 broches est pratiquement devenue le standard de connectique pour la RS232-C.

Soient les signaux :

- **TX** : Transmit Data
- **RX** : Receive Data
- **DTR** : Data Terminal Ready (prêt à recevoir ?)
- **DSR** : Data Set Ready (donnée prête ?)
- **RTS** : Request To Send (donnée à transmettre !)
- **CTS** : Clear To Send (prêt à recevoir !)
- **DCD** : Data Carrier Detect (liaison établie !)
- **RI** : Ring Indicator (demande de communication)



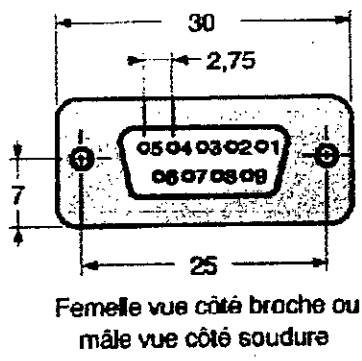


Fig.4.4. brochage d'un connecteur DB9

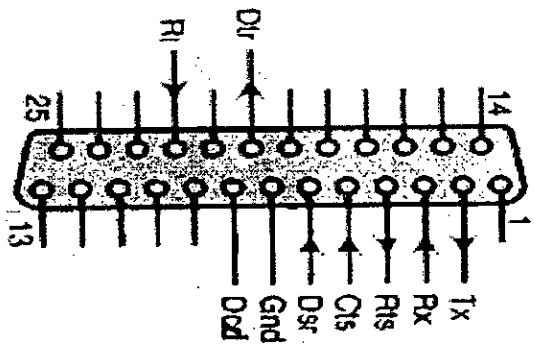


Fig.4.5. Brochage d'un connecteur DB25

Description du protocole :

On retrouve 3 types de liaisons :

Liaison simplex

Liaison half duplex

Liaison full duplex

Type de liaison couramment utilisé

La liaison RS232 la plus répandue chez les électroniciens est celle qui consiste à connecter une carte comportant un microcontrôleur avec un PC. En effet, connecter deux PC ensemble, moyennant un bon schéma de câblage, ne nécessite pas de connaissance particulière en électronique (voir ci-dessous). Donc dans le cadre de notre liaison microcontrôleur vers PC, on aura besoin seulement de 3 lignes: Rx, Tx etGND

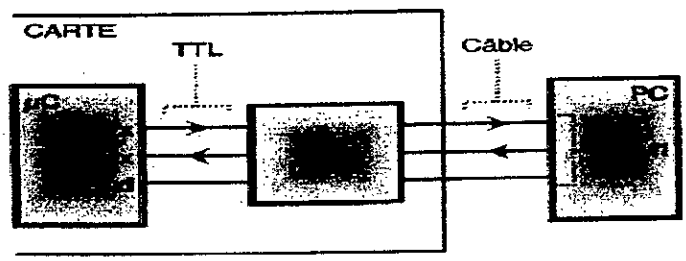


Fig.4.6. schéma simplifié d'une liaison RS232

Description électrique

Les liaisons RS232 entre un microcontrôleur et un PC ne nécessitent, en général, que trois fils si l'on considère qu'il s'agit presque toujours de liaison bidirectionnelle. Si on détaille : Tx (Transmission), Rx (Réception) et GND (Masse). Maintenant, on va s'intéresser aux niveaux de tension présents de chaque côté de la liaison, afin de mieux comprendre la fonction que doit réaliser le driver de RS232. On représente ces niveaux sous la forme d'un tableau.

Niveau Logique	sur SUB-D du P.C.	sur Pin du µC
"0" Low	+12V (+5 à +25V)	0V (Masse)
"1" Hi	-12V (-5 à -25V)	+5V (TTL)

On constate que les tensions normalisées sont égales à +12V et -12V. Il découle du tableau ci-dessus deux constatations évidentes : d'une part les différences de tensions (+12V, -12V, +5V, 0), et d'autre part l'inversion logique des niveaux. Si maintenant on s'attache aux fonctions à remplir, il faudra réaliser une inversion ainsi qu'une mise à niveau des signaux.

Solutions intégrées

Pour remplir ces deux fonctions, on peut citer les solutions communément utilisées dans l'ordre "historique". Dans un premier temps l'utilisation de drivers du type 14C88

et 14C89 comportant chacun quatre drivers de chaque type. Puis dans un souci d'optimisation, on peut utiliser des circuits combinant 2 drivers d'émission et 2 drivers de réception tel que le MC145406 qui, en termes de mise en oeuvre, ressemble énormément aux MC14C88 et MC14C89 : toutefois, on constate que l'on ne dispose pas très souvent du ±12V sur la carte du microcontrôleur et que la réalisation d'une alimentation auxiliaire, uniquement pour driver la RS232, n'est pas toujours facile et souvent encombrante.

C'est pourquoi la société MAXIM a mis sur le marché le célèbre MAX232, Le MAX232 possède, comme le 145406, deux drivers en émission et deux en réception mais également, d'où la nouveauté, un convertisseur à pompe de charge qui, à partir du +5V, crée du +12V (+10V) et du -12V (-10V). On peut noter auprès du MAX232 l'apparition de cinq condensateurs de 1µF qui pour quatre d'entre eux sont destinés directement aux convertisseurs à pompe de charge, et d'un 5ème (souvent oublié !) destiné à filtrer le +5V (Vcc) perturbé par les pointes de courant du convertisseur (commutation). Ainsi l'art du driver RS232 atteint son apogée. Les versions qui suivent correspondent souvent à des produits "costumes" pour des applications spécifiques, mis à part peut-être le MAX233 qui moyennant un nombre de pins plus important fait disparaître les quatre condensateurs externes des pompes de charge. Cependant, le plus utilisé reste le MAX232 qui en prenant de l'âge a été "cloné" entre autres par HARRIS et TEXAS INSTRUMENTS.

Maintenant, si on revient aux fonctions utiles pour notre application, on pourrait très bien utiliser un MAX232 (solution d'autant plus que son prix assez faible n'incite guère à son remplacement).

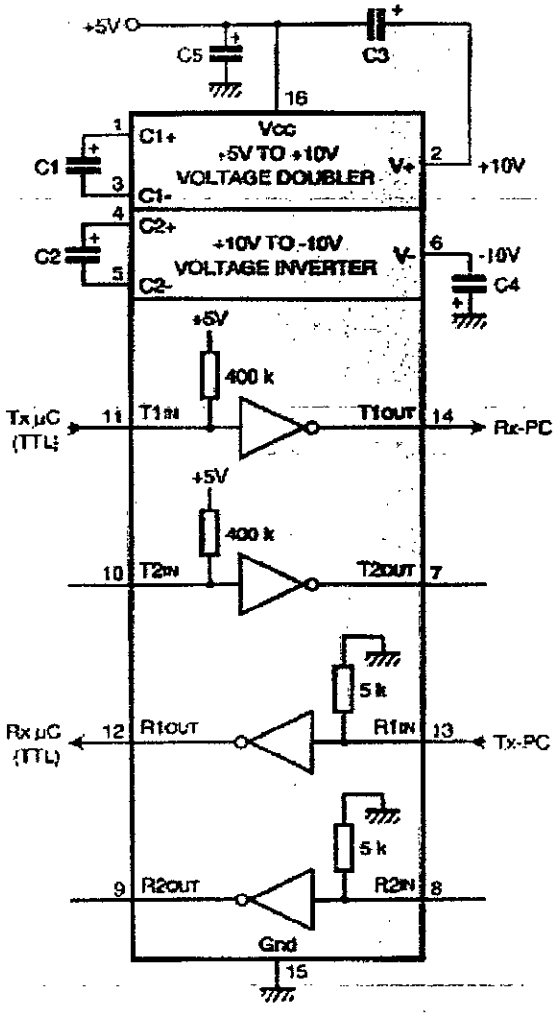


Fig.4.7. schéma interne d'un MAX232

Implémentation de la routine RS232 sur le PIC16F84 :

La routine RS232

Programme :

La programmation d'une routine RS232 revient à programmer une routine de réception et une routine de transmission. Dans notre cas l'automate est utilisé en mode esclave donc attend un signal de début de transmission de la part du pc ou de l'automate maître. Ce signal correspond au bit start, dés que celui-la est reçu une interruption survient au niveau du pic et le programme de réception commence à acquérir les bits sur la pin RB0.

Pour le premier échantillon on attend une durée correspondant à un bit et demi, pour les autres échantillons on attend une durée d'un bit.

Pour une configuration de 9600 bauds, la durée d'un bit et de 104µs, donc on attend 156µs au début et 104µs pour les autres bits.

La méthode utilisée pour détecter la fin de la durée de temporisation est le test du flag du registre intcon t0if. Ce bit est positionné lors d'un débordement du TMR0 qui doit être chargé par une valeur à déterminer X. Pour cela on devra calculer avec précision la durée de temps qui n'a pas été prise en compte par le TMR0 et la soustraire de 156. Cette valeur est utilisée dans l'instruction :

```
movlw - X

addwf tmr0,f
```

On trouve X=135 pour le bit start et X=93 pour les autres bits.

Gestion des erreurs :

L'automate maître envoie un mot instruction qu'on a choisi égal à h'0045' c'est-à-dire "E" en ASCII. Si notre automate reçoit ce mot instruction il envoie sur 2 octets l'état de ses entrée "7" et de ses sorties "4".S'il ne reçoit pas ce mot ou s'il détecte pas les 2 bits stops il positionne le flag d'erreur du registre d'état de la routine RS232, cela entraînera le positionnement à 1 du dernier bit des 2 octets envoyés pour signaler à l'automate maître qu'il y a eu une erreur de communication.

La routine de transmission a pour fonction le positionnement à 1 ou à 0 de rx "portA,2". Il envoie en tout début le bit start, suivi des autres bits du mot mémoire contenu dans le registre txreg, à des intervalles égaux.

La simulation :

Le simulateur nous propose plusieurs modes d'exécutions :

"Run" : Exécution de tout le programme.

"Step Into": exécution pas à pas ou le simulateur s'arrête après l'exécution de chaque instruction.

"Step Over": mode d'exécution semblable à Step Into à l'exception d'exécuter les macros en une seul fois.

MPLAB SIM propose la visualisation des diverses parties du PIC via l'onglet View: on choisira la fenêtre Watch qui permet de visualiser les variables et les registres spéciaux de notre choix.

L'outil debugger permet en outre d'ajouter des points d'arrêts à des endroits précis d'un programme pour permettre à l'utilisateur de visualiser le contenu des divers registres à une étape précise de son programme. Pour nous on choisira la fin de chaque cycle d'acquisition de l'entrée RB0 ou la fin de chaque cycle d'écriture sur RA2.

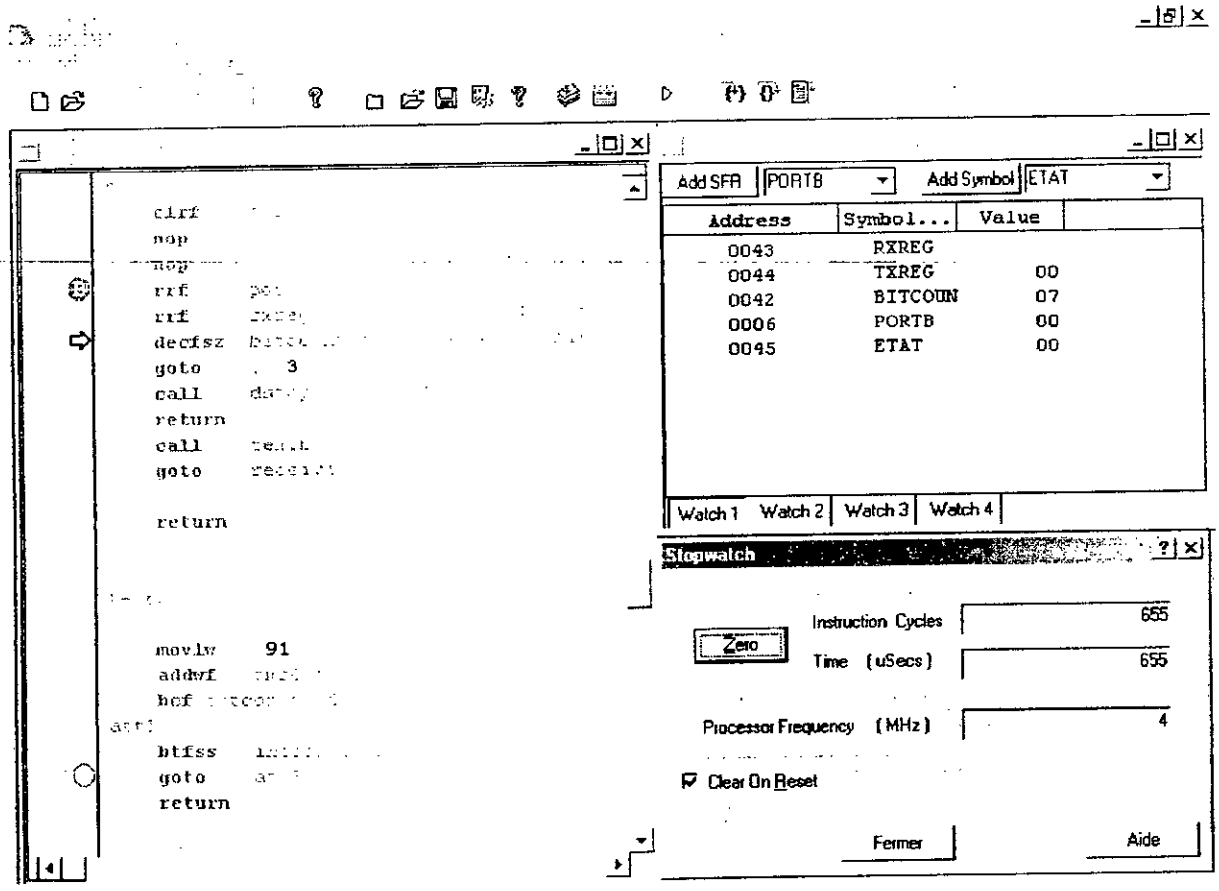


Fig.4.8. Ecran de simulation de la routine RS232

On utilisera l'outil de simulation des événements externes de MPLAB appelé Stimulus qui offre 2 modes d'utilisation, soit le changement du niveau d'une pin d'une façon asynchrone ou d'une façon synchrone ou charger un fichier qui contient une suite d'événements via l'onglet "File Stimulus".

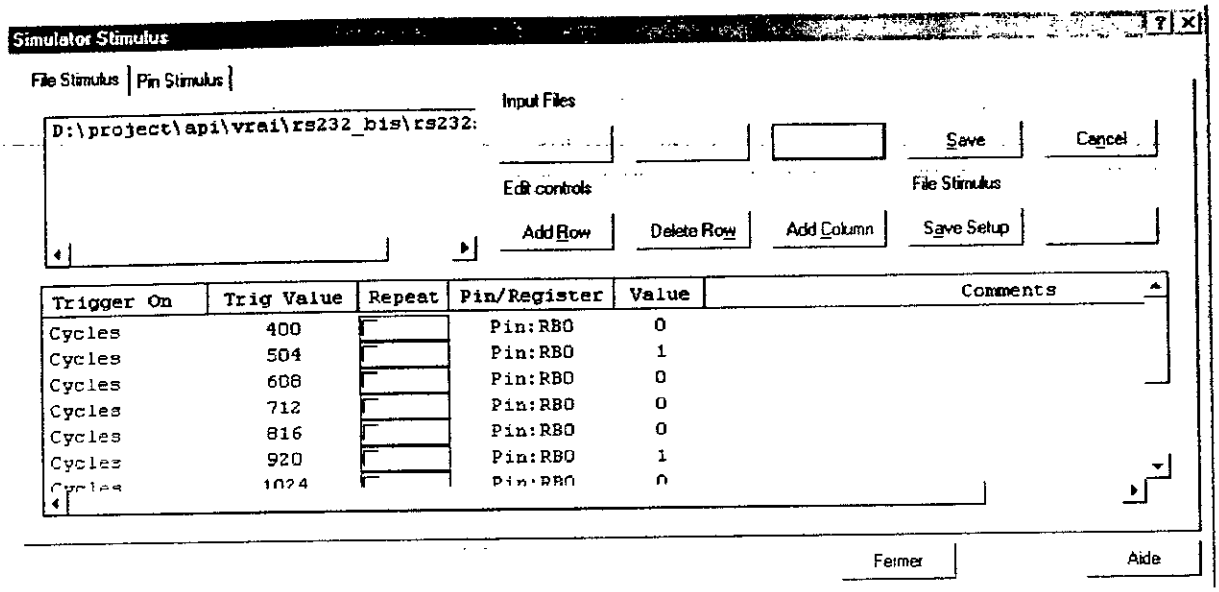


Fig.4.9. Outil de simulation des événements externes

Pour savoir les instants pour lesquels des actions s'ont produites il existe une petite fenêtre "Stop Watch" qui affiche le temps écoulé en secondes et en nombre de cycles.

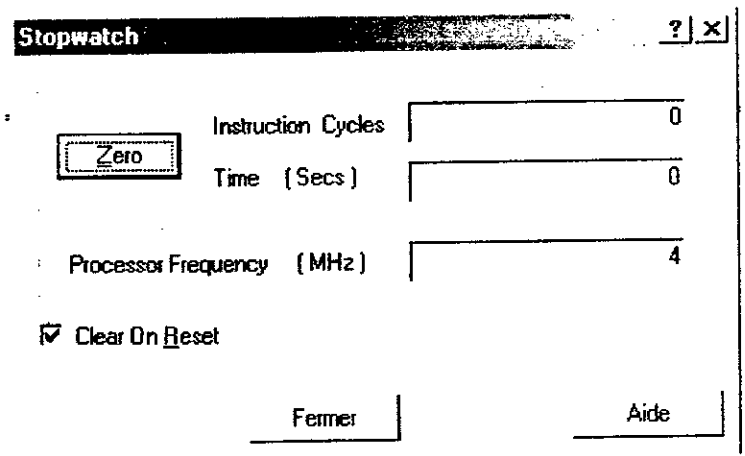


Fig.4.10. fenêtre d'affichage du temps écoulé

Pour la routine de réception on a relevé les instants d'acquisition et on les a comparé aux instants désirés on retrouve le tableau suivant :

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit stop
t (idéal)	556	660	764	868	972	1076	1180	1284	1440
t (relevé)	553	656	759	862	965	1068	1171	1274	1426

Tableau 4.1. Comparaison des instants d'échantillonnage en réception

On remarque que l'erreur et d'autant plus importante qu'on avance dans le temps puisque c'est une erreur additive. D'un autre coté l'erreur pour le dernier bit est de 10µs ce qui est correct.

Pour la routine d'émission on retrouve les résultats suivants :

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit stop1	Bit stop2
t (idéal)	104	208	312	416	520	624	728	832	936	1040
t (relevé)	104	204	307	410	513	616	719	822	921	1021

Tableau 4.2. Comparaison des temps de durée des bits en émission

On remarque que comme la réception le décalage entre les instants attendus et les instants relevés augmente avec le nombre de bits. L'erreur maximale est de 10µs.

2. La partie informatique au niveau du PC

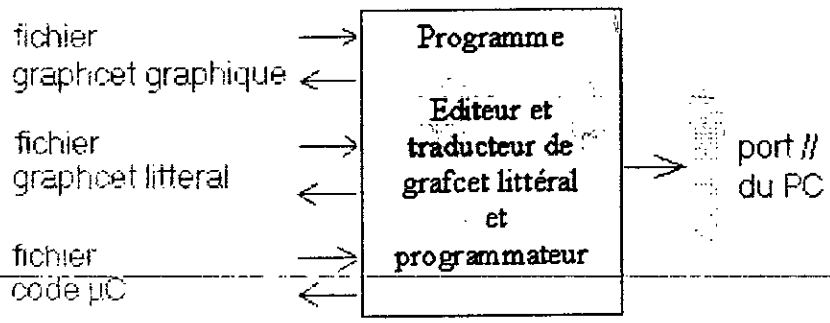


Fig.4.11. Schéma fonctionnel de la partie informatique au niveau du PC

Du point de vue informatique du cote du PC, un programme se charge de tout :
L'éditeur/traducteur de grafcet graphique et programmateur permet de :

- créer un grafcet graphique et de le transformer en grafcet littéral qui est une représentation textuelle du graphe.
- charger un grafcet littéral si vous boudez le graphique.
- programmer le uC avec le grafcet littéral plus l'interpréteur.

a. Rappel sur le grafcet : [8]

Vers les années 70, la complexité croissante des automatismes industriels nécessite de nouveaux outils de modélisation. Sous la direction de Michel Blanchard, le groupe *Systèmes Logiques* de l'AFCET "Association Française pour la Cybernétique Economique et Technique" a répondu à ce besoin en créant en 1975 le Grafcet, dont le principe et le graphisme sont inspirés des Réseaux de Petri (1962, par Carl Adam Petri, mathématicien allemand). A l'origine, le Grafcet est donc un outil de communication qui s'attache exclusivement à la description fonctionnelle des automatismes. Il a été normalisé en France en juin 1982 (NF C 03-190).

On peut utiliser 2 niveaux successifs de spécifications :

GRAFCET niveau1 : spécifications fonctionnelles. On décrit l'enchainement des étapes sans préjuger de la technologie.

GRAFCET niveau2 : on ajoute les spécifications technologiques et opérationnelles

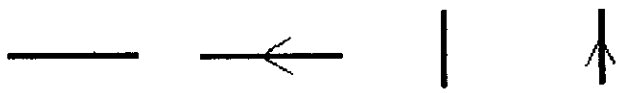
Conçu au départ comme outil de spécification du cahier des charges, le GRAFCET est devenu également un outil pour la synthèse de la commande et un langage de programmation des automates programmables

- définitions

Le GRAFCET (**GRA**phe **F**onctionnel de **C**ommande des **E**tapes et **T**ransitions) est l'outil de représentation graphique d'un cahier des charges.

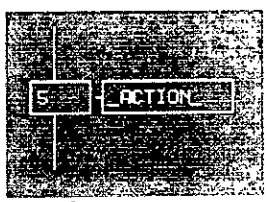
Un **Grafcet** est composé d'**étapes**, de **transitions** et de **liaisons**.

Une **LIAISON** est un arc orienté (ne peut être parcouru que dans un sens). A une extrémité d'une liaison il y a UNE (et une seule) étape, à l'autre UNE transition. On la représente par un trait plein rectiligne, vertical ou horizontal. Une verticale est parcourue de haut en bas, sinon il faut le préciser par une flèche. Une horizontale est parcourue de gauche à droite, sinon le préciser par une flèche.



Une **ÉTAPE** correspond à une phase durant laquelle on effectue une **ACTION** pendant une certaine DUREE (même faible mais jamais nulle). L'action doit être stable, c'est à dire que l'on fait la même chose pendant toute la durée de l'étape, mais la notion d'action est assez large, en particulier composition de plusieurs actions, ou à l'opposé l'inaction (étape dite d'attente).

On représente chaque étape par un carré, l'action est représentée dans un rectangle à gauche, l'entrée se fait par le haut et la sortie par le bas. On numérote chaque étape par un entier positif, même pour plusieurs liaisons partant de l'étape.



Une étape est dite active lorsqu'elle correspond à une phase "en fonctionnement", c'est à dire qu'elle effectue l'action qui lui est associée. On représente quelquefois une étape active à un instant donné en dessinant un point à l'intérieur.

Une **TRANSITION** est une condition de passage d'une étape à une autre. Elle n'est que logique (dans son sens Vrai ou Faux), sans notion de durée. La condition est définie par une **RECEPTIVITE** qui est généralement une expression booléenne (c'est à dire avec des ET et des OU) de l'état des CAPTEURS. On représente une transition par un petit trait horizontal sur une liaison verticale. On note à droite la réceptivité. Dans le cas de plusieurs liaisons arrivant sur une transition, on les fait converger sur une grande double barre horizontale, qui n'est qu'une représentation du dessus de la transition. De même pour plusieurs liaisons partant sous une transition.

b. Le logiciel PROGAPI :

- **Présentation :**

Cette réalisation utilise deux programmes :

1. Le programme d'interprétation au niveau du PIC.
2. Le logiciel de conversion de grafcet graphique, en grafcet littéral et de transfert du grafcet littéral dans l'automate au niveau du PC.

La figure suivante, montre cette organisation.

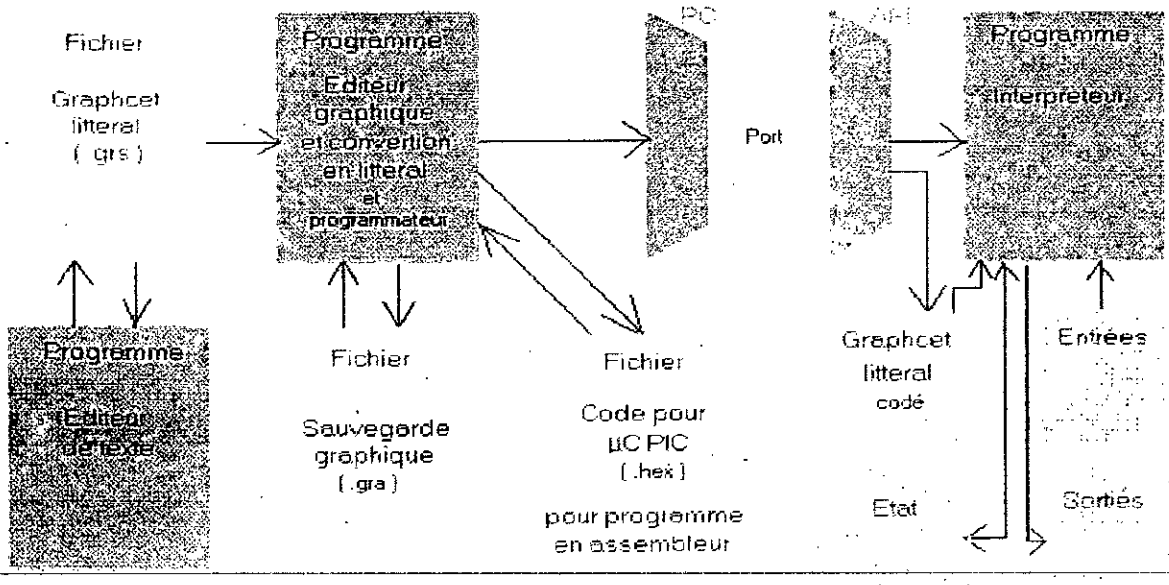


Fig.4.12. schéma fonctionnel du logiciel PROGAPI

Si on désire programmer en grafcet littéral alors, on doit éditer un fichier texte avec les instructions du grafcet littéral. Le nommer avec l'extension .grs . Lancer progapi.exe et choisir charger GRS dans le logiciel. Le programme convertit le grafcet littéral en code et ajoute l'interpréteur.

Si on désire programmer en graphique, alors il suffit de lancer progapi.exe et construire notre grafcet. Le programme transforme ensuite ce graphique en grafcet littéral. Modifier le grafcet littéral éventuellement puis le télécharger. Toutes ces opérations, mise à part la modification du grafcet littéral peuvent se faire à partir du programme progapi.exe.

Examinons maintenant les différents programmes

- **Principe de base**

Ce programme est chargé d'exécuter le grafcet au niveau de l'API. Il dispose d'une table ou est rangé le grafcet littéral à interpréter et d'une zone en RAM appelée zone des bits qui indique l'activité des entrées, sorties, étapes, bits internes et bits systèmes. Ces 2 derniers termes seront expliqués au niveau du grafcet littéral.
Son fonctionnement est le suivant.

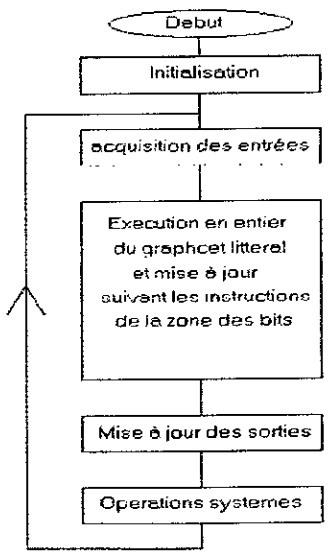


Fig.4.13. Mode de fonctionnement de l'interpréteur du grafcet

Les opérations systèmes consistent à mettre à jour les différents bits systèmes qui procurent des services à l'utilisateur, elles gèrent aussi les temporisations.
Ce programme a été fait en assembleur et est optimisé pour la vitesse.

Le grafcet littéral

Comme nous l'avons vu, le grafcet littéral est un petit langage qui est sensé représenter un grafcet.

Bases

Les étapes sont représenté par * pour les initiales et - pour les autres.
Pour changer d'étape, on utilise >, mais le changement est effectué seulement si l'étape dans lequel il est définit est active et si l'indicateur de test vaut 1.
L'indicateur de test est un bit avec lequel on peut faire les opérations suivantes :

- l bit : chargement de l'indicateur avec la valeur de bit
- ln bit : chargement de l'indicateur avec le complément de la valeur de bit
- a bit : et logique entre l'indicateur et le bit (indicateur = indicateur et bit)
- an bit : et logique entre l'indicateur et le complément du bit (indicateur = indicateur et /bit)
- o bit : ou logique entre l'indicateur et le bit (indicateur = indicateur ou bit)
- on bit : ou logique entre l'indicateur et le complément du bit (indicateur = indicateur ou /bit)
- x bit : ou exclusif entre l'indicateur et le bit (indicateur = indicateur xor bit)
- xn bit : ou exclusif entre l'indicateur et le complément du bit (indicateur = indicateur xor /bit)

Les transitions sont donc réalisées avec l'indicateur et l'instruction >.
= permet d'affecter un bit de la valeur de l'indicateur.
@ permet de mettre des commentaires

L'opérande bit indique le bit qui entre en jeu, il y a 256 bits possibles. On peut mettre directement leurs numéros, mais il faut les connaître et ce n'est pas pratique. Pour simplifier, ces numéros sont remplacés par une chaîne et un numéro il faut oublier de coller la chaîne et son numéro. Cette chaîne est un offset et chaque chaîne indique une zone particulière.

x: étape 0-63

i : entrée 0-31

o : sortie 0-15

bi : bit interne 0-31

bs : bit systmème 0-7

tc: commande de tempo 0-15

tf: indicateur de fin de tempo 0-15

Les bi pour bits internes, sont des bits laissés libres pour l'utilisateur, il peut les utiliser comme mémoire et jouer avec l'indicateur de test.
Les bs ou bits systèmes, sont des bits gérés par le uC qui fournissent des services a l'utilisateur, ainsi le bit bs7 est à "1" lors de la première exécution du grafcet littéral, il passe à "0" ensuite, les bits bs0 à bs6 clignotent afin de réaliser des voyants clignotants très simplement. bs0 clignote à 10Hz (1 pendant "50"mS et 0 pendant "50"ms) bs1 2 fois moins vite et ainsi de suite.
Les bits tc sont les commandes de lancement des 16 temporisations.
Les bits tf indiquent que la temporisation correspondante est terminée. Si le lancement d'une temporisation intervient dans une étape, alors elle doit être testée à la transition qui suit cette étape. Les temporisations sont définies entre 0 et 25.5 secondes au pas de 0.1s et sont précises à +/- 0.1s, donc 0.3 peut durer entre 0.2 et 0.4s.
La définition de la valeur d'une temporisation, se fait de la manière suivante : #t0

135, signifie que la valeur de la temporisation 0 est de 135 fois 100ms, soit 13.5s ici. Ces valeurs sont stockées dans l'EEPROM de donnée du PIC qui peut être reprogrammée 1000000 de fois, donc il n'est pas besoin de recharger le grafcet pour modifier uniquement une temporisation. La programmation permet de modifier les tempos, qu'on peut régler expérimentalement jusqu'a obtenir le résultat escompté. La faible taille de la RAM ne permet pas d'avoir plus de temporisations (bien que les valeurs soit en EEPROM, le décomptage se fait en RAM).

Méthode de programmation

Il est d'usage de respecter une programmation en 3 parties du type :

- PRETRAITEMENT
- TRAITEMENT
- POSTTRAITEMENT

Pré traitement

On y effectue souvent les opérations combinatoires complexes qui entre autre peuvent servir pour les transitions,

Traitement

Cette seconde partie s'occupe de l'évolution du grafcet, on y indique, la dynamique du grafcet, c'est à dire comment passer d'une étape à une autre avec les transitions, mais les sorties n'interviennent pas encore ici.

Post traitement

On affecte les sorties à ce niveau la.

L'interpréteur interprète tout le grafcet littéral dans l'ordre, puis affecte les sorties à la fin de cette interprétation.

Il n'y a pas besoin d'instruction pour séparer ces trois parties; on met dans l'ordre pré traitement suivi de traitement et enfin post traitement.

Divergence en "OU"

La divergence en "OU" ne pose pas de problèmes particuliers, si ce n'est qu'il ne faut pas oublier de la rendre exclusive.

Divergence en "ET"

Normalement il faudrait que la zone des bits ne soit modifiée qu'a la fin des passages d'interprétation, or les 68 octets de RAM du PIC 16F84 ne permettent pas de travailler sur une seconde zone des bits que l'on viendrait recopier sur la première après chaque traitement. En clair, dés qu'on modifie un bit ou on change d'étape, le bit correspondant est immédiatement modifié. Cela pose un problème, lors d'une convergence d'un "et". Pour remédier à cela il suffit de définir un bit interne qui vaudrait 1 si l'une des étapes de la divergence en "et" est active, utiliser par la suite ce bit interne dans la dernière transition de chaque branche.

Mode d'utilisation

Progapi est un logiciel permettant d'editer et convertir un grafcet en fichier hex utilisable sur des automates programmables a base de microcontrôleur PIC. Le grafcet peut être graphique ou litteral. (Un grafcet litteral est une représentation textuelle du graphique).

Le logiciel a besoin d'un fichier de définition (.def) qui décrit entièrement l'automate (code de l'interpréteur +nom des bits + emplacement mémoire pour le grafcet ...) Chaque automate a son fichier et l'utilisateur doit l'indiquer au programme En gros, voici le synoptique du programme :

Editeur --> GRAFCET GRAPHIQUE

| Convertisseur

V + fichier de def

Editeur --> GRAFCET LITTERAL

|

V Convertisseur

FICHIER HEX

Ce programme fonctionne sous Windows (95, 98, NT, 2000...),

Le fichier hex peut ensuite être programme avec un programmateur habituel

3. Le menu

Le menu se compose de familles (ellipses a gauche) et de fonctions (ellipses a droite). Pour sélectionner une fonction d'une famille, il faut cliquer sur la famille puis sur la fonction. Certaines fonctions affichent des écrans d'édition et d'autres lancent des traitements Voici l'arborescence du menu :

Fich Famille : gestion des fichiers

- LD G charger le grafcet graphique a partir d'un fichier

- SV G sauver le grafcet graphique dans un fichier

- LD L charger le grafcet littéral a partir d'un fichier

- SV L sauver le grafcet littéral dans un fichier

- CFG Indique les fichiers a utiliser

- Exit Pour sortir de l'application

GRA Famille : grafcet graphique

- Gril Edition du graph du grafcet graphique

- Etap Sorties à activer pour chaque étape

- Tran Equations des transitions

- Pre Equations de pre traitement

- Post Equations de post traitement

- Cst Valeur des constantes Lit

- Edit

Go Famille : grafcet litteral

- G->L Conversion du grafcet graphique en litteral

- L->H Conversion du grafcet litteral en fichier hex

- G->H Conversion du grafcet graphique en fichier hex

Help Famille : Aide

- Manu Indique l'adresse du site web ou sera le manuel

- Abt Nom du soft, etat, auteur ...

4. Utilisation

Famille FICH

[Fich|CFG]: A chaque ouverture, l'ecran [Fich,CFG] est affiche. L'utilisateur peut alors indiquer les fichiers suivant:

- Le fichier de définition de l'automate ex api1.def

- Le fichier du grafcet graphique ex: essai.gra

- Le fichier du grafcet littéral ex: essai.lit

- Le fichier hexadécimal ex: essai.hex

A la fin de l'application, les nom de ces fichiers seront sauves dans le fichier config.txt afin d'être restitues lors du prochain lancement de l'application

[Fich|LD G] L'utilisateur peut charger un fichier de grafcet graphique,

qui mettra a jour les écrans de la famille graphique. Le nom du fichier graphique est indique dans [Fich|CFG]

[Fich|LD L]: Idem en litteral

[Fich|SV G] Les écrans de la famille graphique seront sauvegarder dans le

fichier graphique sélectionne

[Fich|SV L]Le listing du grafcet littéral sera sauvegarder dans le fichier littéral sélectionne.

[Fich|EXIT] Pour quitter l'application

Famille GRA

[Gra, Gril]: Permet d'éditer le graph du grafcet sur une grille de dimension

40*40. Pour placer un symbole, sélectionner le (click gauche) dans la zone de sélection puis poser le sur la grille (click droit). Certains symboles ont des numéros, pour modifier ces numéros cliquer droit sur le symbole, taper le nombre puis appuyer sur entrée. Pour les transitions, ces numéros correspondent a l'équation portant le même numéro dans [Gra, Tran]. Pour les étapes ces numéros correspondent a ligne portant le même numéro dans [Gra, Etap] et qui indique les sorties a activer lorsque l'étape est active. Enfin les numéros des ancrages (Y) donnes des points de branchement aux symboles (V), ainsi un symbole V signifie que le grafcet continu a partir du symbole Y portant le même numéro. Bien entendu, il ne peut y avoir 2 fois le même numéro pour les transitions, étapes et ancrages.

[Gra, Etap] Permet de lister les sorties à activer durant chaque étape. Pour se déplacer dans la liste il suffit d'utiliser les touches de défilement.

[Gra, Tran] Permet d'éditer les équations des transitions.

[Gra, Pre] Permet d'éditer les équations des prétraitements. Ces équations seront exécutées avant d'exécuter le corps du grafcet, afin par exemple de simplifier les transition.

[Gra, Post]Permet d'éditer les équations des post-traitements. Ces équations seront exécutes après d'exécuter le corps du grafcet, afin par exemple de réaliser des traitement particulier sur les sorties

[Gra, Cst] Permet d'affecter des valeurs aux constantes, comme les valeurs des temporisation : ex t2 100 La temporisation t2 durera 100 unités de temps (100*100ms=10s).

Famille LIT

[Lit,Edit] Permet d'editer le listing du grafcet littéral

Famille GO

[Go, G->L] Convertit le grafcet graphique en grafcet littéral Le listing du grafcet littéral est donc écrasé avec le résultat de la conversion. Ce résultat n'est pas stocké dans le fichier littéral, il faut donc faire [Fich|SV L] on souhaite le sauver. Cette fonction n'indique aucune erreur.

[Go, L->H] Convertit le grafcet littéral en fichier hex directement exploitable par les programmateur de PIC C'est cette fonction qui signale les erreurs (mauvais nom de variables, mauvaises équations ...)

[Go, G->H] Convertit le grafcet graphique en fichier hex. Enchaîne directement [Go, G->L] et [Go,L->H]

Famille Help

[Help, Manu] Indique l'adresse du site web ou sera le manuel

[Help, Abt] Nom du soft, état, auteur ...

5. Les fichiers de définition :

Ces fichiers contiennent l'entière description des automates. Il est déconseille d'y toucher sauf pour changer le nom des variables Ces fichiers textes contiennent :

- Une zone de description pour donner quelques notes a l'utilisateur

- Des informations (emplacement du grafcet, place, uc...)

- Le nom des variables de la zone des bits ex : ox2 correspond a une sortie PORTA2 sur l'api0.

- Le nom des constantes

- Le code hex de l'interpréteur, car en fait les automates interprètent le listing du grafcet littéral, Le fichier hex de sortie est construit en ajoutant à l'interpréteur les instructions du grafcet littéral (y compris les constantes)

- Une zone libre a la fin ou il est bon de mettre les sources de l'interpréteur

Ainsi grâce a ces fichiers, il est possible de créer de nouveaux automates sans modifier le code du soft.

Les bits sont regroupes dans une zone de l'automate appelée zone des bits

Il y en a par exemple 256 pour les automates api0,0i,1,2. Chaque bit a un numéro auquel est associe un nom défini dans les fichiers de définition afin de rendre l'utilisation plus facile. Chaque bit a une fonction particulière

6. Fichiers

- progapi.exe : L'exécutable

- progapi.cpp : Les sources de l'exécutable

- api0.def : définition de l'automate 0

- Fichier.gra : le fichier graphique de l'exemple

- Fichier.lit : le fichier littéral de l'exemple

- Fichier.hex : le fichier hex de l'exemple

- config.txt : le fichier de configuration crée en sortant de l'application et qui contient les nom des fichiers de l'exemple

- mpasm.exe : assembleur qui permet d'assembler l'interpreteur

7. Autres

Pour une modification de l'interpréteur afin de l'adapter des besoins spécifiques alors il suffit de :

- extraire dans un fichier .asm la section ASM du fichier def.

- modifier le fichier asm pour l'adapter à vos besoins.

- assembler le fichier asm pour en faire un hex avec mpasm.

- Copier le fichier def de base dans un nouveau fichier def

- Copier le fichier hex dans la partie HEX du fichier nouveau def

- Copier le fichier asm dans la partie ASM du fichier nouveau def

- Modifiez les paramètres de la zone info. En général adr_start_grafcet, si le code prend plus de place et recouvre la zone à partir de cette adresse.

3. La partie informatique au niveau du µC :

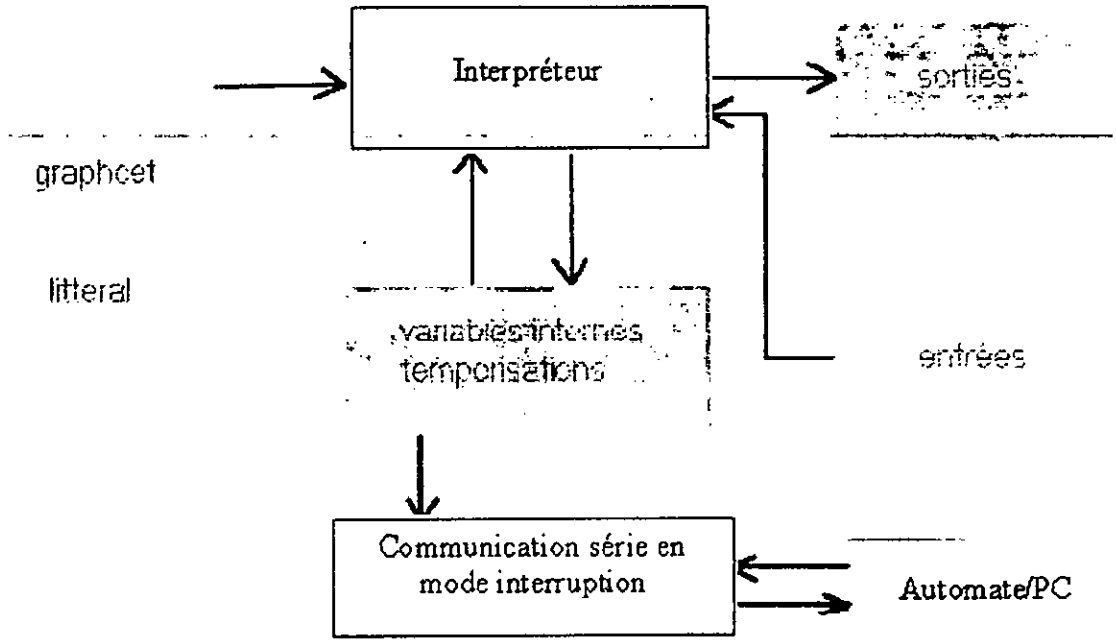


Fig.4.14. schéma fonctionnel de la partie informatique au niveau du µc

Du point de vue informatique du cote du uC, c'est un programme appelé **interpréteur de grafcet** qui gère l'automate en exécutant le grafcet littéral et en mettant à jour les entrées, sorties, variables internes et temporisations.

Intégration de la routine RS232 :

Comme on a vu précédemment la génération du fichier hexadécimal avec le logiciel PROGAPI utilise le fichier de définition [annexe2]. Le manuel d'utilisation de PROGAPI explique la procédure d'ajout d'un programme externe au fonctionnement de l'automate. Dans notre cas on veut intégrer une routine RS232. Lors de la première étape c'est-à-dire la compilation du fichier .asm à l'aide de Mpasm il faudra tenir compte des aspects suivant :

- La partie asm du fichier de définition ne tient pas compte du fichier P16f84.inc, et utilise ses propres assignations donc il est nécessaire d'adapter les assignations de la routine RS232 à celle déjà existantes comme opt___ au lieu de optionreg et ajouter celle qui n'existe pas comme les bits particuliers des registres option et intcon.
- Eviter tout conflit d'utilisation de la mémoire RAM, pour cela on doit répertorier toutes les variables du fichier de définition et leurs adresses; on obtient le schéma suivant :

Special Function Register	0x00 0x0b
Variables de l'interpréteur du grafcet littéral	0x0c 0x1f
Début de la zone des bits étapes X	0x20
X	
Fin de la zone des bits étapes X	0x27
iA	0x28
bi	0x29
bi	0x2a
bi	0x2b
ox	0x2c
bs	0x2d

tc	0x2e
tc	0x2f
tf	0x30
tf	0x31
Début de la 2 zone des bits internes bi	0x32
bi	
Fin de la zone des bits internes bi	0x3f
Zone mémoire vide	0x40 0x4f

Tableau 4.3 Mappage de la zone mémoire

Le choix de la zone mémoire se porte automatiquement sur la dernière zone entre 0x40 et 0x4f pour éviter la zone interpréteur et zone des bits.

- L'exécution de la routine RS232 ne doit pas changer non seulement la zone mémoire utilisé par l'automate mais aussi tout les registres spéciaux. En ce qui concerne les registres TRISA et TRISB les deux programmes utilisent la même configuration de ce coté on a pas de problèmes. Par contre la routine RS232 utilise une configuration du registre d'option qui permet une incrémentation du TMR0 beaucoup plus rapide que celle utiliser par l'automate. Pour cela on doit la restituer à la fin de l'exécution de la routine. D'un autre coté pour autoriser l'interruption sur RB0 il suffit de changer la configuration du registre intcon et mettre 0xb0 au lieu de 0xa0 utilisé précédemment par l'automate. Ce changement ne généra nullement le fonctionnement l'automate puisque il autorise l'ancienne interruption.

A cause de l'utilisation du TMR0 au cours du programme de communication il faudrait remettre à zéros le flag t0if en ajoutant la ligne :

```
Bcf   intcon,t0if
```

Le programme de communication entraînera une erreur au maximum de 65,536ms qui est de même ordre que l'erreur que commet l'automate lui-même "100 ms"

- La routine RS232 dure dans les cas les moins rapides quelques millisecondes, d'un autre coté l'arrivée d'un bit start nécessite une exécution rapide de la routine. Pour ces raisons on a choisi de donner la priorité à la routine RS232 lors de l'apparition d'une interruption en ajoutant le code suivant :

```
btfsc  intcon,intf
goto   initRB0
goto   itsave
```

Ou initRB0 début de la routine RS232, et itsave début de la routine d'interruption de l'automate.

- La zone programme destinée au grafcet littéral s'étend de 0x140 à 0x3FF pour éviter un recouvrement cette zone par le programme de l'interpréteur le programme de communication il suffit de la décaler. On choisit la valeur suivante : 0x180 à 0xFF

Après la prise en considération de tous ces paramètres il ne reste plus maintenant qu'injecter le programme de la routine RS232 au début du code du programme d'interruption et compiler la partie asm à l'aide de Mpasm. Par la suite poursuive la procédure d'ajout d'un programme au fichier de définition. Le fichier de définition finale est dans l'annexe 3

Remarque :

- En examinant attentivement la routine d'interruption finale, on peut se poser la question de savoir si une interruption de RB0 qui survient au cours de l'exécution de l'interruption timer poserait-elle problème?. En calculant le temps de retard en trouvera qu'il est égal dans les pires des cas à 90 µs et posera problème seulement pour les communications à 9600 bauds. Pour les autres vitesses il ne posera pas de problème. Pour remédier à ça il suffit :

_ En ajoutant un test du bit intf au milieu du programme d'interruption de TMR0 et positionner le flag du registre d'état de la routine RS232 si le test est affirmatif, l'effacer lors de l'exécution du programme d'interruption RB0 de cette façon le programme saura qu'il y a eu un temps perdu.

_ diminuer le temps d'attente lors du premier échantillon de 100 µs au lieu des 156 µs habituelles.

- prendre en compte le changement du nombre d'instructions au début de la routine d'interruption pour la temporisation du premier bit.

Exemple d'utilisation de l'automate à base de PIC 16F84 :"gestion d'une chaudière"[16]

A. **Principe de fonctionnement d'une chaudière :**

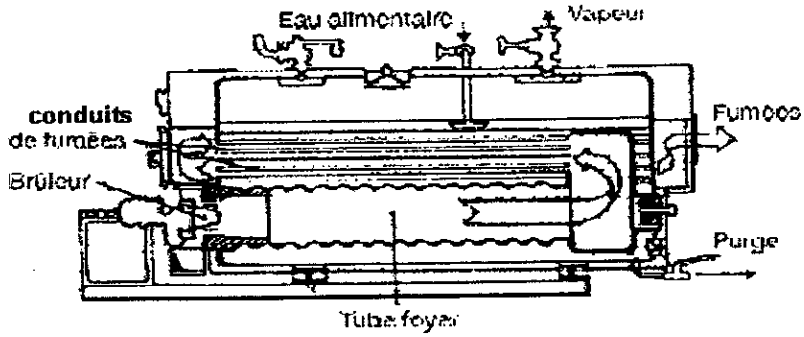


Fig.4.15 schéma simplifié d'une chaudière

Le tube foyer, qui se trouve dans le ballon même de la chaudière, sous le plan d'eau, collecte les gaz chauds en sortie de brûleur. Les gaz chauds, accumulés dans un premier caisson à l'arrière de la chaudière, sont véhiculés par un groupe de tubes immergés dans l'eau du ballon vers un second caisson à l'avant de la chaudière. Un second groupe de tubes immergés emmène les gaz vers un troisième caisson à l'arrière de la chaudière ; ce troisième caisson débouche sur la cheminée pour l'évacuation des fumées vers l'extérieur.

Il y a donc circulation des gaz de combustion dans des tubes assurant, par conduction vers l'eau de la cuve, la vaporisation par apport de calories.

B. Mise en marche :

L'opérateur doit passer par deux étapes :

- La première étape consiste à passer en mode manuel et actionner la pompe jusqu'à dépasser le niveau bas.
- La deuxième étape consiste à passer en mode automatique, à ce moment le séquenceur contrôle tous les défauts possibles (pression fuel, niveau très bas, dérangement brûleur, défaut ventilation, etc....). Après cette séquence de vérification le brûleur est allumé et deux régulations démarrent simultanément :

Régulation du niveau d'eau : Un niveau d'eau trop élevé augmentera la pression et l'eau restera à l'état liquide. D'un autre coté un niveau d'eau trop bas entraînera une dégradation du tube foyer du fait que la chaleur des gaz n'a pas été transmise à l'eau.

Régulation du niveau de pression : Ce niveau doit rester dans une certaine plage est dépends de la consommation des divers organes de l'usine, exemple entre 5 et 7Bars.

Tout défaut jugé important entraînera immédiatement l'arrêt du brûleur, déclenchement d'une alarme sonore et visuelle et demande acquittement auprès de l'opérateur après vérification et réglage du défaut.

C. Analyse du fonctionnement de la chaudière :

On peut résumer les états possibles de la chaudière par :

- Etape initiale : attente d'une commande de l'opérateur
- Etape 1 : pompe en marche en mode manuel
- Etape 2 : pompe à l'arrêt en mode manuel
- Etape 3 : séquenceur et pompe en marche en mode automatique
- Etape 4 : séquenceur et pompe à l'arrêt en mode automatique.
- Etape 5 : séquenceur à l'arrêt et pompe en marche en mode automatique.
- Etape 6 : séquenceur en marche et pompe à l'arrêt en mode automatique.
- Etape 7 : séquenceur et pompe à l'arrêt en mode automatique et pas de défaut.
- Etape 8 : Etape intermédiaire

Les entrées du système sont :

- Entrée 1 : Niveau d'eau bas
- Entrée 2 : Niveau d'eau haut
- Entrée 3 : Niveau bas pression
- Entrée 4 : Niveau haut pression
- Entrée 5 : défaut
- Entrée 6 : Mode manuel/automatique
- Entrée 7 : Acquittement

Les sorties du système sont :

- Sortie 1 : Commande séquenceur
- Sortie 2 : Commande Pompe
- Sortie 3 : Allumage lampe défaut
- Sortie 4 : klaxon

D. Dessiner le schéma du grafcet à l'aide de PROGRAPI

Après établissement du cahier des charges de la chaudière (entrées, sorties, transitions, étapes) on va maintenant préciser les fonctions des bits de la zone des bits :

Bit	Etat1	Etat0
ia0	Non utilisé	Non utilisé
ia1	Niveau au dessus du bas	Niveau au dessous du bas
ia2	Niveau au dessus du haut	Niveau au dessous du haut
ia3	niveau bas pression	En dessous du niveau
ia4	Niveau haut pression	En dessus du niveau
ia5	défaut	Pas de défaut
ia6	manuel	automatique
ia7	Acquittement	Pas d'acquittement
ox0	Commande séquenceur	Décommander séquenceur
ox1	Commande pompe	Décommander pompe
ox2	Non utilisé	Non utilisé
ox3	lampe défaut allumé	Lampe défaut éteinte
ox4	Klaxon en marche	Klaxon à l'arrêt

Tableau 4.4 Désignation de la zone des bits ia et ox

Dans la sous-famille Gril on dessine le grafcet :

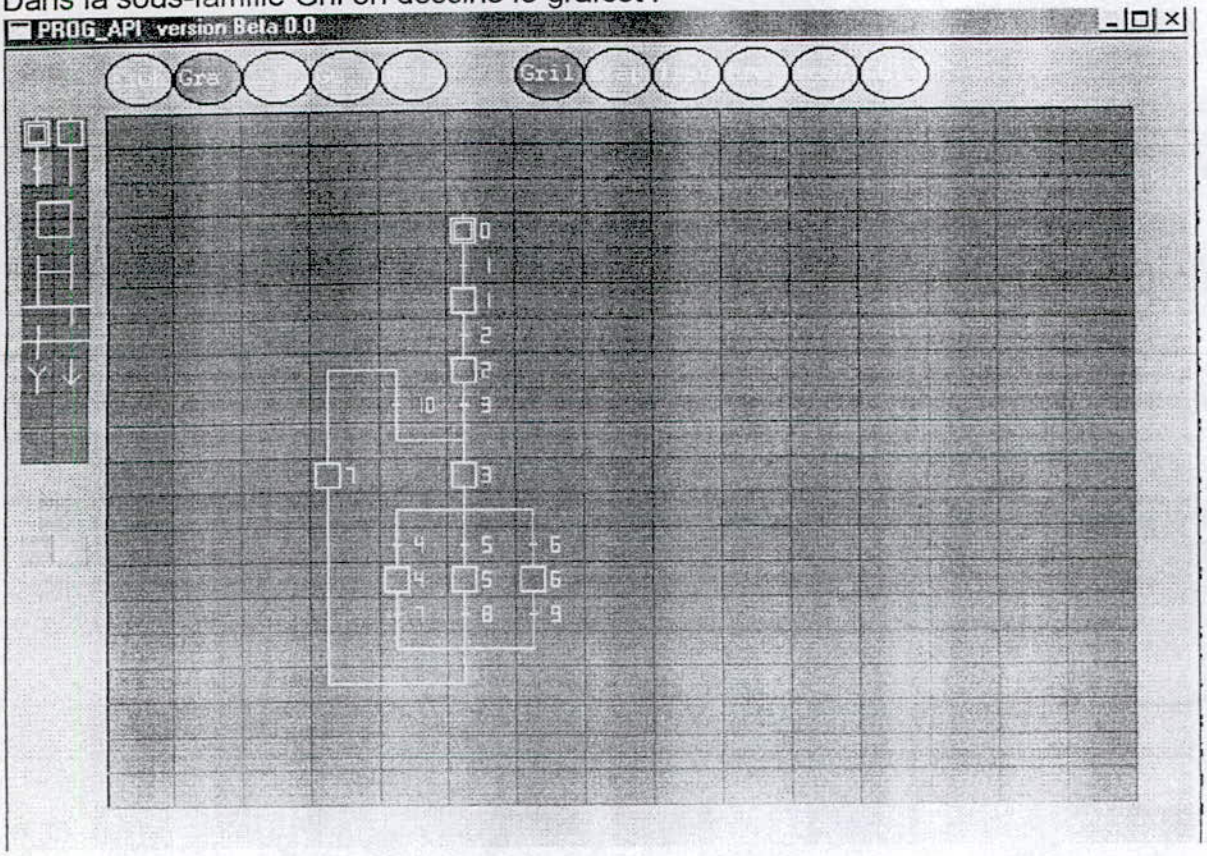


Fig.4.16 Schéma grafcet à l'aide de PRGAPI

Dans la sous-famille étape, on affecte à chacune des étapes les sorties qu'on veut activer :

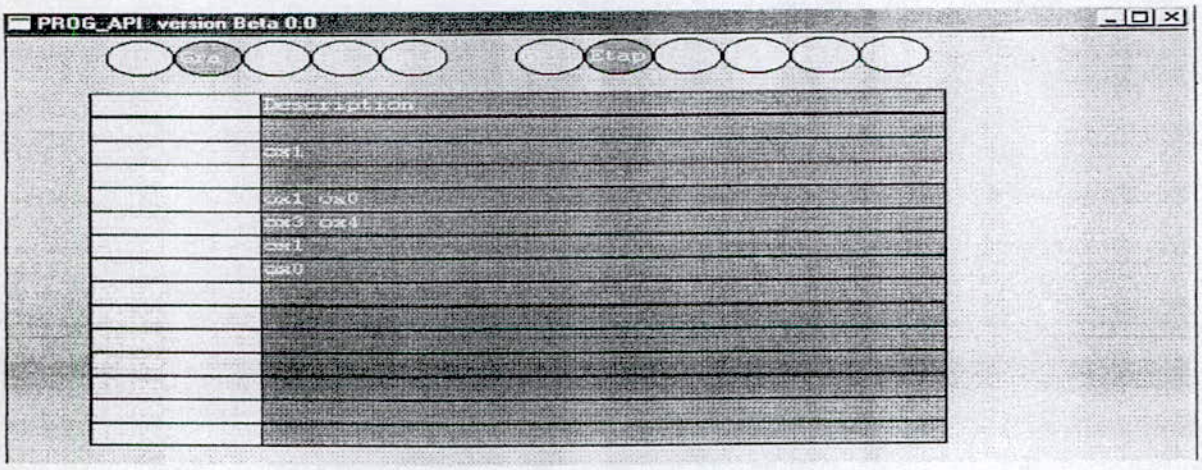


Fig.4.17. Visualisation des sorties sur le logiciel PROGAPI

De même que les étapes, on chargera pour chaque transition l'équation du bit test :

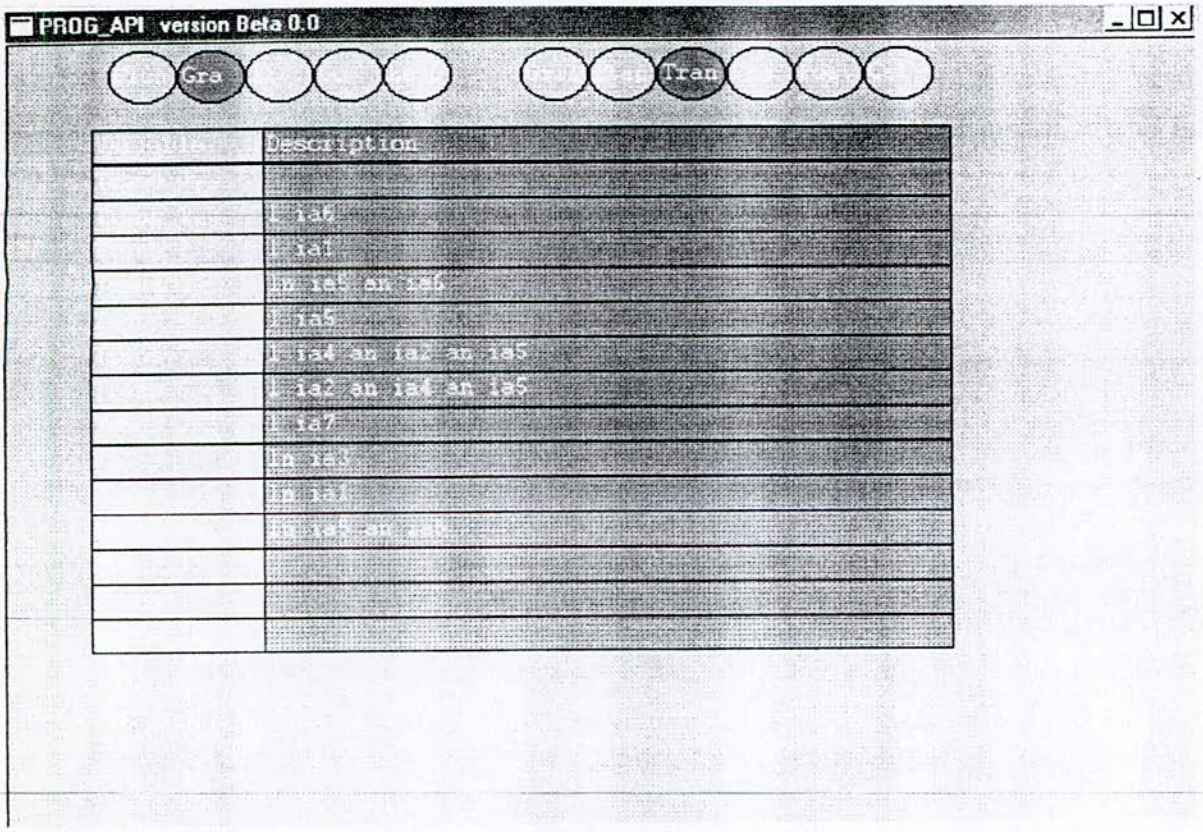


Fig.4.18. visualisation des transitions sur le logiciel PROGAPI

Maintenant il nous reste plus qu'à lancer la conversion du grafcet graphique à un fichier hexadécimal qu'on peut directement injecter dans le PIC.

APV prévoit la mise sur le marché d'un nouveau produit : I-CE fonctionnant sous Windows CE. Qui va réduire les coûts d'installations d'une manière considérable.

Les caractéristiques techniques du modèle I-NT-1-XXX sont :

- Processeur AMD K6 266MHZ
- 64MB RAM
- Disque dur 2.1GB
- Connexion Ethernet 10BaseT en standard
- 4 Ports séries RS232
- 2 ports parallèles
- Connecteurs VGA, LCD, clavier et souris
- 1 Slot d'extension au format PC104

- en option carte Profibus ou Bitbus PC104
- Windows NT 4 + SP 5
- InControl 7.1 "Runtime+" (Edition config.locale)
- En option le firmware ACCOS

2. Description de la partie logicielle :

Comme tout automate l'ACCOS est associé à un ensemble de logiciels indispensables pour l'utilisation de celui-ci. On distingue un logiciel de supervision et de visualisation "OGS", Un autre d'édition et de compilation du programme de l'automate lui-même "ADE" et enfin un utilitaire qui fait la liaison entre l'environnement extérieur et le programme de l'automate "Node".

a. Présentation générale :

Malgré la multiplication des fonctionnalités exigées pour la gamme de logiciels associés à un automate, celle de l'ACCOS offre d'un coté à l'opérateur une interface de supervision et de commande conviviale et au programmateur un environnement riche et facile d'exploitation.

On peut modéliser le fonctionnement et l'interaction de ces logiciels par le schéma suivant :

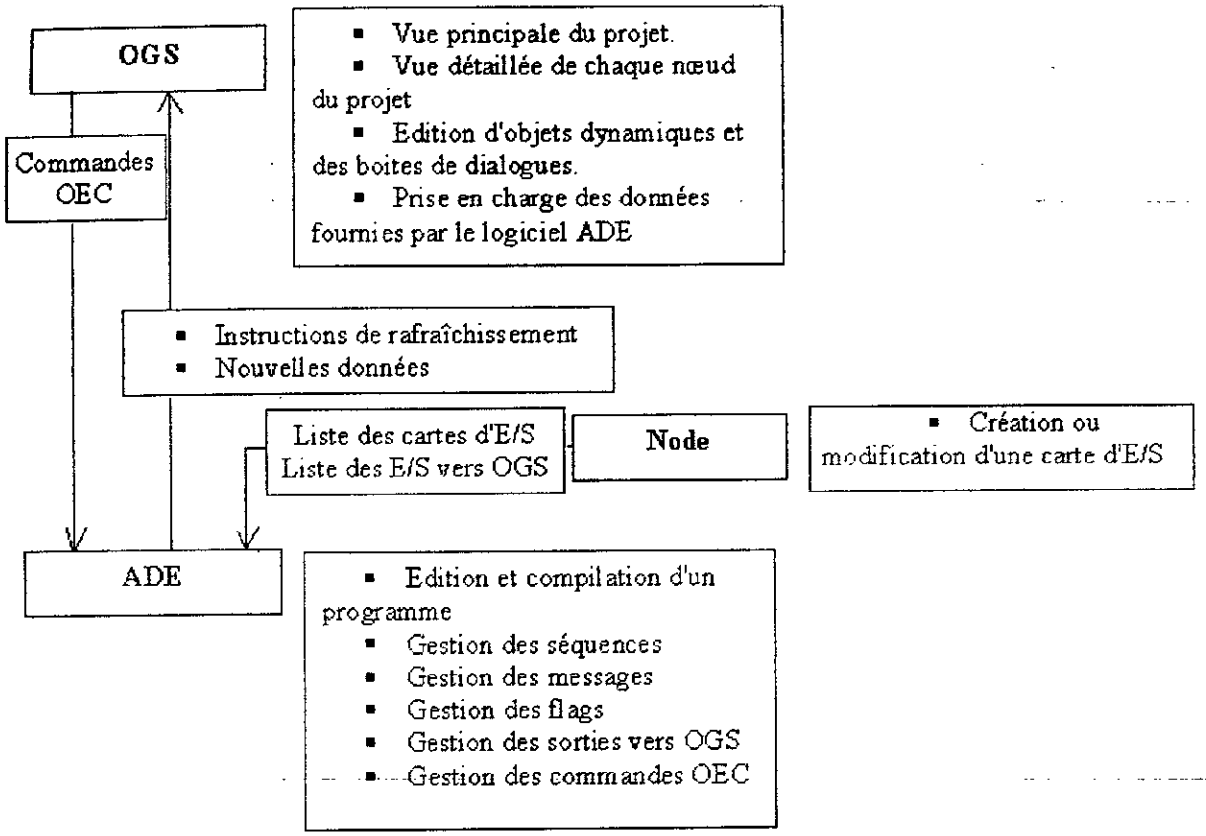


Fig.5.1. Principe de fonctionnement et d'interaction entre les divers composants logiciels d'APV

b. L'utilitaire Node :

NODE (NORMALIZED DOCUMENTS EXPERTS) est un logiciel de nomenclature de composants d'installations agro-alimentaires automatisés.
La vocation actuelle de NODE est d'être utilisé pour indiquer quels sont les matériels employés et de quelles manières ils sont reliés afin de garantir que les différents composants de l'installation sont correctement câblés. Le logiciel NODE est développé pour l'édition de documents normalisés. La création de listing d'Entrées/Sorties des automates pour le câblage électrique fut le premier module implémenté.

NODE se veut être un suivi du matériel employé ainsi qu'un descriptif des éléments selon des normes précises. NODE supporte les systèmes ACCOS 30 et ACCOS 31P,

ainsi que les matériels associés tels que les Racks GPIO, les Racks Accos 2, les Racks Accos 2S et les cartes d'entrées/sorties.

c. L'ADE :

ADE "Automation Development Environment" est un outil indispensable pour modifier et injecter un programme dans l'automate ACCOS. Il offre plusieurs avantages comme la génération automatique des fichiers de références, génération automatique des commentaires, génération automatique de la documentation.
Les programmes d'un automate ACCOS sont très structurés, et doivent se baser sur le modèle d'exécution d'un programme dans l'automate ACCOS qu'on peut représenter par le schéma suivant :

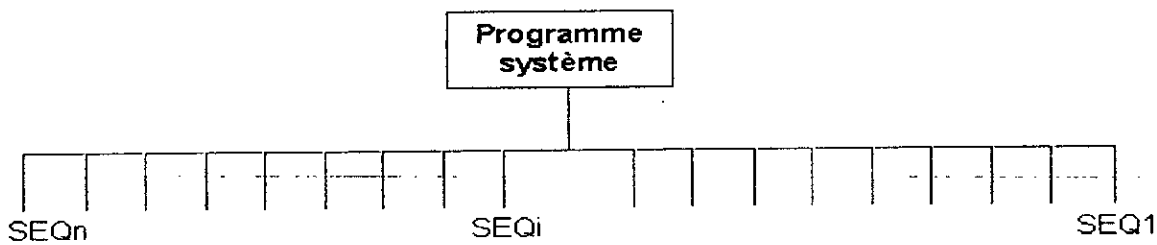


Fig.5.2. Modèle d'exécution

Le programme système exécute plusieurs séquences en parallèle et si l'une d'elle se termine alors elle n'est pas redémarrée automatiquement. On distingue les types de variables suivants :

- Séquences
- Fichiers
- Messages : Ces messages sont utilisés pour répondre à une commande OEC donnée par l'opérateur
- Timers
- Flags permanents
- Flags temporaires
- Registres flottants

- Registres entiers
- Eléments : sont des variables avec lesquelles le programme ADE peut communiquer avec le programme OGS.

Chaque séquence en plus de sa fonction bien spécifique, devra réaliser un ensemble de taches pour se synchroniser avec les autres séquences, positionner des flags pour informer le programme système et par conséquence les autres séquences de son état d'avancement, récupérer les commandes OEC qui viennent de l'opérateur, s'il y a un défaut positionner le flag correspondant dans la zone des variables globales et envoyer un message défaut à l'imprimante pour archiver le listing des défauts afin de l'utiliser ultérieurement.
Le langage utilisé pour programmer les séquences est le "PARACODE"

d. L'OGS :

C'est un logiciel puissant de programmation objet dédié à l'agroalimentaire. Il présente une palette d'objets : objets prédéfinis, formes diverses, cuves, vannes,... à travers un inspecteur d'objets.

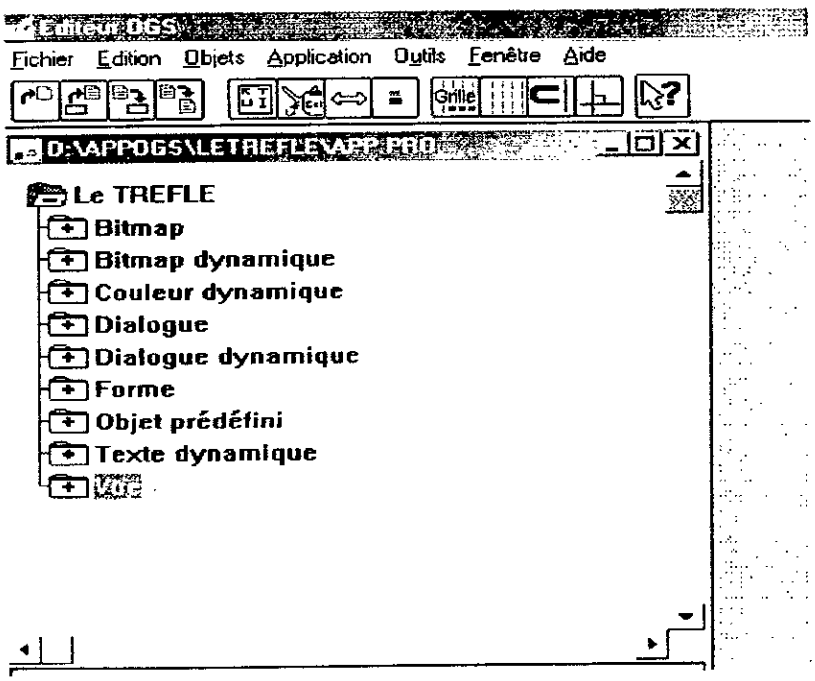


Fig.5.3. Ecran du logiciel OGS

Il offre la possibilité de programmer une vue globale de toute l'usine, et de programmer par la suite la vue détaillée de chacune de ces parties.

V. Implémentation du programme de communication dans l'automate ACCOS :

Faire communiquer l'automate à base d'un PIC 16F84 avec l'automate ACCOS nécessite la connaissance du programme système, ou les fonctionnalités cachées de celui-ci. Or celles-ci ne sont pas disponibles malheureusement. Mais d'un autre coté l'ACCOS offre la possibilité de simuler des programmes, donc on a décidé de programmer la routine RS232 sous forme d'une séquence, récupérer les données de l'automate esclave et rafraîchir l'écran de l'opérateur par les nouvelles données. Le programme RS232 va utiliser une entrée et une sortie logiques de l'automate comme étant Rx et Tx et lors de la simulation l'ACCOS va nous demander lorsqu'il a besoin de lire une entrée il nous donne la main afin de saisir la valeur qu'on veut lui assigner.

Remarque :

Les entrées et sorties de l'automate ne sont pas destinées à la communication mais elles ont été utilisées ici pour des raisons de simulation seulement.

Pour implémenter ce programme il faut suivre les étapes suivantes :

1. Première étape :

Création d'une nouvelle séquence à l'aide de l'utilitaire de visualisation et gestion des variables qu'on va nommer "SUPERVISIN DE LA CHAUDIERE", choisir le matériel qu'il lui est associé pour notre cas SC1 et choisir un numéro : 3200. On obtient dans la zone d'affichage des variables ce qui suit :

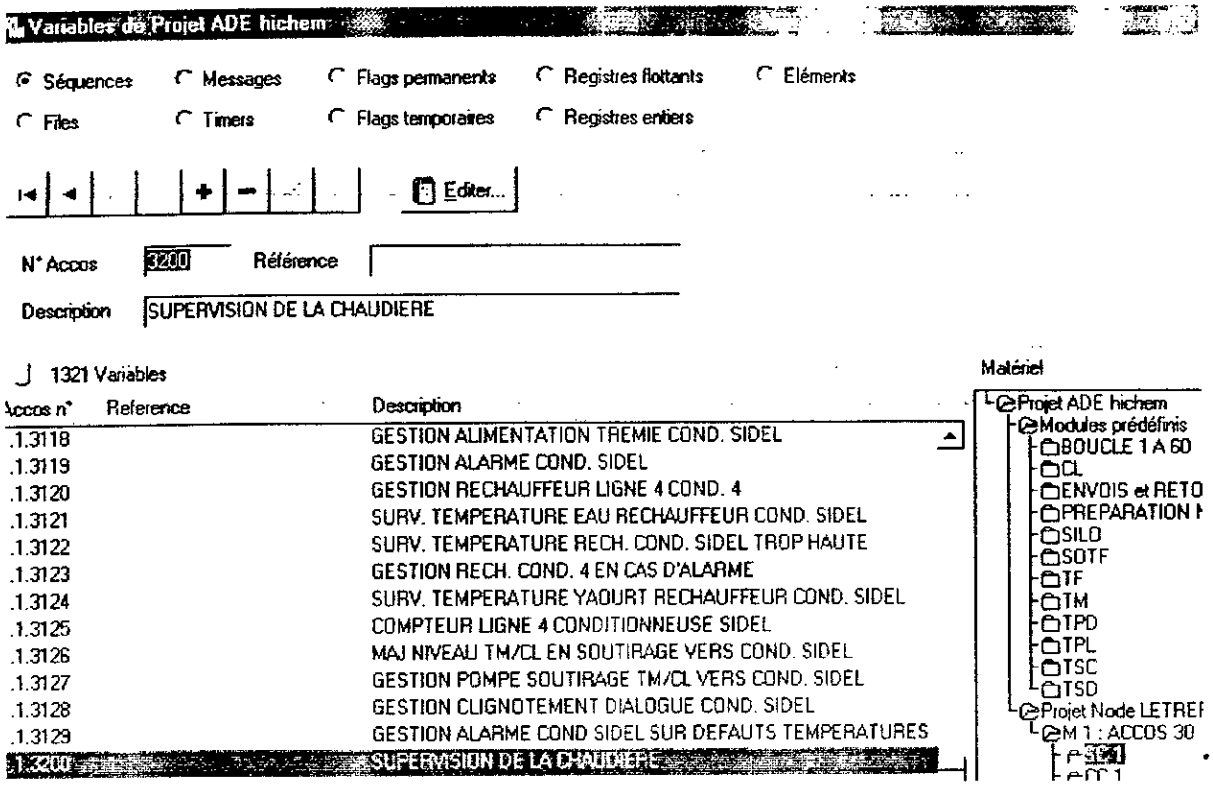


Fig.6.1. fenêtre de visualisation des différentes variables du projet "LETREFLE"

Par la suite on édite le programme de la routine RS232, et on le sauvegarde. On obtient le code de l'Annexe 7.

La première étape consiste à émettre, en utilisant le protocole RS232, un mot instruction sur un octet qui est le caractère "E" et correspondant à H"0045". L'automate esclave, à la réception de ce mot instruction, émet à son tour l'état de ses entrées et sorties en 2 octets en positionnant le dernier bit à 1 en cas d'erreur de communication. A la réception de ces deux octets l'ACCOS commence l'étape de commander ou décommander les éléments qui font office d'interface avec l'OGS en suivant les valeurs des bits reçus.

2. Deuxième étape :

Cette étape consiste à programmer l'interface de l'utilisateur. On commence à insérer un nouveau nœud qu'on va nommer "Supervsion_chaud", insérer les objets dont on a besoin dans la fenêtre courante et on obtient la vue détaillée de la chaudière :

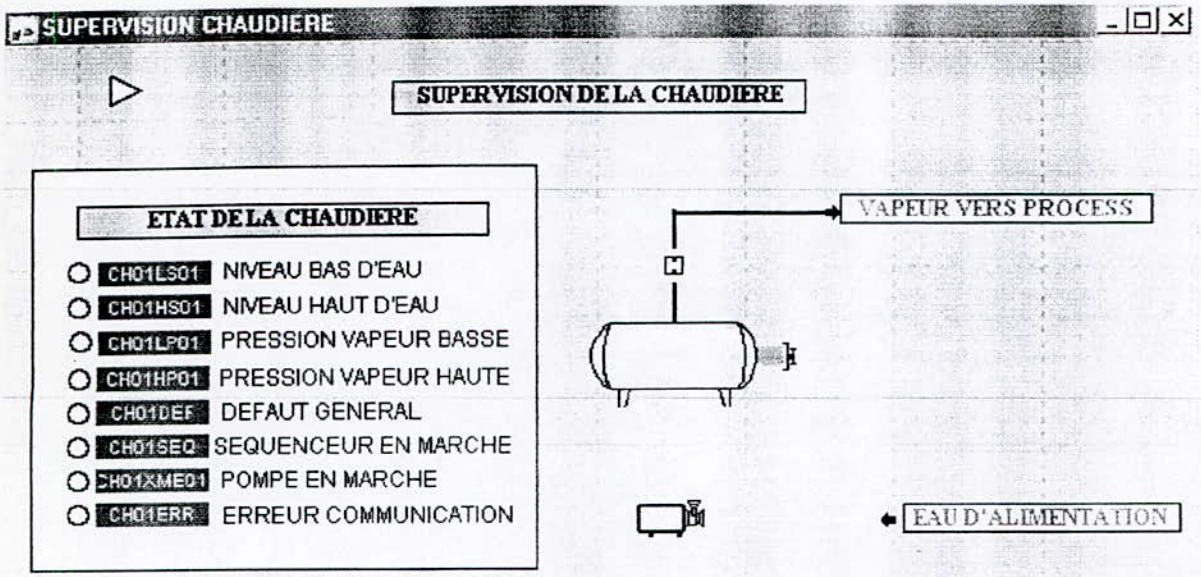


Fig.6.2. Fenêtre de visualisation de l'application "supervision chaudière"

Les expressions des indicateurs doivent correspondre aux noms des éléments qu'on a donné préalablement dans le programme ADE.

Par la suite en intégrera cette vue dont la vue générale du projet TREFLE en l'insérant directement dans l'onglet vue principale. On obtient la figure suivante :

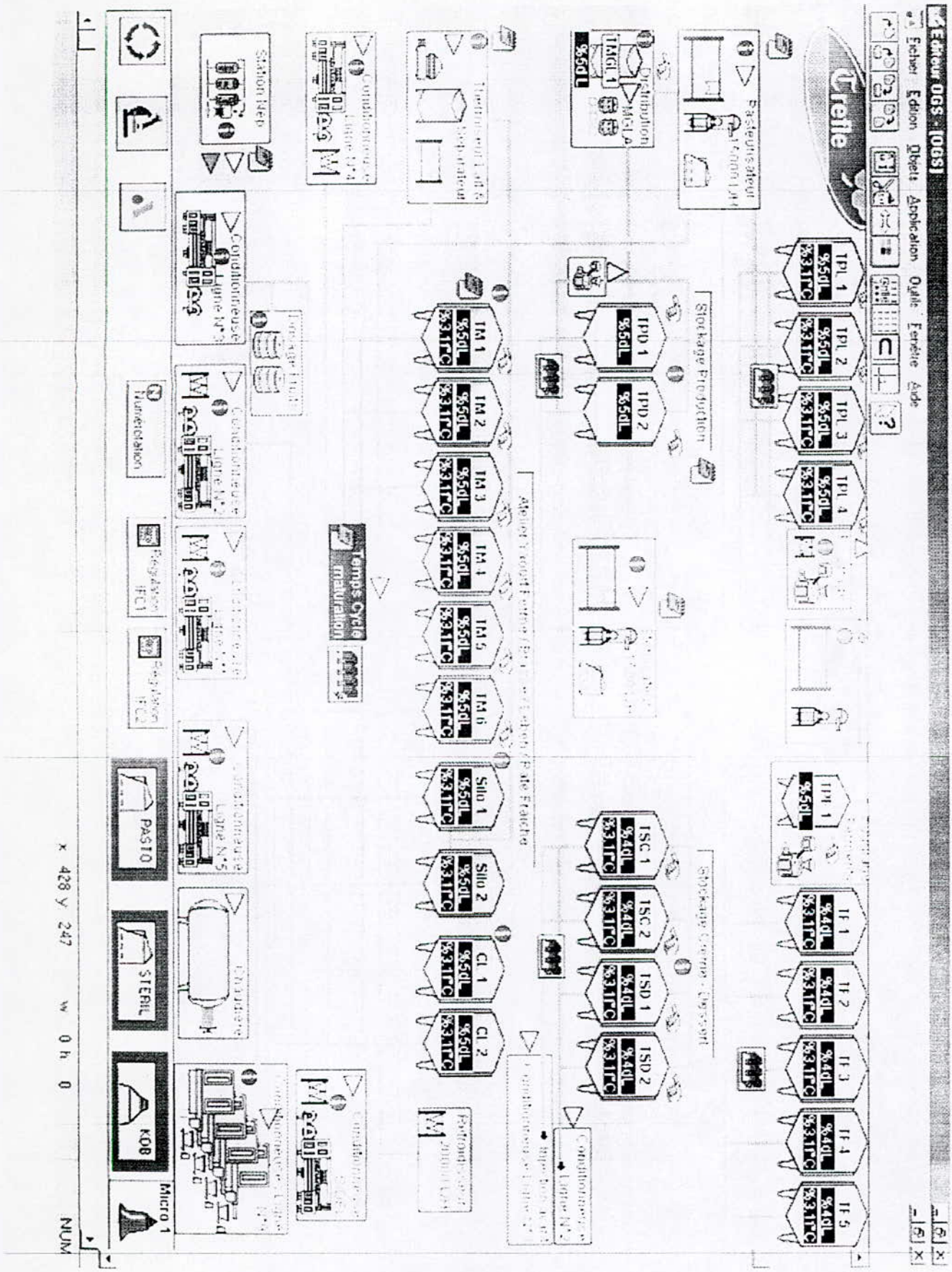


Fig.6.3. Vue globale de l'application "LETREFLE"

Remarque :

Pour des raisons de sécurité et de bon fonctionnement il est conseillé d'ajouter un bouton de prise en charge de la chaudière ou non ▷ . Pour que lorsque la chaudière est hors service, l'automate ne bloque pas et continue de fonctionner normalement.

On doit le paramétrer en spécifiant la boite de dialogue qui doit l'activer, cette boite dialogue doit être créer préalablement. Elle a la forme suivante :

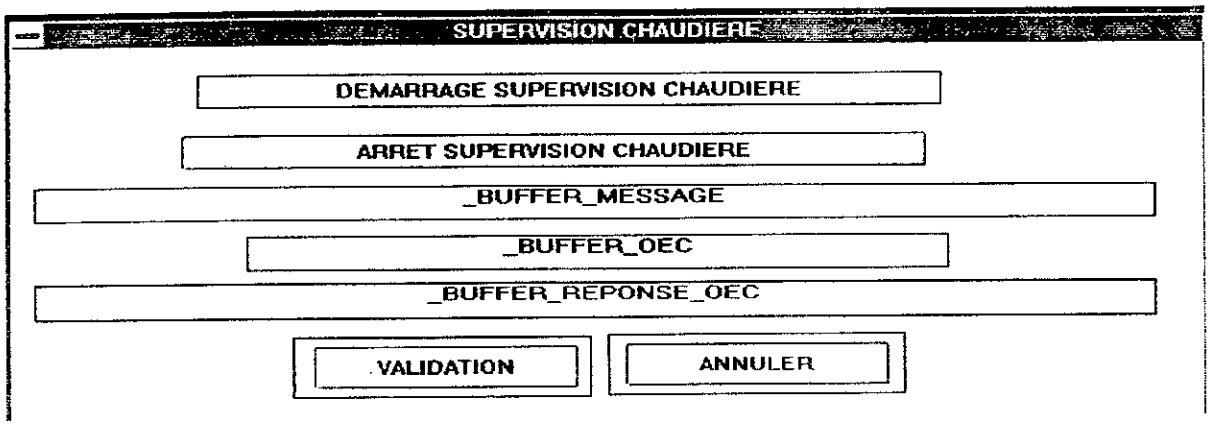


Fig.6.4. boite de dialogue prise en charge chaudière ou pas

On doit spécifier le libellé du flag à positionner pour informer le système qu'on veut ou pas prendre en charge la supervision de la chaudière.

3. Troisième étape :

Modifier les séquences 1.1.51 intitulé "T.O.T OEC" et 1.1.52 pour prendre en charge la demande lancée par l'opérateur. Et créer une nouvelle séquence 1.68 intitulé "OEC CHAUDIERE" pour gérer la demande OEC de la vue "Supervision_chaud" cette séquence est présentée dans l'annexe 8. On retrouve deux parties :

- La première qui teste si la séquence est active ou pas pour éviter par exemple que l'opérateur n'active le programme de supervision alors qu'il est déjà actif.
- Le deuxième standard pour positionner quelques flags nécessaires au bon fonctionnement de l'automate.

Conclusion

La communication entre les automates prend une importance capitale de nos jours, elle permet de coordonner leurs taches, synchroniser leurs fonctions et commander toute une usine à partir d'un seul terminal.

L'évolution rapide des systèmes de communication et l'apparition des diverses normes de communication industrielles qui les ont accompagnées; n'ont fait qu'accélérer l'intégration de ces nouvelles technologies dans le domaine des automates.

La circuit réalisé dans ce travail peut être amélioré en utilisant le multiplexage des entrées et sorties pour augmenter la capacité d'acquisition de l'automate, d'un autre coté un jeu d'instructions évolués entre les deux automates offrira d'autres possibilités à l'utilisateur en termes de changements du mode de fonctionnement de l'automate esclave; changer les valeurs des temporisations...

Le travail réalisé dans ce projet de fin d'étude met en évidence tout l'intérêt de faire communiquer deux automates. La communication a été faite en utilisant la routine RS232. Mais on peut améliorer le mode de communication en utilisant un réseau Ethernet et implémenter le protocole TCP/IP sur l'automate à base de PIC 16F84, d'autant que l'automate ACCOS prend en charge ce protocole.

Liste des annexes

ANNEXE1: Le circuit imprimé de l'automate à base de PIC 16F84

ANNEXE2 : Implantation des composants de l'automate à base de PIC 16F84

ANNEXE3 : La routine RS232

ANNEXE4 : Le fichier de définition original de u-delmas Api0.def

ANNEXE5 : Le fichier de définition modifié Api0c.def

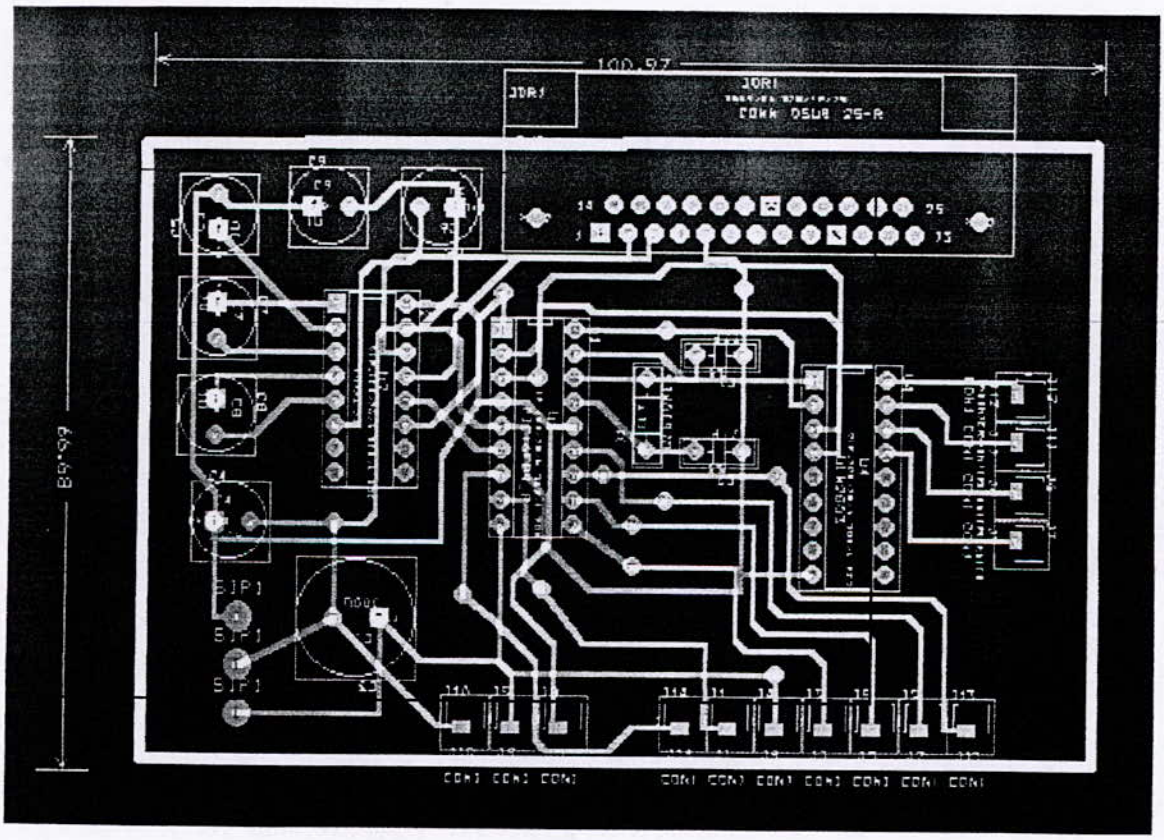
ANNEXE6 : Vue globale de l'automate ACCOS

ANNEXE7 : Séquence 3200 "SUPERVISION CAUDIERE"

ANNEXE 8: Séquence 1.68 "OEC CHAUDIERE"

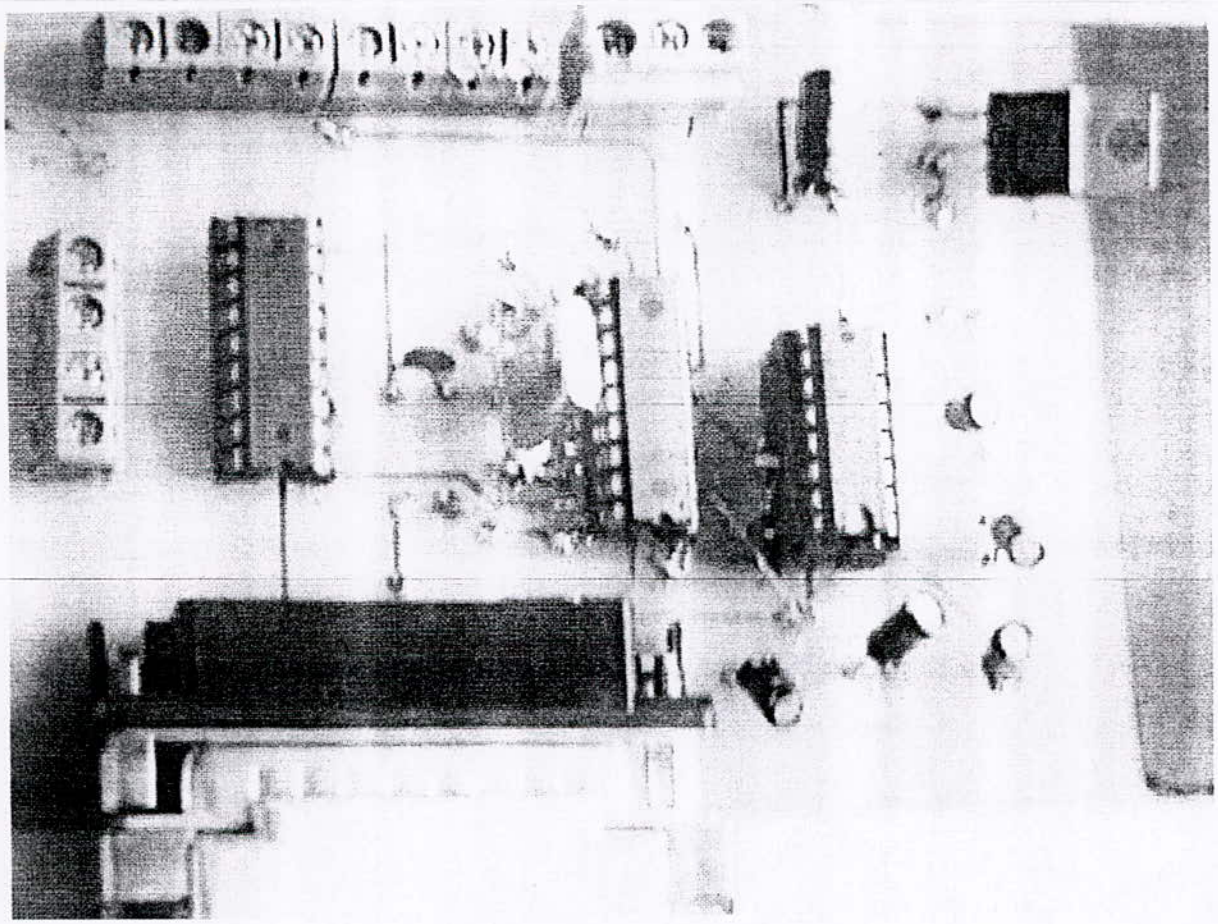
ANNEXE 1

Circuit imprimé de l'automate à base de PIC16F84



ANNEXE 2

Implantation des composants de l'automate à base de PIC 16F84



ANNEXE 3
La routine RS232

```
;*********************************************************************
; Ce fichier contient le programme de communication entre le PIC16F84 *
; et n'importe quel automate ou PC à l'aide d'une interface RS232     *
; RX --->RB0                    *
; TX --->RA2.                   *
; Le programme d'interruption fonctionne en mode interruptible        *
;                               *
;*********************************************************************
;                                                                     *
;    NOM:       programme de communication RS232                      *
;    Date:      01/06/2003                                            *
;    Version:   1.0                                                   *
;    Circuit:   circuit automate à base de pic 16F84 de u-delmas      *
;    Auteur:    Tlemsani hichem                                       *
;                                                                     *
;*********************************************************************
;                                                                     *
;    Fichier requis: P16F84.inc                                       *
;                                                                     *
;                                                                     *
;*********************************************************************


      LIST      p=16F84           ; Définition de processeur
      #include <p16F84.inc>       ; Définitions de variables

      __CONFIG    _CP_Off & _WDT_Off & _PWRTE_ON & _HS_OSC

; '__CONFIG' précise les paramètres encodés dans le processeur au moment de
; la programmation du processeur. Les définitions sont dans le fichier include.
; Voici les valeurs et leurs définitions :
;     _CP_ON            Code protection ON : impossible de relire
;     _CP_OFF           Code protection OFF
;     _PWRTE_ON   Timer reset sur power on en service
;     _PWRTE_OFF  Timer reset hors-service
;     _WDT_ON           Watch-dog en service
;     _WDT_OFF    Watch-dog hors service
;     _LP_OSC           Oscillateur quartz basse vitesse
;     _XT_OSC           Oscillateur quartz moyenne vitesse
;     _HS_OSC           Oscillateur quartz grande vitesse
;     _RC_OSC           Oscillateur à réseau RC

;*********************************************************************
;                              ASSIGNATIONS                           *
;*********************************************************************

OPTIONVAL   EQU   H'0008'     ; Valeur registre option
                              ; Résistance pull-up ON
                              ; Interrupt front déscendant RB0
                              ; Préscaler timer à 1 (pour une communication à
9600 bauds)
                                                    ; préscaler timer à 2 (pour une
communication à 4800 bauds)
                                                    ; préscaler timer à 4 (pour une
communication à 2400 bauds)
```

```
                                                    ; préscaler timer à 8 (pour une
communication à 1200 bauds)
INTERMASK   EQU   H'0090'                  ; Masque d'interruption
                                  ; Interruption sur RB0


;*********************************************************************
;                              DEFINE                                 *
;*********************************************************************

      #define     rx    portb,0           ; pin de réception
      #define     tx    porta,2           ; pin d'émission


;*********************************************************************
;                              MACRO                                  *
;*********************************************************************


bank0 macro
      bcf   status,RP0        ;basculer dans la zone Bank0 de la mémoire
      endm
bank1 macro
      bsf   status,RP0        ;basculer dans la zone Bank1 de la mémoire
      endm

;

;*********************************************************************
;                  DECLARATIONS DE VARIABLES                          *
;*********************************************************************


      CBLOCK 0x40             ; début de la zone variables
      w_temp:1
      status_temp:1
      bitcount:1
      rxreg:1
      txreg:1
      etat:1      ;registre d'état

       ENDC               ; Fin de la zone

;*********************************************************************
;                      DEMARRAGE SUR RESET                            *
;*********************************************************************

      org 0              ; Adresse de départ après reset
      goto    init             ; Adresse 0: initialiser

;*********************************************************************
;                     ROUTINE INTERRUPTION                            *
;*********************************************************************


                  ;sauvegarder registres
                  ;----------------------
     org 0x004          ; adresse d'interruption
initRB0
```

```
      movwf   w_temp    ; sauver registre W
      swapf STATUS,w    ; swap status avec résultat dans w
      movwf status_temp ; sauver status swappé

                  ;main interrupt programm
-------------------------------------------------------------------------------
      bcf         etat,0            ; mettre le flag d'erreur du registre d'état
de communication à zero
      clrf  tmr0        ; inialise tmr0 pour ne pas conter le temps avant le
début de l'interruption
      call  tempbs            ; attendre une durée d'un bit et demi pour
commencer à recevoir
      call  receiv            ;démarrer la séquence de reception pour charger
l'octet reçu dans rxreg
; vérification du mot d'instruction
      movf  rxreg,w           ; transfert du contenu de rxreg dans w
      sublw h'0045'           ; vérification du mot reçu
      btfss status,z    ; verifier bit z
      bsf         etat,0            ; positionner à 1 le bit d'état
      btfsc etat,0            ; tester le bit d'état
      bsf         txreg,7           ; fixer le bit n°7 en cas d'erreur
      call  transmit    ;démarrer la séquence d'emmission de l'état de
l'automate sur 2 octets
      call  transmit
      goto  restorereg  ;fin du programme de communication



                  ;restaurer registres    ----------
                  ;--------------------
restorereg
      bcf         intcon,intf ;effacer le flag d'interrutption
      swapf status_temp,w     ; swap ancien status, résultat dans w
      movwf   STATUS          ; restaurer status
      swapf   w_temp,f  ; Inversion L et H de l'ancien W
                              ; sans modifier Z
      swapf   w_temp,w        ; Réinversion de L et H dans W
                        ; W restauré sans modifier status
      retfie            ; return from interrupt

;*******************************************************************************
;*************************** programme de reception*****************************
;*******************************************************************************

tempbs
      movlw -135        ;chargement de la durée d'attente
      addwf tmr0,f            ;comptabiliser le temps déjas écoulé
      bcf   intcon,t0if       ;effacer le flag
att2
      btfss intcon,t0if ; tester le positionnement du flag
      goto  att2        ; attendre le flag
      return


dstop
      movlw -151        ;chargement de la durée d'attente
      addwf tmr0,f            ;comptabiliser le temps déjas écoulé
      bcf   intcon,t0if       ;effacer le flag
att4
      btfss intcon,t0if ; tester le positionnement du flag
      goto  att4        ; attendre le flag
```

```
      btfss rx                ;tester le bit stop
      bsf         etat,0            ;positionner le flag d'erreur en cas de non
reception du bit stop
      return

receiv

      movlw h'0008'           ; charger le compteur de bits
      movwf bitcount
receivl
      clrf  tmr0        ;remettre à zero pour attendre la durée d'un bit
      nop                           ;le TMR0 est à l'arret
      nop                           ;le TMR0 est à l'arret
      rrf         portb,w           ;mettre le bit reçu dans le bit carry
      rrf         rxreg,f           ;injecter le bit carry dans rxreg
      decfsz      bitcount,f  ;detecter la fin du mot
      goto  $ +3
      call  dstop       ;détéction du bit stop
      return
      call  tempb       ;temporisation pour le prochain bit
      goto  receivl

      return


tempb

      movlw -93               ;chargement de la durée d'attente
      addwf tmr0,f      ;comptabiliser le temps déjas écoulé
      bcf   intcon,t0if       ;effacer le flag
att3
      btfss intcon,t0if ; tester le positionnement du flag
      goto  att3        ; attendre le flag
      return


;*****************************************************************************
;*********************** programme de transmition ****************************
;*****************************************************************************

transmit
      clrf  tmr0
      bcf         tx                ;bit start
      movwf txreg       ;charger le mot à transmettre
      movlw h'0008'
      movwf bitcount    ;charger le compteur mot
      call  tempb       ;temporisation du premier bit
transmitl
      clrf  tmr0        ;mettre à jour le TMR0
      rrf         txreg,f           ;mettre le bit à transmettre dans le bit
carry
      btfsc status,c    ;tester le bit carry
      goto  transmit2   ;saut vers transmit2
      bcf         tx                ;mettre à 0 RA2
transmit3
      decfsz      bitcount,f  ;décrémenter et tester la fin des 8 bits à
transmettre
      goto  transmit4   ;saut vers attendre d'autres bits
      call  tempb       ;temporisation du premier bit stop
```

```
      bsf         tx                ;envoyer bit stop
      call  tempb         ;temporisation du deuxiéme bit stop
      bsf         tx                ;envoyer bit stop
      return
transmit2
      bsf   tx                      ;mettre à 1 RA2
      goto  transmit3   ;retourner vers la suite du programme

transmit4
      call  tempb       ;temporisation du bit prochain
      goto  transmit1   ;retourner vers la suite du programme


;*****************************************************************
;                     INITIALISATIONS                           *
;*****************************************************************

init
      bank1             ; sélectionner banque 1
      movlw h'00ff'     ;mettre en entrée le portb
      movwf portb
      movlw h'0000'     ;mettre en sortie le porta
      movwf porta
        bank0
      clrf  PORTA       ; Sorties portA à 0
      clrf  PORTB       ; sorties portB à 0

                  ; Effacer RAM
                  ; ------------
      movlw 0x0c        ; initialisation pointeur
      movwf FSR               ; pointeur d'adressage indirect
init1
      clrf  INDF        ; effacer ram
      incf  FSR,f       ; pointer sur suivant
      btfss FSR,6       ; tester si fin zone atteinte (>=40)
      goto  init1       ; non, boucler
      btfss FSR,4       ; tester si fin zone atteinte (>=50)
      goto  init1       ; non, boucler



      bank1
      movlw OPTIONVAL   ; charger masque
      movwf OPTION_REG  ; initialiser registre option
      bank0
      bsf         porta,2
      movlw INTERMASK   ; masque interruption
      movwf INTCON            ; charger interrupt control


;*****************************************************************
;                     PROGRAMME PRINCIPAL                       *
;*****************************************************************


start

      goto  start       ;start
```

```
END                    ; directive fin de programme
```

ANNEXE 4
Fichier de définition original de u-delmas
Api0.def

@INTRODUCTION

```
 AAA  PPPP  IIIII  000
A   A P   P   I   0   0
A   A P   P   I   0  00
AAAAA PPPP    I   0 0 0
A   A P       I   00  0
A   A P       I   0   0
A   A P     IIIII  000     API0
```

Ulysse Delmas-Begue (udelmas@chez.com)
15 fev 2002

Ce fichier contient la definition de l'automate "api0" qui propose:
- 350 lignes de graphcet litteral
- zone des bits de 256 bits
- 16 temporisation de 0 a 25.6s (pas 100ms)
- 8 entrees PORTB
- 5 sorties PORTA
- quartz de 4 MHz (1 instruction par us)

Les champs presents sont: (* = champs obligatoires)
- INTRODUCTION : Cette partie
- INFOS *   : Des informations pour progapi
- HEX   *   : Le code de l'interpreteur
- ZB   *   : Nom des bits de la zone des bits
- CST   *   : Constantes
- ASM   : Les sources de l'interpreteur

@INFOS

```
nom                 api0
uc_type             pic_1
uc_taille_flash     400
uc_taille_eeprom     64
taille_zb           256
adr_start_graphcet  140
adr_stop_graphcet   3ff
```

@HEX

```
:020000000528D1
:080008005228A82032204720F5
:10001000AF20C620B2206E2008280D088400840C72
:10002000840C840C1F300405203E84000D08073921
:1000300095000130940095 0A950B1F280800140DB7
:10004000FE3994001C2892010D2000081405031DA0
:10005000920A08000D201409800508000D201408DC
:1000600080040800 9C019D01203084002030940011
:100070008001840A940B3828B520BA200C300C0279
:1000800003190800 9300C0203192E203D289B019A
:100090002D1483160730810083120130810 0A030B7
:1000A0008B0008009800030E970083120408990043
:1000B000AD0AAD0AAD1C66283F30840010309A00AE
:1000C000840A800A800B80039A0B60281908840038
:1000D000170E8300980E180E0B1109002D104030DA
:1000E000840089012E089E001C089B0083201B08A9
:1000F0009C003108B0002F089E001D089B00832043
```

```
:100100001B089D0008000830940001309500 1E086F
:10011000150503 1D8F2815099B05B105A1281B088E
:10012000150503 1D9B2815089B04831608148312CC
:10013000080880 00A12803010007031DA128150855
:10014000B104950D1510840A890A940B87280800BC
:10015000831681138501831 2AC01B22008000609C1
:10016000A80008002C088500080001308A004030F3
:100170008E0008000E08C5208C008E0A0E08C520CF
:100180008D008E0F08008A0A08008200B520BA2070
:100190000A08930001308A000C08303E82001308E0
:1001A0008A00C728080023201208 8F00CF282320A8
:1001B0001208013A8F00CF28232012088F05CF287C
:1001C00023201208013A8F05CF28232012088F041C
:1001D000CF2823201208013A8F04CF2823201208A9
:1001E0008F06CF2823201208013A8F06CF280F1838
:1001F000FB282A20CF282E20CF280D08900023206E
:10020001208910 0CF2811180F1CCF282E2010089B
:060210008D002A20CF281A
:10026000D328D728DC28E028E528E928EE28F2283A
:10027000F728FD28FD280329D228D228D228D22801
:02400E00F13FEE
:00000001FF

@ZB

z  0   x0
z  1   x1
z  2   x2
z  3   x3
z  4   x4
z  5   x5
z  6   x6
z  7   x7
z  8   x8
z  9   x9
z  10  x10
z  11  x11
z  12  x12
z  13  x13
z  14  x14
z  15  x15
z  16  x16
z  17  x17
z  18  x18
z  19  x19
z  20  x20
z  21  x21
z  22  x22
z  23  x23
z  24  x24
z  25  x25
z  26  x26
z  27  x27
z  28  x28
z  29  x29
z  30  x30
z  31  x31
z  32  x32
z  33  x33
z  34  x34
```

```
z  35  x35
z  36  x36
z  37  x37
z  38  x38
z  39  x39
z  40  x40
z  41  x41
z  42  x42
z  43  x43
z  44  x44
z  45  x45
z  46  x46
z  47  x47
z  48  x48
z  49  x49
z  50  x50
z  51  x51
z  52  x52
z  53  x53
z  54  x54
z  55  x55
z  56  x56
z  57  x57
z  58  x58
z  59  x59
z  60  x60
z  61  x61
z  62  x62
z  63  x63

z  64  ia0
z  65  ia1
z  66  ia2
z  67  ia3
z  68  ia4
z  69  ia5
z  70  ia6
z  71  ia7

z  72  bi112
z  73  bi113
z  74  bi114
z  75  bi115
z  76  bi116
z  77  bi117
z  78  bi118
z  79  bi119

z  80  bi120
z  81  bi121
z  82  bi122
z  83  bi123
z  84  bi124
z  85  bi125
z  86  bi126
z  87  bi127

z  88  bi128
z  89  bi129
z  90  bi130
z  91  bi131
```

```
z  92  bi132
z  93  bi133
z  94  bi134
z  95  bi135

z  96  ox0
z  97  ox1
z  98  ox2
z  99  ox3
z  100 ox4
z  101 ox5
z  102 ox6
z  103 ox7

z  104 bs0
z  105 bs1
z  106 bs2
z  107 bs3
z  108 bs4
z  109 bs5
z  110 bs6
z  111 bs7

z  112 tc0
z  113 tc1
z  114 tc2
z  115 tc3
z  116 tc4
z  117 tc5
z  118 tc6
z  119 tc7
z  120 tc8
z  121 tc9
z  122 tc10
z  123 tc11
z  124 tc12
z  125 tc13
z  126 tc14
z  127 tc15

z  128 tf0
z  129 tf1
z  130 tf2
z  131 tf3
z  132 tf4
z  133 tf5
z  134 tf6
z  135 tf7
z  136 tf8
z  137 tf9
z  138 tf10
z  139 tf11
z  140 tf12
z  141 tf13
z  142 tf14
z  143 tf15

z  144 bi0
z  145 bi1
z  146 bi2
z  147 bi3
```

z 148 bi4
z 149 bi5
z 150 bi6
z 151 bi7
z 152 bi8
z 153 bi9
z 154 bi10
z 155 bi11
z 156 bi12
z 157 bi13
z 158 bi14
z 159 bi15
z 160 bi16
z 161 bi17
z 162 bi18
z 163 bi19
z 164 bi20
z 165 bi21
z 166 bi22
z 167 bi23
z 168 bi24
z 169 bi25
z 170 bi26
z 171 bi27
z 172 bi28
z 173 bi29
z 174 bi30
z 175 bi31
z 176 bi32
z 177 bi33
z 178 bi34
z 179 bi35
z 180 bi36
z 181 bi37
z 182 bi38
z 183 bi39
z 184 bi40
z 185 bi41
z 186 bi42
z 187 bi43
z 188 bi44
z 189 bi45
z 190 bi46
z 191 bi47
z 192 bi48
z 193 bi49
z 194 bi50
z 195 bi51
z 196 bi52
z 197 bi53
z 198 bi54
z 199 bi55
z 200 bi56
z 201 bi57
z 202 bi58
z 203 bi59
z 204 bi60
z 205 bi61
z 206 bi62
z 207 bi63
z 208 bi64

```
z  209 bi65
z  210 bi66
z  211 bi67
z  212 bi68
z  213 bi69
z  214 bi70
z  215 bi71
z  216 bi72
z  217 bi73
z  218 bi74
z  219 bi75
z  220 bi76
z  221 bi77
z  222 bi78
z  223 bi79
z  224 bi80
z  225 bi81
z  226 bi82
z  227 bi83
z  228 bi84
z  229 bi85
z  230 bi86
z  231 bi87
z  232 bi88
z  233 bi89
z  234 bi90
z  235 bi91
z  236 bi92
z  237 bi93
z  238 bi94
z  239 bi95
z  240 bi96
z  241 bi97
z  242 bi98
z  243 bi99
z  244 bi100
z  245 bi101
z  246 bi102
z  247 bi103
z  248 bi104
z  249 bi105
z  250 bi106
z  251 bi107
z  252 bi108
z  253 bi109
z  254 bi110
z  255 bi111

@CST

eb 0 t0
eb 1 t1
eb 2 t2
eb 3 t3
eb 4 t4
eb 5 t5
eb 6 t6
eb 7 t7
eb 8 t8
eb 9 t9
```

```
eb a t10
eb b t11
eb c t12
eb d t13
eb e t14
eb f t15

@END

A partir d'ici n'importe quel texte peut etre insere

@ASM

;*****************************************************************
;                          AUTO.ASM
;   Interpreteur pour l'automate programmable a PIC16F84
;*****************************************************************

        LIST    p=16C84 ; PIC16C84 is the target processor

;***** declarations *****

;*** standard ***
INDF    EQU     0x00
TMR0    EQU     0x01
PCL     EQU     0x02
STATUS  EQU     0x03
FSR     EQU     0x04
PORTA   EQU     0x05
PORTB   EQU     0x06
EEDATA  EQU     0x08
EEADR   EQU     0x09
PCLATH  EQU     0x0A
INTCON  EQU     0x0B

OPT___  EQU     0x01
TRISA   EQU     0x05
TRISB   EQU     0x06
EECON1  EQU     0x08
EECON2  EQU     0x09

;*** variables ***

;relatif table
ORDRE   EQU     0x0c
VALEUR  EQU     0x0d
PT      EQU     0x0e

;relatif traitement
TEST    EQU     0x0f
ETAPE   EQU     0x10
SAUT    EQU     0x11

;relatif fonction
VALBIT  EQU     0x12
PCLAT2  EQU     0x13

;relatif general
varA        EQU     0x14
```

```
varB        EQU     0x15
varC        EQU     0x16
S_STATUS    EQU     0x17
S_w         EQU     0x18
S_FSR       EQU     0x19
DECMPT      EQU     0x1a
EC          EQU     0x1b
EC1         EQU     0x1c
EC2         EQU     0x1d
T0          EQU     0x1e

;*** constantes ***
_DEBZB      EQU     0x20
_DEBDCMPT   EQU     0x40
_SWITCH     EQU     0x30

;*** bits particuliers (0-7) ***
CARRY   EQU     0x00
ZERO    EQU     0x02
RBPU    EQU     0x07
RP0     EQU     0x05
LINE_A  EQU     0x04
LINE_B  EQU     0x03
LINE_C  EQU     0x02
LINE_D  EQU     0x01
LINE_CK EQU     0x00
RD      EQU     0x00

;**** programme ****

        ORG     0
        goto    main

        ORG     0x4
        goto    ittmp

        ORG     0x5
main
        call    iniport
        call    inizb
        call    initmp
bmain   call    entrees
        call    traite
        call    sorties
        call    traitmp
        goto    bmain

;*** fonctions ***
;convbit
;vb
;cb
;sb
;inizb
;iniport
;initmp
;ittmp
;traitmp
;entrees
;sorties
;initab
;acqtab
```

```
;traite

;=== convbit ===
;valeur --> ### --> FSR
;          ### --> varA
;          U -- varB
;transforme la valeur en adresse FSR sur la zone des bit et met
;dans varA le masque pour le bon bit
convbit
        movf    VALEUR,w
        movwf   FSR
        rrf     FSR,f
        rrf     FSR,f
        rrf     FSR,f
        movlw   0x1f
        andwf   FSR,w
        addlw   _DEBZB
        movwf   FSR

        movf    VALEUR,w
        andlw   0x07
        movwf   varB
        movlw   1
        movwf   varA
        incf    varB,f
tstconv decfsz  varB,f
        goto    suiconv
        return
suiconv rlf     varA,w
        andlw   0xfe
        movwf   varA
        goto    tstconv

;=== vb ===
;valeur --> ### --> VALBIT
;          U -- varA,varB
;met dans valbit la valeur du bit valeur de la zone des bits
vb
        clrf    VALBIT
        call    convbit
        movf    INDF,w
        andwf   varA,w
        btfss   STATUS,ZERO
        incf    VALBIT,f
        return

;=== cb ===
;valeur --> ###
;          U -- varA,varB
;met a 0 le bit valeur de la zone des bits
cb
        call    convbit
        comf    varA,w
        andwf   INDF,f
        return

;=== sb ===
;valeur --> ###
;          U -- varA,varB
;met a 1 le bit valeur de la zone des bits
sb
```

```
        call    convbit
        movf    varA,w
        iorwf   INDF,f
        return

;===  inizb ===
;           ###
;           U -- varA
; initialise la zone des bits a 0 sauf pour les etapes initiales
inizb
        clrf    EC1
        clrf    EC2

        movlw   _DEBZB
        movwf   FSR
        movlw   0x20
        movwf   varA
binizb0 clrf    INDF
        incf    FSR,f
        decfsz  varA,f
        goto    binizb0

        call    initab
binizb1 call    acqtab
        movlw   0x0c
        subwf   ORDRE,w
        btfsc   STATUS,ZERO
        return
        movlw   0x09
        subwf   ORDRE,w
        btfsc   STATUS,ZERO
        call    sb
        goto    binizb1

;=== initmp ===
;           ###
;           U --
;initialise les interruptions pour la temporisation
initmp
        clrf    EC
        bsf     _DEBZB+0d,0      ;met a 1 le bit de demarage
        bsf     STATUS,RP0
        movlw   0x07
        movwf   OPT___           ;fixe la predivision a 256
        bcf     STATUS,RP0
        movlw   0x01
        movwf   TMR0             ;initialise le timer
        movlw   0xa0
        movwf   INTCON           ;active l'it timer
        return

;=== ittmp ===
;           ###
;           U -- S_STATUS S_w S_FSR DECMPT
;gere les decompteurs de temporisation pendant les it du timer
ittmp
itsave  movwf   S_w
        swapf   STATUS,w
        movwf   S_STATUS
        bcf     STATUS,RP0
        movf    FSR,w
```

```
        movwf   S_FSR

ittrai  incf    _DEBZB+0xd,f
        incf    _DEBZB+0xd,f
        btfss   _DEBZB+0xd,1
        goto    itresto

        movlw   _DEBDCMPT-1
        movwf   FSR
        movlw   0x10
        movwf   DECMPT
bittrai incf    FSR,f
        incf    INDF,f
        decfsz  INDF,f
        decf    INDF,f
        decfsz  DECMPT,f
        goto    bittrai

itresto movf    S_FSR,w
        movwf   FSR
        swapf   S_STATUS,w
        movwf   STATUS
        swapf   S_w,f
        swapf   S_w,w
        bcf     INTCON,2
        retfie

;=== traitmp ===
;           ###
;           U -- varA,varB,EC
;gere les indicateurs de temporisation
traitmp
        bcf     _DEBZB+0xd,0    ;met a 0 le bit de demarage

        movlw   _DEBDCMPT
        movwf   FSR
        clrf    EEADR

        movf    _DEBZB+0xe,w
        movwf   T0
        movf    EC1,w
        movwf   EC
        call    traitmp1
        movf    EC,w
        movwf   EC1
        movf    _DEBZB+0x11,w
        movwf   _DEBZB+0x10

        movf    _DEBZB+0xf,w
        movwf   T0
        movf    EC2,w
        movwf   EC
        call    traitmp1
        movf    EC,w
        movwf   EC2

        return

traitmp1
        movlw   0x08
        movwf   varA
```

```
        movlw   0x01
        movwf   varB

btmp    movf    T0,w
        andwf   varB,w
        btfss   STATUS,ZERO
        goto    t1

t0      comf    varB,w
        andwf   EC,f
        andwf   _DEBZB+0x11,f
        goto    ftmp

t1      movf    EC,w
        andwf   varB,w
        btfss   STATUS,ZERO
        goto    ec1

ec0     movf    varB,w
        iorwf   EC,f

        bsf     STATUS,RP0
        bsf     EECON1,RD
        bcf     STATUS,RP0
        movf    EEDATA,w
        movwf   INDF

        goto    ftmp

ec1     clrw
        addwf   INDF,w
        btfss   STATUS,ZERO
        goto    ftmp
        movf    varB,w
        iorwf   _DEBZB+0x11,f

ftmp    rlf     varB,f
        bcf     varB,0
        incf    FSR,f
        incf    EEADR,f
        decfsz  varA,f
        goto    btmp
        return

;=== iniport ===
;           ###
;           U --
;initialise les ports avec 0 en sortie et les pull-up
iniport
        bsf     STATUS,RP0
        bcf     OPT___,RBPU
        clrf    TRISA
        bcf     STATUS,RP0

        clrf    _DEBZB+0xc
        call    sorties
        return

;=== entrees ===
;           ###
;           U --
```

```
;remplit la zone des bit pour les entrees
entrees
        comf    PORTB,w
        movwf   _DEBZB+0x08
        return

;=== sorties ===
;           ###
;           U --
;met a jour les sorties suivant la zone des bits
sorties
        movf    _DEBZB+0xc,w
        movwf   PORTA
        return

;=== initab ===
;           ###
;           U -- PT
;prepare le pointeur pour pointer sur le premier element
initab
        movlw   0x01
        movwf   PCLATH
        movlw   0x40
        movwf   PT
        return

;=== acqtab ===
;           ### --> ORDRE
;           ### --> VALEUR
;           U -- PT
;remplit ordre et valeur et pointe suer le suivant
acqtab
        movf    PT,w
        call    rechtab
        movwf   ORDRE
        incf    PT,f
        movf    PT,w
        call    rechtab
        movwf   VALEUR
        incfsz  PT,f
        return
        incf    PCLATH,f
        return
rechtab movwf   PCL

;=== traite ===
;           ###
;           U -- PCLAT2,varA,varB,TEST,SAUT,ETAPE,varC
;traite le graphcet
traite
        call    initab
btraite call    acqtab
        movf    PCLATH,w
        movwf   PCLAT2
        movlw   0x01
        movwf   PCLATH
        movf    ORDRE,w
        addlw   _SWITCH
        movwf   PCL

traitee movf    PCLAT2,w
```

```
        movwf   PCLATH
        goto    btraite

traitef return

_0_1    call    vb
        movf    VALBIT,w
        movwf   TEST
        goto    traitee

_1_1n   call    vb
        movf    VALBIT,w
        xorlw   0x01
        movwf   TEST
        goto    traitee

_2_a    call    vb
        movf    VALBIT,w
        andwf   TEST,f
        goto    traitee

_3_an   call    vb
        movf    VALBIT,w
        xorlw   0x01
        andwf   TEST,f
        goto    traitee

_4_o    call    vb
        movf    VALBIT,w
        iorwf   TEST,f
        goto    traitee

_5_on   call    vb
        movf    VALBIT,w
        xorlw   0x01
        iorwf   TEST,f
        goto    traitee

_6_x    call    vb
        movf    VALBIT,w
        xorwf   TEST,f
        goto    traitee

_7_xn   call    vb
        movf    VALBIT,w
        xorlw   0x01
        xorwf   TEST,f
        goto    traitee

_8_ega  btfsc   TEST,0
        goto    ega1
ega0    call    cb
        goto    traitee
ega1    call    sb
        goto    traitee

_9_eti
_a_et   movf    VALEUR,w
        movwf   ETAPE
        call    vb
        movf    VALBIT,w
```

```
        movwf   SAUT
        goto    traitee

_b_va   btfsc   SAUT,0
        btfss   TEST,0
        goto    traitee
        call    sb
        movf    ETAPE,w
        movwf   VALEUR
        call    cb
        goto    traitee


;=== table de saut switch ===
        ORG     0x130
switch
        goto    _0_l
        goto    _1_ln
        goto    _2_a
        goto    _3_an
        goto    _4_o
        goto    _5_on
        goto    _6_x
        goto    _7_xn
        goto    _8_ega
        goto    _9_eti
        goto    _a_et
        goto    _b_va
        goto    traitef
        goto    traitef
        goto    traitef
        goto    traitef

;=== graphcet en code a pqrtir de 0x200 ===

        END
```

ANNEXE 5
Le fichier de définition modifié Api0c.def

@INTRODUCTION

```
 AAA  PPPP  IIIII  000
A   A P   P   I   0   0
A   A P   P   I   0  00
AAAAA PPPP    I   0 0 0
A   A P       I   00  0
A   A P       I   0   0
A   A P     IIIII  000     API0
```

Ulysse Delmas-Begue_tlemsani hichem (udelmas@chez.com,epicure3@hotmail.com)
15 fev 2002
update 01 jui 2003

Ce fichier contient la definition de l'automate "api0" qui propose:
- 286 lignes de graphcet litteral
- zone des bits de 256 bits
- 16 temporisation de 0 a 25.6s (pas 100ms)
- 8 entrees PORTB
- 5 sorties PORTA
- quartz de 4 MHz (1 instruction par us)

Les champs presents sont: (* = champs obligatoires)
- INTRODUCTION : Cette partie
- INFOS *   : Des informations pour progapi
- HEX   *   : Le code de l'interpreteur
- ZB   *   : Nom des bits de la zone des bits
- CST   *   : Constantes
- ASM   : Les sources de l'interpreteur

@INFOS

```
nom                 api0c
uc_type             pic_1
uc_taille_flash     400
uc_taille_eeprom     64
taille_zb           256
adr_start_graphcet 180
adr_stop_graphcet   3ff
```

@HEX

```
:020000040000FA
:020000000528D1
:0800080052280421322047 2098
:100010000B2122210E21CA2008280D088400840CFF
:10002000840C840C1F300405203E84000D08073921
:1000300095000130940095 0A950B1F280800140DB7
:10004000FE3994001C2892010D200008140503 1DA0
:10005000920A08000D201409800508000D201408DC
:1000600080040800 9C019D01203084002030940011
:100070008001840A940B3828112116210C300C02BF
:1000800003190800093 00C0203192E203D289B019A
:1000900020148316073081008312013081 00B030B4
:1000A0008B0008009800030E970083120408990043
:1000B0008B185B28B428831608308100051543107F
:1000C00081017B208A204108453C031D43142808F8
:1000D000C2009E202C08C2004318C2179E20073081
:1000E000810072281908840017 0E8300980E180EDC
```

```
:1000F0008B100B1109007930810708B110B1D7E2825
:10010000080069308107OB110B1D84280618892807
:1001100043140800083OC0008101000000000060CF4
:10012000C10CC00B95288120080098208C2808005D
:10013000A33081070B110B1D9B28080081010511BD
:10014000C2000830C0009820810IC20C0318B028FA
:100150000511C00BB228982005159820051508003S
:100160000515A9289820A428AD0AAD0AAD1CC228FF
:100170003F30840010309A00840A800A800B80038C
:100180009A0BBC2819088400170E8300980E180ECD
:100190000B1109002D104030840089012E089E00AB
:1001A0001C089B00DF201B089C003108B0002F08B2
:1001B0009E001D089B00DF201B089D0008000830E2
:1001C000940013095001E081505031DEB28150944
:1001D0009B05B105FD281B081505031DF72815080B
:1001E0009B048316081483120808800FD2803016D
:1001F0000007031DFD281508B104950D1510840A8C
:10020000890A940BE32808008316811385018312G1
:10021000AC010E2108000609A80008002C08850082
:1002200008000130SA0080308E0008000E0821216D
:100230008C008E0A0E0821218D008E0F08008A0A7C
:10024000080082001121162l0A08930001308A005B
:100250000C08303E820013088A002329080023205E
:1002600012088F002B2923201208013A8F002B2916
:1002700023201208SF052B2923201208013A8F050D
:100280002B29232012088F042B2923201208013A3E
:100290008F042B29232012088F062B2923201208D4
:1002A000013A8F062B290F1857292A202B292E2097
:1002B0002B290D0890002320120891002B291118DA
:0E02C0000F1C2B292E2010088D002A202B2920
:1002E0002F29332938293C29412945294A294E29D2
:1002F000532959295929529F292E292E292E292E299A
:00000001FF
```

```
@ZB

z  0   x0
z  1   x1
z  2   x2
z  3   x3
z  4   x4
z  5   x5
z  6   x6
z  7   x7
z  8   x8
z  9   x9
z  10  x10
z  11  x11
z  12  x12
z  13  x13
z  14  x14
z  15  x15
z  16  x16
z  17  x17
z  18  x18
z  19  x19
z  20  x20
z  21  x21
z  22  x22
z  23  x23
```

```
z  24  x24
z  25  x25
z  26  x26
z  27  x27
z  28  x28
z  29  x29
z  30  x30
z  31  x31
z  32  x32
z  33  x33
z  34  x34
z  35  x35
z  36  x36
z  37  x37
z  38  x38
z  39  x39
z  40  x40
z  41  x41
z  42  x42
z  43  x43
z  44  x44
z  45  x45
z  46  x46
z  47  x47
z  48  x48
z  49  x49
z  50  x50
z  51  x51
z  52  x52
z  53  x53
z  54  x54
z  55  x55
z  56  x56
z  57  x57
z  58  x58
z  59  x59
z  60  x60
z  61  x61
z  62  x62
z  63  x63

z  64  ia0
z  65  ia1
z  66  ia2
z  67  ia3
z  68  ia4
z  69  ia5
z  70  ia6
z  71  ia7

z  72  bi112
z  73  bi113
z  74  bi114
z  75  bi115
z  76  bi116
z  77  bi117
z  78  bi118
z  79  bi119

z  80  bi120
z  81  bi121
```

```
z  82  bi122
z  83  bi123
z  84  bi124
z  85  bi125
z  86  bi126
z  87  bi127

z  88  bi128
z  89  bi129
z  90  bi130
z  91  bi131
z  92  bi132
z  93  bi133
z  94  bi134
z  95  bi135

z  96  ox0
z  97  ox1
z  98  ox2
z  99  ox3
z  100 ox4
z  101 ox5
z  102 ox6
z  103 ox7

z  104 bs0
z  105 bs1
z  106 bs2
z  107 bs3
z  108 bs4
z  109 bs5
z  110 bs6
z  111 bs7

z  112 tc0
z  113 tc1
z  114 tc2
z  115 tc3
z  116 tc4
z  117 tc5
z  118 tc6
z  119 tc7
z  120 tc8
z  121 tc9
z  122 tc10
z  123 tc11
z  124 tc12
z  125 tc13
z  126 tc14
z  127 tc15

z  128 tf0
z  129 tf1
z  130 tf2
z  131 tf3
z  132 tf4
z  133 tf5
z  134 tf6
z  135 tf7
z  136 tf8
z  137 tf9
```

```
z  138 tf10
z  139 tf11
z  140 tf12
z  141 tf13
z  142 tf14
z  143 tf15

z  144 bi0
z  145 bi1
z  146 bi2
z  147 bi3
z  148 bi4
z  149 bi5
z  150 bi6
z  151 bi7
z  152 bi8
z  153 bi9
z  154 bi10
z  155 bi11
z  156 bi12
z  157 bi13
z  158 bi14
z  159 bi15
z  160 bi16
z  161 bi17
z  162 bi18
z  163 bi19
z  164 bi20
z  165 bi21
z  166 bi22
z  167 bi23
z  168 bi24
z  169 bi25
z  170 bi26
z  171 bi27
z  172 bi28
z  173 bi29
z  174 bi30
z  175 bi31
z  176 bi32
z  177 bi33
z  178 bi34
z  179 bi35
z  180 bi36
z  181 bi37
z  182 bi38
z  183 bi39
z  184 bi40
z  185 bi41
z  186 bi42
z  187 bi43
z  188 bi44
z  189 bi45
z  190 bi46
z  191 bi47
z  192 bi48
z  193 bi49
z  194 bi50
z  195 bi51
z  196 bi52
z  197 bi53
```

```
z   198 bi54
z   199 bi55
z   200 bi56
z   201 bi57
z   202 bi58
z   203 bi59
z   204 bi60
z   205 bi61
z   206 bi62
z   207 bi63
z   208 bi64
z   209 bi65
z   210 bi66
z   211 bi67
z   212 bi68
z   213 bi69
z   214 bi70
z   215 bi71
z   216 bi72
z   217 bi73
z   218 bi74
z   219 bi75
z   220 bi76
z   221 bi77
z   222 bi78
z   223 bi79
z   224 bi80
z   225 bi81
z   226 bi82
z   227 bi83
z   228 bi84
z   229 bi85
z   230 bi86
z   231 bi87
z   232 bi88
z   233 bi89
z   234 bi90
z   235 bi91
z   236 bi92
z   237 bi93
z   238 bi94
z   239 bi95
z   240 bi96
z   241 bi97
z   242 bi98
z   243 bi99
z   244 bi100
z   245 bi101
z   246 bi102
z   247 bi103
z   248 bi104
z   249 bi105
z   250 bi106
z   251 bi107
z   252 bi108
z   253 bi109
z   254 bi110
z   255 bi111

@CST
```

```
eb 0 t0
eb 1 t1
eb 2 t2
eb 3 t3
eb 4 t4
eb 5 t5
eb 6 t6
eb 7 t7
eb 8 t8
eb 9 t9
eb a t10
eb b t11
eb c t12
eb d t13
eb e t14
eb f t15


@END

A partir d'ici n'importe quel texte peut etre insere


@ASM
;*****************************************************************
;                          AUTO.ASM
;   Interpreteur pour l'automate programmable a PIC16F84
;*****************************************************************

        LIST    p=16f84 ; PIC16C84 is the target processor

;***** declarations *****

;*** standard ***
INDF    EQU     0x00
TMR0    EQU     0x01
PCL     EQU     0x02
STATUS  EQU     0x03
FSR     EQU     0x04
PORTA   EQU     0x05
PORTB   EQU     0x06
EEDATA  EQU     0x08
EEADR   EQU     0x09
PCLATH  EQU     0x0A
INTCON  EQU     0x0B

OPT___  EQU     0x01
TRISA   EQU     0x05
TRISB   EQU     0x06
EECON1  EQU     0x08
EECON2  EQU     0x09

;*** variables ***

;relatif table
ORDRE   EQU     0x0c
VALEUR  EQU     0x0d
PT      EQU     0x0e

;relatif traitement
```

```
TEST    EQU     0x0f
ETAPE   EQU     0x10
SAUT    EQU     0x11

;relatif fonction
VALBIT  EQU     0x12
PCLAT2  EQU     0x13

;relatif general
varA        EQU     0x14
varB        EQU     0x15
varC        EQU     0x16
S_STATUS    EQU     0x17
S_w         EQU     0x18
S_FSR       EQU     0x19
DECMPT      EQU     0x1a
EC          EQU     0x1b
EC1         EQU     0x1c
EC2         EQU     0x1d
T0          EQU     0x1e

;*** constantes ***
_DEBZB      EQU     0x20
_DEBDCMPT   EQU     0x40
_SWITCH     EQU     0x30

;***entrées***
ia                  equ         0x28

;****sorties*******
ox                  equ         0x2c

;*** bits particuliers (0-7) ***
CARRY   EQU     0x00
ZERO    EQU     0x02
RBPU    EQU     0x07
RP0     EQU     0x05
LINE_A  EQU     0x04
LINE_B  EQU     0x03
LINE_C  EQU     0x02
LINE_D  EQU     0x01
LINE_CK EQU     0x00
RD      EQU     0x00
;----- INTCON Bits --------------------------------------------------

GIE                         EQU     H'0007'
EEIE                        EQU     H'0006'
T0IE                        EQU     H'0005'
INTE                        EQU     H'0004'
RBIE                        EQU     H'0003'
T0IF                        EQU     H'0002'
INTF                        EQU     H'0001'
RBIF                        EQU     H'0000'

;----- OPTION Bits --------------------------------------------------

NOT_RBPU                    EQU     H'0007'
INTEDG                      EQU     H'0006'
T0CS                        EQU     H'0005'
T0SE                        EQU     H'0004'
PSA                         EQU     H'0003'
```

```
PS2                             EQU     H'0002'
PS1                             EQU     H'0001'
PS0                             EQU     H'0000'

;*****************************************************************
;                               ASSIGNATIONS                      *
;*****************************************************************

OPTIONVAL   EQU   H'0008'     ; Valeur registre option
                                ; Résistance pull-up ON
                                ; Interrupt front déscendant RB0
                                ; Préscaler timer à 1 (pour une communication à
9600 bauds)
                                                ; préscaler timer à 2 (pour une
communication à 4800 bauds)
                                                ; préscaler timer à 4 (pour une
communication à 2400 bauds)
                                                ; préscaler timer à 8 (pour une
communication à 1200 bauds)


;*****************************************************************
;                               DEFINE                            *
;*****************************************************************

      #define     rx    portb,0           ; pin de réception
      #define     tx    porta,2           ; pin d'émission

;*****************************************************************
;                               MACRO                             *
;*****************************************************************


bank0 macro
      bcf   status,RP0        ;basculer dans la zone Bank0 de la mémoire
      endm
bank1 macro
      bsf   status,RP0        ;basculer dans la zone Bank1 de la mémoire
      endm

;

;*****************************************************************
;                   DECLARATIONS DE VARIABLES                     *
;*****************************************************************


      CBLOCK 0x40             ; début de la zone variables
      bitcount:1
      rxreg:1
      txreg:1
      etat:1      ;registre d'état

       ENDC               ; Fin de la zone

;**** programme ****

        ORG     0
        goto    main
```

```
        ORG     0x4
        goto    ittmp

        ORG     0x5
main
        call    iniport
        call    inizb
        call    initmp
bmain   call    entrees
        call    traite
        call    sorties
        call    traitmp
        goto    bmain

;*** fonctions ***
;convbit
;vb
;cb
;sb
;inizb
;iniport
;initmp
;ittmp
;traitmp
;entrees
;sorties
;initab
;acqtab
;traite

;=== convbit ===
;valeur --> ### --> FSR
;           ### --> varA
;           U -- varB
;transforme la valeur en adresse FSR sur la zone des bit et met
;dans varA le masque pour le bon bit
convbit
        movf    VALEUR,w
        movwf   FSR
        rrf     FSR,f
        rrf     FSR,f
        rrf     FSR,f
        movlw   0x1f
        andwf   FSR,w
        addlw   _DEBZB
        movwf   FSR

        movf    VALEUR,w
        andlw   0x07
        movwf   varB
        movlw   1
        movwf   varA
        incf    varB,f
tstconv decfsz  varB,f
        goto    suiconv
        return
suiconv rlf     varA,w
        andlw   0xfe
        movwf   varA
        goto    tstconv
```

```
;=== vb ===
;valeur --> ### --> VALBIT
;           U -- varA,varB
;met dans valbit la valeur du bit valeur de la zone des bits
vb
        clrf    VALBIT
        call    convbit
        movf    INDF,w
        andwf   varA,w
        btfss   STATUS,ZERO
        incf    VALBIT,f
        return

;=== cb ===
;valeur --> ###
;           U -- varA,varB
;met a 0 le bit valeur de la zone des bits
cb
        call    convbit
        comf    varA,w
        andwf   INDF,f
        return

;=== sb ===
;valeur --> ###
;           U -- varA,varB
;met a 1 le bit valeur de la zone des bits
sb
        call    convbit
        movf    varA,w
        iorwf   INDF,f
        return

;===  inizb ===
;           ###
;           U -- varA
; initialise la zone des bits a 0 sauf pour les etapes initiales
inizb
        clrf    EC1
        clrf    EC2

        movlw   _DEBZB
        movwf   FSR
        movlw   0x20
        movwf   varA
binizb0 clrf    INDF
        incf    FSR,f
        decfsz  varA,f
        goto    binizb0

        call    initab
binizb1 call    acqtab
        movlw   0x0c
        subwf   ORDRE,w
        btfsc   STATUS,ZERO
        return
        movlw   0x09
        subwf   ORDRE,w
        btfsc   STATUS,ZERO
        call    sb
```

```
        goto    binizbl

;=== initmp ===
;           ###
;           U --
;initialise les interruptions pour la temporisation
initmp
        clrf    EC
        bsf     _DEBZB+0d,0     ;met a 1 le bit de demarage
        bsf     STATUS,RP0
        movlw   0x07
        movwf   OPT___          ;fixe la predivision a 256
        bcf     STATUS,RP0
        movlw   0x01
        movwf   TMR0            ;initialise le timer
        movlw   0xb0
        movwf   INTCON          ;active l'it timer et l'it RB0
        return

;=== ittmp ===
;           ###
;           U -- S_STATUS S_w S_FSR DECMPT
;gere les decompteurs de temporisation pendant les it du timer et le programme
de
;de communication
ittmp
itsave  movwf   S_w
        swapf   STATUS,w
        movwf   S_STATUS
        bcf     STATUS,RP0
        movf    FSR,w
        movwf   S_FSR


      btfsc     intcon,intf
      goto      initRB0
      goto      ittrai

initRB0
;*****************************************************************
;                       INITIALISATIONS                          *
;*****************************************************************

      bank1
      movlw     OPTIONVAL   ; charger masque
      movwf     OPT___      ; initialiser registre option


                ;main interrupt programm
      bsf             porta,2
      bcf             etat,0          ; mettre le flag d'erreur du registre
d'état de communication à zero
      clrf      tmr0        ; inialise tmr0 pour ne pas conter le temps avant
le début de l'instruction
      call      tempbs      ; attendre une durée d'un bit et demi pour
commencer à recevoir
      call      receiv      ;démarrer la séquence de reception pour
charger l'octet reçu dans rxreg
; vérification du mot d'instruction
      movf      rxreg,w
      sublw     h'0045'
```

```
      btfss       status,zero
      bsf               etat,0
      movf        ia,w
      movwf       txreg
      call        transmit    ;démarrer la séquence d'emmission de l'état de
l'automate sur 2 octets
      movf        ox,w
      movwf       txreg
      btfsc       etat,0
      bsf               txreg,7
      call        transmit

      movlw       h'0007'
      movwf       opt___
      goto        itresto1
itresto1
       movf    S_FSR,w
    movwf   FSR
    swapf   S_STATUS,w
    movwf   STATUS
    swapf   S_w,f
    swapf   S_w,w
      bcf         intcon,intf ;effacer le flag d'interrutption
    bcf           intcon,t0if ;effacer le flag du tmr0
    retfie


;*****************************************************************************
;*************************** programme_de_reception**************************
;*****************************************************************************

tempbs
      movlw       -135        ;chargement de la durée d'attente
      addwf       tmr0,f
      bcf         intcon,t0if
att2
      btfss       intcon,t0if
      goto        att2
      return


dstop
      movlw       -151        ;chargement de la durée d'attente
      addwf       tmr0,f
      bcf               intcon,t0if
att4
      btfss       intcon,t0if
      goto        att4
      btfsc       rx
      goto        $ +2
      bsf               etat,0
      return



receiv

      movlw       h'0008'
      movwf       bitcount
receivl
      clrf        tmr0        ;remettre à zero pour attendre la durée d'un bit
```

```
      nop                           ;le TMR0 est à l'arret
      nop                           ;le TMR0 est à l'arret
      rrf               portb,w           ;mettre le bit reçu dans le bit carry
      rrf               rxreg,f           ;injecter le bit carry dans rxreg
      decfsz            bitcount,f  ;detecter la fin du mot
      goto              $+3
      call        dstop       ;détéction du bit stop
      return
      call        tempb
      goto        receiv1

      return


tempb

      movlw       -93
      addwf       tmr0,f
      bcf               intcon,t0if
att3
      btfss       intcon,t0if
      goto        att3
      return


;*********************************************************************************
;************************ programme de transmition ****************************
;*********************************************************************************
transmit
      clrf        tmr0
      bcf               tx                ;bit start
      movwf       txreg       ;charger le mot à transmettre
      movlw       h'0008'
      movwf       bitcount    ;charger le compteur mot
      call        tempb       ;temporisation du premier bit
transmit1
      clrf        tmr0        ;mettre à jour le TMR0
      rrf               txreg,f           ;mettre le bit à transmettre dans le
bit carry
      btfsc       status,carry;tester le bit carry
      goto        transmit2   ;saut vers transmit2
      bcf               tx                ;mettre à 0 RA2
transmit3
      decfsz      bitcount,f  ;décrémenter et tester la fin des 8 bits à
transmettre
      goto        transmit4   ;saut vers attendre d'autres bits
      call        tempb       ;temporisation du premier bit stop
      bsf         tx                ;envoyer bit stop
      call        tempb       ;temporisation du deuxiéme bit stop
      bsf         tx                ;envoyer bit stop
      return
transmit2
      bsf   tx                      ;mettre à 1 RA2
      goto  transmit3   ;retourner vers la suite du programme

transmit4
      call  tempb       ;temporisation du bit prochain
      goto  transmit1   ;retourner vers la suite du programme
```

```
;suite du programme d'interruption
ittrai  incf    _DEBZB+0xd,f
        incf    _DEBZB+0xd,f
        btfss   _DEBZB+0xd,1
        goto    itresto

        movlw   _DEBDCMPT-1
        movwf   FSR
        movlw   0x10
        movwf   DECMPT
bittrai incf    FSR,f
        incf    INDF,f
        decfsz  INDF,f
        decf    INDF,f
        decfsz  DECMPT,f
        goto    bittrai

itresto movf    S_FSR,w
        movwf   FSR
        swapf   S_STATUS,w
        movwf   STATUS
        swapf   S_w,f
        swapf   S_w,w
        bcf     INTCON,2
        retfie

;=== traitmp ===
;           ###
;           U -- varA,varB,EC
;gere les indicateurs de temporisation
traitmp
        bcf     _DEBZB+0xd,0    ;met a 0 le bit de demarage

        movlw   _DEBDCMPT
        movwf   FSR
        clrf    EEADR

        movf    _DEBZB+0xe,w
        movwf   T0
        movf    EC1,w
        movwf   EC
        call    traitmp1
        movf    EC,w
        movwf   EC1
        movf    _DEBZB+0x11,w
        movwf   _DEBZB+0x10

        movf    _DEBZB+0xf,w
        movwf   T0
        movf    EC2,w
        movwf   EC
        call    traitmp1
        movf    EC,w
        movwf   EC2

        return

traitmp1
        movlw   0x08
        movwf   varA
        movlw   0x01
```

```
        movwf   varB

btmp    movf    T0,w
        andwf   varB,w
        btfss   STATUS,ZERO
        goto    t1

tt0     comf    varB,w
        andwf   EC,f
        andwf   _DEBZB+0x11,f
        goto    ftmp

t1      movf    EC,w
        andwf   varB,w
        btfss   STATUS,ZERO
        goto    ec11

ec0     movf    varB,w
        iorwf   EC,f

        bsf     STATUS,RP0
        bsf     EECON1,RD
        bcf     STATUS,RP0
        movf    EEDATA,w
        movwf   INDF

        goto    ftmp

ec11    clrw
        addwf   INDF,w
        btfss   STATUS,ZERO
        goto    ftmp
        movf    varB,w
        iorwf   _DEBZB+0x11,f

ftmp    rlf     varB,f
        bcf     varB,0
        incf    FSR,f
        incf    EEADR,f
        decfsz  varA,f
        goto    btmp
        return

;=== iniport ===
;           ###
;           U --
;initialise les ports avec 0 en sortie et les pull-up
iniport
        bsf     STATUS,RP0
        bcf     OPT___,RBPU
        clrf    TRISA
        bcf     STATUS,RP0

        clrf    _DEBZB+0xc
        call    sorties
        return

;=== entrees ===
;           ###
;           U --
;remplit la zone des bit pour les entrees
```

```
entrees
        comf    PORTB,w
        movwf   _DEBZB+0x08
        return

;=== sorties ===
;           ###
;           U --
;met a jour les sorties suivant la zone des bits
sorties
        movf    _DEBZB+0xc,w
        movwf   PORTA
        return

;=== initab ===
;           ###
;           U -- PT
;prepare le pointeur pour pointer sur le premier element
initab
        movlw   0x01
        movwf   PCLATH
        movlw   0x80
        movwf   PT
        return

;=== acqtab ===
;           ### --> ORDRE
;           ### --> VALEUR
;           U -- PT
;remplit ordre et valeur et pointe suer le suivant
acqtab
        movf    PT,w
        call    rechtab
        movwf   ORDRE
        incf    PT,f
        movf    PT,w
        call    rechtab
        movwf   VALEUR
        incfsz  PT,f
        return
        incf    PCLATH,f
        return
rechtab movwf   PCL

;=== traite ===
;           ###
;           U -- PCLAT2,varA,varB,TEST,SAUT,ETAPE,varC
;traite le graphcet
traite
        call    initab
btraite call    acqtab
        movf    PCLATH,w
        movwf   PCLAT2
        movlw   0x01
        movwf   PCLATH
        movf    ORDRE,w
        addlw   _SWITCH
        movwf   PCL

traitee movf    PCLAT2,w
        movwf   PCLATH
```

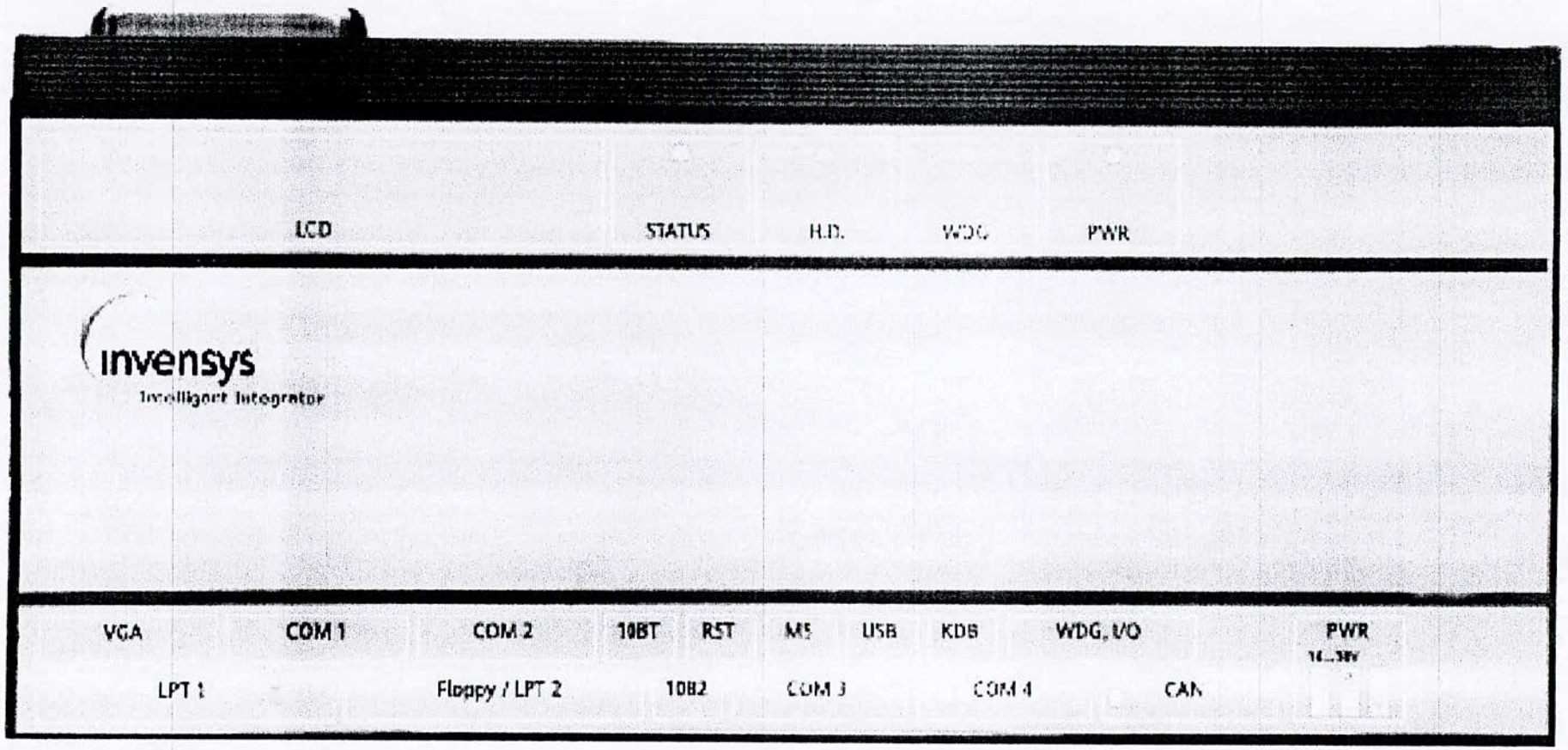
```
        goto    btraite

traitef return

_0_1    call    vb
        movf    VALBIT,w
        movwf   TEST
        goto    traitee

_1_1n   call    vb
        movf    VALBIT,w
        xorlw   0x01
        movwf   TEST
        goto    traitee

_2_a    call    vb
        movf    VALBIT,w
        andwf   TEST,f
        goto    traitee

_3_an   call    vb
        movf    VALBIT,w
        xorlw   0x01
        andwf   TEST,f
        goto    traitee

_4_o    call    vb
        movf    VALBIT,w
        iorwf   TEST,f
        goto    traitee

_5_on   call    vb
        movf    VALBIT,w
        xorlw   0x01
        iorwf   TEST,f
        goto    traitee

_6_x    call    vb
        movf    VALBIT,w
        xorwf   TEST,f
        goto    traitee

_7_xn   call    vb
        movf    VALBIT,w
        xorlw   0x01
        xorwf   TEST,f
        goto    traitee

_8_ega  btfsc   TEST,0
        goto    ega1
ega0    call    cb
        goto    traitee
ega1    call    sb
        goto    traitee

_9_eti
_a_et   movf    VALEUR,w
        movwf   ETAPE
        call    vb
        movf    VALBIT,w
        movwf   SAUT
```

```
        goto    traitee

_b_va   btfsc   SAUT,0
        btfss   TEST,0
        goto    traitee
        call    sb
        movf    ETAPE,w
        movwf   VALEUR
        call    cb
        goto    traitee


;=== table de saut switch ===
        ORG     0x170
switch
        goto    _0_l
        goto    _1_ln
        goto    _2_a
        goto    _3_an
        goto    _4_o
        goto    _5_on
        goto    _6_x
        goto    _7_xn
        goto    _8_ega
        goto    _9_eti
        goto    _a_et
        goto    _b_va
        goto    traitef
        goto    traitef
        goto    traitef
        goto    traitef

;=== graphcet en code a pqrtir de 0x200 ===

        END
```

ANNEXE 6
Vue globale de l'automate ACCOS

APV

Improving Process Profitability...Continuously℠



invensys

An Invensys company

ANNEXE 7
Séquence 3200 "SUPERVISION CAUDIERE"

```
                    /supervision de la chaudière/
/Programme d'emission
DEBUT, DCOM                 CH01ERR
      COMD      1.10.2
     DCOM      1.10.2
     MSATTE    1
     COMD      1.10.2
     MSATTE    1
     DCOM      1.10.2
     MSATTE    1
     COMD      1.10.2
     MSATTE    1
     DCOM      1.10.2
     MSATTE    1
     DCOM      1.10.2
     MSATTE    1
     DCOM      1.10.2
     MSATTE    1
     COMD      1.10.2
     MSATTE    1
     DCOM      1.10.2
     MSATTE    1
        COMD      1.10.2
   /
       /programmme de reception:
       /
/1 octet
DEBUT1,      BCDIN 1.10.1        1             R1.580
       IFBT          1          R1.580        DEBUT1
       MSATTE        1
       BCDIN 1.10.1        1          R1.581
       MSATTE        1
       BCDIN 1.10.1        1          R1.582
       MSATTE        1
       BCDIN 1.10.1        1          R1.583
       MSATTE        1
       BCDIN 1.10.1        1          R1.584
       MSATTE        1
       BCDIN 1.10.1        1          R1.585
       MSATTE        1
       BCDIN 1.10.1        1          R1.586
       MSATTE        1
       BCDIN 1.10.1        1          R1.587
       MSATTE        1
       BCDIN 1.10.1        1          R1.588
       MSATTE        1
       BCDIN 1.10.1        1          R1.589
       IFNBT         1          R1.589        ERROR
/2 octet
DEBUT2,      BCDIN 1.10.1        1             R1.590
       IFBT          1          R1.580        DEBUT2
       MSATTE        1
       BCDIN 1.10.1        1          R1.591
       MSATTE        1
       BCDIN 1.10.1        1          R1.592
       MSATTE        1
       BCDIN 1.10.1        1          R1.593
       MSATTE        1
       BCDIN 1.10.1        1          R1.594
       MSATTE        1
       BCDIN 1.10.1        1          R1.595
       MSATTE        1
       BCDIN 1.10.1        1          R1.596
       MSATTE        1
       BCDIN 1.10.1        1          R1.597
       MSATTE        1
       BCDIN 1.10.1        1          R1.598
       MSATTE        1
```

```
        BCDIN 1.10.1        1           R1.599
        IFNBT       1           R1.599      ERROR
        IFBT        1           R1.598      ERROR
/Programme de rafracichissement des données de l'OGS

        IFBT        1           R1.582      NB
        DCOM CH01LS01                       /NIVEAU BAS CHAUDIERE
        GOTO        NH1
NB,     COMD CH01LS01                       /NIVEAU BAS CHAUDIERE
NH1,    IFBT        1           R1.583      NH
        DCOM CH01HS01                       /NIVEAU D'EAU HAUT CHAUDIERE
        GOTO        NBP1
NH,     COMD CH01HS01                       /NIVEAU D'EAU HAUT CHAUDIERE
NBP1,   IFBT        1           R1.584      NBP
        DCOM CH01LP01                       /PRESSION BAS CHAUDIERE
        GOTO        NHP1
NBP,    COMD CH01LP01                       /PRESSION BAS CHAUDIERE
NHP1,   IFBT        1           R1.585      NHP
        DCOM CH01HP01                       /PRESSION HAUTE CHAUDIERE
        GOTO        DEF1
NHP,    COMD CH01HP01                       /PRESSION HAUTE CHAUDIERE
DEF1,   IFBT        1           R1.586      DEF
        DCOM CH01DEF                        /DEFAUT GENERAL CHAUDIERE
        GOTO        SEQ1
DEF,    COMD CH01DEF                        /DEFAUT GENERAL CHAUDIERE
SEQ1,   IFBT        1           R1.591      SEQ
        DCOM CH01SEQ                        /SEQUENCEUR CHAUDIERE
        GOTO        POMPE1
SEQ,    COMD CH01SEQ                        /SEQUENCEUR CHAUDIERE
POMPE1,     IFBT        1       R1.592      POMPE
        DCOM CH01XME01                      /POMPE ALIMENTATION EAU CHAUDIERE
        GOTO        ATT
POMPE,      COMD CH01XME01                  /POMPE ALIMENTATION EAU CHAUDIERE
ATT,    ATTE        10
        GOTO DEBUT
ERROR,      COMD CH01ERR                    /ERREUR DE COMMUNICATION
        ATTE        10
        GOTO        DEBUT
        STOP
```

ANNEXE 8
Séquence 1.68 "OEC CHAUDIERE"

```
        /-----------/
        / OEC CHAUDIERE /
        /-----------/

        ISSQ   1.68                     /OEC CHAUDIERE
        IFMI   T1.2             ARCH    /MEMO. OEC ARRET
        OEMESS      CHAUDIERE
        IFSAC  1.3200           INTER
        OETSEQ      1.3200              INEX    /SUPERVISION DE LA CHAUDIERE
ARCH,   OEMESS      CHAUDIERE
        OEMODE      1
        IFSNAC      1.3200              INTER
        OETSEQ      1.3200              INEX
        GOTO                    FIN_AN

        /-------------------------/
        / INDICATION DU DEFAUT D'ANALYSE OEC /
        /-------------------------/
        /
INCOR,  CHAR  2     R1.50               / OEC ERRONNE
        ABEY
        /
INTER,  CHAR  3     R1.50               / OEC INTERDIT
        ABEY
        /
INEX,   CHAR  4     R1.50               / OEC INEX
        ABEY
        /
PORT,   CHAR  5     R1.50               / OEC DEFAUT PORT COMMS
        ABEY
        /
PWD1,   CHAR  6     R1.50               / OEC DEFAUT MOT DE PASSE
        ABEY
        /
DIA300, CHAR  7     R1.50               / OEC DEFAUT DIALOGUE A300
        ABEY
        /
DAT300,     CHAR  8     R1.50               / OEC DEFAUT DONNEES A300
        ABEY
        /
DEFDDP,     CHAR  21    R1.50               / OEC DEFAUT PONTAGE %i
        ABEY
        /
DEFELT,     CHAR  23    R1.50               / OEC DEFAUT ELEMENT %i
        ABEY
        /
DEFBV,  CHAR  32    R1.50               / OEC DEFAUT BARRIERE VAPEUR %d
        ABEY
        /
DEFNH,      CHAR  40    R1.50               / OEC DEFAUT NIVEAU HAUT
        ABEY
        /
DEFNM,      CHAR  41    R1.50               / OEC DEFAUT NIVEAU MOYEN
        ABEY
        /
DEFNB,      CHAR  42    R1.50               / OEC DEFAUT NIVEAU BAS
        ABEY
        /
DEFNIV,     CHAR  43    R1.50               / OEC DEFAUT NIVEAU
        ABEY
        /
DEFEVE,     CHAR  44    R1.50               / OEC DEFAUT EVENT
        ABEY
        /
DEFVM,      CHAR  45    R1.50               / OEC DEFAUT VANNE MANUELLE
        ABEY
        /
```

```
DEFTH,    CHAR  46   R1.50          / OEC DEFAUT TROU D'HOMME
      ABEY
      /
DEFTEM,   CHAR  47   R1.50          / OEC DEFAUT TEMPERATURE
      ABEY
      /
DEFPRE,   CHAR  48   R1.50          / OEC DEFAUT PRESSION
      ABEY
      /
DEFFIL,   CHAR  49   R1.50          / OEC DEFAUT FILTRE
      ABEY
      /
DEFDIA,   CHAR  60   R1.50          / OEC DEFAUT DIALOGUE
      ABEY
      /
ETAINC,   CHAR  61   R1.50          / OEC DEFAUT ETAT INCOMPATIBLE
      ABEY
      /
PROINC,   CHAR  62   R1.50          / OEC DEFAUT PRODUIT INCOMPAT.
      ABEY
      /
DEFACQ,   CHAR  63   R1.50          / OEC DEFAUT ACQUIT
      ABEY
      /
DEFREC,   CHAR  64   R1.50          / OEC DEFAUT RECETTE
      ABEY
      /
DEFSEL,   CHAR  66   R1.50          / OEC DEFAUT SELECTION
      ABEY
      /
DEFQU,    CHAR  67   R1.50          / OEC DEFAUT QUANTITE
      ABEY
      /
DEFREM,   CHAR  91   R1.50          / OEC REMPLISSAGE EN COURS
      ABEY
      /
DEFSOU,   CHAR  92   R1.50          / OEC SOUTIRAGE EN COURS
      ABEY
      /
NETTOY,   CHAR  93   R1.50          / OEC NETTOYAGE EN COURS
      ABEY
      /
NEPOCC,   CHAR  94   R1.50          / OEC STATION NETTOYAGE OCCUPEE
      ABEY
      /
NEPNP,    CHAR  95   R1.50          / OEC STATION NEP NON PRETE
      ABEY
      /
NONATT,   CHAR  96   R1.50          / OEC NON EN ATTENTE
      ABEY
      /
NONPRO,   CHAR  97   R1.50          / OEC NON PROPRE
      ABEY
      /
NONSAN,   CHAR  98   R1.50          / OEC NON SANITE
      ABEY
      /
NONSTE,   CHAR  99   R1.50          / OEC NON STERILE
      ABEY
      /
FIN1,     IFCNV R1.48  SE   1     FIN_AN/ NIVEAU DE MOT DE PASSE
      GOTO                        PWD
      /
FIN2,     IFCNV R1.48  SE   2     FIN_AN/ NIVEAU DE MOT DE PASSE
      GOTO                        PWD
      /
FIN3,     IFCNV R1.48  SE   3     FIN_AN/ NIVEAU DE MOT DE PASSE
      GOTO                        PWD
```

```
        /
FIN4,   IFCNV R1.48  SE     4       FIN_AN/ NIVEAU DE MOT DE PASSE
        GOTO                        PWD
        /
FIN5,   IFCNV R1.48  SE     5       FIN_AN/ NIVEAU DE MOT DE PASSE
        GOTO                        PWD
        /
FIN6,   IFCNV R1.48  SE     6       FIN_AN/ NIVEAU DE MOT DE PASSE
        GOTO                        PWD
        /
FIN7,   IFCNV R1.48  SE     7       FIN_AN/ NIVEAU DE MOT DE PASSE
        /
PWD,    CHAR  6      R1.50                  / OEC DEFAUT MOT DE PASSE
        ABEY
        /
FIN_AN,       CHAR  999    R1.50                / OEC OK
        /
        ABEY
        STOP
```

Bibliographie

[1] C.Tavernier, les microcontrôleurs PIC, Application, Dunod, 2000

[2] Microchip, Datasheet 16F8X, DSC 30430, 1998 Microchip technology Inc.

[3] G.Michel, Les A.P.I architecture et applications des automates programmables industriels, Dunod, 1988

[4] S.Thelliez, J.M.Toulotte, Applications industrielles du Grafcet, Eyrolles, 1985

[5] Bigonoff, La programmation des PICs, www.von-info.ch/PIC/bigonoff_01.htm.

[6] Y.Lecourtier, B.Saint-Jean, Introduction aux automatismes industriels, Masson, 1989

[7] G.Gonda, M.Quoirin, Automates programmables, Elan, 1993

[8] Bernard Reeb, Développement des Grafcets, ellipes, 1999

[9] C.Tavernier, Applications industrielles des PIC, Dunod, 2001

[10] Electronique pratique, "Les drivers RS232", N°210 Janvier 1997

[11] SGS-Thomson Microelectronics, Datasheet L7800AB/AC Series, Janvier 1997

[12] Maxim, "+5V-powered, Multichanel RS232 Drivers/Receivers, 19-4323 Rev 10;8/01

[13] SGS-Thomson microelectronics, Datasheet ULN2801A- ULN2802A- ULN2803A- ULN2804A- ULN2805A, Avril 1993

[14] U.Delmas, "Automates programmables industriels API0 et API0i", www.chez.com/udelmas/api0et0i.html

[15] U.Delmas, "Un automate programmable industriel à PIC16F84", www.chez.com/udelmas/autopic2.htm

[16] Manuel d'exploitation et d'entretien, ABB Alstom