TABLE DES MATIERES

LISTE DES FIGURES

API	:	Automate Programmable Industriel
CdCF	:	Cahier des charges fonctionnel
CISC	:	Complex instruction set computer
CMP	:	Chip-level multiprocessing
CPU	:	Central processing unit
CTS	:	Clear to send
DB	:	Data blocs
DCD	:	Data carrier detecte
DSP	:	Digital signal processor
DSR	:	Data set read
DTR	:	Data terminal ready
EAROM	:	Electrically alterable read only memory
EEPROM	:	Electric erasable prom
EPROM	:	Erasable prom
FB	:	Function blocs
FPU	:	Floating point unit
FPU	:	Floating point unit
GEMMA	:	Guide d'etude des modes de marches et d'arrets
GND	:	Ground
GRAFCET	:	Graphe de fonctionnement des commandes des etapes et transitions
GRAFCET	:	Graphe de commande etapes-transitions
IEC 61131-3	:	International electrotechnic commission
ILP	:	Instruction level parallelism
IRQ	:	Interrupt request
ISA	:	Instructions set architecture
LCD	:	Liquid cristal display
MIPS	:	Million d'instructions par seconde
MOA	:	Maitrise d'ouvrage
MOE	:	Maitrise d'œuvre
NUMA	:	Non uniform memory access
PB	:	Program blocs
PIC :	:	Programmable interrupt controller
PLC	:	Programmable logic controller
POU	:	Program organisation unit
REPROM	:	Reprogrammable eprom
RI	:	Ring indicator
RISC	:	Reduced instruction set computer
RTS	:	Request to send
RX	:	Reciver data
SB	:	Sequential blocs ou sfc : sequential flow chart
SMP	:	Symetric multiprocessing
SMT	:	Simultaneous multithreading
TLP	:	Thread level parallelism

TOR	:	Tout ou rien
TX	:	Transmit data
UAL	:	Unite arithmetique et logique
UL	:	Unite logique
USART	:	Universal synchronous asynchronous reciever transmitter
VLIW	:	Very long instruction word

L'eau, en tant que ressource stratégique et vitale, nécessite une bonne gestion afin d'optimiser son exploitation.

Il est alors indispensable de faire appel à des techniques efficaces de supervision au niveau des installations hydrauliques et de leurs accessoires.

En effet la technologie d'enregistrement des données provenant d'une pompe permet de contrôler le fonctionnement et le rendement de la station de pompage pendant toute l'année.

La pompe représente l'un des éléments les plus importants dans une station de pompage comme son nom l'indique.

Mais le problème qui se pose c'est que le prix de l'énergie (électricité) est cher pendant les heures de pointe, c'est pour cela qu'il est préférable de procéder au pompage hors de cette période de temps.

En effet, notre but est de concevoir une carte qui permet la gestion et l'optimisation du fonctionnement des stations de pompage.

L'idée de ce projet est nouvelle, en effet il n'existe pas sur le marché un tel modèle de carte de commande.

En plus l'originalité de notre étude est d'essayer à partir d'une carte simple et surtout non coûteuse d'optimiser le fonctionnement des ouvrages d'extraction d'eau suivant les tarifs JIRAMA (Jour, Pointe et Nuit).

Mais aussi notre carte permet le stockage des données de consommation durant toute une année. Ces données peuvent être transmises à un PC à l'aide d'un simple câble série. (RS 232) pour un éventuel traitement externe (traçage des courbes de modulation…).

Maintenant que nous avons présenté l'idée générale de notre projet, nous allons présenter la démarche que nous avons envisagée en indiquant le contenu des différentes parties qui constituent cet ouvrage.

Le premier chapitre consistera à faire une présentation des généralités sur les Automates Programmable Industriels.

Dans le deuxième chapitre, nous décrirons l'aspect Matériel d'un Automate Programmable Industriel.

Dans le troisième chapitre nous attaquerons l'aspect Logiciel qui va piloter l'Automate Programmable Industriel.

Le quatrième chapitre comportera l'élaboration de différentes étapes pour réaliser un Automate Programmable Industriel.

Enfin, le dernier chapitre contiendra l'Application de l'Automate Programmable Industriel, qui est basée sur la conception d'une carte de commande pour l'«Optimisation de fonctionnement d'une station de pompage en fonction du prix de la JIRAMA », et la simulation de ce circuit avec le logiciel PROTEUS version 7.6.

CHAPITRE1 : GENERALITES SUR LES AUTOMATES PROGRAMMABLES INDUSTRIELS

1.1 Historiques :

L'histoire des systèmes automatiques peut se diviser en trois périodes :

La première période, que l'on peut qualifier de préhistoire de l'automatique s'étend de l'antiquité au milieu du siècle dernier. Des inventeurs géniaux ont conçu des systèmes automatiques de manière purement intuitive.

Les systèmes de commande sont généralement issus des lois naturelles de la physique. Dès 250 avant J.C. nous avons des exemples avec l'horloge automatique à eau de Ktésybios (voir figure 1), la lampe à huile de Philon de Bizance et la machine à doser le vin de Héron d'Alexandrie.

Plus tard, Réaumur, Watt et son régulateur en 1788 (voir figure 2), Jacquard et son métier à cartes perforées, font progresser l'automatisation.

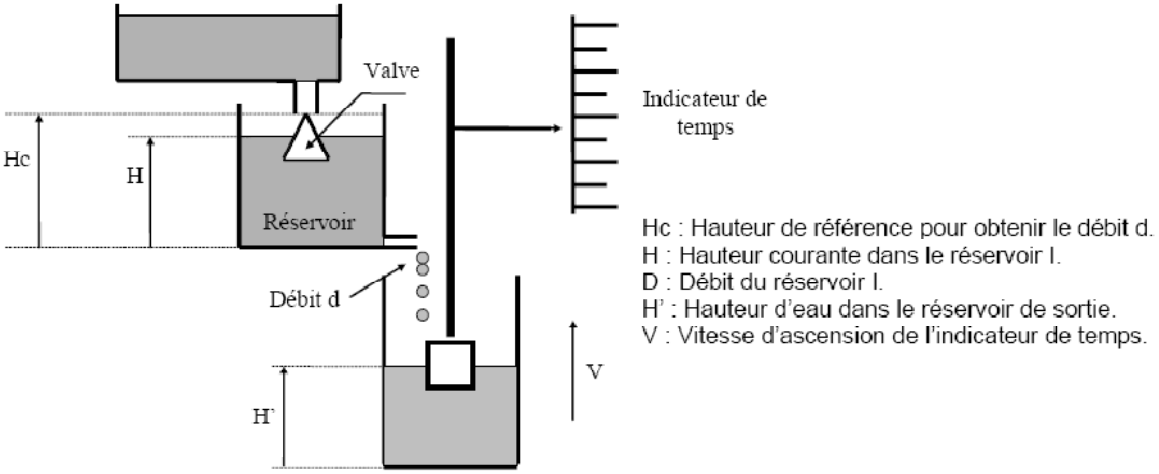


Figure 1.1 : CLEPSYDRE (horloge à eau) de KTESYBIOS

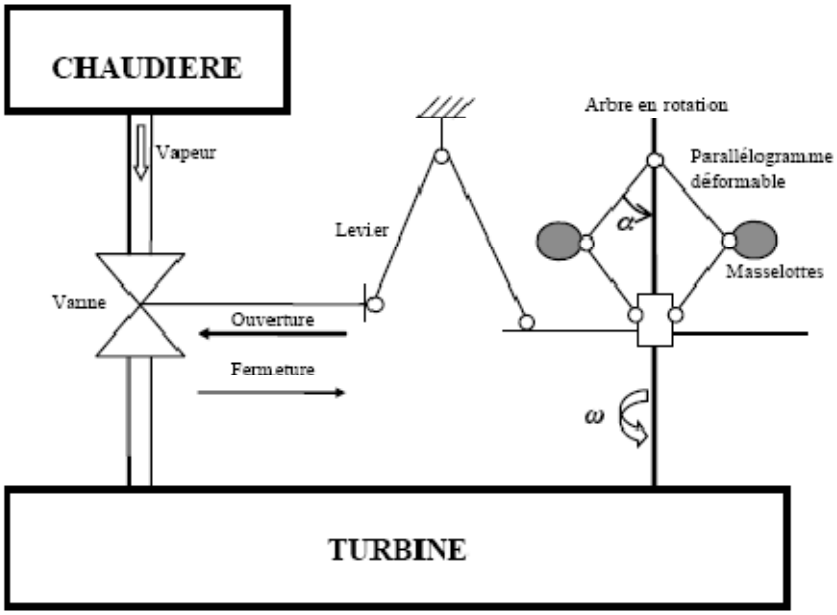


Figure 1.2 : REGULATEUR DE WATT

Le régulateur de WATT a pour but de maintenir la vitesse de rotation ω d'une turbine constante. Le principe consiste à contrôler le débit de vapeur de la vanne à partir de l'effet de la force centrifuge des masselottes due à la rotation de l'arbre et d'un système mécanique déformable.

La deuxième période à partir du milieu du XIXème siècle est caractérisée par la théorie du bouclage (retour) et les applications de l'algèbre de Boole (systèmes combinatoires). L'étude des systèmes est abordée d'un point de vue analytique. Des chercheurs du nom de **Nyquist**, **Bode**, **Black**, **Nichols**, **Hall**, **Evans** ont laissé leur nom à des représentations de systèmes bouclés et ont publié leurs résultats à la fin de la seconde guerre mondiale.

La troisième période débute avec les années cinquante. L'apparition de calculateurs numériques révolutionne le monde de l'automatique. La puissance de calcul disponible fait naître les méthodes dites de l'automatique « moderne ».

Chez Renault, les premiers robots datent des années 70 pour l'assemblage de tôles de carrosserie.

Les années 80 ont vu ensuite le développement de robots hydrauliques avec les premiers ateliers complètement robotisés (une centaine de robots à Douai en 1981 pour l'assemblage de la R9).

Les années 90 ont ensuite vu le développement des robots tout électrique à moteurs auto synchrones et l'apparition de contrôleurs d'axe intégrant l'ensemble des fonctions de contrôle / commande d'une chaîne mono-axe (traitement et surveillance séquentielle, carte d'axe et variateur, mesures et sécurité).

1.2 Définitions : API (ou PLC : Programmable Logic Controller) :

Définition1 :

L'Automate Programmable Industriel (API) est un appareil électronique programmable, adapté à l'environnement industriel, qui réalise des fonctions d'automatisme pour assurer la commande de pré actionneurs et d'actionneurs à partir d'informations logique, analogique ou numérique.

Définition2 :

Un Automate Programmable Industriel (API) est une machine électronique programmable par un personnel non informaticien et destiné à piloter en ambiance industrielle et en temps réel des procédés ou parties opératives.

Définition3 :

Un automate programmable industriel (API) est une machine électronique programmable utilisée pour piloter des systèmes automatisés. Sa flexibilité explique son large domaine d'utilisation, qui comporte certaines applications critiques, où des erreurs de programmation sont susceptibles de causer des dommages humains ou matériels.

Automatique :

Science et technique de l'automatisation, qui étudie les méthodes scientifiques et les moyens technologiques utilisés pour la conception et la construction des systèmes automatiques.

Automatisation :

- L'automatisation est considérée comme l'étape d'un progrès technique où apparaissent des dispositifs techniques susceptibles de seconder l'homme, non seulement dans ses efforts musculaires, mais également dans son travail intellectuel de surveillance et de contrôle. C'est alors l'exécution automatique de tâches industrielles, administratives ou scientifiques, sans intervention humaine.

- L'automatisation de la production consiste à transférer tout ou partie des tâches de coordination, auparavant exécutées par des opérateurs humains, dans un ensemble d'objets techniques appelé PARTIE COMMANDE .La Partie Commande mémorise le SAVOIR FAIRE des opérateurs pour obtenir la suite des actions à effectuer sur les matières d'œuvre afin d'élaborer la valeur ajoutée. Elle exploite un ensemble d'informations prélevées sur la Partie Opérative pour élaborer la succession des ordres nécessaires pour obtenir les actions souhaitées.

Système :

Dans le langage scientifique, le mot « système » peut être défini de la manière suivante: un système consiste en une combinaison de parties (électriques, pneumatiques, thermiques, mécaniques, ..) qui se coordonnent pour concourir à un résultat.

Les entrées et les sorties d'un système :

Les entrées sont les signaux qui apportent au système les informations du milieu extérieur. Les sorties fournissent la réponse du système relative aux entrées. On peut parler de causes (entrées) et d'effets (sorties).

La Partie Opérative :

La Partie Opérative est un ensemble d'éléments ou de dispositifs opératifs plus ou moins mécanisés procédant au traitement des matières d'œuvre afin d'élaborer progressivement la valeur ajoutée.

La Partie Commande:

La Partie Commande d'un système isolé est un ensemble de composants et de constituants de traitement de l'information, destiné :

- à coordonner la succession des actions sur la Partie Opérative,
- à surveiller son bon fonctionnement,
- à gérer les dialogues avec les intervenants,
- à gérer les communications avec d'autres systèmes,
- à assurer le traitement des données et des résultats relatifs au procédé, aux matières d'œuvre, aux temps de production, à la consommation énergétique... (gestion technique).

La commande d'un système :

La « commande » d'un système consiste à exercer à partir des signaux d'entrée, une influence sur le système de manière à obtenir en sortie un comportement déterminé. Ce comportement peut être assimilé à la conduite d'une ou plusieurs grandeurs physiques telles que vitesse, accélération, position, température, effort, ...

La commande manuelle :

Lorsque l'influence sur le système est exercée par l'homme, la commande est dite manuelle.

La commande automatique :

Lorsque l'homme est remplacé par des dispositifs techniques autonomes, la commande est dite automatique.

Un système automatisé :

Un système automatisé est alors un système technique pour lequel tout ou une partie du savoir-faire est confié à une machine.

Différentes catégories de systèmes automatisés

- ***Phénomène de retour (rétro action) :***

Cette notion est fondamentale dans l'étude des systèmes automatiques. Le retour consiste à prendre en compte de manière permanente la situation du système afin d'élaborer le signal de commande. Un système en boucle fermée sera caractérisé par un dispositif de retour. Ainsi, dans ce type de système, le signal de commande sera fonction à la fois du signal d'entrée et du signal de sortie.

- ***Systèmes à contrôle logique :***

Dans ce type de système, les signaux à contrôler sont exclusivement des signaux logiques. Le système obéit à un processus préétabli qui envisage toutes les possibilités d'évolution.
On distingue deux types de systèmes à contrôle logique :

- Un signal d'entrée logique (ou une combinaison de signaux d'entrée) conduit invariablement au même signal de sortie. On parlera de systèmes à logique combinatoire.
- Le signal de sortie est élaboré à partir d'un signal d'entrée logique (ou d'une combinaison de signaux) et doit prendre en compte une chronologie pré-établie qui

porte sur un nombre fini d'opérations. Dans ce cas on parlera de systèmes à évènements discrets ou encore de systèmes séquentiels.

- ***Systèmes à contrôle analogique ou numérique :***

Dans cette catégorie de système, les signaux à contrôler sont des grandeurs de type analogique ou numérique. Dans ce cas, toutes les situations possibles n'étant pas prévisibles (arrivées d'une perturbation par exemple), le déroulement des opérations ne peut pas être prédéterminé. Un tel système, s'il est munis d'un dispositif de retour (c'est à dire qu'une mesure de la situation est en permanence considérée dans l'élaboration du signal de commande) sera dit système asservi.

Lorsque les variables traitées sont à variation continue, on parlera de système continu ou analogique.

D'une manière générale, la plupart des systèmes des systèmes physiques sont continus.

Lorsque les variables sont traitées par un calculateur, elles sont échantillonnées dans le temps, on parlera alors de système discret ou numérique. Les systèmes informatiques font partie de cette catégorie.

Information :

Ensemble de données pouvant être traitées par un système**.** L'information est nécessaire pour transmettre des décisions et pour disposer de renseignements sur l'état du système.

Information discrète :

Signaux logiques :

Un signal logique est une grandeur qui n'admet que 2 états : L'état « travail » (tout) et l'état « repos » (rien). Chacun des ces états est décrit à l'aide des variables booléennes 0 et 1.

Signaux numériques :

L'information numérique discrète découle de la nécessité de discrimination d'une même information à l'aide d'un nombre restreint de variables. Une information numérique est le résultat d'un codage.

Information analogique :

Un signal analogique est une grandeur physique (vitesse, position, température..) qui peut prendre un nombre infini de valeurs avec une variation continue.

Matières d'œuvre:

Une matière d'œuvre peut se présenter sous plusieurs formes. Par exemple :

- un PRODUIT, c'est-à-dire de la matière, à l'état solide, liquide ou gazeux, et sous une forme plus ou moins transformée :

- des objets techniques : lingot, roulement, moteur, véhicule...

- des produits chimiques : pétrole, éthylène, matière plastique...

- des produits textiles : fibre, tissu, vêtement...

- des produits électroniques : transistor, puce, microprocesseur, automate programmable...

- etc.

- qu'il faut : concevoir, produire, stocker, transporter, emballer, utiliser...

- de l'ENERGIE

- sous forme : électrique, thermique, hydraulique...

- qu'il faut : produire, stocker, transporter, convertir, utiliser...

- de l'INFORMATION

- sous forme écrite, physique, audiovisuelle...

- qu'il faut : produire, stocker, transmettre, communiquer, décoder, utiliser...

- des ETRES HUMAINS

- pris individuellement ou collectivement

- qu'il faut : former, informer, soigner, transporter, Servir...

Valeur ajoutée:

La Valeur Ajoutée à ces matières d'œuvre est l'OBJECTIF GLOBAL pour lequel a été défini conçu, réalisé, puis éventuellement modifié, le système. Cette Valeur Ajoutée peut résulter par exemple :

- d'une MODIFICATION PHYSIQUE des matières d'œuvre

- traitement mécanique : usinage, formage, broyage, impression…

- traitement chimique ou biologique

- conversion d'énergie

- traitement thermique : cuisson, congélation...

- traitement superficiel : peinture, teinture...

- d'un ARRANGEMENT PARTICULIER, sans modification des matières d'œuvre

- montage, emballage, assemblage...

- couture, collage...

- d'une MISE EN POSITION particulière, ou d'un TRANSFERT, de ces matières d'œuvre

- manutention, transport, stockage

- commerce

- communication

- d'un PRELEVEMENT D'INFORMATION sur ces matières d'œuvre

- contrôle mesure lecture examens...

Contexte et valeur ajoutée:

La NATURE, la QUANTITE et la QUALITE de la valeur ajoutée peuvent varier pour tenir compte de l'évolution des besoins de la société dans laquelle s'insère le système. Ce qui peut conduire à modifier le système, voire l'abandonner pour en construire un nouveau.

L'environnement, c'est-à-dire le CONTEXTE physique, social, économique, politique, ... joue un rôle essentiel dans le fonctionnement du système et influe sur la qualité et/ou la quantité de la Valeur Ajoutée.

1.3 Objectifs de l'automatisation:

L'automatisation permet d'apporter des éléments supplémentaires à la valeur ajoutée par le système. Ces éléments sont exprimables en termes d'objectifs par :

- accroître la productivité du système c'est-à-dire augmenter la quantité de produits élaborés pendant une durée donnée. Cet accroissement de productivité exprime un gain de valeur ajoutée sous forme :
- d'une meilleure rentabilité,
- d'une meilleure compétitivité.
- améliorer la flexibilité de production ;
- améliorer la qualité du produit grâce à une meilleure répétabilité de la valeur ajoutée
- s'adapter à des contextes particuliers :
- adaptation à des environnements hostiles pour l'homme (milieu salin, spatial, nucléaire...),
- adaptation à des tâches physiques ou intellectuelles pénibles pour l'homme (Manipulation de lourdes charges, tâches répétitives parallélisées...),
- augmenter la sécurité
  Remplacement des agents humains par des machines (guichets automatiques de banques, distributeurs automatiques de carburants)
- minimiser les tâches quotidiennes (arrosage automatique)

1.4 Système de production:

Un SYSTEME DE PRODUCTION est un système à caractère industriel possédant les caractéristiques suivantes :

- l'obtention de la valeur ajoutée présente, pour un ensemble de matières d'œuvre donné, un caractère reproductible,
- la valeur ajoutée peut être exprimée et quantifiée en termes économiques Un système de production répond au besoin d'élaborer des produits, de l'énergie ou de l'information à un coût rentable pour l'utilisateur du système.

L'élaboration progressive de la valeur ajoutée sur les matières d'œuvre est obtenue :

- au moyen d'un ensemble d'éléments ou de dispositifs opératifs, appelés PARTIE OPERATlVE et plus ou moins mécanisés,
- par l'action, à certains moments, d'opérateurs humains et/ou de dispositifs de commande pour assurer la coordination des dispositifs opératifs.

Exemples de système de production :

- usine de fabrication chimique, métallurgique, électronique...

- société de service (informatique...), groupe de presse, banque...

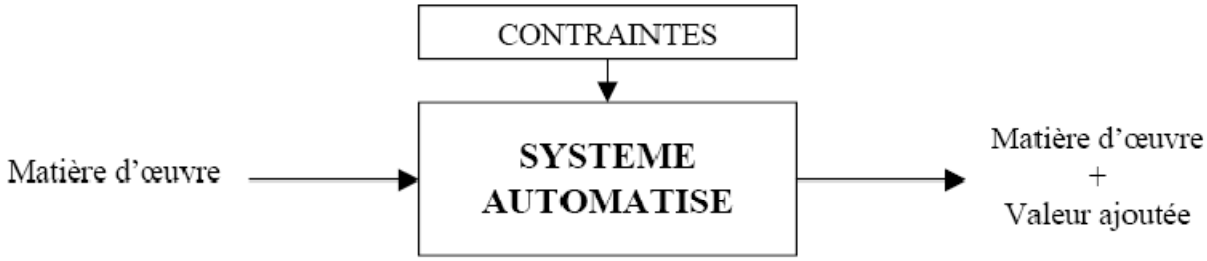


Figure 1.3 : Système de production

La figure ci-dessous illustre le fonctionnement d'un Système Automatisé de Production, tel que l'on en rencontre notamment, mais pas exclusivement, dans l'industrie manufacturière (chaînes d'usinage, de montage, de conditionnement).

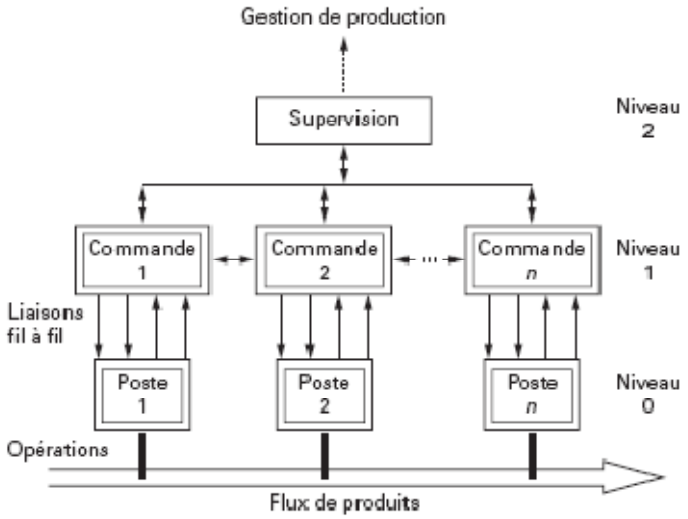


Figure 1.4 : Fonctionnement d'un Système Automatisé

1.5 L'architecture générale d'un automate programmable

1.5.1 Structure des automates :

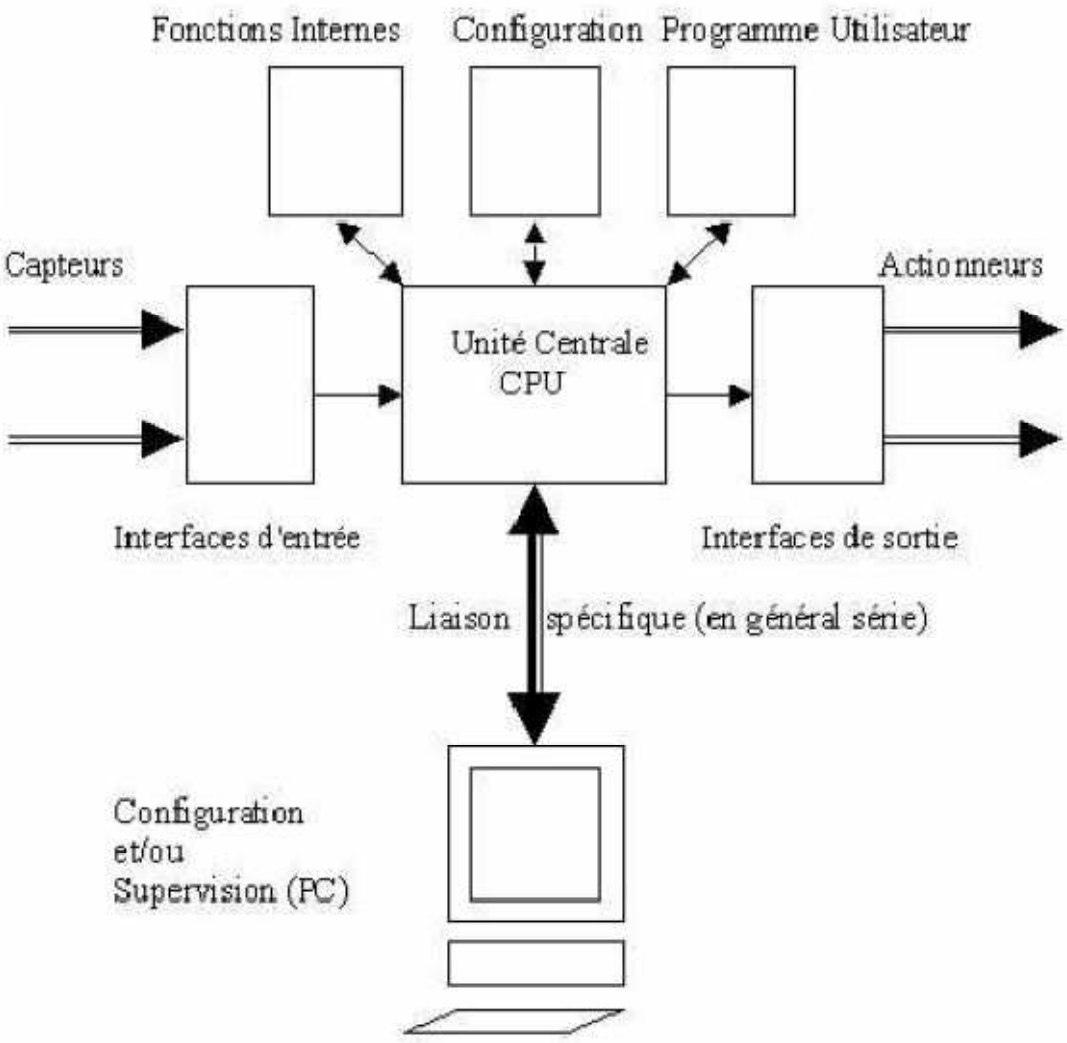


Figure 1.5 : Structure générale des automates

- ***Unité centrale :*** elle gère l'ensemble du processus, elle contient le processeur, les mémoires vives et des mémoires mortes pour une taille débutant à 40 koctets. Elle est programmable directement par console ou par le biais d'une liaison série et d'un logiciel adapté. Cette CPU peut être en RUN (elle exécute le programme), en STOP (toutes les sorties sont au repos, contacts ouverts).
- ***Configuration :*** elle contient les paramètres liés à la structure de l'API et à la structure du réseau informatique.
- ***Fonctions Internes :*** ce sont des fonctions pré-programmées livrées avec l'API qui permettent par exemple d'assurer des temporisations, des régulations… Ces fonctions

peuvent être résidentes dans l'automate ou disponible dans le logiciel de programmation.

- ***Programme Utilisateur :*** c'est la loi de commande, il assure la gestion des sorties en fonction de l'état des entrées et éventuellement du temps. Ce programme est exécuté sous forme cyclique par l'API, le temps de cycle est bien sûr dépendant de la taille du programme et ne doit pas excéder la centaine de millisecondes.
- ***Supervision :*** c'est un ordinateur standard. Il contient le logiciel de programmation (Orphée pour April et Step7 pour Siemens). Ce logiciel permet d'écrire le programme, de le compiler et de le transférer à l'automate.

L'ordinateur peut également servir de poste opérateur pour assurer la conduite de l'unité. Un autre logiciel est alors nécessaire pour assurer le dialogue avec l'automate et une interface opérateur conviviale. Si la liaison entre le PC et la CPU est rompue, l'API continue de dérouler son programme.

- ***Interfaces :*** elles assurent le lien avec le procédé. Ces interfaces peuvent alimenter les boucles d'entrées ou de sorties, dans ce cas, l'automate sera dotée d'une alimentation 24V continue. Elles peuvent être garnies de contacts secs, dans ce cas, une alimentation extérieure devra être intégrée aux boucles d'entrée et de sortie.

1.5.2 Aspect extérieur :

Les automates peuvent être de type compact ou modulaire.

- De type compact, on distinguera les modules de programmation (LOGO de Siemens, ZELIO de Schneider, MILLENIUM de Crouzet ...) des microautomates.

Il intègre le processeur, l'alimentation, les entrées et les sorties. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage rapide, E/S analogiques ...) et recevoir des extensions en nombre limité.

Ces automates, de fonctionnement simple, sont généralement destinés à la commande de petits automatismes.

- De type modulaire, le processeur, l'alimentation et les interfaces d'entrées / sortie résident dans des unités séparées (modules) et sont fixées sur un ou plusieurs racks contenant le "fond de panier" (bus plus connecteurs).

Ces automates sont intégrés dans les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires.

1.5.3 Structure interne d'un automate programmable industriel :

La structure interne d'un API peut se représenter comme suit :

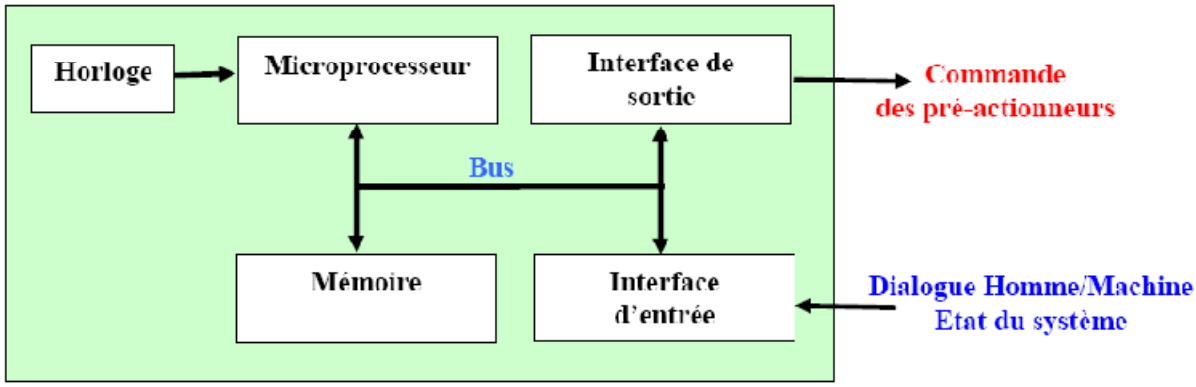


Figure 1.6 : Structure interne d'un automate

L'automate programmable reçoit les informations relatives à l'état du système et puis commande les pré-actionneurs suivant le programme inscrit dans sa mémoire.

Un API se compose donc de trois grandes parties :

• Le processeur ;

• La zone mémoire ;

• Les interfaces Entrées/Sorties

(1) Le microprocesseur :

Le microprocesseur réalise toutes les fonctions logiques ET, OU, les fonctions de temporisation, décomptage, de calcul... à partir d'un programme contenu dans sa mémoire.

Il est connecté aux autres éléments (mémoire et interface E/S) par des liaisons parallèles appelées ' BUS 'qui véhiculent les informations sous forme binaire.

(2) La zone mémoire :

a)- La Zone mémoire va permettre :

- De recevoir les informations issues des capteurs d'entrées
- De recevoir les informations générées par le processeur et destinées à la commande des sorties (valeur des compteurs, des temporisations, …)
- De recevoir et conserver le programme du processus

b)-Action possible sur une mémoire :

- ECRIRE pour modifier le contenu d'un programme
- EFFACER pour faire disparaître les informations qui ne sont plus nécessaires

- LIRE pour en lire le contenu d'un programme sans le modifier

c)- Technologie des mémoires :

- RAM (Random Acces Memory): mémoire vive dans laquelle on peut lire, écrire et effacer (Contient le programme)
- ROM (Read Only Memory): mémoire morte accessible uniquement en lecture.
- EPROM mémoires mortes reprogrammables effacement aux rayons ultra-violets.
- EEPROM mémoires mortes reprogrammables effacement électrique

Remarque :

La capacité mémoire se donne en mots de 8 BITS (Binary Digits) ou octets.

(3) Les interfaces d'entrées/sorties :

Les entrées reçoivent des informations en provenance des éléments de détection (capteurs).

Les sorties transmettent des informations aux pré-actionneurs (relais, électrovannes …) et les éléments des signalisations (voyants).

a)- Interfaces d'entrées :

Elles sont destinées à :

- Recevoir l'information en provenance des capteurs
- Traiter le signal en le mettant en forme, en éliminant les parasites et en isolant électriquement l'unité de commande de la partie opérative.

b)- Interfaces de sorties :

Elles sont destinées à :

- Commander les pré-actionneurs et éléments des signalisations du système
- Adapter les niveaux de tensions de l'unité de commande à celle de la partie opérative du système en garantissant une isolation galvanique entre ces dernières

(4) Alimentation de l'automate programmable industriel :

L'alimentation intégrée dans l'API, fournit à partir des tensions usuelles des réseaux (230 V, 24 V=) les tensions continues nécessaires au fonctionnement des circuits électroniques.

1.6 Fonctionnement général d'un automate programmable industriel :

À partir des informations que lui fournissent les capteurs et, suivant un algorithme déterminé par programmation, élabore les commandes transmises aux actionneurs.

Assure la communication avec l'opérateur (interface avec l'usager) et les autres processeurs qui gèrent la production ou qui interviennent dans le même procédé.

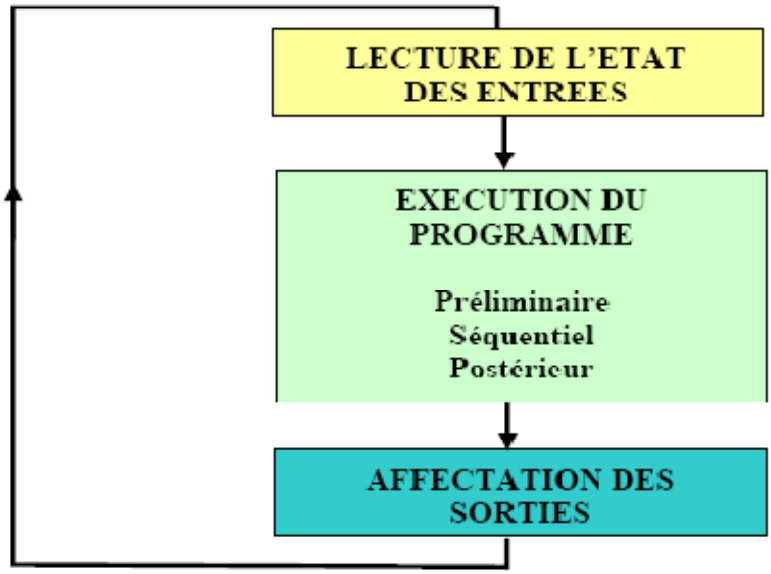


Figure 1.7 : Fonctionnement général d'un automate programmable industriel

Dans un API cyclique, le programme s'exécute dans une boucle permanente. Dans chaque itération de cette boucle ou cycle, trois types d'actions (l'acquisition des entrées, l'exécution du programme et l'affectation des sorties) sont effectuées. L'ordre et la répartition de ces actions dans le cycle conditionnent sa structure. Nous nous intéresserons dans la suite aux API dont le cycle est composé des trois phases suivantes :

Phase d'acquisition des entrées

Durant cette phase, les signaux appliqués à l'interface d'entrée de l'API sont copiés en mémoire dans des emplacements accessibles au programme et qui correspondent aux variables d'entrée. Les variables d'entrée sont uniquement accessibles en lecture. Leurs valeurs resteront ainsi inchangées lors des deux phases suivantes du cycle. En d'autres termes, au moment de l'acquisition des entrées, l'API "prend une photo" de l'environnement physique.

Phase d'exécution du programme

La phase d'exécution du programme permet de calculer les nouvelles valeurs des variables de sortie.

Phase d'affectation des sorties

Les variables de sortie sont affectées à l'interface de sortie pour pouvoir être appliquées aux préactionneurs.

Notons enfin qu'il existe une classe particulière d'API : les API périodiques.

Contrairement aux API cycliques où les cycles sont enchaînés sans attente (si le cycle courant se termine, le cycle suivant est immédiatement commencé), dans les API périodiques, les cycles commencent à des intervalles réguliers, fixés par l'utilisateur et dont la durée doit être supérieure à la somme des durées d'exécution du plus lent chemin du programme et de la phase d'entrée-sortie. Dans ces API, les entrées et les sorties sont toujours lues et écrites périodiquement (deux lectures ou écritures successives sont séparées par la période fixée par l'utilisateur)

CHAPITRE2 : LA DESCRIPTION DES LOGICIELS D'UN API

2.1 Les outils pour le contrôle et la commande d'un API

2.1.1 Le responsable de transfert des programmes ou les modules de communication

(1) Le micro-ordinateur

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.

Automatiquement en transférant le programme à l'aide du logiciel d'assistance, et en réalisant la liaison série entre l'ordinateur et l'automate, on peut transférer le programme vers l'automate.

(2) la console de programmation

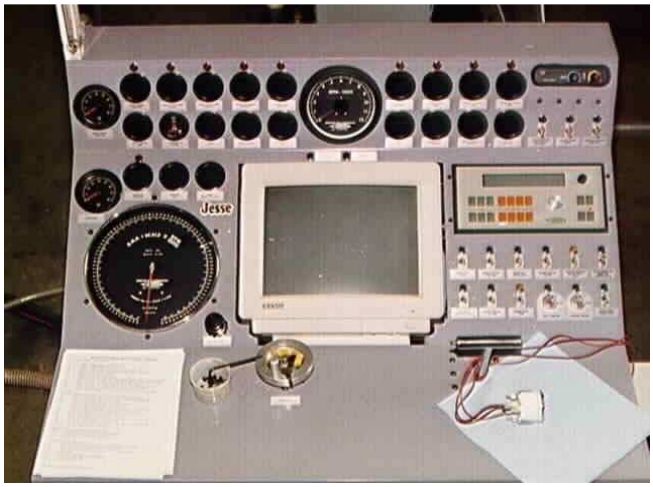


Figure 2.1 : Le consol de programmation

Exemple :

Contrôle à 100 % sur les automates programmables dans les chariots élévateurs:

(TVH ELARGIT SA GAMME DE CONSOLES DE PROGRAMMATION)

Toujours plus de constructeurs de chariots élévateurs utilisent des automates programmables. Cela signifie que le réglage de nombreux paramètres des chariots élévateurs se fait par le biais d'une console de programmation. Cette même console de programmation est également

utilisée pour effectuer un diagnostic d’erreur, si une fausse produit dans la machine. Il s’agit donc d’un appareil indispensable. C’est pourquoi **TVH** dispose d’un vaste choix de consoles.

Un automate est un réglage électronique qui détermine à l’aide de différentes entrées la vitesse de rotation d’un ou de plusieurs moteurs. Un chariot élévateur à fourche électrique est muni d’une commande de traction pour le mouvement de translation, d’une commande à pompe pour l’élévation et d’une commande assistée pour la conduite du véhicule. Chaque automate contient généralement une ou plusieurs cartes imprimées qui transforment tous les signaux d’entrée en un signal de sortie correct.

Pour programmer les automates, il est conseillé d’utiliser une console de programmation adéquate. C’est pourquoi **TVH** propose une série de consoles conçues pour être utilisées en combinaison avec des chariots élévateu**r**s spécifiques. On remarque ainsi le programmateur **Zapi standard**, les programmateurs **Curtis standard** et **Curtis spécial** pour BT-trucks, la console pour les automates **Bosch/Iskra LT200**, une console pour les automates **GR**, le programmateur **Sevcon Mos 90** et le **Sevcon** pour les automates **SC2000** et **Powerpak.**

Cette gamme de consoles est non seulement proposée au client, mais **TVH** même l’utilise pour effectuer des réparations et des essais.

(3) Les borniers de tests

Destinés aux personnels d'entretien, ils permettent de visualiser le programme ou les valeurs des paramètres.

Par exemple :

- Affichage de la ligne de programme ontr?
- Visualisation de l'instruction (code opératoire et adresse de l'opérande)
- Visualisation de l'état des entrées
- Visualisation de l'état des sorties.

Figure 2.2 : Bornier de test

(4) Les unités de dialogue en ligne

Elles sont destinées aux personnels spécialistes de la procédure et non de l'automate programmable, et leur permet d'agir sur certains paramètres :

- Modification des constantes, compteurs temporisations
- Forme des entrées/sorties
- Exportation de parties de programme
- Chargement de programmes en mémoire à partir de cassettes.

Ces borniers se présentent sous la forme enfichable dans l'unité centrale. Il comporte des touches de fonctions, numériques, une visualisation, un dispositif de sécurité, l'ensemble est piloté par un micro-processeur.

2.1.2 Interface de communication

RS-232: communication série (l'information est communiquée, un bit à la fois, sur un seul fil) entre deux dispositifs.

Réseau: lien de communication partagé par plusieurs dispositifs. Chacun d'eux est identifié par une adresse unique. La communication se fait suivant des protocoles déterminés (DeviceNet, Profibus, Ethernet, …)

Le bus interne : il permet la communication de l'ensemble des blocs de l'automate et des éventuelles extensions.

2.1.3 L'exécuteur et les supports des programmes de l'API

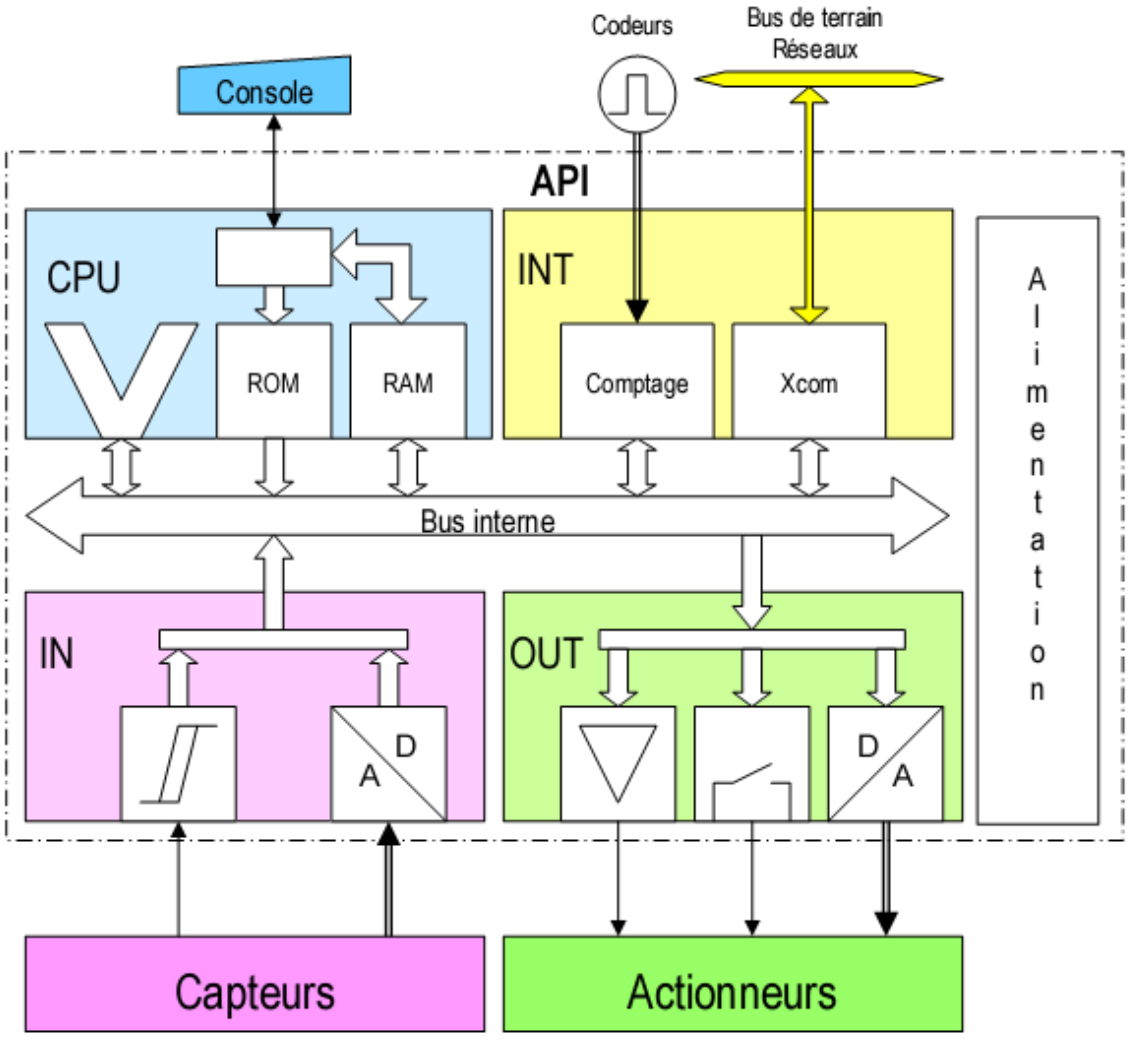


Figure 2.3 : Architecture interne d'un automate programmable

Un automate programmable est constitué de plusieurs éléments. Quelque soit la taille et la puissance de calcul de la machine l'architecture est similaire :

- L'élément central est l'unité de traitement arithmétique et logique (CPU) qui effectue les séquences de programme et les calculs.
- Les programmes sont enregistrés dans une mémoire qui garde l'information même quand l'alimentation électrique est coupée. Une autre mémoire est dédiée au stockage des données ; cette partie de la mémoire peut être ou non volatile, c'est-à-dire qu'elle s'efface quand la tension d'alimentation est coupée.
- Les entrées-sorties sont les liens entre l'automate et son environnement. Leur type dépend des caractéristiques du signal qu'elles doivent capter ou générer : tout ou rien (digitales) pour les signaux binaires, analogiques pour les signaux de mesure ou de consigne.
- Une alimentation pour les circuits électroniques internes. Celle-ci est galvaniquement isolée des circuits de commande.

- Des interfaces de communication (Xcom) qui servent à l'échange d'informations numériques avec le monde extérieur par bus de terrain ou réseaux informatiques. Une de ces interfaces est utilisée pour charge le programme dans la mémoire de l'automate.

3.1.3.1 Le CPU ou le microprocesseur :

Le **microprocesseur** réalise toutes les fonctions logiques ET, OU, les fonctions de temporisation, de comptage, de calcul, ... à partir d'un programme contenu dans sa **mémoire**. Le microprocesseur effectue les opérations logiques et arithmétiques suivant une séquence dictée par les instructions stockées en mémoire, il gère le trafic des données sur le bus interne mais aussi le responsable de la procédure d'auto-diagnostique (mesure pour s'assurer du bon fonctionnement de toutes les composantes de l'automate).

Un des critères de performance d'un processeur est son temps de cycle (le temps pris par l'automate pour interroger les ports d'entrée, exécuter le programme, et mettre à jour les ports de sortie). Ce temps varie de 0.1 à 50 ms/ko.

Il est connecté aux autres éléments (mémoire et interface E/S) par des liaisons **parallèles** appelées '**BUS** ' qui véhiculent les informations sous forme binaire ...

Son rôle consiste d'une part à organiser les différentes relations entre la zone mémoire et les interfaces d'entrées et de sorties et d'autre part à gérer les instructions du programme.

3.1.3.2 La zone mémoires :

La mémoire est issue des différents secteurs du système qui sont :

- le terminal de programmation (introduction du programme);
- le processeur qui gère et exécute le programme.

Elle reçoit également des informations en provenance des capteurs.

- La Zone mémoire va permettre :

- De recevoir les informations issues des capteurs d'entrées
- De recevoir les informations générées par le processeur et destinées à la commande des sorties (valeur des compteurs, des temporisations, …)
- De recevoir et conserver le programme du processus

-Action possible sur une mémoire :

ECRIRE pour modifier le contenu d'un programme

EFFACER pour faire disparaître les informations qui ne sont plus nécessaires

LIRE pour en lire le contenu d'un programme sans le modifier

Il existe dans les automates plusieurs types de mémoires qui remplissent des fonctions différentes :

- Technologie des mémoires :

Conception et élaboration du programme

RAM (Random Acces Memory): mémoire vive dans laquelle on peut lire, écrire et effacer (contient le programme et elle s'efface automatiquement à l'arrêt de l'automate, nécessite une batterie de sauvegarde)

ROM (Read Only Memory): mémoire morte accessible uniquement en lecture.

EPROM mémoires mortes reprogrammables : effacement aux rayons ultra-violets, conservation du programme pendant l'exécution de celle-ci.

EEPROM mémoires mortes reprogrammables : effacement électrique, seulement la lecture est possible.

Remarque :

La capacité mémoire se donne en mots de 8 BITS (Binary Digits) ou octets.

Exemple:

Soit une mémoire de 8 Koctets = 8 x 1024 x 8 = 65 536 BITS. Cette mémoire peut contenir

65 536 informations binaires.

3.1.3.3 Les interfaces :

L'interface d'entrée comporte des adresses d'entrée. Chaque capteur est relié à une de ces adresses. L'interface de sortie comporte de la même façon des adresses de sortie. Chaque préactionneur est relié à une de ces adresses. Le nombre de ces entrées est sorties varie suivant le type d'automate.

3.1.3.4 Les cartes d'E/S :

Les modules d'entrées traduisent les signaux industriels (tension, courant, résistance, pulsation, …) en information logique ou numérique interprétable par le processeur.

Inversement, les modules de sortie traduisent les commandes du processeur en des signaux industriels.

Les cartes d'entrées / sorties sont modulaires, ces modules comportent 1, 4, 8, 16 ou 32 voies (ports) d'entrée et/ou de sortie.

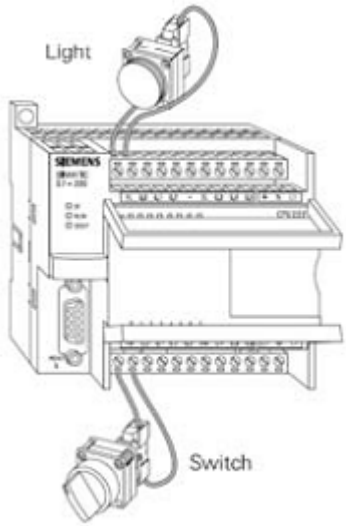


Figure 2.4 : cartes d'E/S

3.1.3.5 Une alimentation électrique :

Tous les automates actuels sont équipés d'une alimentation 240 V 50/60 Hz, 24 V DC. Les entrées sont en 24 V DC et une mise à la terre doit également être prévue.

2.2 La description des logiciels

2.2.1 La norme IEC 61131-3

(1) Normalisation IEC 61131-3

La diversité des applications et l'évolution du matériel ont amené les constructeurs d'automatismes à concevoir des automates de plus en plus complexes.

Un API est programmé à l'aide de langages spécialisés, fournis par son constructeur et utilisables au travers d'une interface (un logiciel sur PC, un pupitre...).

Pour maîtriser la complexité et rendre la programmation des automates plus efficace, des standards industriels ont été adoptés par les automaticiens. Ces standards définissent non seulement les langages mais également la méthodologie de programmation.

A présent ils sont disponibles pour presque toutes les plates-formes du marché et sont normalisés par la Commission Électrotechnique Internationale (CEI – en anglais : International Electrotechnic Commission - IEC), qui réuni fournisseurs, utilisateurs et chercheurs.

Ces langages peuvent être classés en 5 grandes familles. Cependant, deux langages de la même famille et fournis par deux constructeurs différents ne sont pas forcément compatibles, ce qui est de nature à nuire à la portabilité des applications et à limiter la réutilisation du code. C'est pour cette raison que la commission électrotechnique internationale **CEI** a entrepris un grand effort de normalisation visant à uniformiser les langages utilisés dans le domaine de la programmation des API.

En particulier, la norme **CEI-61131-3 [IEC93]** définit la programmation des automates, et propose un cadre qui s'étend de la spécification aux architectures.

(2) Critères de qualité

Pour fonctionner en temps réel de manière sûre et stable dans un environnement industriel, les automates programmables doivent respecter des contraintes très sévères :

Temporelle : Le système doit répondre à une stimulation externe dans un laps de temps connu.

Sûreté : Les automates programmables sont parfois utilisés dans des applications où leur défaillance peut entraîner des blessures ou des dégâts matériels coûteux.

Concurrence : Les processeurs des automates doivent être à même de traiter plusieurs tâches simultanément.

Déterminisme : L'état des sorties est entièrement déterminé par l'histoire des entrées

2.2.2 les principes de réalisation câblée et programmée

(1) La logique câblée :

L'automatisme est obtenu en reliant entre eux les différents constituants de base ou fonctions logiques par câblage.

La logique câblée correspond à un traitement parallèle de l'information. Plusieurs constituants peuvent être sollicités simultanément.

Exemple : X = (a.b) +c

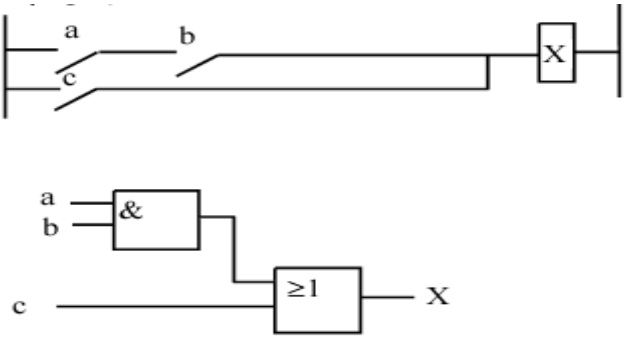


Figure 2.5 : Schéma câblé de X= (ab)+c

Modification : X= (a+b).c

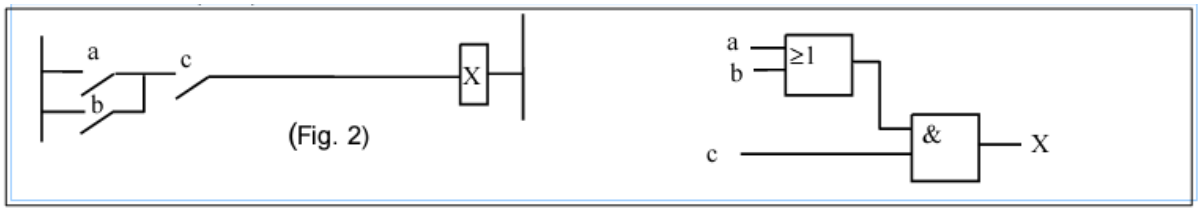


Figure 2.6 : Schéma câblé de X= (a+b).c

(2) La logique programmée :

Elle correspond à une démarche séquentielle, seule une opération élémentaire est exécuté à la fois, c'est un traitement série.

Le schéma électrique est transcrit en une suite d'instruction qui constitue le programme.

En cas de modification des équations avec les mêmes accessoires, l'installation ne comporte aucune modification de câblage seul le jeu d'instructions est modifié.

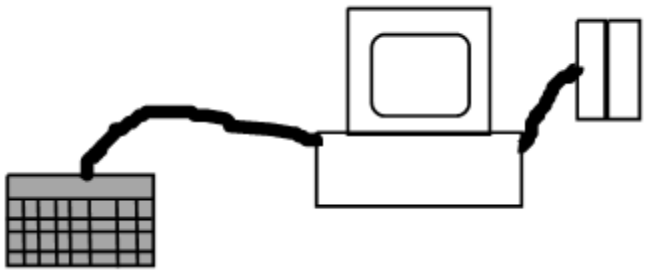


Figure 2.7 : Logique programmée

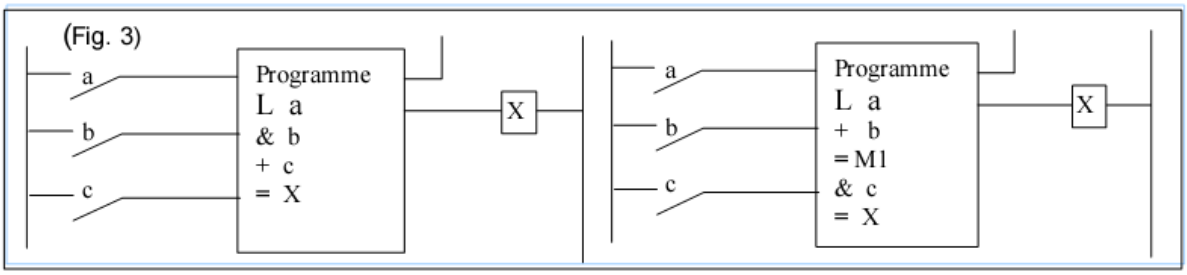


Figure 2.8 : Logique câblée

Ici il suffit de modifier le jeu d'instruction pour obtenir un fonctionnement différent.

2.2.3 le fonctionnement du programme de l'automate

(1) Organisation des programmes

La norme CEI-61131-3 définit l'organisation générale des programmes et les éléments communs qui sont utilisés par tous les langages de programmation.

Comme tous systèmes informatiques les automates programmables utilisent des **variables** qui représentent sous forme numérique l'information qui doit être traitée. La forme de ces données est définie par le **type** des variables : booléen, entier, entier long, réel, date, etc. Pour faciliter la lisibilité des programmes des variables sont nommés par des **symboles**.

Les **ressources** sont l'ensemble des variables auxquelles peuvent accéder les programmes. Ces variables peuvent être liées à la structure matérielle de l'automate (entrées, sorties, horloges, interruptions) ou internes (indicateurs, registres, temporisateurs, compteurs, blocs de données).

Les programmes doivent être structurés pour en faciliter le développement, la portabilité et la maintenance.

Les Unités d'Organisation de Programme ou blocs d'organisation (en anglais : program organisation unit – POU) sont des conteneurs qui contiennent une partie des instructions du programme. Ces blocs peuvent être :

- Des **fonctions standards** (sin(x), sqrt(x), exp(x), etc.) ou définies par le programmeur, elles n'ont pas d'état interne et leur invocation avec les mêmes arguments donne toujours le même résultat.

- Des **blocs fonctionnels** (FB : function blocs) qui contiennent dans la même entité un programme et des données. Les blocs fonctionnels possèdent des arguments et ont un état interne, le résultat produit par leur invocation dépend donc de l'historique.
- Des **blocs programme** (PB : program blocs) regroupent un certain nombre d'instructions ils ne permettent pas le passage d'arguments.
- Des **blocs de données** (DB : data blocs) rassemblent un certain nombre de variables utilisables comme arguments pour l'échange d'information entre les programmes.
- Des **blocs séquentiels** (SB : sequential blocs ou SFC : sequential flow chart) sont utilisés pour l'exécution des programmes écrits en GRAFCET, ces blocs regroupent un certain nombre de bloc d'action et de transition.

Ces différents blocs peuvent être imbriqués, la structure de l'organisation varie en fonction du type de processeur et du fabricant.

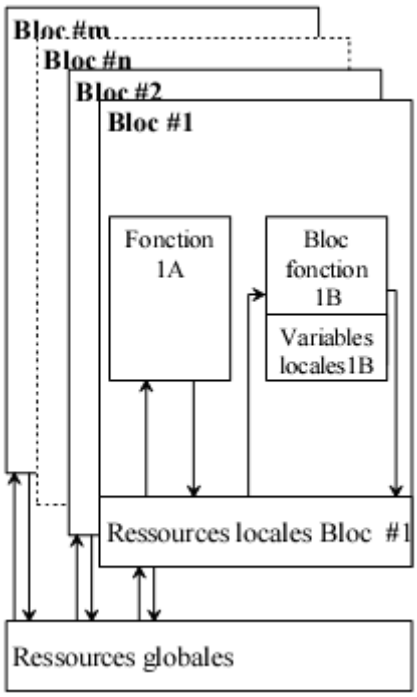


Figure 2.9 : Blocs d'organisation d'un programme d'automatisme

(2) Exécution des programmes

Les programmes des automates s'exécutent en temps réel, c'est-à-dire qu'un ensemble d'instruction doit être traité en un temps donné. Au cours d'un cycle de traitement le processeur effectue un certain nombre de tâches qui s'enchaînent dans un ordre préétabli :

Tâche n° 1 : Traitement interne ou système : fonctions non liées à l'application telles que surveillance du matériel, communication avec des périphériques ou échange de données avec d'autres processeurs.

Tâche n° 2 : Lecture des entrées : toutes les entrées sont lues au même instant. Les cartes d'entrée enregistrent l'état des signaux sur un ordre provenant du processeur. L'image de ces états est ensuite copiée dans des variables qui seront traitées par le programme.

Tâche n° 3 : Exécution des blocs de programme : les différents blocs du programme d'application sont successivement traités.

Tâche n° 4 : Écriture des sorties : les valeurs des variables représentant les sorties sont toutes copiées dans les sorties physiques au même instant sur un signal du processeur.

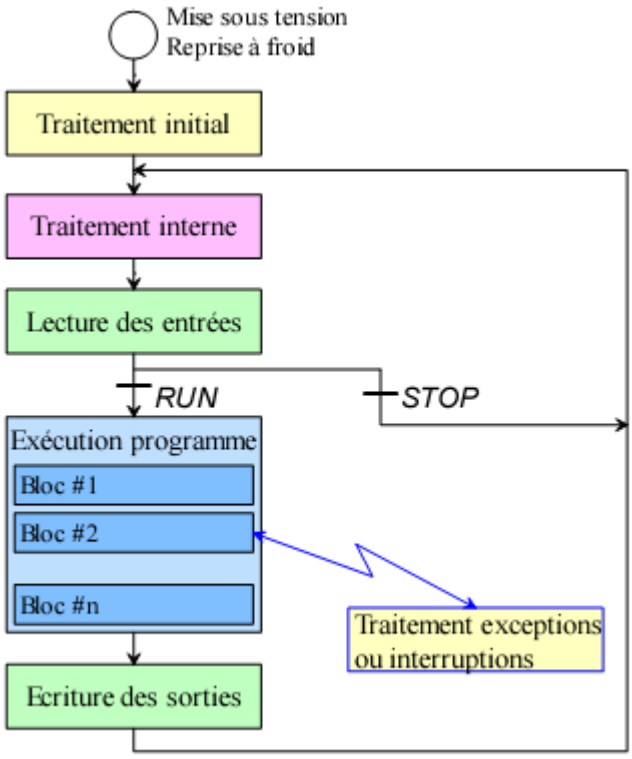


Figure 2.10 : Blocs d'organisation d'un programme d'automatisme

Ces quatre tâchent composent un cycle. Quand l'automate est actif (en RUN) les cycles se succèdent indéfiniment. Une temporisation indépendante du processeur, le chien de garde ou watch-dog, surveille le temps d'exécution de chaque cycle. Si ce temps est dépassé une alarme est signalée.

Quand l'automate est arrêté (en STOP) seules les tâches de traitement système et de lecture des entrées sont effectuées. Le programme n'est plus traité et les sorties ne sont plus mises à jour.

A la mise sous tension de l'automate, appelée également reprise à froid, un bloc de programme particulier, le traitement initial, est exécuté une seule fois. Il permet d'initialiser des variables et de faire les contrôles préliminaires à l'exécution du programme d'application.

Certains blocs de programme s'exécutent si des événements particuliers externes se produisent : activation d'une entrée ou limite de comptage atteinte. Ces événements génèrent une interruption qui déclenche immédiatement l'exécution du bloc de programme. Des événements internes (erreur de calcul, débordement d'un compteur, dépassement du temps de cycle), appelés exceptions, peuvent également déclencher l'exécution de blocs de programme pour la signalisation ou la correction de l'erreur.

Certains automates proposent deux modes de fonctionnement possibles, le choix dépendra de l'application :

- En mode **cyclique** les cycles s'enchaînent les uns après les autres, sans temps d'attente, quelque soit la durée d'exécution du programme. Avantage le temps de réponse est plus court mais la durée de traitement t est différente à chaque cycle. L'**intervalle entre chaque cycle est indéterminé**.

Figure 2.11 : Exécution cyclique du programme

- En mode périodique les cycles sont déclenchés périodiquement par un top d'horloge. Les cycles s'effectuent à intervalles réguliers t ce qui laisse des espaces temporels libres pendant lesquels le processeur ne fait rien, le temps de réponse n'est donc pas

minimum. Ce mode de fonctionnement sera utilisé pour des applications de régulation numérique, car celles-ci requièrent un temps de cycle constant.

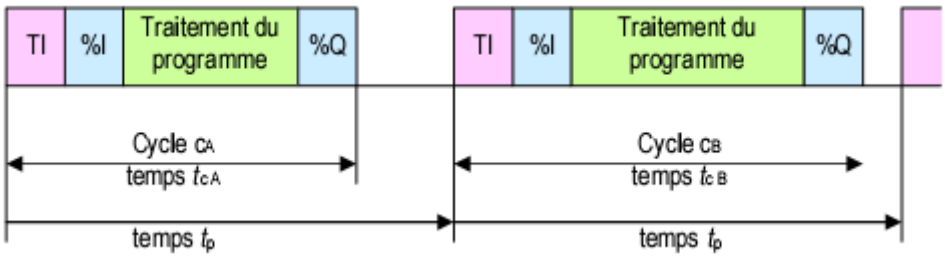


Figure 2.12 : Exécution périodique du programme

2.2.4 Les langages de description et la programmation d'un API

(1) Choix des langages de programmation

Les langages de programmation des automates programmables sont complètement définis par la norme CEI-61131-3 qui distingue trois groupes :

Les **langages textuels** : liste d'instruction (IL) et texte structuré (ST),

Les **langages graphiques** : plans de contacts (LD) et diagramme de blocs fonctionnels (FDB),

Le **langage séquentiel** : diagramme séquentiel (SFC), variante du GRAFCET.

Il est toujours possible d'utiliser plusieurs langages au sein d'un même projet, fonctionnant sur un seul automate programmable. Le choix peut ainsi être guidé par la qualification des programmeurs qui interviennent aux différents stades du projet. Ainsi par exemple :

- Les fonctions sophistiquées seront programmées en texte structuré (ST) par les spécialistes du Ra&D, aboutissant à des sous-programmes ou à des blocs fonctionnels.
- L'automatisation de l'installation ou de la machine à programmer sera programmée en diagramme séquentiel (SFC) et en blocs fonctionnels (SBD) par les constructeurs.
- Les commandes tout-ou-rien des moteurs et autres actionneurs simples seront programmées en plan de contacts (LD), facilitant ainsi la mise en service et la maintenance par du personnel moins qualifié.

(2) Les variables

La syntaxe des symboles des variables accessibles directement est définie. Pour les nommer on associe la notation '%', l'emplacement physique (ressource), la taille et l'adresse absolue. Le tableau ci-dessous donne les indications pour les principaux symboles :

Ressource, 1er préfixe	
%I	entrée
%Q	sortie
%M	mémoire
(%K)	constante

Taille, 2ème préfixe		
X	booléen (*bit*)	1 bit
B	octet (*byte*)	8 bits
W	mot (*word*)	16 bits
D	double mot (*double*)	32 bits
L	mot long (*long*)	64 bits

Tableau 2.1 : Notation normalisée des variables dans les automates programmables

Pour alléger la notation le deuxième préfixe (la taille) est parfois omis quand le symbole représente un bit (X).

EXEMPLES

%MW357 mot 357 de la mémoire interne

%MD81 double mot à l'adresse 81 de la mémoire

%I35 entrée physique 35

%Q13 sortie physique 13

%QB2 octet de sortie 2

(3) Liste d'instructions IL (Instruction List)

Ce langage, proche de l'assembleur des microprocesseurs, est utilisé dans des cas particuliers pour optimiser certaines parties du code si les contraintes temporelles sont importantes. Un programme IL est une suite de ligne d'instructions qui comprend :

Une **étiquette** : (facultative) utilisée pour mettre des points de repère dans le programme,

Un **opérateur** : représenté par un code mnémonique qui indique quelle opération doit effectuer le processeur,

Un **argument** : (éventuel) qui dépend de l'opérateur, indique sur quel objet porte l'opération,

Un **commentaire** : optionnel mais vivement recommandé.

Ces éléments sont mis en colonne pour faciliter la lecture du code.

EXEMPLE

Programme en liste d'instruction IL pour la commande du vérin :

```
;=================================================================
======
; HEIG-VD - TCSiM cours d'électricité automates programmable
; Exemple de programme en liste d'instructions (IL) : commande de verin
; ----------------------------------------------------------------
; V1.0  ABe  30.03.05    Première version
;=================================================================
======

Verin_IL:                       ; Etiquette début de programme
        LD  In_M                ; Lecture entrée In_M
        AND  In_A               ; AND logique avec entrée In_A
        OR  Out_Q               ; OR logique avec sortie Out_Q
        ST  M_temp              ; stocke résultat dans bit temporaire
        LD  In_B                ; Lecture entrée In_B
        NOT                     ; Inverse la valeur
        AND  M_temp             ; AND logique avec variable temporaire
        ST  Out_Q               ; Ecrit résultat dans sortie Out_Q
        END
```

(4) Texte structuré ST (Structured Text)

Le langage texte structuré est dérivé des langages de programmation de haut niveau tels que Pascal, ADA, C, BASIC. Il est utilisé pour programmer des algorithmes complexes qui nécessitent des calculs numériques.

Le jeu d'instructions de ce langage comprend des fonctions, des instructions conditionnelles (IF … THEN … ELSIF … ELSE … END_IF), des instructions de boucle (FOR … DO … END_FOR) et d'autres instructions de contrôle.

EXEMPLE

Programmes en texte structuré ST pour la commande du vérin

```
(*
=================================================================
*)
(* HEIG-VD - TCSiM cours d'électricité automates programmable          *)
(* Exemple de programme en texte structuré (ST) : commande de verin    *)
(* ------------------------------------------------------------------ *)
(* V1.0      ABe 30.03.05              Première version            *)
(*
=================================================================
*)


Verin_ST1: (* Etiquette début de programme *)
Out_Q := ((In_M AND In_A) OR Out_Q) AND NOT(In_B)); (* Equation booléenne *)
RETURN;



(*
=================================================================
*)
(* HEIG-VD - TCSiM cours d'électricité automates programmable          *)
(* Exemple de programme en texte structuré (ST) : commande de verin    *)
(* ------------------------------------------------------------------ *)
(* V2.0      ABe 30.03.05              Variante du programme       *)
(*
=================================================================
*)


Verin_ST2: (* Etiquette début de programme *)
IF ((In_M AND In_A AND NOT(In_B)) THEN (* Equation booléenne *)
   SET Out_Q;
```

```
ELSE
   RESET Out_Q;
END_IF;
RETURN;
```

(5) Plan de contacts LD (Ladder Diagram)

Les plans de contacts sont dérivés des premières réalisations d'automatismes faites avec des relais électromécaniques. C'est le plus ancien des langages normalisés (1969), il a été conçu pour faciliter la transition entre les systèmes à logique câblée et les systèmes à logique programmée. Il permet la représentation graphique sous forme de schémas des équations logiques booléennes. Ce langage est conservé pour maintenir une compatibilité avec des automates d'ancienne génération mais il est de moins en moins utilisé.

Les variables d'entrée des équations sont symbolisées par des 'contacts' –| |– et les variables de sorties par des 'bobines' –( )–. La réalisation des fonctions logiques se fait en reliant ces éléments par des lignes qui symbolisent l'équivalent d'un schéma électrique. L'automate évalue ces réseaux de haut en bas et de gauche à droite.

Les fonctions de base de ce langage sont données dans le tableau ci-dessous. Il convient de remarquer que les symboles diffèrent des contacts définis par la CEI. La raison est que ce langage de programmation a été conçu aux USA, et qu'il a repris les symboles des contacts en usage dans ce pays à l'époque.

Symbole	Désignation	Fonction
–| |–IN_A	Contact direct	Variable d'entrée *IN_A*
–| / |–IN_B	Contact inverse	Inverse de la variable d'entrée IN_B
–| P |–IN_C	Front positif	Vaut '1' pendant le cycle de traitement où la variable *IN_C* à passé de l'état '0' à l'état '1'
–| N |–IN_D	Front négatif	Vaut '1' pendant le cycle de traitement où la variable *IN_D* à passé de l'état '1' à l'état '0'
–( )–OUT_A	Relais direct	Variable de sortie *OUT_A*
–(/)–OUT_B	Relais inverse	Inverse de la variable de sortie *OUT_B*
–(P)–OUT_C	Relais front positif	Variable de sortie *OUT_C* vaut '1' durant le cycle de l'automate où le signal de commande du relais à passé de l'état '0' à l'état '1'
–(N)–OUT_D	Relais front négatif	Variable de sortie *OUT_D* vaut '1' durant le cycle de l'automate où le signal de commande du relais à passé de l'état '0' à l'état '1'
–(S)–OUT_E	Relais Set	Met la variable de sortie *OUT_E* à l'état '1' quand le signal de commande du relais est à l'état '1'. La variable de sortie n'est pas modifiée quand le signal est à l'état '0'.
–(R)–OUT_F	Relais Reset	Met la variable de sortie *OUT_E* à l'état '0' quand le signal de commande du relais est à l'état '1'. La variable de sortie n'est pas modifiée quand le signal est à l'état '0'.

Tableau 2.2 : Symboles graphiques des plans de contacts LD

EXEMPLE

Programme en réseaux de contacts LD pour la commande du vérin.

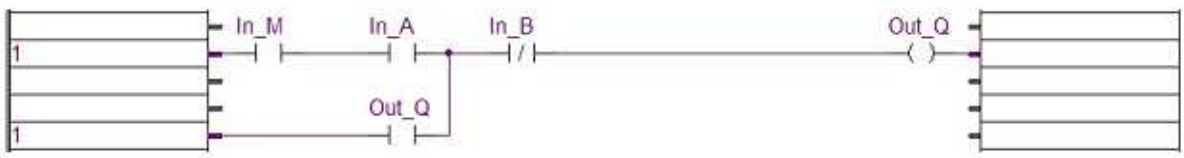


Figure 2.13 : Exemple d'un programme en plan de contacts

(6) Blocs fonctionnels FBD (Function Bloc Diagram)

Ce langage est une évolution des réseaux de contacts. Il permet de traiter non seulement des équations logiques binaires mais également des fonctions beaucoup plus complexes, par exemple des régulateurs PID, faisant intervenir des variables numériques. Le langage FBD est normalement utilisé pour des applications qui traitent un flux continu d'information.

La programmation se fait en interconnectant de manière judicieuse des composants, appelés blocs fonctionnels, pour réaliser une fonction spécifique. Les **blocs fonctionnels** peuvent être :

- de simples fonctions logiques : portes NON, ET, OU
- des fonctions de logique séquentielle : bascules RS, flip-flop, temporisateurs
- des opérateurs arithmétiques ou mathématiques : adition, soustraction, multiplication, division, racine carrée, exponentielle, sinus, etc.
- des opérateurs fonctionnels complexes : multiplexeurs, démultiplexeurs, générateurs d'impulsions, régulateurs, etc.
- des opérateurs fonctionnels spécifiques préprogrammés : commande de moteur, asservissement d'axes, gestionnaires de communication, etc.

Les différents fabricants offrent un catalogue de fonctions de base et des modules dédiés aux différents métiers.

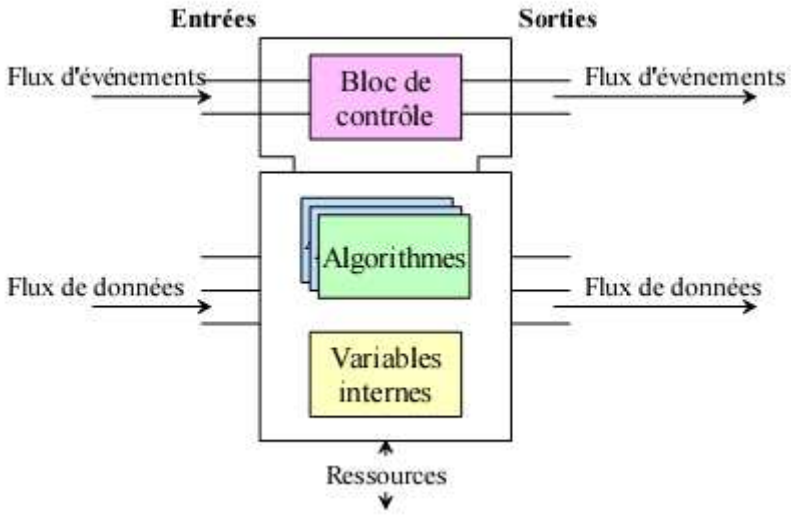


Figure 2.14 : Modèle général d'un bloc fonctionnel

Un bloc fonctionnel est représenté par un rectangle, les signaux d'entrée sont à gauche et les sorties à droite. Cette disposition correspond au sens conventionnel du flux des données dans une représentation schématique. Un bloc fonctionnel exécute un ou plusieurs algorithmes qui peuvent être des opérateurs simples ou des fonctions complexes. Il comporte au besoin des variables internes qui peuvent être des paramètres, c'est-à-dire des constantes fixées au moment de la programmation, ou des variables qui stockent des valeurs intermédiaires utilisées par les algorithmes. Certains blocs fonctionnels ont éventuellement un élément de contrôle qui permet de modifier leur comportement en fonction d'événements externes particuliers : activation ou désactivation du bloc par exemple. Les blocs peuvent également faire directement appel aux ressources de la machine pour des applications particulières : communication, interruptions, etc.

EXEMPLE

Programme en réseaux de contacts LD pour la commande du vérin.

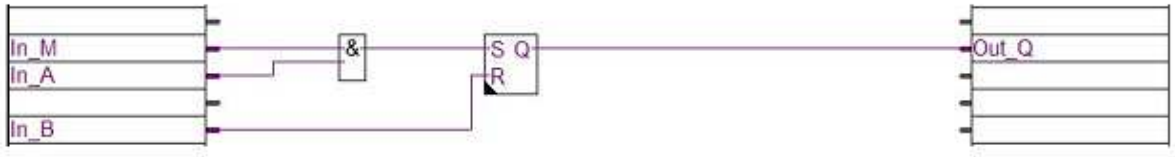


Figure 2.15 : Exemple d'un programme réalisé en blocs fonctionnels

(7) Diagramme fonctionnel en séquence SFC (Sequential Flow Chart)

Ce langage est directement dérivé du GRAFCET et répond aux mêmes règles de base. Il est utilisé pour programmer la commande de systèmes séquentiels. Un diagramme fonctionnel en séquence est constitué d'une succession d'étapes reliées par des transitions. Un programme d'application peut comporter plusieurs diagrammes fonctionnels en séquence indépendants ou non.

Une étape correspond à une situation dans laquelle le comportement du système est invariant par rapport à ses entrées et ses sorties. Une étape peut être active ou inactive. Quand l'étape est active le programme qui lui est associé s'exécute. Si elle est inactive cette partie du programme n'est pas traitée par le processeur. Le programme associé à une étape peut être écrit en langage IL, ST, LD ou FBD.

L'étape initiale est celle qui est activée à l'initialisation du programme. Il ne peut y avoir qu'une seule et unique étape initiale par diagramme.

Une transition indique la possibilité d'évolution entre les étapes. A chaque transition est associé une condition logique appelée réceptivité. Une transition est dite valide si toutes les étapes qui lui sont reliées en amont sont actives. Quand une transition est valide et que sa réceptivité est vraie, le diagramme peut évoluer d'une étape à l'autre ce qui ce fait généralement en un cycle d'automate. La réceptivité peut être écrite en langage IL, ST, LD ou FBD.

EXEMPLE

Diagramme fonctionnel en séquence SFC pour la commande du vérin

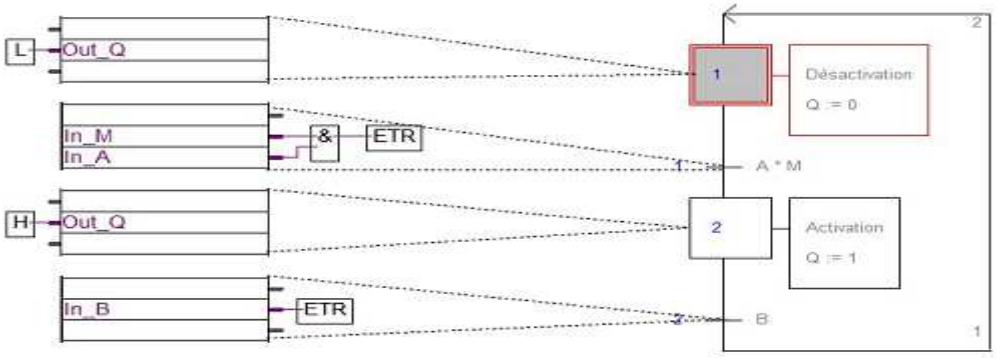


Figure 2.16 : Exemple d'un programme réalisé en diagramme fonctionnel à séquences

2.2.5 Les logiciels de programmation

(1) Mise en œuvre du logiciel PL7 Pro :

Après lancement du logiciel vous devez préciser :

- le processeur et la version de l'API à mettre en œuvre
- l'utilisation de l'outil Grafcet
- l'installation éventuelle d'une mémoire.

Toutes les configurations ci-dessous doivent être confirmées par appui sur l'icône : 

(a) Configuration du processeur :

Les déclarations suivantes peuvent être faites à partir du dossier configuration matérielle du navigateur application en double-cliquant sur le processeur de l'API (position 0) :

- L'**entrée %I1.8** peut être paramétrée pour commander le passage RUN/STOP de l'automate.
- La **sortie %Q2.0** peut être affectée à une fonction d'alarme.
- Démarrage automatique en RUN, cocher la case correspondante.
- Si l'API dispose d'une mémoire EPROM on peut cocher : Sauvegarde / Restitution des **mots internes %MWi**.
- Sélectionner le type d'exécution de la tâche MAST Cyclique ou Périodique (3 à 255 ms).
- Saisir la valeur du chien de garde de la tâche MAST: 10 à 500 ms maximum (250 ms, valeur par défaut). Saisir suivant l'application la valeur de la période de la tâche FAST (2 à 255 ms maximum)...

(b) Choix et configuration des modules :

- Double-cliquer sur la position du rack à configurer.
- Sélectionner dans le champ Famille le type de module (par exemple Analogique, sauf positions 1 et 2).
- Sélectionner dans le champ Module la référence du module à configurer (par exemple TSX AEZ 801).
- Valider par OK, le module est déclaré dans sa position (celle-ci est tramée et contient la référence du module).
- La configuration de chaque module peut-être modifiée par double-clic sur le module : type d'entrée positive (Sink) ou négative (Source), surveillance défaut d'alimentation, affection des entrées aux tâches, filtrage des entrées, fonction, repli des sorties etc.

(c) Adressage des objets de modules d'entrées sorties :

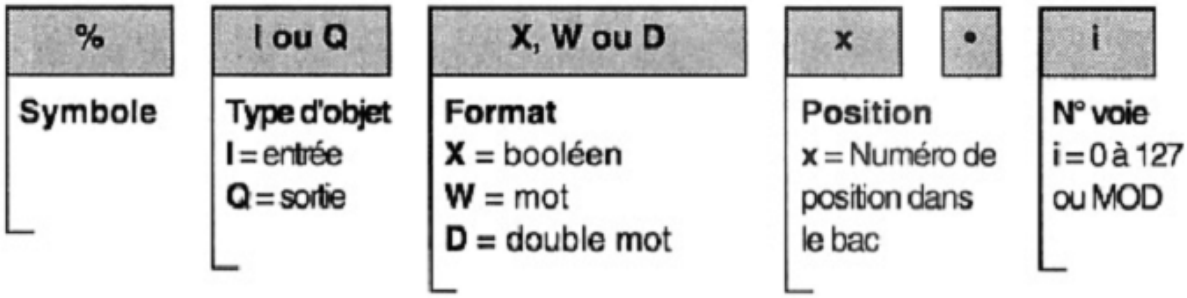


Tableau 2.3 : Adressage des objets

Exemples d'objets :

%I1.6 : Bit de la voie d'entrée n° 6 du module d'entrées TOR placé en position 1 dans le bac.

%IW3.5 : Mot de la voie d'entrée n° 5 du module d'entrées analogiques placé en position 3.

%Q2.8 : Bit de la voie de sortie n° 8 du module de sortie TOR placé en position 2.

%QW4.3 : Mot de la voie de sortie n° 3 du module de sortie analogique placé en position 4.

%Ix.MOD.ERR : Bit. Il indique lorsqu'il est à l'état 1 que le module situé en position x est en défaut.

%Ix.i.ERR : Bit, à l'état 1 il indique que la voie d'entrée i du module situé en position x est en défaut.

(d) Adressage des bits et mots:

Exemples :

%MB5 : Mot interne variable de 8 bits, numéro 5.

%KW12 : Mot constant de 16 bits, numéro 12.

%SW49 à %SW53 : Mots systèmes contenant la date et l'heure courante en BCD.

%M8 : Bit interne, numéro 8.

%S6 : Bit système, son état change toutes les secondes.

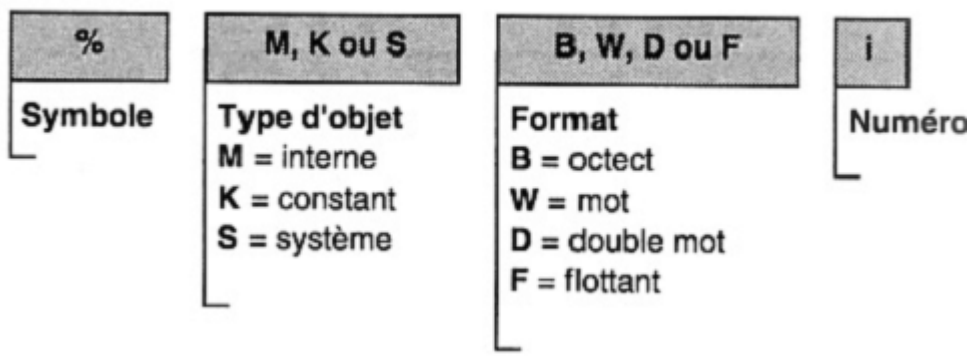


Tableau 2.4 : Adressage des vits et mots

Exemples :

%MW10:X4 : Bit n° 4 du mot interne %MW10 de 16 bits.

%QW5.1:X10 : Bit n° 10 du mot de la voie 1 du module de sortie analogique placé en position 5.

%SW60:X13 : Bit du mot système diagnostic automate. A l'état 1 indique le mode Run de l'automate.

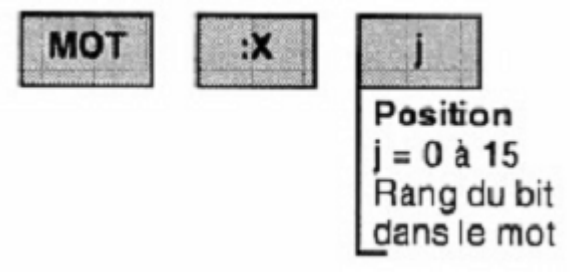


Tableau 2.5 : Bit du mot système

(e) Autres adressages :

%X12 : Bit indiquant l'état de l'étape 12 du GRAFCET. %X12.T : mot, temps d'activité de l'étape 12.

%TM2.V : Mot représentant la valeur courante de la temporisation du bloc fonction temporisateur n°2**.**

%TM3.Q : Bit de sortie indiquant une fin de temporisation du bloc fonction temporisateur n°3.

%C5.P : Mot représentant la valeur de présélection du bloc fonction compteur-décompteur n°5.

%C6.F : Bit de sortie indiquant un débordement du bloc fonction compteur-décompteur n° 6.

(f) Editeur de variables :

Dans l'éditeur de variables du navigateur application il est possible de :

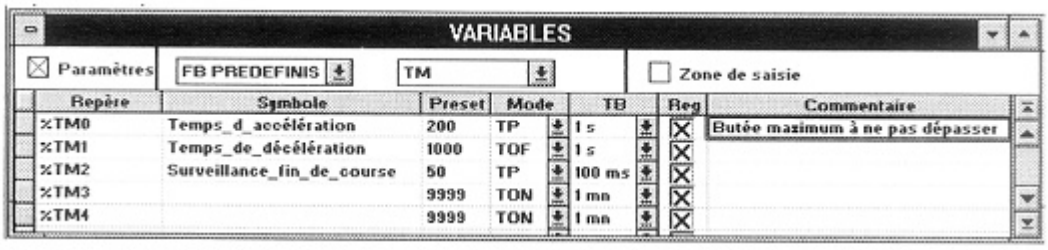
- Symboliser les différents objets de l'application (colonne symbole). Leur identification dans la programmation en sera facilitée. Repère **%Q2.4** symbole **Mot_vent**.
- Paramétrer les blocs fonctions prédéfinis temporisateurs, compteurs, etc.
- Saisir les valeurs des mots constants (colonne valeur) après avoir coché la case paramètres et choisi la base d'affichage (décimal, binaire, hexadécimal, flottant, message).
- 

Figure 2.17 : éditeur de variables

(g) Editeur de programme :

La structure logicielle d'une application peut-être monotâche elle est alors associée à une seule tâche utilisateur avec d'éventuels sous-programmes : la tâche **MAST**.

Exemple de structure monotâche avec Grafcet :

- **La flèche** indique l'ordre d'exécution des 3 modules Prl, chart et Post.

Prl : Traitement préliminaire. Exécuté avant les traitements séquentiel et postérieur, il permet de traiter tous les événements ayant une influence sur ces derniers. C'est donc uniquement dans le traitement préliminaire qu'il faut agir sur les bits associés aux étapes (mise à 0 ou à 1 des bits étapes %Xi ou %Xi.j par les instructions SET et RESET).

- **Chart** : Le Grafcet permet de représenter graphiquement et de façon structurée le fonctionnement d'un automatisme séquentiel.
- **Post** : Traitement postérieur. Il est exécuté après le Grafcet. C'est ici qu'il est recommandé de programmer l'écriture des sorties %Qi.x, pour être certain qu'elle n'a été effectuée qu'une fois pour chacune d'elles.
- **Sr** : Sous-programme. Les modules sous-programmes se programment comme les précédents en tant qu'entités séparées en : langage à contacts, liste d'instructions ou littéral structuré. Les appels aux sous-programmes s'effectuent dans les sections ou depuis un autre sous-programme (8 niveaux d'imbrications maximum). Les sous-programmes sont aussi liés à une tâche, un même sous-programme ne peut pas être appelé depuis plusieurs tâches.

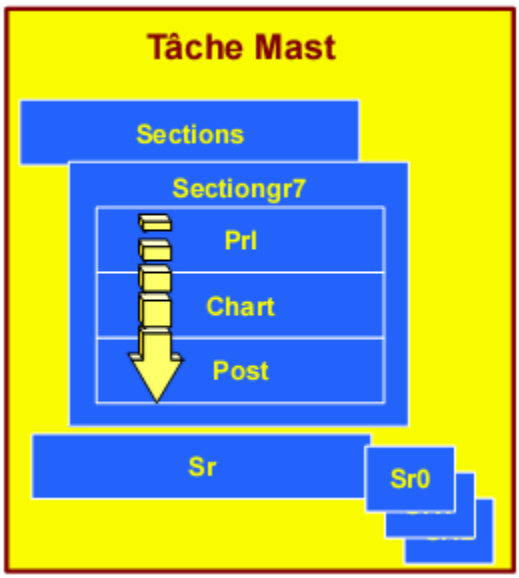


Figure 2.18 : La tâche MAST

(h) Traitement interne :

Le système réalise implicitement la surveillance de l'automate (gestion des bits et mots système, mise à jour des valeurs courantes de l'horodateur, mise à jour des voyants d'état, détection des passages RUN/STOP..) et le traitement des requêtes en provenance du terminal (modifications et animation).

- **Acquisition des entrées** : Ecriture en mémoire de l'état des informations présentes sur les entrées.
- **Traitement du programme :** Exécution du programme application, écrit par l'utilisateur.

- **Mise à jour des sorties :** Ecriture des bits ou des mots de sorties associés aux modules TOR et métier associés à la tâche selon l'état défini par le programme application avec ou sans Grafcet. Les fonctions de régulation doivent être programmées dans une tâche périodique.

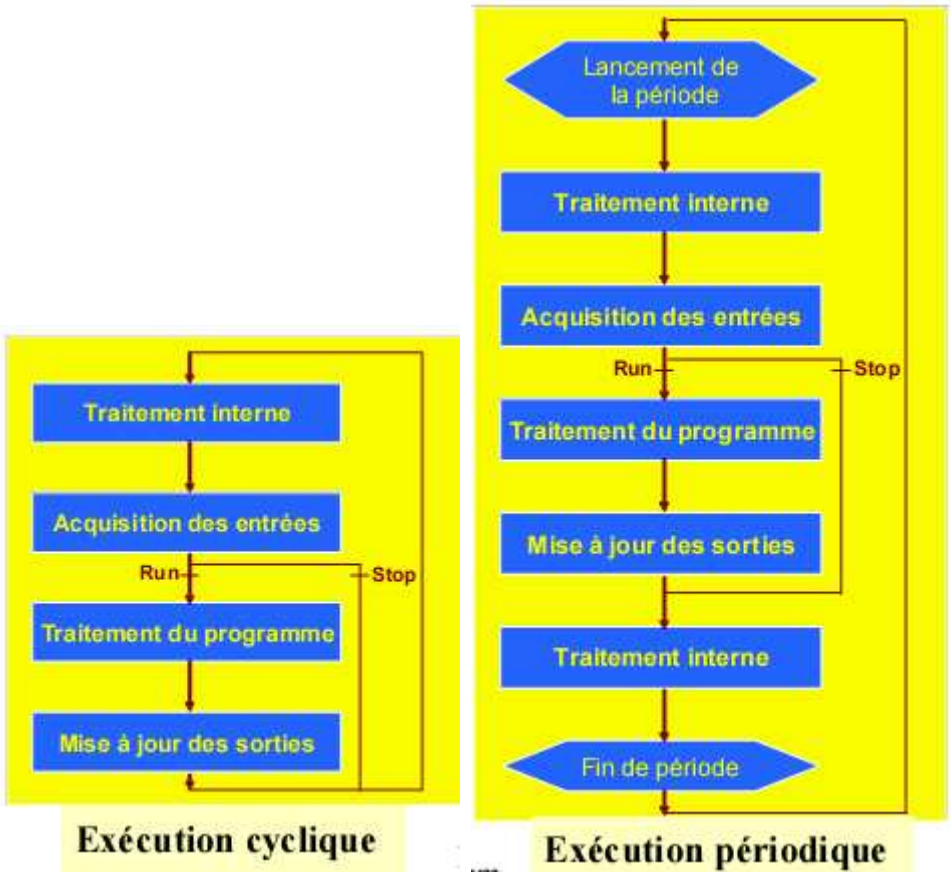


Figure 2.19 : La tâche périodique

(i) Chargement, essai :

Le programme est transféré du PC vers l'API par un câble de type TSX CPU.

- Vérifier sur le bloc de visualisation de l'API le bon fonctionnement des entrées programmées, même en STOP.
- En RUN vous pouvez vérifier le bon fonctionnement des sorties à blanc, puissance coupée.
- En mode connecté on peut observer l'évolution du fonctionnement, des tables d'animation permettent de vérifier les valeurs des objets du système.

(2) Le logiciel STEP 7

(a) Mode d'emploi

Avant de créer le projet, différentes approches sont possibles. En effet, le logiciel STEP 7 est libre de l'ordre pour la procédure de manipulation.

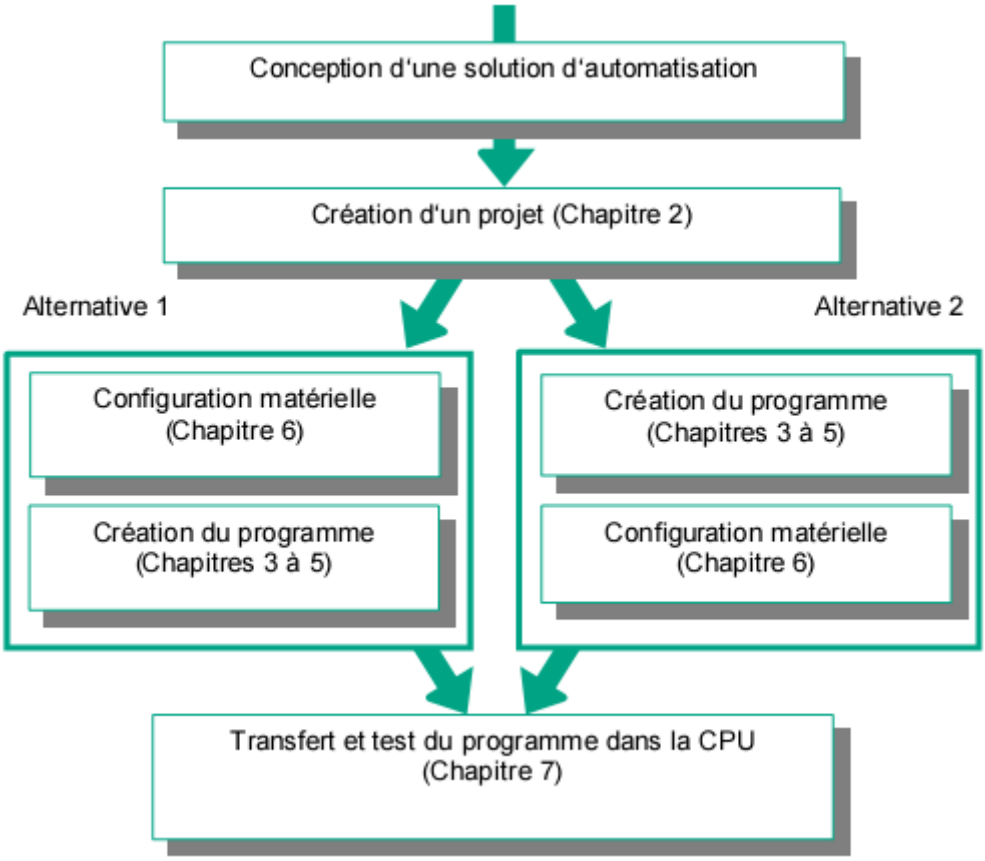


Figure 2.20 : procédure de manipulation

(b) Lancer SIMATIC Manager et créer un projet

Le lancement de STEP 7 fait s'ouvrir le gestionnaire de projets SIMATIC Manager. L'assistant de STEP 7 est par défaut toujours activé. Celui-ci a pour but d'assister dans la création du projet STEP 7. La structure du projet sert à ordonner les données et programmes créés au cours de celui-ci.

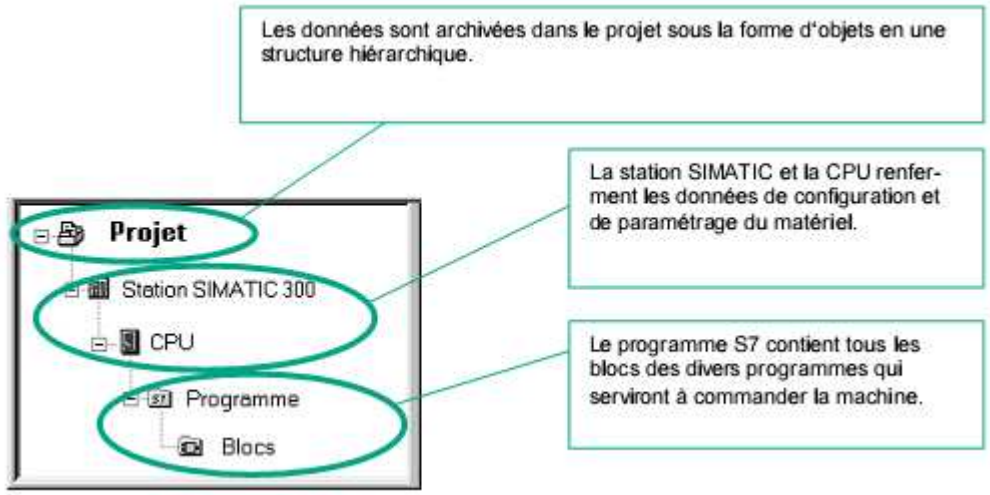


Figure 2.21 : lancement du projet

(c) Structure du projet dans SIMATIC Manager et appel de l'aide de STEP 7

Dès que l'Assistant est refermé, SIMATIC Manager apparaît de nouveau avec la fenêtre du projet "Getting Started" qui vient d'être créé ouverte. C'est à partir de cette fenêtre que toutes les fonctions et les autres fenêtres de STEP 7 pourraient être appelé.

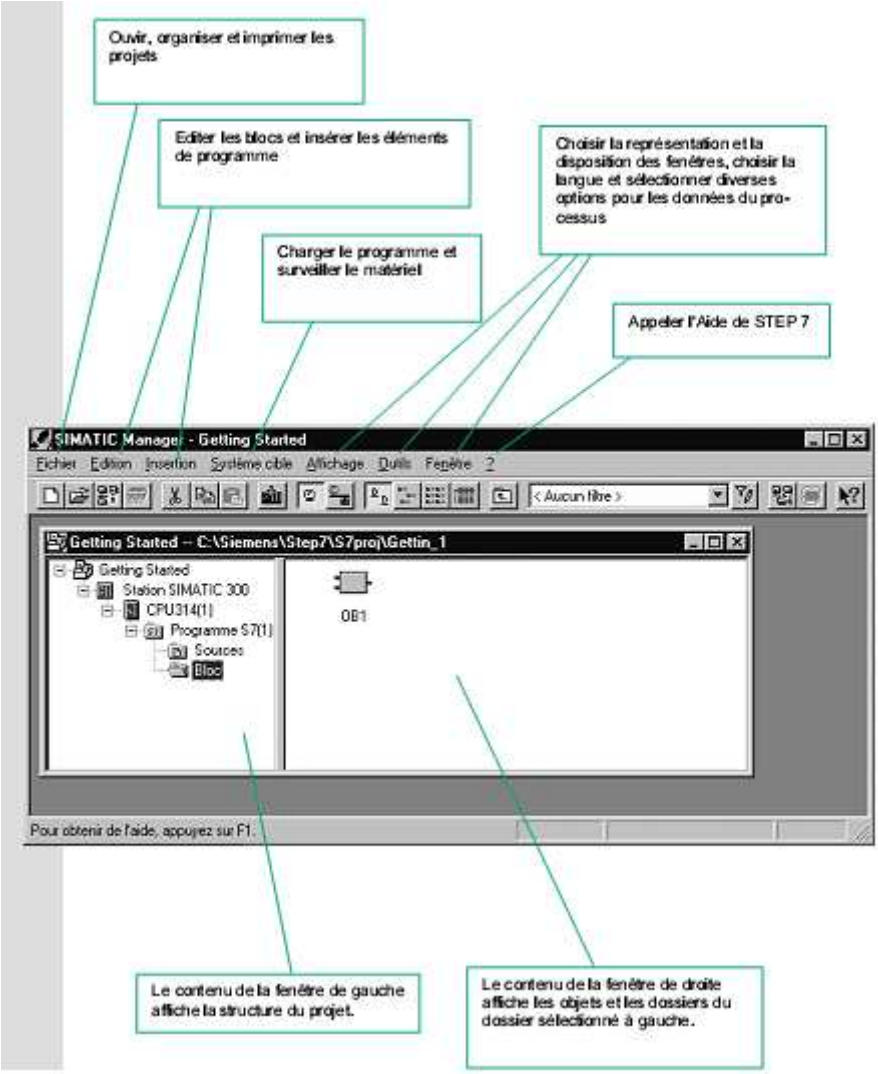


Figure 2.22 : Structure du projet

(d) Création d'un programme dans l'OB1

- Ouvrir l'éditeur de programme dans la vue CONT, LIST ou LOG et ouvrir l'OB1.
- Choisir le langage de programmation : CONT, LIST ou LOG

Pour créer les programmes S7, STEP 7 dispose trois langages de programmation CONT, LIST ou LOG. Dans la pratique, il important de décider pour l'un de ces langages.

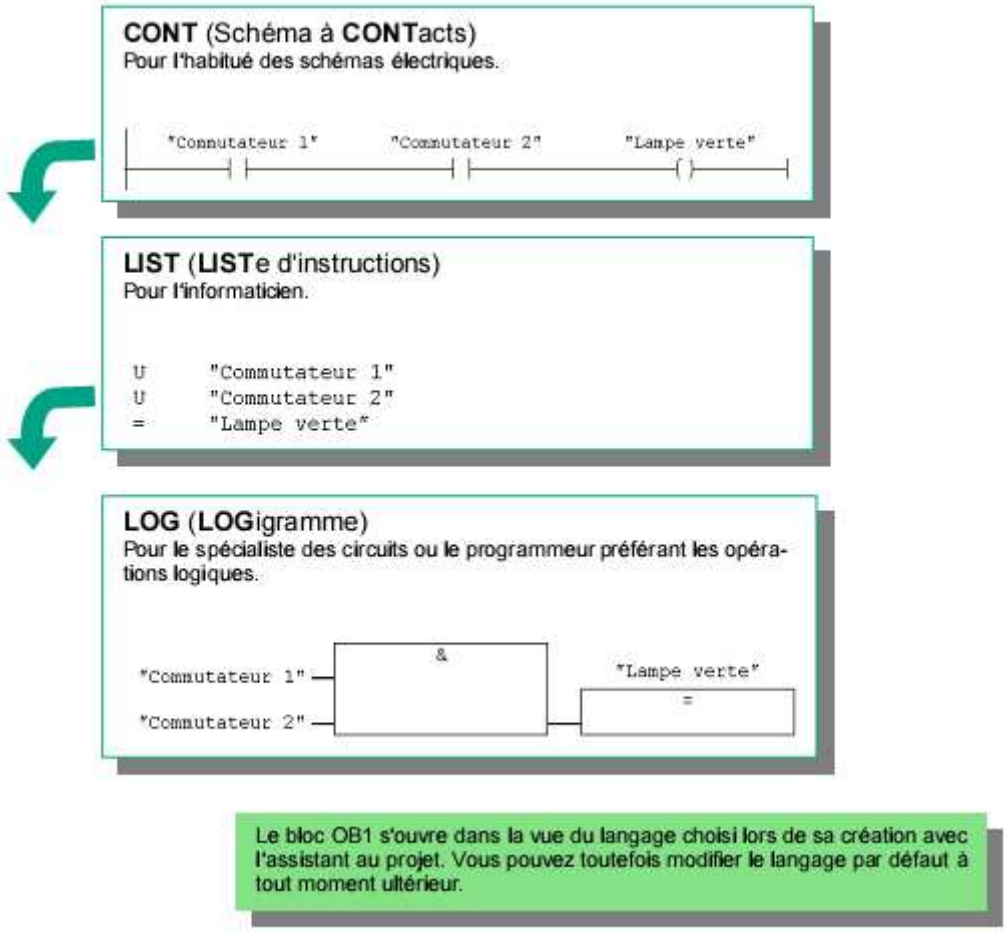


Figure 2.23 : langages de programmation

(e) L'éditeur de programme CONT/LIST/LOG

Les programmes blocs sont édités dans l'éditeur de programme CONT/LIST/LOG. Vous voyez représentée ici à titre d'exemple la vue CONT.

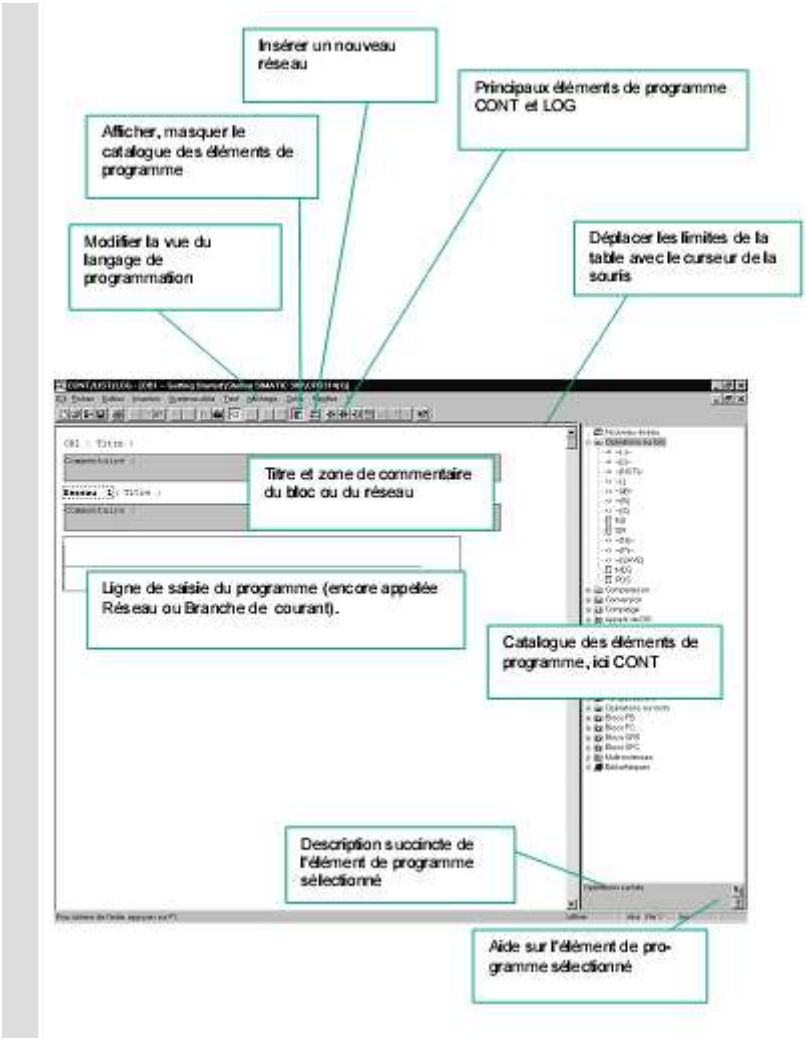


Figure 2.24 : éditeur de programme CONT

(3) Le logiciel WAGO-IO-PRO 32

(a) Description :

Cette note d'application décrit comment lire ou écrire des données d'un automate programmable industriel TSX 17-20 ou TSX 47-20, à partir d'un contrôleur de bus de terrain WAGO.

(b) Câblage :

Le dialogue entre la station WAGO et le TSX 17-20 se fait par l'intermédiaire du câble de configuration du contrôleur de la station et la prise terminale du TSX 17-20. Vu que la liaison série du contrôleur est une liaison série RS 232 et que celle de la prise terminale est une liaison série RS 485, il faut pour établir un dialogue entre les deux équipements utiliser un convertisseur RS 232 / RS 485.

(c) Schéma général :

Figure 2.25 : schéma général

(d) Lire ou écrire des données de l'API :

- Pour pouvoir lire ou écrire des données d'un API, copier les fichiers suivants dans le dossier lib2 de WAGO-IO-PRO 32

  - Sercomm.lib
  - Sercomm.hex
  - Serial_interface_01.lib
  - TSX_DATA.lib

Le répertoire par défaut est C:\Program Files\WAGO-IO-PRO 32\Lib2

- Lancer le logiciel WAGO-I/O-PRO 32 et ouvrir le projet TSX_DATA.pro

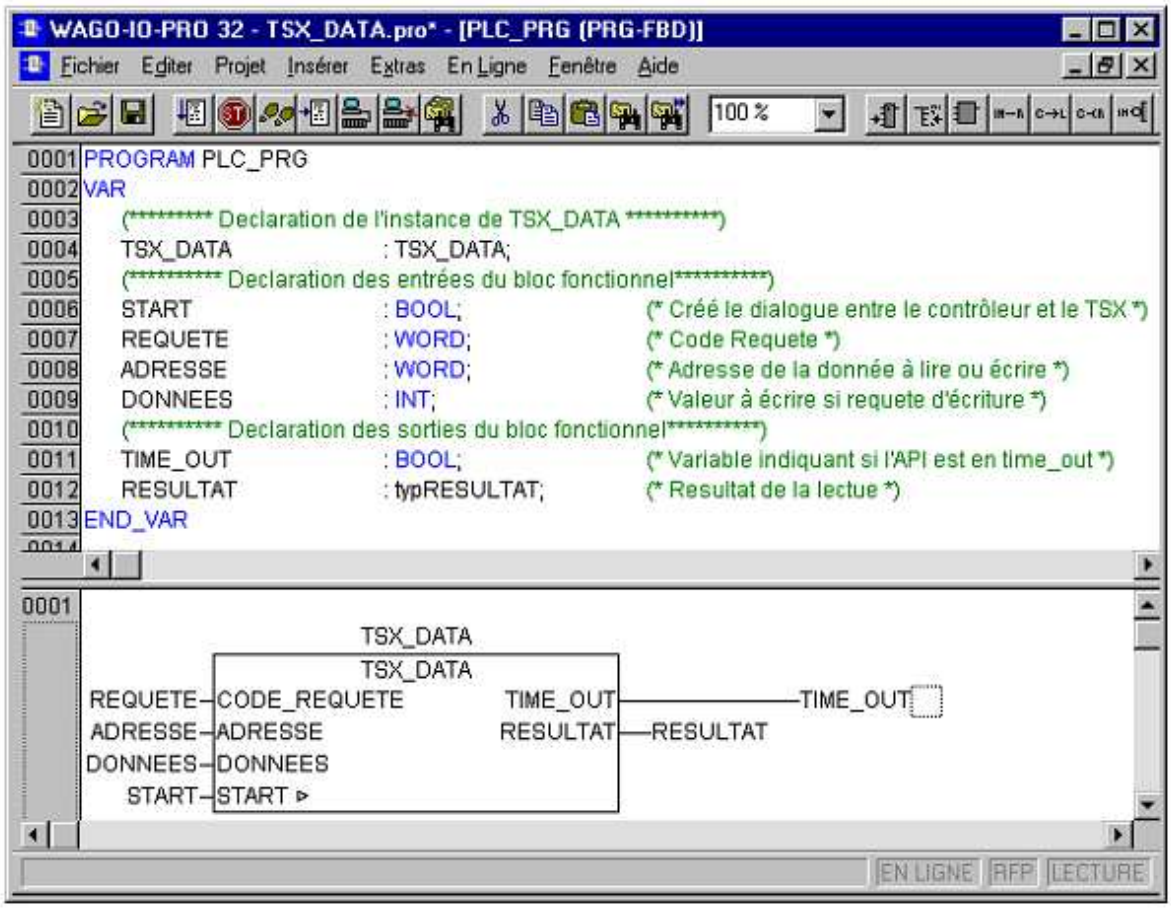


Figure 2.26 : éditeur des données de l'API

(e) Description du bloc fonctionnel TSX_DATA :

Ce bloc fonctionnel réalise la fonction de lecture ou d'écriture de données du TSX.

- Pour réaliser cette fonction il faut remplir les entrées de ce bloc fonctionnel de la façon suivante :

- CODE_REQUETE (WORD) :

- Pour lire les données :

  - 00 : lecture d'un bit (Bi) ;

  - 04 : lecture d'un mot (Wi) ;

  - 05 : lecture d'un mot constant (CWi) ;

  - 06 : lecture d'un mot système (SWi) ;

  - 09 : lecture des paramètres d'un temporisateur (Ti) ;

- 11 : lecture des paramètres d'un compteur (Ci).

- Pour écrire les données :

  - 20 : écriture d'un mot (Wi) ;

  - 25 : écriture présélection compteur (Ci,p).

- **ADRESSE (WORD) :** Adresse de la donnée à lire ou écrire

  - Exemple : Pour un mot interne W1 l'adresse est 1. En généralisant, pour un mot Wi l'adresse est i.

- **START (BOOL) :**

Etablie le dialogue entre la station WAGO et l'automate TSX. Sur un front descendant de celle-ci la fonction demandée a été réalisée.

Le bloc fonctionnel renvoi les paramètres suivants :

- **TIME_OUT (BOOL) :** Variable indiquant si le programme est en time out.

- **RESULTAT (typRESULTAT):**

- **RESULTAT.VALEUR_COURANTE :** valeur courante du mot lu ;

- **RESULTAT.VALEUR_PRESELECTION :** valeur de la présélection pour les compteurs et temporisateurs.

(f) Execution du programme :

- Compiler le projet.

- Télécharger le programme dans le contrôleur, puis le démarrer.

- Pour établir un dialogue avec l'automate remplir les entrées du bloc fonctionnel aux valeurs voulues puis mettre la variable START à TRUE.

CHAPITRE3 : LES ETAPES POUR REALISER UN AUTOMATE PROGRAMMABLE INDUSTRIEL

3.1 Description du cahier des charges

3.1.1 Critères de choix d'un automate :

Le choix d'un automate programmable est en premier lieu le choix d'une société ou d'un groupe et les contacts commerciaux et expériences vécues sont déjà un point de départ.

Les grandes sociétés privilégieront deux fabricants pour faire jouer la concurrence et pouvoir "se retourner" en cas de "perte de vitesse" de l'une d'entre elles.

Le personnel de maintenance doit toutefois être formé sur ces matériels et une trop grande diversité des matériels peut avoir de graves répercussions. Un automate utilisant des langages de programmation de type GRAFCET est également préférable pour assurer les mises au point et dépannages dans les meilleures conditions.

La possession d'un logiciel de programmation est aussi source d'économies (achat du logiciel et formation du personnel). Des outils permettant une simulation des programmes sont également souhaitables.

Il faut ensuite quantifier les besoins :

- Nombre d'entrées / sorties : le nombre de cartes peut avoir une incidence sur le nombre de racks dès que le nombre d'entrées / sorties nécessaires devient élevé.
- Type de processeur : la taille mémoire, la vitesse de traitement et les fonctions spéciales offertes par le processeur permettront le choix dans la gamme souvent très étendue.
- Fonctions ou modules spéciaux : certaines cartes (commande d'axe, pesage ...) permettront de "soulager" le processeur et devront offrir les caractéristiques souhaitées (résolution, ...).
- Fonctions de communication : l'automate doit pouvoir communiquer avec les autres systèmes de commande (API, supervision ...) et offrir des possibilités de communication avec des standards normalisés (Profibus ...).

3.1.2 Le cahier des charges

(1) Definitions :

Un **cahier des charges** est un document (Dans son acception courante un document est généralement défini comme le support physique d'une information.) visant à définir exhaustivement les spécifications de base d'un produit ou d'un service à réaliser. Outre les spécifications de base, il décrit ses modalités d'exécution. Il définit aussi les objectifs à atteindre et vise à bien cadrer une mission.

En interne, le cahier des charges sert à **formaliser les besoins** et à les expliquer aux différents acteurs pour s'assurer que tout le monde est d'accord. Il sert ensuite à sélectionner le prestataire et à organiser la relation tout au long du projet (Un projet est - dans un contexte professionnel - une aventure temporaire entreprise dans le but de créer un produit ou un service). Il est considéré comme un référentiel contractuel partagé par le prestataire et l'équipe interne, ce qui en fait un **outil** (Un outil est un objet finalisé utilisé par un être vivant dans le but d'augmenter son efficacité naturelle dans...) **fondamental** de communication (La communication concerne aussi bien l'homme (communication intrapsychique, interpersonnelle, groupale...) que l'animal...) du chef de projet.

Le cahier des charges est un document contractuel il fait office de contrat entre le client et le prestataire de service.

Les entreprises se basent sur le cahier de charges pour estimer le coût de réalisation du produit dans le cas d'un appel d'offre. Il est donc le premier pas vers un projet maîtrisé.

Un cahier des charges fonctionnel (**CdCF**) est un outil requis pour déceler et exprimer le besoin par une approche systémique.

(2) Il faut se poser les bonnes questions

- Dans quel(s) but(s) faire ce projet ?
- Comment le mettre en place ?
- Qu'est-ce que l'on gagne avec ce projet ?
- Qu'est-ce que l'on perd si on ne le fait pas ?
- Qui va s'assurer du suivi du produit ?
- Qui va s'assurer de la demande du projet?

Penser à faire participer un maximum d'utilisateurs du produit pour savoir quels sont les besoins de chacun par rapport à ce produit.

(3) Structure du cahier des charges

- **Rédaction d'un cahier de charges**

Il faut savoir que le cahier de charges vit tout au long du projet. Imposer une lecture fastidieuse est le meilleur moyen pour que personne ne l'utilise. Sa rédaction doit favoriser les résumés (une page maximum), les listes à puce et les tableaux. En général, il est constitué de quatre parties. La première explique pourquoi le projet existe, quels sont ses objectifs et qui le pilote : rôles respectifs de la Maîtrise d'ouvrage et de la Maîtrise d'œuvre (MOA et MOE), procédures de validation, etc. La seconde énumère les **besoins fonctionnels, techniques et organisationnels** ainsi que les contraintes et les exigences. La troisième partie liste les prestations attendues. La quatrième partie définit le cadre de la réponse : planning de l'appel d'offres, documents attendus, etc.

- **Les conditions juridiques à inclure dans un cahier des charges**

La participation d'un juriste dans la rédaction du cahier de charges est primordiale, car elle assure au minimum le résultat en cas de problèmes imprévus. En effet, l'expérience montre que c'est le meilleur moyen de se faire imposer le contrat du prestataire, donc d'être potentiellement lésé en cas de différend finissant devant les tribunaux.

Pour éviter d'en arriver là, il suffit de définir des pré-requis en amont du projet et d'en faire mention dans le cahier des charges ou d'ajouter un projet de contrat en annexe quand c'est possible. Ce dernier **accélère la rédaction** et donc la négociation (La négociation est la recherche d'un accord, centrée sur des intérêts matériels ou des enjeux quantifiables entre deux négociateurs) du contrat final et garantit un commencement du projet dans un cadre juridique partagé.

Précisément, il y a trois **conditions juridiques** à inclure dans le cahier des charges :

- un **planning précis** (des dates, des délais impératifs avec engagement de résultat et un mécanisme de pénalité en cas de retard) ;
- les **clauses de cession des droits** incluant les droits des éléments de contenu utilisés pour créer le projet;
- les **modalités de validation** (déroulement et supports des validations, répartition des rôles lors de la recette, PV de réception provisoire, PV de réception définitive, etc.).

Par exemple, en droit français, il est important de noter que, le vendeur d'un bien ou d'un service est tenu à une obligation générale d'information vis-à-vis de son client (code de la consommation, Livre Ier, titre Ier, chapitre Ier).

- **Partie technique dans un Cahier des Charges**

La partie technique d'un Cahier des Charges doit se limiter à **énumérer les contraintes techniques avérées**. Le recours à tel ou tel serveur d'applications, le budget (Un budget est un document comptable prévisionnel distinguant les recettes et les dépenses.) de fonctionnement, la plate-forme de déploiement par exemple. L'erreur la plus courante est de confondre préférences et contraintes. Ceci cause des incompréhensions et des remises en causes aussi tardives que dramatiques. Pour remédier à ce problème, il faut confier la rédaction du cahier des charges à un non technicien et fournir le même niveau de détail pour chaque besoin.

On peut aussi faire appel à un expert pour valider la cohérence du cahier des charges.

- **Exemple de cahier des charges d'un site Internet**

Chapitre 1 – Présentation du projet

- Contexte (Étape →3/5) :
- Historique
- Objet du projet
- Organisation du projet
- Comité de pilotage
- Groupe de projet
- Maîtrise d’ouvrage
- Maîtrise d’œuvre
- Prestataire
- Environnement du projet
- Existant fonctionnel
- Existant technique

Chapitre 2 - Description du futur site

- Périmètre du projet
- Positionnement (On peut définir le positionnement comme un choix stratégique qui cherche à donner à une offre (produit, marque...) du futur site
- Description générale du projet

- Description des besoins fonctionnels
- Besoins fonctionnels en « front office »
- Besoins fonctionnels en « back office »
- Points clés fonctionnels
- Description des besoins techniques
- Architecture technique
- Configuration logicielle
- Configuration matérielle
- Sécurité
- Reprise de l'existant

Chapitre 3 - Prestations attendues

- Présentation des prestations attendues
- Cadre de la réponse
- Informations générales
- Documents à remettre lors de l'appel d'offre
- Bordereau de prix
- Critères de choix
- Exigences
- Exigences techniques

Exigences ergonomiques et graphiques

3.2 Définition et description du projet d'application et son environnement

3.2.1 Définition du projet d'application d'un api

L'automate programmable industriel est aujourd'hui le constituant le plus répandu des automatismes.

On le trouve non seulement dans tous les secteurs de l'industrie, mais aussi dans les services (gestion de parkings, d'accès à des bâtiments) et dans l'agriculture (composition et délivrance de rations alimentaires dans les élevages), etc. Il répond aux besoins d'adaptation et de flexibilité et de nombreuses activités économiques actuelles. Cette place majeure soulève bien sûr un certain nombre de questions.

- Dans quel contexte est utilisé l'automate programmable ?
- Comment est son principe de fonctionnement ?

- Quelles sont les possibilités des paramètres à considérer et ses mises à jour ?
- Quelles sont les équations qui peuvent traduire le fonctionnement du système ?
- Etc.

On trouve alors qu'il faut bien définir le domaine d'application de l'automate et son environnement

3.2.2 Description de l'environnement

L'environnement, c'est-à-dire le CONTEXTE physique, social, économique, politique, ... joue un rôle essentiel dans le fonctionnement du système automatisé.

On peut une fois listés ces critères les utiliser via des tableaux, des graphes, ou toute méthode d'aide à la décision, mais la pondération d'un élément, voire sa simple pertinence, sont fonction de l'entreprise et de l'application.

Dans tous les cas, la démarche consiste à quantifier les besoins et à comparer cette mesure aux capacités des matériels proposés par les fournisseurs. Presque toujours, un certain nombre de critères (ergonomie, sûreté de fonctionnement) n'auront pas de mesure fiable et ce sera à l'utilisateur d'en faire une évaluation nécessairement entachée de subjectivité. Nous examinerons les facteurs les plus importants suivant le rôle joué par le dispositif de commande, en attirant l'attention sur des particularités parfois perdues de vue, avant d'envisager les différentes solutions, API ou autre, et, si la solution API est retenue, le choix d'une marque plutôt qu'une autre.

3.3 Décrire l'organisation d'un automate programmable

3.3.1 Structure générale d'un API :

- L'API est construite autour d'un microprocesseur,
- Les entrées sont nombreuses et acceptent des signaux venant des capteurs industriels,
- Les sorties sont traités pour actionner des contacteurs, relais…
- Le langage de programmation est simple et accessible rapidement,
- La mémoire est en partie prise par le programme moniteur (contrôle de fonctionnement de l'automate, gestion interne des traitements).

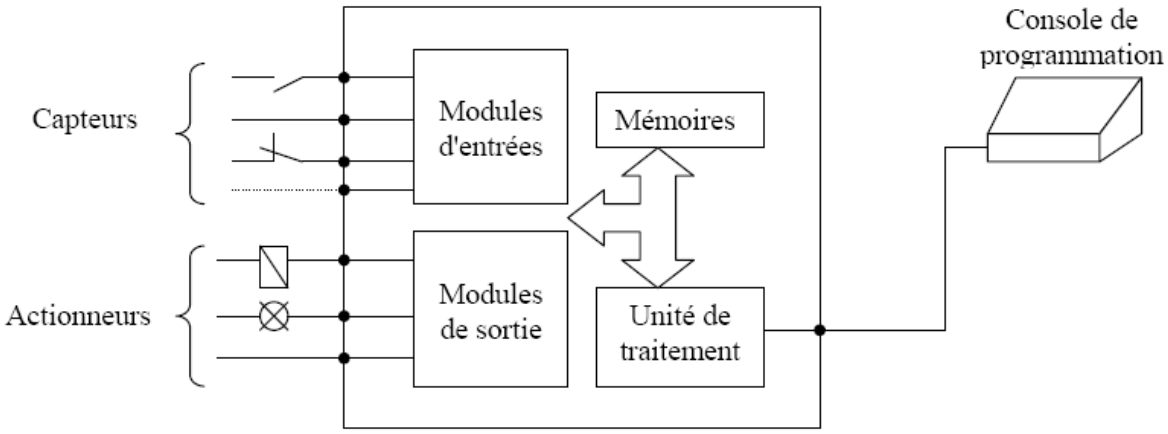


Figure 3.1 : Structure générale d'un api

- Le programme de traitement des informations est stocké en mémoire
- L'unité de traitement pilote le fonctionnement de l'automate
- La console de programmation assure le dialogue entre l'opérateur et l'automate

  Enregistrement du programme et transfert dans l'automate.

3.3.2 Principe de fonctionnement des automates :

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.

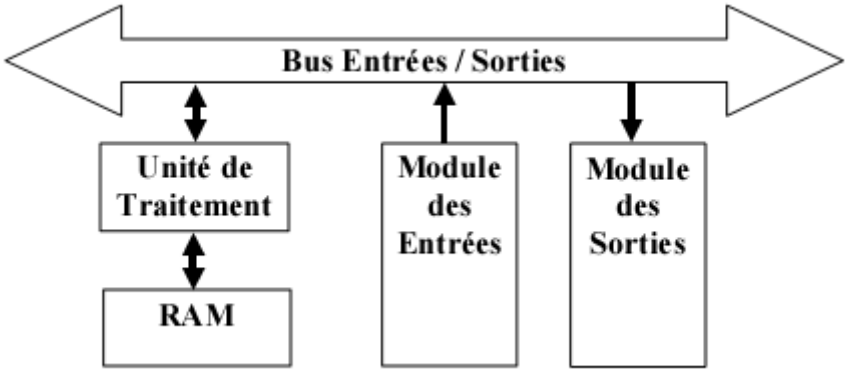


Figure 3.2 : Principe de fonctionnement des automates

En d'autres termes, un automate exécute son programme de manière cyclique.

- Lecture des entrées
- Traitement du programme
- Écriture des sorties

Le temps d'exécution d'un cycle est contrôlé par une temporisation appelée **chien de garde**

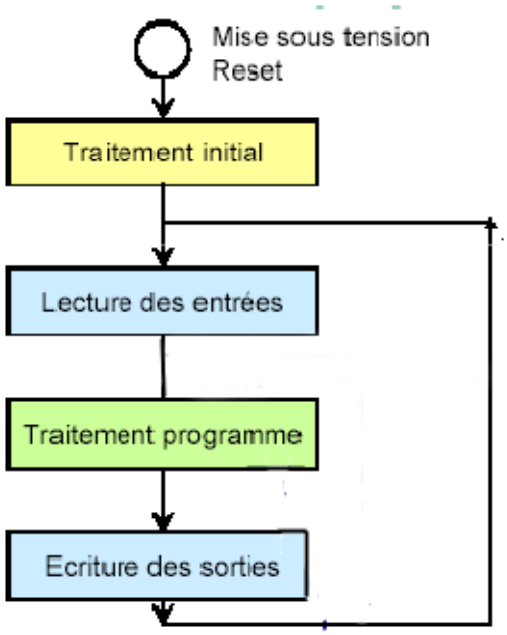


Figure 3.3 : cycle d'exécution

Fonctionnement cyclique:

Les cycles s'enchaînent les uns après les autres, le temps de cycle dépend de la durée d'exécution du programme.

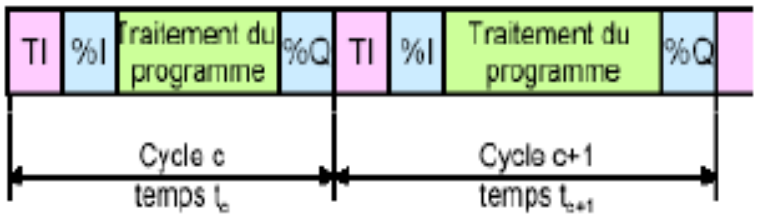


Figure 3.4 : chronogramme d'un cycle

Attention, le temps de cycle doit être compatible avec l'application

Mode périodique:

Il est parfois possible de fixer le temps de cycle (important en cas de régulation)

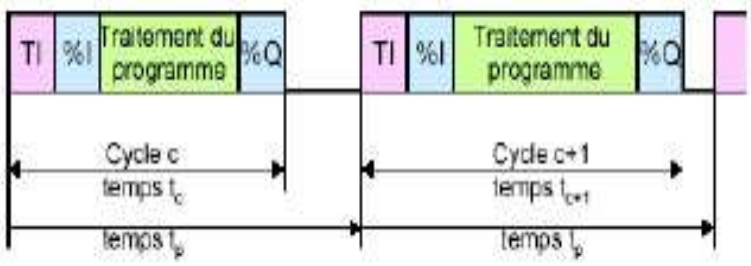


Figure 3.5 : mode périodique

Temps de réponse :

Lorsque l'évolution d'une entrée doit entraîner l'activation d'une sortie, le temps de retard entre les 2 évènements dépend du temps de cycle.

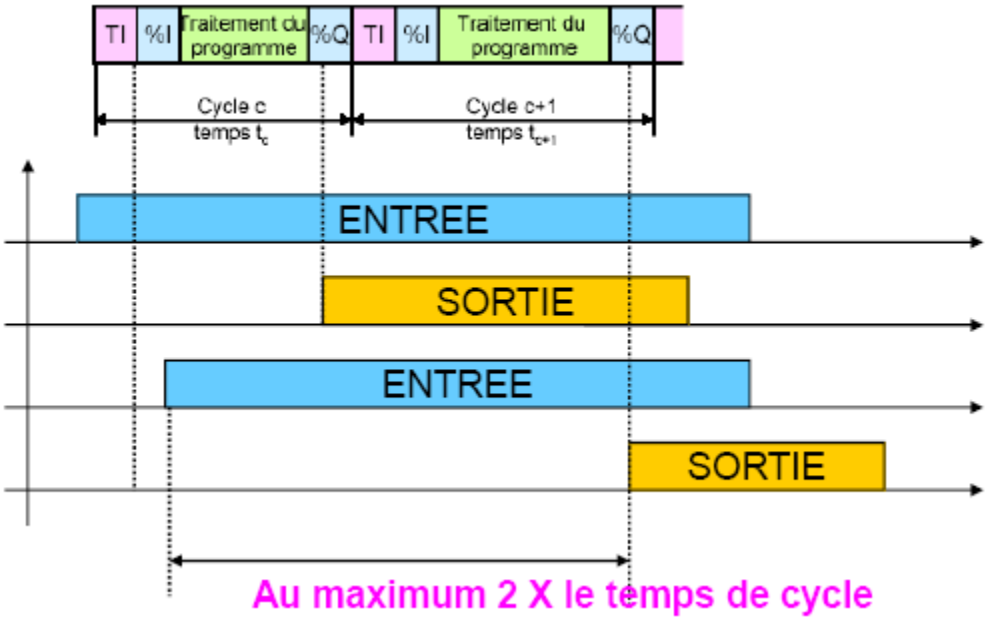


Figure 3.6 : Temps du cycle

3.3.3 Mise en œuvre d'un automate programmable industriel

(1) Prise en compte du système à automatiser

On réalise l'affectation et l'adressage des entrées de l'automate en fonctions des capteurs, boutons de commandes utilisées, puis des sorties de l'automate avec les contacteurs, voyants ...

La description du cycle à réaliser est effectuée et transcrit sur un cahier des charges sous forme de schéma, logigramme, équations logiques, grafcet, gemma.

L'automate programmable étant relié aux préactionneurs (affectation Entrées/ Sorties) et ayant son propre langage de programmation.

Comme exemple explicatif, prenons le **GRAFCET** qu'il faut traduire 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	Etapes 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

Tableau 3.1 : Mise en œuvre d'un système à automatisé

REMARQUE: Le logiciel AUTOMGEN permet l'édition graphique proche des grafcet, puis l'affectation des entrées/sorties, la génération du programme pour l'automate concerné, la simulation du programme, le transfert et la supervision de son exécution.

(2) Ecriture du programme

Il est réalisé avec un ordinateur compatible, à partir du système d'exploitation Windows en utilisant le logiciel du constructeur (PL7 pour Schneider) .

- Il permet de programmer l'application dans différents langages : Schéma à contact Ladder, grafcet, liste d'instruction.

Ou à l'aide d'un logiciel d'assistance à la programmation

- Ces logiciels permettent la saisie directe à partir d'un schéma électrique ou d'un grafcet et ils traduisent automatiquement le cycle à réaliser en code instructions de l'automate sélectionné.

Ou à l'aide d'une console de programmation spécifique à l'automate

(3) Transfert du programme dans l'automate programmable

Le transfert du programme consiste à envoyer le programme réalisé dans la mémoire de l'automate.

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.

Puis on effectue les réglages des différents paramètres de temporisation, comptage ...

(4) Vérification du fonctionnement

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

- ***Prendre connaissance du système*** (dossier technique, des grafcet 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...***

NB :

Il existe deux façons de vérifier le fonctionnement :

- En simulation (sans Partie Opérative).
- En condition réelle (avec Partie Opérative).

Simulation sans P.O.	Condition réelle
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 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.).* ⇒ .... Toutes les évolutions du GEMMA et des grafcets doivent être vérifiées.	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 grafcets doivent être vérifiées.

Tableau 3.2 : Vérification du fonctionnement

(5) Recherche des dysfonctionnements

(a) Causes de dysfonctionnements

Un dysfonctionnement peut avoir pour origine :

- un composant mécanique défaillant (préactionneur, actionneur, détecteur,...).•un câblage incorrect ou défaillant (entrées, sorties).
- un composant électrique ou électronique défectueux (interface d'entrée ou de sortie).
- une erreur de programmation (affectation d'entrées-sorties, ou d'écriture).
- un système non initialisé (étape, conditions initiales...).

(b) Méthode de recherche des causes de dysfonctionnement

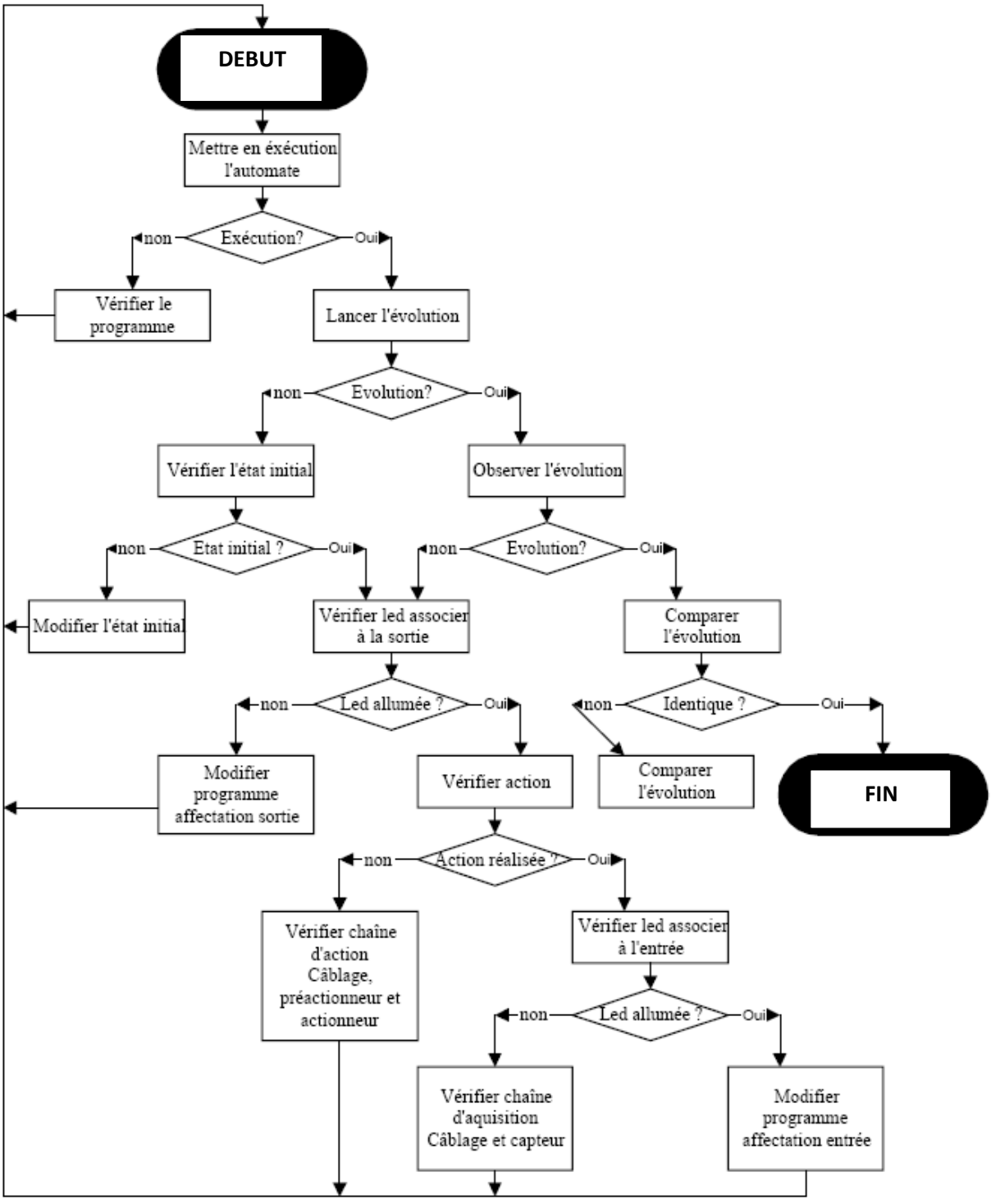


Figure 3.7 : Méthode de recherche des causes de dysfonctionnement

CHAPITRE4 : APPLICATION ET SIMULATION

4.1 Description du cahier des charges

Réaliser une carte de commande pour une station de pompage d'eau à base d'un microcontrôleur de la famille PIC 16f877 permettant la gestion de marche et d'arrêt des motopompes.

En se basant sur les tarifs de l'énergie de la JIRAMA (jour, pointe, nuit), optimiser le fonctionnement des pompes de façon à réduire le coût des factures d'énergie.

4.2 Présentation et étude de la Station de pompage

4.2.1 Introduction

Dans un premier lieu, on va essayer de définir qu'est ce qu'une station de pompage ainsi que le principe de fonctionnement des pompes utilisées.

Dans un deuxième temps, on verra les courbes de modulation de la consommation journalières qui vont nous permettre de dégager un organigramme de fonctionnement optimisé suivant les tarifs d'énergie de l'électricité (Jour/Pointe/Nuit).

Enfin, une partie sera consacrée à la mise en équation du fonctionnement de la station de pompage.

4.2.2 Définition et principe de fonctionnement d'une station de pompage

(1) Qu'est-ce qu'une Station de pompage ?

Une station de pompage est un ouvrage hydromécanique destiné à pomper l'eau potable d'un niveau géographique bas à un niveau plus haut. On distingue dans une station de pompage :

Poste de transformation, groupe électropompe, armoire de commande et de protection, conduit de raccordement de protection, réservoirs, accessoires (vanne, clapets, ….).

(a) Principe de fonctionnement d'une pompe :

Le principe essentiel de la pompe est de refouler une quantité d'eau d'un niveau bas à un niveau haut. C'est pour cela que le fonctionnement de la pompe est composé de deux phases essentielles qui sont :

- L'aspiration.

- Le refoulement.

Mais ces phases sont différentes d'une pompe à une autre suivant des critères bien définis comme le débit, la pression, la hauteur, le rendement et la puissance.

(b) Constitution d'une pompe centrifuge :

Une pompe centrifuge est construite pour répondre à des conditions précises de fonctionnement : débit Q à élever à une hauteur H.

D'une façon générale, une pompe comporte :

- Un organe mobile, la roue ou encore appelé turbine ou impulseur ou rotor.
- Des organes fixes, diffuseur, encore stator qui peut être moulé directement avec le flasque ou démontable.

(2) Les courbes de modulation de la consommation journalières

(a) Prestation des tarifs de la JIRAMA :

	PRIX DE L'ENERGIE (Ariary/KWh)		
Pompage	JOUR	POINTE	NUIT
Eau	200	250	150

Tableau 4.1 : Prix de l'énergie électrique

(b) Interprétation :

D'après le tableau précédent, nous remarquons que le prix de l'énergie est plus élevé pendant les heures de pointe.

L'utilisation d'eau fréquente pendant les heures de pointe alors la consommation est plus élevée durant cette période que pendant les autres heures (jour et nuit).

Notre but consiste alors à optimiser le fonctionnement des pompes suivant ces 3 périodes, c'est-à-dire qu'on veut commander le démarrage et l'arrêt des moteurs.

D'une part si le niveau d'eau atteint son niveau bas, il faut que les moteurs démarrent. D'autre part il faut essayer de minimiser le temps de fonctionnement des motopompes durant la période critique définie par le tableau 8 où le tarif de l 'énergie est cher (heures de pointe). Par la suite il faut démarrer les moteurs de telle sorte qu'on peut avoir un réservoir plein au début de la période critique. Pour assurer cet objectif on va faire une étude sur l'optimisation.

(c) les courbes de modulation journalière:

C'est une étude faite, d'une part pour contrôler l'énergie consommée par les motopompes durant une période bien déterminée (consommation journalière) et d'autre part pour assurer que notre unité de contrôle (microcontrôleur) est capable de stocker certaines données concernant la consommation d'eau en fonction du temps, d'où le traçage de la courbe de modulation annuelle.

Exemple :

Tableau de mesures :

Heure (h)	00 ➔ 04	04 ➔ 08	08 ➔ 12	12 ➔ 16	16 ➔ 20	20 ➔ 24
Consommation (m^3)	5.000	20.000	52.000	39.000	27.000	20.000

Tableau 4.2: Tableau de mesures

Nous avons partagé les heures d'une journée en six périodes, chaque période comporte quatre heures.

Notre unité de contrôle va calculer la quantité d'eau consommée pendant cette période et par la suite nous obtiendrons une courbe de modulation journalière. (figure 55)

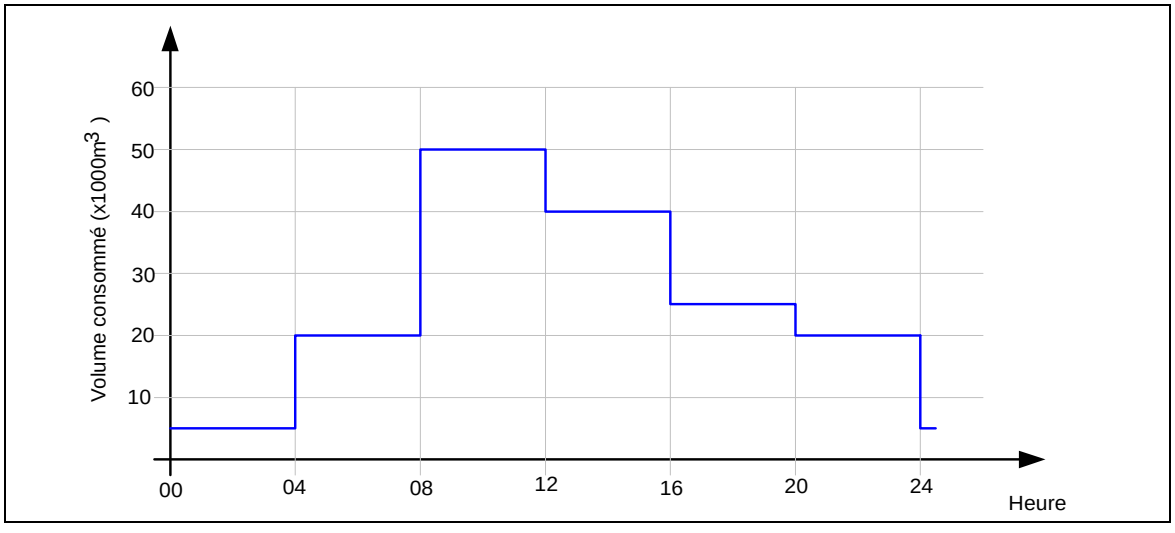


Figure 4.1 : courbe de modulation journalière

Cette étude va être répétée chaque jour, ensuite une courbe de consommation moyenne va être déduite et stockée en mémoire qui servira comme base de données pour les calculs d'optimisation.

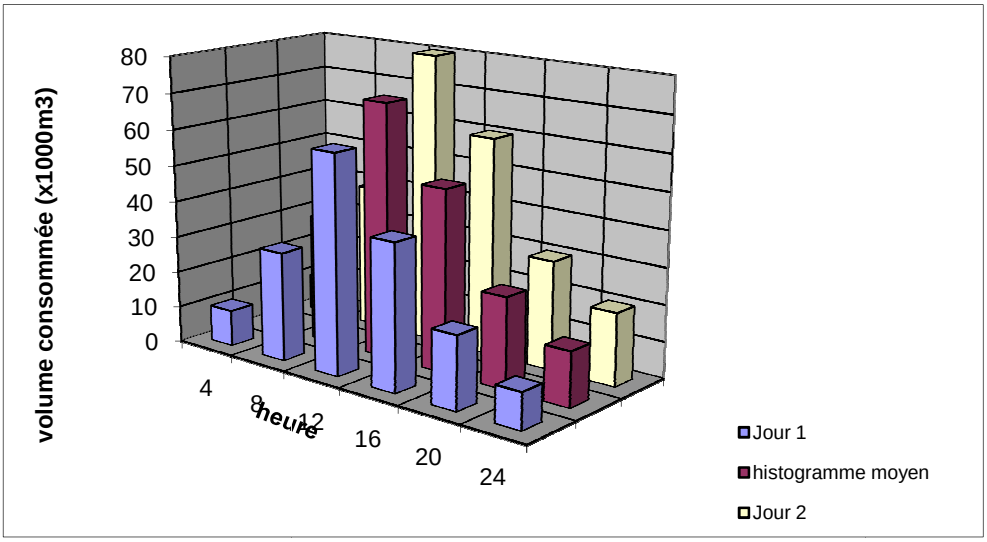


Figure 4.2 : exemple de calcul de la valeur moyenne entre deux jours

(d) Le schéma de la station de pompage

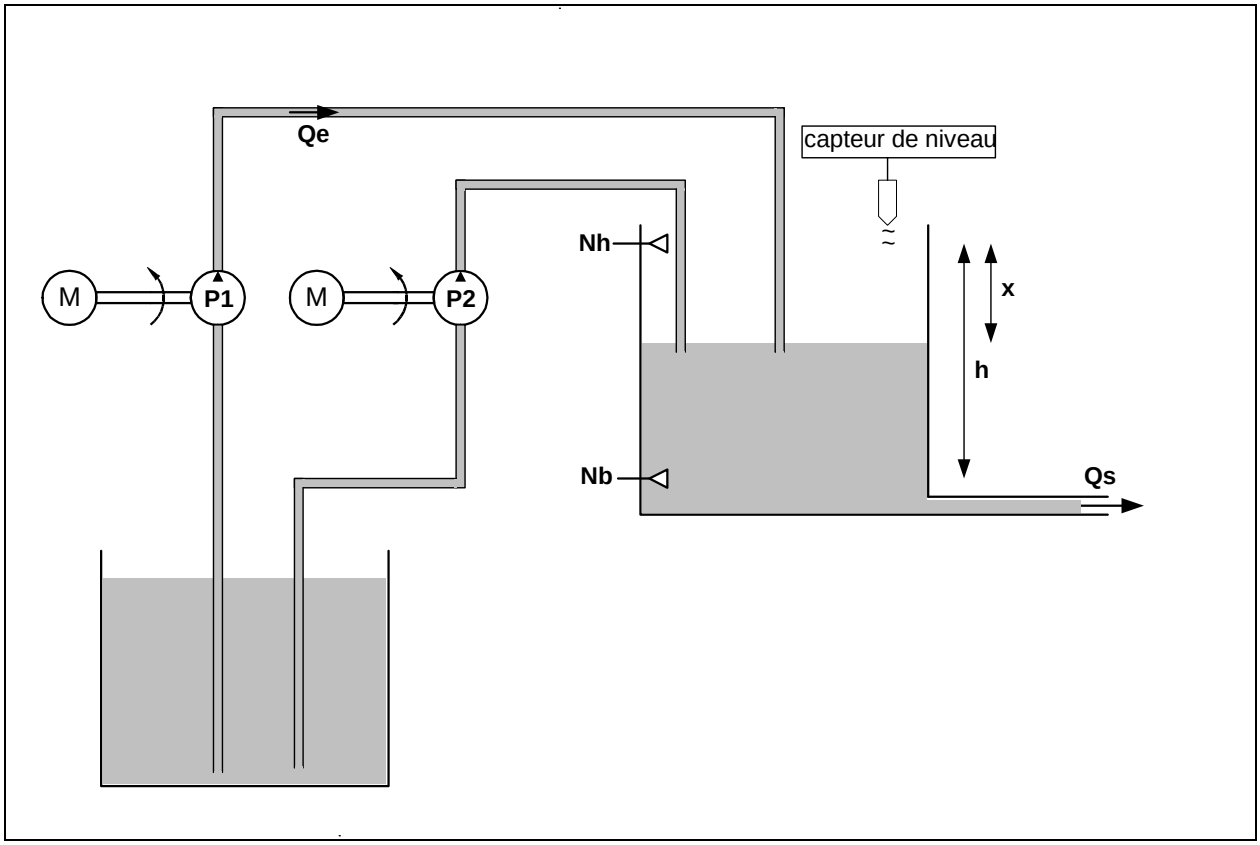


Figure 4.3 : schéma de la station de pompage

Les notations :

P1, P2 : Pompes
M : moteur
$\boldsymbol{Qe}$ **:** Débit des pompes
$\boldsymbol{Qs}$ **:** Débit de consommation
$\boldsymbol{Q1}$ **:** Débit de la pompe 1
$\boldsymbol{S}$ **:** Surface du réservoir
$\boldsymbol{h}$ **:** Hauteur du réservoir
$\boldsymbol{x}$ **:** Volume vide
$\boldsymbol{Nh}$ **:** Capteur de niveau haut
$\boldsymbol{Nb}$ **:** Capteur de niveau bas
$\boldsymbol{hp}$ **:** Période de pointe
$\boldsymbol{hj}$ **:** Période de jour

$\boldsymbol{t}$ *:* Temps

$\boldsymbol{tf}$: Temps de fonctionnement

Etant donné la hauteur ($\boldsymbol{h}$), la surface du réservoir ($\boldsymbol{S}$) et le volume d'eau pompée (débit des pompes constant $\boldsymbol{Qe}$), on peut déterminer par un simple calcul la quantité consommée $\boldsymbol{Qs}$ d'eau pendant un temps $\boldsymbol{t_2 - t_1}$.

$$\boldsymbol{Qs = S.[x(t_2) - x(t_1)] - Qe}$$

(e) Algorithme d'optimisation :

Pour optimiser le fonctionnement d'une station de pompage d'eau, on a établi l'algorithme suivant :

Démarche à suivre :

Les règles de base :

- Si $\boldsymbol{Nb}$ =1 alors P1=1 ;

⇒ Si le niveau bas est atteint alors la pompe principale P1 doit forcément fonctionner pour répondre aux besoins des consommateurs.

- Si $\boldsymbol{Nh}$ =1 alors P1=0 ;

⇒ Si le réservoir est plein c'est à dire que le niveau haut est atteint alors il n'y aura pas de pompage.

Les règles de sécurité :

- Si P1=1 et $\boldsymbol{x}(\boldsymbol{t}$=15mn) - $\boldsymbol{x}(\boldsymbol{t}$=0) >0 Alors P2=1 ;

⇒ Si le pompe P1 fonctionne et le niveau du réservoir diminue pendant un temps de fonctionnement de 15mn alors la pompe de secours P2 doit fonctionner.

- Si P1=1 et Si P2=1 et $\boldsymbol{x}(\boldsymbol{t}$=15mn) - $\boldsymbol{x}(\boldsymbol{t}$=0) < $\boldsymbol{Q1}$ alors P2= 0 ;

⇒ Si P1 et P2 fonctionnent en même temps et si la consommation est inférieure au débit de la pompe P1 alors il faut arrêter la pompe de secours.

Les règles d'optimisation :

- Si $\boldsymbol{t} = \boldsymbol{hp} - \boldsymbol{tf}$ et $\boldsymbol{Qs}(\boldsymbol{hp}) > (\boldsymbol{h} - \boldsymbol{x})$ alors P1=1 ;

⇒ Sachant que le temps de fonctionnement ($\boldsymbol{tf}$) nécessaire pour le remplissage du réservoir est estimé à une durée bien déterminée, notre unité de contrôle va vérifier : est ce que le volume d'eau dans le réservoir suffira à la consommation durant la période de pointe ? Sinon elle entamera le pompage à l'instant heure de pointe ($\boldsymbol{hp} - \boldsymbol{tf}$).

- Si $\boldsymbol{t} = \boldsymbol{hp}$ et $\boldsymbol{Nb}$ =1 alors P1=1 jusqu'à $(\boldsymbol{h} - \boldsymbol{x}) > \boldsymbol{Qs}(\boldsymbol{hp})$;

⇒ Si nous sommes dans la période de pointe et le niveau dans le réservoir est bas alors la pompe principale P1 doit fonctionner le temps nécessaire juste pour remplir la quantité de

consommation prévue pour cette période.

- Si $\boldsymbol{t} = \boldsymbol{hj} - \boldsymbol{tf}$ et $\boldsymbol{Nh}$ =0 alors P1=1 ;

⇒ Notre unité de contrôle doit toujours entamer une période jour par un réservoir plein.

Remarque :

Les règles d'optimisation peuvent être violées pendant le fonctionnement alors que les règles de base et de sécurité ne peuvent jamais être violées

4.2.3 Conclusion

En conclusion, nous pouvons dire que le prix d'énergie est élevé pendant les heures de pointes, nous voulons alors optimiser le fonctionnement des pompes à des conditions précises, c'est à dire commander le démarrage et l'arrêt des moteurs.

Pour assurer cet objectif et répondre à ce problème, nous réalisons une carte électronique à base de microcontrôleur dont le programme de commande sera basé sur l'algorithme de fonctionnement qui a été établi précédemment.

4.3 Décrire l'organisation d'un automate programmable

4.3.1 Structure générale d'un API

Notre carte comme l'indique la figure ci-dessous est composée de plusieurs unités qui assurent le bon fonctionnement de la carte de commande de la station de pompage.

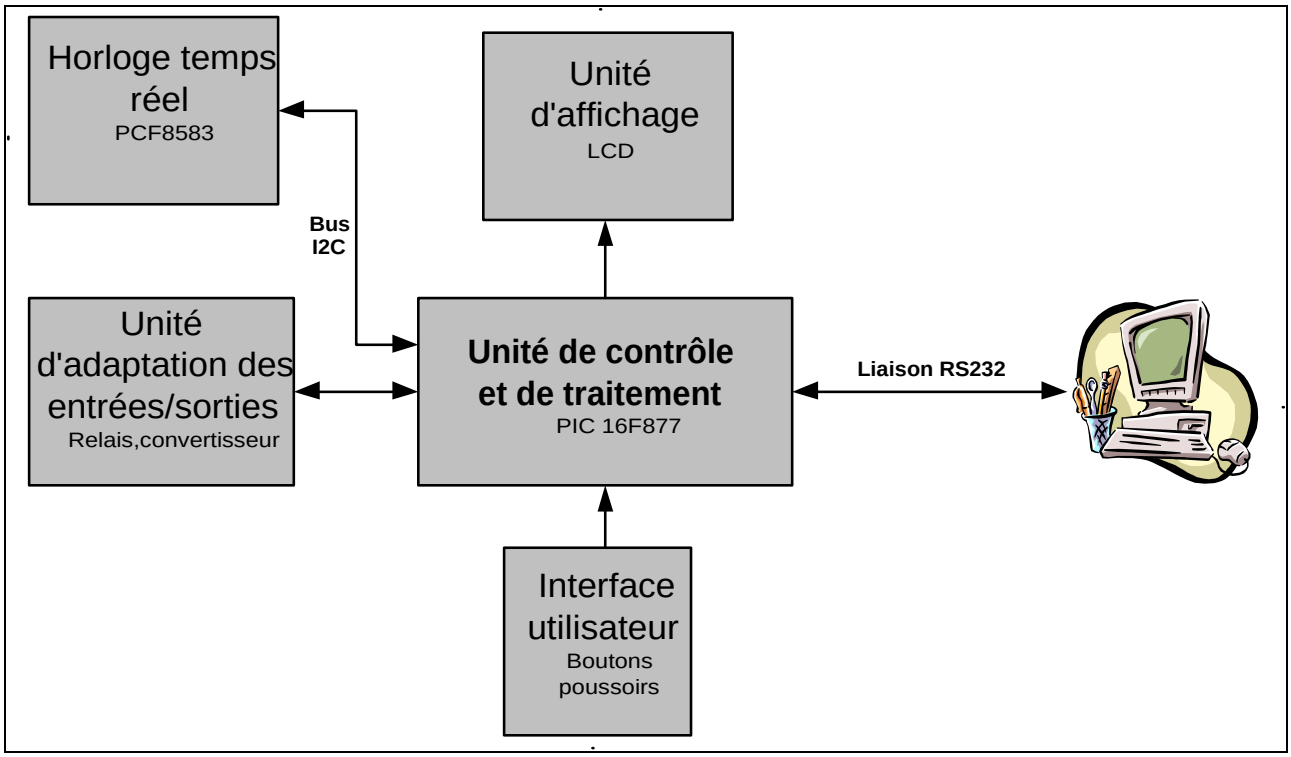


Figure 4.4 : Schéma synoptique de la carte

En effet, on trouve l'unité de traitement et de contrôle des données qui est le microcontrôleur (PIC 16F877).

Un afficheur LCD est lié directement à la carte pour afficher les données.

Un clavier composé de trois touches de commande pour le défilement des instructions contenues dans l'afficheur.

Une horloge temps réel qui joue le rôle d'un vrai calendrier.

Et des actionneurs qui assurent la commande des deux pompes (relais).

(1) Mise en œuvre d'un afficheur L C D :

(a) Présentation de l'afficheur LCD :

Les afficheurs à cristaux liquides sont des modules compacts intelligents et nécessitent peu de composants externes. Ils sont utilisés avec beaucoup de facilité. Ils sont pratiquement les seuls à être utilisés sur les appareils à alimentation par pile.

Plusieurs afficheurs sont disponibles sur le marché et ne se différent pas les unes des autres, seulement par leurs dimensions, (1 à 4 lignes de 6 à 80 caractères), mais aussi par leurs caractéristiques techniques et leurs tension de services.

Certains sont dotés d'un rétro éclairage de l'affichage. Cette fonction fait appel à des LED montées derrière l'écran du module, cependant, cet éclairage est gourmand en intensité (250mA max).

(b) Présentation d'un écran LCD :

Qu'il soit à une ou deux lignes, un afficheur LCD se présente sous la forme suivante :

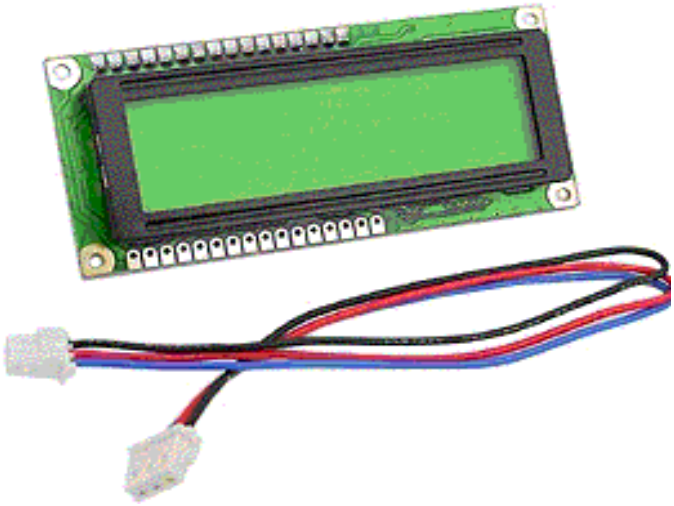


Figure 4.5 : Photo d'un afficheur LCD

Au-dessus de l'écran à cristaux liquides proprement dit, on trouve une série de 14 broches aux rôles Suivantes :

- Broche 1 : masse ;
- Broche 2 : Vcc ;
- Broche 3 : luminosité ;
- Broche 5, R/W : sélection du mode lecture ou écriture :

0	ecriture
1	lecture

Tableau 4.3 : sélection du mode lecture ou écriture

- Broche 6, E : Commande des opérations d'écriture ou de lecture ;
- Broche 7 à 14 : utilisées pour le transfert des données ou des instructions. Le transfert peut se faire sur 8 bits, toutes les broches sont alors utilisées, ou sur 4 bits, dans ce cas, seules les broches 11 à 14 sont utilisées.

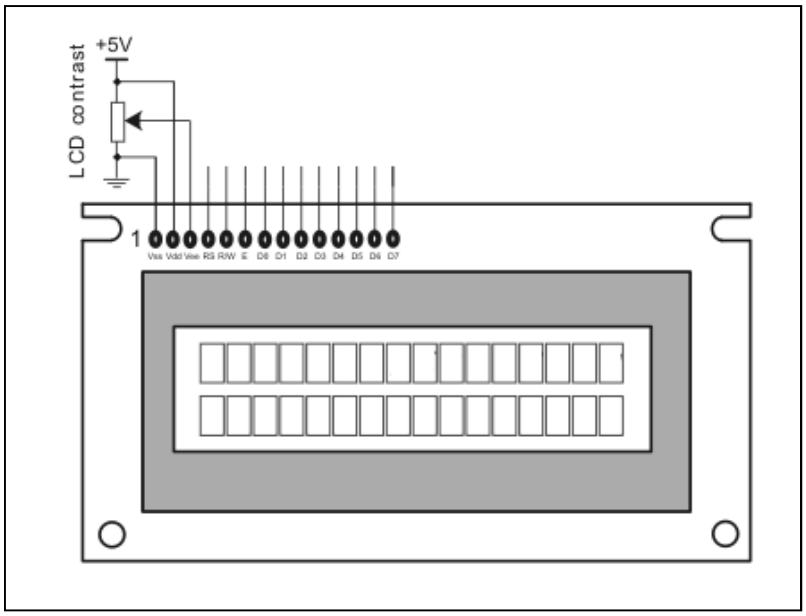


Figure 4.6 : schéma d 'un afficheur LCD

(c) Fonctionnement :

Un afficheur LCD est capable d'afficher tous les caractères alphanumériques usuels et quelques Symboles supplémentaires. Pour certains afficheurs, il est même possible de créer ses propres Caractères.

Chaque caractère est identifié par son code ASCII qu'il faut envoyer sur les lignes D0 à D7 broches 7 A 14. Ces lignes sont aussi utilisées pour la gestion de l'affichage avec l'envoi d'instructions telles que l'effacement de l'écran, l'écriture en ligne 1 ou en ligne 2, le sens de défilement du curseur.

(d) Principales instructions :

Effacement de l'écran en remplissant du caractère « espace »

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	1

Tableau 4.4 : effacement de l'écran

Retour en début de première ligne

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	1	*

Tableau 4.5 : Retour en début de première ligne

Aller en début de seconde ligne

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	0	0	0	0	0

Tableau 4.6 : Aller en début de seconde ligne

Mode d'affichage

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	1	I/D	S

Tableau 4.7 : Mode d'affichage

- Si I\D = 1 : le déplacement du curseur vers la droite ;
- Si I\D = 0 : le déplacement vers la gauche.
- Si S = 1 : le déplacement du texte affiché vers la droite
- Si I\D = 1 vers la gauche.
- Si I\D=0
- Si S=0 : aucun déplacement du texte.

Contrôle d'affichage

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	1	D	C	B

Tableau 4.8 : Contrôle d'affichage

- Si D = 1 : affichage visible.
- Si C = 1 : curseur visible.
- Si B = 1 : inversion.

Déplacement affichage et curseur, sans opération d'écriture

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	1	S\C	R\L	*	*

Tableau 4.9 : Déplacement affichage et curseur

S\L	R\L	ACTIONS
0	0	Déplacement du curseur vers la droite
0	1	Déplacement du curseur vers la gauche
1	0	Déplacement de l'affichage vers la droite
1	1	Déplacement de l'affichage vers la gauche

Tableau 4.10 : Déplacement affichage et curseur, sans opération d'écriture

Fonction

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	DL	N	F	*	*

Tableau 4.11 : fonction

- Si DL = 1 : donnée sur 8 bits, sur 4 bits si DL = 0.
- Si N = 0 : affichage sur 1 ligne, sur 2 ligne si N = 1.
- Si F = 0 : taille des caractères 5x 8,5 x 10 si F= 1.

(e) Connexion de l'afficheur sur la carte :

Dans notre application, nous avons utilisé un écran LCD alphanumérique de 2 lignes et de 24 caractères.

Cet écran est connecté au microcontrôleur sur ses deux ports D et B, on a conservé tous les broches du port D pour lier les données (D0 à D7).

Et on a pris trois broches du port B :

Un ensemble d'instructions spécialisées permet de le piloter très facilement et disponible dans le logiciel micropascal.

Il est également possible de piloter l'écran directement avec des instructions de commande sous forme binaire.

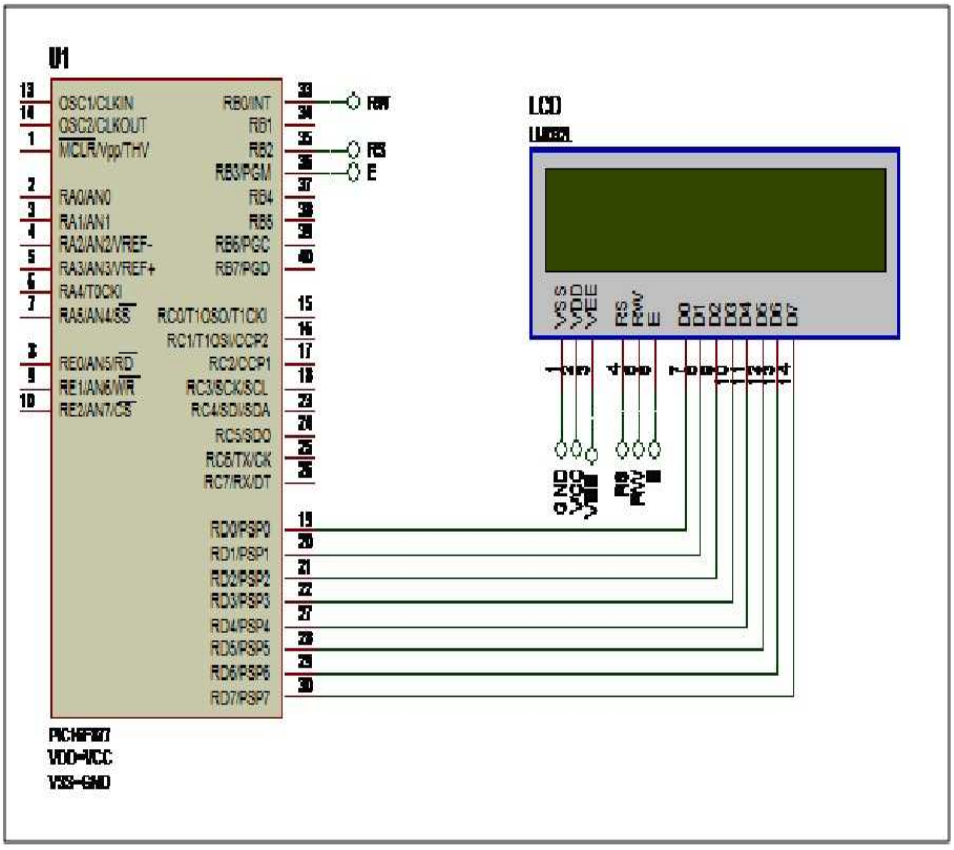


Figure 4.7 : Montage de l'afficheur LCD sur le PIC

(f) La liaison à la norme RS 232 :

(i) Présentation :

Les liaisons séries permettant la communication entre deux systèmes numériques en limitant le nombre de fils de transmission.

La liaison série à la norme RS 232 est utilisée dans tous les domaines de l'informatique (exemple : port de communication com1 et com2 des PC permettant la communication avec des périphériques tels que modem et souris). Elle est de type asynchrone, c'est à dire qu'elle ne transmet pas de signal d'horloge.

(ii) Brochage du port RS232 :

Brochage	Nom
1	DCD
2	RXD
3	TXD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

Tableau 4.12 : Brochage du port RS232

- DCD (Data Carrier Detecte) : Cette ligne est une entrée active à l'état haute. Elle signale à l'ordinateur qu'une liaison a été établie avec un correspondant.
- RX (Reciver Data) : Cette ligne est une entrée. C'est ici que transitent les informations du correspondant vers l'ordinateur.
- TX (Transmit Data) : cette ligne est une sortie. Elle permet de véhiculer des données de l'ordinateur vers le correspondant.
- DTR (Data Terminal Ready) : Cette ligne est une sortie active à l'état haut Elle permet à l'ordinateur de signaler au correspondant que le port série à été libéré et qu'il peut être utilisé s'il le souhaite.
- GND (GrouND) : c'est la masse.
- DSR (Data Set Read) : Cette ligne est une entrée active à l'état haut. Elle permet au correspondant de signaler qu'une donnée est prête.

- RTS (Request To Send) : cette ligne est une sortie active à l'état haut. Elle indique au correspondant que l'ordinateur veut lui transmettre des données..
- CTS (Clear To Send) : Cette ligne est une entrée active à l'état haut. Elle indique à l'ordinateur que le correspondant est prêt à recevoir des donnés.
- RI (RING Indicator) : Cette ligne est une entrée active à l'état haut. Elle permet à l'ordinateur de savoir si un correspondant veut initier une communication avec lui.

(iii)Fonctionnement :

Pour pouvoir dialoguer avec le PC, notre microcontrôleur utilise son module USART signifie (Universal Synchronous Asynchronous Reciever Transmitter).

C'est donc un module qui permet d'envoyer et de recevoir des données en mode série, soit de façon synchrone, soit asynchrone. Le module USART de notre PIC gère uniquement deux pins, à savoir RC6/TX/CK et RC7/RX/DT.

Une liaison série synchrone nécessite une connexion dédiée à l'horloge, donc il reste une seule ligne pour transmettre les données. Alors qu'en mode asynchrone on n'a pas besoin d'une ligne d'horloge, il nous restera alors deux lignes pour communiquer, chacune étant dédiée à un sens de transfert. Nous pourrons donc envoyer et recevoir des données en même temps.

Les liaisons RS 232 sont des liaisons asynchrones très utilisées en informatique. Elle nécessite que l'émetteur et le récepteur soient informés de la vitesse choisie de transfert.

Puisque le récepteur connaît la vitesse du transfert il peut se passer du signal de synchronisation.

Trois lignes sont nécessaires à cette liaison.

TX : transmission de données.

RX: récepteur de données.
GND : masse

Grâce à cette liaison la carte peut servir d'interface entre un PC et un montage extérieure afin d'adapter les signaux TTL du microcontrôleur au standard RS232

Un MAX232 est montée de faucon classique, les lignes RX, TX et la masse sont disponibles sur un connecteur DB9 mâle qui permet ainsi de relier la carte au PC avec un simple câble série.

(g) Le MAX 232 :

Le pic 16F877 utilise les niveaux 0v et 5v pour définir respectivement les signaux <0> et <1>

La norme RS 232 définit des niveaux de +12v et –12v pour établir ces mêmes niveaux. Nous avons donc besoin d'un circuit (driver de bus) chargé de convertir les niveaux des signaux entre PIC et PC.

Ce circuit dispose de :

- Deux blocs dénommés T1et T2 qui convertissent les niveaux entrés en 0v et 5v en signaux sortis sous +12v et -12v. Les entrées de ces blocs sont donc dirigées vers le pic et les sorties sont connectées sur le port RS 232.
- Deux blocs dénommés R1 et R2, qui convertissent les niveaux entrés en +12v /-12v en signaux sortis sous 0v/5v.

Les entrées de ces blocs sont donc connectées sur le port RS232, les sorties sur le PC.

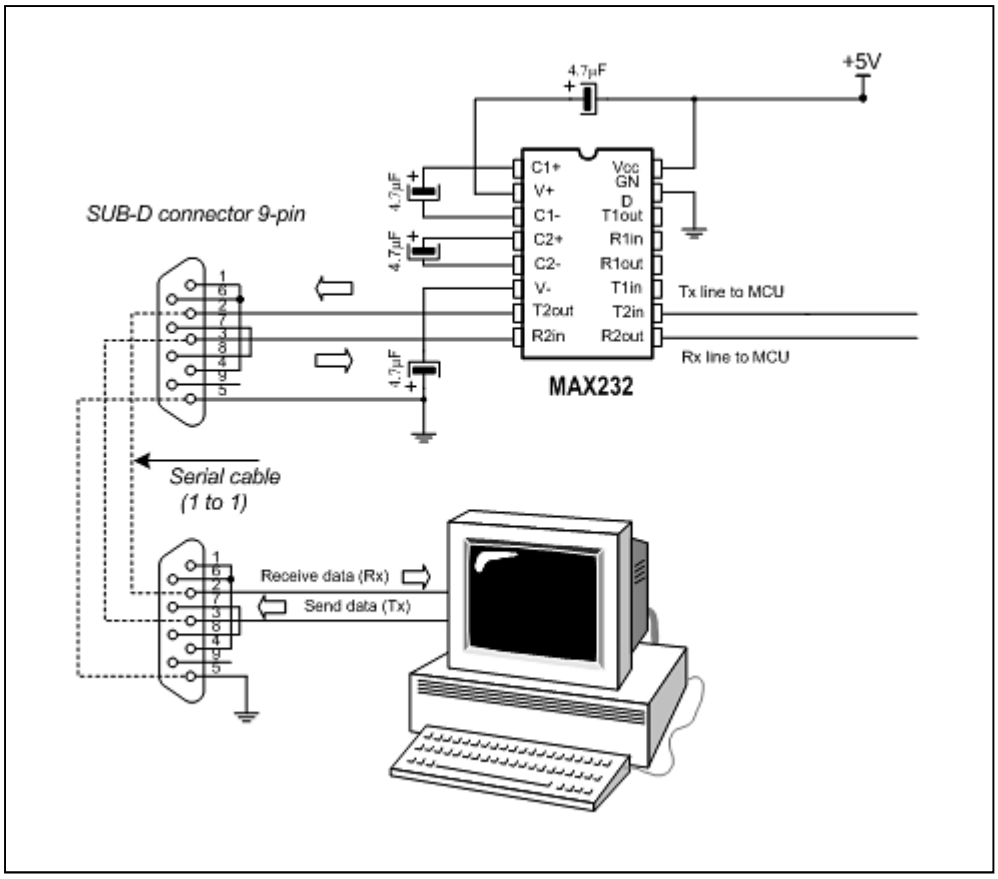


Figure 4.8 : Montage du MAX232 sur le PIC

(h) Le Bus I2C :

Le Bus I2C ou intercommunication a été conçu pour réaliser la liaison entre les circuits intégrés d'une manière platine.

Il se charge de la communication entre les périphériques.

Pour le bus I2C, le niveau dominant étant l'état bas. Les deux lignes SDA et SCL sont donc maintenues au niveau haut tant que le bus est libre.

Sur les deux lignes, le niveau est reconnu bas pour toute tension inférieure à 1,5 v et haut pour toute tension supérieure 3v.

Le bus I2C appartient à la catégorie des bus séries par opposition aux bus parallèles ou les données sont transmises par bloc, les données sont ici envoyées bit par bit par groupe d'octet sur la ligne SDA. La ligne SCL fonctionne comme une horloge sérielle d'un registre à décalage. Tant que la ligne SCL est à l'état haut, les données de la ligne SDA doivent être stables.

Lorsque la ligne SCL est à l'état bas, le circuit qui émet les données peut modifier l'état.

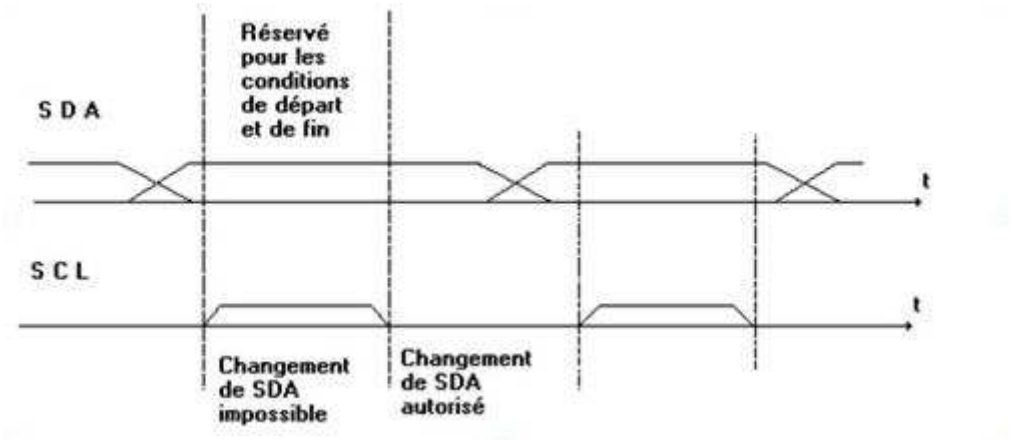


Figure 4.9 : Condition de validité des données sur le bus I2C

Certaines combinaisons particulières de niveaux et de fronts des deux lignes déterminant la condition de départ ou d'arrêt de la transmission des données.

- Condition de départ : Un front descendant sur SDA quand SCL est à l'état haut.
- Condition d'arrêt : Un front montant sur SDA quand SCL est à l'état haut.

(i) Module horloge temps réel :

Puisque le facteur temps est primordial dans le fonctionnement de notre système nous avons pensé à l'utilisation d'un circuit intégré de la famille PCF qui servira d'horloge temps réel.

Dans la famille des circuits intégrés compatibles au bus I2C, le PCF8583 satisfait à cette fonction.

PCF8583 : (ANNEXE 4)

Le PCF8583 fonctionne en véritable horloge calendrier c'est-à-dire en mode de 24 heures et sur une période de vingt quatre ans. Il possède une sortie d'interruption et de la RAM qui possède 232 octets disponibles en plus de ceux de sa propre fonction.

Caractéristiques électriques :

- consommation faible de courant.

- garantie de la fonction d'horloge et de rétention de mémoire sous 1v (et 2µA) ce qui permet de le secourir facilement par une batterie.

Le bloc diagramme de notre circuit nous aide à mieux comprendre les choses :

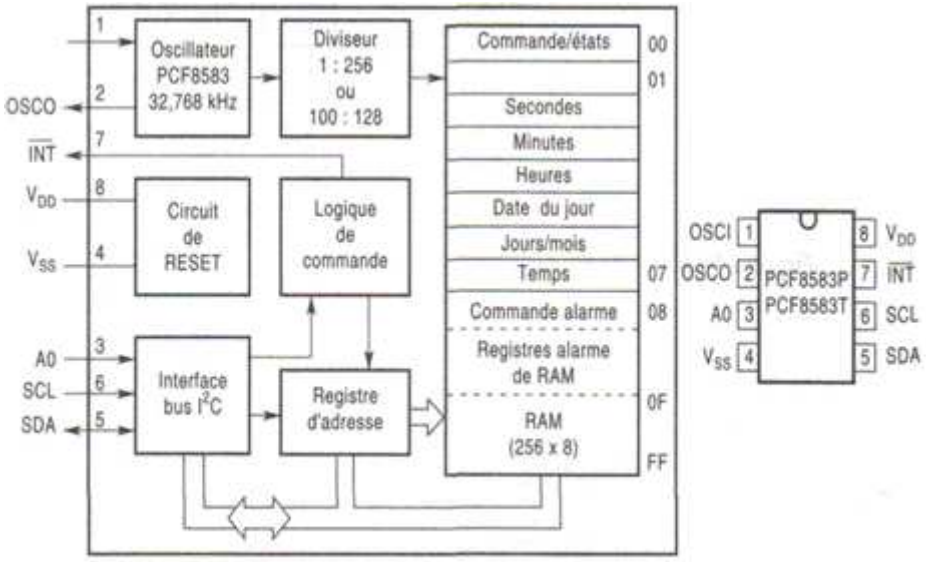


Figure 4.10 : Organisation interne du PCF 8583

(j) Les relais :

Dans notre application nous avons utilisé deux relais dont l'un est principal et l'autre de secours. Ces 2 sorties sont disponibles sur connecteur à vis.

Les relais sont pilotés via un transistor de commande, une diode de roue libre est montée aux bornes du relais pour la protection du transistor.

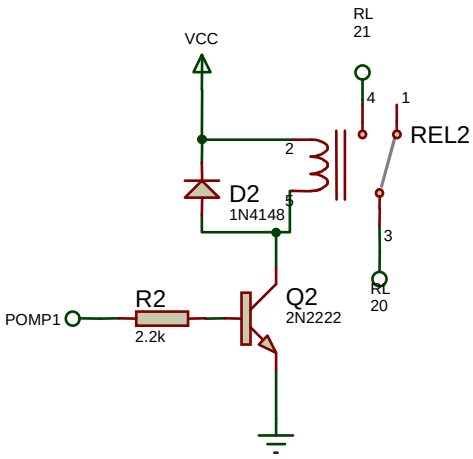


Figure 4.11 : Montage du relais

(k) Le haut-parleur :

Un petit haut-parleur est relié à la broche I/O 38 et sera très appréciable pour agrémenter les programmes de confirmations sonores.

De plus une procédure en µpascale est disponible, rendant la tache de programmation des signaux d'avertissement sonore très facile. Ce qui donnera de plus vie à ce montage.

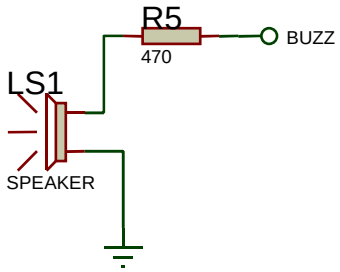


Figure 4.12 : Montage du haut-parleur

(l) Les trois touches de commande :

La carte dispose de trois boutons poussoirs ($SW_{0,}$ $_{SW1}$ et SW_2)qui sont reliés à trois entrées numériques du microcontrôleur (RE0/AN5 ,RE1/AN6et RE2/AN7) en les reliant respectivement aux résistances (R10 ,R9 et R8) de rappel à la source Vcc de moyenne valeurs (2.2 KΩ) pour ne pas consommer plus de l'énergie .

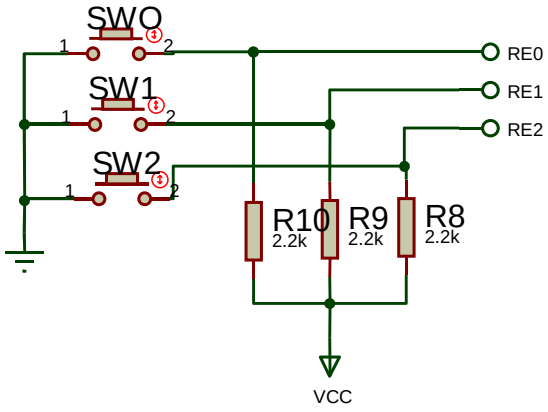


Figure 4.13 : Montage des boutons poussoirs

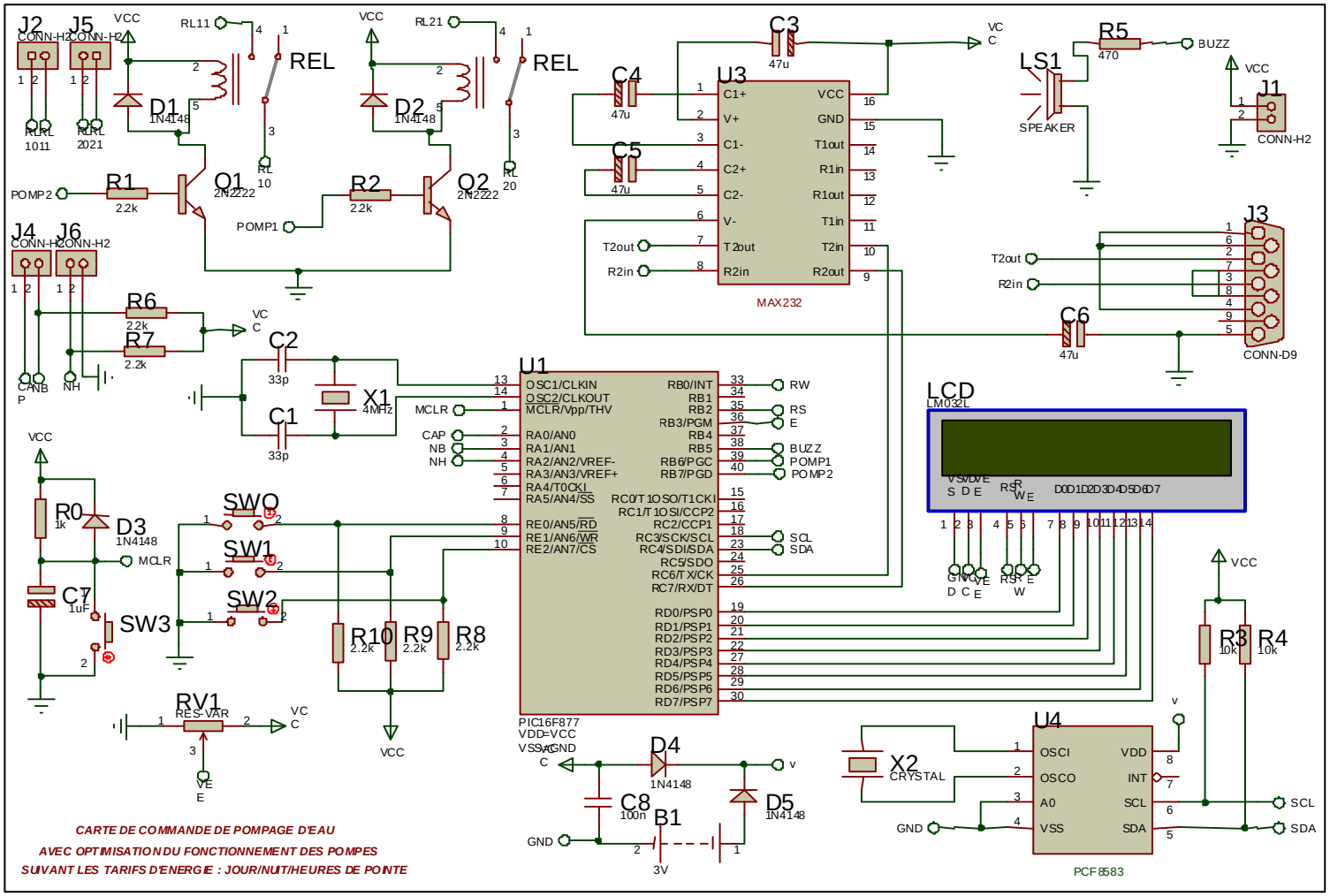


Figure 4.14 : Schéma de la carte de commande

(m) Alimentation :

On désire réaliser une alimentation stabilisée pouvant débiter un courant I_{MAX} et imposer une tension U_{MAX}. On utilise pour cela le montage proposé sur la figure 3.11

Celui-ci est composé :

- d'un transformateur de rapport *M*.
- d'un pont de diodes où chacune d'entre elles possède une tension de seuil notée V_D.
- d'une capacité *C*.
- d'un régulateur intégré *CI*.

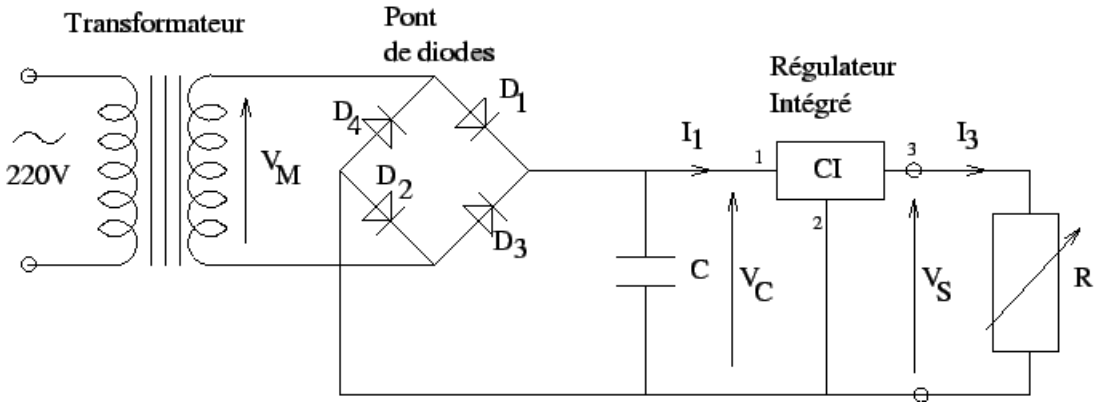


Figure 4.15 : Montage de l'alimentation stabilisée

Pour comprendre la façon de choisir ces différents composants, il faut d'abord s'intéresser au fonctionnement du montage complet.

4.3.2 Fonctionnement du montage.

Le transformateur génère une tension sinusoïdale d'amplitude crête V_M et de fréquence *f=50Hz* (fréquence du secteur). Cette tension est redressée sur ces deux alternances par le pont de diodes. L'association pont de diodes -capacité forme un détecteur de crête. En l'absence de charge R, la tension aux bornes de la capacité est continue et égale à V_M-*2Vd*.

Le régulateur est un circuit intégré (*CI*) générant une tension constante entre ses broches 2 et 3 lorsque la tension entre ses bornes 1 et 2 est supérieure à une tension de seuil notée V_T. De plus, le courant sortant de la broche 2 est négligeable. Le courant débité par le circuit provient donc de son entrée 1 soit : *I1=I3*. En charge, le courant débité par le *CI* va venir

décharger la capacité C. Le courant de décharge est *I3=Vs/R* et il reste constant tant que la tension *Vc* aux bornes du condensateur reste supérieure à la tension de seuil V_T du régulateur.

Pour un fonctionnement normal du montage, il faut que quelle que soit la charge, la tension aux bornes de la capacité soit supérieure à la tension de seuil du régulateur.

(1) Choix des différents éléments (ANNEXE 1)

On néglige dans cette partie, la résistance de sortie du transformateur et le courant inverse des diodes du pont.

(a) Choix de la capacité C.

La figure 3.12 indique l'évolution de la tension aux bornes de la capacité *C* quand le montage fonctionne à vide (V_{Cvide}) et en charge ($V_{Ccharge}$) dans le cas où la capacité *C* serait correctement dimensionnée.

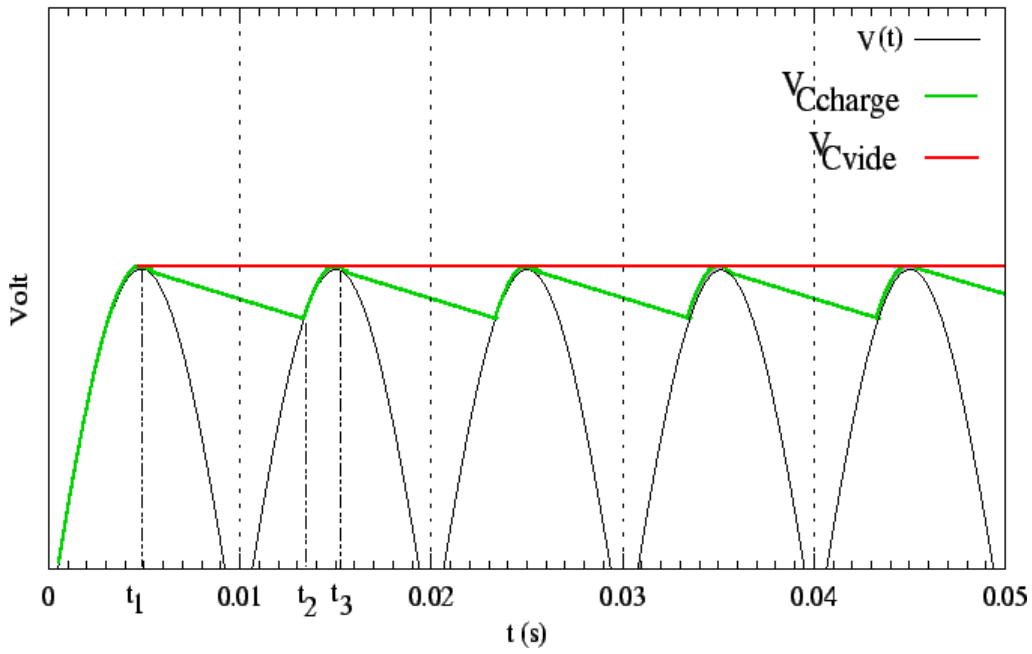


Figure 4.16 : Evolution de la tension aux bornes de la capacité à vide et en charge

On suppose que l'on allume le dispositif à *t=0*. Pour la première période *T* de la tension secteur, trois instants t1, t2 et t3 peuvent être définis :

- t1 est l'instant où la tension *Vc(t)* est maximale.

- t2 est l'instant où la tension *Vc(t)* est égale à la tension *V(t)*.
- t3 est l'instant où la tension *Vc(t)* est de nouveau maximale.

Pour $0<t<t_1$ les diodes D_1et D_2 sont passantes, et la tension *Vc(t)* est égale à la tension *V(t)* : la capacité se charge.

Pour $t_1<t<t_2$, toutes les diodes du pont sont bloquées puisque que la tension *Vc(t) > V(t)*. En fonctionnement à vide, la tension reste constante et égale à V_M. En charge, la tension aux bornes de la capacité est donnée par l'équation (1). (Le calcul détaillé se trouve en **ANNEXE1**)

$$V_C(t) = (V_M - 2.V_D).\left[1-(t-t_1)\frac{V_S}{R.C.(V_M-2.V_D)}\right] \qquad (1)$$

Pour $t_2<t< t_3$, les diodes D_2 et D_4 sont passantes. La capacité se charge, *Vc(t) > V(t)*. Après la valeur maximale de la tension $V\,(t>t_3)$, un cycle identique se reproduit indéfiniment.

La valeur de la capacité *C* est déterminée par la condition :

$$V_{Charge}\,(t_2) \geq V_T$$

$$\sin(2\Pi f t_2) = \frac{V_S(t_2-t_1)}{R.C.V_M} - 1$$

En négligeant la tension de seuil des diodes, on peu écrire à l'instant t_2 :

Il apparaît que l'expression de l'instant t_2 n'est pas soluble analytiquement. Pour poursuivre l'analyse, on confond l'instant t_2 avec l'instant t_3 où la tension *V(t)* est maximale.

L'équation (1) nous donne :

$$V_C(t_3) = V_M.\left[1-(t_3-t_1)\frac{V_S}{R.C.V_M}\right] \geq V_T \quad \Leftrightarrow \quad C \geq \frac{V_S(t_3-t_1)}{R(V_M-V_T)}$$

Pour : $V_S = 5V$ et $I_{MAX} = 1A$ on a : $C_{min} = 1912\mu F$

On prend la valeur normalisée $2200\mu F$

(b) Choix du régulateur.

Ce choix se fait à partir des valeurs de U_{MAX} et I_{MAX} désirées. On trouvera en annexe quelques références de régulateur intégré ainsi que quelques applications.

(c) Choix du transformateur.

Deux points sont à prendre en considération lors du choix du transformateur, la puissance qu'il doit fournir et son rapport M.

(d) Choix du pont de diodes.

A l'instant proche de t_2, les diodes du pont se débloquent et il apparaît alors un pic de courant limité par la résistance de sortie du transformateur. Le calcul de ce pic est complexe. On choisit en général un pont de diodes pouvant débiter un courant égal à 5 fois le courant I_{MAX} . On trouvera en annexe quelques références de ponts de diodes.

(2) Montage final de l'alimentation :

L'alimentation des amplificateurs opérationnels ce fait à l'aide d'une alimentation stabilisée de 5V On se basant sur l'étude précédente nous avons choisi le montage de suivant :

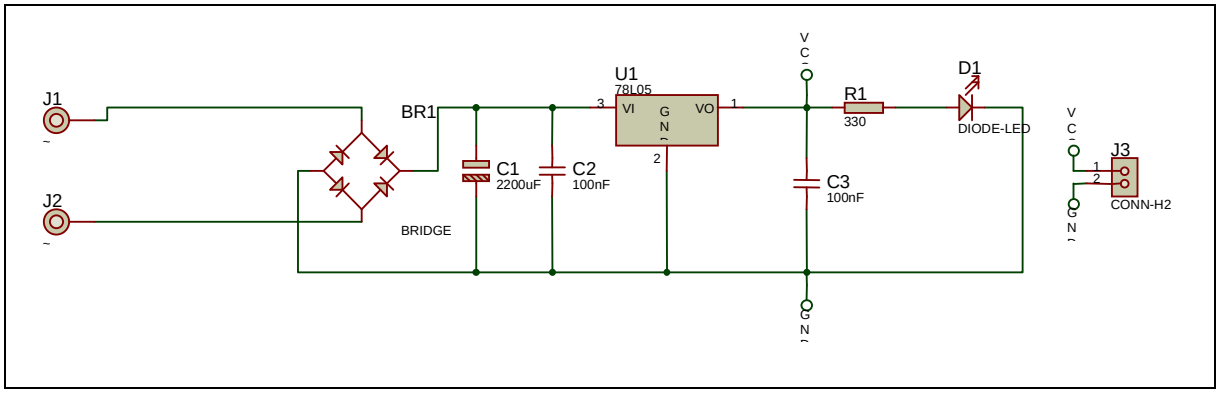


Figure 4.17 : Montage complet de l'alimentation stabilisée

Un transformateur nous fourni une tension de 5v qui est redressée par le pont de diodes et filtrée par les condensateurs de C_1 et C_2. Cette tension est ensuite régulée à $5v$ par $U1$ et $U2$, les capacités C_2 et C_3 servent à filtrer les bruits, leurs valeurs étant données par le constructeur du régulateur

On obtient à la sortie du régulateur une tension de *5 v* assez stable pour ne pas perturber le circuit de commande.

4.3.3 Mise en œuvre d'un automate programmable industriel (simulation)

Pour la simulation, on a utilisé :

- Le logiciel PROTEUS Version 7.6
- Un condensateur pour le réservoir
- 2 LED pour les 2 pompes
- Un interrupteur à 2 positions pour la consommation
- Affichage LCD pour un effet visuel
- Un haut parleur pour la signalisation du réservoir plein

Préliminaire :

Après avoir chargé le fichier pompage.cof dans la carte de commande et le lancement de la simulation, voici les résultats obtenus :

Etape 1 : remplissage du réservoir

Pompe 2 désactivée

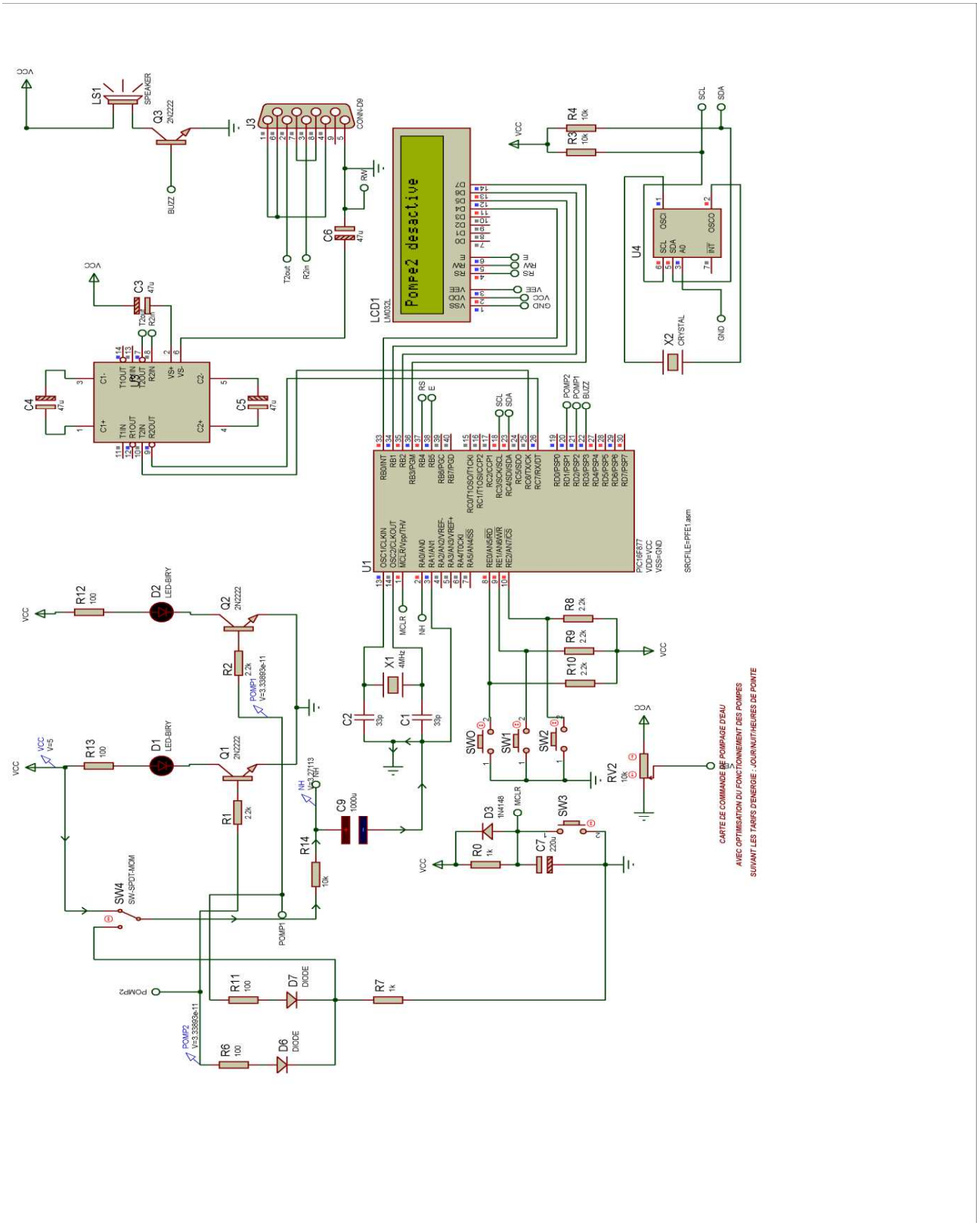


Figure 4.18 : Pompe 2 désactivée

Pompe 1 activée

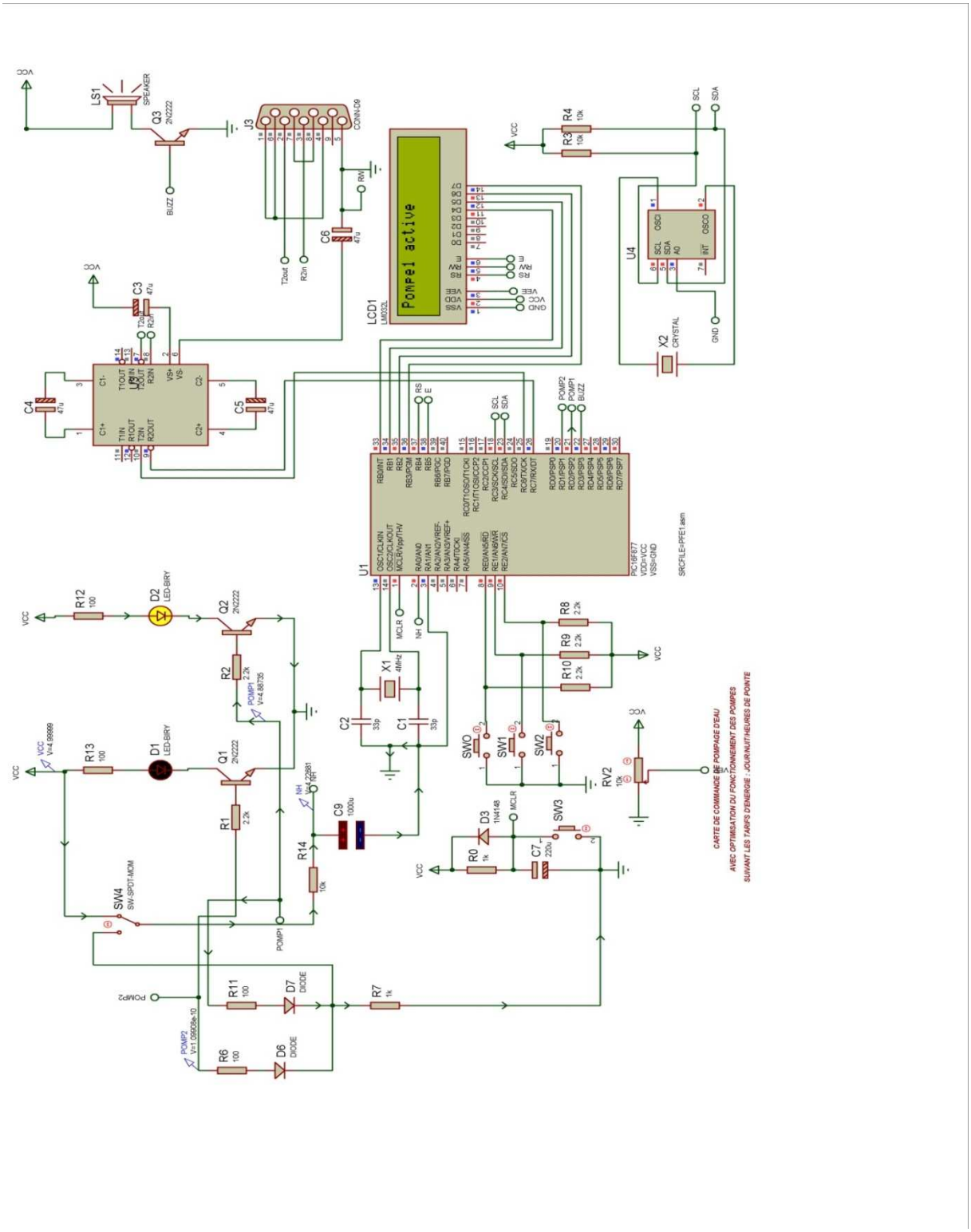


Figure 4.19 : Pompe 1 activée

Pompe 1 désactivée

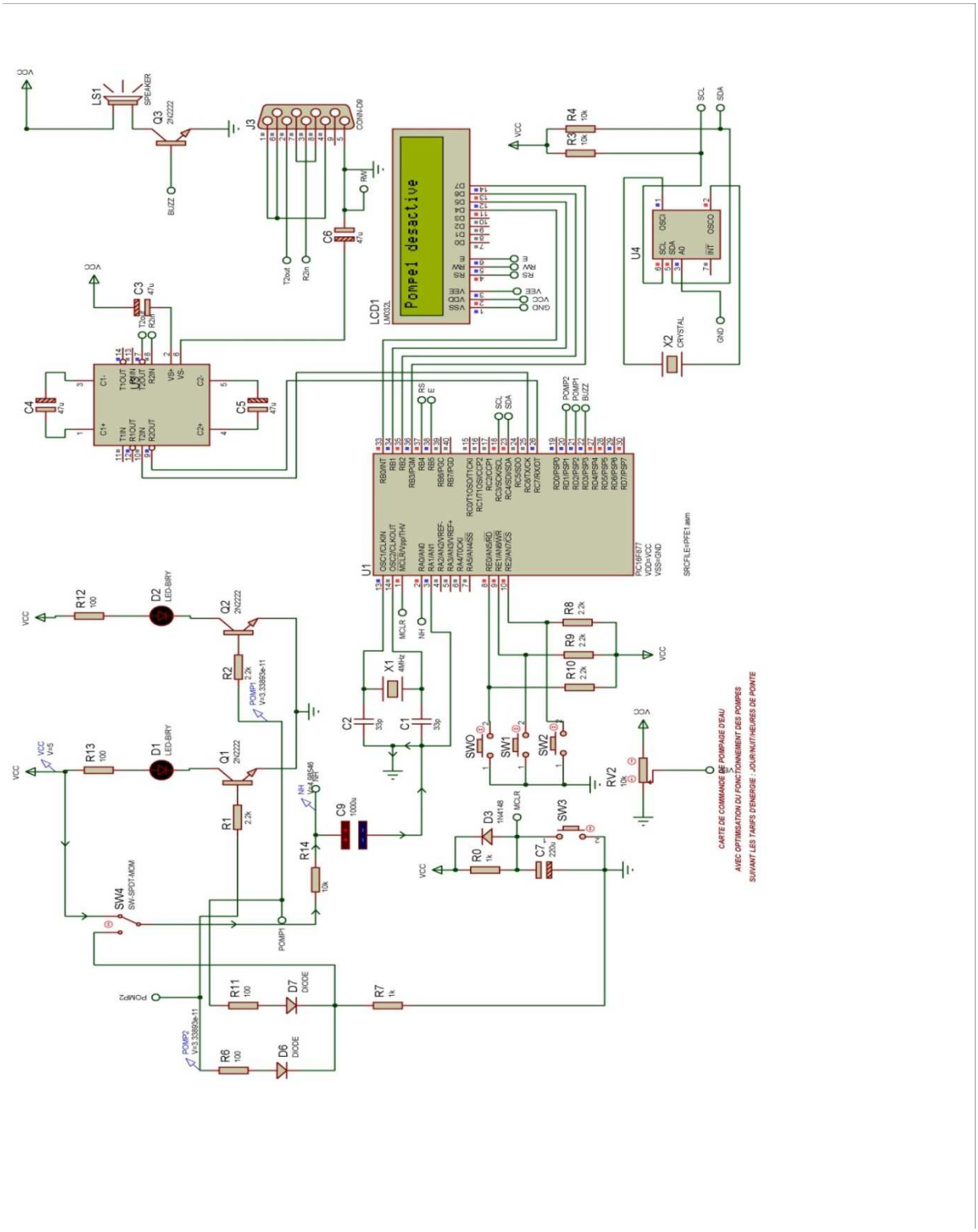


Figure 4.20 : Pompe 1 désactivée

Réservoir plein

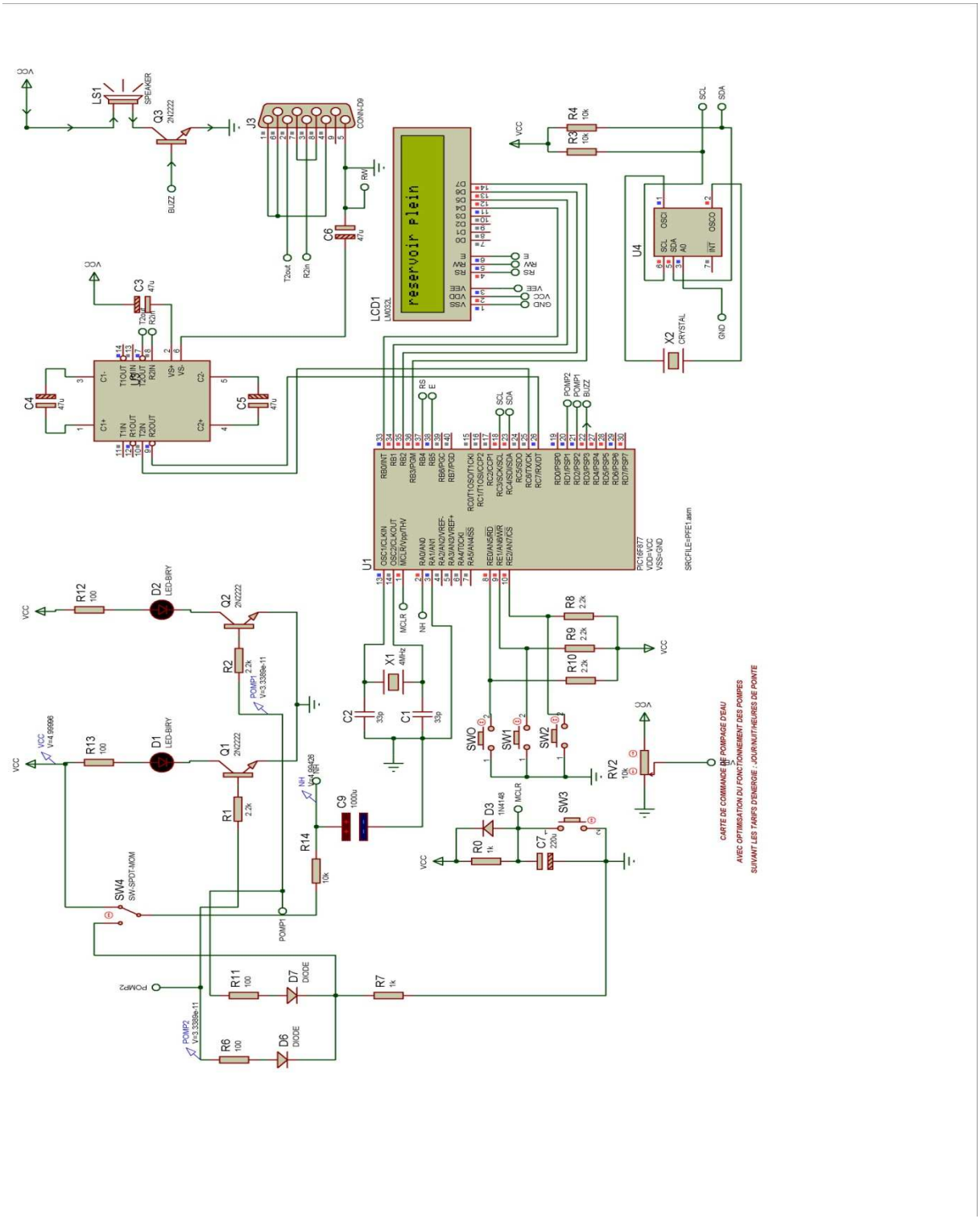


Figure 4.21 : Réservoir plein

Etape 2 : consommation

Pour la consommation, on maintient enfoncer l’interrupteur à 2 position jusqu’à l’activation des 2 pompes P1, P2

Pompes P1, P2 activées

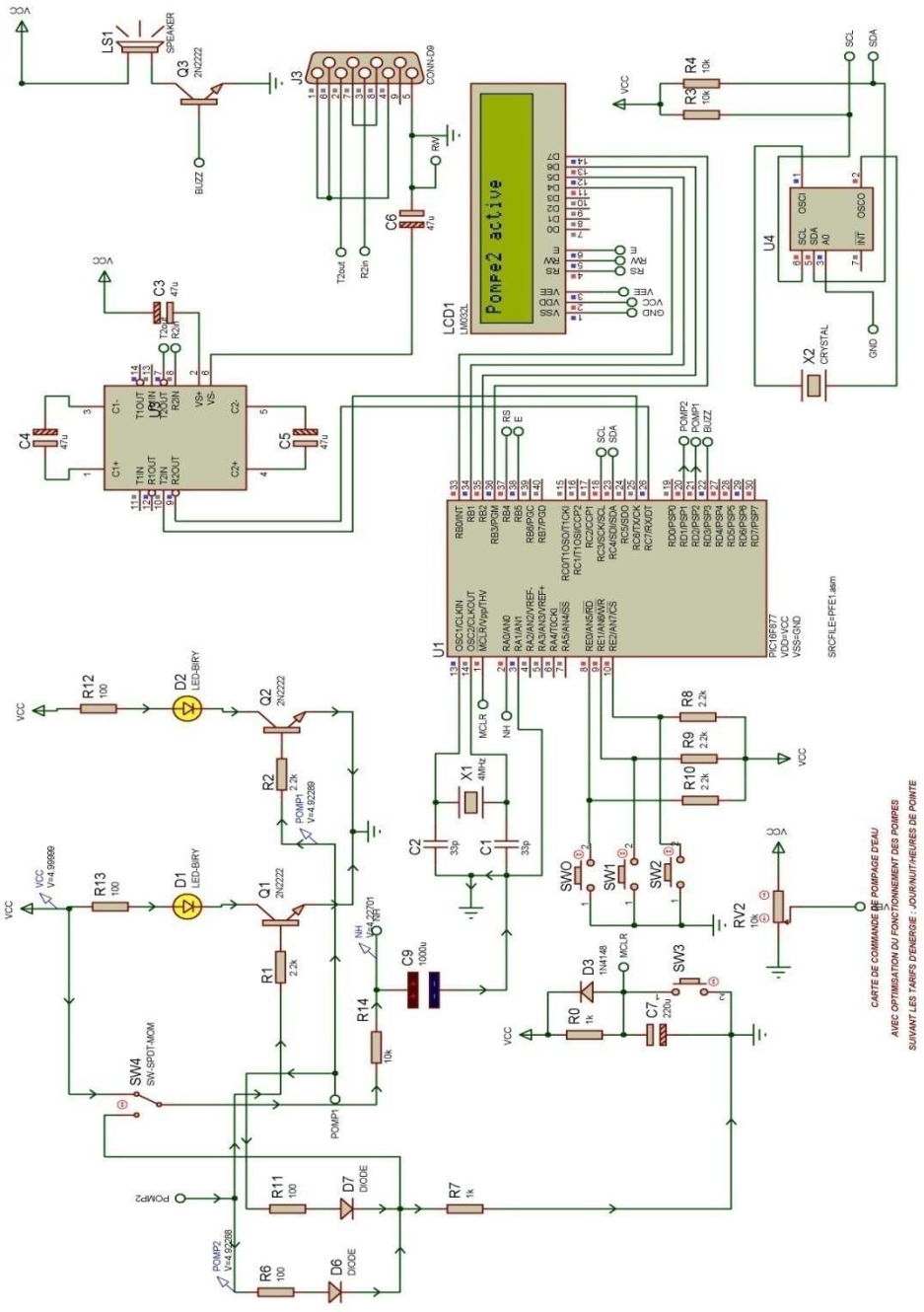


Figure 4.22 : Pompes P1, P2 activées

Etape 3 : Les boutons poussoirs

Bouton SW0 : Pour activer la pompe P1

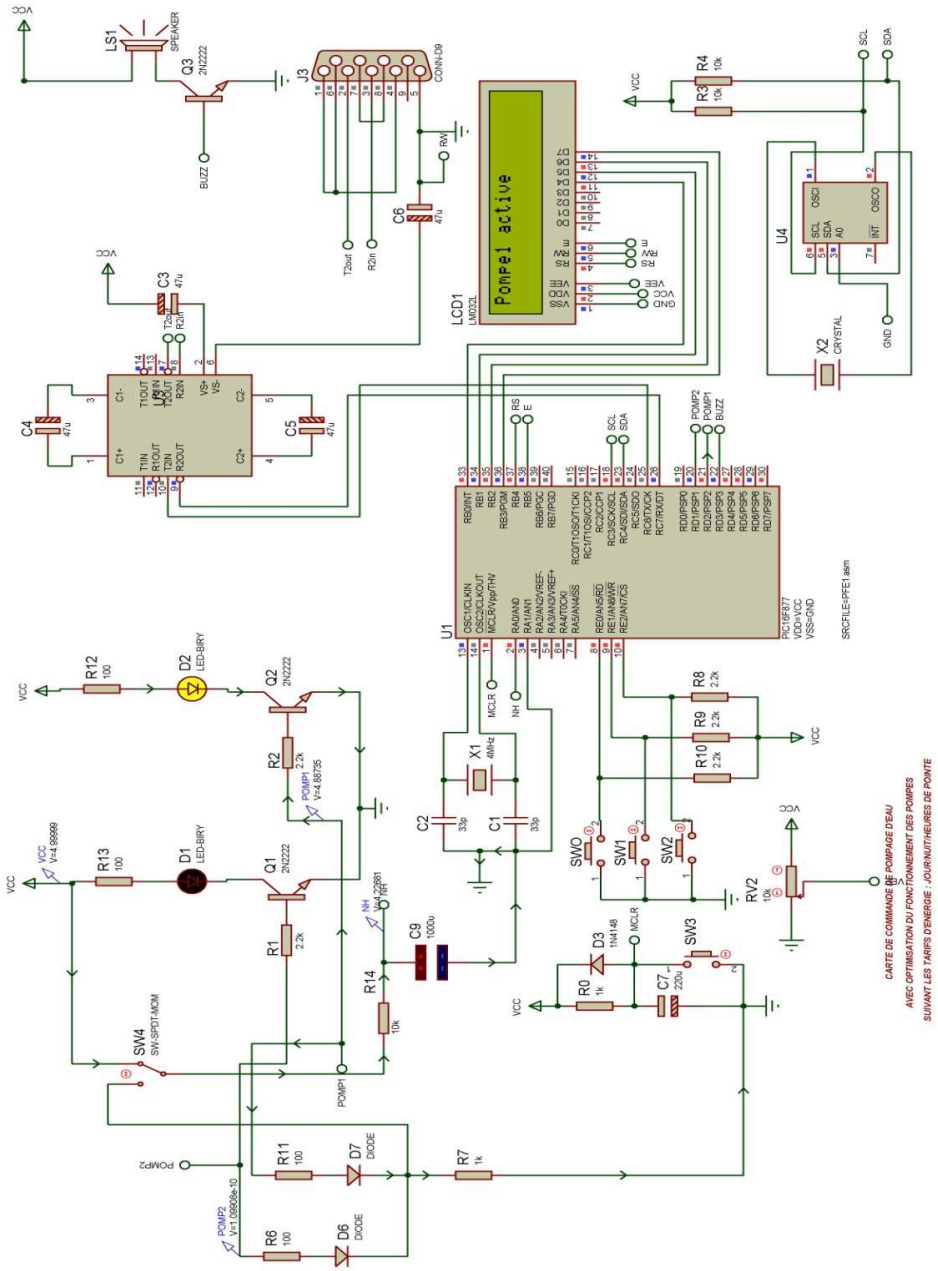


Figure 4.23 : Pompe 1 activée par le bouton SW0

Bouton SW1 : Pour activer la pompe P2

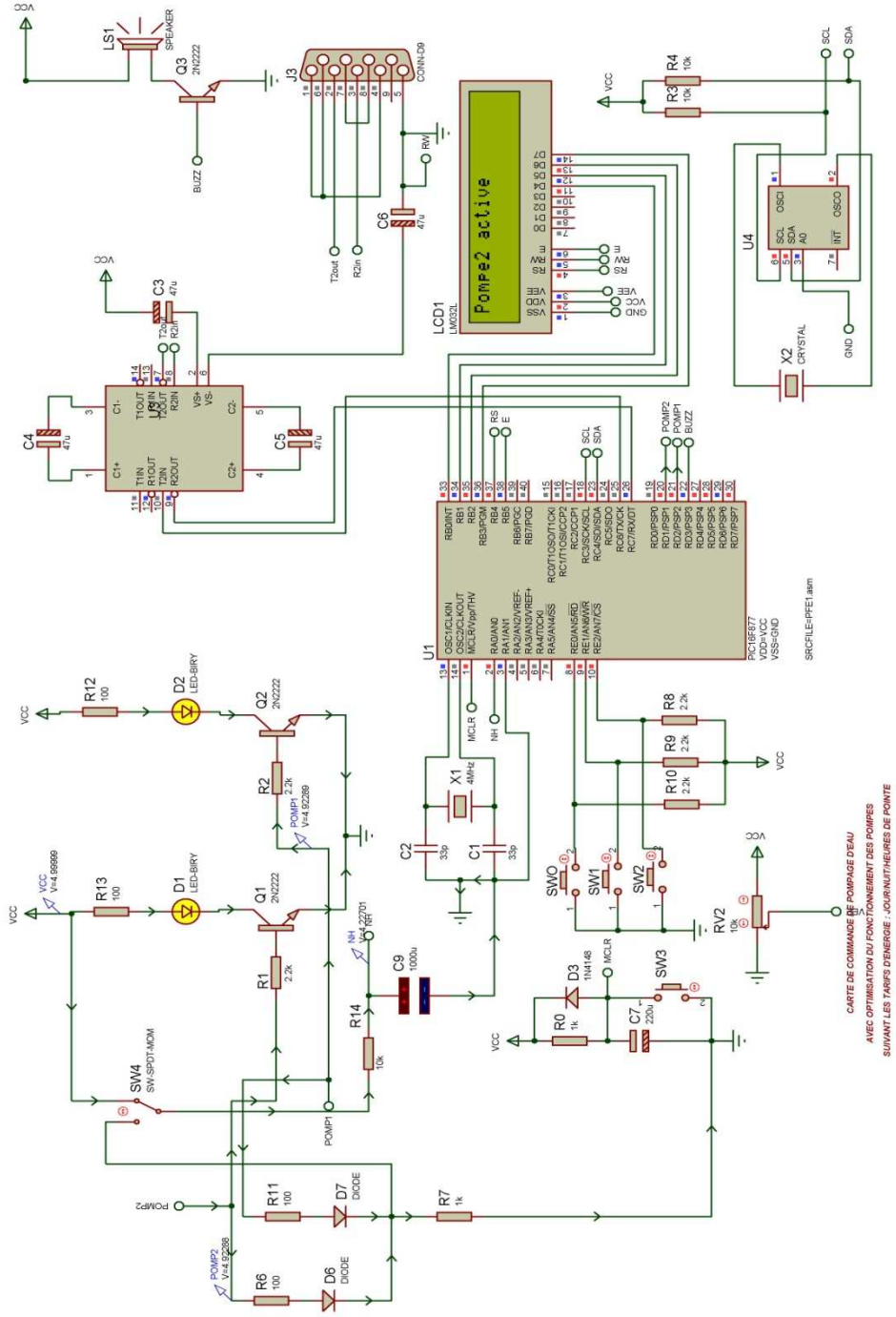


Figure 4.24 : Pompe 2 activée par le bouton SW1

Bouton SW2 :

Pour effacer l'écran et désactiver les pompes

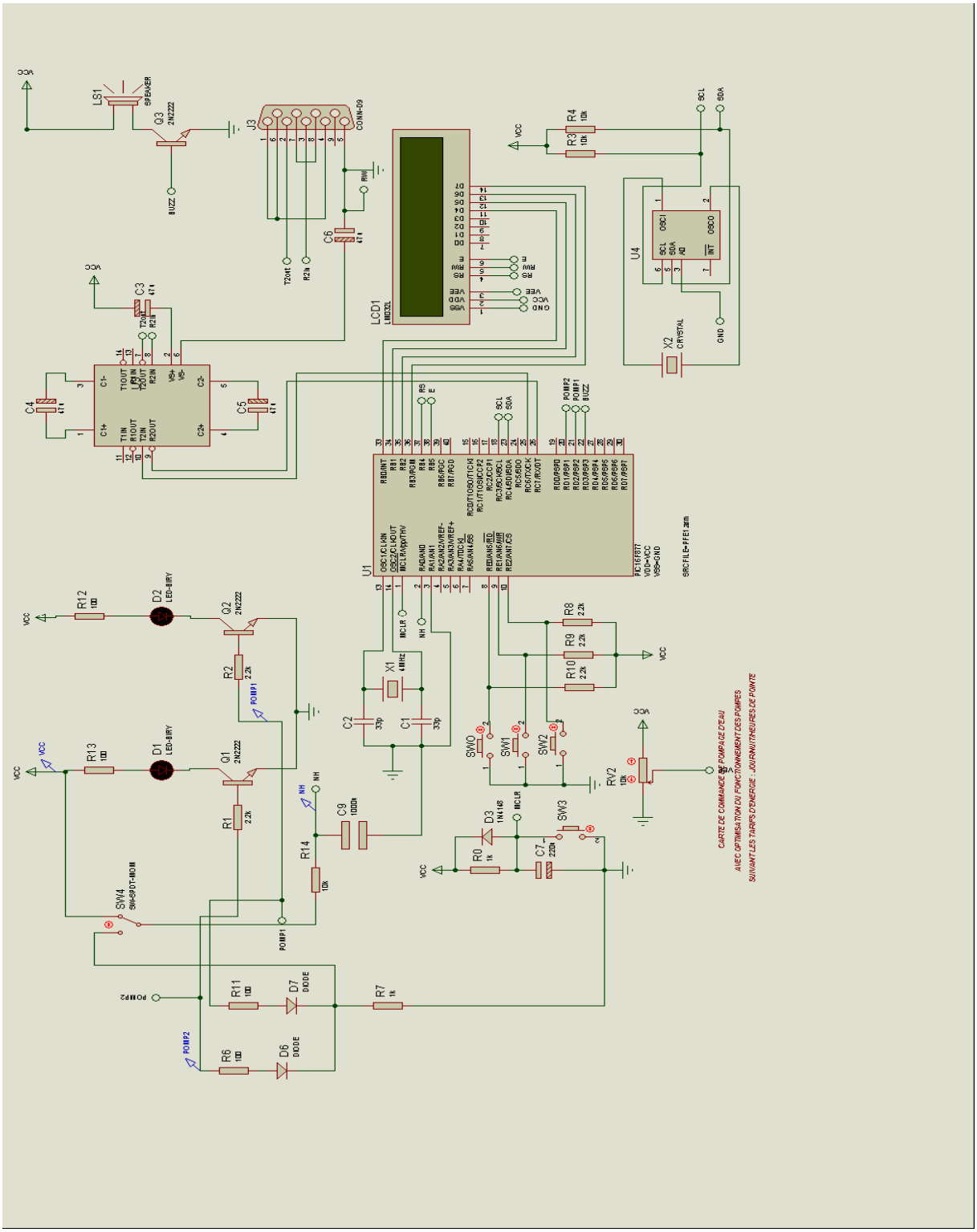


Figure 4.25 : effacement de l'écran et désactivation des pompes par le bouton SW2

Bouton SW3 :

Pour réinitialiser la carte de commande

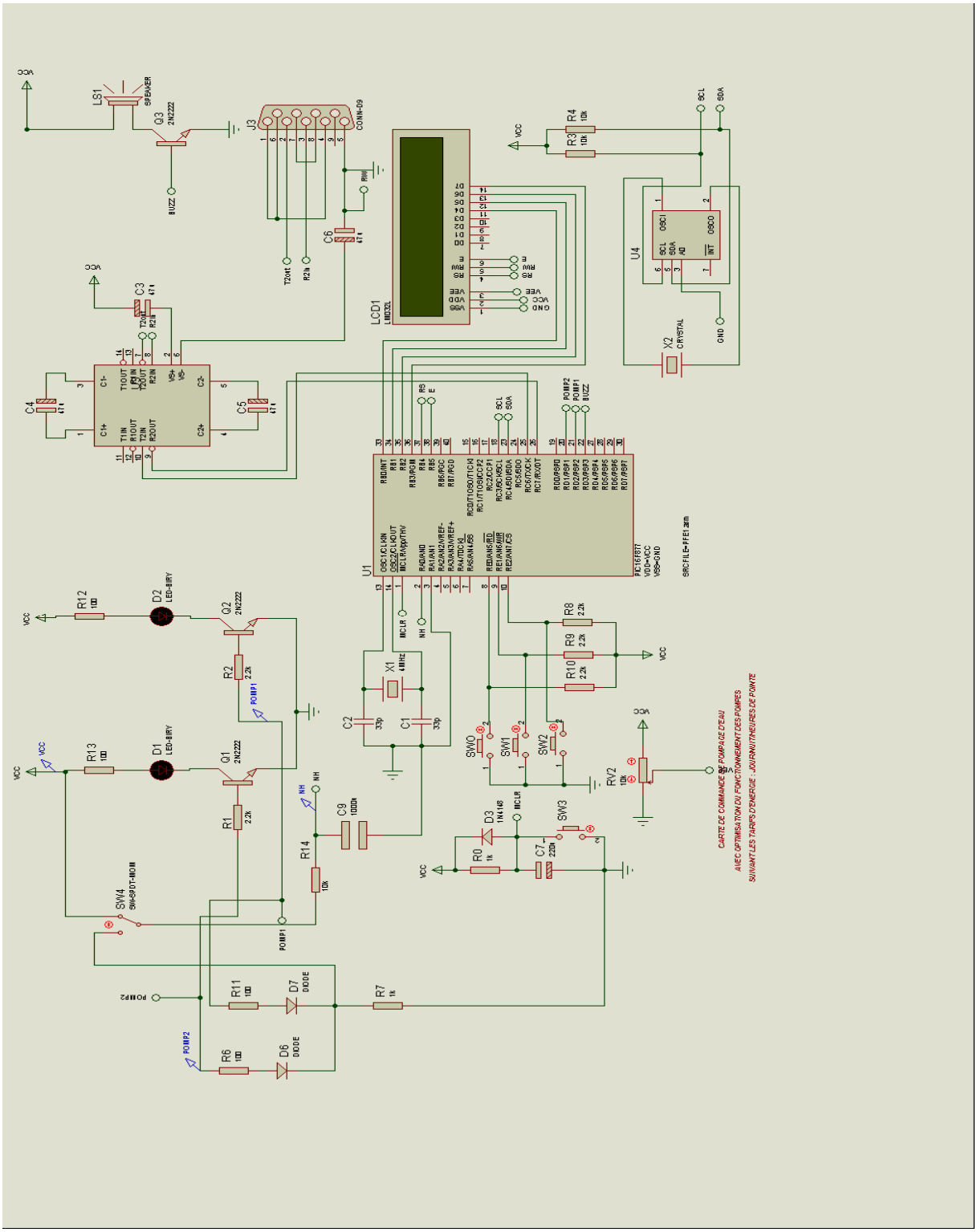


Figure 4.26 : Réinitialisation de la carte de commande par le bouton SW

4.4 Conclusion

Notre stratégie dans la partie conception est basée sur La simplicité des montages et le moindre coût, pour assurer le bon fonctionnement de notre carte. Pour réduire le nombre de connexions et de circuits nous avons utilisé trois boutons poussoirs au lieu d'un clavier matriciel.

Nous avons également pensé utiliser la mémoire EEPROM du PIC ainsi que les 240 octets disponibles dans le circuit PCF8583 au lieu d'utiliser un circuit mémoire externe.

C'est vrai que nous sommes limités par la taille de la mémoire mais, sachant que l'évolution de la consommation de l'eau varie d'une façon lente dans le temps, la taille mémoire disponible est largement suffisante pour assurer le bon fonctionnement de notre système.

Partant d'une étude générale des stations de pompage tel que la description de ces équipements et leur principe de fonctionnement, on a pu montrer l'importance de l'optimisation de fonctionnement des ouvrages d'extraction d'eau, puisque le prix de l'énergie électrique est variable suivant la période du jour, notre système va permettre à la société d'économiser une bonne partie du capital destiné au pompage d'eau. Nous avons également montré, par la suite, que la consommation d'eau varie durant la journée.

Ceci a été fait par une étude complète comportant des courbes de modulation sur la consommation d'eau en fonction de temps.

Ensuite nous avons cherché à faire, dans la partie conception matérielle, une présentation des différents composants utilisés dans notre système tel que l'afficheur LCD, l'horloge de temps, MAX 232…).

Pour finir nous avons réalisé une simulation de notre projet afin de pouvoir le tester et déterminer ses performances et ses limites, nous avons également fait la conception du circuit d'alimentation stabilisée que nous avons utilisé pour notre montage.

L'intégration de l'automate programmable renforce le degré de fiabilité de l'équipement et offre une très grande adaptabilité face aux évolutions de l'environnement.

Aujourd'hui, l'automate programmable n'est plus seulement une machine séquentielle mais il est beaucoup plus considéré comme un calculateur de processus grâce aux énormes progrès quant à la structure de base, la qualité et la diversité des outils proposés.

Son intégration sur Fieldbus (Profibus, WorldFip), sur Ethernet (Standard TCP-IP), accroît ses possibilités et constitue un passage obligé pour augmenter la performance des processus. Comme la commande à distance et l'optimisation d'un réseau de station de pompage.

ANNEXE 1 : Calcul pour le choix des condensateurs

$$V_C(t_1) = V_M - 2.V_D$$

$$V_C(t) = \frac{1}{C}\int_{t_1}^{t} i(t).dt + V_C(t_1)$$

Or :

Le condensateur se décharge $\Leftrightarrow i(t) = -I_1 = -I_2 = -\frac{V_S}{R}$

$$\Leftrightarrow V_C(t) = -\frac{V_S}{RC}\int_{t_1}^{t} dt + V_M - 2.V_D$$

$$\Leftrightarrow V_C(t) = -\frac{V_S}{RC}(t - t_1) + V_M - 2.V_D$$

$$\Leftrightarrow V_C(t) = (V_M - 2.V_D).\left[1 - (t - t_1).\frac{V_S}{RC.(V_M - 2.V_D)}\right]$$

On peut écrire à l'instant t_2 : $V_C(t_2) = (V_M - 2.V_D).\left[1 - (t_2 - t_1).\frac{V_S}{RC.(V_M - 2.V_D)}\right] = -V_M \sin(2\Pi f t_2) - 2V_D$

En négligeant la tension de seuil des diodes, on a :

$$\sin(2\Pi f t_2) = \frac{V_S(t_2 - t_1)}{RC.(V_M - 2.V_D)} - 1$$

ANNEXE 2 : Régulateur de tension positive à 3 broches

La série de régulateurs à trois broches LM78XX est disponible dans de nombreuses valeurs de tensions de sortie fixes et est très utile dans nombre d'applications. Bien que conçus pour fournir des tensions de sortie fixes, ces circuits peuvent également délivrer des tensions et courants réglables à l'aide de quelques composants extérieurs.

LM78XXC
La série LM78XX est disponible en boîtier aluminium TO-3 qui peut délivrer jusqu'à 1 A si on utilise un refroidisseur approprié. Ce boîtier possède une limitation en courant interne pour ne pas dépasser les limites de sécurité en courant de pointe. Une plage de sécurité est prévue pour le transistor de sortie permettant de limiter la puissance interne dissipée. Si celle-ci devient trop importante pour le refroidisseur utilisé, le circuit de disjonction thermique est activé pour éviter une surchauffe du circuit intégré.
Des efforts considérables ont été faits pour rendre la série LM78XX facile à mettre en œuvre et réduire au minimum le nombre de composants extérieurs. Il n'est pas nécessaire de découpler la sortie bien que ceci améliore la réponse aux transitoires. Le découplage de l'entrée n'est nécessaire que dans le cas où le régulateur soit éloigné du condensateur de filtrage de l'alimentation.

Caractéristiques :
- Courant de sortie d'un moins 1 A
- Protection thermique interne contre les surcharges
- Aucun composant externe nécessaire
- Plage de sécurité pour le transistor de sortie
- Limitation interne du courant de court-circuit
- Disponible en boîtier aluminium TO-3

Type	U_{out} (V)	I_{out} (A) 78XXC	78LXX	78MXX	U_{in} (V) min.	max.
7805	5	1	0,1	0,5	7,5	20
7806	6	1	0,1	0,5	8,6	21
7808	8	1	0,1	0,5	10,6	23
7810	10	1	0,1	0,5	12,7	25
7812	12	1	0,1	0,5	14,8	27
7815	15	1	0,1	0,5	18	30
7818	18	1	0,1	0,5	21	33
7824	24	1	0,1	0,5	27,3	38

ANNEXE 3 : PIC 16F877

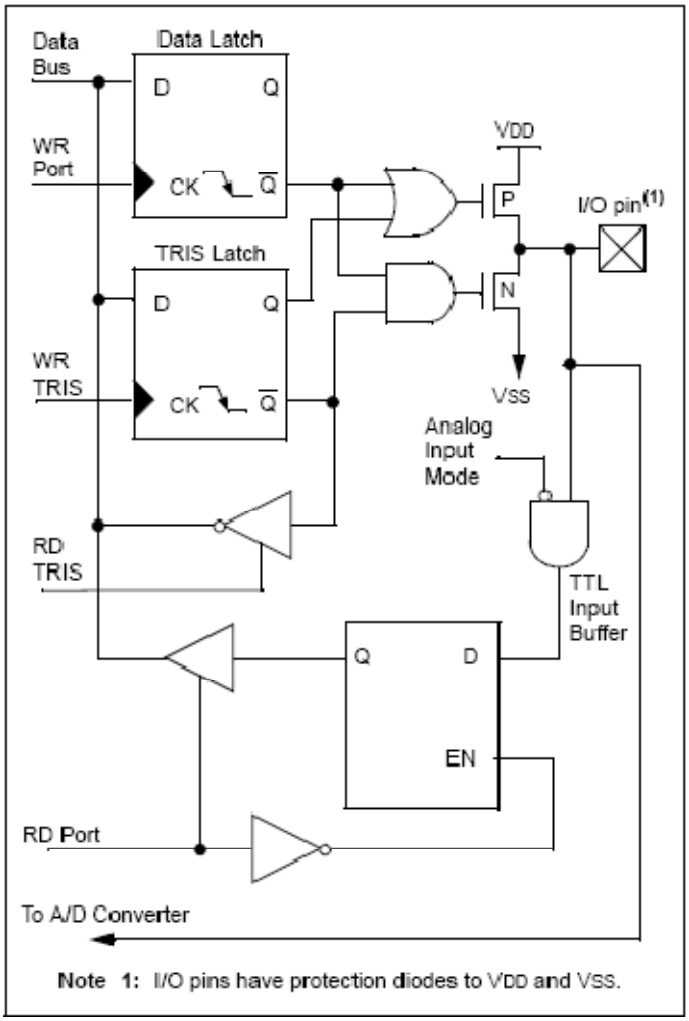


Figure A.1 : schéma des ports RA3 :RA0 et RA5

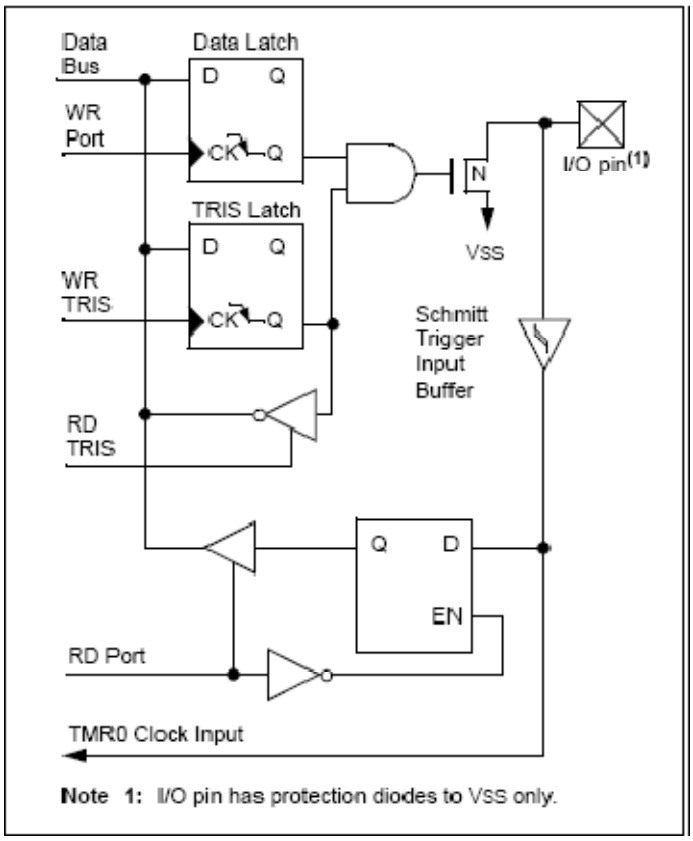


Figure A.2 : schéma des ports RA3 :RA0 et RA5

Exemple d'initialisation du port A :

```
BCF     STATUS, RP0   ;
BCF     STATUS, RP1   ; Bank0
CLRF    PORTA         ; Initialize PORTA by
                      ; clearing output
                      ; data latches
BSF     STATUS, RP0   ; Select Bank 1
MOVLW   0x06          ; Configure all pins
MOVWF   ADCON1        ; as digital inputs
MOVLW   0xCF          ; Value used to
                      ; initialize data
                      ; direction
MOVWF   TRISA         ; Set RA<3:0> as inputs
                      ; RA<5:4> as outputs
                      ; TRISA<7:6>are always
                      ; read as '0'.
```

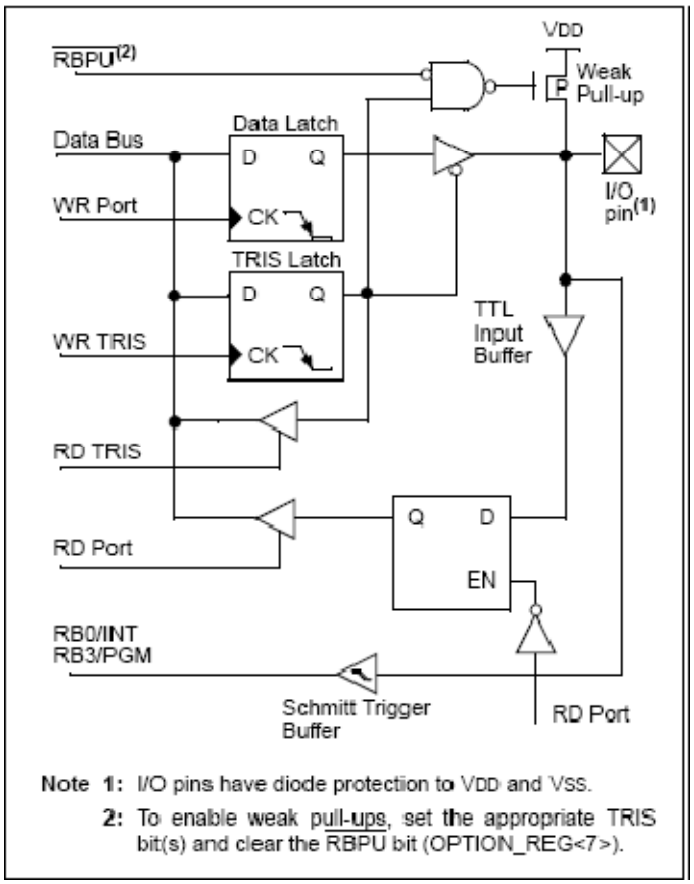
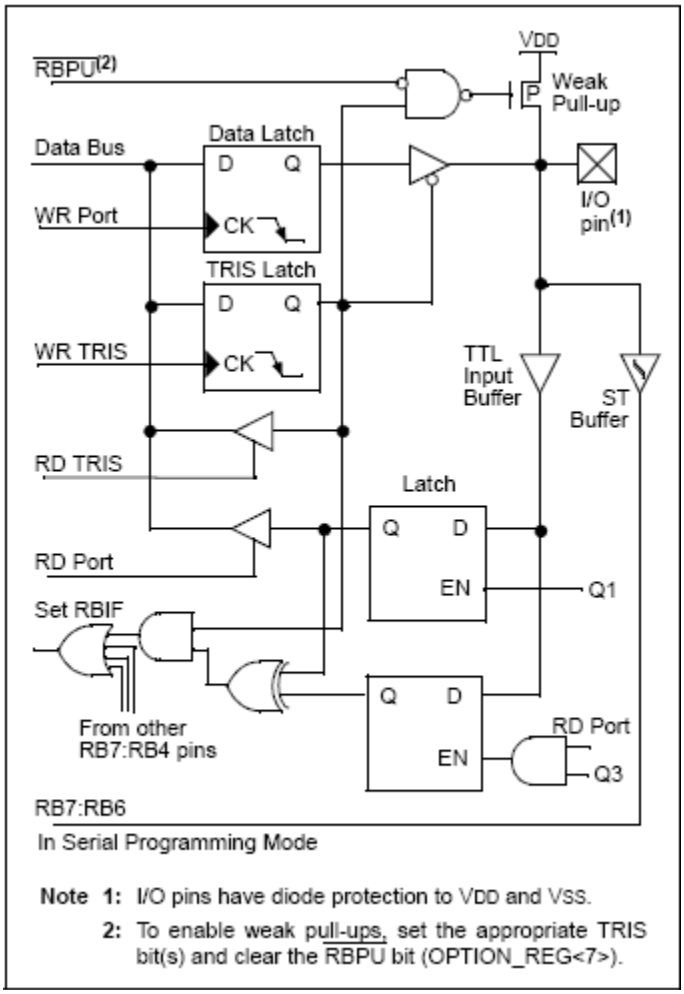


Figure A.3 : schéma des ports RB3 :RB0



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (OPTION_REG<7>).

Figure A.4 : schéma des ports RB7 :RB4

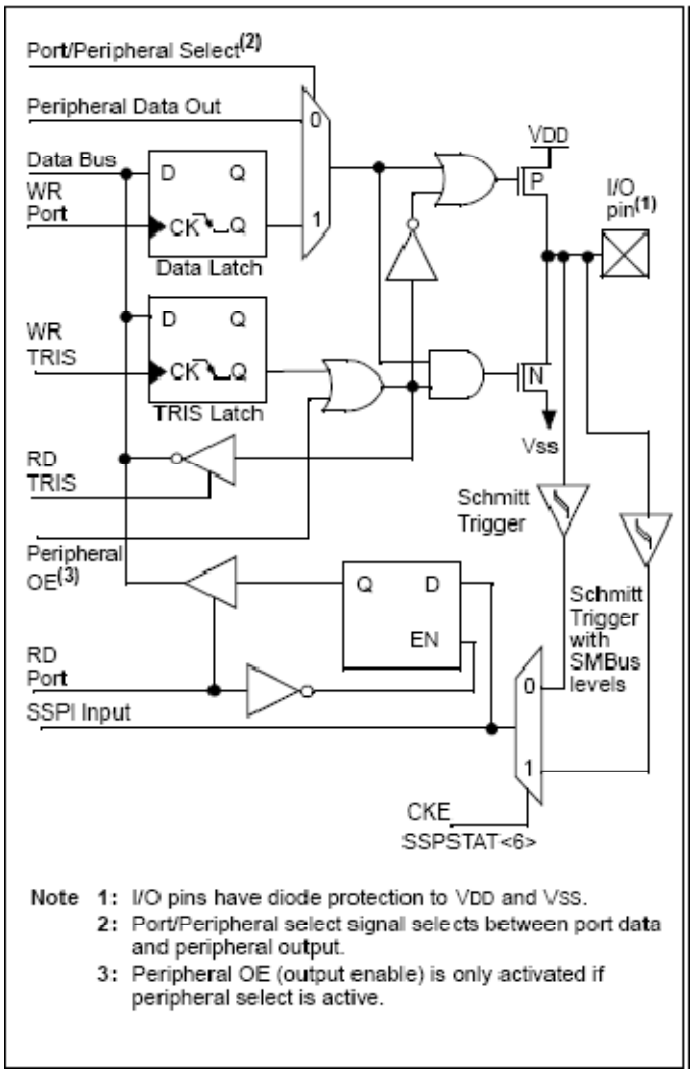


Figure A.5 : schéma des ports RC4 :RC3

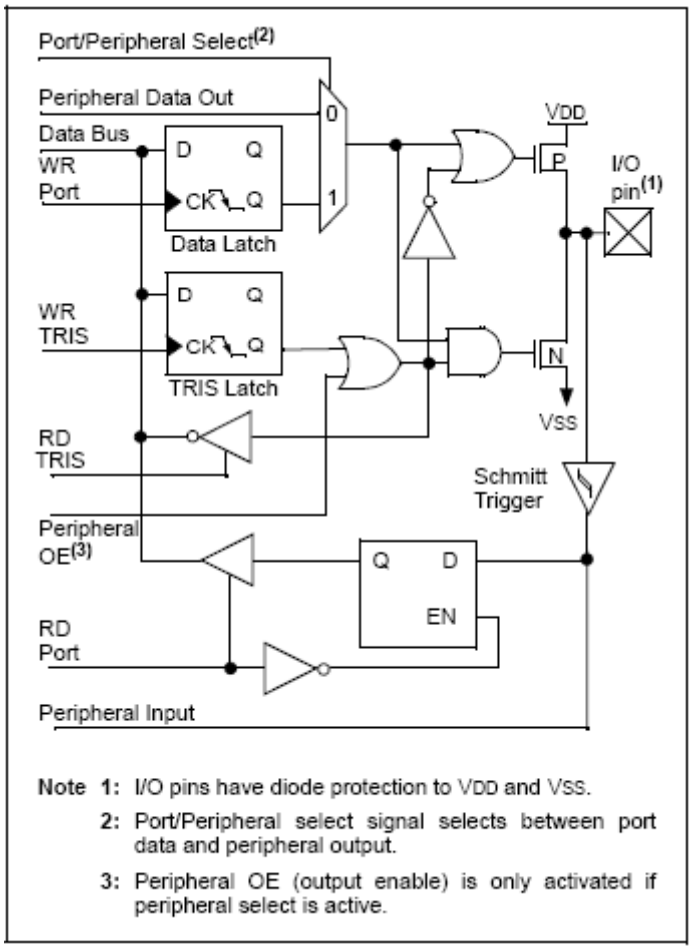


Figure A.6 : schéma des ports RC2 :RC0, RC7 :RC5

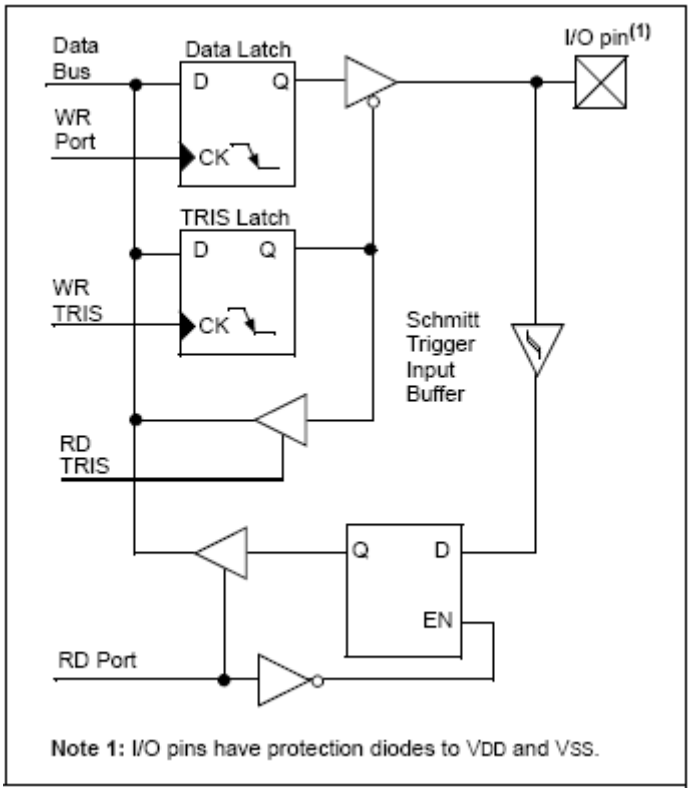


Figure A.7 : schéma des ports D (mode I/O)

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL(1)	Input/output port pin or parallel slave port bit0.
RD1/PSP1	bit1	ST/TTL(1)	Input/output port pin or parallel slave port bit1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or parallel slave port bit2.
RD3/PSP3	bit3	ST/TTL(1)	Input/output port pin or parallel slave port bit3.
RD4/PSP4	bit4	ST/TTL(1)	Input/output port pin or parallel slave port bit4.
RD5/PSP5	bit5	ST/TTL(1)	Input/output port pin or parallel slave port bit5.
RD6/PSP6	bit6	ST/TTL(1)	Input/output port pin or parallel slave port bit6.
RD7/PSP7	bit7	ST/TTL(1)	Input/output port pin or parallel slave port bit7.

Legend: ST = Schmitt Trigger input, TTL = TTL input
Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

Tableau A.2 : Fonction du port D

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

Tableau A.3 : Sommaire de registre associé avec le port D

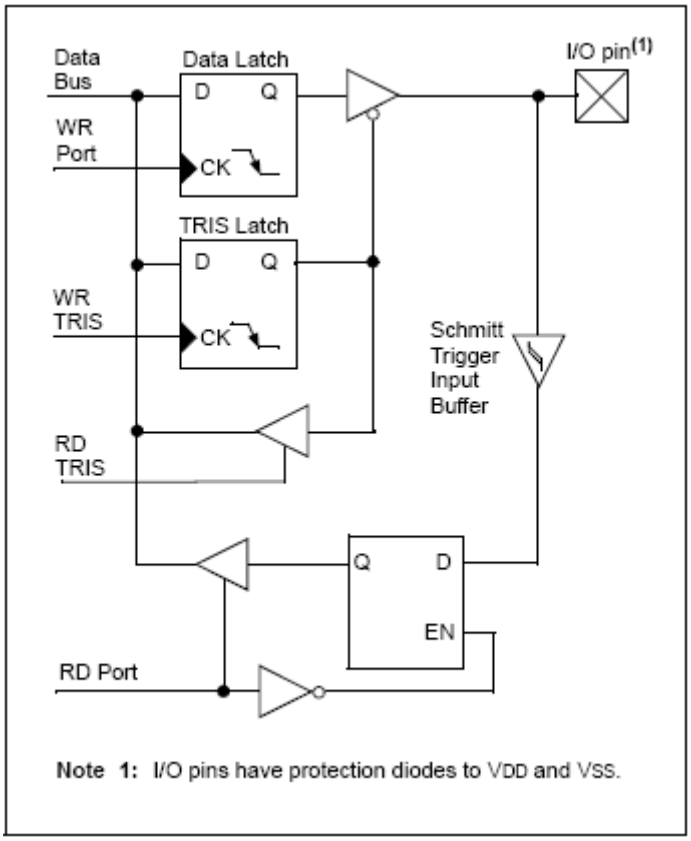


Figure A.8 : Schéma des ports E (mode I/O)

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected)
RE1/$\overline{WR}$/AN6	bit1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected)
RE2/$\overline{CS}$/AN7	bit2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input
Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

Tableau A.4 : Fonction du port E

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

Tableau A.5: Sommaire de registre associé avec le port E

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on $\overline{MCLR}$, WDT
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	0000 00-0
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	--0u 0000
89h(1)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
09h(1)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.
Note 1: These registers/bits are not available on the 28-pin devices.

Tableau A.6 : Registres/Bits associés avec A/D

ANNEXE 4 : PCF 8583

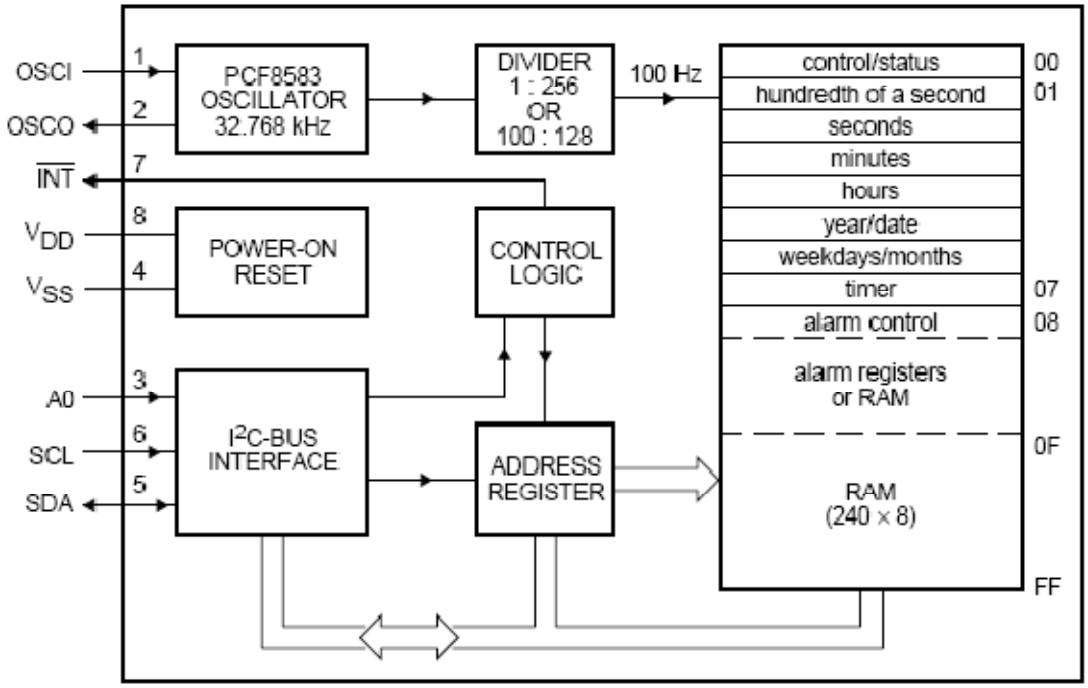


Figure A.9 : Schéma bloc du PCF8583

SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

Tableau A.7 : Brochage

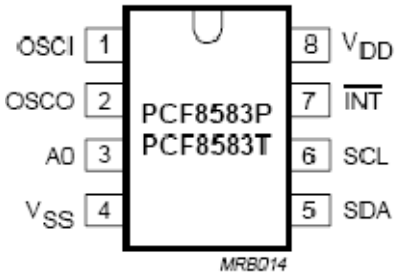


Figure A.10 : Schéma du brochage

Bibliographie

Les livres :

[1] E 301 Circuits Séquentiels
[2] E 302 Analyse et Conception de Circuits Électron.
[3] E 303 Circuits Analogiques
[4] E 310 Technique de l'EN Discrète - Quadripôles
[5] E 311 Technologie des Circuits Numériques
[6] E 332 Microprocesseurs I
[7] E 331 Programmation en C
[8] E 403 Circuits Mémoires
[9] E 412 Programmation en Langage Machine
[10] E 435 Mesure des Grandeurs Physiques et Capteurs
[11] E533 Gestion de la Qualité
[12] E 501 Microprocesseurs II
[13] Rapport de stage d'initiation février 2004/2005.
[14] Projet de fin d'étude : Maintenance d'une station de pompage, 13 janvier 2005.
[15] ALAIN REBOUX, « s'initier à la programmation des pics », Dunod.
[16] CHRISTIAN TRAVENIER, « Description et mise en œuvre », Dunod.
[17] BERNARD ODANT « Microcontrôleur description et mise en œuvre », Dunod.
[18] PIERRE MAYE, « Connaître les composants électronique », Dunod.
[19] Alain GONZAGA, « Les automates programmables industriels »
[20] Rapport de stage de DEA élaboré par : Mohamed El Mongi BEN GAID
Encadré par : Antoine PETIT et Philippe SCHNOEBELEN, « Modélisation et vérification des aspects temporisés des langages pour automates programmables industriels »
[21] Michel Bertrand, « Automates programmables industriels », 9 mars 2009
[22] Présenté par C. VRIGNON et M. THENAISIE, « L'automatisation », 17 octobre 2005
[23] M. Ben Gaid, « Modélisation et vérification des aspects temporisés des langages pour automates programmables industriels ». Mémoire de DEA. Septembre 2003.
[24] A. Mader. 'A Classification of PLC Models and Applications'
[25] J.M. Roussel, J.J. Lesage, « Définition d'un cadre formel Pour l'expression et la vérification de propriétés d'un modèle grafcet » 28 Mars 1996
[26] O. Rossi, « Validation formelle de programmes Ladder Diagram pour Automates Programmables Industriels », Thèse de Doctorat en EEA, ENS de Cachan, juin 2003
[27] S. Yovine, « Méthodes et outils pour la vérification symbolique de systèmes temporises » Master thesis, Institut National Polytechnique de Grenoble, France, May 1993.
[28] Philippe LE BRUN, « Automates programmables industriels », Décembre 1999

Les sites :

[29] http://www.microchip.com
[30] http://matthieu.benoit.free.fr/pic 16.htm
[31] http://www.courselec.free.fr
[32] http://www.lucas-nuelle.com
[33] http://www.wago.com
[34] http://fr.wikipedia.org
[35] http://www.electrome.fr
[36] http://www.erm-automatismes.com

Auteurs : **ANDRIANAIVO Solofoniaina Jean Henri**
RAKOTOMAVO Rijaniaina Michaël

Titre : Optimisation du fonctionnement d'une station de pompage

Nombre de pages : 105
Nombre de figures : 76
Nombre de tableaux : 26

Résumé :

Un automate programmable industriel est très répandu de nos jours et est adaptable sur plusieurs applications, ceci du point de vue traitement de l'information, des composants nécessaires, ainsi que du langage utilisé.

Le but de notre étude est la conception de la carte de commande simple et à bas prix qui gère le bon fonctionnement d'une station de pompage pour ne pas utilisé un automate programmable qui coûte cher et très rare à Madagascar.

Cette carte de commande nécessite une unité de contrôle et de traitement qui est le microcontrôleur PIC 16f877, une horloge à temps réel PCF 8583, une unité d'adaptation des entrées/sorties, une interface « utilisateur » et une unité d'affichage LCD.

Suivant différentes règles, cette carte, prend en compte les paramètres des entrées/sorties, optimise la consommation en énergie électrique de la station de pompage suivant le tarif de la JIRAMA (jour, pointe, nuit).

La réalisation et la simulation sont assurées par le logiciel PROTEUS Version 7.6. Le programme qui pilote la carte est écrit en ASSEMBLEUR et en Langage C.

Mots Clés : Automates programmables industriels

Rapporteur : Monsieur RATSIMBA Mamy Nirina

Adresse des auteurs : **Bloc 16/1 Cité des 67Ha Sud Tanambao 101**

VB 72 ES Bis Ambatoroka Antananarivo 101