République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de L'**E**nseignement **S**upérieur et de la
Recherche **S**cientifique

Université **M**ouloud **M**ammeri De **T**izi-**O**uzou

Faculté De **G**énie **E**lectrique Et D'informatique
DEPARTEMENT D'AUTOMATIQUE

Projet de Fin d'Etude
De MASTER PROFESSIONNEL
Spécialité : Automatique industriel

Présenté par
Rabah DJEBRA
Bilal BOUSSADIA

Mémoire dirigé par **Takfarinas CHELLI**

Thème

Conception et simulation d'une carte de commande à base de microcontrôleur pour l'unité de transfert dans la briqueterie Irdjen

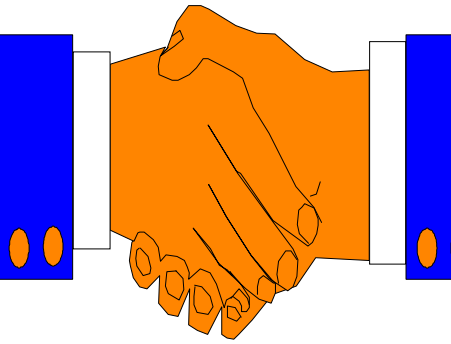
Mémoire soutenu publiquement le 08/ 07 / 2018 devant le jury composé de :

Mr Idjeri
Grade, Lieu d'exercice, Président

Mme Bouguandor
Grade, Lieu d'exercice, Examinateur

2017/2018

Préparé à : Briqueterie Irdjen



Remerciements

Nous tenons à remercier vivement notre promoteur Mr T.CHELLI, pour son aide précieuse et de nous avoir fait profiter de sa rigueur scientifique, de son

Expérience et de nous avoir encouragés tout au long de ce travail. On le

Remercie sincèrement pour ses conseils, sa patience et sa disponibilité tout au

Long de notre projet.

Ainsi qu'à notre Co-promoteur Mr G. Salem, de nous avoir pris en charge et bien encadré durant notre stage, ses remarques et ses conseils nous ont sérieusement permis de réaliser notre travail dans sa meilleure forme.

Nous remercions Le président du jury et les examinateurs d'avoir accepté de juger notre travail.

Nos remerciements s'adressent également au personnel de la briqueterie IRDJEN et particulièrement l'effectif du l'unité transfert qui ont contribués à notre formation durant notre stage.

Nous remercions ainsi Mr D. Lyes, de nous avoir réalisé le plan de l'unité de transfert sur AUTOCAD.

Ainsi qu'à R. Marzouk de nous avoir aidé sur le domaine logistique.

Tous nos remerciements vont à tous les enseignants qui ont contribué à notre formation durant notre cursus universitaire, pour le riche savoir qu'ils nous ont transmis avec rigueur et dévouement.

Enfin, nous tenons à remercier également toutes les personnes ayant contribué de près ou de loin à la réalisation de notre travail.

Dédicaces

Je dédie ce modeste travail réalisé à dieu à :

- *Mes très chers parents qui m'ont soutenu durant tout mon cursus d'études, que je ne pourrai jamais assez remercier pour leur sacrifices, que dieu les garde.*
- *Mon cher frère : Saïd.*
- *Mes très chères sœurs : Samia, Hanane.*
- *Touts mes cousins, toute la famille et mes amis.*
- *Mon binôme Bilal, et toute sa famille*

Rabah

Dédicaces

Je dédie ce modeste travail réalisé à dieu à :

- *Mes très chers parents qui m'ont soutenu durant tout mon cursus d'études, que je ne pourrai jamais assez remercier pour leur sacrifices, que dieu les garde.*
- *A la mémoire de ma très chère tente Rafika que dieu l'accueil dans son vaste paradis.*
- *Mes très chères sœurs : Souraya et Melissa.*
- *Touts mes amis : Meziane, Massinissa, Bilal, Nadjia, Lyes, Mouloud, Marzouk, Nabil.*
- *Mon binôme Rabah, et toute sa famille.*

Bilal

Préambule : Présentation de l'entreprise

Chapitre 1 : Fonctionnement et modélisation de l'unité transfert

Chapitre 2 : Systèmes à microprocesseur et à microcontrôleur

Chapitre 3 : Circuit de commande de l'unité transfert

Chapitre 4 : Programmation et simulation de la carte de commande

Préambule : Présentation de l'entreprise

Chapitre 1 : Fonctionnement et modélisation de l'unité de séchage

Chapitre 2 : Systèmes à microprocesseur et à microcontrôleurs

Chapitre 3 : Circuit de commande de l'unité transfert

Préambule : Présentation de l'entreprise

Chapitre 1 : Fonctionnement et modélisation de l'unité transfert

Chapitre 2 : Systèmes à microprocesseur et à microcontrôleur

Chapitre 3 : Circuit de commande de l'unité transfert

Chapitre 4 : Programmation et simulation de la carte de commande

Préambule : Présentation de l’entreprise d’accueil

1-Présentation de l'entreprise :

1.1-Situation géographique :

La briqueterie d'TRDJEN est située sur la route nationale n°15, allant vers Larbaa Nath Irathen, dans la zone industrielle d'Oued Aissi, est une entreprise de droit privé. Depuis 2006. Dénommée avant la privatisation ALTEC qui était affiliée à l'entreprise des produits rouges de centre, avant d'être privatisée. Son activité a débuté en 1998 et elle couvre une surface de 16750 m².

Son principal domaine d'activité est la fabrication et la commercialisation des briques rouges avec argile et ajouts qui subissent un processus de transformation physico-chimique. Ella a une capacité théorique plus de 200000 T/an.

1.2-Pièces à fabriquer:

Les pièces dont la fabrication a été prévue sont détaillées ci-après, avec dimension et poids se référant à la matière une fois cuit, comme indiqué dans la table (**Tab.0.1).**

Tab.0.1 : table représentent les pièces fabriqué et leur caractéristique.

Dénomination	Coupe Mm	Largeur Mm	Hauteur mm	Poids Kg
Brique creuse à 8 trous	300	100	200	3.8
Brique creuse à 12 trous	300	150	200	5.6

1.3- Matière premières :

Les matières de bases utilisées comme point de départ sont l'argile avec un taux de 75% et le tuf 25%, qui sont traités par voie humide, avec broyeur laminoir, toutes fois en fonction de l'humidité réelle d'entrée et des types d'argiles à utilisées.

On varie la préparation de l'argile de la façon qu'on désire soit en passant par une voie sèche ou bien humide.

1.4- Structure de l'entreprise:

La structure de l'usine est en ateliers selon le schéma commun à toutes les briqueteries selon le schéma sur la figure **(Fig.0.1)** ci-dessus :

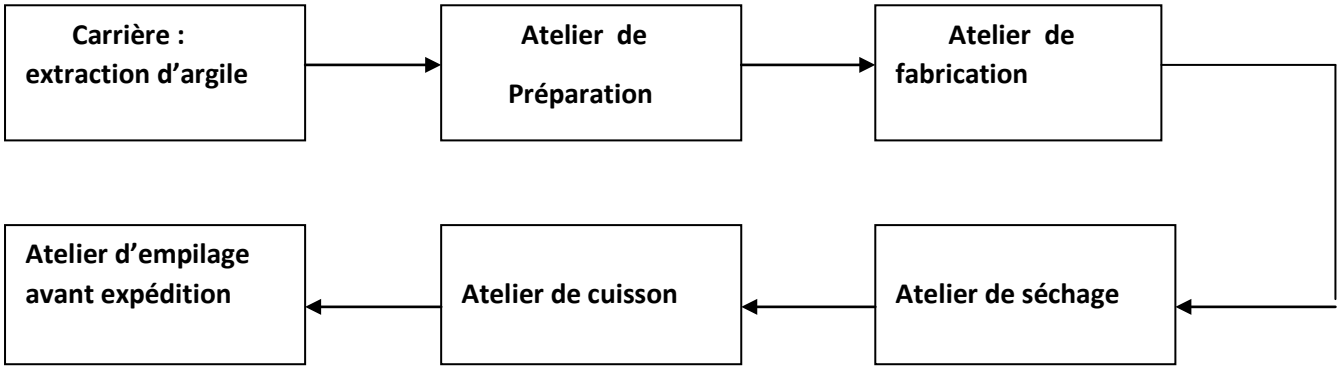


Fig.0.1 : structure de l'usine Irdjen.

1.5- les équipements utilisés :

Les équipements installés sont de fabrications italiennes telles que **BONGIOANNI, SYMBOL, INSTALAT**, et aussi allemande telles que **CERAMDRY, TECTON.**

Les machines constituent la chaine de productions sont dépendent l'une de l'autres pour une production continue.

En tenant compte de la complicité et la diversité des taches pour assurer une production qualitative et quantitative, on fait appel à un effectif de 135 personnes repartie sur les différentes taches en 3 équipes de 35 personnes chaque 'une durant la longueur de la journée, et cela pour assurer un rendement efficace sur 24h et cela dans tout les domaines d'activités :

- Production
- Maintenances
- Administration
- Hygiène
- Sécurité
- Commerciale

1.6- Organigramme de l'usine d'IRDJEN :

La briqueterie IREDJEN se structure selon l'organigramme indiqué dans la figure (**Fig.0.2**).

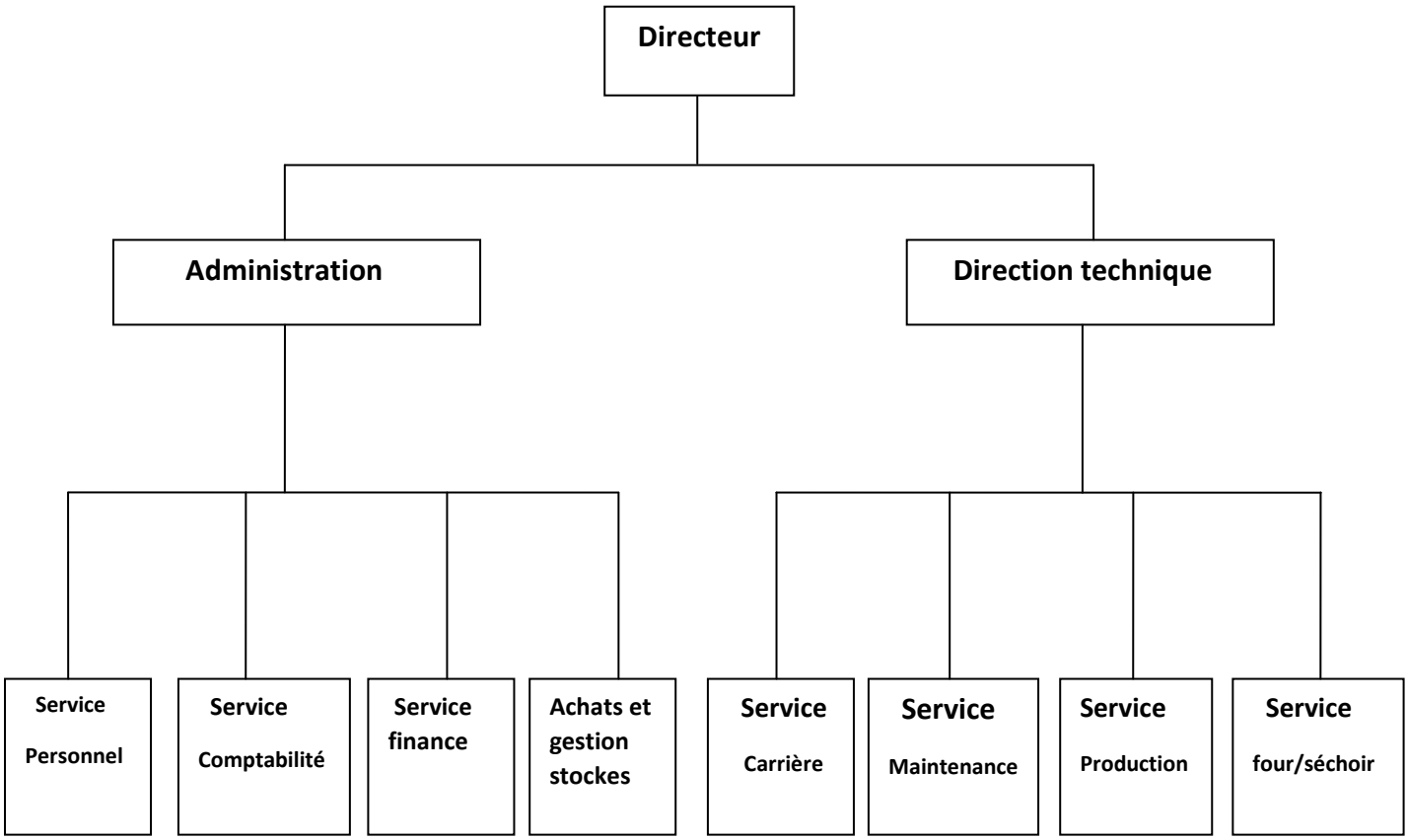


Fig.0.2 : Organigramme de l'usine Irdjen.

INTRODUCTION GENERALE

Dans le but de dominer le marché, les entreprises de productions s'évoluent régulièrement. L'évolution industrielle de ces dernières est la course infinie vers la perfection, pour l'objectif de répondre à toutes les exigences que ça soit de nature technique, économique ou humaines.

De nos jours, le développement des sociétés industrielles amène a une obligation d'amélioration des systèmes automatisés et un développement intense.

Actuellement la plupart des systèmes industriels automatisés sont à base d'automates programmables qui ont fournis des solutions pour satisfaire les exigences et les critères demandés, cependant lorsqu'on consulte leurs couts élevés et leur besoins matériels surchargés, ça devient un inconvénient pour leur utilisations et provoquent une concurrence dans le domaine de la recherche scientifique pour trouver la solution de rechange optimale, d'où la naissance de plusieurs façons et nouvelles technologies et parmi elle, les microcontrôleurs.

L'idée de l'utilisation des microcontrôleurs est née de la nécessité de disposer pour certains systèmes d'une commande avec des performances assez élevées, de plus leurs indéniables avantages sur la logique câblée. En effet pour modifier le fonctionnement d'une application, il suffit de modifier le programme sans refaire le câblage. Les microcontrôleurs possèdent également la puissance d'un microprocesseur mais ils ont un atout en plus, du fait qu'ils possèdent des périphériques intégrés dans le même boitier. Les microcontrôleurs sont actuellement les plus utilisés dans les montages nécessitant les commandes des appareils et des équipements

Dans ce projet, notre contribution intervient au niveau de l'automatisme partiel, d'une unité de transfert des chariots vers la station séchage dans une briqueterie, L'unité de transfert joue un rôle important dans la fabrication des brique, car c'est elle qui assure le transport des briques de l'entrées jusqu'à la sortie de séchoir.

On va collecter tous les données, les liaisons entre les différents relais, acquérir les outils et méthodes afin de gérer de façon optimale un projet d'automatisme industriel et commander cette station à l'aide d'une carte de commande à base de microcontrôleur. Ce mémoire se compose de quatre chapitres qui sont exposés selon l'ordre suivant :

- **le chapitre 1** est destiné au fonctionnement ainsi que la modélisation de l'unité de transfert.
- **Le chapitre 2** est consacré à la présentation des systèmes à microprocesseurs et à microcontrôleurs.
- **Le chapitre 3** faits l'objet du circuit de commande de l'unité de transfert à l'aide de **« PROTEUS ».**
- **le chapitre 4** est dédié à la Programmation et simulation de la carte de commande.
- **En fin**, notre travail est terminé par une conclusion.

CHAPITRE 1 : Fonctionnement et modélisation de l’unité transfert

1-Introduction

Avant de procéder à la programmation, et pour réaliser notre carte de commande on doit d'abord connaitre notre système, ses composants et sa technologie. Pour passer en suite a l'élaboration d'un cahier de charge qui détaille le fonctionnement et à la fin, faire la modélisation pour définir les étapes et les conditions.

2-Présentation du système :

Notre système est un ensemble de machines automatisés relié entre elles pour réaliser une fonction prédéfinis qui est le transfert des brique de l'unité de production vers l'unité de séchage qui est 'le séchoir industriel' dans le but d'évaporer l'eau présent dans les brique et l'évacuation de l'humidité, qui est du à leur façonnage, Le système est constitué de trois zones indépendante :

2.1-La zone d'entrée :

Elle représente la partie de chargement des chariots comme l'indique la (**Fig.1.1)**, et leurs déplacements vers la porte d'entrée de séchoir. Elle se compose de deux voies et une table rotative montrée sur (**Fig.1.2**.).

Fig.1.1: image d'un chariot chargé.

Fig.1.2 : image de la table rotative.

2.2-La zone de séchage :

Elle se traduite par le lieu d'évaporation de l'eau dans les briques et l'évacuation de l'humidité, elle se compose de deux transporteur de chariots, un à l'entrée et un autre à la sortie et elle est séparé en 6 voies d'entrées et 7 voies de sorties, avec une capacité de 23 chariots chacune. La figure **(Fig1.3)** représente l'entrée du séchoir.

Fig.1.3 : image de la porte d'entrée de séchoir.

2.3-La zone de sortie :

c'est la partie d'évacuation des chariots séchés vers la porte de sortie de séchoirs et leurs déplacements à l'intermédiaire d'un autre accrocheur (**Fig.1.4)**, vers la prochaine étape (chargement et préparation de l'expédition vers la cuisson).

Fig.1.4 : image d'un accrocheur.

3-Fonctionnement du système :

A l'entrée, après leur chargements les chariots à briques se déplacent horizontalement sur deux voies perpendiculaires à l'aide de trois accrocheurs placés sur les deux voies et commandés avec des moteurs triphasés à deux sens de rotation et la liaison entre les deux voies se fait avec une table rotative avec un angle de 90° commandé par un autre moteur triphasé avec deux sens de rotation, à la fin de ce parcours le chariots rentre dans le séchoir.

A l'intérieur du séchoir il y'a un transporteur qui assure le chargement de chariot dans une voie de séchage prédéfinie avec une condition 'de gauche à droite' et un autre transporteur à la

sortie de cette voie de séchage qui assure à son tour le transport de chariot vers la voie de sortie , avec comme ordre de sélection des vois 'de droite à gauche'.

Chaque transporteur se déplace avec un moteurs triphasé à deux sens de rotations après la désactivation des bloqueurs. Et le déchargement de chariot vers la voie de séchage est effectué à l'aide d'un vérin qui se trouve sur le transporteur d'entrée, par contre le chargements sur le transporteur de sortie est fait pour cause de surpoids de 24éme chariot par la règle de premier entré premier chargé c'est-à-dire le chargement du 24éme chariot pousse le dernier chariots ce qui provoque la sortie de premier et son chargements sur le transporteur de sortie après cela son évacuation par la porte de sortie à l'aide d'un accrocheur.

4-Les éléments utilisés dans le système :

4.1-Les moteurs triphasés :

Les moteurs asynchrones triphasés sont utilisés pour transformer l'énergie électrique en énergie mécanique grâce à des phénomènes électromagnétiques.il est représenté sur la (**Fig.1.5)**. **[1]**

Fig.1.5 : moteur triphasé.

4.2- Les vérins :

Un vérin est un actionneur pneumatique **(Fig.1.6)** qui est soumis à des pressions d'air comprimé qui permettent d'obtenir des mouvements dans un sens puis dans l'autre. Les mouvements obtenus peuvent être linéaires ou rotatifs.

Dans l'unité de transfert il y'a trois vérins le premier est utilisé pour décharger les chariots de transporteur d'entrée vers les voies de séchages et les deux autre placé sur les transporteurs pour activer et désactiver les bloqueurs. **[2]**

Fig.1.6 : un vérin.

4.3-Les contacteurs :

Appareil électromagnétique de connexion ayant une seule position de repos, commandé électriquement et capable d'établir, de supporter et d'interrompre des courants dans des conditions normales du circuit. C'est un appareil de commande et de contrôle capable d'effectuer un grand nombre de manœuvres sous des courants de charges normaux, il est représenté sur la (**Fig.1.7)**. **[2]**

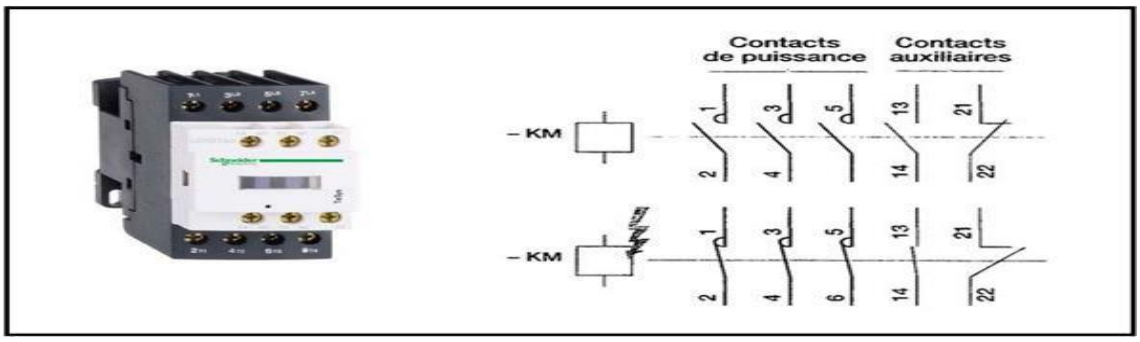


Fig.1.7 : contacteur.

4.4-Les Relais :

Utilisables en alternatif et continu, ils sont destinés à assurer une protection thermique contre les surcharges faibles et prolongées. Et assurer aussi le bon fonctionnement des actionneurs **(Fig.1.8).**

Différentes type de relais :

- **Relais monostable :** C'est le plus courant des relais, lorsque sa bobine est sous tension, l'armature mobile actionne les contacts qui changent d'état. Lorsque le courant cesse, l'armature revient à la position initiale ainsi que les contacts.
- **Relais bistable :** Ce relais comporte généralement deux bobines montées en opposition. La mise sous tension d'une bobine déplace l'armature mobile et ses contacts qui restent en position par un système magnétique ou mécanique quand la bobine n'est plus alimentée. Pour changer la position il faut alimenter brièvement l'autre bobine. **[2]**

Fig.1.8 : image d'un relai thermique.

4.5-Les distributeurs (Pré-actionneurs pneumatiques) :

Ce sont des constitués chargés de distribuer l'énergie pneumatique vers les actionneurs pneumatique sur ordre constituant de commande **(Fig.1.9).** Ces ordres supportés par un signal électrique, Il commande la circulation de l'énergie entre la source et l'actionneur. Il permet de contrôler le mouvement d'un actionneur, choisir le sens de circulation d'un fluide et démarrer ou arrêter la circulation d'un fluide, et dans notre système on a deux types de distributeurs **(Fig.1.10).** **[2]**

- Distributeur bistable avec deux commandes électriques
- Deux distributeurs monostables commande électrique avec un retour a ressort pour les vérins des bloqueurs.

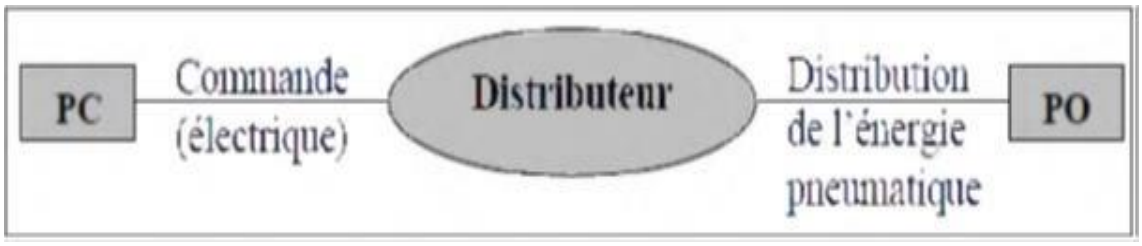


Fig.1.9: Structure de pré-actionneur pneumatique.

Ils permettent de commuter et contrôler la circulation des fluides sous pression et assurent diverses fonctions :

- Contrôle de mouvement de la tige d'un vérin.
- Choisir le sens de circulation d'un fluide,
- Exécuter des fonctions logiques.
- Démarrer ou arrêter la circulation d'un fluide,
- Etre des capteurs de position (pressostat, vacuostas).

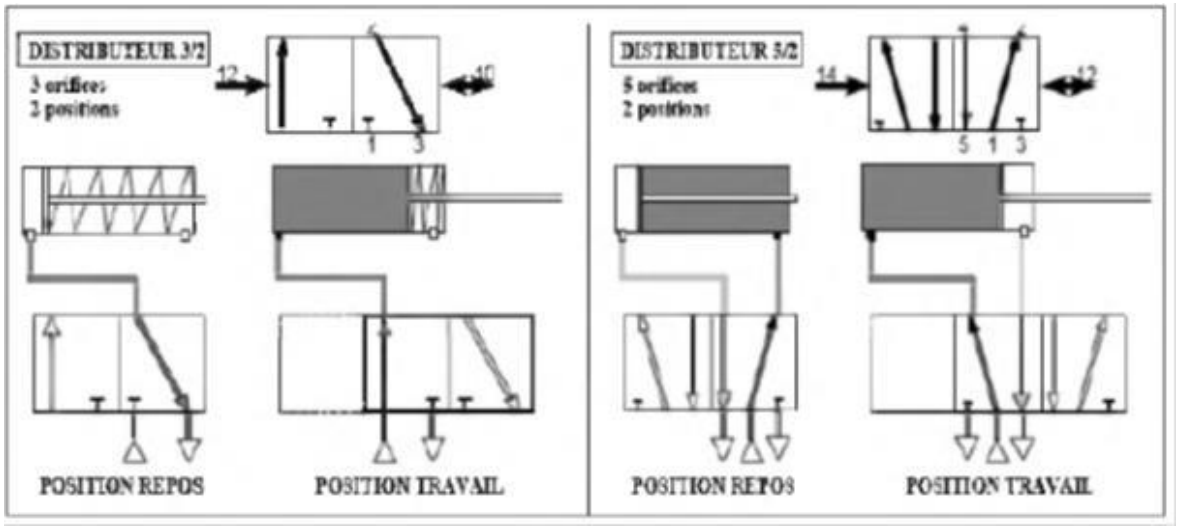


Fig.1.10: Fonctionnement de distributeur.

4.6-Les Capteurs :

Un capteur est un dispositif transformant l'état d'une grandeur physique observée en une grandeur utilisable, son fonctionnement est expliquer sur la (**Fig.1.11**). **[2]**

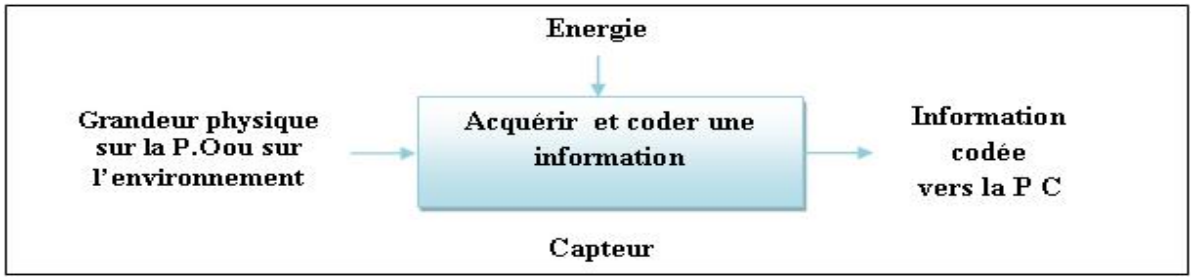


Fig.1.11: principe de fonctionnement d'un capteur.

4.7-Les Protections :

- ✓ ***Disjoncteur magnétothermique :*** destiné pour la protection contre les surcharges électriques et les courts circuits, il est représenté sur (**Fig.1.12) :**

Fig.1.12 : Disjoncteur magnétothermique et son symbole.

- ✓ ***Disjoncteurs moteurs magnétiques*** **:** pour la protection contre les courts circuits comme le montre la **(Fig.1.13) :**

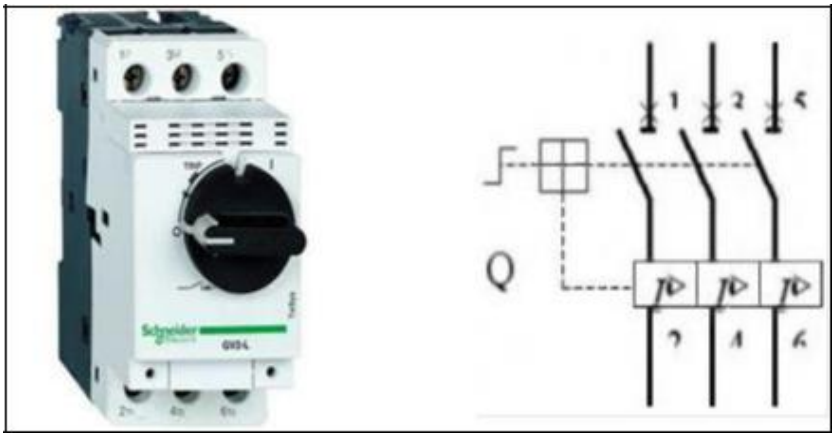


Fig.1.13 : Disjoncteur moteurs magnétique et son symbole

5-Cahier de charge:

Après le chargement de dernier étage du chariot à briques, ce dernier est dirigé vers le séchoir pour évaporer l'humidité présente dans les briques, le transfert de ce chariot est réalisé à travers deux rails reliés entre elles par une table rotative.

La poussée du chariot se fait a l'aide de trois accrocheurs avec deux têtes chaque un se placent en dessous du chariot entrainent avec elles le déplacement horizontale de celui-ci, Ces accrocheurs la sont commandés par des moteurs asynchrones à deux sens de rotation.

A l'arrivée du chariot devant la porte d'entrée il s'arrête et celle-ci s'ouvre complètement puis le chariot continue son déplacement a l'intérieur de séchoir, après l'entrée complète du chariot, il s'arrête et la porte se referme complètement puis il reprend son déplacement jusqu'au chargement sur le transporteur d'entrée qui se trouve sur sa position initiale ci à dire la voie 1.

A la fin de cette opération le transporteur d'entrée et celui de sortie se préparent à quitter leur poste pour rejoindre la position indiquer sur le pupitre de commande et en même temps le bloqueur de sécurité se désactivent pour autorisé le changement de position.

A l'arrivée des transporteurs sur la voie indiquer l'opération déchargement commence et cela a travers un vérin placé sur le transporteur d'entrée qui pousse le chariot sur la voie indiquer ce que provoque la sortie d'un autre chariot dans le coté opposé (pousser pour pousser un autre) et son chargement sur le transporteur de sortie déjà en place.

Pour terminer le transporteur de sortie se dirige vers la voie de sortie et à son arrivée la porte s'ouvre et a ce moment un accrocheur viens le déplacé et l'emmène en dehors de séchoir alors la porte se ferme.

Le circuit parcouru par le chariot depuis la première étape de chargement jusqu'à sa sortie de séchoir est représenté sur la figure (**Fig.1.14).**

6-Modélisation de notre système par « Grafcet » :

6.1-Définition de grafcet :

La signification de GRAFCET est graphique fonctionnel de commande étapes/transition le Grafcet est un outil graphique de définition pour l'automatisme séquentiel, en tout ou rien. Mais il est également utilisé dans beaucoup de cas combinatoires, dans le cas où il y a une séquence à respecter mais où l'état des capteurs suffirait pour résoudre le problème en combinatoire. Il utilise une représentation graphique. C'est un langage clair, strict mais sans ambiguïté, permettant par exemple au réalisateur de montrer au donneur d'ordre comment il a compris le cahier des charges. **[3]**

>> On trouve deux représentations de grafcet :

6.1.1-La représentation fonctionnelle ou de niveau 1 :

Elle donne une interprétation de la solution retenue pour un problème posé, en précisant la coordination des tâches opératives. Elle permet une compréhension globale du système.

6.1.2-La représentation technologique ou de niveau 2 :

Elle donne une interprétation en tenant compte des choix technologique relatifs à la partie de commande de l'automatisme, le type et la désignation des appareillages.

6.2-Table des mnémoniques :

Tab.1.1 : Table des mnémoniques.

Entrées/sorties	***Significations***
S1	Dernier étage de chariot chargé
S2, S3, S4	Capteurs de la position pour l'accrocheur A1 sur la voie 1
C1	Table rotative à gauche
C2	Table rotative à droite
B1	Capteur de la présence du chariot devant la table rotative
B2, B3	Capteurs de la présence du chariot sur la table rotative
S5, S6, S7, S8	Capteur de la position pour l'accrocheur A2 sur la voie 2
B4, B5	Capteur de présence du chariot au milieu de la voie 2
B6, B7	Capteur de présence du chariot devant la porte d'entrée
S9, S10, S11, S12, S13	Capteur de position de l'accrocheur A3 sur la voie d'entrée de séchoir
B8, B9	Capteur de présence du chariot devant le transporteur d'entrée
B11, B12	Capteur de présence du chariot sur le transporteur d'entrée
CPE1	Porte d'entrée de séchoir ouverte
CPE2	Porte d'entrée de séchoir fermé
EA1	Bloqueurs du transporteur d'entrée activée
EA2	Bloqueurs du transporteur d'entrée désactivée
S17	Présence du transporteur d'entrée devant la voie de séchage 1 (position initiale)
S19	Présence du transporteur d'entrée devant la voie de séchage 2
S21	Présence du transporteur d'entrée devant la voie de séchage 3
S23	Présence du transporteur d'entrée devant la voie de séchage 4
S25	Présence du transporteur d'entrée devant la voie de séchage 5
S27	Présence du transporteur d'entrée devant la voie de séchage 6
B13, B14	Présence du chariot sur le transporteur de sortie
S29	Présence du transporteur de sortie devant la voie de séchage 1
S31	Présence du transporteur de sortie devant la voie de séchage 2
S33	Présence du transporteur de sortie devant la voie de séchage 3

S35	Présence du transporteur de sortie devant la voie de séchage 4
S37	Présence du transporteur de sortie devant la voie de séchage 5
S39	Présence du transporteur de sortie devant la voie de séchage 6
S40	Présence du transporteur de sortie à la voie 7 devant la porte de sortie de séchoir
S41, S42, S43	Capteurs de position pour l'accrocheur A4 sur la voie de sortie de séchoir
CPS1	Porte de sortie ouverte
CPS2	Porte de sortie fermée
SA1	Bloqueurs du transporteur de sortie activé
SA2	Bloqueurs du transporteur de sortie désactivé
CV1	Vérin porté sur le transporteur d'entrée entrée
CV2	Vérin porté sur le transporteur d'entrée sortie
KM11	Marche avant de moteur relié à l'accrocheur A1
KM12	Marche arrière de moteur relié à l'accrocheur A1
KM21	Marche avant de moteur relié à l'accrocheur A2
KM22	Marche arrière de moteur relié à l'accrocheur A2
KM31	Marche avant de moteur relié à l'accrocheur A3
KM32	Marche arrière de moteur relié à l'accrocheur A3
KM41	Marche avant de moteur relié à l'accrocheur A4
KM42	Marche arrière de moteur relié à l'accrocheur A4
KMRD	Rotation à droite du moteur de la table rotative
KMRG	Rotation à gauche du moteur de la table rotative
KMP11	Ouverture de la porte d'entré
KMP12	Fermeture de la porte d'entré
KMP21	Ouverture de la porte de sortie
KMP22	Fermeture de la porte de sortie
KMTE1	Marche avant du moteur de transporteur d'entrée
KMTE2	Marche arrière du moteur de transporteur d'entrée
KMTS1	Marche avant du moteur de transporteur de sortie
KMTS2	Marche arrière du moteur de transporteur de sortie
EV1	Electrovanne qui commande la sortie de vérin porté sur le transporteur d'entrée
EV2	Electrovanne qui commande la rentrée de vérin porté sur le transporteur d'entrée
EV3	Electrovanne qui désactive les bloqueurs du transporteur d'entrée
EV4	Electrovanne qui désactive les bloqueurs du transporteur de sortie
AR	Arrêt d'urgence activé
‾‾ AR	Arrêt d'urgence désactivé

6.3- Grafcet fonctionnel ou niveau 1 de l'unité de transfert :

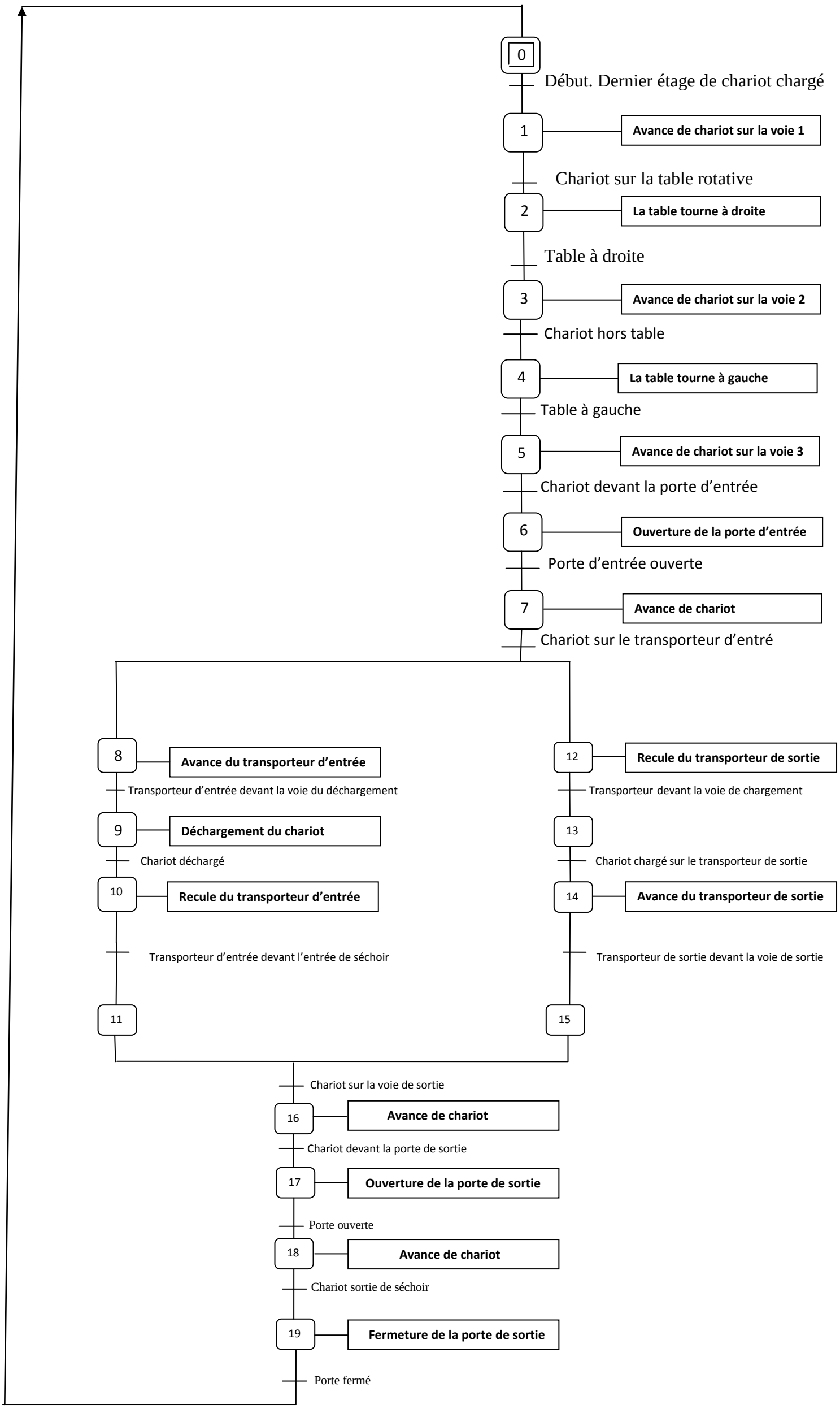


Fig.1.15 : Grafcet fonctionnel ou niveau 1 de l'unité transfert.

6.4.4. Sous grafcet numéro 4:

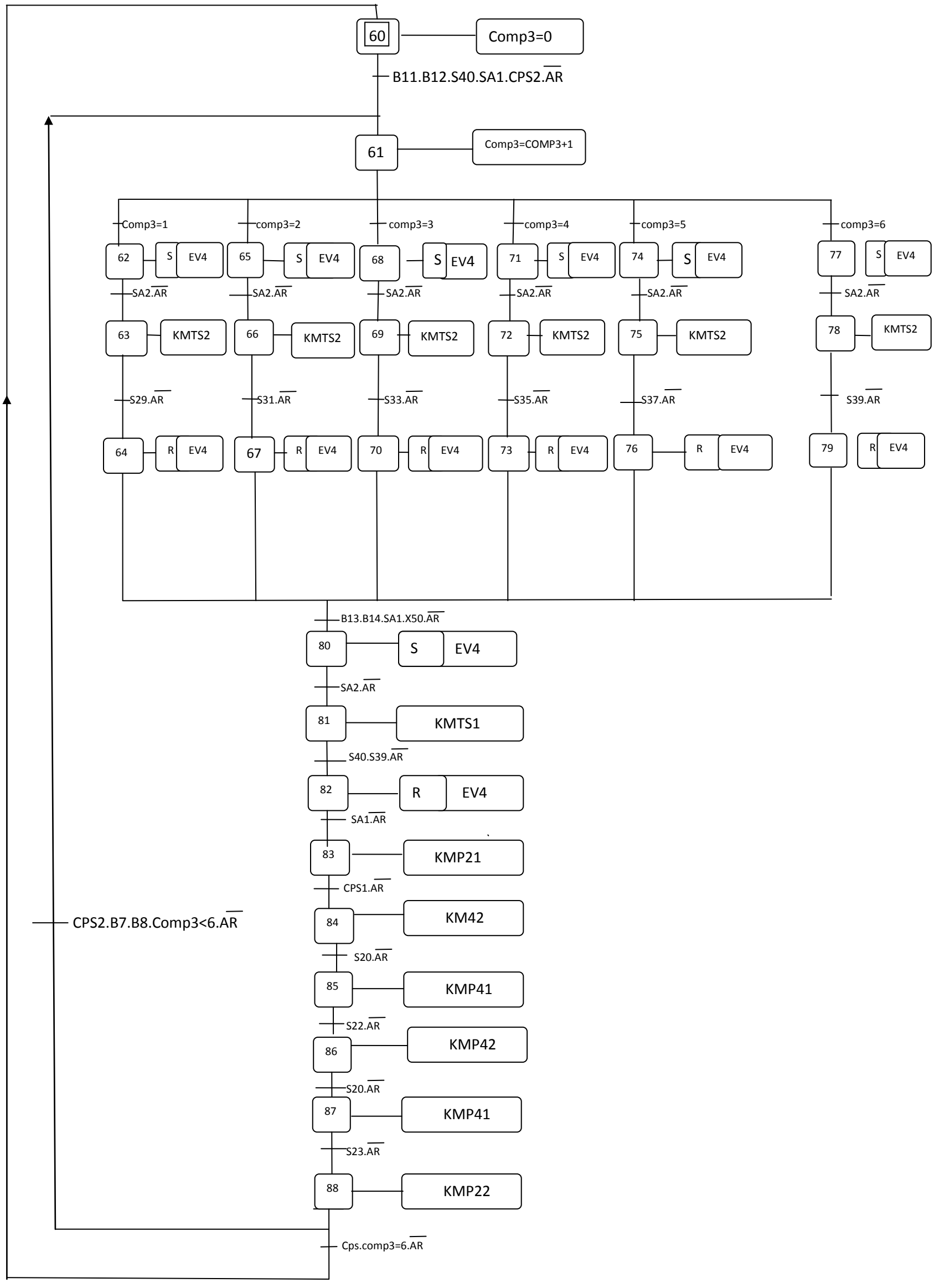


Fig.1.19 : Sous grafcet numéro 4 de l'unité de transfert.

6.4- Grafcet technologique ou niveau 2 de l'unité de transfert :

6.4.1- Sous Grafcet numéro 1:

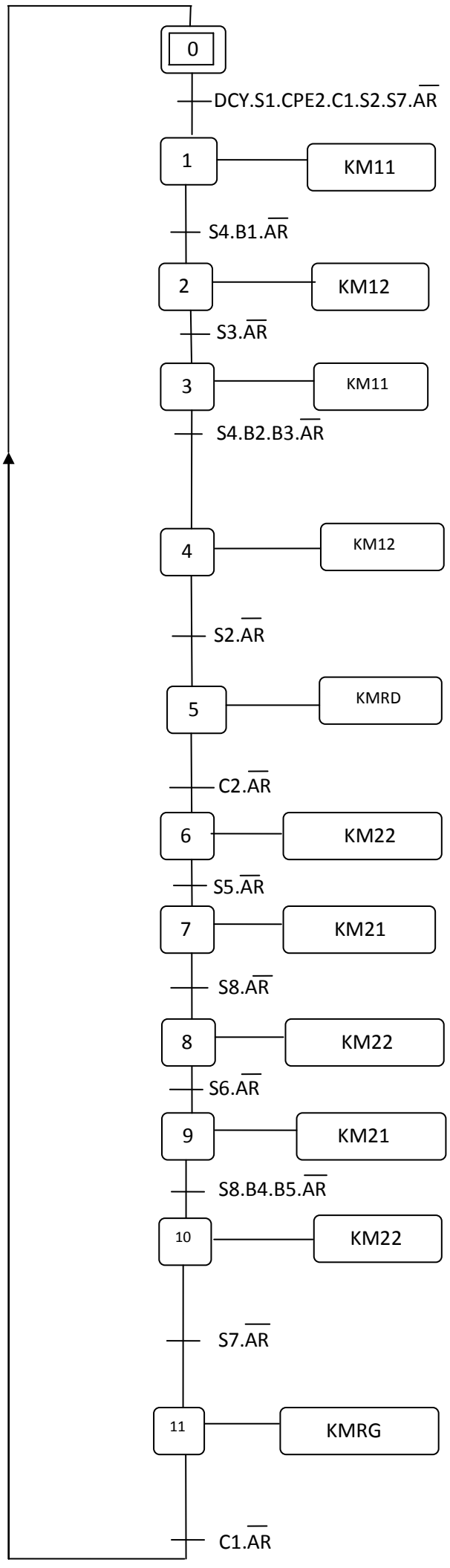


Fig.1.16: Sous grafcet numéro 1 de l'unité transfert.

6.4.2- Sous grafcet numéro 2:

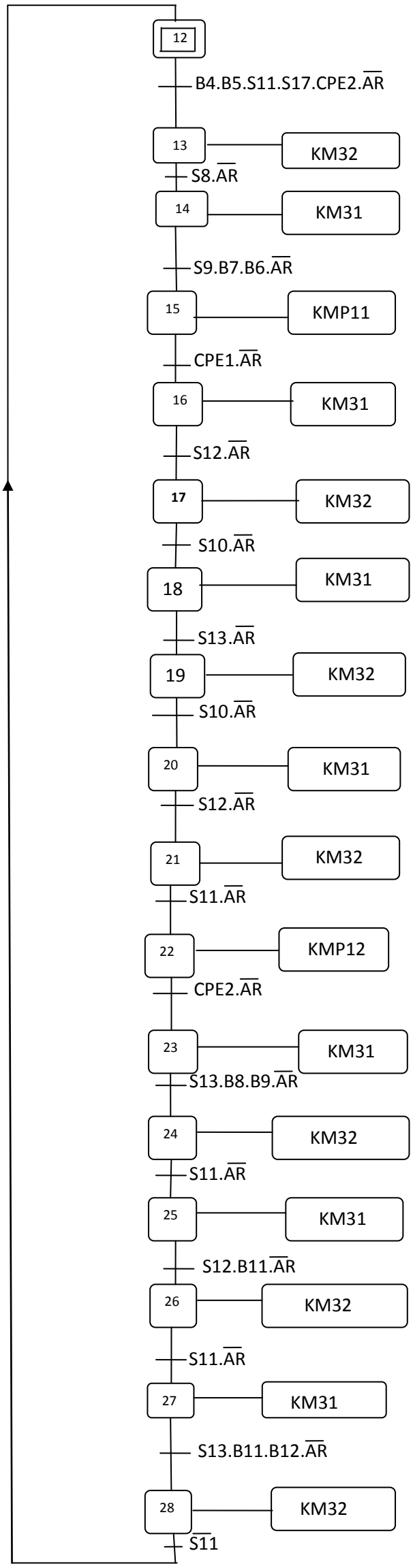


Fig.1.17 : Sous grafcet numéro 2 de l'unité transfert.

6.4.3- Sous grafcet numéro 3:

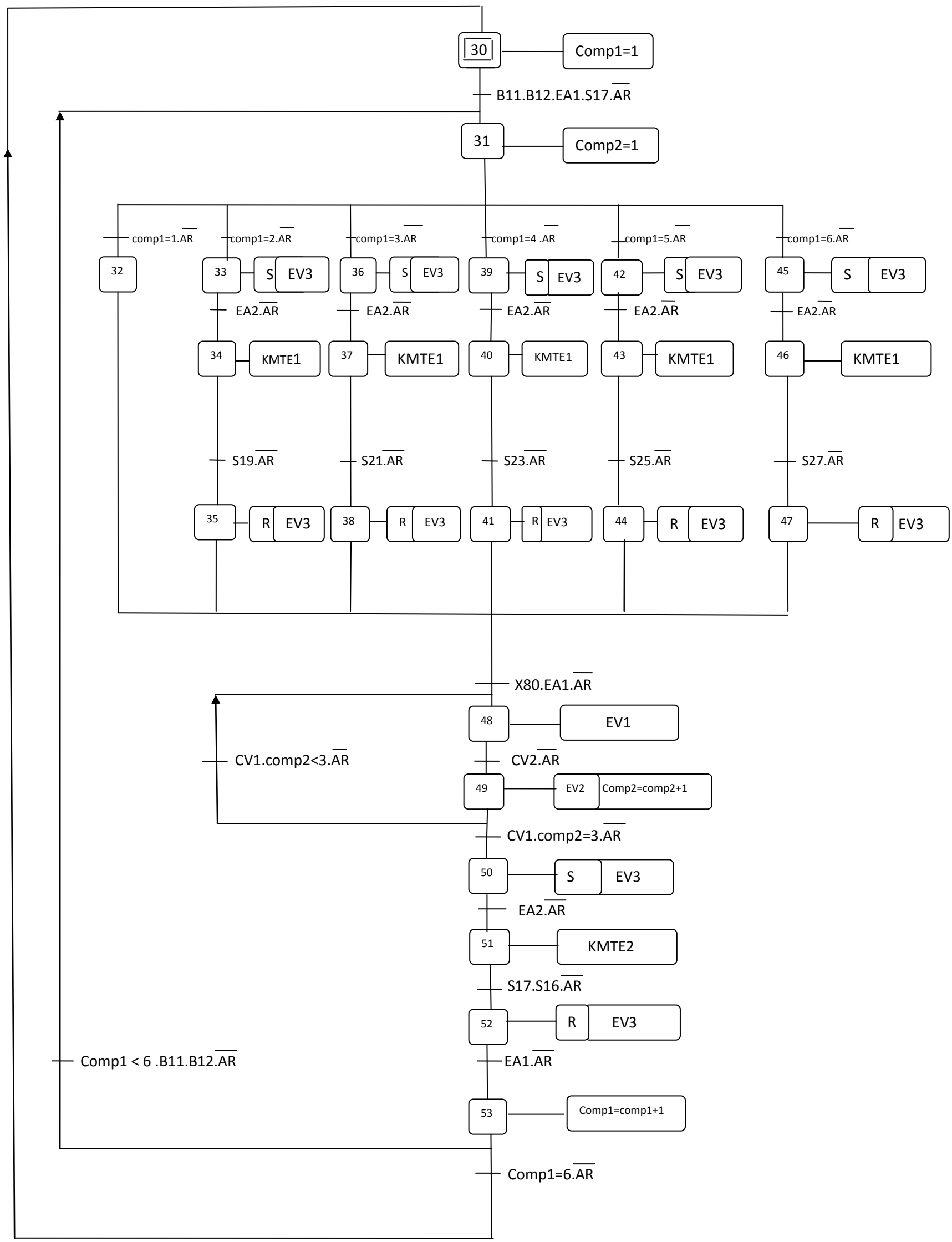


Fig.1.18 : Sous grafcet numéro 3 de l'unité transfert.

6.5- Grafcet principal:

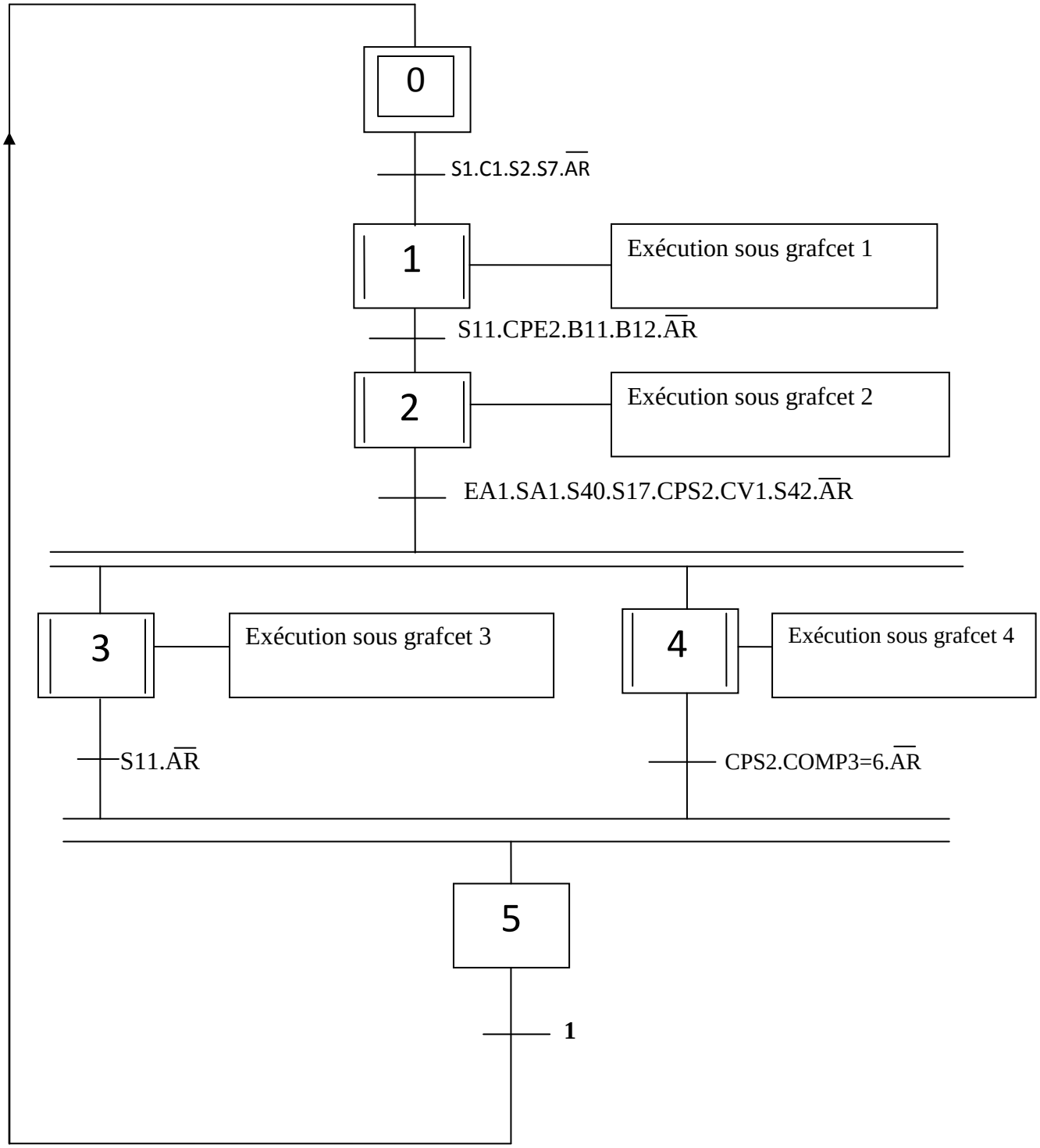


Fig.1.20 : Grafcet principal de l'unité transfert.

7- Conclusion :

Les systèmes de production automatisés deviennent indispensables pour obtenir une compétitivité des produits fabriqués de haute qualité. Dans ce chapitre on a vu en générale la structure des systèmes de productions automatises et l'instrumentation liés à ces systèmes pour la communication, la distribution d'énergie et la protection des équipements. Nous avons présenté les différents parties contrôle, puissance, communication et l'appareillage d'un système automatisé.

Le Grafcet nous a permet de modéliser le système dans le but de décrypter son cahier de charge d'un côté, et d'un autre de différencier les entrées et les sortie et déterminer les actionneurs, ce qui amène à comprendre son fonctionnement, pour pouvoir enfin réalisé notre projet et écrire son programme.

CHAPITRE 2 : systèmes à microprocesseur et à microcontrôleur

1-INTRODUCTION :

Les microcontrôleurs sont très utilisés dans le monde de l'industrie. Et pour programmer un microcontrôleur ; il est nécessaire de connaître sa structure interne : registres, mémoires, ports d'entrées sorties, et toutes leurs possibilités.

Par rapport à des systèmes électroniques à base de microprocesseurs et autres composants séparés, les microcontrôleurs permettent de diminuer la taille, la consommation électrique et le coût des produits. Ils ont ainsi permis de contrôler l'utilisation de l'informatique dans un grand nombre de produits et de procédés. **[4]**

2-Système à microprocesseur :

2.1-Définition :

Le processeur (CPU, pour Central Processing Unit, soit Unité Centrale de Traitement) est le cerveau de l'ordinateur. Il permet de manipuler des informations numériques, c'est-à-dire des informations codées sous forme binaire, et d'exécuter les instructions stockées en mémoire, Le premier microprocesseur (Intel 4004) a été inventé en 1971. Il s'agissait d'une unité de calcul de 4 bits, cadencé à 108 kHz. Depuis, la puissance des microprocesseurs augmente exponentiellement. **[4]**

2.2-Structure interne d'un microprocesseur :

Le microprocesseur est composé de trois parties :

2.2.1-L'unité de contrôle :

C'est la partie la plus complexe du processeur. Elle se compose en plusieurs parties et registres dont elle doit assurer la coordination. **[5]**

2.2.1.1-Le registre d'instruction :

Ce registre reçoit les instructions à exécuté, il contient l'instruction courant à exécuter. **[5]**

2.2.1.2-Le décodeur :

Il interprète l'instruction chargée dans le registre d'instruction. **[5]**

2.2.1.3-Le séquenceur :

Il est capable de d'ordonnancer les divers opérations élémentaires du processeur nécessaires pour exécuter l'instruction. **[5]**

2.2.1.4-Le compteur ordinal :

Est un registre particulier qui contient à tout instant, l'adresse de l'instruction suivante à exécuter. **[5]**

2.2.1.5-Le registre d'état :

Est un registre particulier, il représente à tout moment l'état de processeur. **[5]**

2.2.2-L'unité de calcule :

Comme son nom l'indique, effectue tous les calculs au sein du processeur. À côté des opérations arithmétiques, elle peut aussi procéder à des opérations logiques. **[5]**

2.2.2.1- L'unité arithmétique et logique (UAL):

Elle reçoit ses opérandes (les octets qu'elle manipule) du bus de données. Celles-ci peuvent provenir de registres ou de la mémoire. A la fin d'une opération, l'UAL peut aller modifier certains bits du registre d'état. **[5]**

Cette unité peut exécuter deux types d'opérations :

- **Opérations arithmétiques** : Elles incluent l'addition et la soustraction qui sont des opérations de base (une soustraction est une addition avec le complément à deux), la multiplication et la division. Les données traitées sont considérées dans des représentations entières.

- **Opérations logiques** : Ces opérations sont effectuées bit à bit sur les bits de même poids de deux mots, tel que ET, OU, NOT OU EXCLUSIF, de même les opérations de rotation et de décalage (arithmétique et logique)

2.2.2.2-L'accumulateur :

Il s'agit d'un registre d'usage général recevant des opérandes, des résultats intermédiaires ou des résultats provenant de l'unité arithmétique et logique. Ils évitent des appels fréquents à la mémoire, réduisant ainsi les temps de calcul. Donc la plupart des opérations arithmétiques et logiques se font dans l'accumulateur. **[5]**

2.2.3-Bus de communication :

Il s'agit de trois bus (bus d'adresse, bus de données et bus de commandes) :

2.2.3.1- Le bus d'adresses :

Il permet d'adresser un élément par le microprocesseur .il est unidirectionnel .il détermine la capacité maximale d'adressage du système, c'est à dire le nombre maximum de mots de la mémoire associée (ex : 16 bits "adressent" 64 Kmots).

2.2.3.2- Le bus de données :

C'est un ensemble de fils bidirectionnels qui va permettre le transfert des données entre les différents éléments du système. C'est par ce bus que sont transmises les données qui doivent être traitées par le microprocesseur. A l'inverse, c'est également par ce bus que transitent les résultats en sortie du microprocesseur. Autrement dit, toutes les données entrantes et sortantes du microprocesseur sont véhiculées par le bus de données qui fixe la longueur du mot échangé avec la mémoire. **[5]**

2.2.3.3- bus de commandes :

Ce processeur utilise le bus de commandes pour indiquer la direction de la transaction sur le bus de données. **[5]**

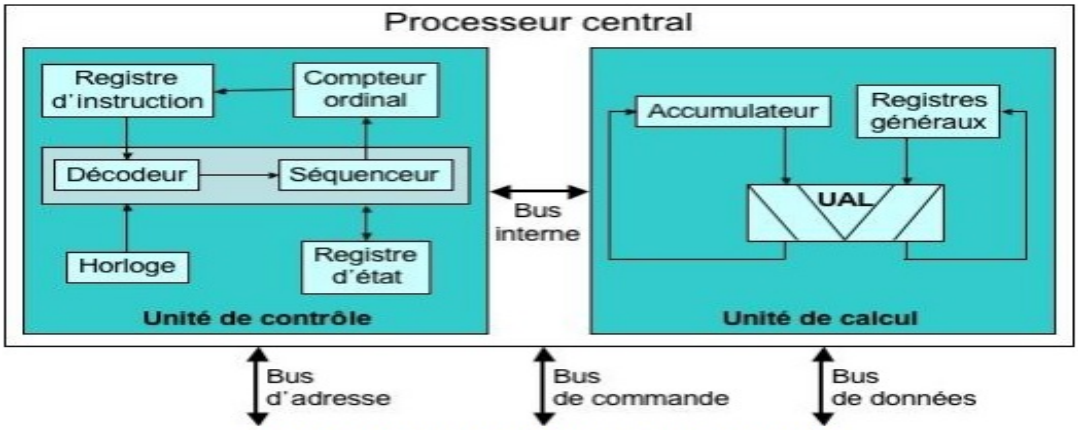


Fig.2.1 : Structure interne d'un microprocesseur.

2.3-Les différentes architectures d'un microprocesseur :

2.3.1-L'architecture Von Neumann :

Cette architecture se présente avec les caractéristiques suivantes :

- ✓ Un seul chemin d'accès à la mémoire :
  - Un bus de données (programme et données).
  - Un bus d'adresse (programme et données).
- ✓ Architecture des processeurs d'usage général :
  - Goulot d'étranglement pour l'accès à la mémoire. **[6]**

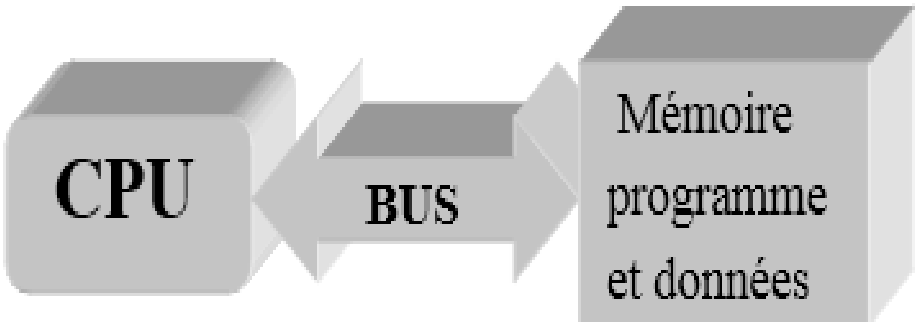


Fig.2.2: architecture Von Neumann.

2.3.2-L'architecture Harvard:

Cette architecture se présente avec les caractéristiques suivantes :

- ✓ Séparations des mémoires programme et données
  - Un bus de données programme
  - Un bus de données pour les données
  - Un bus d'adresse programme
  - Un bus d'adresse pour les données
- ✓ Meilleure utilisation du CPU :
  - Chargement du programme et des données en parallèles. **[6]**

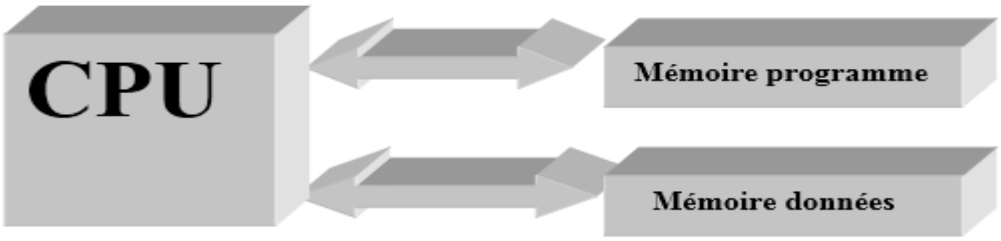


Fig.2.3 : architecture Harvard.

2.4-Fonctionnement d'un système à base de microprocesseur :

2.4.1-Les interruptions :

Les interruptions permettent au matériel (périphérique) de communiquer avec le processeur. Les interruptions sont de deux types : les interruptions matérielles.et les interruptions logicielle. Dans certains cas, on désire que le processeur réagisse rapidement à un évènement extérieur. Si un périphérique nécessite une intervention, il génère lui-même une demande d'interruption.

Une interruption est signalée au processeur par un signal électrique sur une borne spéciale. Lors de la réception de ce signal, le processeur (traite) l'interruption dès la fin de l'instruction qu'il était en train d'exécuter. Le traitement de l'interruption consiste soit :

- à l'ignorer et passer normalement à l'instruction suivante : C'est possible uniquement pour certaines interruptions, nommées interruptions masquable. Il est en effet parfois nécessaire de pouvoir ignorer les interruptions pendant un certain temps, pour effectuer des traitements très urgents par exemple. Lorsque le traitement est terminé, le processeur démasque les interruptions et les prend alors en compte.

- à exécuter un traitant d'interruption (interrupthandler) : Un traitant d'interruption est un programme qui est appelé automatiquement lorsqu'une interruption survient.

- Parfois le microprocesseur est sollicité par plusieurs interruptions en même temps : pour répondre à ces appels un ordre de priorité est souvent pris en compte pour leurs traitements. **[6]**

2.4.2-L'écriture en mémoire (WRITE):

Pour écrire une donnée dans la mémoire le microprocesseur doit placer l'adresse de la donné sur le bus d'adresses (son emplacement dans la mémoire) puis il place la donnée sur le bus de données et enfin génère le signal WRITE (ordre d'écriture dans la mémoire).

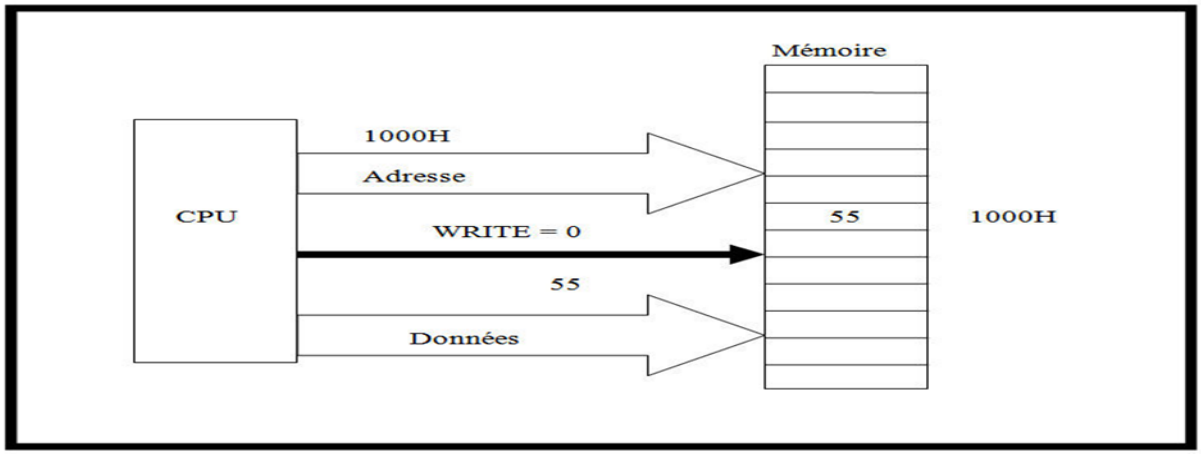


Fig.2.4: l'écriture en mémoire (write).

2.4.3-La lecture de la mémoire (READ):

Pour lire une donnée de la mémoire le microprocesseur doit connaître son emplacement, en effet il dépose son adresse sur le bus d'adresses puis génère le signal READ (il demande une opération de lecture de la mémoire) alors la donnée sera acheminée vers le microprocesseur à travers le bus de données, qui sera stockée dans un registre dans le microprocesseur. **[7]**

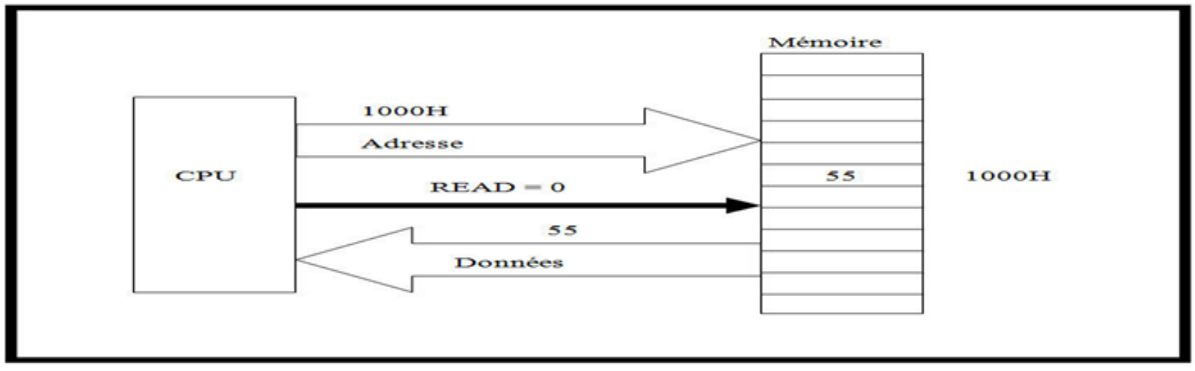


Fig.2.5 : la lecture en mémoire (Read)

2.4.4-Accès direct à la mémoire :

Lorsqu'un transfert en mémoire est nécessaire de la mémoire RAM à un port d'E/S, la CPU lit le premier octet en mémoire et le charge dans l'un des registres du microprocesseur. La CPU écrit ensuite l'octet rangé précédemment sur le port d'E/S approprié.

Il en résulte que le microprocesseur effectue des opérations de lecture et d'écriture répétées. Ainsi un certain temps est perdu entre le traitement de chaque octet. Pour remédier à ce problème, une procédure est mise au point pour l'accès direct à la mémoire (Direct Memory Access), qui permet de transférer des données de la mémoire RAM au port d'E/S sans passer par le microprocesseur. Pour cela, un contrôleur DMA, qui reprend le rôle de la CPU, c'est à dire qu'il gère les transferts de la RAM aux ports d'E/S. **[6]**

3-Système a microcontrôleur :

3.1-Définition :

Un microcontrôleur est une unité de traitement de l'information qui intègre un maximum de fonctions dans un même boitier. L'intégration de ces fonctions dans le même environnement permet de créer des applications plus simplement. Chaque fabricant a ses familles de microcontrôleur. Une famille se caractérise par un noyau commun (le, le jeu d'instruction, microprocesseur…). **[7]**

Fig.2.6: un microcontrôleur.

3.2-la structure interne d'un microcontrôleur :

On peut décomposer la structure interne d'un microcontrôleur en trois parties :

-Les mémoires, -Le processeur, -Les périphériques. **[7]**

3.2.1-Les mémoires :

Elles sont chargées de stocker le programme qui sera exécuté ainsi que les données nécessaires et les résultats obtenus.

3.2.2-Le processeur :

C'est le cœur du système puisqu'il est chargé d'interpréter les instructions du programme en cours d'exécution et de réaliser les opérations qu'il contient .Au sein du processeur, l'unité arithmétique et logique (ALU) interprète, traduit et exécute les instructions de calcul.es nécessaires et les résultats obtenus.

3.2.3-Les périphériques :

Ils ont pour tâche de connecter le processeur avec le monde extérieur dans les deux sens. Soit le processeur fournit des informations vers l'extérieur (périphérique de sortie), soit il en reçoit (périphérique d'entrée).

3.3-Rôle d'un système à microcontrôleur :

Un système à microcontrôleur permet :

- D'acquérir des entrées logiques et analogiques représentant l'état du système technique.
- d'interpréter, la signification de ces entrées.
- de calculer, mémoriser, récupérer des variables logicielles intermédiaires, d'agir sur des sorties logiques et analogiques en fonction des entrées et des calculs réalisés de manière à modifier le fonctionnement du système technique (commande moteur, affichage d'informations,…).
- de communiquer par des liaisons séries avec d'autres systèmes techniques et/ou un ordinateur. **[8]**

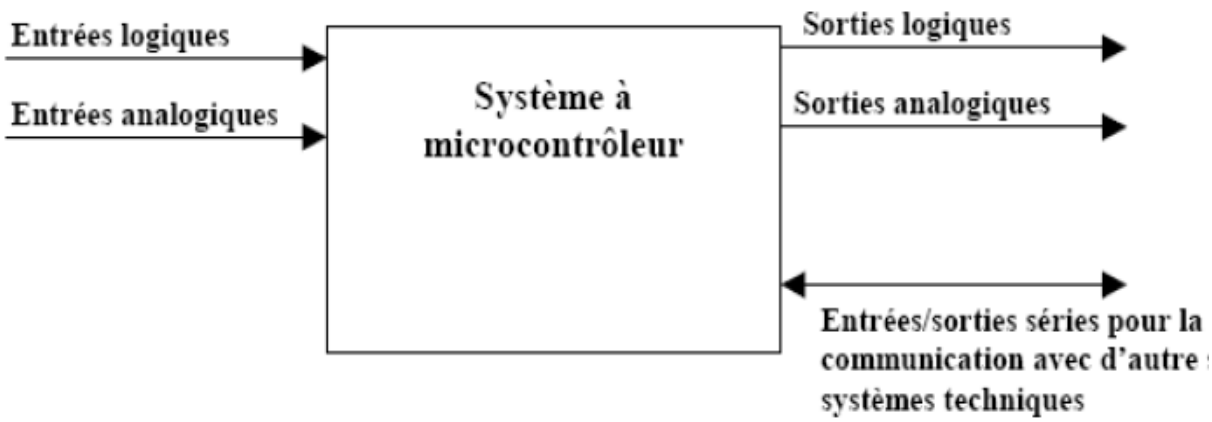


Fig.2.7 : rôle d'un système a microcontrôleur.

3.4-Le contrôle du microcontrôleur :

3.4.1-L'horloge du microcontrôleur :

Elle va donner la référence temporelle au microcontrôleur pour exécuter les instructions.

L'horloge d'un microprocesseur est souvent réalisée grâce à un Quartz. Il existe certains microcontrôleurs qui ont la possibilité de sélectionner une horloge interne (sans composants externes) ce qui permet d'utiliser les broches de l'horloge pour d'autres périphériques. **[8]**

3.4.2- Le chien de garde du microcontrôleur :

C'est une structure, qui peut être interne ou externe au microcontrôleur, qui permet de vérifier le bon déroulement du programme. Le microcontrôleur envoie des impulsions espacées de durées fixes au chien de garde. Tant que les impulsions espacées de durées fixes arrivent au chien de garde, tout se passe bien.

Par contre dès que le chien de garde détecte l'absence d'une impulsion (le programme est bloqué), il produit une mise à zéro du programme de gestion du système technique de manière à débloquer le programme. **[8]**

3.4.3-Le reset à la mise sous tension :

Tout microcontrôleur a besoin d'un temps minimum avant de pouvoir commencer à lancer le programme. Ce temps est donné par la documentation constructrice. Il faut par conséquent produire un signal de reset d'une durée supérieur à la mise sous tension. **[8]**

3.5-Microcontrôleur de la Famille Pic :

3.5.1-Définition d'un pic :

Un PIC (Programmable Interface Controller) est un microcontrôleur de MicrochipTechnology Un PIC est un composant dit RISC (Reduced Instructions Set Computer), ou encore composant à jeu d'instruction réduit. Ces microcontrôleurs sont conçus sur une architecture dite HAVARD, elle est basée sur deux bus, un pour les données et l'autre pour les instructions (bus de programme). **[9]**

3.5.2- Classification des PIC :

La famille des PIC de Microchip est subdivisée en 3 grandes, familles comportant chacune plusieurs références :

- **La famille Base-Line :** qui utilise des mots d'instructions de 12 bits.
- **La famille Mid-Range :** qui utilise des mots de 14 bits (et dont font partie les 16FXXX).
- **La famille High-End :** qui utilise des mots de 16 bits (18FXXX).

3.5.3- Identification des PIC :

Un PIC est généralement identifié par une référence de la forme suivante : xx(L) XXyy-zz

xx : famille du composant, actuellement « 12, 14, 16,17 et 18 »

L : tolérance plus importante de la plage de tension.

XX : type de programme : **C** : EPROM ou EEPROM ; **F** : flash

yy : identificateur.

zz : vitesse maximale du quartz de pilotage.

3.5.4-Le choix d'un PIC :

Le choix d'un pic est directement lié à l'application envisagée: Il faut dans un premier temps déterminer le nombre d'entrées/sorties nécessaires pour l'application. Ce nombre d'entrées/sorties nous donne une idée sur la famille du PIC. Il faut ensuite déterminer si l'application nécessite un convertisseur Analogique/ Numérique ce qui va centrer un peu plus vers le choix du PIC. La rapidité d'exécution est un élément important, il faut consulter les DATA-BOOK pour vérifier la compatibilité entre la vitesse maximale du PIC choisi et la vitesse max nécessaire au montage. La taille de la RAM interne et la présence ou nom d'une EEPROM pour mémoriser des données est également important pour l'application souhaitée. La longueur de programme de l'application détermine la taille de la mémoire programme du PIC recherché. Afin de choisir un PIC adéquat à notre projet, nous avons pensé à l'utilisation du PIC18F4550.

3.5.5-Principe de fonctionnement d'un pic :

Un microcontrôleur exécute des instructions, on définit « le cycle instruction » comme le temps nécessaire à l'exécution d'une instruction. A ne pas confondre avec la notion du cycle d'horloge qui correspond au temps nécessaire à l'exécution d'une opération élémentaire. Chaque cycle instruction dure 4 coup d'horloge, Une instruction est exécutée en deux phases :

- la phase de recherche du code binaire de l'instruction stocké dans la mémoire de programme.
- la phase d'exécution ou le code de l'instruction est interprété par le processeur et exécuté.

En effet, comme les instructions issues de la mémoire de programme circulent sur un bus différent de celui sur lequel circulent les données, ainsi le processeur peut effectuer la phase de recherche d'une instruction pendant qu'il exécute l'instruction précédente. **[10]**

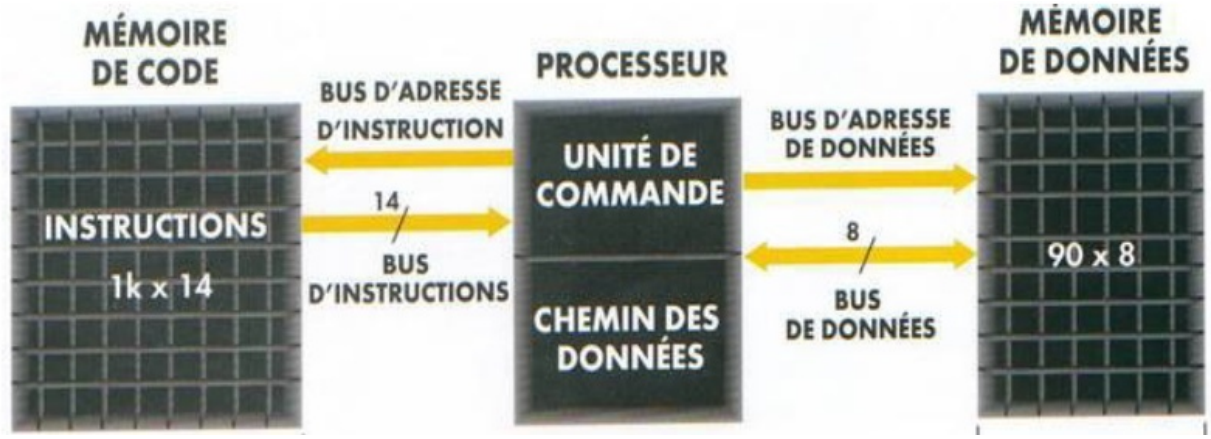


Fig. 2.8 : principe de fonctionnement des PIC avec représentation du chemin

De circulation des instructions et de données

3.5.6-Jeu d'instruction :

La figure suivante présente l'ensemble des 35 instructions d'un processeur PIC ; ce jeu est commun à la plupart de microcontrôleurs du fabricant. Pour avoir le détail de chaque instruction, il faut se reporter à la documentation du constructeur. **[10]**

Tab 2.1 : jeu d'instructions.

Mnémonique, Opérande	**Description**	**Code Opératoire (14 bits)** MSbLsb	**Statu**
Opérations sur des Octets			
ADDWF f, d	Add W and f	00 0111 dfff ffff	C, DC, Z
ANDWF f, d	AND W with f	00 0101 dfff ffff	Z
CLRF f, d	Clear f	00 0001 1fff ffff	Z
CLRW -	Clear W	00 0001 0xxx xxxx	Z
COMF f, d	Complément f	00 1001 dfff ffff	Z
DECF f, d	Décrément f	00 0011 dfff ffff	Z
DECFSZf, d	Décrément f Skip if 0	00 1011 dfff ffff	
INCF f, d	Incrément f	00 1010 dfff ffff	Z
INCFSZ f, d	incrément f Skip if 0	00 1111 dfff ffff	
IORWF f, d	Inclusive OR W with f	00 0100 dfff ffff	Z
MOVF f, d	Move f	00 1000 dfff ffff	Z
MOVWF f	Move W to f	00 0000 1fff ffff	
NOP -	No Operation	00 0000 0xx0 0000	
RLF f, d	Rotate Left f through Carry	00 1101 dfff ffff	C
RRF f, d	Rotate Right f throuch Carry	00 1100 dfff ffff	C
SUBWF f, d	Substract W from f	00 0010 dfff ffff	C, DC, Z
SWAPF f, d	Swap nybbleswith f	00 1110 dfff ffff	
XORWF f, d	Exclusive OR W with f	00 0110 dfff ffff	Z
Opérations sur des bits (registres)			
BCF f, b	Bit Clear f	01 00bb bfff ffff	
BSF f, b	Bit Set f	01 01bb bfff ffff	
BTFSC f, b	Bit Test f, Skip if Clear	01 10bb bfff ffff	
BTFSS f, b	Bit Test f, Skip if Set	01 11bb bfff ffff	
Opérations immédiates (W) et de contrôle			
ADDLWk	AddLiteral and W	11 111x kkkk kkkk	C, DC, Z
ANDLWk	AND Literal and W	11 1001 kkkk kkkk	Z

CALL k	Call subroutine	10	0kkk	kkkk	kkkk	
CLRWDT-	ClearWatchdogTimer	00	0000	0110	0100	T0n, PDn
GOTO k	Go To address	10	1kkk	kkkk	kkkk	
IORLWk	Inclusive OR literal with W	11	1000	kkkk	kkkk	Z
MOVLWk	Move Literal to W	11	00xx	kkkk	kkkk	
RETFIE -	Return frominterrupt	00	0000	0000	1001	
RETLWk	Return with literal in W	11	01xx	kkkk	kkkk	
RETURN-	Return fromSubroutine	00	0000	0000	1000	
SLEEP-	Go into standby mode	00	0000	0110	0011	T0n, PDn
SUBLW k	Substract W fromliteral	11	110x	kkkk	kkkk	C, DC, Z
XORLWk	Exclusive OR literal with W	11	1010	kkkk	kkkk	Z

Notations:

F: Registre (File Register)

d: Destination de l'opération (W ("0") ou f ("1"))
k : Valeur immédiate (Literal value)
b : Position du bit dans l'octet

4- Conclusion :

Les systèmes à microprocesseurs et à microcontrôleurs sont devenus assez simples à mettre en œuvre, mais seule une compréhension en profondeur permet de tirer parti au maximum de leur potentiel dans de nombreuses applications donc ils sont et continueront à être largement utilisés pour les applications de régulation, et de commande de processus et dans le monde de l'industrie, notamment dans les systèmes embarqués et ça grâce à Leur polyvalence, leur puissance et leur taille, les rendent intéressants.

CHAPITRE 3 : Circuit de commande de l’unité transfert

1-Introduction

Pour la simplicité et la facilité de la programmation, plusieurs langages ont été évolués dans le temps. En cherchant le compilateur le plus adapté aux microcontrôleurs PIC, on trouve le MikroBasic, MikroC et MikroPascal, et pour la simulation on trouve le logiciel (PROTEUS ISIS).

2-Définition du logiciel "PROTEUS" :

Proteus est une suite de logiciels permettant la CAO (conception assisté par ordinateur) électronique éditée par la société Lab Center Electronics. Proteus est composé de deux logiciels principaux : ISIS, permettant entre autres la création de schémas et la simulation électrique, et ARES, dédié à la création de circuits imprimés.

Proteus est un logiciel regroupant ISIS, ARES, PROSPICE et VSM. Tous ces modules sont destinés à l'électronique, grâce à ce logiciel, nous pouvons réaliser des schémas structurels et simuler.

Ce logiciel est bien connu et utilisé dans de nombreuse entreprises et organismes de formations, Outre la popularité de l'outil, Proteus possède d'autres avantages

- Pack contenant des logiciels facile et rapide à comprendre et utiliser
- Le support technique est performant
- L'outil de création de prototype virtuel permet de réduire les coûts matériel et logiciel lors de la conception d'un projet. **[11]**

2.1-Présentation générale:

Cette suite logicielle est très connue dans le domaine de l'électronique. De nombreuses entreprises et organismes de formation utilisent cette suite logicielle. Outre la popularité de l'outil, Proteus possède d'autres avantages.

- L'outil de création de prototype virtuel permet de réduire les coûts matériels.

- Logiciel lors de la conception d'un projet. **[11]**

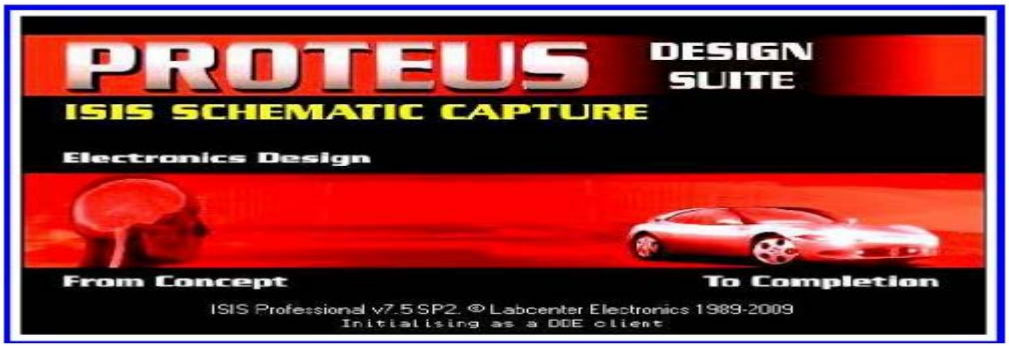


Fig.3.1 : LOGO logiciel Proteus.

2.2-ISIS:

Il est principalement connu pour éditer des schémas électriques. le logiciel permet également de simuler ces schémas ce qui permet de déceler certaines erreurs dès l'étape de conception. Indirectement, les circuits électriques conçus grâce à ce logiciel peuvent être utilisé dans des documentations car le logiciel permet de contrôler la majorité de l'aspect graphique des circuits. **[11]**

Fig.3.2 : Logo ISIS.

2.3-ARES :

Le logiciel ARES est un outil d'édition et de routage qui complète parfaitement ISIS. Un schéma électrique réalisé sur ISIS peut alors être importé facilement sur ARES pour réaliser le PCB de la carte électronique. Bien que l'édition d'un circuit imprimé soit plus efficiente lorsqu'elle est réalisée manuellement, ce logiciel permet de placer automatiquement les composants et de réaliser le routage automatiquement. **[11]**

3-Les éléments de circuit :

3.1-LE MICROCONTRÔLEUR PIC18F4550:

Le PIC18F est l'une des familles de microcontrôleurs PIC et les PIC18F4550 en sont membres. Le PIC18F4550 est l'un des nombreux microcontrôleurs avancés de l'ère de la micro-puce. Ce microcontrôleur est très célèbre entre amateur et débutant en raison de ses fonctionnalités et de ses fonctions ainsi que de l'intégration ADC et USB. Il existe différents packages tels que DIP, QPF et QPN de PIC18F4550 qui sont actuellement disponibles. **[12]**

3.1.1-Caractéristiques principales du pic18f4550 :

Les caractéristiques principales du 18F4550 sont résumées comme suit :

- 35 lignes d'entrées/sorties, répartis comme suit :
  - Un port de 7 lignes (port A)
  - Un port de 8 lignes (port B)
  - Un port de 8 lignes (port C)
  - Un port de 8 lignes (port D)

- Un port de 4 lignes (port E)

- Alimentation sous 5 Volts.
- Une mémoire de programme de type flash (32Ko (32768 Octets) mots de 16 bits).
- Une mémoire RAM utilisateur de 2048 Octets.
- Une mémoire EEPROM de 256 Octets emplacements.
- Une interface I2C pour la gestion d'un bus à 2 fils.
- Facilité de programmation comme tous les PIC.

Tab.3.1 : Le tableau présente tous les caractéristiques générales de PIC18F4550.

La caractéristique	**Valeur**
Fréquence Horloge MHz	48MHz
Mémoire programme FLASH	32Ko
Mémoire données (RAM)	2048 Octets
Mémoire EEPROM	256 Octets
Interruptions	20
Ports parallèles (Nombre des lignes)	A(7), B(8) C(8), D(8), E(4)
Timers	4 (3×16bit + 1×8bit)
CAN 10-bit	13
Instructions	75
Vitesse du CPU (MIPS)	12
Périphériques de communication Numérique	1-A/E/USART, 1-MSSP (SPI/I2C)
Capture / Comparateur / Périphériques PWM	1 CCP, 1 ECCP
Comparateurs	2
USB (canaux, vitesse, respect)	1, Full Speed, USB 2.0
Plage de tension de fonctionnement (V)	de 2 à 5,5
Température (C)	-40 À 85

3.1.2-Brochage du PIC18F4550 :

Il a 40 branches comme le montre la figure ci-dessus. **[12]**

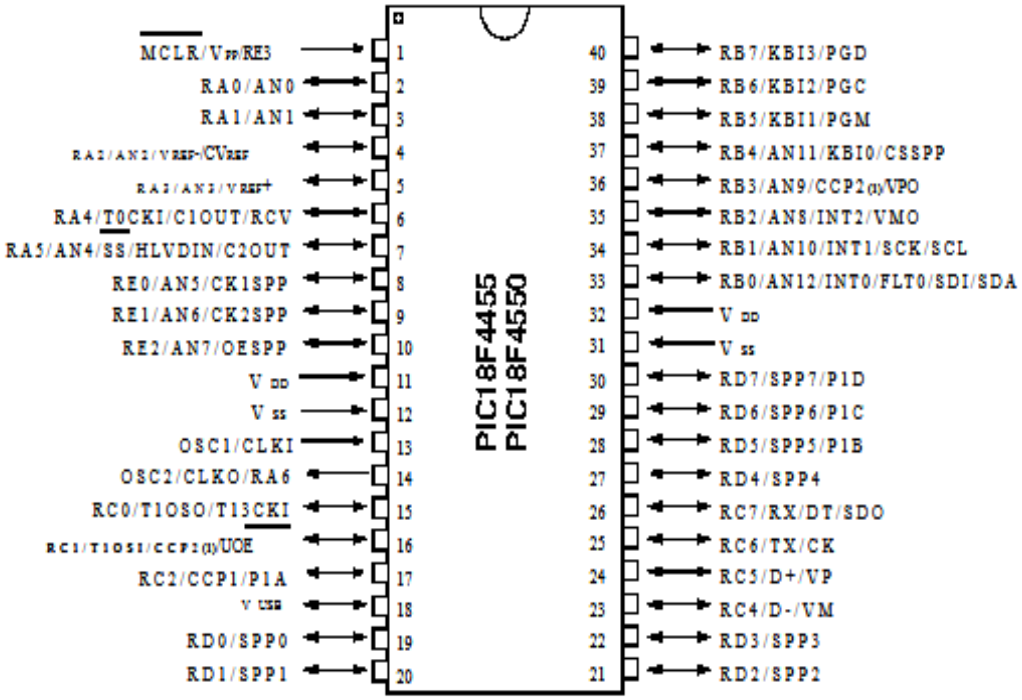


Fig.3.3: Brochage du pic 18f4550.

- On peut distinguer sur ce schéma :

  - L'alimentation : VDD (+5V) et Vss (0V)
  - Les bornes du quartz (oscillateur a quartz) : OSC1 et OSC2

- L'entrée RESET (MCLR: Master Clear)
- Les différents ports d'Entrées/Sorties : RAx, RBx, RCx, RDx, Rex

3.1.3-Architecture interne du PIC18F4550 :

Il est constitué des éléments suivants et représentée par la figure **(fig3.4)** :

- Quatre ports d'entrées/sorties

- Une unité arithmétique et logique (ALU)

- Quatre compteurs (Timers) Timer0, Timer1, Timer2, Timer3

- Un compteur de programme (program conter)

- Une mémoire RAM, 2048 octets

- Une mémoire EEPROM de 256 octets de données

- Un registre contenant le code de l'instruction à exécuter (IR)

- Convertisseur analogique numérique 13 canaux 10bit

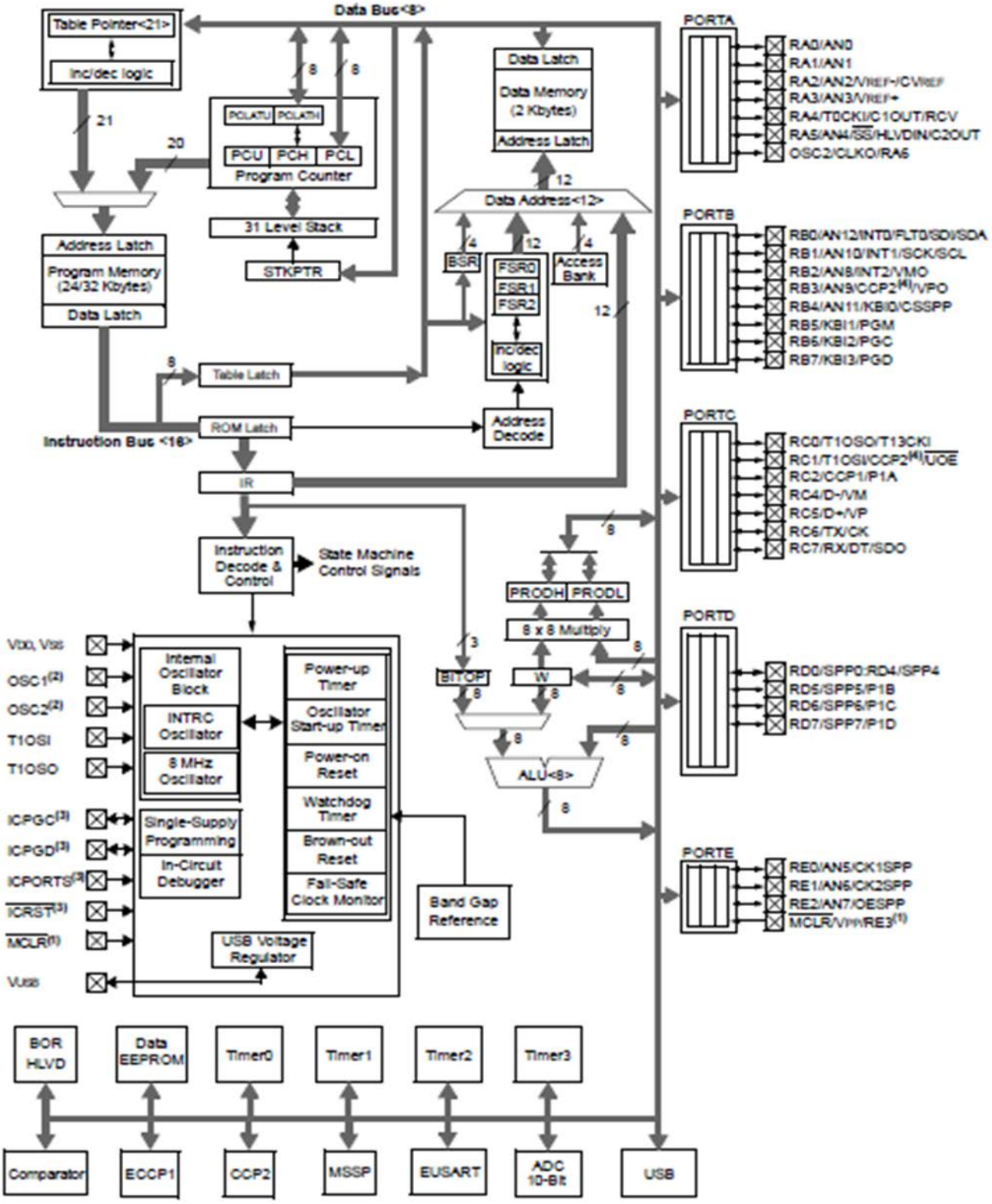


Fig.3.4: Architecture interne du PIC18F4550.

3.2-LE BUFFER 74HC245 :

3.2.1-Définition d'un buffer :

Le buffer est un composant électronique utilisé pour séparer la partie commande et la partie puissance , il permet aussi de raccorder plus d'utilisateurs sur la sortie d'un circuit. Et d'augmenter le nombre d'entrée/sortie d'un microcontroleur. **[13]**

Et pour réaliser notre projet, nous avons besoin de onze buffers 8 bits à trois états,huit buffers pour les entrées et trois buffers pour commander les sortie ; et notre choix s'est porté sur le buffer 74HC245

3.2.2-Déscription du buffer 74hc245 :

Le 74HC245 est un émetteur-récepteur octal , il contient 20 pins doté de sorties compatibles avec le bus à 3 états non inverseurs dans les deux sens d'envoi et de réception. Le buffer 74hc245 dispose d'une pin (OE) pour une mise en cascade facile et l'activation du buffer et d'une autre pin (DIR) pour le contrôle de la direction. OE contrôle les sorties afin que les bus soient isolés efficacement.

Le buffer 74HC245 avec ça possibilité d'augmenter les nombre d'entrée sortie afin d'atteindre le nombre existant dans notre système est une solution fiable pour permettre de réaliser et simuler notre application. **[14]**

3.2.3-Brochage du 74hc245 :

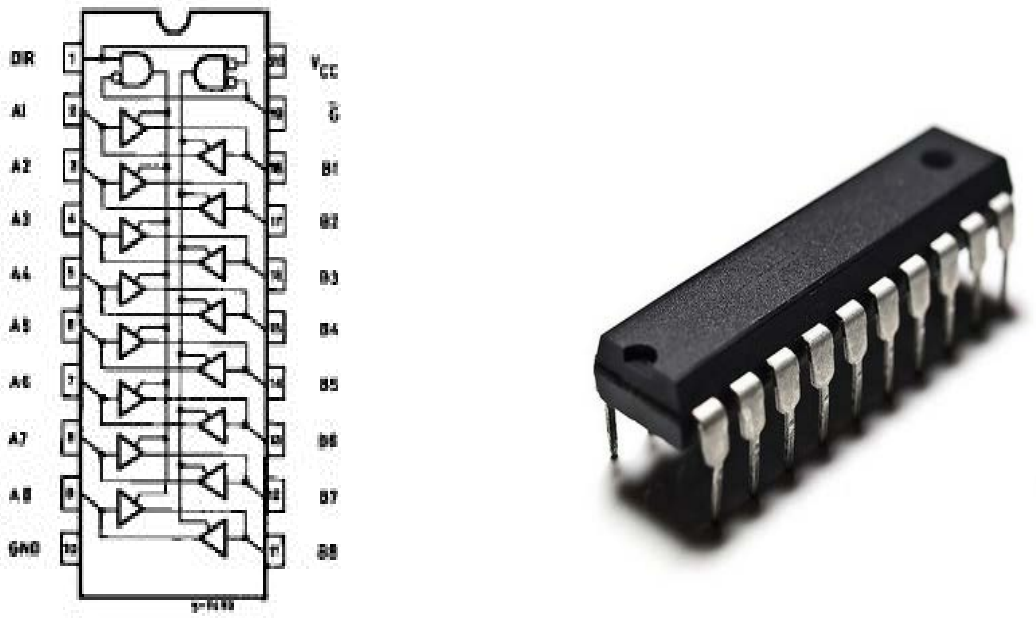


Fig.3.5 : Brochage du buffer 74hc245.

3.2.4-Description des pins :

Tab.3.2 : Description des pins du buffer 74hc245.

Symbole	Pin	Description
DIR	1	contrôle de direction
A0, A1, A2, A3, A4, A5, A6, A7	2, 3, 4, 5, 6, 7, 8, 9	Entrées/sorties
GND	10	0 volt
B7, B6, B5, B4, B3, B2, B1, B0	11, 12, 13, 14, 15, 16, 17,18	Entrées/sorties
OE	19	entrée de validation de sortie
VCC	20	tension d'alimentation

3.3-LE DECODEUR:

Le décodeur est composant électronique, son rôle est de sélectionner entre autres, une adresse précise et aussi utilisé pour contrôler ou commander les éléments électroniques d'un circuit.

3.3.1-Le décodeur 74LS138 :

Le 74LS139 est un décodeur 3 vers 8, il a trois entrées de sélection (A, B, et C), huit sorties (Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7), il a aussi trois entrées d'activation, deux actives High (E1 et E2) et un actif Bas (E3) Cette fonction d'activation multiple permet un parallèle facile. **[15]**

Tab.3.3 : table de vérité de décodeur 74HC138.

E1	E2	E3	A	B	C	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

3.3..2-Le décodeur 74LS139 :

Le 74LS139 est un décodeur 2 vers 4 En tenant en considération les caractéristique de décodeur et le besoin de notre application, on utilisé le 74LS139 Pour gérer et piloter les trois buffers des sortie. **[16]**

Tab.3.4 : table de vérité de décodeur 74LS139.

E	A	B	Y1	Y2	Y3	Y4
1	**x**	**x**	**1**	**1**	**1**	**1**
0	**0**	**0**	**0**	**1**	**1**	**1**
0	**0**	**1**	**1**	**0**	**1**	**1**
0	**1**	**0**	**1**	**1**	**0**	**1**
0	**1**	**1**	**1**	**1**	**1**	**0**

3.4-Le registre mémoire 74198 :

3.4.1-Description générale :

Le registre 74198 c'est un registre a décalage constitué de 8 bascules commandées par le même signal d'horloge, afin de stocker et de déplacer les données à 8 bits qu'il reçoit à ses entrées.

Il dispose de modes de chargement parallèle asynchrone, de maintien, de déplacement vers la droite et vers la gauche, déterminés par l'entrée select (s0, s1). Les changements d'état sont initiés par le front montant de l'horloge, une entrée de réinitialisation asynchrone (MR) remplace toutes les autres entrées et efface le registre. **[17]**

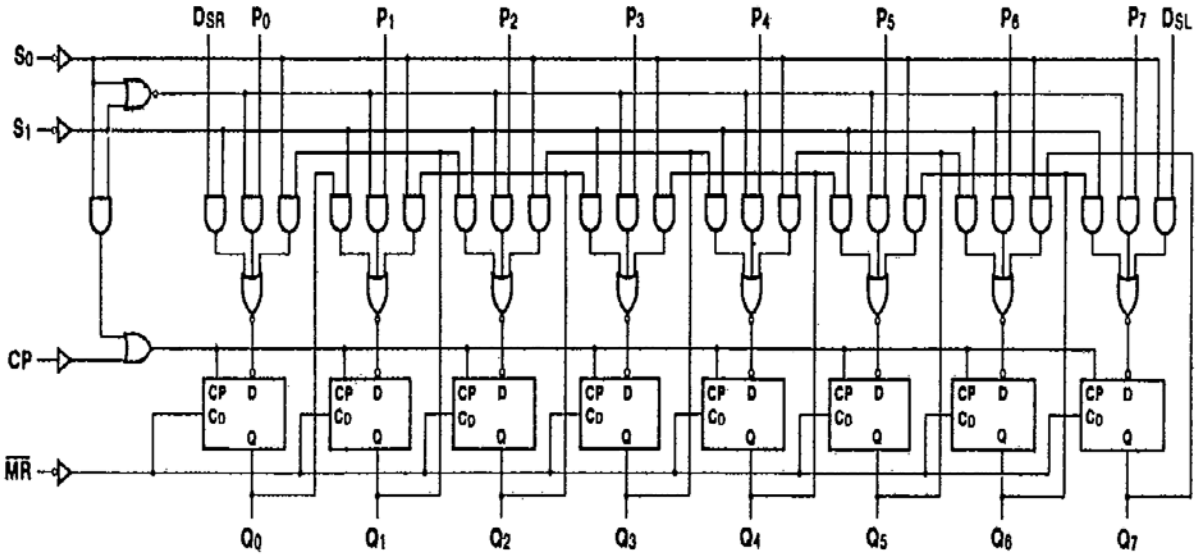


Fig.3.6 : Le diagramme logique du registre 74198.

3.4.2-Brochage du registre :

Pin	Description
S0, s1	Sélectionner le mode des entrées
P0…p7	Les entrées
DSR	Décalage à droite
DSL	Décalage à gauche
CP	Signale d'horloge
MR	Entrée de réinitialisation
Q0…Q7	Les sorties

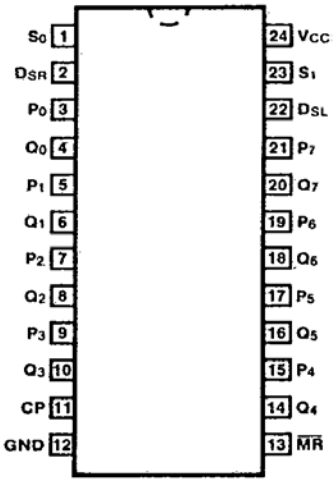


Fig.3.7 : brochage du registre 74198.

3.5- Répartition des entrées /sorties :

3.5.1-Les entrées :

Les entrées sont répartit sur 8 buffers d'entrées comme le montre la tables suivante **:**

Tab.3.5: répartition des entrées sur les buffers.

Buffers d'entrées	Entrée et bits							
	b7	b6	b5	b4	b3	b2	b1	b0
BE1	C2	B3	B2	B1	S4	S3	S2	S1
BE2	C1	S5	S6	S7	S8	B4	B5	Cpe1
BE3	Cpe2	B12	B11	B7	B6	S17	S11	S9
BE4		Cv2	Cv1	B9	B8	S13	S12	S110
BE5	S40	S29	Cps2	Cps1	Sa2	Sa1	Ea2	Ea1
BE6	S37	S25	S35	S23	S33	S21	S31	S19
BE7		S45	S44	S43	S42	S41	S39	S27
BE8						Arr	B14	B13

3.5.2-Les sortie :

Les sorties sont répartie sur trois buffers de sortie comme suit :

Tab.3.6: répartition des sorties sur les buffers.

Buffers d'entrées	Entrée et bits							
	b7	b6	b5	b4	b3	b2	b1	b0
BE1	KM32	KM31	KMRG	KM21	KM22	KMRD	KM12	KM11
BE2	KMTS2	KMTS1	EV4	EV3	KMTE2	KMTE1	KMP12	KMP11
BE3	KM42	KM41	KMP22	KMP21	EV2	EV1		

4- Réalisation du circuit de commande sur ISIS Proteus :

Pour réaliser le circuit de commande sur proteus on doit d'abord choisir nos composants dans les différentes bibliothèque du logiciel, en suite, il faut respecter l'emplacement des broches pour assurer le bon fonctionnement du système. Et établir les liaisons entre les éléments du circuit et les entrées/sorties.

Dans la mise en œuvre de notre circuit, on s'est appliqué sur la simplification des brochages entre les différents pins des composants pour éviter d'encombrer le circuit, pour cela on a recouru à l'utilisation des éléments IN/OUT du mode terminal, afin de diminué le nombre des fils qui relié tout les éléments de circuit.

Pour commencer on a utilisé le port D de pic comme entrée pour recevoir l'information venu de nos buffers d'entrées, après cela on a réservé le port A pour contrôler et commander les deux décodeurs, ce qui permet de gérer la sélection des buffers comme indiqué sur les tables **(Tab.3.1 et Tab.3.2)**, et aussi généré un signal d'horloge dans la broche RA5 pour les registre mémoire et on a défini le port B comme sortie pour transfère de l'information vers les buffers de sortie.

- Sélection des buffers d'entrées :

Tab.3.7 : sélection des buffers d'entrées.

Broches de port A			**Décodeur**			**Buffers d'entrées sélectionnés**
RA4	**RA3**	**RA2**	**C**	**B**	**A**	
0	**0**	**0**	**0**	**0**	**0**	**BUFFER BE1**
0	**0**	**1**	**0**	**0**	**1**	**BUFFER BE2**
0	**1**	**0**	**0**	**1**	**0**	**BUFFER BE3**
0	**1**	**1**	**0**	**1**	**1**	**BUFFER BE4**
1	**0**	**0**	**1**	**0**	**0**	**BUFFER BE5**
1	**0**	**1**	**1**	**0**	**1**	**BUFFER BE6**
1	**1**	**0**	**1**	**1**	**0**	**BUFFER BE7**
1	**1**	**1**	**1**	**1**	**1**	**BUFFER BE8**

- Sélection des buffers de sortie **:**

Tab.3.8 : sélection des buffers de sorties.

Broches de port A		**Décodeur**		**Buffers de sorties**
RA0	**RA1**	**A**	**B**	
0	**0**	**0**	**0**	**BUFFER BS1**
0	**1**	**0**	**1**	**BUFFER BS2**
1	**0**	**1**	**0**	**BUFFER BS3**

En ce qui concerne la simulation de notre circuit, les entrées /sorties définies dans la table mnémonique **(tab1.0)** dans le chapitre 1, on les a met en œuvre à travers un ensemble de contacteurs et des led pour les actionneurs respectivement.

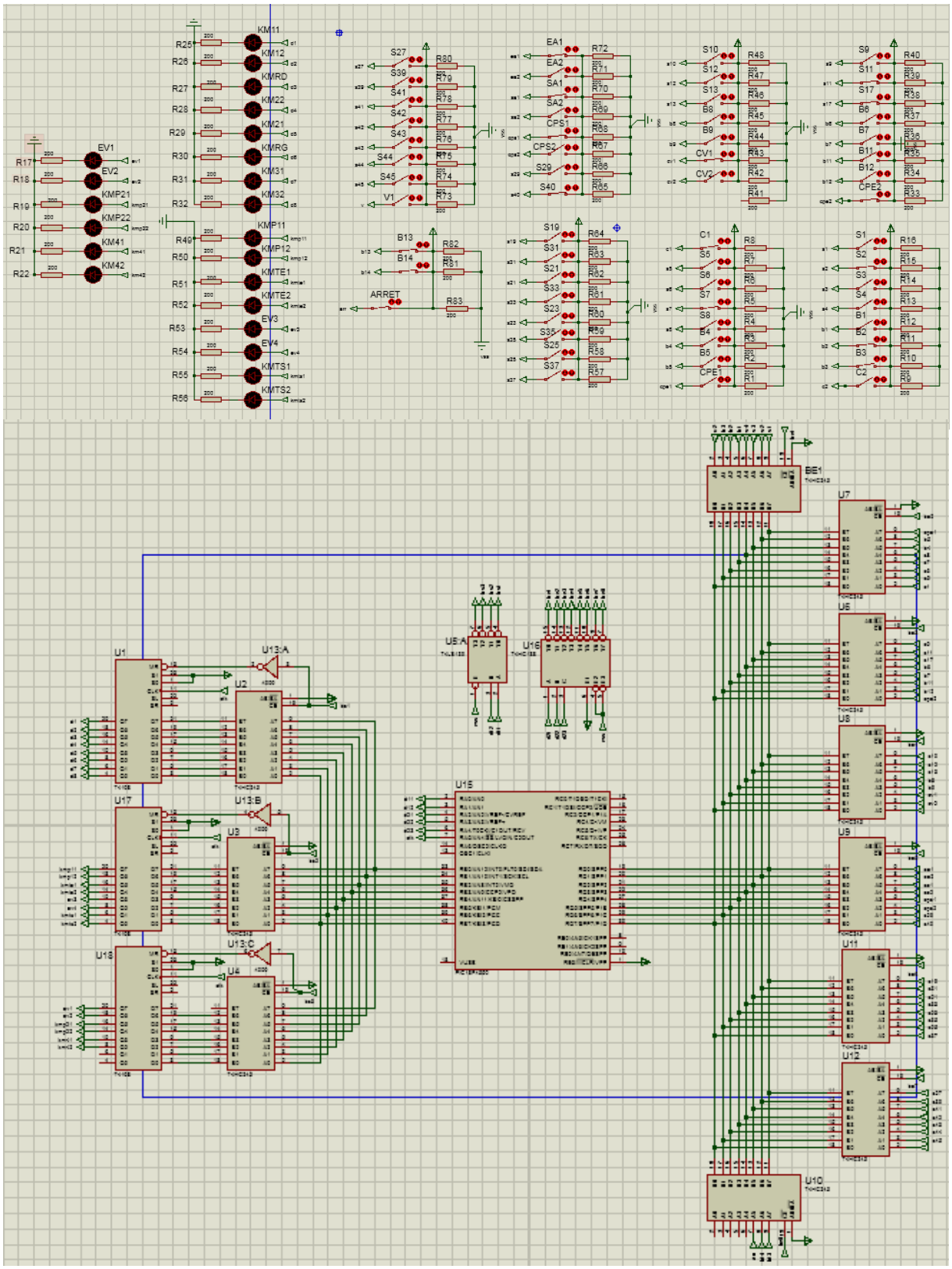


Fig.3.8 : le schéma de circuit de command sur Proteus.

5- Conception du circuit imprimé :

Après y avoir terminé le circuit ISIS on transfert le schéma vers ARES pour la conception de circuit imprimé de notre carte de commande représenté sur la figure **(fig.3.9).**en utilisant le routage automatique. Tandis que la figure **(fig.3.10)** représente la vue de dessus du circuit.

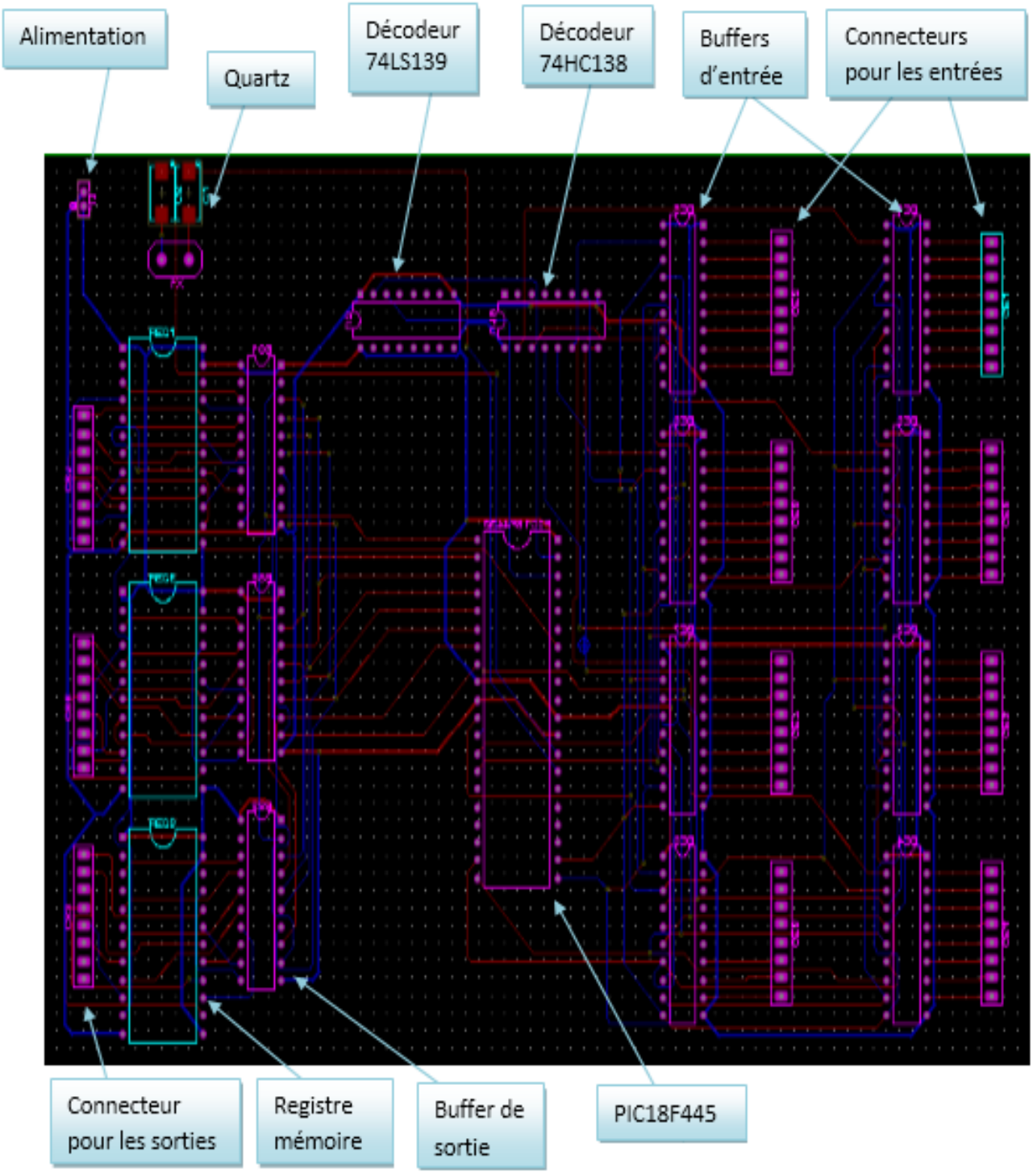


Fig.3.9: le circuit imprimé du la carte de commande.

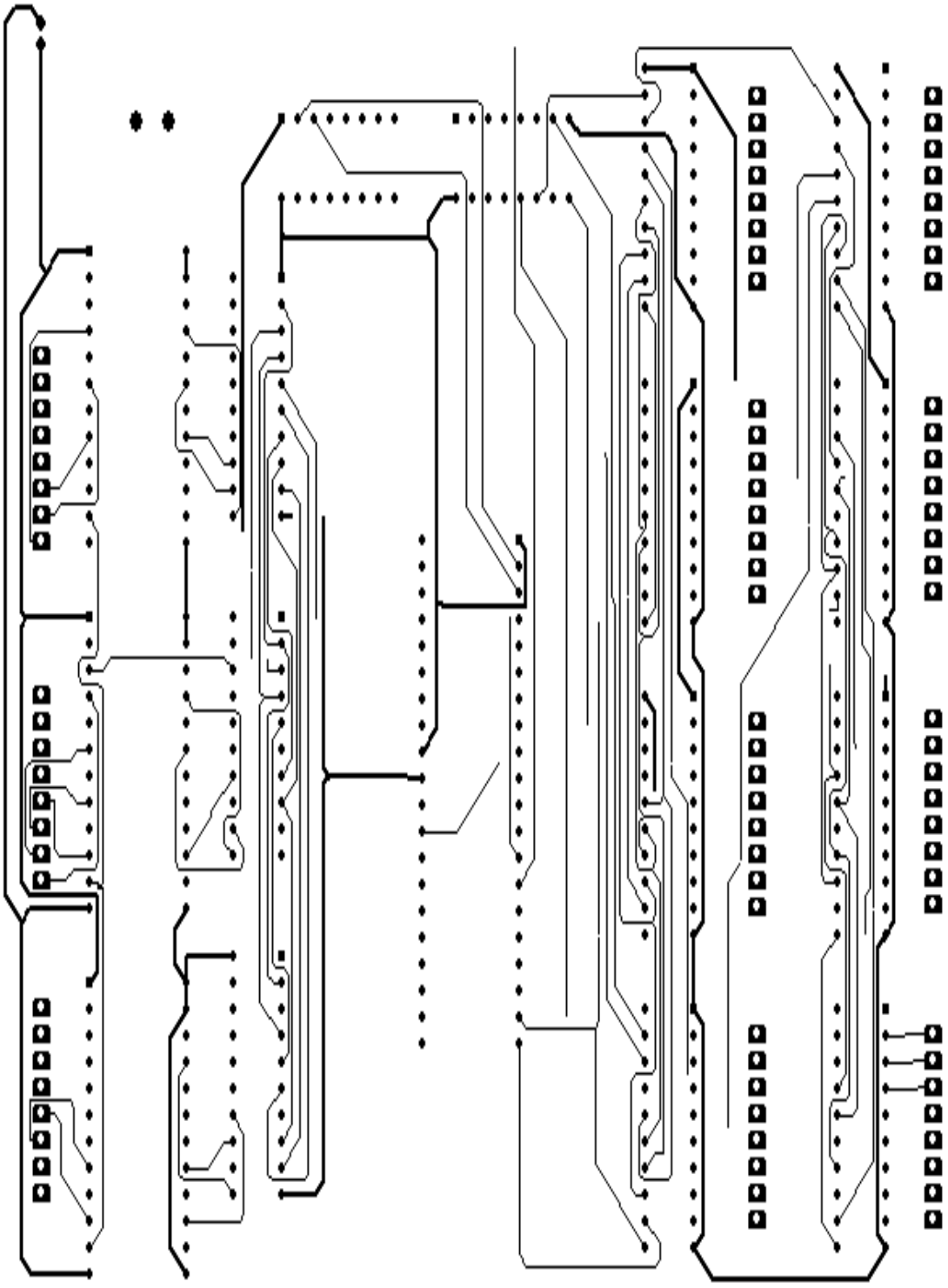


Fig.3.10 : le circuit imprimé face bas.

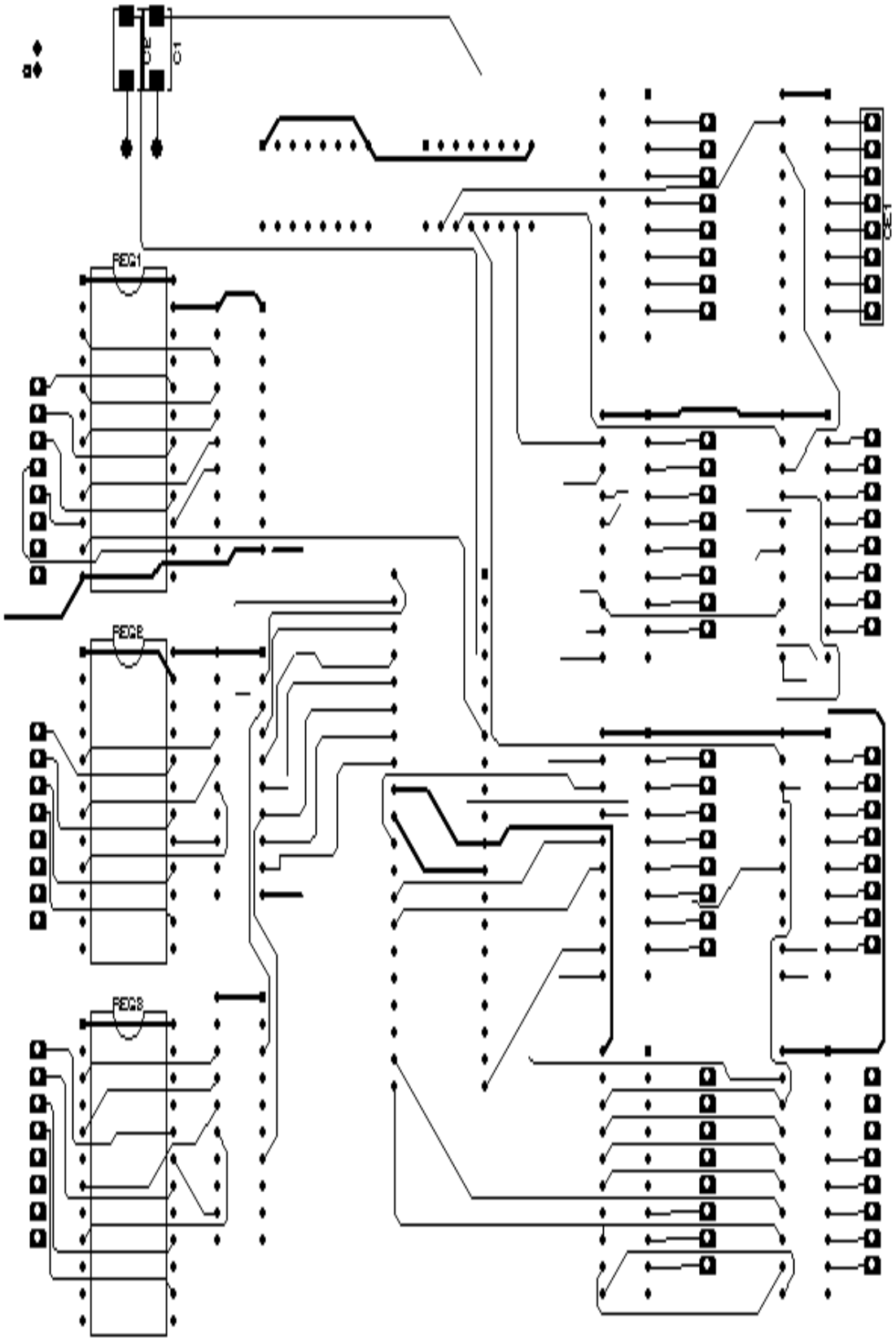


Fig.3.11 : Circuit imprimé vu de haut.

6-Visualisation 3D de la maquette de notre circuit :

Après y avoir fini avec le schéma vers ARES on lance le simulateur 3D pour visualiser la maquette de notre carte de commande, comme le montre la figure **(Fig.3.11).**

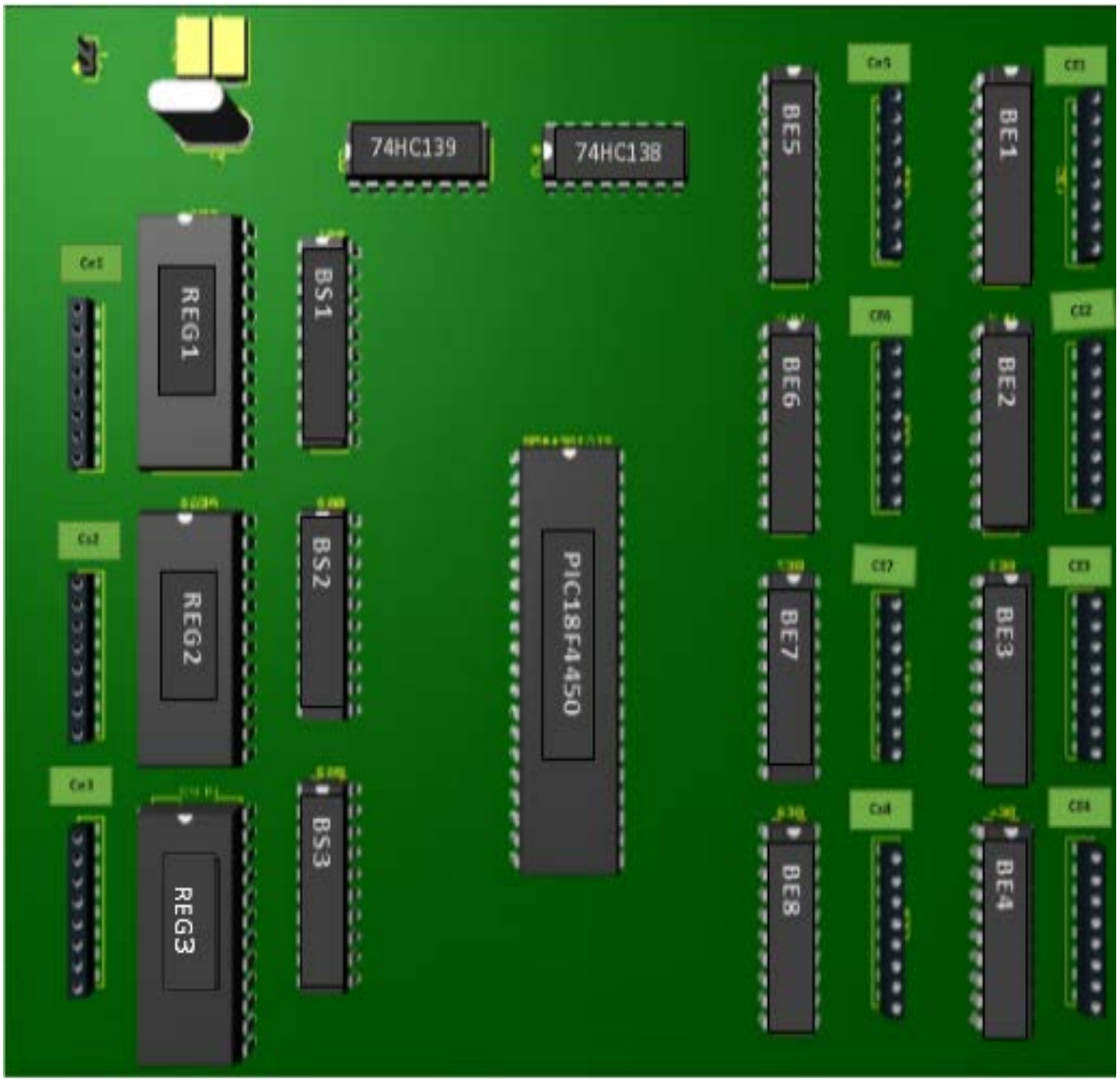


Fig.3.12 : maquette 3D de la carte de commande.

7- Conclusion :

Le logiciel Proteus non seulement nous offre la possibilité de tester n'importe quel système avant de le réaliser sur le plan réel, de corriger les problèmes de fonctionnement et les erreurs de programmation. En effet, il nous facilite la réalisation des cartes de commandes et le câblage des éléments de leurs circuits électriques. Proteus possède un pack contenant des logiciels facile et rapide à comprendre et à utiliser, et l'outil de création de maquette virtuel permet de réduire les coûts matériels et logiciels lors de la conception d'un projet.

CHAPITRE 4 : Programmation et simulation de la carte de commande

1-Introduction :

Après la modélisation de notre système et la réalisation de circuit de commande sur Proteus, on se porte sur la mise en œuvre de programme de simulation.

Pour la programmation des pics il existe plusieurs logiciels avec des langages différents, qui permettent tous de réaliser notre projet grâce à une prise en main très intuitive à l'image de Mplab, C++, Assembleur…etc.

Pour mettre en œuvre cela on s'est appuyer sur le logiciel de programmation **'MikroC pro for pic',** notre choix s'est porté sur cette utile par apport à ça simplicité d'utilisation et ces nombreuses amélioration de fonctionnalité.

2-Compilateur MikroC PRO pour PIC:

La nouvelle version appelée mikroC PRO dispose de très nombreuses améliorations du compilateur mikroC : nouvelles variables utilisables, nouvelle interface IDE, amélioration des performances du linker et de l'optimisateur, cycle de compilation plus rapide, code machine généré plus compact (jusqu'à 40 % suivant les cas), nouveaux PIC supportés, environnement de développement encore plus ergonomique, nouveaux exemples d'applications, etc. **[18]**

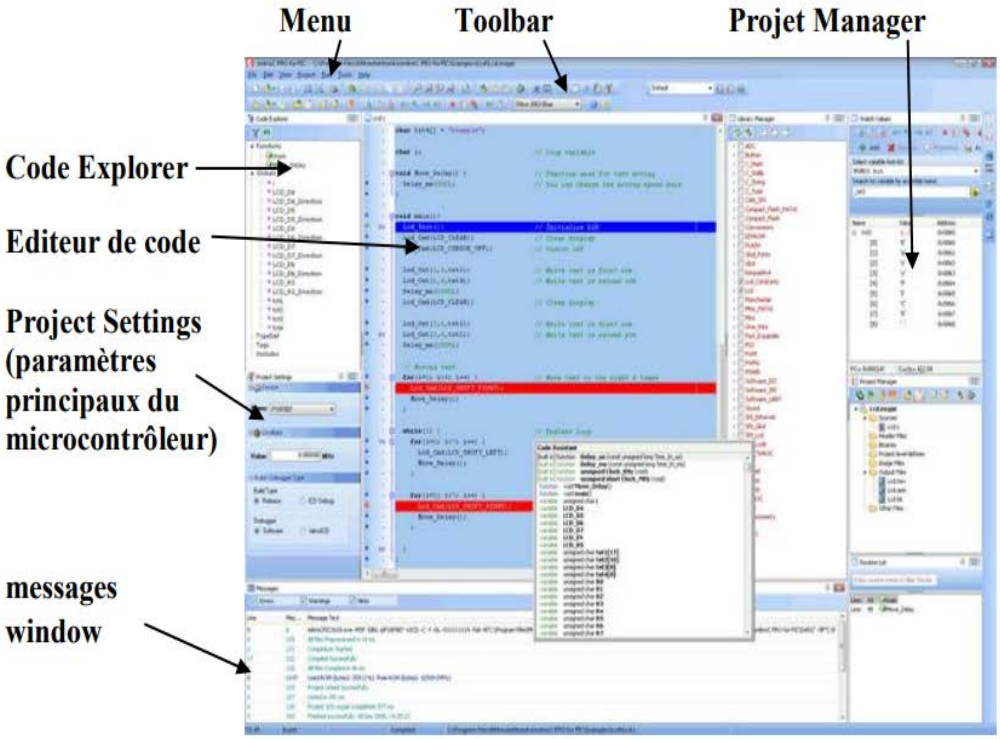


Fig.4.1: l'environnement IDE de compilateur MikroC.

3-Langage et compilateur MikroC pour PIC:

Le langage mikroC pour PIC a trouvé une large application pour le développement de système embarqués sur la base de microcontrôleur. Il assure une combinaison de l'environnement de programmation avancée IDE (Integrated Development Environment), et d'un vaste ensemble de bibliothèques pour le matériel, de la documentation complète et d'un

grand nombre des exemples. Le compilateur mikroC pour PIC bénéficie d'une prise en main facile et d'une ergonomie sans faille. Ses très nombreux outils intégrés (mode simulateur, terminal de communication Ethernet, terminal de communication USB, gestionnaire pour afficheurs 7 segments, analyseur statistique, correcteur d'erreur, explorateur de code, modeDébugICD...) associé à sa capacité à pouvoir gérer la plupart des périphériques rencontrés dans l'industrie (Bus I2C™, 1Wire™, SPI™, RS485, Bus CAN™, USB, gestion de cartes compact Flash et SD™/MMC™, génération de signaux PWM, afficheurs LCD alphanumériques et graphiques, afficheurs LEDs à 7 segments, etc...) en font un outil de développement incontournable pour les systèmes embarqués, sans aucun compromis entre la performance et la facilité de débogage.**[18]**

4-Editeur de code :

L'éditeur de code est le même que toute éditeur de texte standard pour l'environnement de Windows, y compris Copie, Coller, Annuler les actions etc… Il possède en plus des ressources comme suit :

- Coloration syntaxique réglable
- Assistant de code
- Assistant de paramètre
- Mise en forme automatique. **[18]**

5-Application :

Dans cette partie on se focalise sur la programmation de notre pic18f4550 en utilisant le mikroC pour pic à fin de réalisé le fonctionnement cité dans le cahier de charge.

5.1-Les étapes suivies dans la programmation :

5.1.1-Déclaration des variables :

Durant l'exécution du programme, on a besoin de garder quelques conditions activées pour les vérifier dans les étapes suivantes. Et pour ca on doit déclarer ces variables sous le type **«bit »** et une variable de type **« char »** pour mémoriser le continu de port B avant de le mettre à zéro en cas d'arrêt d'urgence **,** et on doit aussi initialiser les deux compteurs qu'on a déclarés sous le type de variable **« Int ».**

```
bit s1,s2,s3,s4,s5,s6,s7,s13,s17,s40,s11;
bit a1,a2,a3,a4,a5,a11,a12;
bit c1,c2;
bit cep1,cep2;
 int j;
```

Fig.4.2 : Déclaration des variables.

5.1.2-Configuration du pic :

Avant la saisie de notre programme, il faut configurer le pic en spécifient les entrée, les sorties et l'utilisation des valeurs numériques seulement en mettant les quatre premiers bits du registre ADCON1 à 1.

```
TRISD=0XFF; // portD en entrée
TRISA=0X00; // le portA en sortie
TRISB=0X00; //le portB en sortie
ADCON1=0X0f; // utilisation des valeurs numeriques seulement
```

Fig.4.3: configuration du pic.

5.1.3-Simplification des étapes:

Dans le but de simplifié notre programme, on a rassemblé toutes les instructions qui nos permet de réaliser chaque actions qui se répète plusieurs fois dans le programme puis on les a met sous forme de fonctions comme le montre la figure ci-dessus :

```
void avance m1(){
      PORTB.f0=1;     // actionner le contacteur KM11(led KM11)
      PORTA.f5=1;     // front montant pour le registre memoire
      delay ms(10);   // attente
      PORTA.F5=0;     // front descendant
      return;
}
```

Fig.4.4: exemple instructions regroupé pour réalisé une étape.

5.1.4-Sous programmes :

Les étapes déclarées au début de notre programme sous forme de fonctions on leur fait appel dans des sous programmes exécutent des taches, qui est le moyen le plus facile de bien organisé et simplifié et facilité la compréhension du programme.

5.1.5-L'arrêt d'urgence :

Pour assurer la sécurité de personnel et de matériels utilisés dans l'unité transfert, on a recouru à cette option qui permet de mettre le système au repos en cas de problème ou disfonctionnement.

```
void arret(){
    PORTB=0x00;       // mettre le portB a zero
    PORTA.f5=1;       //front montant pour le registre memoire
    delay ms(10);     // attente
    PORTA.f5=0;       // front descendant
    return ;
}

void arret d urg(){
  //  selection du buffer BE8
       PORTA.f2=1;
       PORTA.f3=1;
       PORTA.f4=1;
      while(1)
 {    if (PORTD.f2==0)    // arret d'urgence activée
   { Ac=PORTB;           // memorisation du portB
     arret();           // mettre le portB a zero
     break;}
     else if(PORTD.f2==1) // arret d'urgence non activée
   {  Ac=PORTB;           // memorisation du portB
    break;}
 }
   while(1)
 {    if(PORTD.f2==1)
    {PORTB=Ac;           // mettre le portB a l'etat normale
     PORTA.F5=1;         // front montant pour le registre memoire
     delay ms(10);       // attente
     PORTA.f0=0;         // front descendant
    break;}
 }
       return;
```

Fig.4.5 : programme de l'arrêt d'urgence.

5.1.6-Programme principal :

Le programme principal fait avec le réassemblage des trois sous-programmes, et chacun est conditionné selon le cahier de charge.

5.2-Le programme MikroC qui correspond au cahier de charge :

Nous avons écrit un programme qui permet au microcontrôleur PIC18F4550 de bien commander et contrôler notre système, Apres avoir terminé le programme et s'assuré qu'il n'y ait pas d'erreurs. Nous cliquons sur le (build projet). Le MikroC va compiler automatiquement le programme écrit en langage MikroC en programme langage machine (Numérique) qui s'appelle le fichier hexadécimal avec l'extension HEX(FILE.HEX).

```
          // Programme MikroC de la carte de commande

bit s13,s17,s11;
bit a4,a5,a11,a12;     // déclaration des variables
int j,i;
char Ac;

void avance_m1(){
       PORTB.f1=0;      // desactiver le contacteur MK12
       PORTB.f0=1;     // actionner le contacteur KM11(led KM11)
       PORTA.f5=1;     // front montant pour le registre memoire
       delay_ms(10);   // attente
       PORTA.F5=0;     // front descendant
       return;
}

void recule_m1(){
       PORTB=0x02;     // actionner le contacteur KM12(led KM12)
       PORTA.f5=1;     //front montant pour le registre memoire
       delay_ms(10);   // attente
       PORTA.f5=0;     // front descendant
       return;
}

void avance_m2(){
        PORTB.f3=0;      // desactiver le contacteur KM22(led KM22)
        PORTB.f4=1;      // actionner le contacteur KM21(led KM21)
        PORTA.f5=1;      // front montant pour le registre memoire
        delay_ms(10);    // attente
        PORTA.f5=0;      // front descendant
        return;
}

void recule_m2(){
        PORTB.f4=0;      // actionner le contacteur KM21(led KM21)
        PORTB.f3=1;      // actionner le contacteur KM22(led KM22)
        PORTA.f5=1;      //front montant pour le registre memoire
        delay_ms(10);    // attente
        PORTA.f5=0;      // front descendant
        return;
}

void avance_m3(){
   PORTB.F7=0;          // desactiver le contacteur KM32(led KM32)
   PORTB.F6=1;          // actionner le contacteur KM31(led KM31)
   PORTA.F5=1;          //front montant pour le registre memoire
   delay_ms(10);        // attente
   PORTA.F5=0;          // front descendant
   return ;
}

void recule_m3(){
   PORTB.F6=0;        // désactiver le contacteur KM31(led KM31)
   PORTB.f7=1;        // actionner le contacteur KM32(led KM32)
   PORTA.F5=1;        //front montant pour le registre memoire
   delay_ms(10);      // attente
   PORTA.F5=0;        // front descendant
   return ;
}

void avance_m4(){
```

```
     PORTB.f5=0;     // désactiver le contacteur KM42(led KM42)
     PORTB.f4=1;     // actionner le contacteur KM41(led KM41)
     PORTA.f5=1;     //front montant pour le registre memoire
     delay_ms(10);   // attente
     PORTA.f5=0;     // front descendant
     return;
}

void recule_m4(){
     PORTB.f4=0;     // désactiver le contacteur KM41(led KM41)
     PORTB.f5=1;     // actionner le contacteur KM42(led KM42)
     PORTA.f5=1;     //front montant pour le registre memoire
     delay_ms(10);   // attente
     PORTA.f5=0;     // front descendant
     return;
}

void recule_trans2(){
PORTB.f6=0;         // désactiver le contacteur KMTS1
PORTB.F7=1;         //activer le contacteur KMTS2
PORTA.f5=1;         //front montant pour le registre memoire
delay_ms(10);       // front descendant
PORTA.f5=0;
return;
}

void recule_trans1(){
PORTB.f2=0;         // desactiver le contacteur KMTE1(avance transporteur d
'entrée)
PORTB.F3=1;         //activer le contacteur KMTE2( recule transporteur d'en
trée)
PORTA.f5=1;         //front montant pour le registre memoire
delay_ms(10);       // attente
PORTA.f5=0;         // front descendant
return;
}

void avance_trans2(){
PORTB.f7=0;         // desactiver le contacteur KMTS2(recule transporteur d
e sortie)
PORTB.F6=1;         //activer le contacteur KMTS1( avance transporteur de s
ortie)
PORTA.f5=1;         //front montant pour le registre memoire
delay_ms(10);       // attente
PORTA.f5=0;         // front descendant
return;
}

void avance_trans1(){
PORTB.f3=0;         // desactiver le contacteur KMTE2(recule transporteur d
'entrée)
PORTB.F2=1;         //activer le contacteur KMTE1( avance transporteur d'en
trée)
PORTA.f5=1;         //front montant pour le registre memoire
delay_ms(10);       // attente
PORTA.f5=0;         // front descendant
return;
}

void arret(){
    PORTB=0x00;      // mettre le portB a zero
```

```
    PORTA.f5=1;      //front montant pour le registre memoire
    delay_ms(10);    // attente
    PORTA.f5=0;      // front descendant
    return ;
}

void arret_d_urg(){
  //  selection du buffer BE8
       PORTA.f2=1;
       PORTA.f3=1;
       PORTA.f4=1;
      while(1)
 {   if (PORTD.f2==0)   // arret d'urgence activée
   { Ac=PORTB;          // memorisation du portB
     arret();          // mettre le portB a zero
     break;}
     else if(PORTD.f2==1) // arret d'urgence non activée
   {  Ac=PORTB;           // memorisation du portB
    break;}
 }
   while(1)
 {   if(PORTD.f2==1)
    {PORTB=Ac;          // mettre le portB a l'etat normale
     PORTA.F5=1;        // front montant pour le registre memoire
     delay_ms(10);      // attente
     PORTA.f0=0;        // front descendant
    break;}
 }
       return;
}


 //   dechargement du chariot sur la voie de séchage et retour des transpor
teurs
void dechargement(){
  PORTA=0x0E;         // selectionner les buffers BE4 ET BS3
  arret();           //   desactiver les bloqueurs
      for(i=1;i<=3;i++)     // 3 cycle pour le verin porté sur T1
    {while(1)
  {   if (PORTD.f5==1) // C1=1
   { PORTB.f1=0;      // desactiver EV2
     PORTB.F0=1;      // activer EV1
     PORTA.f5=1;      //front montant pour le registre memoire
     delay_ms(10);    // attente
     PORTA.f5=0;      // front descendant
     break;}
   }
     while(1)
 {  if (PORTD.f6==1)    //C2=1
     {PORTB.F0=0;      // desactiver EV1
     PORTB.f1=1;       // activer EV2
     PORTA.f5=1;       //front montant pour le registre memoire
     delay_ms(10);     // attente
     PORTA.f5=0;       // front descendant
    break;}
 }
 }
     while(1)
 {
 PORTA=0x1D;          // selectionner les buffers BE8 et BS2
   if (PORTD==0x07)   // b13=b14=1
```

```
   {PORTB=0x30;       // desactiver les bloqueurs
    PORTA.f5=1;       // front montant pour le registre memoire
    delay_ms(10);     // attente
    PORTA.f5=0;       // front descendant

    break;}
 }
      while(1)
 {   PORTA=0x11;                    // selectionner le buffer BE5
    if(PORTD.f1==1&PORTD.f3==1)  // bloqueurs desactiver
   {recule_trans1();             // retour du transport d'entrée a la posit
io initiale
    avance_trans2();             // retour du transport d'entrée a la voie
de sortie
    PORTA=0x09;                 // selection du buffer BE3
    break;}
 }
   while(1)
{   if (PORTD.f2==1)     // S17  =1
   {PORTB.f4=0;          // activer les bloqueurs de T1
    PORTB.f3=0;          // arreter le transporteur1
    PORTA.f5=1;          // front montant pour le registre memoire
    delay_ms(10);        // attente
    PORTA.f5=0;         // front descendant
    PORTA=0x11;         //BE5
    break;}
}
   while(1)
{   if (PORTD.f7==1)     // S40=1
   {PORTB.f5=0;          // activer les bloqueurs  T2
    PORTB.f6=0;          // arreter le transporteur 2
    PORTA.f5=1;          // front montant pour le registre memoire
    delay_ms(10);        // attente
    PORTA.f5=0;          // front descendant
    break;}
 }
 return;
 }
  // les voies de séchage
  void voie1(){
     while(1)
 {    if (PORTD.F1==1&PORTD.f3==1)
    {PORTA=0x11;         // selectionner les buffers BE5 et BS2
     arret();            // mettre le bortB a zero
     PORTB.f4=1;         // desactiver les bloqueurs de T1 (ev3)
     PORTA.f5=1;         // front montant pour le registre memoire
     delay_ms(10);       // attente
     PORTA.f5=0;         // front descendant
     break;}
    }
        while(1)
   {   if(PORTD.f3==1)      // les bloqueurs de T1 sont desactivés
       {PORTB.f5=1;        // desactiver les bloqueurs de T1 (ev3)
       recule_trans2();    // deplacement de T2 vers la voie de sechage
       break;}
   }
         while(1)
   {    PORTA=0x11;        // selectionner le buffer BE5
     if(PORTD.F6==1)       // s29=1
       {arret();           // arret de trans
```

```
        dechargement(); // activer le verin et retour des transporteur a la
position init
        break;}
    }
    return;
}

 void voie2(){
        while(1)
   {     PORTA=0x11;
        if(PORTD.f1==1)    // les bloqueurs de T1 sont desactivés
      {PORTB.f4=1;        // desactiver les bloqueurs de T1 (ev3)
       PORTB.f5=1;        // desactiver les bloqueurs de T2 (ev4)
       avance_trans1();   // deplacement de T1 vers la voie de sechage
       recule_trans2();   // deplacement de T2 vers la voie de sechage
       break;}
   }
     while(1)
   {    PORTA=0x15;      // selectionner le buffer BE6
     if(PORTD.F0==1)     // S19=1
       {arret();         // arret de T1 et activation de ses bloqueurs
        PORTB.f5=1;
        recule_trans2(); // deplacement de T2 vers la voie de sechage
        break;}
    }
        while(1)
    {  if(PORTD==0x03)       // S19=S31=1
        {arret();             // mettre le portB a zero
         dechargement();     // activer le verin et retour des trans a la p
osition init
        break;}
    }
 }

 void voie3(){
        while(1)
   {        PORTA=0x11;
        if(PORTD.f1==1)        // les bloqueurs de T1 sont desactivés
      {PORTB.f4=1;             // desactiver les bloqueurs de T1 (ev3)
       PORTB.f5=1;             // desactiver les bloqueurs de T2 (ev4)
       avance_trans1();       // deplacement de T1 vers la voie3
       recule_trans2();       // deplacement de T2 vers la voie3
       break;}
   }
         while(1)
   {    PORTA=0x15;         // selectionner le buffer BE6
     if(PORTD.F2==1)        // s21=1
       {arret();            // arret de trans1
        PORTB.f5=1;
        recule_trans2();   // deplacement de T2 vers la voie3
        break;}
    }
        while(1)
    {  if(PORTD==0x0C)    //s21=s33=1
        {arret();         // arret des deux transporteurs
         dechargement();  // activer le verin et retour des trans a la posi
tion init
        break;}
    }
}
```

```
void voie4(){
         while(1)
   {     PORTA=0x11;
      if(PORTD.f1==1)       // les bloqueurs de T1 sont desactivés
      {PORTB.f4=1;           // desactiver les bloqueurs de T1 (ev3)
       PORTB.f5=1;           // desactiver les bloqueurs de T2 (ev4)
       avance_trans1();      // deplacement de T1 vers la voie4
       recule_trans2();      // deplacement de T2 vers la voie4
       break;}
   }
       while(1)
   {    PORTA=0x15;      // selectionner le buffer BE6
     if(PORTD.F4==1)      // s23=1
       {arret();          // arret de trans
        PORTB.f5=1;         // desactiver les bloqueurs de T2 (ev4)
        recule_trans2();     // deplacement de T4 vers la voie4
        break;}
    }
         while(1)
    {  if(PORTD==0x30)   //s23=s35=1
        {arret();         // arret des deux chariots
         dechargement();  // activer le verin et retour des trans a la posi
tion init
        break;}
    }
}

void voie5(){
         while(1)
   {       PORTA=0x11;
      if(PORTD.f1==1)   // les bloqueurs de T1 sont desactivés
     {PORTB.f4=1;      // desactiver les bloqueurs de T1 (ev3)
      PORTB.f5=1;
      avance_trans1();  // deplacement de T1 vers la voie5
      recule_trans2();  // deplacement de T2 vers la voie5
      break;}
   }
      while(1)
   {    PORTA=0x15;      // selectionner le buffer BE6
     if(PORTD.F7==1)      // s37=1
       {arret();          // arret de trans2
        PORTB.f5=1;       // desactiver les bloqueurs de T2 (ev4)
        avance_trans1();    // deplacement de T1 vers la voie5
        break;}
    }
         while(1)
    {  if(PORTD==0xC0)   //s37=s25=1
        {arret();         // arret des deux chariots
         dechargement();  // activer le verin et retour des trans a la posi
tion init
        break;}
    }
}

void voie6(){
     while(1)
   {     PORTA=0x11;
      if(PORTD.f1==1) // les bloqueurs de T1 sont desactivés
     {PORTB.f4=1;    // desactiver les bloqueurs de T1 (ev3)
      PORTB.f5=1;    // desactiver les bloqueurs de T2 (ev4)
      avance_trans1();   // deplacement de T1 vers la voie6
```

```
       recule_trans2();   // deplacement de T2 vers la voie6
       break;}
   }
      while(1)
   {    PORTA=0x19;      // selectionner le buffer BE6
     if(PORTD.F1==1)      // s39=1
       {arret();          // arret de trans1
        PORTB.f5=1;
        avance_trans1();    // deplacement de T1 vers la voie5
        break;}
    }
         while(1)
    {  if(PORTD==0x03)   //s39=s27=1
        {arret();         // arret des deux chariots
          dechargement();  // activer le verin et retour des trans a la pos
ition init
        break;}
    }
}

void sousprog1() {
  etape0:                  // debut de la premiere etape
            PORTA=0x00; // selectionner les buffers BE1 et BS1
     while(1)
   {    if (PORTD.f0==1) // si S1=1
      {avance_m1();      //deplacement de l'accrocheur A1 en avant
       arret_d_urg();    // arret d urgence
       break; }         // sortir de la boucle
   }
     while(1)
   {   PORTA=0x00; // selectionner les buffers BE1 et BS1
       if (PORTD==0x18)// si les capteurs S4,B1 sont activés
      {recule_m1();     //deplacement de l'accrocheur A1 en arriere
         arret_d_urg();    // arret d urgence
       break; }        // sortir de la boucle
   }
     while(1)
   {     PORTA=0x00; // selectionner les buffers BE1 et BS1
      if (PORTD.f2==1) // si S3=1
      {avance_m1();      //deplacement de l'accrocheur A1 en avant
        arret_d_urg();    // arret d urgence
       break; }         // sortir de la boucle
   }
     while(1)
   {    PORTA=0x00; // selectionner les buffers BE1 et BS1
         if (PORTD==0x68) //si S4,B2 et B3 sont activés
      {recule_m1();        //deplacement de l'accrocheur A1 en arriere
       arret_d_urg();    // arret d urgence
       break; }           // sortir de la boucle
    }
    while(1)
   {    PORTA=0x00; // selectionner les buffers BE1 et BS1
        if (PORTD.f1==1)  // si S2=1
      {PORTB=0x04;        // rotatioin de la table rotative(contacteur KMRD
)
       arret_d_urg();    // arret d urgence
       PORTA.f5=1;        //front montant pour le registre memoire
       delay_ms(10);      // attente
       PORTA.f5=0;        // front descendant
       break; }           // sortir de la boucle
   }
```

```
    while(1)
   {    PORTA=0x00; // selectionner les buffers BE1 et BS1
         if (PORTD.f7==1) // si C2=1
      {arret();          // metre le PORTB a zero
       recule_m2();      //deplacement de l'accrocheur A2 en arriere
       arret_d_urg();    // arret d urgence

       break; }         // sortir de la boucle
    }
     while(1)
    {   PORTA=0x04;      // selection du BUFFER BE2
        if (PORTD.f6==1)//si S5=1
       {avance_m2();     //deplacement de l'accrocheur A2 en avant
        arret_d_urg();    // arret d urgence
        break; }          // sortir de la boucle
    }
    while(1)
   {   PORTA=0x04;      // selection du BUFFER BE
     if (PORTD.f3==1) // si S8=1
      {recule_m2();      //deplacement de l'accrocheur A2 en arriere
       arret_d_urg();    // arret d urgence
       break; }        // sortir de la boucle
    }
   while(1)
    {   PORTA=0x04;      // selection du BUFFER BE
         if (PORTD.f5==1)// si S6=1
       {avance_m2();     //deplacement de l'accrocheur A2 en avant
        arret_d_urg();    // arret d urgence
        break; }         // sortir de la boucle
    }
    while(1)
   {     PORTA=0x04;      // selection du BUFFER BE
           if (PORTD==0x0E) // si S8,B4 et B5 sont activés
      {a4=1;
       a5=1;
       recule_m2();      //deplacement de l'accrocheur A2 en arriere
       arret_d_urg();    // arret d urgence
       break; }         // sortir de la boucle
    }
    while(1)
   {   PORTA=0x04;      // selection du BUFFER BE
          if (PORTD.f4==1) // si S7=1
      {PORTB=0x20;      // rotation de la table rotative a gauche(led KMRG)
       arret_d_urg();    // arret d urgence
       PORTA.f5=1;      // front montant pour le registre memoire
       delay_ms(10);    // attente
       PORTA.f5=0;      // front descendant
       break; }         // sortir de la boucle
    }
    while(1)
   {  PORTA=0x04;      // selection du BUFFER BE
        if (PORTD.f7==1) //  si C1=1
      {arret();  // mettre le bortB a zero
       arret_d_urg();    // arret d urgence
       break; }          //  sortir de la boucle
    }
    return;
}

void sousprog2(){
  etape2:                 // debut de la premiere etape
```

```
                        // selectionner les buffers BB2
      while(1)
    {  PORTA=0x04;
         if (PORTD.f1==1 & PORTD.F2==1) // B4=B5=1
       { a4=1;          // capteur B4=1
         a5=1;          // capteurB5=1
         PORTA=0x08;    // selection des buffeers BE3 et BS1
         break;}        // sortir de la boucle
    }
      while(1)
    {    if (PORTD==0x86 & a4==a5==1)// conditions initials vérifiés
        {recule_m3();    // activer le moteur M3 (marche arriere)
         arret_d_urg();    // arret d urgence
         break;          // sortir de la boucle
         }
    }
      while(1)
    {       PORTA=0x04;     // selection des buffers BE2 et BS1
       if ( PORTD.f3==1) // capteur S8=1
       {avance_m3();      // activer le moteur M3 (marche avant)
        arret_d_urg();    // arret d urgence

        break;}           // sortir de la boucle
     }
      while(1)
     {    PORTA=0x08;       // selection des buffeers BE3 et BS1
       if (PORTD.f0==1&PORTD.f3==1&PORTD.f4==1)// les capteurs S9 B6 et B7
 sont activés
         {arret();
          PORTA.F0=1;        // selection du buffer BS2
          PORTB.f0=1;        // activer le contacteur KMP11(ouverture de l
a porte d'entrée)
          arret_d_urg();    // arret d urgence
          PORTA.f5=1;        // front montant pour le registre memoire
          delay_ms(10);      // attente
          PORTA.f5=0;        // front descendant

          break;}            // srtir de la boucle
     }
      while(1)
     {     PORTA=0x06;        // selection des buffers BE2 et BS2
       if (PORTD.f0==1)   // capteur CPE1=1 (porte ouverte)
       {arret();           // mettre le portD a zero
        PORTA=0x04;       // selection des buffers BE2 et BS1
        avance_m3();      // activer le moteur M3 (marche avant)

        break;}           // srtir de la boucle
     }
      while(1)
     {  PORTA=0x0C;       // selection des buffers BE4 et BS1
        if (PORTD.f1==1)  // capteur S12=1
        {recule_m3();      // activer le moteur M3 (marche arriere)
         arret_d_urg();    // arret d urgence
         break;}          // srtir de la boucle
     }
        while(1)
     {  PORTA=0x0C;       // selection des buffers BE4 et BS1
        if (PORTD.f0==1) // capteur S10=1
         {avance_m3();     // activer le moteur M3 (marche avant)
          arret_d_urg();    // arret d urgence
          break; }         // srtir de la boucle
```

```
        }
          while(1)
       {  PORTA=0x0C;        // selection des buffers BE4 et BS1
          if (PORTD.f2==1)   // capteur S13=1
            {recule_m3();       // activer le moteur M3 (marche arriere)
             arret_d_urg();    // arret d urgence
             break;}            // srtir de la boucle
       }
          while(1)
       { PORTA=0x0C;       // selection des buffers BE4 et BS1
          if (PORTD.f0==1)    // capteur S10=1
            {avance_m3();        // activer le moteur M3 (marche avant)
             arret_d_urg();    // arret d urgence
             break;}             // srtir de la boucl
       }
          while(1)
       { PORTA=0x0C;       // selection des buffers BE4 et BS1
          if (PORTD.f1==1)    // capteur S12=1
            {recule_m3();        // activer le moteur M3 (marche arriere)
              arret_d_urg();    // arret d urgence

             break;}             // srtir de la boucle
       }
          while(1)
       {    PORTA=0x09;          // selection des buffers BE3 et BS2
             if (PORTD.f1==1)   // capteur S11 activé
           {PORTB=0x02;         // activer le contacteur KMP12(fermer la port
e d'entrée)
             arret_d_urg();    // arret d urgence
            PORTA.f5=1;        // front montant pour le registre memoire
            delay_ms(10);      // attente
            PORTA.f5=0;        // front descendant

            break;}            // sortir de la boucle
        }
          while(1)
       {     PORTA=0x08;         // selection des buffeers BE3 et BS1
             if (PORTD.f7==1)  // capteur CPE2=1(porte fermée)
            {arret();          // mettre le portD a zero
             avance_m3();     // activer le moteur M3 (marche avant)
             arret_d_urg();    // arret d urgence

             break;}          // sortir de la boucle
       }
          while(1)
       {      PORTA=0x0C;      // selection des buffers BE4 et BS1
        if (PORTD==0x3C)// les capteurs S13,B8 et B9 sont a 1
            {recule_m3();   // activer le moteur M3 (marche arriere)
             arret_d_urg();    // arret d urgence

             break;}        // sortir de la boucle
       }
           while(1)
       {    PORTA=0x08;    // selection des buffeers BE3 et BS1
           if (PORTD.f1==1) // capteur S11=1
            {avance_m3();     // activer le moteur M3 (marche avant)
              arret_d_urg();    // arret d urgence
             break;}          // sortir de la boucle
       }
           while(1)
       {    PORTA=0x08;    // selection des buffeers BE3 et BS1
```

```
          if (PORTD.f5==1) // capteur B11=1
           { a11=1;            // sauvgarder l'info

             break;}           // sortir de la boucle
       }
          while(1)
       {   PORTA=0x0c;        // selection des buffers BE4 et BS1
              if (PORTD.f1==1 & a11==1) // les capteurs S12 et B11 sont a1
           {recule_m3();       // activer le moteur M3 (marche arriere)
            arret_d_urg();    // arret d urgence

             break;}           // sortir de la boucle
       }
           while(1)
       {   PORTA=0x08;        // selection des buffeers BE3 et BS1
             if (PORTD.f1==1) // capteur S11 activé
           {avance_m3();     // activer le moteur M3 (marche avant)
             arret_d_urg();    // arret d urgence
            break;}          // sortir de la boucle
       }
            while(1)
       {     PORTA=0x0C;      // selection des buffers BE4 et BS1
              if (PORTD.f2==1) // capteur S13=1
           {s13=1;           // sauvgarder l'info
            PORTA=0x08;      // selection des buffeers BE3 et BS1
            break;}          // sortir de la boucle
       }
            while(1)
       {   PORTA=0x08;      // selection des buffeers BE3 et BS1
       if (PORTD.f5==1 & PORTD.f6==1 & PORTD.f2==1 & s13==1) // S13=s17=B11=
B12=1
           {a11=1;           // sauvgarder les informations
            a12=1;
            s17=1;
            recule_m3();      // activer le moteur M3 (marche arriere)
            arret_d_urg();    // arret d urgence
            break;}           // sortir de la boucle
       }
            while(1)
       {   PORTA=0x08;      // selection des buffeers BE3 et BS1
              if (PORTD.f1==1)// capteur S11 activé
           { s11=1;
             arret();       // mettre le portD a zero
             break;}        // sortir de la boucle
        }
     return;
}

void sousprog3(){
   etape3:             // debut de la premiere etape

        while(1)
   {     PORTA=0x11;
              if(PORTD==0xA5) // cond init
         { PORTB=0x30;        //EV3 et EV4(desactiver les bloqeurs de T2 et
 T)
           arret_d_urg();    // arret d urgence
           PORTA.F5=1;        // front montant pour le registre memoire
           delay_ms(10);      // attente
           PORTD.f5=0;        // front descendant
           break;}
```

```
   }
voies:      while(1)
   {    PORTA=0x11;
       if (j==1)
     {voie1();                  // choisir la voie 1
      arret_d_urg();    // arret d urgence
       goto suite;
        break;}
       else if (j==2)
     {voie2();                  // choisir la voie 2
      arret_d_urg();    // arret d urgence
      goto suite;
        break;}
       else if (j==3)
     {voie3();                  // choisir la voie 3
      arret_d_urg();    // arret d urgence
      goto suite;
        break;}
     else if (j==4)
     {voie4();                  // choisir la voie 4
      arret_d_urg();    // arret d urgence
      goto suite;
        break;}
     else if (j==5)
     {voie5();                  // choisir la voie 5
      arret_d_urg();    // arret d urgence
      goto suite;
        break;}
     else if (j==6)
     {voie6();                  // choisir la voie 6
      arret_d_urg();    // arret d urgence
      goto suite;
        break;}
     else if (j==7)
     {j=1;                      // initialisation du compteur
      goto voies;
        break;}
    }
     while(1)
{ suite:      PORTA=0x0A;   // selection du buffer BE3
     if (PORTD.f2==1) // s17=1 (chariot T1 devant la voie 1
       {s17=1;         // sauvgarder l info
        arret_d_urg();    // arret d urgence

        break;}
  }
       while(1)
 {        PORTA=0x12;      // selection du buffer BE5 et BS3
    if(PORTD.f7==1&PORTD.f2==1&PORTD.F0==1&s17==1)  // s40=1(tras2 devant l
a sortie)et bloqueurs activer
       {arret();          // mettre le portB a zero
        PORTB.f2=1;       // ouvrir la porte de sortie
        arret_d_urg();    // arret d urgence
        PORTA.f5=1;       // front montant pour le registre memoire
        delay_ms(10);     // attente
        PORTA.f5=0;       // front descendant
        break;}
 }
      while(1)
 {   PORTA=0x12;       // selection du buffer BE5 et BS3
       if (PORTD.f4==1)  // cps1=1 (porte de sortie ouverte)
```

```
    {arret();
     recule_m4();      // deplacement de l'accrocheur A4 en arriere
     arret_d_urg();    // arret d urgence
     break;}
 }
      while(1)
 {    PORTA=0x1A;      // selection des buffers BE7 et BS3
      if(PORTD.f2==1)
    {arret();
     avance_m4();    // deplacement de l'accrocheur A4 en avant
     arret_d_urg();    // arret d urgence
     break;}
 }
     while(1)
 { PORTA=0x1A;      // selection des buffers BE7 et BS3
  if (PORTD.f3==1)     // s42=1
    {recule_m4();       // deplacement de l'accrocheur A4 en arriere
     arret_d_urg();    // arret d urgence
     break;}
}
    while(1)
 {    PORTA=0x1A;      // selection des buffers BE7 et BS3
      if(PORTD.f2==1)    // s41=1
    {arret();
     avance_m4();       // deplacement de l'accrocheur A4 en avant
     arret_d_urg();    // arret d urgence
     break;}
 }
  while(1)
 {      PORTA=0x1A;      // selection des buffers BE7 et BS3
      if(PORTD.f4==1)    // s43=1
    {arret();
     PORTB.f3=1;        // fermer la porte de sortie du sechoir
     arret_d_urg();    // arret d urgence
     PORTA.f5=1;        // front montant pour le registre memoire
     delay_ms(10);      // attente
     PORTA.f5=0;        // front descendant

     break;}
 }
  return;
}

void main() {
   // configuration du pic18f4450
             TRISD=0XFF; // portD en entrée
             TRISA=0X00; // le portA en sortie
             TRISB=0X00; //le portB en sortie
             ADCON1=0X0f; // utilisation des valeurs numeriques seulement
 boucle: j=j+1;
     PORTA=0x00; // selectionner les buffers BE1 et BS1
     while(1)
   {    if (PORTD.f0==1) // si S1=1
      {sousprog1();       // execution de sous programme 1
       break;}
   }
      while(1)
   {
          PORTA=0x08;             // selectionner les buffers BB3
      if (PORTD==0x86 & a4==a5==1)// conditions initials vérifiés
          { sousprog2();            // execution de sous programme 2
```

```
            break;}
   }
      while(1)
   {          PORTA=0x011;         //selection des buffers BE5 et BS2
         if(s17==a11==a12==1)    //  S17=B11=B12=1
         {sousprog3();            // execution de sous programme 3
           break;}
   }
    while(1)
{     if(PORTD.f5==1)          // cps1=1(porte de sortie fermée)
    {arret();          // metre le portB a zero
     goto boucle;
     break;}


                  //fin de programme
```

5.3- Compilation :

Après avoir créé le projet et écrit le code source, on passe à l'exécution du programme en sélectionnant **Project >> Build** à partir du menu déroulant ou cliquez sur l'icône **Build** dans la barre d'outils du projet, et la compilation de notre programme est représenté sur la figure **(Fig.4.6).**

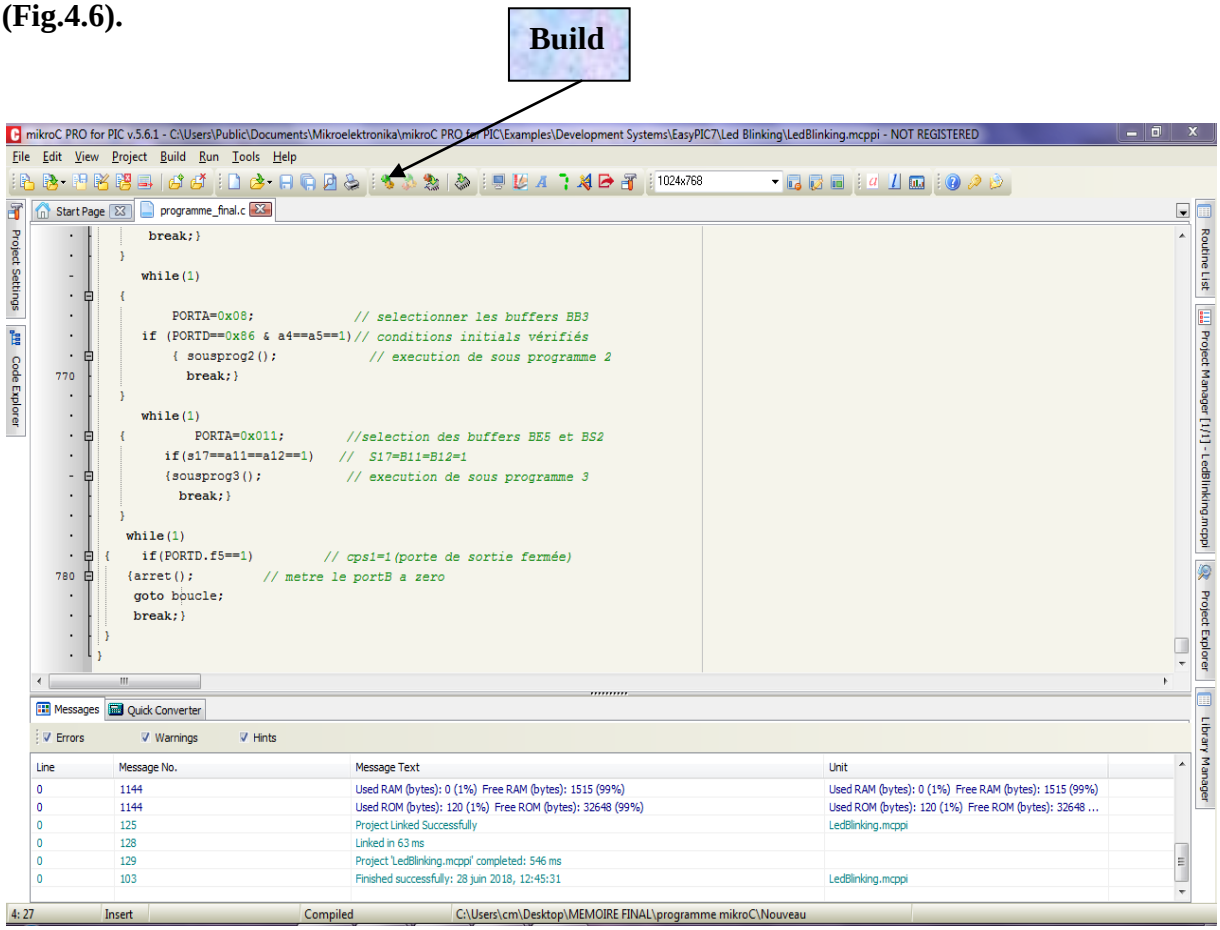


Fig.4.6 : Compilation de programme MikroC.

6- Conclusion :

Après identification MikroC Pro pour PIC, on a conclu que le mikroc est l'un des meilleurs logiciels pour programmation et simulation des microcontrôleurs.

Par conséquent, nous avons pu arriver à la programmation du pic 18F4550 avec ce logiciel afin de réaliser notre projet d'étude.

Et pour éclairer les choses, on cité plusieurs étapes de mise en œuvre de notre programme, et ça compilation, et on a bien montré les techniques essentielles qu'il faut avoir pour réalisé un programme simple et facile à l'aide de MikroC.

Nous avons donné des exemples simples et clairs qui aident finalement le lecteur à bien s'initier avec la commande par les microcontrôleurs

Après la compilation sur MikroC, et le chargement du programme dans notre Microcontrôleur, on obtient exactement le fonctionnement voulu en simulant le projet ISIS et on suivant les étapes et les conditions citées dans notre cahier de charge.

CONCLUSION GENERALE

Notre projet comporte un travail théorique et pratique, son objectif consiste sur «La réalisation d'une carte de commande à base de microcontrôleur », pour commander une unité de transfert de chariots dans la briqueterie « Irdjen ».

Nous nous sommes consacrés à réaliser un objet de commande qui permettrai de remplacer l'utile le plus couramment utilisé dans l'industrie qui est l'automate programmable, en tenant en considération l'optimisations et la diminution des couts des matériels d'un coté et la facilitations de maintenance matériels et logistique d'un autre .

Grâce à ce stage nous pouvons maitriser un certain nombre d'instruments et d'outils indispensables pour un automaticien. Et nous avons découvert la réalité du monde industriel, et la modélisation de l'unité de transfert nous a permet d'acquérir une expérience professionnelle et nous a offert des connaissances sur les systèmes automatisés et les grandes chaines industrielles de production.

Le circuit de commande est réalisé afin de contrôler les actionneurs associés, et l'ensemble de système est commandé par un microcontrôleur qui en utilisant les informations actuelles, décider de l'action à prendre. Pour notre cas on a utilisé le microcontrôleur PIC18F4550, dont ces caractéristiques nous ont aidés en facilitant la tache surtout en ce qui concerne sa programmation.

Après avoir terminé avec succès notre projet, nous avons pu avoir une bonne idée et une initiation claire sur le contrôle des systèmes électriques en utilisant des microcontrôleurs, et ça nous a offert une expérience pratique.

Cette étude offre les éléments essentiels de la commande avancée utilisé par la technologie de pointe indispensable dans les différents domaines surtout que pratiquement tous les fabricants de microprocesseurs proposent une ou plusieurs gammes de microcontrôleurs ce qui facilite leur utilisations pour commander des processus industriels.

D'autre part, nous avons montré qu'il faut avoir chez le futur ingénieur des acquits de devers disciplines (électrique, électronique, informatique, ...), pour conquérir le domaine de la commande par les microcontrôleurs.

En définitive, la perspective de notre travail c'est exploiter un microcontrôleur pour commander un processus industriel et nous pouvons dire que la carte de commande à base de microcontrôleur est adaptable à notre environnement et le PIC 18F4550 peut bien jouer le rôle d'unité de commande pour l'unité de transfert des chariots.

REFERENCE BIBLIOGRAPHIQUE

[1] : M. Bapio BAYALA. Support cours LA MACHINE ASYNCHRONE. Université Renne 2010.
[2] : M.Messaad. Université Bejaia. Support cours capteurs, actionneur.2014/2015
[3] : L.BERGOUGNOUX. Support cours Automates Programmables Industriel. POLYTECH Marseille. 2004/2005.
[4] : A. Mazidi. Support cours .systèmes à microprocesseurs. Université Texas. 2014/2015
[5] : M. Benmakhlouf. Généralités sur les systèmes à base de microprocesseur. Université Ouargla. 2014
[6] : F.Mallet.Université Nice. Support cours. Architecture d'un microprocesseur.2012/2013
[7]: M.Bouaraba.Université Bouira. Support Cours Microcontrôleur.2011/1012
[8] : Philippe. Hoppenot. Université d'Evry. Support cours. Systèmes à base-microprocesseur. 2016.
[9] : G BERTHOME. Lycée Mireille GRENET. Organisation fonctionnelle d'un système à microcontrôleur.2009/2010
[10]: Adjiba Brahim et Chalghoum Abdelmonaim. << Commande des équipements électriques par microcontrôleurs "Simulations et Réalisations" >> Mémoire de Fin d'Étude
En vue de l'obtention du diplôme de MASTER ACADEMIQUE. Université d'El-Oued. Année 2014/2015.
[11] : Université Kenitra. Support cours .instructions pic.2015/2016.
[12] : Danny.Causey. Introduction à proteus.2011
[13]: Datasheet PIC18F4450.
[14]: Pierre. Auger. Université Paris 11. Support cours utilisation buffer 74HC245.2013
[15]: Datasheet 74HC245.
[16]: Datasheet 74LS139.
[17]: Datasheet 74LS139.
[18]: Datasheet 74198.

Résumé

Notre projet consiste à remplacer l’automate programmable dans l’unité de transfert des chariots au niveau de la briqueterie d’IRDJEN par une carte de commande à base de microcontrôleurs, pour économiser les couts et aussi simplifier le circuit de commande.

Comme premier étape nous avons étudié le fonctionnement de l'unité et on a modélisé le système en utilisant le grafcet, ensuite, on a donné une initiation aux les systèmes à microprocesseur et à microcontrôleur et aussi quelques définitions sur les microcontrôleurs de la famille PIC.

Pour la conception de circuit de commande nous avons choisi le logiciel « PROTEUS », comme une unité de commande on a choisi le PIC18F4550, et puisque l’unité contient un nombre important d’entrées/sorties on a utilisé des buffers (74HC245) pour augmenter le nombre nombre d’E/S du pic. Et on a géré la sélection de ces buffers par deux décodeurs.

Dans la partie programmation nous avons utilisé le logiciel MikroC. Et après la simulation, on a chargé le programme dans le PIC18F4550.

A la fin. On a réussi notre application et on a obtenu le bon fonctionnement qui correspond au cahier de charge. Et on constate que le microcontrôleur PIC18F4550 peut bien jouer le role d’une unité de commande pour les systèmes industriel de production.

Mots clés

GRAFCET, systèmes à microprocesseur, systèmes à microcontrôleurs, Les Pic, PIC18F4550, buffer 74HC245, décodeur 74LS138, décodeur 74LS139, registre mémoire 74198, proteus, ISIS, ARES, MikroC.