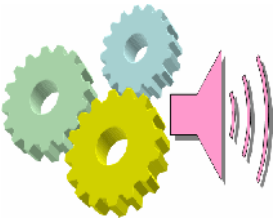
ROYAUME DU MAROC

MINISTERE DE L'ÉDUCATION NATIONALE

Académie de Casablanca Settat

Direction Provinciale de Mohammedia



Nom : ……

Prénom : ……

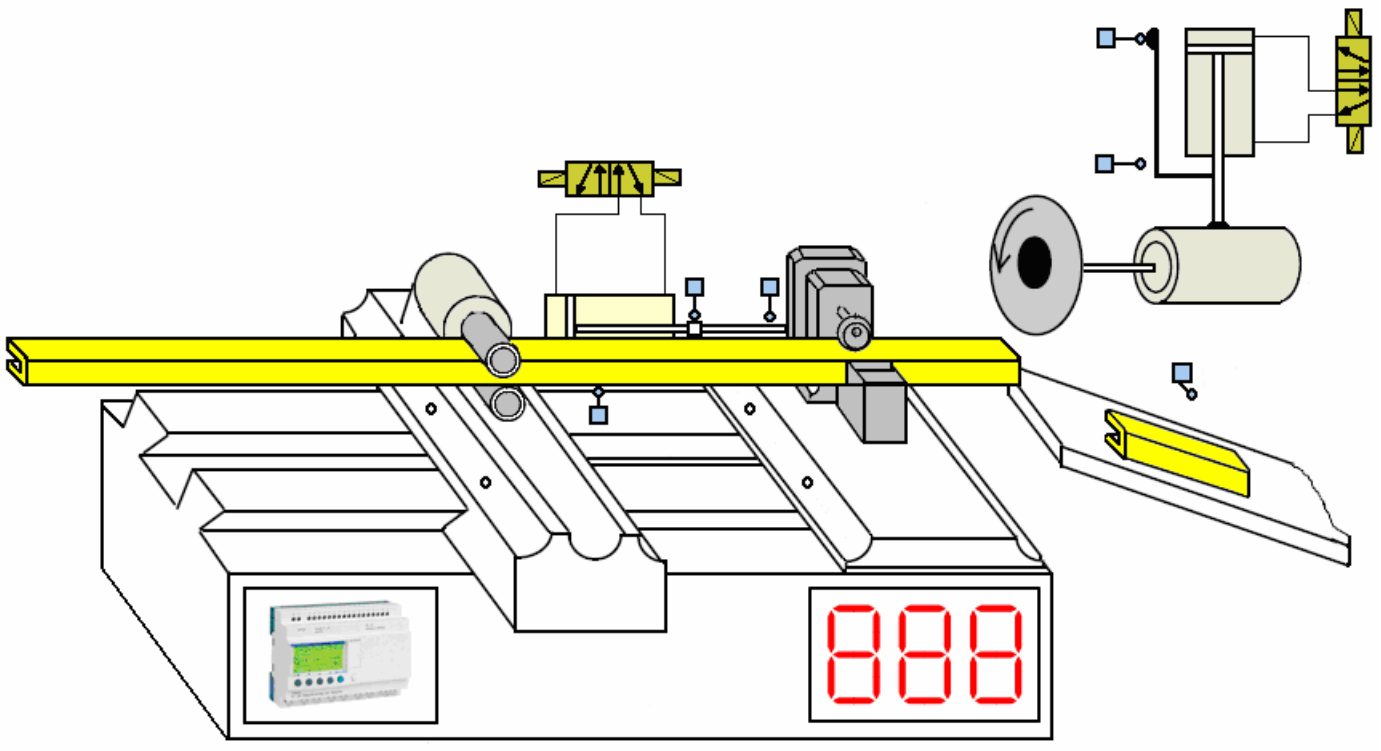
Classe : **2STE**…

Lycée Qualifiant Technique Mohammedia

Sciences de l'ingénieur

Système n°1 :

Tronçonneuse Automatique



Sciences et Technologies Électriques Niveau 2

Professeur : **MAHBAB**

المملكة المغربية وزارة التربية الوطنية

الثانوية التأهيلية التقنية المحمدية

Le dossier comporte au total 67 pages :

Sujet : Tronçonneuse Automatique

- ☞ Le sujet comporte au total 19 pages.
- ☞ Le sujet comporte 3 types de documents :

- **Pages 01 à 03** : Socle du sujet comportant les situations d'évaluation (SEV) ;
- **Pages 04 à 09** : Documents ressources portant la mention **DRES XX**
- **Pages 10 à 19** : Documents réponses portant la mention **DREP XX**

19 pages

Unité A.T.C

Fiches cours :

- Fiche cours n°01 : **Le GRAFCET**
- Fiche cours n°02 : **L'organigramme**
- Fiche cours n°03 : **Système à microprocesseur**
- Fiche cours n°04 : **Les mémoires électroniques**
- Fiche cours n°05 : **Les interfaces d'entrée sortie**
- Fiche cours n°06 : **Le microprocesseur**
- Fiche cours n°07 : **Le microcontrôleur PIC 16 F 84**
- Fiche cours n°08 : **Le jeu d'instruction**
- Fiche cours n°09 : **Configuration des ports A et B**

27 pages

Activités :

- Activité n°01 : **Commande d'un chariot**
- Activité n°02 : **Commande d'un chariot (Temporisation)**
- Activité n°03 : **Système de tri de caisse**
- Activité n°04 : **Commande de deux chariots**

19 pages

TRONÇONNEUSE AUTOMATIQUE

1. PRÉSENTATION DU SYSTÈME :

La figure ci-dessous représente un système de tronçonnage utilisé pour le découpage des barres d'aluminium en forme de profilé : **U**

2. FONCTIONNEMENT ET DESCRIPTION DU SYSTÈME :

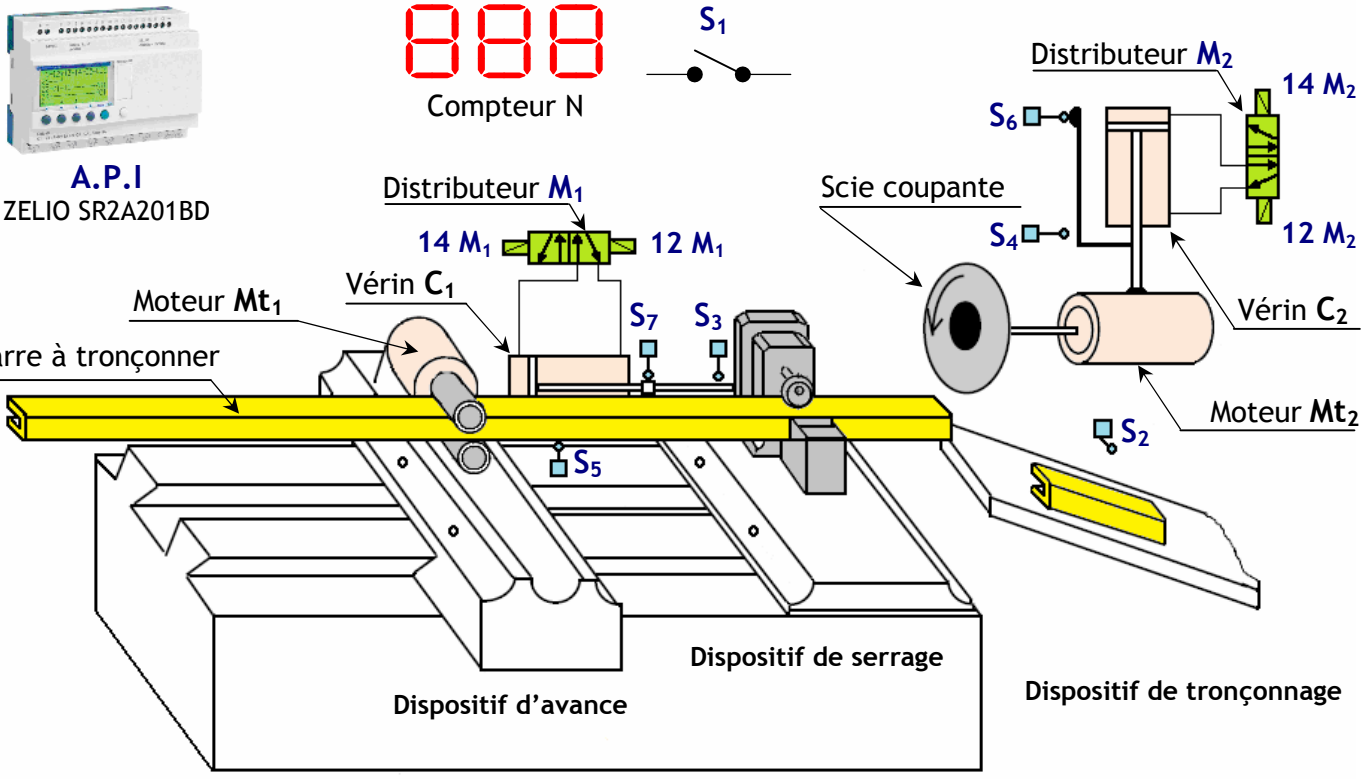
Le système permet le découpage d'une barre d'aluminium en **20 morceaux** de longueur prédéterminée. Le fonctionnement est le suivant :

- La barre est introduite manuellement entre les deux rouleaux ; sa présence est détectée par le capteur S_5 ;
- L'action sur le bouton poussoir S_1 met le moteur Mt_1 en marche. Le dispositif d'avance entraîne la barre ;
- Lorsque celle ci actionne le capteur S_2. Le moteur Mt_1 s'arrête et le vérin pneumatique C_1 provoque le serrage de la barre grâce à un dispositif approprié ;
- Une fois la barre est serrée (capteur S_3 actionné), le moteur Mt_2 fonctionne et le vérin pneumatique C_2 fait descendre le dispositif de tronçonnage ;
- La fin du tronçonnage est détectée par le capteur S_4 provoquant ainsi l'arrêt du moteur Mt_2 et la rentrée de la tige du vérin C_2, (détectée par le capteur S_6) ;
- Lorsque le capteur S_6 est actionné, la barre est desserrée (rentrée de la tige du vérin C_1).
- Lorsque celle ci actionne le capteur S_7. Le **compteur** est incrémenté de **1** ;
- Le comptage des morceaux découpés se fait par un compteur **N**.

Le cycle décrit précédemment se répète tant que le nombre de morceaux découpés reste inférieur à **20**; dans le cas contraire, on provoque l'arrêt du système et l'initialisation du compteur. Pour des raisons de sécurités, la barre ne peut avancer que si le moteur Mt_2 est en position haute.

Les moteurs Mt_1 et Mt_2 sont respectivement alimentés par des contacteurs KM_1 et KM_2.

Les vérins C_1 et C_2 sont respectivement alimentés par des distributeurs M_1 et M_2.



SEV 1 — ÉTUDE FONNCTIONNELLE DU SYSTÈME

RESSOURCES À EXPLOITER : **DRES 01** page 04, 'Description' et 'Fonctionnement' page 01.

Tâche 1 — Analyse fonctionnelle globale

✍ Répondre sur le document **DREP 01** page 10

1. Citer la fonction **globale** du système ;
2. Quel type d'**énergie** reçoit le système ;
3. Indiquer la nature de la matière d'œuvre en cochant la case correspondante ;
4. Donner la valeur ajoutée du système (VA) ;
5. Donner le rôle des organes de contrôles S_2, S_3, S_4 et S_5 ;
6. Donner la **nature** de l'information délivrée par ces capteurs ;
7. Compléter l'actigramme **A-0** du système.

Tâche 2 — Analyse fonctionnelle structurelle

8. Sur le document **DREP 01** page 10, compléter la modélisation de la relation entre les deux **rouleaux**, le moteur Mt_1 et le contacteur KM_1 ;
9. Le F.A.S.T du **DREP 02** page 11 définit la fonction globale (**FG**) du système étudié. Indiquer sur ce document, pour chacune des fonctions technique le **processus** qui lui est associé ;
10. Sur le document **DREP 03** page 12, compléter le modèle fonctionnel du système (éléments de la **P.O**, éléments de la **P.C**, éléments de l'**I.H.M**) ;
11. Sur le document **DREP 03** page 12, proposer deux autres solutions constructives pour réaliser la fonction FT_6.

SEV 2 — COMMANDE PAR L'A.P.I ZELIO SR2A201BD

RESSOURCES A EXPLOITER : **DRES 01, 02, 03** pages 04, 05 et 06

Tâche 1 — L'outil graphique normalisé 'GRAFCET'

La commande de la tronçonneuse est réalisée par un module logique programmable de type **ZELIO SR2A201BD** dont les caractéristiques sont données au **DRES 01** page 04.

Pour la programmation des automates programmables industriels, on utilise le GRAFCET :

✍ Répondre sur le document **DREP 04** page 13

12. Compléter le **vide** de la figure 1 par les mots correspondants (les éléments d'un GRAFCET);
13. Relier par des **flèches** les étiquettes de la figure 2 ;
14. Considérons la séquence de la figure 3, répondre aux **questions proposées** ;
15. Sur le document **DREP 05** page 14, compléter le **vide** des différentes figures par les mots correspondants (les structures de base d'un GRAFCET).

Tâche 2 — GRAFCET de fonctionnement du système

✍ Répondre sur le document **DREP 06** page 15

16. En se référant au **DRES 01** page 04 et au document de la page 01, compléter le GRAFCET du point de vue **système** traduisant le fonctionnement normal de la tronçonneuse ;

17. En se référant au **DRES 01** page 04 et au GRAFCET du point de vue système, compléter le GRAFCET du point de vue partie **commande** traduisant le fonctionnement normal de la tronçonneuse.

Tâche 3 — Programmation en langage LADDER

Pour la programmation de l'automate, on utilise le langage LADDER (voir **DRES 02**). Pour le comptage des morceaux coupés, on utilise le compteur C_1 de l'A.P.I (voir **DRES 03**).

Le langage LADDER est une succession " de réseaux de contacts " véhiculant des informations logiques depuis les entrées vers les sorties. Le résultat dépend des fonctions programmées.

18. En se référant au **DRES 01** page 04 et au GRAFCET du point de vue partie commande, compléter sur le document **DREP 07** page 16, le GRAFCET du point de vue **A.P.I** traduisant le fonctionnement normal de la tronçonneuse ;

A fin de préparer la phase de programmation, on doit rédiger les équations des sorties. A partir du GRAFCET point de vue partie commande codé :

19. On demande d'établir les équations de **récurrence** du GRAFCET sur le **DREP 07** page 16 ;

20. En déduire le programme **LADDER** correspondants sur le **DREP 08** page 17.

SEV 3 — COMMANDE PAR LE MICROCÔNTROLEUR PIC 16F84

RESSOURCES A EXPLOITER : **DRES 04**, **05**, **06** pages 07, 08 et 09

Afin d'élargir le champ d'exploitation du système, on propose d'étudier la commande de la tronçonneuse par un système électronique à base du microcontrôleur le PIC16F84. Le schéma de la carte de commande à µC est donné au **DRES 04**, **05** pages 07 et 08 :

- Pour les entrées : seul le schéma de câblage du capteur S_5 est représenté ;
- Pour les sorties : seul le schéma du moteur Mt_1 (Positionner la barre) est représenté.
- Ces montages permettent une isolation galvanique des entrées et des sorties du microcontrôleur ;
- Ces montages, on les trouve intégrés dans les modules d'entrée et de sortie d'un A.P.I.

Tâche 1 — Conditionnement des signaux

Les capteurs S_2, S_3, S_4, S_5, S_6 et S_7 sont des détecteurs (ou interrupteurs) de position à action mécanique. L'information délivrée par ces capteurs doit être conditionnée avant d'attaquer les entrées du microcontrôleur.

✍ Répondre sur le document **DREP 09** page 18

21. Décrire le **fonctionnement** de ces capteurs ;

22. Donner le nom et le rôle des blocs F_1, F_2, F_3 et F_4.

Tâche 2 — Organigramme et programme de fonctionnement

23. En se référant aux **DRES 04** pages 07 et au GRAFCET du point de vue partie commande, compléter sur le document **DREP 09** page 18 l'**organigramme** traduisant le fonctionnement normal de la tronçonneuse ;

24. Sur le document **DREP 09** page 18, compléter le programme d'**initialisation** du µC correspondant à la configuration donnée sur le document **DRES 04** ;

25. En se référant à l'organigramme traduisant le fonctionnement normal de la tronçonneuse, compléter le **programme** du document **DREP 10** page 19.

DRES 01

Module logique ZELIO SR2A201BD

1. Présentation :

Grâce à sa facilité de mise en œuvre et sa simplicité de programmation, le module logique Zelio **SR2A201BD** est destiné à la réalisation de petits équipements d'automatisme.
Ses caractéristiques principales sont les suivantes :

- 10 entrées TOR (I_1 à I_A) ;
- 2 entrées mixtes (TOR/Analogique I_B à I_C) ;
- 8 sorties à relais (Q_1 à Q_8) ;
- Interface Homme/machine avec boutons et affichage LCD ;
- 28 relais auxiliaires (M_1 à M_V) ;
- 16 temporisateurs (T_1 à T_G) ;
- 16 Compteurs (C_1 à C_G) ;
- 8 comparateurs de compteurs (V_1 à V_8) ;
- Langages de programmation LADDER et FBD.

2. Affection des entrées/sorties :

2.1. Affection des sorties :

Actions	Actionneurs		Préactionneurs		A.P.I
Positionner la barre	Moteur Mt_1		Contacteur	KM_1	Q_1
Serrer la barre	Vérin C_1	SC_1	Distributeur M_1	$14\ M_1$	Q_2
desserrer la barre		RC_1		$12\ M_1$	Q_3
Couper la barre	Moteur Mt_2		Contacteur	KM_2	Q_4
	Vérin C_2	SC_2	Distributeur M_2	$14\ M_2$	Q_5
Positionner la tronçonneuse en haut		RC_2		$12\ M_2$	Q_6
Compter le nombre de morceaux coupés	Compteur N				CC_1

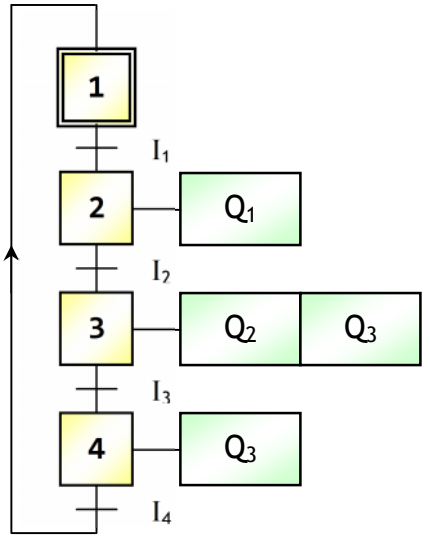
2.2. Affection des entrées :

Compte-rendu et consigne	Capteur et bouton		A.P.I
Départ cycle	Bouton poussoir	S_1	I_1
Barre positionnée	Détecteur mécanique à levier	S_2	I_2
Barre serrée	Détecteur mécanique à levier	S_3	I_3
Barre coupée	Détecteur mécanique à levier	S_4	I_4
Présence barre	Détecteur mécanique à levier	S_5	I_5
Tronçonneuse positionnée en haut	Détecteur mécanique à levier	S_6	I_6
Barre desserrée	Détecteur mécanique à levier	S_7	I_7

DRES 02

Réalisation de GRAFCET en langage LADDER

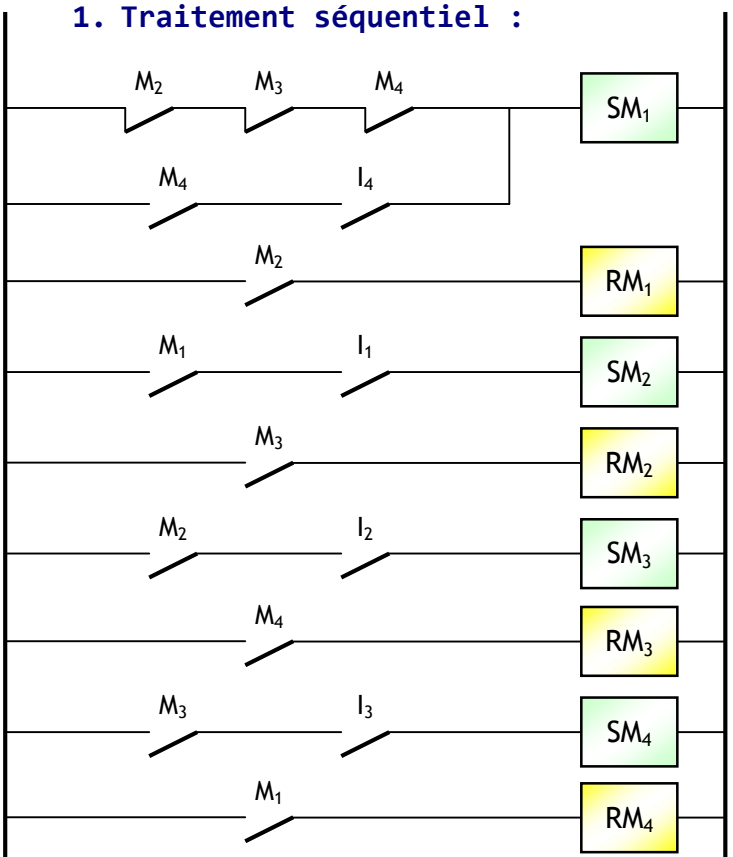
Soit le GRAFCET point de vue partie commande codé API suivant :



On associe à chaque étape **n°i** du GRAFCET un bit interne de l'API **Mi :**

- à l'étape 1 on associe un bit M_1 ;
- à l'étape 2 on associe un bit M_2 ;
- à l'étape 3 on associe un bit M_3 ;
- à l'étape 4 on associe un bit M_4.

1. Traitement séquentiel :



En plus de son activation par l'étape précédente (M_4), l'étape initiale 1 (M_1) doit être active au démarrage quand toutes les autres étapes ne sont pas actives :

$$SM_1 = M_4.I_4 + \overline{M_2}.\overline{M_3}.\overline{M_4}$$

L'étape initiale 1 (M_1) doit être désactivée lorsque l'étape 2 (M_1) est active :

$$RM_1 = M_2$$

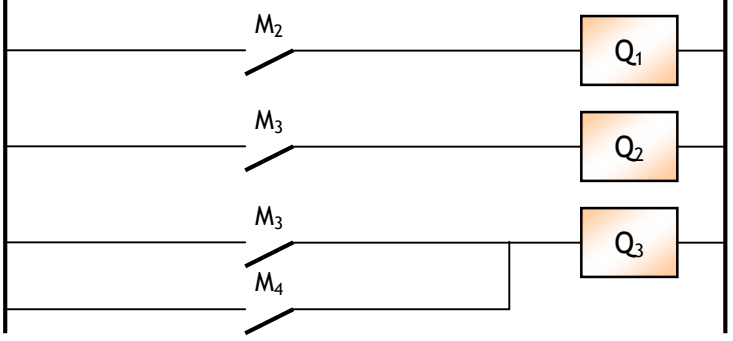
L'étape 2 (M_2) doit être active lorsque l'étape 1 est active (M_1=1) et la réceptivité I_1 est vraie (I_1=1) :

$$SM_2 = M_1.I_1$$

L'étape initiale 2 (M_2) doit être désactivée lorsque l'étape 3 est active (M_3=1) :

$$RM_2 = M_3$$

2. Traitement postérieur :



Q_1 = 1 si on est dans l'étape 2 :

$$Q_1 = M_2$$

Q_2 = 1 si on est dans l'étape 3 :

$$Q_2 = M_3$$

Q_3 = 1 si on est dans l'étape 3 ou dans l'étape 4 :

$$Q_3 = M_3 + M_4$$

DRES 03

Les blocs fonctions 'COMPTEUR'

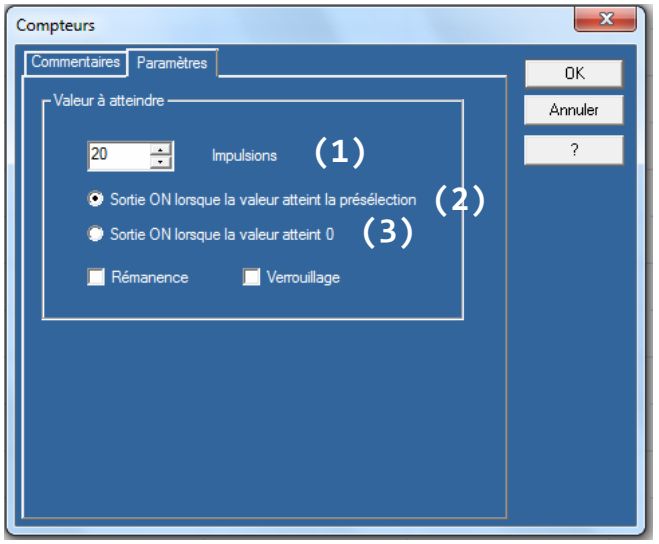
1. Présentation :

Le module logique Zelio **SR2A201BD** possède 16 blocs compteurs de C_1 à C_G.

No					Commentaire
01	C1	C	D	R	
02	C2	C	D	R	
03	C3	C	D	R	
04	C4	C	D	R	
05	C5	C	D	R	
06	C6	C	D	R	
07	C7	C	D	R	
08	C8	C	D	R	
09	C9	C	D	R	
10	CA	C	D	R	
11	CB	C	D	R	
12	CC	C	D	R	
13	CD	C	D	R	
14	CE	C	D	R	
15	CF	C	D	R	
16	CG	C	D	R	

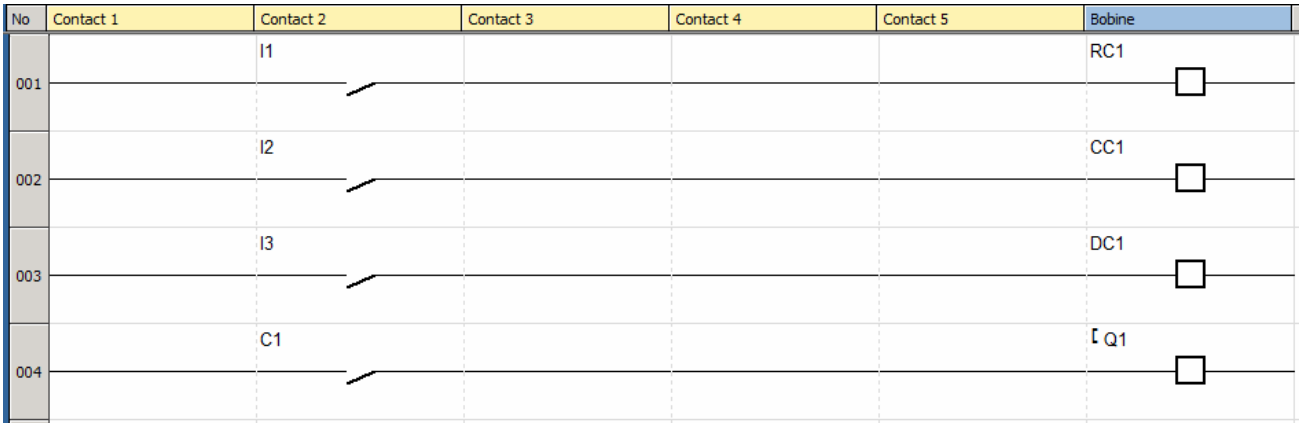
Cette fonction permet de compter ou décompter des impulsions jusqu'à une valeur de présélection définie dans la fenêtre de paramétrage.
Le bloc fonction Compteur possède une entrée comptage **CC_X** (à chaque excitation de la bobine, le compteur s'incrémente ou se décrémente de 1 selon le sens de comptage choisi), une entrée Remise à zéro **RC_X**, une entrée sens de comptage **DC_X** (le bloc décompte si cette entrée est activée) et une sortie **C_X** permettant de savoir le niveau commandé par le compteur. Lorsque la valeur de présélection est atteinte, cette sortie passe à 1 jusqu'à la remise à zéro ou le comptage dans l'autre sens. La valeur de comptage et la valeur de présélection peuvent être visualisées à l'écran du module.

2. La fenêtre de paramétrage



- Le champ **(1) :** permet de saisir la valeur à atteindre (valeur de présélection) ;
- Le champ **(2) :** compter vers la présélection, alors la sortie Cx est ON lorsque la valeur atteint la présélection ;
- Le champ **(3) :** décompter à partir de la présélection, alors la sortie Cx est ON lorsque la valeur atteint 0.

3. Exemple :



A chaque appui sur I_2, le compteur (**CC_1**) s'incrémente. La fermeture de **I_3** change la direction de comptage (**DC_1**), le compteur se décrémente. Quand la valeur de présélection (ici 20) est atteinte, **C_1** sera à l'état haut, la sortie **Q_1** aussi. L'appui sur **I_1** remet le compteur à zéro (**RC_1**).

DRES 04

Microcontrôleur PIC 16F84

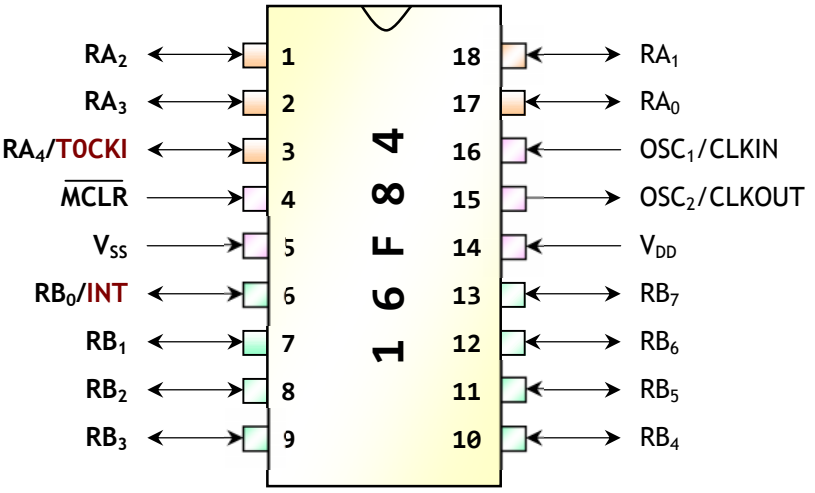
1. Présentation :

Le PIC16F84 est logé dans un boîtier 18 broches.

Caractéristiques principales :

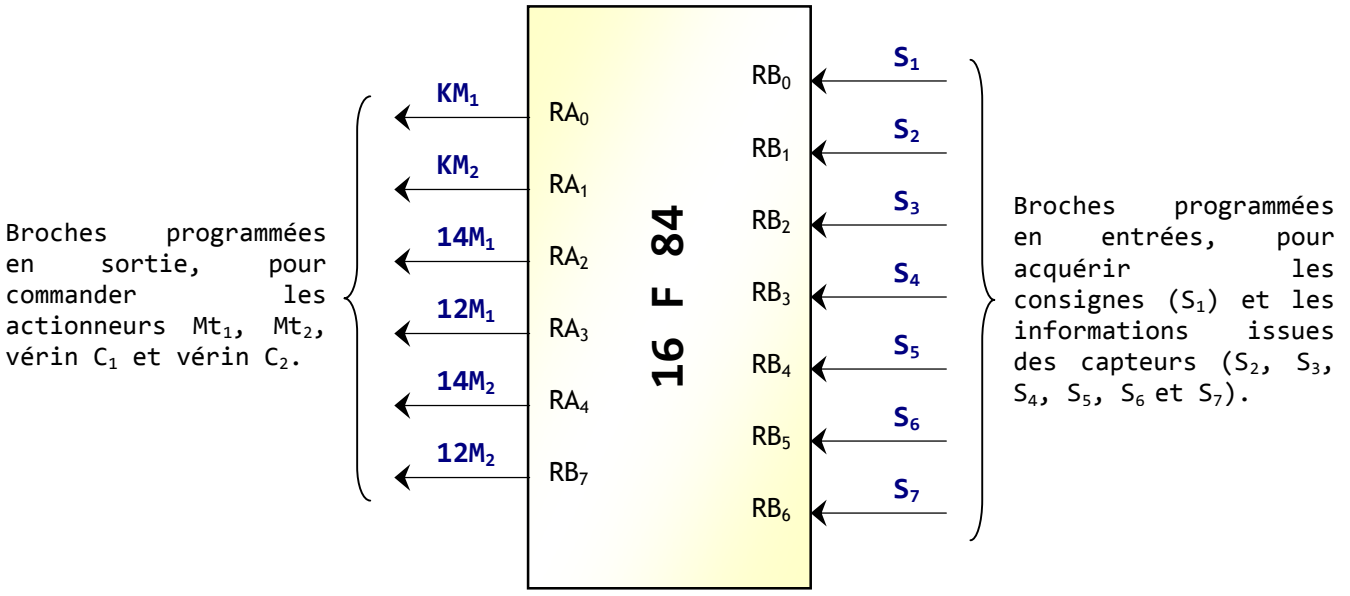
- 1K de mémoire programme ;
- 68 octets de RAM ;
- 64 octets D'EEPROM ;
- 13 entrées/sorties réparties en 2 ports : PORTA = 5 bits et PORTB = 8 bits ;
- 4 sources d'interruption ;
- 1 Timer/Compteur ;
- 1 Chien de garde ;
- MODE SLEEP (pour une faible consommation) ;
- 4 Sources d'oscillateur sélectionnable ;
- Protection du code ;
- 35 instructions seulement.

Brochage :



V_{DD}, V_{SS} : Broches d'alimentation (3 à 5.5 v) ;
OSC_1, OSC_2 : Signaux d'horloge, peuvent recevoir un circuit RC ou un résonateur ;
CLKINT : Peut être connectée à une horloge externe de 0 à 4, 10 ou 20 Mhz ;
MCLR : Reset ou Master Clear;
T0CKI : entrée d'horloge externe du Timer 0 ;
INT : entrée d'interruption externe ;
RA_0...RA_4 : 5 E/S du PORTA ;
RB_0...RB_7 : 8 E/S du PORTB.

2. Affection des entrées/sorties :

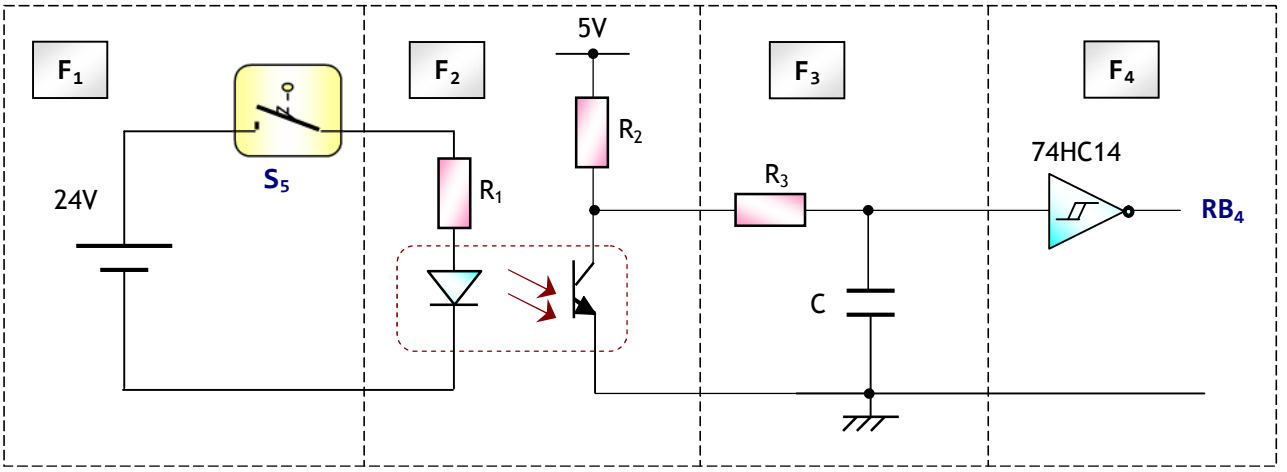


Le registre d'adresse $0C_H$ (case mémoire interne du PIC 16F84) est utilisé pour compter le nombre de morceaux coupés.

Le microcontrôleur reçoit les informations issues des capteurs et les consignes sur un PORT d'entrée, les traite et élabore les ordres pour commander les actionneurs qui sont connectés sur un PORT de sortie.

DRES 05 — **Lecture et écriture des entrées/sorties**

1. Acquisition de présence (présence barre) :

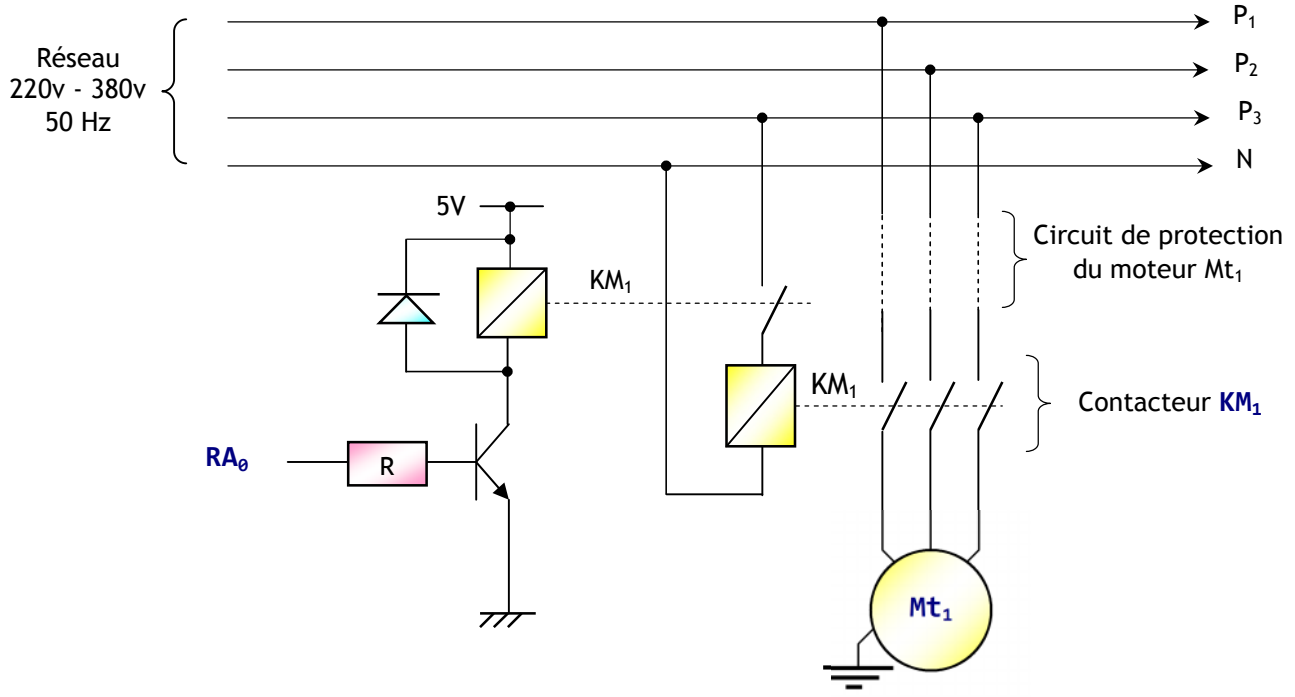


Le capteur S_5 est utilisé pour détecter la présence de la barre.

Lorsque la barre est introduite entre les 2 rouleaux, par contact, l'interrupteur S_5 (capteur) se ferme. La diode s'allume, le phototransistor se sature et RB_4 passe à 1(compte rendu : Présence barre).

Les informations concernant l'ordre départ cycle, l'état de la barre et de la tronçonneuse sont fournies par le bouton S_1 et les capteurs S_2, S_3, S_4, S_5, S_6, S_7. Ces informations sont acquises sur les broches RB_0, RB_1, RB_2, RB_3, RB_4, RB_5 et RB_6. Donc le PORTB doit être programmé en ENTREE sauf la broche RB_7 qui doit être programmée en SORTIE.

2. Commande des actionneurs (Mt_1) :



Le relais KM_1 est commandé par la broche RA_0 du microcontrôleur ; le PIC 16 F 84.
Si RA_0 = 1 alors le transistor est saturé, la bobine du relais KM_1 est excitée ; les contacts sont fermés et le moteur Mt_1 est alimenté (Action : Positionner la barre).

Les broches RA_0, RA_1, RA_2, RA_3, RA_4 et RB_7 sont utilisées pour commander respectivement le moteur Mt_1, le moteur Mt_2, le vérin C_1 et le vérin C_2. Donc le PORTA et la broche RB_7 du PORTB doivent être programmés en SORTIE.

DRES 06

Jeu d'instruction du microcontrôleur 16F84

1. Jeu d'instruction :

Mnemonic, operonds		Description	Cycles	14-bit opcode MSB LSB	Status affected
BYTE ORIENTED FILE REGISTER OPERATIONS					
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C, DC, Z
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z
CLRF	f	Clear f	1	00 0001 1fff ffff	Z
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z
DECFSZ	f, d	Decrement f, skip if 0	1(2)	00 1011 dfff ffff	
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z
INCFSZ	f, d	Increment f, skip if 0	1(2)	00 1111 dfff ffff	
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z
MOVWF	f	Move W to f	1	00 0000 1fff ffff	
NOP	-	No operation	1	00 0000 0xx0 0000	
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C
SUBWF	f, d	Subsract W from f	1	00 0010 dfff ffff	C, DC ,Z
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff	
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z
BIT ORIENTED FILE REGISTER OPERATIONS					
BCF	f, b	Bit clear f	1	01 00bb bfff ffff	
BSF	f, b	Bit set f	1	01 01bb bfff ffff	
BTFSC	f, b	Bit test f, skip if clear	1(2)	01 10bb bfff ffff	
BTFSS	f, b	Bit test f, skip if set	1(2)	01 11bb bfff ffff	
LITERAL AND CONTROL OPERATIONS					
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C, DC, Z
ANDLW	k	AND literal With W	1	11 1001 kkkk kkkk	Z
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk	___ ___
CLRWDT	-	Clear watchdog Timer	1	00 0000 0101 0100	$\overline{TO}$, $\overline{PD}$
GOTO	k	Go to address	2	10 1kkk kkkk kkkk	
IORLW	k	Inclusive OR literal With W	1	11 1000 kkkk kkkk	Z
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk	
RETFIE	-	Return from interrupt	2	00 0000 0000 1001	
RETLW	k	Return with literal to W	2	11 01xx kkkk kkkk	
RETURN	-	Return from subroutine	2	00 0000 0000 1000	___ ___
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$, $\overline{PD}$
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C, DC, Z
XORLW	k	Exclusive OR literal With W	1	11 1010 kkkk kkkk	Z

2. Configuration des PORTS :

Tous les ports sont pilotés par deux registres :

- Le registre de **PORTx**, si le **PORTx** ou certaines lignes de **PORTx** sont configurées en sortie, ce registre détermine l'état logique des sorties ;
- Le registre **TRISx**, c'est le registre de direction. Il détermine si le **PORTx** ou certaines lignes de Port sont en entrée ou en sortie. L'écriture d'un **1** logique correspond à une **entrée** (**1** comme **I**nput) et l'écriture d'un **0** logique correspond à une sortie (**0** comme **O**utput) ;
- Les registres **TRISx** appartiennent à la **BANQUE 1** des **SFR**. Lors de l'initialisation du **µC** il ne faut pas oublier de changer de page mémoire pour les configurer ;
- Pour accéder aux banques mémoire, on utilise le bit **RP_0** ($5^{ième}$ bit du registre **STATUS**) :

RP_0 = 0	Accès à la banque 0	RP_0 = 1	Accès à la banque 1

DREP 01

ANALYSE FONNCTIONNELLE GLOBALE

Q.1:

.....

Q.2:

.....

Q.3:

Energie [...] Matière [...] Information [...]

Q.4:

.....

Q.5:

S_2 : .....

S_3 : .....

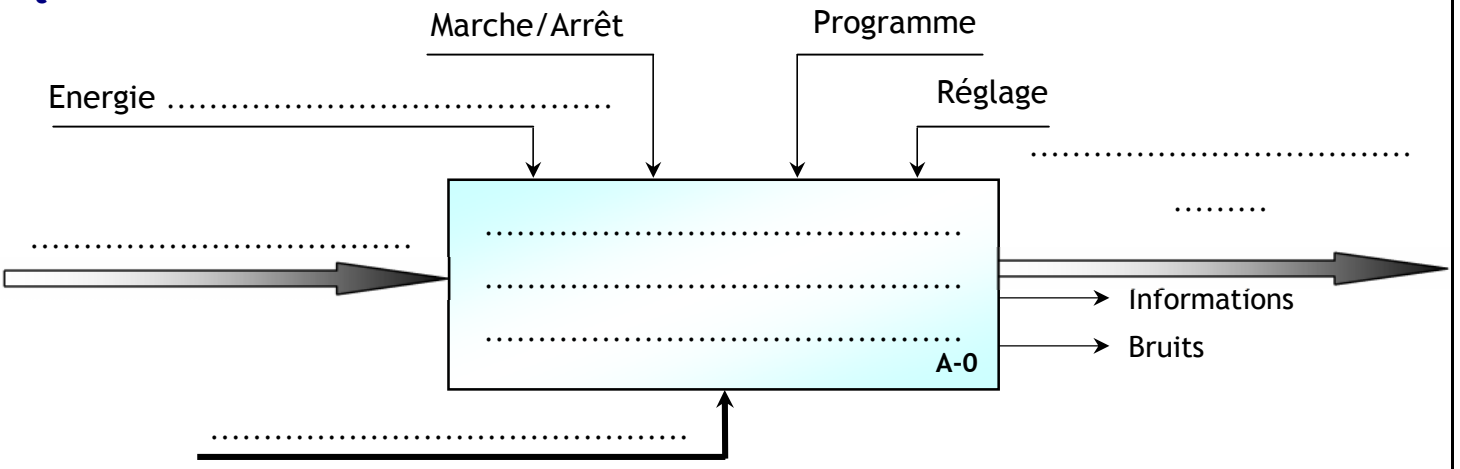
S_4 : .....

S_5 : .....

Q.6:

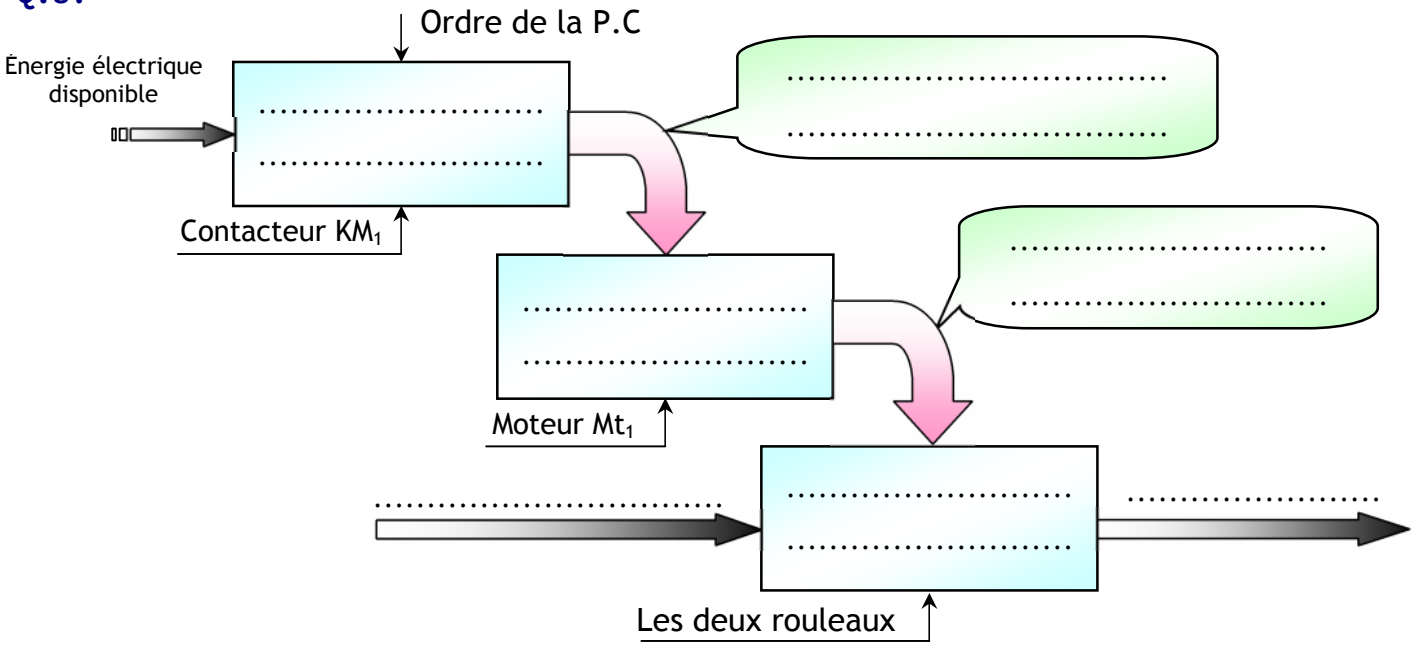
.....

Q.7:



ANALYSE FONNCTIONNELLE STRUCTURELLE

Q.8:



Q.9:

DREP 02

Fonction de service

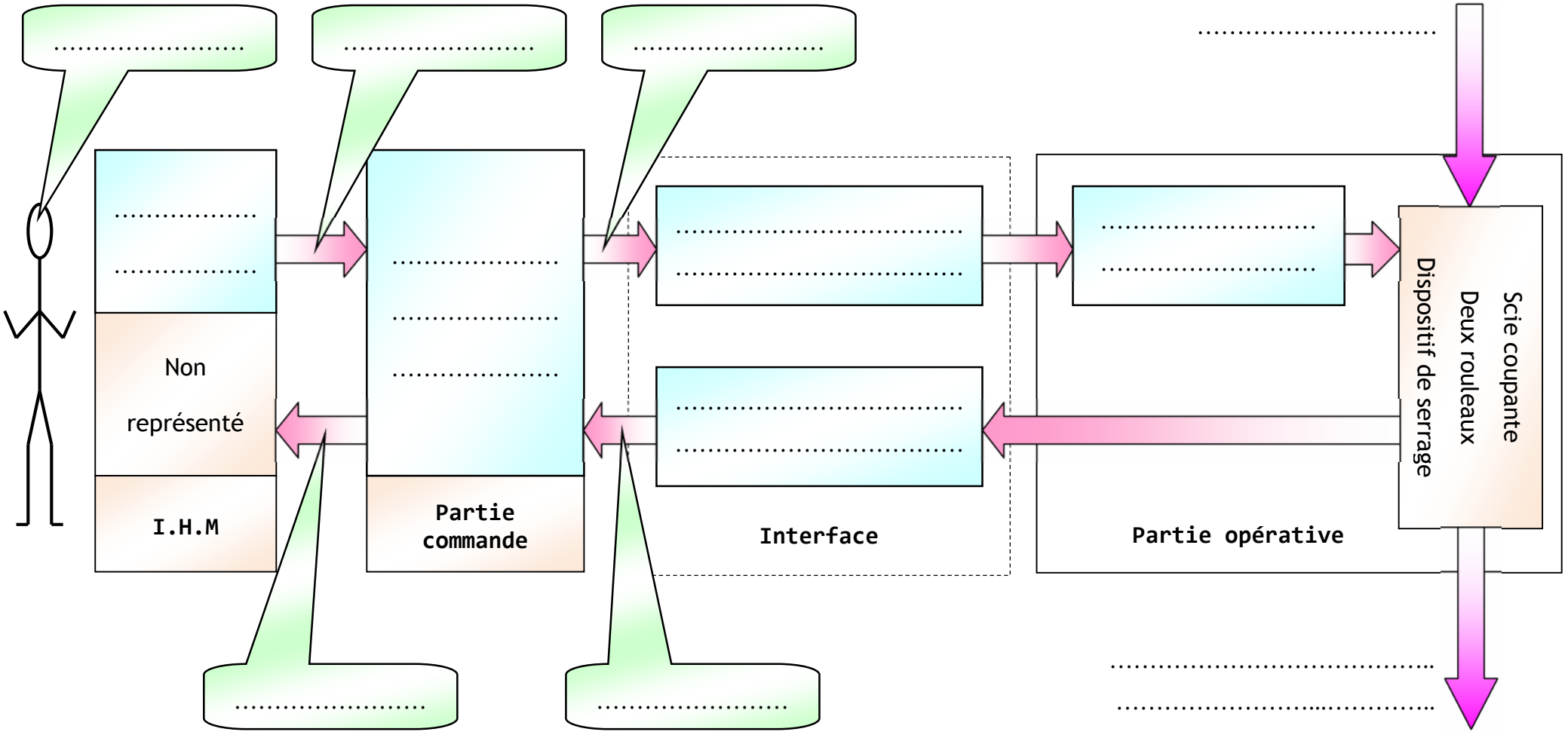
Découper des barres d'aluminium en forme de profilé «U», en 20 morceaux

Fonctions techniques

				Solutions constructives
FT_1	Positionner la barre	FT_{11}	Transformer l'énergie électrique en énergie mecanique de rotation	.....
		FT_{12}	Entrainer la barre	.....
FT_2	Serrer/Desserrer la barre	FT_{21}	Transformer l'énergie électrique en énergie mecanique de translation	.....
		FT_{22}	Serrer/Desserrer la barre	Dispositif de serrage
FT_3	Positionner la tronçonneuse	FT_{41}	Animer la tronçonneuse d'un mouvement de translation	.....
FT_4	Couper la barre	FT_{42}	Animer la tronçonneuse d'un mouvement de rotation	.....
		FT_{43}	Couper la barre	.....
FT_5	Saisir le nombre de morceaux coupés			.....
FT_6	Traiter les informations			.....

DREP 03

Q.10:



Q.11:

.....

DREP 04

L'OUTIL GRAPHIQUE NORMALISÉ 'GRAFCET'

Q.12:

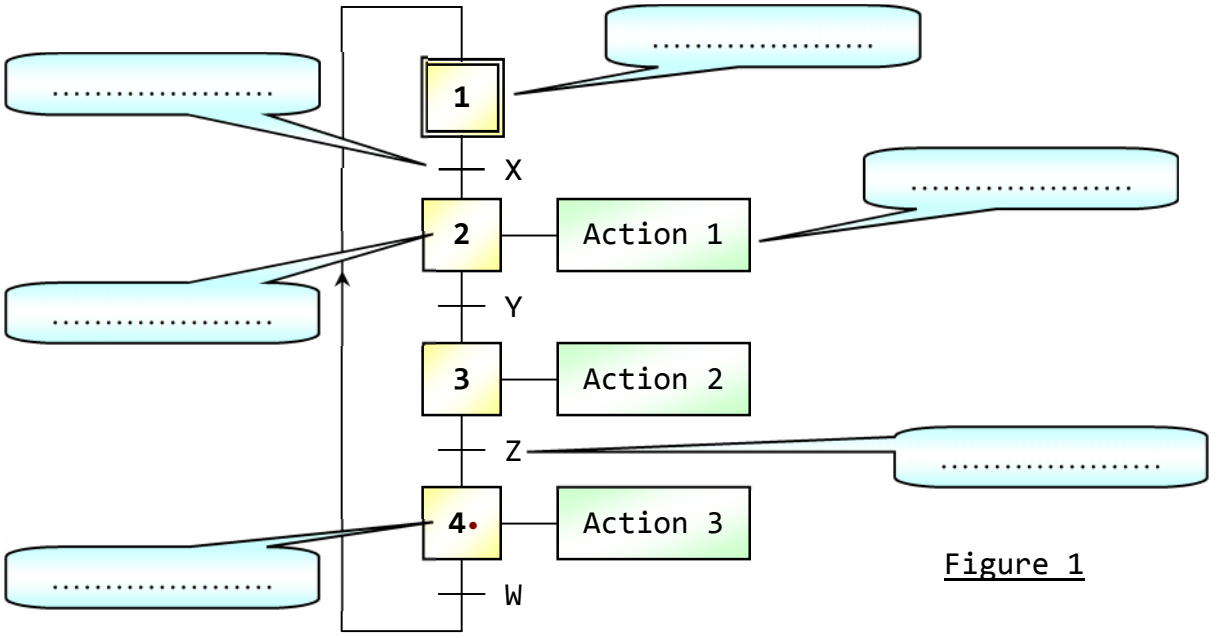


Figure 1

Q.13:

Réceptivité	Validée
Transition	Vraie
Étape	Active
	Non validée
	Fausse
	Inactive

Figure 2

Q.14:

- Est-ce que l'étape 5 est active : .....
- Est-ce que l'étape 10 est active : .....
- La transition T_{6-7} est elle validée : .....
- Pourquoi : .....
- La transition T_{5-6} est elle validée : .....
- Pourquoi : .....
- La transition T_{5-10} est elle franchissable : .....
- Pourquoi : .....

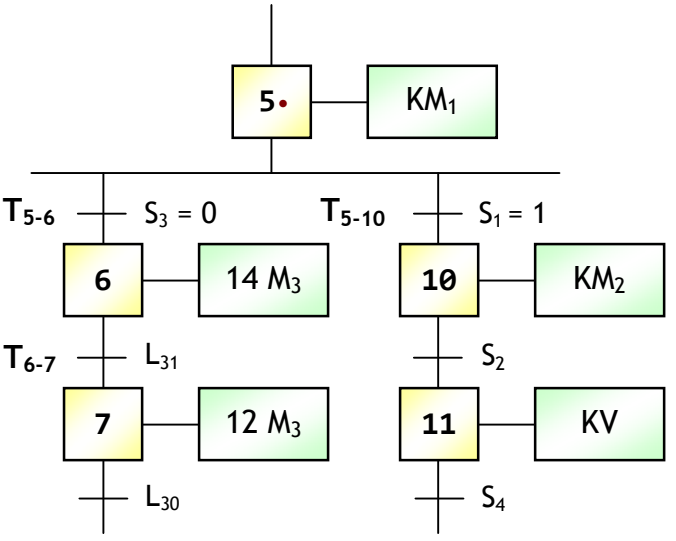
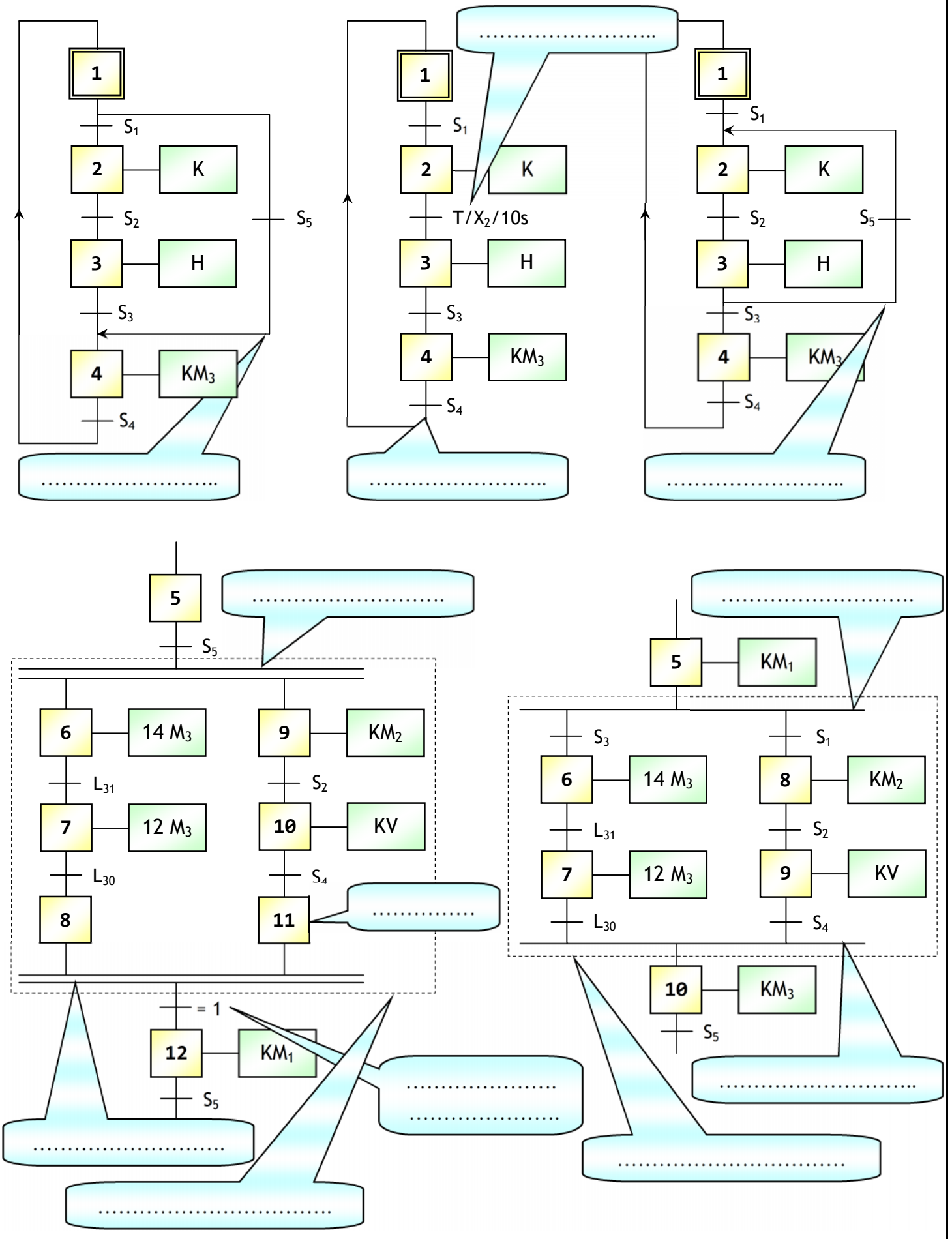


Figure 3

Q.15:

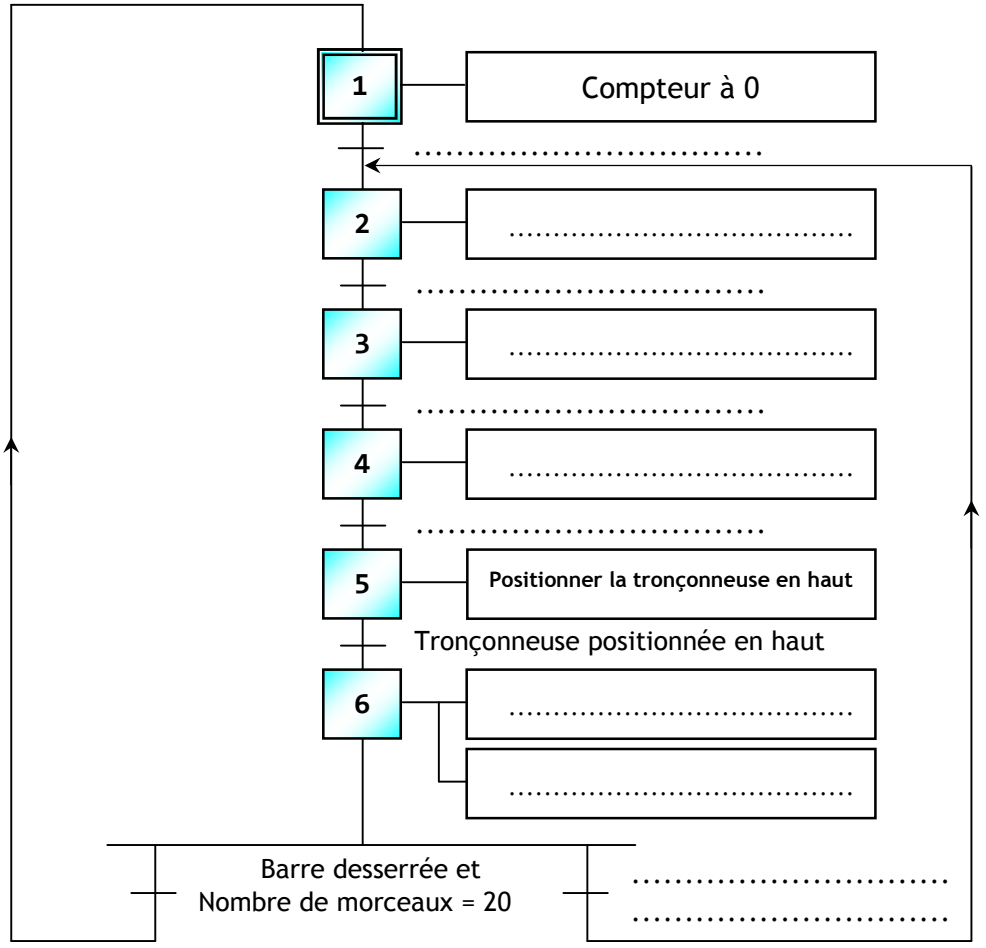
DREP 05


1
S1
2
K
S2
S5
3
H
S3
4
KM3
S4
.....
T/X2/10s
5
S5
6
14 M3
9
KM2
L31
S2
7
12 M3
10
KV
L30
S4
8
11
= 1
12
KM1
KM1
S3
S1
8
L31
S2
9
L30
S4
10
S5

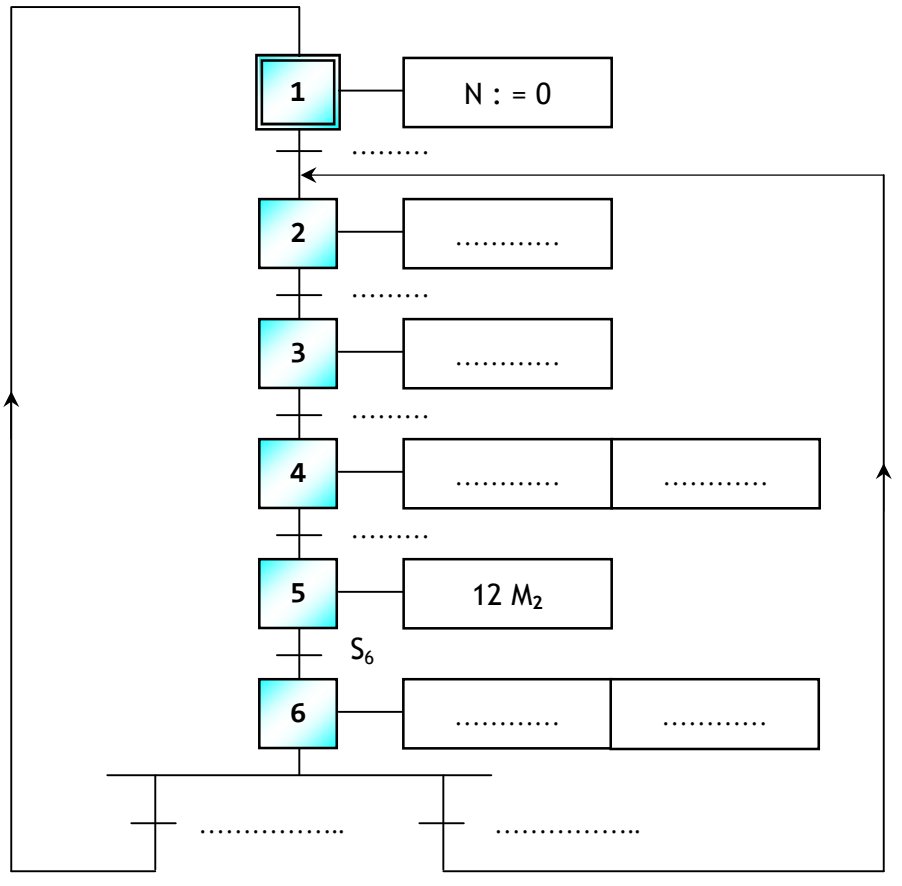
DREP 06

GRAFCET DE FONNCTIONNEMENT DU SYSTÈME

Q.16:



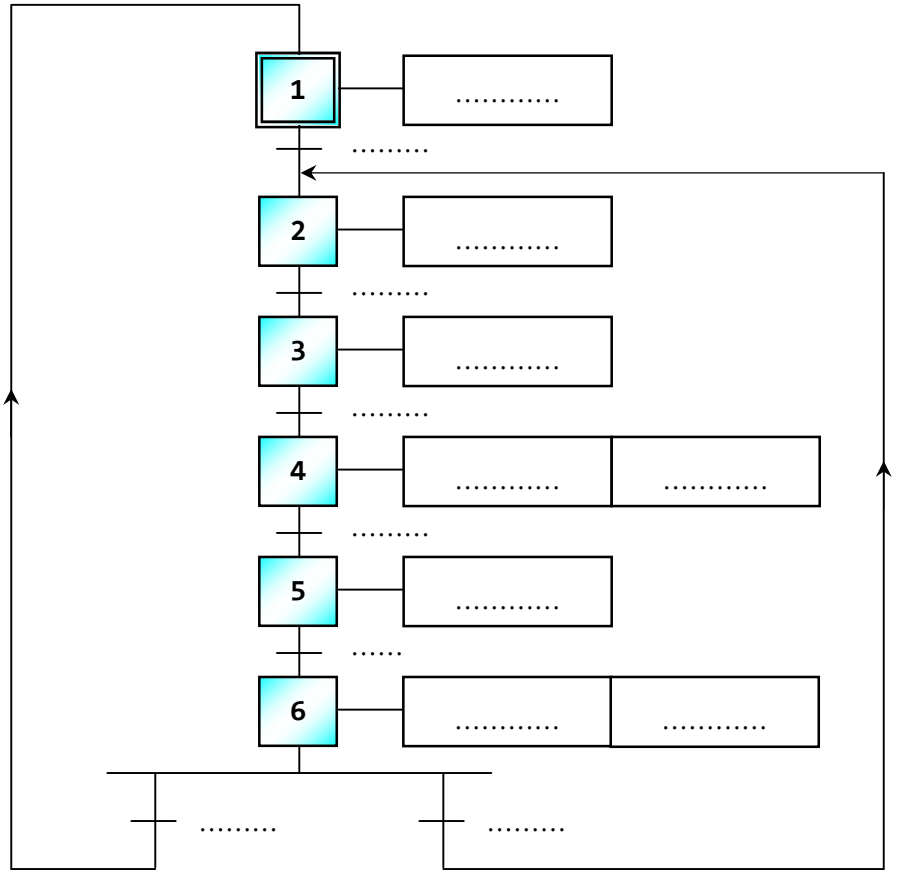
Q.17:



PROGRAMMATION EN LANGAGE LADDER

DREP 07

Q.18:



Q.19:

Traitement séquentiel :

N° d'étape	Activation	Désactivation
1	SM_1 = ... + Init	RM_1 = ...
2	SM_2 = ...	RM_2 = ...
3	SM_3 = ...	RM_3 = ...
4	SM_4 = ...	RM_4 = ...
5	SM_5 = ...	RM_5 = ...
6	SM_6 = ...	RM_6 = ...

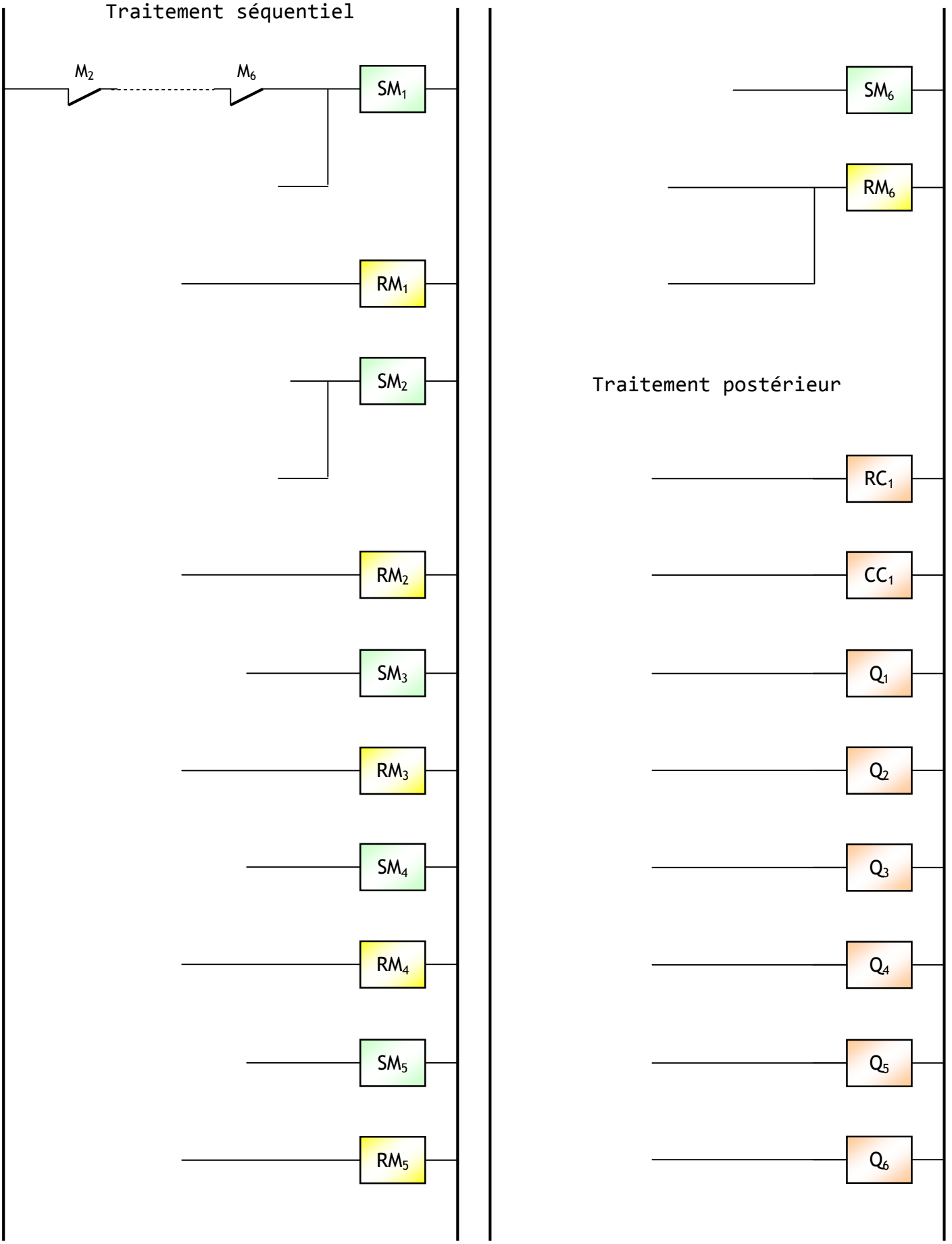
Init = $\overline{M_2}.\overline{M_3}.\overline{M_4}.\overline{M_5}.\overline{M_6}$

Traitement postérieur :

Actions	
Q_1 = ...	Q_5 = ...
Q_2 = ...	Q_6 = ...
Q_3 = ...	RC_1 = ...
Q_4 = ...	CC_1 = ...

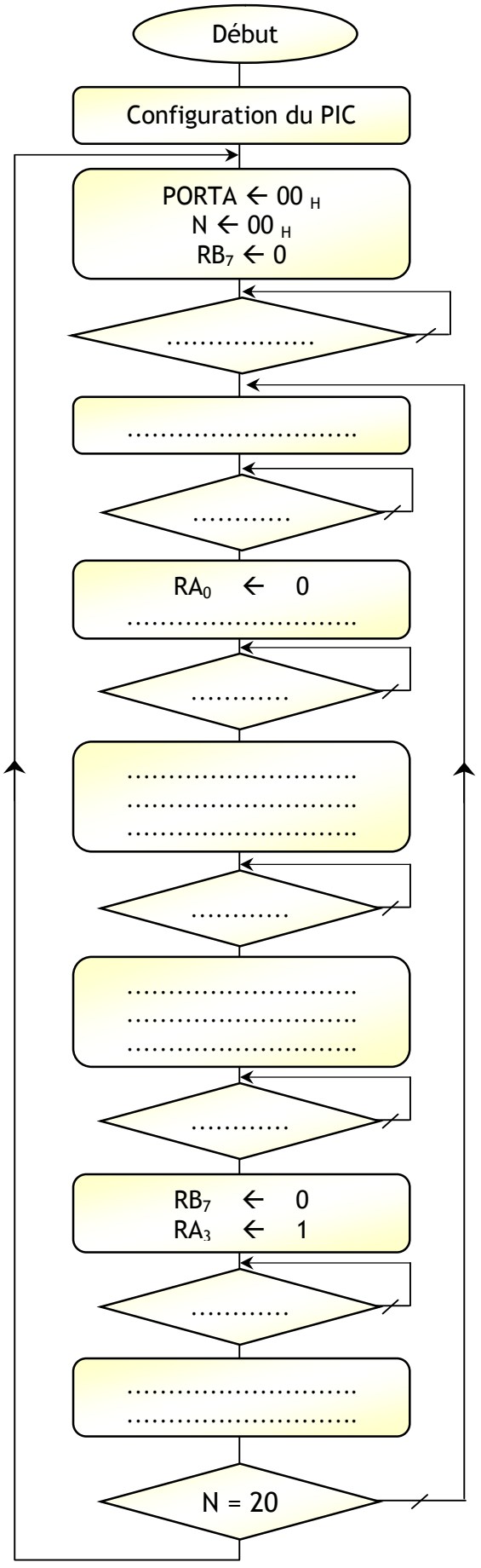
Q.20:

DREP 08


Traitement séquentiel
M2
M6
SM1
RM1
SM2
RM2
SM3
RM3
SM4
RM4
SM5
RM5
SM6
RM6
Traitement postérieur
RC1
CC1
Q1
Q2
Q3
Q4
Q5
Q6

DREP 09

Q.23: ORGANIGRAMME DE FONCTIONNEMENT



CONDITIONNEMENT DES SIGNAUX

Q.21:

.....

.....

.....

.....

.....

.....

.....

.....

Q.22:

Bloc F_1

Nom	.....
Rôle	.....

Bloc F_2

Nom	.....
Rôle	.....

Bloc F_3

Nom	.....
Rôle	.....

Bloc F_4

Nom	.....
Rôle	.....

Q.24: PROGRAMME D'INITIALISATION

..... ; Accès à la BANK 1

..... ; PORTA en sortie

..... ;

..... ; Configuration du PORTB

..... ; Accès à la BANK 0

DREP 10

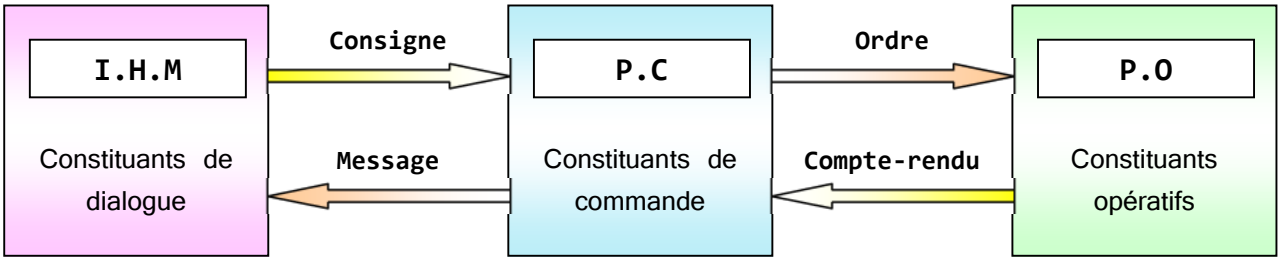
Q.25: PROGRAMME DE FONCTIONNEMENT

```
LAB0        CLRF      PORTA          ; État de repos (aucune action)
            BCF       PORTB, 7       ;
            CLRF      0x0C           ; Compteur à 0
LAB1        MOVF      PORTB, W       ;
            ANDLW     B'00010001'    ;
            SUBLW     B'00010001'    ; Présence barre et Départ cycle?
            BTFSS     STATUS, Z      ;
            GOTO      LAB1           ;
Reprendre   .....                    ; Positionner la barre
LAB2        .....                    ; Barre positionnée?
            .....                    ;
            .....                    ;
            .....                    ; Serrer la barre
LAB3        .....                    ; Barre serrée?
            .....                    ;
            .....                    ;
            .....                    ;
            .....                    ; Couper la barre
LAB4        .....                    ; Barre coupée?
            .....                    ;
            .....                    ;
            .....                    ;
            .....                    ; Positionner la tronçonneuse
LAB5        .....                    ; Tronçonneuse positionnée?
            .....                    ;
            .....                    ;
            .....                    ; Desserrer la barre
LAB6        .....                    ; Barre desserrée?
            .....                    ;
            .....                    ;
            INCF      0x0C, 1        ; Incrémenter compteur
            MOVF      0x0C, W        ;
            SUBLW     D'20'          ;
            BTFSS     STATUS, Z      ; Compteur = 20
            GOTO      Reprendre      ; Couper encore la barre
            GOTO      LAB0           ; La barre est coupée en 20 morceaux
            END                      ; Fin du fichier
```

1. Structure d'un système automatisé :

Un système automatisé se compose de deux parties qui coopèrent :

- Une partie opérative constituée du processus à commander, des actionneurs qui agissent sur ce processus et des capteurs permettant de mesurer son état.
- Une partie commande qui élabore les ordres pour les actionneurs en fonction des informations issues des capteurs et des consignes. Cette partie commande peut être réalisée par des circuits câblés, ou par des dispositifs programmables (automates, calculateurs).



2. Cahier des charges d'un automatisme logique :

Le cahier des charges décrit :

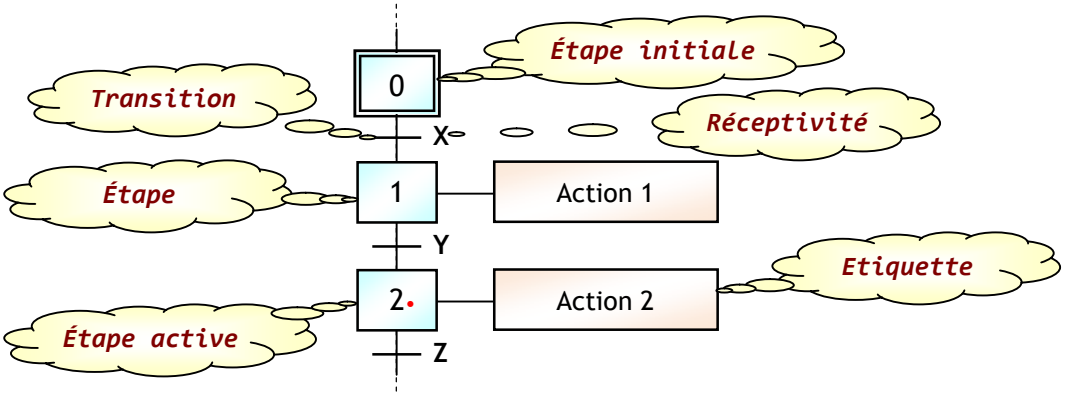
- les relations entre la partie commande et la partie opérative ;
- les conditions d'utilisation et de fonctionnement de l'automatisme.

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

3. GRAFCET :

3.1. Définition :

- Le GRAFCET (Graphe de Contrôle Etape-Transition) est un outil graphique normalisé permettant de spécifier le cahier des charges d'un automatisme séquentiel ;
- Le GRAFCET est une représentation graphique alternée d'étapes et de transitions. Une seule transition doit séparer deux étapes ;
- L'étape caractérise un comportement invariant du système technique au moment donnée, elle peut être active ou inactive ;
- L'action associée à une étape quelconque caractérise ce que doit faire le système lorsque cette dernière est active ;
- Les actions associées aux étapes sont inscrites dans les étiquettes ;
- La transition indique le critère d'évolution entre deux étapes consécutives ;
- Une réceptivité est une condition logique qui conditionne la transition d'une étape à la suivante ;
- l'étape initiale est représentée par un double carreau ;
- Les étapes qui se succèdent sont reliées par des segments orientés auxquels sont associées des transitions (trais horizontaux coupant les segments orientés).



3.2. Règles de syntaxe :

- **Règle N°1** «situation initiale» :

L'étape initiale est considérée initialement active ce qui permet l'évolution de l'automatisme des que la réceptivité associée à la première transition devient vraie.

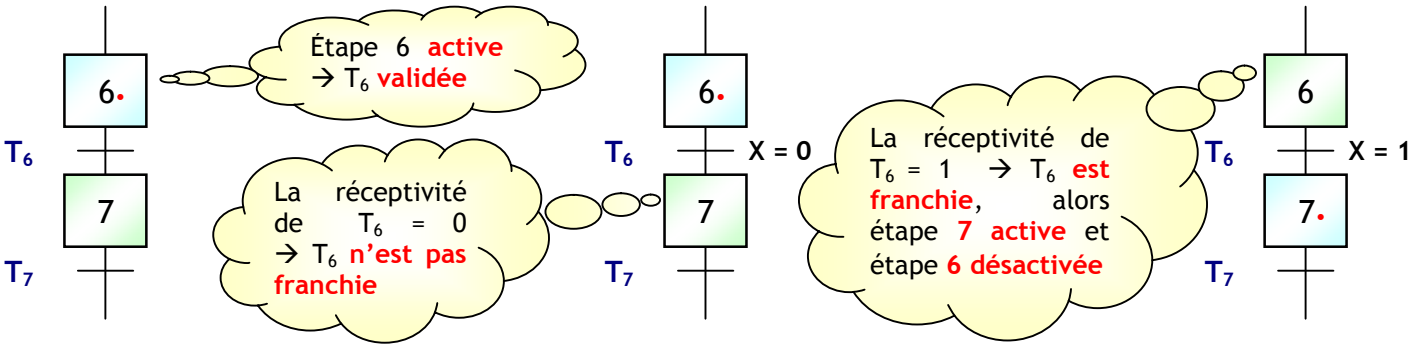
- **Règle N°2** «franchissement d'une transition» :

Une transition est franchie lorsque l'étape associée est active et la réceptivité associée à cette transition est vraie.

- **Règle N°3** «évolution des étapes actives» :

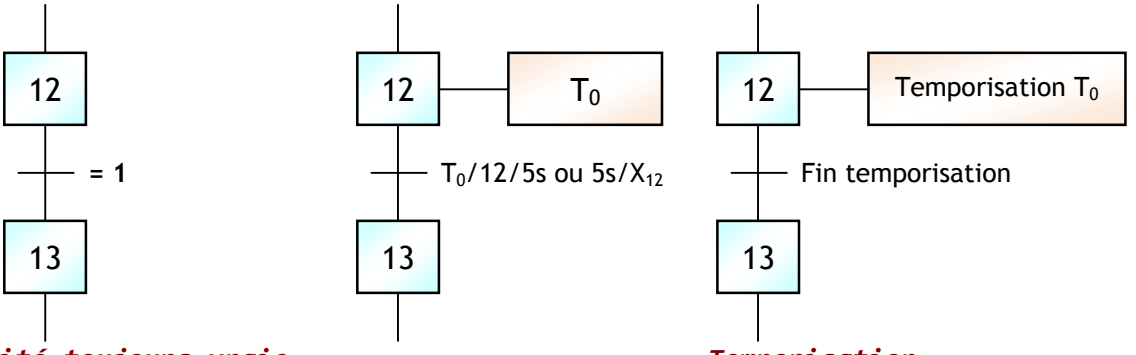
Le franchissement d'une transition provoque simultanément :

- la désactivation de toutes les étapes immédiatement précédentes reliées à cette transition ;
- l'activation de toutes les étapes immédiatement suivantes reliées à cette transition.



Il y a des cas particuliers de réceptivité, on en cite 2 :

- **Temporisation :** Pour faire intervenir le temps dans une réceptivité, il suffit d'indiquer après le repère "**t**" son origine et sa durée. L'origine sera l'instant de début de l'activation de l'étape déclenchant la temporisation. La notation **t/12/5s** signifie que la réceptivité sera vraie **5** secondes après l'activation de l'étape repérée **12**. La notation normalisée s'écrit **5s/X12** ;
- **Réceptivité toujours vraie :** une telle réceptivité s'écrit "**= 1**". Le franchissement de cette transition se fera dès que la ou les étapes immédiatement antérieures seront actives sans autre condition.



Réceptivité toujours vraie *Temporisation*

4. Structure de base d'un GRAFCET :

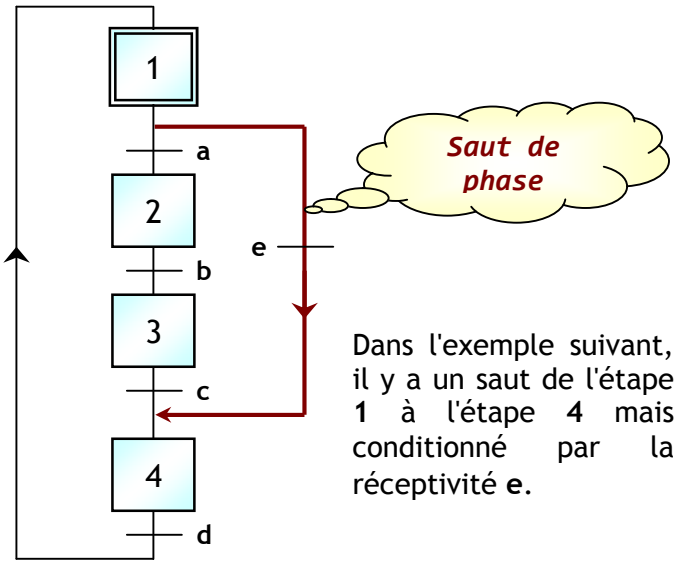
4.1. La séquence linéaire :

Une séquence linéaire est composée d'un ensemble d'étapes successives où chaque étape est suivie d'une seule transition et chaque transition n'est validée que par une seule étape.

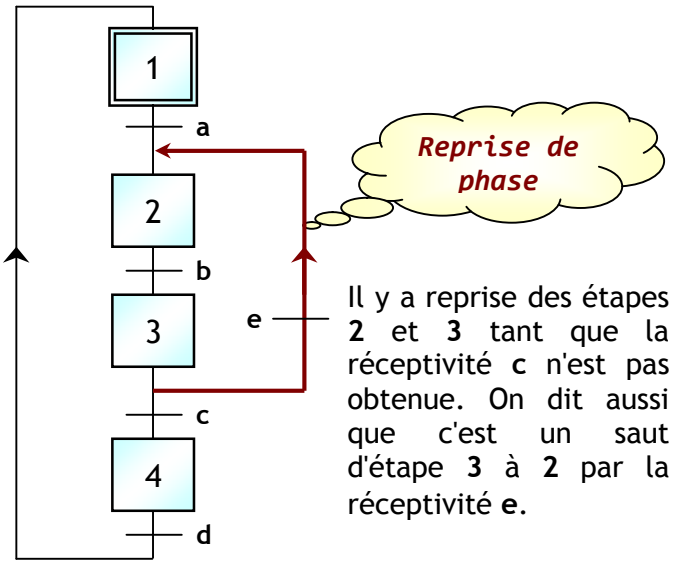
4.2. Saut en avant et saut en arrière :

- Le saut en avant permet de sauter 1 ou plusieurs étapes lorsque les actions à réaliser deviennent inutiles.

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

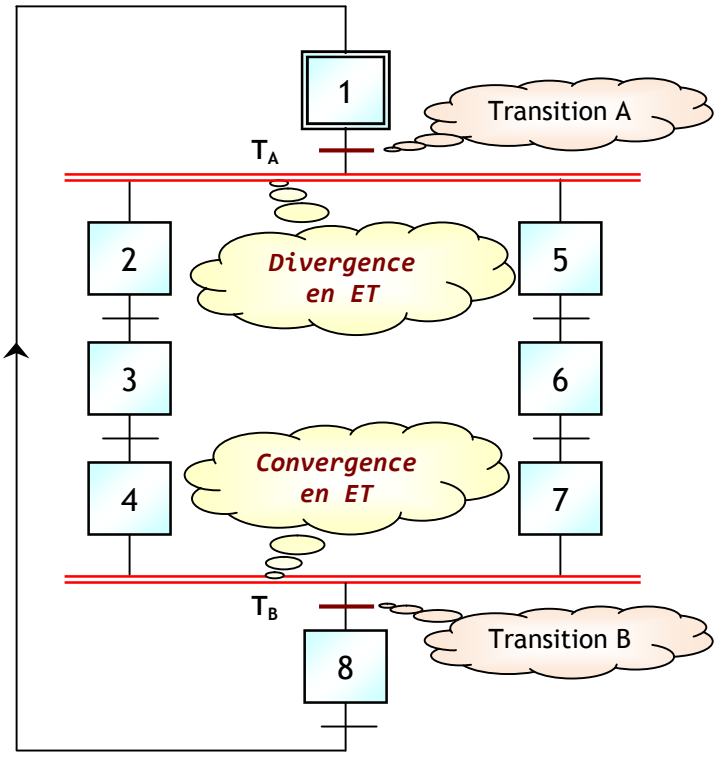


Dans l'exemple suivant, il y a un saut de l'étape **1** à l'étape **4** mais conditionné par la réceptivité **e**.



Il y a reprise des étapes **2** et **3** tant que la réceptivité **c** n'est pas obtenue. On dit aussi que c'est un saut d'étape **3** à **2** par la réceptivité **e**.

4.3. Les séquences simultanées :

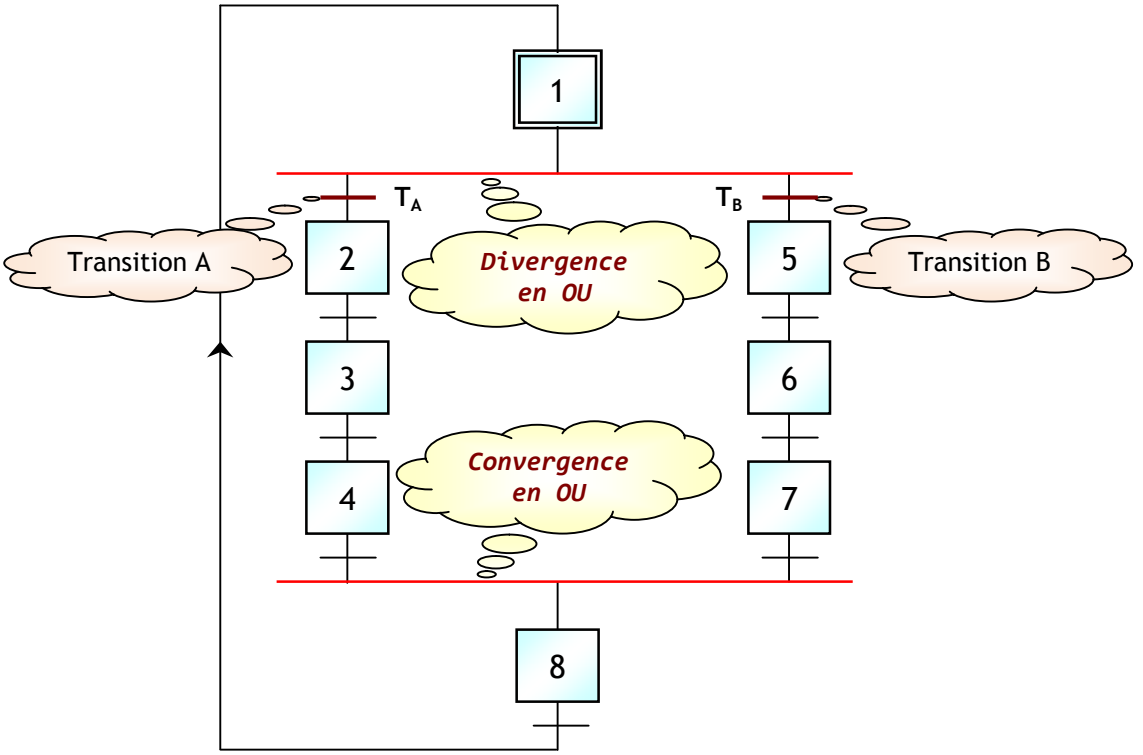


Lorsque le franchissement d'une transition conduit à activer simultanément plusieurs séquences d'étapes, on obtient des séquences simultanées qui s'exécuteront parallèlement mais indépendamment. C'est-à-dire, l'évolution de chacune des séquences d'étapes dépendra des conditions d'évolution du système automatisé.

- **Divergence en ET :** lorsque la transition T_A est franchie, les étapes **2** et **5** sont actives.
- **Convergence en ET :** la transition T_B sera validée lorsque les étapes **4** et **7** seront actives et si la réceptivité associée à cette transition est vraie, alors celle-ci est franchie.
- **Remarque :**
  - Après une divergence en ET, on trouve une convergence en ET ;
  - Le nombre de branches parallèles peut-être supérieur à 2 ;

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

4.4. Sélection de séquences :



Une structure alternative permet d'effectuer un choix unique d'évolution entre plusieurs étapes en aval à partir d'une seule étape en amont.

- **Divergence en OU :** l'évolution du système vers une branche dépend des réceptivités des transitions T_A et T_B associées aux transitions.
- **Convergence en OU :** après l'évolution dans une branche, il y a convergence vers une étape commune.
- **Remarque :**
  - A et B ne peuvent être vrais simultanément (conflit) ;
  - Après une divergence en OU, on trouve une convergence en OU ;
  - Le nombre de branches peut-être supérieur à 2 ;
  - La convergence de toutes les branches ne se fait pas obligatoirement au même endroit.

5. Différents points de vue d'un GRAFCET :

Suivant les différents points de vue (utilisateur, technico-commercial, concepteur-réalisateur, etc.), on peut distinguer plusieurs 3 types de GRAFCET :

- GRAFCET d'un point de vue du système ;
- GRAFCET d'un point de vue de la partie opérative (P.O) ;
- GRAFCET d'un point de vue de la partie commande (P.C).

NB :

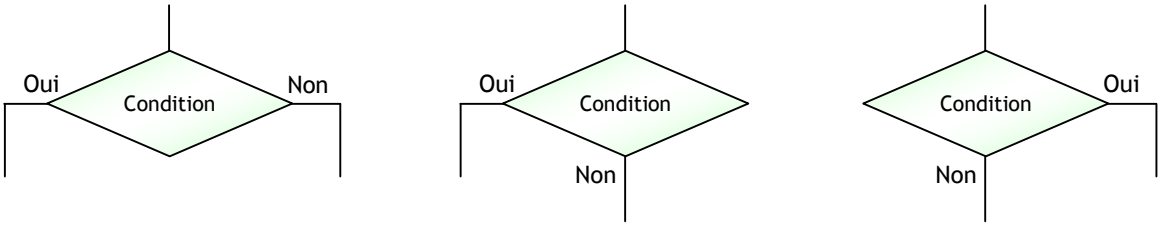
Le GRAFCET d'un point de vue de la P.O et le GRAFCET d'un point de vue de la P.C doivent avoir le même nombre d'étapes.

1. Organigramme :

1.1. Définition :

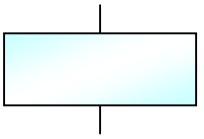
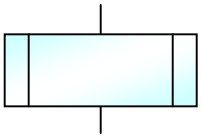
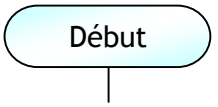
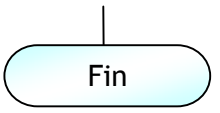
L'organigramme est la représentation graphique de l'algorithme, il permet de représenter chaque opération élémentaire au moyen d'un symbole graphique normalisé.

1.2. Symboles de test logique :



L'opération de test logique se fait sur une condition. Le résultat de cette opération implique le choix d'une voix parmi plusieurs. Le symbole de test logique est couramment employé pour représenter une décision ou un aiguillage.

1.3. Symboles de traitement :

 Symbole de renvoi utilisé 2 fois pour assurer la continuité lorsqu'une partie de la ligne de liaison n'est pas représentée	 Opération ou groupe d'opérations sur des données, instructions ou opération pour laquelle il n'existe aucun symbole normalisé.	 Commentaires : symbole utilisé pour donner des indications marginales
 Groupe d'opérations considérées comme une seule opération sous programme.	 Début d'un programme	 Fin d'un programme

1.4. Règles de construction :

- Centrer l'organigramme sur une feuille ;
- Construire l'organigramme afin que sa lecture s'effectue verticalement ;
- Les lignes de liaison entre symboles ne doivent pas en principe se couper ;
- Une ligne de liaison doit toujours arriver sur le haut et au centre d'un symbole ;
- Les commentaires sont à placer de préférence à droite, et les renvois de branchement à gauche.

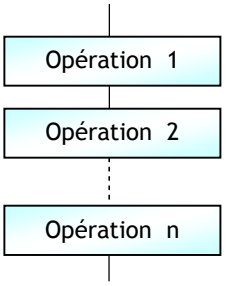
2. Structures algorithmiques élémentaires :

2.1. La structure séquentielle ou linéaire :

Algorithme

Opération 1
Opération 2
|
|
Opération n

Organigramme



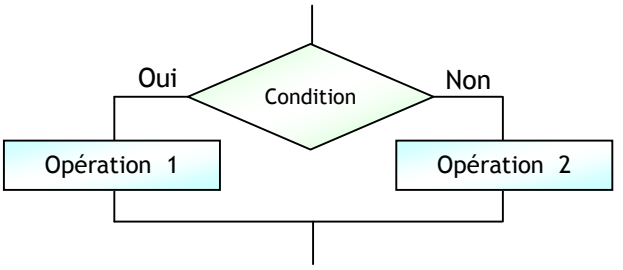
2.2. La structure conditionnelle ou alternative :

2.2.1. Cas d'une sélection simple :

Algorithme

- Si condition alors
  Opération 1
  Si non
  Opération 2
- Fin si

Organigramme



Remarque :

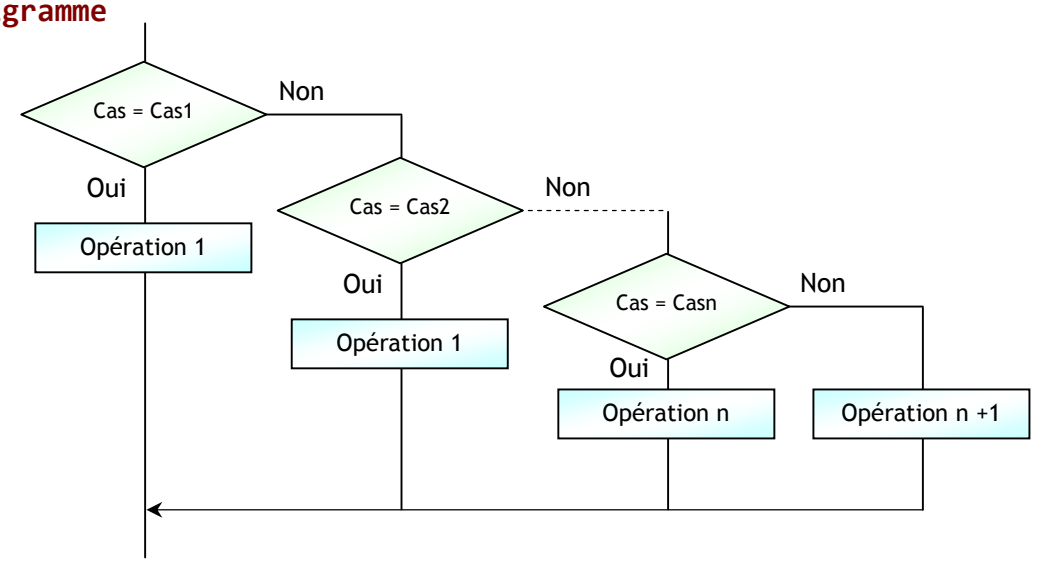
- La condition doit forcément s'énoncer au moyen d'une préposition logique ;
- L'une des deux opérations peut ne pas exister, ce qui fait disparaître le sinon.

2.2.2. Cas d'une sélection multiple :

Algorithme

- Selon cas
  Cas 1 : Opération 1
  Cas 2 : Opération 2
  |
  Cas n : Opération n
- Autrement
  Opération n+1
- Fin selon

Organigramme



Remarque :

- Le 'autrement' disparaît si l'opération n+1 n'existe pas.
- Si n est grand, l'organigramme devient très important et mal adapté.

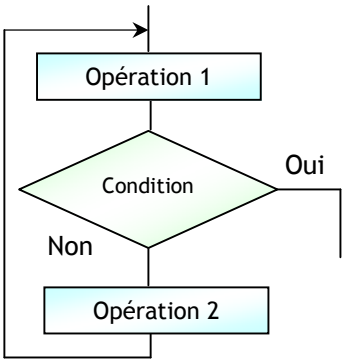
2.3. La structure itérative ou de répétition :

2.3.1. Cas d'une répétition non contrôlée :

Algorithme

- Itérer
  Opération 1
  Sortir si condition
  Opération 2
- Fin Itérer

Organigramme



Il s'agit d'une structure de boucle pour laquelle on ne peut sortir que si la condition est remplie.

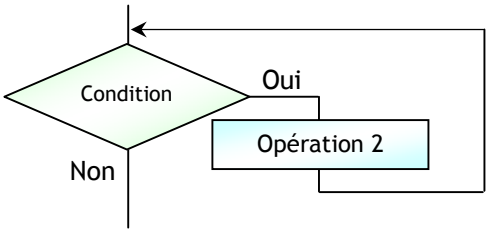
Remarque : deux cas particuliers sont très courants

- **Cas 1** : l'opération 1 n'existe pas, la structure de la boucle se décrit alors de la façon suivante :

Algorithme

- Tant que condition
  Faire opération 2
- Fin tant que

Organigramme

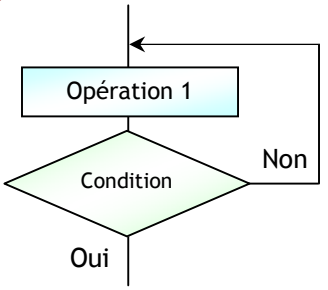


- **Cas 2** : l'opération 2 n'existe pas, la structure de la boucle se décrit alors de la façon suivante :

Algorithme

- Répéter
  Opération 1
- Jusqu'à ce que condition

Organigramme



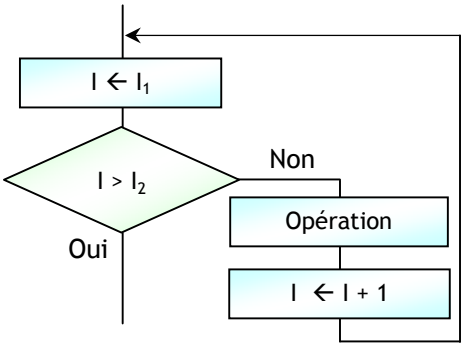
2.3.2. Cas d'une répétition contrôlée :

Il s'agit d'une structure de boucle évoluée qui se répète un nombre limité de fois, défini au préalable. Elle peut se décrire de la façon suivante :

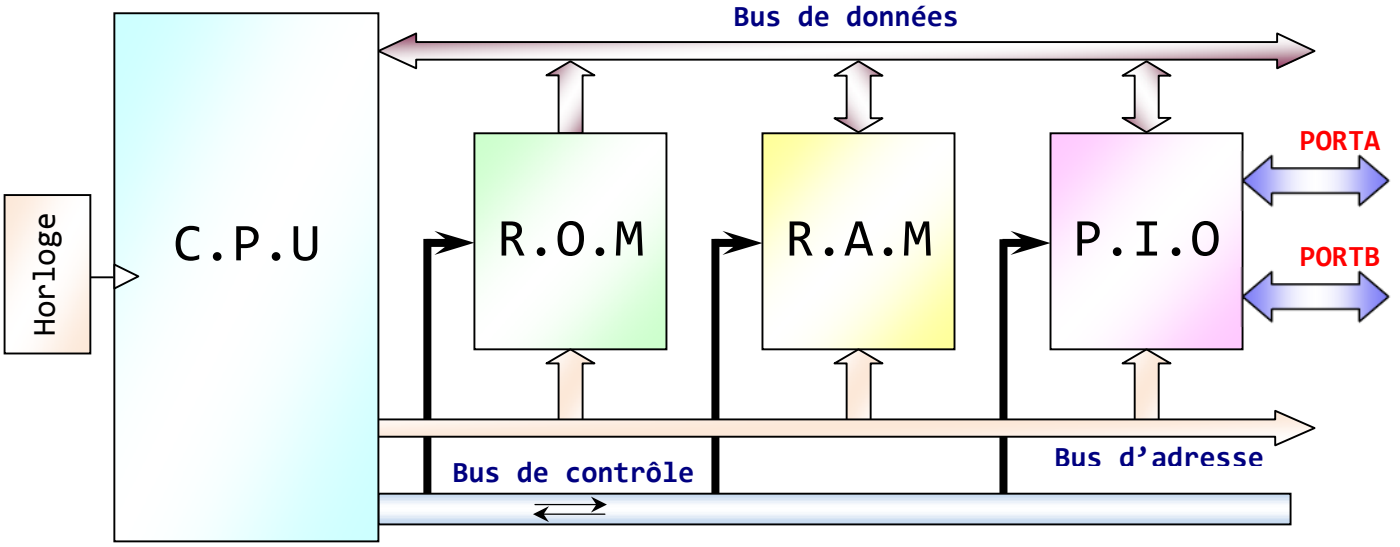
Algorithme

- Pour I de I_1 à I_2
  Faire opération
- Fin pour

Organigramme



1. Système minimum :



1.1. CPU :

L'unité centrale de traitement -Central Unit Processor - effectue le traitement des données et exécute les programmes.

Les opérations de bases réalisées par la CPU sont :

- Opérations logiques: AND, OR, XOR, Inversion… ;
- Opérations arithmétiques : Addition, soustraction, incrémentation, décrémentation … ;
- Déplacement et transfert de données.

La CPU génère 3 bus :

- Bus de données : bus bidirectionnel qui transporte les données ;
- Bus d'adresse : bus unidirectionnel qui transporte les adresses ;
- Bus de contrôle : bus qui transporte les différents signaux de contrôles nécessaire au fonctionnement du système.

Exemple de µP : Z80 - Z8000 de Zilog 6809 - 68000 de Motorola 8085 - 8086 de Intel.

1.2. RAM :

Mémoire à accès aléatoire -de l'anglais Random acces memory - permet de stocker les variables, les données et les programmes temporaires. C'est une mémoire volatile à lecture et à écriture.

Exemple de RAM : RAM 6164 de capacité 8 k octets.

1.3. ROM :

Mémoire non volatile à lecture seule -de l'anglais Read Only Memory- permet de stocker des programmes et des données constants.

Exemple de ROM : EPROM 2764 de 8 k octets.

1.4. Interface d'entrées sorties :

Permet de communiquer avec le monde extérieur (Imprimante, disque dur, Ecran, Clavier), à travers des lignes d'entrées sorties appelées PORT.

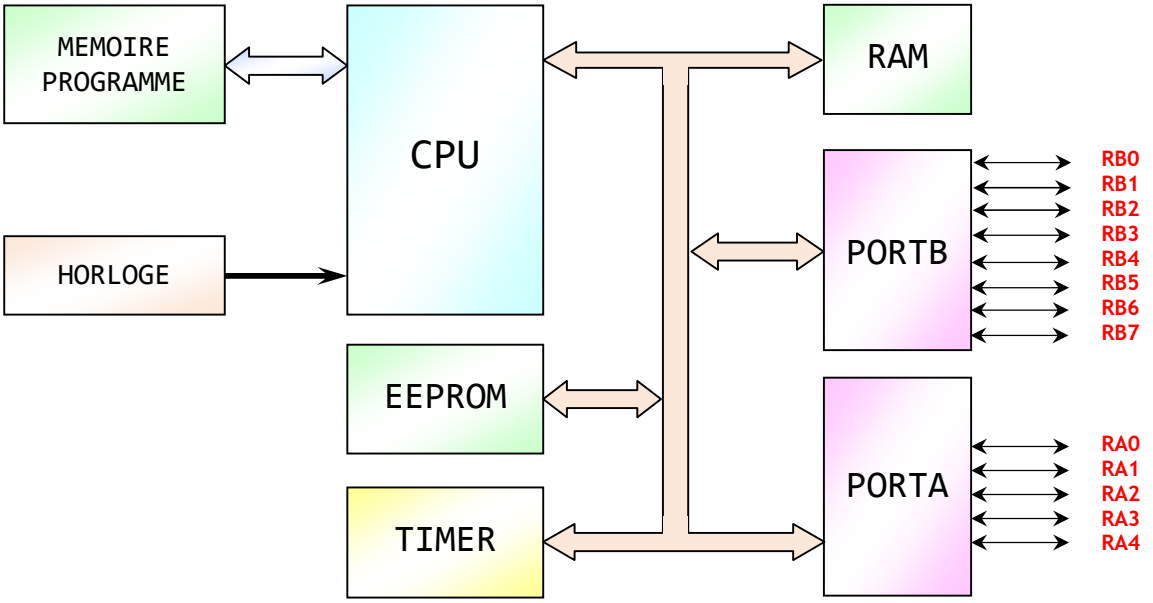
Exemple d'interface d'E/S : PIA 6821 de Motorola - PIO 8255 de Intel.

2. Les microcontrôleurs :

Les microcontrôleurs sont des composants programmables. Ils intègrent dans un seul boîtier l'environnement minimal d'un système à microprocesseur (l'UC, la RAM, l'EPROM et les interfaces). Ils sont présents dans la plupart des systèmes électroniques embarqués ou dédiés à une application unique. Il en existe de nombreux modèles différents, parmi les plus courants : le **8051** d'Intel, le **68HC11** de Motorola... et les **PIC** de Micro chip.

3. Description générale du PIC 16F84 :

Le synoptique simplifié est donné ci-contre :



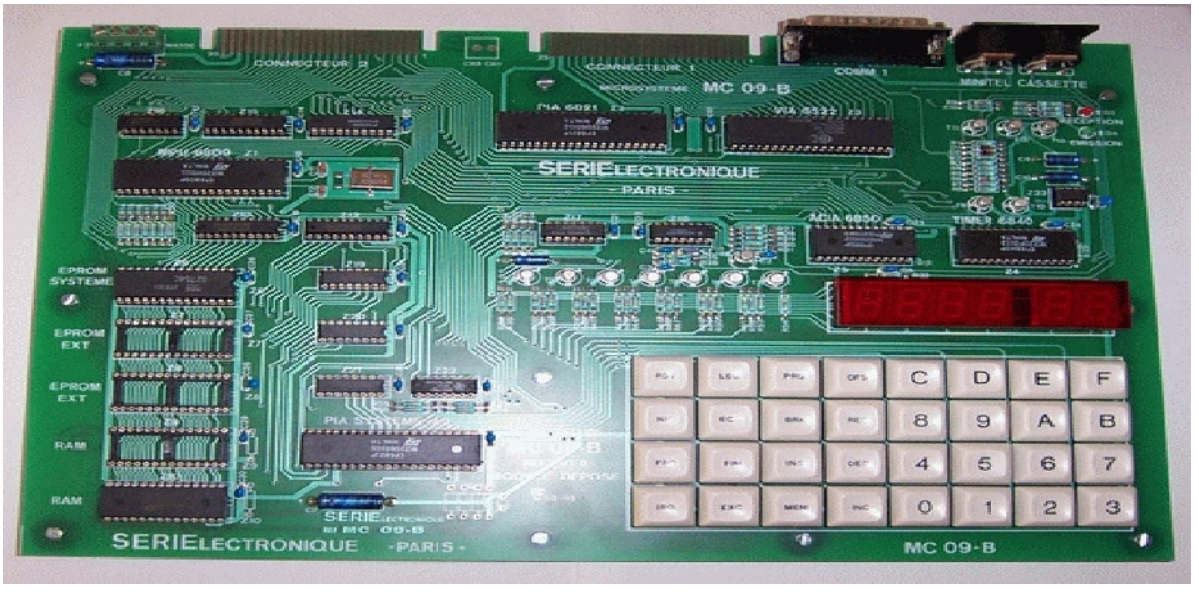
- **EEPROM de Programme :** Cette mémoire stock le programme.
- **RAM :** Cette mémoire stock les variables et les données temporaires.
- **EEPROM de données :** Cette mémoire stock les constantes et les données semi permanentes.
- **TIMER :** C'est compteur (de temps) modulo 256 – 8 bits –
- **PORTA et PORTB :** Ports pour communiquer avec l'extérieur.

Les ports sont **bidirectionnels**, ce qui signifie qu'ils peuvent être configurés et utilisés comme des entrées ou des sorties. Le microcontrôleur reçoit les informations sur un port d'entrée :

- Informations logiques issues de capteurs sur un ou plusieurs bits d'un port d'entrée ;
- Informations numériques codées sur 8 bits sur un port entier ;
- Informations analogiques variables dans le temps, si le PIC est doté d'un convertisseur analogique / numérique.

Le microcontrôleur traite ces données et les utilisent pour commander des circuits qui sont connectés sur un port de sortie.

4. Kit didactique d'un système programmé à base du microprocesseur 6809 :



1. Présentation :

Une cellule mémoire est un élément bistable capable d'emmagasiner puis de restituer un bit d'information ('0' ou '1'). EX : bascule, Disquette…

2. Mémoires électroniques :

2.1. Mémoire morte (ROM) :

C'est une mémoire à lecture seule, son contenu est non modifiable, elle reste inchangée même s'il y a coupure d'alimentation. On dit alors qu'elle est non volatile.

Les ROM sont utilisées pour stocker des informations figées telles que des programmes fixes dans des machines programmées ou les tables de conversion de données.

Le contenu est fixé à la construction ou par l'utilisateur et la disparition de l'alimentation électrique n'altère pas le contenu.

2.2. Mémoire vive (RAM) :

C'est une mémoire à accès aléatoire, on peut à chaque instant changer son contenu. Les RAM perdent leurs informations si on coupe l'alimentation, on dit qu'elles sont volatiles.

Dès qu'un système doit conserver temporairement des informations, la RAM trouve sa place. En informatique, elles sont largement mises en œuvre en quantités importantes (plus de 16 Mo en micro informatique et plusieurs centaines de méga octets en mini informatique).

2.3. Les mémoires programmables et effaçables par l'utilisateur :

Les mémoires programmables sont intermédiaires entre les RAM et les ROM. Leur contenu peut être défini par l'utilisateur et subsister sans alimentation électrique.
On en rencontre de différentes familles :

- Les **PROM** (Programmable ROM) : sont composées de fusibles que l'on peut détruire une seule fois ;
- Les **EPROM** (Erasable PROM) : ce sont des mémoires effaçables par ultraviolet et programmables électriquement ;
- Les **EEPROM** (Electrical Erasable PROM) : ce sont des mémoires effaçables et programmables électriquement.

3. Organisation interne :

3.1. Capacité :

C'est la quantité d'information qui peut être stockée dans la mémoire. Elle s'exprime en bits ou en mots de n bits. Par exemple : 64Ko, 4Mo, 2Go (o : octet)

- 1 K = 2^{10} = 1024 ;
- 1 M = 2^{20} = 1.048.576 ;
- 1 G = 2^{30} = 1.073.741.824.

3.2. Longueur de mot :

C'est la façon avec laquelle les bits sont organisés ou rangés, en général par mot de 8 bits ou de 4 bits.

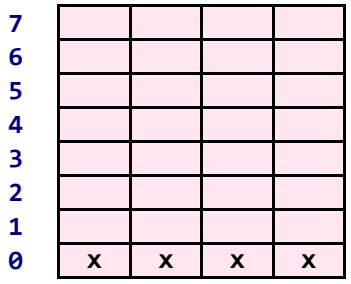
- Un mot de 4 bits : 1 Quartet
- Un mot de 8 bits : 1 Octet

3.3. Adresse :

Pour identifier les mots on donne à chacun une adresse, on dit alors case mémoire d'adresse 40, case mémoire d'adresse FF …… etc.

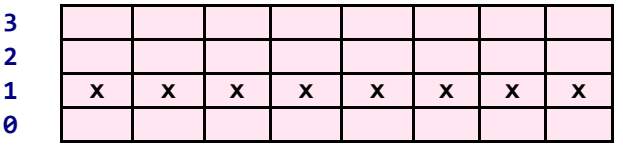
3.4. Le temps d'accès :

C'est le temps qui s'écoule entre une demande d'information et le moment où elle est effectivement disponible.



8 mots de 4 bits ou 8 quartets

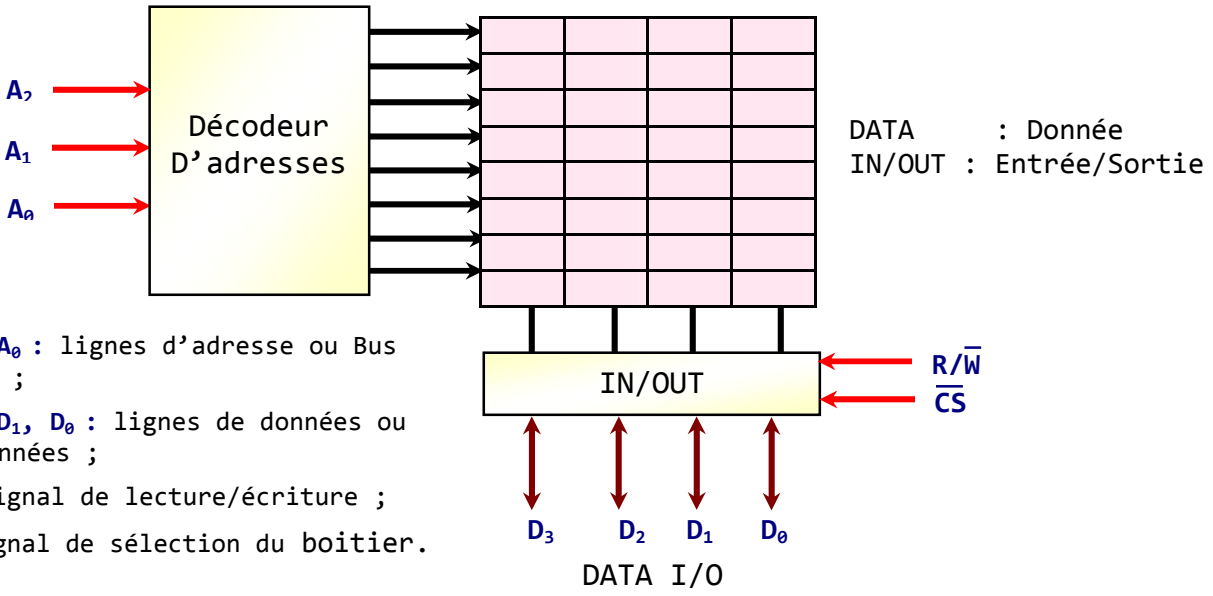
8 mots de 4 bits = 32 bits



4 mots de 8 bits ou 4 octets

4 mots de 8 bits = 32 bits

3.5. Décodage d'adresse :



- A_2, A_1, A_0 : lignes d'adresse ou Bus d'adresse ;
- D_3, D_2, D_1, D_0 : lignes de données ou Bus de données ;
- $R/\overline{W}$: signal de lecture/écriture ;
- $\overline{CS}$: signal de sélection du boitier.

Capacité = 2 $^{\text{nombre de lignes d'adresse}}$ x nombre de lignes de données

Pour l'exemple ci-dessus : Capacité = 2^3 x 4 bits = 8 x 4 bits = 32 bits

Capacité = 8 x 4 bits = 8 q

On peut donc utiliser une mémoire soit en :

- **lecture :**
  - Appliquer le mot adresse sur le bus d'**adresse** ;
  - Sélectionner le boîtier mémoire en appliquant un niveau logique **bas** sur la ligne $\overline{CS}$;
  - Sélectionner le mode lecture en appliquant un niveau logique **haut** sur la ligne $R/\overline{W}$;
- **écriture :**
  - Appliquer le mot d'adresse sur le bus d'**adresse** ;
  - Appliquer le mot de donnée sur le bus de **données** ;
  - Sélectionner le boîtier mémoire en appliquant un niveau logique **bas** sur la ligne $\overline{CS}$;
  - Sélectionner le mode écriture en appliquant un niveau logique **bas** sur la ligne $R/\overline{W}$;

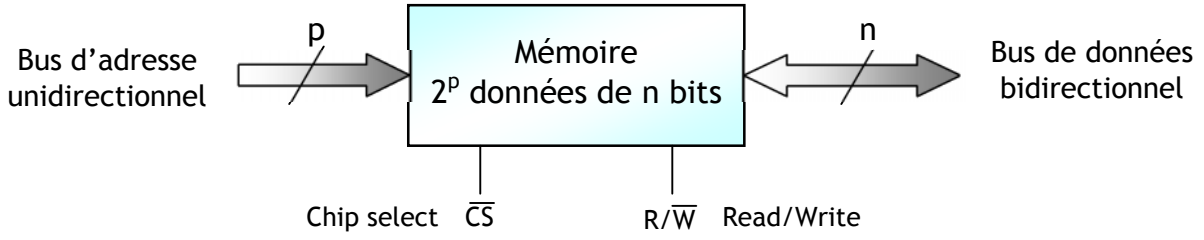


Schéma fonctionnel d'une mémoire

1. Introduction :

1.1. Définition :

Une interface est un circuit d'adaptation entre le µp et le périphérique. Elle établit la compatibilité entre les entées sorties du µp et celles des périphériques, à plusieurs niveaux :

- Au niveau des types de transmission série ou parallèle ;
- Au niveau de la vitesse de transmission, car les périphériques sont très lents par rapport au µp.

1.2. Périphérique :

On appelle périphérique tout organe ou dispositif externe au système minimum qui échange des informations avec le µp.

Il exit deux types de périphérique :

- **Organe d'entrée :** Il envoie les informations vers le µp (le clavier) ;
- **Organe de sorties :** Il reçoit les informations du µp (l'écran).

2. Périphériques d'entrées sorties parallèles :

2.1. Définition :

Une liaison // consiste à envoyer l'information de mot de n bits, véhiculés de façon simultané sur n fils.

2.2. Structure et fonctionnement :

Une interface parallèle programmable comporte :

- Des registres tampon de sorties : Les transferts d'entrées sorties se ramènent à une lecture ou écriture de ces registres appelés ports ;
- Registre de commandes : qui permet de configurer le port en entrée ou en sortie selon l'utilisation.

2.3. Programmation de l'interface :

Une interface s'adaptera à un périphérique donné par une programmation de l'interface.

Cette programmation consiste à configurer les ports de l'interface en entrée ou en sortie selon l'utilisation.

3. Périphériques d'entrées sorties séries :

3.1. Définition :

Une liaison série consiste à transmettre les données de façon successive (bit par bit) sur un seul et unique fil.

Les avantages de la liaison série sont énormes :

- utilisation d'un nombre réduit de fil ;
- Fiabilité de l'information due au contrôle de mot du l'émetteur jusqu'au récepteur.

Il y a 2 types de liaison série : **synchrone** et **asynchrone**.

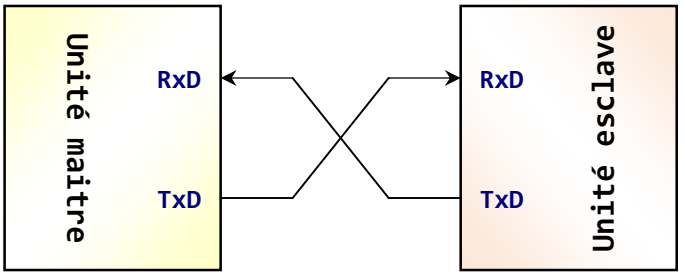
3.2. Liaison série asynchrone :

Ce dispositif ne possède pas de signal d'horloge de synchronisation. Les unités en liaison possèdent chacune une horloge interne cadencée à la même fréquence.

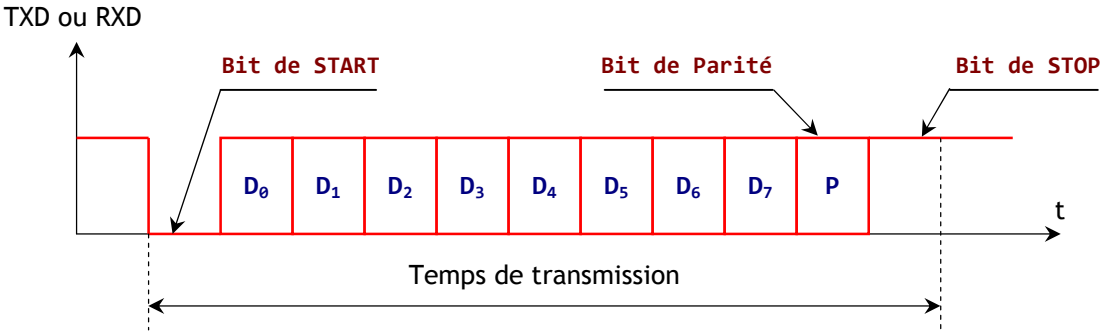
Lorsqu'une unité veut émettre un mot binaire, elle génère un front descendant sur sa ligne émettrice. A la fin de l'émission de ce mot, la ligne repasse au niveau haut.

La donnée à transmettre peut contenir un bit supplémentaire appelé "parité" et servant à la correction d'erreurs.

Chronogrammes :



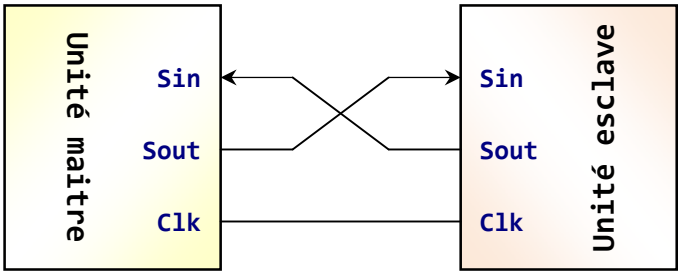
- **RxD :** ligne de réception des données.
- **TxD :** ligne de transmission des données.



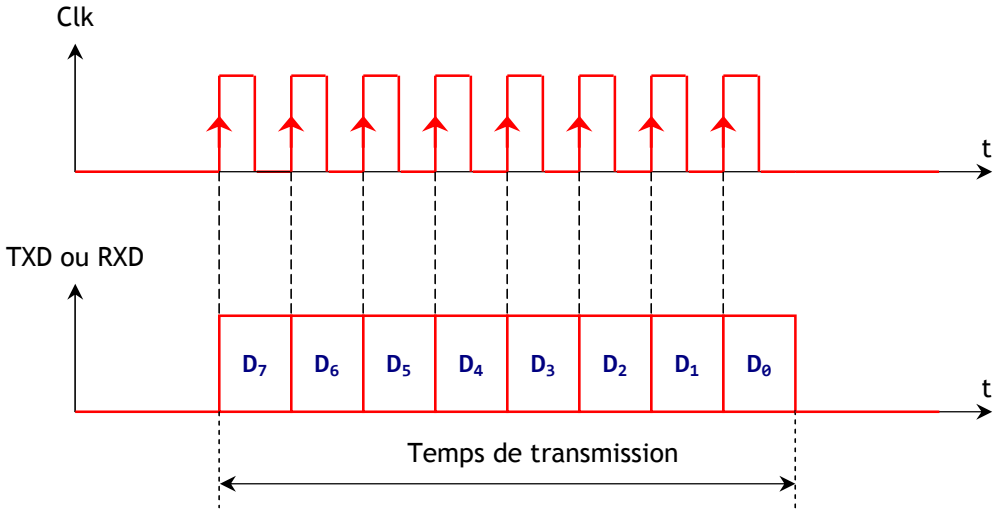
3.3. Liaison série synchrone :

Dans ce dispositif la transmission est synchronisée par un signal d'horloge émis par l'unité maître.

Chronogrammes :



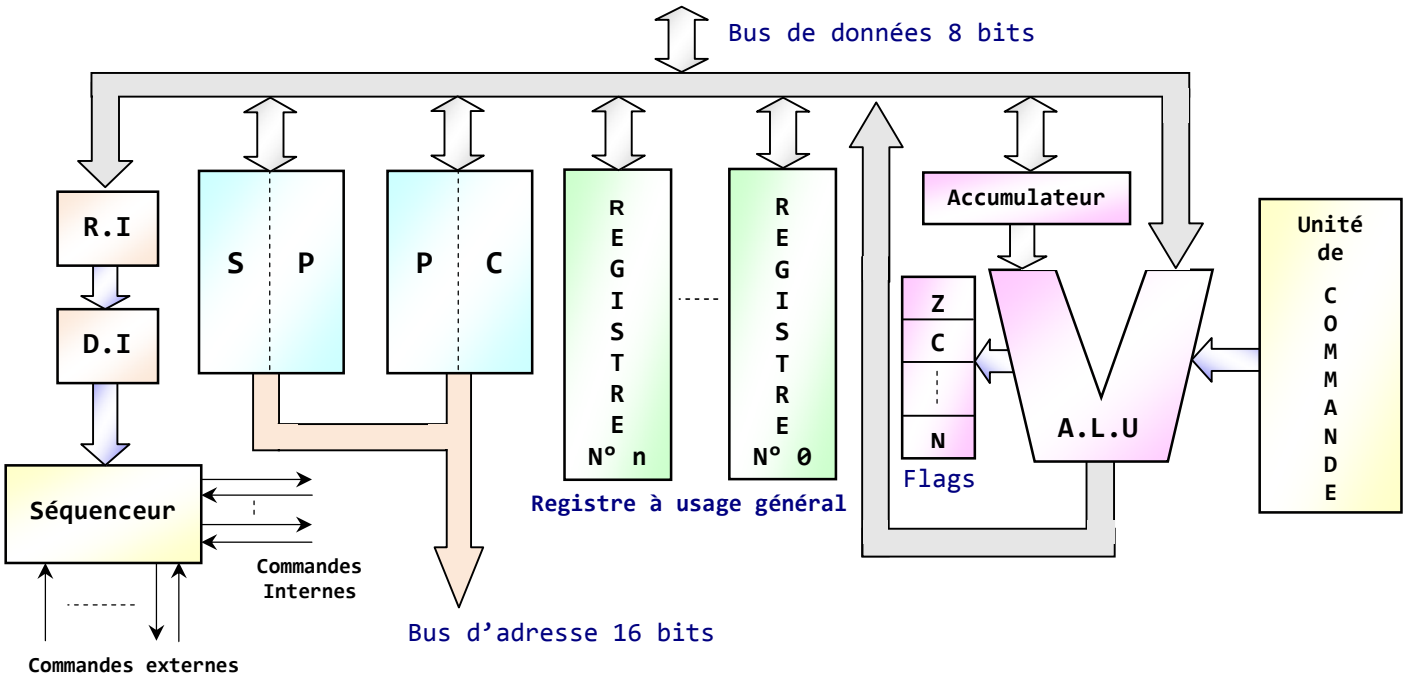
- **Sin :** ligne de réception des données.
- **Sout :** ligne de transmission des données.
- **Clk :** horloge de synchronisation



4. Exemples:

- Liaison // :
  Port **Centronics** (Pour imprimante parallèle).
- Liaison série :
  Port **USB** (Universal Serial Bus);
  Liaison **RS232** (liaison série point à point) ;
  Liaison **RS485** (liaison série multipoints).

1. Structure générale d'un microprocesseur 8 bits :



2. Unité arithmétique et logique :

L'A.L.U accomplie les opérations arithmétique et logiques :

- Addition, soustraction, négation …
- Et, ou inclusif, ou exclusif, inversion …

Elle est dotée à l'une de ces entrées d'un registre spécial appelé '**Accumulateur**'.

3. Registre d'état :

C'est le registre des indicateurs d'état (**Flags**) ; leur rôle est de mémoriser les situations exceptionnelles qui peuvent survenir pendant le fonctionnement du µP.

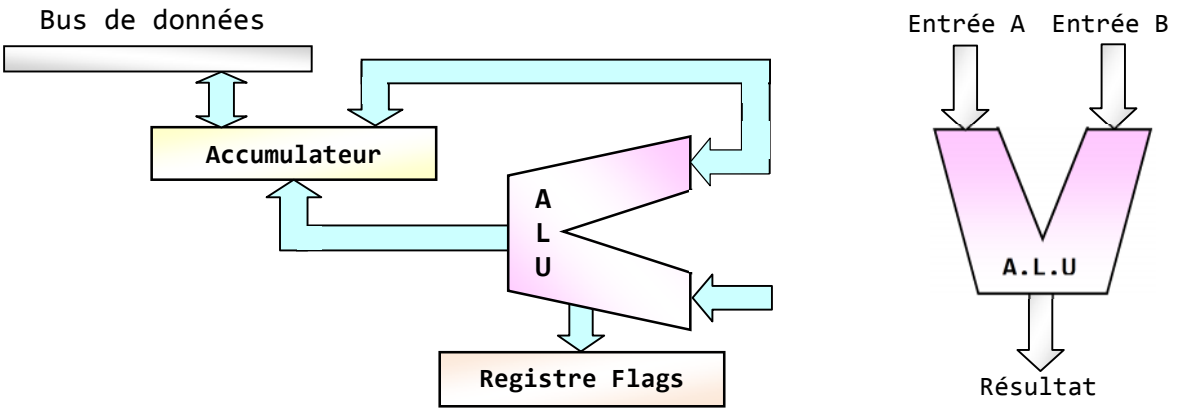
Exemple d'indicateur :

- **C** : Retenue (si C =1 il y a retenue, si C = 0 pas de retenue) ;
- **DC** : Demi retenue (si DC =1 il y a demi retenue, si c = 0 pas de demi retenue) ;
- **Z** : Zéro (si z =1 résultat = 0, si z = 0 résultat ≠ 0).

Le contenu du registre d'états est testé en général par des instructions spécialisées tel que les instructions conditionnelles.

4. Accumulateur :

C'est le registre principal d'un microprocesseur 8bits, il présente la particularité de pouvoir être désigné, dans la même instruction, à la fois comme opérande et comme opérande résultat.



5. Registre à usage universel (général) :

Leur rôle est de permettre à l'A.L.U de manipuler les données à grande vitesse. Ils sont généralement numérotés de 0 jusqu'à n, leur rôle n'est pas défini à l'avance. C'est pour quoi on les appelle registres à usage universel. Ils peuvent contenir n'importe quelle valeur utilisée par le programme.

6. Le compteur de programme :

Le compteur de programme ou Program Counter est présent dans tous les processeurs, il contient toujours l'adresse de la prochaine instruction à exécuter. Une fois le code de l'instruction est récupéré, le PC est incrémenté automatiquement pour pointer l'instruction suivante.

7. Le pointeur de pile :

Le pointeur de pile ou Stack Pointer contient toujours une adresse particulière ou la valeur du PC est stockée temporairement. Le SP est indispensable au fonctionnement des sous programmes et des interruptions.

8. Registre d'instructions :

Le registre d'instructions est un registre 8bits, dont le rôle est de stocker le code opération de l'instruction en cours d'exécution, ce registre n'est pas programmable.

9. Décodeur d'instructions :

Son rôle est d'identifier le code opération de l'instruction présente sur le RI, et générer la séquence correcte des signaux interne et externe qui permettra l'exécution de l'instruction.

10. Les cycles de fonctionnement d'un microprocesseur :

Quelque soit le processeur, le cycle de fonctionnement se fait en 3 cycles :

- Récupération de l'instruction ;
- Décodage de l'instruction ;
- Exécution de l'instruction.

1. Introduction :

1.1. Description :

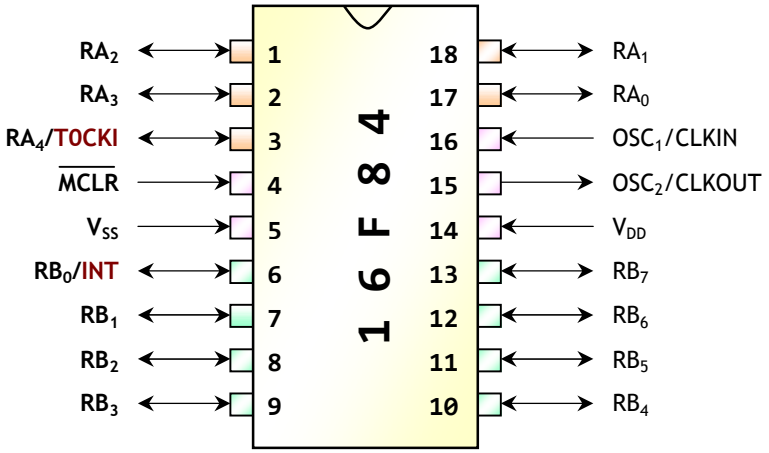
Le PIC (**P**rogrammable **I**nterface **C**ontroler) **16F84** est un microcontrôleur 8 bits de faible coût, produit par la société **MicroChip**. C'est un composant électronique qui regroupe dans un même boîtier tous les éléments vitaux d'un système programmé : CPU, RAM, ROM, Interfaces d'entrées/sorties.
De plus, avec son jeu d'instructions réduit, il est très agréable à utiliser lorsqu'on désire s'initier à l'étude d'un microcontrôleur.

1.2. Caractéristiques :

- 1K de mémoire programme ;
- 68 octets de RAM ;
- 64 octets D'EEPROM ;
- 13 entrée/sortie réparties en 2 ports ;
- PORTA et PORTB ;
- 4 sources d'interruption ;
- 1 Timer/Compteur ;
- 1 Chien de garde ;
- MODE SLEEP (faible consommation) ;
- 4 Sources d'oscillateur sélectionnable ;
- Protection du code ;
- 35 instructions seulement.

2. Structure externe du PIC 16F84 :

2.1. Brochage :



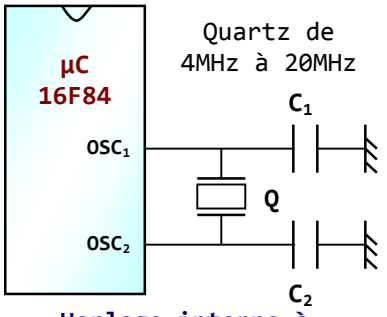
- **VDD, VSS** : Broches d'alimentation (3 à 5.5 v) ;
- **OSC1, OSC2** : Signaux d'horloge ;
- **CLKINT** : horloge externe ;
- **$\overline{MCLR}$** : Reset ou Master Clear;
- **T0CKI** : entrée d'horloge externe du Timer 0 ;
- **INT** : entrée d'interruption externe ;
- **RA_0...RA_4** : 5 E/S du PORTA ;
- **RB_0...RB_7** : 8 E/S du PORTB.

2.2. Les signaux d'horloge :

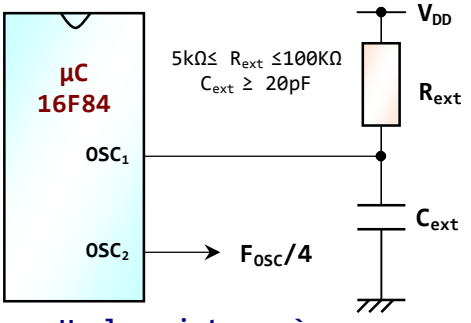
Le fonctionnement du µC 16F84 nécessite une horloge qui rythme l'exécution des instructions du programme.

On distingue trois modes d'horloge :

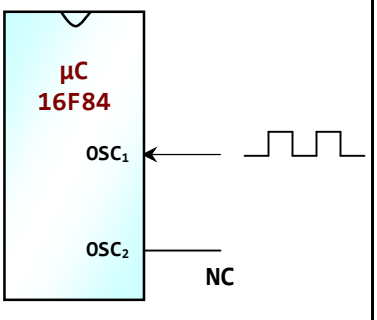
- **Horloge interne à quartz** : Avec l'oscillateur à quartz, on peut avoir des fréquences allant de 4 MHz jusqu'à 20 MHz selon le type du µC.
- **Horloge interne à circuit RC** : Avec un oscillateur à circuit RC, la fréquence de l'oscillation dépend de la tension V_{DD} et des éléments R_{ext} et C_{ext}.
- **Horloge externe** : Application d'un signal horloge externe.



Horloge interne à quartz



Horloge interne à circuit RC

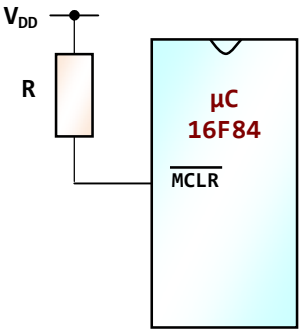


Horloge externe

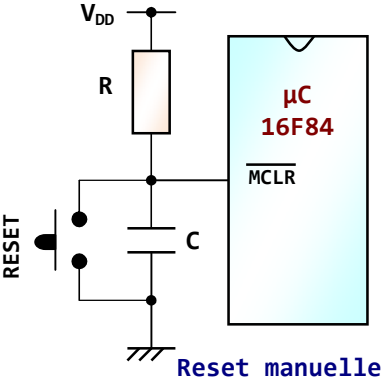
2.3. Source de Reset :

Le Reset du µC 16F84 peut avoir plusieurs causes :

- Une mise sous tension **POR** (**P**ower **O**n **R**eset) ;
- Une mise à 0 de la broche **MCLR** (Reset manuelle) ;
- Un débordement du timer du chien de garde **WDT**.



Power On Reset



Reset manuelle

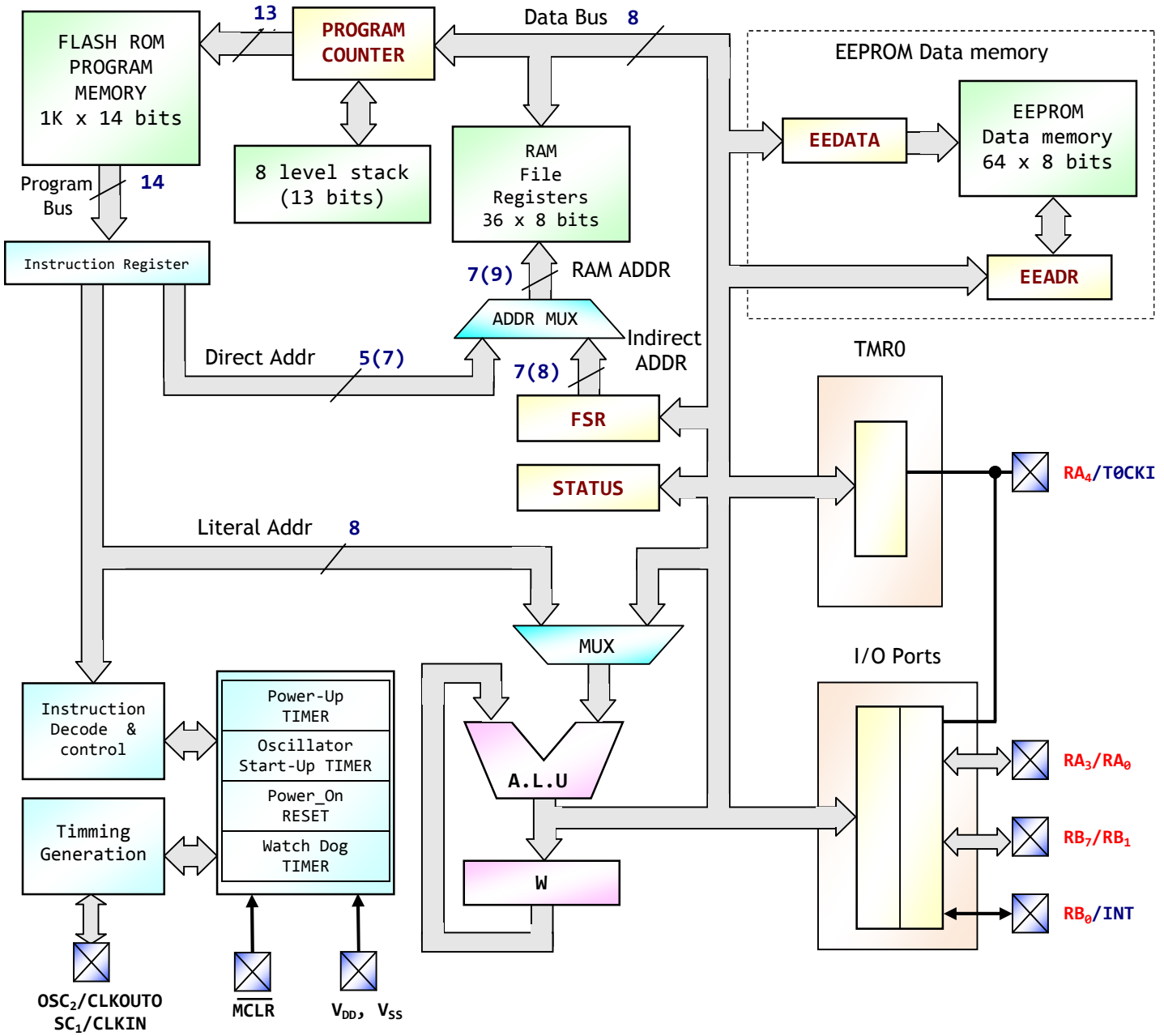
3. Structure interne du PIC 16F84 :

3.1. Horloge système :

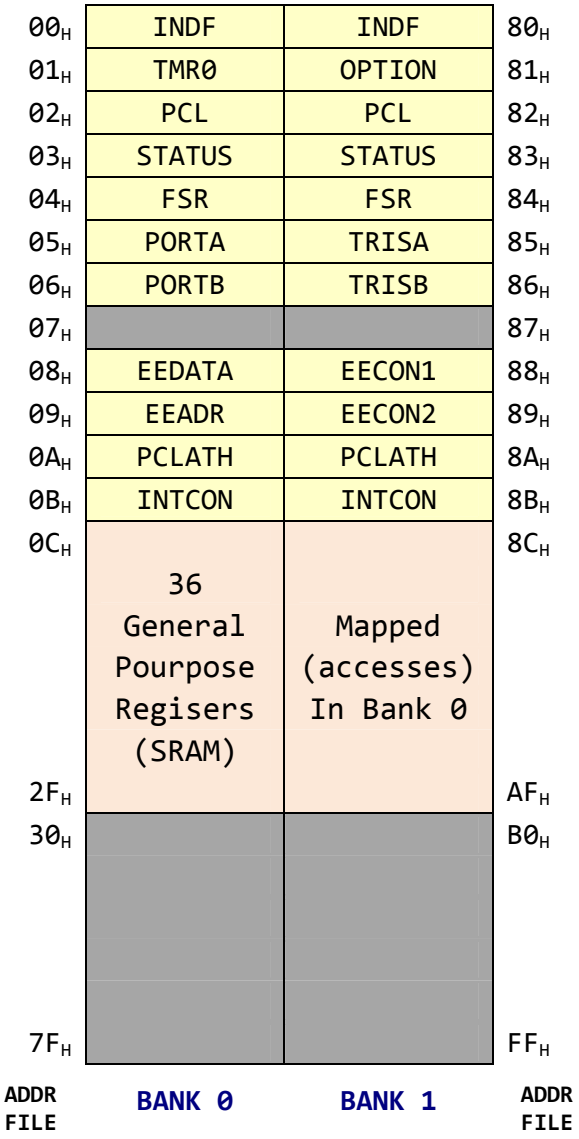
L'horloge système ou horloge instruction, est la base du temps interne qui cadence le fonctionnement du µC. Elle s'obtient en divisant la fréquence de l'oscillateur par **4**.

Exemple : Avec un quartz de 4 MHz, on obtient une horloge instruction de 1 MHz, soit le temps pour exécuter une instruction de 1 µs.

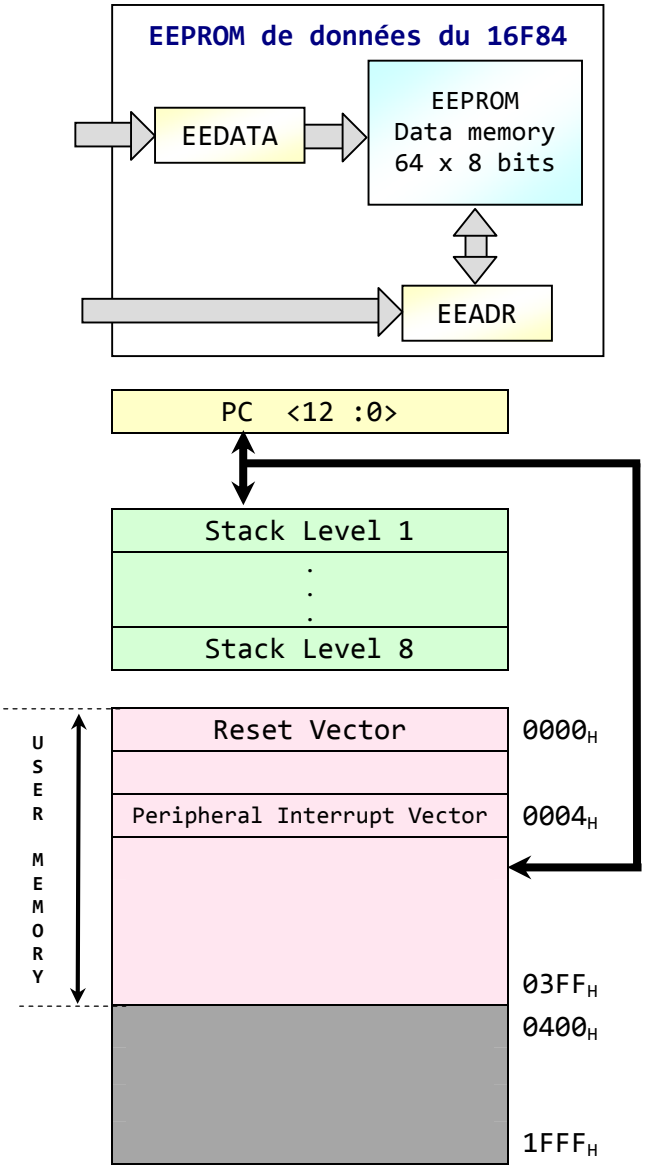
3.2. Description :



3.3. Les Mémoires :



RAM du 16F84



EEPROM programme du 16F84

3.3.1 EEPROM de Programme :

Cette mémoire de **1K** stock le programme. L'emplacement de celui-ci peut être à n'importe quel endroit de la mémoire. Suite à un RESET ou lorsqu'on l'alimente, le PIC16F84 commence toujours à l'adresse **0000_H** (Vecteur RESET).

De plus, lorsqu'il y a une interruption, et si celle-ci est validée, le microcontrôleur va à l'adresse **0004_H** (Vecteur d'interruption).

3.3.2 RAM de données :

Cette mémoire de **68 octets** est divisée en deux parties. Une partie appelée **SFR**, l'autre le **GPR**, qui sont encore divisée en deux pages (**Bank 0** et **Bank 1**). Toutes les données de la mémoire sont appelées registres y compris les données utilisateurs.

- **Le SFR (Special Function Registers):**

Le SFR est l'ensemble des registres qui permet de configurer tous les modules internes du PIC16F84 (les PORTS, le TIMER, la gestion des interruptions, etc....).

- **Le GPR (General Purpose Register):**

Le GPR est l'ensemble des registres à usage général, utilisé par le programmeur pour stocker les variables et les données.

3.3.3 EEPROM de données :

Une EEPROM de **64 octets** est disponible pour y stocker les données semi permanentes. Pour accéder à cette mémoire, on utilise les registres **EEADR**, **EEDATA**. 2 registres de contrôle sont associés à cette mémoire **$EECON_1$** et **$EECON_2$**.

3.3.4 La pile :

Un groupe de **8** registres de **13** bits, son rôle est de sauvegarder temporairement le contenu du **PC** lors de l'appel d'un sous programme ou du service d'une interruption. La pile n'est pas dans le fichier des registres, donc elle n'est pas manipulable par le programmeur.

3.4. Registre de configuration :

Pendant la phase de la programmation du µC, on programme aussi un registre de configuration logé dans la mémoire EEPROM. Ce registre est un mot de **14** bits qui permet de :

- Choisir le type de l'oscillateur pour l'horloge ;
- Valider ou non le timer du watchdog WDT ;
- Autoriser ou non une temporisation à la mise sous tension ;
- Interdire ou non la lecture des mémoires de programme et de données.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
CP	CP	CP	CP	CP	CP	DP	CP	CP	CP	PWRTE	WTDE	$FOSC_1$	$FOSC_0$

- **Bits $FOSC_0$ et $FOSC_1$: Sélection du type d'oscillateur pour l'horloge**
  - $FOSC_1FOSC_0$ = 11 : Oscillateur à circuit RC jusqu'à 4 MHz ;
  - $FOSC_1FOSC_0$ = 10 : Oscillateur HS, quartz haute fréquence, jusqu'à 20 MHz ;
  - $FOSC_1FOSC_0$ = 01 : Oscillateur XT, quartz standard jusqu'à 4 MHz ;
  - $FOSC_1FOSC_0$ = 00 : Oscillateur LP, quartz basse fréquence, jusqu'à 200 KHz.
- **Bit WDTE : Validation du timer du watchdog WDT**
  - WDTE = 1 : Timer du watchdog validé ;
  - WDTE = 0 : Timer du watchdog validé inhibé.
- **Bit PWRTE : Validation d'une temporisation à la mise sous tension**

Le µC possède un timer permettant de retarder de 72 ms le lancement du programme après la mise sous tension. Ce délai maintient le µC à l'arrêt et permet ainsi à la tension d'alimentation de bien se stabiliser.

  - PWRTE = 1 : le µC démarre tout de suite ;
  - PWRTE = 0 : le µC attend 72 ms.
- **Bits CP : Protection en lecture de la mémoire programme**
  - CP = 1 : pas de protection du code programme ;
  - CP = 0 : protection du code programme activée.
- **Bits DP : Protection en lecture de la mémoire des données**
  - DP = 1 : pas de protection des données mémoire ;
  - DP = 0 : protection des données mémoire activée.

3.5. ALU et le registre W :

Le registre **W**, qui n'a pas d'adresse, est un registre de travail de **8** bits.

L'ALU est une unité arithmétique et logique de **8** bits qui réalise les opérations entre **W** et n'importe quel autre registre **f** ou constante **k**. Le résultat de l'opération peut être placé soit dans **W** soit dans **f**.

L'ALU est associée au registre d'état **STATUS** par les bits **Z**, **C** et **DC** :

7	6	5	4	3	2	1	0
IRP	RP_1	RP_0	/TO	/PD	Z	DC	C

3.6. Registres spéciaux :

Ces registres spéciaux font partie du **SFR** (Special Function Registers) et configurent le µC. Certains registres initialisent les périphériques alors que d'autres sont utilisés par le CPU.

- Les registres utilisés par le CPU :
  INDF, **FSR**, **STATUS**, **INTCON** et **PC** : **PCLATH-PCL**.
- Les registres utilisés par le PORTA :
  TRISA et **PORTA**.
- Les registres utilisés par le PORTB :
  TRISB et **PORTB**.
- Les registres utilisés par le TIMER :
  TMR0 et **OPTION**.
- Les registres utilisés par l'EEPROM :
  EEDATA, **EEADR**, **EECON$_1$** et **EECON$_2$**.

4. Les ports A et B :

4.1. Particularités du port A :

- Le port A désigné par **PORTA** est un port bidirectionnel de **5 bits** (RA_0 à RA_4) ;
- La configuration de direction du port A est déterminée avec le registre **TRISA** ;
- Les broches **RA_0** à **RA_3** sont des entrées/sorties compatibles **TTL** ;
- La broche **RA_4** est multiplexée avec l'entrée d'horloge du registre **TMRO** ;
- La broche **RA_4** est une sortie à **drain ouvert**, il ne faut pas oublier de mettre une résistance externe vers **V_{DD}**.
- Chaque broche du port A configurée en sortie peut fournir un courant de **20 mA** au maximum, mais tout le port A configuré en sortie ne peut pas débiter un courant total supérieur à **50 mA**.
- Chaque broche du port A configurée en entrée peut accepter un courant de **25 mA** au maximum, mais tout le port A configuré en entrée ne peut pas accepter un courant total supérieur à **80 mA**.

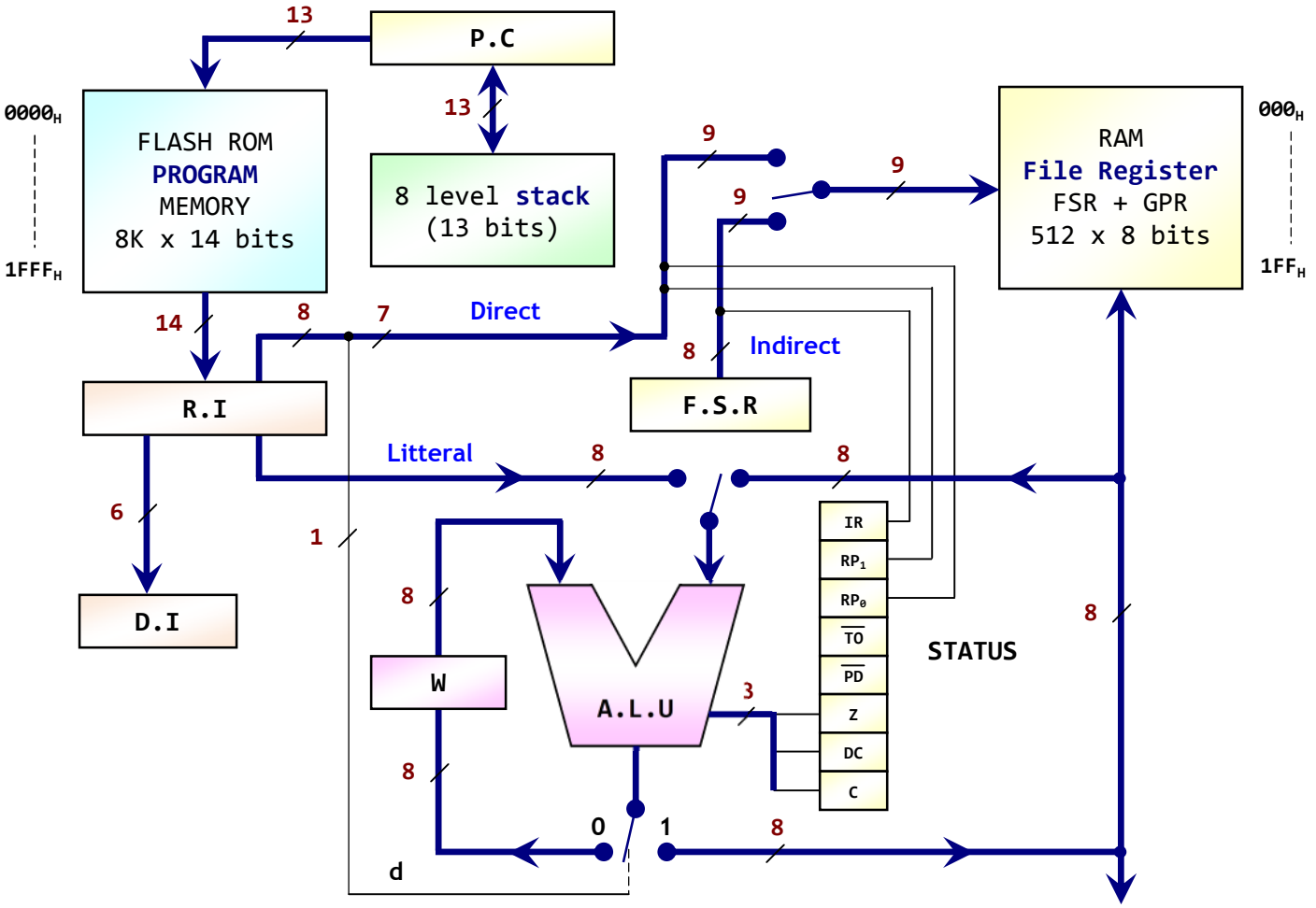
4.2. Particularités du port B :

- Le port B désigné par **PORTB** est un port bidirectionnel de **8 bits** (RB_0 à RB_7) ;
- La configuration de direction du port B est déterminée avec le registre **TRISB** ;
- Toutes les broches sont des entrées/sorties compatibles **TTL** ;
- Les entrées du port B bénéficient d'un "**tirage au plus**" interne - connectées à des résistances de rappel à **V_{DD}** - ;
- La broche **RB_0** est multiplexée avec l'interruption **INT** ;
- Les broches **RB_4** à **RB_7**, à condition qu'elles soient configurées en ENTREE, peuvent générer une **interruption** lorsqu'elles changent d'états ;
- Chaque broche du port B configurée en sortie peut fournir un courant de **20 mA** au maximum, mais tout le port B configuré en sortie ne peut pas débiter un courant total supérieur à **100 mA** ;
- Chaque broche du port B configurée en entrée peut accepter un courant de **25 mA** au maximum, mais tout le port B configuré en entrée ne peut pas accepter un courant total supérieur à **150 mA**.

1. Introduction

Les µC PIC 16Fxxx possèdent un jeu de **35** instructions. Chaque instruction est codée sur un mot de **14** bits qui contient le code opération ainsi que l'opérande. A part les instructions de saut, toutes les instructions sont exécutées en un **cycle** d'horloge. Sachant que l'horloge fournie au µC est prédivisée par **4**, si on utilise par exemple un quartz de 4 MHz, on obtient donc 1000000 cycles/seconde, cela nous donne une puissance de l'ordre de 1 **MIPS** (1 **M**illion d'**I**nstructions **P**ar **S**econde).

2. Structure simplifiée du PIC 16F877 :



3. Le registre STATUS :

7	6	5	4	3	2	1	0
IRP	RP_1	RP_0	/TO	/PD	Z	DC	C

C'est le registre d'état, il contient :

- 5 bits, témoins (drapeaux) caractérisant le résultat de l'opération réalisée par la CPU (à lecture seule) :
  - **/TO** : (Time Out) débordement du timer WDT ;
  - **/PD** : (Power Down) caractérise l'activité du chien de garde WDT ;

RESET	Etat	/TO	/PD	PC
Mise sous tension		1	1	0000
0 sur broche MCLR	Normal	NA	NA	0000
	Sleep	NA	0	0000
Chien de garde	Normal	0	1	0000
	Sleep	0	0	PC + 1

- **Z** : (zéro) résultat nul pour une opération arithmétique et logique ;
- **DC** : (digit carry) retenue sur un quartet (4 bits) ;
- **C** : (carry) retenue sur un octet (8 bits).

- 2 bits - **RP_1** et **RP_0** - de sélection de banc (Ecriture/lecture) ;
- 1 bit - **IRP** - de sélection de page (Ecriture/lecture).

4. Les modes d'adressages :

- **Mode littéral :** Dans ce mode d'adressage, la donnée est intégrée avec le code de l'opération.

```
MOVLW   0x55 ; charger la valeur 0x55 dans le registre W
```

- **Mode direct :** Les 7 premiers bits de l'adresse du fichier, sont intégrés avec le code de l'opération. Les 2 bits RP_0, RP_1 du registre STATUS sont utilisés pour compléter l'adresse sur 9 bits.

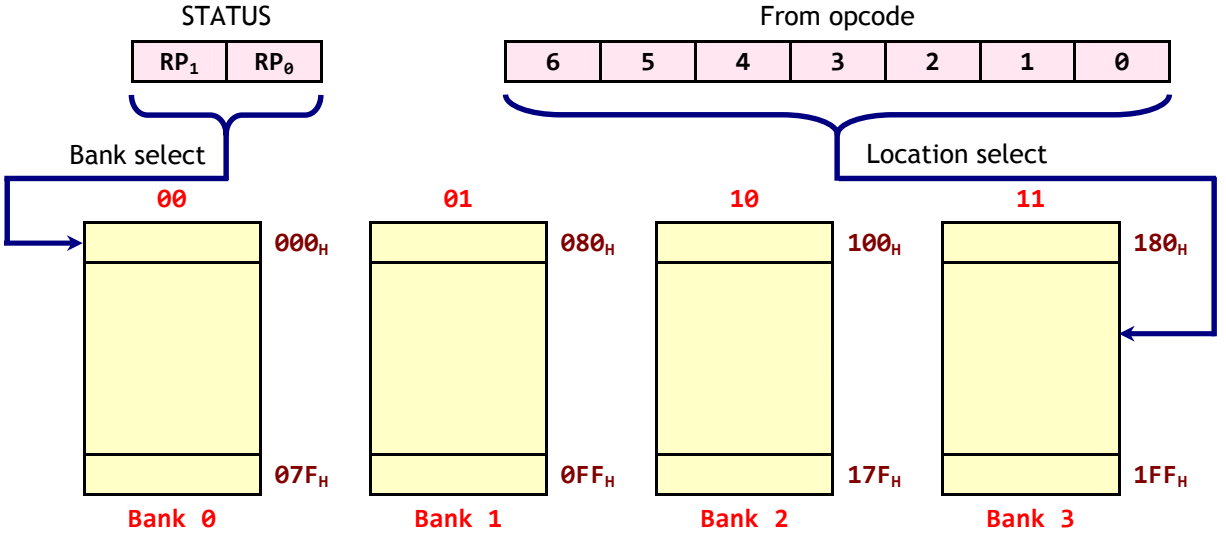
```
MOVF    0x55, W ; charger le contenu de l'adresse 0x55 dans W
```

- **Mode indirect :** L'adresse du fichier est formée par le contenu du FSR et le bit IRP du registre STATUS.

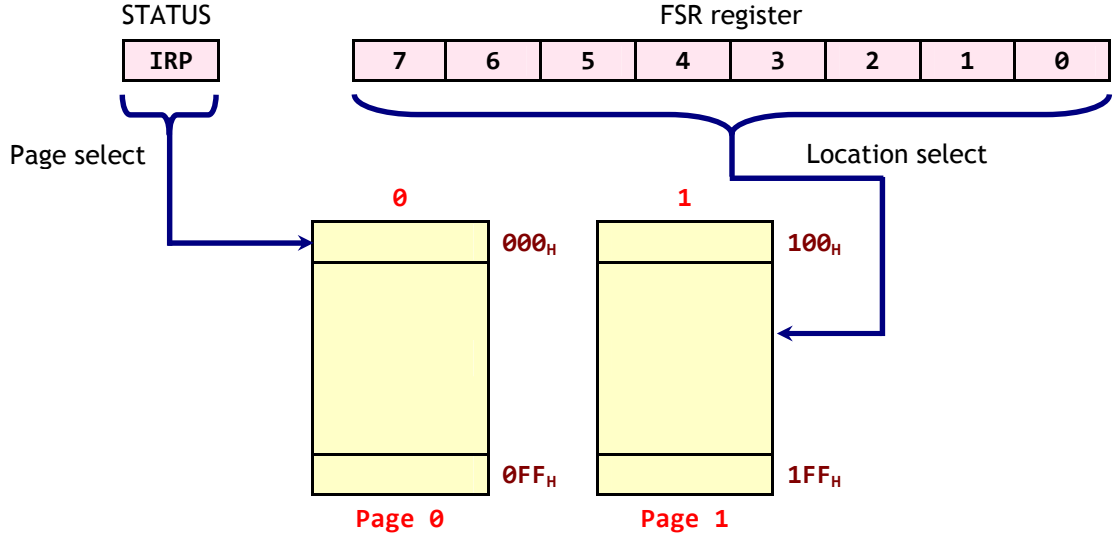
```
MOVF    INDF, W ; charger le contenu de la case mémoire pointée par le FSR.
```

5. Plan mémoire :

5.1. Plan mémoire en mode direct :



5.2. Plan mémoire en mode indirect :

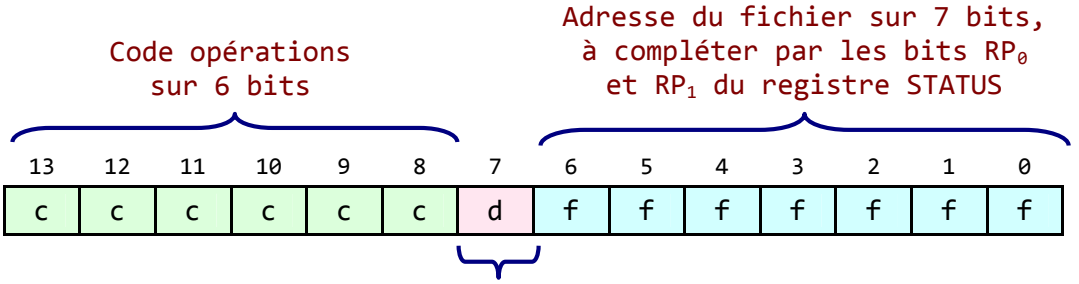


6. Le jeu d'instruction :

Le PIC16F84 a un jeu d'instructions relativement limité mais possède une architecture interne (RISC) qui permet une programmation efficace et rapide (toutes les instructions, exceptées les sauts, s'exécute en un cycle d'horloge).

6.1. Instructions opérant sur les registres :

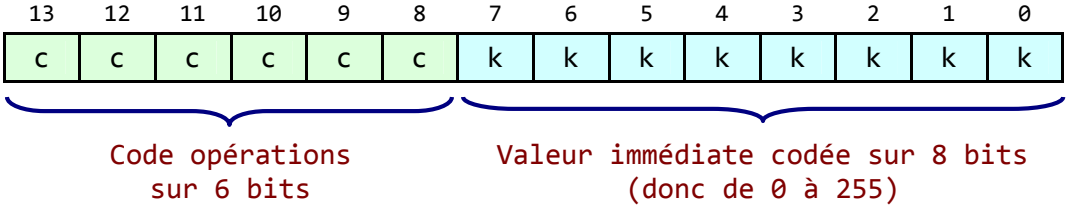
Ce sont des instructions qui manipulent les registres (RAM). Elles sont codées de la manière suivante :



Instructions opérant sur les registres			indicateurs	cycles	Code opération
ANDWF	**F, d**	ET entre W et le fichier f → (W ou f ? d)	Z	1	00.0101.dfff.ffff
IORWF	**F, d**	Ou inclusif entre W et f → (W ou f ? d)	Z	1	00.0100.dfff.ffff
XORWF	**F, d**	Ou exclusif entre W et f → (W ou f ? d)	Z	1	00.0110.dfff.ffff
COMF	**F, d**	Complément à 1 de f → (W ou f ? d)	Z	1	00.1001.dfff.ffff
SWAPF	**F, d**	Permute les deux quartets de f → (W ou f ? d)		1	00.1110.dfff.ffff
ADDWF	**F, d**	Ajouter W au fichier f → (W ou f ? d)	C, DC, Z	1	00.0111.dfff.ffff
SUBWF	**F, d**	Soustraire W du fichier f → (W ou f ? d)	C, DC, Z	1	00.0010.dfff.ffff
DECF	**F, d**	Décrémenter le fichier f → (W ou f ? d)	Z	1	00.0011.dfff.ffff
DECFSZ	**F, d**	Décrémenter f → (W ou f ? d), sauter si 0		1(2)	00.1011.dfff.ffff
INCF	**F, d**	Incrémenter le fichier f → (W ou f ? d)	Z	1	00.1010.dfff.ffff
INCFSZ	**F, d**	Incrémenter f → (W ou f ? d), sauter si 0		1(2)	00.1111.dfff.ffff
RLF	**F, d**	Rotation à gauche de f à travers C → (W ou f ? d)	C	1	00.1101.dfff.ffff
RRF	**F, d**	Rotation à droite de f à travers C → (W ou f ? d)	C	1	00.1100.dfff.ffff
CLRF	**F**	Effacer le fichier f	Z	1	00.0001.1fff.ffff
CLRW		Effacer le registre de travail W	Z	1	00.0001.0xxx.xxxx
MOVF	**F, d**	Charge f dans W si d= 0, si non réécrit f dans f	Z	1	00.1000.dfff.ffff
MOVWF	**F**	Stocker W dans le fichier f		1	00.0000.1fff.ffff

6.2. Instructions opérant sur les constantes :

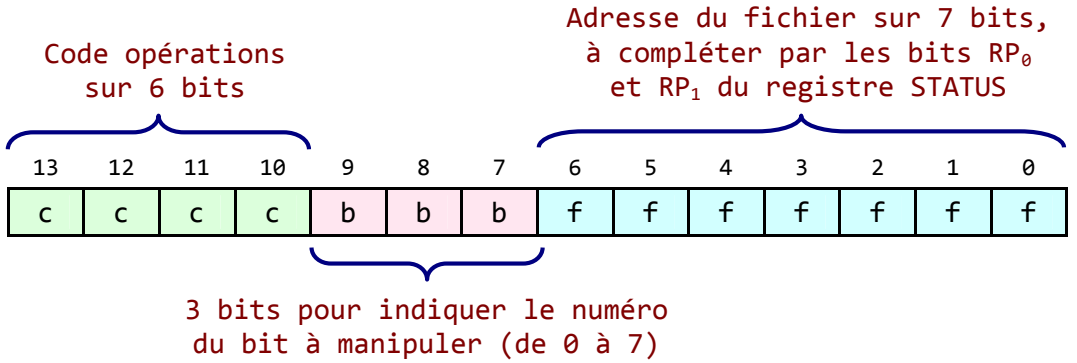
Ce sont des instructions qui manipulent des données qui sont codées dans l'instruction directement. Elles sont codées de la manière suivante :



Instructions opérant sur les constantes			indicateurs	cycles	Code opération
ADDLW	**k**	Ajouter la constante k à W : W ← W + K	C, DC, Z	1	11.111x.kkkk.kkkk
SUBLW	**k**	Soustraire W de la constante k : W ← K - W	C, DC, Z	1	11.110x.kkkk.kkkk
ANDLW	**k**	ET entre la constante k et W : W ← K et W	Z	1	11.1001.kkkk.kkkk
IORLW	**k**	Ou inclusif entre k et W : W ← K ou W	Z	1	11.1000.kkkk.kkkk
XORLW	**k**	Ou exclusif entre k et W : W ← K ou ex W	Z	1	11.1010.kkkk.kkkk
MOVLW	**k**	Charger la constante k dans W : W ← K		1	11.00xx.kkkk.kkkk
RETLW	**k**	Retour d'un sous programme avec k dans W		2	11.01xx.kkkk.kkkk

6.3. Instructions opérant sur les bits d'un fichier :

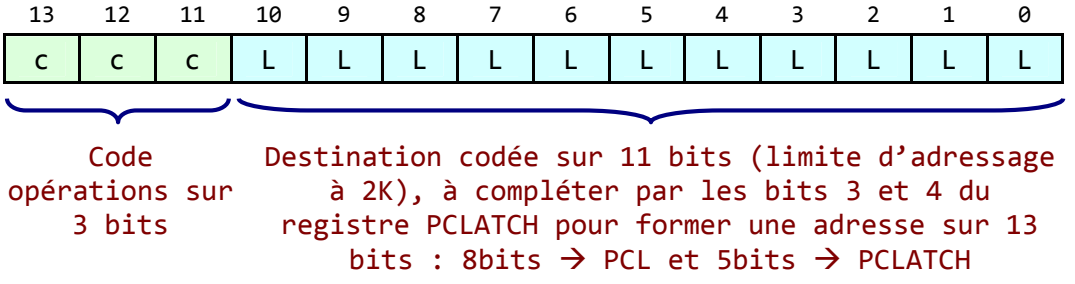
Ce sont des instructions destinées à manipuler directement les bits d'un registre d'une case mémoire. Elles sont codées de la manière suivante :



Instructions opérant sur les bits d'un fichier			indicateurs	cycles	Code opération
BCF	**F, b**	Mettre à zéro le bit numéro b du registre f		1	01.00bb.bfff.fff
BSF	**F, b**	Mettre à un le bit numéro b du registre f		1	01.01bb.bfff.fff
BTFSC	**F, b**	Tester le bit b de f, sauter une instruction si 0		1(2)	01.10bb.bfff.fff
BTFSS	**F, b**	Tester le bit b de f, sauter une instruction si 1		1(2)	01.11bb.bfff.fff

6.4. Instructions de saut/appel et instructions générales :

Les instructions de saut/appel provoquent une rupture dans la séquence de déroulement du programme. Elles sont codées de la manière suivante :



Instructions générales			indicateurs	cycles	Code opération
GOTO	**L**	Branchement à la ligne de label L		2	10.1LLL.LLLL.LLLL
CALL	**L**	Branchement à un sous programme de label L		2	10.0LLL.LLLL.LLLL
RETURN		Retourner d'un sous programme		2	00.0000.0000.1000
RETFIE		Retourner d'un s. programme d'interruption		2	00.0000.0000.1001
CLRWTD		Effacer le Watchdog Timer	TO, PD	1	00.0000.0101.0100
SLEEP		Mettre le µC en mode veille	TO, PD	1	00.0000.0110.0011
NOP		Pas d'opération		1	00.0000.0xx0.0000

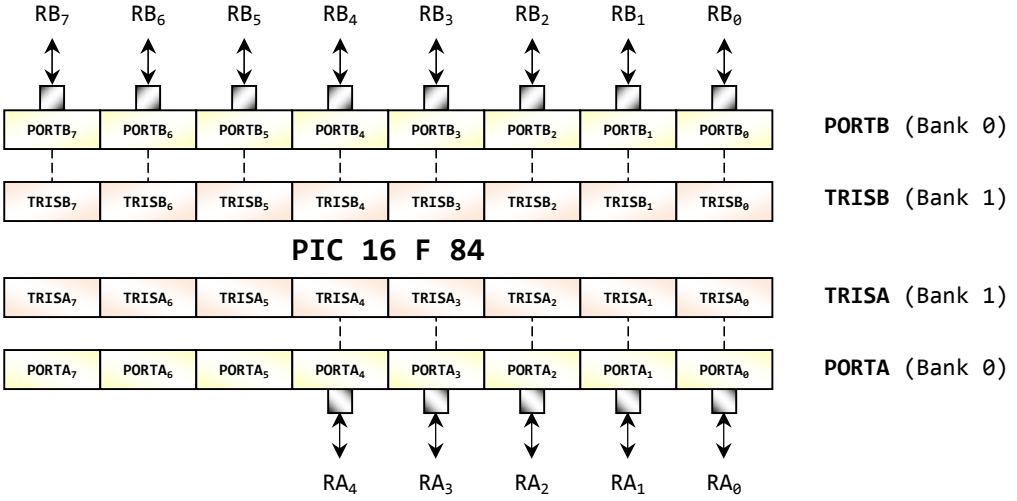
1. Introduction :

Pour communiquer avec l'extérieur le PIC 16F84 dispose de 2 ports : **PORTA et PORTB**. Les ports sont **bidirectionnels**, ce qui signifie qu'ils peuvent être configurés et utilisés comme des entrées ou des sorties.

Le microcontrôleur reçoit les informations sur un port d'entrée :

- informations logiques issues de capteurs sur un ou plusieurs bits d'un port d'entrée,
- informations numériques codées sur 8 bits sur un port entier.
- informations analogiques variables dans le temps, si le PIC est doté d'un convertisseur analogique / numérique.

Le microcontrôleur traite ces données et les utilisent pour commander des circuits qui sont connectés sur un port de sortie.



2. Le port d'E/S PORTA :

Le port A désigné par **PORTA** est un port de **5** bits (RA_0 à RA_4). RA_5, RA_6 et RA_7 ne sont pas accessibles. 2 registres pour utiliser ce port :

- **TRISA : Registre de direction ou registre de configuration du PORTA**

La configuration de direction se fait par positionnement des bits du registre TRISA, de la manière suivante :

- Si le bit **$TRISA_i = 0$** alors la broche **RA_i** du PORTA est configurée en **sortie** ;
- Si le bit **$TRISA_i = 1$** alors la broche **RA_i** du PORTA est configurée en **entrée** ;

- **PORTA : Registre de sortie du PORTA**

La lecture ou l'écriture des broches **RA_0** … **RA_4**, se ramène tout simplement à une lecture ou écriture du registre **PORTA**.

La broche **RA_4** est multiplexée avec l'entrée d'horloge du registre **TMR0**.

3. Le port d'E/S PORTB :

Le port B désigné par **PORTB** est un port de **8** bits (RB_0 à RB_7). 2 registres pour utiliser ce port :

- **TRISB : Registre de direction ou registre de configuration du PORTB**

La configuration de direction se fait par positionnement des bits du registre TRISB, de la manière suivante :

- Si le bit **$TRISB_i = 0$** alors la broche **RB_i** du PORTA est configurée en **sortie** ;
- Si le bit **$TRISB_i = 1$** alors la broche **RB_i** du PORTA est configurée en **entrée** ;

- **PORTB : Registre de sortie du PORTB**

La lecture ou l'écriture des broches **RB_0** … **RB_7**, se ramène tout simplement à une lecture ou écriture du registre **PORTB**.

La broche **RB_0** est multiplexée avec l'interruption **INT**.

4. Configuration d'un PORTx :

Les registres TRISx appartiennent à la Bank 1 des SFR. Lors de l'initialisation du µC, il ne faut pas oublier de changer de page mémoire pour les configurer.

Pour configurer un PORTx, il faut :

- Accéder à la **Bank 1** ;
- Déterminer le **mot** à mettre dans le registre TRISx ;
- Mettre ce **mot** dans le registre de travail **W** ;
- Transférer le contenu de **W** dans le registre **TRISx** ;
- Accéder à la **Bank 0**, pour pouvoir accéder au PORTx.

5. Exemples d'application :

Exemple 1 :

Configurer le PORTB en entrée, lire le contenu du PORTB et mettre le résultat dans la case mémoire d'adresse $0C_H$.

```
..............................  ; Accès à la Bank1
..............................  ;
..............................  ; PORTB en entrée
..............................  ; Accès à la Bank0
..............................  ; W ← PORTB
..............................  ; (0C_H) ← W
```

Bit n°	7	6	5	4	3	2	1	0
PORTB	...	...	...	...	...	...	...	...
TRISB	...	...	...	...	...	...	...	...
Hex	...				...			

Exemple 2 :

Configurer le PORTB en sortie et initialiser le PORTB à FF_H.

```
..............................  ; Accès à la Bank1
..............................  ;
..............................  ; PORTB en sortie
..............................  ; Accès à la Bank0
..............................  ; W ← FF_H
..............................  ; PORTB ← W
```

Bit n°	7	6	5	4	3	2	1	0
PORTB	...	...	...	...	...	...	...	...
TRISB	...	...	...	...	...	...	...	...
Hex	...				...			

Exemple 3 :

Configurer les broches :

- RB_1, RB_3, RB_5, RB_7 du PORTB en entrée ;
- RB_0, RB_2, RB_4, RB_6 du PORTB en sortie ;
- RA_4 du PORTA en entrée ;
- RA_0, RA_1, RA_2, RA_3 du PORTA en sortie.

```
..............................  ; Accès à la Bank1
..............................  ;
..............................  ; Configuration PORTB
..............................  ;
..............................  ; Configuration PORTA
..............................  ; Accès à la Bank0
```

Bit n°	7	6	5	4	3	2	1	0
PORTB	...	...	...	...	...	...	...	...
TRISB	...	...	...	...	...	...	...	...
Hex	...				...			
PORTA	...	...	...	...	...	...	...	...
TRISA	...	...	...	...	...	...	...	...
Hex	...				...			

Exemple 4 :

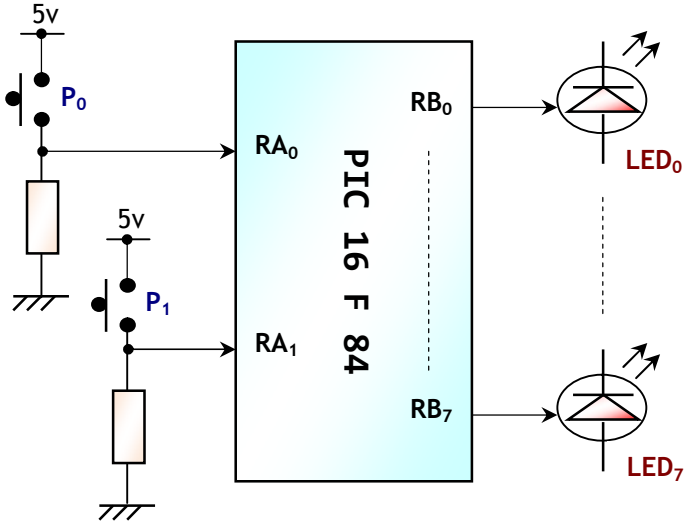
Configurer les broches :
- RB_0, RB_1, RB_6, RB_7 en entrée et RB_2, RB_3, RB_4, RB_5 en sortie ;
- RA_2, RA_3, RA_4 en entrée et RA_0, RA_1 en sortie ;

```
..............................  ; Accès à la Bank1
..............................  ;
..............................  ; Configuration PORTB
..............................  ;
..............................  ; Configuration PORTA
..............................  ; Accès à la Bank0
```

Bit n°	7	6	5	4	3	2	1	0
PORTB	…	…	…	…	…	…	…	…
TRISB	…	…	…	…	…	…	…	…
Hex	…				…			
PORTA	…	…	…	…	…	…	…	…
TRISA	…	…	…	…	…	…	…	…
Hex	…				…			

Exemple 5 :

Soit le montage suivant :

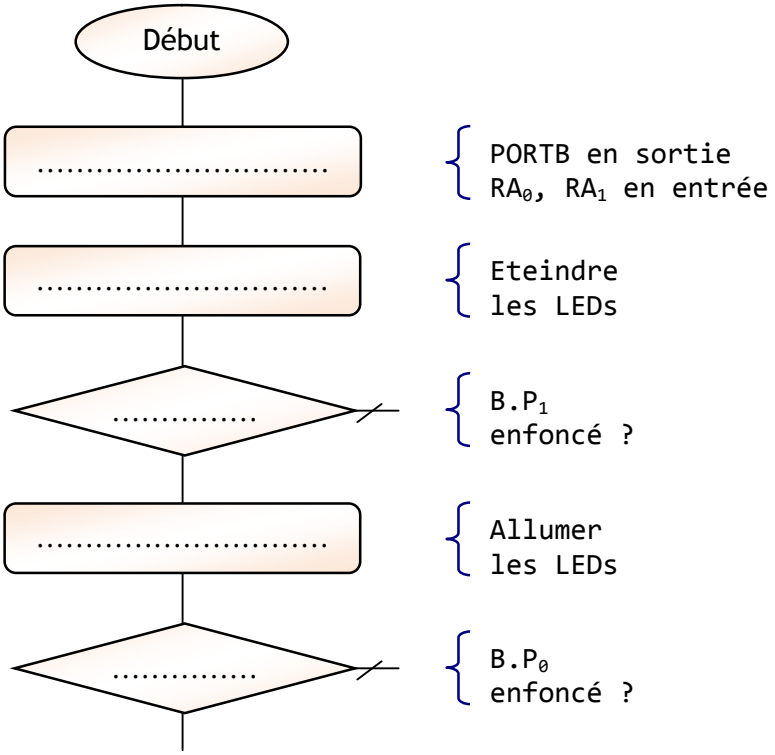


Fonctionnement :
- Au repos les LEDs sont éteintes ;
- Une action sur le bouton poussoir **P_1 allume** toutes les LEDs ;
- Une action sur le bouton poussoir **P_0 éteint** toutes les LEDs.

Travail Demandé :

Faire un programme qui permet d'avoir le fonctionnement décrit ci-dessus.

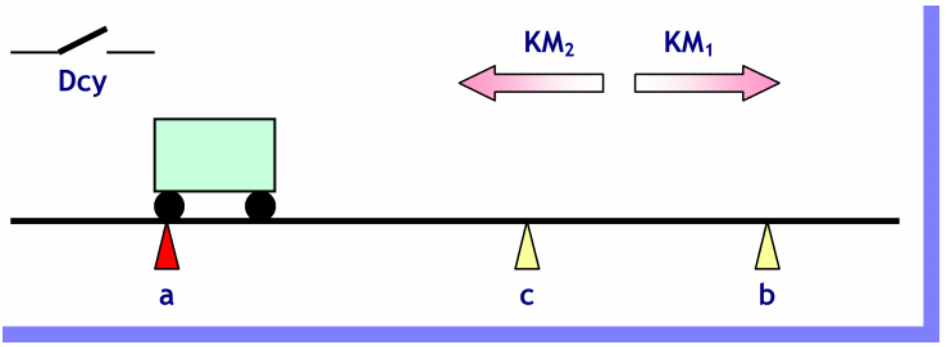
Organigramme :



Programme assembleur :

```
                ..............................
                ..............................
                ..............................
                ..............................
                ..............................
                ..............................
LAB1            ..............................
LAB2            ..............................
                ..............................
                ..............................
                ..............................
LAB3            ..............................
                ..............................
                ..............................
```

1. Cahier des charges :



Après l'ordre de départ cycle « dcy », le chariot part jusque b, revient en c, repart en b puis rentre en a.

2. Identification des entrées et des sorties :

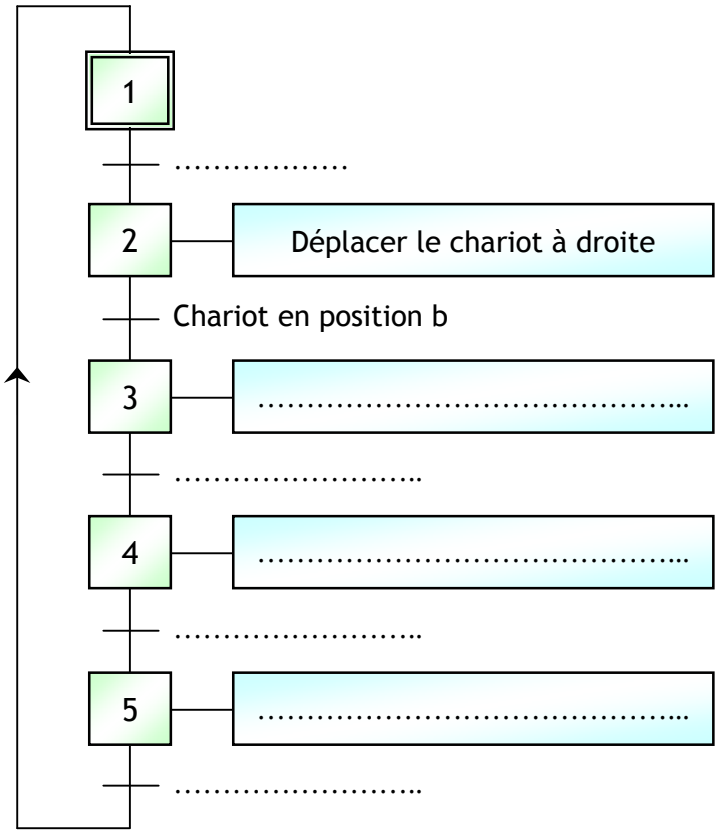
Mouvement	Actionneur	Ordres
Déplacer le chariot à droite	Moteur **M**	KM_1
Déplacer le chariot à gauche	Moteur **M**	KM_2

Compte-rendu	Capteur	Mnem.
Chariot en position a	Détecteur mécanique à levier	a
Chariot en position b	Détecteur mécanique à levier	b
Chariot en position c	Détecteur mécanique à levier	c

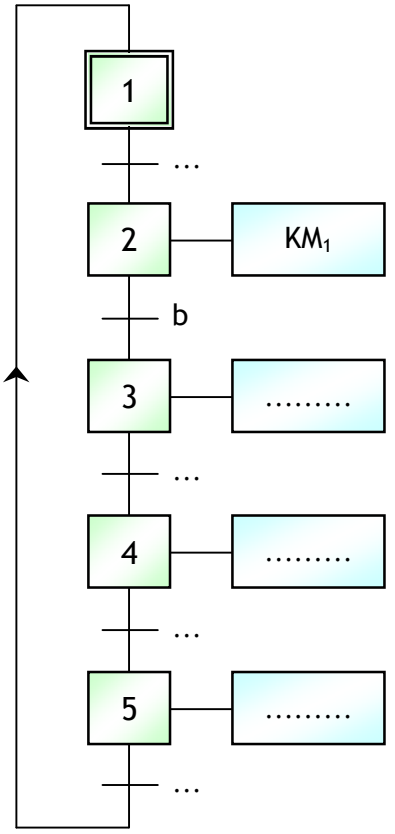
Consigne	Constituant	Mnem.
Départ cycle	Bouton poussoir	Dcy

3. GRAFCET :

3.1. GRAFCET P.O :

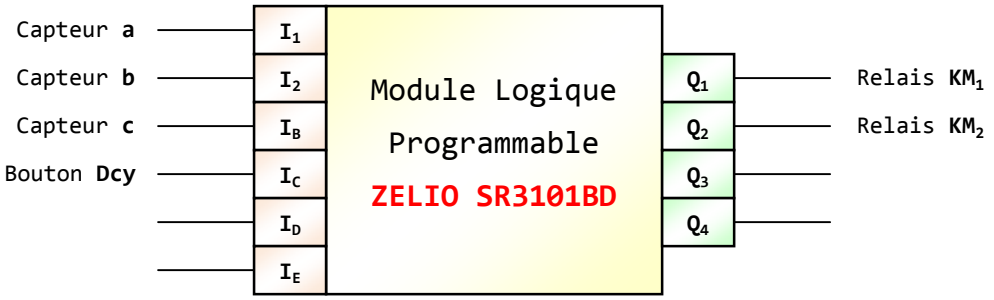


3.2. GRAFCET P.C :

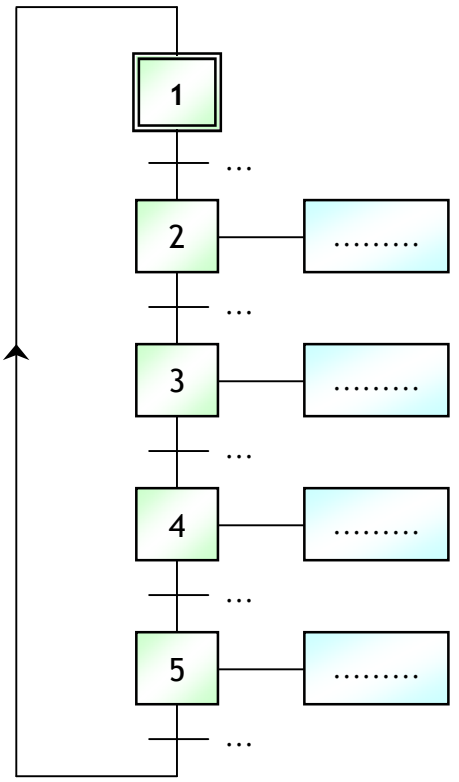


4. Commande par l'A.P.I ZELIO SR3101BD :

4.1. Affectation des entrées/sorties :



4.2. GRAFCET P.C codé A.P.I :

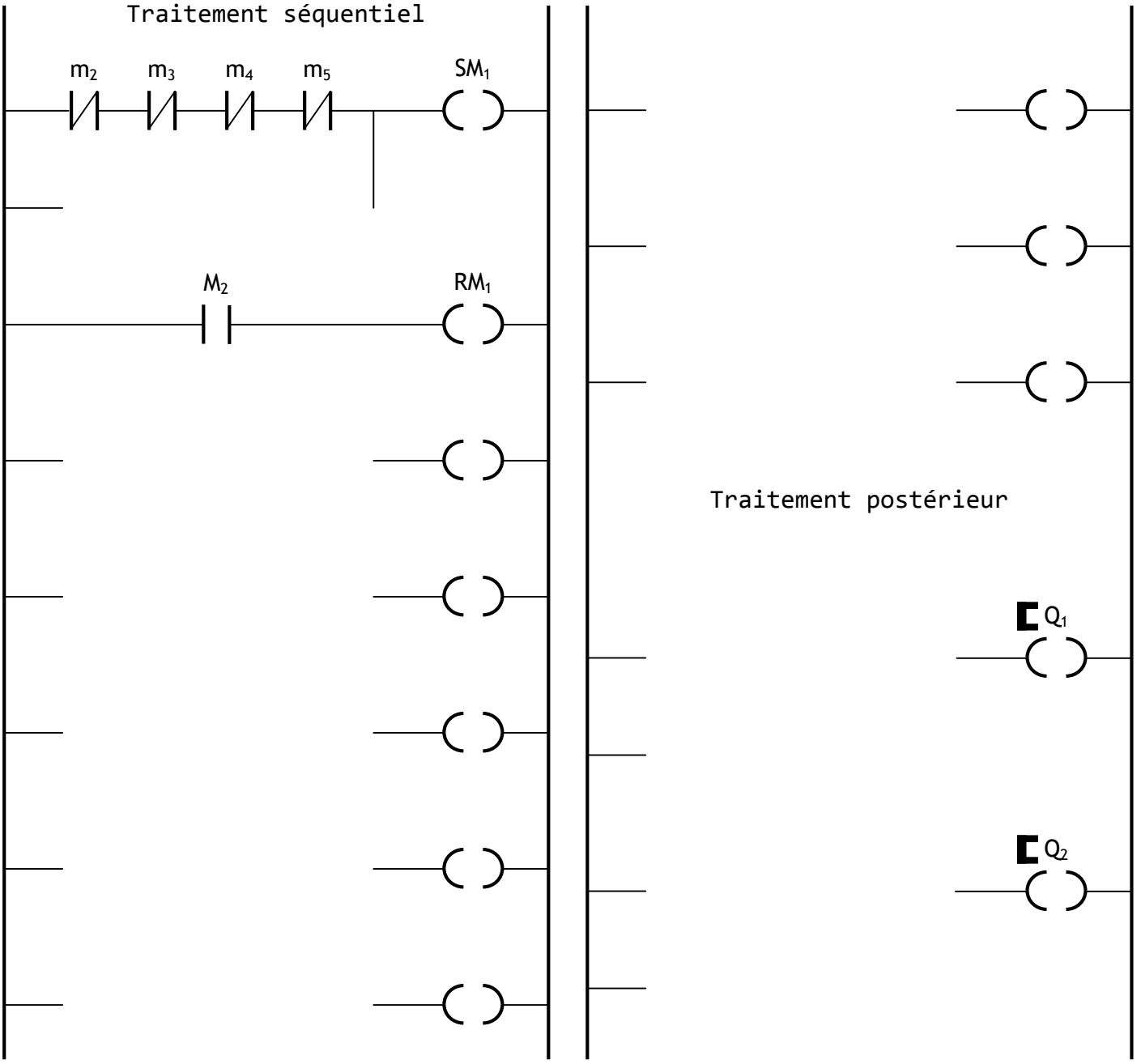


4.3. Mise en équation :

Étapes	Activation	Désactivation
1	$SM_1=\overline{M_2}.\overline{M_3}.\overline{M_4}.\overline{M_5} + I_1.M_5$	$RM_1 = M_2$
2	SM_2 = ...	RM_2 = ...
3	SM_3 = ...	RM_3 = ...
4	SM_4 = ...	RM_4 = ...
5	SM_5 =.....	RM_5 = ...

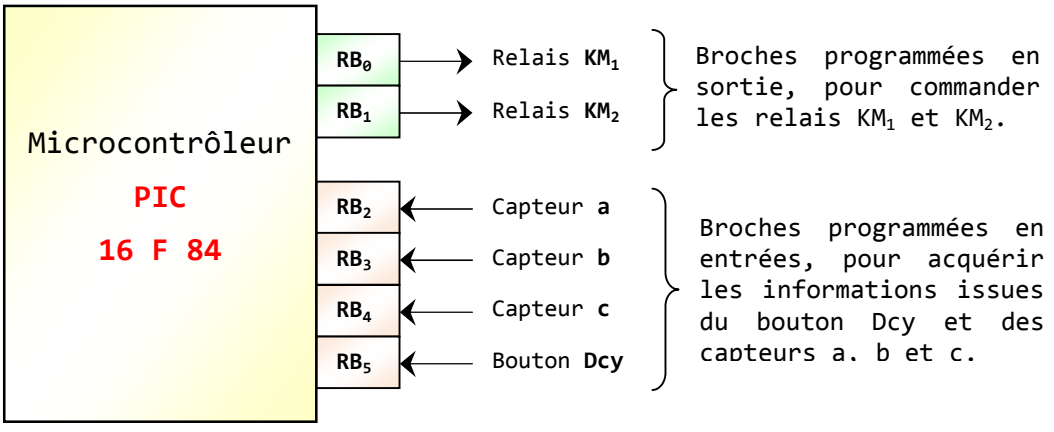
Actions	Q_1 = ...	Q_2 = ...

4.4. Programme LADDER :

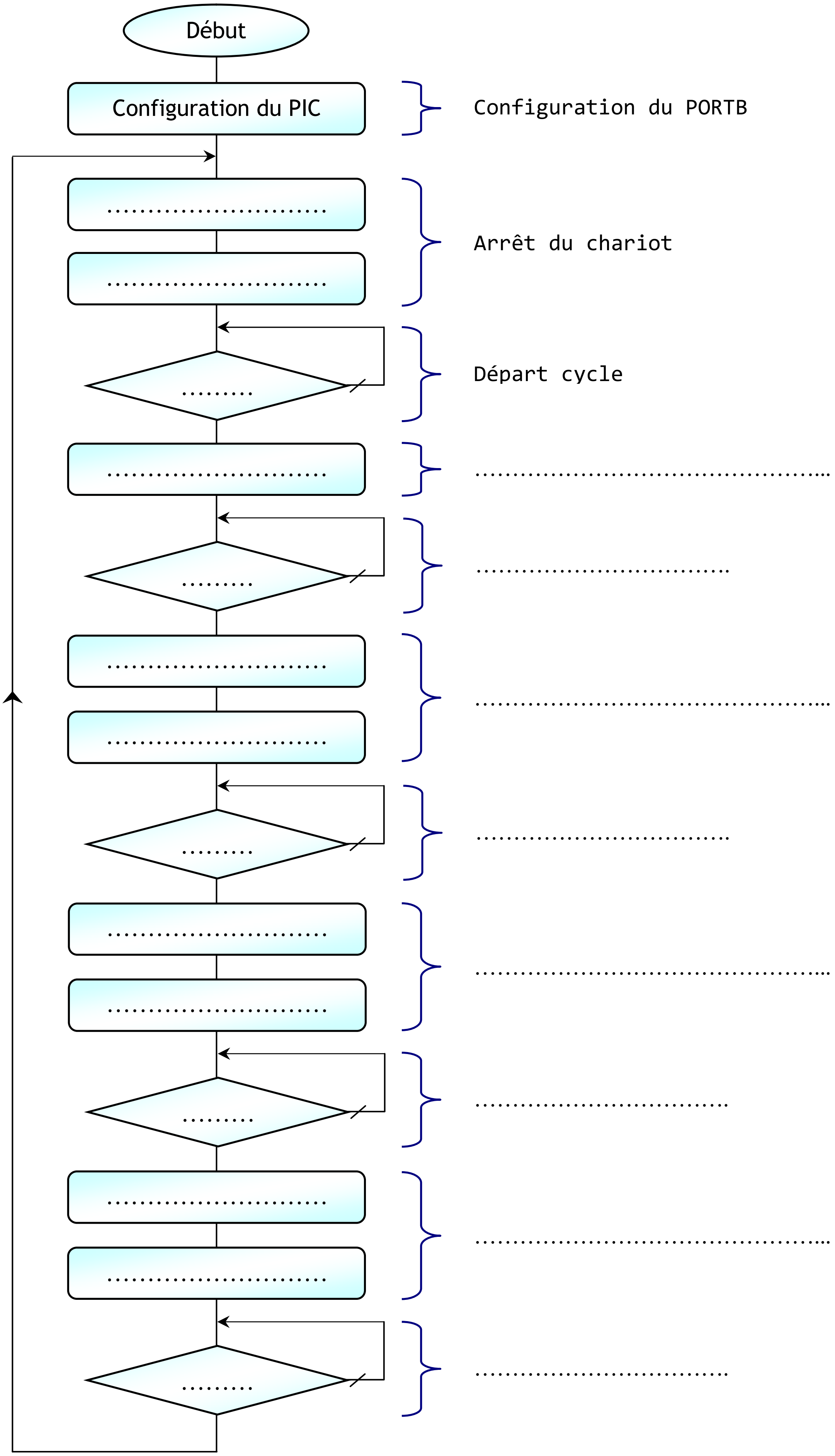


5. Commande par le microcontrôleur PIC 16 F 84 :

5.1. Affectation des entrées/sorties :



5.2. Organigramme :



5.3. Programme ASSEMBLEUR :

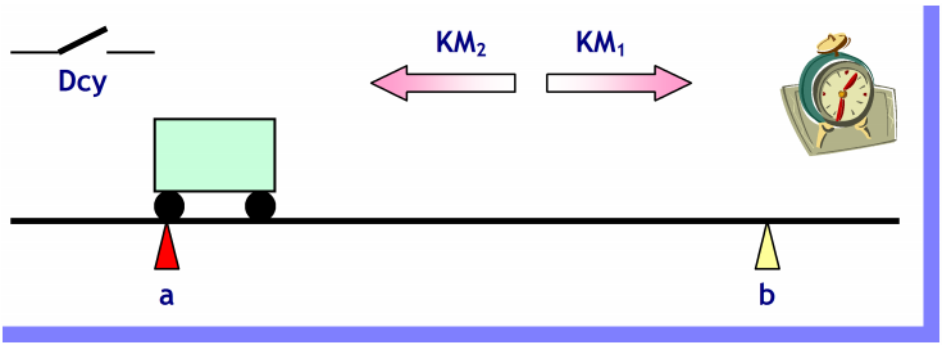
Programme de configuration

```
                .....          ; Accès à la BANK 1
                .....          ;
                .....          ; Configuration du PORTB
                .....          ; Accès à la BANK 0
```

Programme principal

```
Lab1    BCF     PORTB, 0       ;
        BCF     PORTB, 1       ; Arrêt du chariot
Lab2    BTFSS   PORTB, 5       ;
        GOTO    Lab2           ; Départ cycle
                .....          ; Déplacer le chariot à droite
Lab3            .....          ; Chariot en position b
                .....          ;
                .....          ;
                .....          ; Déplacer le chariot à gauche
Lab4            .....          ; Chariot en position c
                .....          ;
                .....          ; Déplacer le chariot à droite
                .....          ;
Lab5            .....          ; Chariot en position b
                .....          ;
                .....          ;
                .....          ; Déplacer le chariot à gauche
Lab6            .....          ; Chariot en position a
                .....          ;
                .....          ; Reprendre
                END            ; Fin du fichier
```

1. Cahier des charges :



Après l'ordre de départ cycle « dcy », le chariot part jusque b, reste en b un temps T_1 de 15 s puis rentre en a.

2. Identification des entrées et des sorties :

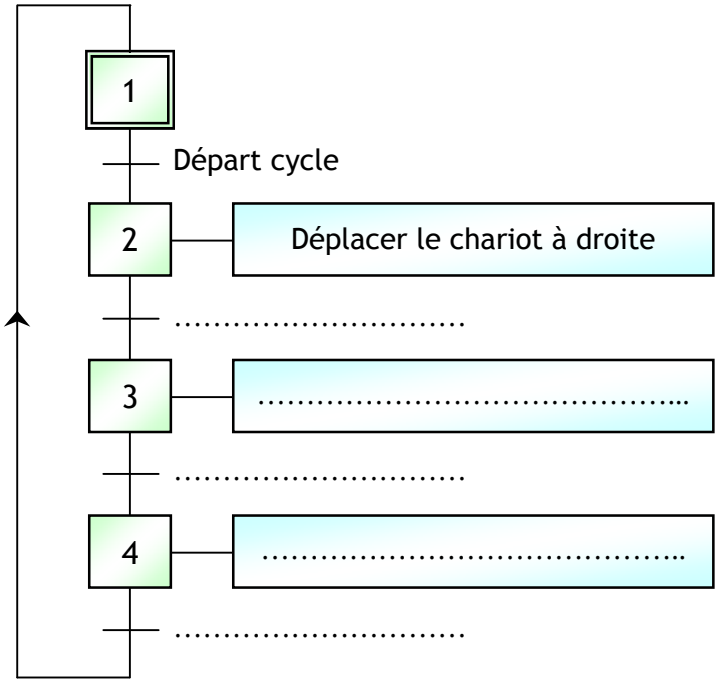
Mouvement	Actionneur	Ordres
Déplacer le chariot à droite	Moteur **M**	KM_1
Déplacer le chariot à gauche	Moteur **M**	KM_2

Compte-rendu	Capteur	Mnem.
Chariot en position a	Détecteur mécanique à levier	a
Chariot en position b	Détecteur mécanique à levier	b

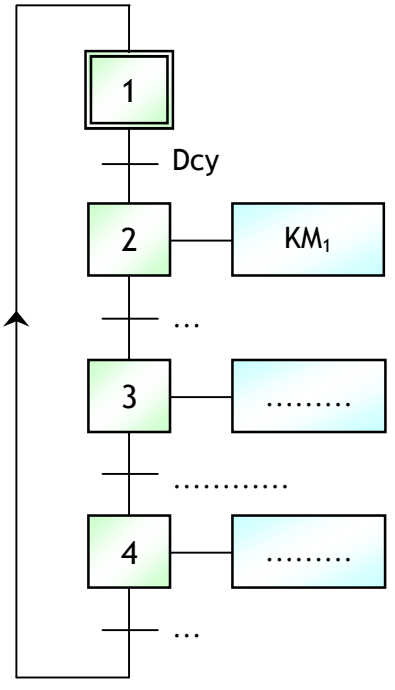
Consigne	Constituant	Mnem.
Départ cycle	Bouton poussoir	Dcy

3. GRAFCET :

3.1. GRAFCET P.O :

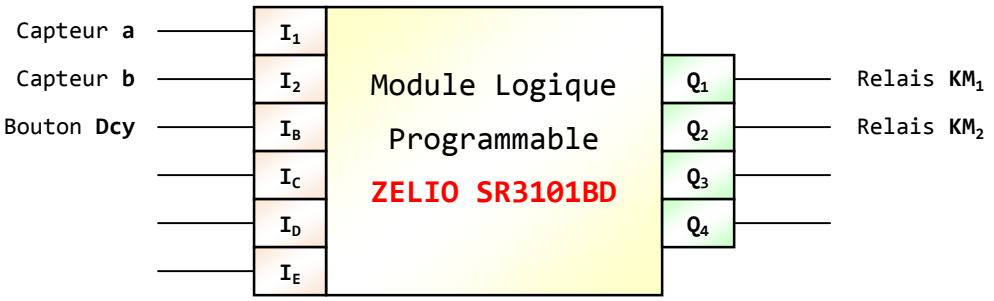


3.2. GRAFCET P.C :



4. Commande par l'A.P.I ZELIO SR3101BD :

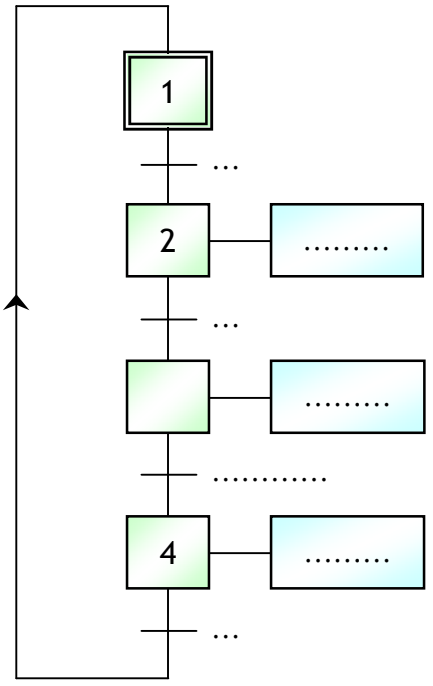
4.1. Affectation des entrées/sorties :



Pour la temporisation de 15s, on utilise le bloc temporisateur T_1 de l'A.P.I ZELIO **SR3101BD** :

- TT_1 : entrée de déclenchement de la temporisation ;
- T_1 : sortie de fin de temporisation.

4.2. GRAFCET P.C codé A.P.I :

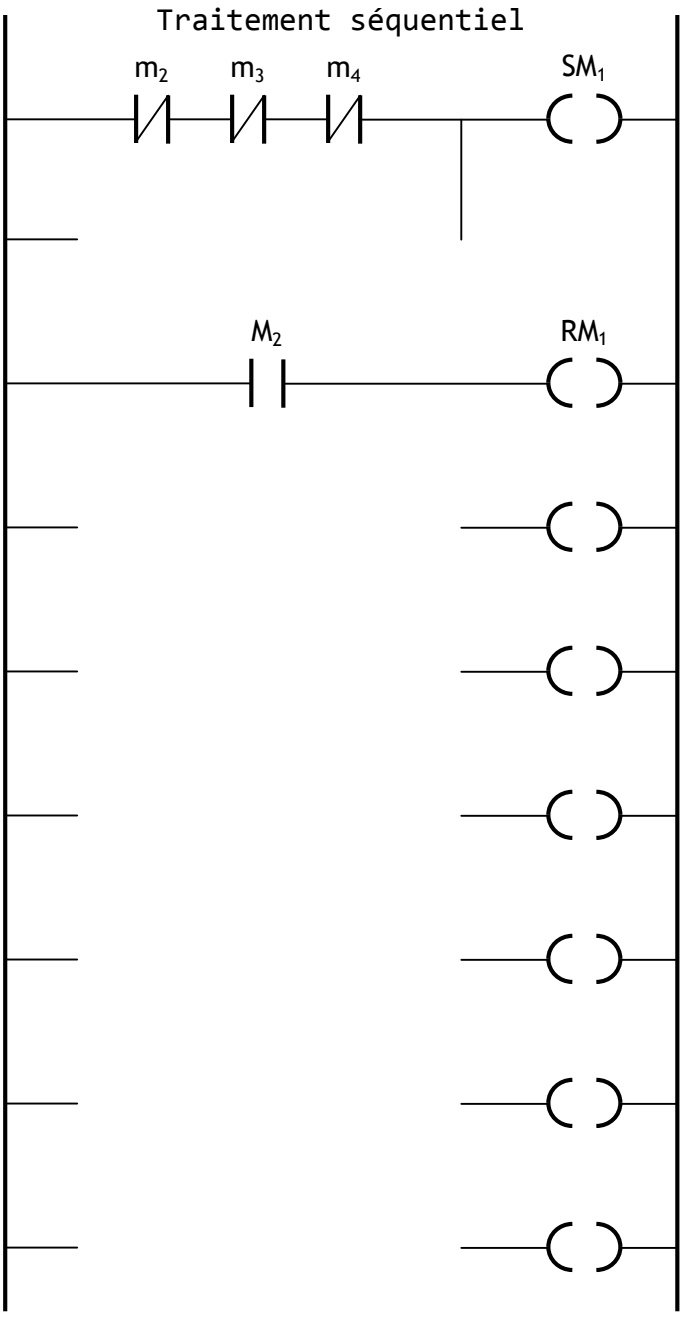


4.3. Mise en équation :

Étapes	Activation	Désactivation
1	$SM_1=\overline{M_2}.\overline{M_3}.\overline{M_4}. + M_4.I_1$	$RM_1 = M_2$
2	SM_2 = ...	RM_2 = ...
3	SM_3 = ...	RM_3 = ...
4	SM_4 = ...	RM_4 = ...

Actions	Q_1 = ...	Q_2 = ...	TT_1 = ...

4.4. Programme LADDER :



LES BLOCS FONCTIONS 'TEMPORISATION'

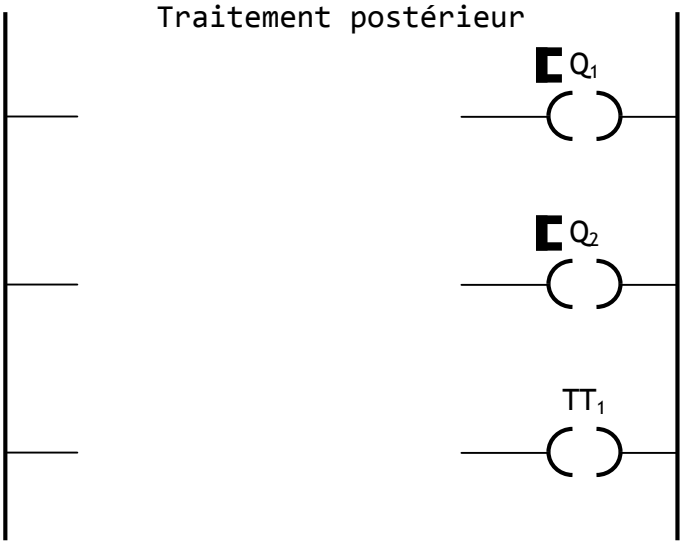
Le module logique Zelio **SR3101BD** possède 16 blocs temporisateurs de T_1 à T_G.
Le bloc fonction Temporisateur permet de temporiser des actions. Il possède :

- une entrée de remise à zéro **RT** ;
- une entrée de commande **TT**;
- une sortie fin de temporisation **T** ou **t**;
- une valeur de présélection.

TT (commande) : Utilisé comme bobine, cet élément représente l'entrée de commande du Bloc fonction Temporisation. Son fonctionnement dépend du type utilisé.

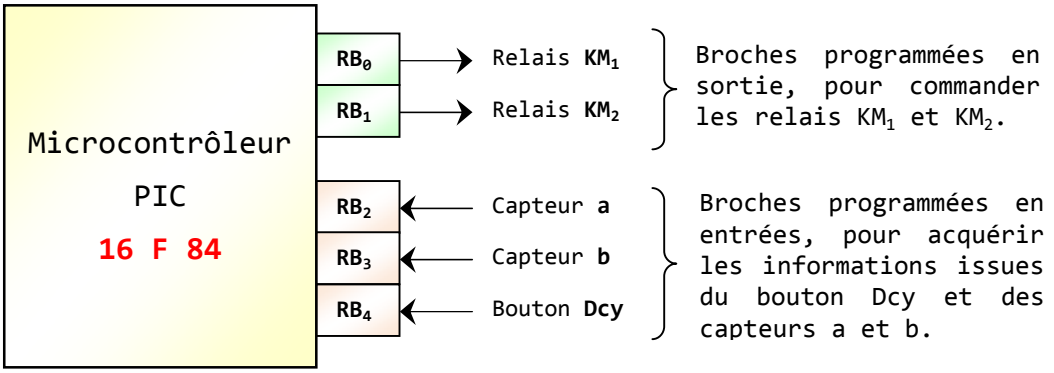
RT (remise à 0) : Utilisé comme bobine, cet élément représente l'entrée de remise à zéro. L'excitation de la bobine a pour effet de :

- remettre à zéro la valeur courante de la Temporisation ;
- désactiver le contact T, le bloc est prêt pour un nouveau cycle de temporisation.



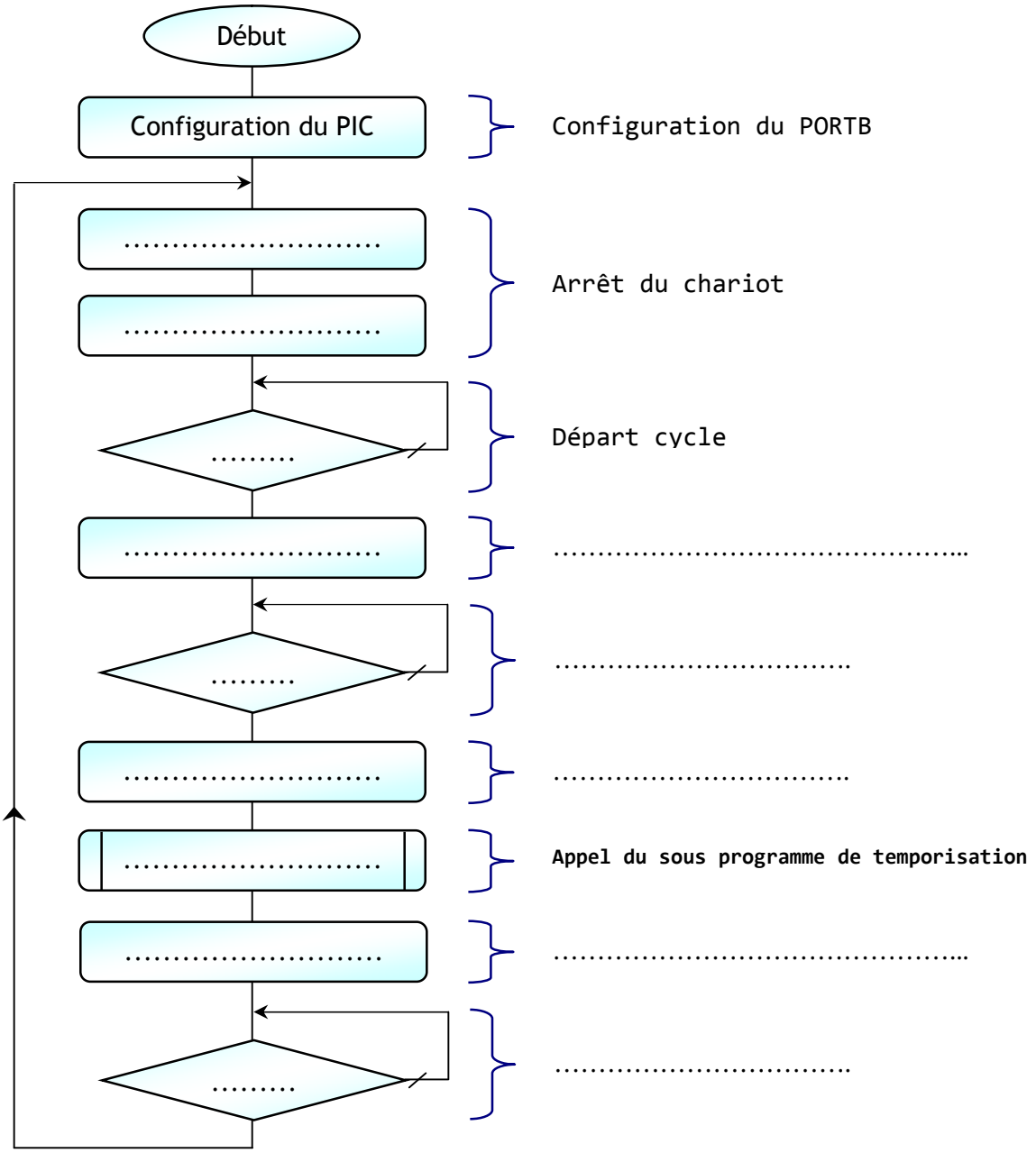
5. Commande par le microcontrôleur PIC 16 F 84 :

5.1. Affectation des entrées/sorties :



Pour la temporisation de 15s, on utilise un sous de temporisation **Tempo**.

5.2. Organigramme :



5.3. Programme ASSEMBLEUR :

____Programme de configuration____

.....

.....

.....

.....

_____Programme principal _____

Lab$_1$ .....

.....

Lab$_2$ BTFSS PORTB, 4

GOTO Lab$_2$

.....

Lab$_3$ .....

.....

.....

.....

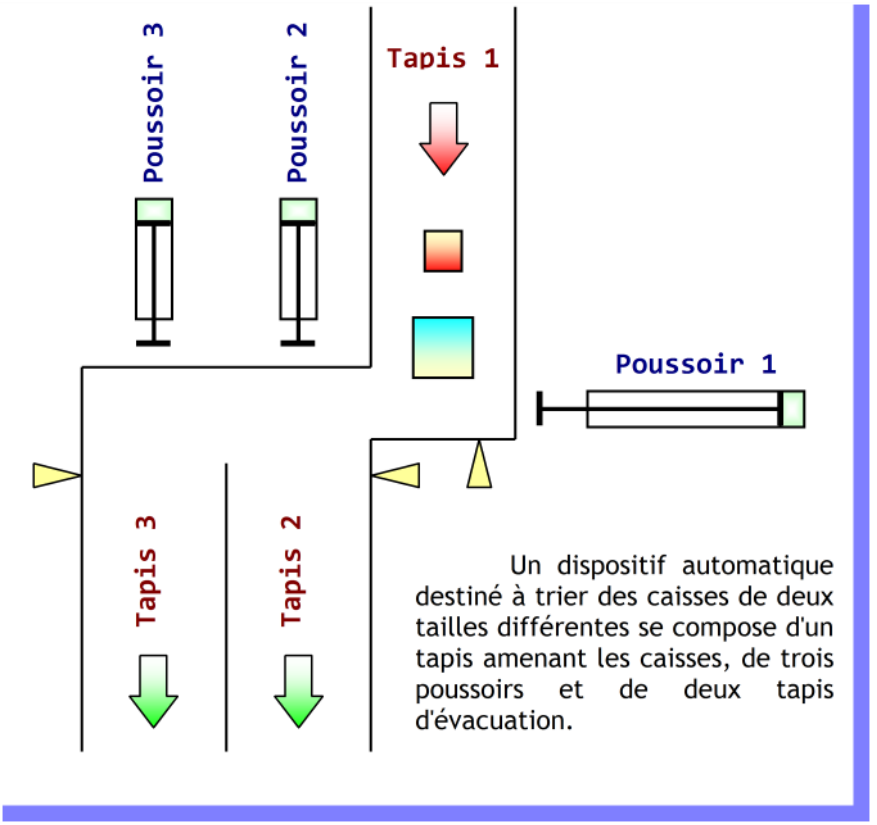
.....

Lab$_4$ .....

.....

.....

1. Cahier des charges :



Le poussoir 1 pousse les petites caisses devant le poussoir 2 qui, à son tour, les transfère sur le tapis d'évacuation 2, alors que les grandes caisses sont poussées devant le poussoir 3, ce dernier les évacue sur le tapis 3. Pour effectuer la sélection des caisses, un dispositif de détection placé devant le poussoir 1 permet de reconnaître sans ambiguïté le type de caisse qui se présente.

N.B : Les poussoirs 2 et 3 ne peuvent avancer que lorsque le poussoir 1 est en arrière.

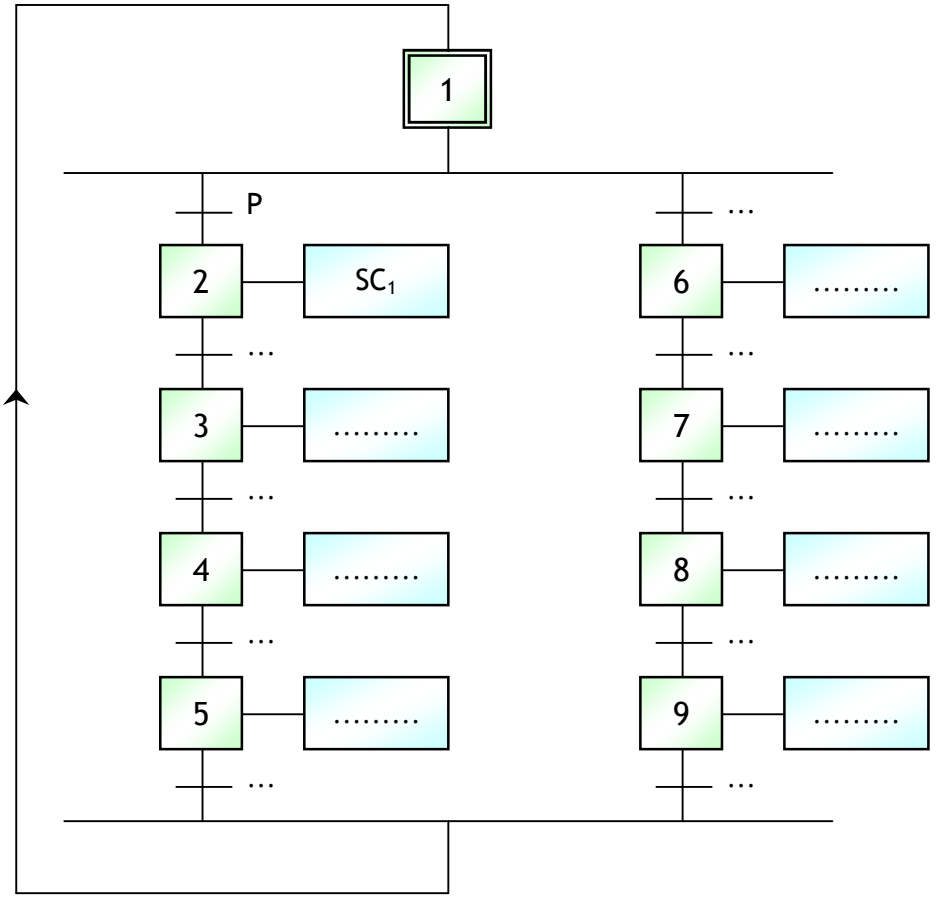
2. Identification des entrées et des sorties :

Mouvement	Actionneur et action		Ordres
Sortir la tige du vérin 1	Poussoir 1 (Vérin)	SC_1	$14\ M_1$
Rentrer la tige du vérin 1		RC_1	$12\ M_1$
Sortir la tige du vérin 2	Poussoir 2 (Vérin)	SC_2	$14\ M_2$
Rentrer la tige du vérin 2		RC_2	$12\ M_2$
Sortir la tige du vérin 3	Poussoir 3 (Vérin)	SC_3	$14\ M_3$
Rentrer la tige du vérin 3		RC_3	$12\ M_3$

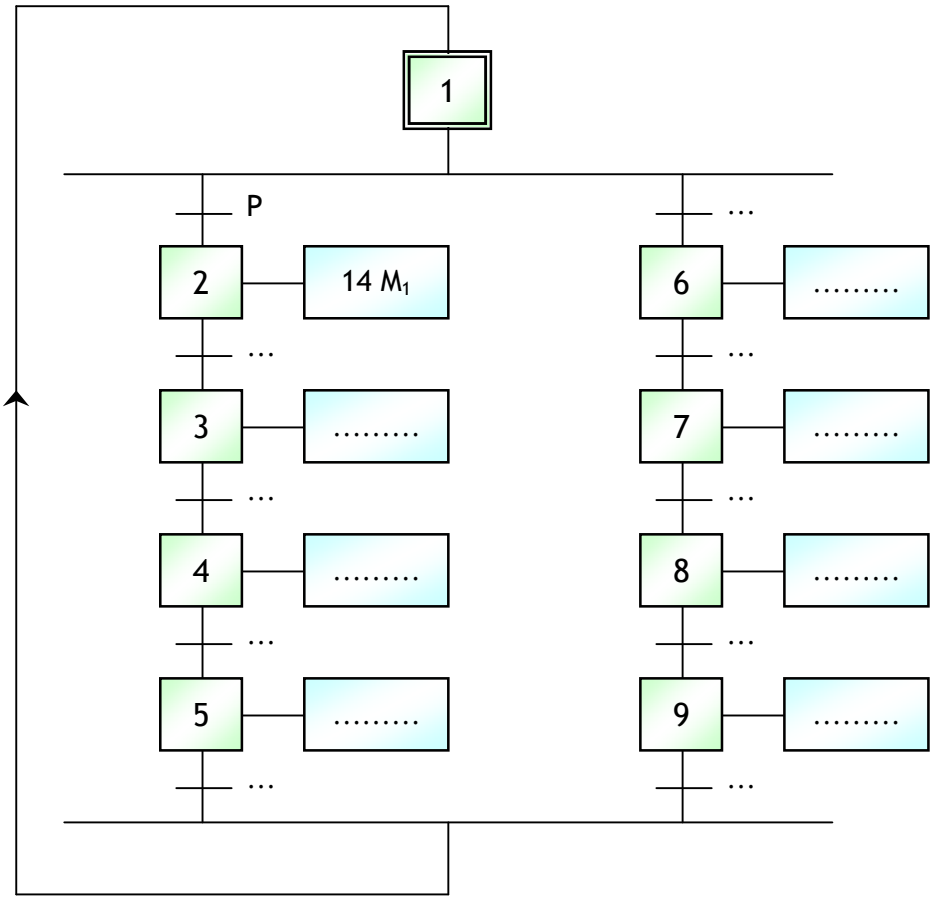
Compte-rendu	Capteur	Mnem.
P_1 en arrière	Détecteur mécanique à levier	L_{10}
P_2 en arrière	Détecteur mécanique à levier	L_{20}
P_3 en arrière	Détecteur mécanique à levier	L_{30}
Caisse sur tapis 2	Détecteur mécanique à levier	L_{21}
Caisse sur tapis 3	Détecteur mécanique à levier	L_{31}
Caisse devant P_2	Détecteur mécanique à levier	L_{112}
Caisse devant P_3	Détecteur mécanique à levier	L_{113}
Grande caisse	Détecteur mécanique à levier	G
Petite caisse	Détecteur mécanique à levier	P

3. GRAFCET :

3.1. GRAFCET P.O :

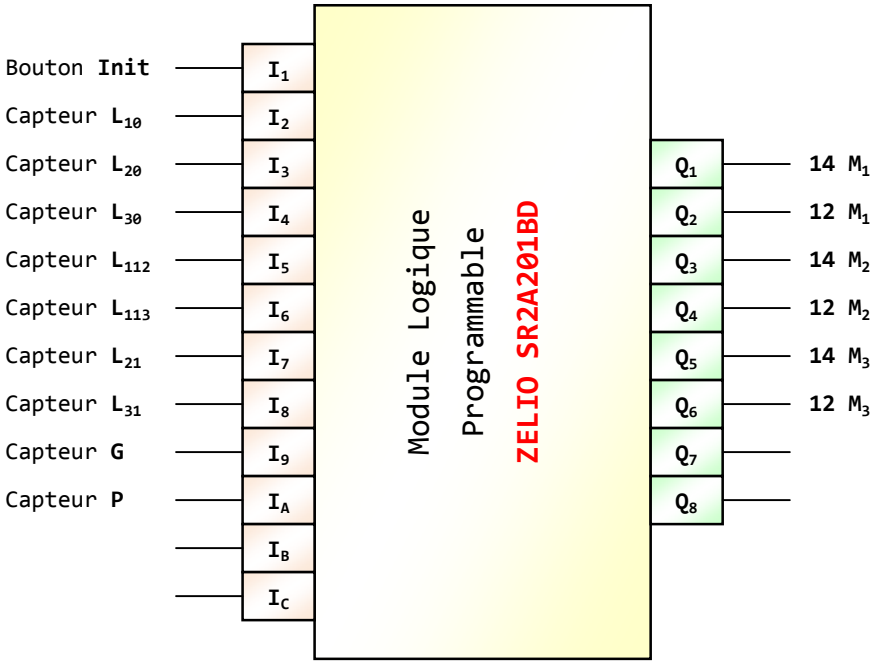


3.2. GRAFCET P.C :



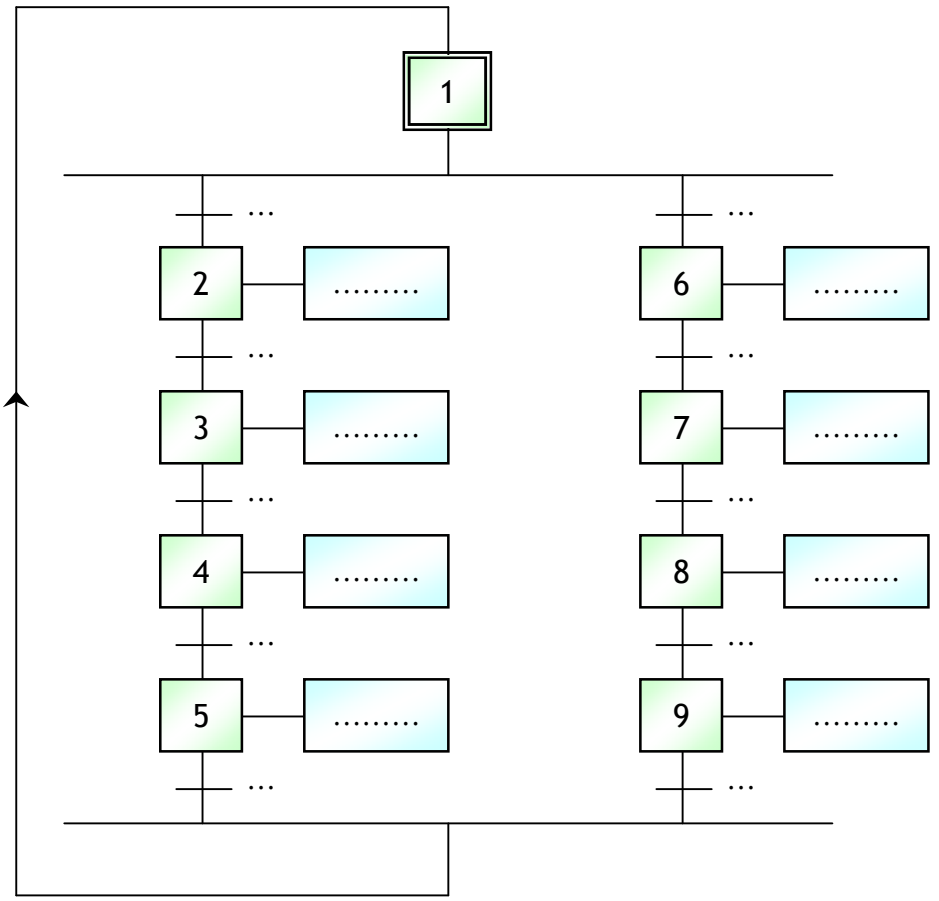
4. Commande par l'A.P.I ZELIO SR2A201BD :

4.1. Affectation des entrées/sorties :



Init : Bouton d'initialisation du système.

4.2. GRAFCET P.C codé A.P.I :

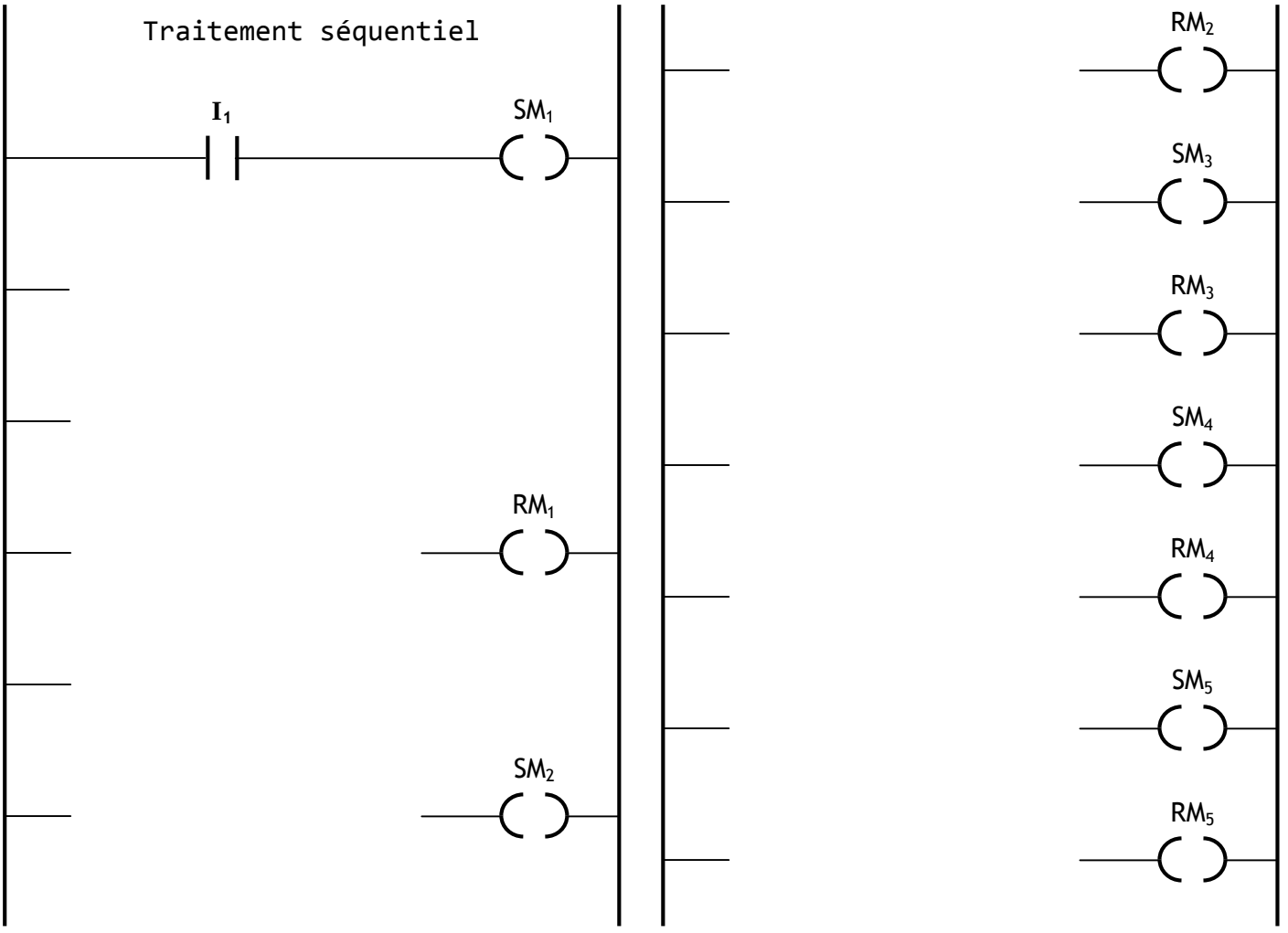


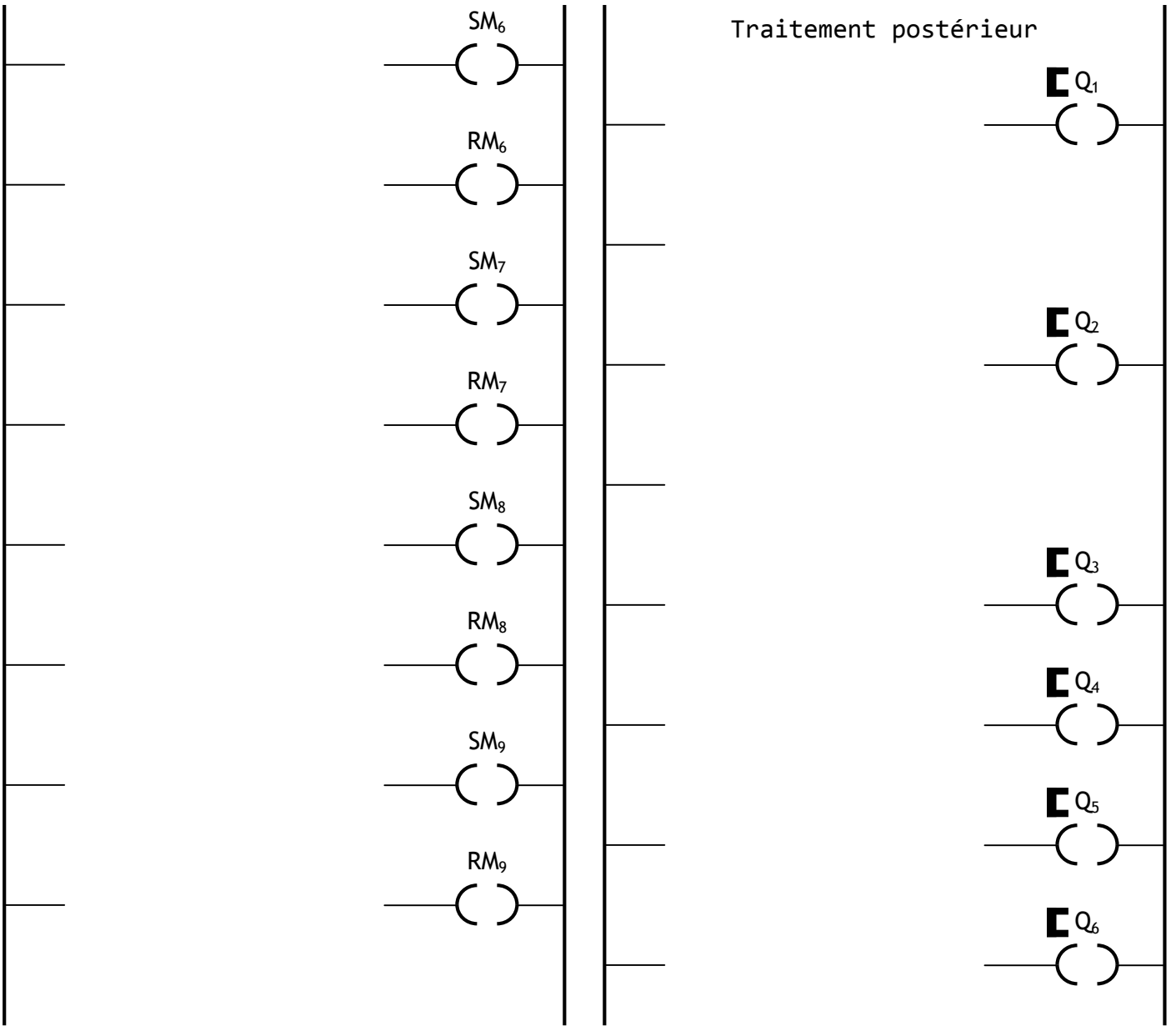
4.3. Mise en équation :

Étapes	Activation	Désactivation
1	$SM_1 = I_1 +$	RM_1 = ...
2	SM_2 = ...	RM_2 = ...
3	SM_3 = ...	RM_3 = ...
4	SM_4 = ...	RM_4 = ...
5	SM_5 = ...	RM_5 = ...
6	SM_6 = ...	RM_6 = ...
7	SM_7 = ...	RM_7 = ...
8	SM_8 = ...	RM_8 = ...
9	SM_9 = ...	RM_9 = ...

Actions	Q_1 = ...	Q_4 = ...
	Q_2 = ...	Q_5 = ...
	Q_3 = ...	Q_6 = ...

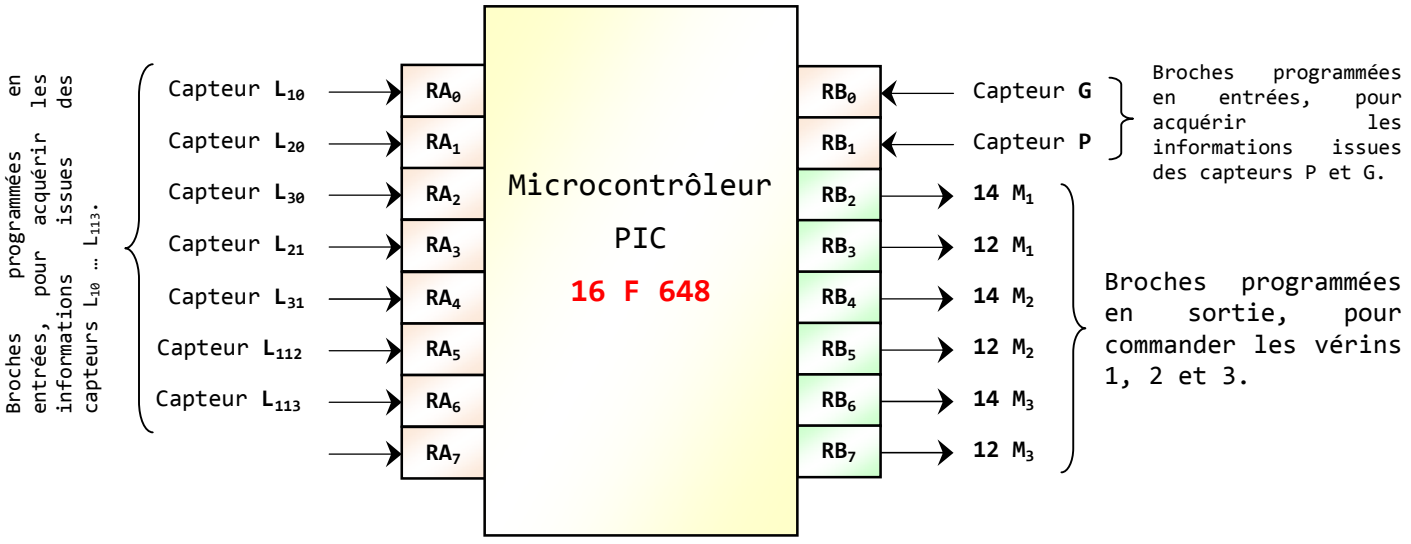
4.4. Programme LADDER :



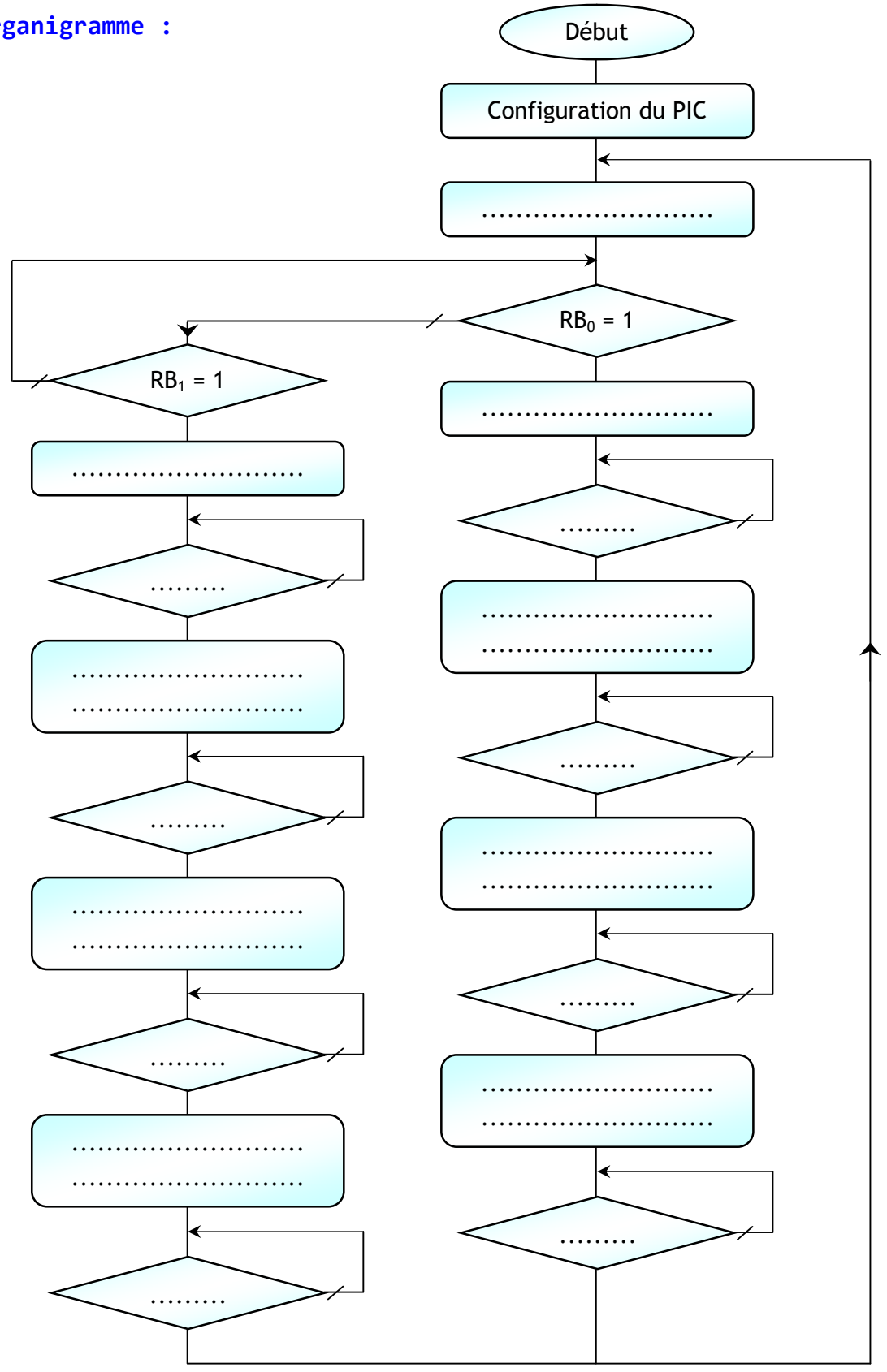


5. Commande par le PIC 16 F 648 :

5.1. Affectation des entrées/sorties :



5.2. Organigramme :



5.3. Programme Assembleur :

Programme de configuration

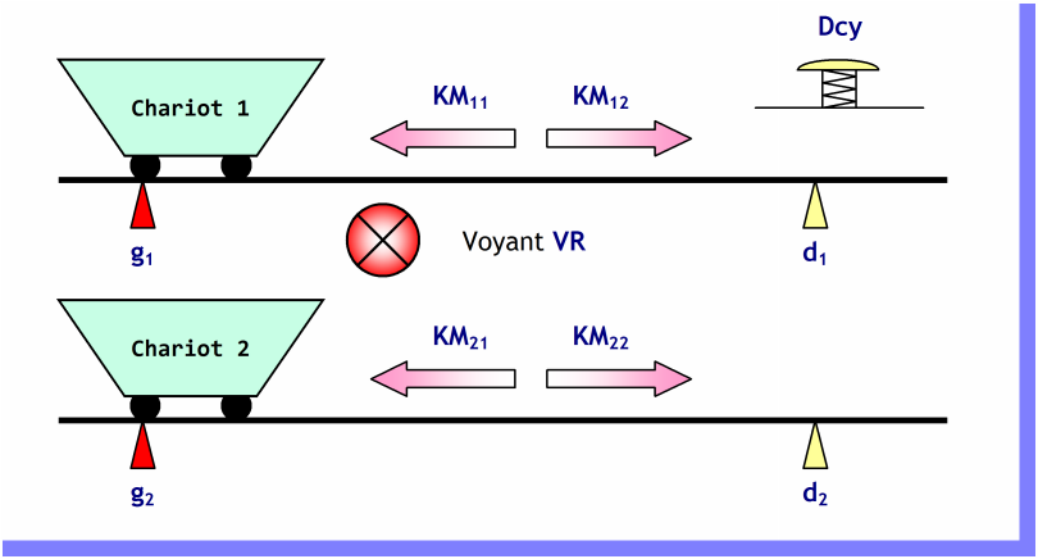
```
.....      ;Accès à la BANK 1
.....      ;
.....      ;PORTA en entrée
.....      ;
```

```
                ..................          ; configuration du PORTB
                BCF     STATUS, 5           ; Accès à la BANK 0
```

Programme principal

```
Lab1            ..................          ; Initialisation des préactionneurs à 0
Lab2            ..................          ; Grande caisse ?
                GOTO    Lab3                ;
                GOTO    G_c                 ;
Lab3            ..................          ; Petite caisse ?
                ..................          ;
P_c             ..................          ; Avance  du poussoir 1
Lab4            ..................          ; Caisse devant P2 ?
                ..................          ;
                ..................          ; Pas d'avance du poussoir 1
                ..................          ; Recule du poussoir 1
Lab5            ..................          ; P1 en arrière ?
                ..................          ;
                ..................          ; Pas de Recule du poussoir 1
                ..................          ; Avance du poussoir 2
Lab6            ..................          ; Caisse sur tapis 2 ?
                ..................          ;
                ..................          ; Pas d'avance du poussoir 2
                ..................          ; Recule du poussoir 2
Lab7            ..................          ; P2 en arrière ?
                ..................          ;
                GOTO    Lab1                ; Reprendre
G_c             ..................          ; Avance  du poussoir 1
Lab8            ..................          ; Caisse devant P3
                ..................          ;
                ..................          ; Pas d'avance du poussoir 1
                ..................          ; Recule du poussoir 1
Lab9            ..................          ; P1 en arrière ?
                ..................          ;
                ..................          ; Pas de Recule du poussoir 1
                ..................          ; Avance du poussoir 3
Lab10           ..................          ; Caisse sur tapis 3 ?
                ..................          ;
                ..................          ; Pas d'avance du poussoir 3
                ..................          ; Recule du poussoir 3
Lab11           ..................          ; P3 en arrière ?
                GOTO    Lab11               ;
                GOTO    Lab1                ; Reprendre
```

1. Cahier des charges :



- Après appui sur départ cycle « dcy », les chariots partent pour un aller retour.
- Quand les deux chariots sont en position gauche, un voyant rouge s'allume pendant un temps T_1 de 20 s ; ainsi on peut lancer un nouveau départ cycle.
- La commande du système est réalisée par l'A.P.I ZELIO SR3101BD.

2. Identification des entrées et des sorties :

Mouvement	Actionneur	Ordres	Sortie API
Déplacer le chariot 1 à droite	Moteur M_1	KM_{12}	Q_1
Déplacer le chariot 1 à gauche	Moteur M_1	KM_{11}	Q_2
Déplacer le chariot 2 à droite	Moteur M_2	KM_{22}	Q_3
Déplacer le chariot 2 à gauche	Moteur M_2	KM_{21}	Q_4

Message	Voyant	Mnem.	Sortie API
Allumer voyant rouge	Voyant rouge	VR	Q_5

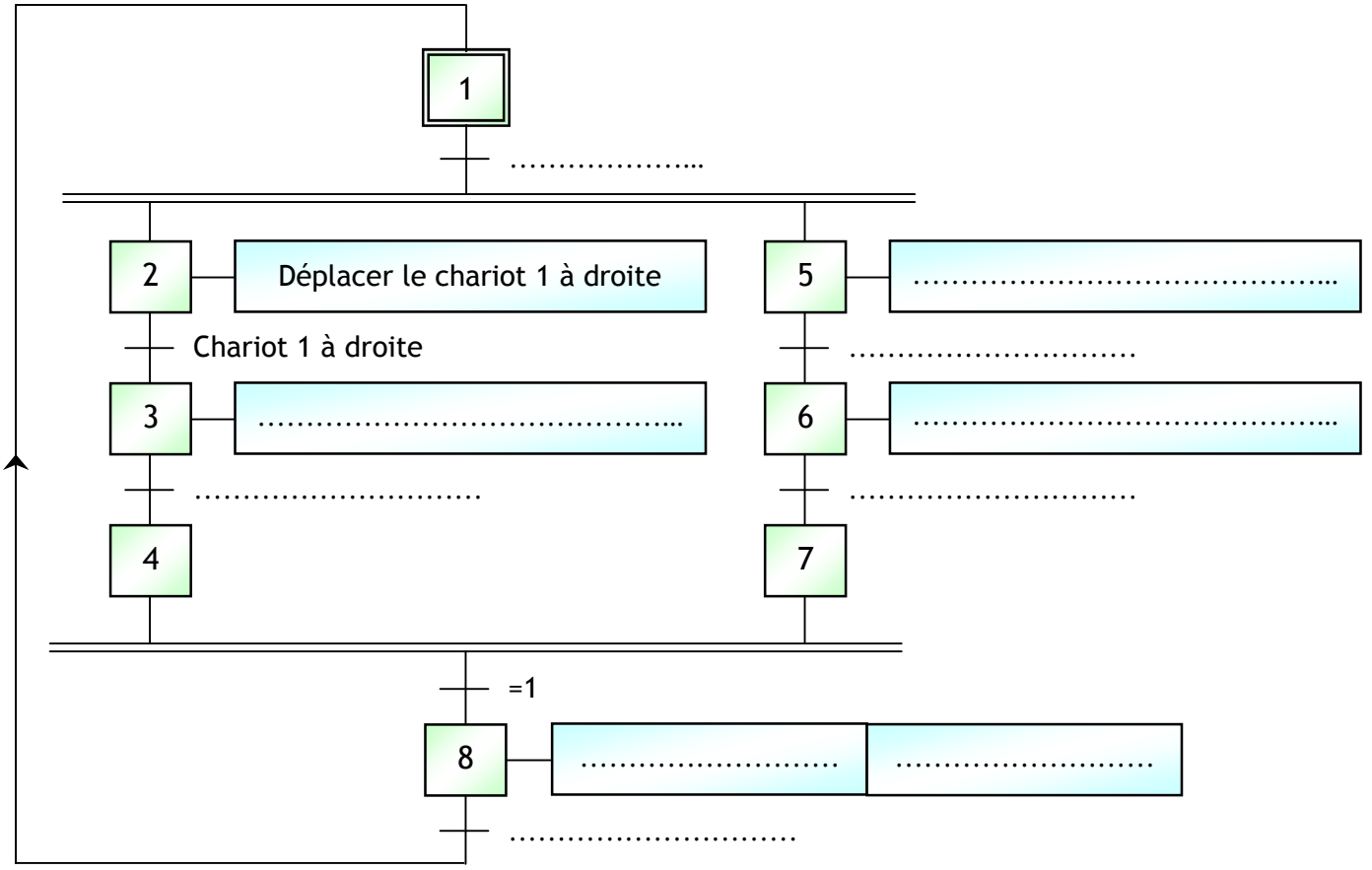
Compte-rendu	Capteur	Mnem.	Entrée API
Chariot 1 à gauche	Détecteur mécanique à levier	g_1	I_1
Chariot 1 à droite	Détecteur mécanique à levier	d_1	I_2
Chariot 2 à gauche	Détecteur mécanique à levier	g_2	I_B
Chariot 2 à droite	Détecteur mécanique à levier	d_2	I_C

Consigne	Constituant	Mnem.	Entrée API
Départ cycle	Bouton poussoir	Dcy	I_D

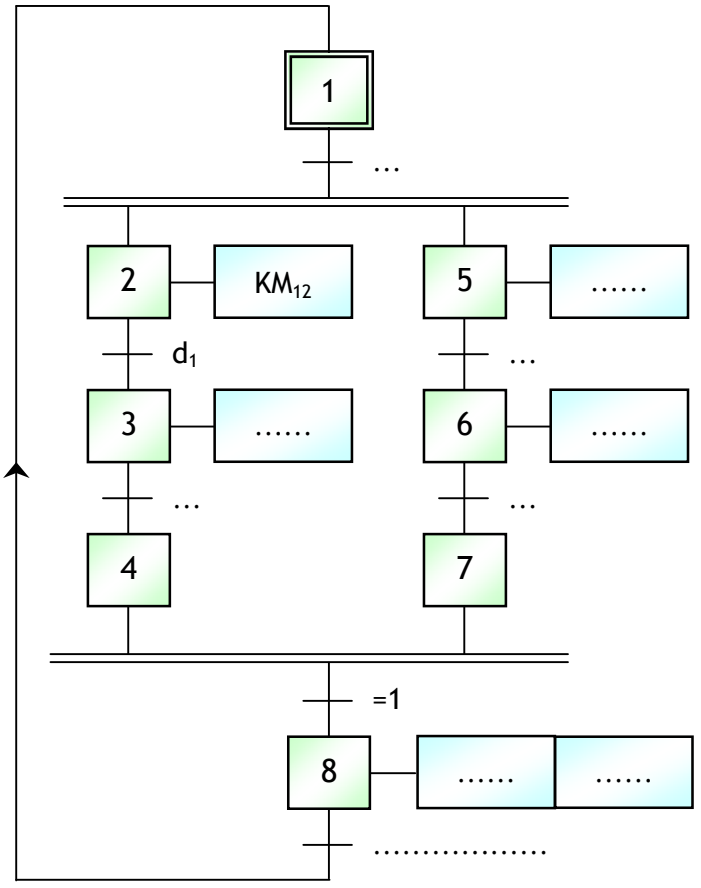
Temporisation	Constituant	Mnem.	API
Temporisation 20 s	Temporisateur T_1	T_1	TT_1

3. GRAFCET :

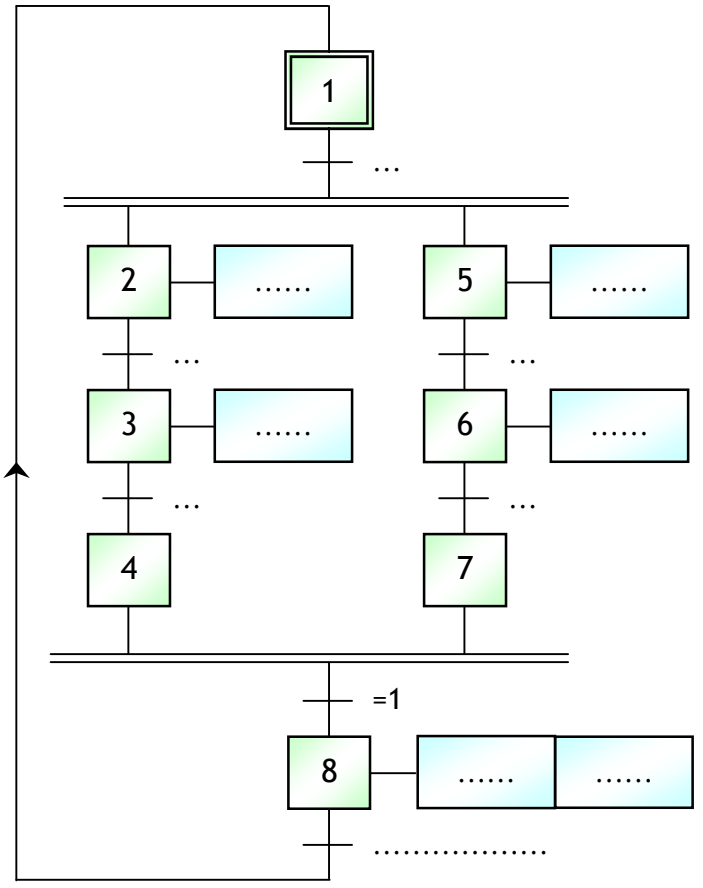
3.1. GRAFCET P.O :



3.2. GRAFCET P.C :



3.3. GRAFCET A.P.I :



4. Programme LADDER :

4.1. Mise en équation :

Étapes	Activation	Désactivation
1	SM_1 = Init + ...	RM_1 = ...
2	SM_2 = ...	RM_2 = ...
3	SM_3 = ...	RM_3 = ...
4	SM_4 = ...	RM_4 = ...
5	SM_5 = ...	RM_5 = ...
6	SM_6 = ...	RM_6 = ...
7	SM_7 = ...	RM_7 = ...
8	SM_8 = ...	RM_8 = ...

Actions
Q_1 = ...
Q_2 = ...
Q_3 = ...
Q_4 = ...
Q_5 = ...
TT_1 = ...

Init = $\overline{M_2}.\overline{M_3}.\overline{M_4}.\overline{M_5}.\overline{M_6}.\overline{M_7}.\overline{M_8}$

4.2. Programme LADDER :