ROYAUME DU MAROC
MINISTERE DE L'EDUCATION NATIONALE
Académie de Casablanca
DÉLÉGATION DE MOHAMMEDIA
Lycée Technique Mohammedia

Matière :	Science de l'Ingénieur - **A.T.C** -	Pr.MAHBAB
Section :	Sciences et Technologies Électriques	Système n°5

CORRECTION

❖ Sujet :

PARKING AUTOMATIQUE

08 pages

❖ Exercices d'application:

- Fiche cours n°22 « *EEPROM DU 16F84* »

01 page

❖ 2 TD:

- TD n°8 « *Partition d'un GRAFCET et notion de Tâche* »
- TD n°9 « *Programmation des PLD* »

09 pages

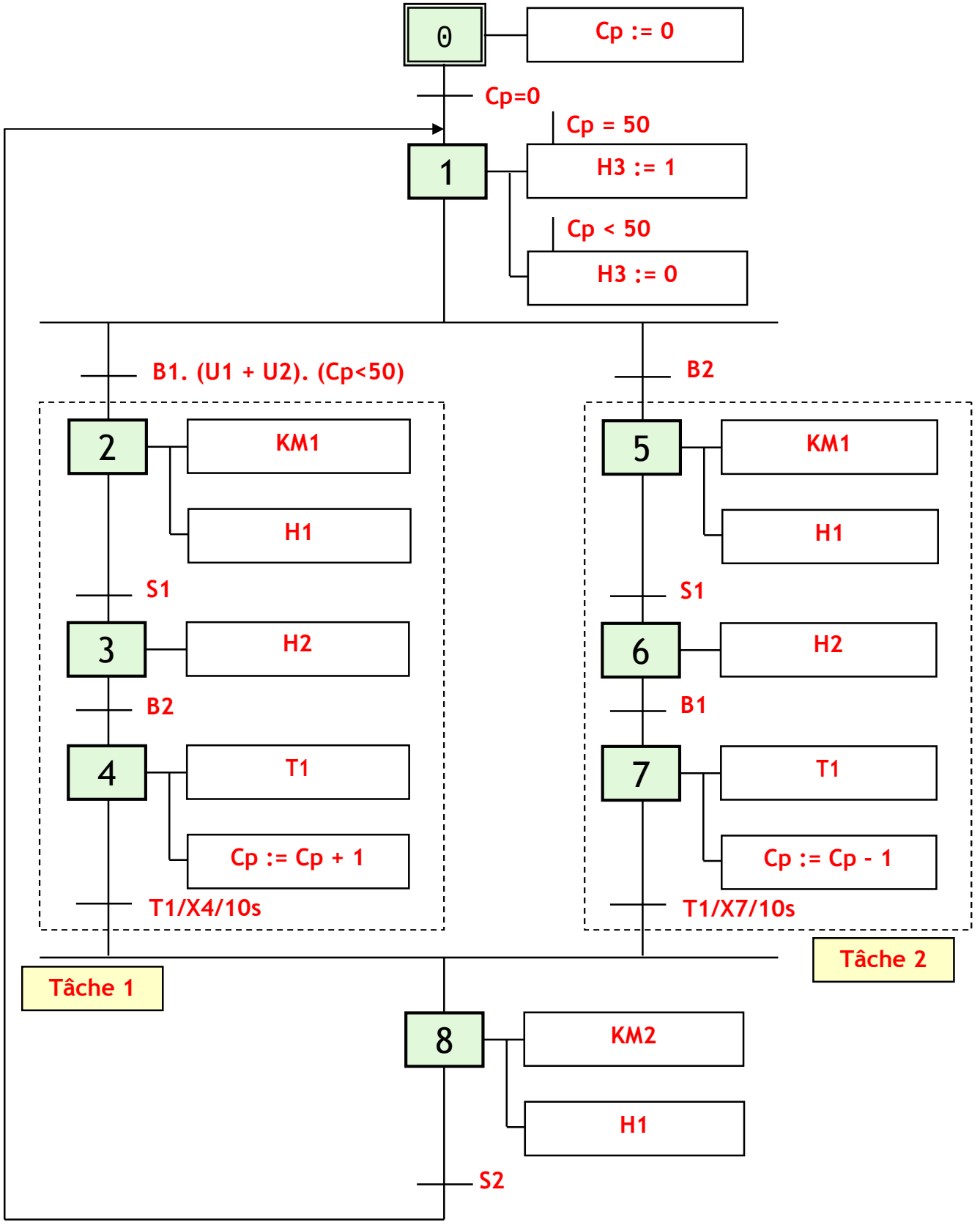
Nombre de pages : **19**

CLASSES : 2 STE

DREP 01

CORRECTION

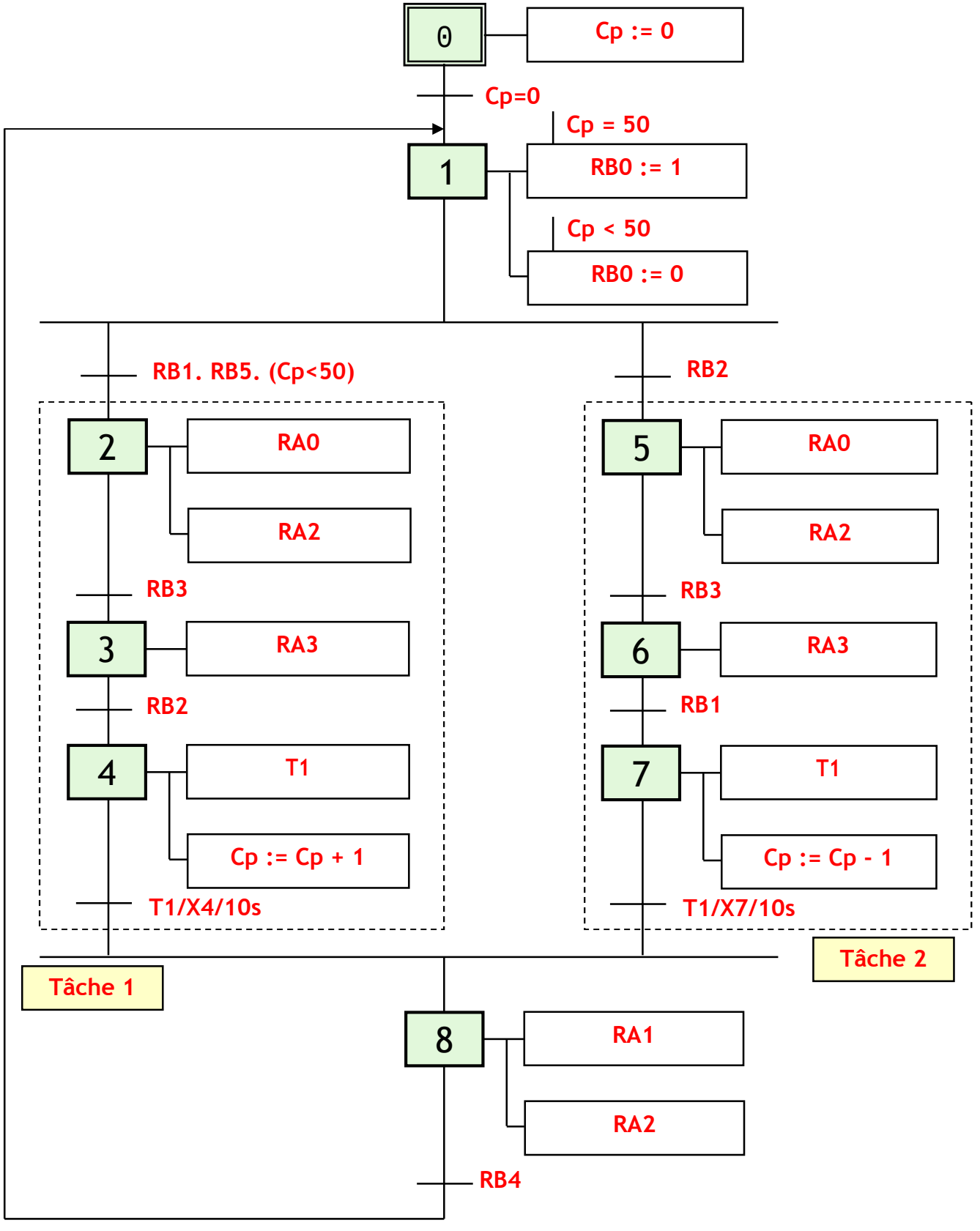
GRAFCET point de vue partie commande



DREP 02

CORRECTION

GRAFCET point de vue partie commande codé PIC 16 F 84

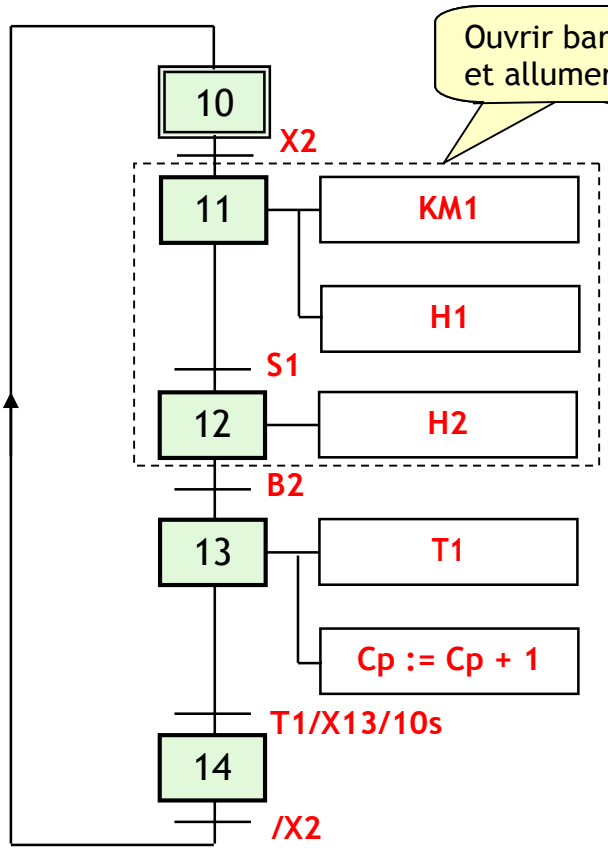


DREP 03

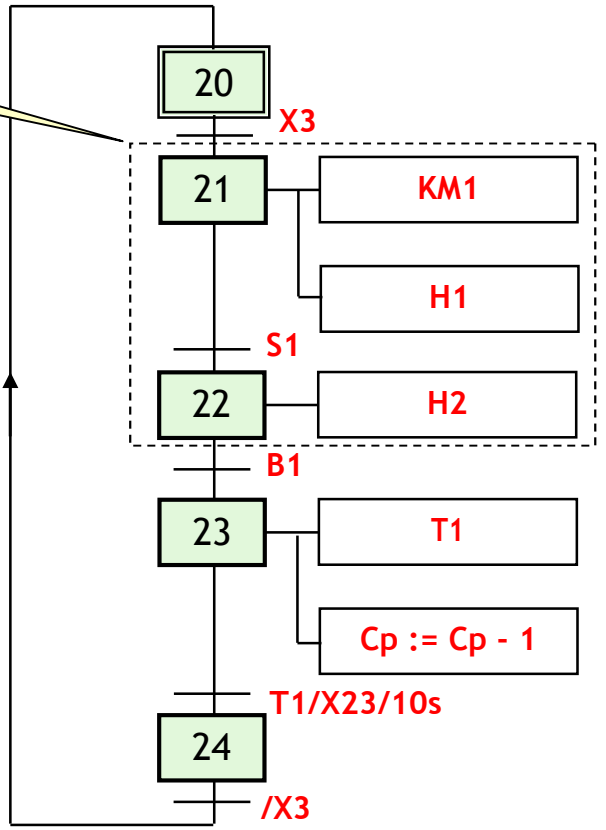
CORRECTION

Partition du GRAFCET global (P.C)

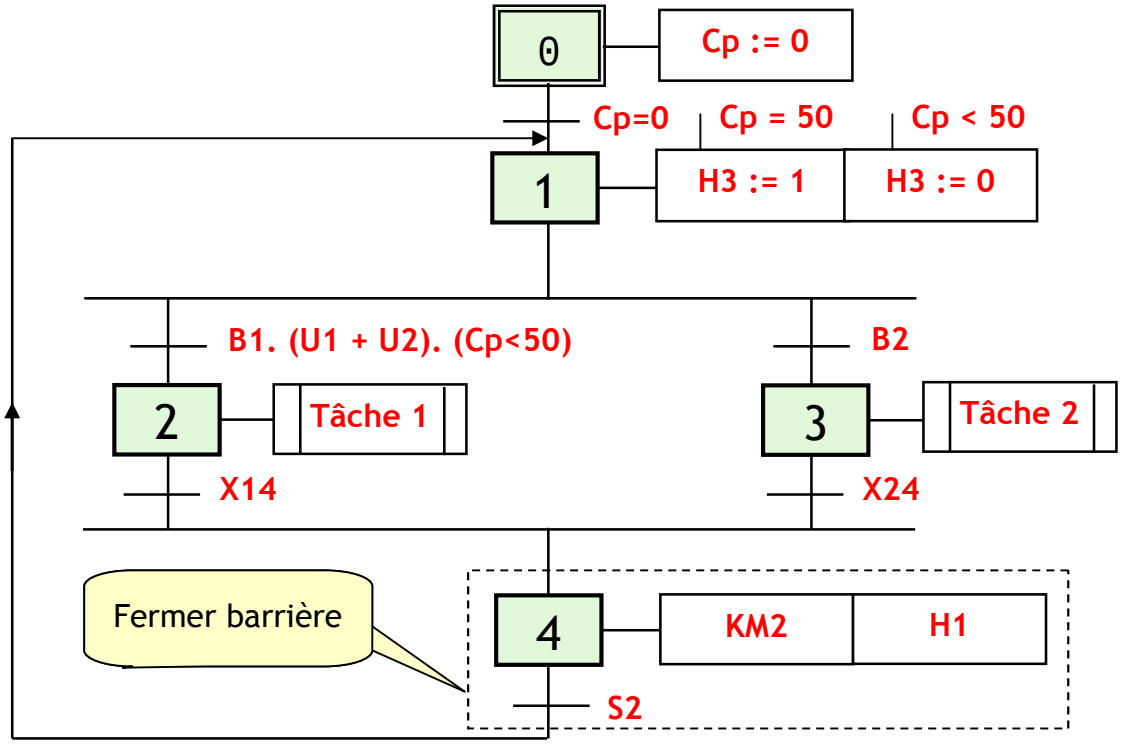
GRAFCET Tâche 1



GRAFCET Tâche 2



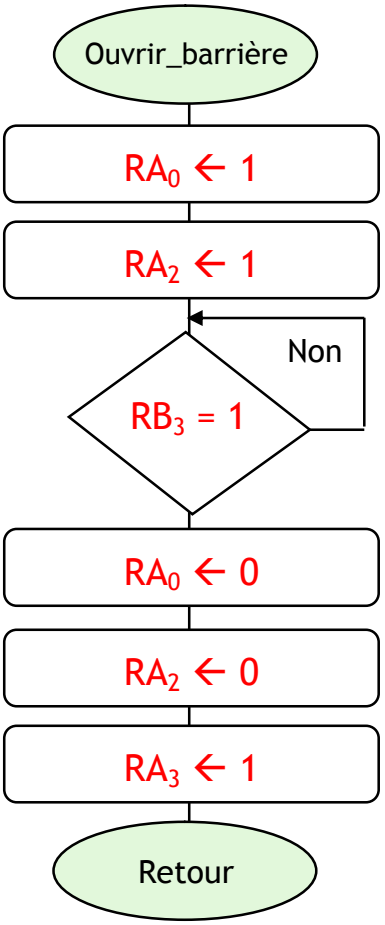
GRAFCET PRINCIPAL



DREP 04

CORRECTION

Sous programme ouvrir_Barrière

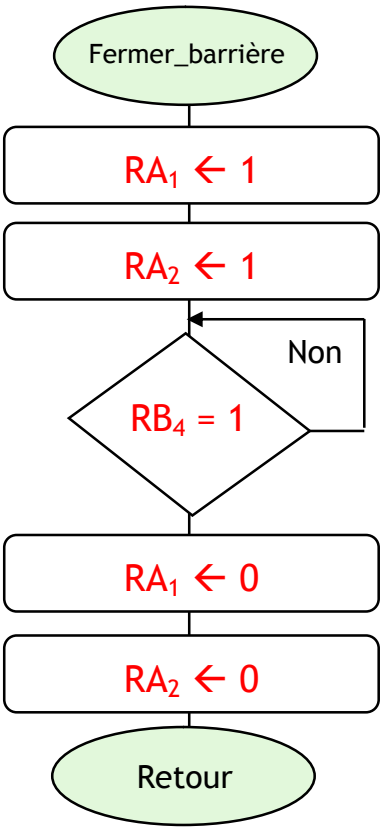


Organigramme

Programme

```
Ouvrir_barriere   BSF     PORTA, 0
                  BSF     PORTA, 2
LAB5              BTFSS   PORTB, 3
                  GOTO    LAB5
                  BCF     PORTA, 0
                  BCF     PORTA, 2
                  BSF     PORTA, 3
                  RETURN
```

Sous programme fermer_Barrière



Organigramme

Programme

```
Fermer_barriere   BSF     PORTA, 1
                  BSF     PORTA, 2
LAB6              BTFSS   PORTB, 4
                  GOTO    LAB6
                  BCF     PORTA, 1
                  BCF     PORTA, 2
                  RETURN
```

DREP 05

CORRECTION

Sous programme Lecture_EEPROM

```
Lecture_EEPROM  MOVLW   0x02          ;
                MOVWF   EEADR         ; l'adresse à lire
                BSF     STATUS, RP0   ; Bank 1
                BSF     EECON1, RD    ; lecture EPROM
                BCF     STATUS, RP0   ; Bank 0
                MOVF    EEDATA, W     ; EEDATA dans W
                MOVWF   Cp            ; W dans Cp
                RETURN                ;
```

Sous programme Ecriture_EEPROM

```
Ecriture_EEPROM MOVLW   0x02          ;
                MOVWF   EEADR         ; définition de l'adresse
                MOVF    Cp, W         ;
                MOVWF   EEDATA        ; définition de la donnée
                BSF     STATUS, RP0   ; Bank 1
                BSF     EECON1, WREN  ; autorisation de l'écriture
                MOVLW   0x55          ;
                MOVWF   EECON2        ; écriture de 0x55
                MOVLW   0xAA          ;
                MOVWF   EECON2        ; écriture de 0xAA
                BSF     EECON1, WR    ; écriture dans EEPROM
LAB7            BTFSS   EECON, EEIF   ;
                GOTO    LAB7          ; écriture terminée
                BCF     EECON, EEIF   ; remise à zéro du témoin EEIF
                BCF     STATUS, RP0   ; Bank 0
                RETURN                ;
```

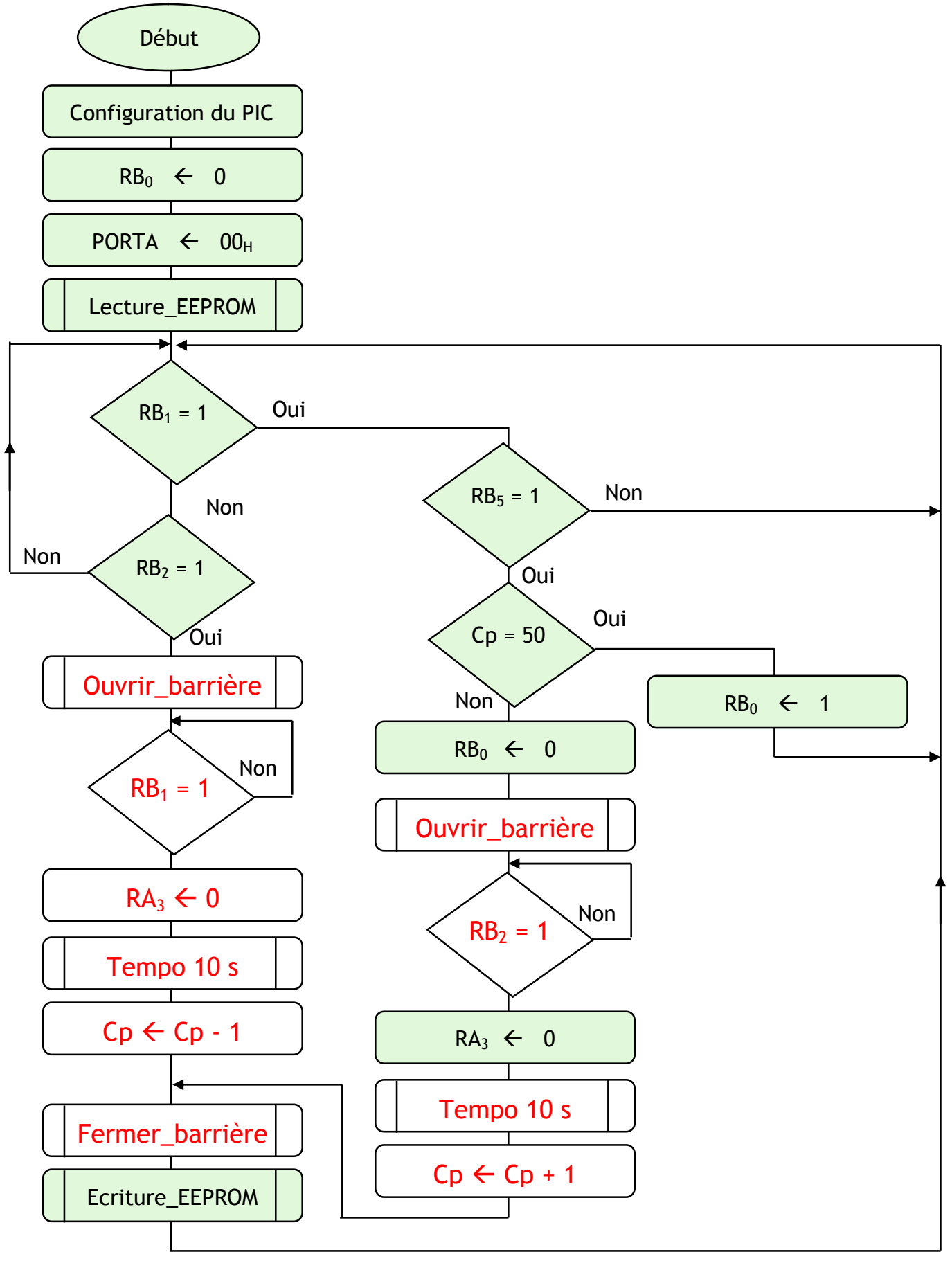
Sous programme TEMPO

```
TEMPO   MOVLW   D'100'        ; doit compter 156 impulsions
        MOVWF   TMR0          ; écriture de 100
LAB8    BTFSS   INTCON, TOIF  ; fin de comptage
        GOTO    LAB8          ;
        BCF     INTCON, TOIF  ; remise à zéro du témoin TOIF
        RETURN                ;
```

DREP 06

CORRECTION

Organigramme à compléter



DREP 07

CORRECTION

Programme à compléter

Initialisation du PIC 16 F 84

```
        BSF     STATUS, 5       ; accès à la BANK 1
        MOVLW   B'00001000'     ;
        MOVWF   TRISA           ; configuration du PORTA
        MOVLW   B'00111110'     ;
        MOVWF   TRISB           ; configuration du PORTB
        MOVLW   B'00110101'     ;
        MOVWF   OPTION          ; configuration du TIMER 0
        BCF     STATUS, 5       ; accès à la BANK 0
```

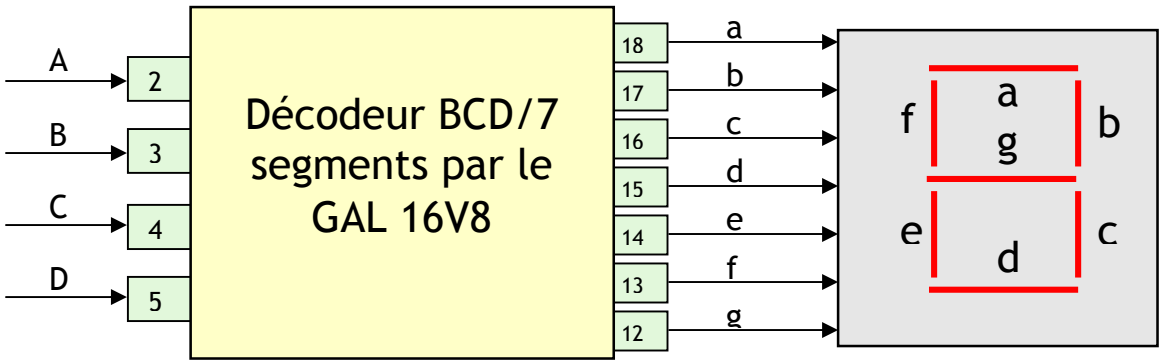
Programme principal

```
        CLRF    PORTA           ; moteur et voyants au repos
        CALL    Lecture_EPROM   ; nombre de véhicule dans Cp
DEBUT   BTFSS   PORTB, 1        ; test si entrée
        GOTO    SORTIE          ;
        GOTO    ENTRÉE          ;
SORTIE  BTFSS   PORTB, 2        ; test si sortie
        GOTO    DEBUT           ;
        CALL    Ouvrir_barriere ; ouvrir la barrière
LAB1    BTFSS   PORTB, 1        ; test si véhicule est sortie
        GOTO    LAB1            ;
        BCF     PORTA, 3        ; éteindre vert
        CALL    TEMPO           ; Temporisation T1
        DECF    Cp, 1           ; décrémenter compteur
        GOTO    LAB4            ;
ENTREE  BTFSS   PORTB, 5        ; test si autorisation
        GOTO    DEBUT           ;
        MOWLW   D'50'           ;
        SUBWF   Cp, W           ; comparer Cp et 10
        BTFSS   STATUS, Z       ; saut si Cp = 10
        GOTO    LAB2            ; parking disponible
        BSF     PORTB, 0        ; allumer jaune
        GOTO    DEBUT           ; parking complet
LAB2    BCF     PORTB, 0        ; éteindre jaune
        CALL    Ouvrir_barriere ; ouvrir la barrière
LAB3    BTFSS   PORTB, 2        ; test si véhicule est entré
        GOTO    LAB3            ;
        BCF     PORTA, 3        ; éteindre vert
        CALL    TEMPO           ; Temporisation T1
        INCF    Cp, 1           ; incrémenter compteur
LAB4    CALL    Fermer_barriere ; fermer la barrière
        CALL    Ecriture_EPROM  ; enregistrer compteur
        GOTO    DEBUT           ;
        END                     ; directive de fin de programme
```

DREP 08

CORRECTION

Décodeur BCD/7 segments par le GAL 16V8



Programme 'ABEL'

```
MODULE BCD
TITLE 'Décodeur BCD 7 segments '
DECLARATIONS
        BCD  device 'P16V8;                          // PLD à programmer
        A, B, C, D pin   2, 3, 4, 5 ;                // Variables d'entrée
        a, b, c, d, e, f, g  pin 18, 17, 16, 15, 14, 13, 12
        istype 'com,buffert' ;                       // Variables sortie
        N = [D, C, B, A] ;                           // Définition du bus N
        S = [a, b, c, d, e, f, g] ;                  // Définition du bus S
EQUATIONS
        When N = 0 THEN S = [1, 1, 1, 1, 1, 1, 0] ELSE
        When N = 1 THEN S = [0, 1, 1, 0, 0, 0, 0] ELSE
        When N = 2 THEN S = [1, 1, 0, 1, 1, 0, 1] ELSE
        When N = 3 THEN S = [1, 1, 1, 1, 0, 0, 1] ELSE
        When N = 4 THEN S = [0, 1, 1, 0, 0, 1, 1] ELSE
        When N = 5 THEN S = [1, 0, 1, 1, 0, 1, 1] ELSE
        When N = 6 THEN S = [1, 0, 1, 1, 1, 1, 1] ELSE
        When N = 7 THEN S = [1, 1, 1, 0, 0, 0, 0] ELSE
        When N = 8 THEN S = [1, 1, 1, 1, 1, 1, 1] ELSE
        When N = 9 THEN S = [1, 1, 1, 1, 0, 1, 1];
TEST_VECTORS    (N -> [a, b, c, d, e, f, g])
                0 -> [1, 1, 1, 1, 1, 1, 0] ;
                1 -> [0, 1, 1, 0, 0, 0, 0] ;
                2 -> [1, 1, 0, 1, 1, 0, 1] ;
                3 -> [1, 1, 1, 1, 0, 0, 1] ;
                4 -> [0, 1, 1, 0, 0, 1, 1] ;
                5 -> [1, 0, 1, 1, 0, 1, 1] ;
                6 -> [1, 0, 1, 1, 1, 1, 1] ;
                7 -> [1, 1, 1, 0, 0, 0, 0] ;
                8 -> [1, 1, 1, 1, 1, 1, 1] ;
                9 -> [1, 1, 1, 1, 0, 1, 1] ;
END BCD.
```

EEPROM DU 16 F 84

Ecriture et lecture de l'EEPROM :

Lecture d'une donnée :

- Placer l'**adresse** de la donnée à lire dans **EEADR**.
- Mettre le bit **RD** de EECONN1 à **1**.
- Lire le contenu du registre **EEDATA.**

```
        BCF     STATUS, RP0     ; Bank 0
        MOVLW   Adresse         ;
        MOVWF   EEADR           ; l'adresse à lire
        BSF     STATUS, RP0     ; Bank 1
        BSF     EECON1, RD      ; lecture EPROM
        BCF     STATUS, RP0     ; Bank 0
        MOVF    EEDATA, W       ; W ← EEDATA
```

Ecriture d'une donnée :

- Placer l'adresse de la donnée à écrire dans EEADR.
- Placer la donnée à écrire dans EEDATA.
- Mettre le bit WREN de EECONN1 à 1 pour autoriser l'écriture.
- Placer 0x55 dans EECON2.
- Placer 0xAA dans EECON2.
- Mettre le bit WR de EECONN1 à 1.
- Attendre que le bit EEIF soit à 1.
- On peut utiliser l'interruption produite par EEIF en la validant par le bit EEIE de INTCON.
- N'oublier pas de remettre EEIF à 0.

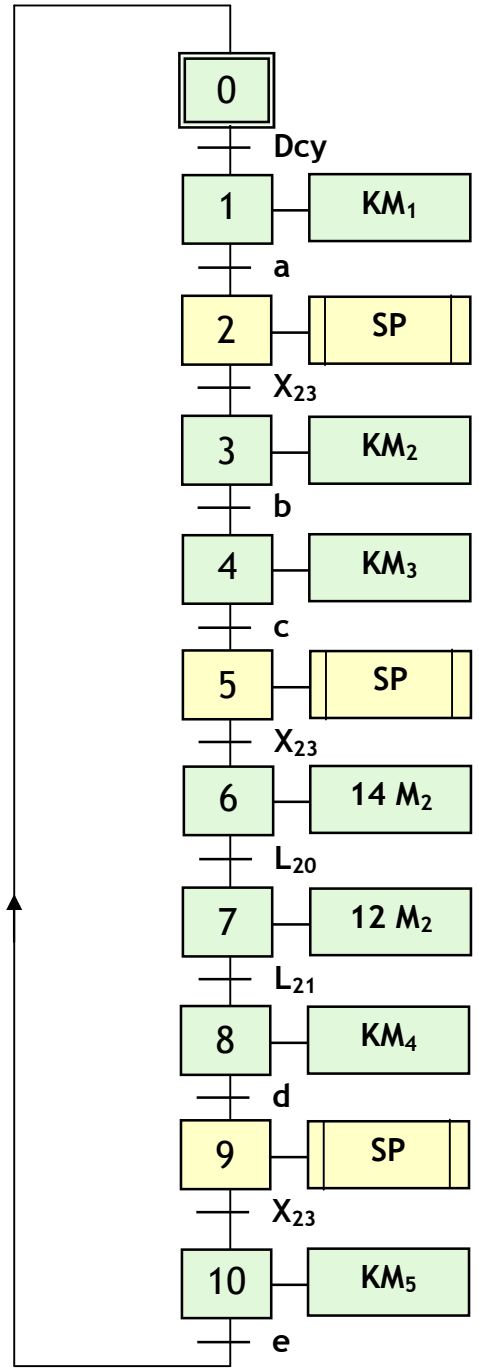
```
        BCF     STATUS, RP0     ; Bank 0
        MOVLW   Adresse         ;
        MOVWF   EEADR           ; définition de l'adresse
        MOVLW   Donnée          ;
        MOVWF   EEDATA          ; définition de la donnée
        BSF     STATUS, RP0     ; Bank 1
        BSF     EECON1, WREN    ; autorisation de l'écriture
        MOVLW   0x55            ;
        MOVWF   EECON2          ; écriture de 0x55
        MOVLW   0xAA            ;
        MOVWF   EECON2          ; écriture de 0xAA
        BSF     EECON1, WR      ; écriture dans EEPROM
Lab     BTFSS   EECON, EEIF     ;
        GOTO    Lab             ; écriture terminée
        BCF     STATUS, RP0     ; Bank 0
```

PARTITION D'UN GRAFCET

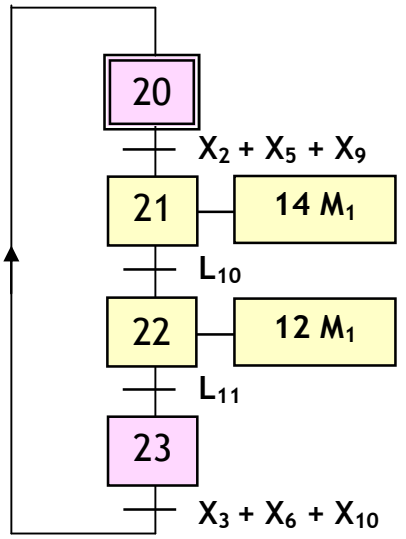
Coordination Verticale asynchrone :

GRAFCET n°1 :

Le GRAFCET n°1 contient une séquence répétitive. On veut réécrire ce GRAFCET d'une manière simplifiée en considérant la séquence répétitive comme une tâche (sous programme) intitulée SP. Compléter le GRAFCET **principal** et le GRAFCET de tâche **SP**.



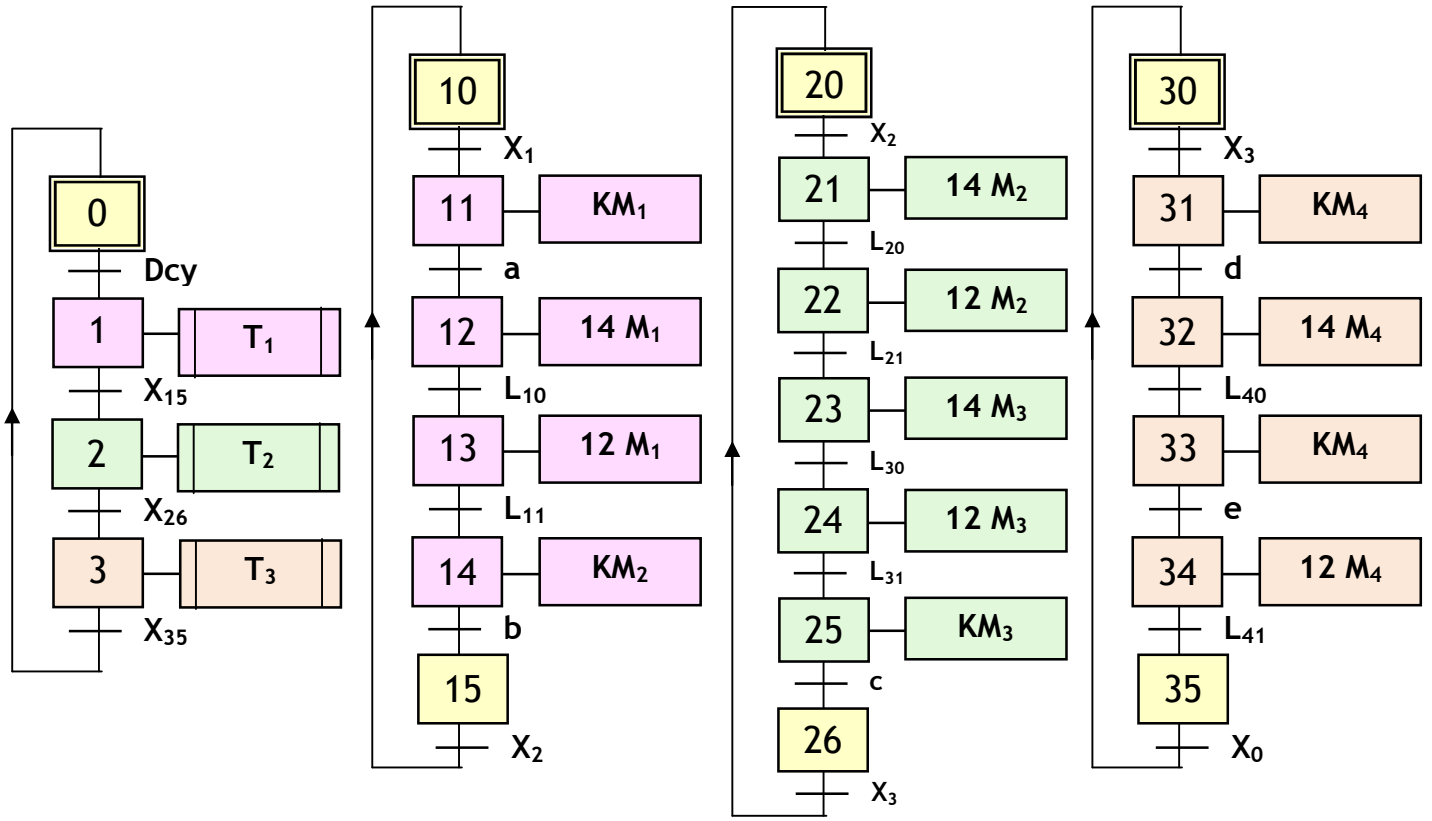
GRAFCET principal



GRAFCET de tâche SP

GRAFCET n°2 :

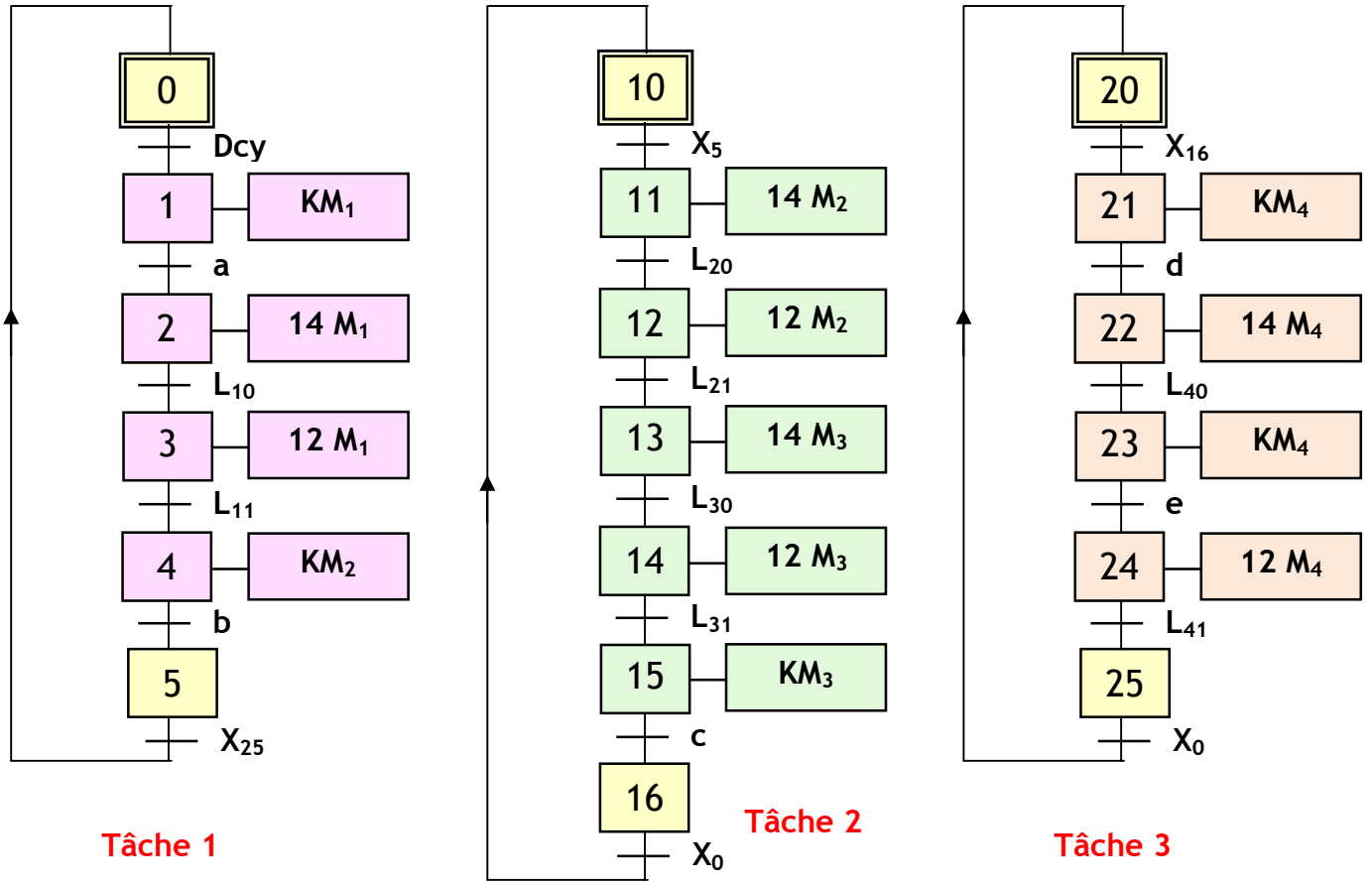
On veut réécrire le GRAFCET n°2 d'une manière simplifiée. Compléter le GRAFCET **principal** et les GRAFCETS de tâche **1, 2 et 3**.



GRAFCET principal **Tâche 1** **Tâche 2** **Tâche 3**

Coordination Horizontale :

On veut réécrire ce GRAFCET n°2 d'une manière simplifiée. Compléter les GRAFCETS **Tâche 1, 2 et 3**.



Tâche 1 **Tâche 2** **Tâche 3**

PROGRAMMATION DES PLD

1- Comparateur binaire 4 bits :

Description par équations :

```
MODULE comparateur
TITLE 'comparateur logique 4 bits'
DECLARATIONS
        Comparateur  device 'P16V8';                    // PLD à programmer
        B3, B2, B1, B0 pin 9, 8, 7, 6 ;                 // Variables d'entrée
        A3, A2, A1, A0 pin 5, 4, 3, 2 ;                 // Variables d'entrée
        A = [A3, A2, A1, A0] ;                          // Définition du bus A
        B = [B3, B2, B1, B0] ;                          // Définition du bus B
        S, I, E pin 15, 16, 17 istype 'com, buffer ';   // Variables de sortie
EQUATIONS
        S = A > B     ;
        E = A = = B   ;
        I = A < B     ;
TEST_VECTORS   ([A, B] -> [S, E, I]);
                        [5, 5]   -> [0, 1, 0];
                        [9, 12] -> [0, 0, 1];
                        [7, 6]   -> [1, 0, 0];
END comparateur
```

Utilisation de **When …. Then ….**

```
EQUATIONS
        When A > B then S = 1     ;
        When A = = Bthen E = 1    ;
        When A < B then I =1      ;
```

2- Multiplexeur 4 bits :

Description par équations :

A- Solution 1 :

```
MODULE Multiplexeur
TITLE 'Multiplexeur 4 bits'
DECLARATIONS
        Multiplexeur  device 'P16V8';          // PLD à programmer
        E3, E2, E1, E0 pin 9, 8, 7, 6 ;        // Variables d'entrée
        A1, A0 pin 5, 4;                       // Variables d'entrée
        S pin 16 istype 'com, buffer ';        // Variables de sortie
EQUATIONS
        S = E3 & A1 & A0 # E2 & A1 &! A0 # E1 &! A1 & A0 # E0 &! A1 &! A0 ;
TEST_VECTORS   ([E3, E2, E1, E0, A1, A0] -> [S]);
                        [0, 0, 0, 0, 0, 0] -> [0];
                        [0, 0, 0, 1, 0, 0] -> [1];
                        [0, 0, 0, 0, 0, 1] -> [0];
                        [0, 0, 1, 0, 0, 1] -> [1];
                        [0, 0, 0, 0, 1, 0] -> [0];
                        [0, 1, 0, 0, 1, 0] -> [1];
                        [0, 0, 0, 0, 1, 1] -> [0];
                        [1, 0, 0, 0, 1, 1] -> [1];
END Multiplexeur.
```

B- Solution 2 :

```
MODULE Multiplexeur
TITLE 'Multiplexeur 4 bits'
DECLARATIONS
        Multiplexeur  device 'P16V8';                  // PLD à programmer
        E3, E2, E1, E0, A1, A0 pin 9, 8, 7, 6, 5, 4 ;  // Variables d'entrée
        A = [A1, A0] ;                                 // Définition du bus A
        E = [E3, E2, E1, E0] ;                         // Définition du bus E
        S pin 16  istype 'com, buffer ';               // Variables de sortie
EQUATIONS
        When A = [0, 0] then S = E0; // ou When A = 0 then S = E0;
        When A = [0, 1] then S = E1; // ou When A = 1 then S = E1;
        When A = [1, 0] then S = E2; // ou When A = 2 then S = E2;
        When A = [1, 1] then S = E3; // ou When A = 3 then S = E3;

TEST_VECTORS   ([E3, E2, E1, E0, A1, A0] -> [S]); // ou ([E, A] -> [S]);
                 [0, 0, 0, 0, 0, 0] -> [0];       // ou   [0, 0] -> [0];
                 [0, 0, 0, 1, 0, 0] -> [1];       // ou   [1, 0] -> [1];
                 [0, 0, 0, 0, 0, 1] -> [0];       // ou   [0, 1] -> [0];
                 [0, 0, 1, 0, 0, 1] -> [1];       // ou   [2, 1] -> [1];
                 [0, 0, 0, 0, 1, 0] -> [0];       // ou   [0, 2] -> [0];
                 [0, 1, 0, 0, 1, 0] -> [1];       // ou   [4, 2] -> [1];
                 [0, 0, 0, 0, 1, 1] -> [0];       // ou   [0, 3] -> [0];
                 [1, 0, 0, 0, 1, 1] -> [1];       // ou   [8, 3] -> [1];
END Multiplexeur.
```

3- Démultiplexeur 4 bits :

3.1- Description par équations :

A- Solution 1 :

```
MODULE Demultiplexeur
TITLE 'Démultiplexeur 4 bits'
DECLARATIONS
        Demultiplexeur device 'P16V8';                          // PLD à programmer
        E, G, A1, A0 pin 7, 6, 5, 4;                            // Variables d'entrée
    S3, S2, S1, S0 pin 12, 13, 14, 15 istype 'com, buffer '; // Variables de sortie
EQUATIONS
        S3 = E & A1 & A0 &! G;
        S2 = E & A1 &! A0 &! G;
        S1 = E &! A1 & A0 &! G;
        S0 = E &! A1 &! A0 &! G;
TEST_VECTORS   ([G, E, A1, A0]      -> [S3, S2, S1, S0]);
                 [1,.X.,.X.,.X.]    -> [0, 0, 0, 0];
                 [0, 0, 0, 0]       -> [0, 0, 0, 0];
                 [0, 1, 0, 0]       -> [0, 0, 0, 1];
                 [0, 0, 0, 1]       -> [0, 0, 0, 0];
                 [0, 1, 0, 1]       -> [0, 0, 1, 0];
                 [0, 0, 1, 0]       -> [0, 0, 0, 0];
                 [0, 1, 1, 0]       -> [0, 1, 0, 0];
                 [0, 0, 1, 1]       -> [0, 0, 0, 0];
```

```
                    [0, 1, 1, 1]        -> [1, 0, 0, 0];
END Demultiplexeur.
```

B- Solution 2 :

```
MODULE Demultiplexeur
TITLE 'Demultiplexeur 4 bits'
DECLARATIONS
        Demultiplexeur  device 'P16V8';                          // PLD à programmer
        E, G, A1, A0 pin 7, 6, 5, 4;                             // Variables d'entrée
        S3, S2, S1, S0 pin 12, 13, 14, 15 istype 'com, buffer '; // Variables de sortie
        A = [A1, A0] ;                                           // Définition du bus A
        S = [S3, S2, S1, S0] ;                                   // Définition du bus S
EQUATIONS
        When (G == 0) & (A == 0) then S0 = E;
        When (G == 0) & (A == 1) then S1 = E;
        When (G == 0) & (A == 2) then S2 = E;
        When (G == 0) & (A == 3) then S3 = E;
TEST_VECTORS   ([G, E, A]       -> [S]);
                [1,.X.,.X.]     -> [0];
                [0, 0, 0]       -> [0];
                [0, 1, 0]       -> [1];
                [0, 0, 1]       -> [0];
                [0, 1, 1]       -> [2];
                [0, 0, 2]       -> [0];
                [0, 1, 2]       -> [4];
                [0, 0, 3]       -> [0];
                [0, 1, 3]       -> [8];
END Demultiplexeur.
```

3.2- Description par table de vérité :

```
MODULE Demultiplexeur
TITLE 'Demultiplexeur 4 bits'
DECLARATIONS
        Demultiplexeur device 'P16V8';                           // PLD à programmer
        E, G, A1, A0 pin 7, 6, 5, 4;                             // Variables d'entrée
        S3, S2, S1, S0 pin 12, 13, 14, 15 istype 'com, buffer '; // Variables de sortie
TRUTH_TABLE       ([G, E, A1, A0] -> [S3, S2, S1, S0]);
                [1,.X.,.X.,.X.] -> [0, 0, 0, 0];
                [0, 0, 0, 0]    -> [0, 0, 0, 0];
                [0, 0, 0, 1]    -> [0, 0, 0, 0];
                [0, 0, 1, 0]    -> [0, 0, 0, 0];
                [0, 0, 1, 1]    -> [0, 0, 0, 0];
                [0, 1, 0, 0]    -> [0, 0, 0, 1];
                [0, 1, 0, 1]    -> [0, 0, 1, 0];
                [0, 1, 1, 0]    -> [0, 1, 0, 0];
                [0, 1, 1, 1]    -> [1, 0, 0, 0];
TEST_VECTORS   ([G, E, A1, A0]          -> [S3, S2, S1, S0]);
                [1,.X.,.X.,.X.] -> [0, 0, 0, 0];
                [0, 0, 0, 0]    -> [0, 0, 0, 0];
                [0, 0, 0, 1]    -> [0, 0, 0, 0];
                [0, 0, 1, 0]    -> [0, 0, 0, 0];
```

Autre solution

```
([G, A1, A0] -> [S3, S2, S1, S0]);
[1,.X.,.X.]     -> [0, 0, 0, 0];
[0, 0, 0]       -> [0, 0, 0, E];
[0, 0, 1]       -> [0, 0, E, 0];
[0, 1, 0]       -> [0, E, 0, 0];
[0, 1, 1]       -> [E, 0, 0, 0];
```

```
                [0, 0, 1, 1]   -> [0, 0, 0, 0];
                [0, 1, 0, 0]   -> [0, 0, 0, 1];
                [0, 1, 0, 1]   -> [0, 0, 1, 0];
                [0, 1, 1, 0]   -> [0, 1, 0, 0];
                [0, 1, 1, 1]   -> [1, 0, 0, 0];
END Demultiplexeur.
```

4- Décodeur BCD 7 segments :

Description par table de vérité :

```
MODULE BCD
TITLE 'Décodeur BCD 7 segments '
DECLARATIONS
        BCD  device 'P16V8;                                  // PLD à programmer
        A, B, C, D pin   2, 3, 4, 5 ;                        // Variables d'entrée
        a, b, c, d, e, f, g pin 18, 17, 16, 15, 14, 13, 12;  // Variables sortie
        N = [D, C, B, A] ;                                   // Définition du bus N
TRUTH_TABLE     (N -> [a, b, c, d, e, f, g])
                0 -> [1, 1, 1, 1, 1, 1, 0] ; // affichage du 0 dans le cas
                1 -> [0, 1, 1, 0, 0, 0, 0] ; // d'un afficheur cathode commune
                2 -> [1, 1, 0, 1, 1, 0, 1] ;
                3 -> [1, 1, 1, 1, 0, 0, 1] ;
                4 -> [0, 1, 1, 0, 0, 1, 1] ;
                5 -> [1, 0, 1, 1, 0, 1, 1] ;
                6 -> [1, 0, 1, 1, 1, 1, 1] ;
                7 -> [1, 1, 1, 0, 0, 0, 0] ;
                8 -> [1, 1, 1, 1, 1, 1, 1] ;
                9 -> [1, 1, 1, 1, 0, 1, 1] ;
TEST_VECTORS    (N -> [a, b, c, d, e, f, g])
                0 -> [1, 1, 1, 1, 1, 1, 0] ;
                1 -> [0, 1, 1, 0, 0, 0, 0] ;
                2 -> [1, 1, 0, 1, 1, 0, 1] ;
                3 -> [1, 1, 1, 1, 0, 0, 1] ;
                4 -> [0, 1, 1, 0, 0, 1, 1] ;
                5 -> [1, 0, 1, 1, 0, 1, 1] ;
                6 -> [1, 0, 1, 1, 1, 1, 1] ;
                7 -> [1, 1, 1, 0, 0, 0, 0] ;
                8 -> [1, 1, 1, 1, 1, 1, 1] ;
                9 -> [1, 1, 1, 1, 0, 1, 1] ;
END BCD.
```

5- Compteur modulo 4 :

5.1- Description par diagramme d'état :

```
MODULE Compteur
TITLE 'Compteur  modulo 4'
DECLARATIONS
        Compteur device 'P16V8' ;                    // PLD à programmer
        H, R pin  1, 2 ;                             // Variables d'entrée
        Q0, Q1 pin 15, 14 Istype 'reg, buffer ';     // Variables de sortie
EQUATIONS
```

```
        [Q1, Q0].clk = H ;
        [Q1, Q0].AR = ! R;
QSTATE      =       [Q1, Q0] ;
A           =       [0, 0] ;
B           =       [0, 1] ;
C           =       [1, 0];
D           =       [1, 1];
STATE_DIAGRAM   QSTATE
        State A:    GOTO B;
        State B:    GOTO C;
        State C:    GOTO D;
        State D:    GOTO A;
TEST_VECTORS    ([R, H,]        -> [Q1, Q0])
                 [0, .X.]       -> [0, 0] ;
                 [1, . C.]      -> [0, 1] ;
                 [1, . C.]      -> [1, 0] ;
                 [1, . C.]      -> [1, 1] ;
                 [1, . C.]      -> [0, 0] ;
END Compteur
```

5.2- Description par équation :

```
MODULE Compteur
TITLE 'Compteur  modulo 4'
DECLARATIONS
        Compteur device 'P16V8' ;                   // PLD à programmer
        H, R pin  1, 2 ;                            // Variables d'entrée
        Q0, Q1 pin 15, 14 Istype 'reg, buffer ';    // Variables de sortie
        Q = [Q1,Q0] ; // Définition du bus Q
EQUATIONS
        Q.clk = H ;
        Q.AR = ! R;
        When R == 0 then Q := 0 ELSE Q := Q + 1 ;

TEST_VECTORS    ([R, H,]        -> [Q1, Q0])
                 [0, .X.]       -> [0, 0] ;
                 [1, . C.]      -> [0, 1] ;
                 [1, . C.]      -> [1, 0] ;
                 [1, . C.]      -> [1, 1] ;
                 [1, . C.]      -> [0, 0] ;
END Compteur
```

5.3- Description par Table de vérité :

```
MODULE Compteur
TITLE 'Compteur  modulo 4'
        Compteur device 'P16V8'
DECLARATIONS
        Compteur device 'P16V8' ;                   // PLD à programmer
        H, R pin  1, 2 ;                            // Variables d'entrée
        Q0, Q1 pin 15, 14 Istype 'reg, buffer ';    // Variables de sortie
EQUATIONS
        [Q1, Q0].clk = H ;
```

```
        [Q1, Q0].AR = ! R;
TEST_VECTORS    ([R, H,]      :> [Q1, Q0])
                 [0, .X.]     :> [0, 0] ;
                 [1, . C.]    :> [0, 1] ;
                 [1, . C.]    :> [1, 0] ;
                 [1, . C.]    :> [1, 1] ;
                 [1, . C.]    :> [0, 0] ;
TEST_VECTORS    ([R, H,]      -> [Q1, Q0])
                 [0, .X.]     -> [0, 0] ;
                 [1, . C.]    -> [0, 1] ;
                 [1, . C.]    -> [1, 0] ;
                 [1, . C.]    -> [1, 1] ;
                 [1, . C.]    -> [0, 0] ;
END Compteur
```

6- Compteur modulo 5 réversible :

6.1- Description par diagramme d'état :

```
MODULE Compteur_reversible
TITLE 'Compteur  modulo 5'
DECLARATIONS
        Compteur device 'P16V8' ;                          // PLD à programmer
        H, R, Y pin  1, 2, 3 ;                             // Variables d'entrée
        Q0, Q1, Q2 pin 14, 15, 16 Istype 'reg, buffer '; // Variables sortie
EQUATIONS
        [Q2, Q1, Q0].clk = H ;
        [Q2, Q1, Q0].AR = ! R;
QSTATE      =       [Q2, Q1, Q0];
A           =       [0, 0, 0];
B           =       [0, 0, 1];
C           =       [0, 1, 0];
D           =       [0, 1, 1];
E           =       [0, 1, 1];
STATE_DIAGRAM   QSTATE
        State A:    When Y = = 1 Then B Else E;
        State B:    When Y = = 1 Then C Else A;
        State C:    When Y = = 1 Then D Else B;
        State D:    When Y = = 1 Then E Else C;
        State E:    When Y = = 1 Then A Else D;
TEST_VECTORS    ([R, H, Y]      -> [Q2, Q1, Q0])
                 [0, .X., .X.]  -> [0, 0, 0] ;
                 [1, .C., 1]    -> [0, 0, 1] ;
                 [1, .C., 1]    -> [0, 1, 0] ;
                 [1, .C., 1]    -> [0, 1, 1] ;
                 [1, .C., 1]    -> [1, 0, 0] ;
                 [1, .C., 1]    -> [0, 0, 0] ;
                 [1, .C., 0]    -> [1, 0, 0] ;
                 [1, .C., 0]    -> [0, 1, 1] ;
                 [1, .C., 0]    -> [0, 1, 0] ;
                 [1, .C., 0]    -> [0, 0, 1] ;
                 [1, .C., 0]    -> [0, 0, 0] ;
END Compteur_reversible
```

6.2- Description par table de vérité :

```
MODULE Compteur_reversible
TITLE 'Compteur  modulo 5'
DECLARATIONS
        Compteur device 'P16V8' ;                          // PLD à programmer
        H, R, Y pin  1, 2, 3 ;                             // Variables d'entrée
        Q0, Q1, Q2 pin 14, 15, 16 Istype 'reg, buffer ';   // Variables sortie
EQUATIONS
        [Q2, Q1, Q0].clk = H ;
        [Q2, Q1, Q0].AR = ! R;
TRUTH_TABLE      ([R, Y, Q2, Q1, Q0] :> [Q2, Q1, Q0])
                 [0, .X., .X., .X., .X.] :> [0, 0, 0] ;
                 [1, 1, 0, 0, 0]  :> [0, 0, 1] ;
                 [1, 1, 0, 0, 1]  :> [0, 1, 0] ;
                 [1, 1, 0, 1, 0]  :> [0, 1, 1] ;
                 [1, 1, 0, 1, 1]  :> [1, 0, 0] ;
                 [1, 1, 1, 0, 0]  :> [0, 0, 0] ;
                 [1, 0, 0, 0, 0]  :> [1, 0, 0] ;
                 [1, 0, 1, 0, 0]  :> [0, 1, 1] ;
                 [1, 0, 0, 1, 1]  :> [0, 1, 0] ;
                 [1, 0, 0, 1, 0]  :> [0, 0, 1] ;
                 [1, 0, 0, 0, 1]  :> [0, 0, 0] ;
TEST_VECTORS     ([R, H, Y]      -> [Q2, Q1, Q0])
                  [0, .X., .X.]   -> [0, 0, 0] ;
                  [1, . C., 1]     -> [0, 0, 1] ;
                  [1, . C., 1]     -> [0, 1, 0] ;
                  [1, . C., 1]     -> [0, 1, 1] ;
                  [1, . C., 1]     -> [1, 0, 0] ;
                  [1, . C., 1]     -> [0, 0, 0] ;
                  [1, . C., 0]     -> [1, 0, 0] ;
                  [1, . C., 0]     -> [0, 1, 1] ;
                  [1, . C., 0]     -> [0, 1, 0] ;
                  [1, . C., 0]     -> [0, 0, 1] ;
                  [1, . C., 0]     -> [0, 0, 0] ;
END Compteur_reversible.
```

6.3- Description par équation :

```
MODULE Compteur_reversible
TITLE 'Compteur  modulo 5'
DECLARATIONS
        Compteur device 'P16V8' ;                          // PLD à programmer
        H, R, Y pin  1, 2, 3 ;                             // Variables d'entrée
        Q0, Q1, Q2 pin 14, 15, 16 Istype 'reg, buffer ';   // Variables sortie
        Q = [Q2, Q1, Q0] ;                                 // Définition du bus Q
EQUATIONS
        Q.clk  = H ;
        Q.AR  = ! R;
        When R == 0 then Q := 0 Else When H == 1 then Q := Q + 1
                                     Else Q := Q - 1 ;
        When Q > 4 then Q := 0 ;
END Compteur_reversible.
```