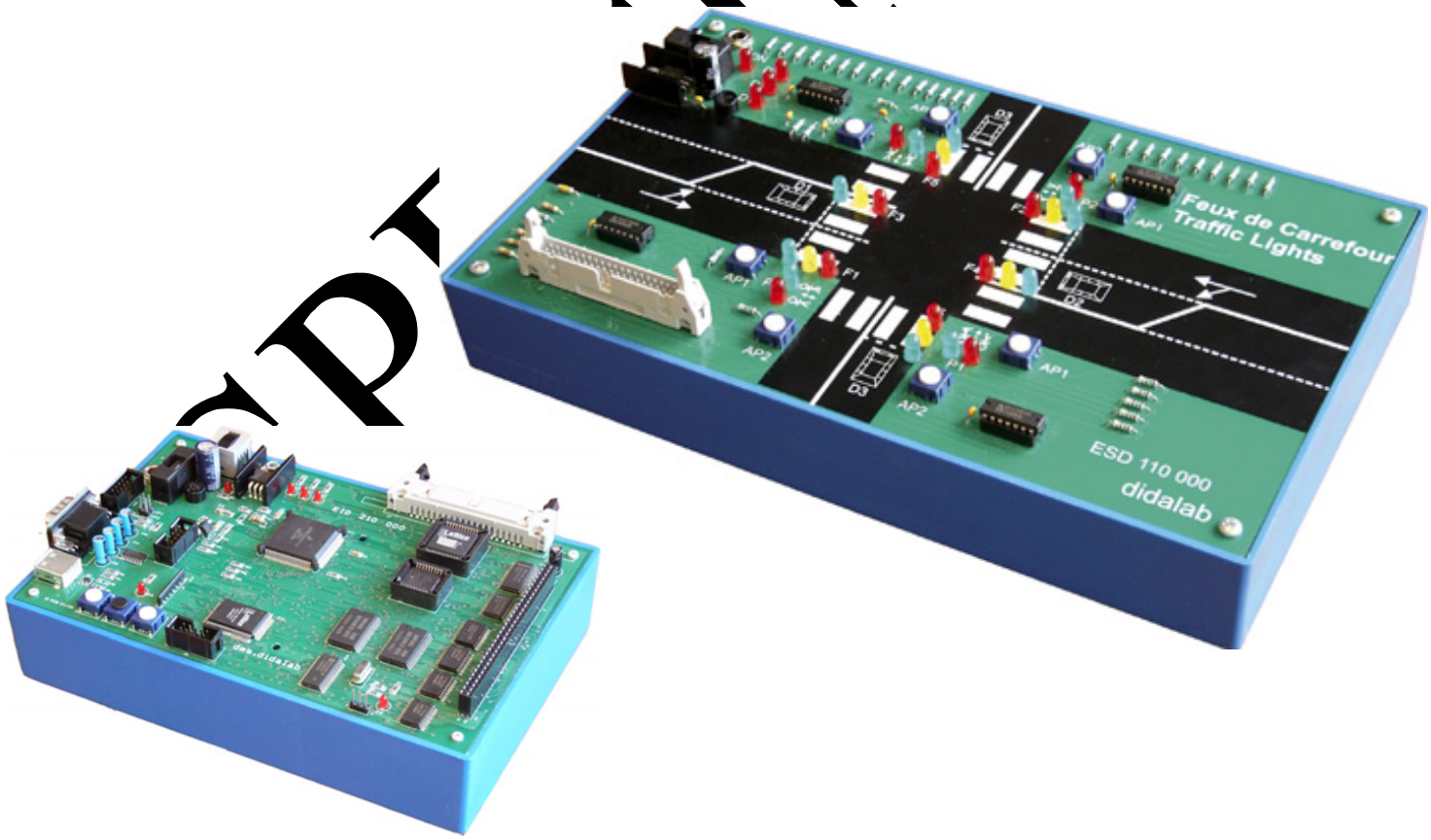
MANUEL
de
TRAVAUX PRATIQUES

Système EID210
+
Module "Feux de carrefour "



didalab
ENSEIGNEMENT SUPERIEUR

DIDALAB
5 Rue du Groupe Manoukian
78990 Elancourt
Tel: 01.30.66.08.88 / Fax: 01.30.66.72.20
ge@didalab.fr

SOMMAIRE

TP 1 : CYCLE SIMPLE SANS APPEL PIETONS NI DETECTION VOITURES

1.1 Sujet

Objectif :	Etre capable d'activer les différents feux de la maquette " Feux de carrefour" Etre capable de représenter par un grafcet l'enchaînement séquentiel défini par un cahier des charges. Etre capable de programmer en langage assembleur un enchaînement séquentiel représenter par un grafcet. Etre capable de réaliser des temporisations par une boucle d'attente de type logiciel.
Cahier des charges :	On désire réaliser le cycle suivant : - voies principales (feux P1, P2 au vert) pendant 12S puis, - passage en état orange pendant 3S puis, - voies secondaires (feux P2 et P1 au vert) pendant 3S puis, - passage en état orange pendant 3S puis - boucler. Les attentes seront réalisées par boucle logiciel

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,

Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 100 000

Câble de liaison USB ou à défaut câble RS232, Réf : EGD 000 003

Alimentation AC/AC 8V, 1 A Réf : EGD000001,

Carte feux de carrefour réf : EID 002 000,

Temps alloué : 4 heures

1.2 Eléments de solution

1.2.1 Activation des sorties

Tableau d'affectation des feux aux bits des ports :

Feu **F3** Bit Port A Bit HSRR1	Vert Orange Rouge 7 6 15 14 13 12	Feu **F2** Bit Port A Bit HSRR1	Vert Orange Rouge 5 4 3 11 10 9 8 7 6	Feu **F1** Bit Port A Bit HSRR1	Vert Orange Rouge 2 1 0 5 4 3 2 1 0
Feu **F5** Bit Port B Bit HSRR0	Vert Orange Rouge 6 5 4 13 12 11 10 9 8	Feu **F4** Bit Port B Bit HSRR0	Vert Orange Rouge 3 2 1 7 6 5 4 3 2	Feu **F3** Bit Port B Bit HSRR0	Vert Orange Rouge 0 1 0
Feu **P2** Bit Port C	Vert Rouge 2 1	Feu **P1** Bit Port C	Vert Rouge 0	Feu **P1** Bit Port B Bit HSRR0	Vert Rouge 7 15 14

Les feux sur les ports A et B sont allumés par un couple binaire « 0 1 » dans l'emplacement correspondant du registre HSRR. Un feu sera éteint si on donne la valeur « 1 0 » au même couple
Les feux sur le port C sont allumés par un 1 dans le registre de donnée associé au port C (Label noté Port_C).

Exemple :
On souhaite autoriser le passage voiture sur les voies F1 et F2 uniquement et le passage piétons sur P2 soit :
- Feux F1 , F2 et P2 au vert
- Feux F3, F4, F5 et P1 au rouge

On devra écrire les combinaisons binaires suivantes :
-> Pour le registre HSRR1 qui permet de définir les états du port A

Feu **F3** Bit Port A Bit HSRR1	Vert Orange Rouge 0 1 1 0 0 1	Feu **F2** Bit Port A Bit HSRR1	Vert Orange Rouge 1 0 0 0 1 1 0 1 0	Feu **F1** Bit Port A Bit HSRR1	Vert Orange Rouge 1 0 0 0 1 1 0 1 0

HSRR1 = 1001 0110 1001 1010 en binaire = $969A (codage Hexadécimal)

-> Pour le registre HSRR0 qui permet de définir les états du port B

Feu **F5** Bit Port B Bit HSRR0	Vert Orange Rouge 0 0 1 1 0 1 0 0 1	Feu **F4** Bit Port B Bit HSRR0	Vert Orange Rouge 0 0 1 1 0 1 0 0 1	Feu **F3** Bit Port B Bit HSRR0	Vert Orange Rouge 0 1 0
Feu **P1** Bit Port B Bit HSRR0	Vert Rouge 0 1 0				

HSRR0 = 1010 1001 1010 0110 en binaire = $A9A6 (codage Hexadécimal)

-> Pour le registre de donnée du port C (de label Port_C)

Feu **P2** Bit Port C	Vert 0	Rouge 1	Feu **P1** Bit Port C	Vert 0	Rouge

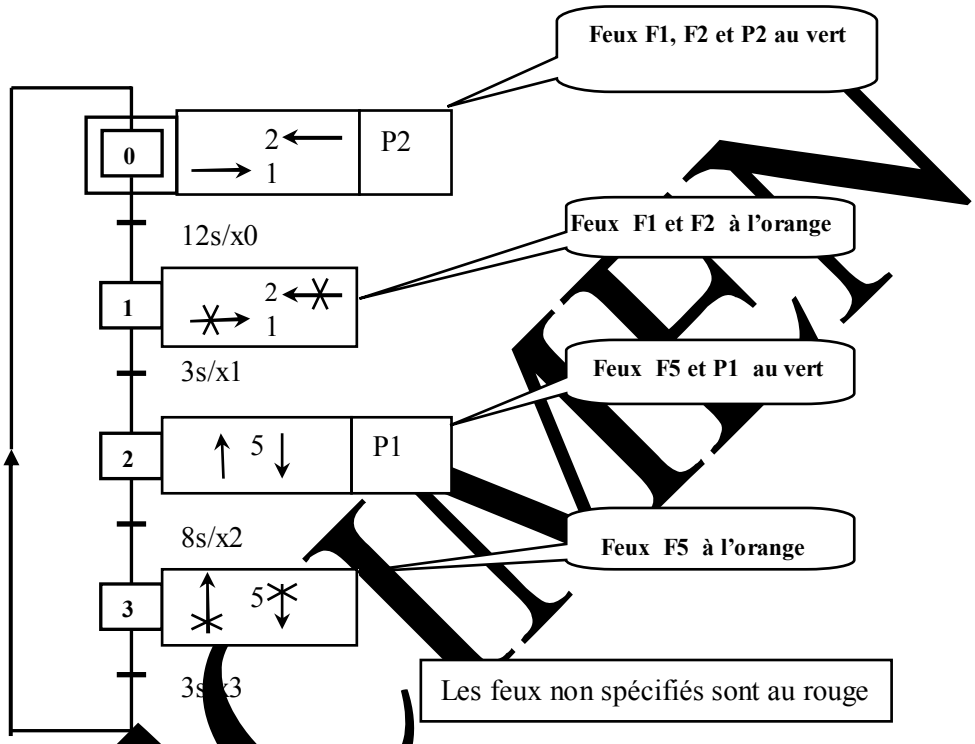
Port_C = xxxxx010xxxxxxxx = $0200

1.2.2 Représentation du cahier des charges par un Grafcet

Pour simplifier la représentation on adopte les conventions suivantes:

- On représente par une flèche la voie avec feu au vert.
- On représente par un flèche barrée une voie avec feu à l'orange.
- Un feu non représenté ni sité est au rouge.

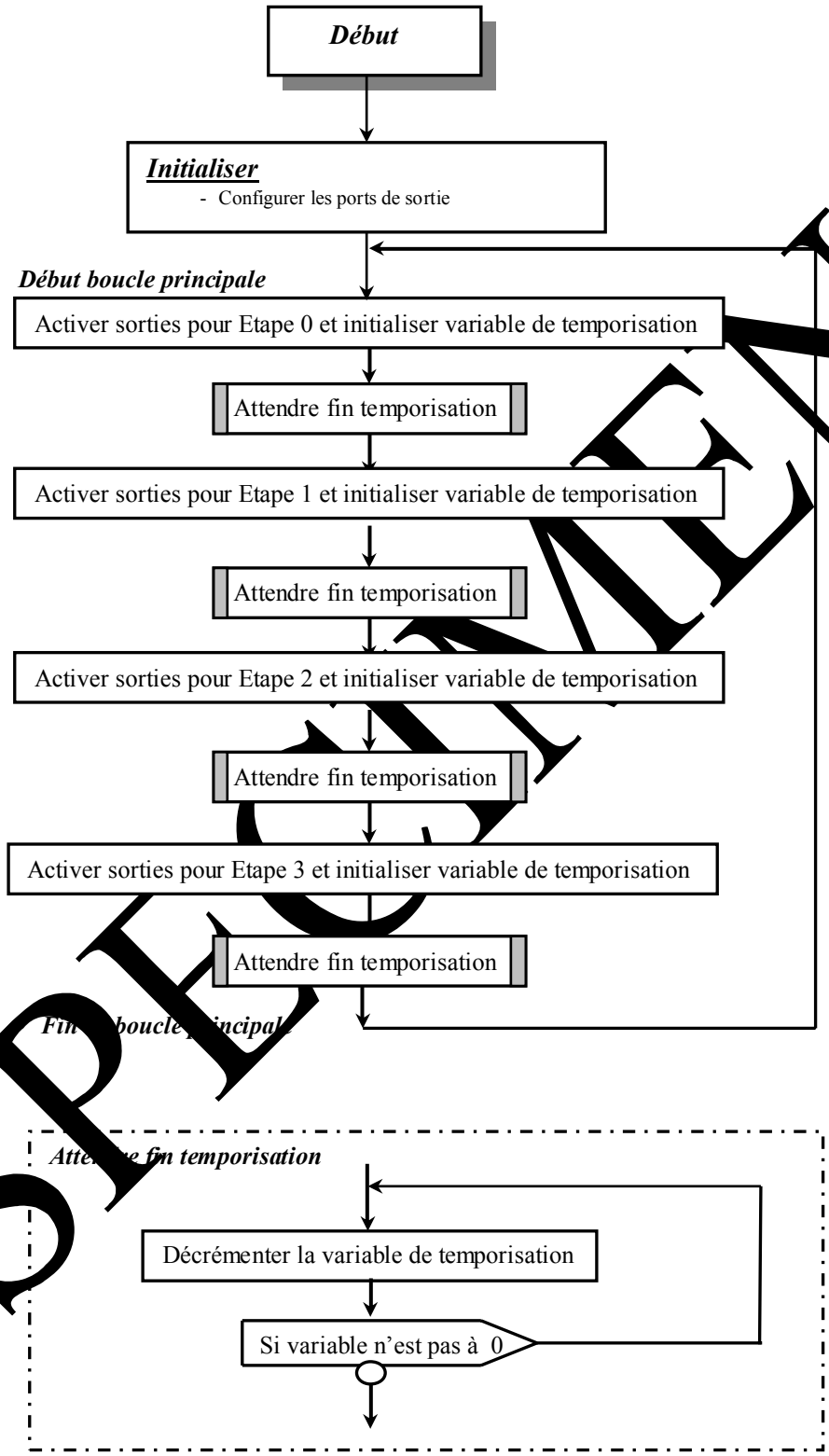
Ce qui donne pour le cahier des charge du sujet :



On trouvera en ANNEXE le tableau détaillant la détermination des mots binaires à charger dans les différents registres.

N° Etape	CONTENU DES REGISTRES (en hexadécimal) HSRR0	HSRR1	Port C
0	69A6	969A	0400
1	69A6	99A6	0200
2	9AA6	9A69	0300
3	66A6	9A69	0200

1.2.3 Organigramme de programmation du grafcet



1.2.4 Programme en assembleur A68xxx

```
*************************************************************************
*                          TP EID210 + FEU  DE  CARREFOUR                 *
*************************************************************************
*     Cahier des charges TP N°1:                                         *
*    ************************                                            *
*    -  Permutations régulières : voies principales -  voies secondaires *
*    - Les appels piétons et détection de présences voitures ne sont pas gérés *
*    - Les temporisations sont réalisées par des boucles d'attente de type logiciel *
*  NOM du FICHIER:  Feu_Carf_TP1.SRC                                    *
*************************************************************************
        * Inclusion du fichier définissant les différents labels
        include 68332.def
*   Définition de constantes        *
*********************************
*********************************
*  Début du programme exécutable   *
*********************************
        section                 code
*       INITIALISER
************************
* Configurer le port A    en mode "Discet Input Output"   (DIO)-> code $8
DEBUT move.w            #$8888,CFSR3        * CHA0 à CHA3 en mode "DIO"
                move.w          #$8888,CFSR2        * CHA4 à CHA7 en mode "DIO"
                move.w          $8888,CFSR1         * CHA8 à CHA11 en mode "DIO"
                move.w          #$8888,CFSR0        * CHA12 à CHA15 en mode "DIO"
* Définir les priorités
                move.w          #$FFFF,CPR1         * Tous les bits de PA en priorité haute
                move.w          #$FFFF,CPR0         * Tous les bits de PB en priorité haute
* Tous les feux sont au rouge
                move.w          #$9A69,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
                move.w          #$69A6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
                move.w          #$0700,DIR_Port_C   * Les 3 bits lsb du port C en sortie
                move.w          #$0200,Port_C       *  Pour les feux sur le port C
*       BOUCLE PRINCIPALE
************************
Deb_BP
*  ETAPE n°0  Autorisation voies principales (Feux 1 et 2 au vert)
                move.w          #$969A,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
                move.w          #$69A6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
                move.w          #$0400,Port_C       *  Piétons 2 au VERT
*  Boucle d'attente d'environ 12 secondes
                move.l          #$00DFCFFF,d2
ATT1            sub.l           #1,d2
                bne             ATT1
* ETAPE n°1  Les Feux 1 et 2 passent au orange
                move.w          #$9A69,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
                move.w          #$9AA6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
                move.w          #$0200,Port_C       *  Piétons 2 passe au ROUGE
*  Boucle d'attente d'environ 3 secondes
                move.l          #$004FCFFF,d2
ATT2            sub.l           #1,d2
                bne             ATT2
* ETAPE n°2  Les Feux 5 passent au vert
                move.w          #$9A69,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
                move.w          #$9AA6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
                move.w          #$0300,Port_C       *  Piétons 1 passe au VERT
*  Boucle d'attente d'environ 7 secondes
                move.l          #$009FCFFF,d2
ATT3            sub.l           #1,d2
                bne             ATT3
* ETAPE n°3  Les Feux 5 passent à l'orange
                move.w          #$9A69,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
                move.w          #$66A6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
                move.w          #$0200,Port_C       *  Piétons 1 passe au rouge
*  Boucle d'attente d'environ 3 secondes
                move.l          #$004FCFFF,d2
ATT4            sub.l           #1,d2
                bne             ATT4
* boucler
                bra             Deb_BP
*       Fin de la boucle principale et Fin du programme principal
                end                             *       Fin du listing
```

1.2.5 Programme en C

```
/*****************************************************************************************
*                          TP SUR EID210 avec FEU  DE  CARREFOUR                         *
*****************************************************************************************
*     Cahier des charge  TP N°1 :                                                       *
*   *********************************                                                   *
*   - Permutations régulières:  voies principales - voies secondaires                   *
*   - Les appels piétons et détection de présences voitures ne sont pas gérés           *
*   - Les temporisations sont réalisées par des boucles d'attente de type logiciel      *
*                                                                                       *
*  NOM du FICHIER:  Feu_Carf_TP1.C                                                      *
*  *********************                                                                *
*****************************************************************************************/
/* Liste des fichiers à inclure */
#include "Cpu_reg.h"              // Voir listing en "ANNEXE"
#include "EID210_reg.h"           // Voir listing en "ANNEXE"
#include "Structures_Donnees.h"// Voir listing en "ANNEXE"
#include "feux_carrefour.h"       // Voir listing en "ANNEXE"
void Active_Etat(int);
void Attendre(int);

/*************************************
*       FONCTION PRINCIPALE      *
*************************************/
main()
{
/*            INITIALISER
*******************************/
/* Définition des "directions" du port C "   */
Dir_PortC=0x07;
PortC=0x00;
/*  Configurer le port A    en mode "Discet Input Output"   (DIO)-> code $8888 */
CFSR3=0x8888;                           /*  CHA0 à CHA3 en mode "DIO"                  */
CFSR2=0x8888;                           /*  CHA4 à CHA7 en mode "DIO"                  */
CFSR1=0x8888;                           /*  CHA8 à CHA11 en mode "DIO"                 */
CFSR0=0x8888;                           /*  CHA12 à CHA15 en mode "DIO"                */
/* Définir les priorités */
CPR1=0xFFFF;                            /* Tous les bits de PA en priorité haute       */
CPR0=0xFFFF;                            /* Tous les bits de PB en priorité haute       */

/*            BOUCLE PRINCIPALE
************************************************/
do
{           Active_Etat(1);             /* Autorisation des voies principales(feux 1 et 2 au vert)*/
            Attendre(12);               /* temporisation de 12 secondes                */
            Active_Etat(11);            /* les feux 1 et 2 passent à l'orange          */
            Attendre(3);                /* temporisation de 3 secondes                 */
            Active_Etat(7);             /* Autorisation des voies secondaires (feux 5 au vert) */
            Attendre(8);                /* temporisation de 8 secondes                 */
            Active_Etat(8);             /* les feux 5 passent à l'orange               */
            Attendre(3);                /* temporisation de 3 secondes                 */
} while(1);    /* Fin de la boucle principale
}   /*      Fin de la fonction principale
   ************************************************/

/*fonction d'affectation des sorties sur port C
************************************************/
void Active_Etat(int j)                 /* Affectation des registres de sorties du port C en fonction de l'étape active */
{
            switch(j)   /* numéro de l'étape */

            case 1 :    {HSRR0=0x69A6;HSRR1=0x969A;PortC=0x04;break;}        //  Voir tableau en "ANNEXE"
            case 7 :    {HSRR0=0x9AA6;HSRR1=0x9A69;PortC=0x03;break;}
            case 8 :    {HSRR0=0x66A6;HSRR1=0x9A69;PortC=0x02;break;}
            case 11 :   {HSRR0=0x69A6;HSRR1=0x99A6;PortC=0x02;break;}
            default :   {HSRR0=0xAAAA;HSRR1=0xAAAA;PortC=0x00;break;}        // On éteint tout
            }
}

/*Boucle d'attente logicielle
*********************************/
void Attendre(int Duree) // Duree passée en paramètre
{
int j,fin;
fin=Duree*250000;
for(j=0;j<=fin;j++); //boucle d'attente
}
```

TP 2 : CYCLE COMPLET SANS TRAITEMENT DES APPELS PIETONS NI DETECTION DES VOITURES

2.1 Sujet

Objectif :	**Capacités compémentaires:** Etre capable de programmer un enchaînement séquentiel complet prédéfini Etre capable de réaliser une boucle de temporisation à l'aide du timer interne au microcontrôleur
Cahier des charges :	le cycle doit comporter les bifurcations soit: - voies principales (feux F1, F2 et P2 au vert) pendant 10S puis, - passage en état orange du feu F1 pendant 3S puis, - voie principale 2 (feu F2) avec bifurcation 4 (F4) pendant 8S puis, - passage en état orange des feux F2et F4 pendant 3S puis, - voies secondaires (feux F5 et P1 au vert) pendant 3S puis, - passage en état orange pendant 3S puis, - voies principales (feux F1, F2 et P2 au vert) pendant 10S puis, - passage en état orange du feu F2 pendant 3S puis, - voie principale 1 (feu F1) avec bifurcation 3 (F3) pendant 8S puis, - passage en état orange des feux F1et F4 pendant 3S puis, - voies secondaires (feux F5 et P1 au vert) pendant 3S puis, - BOUCLER Les attentes seront réalisées par boucle logiciel puis par timer interne au micro-contrôleur

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,

Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 100 000

Câble de liaison USB, ou à défaut câble RS232, Réf : EGD 000 003

Alimentation AC/AC 8V, 1 A Réf : EGD000001,

Carte feux de carrefour réf : EID 002 000,

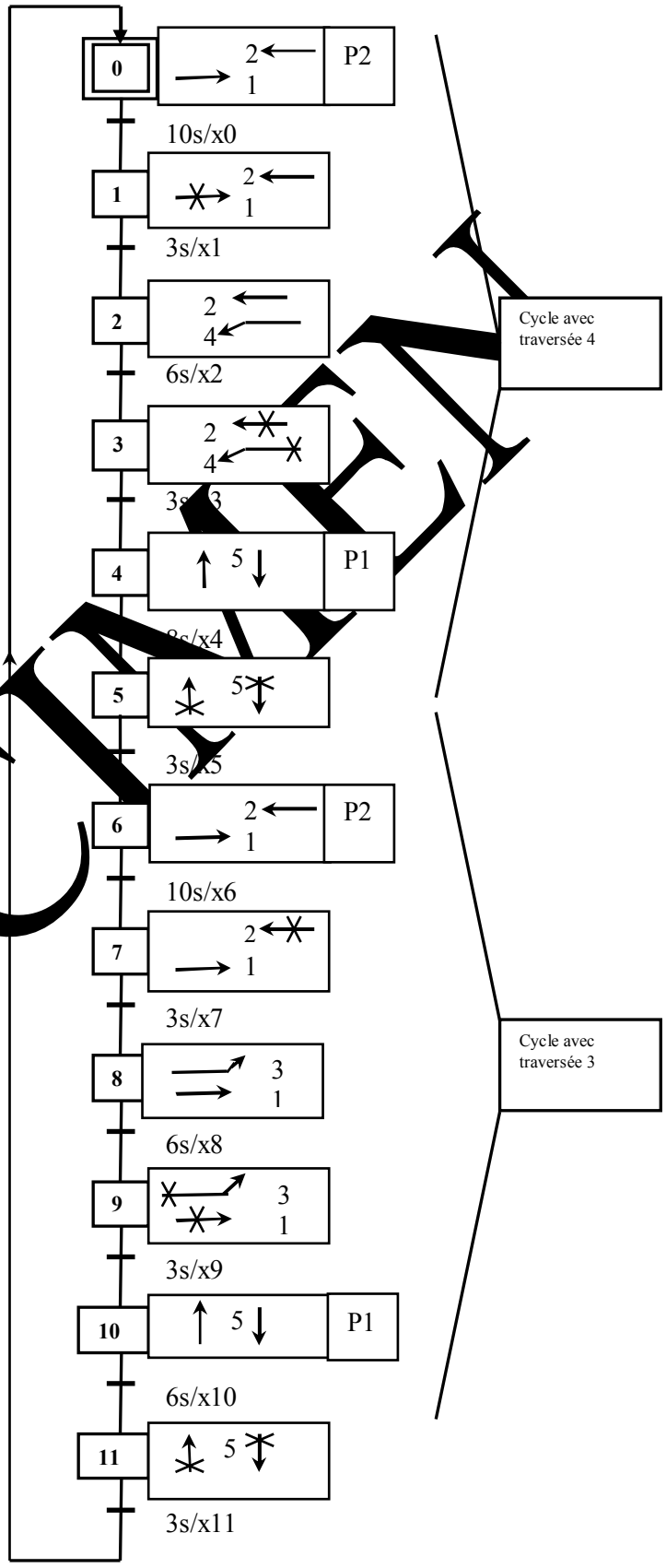
Temps alloué : 4 heures

2.2 Eléments de solution

2.2.1 Grafcet

On trouvera en ANNEXE le tableau détaillant la détermination des mots binaire à charger dans les différents registres

N° Etape	CONTENU DES REGISTRES (en hexadécimal) HSRR0	HSRR1	Port C
0	69A6	969A	0400
1	69A6	96A6	0200
2	696A	96A9	0200
3	699A	99A9	0200
4	9AA6	9A69	0300
5	66A6	9A69	0200
6	69A6	969A	0400
7	69A6	999A	0200
8	69A5	AA5A	0200
9	69A6	6A66	0200
10	9AA6	9A69	0300
11	66A6	9A69	0200



2.2.2 Programme en Assembleur A68xxx avec temporisation 'logicielle'

```
        ************************************************************************
        *                        TP EID210 + FEU  DE  CARREFOUR                *
        ************************************************************************
        *     Cahier des charges TP N°2 Variante 1:                            *
        *   *********************************                                  *
        *   - Permutations régulières  voies principales -  Traversée 4 - voies secondaires  *
        *               voies principales -  Traversée 3 - voies secondaires ..etc  *
        *   - Les appels piétons et détection de présences voitures ne sont pas gérés  *
        *   -  Les temporisations sont réalisées par des boucles d'attente de type logiciel  *
        *  NOM du FICHIER:  Feu_Carf_TP2-1.SRC                               *
        ******************                                                     *
        ************************************************************************
        * Inclusion du fichier définissant les différents labels
        include 68332.def
*   Définition de constantes        *
************************************
*  Début du programme exécutable          *
************************************
        section                 code
*       INITIALISER
*************************
* Configurer le port A    en mode "Discet Input Output"   (DIO)-> code $8
DEBUT   move.w          #$8888,CFSR3        * CHA0 à CHA3 en mode "DIO"
        move.w          #$8888,CFSR2        * CHA4 à CHA7 en mode "DIO"
        move.w          #$8888,CFSR1        * CHA8 à CHA11 en mode "DIO
        move.w          #$8888,CFSR0        * CHA12 à CHA15 en mode "DIO
 Définir les priorités
        move.w          #$FFFF,CPR1         * Tous les bits de PA en priorité haut
        move.w          #$FFFF,CPR0         * Tous les bits de P  priorité haute
* Tous les feux sont au rouge
        move.w          #$9A69,HSRR1        *  Pour les F x sur le p t A (C
        move.w          #$69A6,HSRR0        *  Pour les  r sur le port  (CH8 à 1
        move.w          #$0700,DIR_Port_C   *  Les 3 bits l  port C  en rtie
        move.w          #$0200,Port_C       *  Po les feux  port C
*       BOUCLE PRINCIPALE
*****************************
Deb_BP
*  ETAPE n°0  Autorisation voies principales (Feux  et 2 au vert) * (  Début du c  incluant la traversée 4)
************************************************************************************
        move.w          #$969A,H RR1        Pour les Feux sur e port A (CH0 à 7)
        move.w          #$69A ,HSRR0        ur les Feux su le port B (CH8 à 15)
        move.w          #$0 0,Port_C        *  2 au V RT
*  Boucle d'attente d'environ 12 se des
        move.l          #$0 CFFF 2
ATT1    sub.l           #1,d2
        bne             TT1
* ETAPE n°1  Le Feu  passe à  nge
        move.w          #  6A6,HSRR1       *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #  9A6,HSRR0       *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C       *  Piétons 2 passe au ROUGE
*  Boucle  tente d'environ  condes
        move.l          #$004FCFFF,d2
ATT2    sub.l           #1
                        ATT2
*  ETAPE  Autorisation  versée 4 (Feux 2 et 4 au vert)
*****************************************************
        move.w          #$96A9,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$696A,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C                   *  Piétons  au rouge
*  Boucle d'attente d'environ 8 secondes
        move.l          #$009FCFFF,d2
ATT3    sub.l           #1,d2
        bne             ATT3
* ETAPE n°3  Les Feux 2 et 4 passent à l'orange
        move.w          #$99A9,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$699A,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C       *  Piétons au ROUGE
*  Boucle d'attente d'environ 3 secondes
        move.l          #$004FCFFF,d2
ATT4    sub.l           #1,d2
        bne             ATT4
```

```
* ETAPE n°4  Les Feux 5 passent au vert
************************************
        move.w          #$9A69,HSRR1    *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$9AA6,HSRR0    *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0300,Port_C   *  Piétons 1 passe au VERT
*  Boucle d'attente d'environ 8 secondes
        move.l          #$009FCFFF,d2
ATT5    sub.l           #1,d2
        bne             ATT5
* ETAPE n°5  Les Feux 5 passent à l'orange
        move.w          #$9A69,HSRR1    *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$66A6,HSRR0    *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C   *  Piétons 1 passe au rouge
*  Boucle d'attente d'environ 3 secondes
        move.l          #$004FCFFF,d2
ATT6    sub.l           #1,d2
        bne             ATT6
*  ETAPE n°6  Autorisation voies principales (Feux 1 et 2 au vert)
*   ( Début du cycle incluant la traversée 3)
*********************************************************
        move.w          #$969A,HSRR1    *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0    *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0400,Port_C   *  Piétons 2 au VERT
*  Boucle d'attente d'environ 12 secondes
        move.l          #$00DFCFFF,d2
ATT7    sub.l           #1,d2
        bne             ATT7
* ETAPE n°7  Le Feu 2 passe à l'orange
        move.w          #$999A,HSRR1    *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0    *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C   *  Piétons 2 passe au ROUGE
*  Boucle d'attente d'environ 3 secondes
        move.l          #$004FCFFF,d2
ATT8    sub.l           #1,d2
        bne             ATT8
*  ETAPE n°8  Autorisation traversée 3  (Feux 1 et 3 au vert)
*********************************************
        move.w          #$AA5A,HSRR1    *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A5,HSRR0    *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C   *  Piétons  au rouge
*  Boucle d'attente d'environ 8 secondes
        move.l          #$009FCFFF,d2
ATT9    sub.l           #1,d2
        bne             ATT9
* ETAPE n°9  Les Feux 1 et 3 passent à l'orange
        move.w          #$9A69,HSRR1    *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0    *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C   *  Piétons au ROUGE
*  Boucle d'attente d'environ 3 secondes
        move.l          #$004FCFFF,d2
ATT10   sub.l           #1,d2
        bne             ATT10
* ETAPE n°10  Les Feux 5 passent au vert
*****************************************
        move.w          #$9A69,HSRR1    *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$9AA6,HSRR0    *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0300,Port_C   *  Piétons 1 passe au VERT
*  Boucle d'attente d'environ 8 secondes
        move.l          #$009FCFFF,d2
ATT11   sub.l           #1,d2
        bne             ATT11
* ETAPE n°11  Les Feux 5 passent à l'orange
        move.w          #$9A69,HSRR1    *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$66A6,HSRR0    *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C   *  Piétons 1 passe au rouge
*  Boucle d'attente d'environ 3 secondes
        move.l          #$004FCFFF,d2
ATT12   sub.l           #1,d2
        bne             ATT12
        bra             Deb_BP          * boucler
*       Fin de la boucle principale, Fin du programme principal
*********************************************************
          end                           * Fin du listing
```

2.2.3 Programmation en C (avec boucle d'attente logicielle)

Complément sur l'algorithme de programmation en "C"

On remarque que le système étudié comporte 15 états possibles, comme le montre le tableau donné ci-contre.
On rappelle la représentation adoptée:

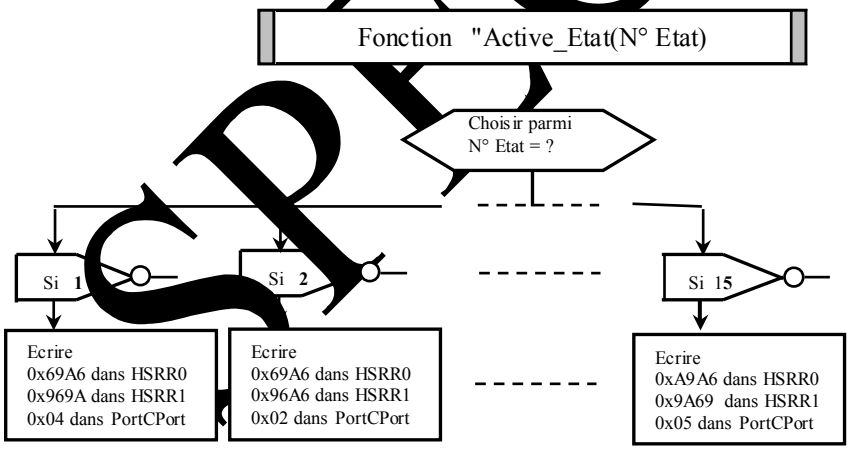
- On représente par une flèche la voie avec feu au vert.
- On représente par un flèche barrée une voie avec feu à l'orange.
- Un feu non représenté ni sité est au rouge.

Pour le grafcet à programmer, on établit la correspondance entre une étape et l'état associé:

N° Etape	0	1	2	3	4	5	6	7	8	9	10	11
N° Etat	1	2	3	4	7	8	1	9	5	6	7	8

Lorsque l'on veut activer un état, on appelle une fonction appelée "Active_Etat(N° Etat)" , où le paramètre passé lors de l'appel de la fonction n'est autre que le n° de l'état.
Exemple d'appel de la fonction en langage "C":
Active_Etat(3); // Activer l'état n° 3

C'est dans la fonction elle même que l'on charge les différents registres avec les valeurs adéquates (Ces valeurs sont définies dans le tableau donné en Annexe).



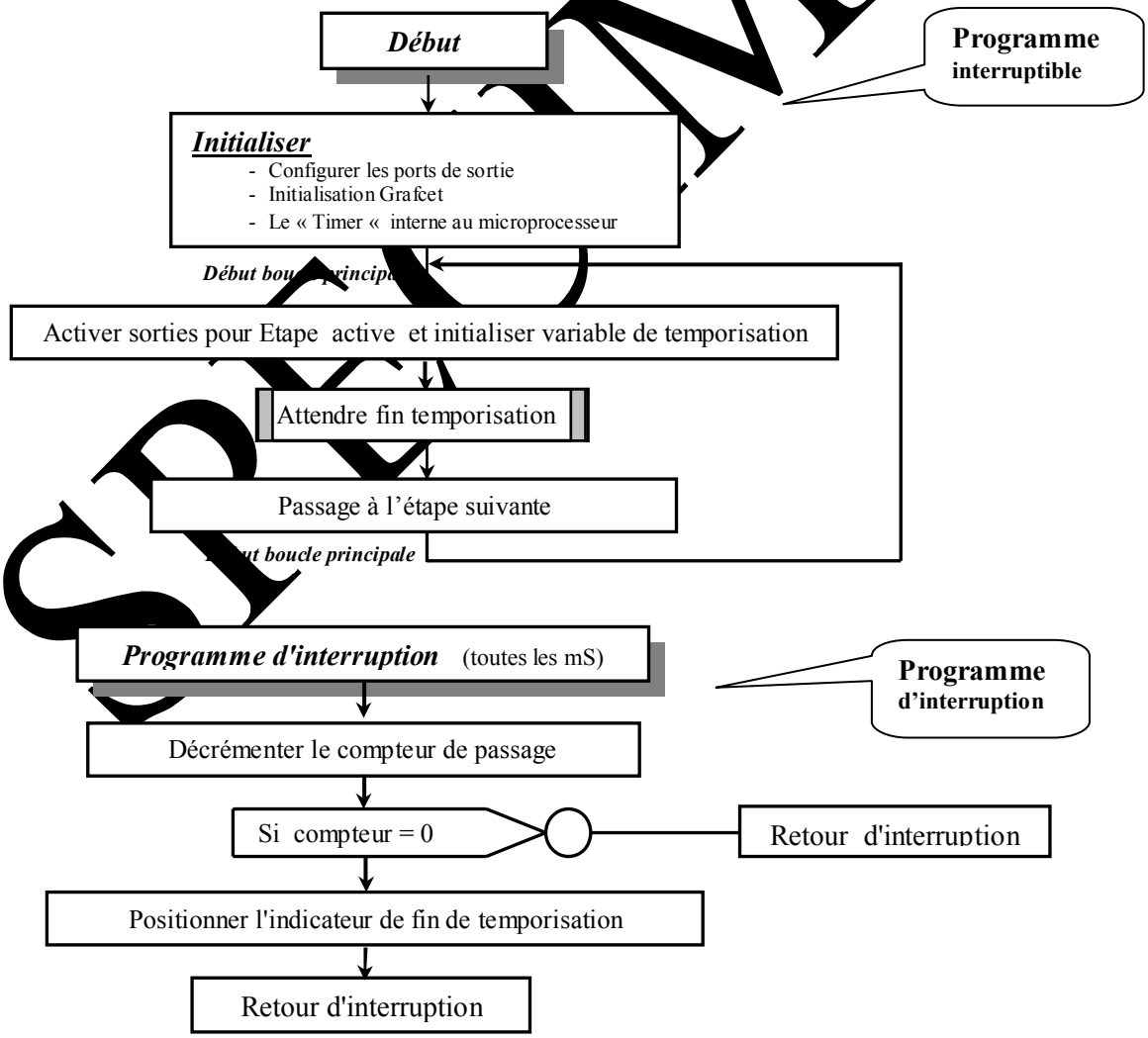
N° Etat	Etat des feux	
1	2 ← 1 →	P2
2	2 ← 1 ✳	
3	2 ← 4 ←	
4	2 ✳ 4 ← ✳	
5	3 → 1 →	
6	3 ✳ 1 ✳	
7	5 ↑ ↓	P1
8	5 ✳ ✳	
9	2 ✳ ← 1 →	
10	3 ✳ 1 →	
11	2 ✳ 1 ✳	
12	2 ✳ ← 1 →	
13	2 ← 4 ✳	
14	3 ✳ 1 →	
15	Aucun feu F Mais feu P	

Remarque: Dans l'exemple présent, seuls les états de N° de 1 à 9 sont activés.

```
/*****************************************************************************************************
*                              TP SUR EID210 avec FEU  DE  CARREFOUR                                 *
*****************************************************************************************************
* Cahier des charges TP 2 Variante 1 :                                                               *
*             - permutations régulières : voies principales-traversée 4-voies                        *
*               secondaires-voies principales-traversée 3-voies secondaires..etc                     *
*             - les appels piéton et détections de présence de voiture ne sont  pas gerés            *
*             - les temporisations sont réalisées par des boucles d'attente de type logiciel         *
*                                                                                                    *
* NOM du FICHIER:  Feu_Carf_TP2-1.C                                                                  *
*                                                                                                    *
*****************************************************************************************************/

/* Liste des fichiers à inclure */
#include "Cpu_reg.h"              // Voir listing en "ANNEXE"
#include "EID210_reg.h"           // Voir listing en "ANNEXE"
#include "Structures_Donnees.h"// Voir listing en "ANNEXE"
#include "feux_carrefour.h"       // Voir listing en "ANNEXE"
void Active_Etat(int);
void Attendre(int);
int Etat,Duree;
/*****************************************************
*       FONCTION PRINCIPALE                          *
*****************************************************/
main()
{
/*            INITIALISER
************************************/
/* Définition des "directions" du port C "   */
PortCDir=0x07;
PortCPort=0x00;
/*  Configurer le port A    en mode "Discet Input Output"   (DIO)-> code $8  */
CFSR3=0x8888;        /*  CHA0 à CHA3 en mode "DIO"              */
CFSR2=0x8888;        /*  CHA4 à CHA7 en mode "DIO"              */
CFSR1=0x8888;        /*  CHA8 à CHA11 en mode "DIO"             */
CFSR0=0x8888;        /*  CHA12 à CHA15 en mode "DIO"            */
/* Définir les priorités */
CPR1=0xFFFF;         /* Tous les bits de PA en priorité haute   */
CPR0=0xFFFF;         /* Tous les bits de PB en priorité haut    */

/*            BOUCLE PRINCIPALE
************************************************/
do
{
            Active_Etat(1);                 /*Autori ion des voies principales(feux 1 et 2 au vert)   */
            Attendre(10);                   /*tempor ion de 12 secondes                               */
            Active_Etat(2);                 /*le feu 1 p  à l'orange                                  */
            Attendre(3);                    /*temporisa  3 secon s                                    */
            Active_Etat(3);                 /*Aut risation  et 4 (feux 2 et 4 au vert)                */
            Attendre(6);                    /*tem isation de 6 secondes                               */
            Active_Etat(4);                 /*l feux 2 et  passent à l'orange                         */
            Attendre(3);                    /*temporisation de 3 secondes                             */
            Active_Etat(7);                 /*utorisation  la voie 5 (feux 5 au vert)                 */
            Attendre(8);                    /*  porisat n de 8 secondes                               */
            Active_E  (8);                  /*le  x  passent à l'orange                               */
            Attendre(3);                    /*temp isation de 3 secondes                              */
            Active_Etat(1);                 /*Autorisation des voies principales(feux 1 et 2 au vert) */
            Attendre(10);                   /*temporisation de 10 secondes                            */
            Active_Etat(9);                 /*le feu 2 passe à l'orange                               */
            Attendre(3);                    /*temporisation de 3 secondes                             */
            Active_Etat(5);                 /*Autorisation des voies 1 et 3 (feux 1 et 3 au vert)     */
            Attendre(6);                    /*temporisation de 6 secondes                             */
            Active_Etat(6);                 /*les feux 1 et 3 passent à l'orange                      */
            Attendre(3);                    /*temporisation de 3 secondes                             */
            Active_Etat(7);                 /*Autorisation de la voie 5 (feux 5 au vert)              */
            Attendre(6);                    /*temporisation de 6 secondes                             */
            Active_E  (8);                  /*les feux 5 passent à l'orange                           */
            Attendre(3);                    /*temporisation de 3 secondes                             */
            } while(1);    /* Fin de la boucle principale *
}   /*      Fin de la fonction principal
    ***********************************************/
```

```
/*  Fonction d'affectation des sorties sur port C
*********************************************************/
void Active_Etat(int j)              /*Affectation des registres de sorties en fonction du N° de l'état à activer*/
{
        switch(j) /* j:numéro de l'état */
        {
        case 1 :   {HSRR0=0x69A6;HSRR1=0x969A;PortC=0x04;break;}
        case 2 :   {HSRR0=0x69A6;HSRR1=0x96A6;PortC=0x02;break;}
        case 3 :   {HSRR0=0x696A;HSRR1=0x96A9;PortC=0x02;break;}
        case 4 :   {HSRR0=0x699A;HSRR1=0x99A9;PortC=0x02;break;}
        case 5 :   {HSRR0=0x69A5;HSRR1=0xAA5A;PortC=0x02;break;}
        case 6 :   {HSRR0=0x69A6;HSRR1=0x6A66;PortC=0x02;break;}
        case 7 :   {HSRR0=0x9AA6;HSRR1=0x9A69;PortC=0x03;break;}
        case 8 :   {HSRR0=0x66A6;HSRR1=0x9A69;PortC=0x02;break;}
        case 9 :   {HSRR0=0x69A6;HSRR1=0x999A;PortC=0x02;break;}
        default :  {HSRR0=0xAAAA;HSRR1=0xAAAA;PortC=0x00;break;}        // On éteint tout
        }
} /* FIN de la fonction d'affectation des sorties */

/* Fonction d'attente logicielle
***********************************/
void Attendre(int Duree)          // Duree passée en paramètre
{
int j,fin;
fin=Duree*250000;
for(j=0;j<=fin;j++);  //boucle d'attente
} /* FIN de la fonction d'attente logicielle */
```

2.2.4 Organigramme avec temporisation par « Timer » interne au microprocesseur.



2.2.5 Programme en Assembleur A68xxx avec utilisation 'Timer'

```
*****************************************************************************************
*                          TP EID210 + FEU  DE  CARREFOUR                               *
*****************************************************************************************
*     Cahier des charges TP N°2   Variante 2:                                           *
*   ***********************************                                                 *
*   - Permutations régulières : voies principales  puis  Traversée 4  puis voies principales, puis Traversée 4      *                    *
puis voies secondaires  puis voies principales   puis Traversée 3   puis   voies secondaires ..etc                  *
*   - Les appels piétons et détection de présences voitures ne sont pas gérés         *
*   - Les temporisations sont réalisées à l'aide du timer du 68332                    *
*                                                                                     *
*  NOM du FICHIER:  Feu_Carf_TP2-2.SRC                                                *
**********************                                                               *
*****************************************************************************************
        * Inclusion du fichier définissant les différents labels
        include 68332.def
*       Déclaration des variables           *
**************************************
        section     var
COMPTEUR        ds.l    1
INDICATEUR      ds.b    1
*  Début du programme exécutable        *
**************************************
        section             code
*       INITIALISER
*************************
* Configurer le port A    en mode "Discet Input Output"   (DIO)-> code $8
DEBUT   move.w          #$8888,CFSR3            * CHA0 à CHA3 en mode "DIO"
        move.w          #$8888,CFSR2            * CHA4 à CHA7 en mode "DIO"
        move.w          #$8888,CFSR1            * CHA8 à CHA11 en mode "DIO"
        move.w          #$8888,CFSR0            * CHA12 à CHA15 en mode "DIO"
* Définir les priorités
        move.w          #$FFFF,CPR1             * Tous les bits de PA en priorité haute
        move.w          #$FFFF,CPR0             * Tous les bits de PB en priorité haute
* Tous les feux sont au rouge
        move.w          #$9A69,HSRR1            *  Pour les feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les feux sur le port B (CH8 à 15)
        move.w          #$0700,DIR_Port_C       *  Les 3 bits ls du port C  en sortie
        move.w          #$0200,Port_C           *  Pour les feux sur le port C
* Configurer la base de temps
        move.l          #96,d0                  * 96 est le n° du vecteur d'interruption
        move.l          #it_bt,a1               * f_it-bt est l'adresse de la fonction d'interruption
        asl.l           #2,d0
        add.l           #$1000,vect,d0          * Initialiser la table des vecteurs
        move.l          d0,a0
        move.l          a1,(a0)
        move.l          #1000,COMPTEUR          * 1000*1mS = 1S
        move.b          #0,INDICATEUR           * de fin de comptage
        move.w          #$0008,PITR             * 1 interruption toutes les 1 ms
        move.w          #$0760,PICR
*************************************
*       BOUCLE PRINCIPALE             *
*************************************
Deb_BP
*  Autorisation des voies (Feux 1 et 2 au vert)
*  (  Début du cycle incluant la traversée 4)
*************************************
        move.w          #$969A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0400,Port_C           *  Piétons 2 au VERT
*  Init tempo d'environ 12 secondes
        move.l          #12000,COMPTEUR         * 12000*1mS = 12S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT1    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT1
* Le Feu 1 passe à l'orange
        move.w          #$96A6,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons 2 passe au ROUGE
```

```
*  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR          * 3000*1mS = 3S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT2    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT2
*  Autorisation traversée 4 (Feux 2 et 4 au vert)
****************************************
        move.w          #$96A9,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$696A,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons  au rouge
*  Init tempo d'environ 8 secondes
        move.l          #12000,COMPTEUR         * 8000*1mS = 8S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT3    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT3
* Les Feux 2 et 4 passent à l'orange
        move.w          #$99A9,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$699A,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons au ROUGE
*  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR          * 3000*1mS = 3S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT4    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT4

* Les Feux 5 passent au vert
*************************
        move.w          #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$9AA6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0300,Port_C           *  Piétons 1 passe au VERT
*  Init tempo d'environ 8 secondes
        move.l          #8000,COMPTEUR          * 8000*1mS = 8S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT5    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT5
* Les Feux 5 passent à l'orange
        move.w          #$9969,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$9A66,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons 1 passe au rouge
*  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR          * 3000*1mS = 3S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT6    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT6

*  Autorisation voitures latérales (Feux 1 et 2 au vert)
*  ( Début de la quatrième traversée 3)
*****************************************
        move.w          #$969A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0400,Port_C           *  Piétons 2 au VERT
*  Init tempo d'environ 12 secondes
        move.l          #12000,COMPTEUR         * 12000*1mS = 12S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT7    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT7
* Le Feu 2 passe à l'orange
        move.w          #$999A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons 2 passe au ROUGE
```

```
*  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR          * 3000*1mS = 3S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT8    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT8
*  Autorisation traversée 3  (Feux 1 et 3 au vert)
****************************************
        move.w          #$AA5A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A5,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons  au rouge
*  Init tempo d'environ 8 secondes
        move.l          #8000,COMPTEUR          * 8000*1mS = 8S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT9    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT9
* Les Feux 1 et 3 passent à l'orange
        move.w          #$6A66,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons au ROUGE
*  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR          * 3000*1mS = 3S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT10   move.b      INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT10
* Les Feux 5 passent au vert
************************************
        move.w          #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$9AA6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0300,Port_C           *  Piétons 1 passe au VERT
*  Init tempo d'environ 8 secondes
        move.l          #8000,COMPTEUR          * 8000*1mS = 8S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT11   move.b      INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT11
* Les Feux 5 passent à l'orange
        move.w          #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$9AA6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons 1 passe au rouge
*  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR          * 3000*1mS = 3S
        move.b          #$00,INDICATEUR         * de fin de comptage
*  Boucle d'attente de fin de temporisation
ATT12   move.b      INDICATEUR,D2
        cmp.b           #01,D2
        bne             ATT12
* boucle
        bra             BP
*       Fin de la boucle principale
************************************
*       Fin du programme principal
****************************************
*********************************************
*       FONCTION D'INTERRUPTION             *
*       associée à la base de temps         *
*********************************************
it_bt   sub.l           #$00000001,COMPTEUR
        cmp.l           #$00000000,COMPTEUR
        bne             it_ret                  * Retourner si pas égal à 0
        move.b          #$01,INDICATEUR         * C'est la fin tempo
        move.l          #1000,COMPTEUR          * Réinit tempo
it_ret  rte                                     * Retour d'interruption
*  Fin de la fonction d'interruption
************************************
*       Fin du fichier source assembleur
************************************
          end
```

2.2.6 Programme en C (avec timer)

Complément sur l'algorithme de programmation en "C"

L'algorithme mis en oeuvre dans le programme donné ci-après correspond à celui décrit dans le chapitre 3 (méthode dite "A recherche de l'étape active").

Rôles des variables utilisée pour la gestion des temporisations:

Tempo_en_Cours	Est à 1 lorsqu'une tempo est en cours, sinon est à 0
CPTR_mS	Compteur de milli-secondes (est incrémenté à chaque mS lorsqu'une temporisation est en cours) est remis à 0 lorsqu'il atteint 1000.
CPTR_S	Compteur de secondes (est incrémenté à chaque fois que le compteur de mS at[illegible])
Valeur_Tempo	Nombre de secondes correspondant à la durée souhaitée de la temporisation
Fin_Tempo	Est positionnée à 1 lorsque la temporisation en cours arrive à expiration (si **CPTR_S** at[illegible]int **Valeur_Tempo**)

```
/*****************************************************************************************
*                          TP SUR EID210 avec FEU  DE  CARREFOUR                         *
*****************************************************************************************
* Cahier des charges  TP N°2  Variante 2:                                                *
*             - permutations régulières : voies principales-traversée 4-voies            *
*               secondaires-voies principales-traversée 3- voies secondaires..etc        *
*             - les appels piéton et détections de présence de voitures sont gérés       *
*             - les temporisations sont réalisées par "Timer" programmable int. 68332    *
*                                                                                        *
*  NOM du FICHIER:  Feu_Carf_TP2-2.C                                                     *
*****************************************************************************************/
/* Liste des fichiers à inclure */
//*********************************
#include "Cpu_reg.h"
#include "EID210_reg.h"
#include "Structures_Donnees.h"
#include "feux_carrefour.h"      //  Voir listing en "ANNEXE"

/* Declaration des variables
*********************************/
void TE0(void),TE1(void),TE2(void),TE3(void),TE4(void),TE5(void),TE6(void),TE7(void);  // déclaration des sous fonctions
void TE8(void),TE9(void),TE10(void),TE11(void);                                        // déclaration des sous fonctions
void Active_Etat(int);                                                 // déclaration des sous fonctions
void Init_Tempo(int);                                                  // déclaration des sous fonctions
int Etat,Duree,CPTR_mS,Valeur_Tempo;                                   /* déclaration des variables et compteurs */
unsigned char Fin_Tempo,Tempo_en_Cours,Appel_Memorise; /*  déclaration variables tempo et mémoire d'appel

/* fonction d'interruption pour tempo  (Toutes les mS)
*****************************************************/
void irq_bt()
{
        if (Tempo_en_Cours=
                { CPTR_mS+
                if (CPTR_mS==1000) CPTR_S++,CPTR_mS=0;
                  if (CPTR_S>=Valeur_Tempo)Fin_Tempo=1,Tempo_en_Cours=0,CPTR_S=0;
                }
}

/* fonction d'attente
************************/
void Init_Tempo(int Duree)
{
Tempo_en_Cours=1;
Valeur_Tempo = Duree;
Fin_Tempo=0;
CPTR_mS=0;
}
```

```
/*************************************************
*          FONCTION PRINCIPALE                  *
*************************************************/
main()
{
/*            INITIALISER
**********************************/
/* Définition des "directions" du port C "   */
Dir_PortC=0x07;
PortC=0x00;


/*  Configurer le port A    en mode "Discet Input Output"   (DIO)-> code $8  */
CFSR3=0x8888;       /*  CHA0 à CHA3 en mode "DIO"             */
CFSR2=0x8888;       /*  CHA4 à CHA7 en mode "DIO"             */
CFSR1=0x8888;       /*  CHA8 à CHA11 en mode "DIO"            */
CFSR0=0x8888;       /*  CHA12 à CHA15 en mode "DIO"           */

/* Définir les priorités */
CPR1=0xFFFF;        /* Tous les bits de PA en priorité haute  */
CPR0=0xFFFF;        /* Tous les bits de PB en priorité haute  */

// Pour base de temps
//******************
CPTR_mS=0;                              // initialisation compteurs millisecondes
CPTR_S=0;                               // initialisation compteurs secondes
SetVect(96,&irq_bt);                // mise en place de l'autovecteur
PITR = 0x0008;                      // Une interruption toutes les millisecondes
PICR = 0x0760;                      //  96 = 60H

// Pour grafcet , activation de l'étape 0
//*************************************************
Etat_grafcet=0x0001;                /*Initialisation du grafcet (Etape 0 activée) */
Active_Etat(1);                             // activation état 1 de l'étape 0
Appel_Memorise=0;                   // init memoire d'appel
Tempo_en_Cours=0;                   // init variables tempo
Fin_Tempo=0;
// Activation tempo de 10 S
Init_Tempo(10);                 /* Durée tempo en Seconde passée comme paramètre */

/*            BOUCLE PRINCIPALE
*************************************************/
do
{
 if (AE0==1) TE0(); // boucle de recherche de l'étape active
 if (AE1==1) TE1();
 if (AE2==1) TE2();
 if (AE3==1) TE3();
 if (AE4==1) TE4();
 if (AE5==1) TE5();
 if (AE6==1) TE6();
 if (AE7==1) TE7();
 if (AE8==1) TE8();
 if (AE9==1) TE9();
 if (AE10==1) TE10();
 if (AE11==1) TE11();

} while(1);             /* Fin de la boucle principale */
} //  Fin de la fonction principale

/*            FONCTIONS DE TRAITEMENT DES ETAPES
*************************************************************************/
void TE0(void)          /*Autorisation des voies principales(feux 1 et 2 au vert)*/
{
        if(Fin_Tempo==1) AE0=0, AE1=1,Active_Etat(2), Init_Tempo(3);
        /*  activation de l'état 2 en étape 1,  lancement de la tempo de 3 s de 'étape 1        */
}

void TE1(void)          /*le feu 1 passe à l'orange*/
{
        if(Fin_Tempo==1) AE1=0, AE2=1,Active_Etat(3), Init_Tempo(6);
        /* activation de l'état 3 en étape 2,  lancement de la tempo de 6 s de l'étape 2       */
}

void TE2(void)          /*Autorisation des voies 2 et 4 (feux 2 et 4 au vert)*/
{
        if(Fin_Tempo==1) AE2=0, AE3=1,Active_Etat(4), Init_Tempo(3);
        /* activation de l'état 4 en étape 3, lancement de la tempo de 3 s de l'étape 3       */
}

void TE3(void)          /*les feux 2 et 4 passent à l'orange*/
{
        if(Fin_Tempo==1) AE3=0, AE4=1,Active_Etat(7), Init_Tempo(8);
        /* activation de l'état 7 en étape 4, lancement de la tempo de 8 s de l'étape 4       */
```

```
}

void TE4(void)          /*Autorisation de la voie 5 (feux 5 au vert)*/
{
        if(Fin_Tempo==1) AE4=0, AE5=1,Active_Etat(8), Init_Tempo(3);
        /* activation de l'état 8 en étape 5,  lancement de la tempo de 3 s de l'étape 5          */
}

void TE5(void)          /*les feux 5 passent à l'orange*/
{
        if(Fin_Tempo==1) AE5=0, AE6=1,Active_Etat(1), Init_Tempo(10);
         /* activation de l'état 1 en étape 6, lancement de la tempo de 10 s de l'étape 6          */
}

void TE6(void)          /*Autorisation des voies principales(feux 1 et 2 au vert)*/
{
        if(Fin_Tempo==1) AE6=0, AE7=1,Active_Etat(9), Init_Tempo(3);
        /* activation de l'état 9 en étape 7,  lancement de la tempo de 3 s de l'étape 7          */
}

void TE7(void)          /*le feu 2 passe à l'orange*/
{
        if(Fin_Tempo==1) AE7=0, AE8=1,Active_Etat(5), Init_Tempo(6);
         /* activation de l'état 5 en étape 8, lancement de la tempo de 6 s de l'étape 8          */
}


void TE8(void)          /*Autorisation des voies 1 et 3 (feux 1 et 3 au vert)*/
{
        if(Fin_Tempo==1) AE8=0, AE9=1,Active_Etat(6), Init_Tempo(3);
        /* activation de l'état 6 en étape 9, lancement de la tempo de 3 s de l'étape              */
}

void TE9(void)          /*les feux 1 et 3 passent à l'orange*/
{
        if(Fin_Tempo==1) AE9=0, AE10=1,Active_Etat(7), Init_Tempo(
        /* activation de l'état 7 en étape 10, lancement de la tempo de 6                 pe 10    */
}

void TE10(void)         /*Autorisation de la voie 5 (feux 5 au vert) */
{
        if(Fin_Tempo==1) AE10=0, AE11=1,Active_Etat(3), Init_Tem          );
        /* activation de l'état 8 en étape 11,  lancement de la tempo de                de l'étape 1          */
}

void TE11(void)         /*les feux 5 passent à l'orang          */
{
        if(Fin_Tempo==1) AE11=0, AE0=1,Active_Etat(1), Init_Tempo(1          );
         /* activation de l'état 1 en étape 0, lancement de la tempo de          0 s de l'étape 0          */
}

   /*       Fin de la définition des fonctions de traitement des étapes
    ************************************************************************************************/


/* fonction d'affectation des sorties
*************************************************/

void Active_Etat(int j)             /*affectation des registres de sorties en fonction du N° de l'état à activer*/
{
                (i) /* j:numéro de l'état */
        {
        case 1 :    {HSRR0=0x          6;HSRR1=0x969A;PortC=0x04;break;}
        case 2 :    {HSRR0=0x69          ;HSRR1=0x96A6;PortC=0x02;break;}
                    =0x69          ;HSRR1=0x96A9;PortC=0x02;break;}
        case 4 :    {HSRR          0x699A;HSRR1=0x99A9;PortC=0x02;break;}
        case 5 :    {HSRR          0x69A5;HSRR1=0xAA5A;PortC=0x02;break;}
        case 6 :    {HSRR          =0x69A6;HSRR1=0x6A66;PortC=0x02;break;}
        case 7 :    {HSRR0=0x9AA6;HSRR1=0x9A69;PortC=0x03;break;}
        case 8 :    {HSRR0=0x66A6;HSRR1=0x9A69;PortC=0x02;break;}
        case 9 :    {HSRR0=0x69A6;HSRR1=0x999A;PortC=0x02;break;}
        default :   {HSRR0=0xAAAA;HSRR1=0xAAAA;PortC=0x00;break;}          // On éteint tout
        }
}

// FIN de la définition des fonctions
// FIN  du listing
```

TP 3 : CYCLE COMPLET AVEC TRAITEMENT DES APPELS PIETON MAIS SANS DETECTION VOITURES

3.1 Sujet

Objectif :	**Capacités complémentaires:** Etre capable d'acquérir des entrées qui imposent des ruptures de séquence. Etre capable de structurer la microprogrammation d'un grafcet comportant des ruptures de séquence.
Cahier des charges :	Le cycle avec bifurcations (voir TP précédent) doit pouvoir être rompu si un appel piéton intervient. S' il y a appui sur l'un des boutons d'appel, le cycle normal est interrompu afin de d'autoriser le passage des piétons. On passe alors dans un état où tous les feux « voitures » sont au rouge et les deux feux « piétons » au vert Cet état dure une dizaine de secondes. L'entrée dans cet état ne peut se faire qu'après un passage à l'orange des feux « voitures » qui étaient au vert. Les attentes seront réalisées par timer interne au micro-contrôleur

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,

Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 100 000

Câble de liaison USB, ou à défaut câble RS232, Réf : EGD 000 003

Alimentation AC/AC 8V, 1 A Réf : EGD000001,

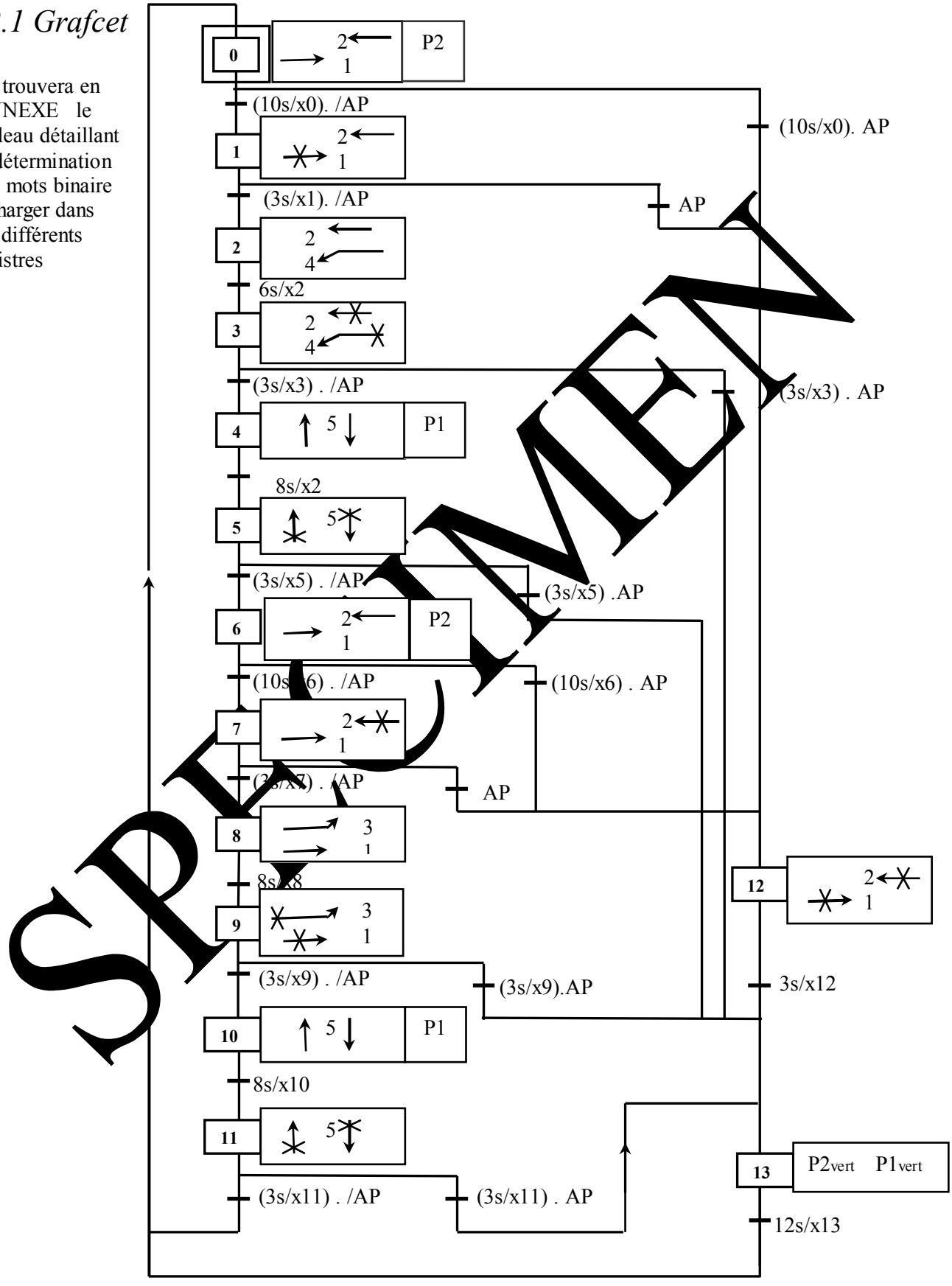
Carte feux de carrefour réf : EID 002 000,

Temps alloué : 4 heures

3.2 Eléments de solution

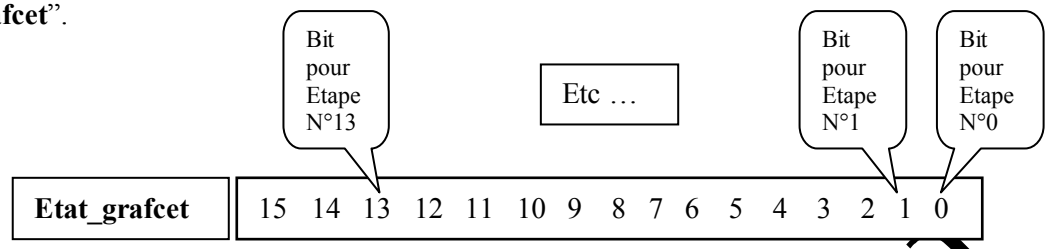
3.2.1 Grafcet

On trouvera en ANNEXE le tableau détaillant la détermination des mots binaire à charger dans les différents registres

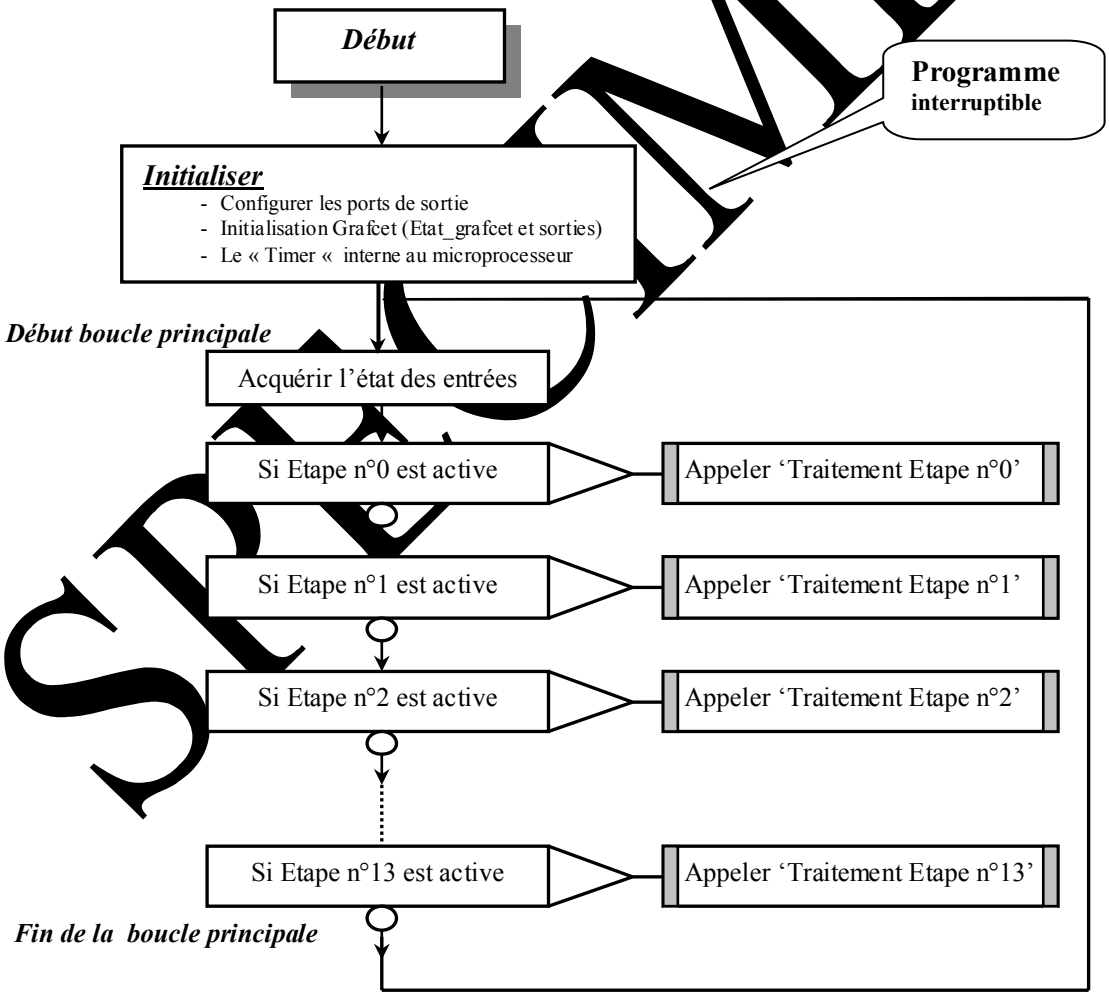


3.2.2 Organigramme de programmation du grafcet

Principe:
On affecte à chaque étape une variable binaire rangée dans une variable globale dont le label choisi est "**Etat_grafcet**".

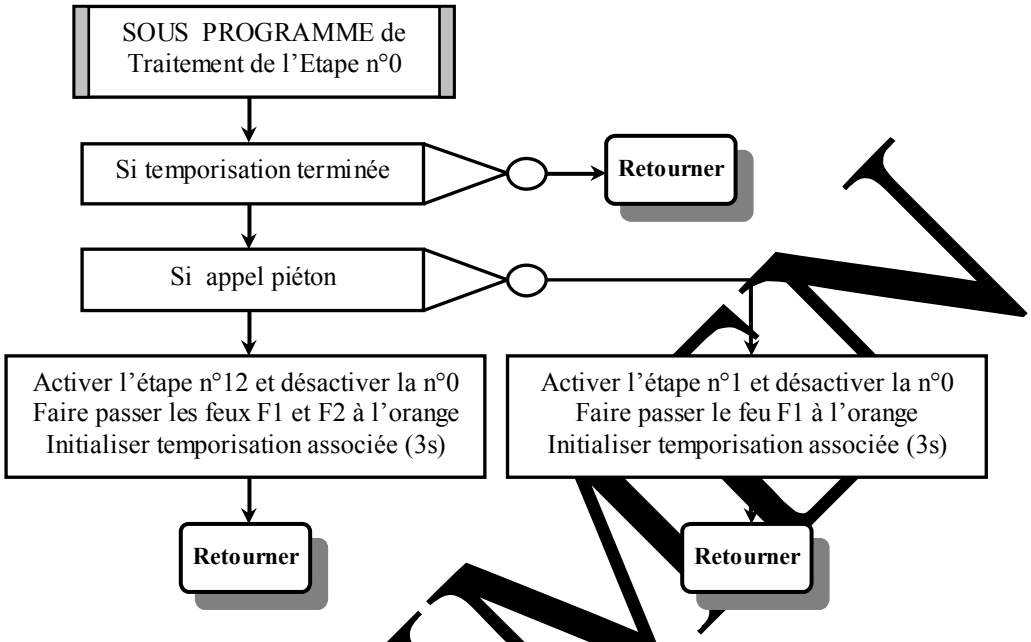


Lorsqu'une étape est active, le bit associé est mis à '1' logique. Naturellement, lorsqu'elle ne l'est pas, il est mis à '0'.
L'initialisation de la variable '**Etat_grafcet'** sera donc $0001 (valeur Hexadécimale).
La boucle principale du programme principal comporte une recherche de l'étape active où l'on teste successivement les bits de la variable '**Etat_grafcet'**.
Lorsque l'on trouve un bit à 1, c'est que l'étape correspondante est active. On appelle alors un sous programme où l'on teste les réceptivités avales, associée cette étape active.



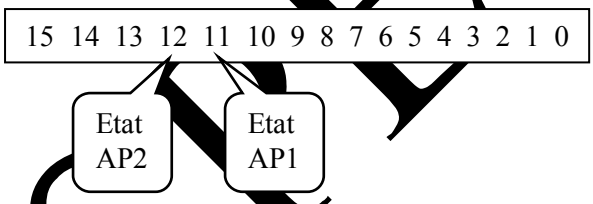
Dans un sous programme de traitement d'étape on teste les réceptivités avales.
Si l'une des réceptivités testée est vraie, on fait évoluer la variable 'Etat_grafcet' ainsi que les sortie associées.

Exemple pour le sous programme de traitement de l'étape 0 :



Acquisition des appels piétons

Les entrées associées aux appels piétons sont accessibles sur le port C : bits de rang 4 et 3.
On lit le registre de donnée du port C, appelé '**Port_C'** sur 16 bit. Les états du port C sont sur les 8 bit de poids fort.



Quand on appuie sur une touche, on lit un '0'.
On pourra effectuer les actions suivantes :

- Lire Port_C (un 'word')
- Faire un ET logique avec $1800
- Comparer le résultat avec $1800
- Si ce n'est pas égal, c'est qu'il y a eu un appel.

Valeurs des registres pour activations des sorties

N° Etape	CONTENU DES REGISTRES (en hexadécimal) HSRR0	HSRR1	Port C
0	69A6	969A	0400
1	69A6	96A6	0200
2	696A	96A9	0200
3	699A	99A9	0200
4	9AA6	9A69	0300
5	66A6	9A69	0200
6	69A6	969A	0400
7	69A6	999A	0200
8	69A5	AA5A	0200
9	69A6	6A66	0200
10	9AA6	9A69	0300
11	66A6	9A69	0200
12	69A6	99A6	0200
13	A9A6	9A69	0500

3.2.3 Programme en Assembleur A68xxx avec "timer"

```
        ********************************************************************************************
        *                          TP EID210 + FEU  DE  CARREFOUR                                  *
        ********************************************************************************************
        *    Cahier des Charges TP N° 3 :                                                          *
        *   ************************                                                               *
        *   -  Avec la prise en compte des appels piétons                                          *
        *   -  S'il n'y a pas d'appel pièton, il y a permutation régulière  des feux: voies principales  puis  Traversée 4   *
        *      puis voies secondaires  puis voies principales  puis Traversée 3   puis   voies secondaires ..etc   *
        *   -  Les présences véhicules ne sont pas gérés                                           *
        *   -  Le fonctionnement a été décrit à l'aide d'un grafcet                                *
        *   -  Les temporisations sont réalisées à l'aide du timer du 68332                        *
        *                                                                                          *
        *  NOM du FICHIER:     Feu_Carf_TP3.SRC                                                    *
        *****************                                                                          *
        ********************************************************************************************
        * Inclusion du fichier définissant les différents labels
        include EID210.def
****************************************
*       Déclaration des variables      *
****************************************
        section     var
COMPTEUR        ds.l    1
Etat_grafcet    ds.w    1
INDICATEUR      ds.b    1       * pour indiquer fin temporisation
MEM_AP ds.b     1               * pour MEMoire Appel Pièton
***********************************
*  Début du programme exécutable  *
***********************************
        section             code
*       INITIALISER
*************************
* Configurer le port A    en mode "Discet Input Output"   (DIO)->  a $8
DEBUT   move.w              #$8888,CFSR3            * CHA0 à CHA3 en mode "DIO"
        move.w              #$8888,CFSR2            * CHA4 à CHA7 en mode "DIO"
        move.w              #$8888,CFSR1            * CHA8 à CHA11 en mode "DIO"
        move.w              #$8888,CFSR0            * CHA12 à CHA15 en mode "DIO"
* Définir les priorités
        move.w              #$FFFF,CPR1             * Tous les bits de PA en priorité haute
        move.w              #$FFFF,CPR0             * Tous les bits de PB en priorité haute
* Configurer la base de temps
        move.l              #96,d0                  * 96 est le n° du vecteur d'interruption
        move.l              #it_1,a1                * it_1 est l'adresse de la fonction d'interruption
        asl.l               #2,d0
        add.l               #tab_vect,d0            * Initialiser la table des vecteurs
        move.l              d0,a0
        move.l              a1,(a0)
        move.l              #8000,COMPTEUR          * 8000*1mS = 8S
        move.b              #0,INDICATEUR           * de fin de comptage
        move.w              #$0008,PITR             * 1 interruption toutes les 1 ms
        move.w              #$0760,PICR
* Pour configurer le port C
        move.w              #$0700,DIR_Port_C       *  Les 3 bits lsb du port C  en sortie
* Initialisation du grafcet
*************************
*  Etape n°0 active à l'initialisation
        move.w              #$0001,Etat_grafcet     * Mémoire d'activation des étapes
*  Initialisation des actions associée à l'étape n°0
*  Autorisation voies principales (Feux 1 et 2 au vert)
        move.w              #$969A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w              #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w              #$0400,Port_C           *  Piétons 2 au VERT
        move.b              #0,MEM_AP               * Init MEMmoire Appel Pièton
***********************************
*        BOUCLE PRINCIPALE        *
***********************************
Deb_BP
* Lecture de l'état des entrées "Appels piétons"
        move.w              Port_C,d0
        andi.w              #$1800,d0               * Pour isoler les 2 bits d'appel piétons
        cmp.w               #$1800,d0
        beq                 Test_E0                 * Sortir si pas d'appui détecté
        move.b              #1,MEM_AP               * Mise à 1 mémoire appel pièton
```

```
* Boucle de recherche de l'étatpe active
Test_E0   cmp.w    #$0001,Etat_grafcet        * Tester si Etape n°0 est active
          bne      Test_E1
          bsr      T_E0                       * Vers traitement de l'étape 0
Test_E1   cmp.w    #$0002,Etat_grafcet        * Tester si Etape n°1 est active
          bne      Test_E2
          bsr      T_E1                       * Vers traitement de l'étape 1
Test_E2   cmp.w    #$0004,Etat_grafcet        * Tester si Etape n°2 est active
          bne      Test_E3
          bsr      T_E2                       * Vers traitement de l'étape 2
Test_E3   cmp.w    #$0008,Etat_grafcet        * Tester si Etape n°3 est active
          bne      Test_E4
          bsr      T_E3                       * Vers traitement de l'étape 3
Test_E4   cmp.w    #$0010,Etat_grafcet        * Tester si Etape n°4 est active
          bne      Test_E5
          bsr      T_E4                       * Vers traitement de l'étape 4
Test_E5   cmp.w    #$0020,Etat_grafcet        * Tester si Etape n°5 est active
          bne      Test_E6
          bsr      T_E5                       * Vers traitement de l'étape 5
Test_E6   cmp.w    #$0040,Etat_grafcet        * Tester si Etape n°6 est active
          bne      Test_E7
          bsr      T_E6                       * Vers traitement de l'étape 6
Test_E7   cmp.w    #$0080,Etat_grafcet        * Tester si Etape n°7est active
          bne      Test_E8
          bsr      T_E7                       * Vers traitement de l'étape 7
Test_E8   cmp.w    #$0100,Etat_grafcet        * Tester si Etape n°8 est active
          bne      Test_E9
          bsr      T_E8                       * Vers traitement de l'étape 8
Test_E9   cmp.w    #$0200,Etat_grafcet        * Tester si Etape n°9 est active
          bne      Test_E10
          bsr      T_E9                       * Vers traitement de l'étape 9
Test_E10  cmp.w    #$0400,Etat_grafcet        * Tester si Etape n°10 est active
          bne      Test_E11
          bsr      T_E10                      * Vers traitement de l'étape 10
Test_E11  cmp.w    #$0800,Etat_grafcet        * Tester si Etape n°11 est active
          bne      Test_E12
          bsr      T_E11                      * Vers traitement de l'étape 11
Test_E12  cmp.w    #$1000,Etat_grafcet        * Tester si Etape n°12 est active
          bne      Test_E13
          bsr      T_E12                      * Vers traitement de l'étape 12
Test_E13  cmp.w    #$2000,Etat_grafcet        * Tester si Etape n°13 est active
          bne      Test_Fin
          bsr      T_E13                      * Vers traitement de l'étape 13
* FIN de la boucle de recherche de l'étatpe active
Test_Fin  bra      Deb_BP                     * Aller au début de la boucle principale
*         Fin de la boucle principale
*         Fin du programme principal
**********************************************
**********************************************
*  SOUS PROGRAMME DE TRAITEMENT DES ETAPES   *
**********************************************
*         Traitement de l'étape n°0          *
**********************************************
T_E0      move.b     INDICATEUR,D2
          cmp.b      #$00,D2
          bne        T_E0_r                     * Sortir si la temporisation n'est pas terminée
          #0,MEM_AP                             * Pour savoir  si appels piétons
          bne        T_E0_1                     * sortir si appel pièton
          *On a la fin tempo sans appel pièton, donc on passe à l'étape 1 (Le Feu 1 passe à l'orange)
          move.w     #$96A6,HSRR1               *  Pour les Feux sur le port A (CH0 à 7)
          move.w     #$69A6,HSRR0               *  Pour les Feux sur le port B (CH8 à 15)
          move.w     #$0200,Port_C                 *  Piétons 2 passe au ROUGE
          * Passage à l'étape n°1
          move.w     #$0002,Etat_grafcet        * le bit de rang  1  passe à 1 et les autres sont  à 0

          move.l     #3000,COMPTEUR             *  Init tempo d'environ 3 secondes       3000*1mS = 3S
          move.b     #$00,INDICATEUR            * de fin de comptage
          rts                                   * Retour dans la boucle principal
        * On a la fin tempo et un appel pièton  -> Passage à l'étape n°12 , les Feux 1 et 2  passent  à l'orange
T_E0_1    move.b     #0,MEM_AP                  * Mise à 0 mémoire appel pièton
          move.w     #$1000,Etat_grafcet        * Le bit de rang 12  passe à 1 et les autres sont à 0
          move.w     #$99A6,HSRR1               *  Pour les Feux sur le port A (CH0 à 7)
          move.w     #$69A6,HSRR0                  *  Pour les Feux sur le port B (CH8 à 15)
          move.w     #$0200,Port_C                 *  Piétons 2 au ROUGE
```

```
        move.l      #3000,COMPTEUR      *  Init tempo d'environ 3 secondes 3000*1mS = 3S
        move.b      #$00,INDICATEUR     * de fin de comptage
T_E0_r  rts         *Fin du traitement de l'étape  n°0, retour dans la boucle principale
*******************************************************************************
*       Traitement de l'étape n°1       *
*************************************
T_E1    cmp.b       #0,MEM_AP           * Pour savoir  si appels piétons
        bne         T_E1_1              * sortir si appel pièton
        move.b  I   NDICATEUR,D2        * Test si fin temporisation
        cmp.b       #01,D2
        bne         T_E1_r              * Sortir si la temporisation n'est pas terminée
        * On a la fin tempo sans appel pièton, donc on passe à l'étape 2
        * Le Feu 4 passe au vert
        move.w      #$96A9,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w      #$696A,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w      #$0200,Port_C       *  Piétons 2 passe au ROUGE
        *  Init tempo d'environ 6 secondes
        move.l      #6000,COMPTEUR      * 6000*1mS = 6S
        move.b      #$00,INDICATEUR     * de fin de comptage
        * Passage à l'étape n°2
        move.w      #$0004,Etat_grafcet * le bit de rang  2  passe à 1 et les autres sont à 0
        rts                             Retour dans la boucle principal
T_E1_1  * On a un appel pièton, donc passage à l'étape 12  * Passage à l'étape n°12 , le Feu  2  passe à l'orange
        move.b      #0,MEM_AP           * Mise à 0 mémoire appel pi
        move.w      #$1000,Etat_grafcet * Le bit de rang 12  passe à 1 et les autres sont à 0
        move.w      #$99A6,HSRR1        *  Pour les Feux sur le port A (CH0 à
        move.w      #$69A6,HSRR0        *  Pour les Feux sur le B (CH8 à 15
        move.w      #$0200,Port_C       *  Piétons 2 passe au ROUGE
        move.l      #3000,COMPTEUR      * Init tempo 3000*1mS = 3S
        move.b      #$00,INDICATEUR     * de fin de comptage
T_E1_r  rts                             * Fin du traitement de l'étape  n°1, retour dans la boucle principale
*******************************************************************************
*       Traitement de l'étape n°2       *
*************************************
T_E2    move.b      INDICATEUR,D2
        cmp.b       #01,D2
        bne         T_E2_r              * Sortir si la temporisation n'est pas terminée
        * La temporisation est terminée alors passage à l'étape 3
        move.w      #$0008,Etat_grafcet * Le bit de rang  passe à 1 et les autres sont à 0
        * Les Feux 2 et 4 passent à l'orange
        move.w      #$99A9,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w      #$699A,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w      #$0200,Port_C       *  Piétons 1 passe au rouge
        move.l      #3000,COMPTEUR      * Init tempo 3000*1mS = 3S
        move.b      #$00,INDICATEUR     * de fin de comptage
T_E2_r  rts                             * Fin du traitement de l'étape  n°2, retour dans la boucle principale
*******************************************************************************
*       Traitement de l'étape n°3       *
*************************************
T_E3    move.b      INDICATEUR,D2
        cmp.b       #01,D2
        bne         T_E3_r              * Sortir si la temporisation n'est pas terminée
        *           #$0,MEM_AP          * Pour savoir  si appels piétons
        *bne        T_E3_1              * sortir si appel pièton
        * On a la fin tempo  et pas d'appel pièton,  donc on passe à l'étape 4   *  feux 5 passent au vert
        move.w      #$0010,Etat_grafcet * Le bit de rang 4  passe à 1 et les autres sont à 0
        move.w      #$9A69,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w      #$9AA6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w      #$0300,Port_C       *  Feux Piétons P 1 au vert
        move.l      #8000,COMPTEUR      * Init tempo 8000*1mS = 8S
        move.b      #$00,INDICATEUR     * de fin de comptage
        rts                             *retour dans la boucle principale
        * On a la fin tempo  et  appel pièton donc on passe à l'étape 13
T_E3_1  move.w      #$2000,Etat_grafcet   * Le bit de rang 13  passe à 1 et les autres sont à 0
        move.w      #$9A69,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w      #$A9A6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w      #$0500,Port_C       *  Feux Piétons  au vert
        move.b      #0,MEM_AP           * Mise à 0 mémoire appel pièton
        move.l      #8000,COMPTEUR      * Init tempo d'environ 10000*1mS = 10S
        move.b      #$00,INDICATEUR     * de fin de comptage
T_E3_r  rts                             * Fin du traitement de l'étape  n°3, retour dans la boucle principale
*******************************************************************************
```

```
*          Traitement de l'étape n°4          *
***************************************
T_E4       move.b     I          NDICATEUR,D2
           cmp.b                 #01,D2
           bne                   T_E4_r                  * Sortir si la temporisation n'est pas terminée
           * Fin tempo, donc on passe à l'étape 5 , Feux 5 passent à l'orange
           move.w                #$0020,Etat_grafcet     * Le bit de rang  5  passe à 1 et les autres sont à 0
           move.w                #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
           move.w                #$A6A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
           move.w                #$0200,Port_C           *  Piétons  au rouge
           move.l                #3000,COMPTEUR          * Init tempo 3000*1mS = 3S
           move.b                #$00,INDICATEUR         * de fin de comptage
T_E4_r     rts                   * Fin du traitement de l'étape  n°4, retour dans la boucle principale
*****************************************************************************
*          Traitement de l'étape n°5          *
***************************************
T_E5       move.b                INDICATEUR,D2
           cmp.b                 #01,D2
           bne                   T_E5_r                  * Sortir si la temporisation est terminée
           cmp                   #$0,MEM_AP              * Pour savoir  si appels piétons
           bne                   T_E5_1                  * sortir si appel pièton
           * On a fin tempo sans appel pièton donc on passe à l'étape 6 (1 et 2 au vert)
           move.w                #$0040,Etat_grafcet     * Le bit de rang 6  passe à 1 et l[illegible] autres sont à 0
           move.w                #$969A,HSRR1            *  Pour les Feux sur le port A [illegible] à 7)
           move.w                #$69A6,HSRR0            *  Pour les Feux sur le port B (CH[illegible]15)
           move.w                #$0200,Port_C           *  Piétons  au rouge
           move.l                #10000,COMPTEUR         * Init tempo 10000*1m[illegible]10S
           move.b                #$00,INDICATEUR         * de fin de comptage
           rts
           * On a la fin tempo avec un appel pièton donc on passe à l'étape 13
T_E5_1     move.w                #$2000,Etat_grafcet     * Le bit de rang 13  passe à[illegible] et les [illegible]res sont à 0
           move.w                #$9A69,HSRR1            *  Pour [illegible] le port A (CH0 à
           move.w                #$A9A6,HSRR0            *  Pour les F[illegible]ux sur [illegible]t B (CH8 à 15)
           move.w                #$0500,Port_C           *  [illegible]eux Piétons au vert
           move.b                #0,MEM_AP               *  [illegible] à 0 mémoire appel pièton
           move.l                #12000,COMPTEUR         * Init[illegible]po 12000*[illegible]S = 12S
           move.b                #$00,INDICATEUR         * de fin [illegible]omptage
T_E5_r     rts                   * Fin du traitement de [illegible]étape  n°5, retour da[illegible] boucle principale
*****************************************************************************
*          Traitement de l'étape n°6          *
***************************************
T_E6       move.b                INDICATEUR,D2
           cmp.b                 #01,D2
           bne                   T_E6_r                  * Sort[illegible]si la temporisation n'est pas terminée
           cmp                   #$[illegible]MEM_AP      [illegible]ur savoir  si appels piétons
           bne                   T_[illegible]           * sortir si appel pièton
           * On a la fin tempo et pas d[illegible]el p[illegible]étons, on pas[illegible]e à l'étape 7  (Feu 2 passe à l'orange)
           move.w                #$0080,[illegible]grafcet  * Le bit de rang 7  passe à 1 et les autres sont à 0
           move.w                #[illegible]099A,HS[illegible]  *  Pour les Feux sur le port A (CH0 à 7)
           move.w                #[illegible]9A6,HSR[illegible]  *  Pour les Feux sur le port B (CH8 à 15)
           move.w                #[illegible]200,Port_C  *  Feux Piétons au Rouge
           move.l                [illegible]3000,COMPTEUR  * Init tempo 30000*1mS = 3S
           [illegible]           #$00,INDICATEUR         * de fin de comptage
           rts
           * On a la fin tempo et [illegible]appel piétons, On passe à l'étape 12 (feux 1 et 2 à l'orange)
T_E6_1     move.w                #$[illegible]0,Etat_grafcet  * Le bit de rang 12  passe à 1 et les autres sont à 0
           move.w                #$099A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
           move.w                #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
           move.w                #$0200,Port_C           *  Feux Piétons  au ROUGE
           move.b                #0,MEM_AP               * Mise à 0 mémoire appel pièton
           move.l                #3000,COMPTEUR          * Init tempo 3000*1mS = 3S
           move.b                #$00,INDICATEUR         * de fin de comptage
T_E6_r     rts                   * Fin du traitement de l'étape  n°6, retour dans la boucle principale
*****************************************************************************
*          Traitement de l'étape n°7          *
***************************************
T_E7       move.b                INDICATEUR,D2
           cmp.b                 #01,D2
           bne                   T_E7_r                  * Sortir si la temporisation n'est pas terminée
           cmp                   #$0,MEM_AP              * Pour savoir  si appels piétons
           bne                   T_E7_1                  * sortir si appel pièton
```

```
        * On la fin tempo sans appel piétons, on passe à l'étape 8, Feux 1 et 3 passent au vert
          move.w        #$0100,Etat_grafcet     * Le bit de rang 8 passe à 1 et les autres sont à 0
          move.w        #$AA5A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
          move.w        #$69A5,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
          move.w        #$0200,Port_C           *  Piétons au ROUGE
          move.l        #8000,COMPTEUR          * Init tempo 8000*1mS = 8S
          move.b        #$00,INDICATEUR         * de fin de comptage
          rts
          * On a un appel piétons, On passe à l'étape 12 (Feux 1 et 2 à l'orange)
T_E7_1    move.w        #$1000,Etat_grafcet     * Le bit de rang 12  passe à 1 et les autres sont à 0
          move.w        #$99A6,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
          move.w        #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
          move.w        #$0200,Port_C           *  Feux Piétons au ROUGE
          move.b        #0,MEM_AP               * Mise à 0 mémoire appel pièton
          move.l        #3000,COMPTEUR          * Init tempo 3000*1mS = 3S
          move.b        #$00,INDICATEUR         * de fin de comptage
T_E7_r    rts           * Fin du traitement de l'étape  n°7, retour dans la boucle principale
*******************************************************************************
***********************************
*       Traitement de l'étape n°8       *
***********************************
T_E8      move.b        INDICATEUR,D2
          cmp.b         #01,D2
          bne           T_E8_r                  * Sortir si la temporisation n'est pas terminée
          * Fin tempo , donc on passe à l'étape 9 , Feux 1 et 3 passent à l'orange
          move.w        #$0200,Etat_grafcet     * Le bit de rang 9  passe à 1 et les autres sont à 0
          move.w        #$6AA6,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
          move.w        #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
          move.w        #$0200,Port_C           *  Piétons au rouge
          move.l        #3000,COMPTEUR          * Init tempo 3000*1mS = 3S
          move.b        #$00,INDICATEUR         * de fin de comptage
T_E8_r    rts           * Fin du traitement de l'étape  n°8, retour dans la boucle principale
*******************************************************************************
***********************************
*       Traitement de l'étape n°9       *
***********************************
T_E9      move.b    INDICATEUR,D2
          cmp.b         #01,D2
          bne           T_E9_r                  * Sortir si la temporisation n'est pas terminée
          cmp           #$0,MEM_AP              * Pour savoir s'il y a appels piétons
          bne           T_E9_1                  * sortir s'il y a appel pièton
          * On la fin tempo sans appel piétons, on passe à l'étape 10, Feux 5 passent au vert
          move.w        #$0400,Etat_grafcet     * Le bit de rang 9 et les autres sont à 0
          move.w        #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
          move.w        #$A6A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
          move.w        #$0200,Port_C           *  Piétons 1  au Vert
          move.l        #0800,COMPTEUR          * Init tempo 8000*1mS = 8S
          move.b        #$00,INDICATEUR         * de fin de comptage
          rts           * Retour dans la boucle de recherche étape active
          * On a la fin tempo et un appel piéton on passe à l'étape 13 (Feux voitures au rouge)
T_E9_1    move.w        #1000,Etat_grafcet      * Le bit de rang 13  passe à 1 et les autres sont à 0
          move.w        #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
          move.w        #$A9A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
          move.w        #$0500,Port_C           *  Feux Piétons au vert
          move.b        #0,MEM_AP               * Mise à 0 mémoire appel pièton
          move.l        #12000,COMPTEUR         * Init tempo 12000*1mS = 12S
          move.b        #$00,INDICATEUR  * de fin de comptage
T_E9_r    rts           * Fin du traitement de l'étape  n°9, retour dans la boucle principale
*******************************************************************************************************
***********************************
*       Traitement de l'étape n°10      *
***********************************
T_E10     move.b    INDICATEUR,D2
          cmp.b         #01,D2
          bne           T_E10_r                 * Sortir si la temporisation n'est pas terminée
          * Fin tempo, donc on passe à l'étape 11 , Feux 5 passent àl'orange
          move.w        #$0800,Etat_grafcet     * Le bit de rang 11  passe à 1 et les autres sont à 0
          move.w        #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
          move.w        #$A6A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
          move.w        #$0200,Port_C           *  Piétons au rouge
          move.l        #3000,COMPTEUR          * Init tempo 3000*1mS = 3S
          move.b        #$00,INDICATEUR         * de fin de comptage
T_E10_r   rts           * Fin du traitement de l'étape  n°10, retour dans la boucle principale
```

```
************************************
*       Traitement de l'étape n°11         *
************************************
T_E11   move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             T_E11_r                 * Sortir si la temporisation est terminée
        cmp             #$0,MEM_AP              * Pour savoir  si appels piétons
        bne             T_E11_1                 * sortir si appel pièton
        * On a fin tempo sans appel pièton donc on passe à l'étape 0 (1 et 2 au vert)
        move.w          #$0001,Etat_grafcet     * Le bit de rang 0  passe et les autres sont à 0
        move.w          #$969A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons  au rouge
        move.l          #10000,COMPTEUR         * Init tempo 10000*1mS = 10S
        move.b          #$00,INDICATEUR         * de fin de comptage
        rts
        * On a la fin tempo avec un appel pièton donc on passe à l'étape 13
T_E11_1 move.w          #$2000,Etat_grafcet     * Le bit de rang 13  passe à 1 et les autres sont à 0
        move.w          #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0500,Port_C           *  Feux Piétons  au vert
        move.b          #0,MEM_AP               * Mise à 0 mémoire appel pièton
        move.l          #10000,COMPTEUR         * Init tempo 10000*1mS = 10S
        move.b          #$00,INDICATEUR         * de fin de comptage
T_E11_r rts                     * Fin du traitement de l'étape  n°11, retour dans la boucle principale
************************************
*       Traitement de l'étape n°12         *
************************************
T_E12   move.b    INDICATEUR,D2
        cmp.b           #01,D2
        bne             T_E12_r                 * Sortir si la temporisation est terminée
        * On a la fin tempo sans appel pièton donc on passe à l'étape 13 (Feux piétons au rouge)
        move.w          #$2000,Etat_grafcet     * Le bit de rang 13 passe à 1 et les autres sont à 0
        move.w          #$9A69,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$A9A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0500,Port_C           *  Feux Piétons  au vert
        move.l          #10000,COMPTEUR         * Init tempo 10000*1mS = 10S
        move.b          #$00,INDICATEUR         * de fin de comptage
T_E12_r rts                     * Fin du traitement de l'étape  n°12, retour dans la boucle principale
************************************
*       Traitement de l'étape n°13         *
************************************
T_E13   move.b    INDICATEUR,D2
        cmp.b           #01,D2
        bne             T_E13_r                 * Sortir si la temporisation est terminée
        * On a fin tempo sans appel pièton donc on passe à l'étape 0 (1 et 2 au vert)
        move.w          #$0001,Etat_grafcet     * Le bit de rang 0  passe et les autres à 0
        move.w          #$969A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons  au rouge
        move.l          #8000,COMPTEUR          * Init tempo 8000*1mS = 8S
        move.b          #$00,INDICATEUR         * de fin de comptage
T_E13_r rts                     * Fin du traitement de l'étape  n°13, retour dans la boucle principale
************************************************
*       FONCTION D' INTERRUPTION                *
*       associée à la base de temps             *
************************************************
it_bt   sub.l           #$00000001,COMPTEUR
        cmp.l           #$00000000,COMPTEUR
        bne             it_ret                  * Retourner si pas égal à 0
        move.b          #$01,INDICATEUR  * C'est la fin tempo
        move.l          #$5000,COMPTEUR * réinit tempo
it_ret  rte                                     * Retour d'interruption
*  Fin de la fonction d'interruption
************************************
*       Fin du fichier source assembleur
********************************************
        end
```

3.2.4 Programmation en C

```
/*****************************************************************************************
*            TP SUR EID210 avec FEU  DE  CARREFOUR
*****************************************************************************************
*
*  Cahier des charges  TP N°3
*            - prise en compte des appels piétons
*            - si il n'y a pas d'appels piétons, permutation régulière des feux :
*              voies principales puis traverséé 4 puis voies secondaires puis
*              voies principales puis traversée 3 puis voies secondaires... etc
*            - les présences des véhicules ne sont pas gerées
*  NOM du FICHIER:  Feu_Carf_TP3.C
*****************************************************************************************/
/* Liste des fichiers à inclure */
//*******************************
#include "Cpu_reg.h"
#include "EID210_reg.h"
#include "Structure_Donnees.h"
#include "feux_carrefour.h"

/* Declaration des variables
***********************************/
void TE0(void),TE1(void),TE2(void),TE3(void),TE4(void),TE5(void),TE6(void),TE7(void);
void TE8(void),TE9(void),TE10(void),TE11(void),TE12(void),TE13(void);
void Active_Etat(int);
void Init_Tempo(int);
int Etat,Duree,CPTR_mS,CPTR_S,Valeur_Tempo;
unsigned char Fin_Tempo,Tempo_en_Cours,Appel_Memorise;

/* fonction d'attente
***********************************/
void Init_Tempo(int Duree)
{
Tempo_en_Cours=1;
Valeur_Tempo = Duree;
Fin_Tempo=0;
CPTR_mS=0;
}

/* fonction d'interruption pour tempo  (Toutes les mS)
*************************************************************/
void irq_bt()
{           if (Tempo_en_Cours==1)
                  { CPTR_mS++;
            if (CPTR_mS ==1000) CPTR_S++,CPTR_mS=0,
                   if (CPTR_S==Valeur_Tempo)Fin_Tempo=1,Tempo_en_Cours=0,CPTR_S=0;
                  }
}

/*****************************
*      FONCTION PRINCIPALE                  *
*****************************************/
main()
{
/*          INITIALISER
*****************************/
/* Définition des "directions" du port C "  */
Dir_PortC=0x07;
PortC=0;

/*  Configurer le port A   en mode "Discet Input Output"   (DIO)-> code $8  */
CFSR3=0x8888;       /*  CHA0 à CHA3 en mode "DIO"            */
CFSR2=0x8888;       /*  CHA4 à CHA7 en mode "DIO"            */
CFSR1=0x8888;       /*  CHA8 à CHA11 en mode "DIO"           */
CFSR0=0x8888;       /*  CHA12 à CHA15 en mode "DIO"          */

/* Définir les priorités */
CPR1=0xFFFF;        /* Tous les bits de PA en priorité haute  */
CPR0=0xFFFF;        /* Tous les bits de PB en priorité haute  */
// Pour base de temps
//**********************
CPTR_mS=0;
CPTR_S=0;
//CPTR_prec=0;
SetVect(96,&irq_bt);              // mise en place de l'autovecteur
PITR = 0x0008;                    // Une interruption toutes les millisecondes
PICR = 0x0760;                    //  96 = 60H

// Pour grafcet , activation de l'étape 0
//***************************************
Etat_grafcet=0x0001;              /*Initialisation du grafcet (Etape 0 activée) */
```

```
Active_Etat(1);
Appel_Memorise=0;
Tempo_en_Cours=0;
Fin_Tempo=0;
// Activation tempo de 10 S
Init_Tempo(10);                 /* Durée tempo en Seconde passée comme paramètre */

/*              BOUCLE PRINCIPALE
***********************************************/
do
{
 if (AE0==1) TE0();
 if (AE1==1) TE1();
 if (AE2==1) TE2();
 if (AE3==1) TE3();
 if (AE4==1) TE4();
 if (AE5==1) TE5();
 if (AE6==1) TE6();
 if (AE7==1) TE7();
 if (AE8==1) TE8();
 if (AE9==1) TE9();
 if (AE10==1) TE10();
 if (AE11==1) TE11();
 if (AE12==1) TE12();
 if (AE13==1) TE13();
} while(1);             /* Fin de la boucle principale */
} //  Fin de la fonction principale

/*              FONCTIONS DE TRAITEMENT DES ETAPES
*************************************************************************/

void TE0(void)                  /*Autorisation des voies principales(feux 1 et 2 au vert)*/
{
        if(Fin_Tempo==1)
                { if(Appel_Memorise==1)AE0=0,AE12=1,Active_Etat(11),Init_Tempo(3),Appel_Memorise=0;
                  else AE1=1,AE0=0,Active_Etat(2),Init_Tempo(3);
                }

        else if ((AP1==0)||(AP2==0)) Appel_Memorise=1;
}
void TE1(void)                  /*le feu 1 passe à l'orange*/
{
        if ((AP1==0)||(AP2==0)) Appel_Memorise=1;
        if (Appel_Memorise==1) AE1=0, AE12=1, Active_Etat(11),Init_Tempo(3), Appel_Memorise=0;
        else if (Fin_Tempo==1) AE1=0, AE2=1, Active_Etat(3),Init_Tempo(6);

}
void TE2(void)                  /*Autorisation des voies 2 et 4 (feux 2 et 4 au vert)*/
{
        if(Fin_Tempo==1) AE2=0, AE3=1,Active_Etat(9), Init_Tempo(3);
        if ((AP1==0)||(AP2==0)) Appel_Memorise=1;                       // memorisation pieton pour étape suivante
}
void TE3(void)                  /*les feux 2 et 4 passent à l'orange*/
{
        if(Fin_Tempo==1)
                { if(Appel_Memorise==1)AE3=0,AE13=1,Active_Etat(15),Init_Tempo(12),Appel_Memorise=0;
                  else AE4=1,AE3=0,Active_Etat(4),Init_Tempo(8);


        else if ((AP1==0)||(AP2==0)) Appel_Memorise=1;
}
void TE4(void)                  /*Autorisation de la voie 5 (feux 5 au vert)*/
{
        if(Fin_Tempo==1) AE4=0, AE5=1,Active_Etat(8), Init_Tempo(3);
                ||(AP2==0)) Appel_Memorise=1;           // memorisation pieton pour étape suivante
}
void TE5(void)                  /*les feux 5 passent à l'orange*/
{
        if(Fin_Tempo==1)
                { if(Appel_Memorise==1)AE5=0,AE13=1,Active_Etat(15),Init_Tempo(12),Appel_Memorise=0;
                  else AE6=1,AE5=0,Active_Etat(1),Init_Tempo(10);
                }
        else if ((AP1==0)||(AP2==0)) Appel_Memorise=1;
}
```

```
void TE6(void)                  /*Autorisation des voies principales(feux 1 et 2 au vert)*/
{
        if(Fin_Tempo==1)
                { if(Appel_Memorise==1)AE6=0,AE12=1,Active_Etat(11),Init_Tempo(3),Appel_Memorise=0;
                  else AE7=1,AE6=0,Active_Etat(9),Init_Tempo(3);
                }

        else if ((AP1==0)||(AP2==0)) Appel_Memorise=1;
}
void TE7(void)                  /*le feu 2 passe à l'orange*/
{
        if ((AP1==0)||(AP2==0)) Appel_Memorise=1;
        if (Appel_Memorise==1) AE7=0, AE12=1,Active_Etat(11),Init_Tempo(3), Appel_Memorise=0;
        else if (Fin_Tempo==1) AE7=0, AE8=1,Active_Etat(5),Init_Tempo(8);

}
void TE8(void)                  /*Autorisation des voies 1 et 3 (feux 1 et 3 au vert)*/
{
        if(Fin_Tempo==1) AE8=0, AE9=1,Active_Etat(6), Init_Tempo(3);
        if ((AP1==0)||(AP2==0)) Appel_Memorise=1;                       // memorisation pieton pour étape suivante
}
void TE9(void)                  /*les feux 1 et 3 passent à l'orange*/
{
        if(Fin_Tempo==1)
                { if(Appel_Memorise==1)AE9=0,AE13=1,Active_Etat(15),Init_Tempo(12),Appel_Memorise=0;
                  else AE10=1,AE9=0,Active_Etat(7),Init_Tempo(8);
                }

        else if ((AP1==0)||(AP2==0)) Appel_Memorise=1;
}
void TE10(void)                 /*Autorisation de la voie 5 (feux 5 au vert)*/
{
        if(Fin_Tempo==1) AE10=0, AE11=1,Active_Etat(8), Init_Tempo(3);
        if ((AP1==0)||(AP2==0)) Appel_Memorise=1;                       // memorisation pieton pour étape suivante
}
void TE11(void)                 /*les feux 5 passent à l'orange*/
{
        if(Fin_Tempo==1)
                { if(Appel_Memorise==1)AE11=0,AE12=1,Active_Etat(15),Init_Tempo(12),Appel_Memorise=0;
                  else AE0=1,AE11=0,Active_Etat(9),Init_Tempo(10);
                }

        else if ((AP1==0)||(AP2==0)) Appel_Memorise=1;
}
void TE12(void)                 /*les feux 1 et 2 passent à l'orange*/
{
        if(Fin_Tempo==1) AE12=0, AE13=1,Active_Etat(15), Init_Tempo(12);
}
void TE13(void)                 /*tous les feux piétons sont au vert */
{
        if(Fin_Tempo==1) AE13=0, AE0=1,Active_Etat(10), Init_Tempo(10);
}
   /*       Fin de la définition des étapes
    ***************************************************************/

/* fonction d'affectation des sorties
*****************************************************************/
void Active_Etat(int j)         /*Affectation des registres de sorties en fonction du N° de l'état à activer*/
{
        switch(j)   /* j:numéro de l'état */
        {
        case 1 :    {HSRR0=0x69A6;HSRR1=0x969A;PortC=0x04;break;}
        case 2 :    {HSRR0=0x69A6;HSRR1=0x96A6;PortC=0x02;break;}
        case 3 :    {HSRR0=0x696A;HSRR1=0x96A9;PortC=0x02;break;}
        case 4 :    {HSRR0=0x699A;HSRR1=0x99A9;PortC=0x02;break;}
        case 5 :    {HSRR0=0x69A5;HSRR1=0xAA5A;PortC=0x02;break;}
        case 6 :    {HSRR0=0x69A6;HSRR1=0x6A66;PortC=0x02;break;}
        case 7 :    {HSRR0=0x9AA6;HSRR1=0x9A69;PortC=0x03;break;}
        case 8 :    {HSRR0=0x66A6;HSRR1=0x9A69;PortC=0x02;break;}
        case 9 :    {HSRR0=0x69A6;HSRR1=0x999A;PortC=0x02;break;}
        case 11:    {HSRR0=0x69A6;HSRR1=0x99A6;PortC=0x02;break;}
        case 15:    {HSRR0=0xA9A6;HSRR1=0x9a69;PortC=0x05;break;}
        default :   {HSRR0=0xAAAA;HSRR1=0xAAAA;PortC=0x00;break;}           // On éteint tout
        } // Fin du "switch"
} // Fin de la fonction d'affectation des sorties
```

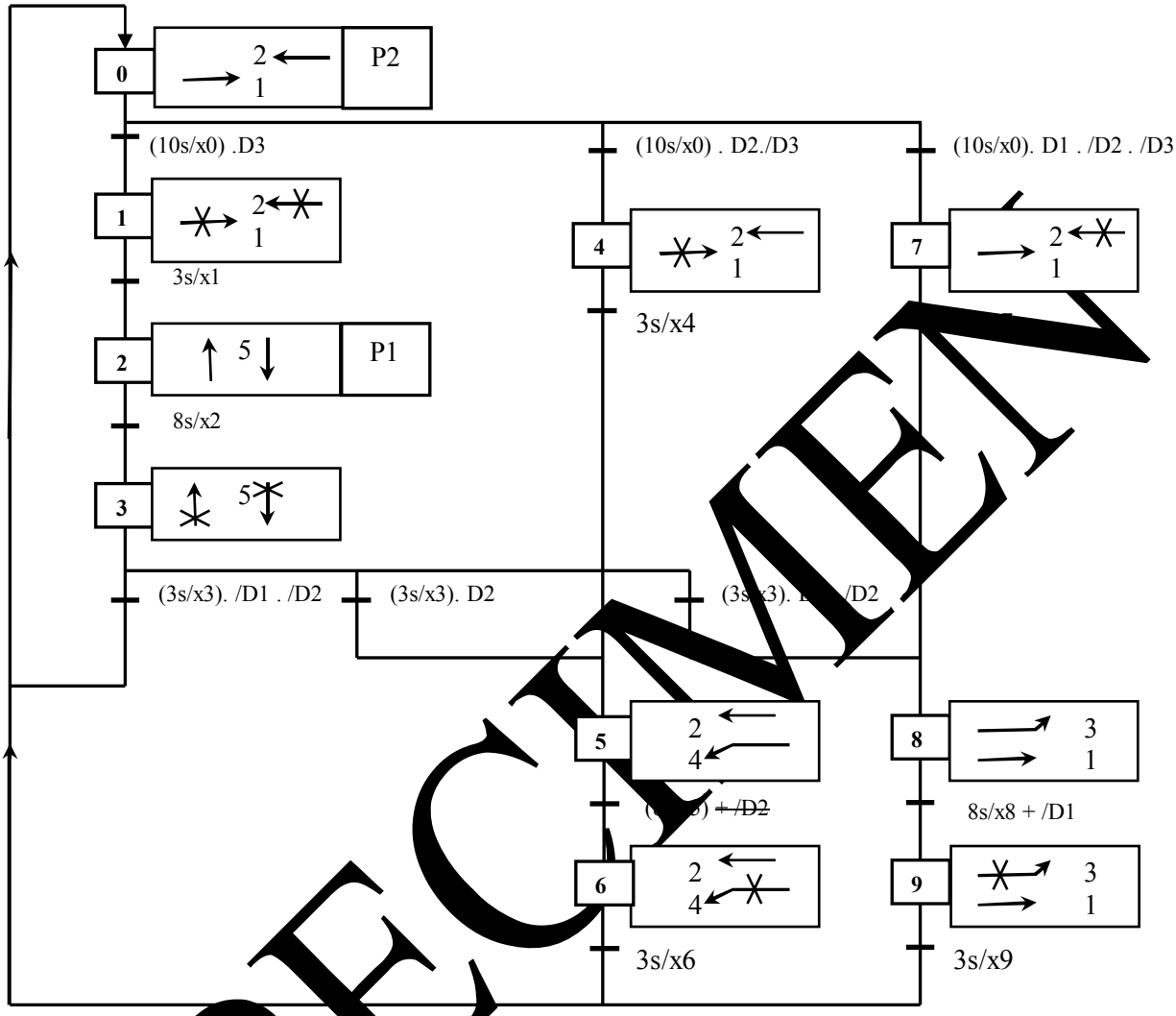
TP 4 : CYCLE AVEC PRISES EN COMPTE DES DETECTIONS VOITURES MAIS SANS TRAITEMENT DES APPELS PIETONS

4.1 Sujet

Objectif :	Capacités complémentaires: Etre capable de représenter par un grafcet un cahier des charges complexe. Etre capable de micro programmer un grafcet présentant de nombreuses divergences et convergences de type OU.
Cahier des charges :	Dans l'état normal, seuls les deux voies principales sont autorisées (Feux F1 et F2 au vert), les voies piétons n°2 également (P2 au vert). Les voitures qui tournent à droite doivent respecter la priorité des piétons. On ne sort de cet état que lorsqu'une voiture est détectée sur l'une des voies secondaires (capteur D3 sur voies 5) ou sur les voies de bifurcations (capteur D1 sur voie 3 et capteur D2 sur voie 4). A partir de l'état normal décrit précédemment il peut y avoir détection sur l'un (ou plusieurs) des 3 capteurs D1, D2, D3. On imposera la prise en compte avec la hiérarchie suivante : P(D3) > P(D2) > P(D1) ➔ 1- Si une voiture actionne D3, les feux F1 et F2 passent à l'orange puis, au bout de 3S, passent au rouge en même temps que les feux F5 passent au vert. Cet état se prolonge 8S avant que les feux F5 passent à l'orange pendant 3S et que l'on retrouve l'état initial (où l'on devra rester au moins 10S). ➔ 2- Si une voiture actionne D2 (alors que D3 n'est pas activé), le feu F1 passe à l'orange puis, au bout de 3S, passe au rouge en même temps que F4 passe au vert. Cet état se prolonge 8S avant que le feu F4 passe à l'orange pendant 3S et que l'on retrouve l'état initial (Où l'on devra rester au moins 10S). ➔ 3- Si une voiture actionne D1 (alors que D3 et D2 ne sont pas activés), le feu F2 passe à l'orange puis, au bout de 3S, passe au rouge en même temps que F3 passe au vert. Cet état se prolonge 8S avant que le feu F3 passe à l'orange pendant 3S et que l'on retrouve l'état initial (Où l'on devra rester au moins 10S). ➔ A partir de l'état ou les feux F5 sont à l'orange depuis 3S, si une voiture actionne D2, on rejoint la condition repérée 2-. ➔ A partir de l'état ou les feux F5 sont à l'orange depuis 3S, si une voiture actionne D2 (alors que D1 n'est pas actionné, on rejoint la condition repérée 3-.

4.2 Eléments de solution

4.2.1 Grafcet



Valeurs des registres pour activation des sorties
Voir en Annexe le tableau des valeurs en binaire

Acquisition des états détecteurs présence voitures
Les entrées associées aux capteurs de présence voiture sont accessibles sur le port C :
- bits de rang 5 pour détecteur repéré D1
- bits de rang 6 pour détecteur repéré D2
- bits de rang 7 pour détecteur repéré D3

On lit le registre de donnée du port C (label '**Port_C'** sur 16 bit. Les états du port C étant sur les 8 bit de poids fort.
Quand une voiture est détectée, on lit un '0'.
On pourra effectuer les actions suivantes :
- Lire Port_C (un 'word')
- Faire un ET logique avec $E000
- Comparer le résultat avec $E000
- Si ce n'est pas égal, c'est qu'il y a une présence voiture.

N° Etape	CONTENU DES REGISTRES (en hexadécimal) HSRR0	HSRR1	Port C
0	69A6	969A	0400
1	69A6	99A6	0200
2	9AA6	9A69	0300
3	66A6	9A69	0200
4	69A6	96A6	0200
5	696A	96A9	0200
6	699A	96A9	0200
7	69A6	999A	0200
8	69A5	AA5A	0200
9	69A6	6A5A	0200

4.2.2 Programme en Assembleur A68xxx avec timer

```
        ****************************************************************
        *                    TP EID210 + FEU DE CARREFOUR              *
        ****************************************************************
        *    Cahier des charge:                                        *
        *   *****************                                          *
        *   - Avec la détection de présences voitures                  *
        *   - Les appels piétons ne sont pas gérés                     *
        *   - Le fonctionnement a été décrit à l'aide d'un grafcet     *
        *   - Les temporisations sont réalisées à l'aide du timer du 68332 *
        *                                                               *
        *  NOM du FICHIER: Feu_Carf_TP4.SRC                           *
        **********************                                         *
        ****************************************************************
        * Inclusion du fichier définissant les différents labels
        include 68332.def
************************************
*       Déclaration des variables  *
************************************
        section     var
COMPTEUR        ds.l    1
Etat_grafcet    ds.w    1
INDICATEUR      ds.b    1
************************************
*  Début du programme exécutable   *
************************************
        section             code
*       INITIALISER
*************************
* Configurer le port A   en mode "Discet Input Output"  (DIO)-> code $8
DEBUT   move.w          #$8888,CFSR3        * CHA0 à CHA3 en mode "DIO"
        move.w          #$8888,CFSR2        * CHA4 à CHA7 en mode "DIO"
        move.w          #$8888,CFSR1        * CHA8 à CHA11 en mode "DIO"
        move.w          #$8888,CFSR0        * CHA12 à CHA15 en mode "DIO"
* Définir les priorités
        move.w          #$FFFF,CPR1         * Tous les bits de PA en priorité haute
        move.w          #$FFFF,CPR0         * Tous les bits de PB en priorité haute
* Configurer la base de temps
        move.l          #96,d0              * 96 est le n° du vecteur d'interruption
        move.l          #it_bt,a1           * f_it-bt est l'adresse de la fonction d'interruption
        asl.l           #2,d0
        add.l           #tab_vect,d0        * Initialiser la table des vecteurs
        move.l          d0,a0
        move.l          a1,(a0)
        move.l          #1000,COMPTEUR      * 1000*1mS = 1S
        move.b          #$00,INDICATEUR     * de fin de comptage
        move.w          #$0008,PICR         * 1 interruption toutes les 1 ms
        move.w          #$0760,PICR
* Initialisation du grafcet
*************************
*  Etape n° 0 active à l'initialisation
        move.w          #$0001,Etat_grafcet     * Mémoire d'activation des étapes
*  Initialisation des actions associées à l'étape n°0
*  Autorisation voies principales (Feux 1 et 2 au vert)
        move.w          #$69A,HSRR1         *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0400,Port_C       *  Piétons 2 au VERT
* Configurer en sorties les 3 bits du port QS où sont connectées les diodes
*  Pour visualisation de l'activation du grafcet
        move.w          #$0070,PQSCTR       * 3 sorties sur LED
        move.w          #$0070,d3           * Pour visu  n° étape 0
************************************
*       BOUCLE PRINCIPALE
************************************
Deb_BP
* Lecture de l'état des entrées
        move.w          Port_C,d0
        and.w           #$E000,d0           * Pour isoler les bits de présence voiture
        eor.w           #$E000,d0           * Pour avoir des 1 si présence voiture
* Boucle de recherche de l'étatpe active
Test_E0 cmp.w           #$0001,Etat_grafcet     * Tester si Etape n°0 est active
        bne             Test_E1
        bsr             T_E0                * Vers traitement de l'étape 0
```

```
Test_E1   cmp.w     #$0002,Etat_grafcet     * Tester si Etape n°1 est active
          bne       Test_E2
          bsr       T_E1                    * Vers traitement de l'étape 1
Test_E2   cmp.w     #$0004,Etat_grafcet     * Tester si Etape n°2 est active
          bne       Test_E3
          bsr       T_E2                    * Vers traitement de l'étape 2
Test_E3   cmp.w     #$0008,Etat_grafcet     * Tester si Etape n°3 est active
          bne       Test_E4
          bsr       T_E3                    * Vers traitement de l'étape 3
Test_E4   cmp.w     #$0010,Etat_grafcet     * Tester si Etape n°4 est active
          bne       Test_E5
          bsr       T_E4                    * Vers traitement de l'étape 4
Test_E5   cmp.w     #$0020,Etat_grafcet     * Tester si Etape n°5 est active
          bne       Test_E6
          bsr       T_E5                    * Vers traitement de l'étape 5
Test_E6   cmp.w     #$0040,Etat_grafcet     * Tester si Etape n°6 est active
          bne       Test_E7
          bsr       T_E6                    * Vers traitement de l'étape 6
Test_E7   cmp.w     #$0080,Etat_grafcet     * Tester si Etape n°7est active
          bne       Test_E8
          bsr       T_E7                    * Vers traitement de l'étape 7
Test_E8   cmp.w     #$0100,Etat_grafcet     * Tester si Etape n°8 est active
          bne       Test_E9
          bsr       T_E8                    * Vers traitement de l'étape 8
Test_E9   cmp.w     #$0200,Etat_grafcet     * Tester si Etape n°9 est active
          bne       Visu_EG
          bsr       T_E9                    * Vers traitement de l'étape 9

* Visualisation Etat GRAFCET sur port "QS" (LEDs D12, D11, D10  de l'unité centrale)
Visu_EG   move.w    d3,PORTQS               * Charger sur le port QS
          bra       Deb_BP
*         Fin de la boucle principale
*************************************
*         Fin du programme principal
*************************************
*****************************************************
*  SOUS PROGRAMME DE TRAITEMENT DES ETAPES          *
*****************************************************
*         Traitement de l'étape n°0       *
*************************************
T_E0      move.b    INDICATEUR,D2
          cmp.b     #01,D2
          bne       T_E0_S                  * Sortir si la temporisation n'est pas terminée
          cmp       #$0000,d0
          bne T_E0_1                        * sortir si on n'a pas  /D3et /D2 et D1
          * on a la combinaison  /D3et /D2et D1
          * Le Feu 2 passe à l'orange
          move.w    #$99A6,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
          move.w    #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
          move.w    #$0200,Port_C           *  Piétons 2 passe au ROUGE
          *  Init tempo d'environ 3 secondes
          move.l    #3000,COMPTEUR   * 3000*1mS = 3S
          move.b    #00,INDICATEUR   * de fin de comptage
          * Passage à l'étape n°7
          move.w    #$0080,Etat_grafcet     * le bit de rang  7  passe à 1 et le 0 à 0
          move.w    #$0000,d3               * Pour visu  n° étape 7
          * Retour dans la boucle principal
          rts
T_E0_1    and       #$C000,d0               * On masque l'entrée D1
          cmp       #$4000,d0               * D2 est prioritaire sur D1
          bne       T_E0_2                  * sortir si on n'a pas  /D3et D2
          * on a la combinaison  /D3et D2
          * Passage à l'étape n°4 , le Feu 1 passe à l'orange
          move.w    #$0010,Etat_grafcet     * Le bit de rang 4  passe à 1 et celui 0 à 0
          move.w    #$0030,d3               * Pour visu  n° étape 4
          move.w    #$96A6,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
          move.w    #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
          move.w    #$0200,Port_C           *  Piétons 2 passe au ROUGE
          *  Init tempo d'environ 3 secondes
          move.l    #3000,COMPTEUR          * 3000*1mS = 3S
          move.b    #00,INDICATEUR          * de fin de comptage
          *  Retour dans la boucle principal
          rts
```

```
T_E0_2  and     #$8000,d0       * On masque l'entrée D2 (D1 déjà masqué
        cmp     #$8000,d0       * D3 est prioritaire sur D2 et sur D1
        bne     T_E0_3          * sortir si on n'a pas D3
        * On a D3
        * Passage à l'étape n°1, les Feux 1 et 2 passent à l'orange
        move.w  #$0002,Etat_grafcet     * Le bit de rang 1 passe à 1 et celui 0 à 0
        move.w  #$0060,d3               * Pour visu n° étape 1
        move.w  #$99A6,HSRR1            * Pour les Feux sur le port A (CH0 à 7)
        move.w  #$69A6,HSRR0            * Pour les Feux sur le port B (CH8 à 15)
        move.w  #$0200,Port_C           * Piétons 2 passe au ROUGE
        * Init tempo d'environ 3 secondes
        move.l  #3000,COMPTEUR  * 3000*1mS = 3S
        move.b  #$00,INDICATEUR * de fin de comptage
        * Retour dans la boucle principal
T_E0_3  rts             *Fin du traitement de l'étape n°0, retour dans la boucle principale
********************************************************************************
*       Traitement de l'étape n°1       *
*****************************************
T_E1    move.b  INDICATEUR,D2
        cmp.b   #01,D2
        bne     T_E1_1          * Sortir si la temporisation n'est pas terminée
        * La temporisation est terminée alors passage à l'étape 2
        move.w  #$0004,Etat_grafcet     * Le bit de rang 2 passe à 1 et celui 1 à 0
        move.w  #$0050,d3               * Pour visu n° étape 2
        * Les Feux 5 passent au vert
        move.w  #$9A69,HSRR1            * Pour les Feux sur le port A (CH0 à 7)
        move.w  #$9AA6,HSRR0            * Pour les Feux sur le port B (CH8 à 15)
        move.w  #$0300,Port_C           * Piétons 1 passe au VERT
        * Init tempo d'environ 6 secondes
        move.l  #6000,COMPTEUR          * 6000*1mS = 6S
        move.b  #$00,INDICATEUR         * de fin de comptage
T_E1_1  rts             * Fin du traitement de l'étape n°1, retour dans la boucle principale
********************************************************************************
*       Traitement de l'étape n°2       *
*****************************************
T_E2    move.b  INDICATEUR,D2
        cmp.b   #01,D2
        bne     T_E2_1          * Sortir si la temporisation n'est pas terminée
        * La temporisation est terminée alors passage à l'étape 3
        move.w  #$0008,Etat_grafcet     * Le bit de rang 3 passe à 1 et celui 2 à 0
        move.w  #$0040,d3               * Pour visu n° étape 3
        * Les Feux 5 passent à l'orange
        move.w  #$9A69,HSRR1            * Pour les Feux sur le port A (CH0 à 7)
        move.w  #$99A6,HSRR0            * Pour les Feux sur le port B (CH8 à 15)
        move.w  #$0200,Port_C           * Piétons 1 passe au rouge
        * Init tempo d'environ 3 secondes
        move.l  #3000,COMPTEUR  * 3000*1mS = 3S
        move.b  #$00,INDICATEUR * de fin de comptage
T_E2_1  rts             * Fin du traitement de l'étape n°2, retour dans la boucle principale
********************************************************************************
*       Traitement de l'étape n°3       *
*****************************************
T_E3    move.b  INDICATEUR,D2
        cmp.b   #01,D2
        bne     T_E3_3          * Sortir si la temporisation n'est pas terminée
        move.w  #$8000,d0       * On masque l'entrée D3
        cmp     #$2000,d0
        bne T_E3_1              * sortir si on n'a pas /D2 et D1
        * On a D1 et /D2 donc on va à l'étape 8 , feux 1 et 3 passent au vert
        move.w  #$0100,Etat_grafcet     * Le bit de rang 8 passe à 1 et celui 3 à 0
        move.w  #$AA5A,HSRR1            * Pour les Feux sur le port A (CH0 à 7)
        move.w  #$69A5,HSRR0            * Pour les Feux sur le port B (CH8 à 15)
        move.w  #$0200,Port_C           * Piétons au rouge
        * Init tempo d'environ 3 secondes
        move.l  #3000,COMPTEUR          * 3000*1mS = 3S
        move.b  #$00,INDICATEUR         * de fin de comptage
        rts                             *retour dans la boucle principale
```

```
T_E3_1  and             #$4000,d0           * On masque l'entrée D1
        cmp             #$4000,d0
        bne T_E3_2                          * sortir si on n'a pas  D2
        *  On a D2 donc on va à l'étape 5, feux 2 et 4 passent au vert
        move.w          #$0020,Etat_grafcet * Le bit de rang 5  passe à 1 et celui 3 à 0
        move.w          #$0020,d3           * Pour visu  n° étape 5
        move.w          #$96A9,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$696A,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C       *  Piétons  au rouge
        *  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR      * 3000*1mS = 3S
        move.b          #$00,INDICATEUR     * de fin de comptage
        rts                                 *retour dans la boucle principale
T_E3_2  * On n'a pas D1 ni D2, donc on retourne à l'étape 0, Feux 1 et 2 passent au vert
        move.w          #$0001,Etat_grafcet * Le bit de rang 0  passe à 1 et celui 3 à 0
        move.w          #$0070,d3           * Pour visu  n° étape 0
        move.w          #$969A,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0400,Port_C       *  Piétons 2 au VERT
        *  Init tempo d'environ 10 secondes
        move.l          #10000,COMPTEUR * 10000*1mS = 10S
        move.b          #$00,INDICATEUR  * de fin de comptage
T_E3_3  rts                 * Fin du traitement de l'étape  n°3, retour dans la boucle principale
*****************************************************************************
*       Traitement de l'étape n°4       *
*****************************************
T_E4    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             T_E4_2              * Sortir si la temporisation n'est pas terminée
        * Fin tempo, donc on passe à l'étape 5 , Feux 2 et 4 passent au vert
        move.w          #$0020,Etat_grafcet * Le bit de rang 5  passe à 1 et celui 3 à 0
        move.w          #$0020,d3           * Pour visu  n° étape 05
        move.w          #$96A9,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$696A,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C       *  Piétons  au rouge
        *  Init tempo d'environ 6 secondes
        move.l          #6000,COMPTEUR      * 6000*1mS = 6S
        move.b          #$00,INDICATEUR     * de fin de comptage
T_E4_2  rts                 * Fin du traitement de l'étape  n°4, retour dans la boucle principale
*****************************************************************************
*       Traitement de l'étape n°5       *
*****************************************
T_E5    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        beq             T_E5_2              * Sortir si la temporisation est terminée
T_E5_1  and             #$4000,d0           * On sélectionne l'entrée D2
        cmp             #$0000,d0
        bne             T_E5_3              * Sortir si D2 n'est pas repassé à 0
        * Fin tempo ou D2 repassé à 0, donc on passe à l'étape 6 , Feu 4 passe à l'orange
T_E5_2  move.w          #$0040,Etat_grafcet * Le bit de rang 6  passe à 1 et celui 5 à 0
        move.w          #$0010,d3           * Pour visu  n° étape 6
        move.w          #$96A9,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$699A,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C       *  Piétons  au rouge
        *  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR      * 3000*1mS = 3S
        move.b          #$00,INDICATEUR     * de fin de comptage
T_E5_3  rts                 * Fin du traitement de l'étape  n°5, retour dans la boucle principale
*****************************************************************************
*       Traitement de l'étape n°6       *
*****************************************
T_E6    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             T_E6_1              * Sortir si la temporisation n'est pas terminée
        * On retourne à l'étape 0, Feux 1 et 2 passent au vert
        move.w          #$0001,Etat_grafcet * Le bit de rang 0  passe à 1 et celui 3 à 0
        move.w          #$0070,d3           * Pour visu  n° étape 0
        move.w          #$969A,HSRR1        *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0        *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C       *  Piétons 2  au Vert
        *  Init tempo d'environ 10 secondes
        move.l          #10000,COMPTEUR * 10000*1mS = 10S
        move.b          #$04,INDICATEUR     * de fin de comptage
T_E6_1  rts                 * Fin du traitement de l'étape  n°6, retour dans la boucle principale
*****************************************************************************
```

```
*       Traitement de l'étape n°7       *
*****************************************
T_E7    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             T_E7_1                  * Sortir si la temporisation n'est pas terminée
        * On passe à l'étape 8, Feux 1 et 3 passent au vert
        move.w          #$0100,Etat_grafcet     * Le bit de rang 08 passe à 1 et celui 7 à 0
        move.w          #$0070,d3               * Pour visu  n° étape 0  (au lieu de 8)
        move.w          #$AA5A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A5,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons  au ROUGE
        *  Init tempo d'environ 6 secondes
        move.l          #6000,COMPTEUR   * 8000*1mS = 6S
        move.b          #$00,INDICATEUR  * de fin de comptage
T_E7_1  rts             * Fin du traitement de l'étape  n°7, retour dans la boucle principale
*******************************************************************************
*       Traitement de l'étape n°8       *
*****************************************
T_E8    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             T_E8_1                  * Sortir si la temporisation n'est pas terminée
        bra             T_E5_2
T_E8_1  and             #$2000,d0               * On sélectionne l'entrée D1
        cmp             #$0000,d0
        bne             T_E8_3                  * Sortir si D2 n'est pas passé à 0
        * Fin tempo OU D1=0, donc on passe à l'étape 9 , Feu 3 passe à l'orange
T_E8_2  move.w          #$0200,Etat_grafcet     * Le bit de rang 9  passe à 1 et celui 8 à 0
        move.w          #$0070,d3               * Pour visu  n° étape 0  (au lieu de 9)
        move.w          #$6A5A,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons au rouge
        *  Init tempo d'environ 3 secondes
        move.l          #3000,COMPTEUR          * 3000*1mS = 3S
        move.b          #$00,INDICATEUR         * de fin de comptage
T_E8_3  rts             * Fin du traitement de l'étape n°8, retour dans la boucle principale
*******************************************************************************
*       Traitement de l'étape n°9       *
*****************************************
T_E9    move.b          INDICATEUR,D2
        cmp.b           #01,D2
        bne             T_E9_1                  * Sortir si la temporisation n'est pas terminée
        * On retourne à l'étape 0, Feux 1 et 2 passent au vert
        move.w          #$0001,Etat_grafcet     * Le bit de rang 0  passe à 1 et celui 3 à 0
        move.w          #$0070,d3               * Pour visu  n° étape 0
        move.w          #$96A5,HSRR1            *  Pour les Feux sur le port A (CH0 à 7)
        move.w          #$69A6,HSRR0            *  Pour les Feux sur le port B (CH8 à 15)
        move.w          #$0200,Port_C           *  Piétons 2  au Vert
        *  Init tempo d'environ 10 secondes
        move.l          #10000,COMPTEUR * 10000*1mS = 10S
        move.b      #$00,INDICATEUR  * de fin de comptage
T_E9_1  rts             * Fin du traitement de l'étape  n°6, retour dans la boucle principale
*******************************************************************************
*       FONCTION D'INTERRUPTION         *
*       à la base de temps              *
*****************************************
it_bt   sub.l           #$00000001,COMPTEUR
        cmp.l           #$00000000,COMPTEUR
        bne             it_ret                  * Retourner si pas égal à 0
        move.b          #$01,INDICATEUR         * C'est la fin tempo
        *move.l         #1000,COMPTEUR          * Réinit tempo
it_ret  rte                                     * Retour d'interruption
*  Fin de la fonction d'interruption
*********************************
*       Fin du fichier source assembleur
*********************************
        end
```

4.2.3 Programme en C

```
/*****************************************************************************************************
*                                TP SUR EID210 avec FEU  DE  CARREFOUR                                *
*****************************************************************************************************
*     Cahier des charge TP N°4:                                                                     *
*    *********************************                                                              *
*              - Avec la detection de présence voitures                                             *
*              - Les appels piétons ne sont pas gerés                                               *
*              - description du fonctionnement selon le grafcet du recueil de TP                   *
*              - utilisation du "timer"                                                             *
*                                                                                                   *
*                                                                                                   *
*  NOM du FICHIER:  Feu_Carf_TP4.C                                                                  *
*                                                                                                   *
*****************************************************************************************************/
/* Liste des fichiers à inclure */
//*********************************
#include "Cpu_reg.h"
#include "EID210_reg.h"
#include "Structures_Donnees.h"
#include "feux_carrefour.h"

/* Declaration des variables
*********************************/
void TE0(void),TE1(void),TE2(void),TE3(void),TE4(void),TE5(void),TE6(void),TE7(void);
void TE8(void),TE9(void);
void Active_Etat(int);
void Init_Tempo(int);
int Etat,Duree,CPTR_mS,CPTR_S,Valeur_Tempo;
unsigned char Fin_Tempo,Tempo_en_Cours,Voiture_1_Memorise,Voiture_2_Memorise,Voiture_3_Memorise;

/* fonction d'attente
************************/
void Init_Tempo(int Duree)
{
Tempo_en_Cours=1;
Valeur_Tempo = Duree;
Fin_Tempo=0;
CPTR_mS=0;
}

/* fonction d'interruption pour tempo  (Toutes les ms)
*****************************************************/
void irq_bt()
{       if (Tempo_en_Cours=
                { CPTR_mS++,
                  if (CPTR_mS >=
                  if (
                }
}

/*****************************************
*      FONCTION PRINCIPALE                *
*****************************************/
main()
{
/*          INITIALISATIONS
*****************/
/* Définition des "directions" du port C "   */
Dir_PortC=0x07;
PortC=0x00;

/*  Configurer le port A en mode "Discet Input Output"   (DIO)-> code $8  */
CFSR3=0x8888;       /*  CHA0 à CHA3 en mode "DIO"              */
CFSR2=0x8888;       /*  CHA4 à CHA7 en mode "DIO"              */
CFSR1=0x8888;       /*  CHA8 à CHA11 en mode "DIO"             */
CFSR0=0x8888;       /*  CHA12 à CHA15 en mode "DIO"            */

/* Définir les priorités */
CPR1=0xFFFF;        /* Tous les bits de PA en priorité haute  */
CPR0=0xFFFF;        /* Tous les bits de PB en priorité haute  */

// Pour base de temps
//******************
CPTR_mS=0;
CPTR_S=0;
SetVect(96,&irq_bt);            // mise en place de l'autovecteur
PITR = 0x0008;                  // Une interruption toutes les millisecondes
PICR = 0x0760;                  //  96 = 60H
```

```
// Pour grafcet , activation de l'étape 0
//**************************************
Etat_grafcet=0x0001;              /*Initialisation du grafcet (Etape 0 activée) */
 Active_Etat(1);

Voiture_1_Memorise=0;
Voiture_2_Memorise=0;
Voiture_3_Memorise=0;
Tempo_en_Cours=0;
Fin_Tempo=0;
// Activation tempo de 10 S
Init_Tempo(10);                   /* Durée tempo en Seconde passée comme paramètre */


/*              BOUCLE PRINCIPALE
************************************************/
do
{
 if (AE0==1) TE0();
 if (AE1==1) TE1();
 if (AE2==1) TE2();
 if (AE3==1) TE3();
 if (AE4==1) TE4();
 if (AE5==1) TE5();
 if (AE6==1) TE6();
 if (AE7==1) TE7();
 if (AE8==1) TE8();
 if (AE9==1) TE9();

} while(1);               /* Fin de la boucle principale */
} //  Fin de la fonction principale

/*              FONCTIONS  DE TRAITEMENT DES ETAPES
*****************************************************************************/
void TE0(void)                          /*Autorisation des voies prin      1 et 2 a    vert)*/
{
        if ((Fin_Tempo==1)&&((Voiture_1_Memorise==1)||(Voi    e_2_Memor    ==1)||(V      _3_Memorise==1)))
                { if (Voiture_3_Memorise==1)
                        { AE0=0,AE1=1,Active_Etat(11),Init_    po(3);
                          Voiture_3_Memorise=0,Vo    ure_2_Me    ise=0,Voitu    1_Memorise=0;
                        }
                 else
                        { if (Voiture_2_Mem    rise==1)
                                { AE4=1    E0=0,Active_Etat(2),Init_Te    po(3);
                                  Voitur    2_Memorise=0,Voitu    e_1_Memorise=0,Voiture_3_Memorise=0;
                                }
                                 else
                                { if (Voiture    Memorise==1)
                                        { A    AE0=0    ctive_Etat(9),Init_Tempo(3);
                                          Voitu    _Memorise=0,Voiture_2_Memorise=0,Voiture_3_Memorise=0;
                                        }
                                }
                        }

        else { if (D1==0) V    ure_1_Mem    =1;  // memorisation presence voiture
                if (    ==0) V    ure_2_Memo    se=1;
                if (D    0) Voiture_3_Memorise=1;
             }
}
void TE    (void)                        /*les feux 1 et 2 passent à l'orange*/
{
                    AE1=0, AE2=1,Active_Etat(7), Init_Tempo(8);

}
void TE2(void)                          /*Autorisation de la voie 5 (feux 5 au vert)*/
{
        if(Fin_Tem    o==1) AE2=0, AE3=1, Active_Etat(8),Init_Tempo(3);
}
void TE3(void)                          /*les feux 5 passent à l'orange*/
{
        if (Fin_Tempo==1)
                { if(Voiture_2_Memorise==1) AE3=0,AE5=1,Active_Etat(3),Init_Tempo(8),Voiture_2_Memorise=0,Voiture_1_Memorise=0;
                  else if (Voiture_1_Memorise==1) AE8=1,AE3=0,Active_Etat(5),Init_Tempo(8),Voiture_1_Memorise=0;
                          else AE0=1,AE3=0,Active_Etat(1),Init_Tempo(10);
                }

        else {if (D1==0) Voiture_1_Memorise=1;
              if (D2==0) Voiture_2_Memorise=1;
             }
}
```

```
void TE4(void)                          /*le feu 1 passe à l'orange*/
{

        if (Fin_Tempo==1) AE4=0, AE5=1,Active_Etat(3), Init_Tempo(8);
}
void TE5(void)                          /*Autorisation des voies 2 et 4 (feux 2 et 4 au vert)*/
{
        if ((Fin_Tempo==1) /* || (D2==1) */ ) AE5=0, AE6=1,Active_Etat(13), Init_Tempo(3);
}
void TE6(void)                          /*le feu 4 passe à l'orange*/
{
        if(Fin_Tempo==1) AE6=0, AE0=1,Active_Etat(1), Init_Tempo(10);
}
void TE7(void)                          /*le feu 2 passe à l'orange*/
{
        if(Fin_Tempo==1) AE7=0, AE8=1,Active_Etat(5), Init_Tempo(8);

}
void TE8(void)                          /*Autorisation des voies 1 et 3 (feux 1 et 3 au vert)*/
{
        if((Fin_Tempo==1) /* || (D1==1) */ ) AE8=0, AE9=1,Active_Etat(10), Init_Tempo(3);
}
void TE9(void)                          /*le feu 3 passe à l'orange*/
{
        if(Fin_Tempo==1) AE9=0, AE0=1, Active_Etat(1),Init_Tempo(10);
}

  /*    Fin de la définition des etapes
   **********************************************/

/* fonction d'affectation des sorties
**********************************************/

void Active_Etat(int j)             /*Affectation des registres de sorties de PortC
          en fonction de l'état     des étapes*/
{
        switch(j)   /* j:numéro de l'état */
        {
        case 1 :  {HSRR0=0x69A6;HSRR1=0x969A;PortC=0x04;break;}
        case 2 :  {HSRR0=0x69A6;HSRR1=0x96A6;PortC=0x02;break;}
        case 3 :  {HSRR0=0x696A;HSRR1=0x96A9;PortC=0x02;break;}
        case 5 :  {HSRR0=0x69A5;HSRR1=0xAA6A;PortC=0x02;break;}
        case 7 :  {HSRR0=0x9AA6;HSRR1=0x9969;PortC=0x03;break;}
        case 8 :  {HSRR0=0x66A6;HSRR1=0x9969;PortC=0x02;break;}
        case 9 :  {HSRR0=0x69A6;HSRR1=0x99AA;PortC=0x02;break;}
        case 10 : {HSRR0=0x69A6;HSRR1=0x6AA9;PortC=0x02;break;}
        case 11:  {HSRR0=0x69A6;HSRR1=0x99A9;PortC=0x02;break;}
        case 13 : {HSRR0=0x699A;HSRR1=0x96A5;PortC=0x02;break;}
        default :  {HSRR0=0xAAAA;HSRR1=0xAAAA;PortC=0x00;break;}          // On éteint tout
        }
}
```

ANNEXE

ANNEXE 1 Fichier de définitions pour programme en Assembleur

Nom du fichier EID210.def

```
nolist
MONITEUR        EQU     $400            * retour au moniteur
* Pour le micro-contrôleur 68332
*  Définition des adresses des registres du module QSM
PORTQS  EQU     $FFFC14
PQSCTR  EQU     $FFFC16
SCCR1           EQU     $FFFC0A
SCCR0           EQU     $FFFC08
SCSR            EQU     $FFFC0C
SCDR            EQU     $FFFC0E
TDRE            EQU     $0100
RDRF            EQU     $0040
RAF             EQU     $0020
* SIM
PITR            EQU     $FFFA24
PICR            EQU     $FFFA22
* définition des registres du module TPU
TRAMMCR         EQU     $FFFB00
TRAMTST         EQU     $FFFA04
TRAMBAR         EQU     $FFFA20
TPUMCR EQU      $FFFE00
TCR             EQU     $FFFE02
CFSR0           EQU     $FFFE0C
CFSR1           EQU     $FFFE0E
CFSR2           EQU     $FFFE10
CFSR3           EQU     $FFFE12
HSQR0           EQU     $FFFE14
HSQR1           EQU     $FFFE16
HSRR0           EQU     $FFFE18
HSRR1           EQU     $FFFE1A
CPR0            EQU     $FFFE1C
CPR1            EQU     $FFFE1E
CTRL_TPU0       EQU     $FFFF00
CTRL_TPU1       EQU     $FFFF10
CTRL_TPU2       EQU     $FFFF20
CTRL_TPU3       EQU     $FFFF30
CTRL_TPU4       EQU     $FFFF40
CTRL_TPU5       EQU     $FFFF50
CTRL_TPU6       EQU     $FFFF60
CTRL_TPU7       EQU     $FFFF70
CTRL_TPU8       EQU     $FFFF80
CTRL_TPU9       EQU     $FFFF90
CTRL_TPU10      EQU     $FFFFA0
CTRL_TPU11      EQU     $FFFFB0
CTRL_TPU12      EQU     $FFFFC0
CTRL_TPU13      EQU     $FFFFD0
CTRL_TPU14      EQU     $FFFFE0
CTRL_TPU15      EQU     $FFFFF0

* Pour les fonctions périphériques de la carte EID210
* controle , Port_C, CNA, CAN, PC104
CTRL            EQU     $900000
REG_ETAT        EQU     CTRL
Port_C          EQU     CTRL+$100
DIR_Port_C      EQU     Port_C+2
CNA             EQU     $B10000
SA0             EQU     CNA
CAN             EQU     $B20000
PC104           EQU     $B30000

* Table des vecteurs en RAM
Tab_vect  EQU   $800000
          list
```

ANNEXE 2 Fichiers de définitions inclus dans programmes en "C"

```
//=====================================================================
//          STRUCTURES DE DONNEES UTILES A LA PROGRAMMATION EN  C
//=====================================================================
//          Nom du fichier:     Structures_donnees.h
//          Date de création:       Aout 2001
//          Date de dernière modification:  Janv 2003
//          Auteurs:      T. HANS  J.L. ROHOU
//=====================================================================

/* Redéfinition des types */
                    typedef unsigned char       BYTE;
                    typedef unsigned short      WORD;
                    typedef unsigned long       ULONG;

/*          Union pour accéder à un octet (BYTE) soit en direct, soit en individualisant les 8 bits
//*************************************************************************************************/
          union byte_bits
          {         struct
                    {         unsigned char b7:1;
                              unsigned char b6:1;
                              unsigned char b5:1;
                              unsigned char b4:1;
                              unsigned char b3:1;
                              unsigned char b2:1;
                              unsigned char b1:1;
                              unsigned char b0:1;
                    }bit;
                    BYTE valeur;
          };

/*          Union pour accéder à un mot de 16 bit  soit en direct soit en individualisant les 16 bits
//*************************************************************************************************/
          union word_bits
          {         struct
                    {         unsigned char b15:1;
                              unsigned char b14:1;
                              unsigned char b13:1;
                              unsigned char b12:1;
                              unsigned char b11:1;
                              unsigned char b10:1;
                              unsigned char b9:1;
                              unsigned char b8:1;
                              unsigned char b7:1;
                              unsigned char b6:1;
                              unsigned char b5:1;
                              unsigned char b4:1;
                              unsigned char b3:1;
                              unsigned char b2:1;
                              unsigned char b1:1;
                              unsigned char b0:1;
                    }bit;
                    WORD valeur;
          };

/*          Union pour accéder à un mot de 16 bit  soit en direct
            soit en individualisant les 16 bits par ensembles de 2 bits  (Utile pour les sorties TPU)
//*************************************************************************************************/
          union word_duos
          {         struct
                    {         unsigned char duo7:2;
                              unsigned char duo6:2;
                              unsigned char duo5:2;
                              unsigned char duo4:2;
                              unsigned char duo3:2;
                              unsigned char duo2:2;
                              unsigned char duo1:2;
                              unsigned char duo0:2;

                    } duo;
                    WORD valeur;
          };
```

// Suite page suivante

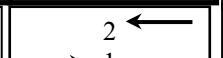
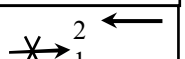
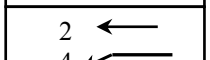
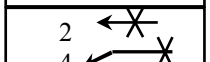
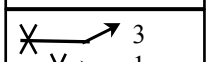
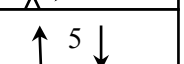
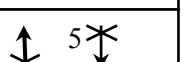
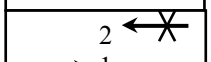
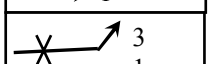
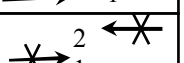
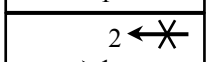
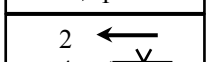
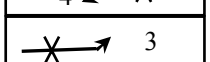
```
/*  Structure pour accéder à un mot de 16 bits soit en direct soit en séparant sur 8 bits de poids forts (b15 à b8)
    et gardant unis les 8 bits de poids faibles (O_lsb)
 ***************************************************************************************************************/
        union word_bits_octet
        {       struct
                {       unsigned char b15:1;
                        unsigned char b14:1;
                        unsigned char b13:1;
                        unsigned char b12:1;
                        unsigned char b11:1;
                        unsigned char b10:1;
                        unsigned char b9:1;
                        unsigned char b8:1;
                        unsigned char O_lsb:8;
                } bits_octet;
                WORD val_wbo;
        };

/*  Structure pour séparer l'octet de poids forts (O_msb) de l'octet de poids faibles (O_lsb)
    d'un mot de 16 bits, avec l'octet de poids fort pouvant être séparé en 8 bits individuels
 ***************************************************************************************************************/
struct word_bytes
        { union byte_bits O_msb;
          unsigned char O_lsb;
         };

/*  Union pour accéder à un mot de 16 bits (WORD) soit en direct soit en séparant sur 8 bits de poids faibles (b0 à b7)
   et gardant unis les 8 bits de poids forts (O_msb)
 ***************************************************************************************************************/
union word_octet_bits
        {       struct
                {       unsigned char O_msb:8;
                        unsigned char b7:1;
                        unsigned char b6:1;
                        unsigned char b5:1;
                        unsigned char b4:1;
                        unsigned char b3:1;
                        unsigned char b2:1;
                        unsigned char b1:1;
                        unsigned char b0:1;
                } octet_bits;
                WORD valeur;
        };
// Fin de fichier
```

```
/*=============================================================================================================
 *   DEFINITION DES LABELS ET ADRESSES PERMETTANT L'ACCES AUX REGISTRES DU MICROPROCESSEUR 68332
 * -------------------------------------------------------------------------------------------------------------
 *          Nom du fichier : CPU_REG.H
 *          Date de création :  Août 2001
 *          Date de la dernière révision :  Février 2002
 *          Auteurs:  J.M. ROHOU          T. HANS
 *============================================================================================================*/

#ifndef _CPU_H
#define _CPU_H
/*************************************
 *
 *   " System Integration Module "
 *************************************/
#define VBR             *(WORD*) 0x000000               /* Vector Base Register Vecteur de base 5-6*/

/* registre de configuration du system integration module */
#define Simcr           *(WORD*) 0xFFFA00               /* regitre de controle de la configuration du SIM */
#define Syncr           *(WORD*) 0xFFFA04               /* registre de controle de l'horloge æp */
#define Sypcr           *(WORD*) 0xFFFA21               /* registre de controle du systeme de protection */
#define Picr            *(WORD*) 0xFFFA22               /* registre de controle des interruptions */
#define Pitr            *(WORD*) 0xFFFA24               /* registre de controle du timer d'interruption */
#define Rsr                     *(WORD*) 0xFFFA07               /* reset register */
```

// Suite page suivante

```
/* registre de configuration des chip selects */
#define     Cspar0     *(WORD*) 0xFFFA44          /* reg. de contr. de la config du CSboot et des chip selects CS0 à CS5 */
#define     Cspar1     *(WORD*) 0xFFFA46          /* regitre de controle de la configuration des chips selects CS6 à CS10 */
#define     Csbarbt *(WORD*)    0xFFFA48          /* regitre de configuration de l'adresse de base du chip select BOOT */
#define     Csorbt     *(WORD*) 0xFFFA4A          /* regitre de controle des options de configuration du chip select BOOT */
#define     Csbar0     *(WORD*) 0xFFFA4C          /* regitre de configuration de l'adresse de base du chip select CS0 */
#define     Csor0      *(WORD*) 0xFFFA4E          /* regitre de controle des options de configuration du chip select CS0 */
#define     Csbar1     *(WORD*) 0xFFFA50          /* regitre de configuration de l'adresse de base du chip select CS1 */
#define     Csor1      *(WORD*) 0xFFFA52          /* regitre de controle des options de configuration du chip select CS1 */
#define     Csbar2     *(WORD*) 0xFFFA54          /* regitre de configuration de l'adresse de base du chip select CS2 */
#define     Csor2      *(WORD*) 0xFFFA56          /* regitre de controle des options de configuration du chip select CS2 */
#define     Csbar3     *(WORD*) 0xFFFA58          /* regitre de configuration de l'adresse de base du chip select CS3 */
#define     Csor3      *(WORD*) 0xFFFA5A          /* regitre de controle des options de configuration du chip select CS3 */
#define     Csbar4     *(WORD*) 0xFFFA5C          /* regitre de configuration de l'adresse de base du chip select CS4 */
#define     Csor4      *(WORD*) 0xFFFA5E          /* regitre de controle des options de configuration du chip select CS4 */
#define     Csbar5     *(WORD*) 0xFFFA60          /* regitre de configuration de l'adresse de base du chip select CS5 */
#define     Csor5      *(WORD*) 0xFFFA62          /* regitre de controle des options de configuration du chip select CS5 */
#define     Csbar6     *(WORD*) 0xFFFA64          /* regitre de configuration de l'adresse de base du chip select CS6 */
#define     Csor6      *(WORD*) 0xFFFA66          /* regitre de controle des options de configuration du chip select CS6 */
#define     Csbar7     *(WORD*) 0xFFFA68          /* regitre de configuration de l'adresse de base du chip select CS7 */
#define     Csor7      *(WORD*) 0xFFFA6A          /* regitre de controle des options de configuration du chip select CS7 */
#define     Csbar8     *(WORD*) 0xFFFA6C          /* regitre de configuration de l'adresse de base du chip select CS8 */
#define     Csor8      *(WORD*) 0xFFFA6E          /* regitre de controle des options de configuration du chip select CS8 */
#define     Csbar9     *(WORD*) 0xFFFA70          /* regitre de configuration de l'adresse de base du chip select CS9 */
#define     Csor9      *(WORD*) 0xFFFA72          /* regitre de controle des options de configuration du chip select CS9 */
#define     Csbar10 *(WORD*)    0xFFFA74          /* regitre de configuration de l'adresse de base du chip select CS10 */
#define     Csor10     *(WORD*) 0xFFFA76          /* regitre de controle des options de configuration du chip select CS10 */
/*******************************
*       Pour le  TPU
 * " Time Processor Unit "
/*******************************/
#define CFSR0       *(short *)(0xFFFE0C)        // Channel Function Select Register
#define CFSR1       *(short *)(0xFFFE0E)        // Permet de définir la fonction souhaitée pour chacun des
#define CFSR2       *(short *)(0xFFFE10)        // 15 canaux (lignes) d'entrée-sortie (CPU0 à CPU15)
#define CFSR3       *(short *)(0xFFFE12)        // 4 bits sont affectés à un même canal
#define HSQR0       *(short *)(0xFFFE14)        // Host Sequence Queue Register
#define HSQR1       *(short *)(0xFFFE16)        // Permet d'effectuer un échange des canaux configurés en entrée
#define HSRR0       *(short *)(0xFFFE18)        // Host Sequence Request Register
#define HSRR1       *(short *)(0xFFFE1A)        // Permet l'accès aux canaux d'entrée-sortie
#define CPR0        *(short *)(0xFFFE1C)        // Chanel Priority Register
#define CPR1        *(short *)(0xFFFE1E)        // Permet de définir les niveaux de priorité
#define CTRL_TPU0  *(short *)(0xFFFF00)         // ControlR
#define CTRL_TPU1  *(short *)(0xFFFF10)         // Une zone de 8 mots qui est réservée à chaque canal d'entrée sortie
#define CTRL_TPU2  *(short *)(0xFFFF20)
#define CTRL_TPU3  *(short *)(0xFFFF30)
#define CTRL_TPU4  *(short *)(0xFFFF40)
#define CTRL_TPU5  *(short *)(0xFFFF50)
#define CTRL_TPU6  *(short *)(0xFFFF60)
#define CTRL_TPU7  *(short *)(0xFFFF70)
#define CTRL_TPU8  *(short *)(0xFFFF80)
#define CTRL_TPU9  *(short *)(0xFFFF90)
#define CTRL_TPU10*(short *)(0xFFFFA0)
#define CTRL_TPU11*(short *)(0xFFFFB0)
#define CTRL_TPU12*(short *)(0xFFFFC0)
#define CTRL_TPU13*(short *)(0xFFFFD0)
#define CTRL_TPU14*(short *)(0xFFFFE0)
#define CTRL_TPU15*(short *)(0xFFFFF0)

// Pour le temporisateur programmable
#define PITR        *(short *)(0xFFFA24) //
#define PICR        *(short *)(0xFFFA22)


/*******************************************
 *       Pour le QSM
 *   "Queued Serial Module"
 *******************************************/
#define QSMCR  *(WORD*)                 0xFFFC00
#define QILVR  *(WORD*)                 0xFFFC04
#define SCCR0  *(WORD*)                 0xFFFC08
#define SCCR1  *(WORD*)                 0xFFFC0A
#define SCSR   *(WORD*)                 0xFFFC0C
#define SCDR   *(WORD*)                 0xFFFC0E

#define PORTQS *(WORD*)                 0xFFFC14
#define PQSCTR *(WORD*)                 0xFFFC16   /* PQSPAR-DDRQS */

#define TDRE   (WORD)          0x0100
#define RDRF   (WORD)          0x0040
#define RAF              (WORD) 0x0020

#endif
```

```
//=====================================================================
//          DECLARATIONS DES ADRESSES DES ELEMENTS DE LA CARTE EID210
//---------------------------------------------------------------------------------------------------
//          Nom du fichier:     EID210_reg.h
//          Date de création:       Aout 2001
//          Date de dernière modification: Janv 2003
//          Auteurs:      T. HANS  J.L. ROHOU
//=====================================================================

#ifndef _EID210_reg.h
#define _EID210_reg.h

/*      Version materielle et logicielle      */
/*====================================*/
#define VERSION_HARD        0x00      /* Version et revision du hard */
#define REVISION_HARD       0x00
#define VERSION_SOFT        0x00      /* Version et revision du programme */
#define    REVISION_SOFT    0x00

 /* Adresses de bases des périphériques */
#define    CTRL                                  0x900000          /*          CPLD de contrôle  */
#define    REG_ETAT (*(union word_bits_octet*)   (CTRL+0x00))      /* registre d'état (en lecture uniquement)   */
#define    REG_CTRL (*(WORD*)                    (CTRL+0x02))
#define    A_Port_C  (*(struct word_bytes*)      (CTRL+0x100))     /*  Accès au registre de donnée du Port C */
#define    A_Dir_Port_C (*(struct word_bytes*)   (CTRL+0x102))     /* Accès au registre de direction du port C */
#define    USB                                   0xB00000          /*  Ad. de base Port USB          CS3       */
#define    CNA                                    0xB10000         /*  Ad. de base du C      Num ->      CS4 */
#define    SA0                 (*(BYTE*)         (CNA+0x00))       /*          Sortie Analo      voie 0  (SA0)  */
#define    SA1                 (*(BYTE*)         (CNA+0x02))       /*          Sortie Analogi    voie 0  (SA0)  */
#define    SA2                 (*(BYTE*)         (CNA+0x04))       /*          Sortie Analogique      0  (SA0)  */
#define    SA3                 (*(BYTE*)         (CNA+0x06))       /*          Sortie Analogique      0  (SA0)  */
#define    CAN                 (*(WORD*)         0xB20000)         /*  Convertisseur A->N      Num          CS5  */
#define    BUS                                   0xB30000          /*  Ad. de base du      Num -> A      CS7  */

 /* Pour accéder au différentes information du registre d'état */
 /* Bits accessibles en lecture uniquement */
 #define Etat_reset             REG_ETAT.bits_octet.b15     /* RESERVE      ne tourne pas si RESET!   */
 #define E_Irq_CAN              REG_ETAT.bits_octet.b14     /*  tat du Bit de fin de co      ion Ana -> Num   */
 #define E_Irq_USB              REG_ETAT.bits_octet.b13     /*  t du Bit d'état de la ligne interruption USB   */
 #define E_Irq4_Bus             REG_ETAT.bits_octet.b12     /* Etat du Bit d'état ligne IRQ4 du BUS   */
 #define E_Irq3_Bus             REG_ETAT.bits_octet.b11     /* Etat du Bit d'état ligne IRQ3 du BUS   */
 #define E_Irq2_Bus             REG_ETAT.bits_octet.b10     /* Etat du Bit d'état ligne IRQ2 du BUS   */
 #define E_Irq1_Bus             REG_ETAT.bits_octet.b9      /* Etat du Bit d'état ligne IRQ1 du BUS   */
 #define S_Controle             REG_ETAT.bits_octet.b8      /* Etat du Etat "Switch de Contrôle   */
 #define N_VERSION              REG_ETAT.bits_octet.O_lsb   /*  N° de VERSION soft PLD sur 8 bits   */


/*  Pour autoriser (valider) les interruptions */
#define VALID_IRQs  (*(union word_octet_bits*) (0x90      ))
#define Valid_IrqCtrl           VALID_IRQs.octet_bits.b
#define Valid_IrqCan            VALID_IRQs.octet_bits.b2
#define Valid_Irq1              VALID_IRQs.octet_bits.b3
#define Valid_Irq2              VALID_IRQs.octet_bits.b4
#define Valid_Irq3              VALID_IRQs.octet_bits.b5
#define Valid_Irq4              VALID_IRQs.octet_bits.b6
#define Valid_IrqUsb            VALID_IRQs.octet_bits.b7
#define Valid_unuse             VALID_IRQs.octet_bits.b0

// Pour une gestion directe du Port C
#define Port_C          A_Port_C.O_msb.valeur
#define PortC           Port_C
#define Dir_Port_C      A_Dir_Port_C.O_msb.valeur
#define DirPortC Dir_Port_C
#define
#define PortCDir Dir_Port_C
#define PortC_Dir Dir_Port_C
#define PC0             A_Port_C.O_msb.bit.b0
#define PC1             A_Port_C.O_msb.bit.b1
#define PC2             A_Port_C.O_msb.bit.b2
#define PC3             A_Port_C.O_msb.bit.b3
#define PC4             A_Port_C.O_msb.bit.b4
#define PC5             A_Port_C.O_msb.bit.b5
#define PC6             A_Port_C.O_msb.bit.b6
#define PC7             A_Port_C.O_msb.bit.b7

// Pour la gestion du convertisseur A->N
#define Fin_Conv_AN E_Irq_CAN

#endif
```

ETAT (Utilisé pour le language C)	ANNEXE Tableau des valeurs	Feu P2			Feu P1			Feu F5			Feu F4			Feu F3			Feu F2			Feu F1		
		Port C			Port B									Port A								
		2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
1		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		1	0	0	01	10	10	01	10	10	01	10	10	01	01	10	10	01	10	10	69A6	969A	0400
2		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	10	01	10	10	01	01	10	10	10	01	10	69A6	96A6	0200
3		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	01	10	10	10	10	01	01	10	10	10	10	01	696A	96A9	0200
4		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	01	10	10	10	01	10	01	10	10	10	01	699A	99A9	0200
5		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	10	01	01	10	10	10	10	01	01	10	10	69A5	AA5A	0200
6		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	10	01	10	01	10	10	10	01	10	01	10	69A6	6A66	0200
7		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	1	10	01	10	10	10	10	01	10	10	01	10	10	01	10	10	01	9AA6	9A69	0300
8		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	1	10	10	01	10	10	10	01	10	10	01	10	10	01	10	10	01	66A6	9A69	0200
9		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	10	01	10	10	01	10	01	10	01	10	10	69A6	999A	0200
10		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	10	01	10	01	10	10	10	01	01	10	10	69A6	6A5A	0200
11		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	10	01	10	10	01	10	01	10	10	01	10	69A6	99A6	0200
12		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	10	01	10	10	01	10	01	10	01	10	10	69A6	999A	0200
13		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	01	10	10	10	01	01	10	10	10	10	01	699A	96A9	0200
14		V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		0	1	0	01	10	10	01	10	10	01	10	01	10	10	10	01	01	10	10	69A6	6A5A	0200
15	Aucun feu F Mais feu P	V	R	V	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R	V	O	R
		1	0	1	10	10	10	01	10	10	01	10	10	01	10	10	01	10	10	01	A9A6	9A69	0500

CONTENU DES REGISTRES (en hexa)