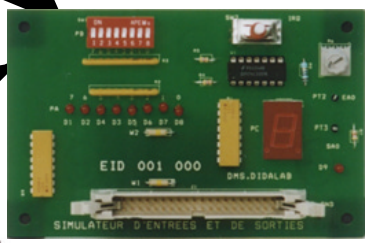
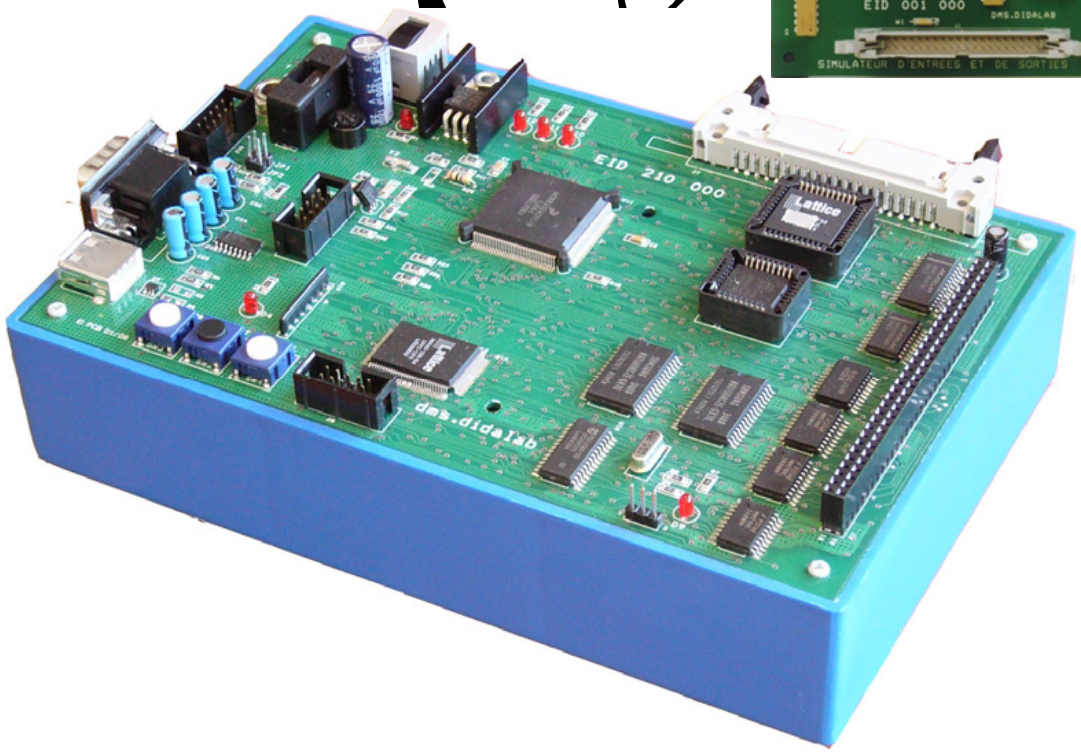
MANUEL
de
TRAVAUX PRATIQUES

Système EID210
+
Module "Simulateur d'entrées sorties "





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

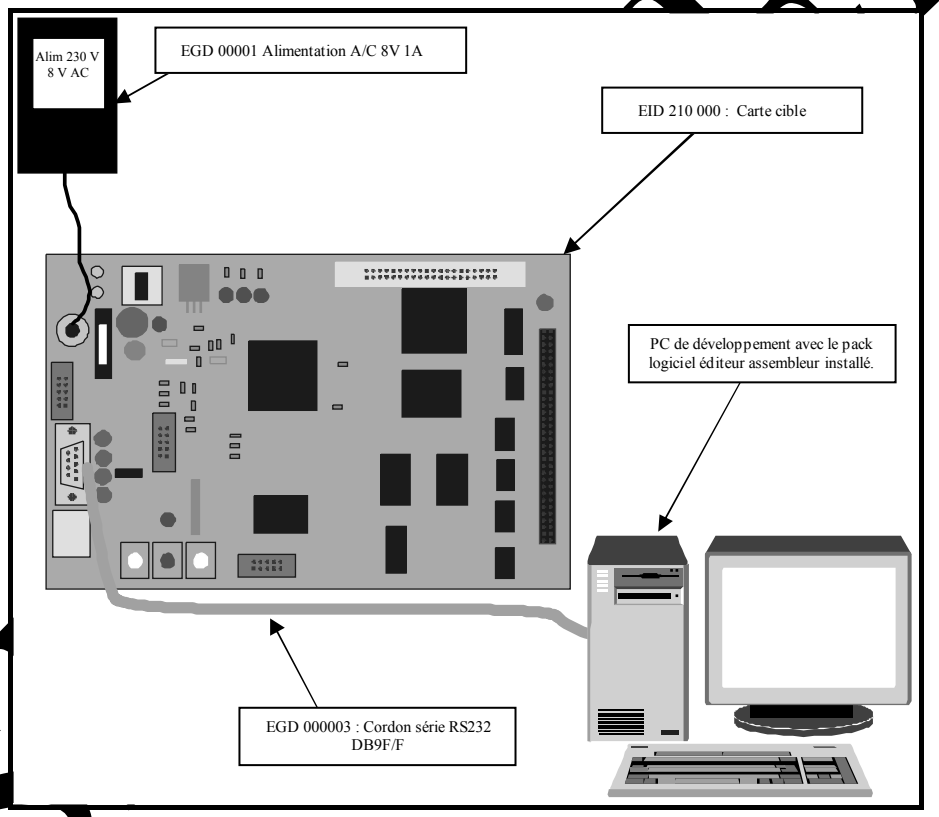
INSTALLATION DU MATERIEL

-> **Relier** la carte EID 210 000 au PC de développement en assembleur (livré avec le matériel et préalablement installé conformément à la notice technique). par le câble USB ou par défaut par le câble série RS232

-> **Connecter** le boîtier alimentation sur la carte EID 210 000, (7 à 12 V AC ou DC),

-> **Connecter** le simulateur d'entrées sorties EID 001 000 à l' EID 210 000(carte CPU).

-> **Appuyer** sur le bouton Marche Arrêt de la carte EID 210 000, la lampe témoin rouge doit s'allumer.



TP 1 CHENILLARD AVEC LEDS SUR PORT A

1.1 Sujet

Objectif :	Etre capable de configurer un port en sortie et d'activer ces sorties (Ce port faisant partie du « TPU » du micro-contrôleur. Etre capable d'analyser un cahier des charges imposé et d'écrire un programme pour le satisfaire. Etre capable de réaliser une temporisation de type logiciel puis en utilisant la fonction "TIMER" du microcontrôleur. Etre capable de réalise un sous programme en langage d'assemblage
Cahier des charges :	On souhaite réaliser un « chenillard » à l'aide des 8 Leds connectées sur le port A de la carte processeur EID210. Ce qui conduit à la séquence suivante : D8 allumée, puis attendre 0,5S environ, puis D7 allumée, puis attendre 1S etc… Après que D1 soit allumée, on boucle c'est à dire qu'on repasse à D8. ***Variante 1*** La temporisations se fera par une boucle d'attente logiciel. ***Variante 2*** La temporisations se fera en utilisant le "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 210 000
Câble de liaison USB, ou à défaut câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,
Simulateur d'entrées sorties réf : EID001000

Durée : 4 heures

1.2 Eléments de solution

1.2.1 Activation des sorties

Configurer les bits du port en sortie:

D'après le schéma de la carte simulateur donné en ANNEXE, les LEDs sont connectées sur le ports A de la carte processeur EID210.
Le port A est connecté sur les sorties CHANNEL0 (TPU0) <-> PA0 à CHANNEL7 (TPU7) <-> PA7 du microcontrôleur.
Le mode de fonctionnement d'une ligne TPU du micro-contrôleur est défini en chargeant un code sur 4 bits dans un registre prévu à cet effet (Registres CFSRi avec i=0,1,2,3 sur 16 bits, définis dans le fichier à inclure "**EID210.def**"), suivant la correspondance:

Bit de CFSR2	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Liaison TPU (N° CHANNEL)	7	6	5	4
Bit Port B	7	6	5	4
Bit de CFSR3	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Liaison TPU (N° CHANNEL)	3	2	1	0
Bit Port B	3	2	1	0

Dans notre cas, le mode de fonctionnement souhaité est un mode de sortie simple.
Dans la documentation technique du microcontrôleur [illegible] et plus particulièrement dans le chapitre consacré au "**TPU**", ce mode d'entées sorties simples est noté "**DIO**" (**D**iscret **I**nput **O**utput).
Dans ce cas, le code binaire sur 4 bits à charger dans les registres de configuration doit être: 1000 = $8

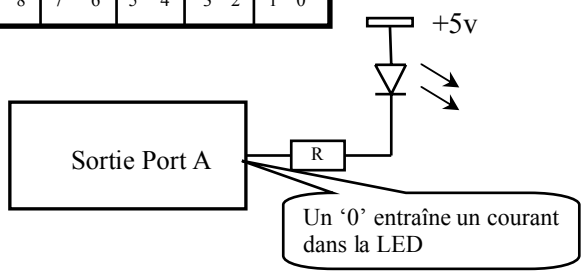
Soit les instructions permettant de configurer le port A en mode "DIO":

```
move.w    #$8888,CFSR2
move.w    #$8888,CFSR3
```

Activer un bit du port de sortie (Port A) à '1' ou à '0'

Pour mettre à '1' logique un bit du port A il faut mettre un binôme binaire "0 1" dans l'emplacement correspondant du registre HSSR1 (dont l'adresse est définie dans le fichier de définition EID210.def), pour mettre à '0' logique il faut mettre la valeur "1 0 " au même binôme.

Repére de la Led	D1	D2	D3	D4	D5	D6	D7	D8
N° du Bit dans le port A	7	6	5	4	3	2	1	0
N° "CHANNEL" du TPU	7	6	5	4	3	2	1	0
Bits du registre HSSR1	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0



Pour allumer une Led, il faut écrire un '0' logique sur le bit du port correspondant
(comme le montre la figure ci-contre)

Exemple :
On souhaite allumer la Led D8 et éteindre les autres, il faudra donc charger le registre HSSR1 par la valeur :

N° de la diode concernée :	D0	D2	D3	D4	D5	D6	D7	D8	
Valeur port sortie port :	1	1	1	1	1	1	1	0	
Valeur dans registre HSRR1 :	01	01	01	01	01	01	01	10	= $5556

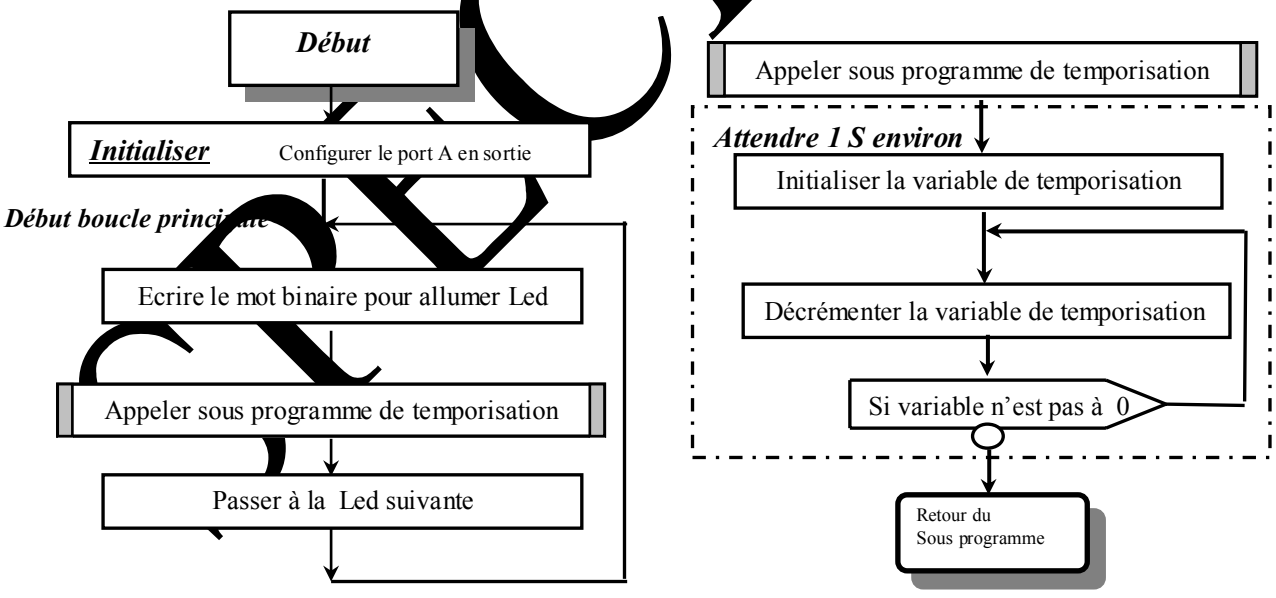
Pour réaliser le chenillard suivant la séquence imposée par le cahier des charges, il faudra écrire la séquence suivante dans le registre HSSR1:

Repères Leds	D1	D2	D3	D4	D5	D6	D7	D8	
Etats logiques	1	1	1	1	1	1	1	0	
Valeur registre HSSR1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	1 0	-> $5556
Etats logiques	1	1	1	1	1	1	0	1	
Valeur registre HSSR1	0 1	0 1	0 1	0 1	0 1	0 1	1 0	0 1	-> $5559
Etats logiques	1	1	1	1	1	0	1	1	
Valeur registre HSSR1	0 1	0 1	0 1	0 1	0 1	1 0	0 1	0 1	-> $5565
Etats logiques	1	1	1	1	0	1	1	1	
Valeur registre HSSR1	0 1	0 1	0 1	0 1	1 0	0 1	0 1	0 1	-> $5595
Etats logiques	1	1	1	0	1	1	1	1	
Valeur registre HSSR1	0 1	0 1	0 1	1 0	0 1	0 1	0 1	0 1	-> $5655
Etats logiques	1	1	0	1	1	1	1	1	
Valeur registre HSSR1	0 1	0 1	1 0	0 1	0 1	0 1	0 1	0 1	-> $5955
Etats logiques	1	0	1	1	1	1	1	1	
Valeur registre HSSR1	0 1	1 0	0 1	0 1	0 1	0 1	0 1	0 1	-> $6555
Etats logiques	0	1	1	1	1	1	1	1	
Valeur registre HSSR1	1 0	0 1	0 1	0 1	0 1	0 1	0 1	0 1	-> $9555

Réalisation d'une temporisation de type logiciel:

On réalise une temporisation logicielle en initialisant une variable à une certaine valeur et en décrémentant celle-ci jusqu'à ce qu'elle devienne nulle. La durée d'exécution de cette boucle de décrémentation, exécutée "n" fois (n = valeur initiale de la variable) constitue le laps de temps souhaitée. Dans le programme donné ci-après la variable est contenue dans le registre

1.2.2 Organigramme, solution avec temporisation logiciel :



1.2.3 Programme en assembleur A68xxx avec boucle de temporisation logicielle

```
        ************************************************************************
        *               TP  SUR EID210 AVEC SIMULATEUR D'ENTREES SORTIES       *
        *               Faire un un chenillard sur les Leds du port   A        *
        ************************************************************************
        * Cahier des charges :                                                 *
        ******************                                                     *
        *  Rem: Ces LEDs sont connectées sur le port A                         *
        *          Lignes TPU du 68332: TPU0 à TPU7  (CHA0  à CHA7)            *
        *    Les leds s'allument suivant le cycle                              *
        *            D8 -> D7 -> .... -> D1  puis  -> D8 -> D7  ... etc        *
        *   Une led s'allume par un 0                PA0=0 -> PA1=0  .... etc  *
        *    Le temps pendant lequel une led s'allume est d'environ 1S         *
        *    Ce temps sera généré par une boucle "logiciel"                    *
        *                                                                      *
        *  NOM du FICHIER:  CHENI_1_PA.SRC                                     *
        **********************                                               *
        ************************************************************************
        * Inclusion du fichier définissant les différents labels
        include EID210.def
        section                 code
************************************
*       PROGRAMME PRINCIPAL       *
************************************
*       INITIALISER
*************************
* Configurer le port A    en mode "Discet Input Output" -> code $8
DEBUT   move.w  #$8888,CFSR3            * CHA0 à CHA3 en mode "DIO"
        move.w  #$8888,CFSR2            * CHA4 à CHA7 en mode "DIO"
        move.w  #$FFFF,CPR1             * Tous les bits en priorité haute

*       BOUCLE PRINCIPALE
************************************
Deb_BP move.w   #$5556,HSRR1            * Allumer uniquement D8
        jsr     ATT
        move.w  #$5559,HSRR1            * Allumer uniquement D7
        jsr     ATT
        move.w  #$5565,HSRR1            * Allumer uniquement D6
        jsr     ATT
        move.w  #$5595,HSRR1            * Allumer uniquement D5
        jsr     ATT
        move.w  #$5655,HSRR1            * Allumer uniquement D4
        jsr     ATT
        move.w  #$5955,HSRR1            * Allumer uniquement D3
        jsr     ATT
        move.w  #$6555,HSRR1            * Allumer uniquement D2
        jsr     ATT
        move.w  #$9555,HSRR1            * Allumer uniquement D1
        jsr     ATT
        bra     Deb_BP
*       Fin du programme principal
************************************

************************************
*       LES SOUS-PROGRAMMES       *
************************************

*       Sous programme d'attente d'environ 1 seconde
***********************************************
ATT     move.l  #$001FFFFF,d2
ATT1    sub.l   #1,d2
        bne     ATT1
        rts                             * Retour de sous programme
Fin du sous programme
******************

        end
*  Fin du fichier source assembleur
******************************
```

1.2.4 Programme en langage C avec boucle de temporisation logicielle

```
/*****************************************************************************
*     TP sur  EID210 avec SIMULATEUR D'ENTREES SORTIES      *
*****************************************************************************
*  Faire un chenillard avec les 8 leds du Simulateur  d'E/S                 *
*  Rem: Ces LEDs sont connectées sur le port A                              *
*          Lignes TPU du 68332: TPU0 à TPU7  (CHA0  à CHA7)                 *
* CAHIER DES CHARGES:                                                       *
*  Les leds s'allument suivant le cycle                                     *
*            D8 -> D7 -> .... -> D1  puis  -> D8 -> D7  ... etc             *
* Une led s'allume par un 0                                                 *
*             PA0=0 -> PA1=0  .... etc                                      *
*  Le temps pendant lequel une led s'allume est d'environ 1S                *
*  Ce temps sera généré par une boucle "logiciel"                           *
*---------------------------------------------------------------------------*
*  NOM du FICHIER:  CHENILLARD_1_PA.C                                       *
*****************************************************************************/
/* Liste des fichiers à inclure */
#include "Cpu_reg.h"
#include "EID210_reg.h"
#include "Structures_donnees.h"
//  DECLARATIONS
//*******************
void ATTendre(int);

//==============================
//              FONCTION PINCIPALE
//==============================
main()
{
//Initialisations
//****************
/* Initialisation du port A en sortie*/
CFSR3=0x8888;                   /* CHA0 à CHA3 en mode "DIO" */
CFSR2=0x8888;                   /* CHA4 à CHA7 en mode "DIO" */
CPR1=0xFFFF;                    /* Tous les bits en priorité haute */

        //Début de la fonction principale
        //************************************
    do
        {
        //Allumage succéssif des LED
        HSRR1=0x5556;                   /* D8*/
        //Appeler le sous-programme TEMPORISATION
        ATTendre(1);
        HSRR1=0x5559;                   /* D7 */
        //Appeler le sous-programme TEMPORISATION
        ATTendre(1);
        HSRR1=0x5565;                   /* D6 */
        //Appeler le sous-programme TEMPORISATION
        ATTendre(1);
        HSRR1=0x5595;                   /* D5 */
        //Appeler le sous-programme TEMPORISATION
        ATTendre(1);
        HSRR1=0x5655;                   /* D4 */
        //Appeler le sous-programme TEMPORISATION
        ATTendre(1);
        HSRR1=0x5955;                   /* D3 */
        //Appeler le sous-programme TEMPORISATION
        ATTendre(1);
        HSRR1=0x6555;                   /* D2 */
        //Appeler le sous-programme TEMPORISATION
        ATTendre(1);
        HSRR1=0x9555;                   /* D1 */
        //Appeler le sous-programme TEMPORISATION
        ATTendre(1);
        }while(1);  // Fin de la boucle principale
}
//    Fin de la fonction principale
//*********************************
//================================
//              FONCTION de temporisation
//================================
void ATTendre(int nb)    // Le paramétre passé (nb) donne le nombre de secondes
{
int i;
for (i=(400000*nb);i>0;i--);            /*Décrémentation de la variable de temporisation*/
}
```

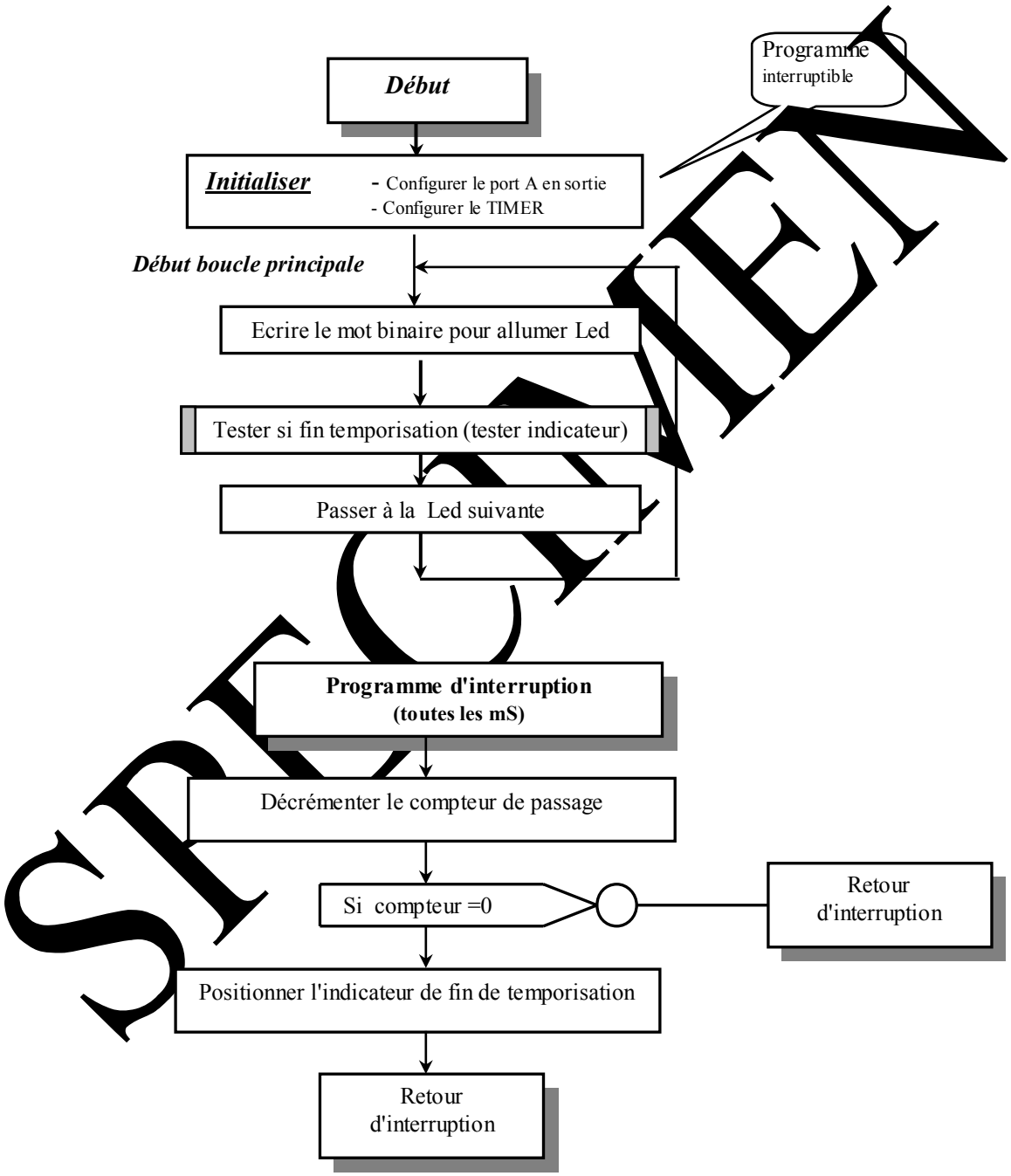
1.2.5 Organigramme, solution avec temporisation réalisée par fonction "Timer" du micro-contrôleur

Pour obtenir une interruption périodique toutes les 1 mS il faut initialiser les deux registres dont les labels ont été définis dans le fichier EID210.def:

"**PICR**" (**P**eriodic **I**nterrupt **C**ontrol **R**egister) à $0760

"**PITR**" (**P**eriodic **I**nterrupt **T**imer **R**egister) à $0008.

Par ailleurs, il faudra initialiser la table des vecteurs et prévoir un programme d'interruption.



1.2.6 Programme en assembleur A68xxx, solution avec "Timer" du microcontrôleur

```
        ************************************************************************
        *               TP  SUR EID210 AVEC SIMULATEUR D'ENTREES SORTIES       *
        *               Faire un un chenillard sur les Leds du port   A        *
        ************************************************************************
        * Cahier des charges :                                                 *
        ******************                                                     *
        *  Rem:       Les  lignes du port A sont connectées sur lignes du TPU du 68332:  *
        *                        TPU0 à TPU7  (CHA0  à CHA7)                   *
        *                 Lignes TPU du 68332: TPU0 à TPU7  (CHA0  à CHA7)     *
        *                 Une led s'allume par un 0                            *
        *                        PA0=0 -> PA1=0  .... etc                      *
        *                 Le temps pendant lequel une led s'allume est d'environ 1S  *
        *                 Ce temps sera généré par le Timer du 68332           *
        *  NOM du FICHIER:  CHENI_2_PA.SRC  (solution 2)                       *
        ******************                                                     *
        ************************************************************************
        * Inclusion du fichier définissant les différents labels
        include EID210.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" -> code $8
DEBUT   move.w              #$8888,CFSR3            * CHA0 à CHA3 en mode "DIO"
        move.w              #$8888,CFSR2            * CHA4 à CHA7 en mode "DIO"
        move.w              #$FFFF,CPRI             * Tous les bits en priorité haute
        move.w              #$0003,CTRL_TPU0
* Configurer la base de temps
        move.l              #96                     * 96 est le n° du vecteur d'interruption
        move.l              #it_bt,d0               * f_it-bt est l'adresse de la fonction d'interruption
        asl.l               #2,d0
        add.l               #b_vect,d0              * Initialiser la table des vecteurs
        move.l              d0,a0
        move.l              a0,(a0)
        move.l              #1000,COMPTEUR          * 1000*1mS = 1S
        move.b              #$00,INDICATEUR         * de fin de comptage
        move.w              #$0008,PITR             * 1 interruption toutes les 1 ms
        move.w              #$0060,PICR

*       PROGRAMME PRINCIPALE
*****************************
Deb_BP  move.w              #$5556,HSRR1            * Allumer uniquement D8
ATT1    move.b              INDICATEUR,D2           * Attente fin tempo
        cmp.b               #01,D2
        bne                 ATT1
        move.l              #1000,COMPTEUR          * 1000*1mS = 1S
        move.b              #$00,INDICATEUR         * de fin de comptage

        move.w              #$5559,HSRR1            * Allumer uniquement D7
ATT2    move.b              INDICATEUR,D2           * Attente fin tempo
        cmp.b               #01,D2
        bne                 ATT2
        move.l              #1000,COMPTEUR          * 1000*1mS = 1S
        move.b              #$00,INDICATEUR         * de fin de comptage
        move.w              #$5565,HSRR1            * Allumer uniquement D6
```

```
ATT3    move.b      INDICATEUR,D2       * Attente fin tempo
        cmp.b       #01,D2
        bne         ATT3
        move.l      #1000,COMPTEUR      * 1000*1mS = 1S
        move.b      #$00,INDICATEUR     * de fin de comptage

        move.w      #$5595,HSRR1        * Allumer uniquement D5
ATT4    move.b      INDICATEUR,D2       * Attente fin tempo
        cmp.b       #01,D2
        bne         ATT4
        move.l      #1000,COMPTEUR      * 1000*1mS = 1S
        move.b      #$00,INDICATEUR     * de fin de comptage

        move.w      #$5655,HSRR1        * Allumer uniquement D4
ATT5    move.b      INDICATEUR,D2       * Attente fin tempo
        cmp.b       #01,D2
        bne         ATT5
        move.l      #1000,COMPTEUR      * 1000*1mS = 1S
        move.b      #$00,INDICATEUR     * de fin de comptage

        move.w      #$5955,HSRR1        * Allumer uniquement D3
ATT6    move.b      INDICATEUR,D2       * Attente fin tempo
        cmp.b       #01,D2
        bne         ATT6
        move.l      #1000,COMPTEUR      * 1000*1mS = 1S
        move.b      #$00,INDICATEUR     * de fin de comptage

        move.w      #$6555,HSRR1        * Allu[illegible]ment D2
ATT7    move.b      INDICATEUR,D2       * Attente fin tem[illegible]
        cmp.b       #01,D2
        bne         ATT7
        move.l      #1000,COMPTEUR      * 10[illegible]1mS = 1S
        move.b      #$00,INDICATEUR     * de fin[illegible] comptage

        move.w      #$9555,HSRR1        * Allumer u[illegible]ement D1
ATT8    move.b      INDICATEUR,D2       * Attente fin te[illegible]
        cmp.b       #01,D2
        bne         ATT8
        move.l      #1000,C[illegible]RTEUR     * 1000*1[illegible]mS = 1S
        move.b      #$00,[illegible]NDICATEUR   * de fin de comptage

        bra         [illegible]RP       * Fin de boucle principale
*       Fin du programme principal
*******************************



*******************************************
*       FONCTION D' INTE[illegible]UPTION  *
*       associée à la base de tem[illegible]  *
*******************************************
it_bt                   #$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 du programme d'interruption
****************************

        end
*  Fin du fichier source
```

1.2.7 Programme en Langage C, solution avec "Timer" du microcontrôleur

```
            /*************************************************************************
            *    TP sur  EID210 avec SIMULATEUR D'ENTREES SORTIES    *
            *************************************************************************
            *              Test du port C qui pilote l'afficheur 7 segments        *
            *               pour afficher un nombre décimal  ( 0 à 9)              *
            * CAHIER DES CHARGES  :                                                *
            ***************************                                           *
            * L'afficheur affiche un nombre décimal (entre 0 et 9)                 *
            * qui s'incrémente environ toutes les secondes                         *
            * Ce temps sera généré par Timer en interruption                       *
            *----------------------------------------------------------------------*
            * NOM du FICHIER:  Aff_7S_HI.C                                         *
            **********************                                               *
            *************************************************************************/
#include "EID210_reg.h"
#include "Simulateur_ES.h"
#include "CPU_reg.h"
#include "Structures_Donnees.h"

//********************
//  DECLARATIONS
//********************
// Les prototypes des fonctions
void AFFICHER(unsigned char);
/* Déclaration des variables pour la temporisation */
int CPTR_mS;                                    /* déclaration des variables et compteurs de millisecondes */
unsigned char Fin_tempo,Tempo_en_cours;         /*  déclaration variables pour gestion temporisation


/* FONCTION d'interruption pour temporisation  (base de temps: toutes les mS)
=============================================================================*/
void irq_bt()
{           if (Tempo_en_cours==1)
                    { CPTR_mS++;
                if (CPTR_mS ==1000) Fin_tempo=1,Tempo_en_cours=0,CPTR_mS=0;
                    }
} // Fin de la définition de la fonction d'interruption pour la temporisation

//============================
//          FONCTION PRINCIPALE
//============================
main()
{
//Varirables locales
//********************
unsigned char compteur;

//  Initialisations
//*****************
// Pour le port C
Dir_Port_C=0xFF;    // Tous les bits du port C sont en sortie
Port_C=0x00;                    // On éteint tous les segments
// Pour base de temps
CPTR_mS=0;                              // initialisation compteurs millisecondes
SetVect(96,&irq_bt);            // Mise en place de l'autovecteur
PITR = 0x0008;                  // Une interruption toutes les millisecondes
PICR = 0x0060;                  //  96 = 60H
Tempo_en_cours=0;               // Temporisation inactive

//Début de la boucle principale
//*****************************
            do
            {           for (compteur=0;compteur<=15;compteur++)
                        {
                                    // AFFICHER
                                    Port_C=Tab_Valeurs_7Seg[compteur];
                                    //Pour attendre 1S
                                    Tempo_en_cours=1,Fin_tempo=0;           // Initialisation pour tempo
                                    do{}while(Fin_tempo==0);                // Attendre fin tempo
                        }
             /*Remise à 0 du compteur */
             compteur=0;
            }while (1);
}
// Fin de la fonction principale
//*********************************
```

TP 2 RECOPIE D'UN PORT D'ENTREE 8 BITS SUR PORT DE SORTIE 8 BITS.

2.1 Sujet

Objectif :	**Capacités complémentaires:** Etre capable de configurer un port d'entrées / sorties en entrée. Etre capable d'acquérir l'état de ces entrées (technique d'acquisition propre au TPU du micro-contrôleur 68332).
Cahier des charges :	L'état des 8 "SWITCHS" connectés sur le port E de la carte EID210 est recopié sur les 8 LEDs qui elles, sont connectées sur le port A.

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332. Réf : EID 210 000
Câble de liaison USB, ou à défaut câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A. Réf : EGD000001,
Simulateur d'entrées sorties réf : EID001000

Durée : 4 heures

2.2 Eléments de solution

2.2.1 Acquisition des états des "Switchs"

Configurer les bits du port B en entrée:

D'après le schéma de la carte simulateur donné en ANNEXE, les "switchs" du simulateur d'entrées / sorties sont connectés sur le port B de la carte EID210.
D'après le schéma de la carte EID210, les bits du port B font partie des sorties TPU du micro-contrôleur 68332, avec la correspondance donnée ci-après:

N° bit TPU	15	14	13	12	11	10	9	8
N° bit Port B	7	6	5	4	3	2	1	0

Le mode de fonctionnement d'une ligne TPU du micro-contrôleur est défini en chargeant un code sur 4 bits dans un registre prévu à cet effet (Registres CFSRi avec i=0,1,2,3 sur 16 bits, définis dans le fichier à inclure "**EID210.def**"), suivant la correspondance:

Bit de CFSR0	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Liaison TPU (N° CHANNEL)	15	14	13	12
Bit Port B	7	6	5	4
Bit de CFSR1	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Liaison TPU (N° CHANNEL)	11	10	9	8
Bit Port B	3	2	1	0

Dans notre cas, le mode de fonctionnement souhaité est un mode d'entrée simple.
Dans la documentation technique du microcontrôleur 68332, et plus particulièrement dans le chapitre consacré au "**TPU**", ce mode d'entées / sorties simples est noté "**DIO**" (**D**iscret **I**nput **O**utput).
Dans ce cas, le code binaire sur 4 bits à charger dans les registres de configuration doit être: 1000 = $8

Soit les instructions permettant de configurer le port B en mode "DIO":

```
move.w    #$8888,CFSR0
move.w    #$8888,CFSR1
```

Technique d'acquisition de l'état d'une entrée

Lorsque l'on souhaite acquérir l'état d'une entrée, il faut écrire le binôme logique "11" aux emplacement correspondant du registre de service HSSR0:

Repére "SWITCH"	1	2	3	4	5	6	7	8
N° du Bit dans le port B	7	6	5	4	3	2	1	0
N° "CHANNEL" du TPU	15	14	13	12	11	10	9	8
Bits du registre HSSR0	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0

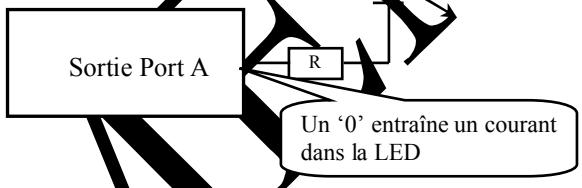
Si on souhaite acquérir les états de l'ensemble des 8 "SWITCHs", il faudra donc charger tous des binômes "11" dans le registre HSRR0, soit la valeur $FFFF.
A l'écriture du binôme binaire "11", il y a mémorisation de l'état de l'entrée dans une mémoire (dite mémoire d'état) de 16 bits réalisant une pile de type FIFO (First In First Out). Le dernier état mémorisé étant le bit de rang 15.
Il suffira donc de tester ce bit de rang 15 pour connaître l'état de l'entrée à l'instant de la dernière écriture du binôme "11" dans le registre HSRR0.
L'adresse de la mémoire d'état du "CHANNEL" de rang i est déduite des définitions données dans le fichier EID210.def (fichier à inclure), grâce à l'opération CH_etat+CTRL_TPi

2.2.2 Activation des sorties

Les LEDs sont connectées sur le ports A de la carte processeur EID210.
Le port A est connecté sur les sorties CH0 (TPU0) <-> PA0 à CH7 (TPU7) <-> PA7 du microcontrôleur.
Pour mettre à '1' logique un bit du port il faut mettre un binôme binaire « 0 1 » dans l'emplacement correspondant du registre HSRR1 (dont l'adresse est définie dans le fichier de définition EID210.def), pour mettre à '0' logique il faut mettre la valeur « 1 0 » au même binôme

Repére de la Led	D1	D2	D3	D4	D5	D6	D7	D8
N° du Bit dans le port A	7	6	5	4	3	2	1	0
Bits du registre HSSR1	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0

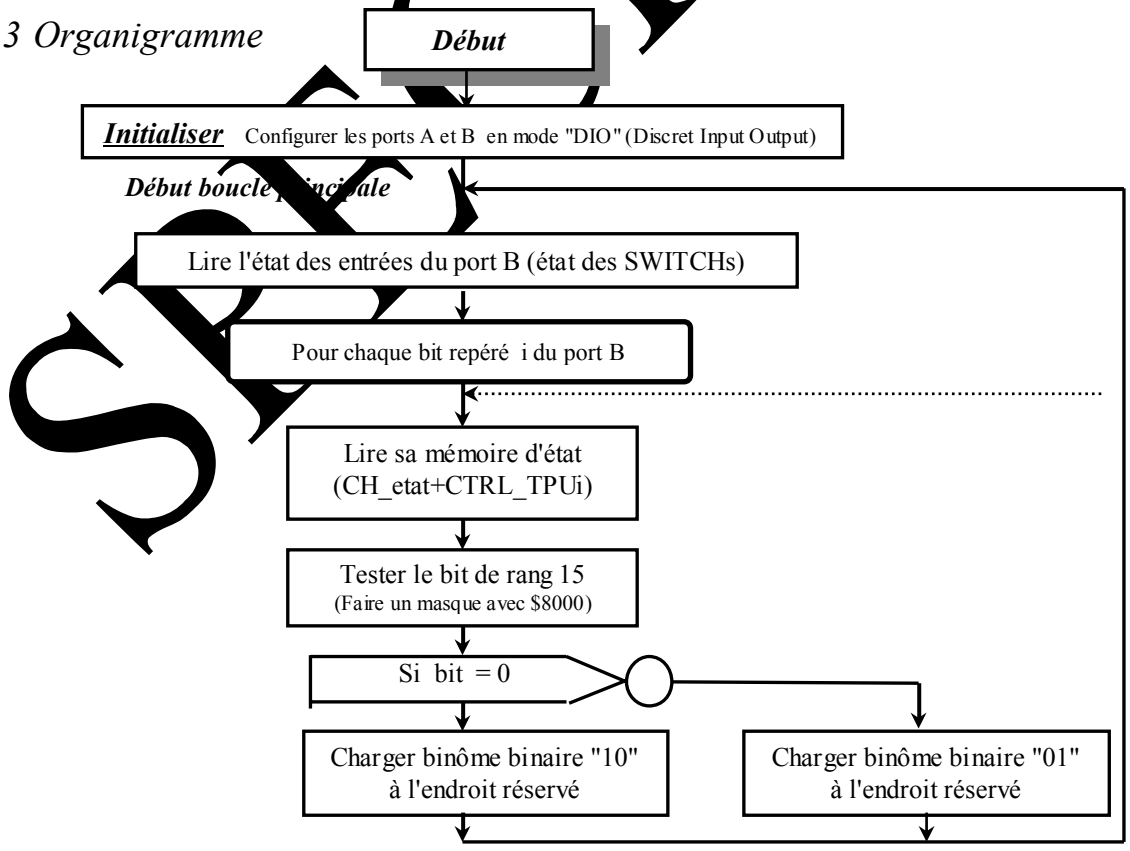
Pour allumer une Led, il faut écrire un '0' logique sur le bit du port correspondant
(comme le montre la figure ci-contre)



Exemple :
On souhaite allumer la Led D8 et éteindre les autres, il faudra donc charger le registre HSSR1 par la valeur :

N° de la diode concernée :	D0	D2	D3	D4	D5	D6	D7	D8	
Valeur port sortie port :	1	1	1	1	1	1	1	0	
Valeur dans registre HSRR1 :	01	01	01	01	01	01	01	10	= $5556

2.2.3 Organigramme



2.2.4 Programme en assembleur A68xxx

```
        ************************************************************************
        *               TP  SUR EID210 AVEC SIMULATEUR D'ENTREES SORTIES       *
        *               Recopier l'état des "Switchs"  (du port B) sur les Leds (du port A)  *
        ************************************************************************
        *  Cahier des charges                                                  *
        ********************                                                   *
        *  Les états des "SWITCHs" sont recopiés sur les LEDs                  *
        *  Rem: Ces LEDs sont connectées sur le port A                         *
        *           ->Lignes TPU du 68332: TPU0 à TPU7  (CHA0  à CHA7)         *
        *               Les  "SWITCHs" sont connectés sur port B               *
        *   ->Lignes TPU du 68332: TPU0 à TPU7  (CHA8  à CHA15)                *
        *                                                                      *
        *                                                                      *
        *  NOM du FICHIER:  RECOP_BsA.SRC                                      *
        ********************                                                   *
        ************************************************************************
        * Inclusion du fichier définissant les différents labels
        include EID210.def
*   Définition de constantes        *
*****************************
CH_etat equ             2
*************************************
*  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

* Les états des entrées sont mémorisés dans les registres d'état
* lorsque l'on écrira des 11 dans le registre de service
        move.w          #$AAAA,SCSR0            * Les entrées seront mémorisées

* Eteindre toutes les leds
        move.w          #$FFFF,HSRR1            * Tous les bits PA à 1
        move.w          #$5555,d0               * d0 -> image de HSRR1

*       BOUCLE PRINCIPALE
*************************************
Deb_BP * Acquérir les entrées
        move.w          #$FFFF,HSRR0

        * tester l'état de PB7 puis positionner PA7
        move.w          CH_etat+CTRL_TPU8,d4    * Lire registre d'état de PB7
        btst            #15,d4
        beq             b7_0
        bclr            #15,d0                  * Si  1 , préparer pour faire PA7=1
        bset            #14,d0
        bra             b7_1
b7_0    bset            #15,d0                  * Si 0 , préparer pour faire PA7=0
        bclr            #14,d0
b7_1    move            d0,HSRR1                * Charger sur port A

        * tester l'état de PB6 puis positionner PA6
        move.w          CH_etat+CTRL_TPU9,d4    * Lire registre d'état de PB6
        btst            #15,d4
        beq             b6_0
        bclr            #13,d0                  * Si  1 , préparer pour faire PA6=1
        bset            #12,d0
        bra             b6_1
b6_0    bset            #13,d0                  * Si 0 , préparer pour faire PA6=0
        bclr            #12,d0
b6_1    move            d0,HSRR1                * Charger sur port A
*   Suite du programme page suivante
```

```
        * tester l'état de PB5 puis positionner PA5
        move.w      CH_etat+CTRL_TPU10,d4   * Lire registre d'état de PB5
        btst        #15,d4
        beq         b5_0
        bclr        #11,d0                  * Si 1 , préparer pour faire PA5=1
        bset        #10,d0
        bra         b5_1
b5_0    bset        #11,d0                  * Si 0 , préparer pour faire PA5=0
        bclr        #10,d0
b5_1    move        d0,HSRR1                * Charger sur port A

        * tester l'état de PB4 puis positionner PA4
        move.w      CH_etat+CTRL_TPU11,d4   * Lire registre d'état de PB4
        btst        #15,d4
        beq         b4_0
        bclr        #9,d0                   * Si 1 , préparer pour faire PA4=1
        bset        #8,d0
        bra         b4_1
b4_0    bset        #9,d0                   * Si 0 , préparer pour faire PA4=0
        bclr        #8,d0
b4_1    move        d0,HSRR1                * Charger sur port A

        * tester l'état de PB3 puis positionner PA3
        move.w      CH_etat+CTRL_TPU12,d4   * Lire registre d'état de PB3
        btst        #15,d4
        beq         b3_0
        bclr        #7,d0                   * Si 1 , préparer pour faire PA3=1
        bset        #6,d0
        bra         b3_1
b3_0    bset        #7,d0                   * Si 0 , préparer pour faire PA3=0
        bclr        #6,d0
b3_1    move        d0,HSRR1 * Charger sur port A

        * tester l'état de PB2 puis positionner PA2
        move.w      CH_etat+CTRL_TPU13,d4   * Lire registre d'état de PB2
        btst        #15,d4
        beq         b2_0
        bclr        #5,d0                   * Si 1 , préparer pour faire PA2=1
        bset        #4,d0
        bra         b2_1
b2_0    bset        #5,d0                   * Si 0 , préparer pour faire PA2=0
        bclr        #4,d0
b2_1    *move       d0,HSRR1                * Charger sur port A

        * tester l'état de PB1 puis positionner PA1
        move.w      CH_etat+CTRL_TPU14,d4   * Lire registre d'état de PB1
        btst        #15,d4
        beq         b1_0
        bclr        #3,d0                   * Si 1 , préparer pour faire PA1=1
        bset        #2,d0
        bra         b1_1
b1_0    bset        #3,d0                   * Si 0 , préparer pour faire PA1=0
        bclr        #2,d0
b1_1    move        d0,HSRR1                * Charger sur port A

        * tester l'état de PB0 puis positionner PA0
        move.w      CH_etat+CTRL_TPU15,d4   * Lire registre d'état de PB0
        btst        #15,d4
        beq         b0_0
        bclr        #1,d0                   * Si 1 , préparer pour faire PA0=1
        bset        #0,d0
        bra         b0_1
b0_0    bset        #1,d0                   * Si 0 , préparer pour faire PA0=0
        bclr        #0,d0
b0_1    move        d0,HSRR1                * Charger sur port A

        bra         Deb_BP                  * Fin de boucle principale -> boucler
Fin du programme principal
************************
Fin du fichier
**************
        end
```

2.2.5 Programme en langage C

```
            /*************************************************************************
            *    TP sur  EID210 avec SIMULATEUR D'ENTREES SORTIES     *
            *************************************************************************
            * Les états des "SWITCHs" sont recopiés sur les LEDs          *
            *                                                             *
            * CAHIER DES CHARGES:                                         *
            * ************************                                    *
            *            Les LEDs sont connectées sur le Port A           *
            *    ->Lignes TPU du 68332: TPU0 à TPU7  (CHA0  à CHA7)       *
            *            Les  "SWITCHs" sont connectés sur Port B         *
            *    ->Lignes TPU du 68332: TPU0 à TPU7  (CHA8  à CHA15)      *
            *                                                             *
            * ----------------------------------------------------------- *
            * NOM du FICHIER:  RECOP_BsurA.C                              *
            * ******************                                          *
            *************************************************************************/
/* Liste des fichiers à inclure */
#include "Cpu_reg.h"
#include "EID210_reg.h"
#include "Simulateur_ES.h"
#include "Structures_donnees.h"

/**************************************
*       FONCTION PRINCIPALE
**************************************/
main()
{
/*          INITIALISER
*****************************/
/*  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  */

/* Les états des entrées sont mémorisés dans les registres d'état
 * Lorsque l'on écrira des 11 dans le registre de service                */
HSQR0=0xAAAA;       /* Les entrées seront mémorisées           */
/* Eteindre toutes les leds */
HSRR1=0x5555;       /* Tous les bits PA à 1 */

/*          BOUCLE PRINCIPALE
*****************************************/
do
{           /* Acquérir les entrées */
            HSRR0=0xFFFF;
            /*  tester l'état de PB7  puis positionner l'image de PA7  */
            if(PB7==1)PA7=un;
            else PA7=zero;
            /*  tester l'état de PB6  puis positionner l'image de PA6  */
            if(PB6==1)PA6=un;
            else PA6=zero;
            /*  tester l'état de PB5  puis positionner l'image de PA5 */
            if(PB5==1)PA5=un;
            else PA5=zero;
            /* tester l'état de PB4  puis positionner l'image de PA4  */
            if(PB4==1)PA4=un;
            else PA4=zero;
            /* tester l'état de PB3  puis positionner l'image de PA3  */
            if(PB3==1)PA3=un;
            else PA3=zero;
            /* tester l'état de PB2  puis positionner l'image de PA2  */
            if(PB2==1)PA2=un;
            else PA2=zero;
            /* tester l'état de PB1  puis positionner l'image de PA1  */
            if(PB1==1)PA1=un;
            else PA1=zero;
            /* tester l'état de PB0  puis positionner l'image de PA0  */
            if(PB0==1)PA0=un;
            else PA0=zero;
            // On actualise les 8 sorties (Recopie de l'image du port A)
            HSRR1=Image_PA;
} while(1);     /* Fin de la boucle principale */

}   /*          Fin de la fonction principale
    *****************************************/
```

TP 3 COMMANDE DE L'AFFICHEUR 7 SEGMENTS

3.1 Sujet

Objectif :	**Capacités complémentaires:** Etre capable d'afficher une donnée numérique à l'aide d'un afficheur '7 segments'. Etre capable de transcoder un information '**D**écimal **C**odé **B**inaire' (DCB) en une information de commande d'un afficheur 7 segment. Etre capable de programmer une structure informatique de type 'CHOISIR PARMI'
Cahier des charges :	***Variante 1: Affichage décimal*** On désire voir s'afficher sur l'afficheur 7 segments, les chiffres de 0 à 9 avec une périodicité d'une seconde environ par chiffre. ***Variante 2: Affichage hexadécimal*** On désire voir s'afficher sur l'afficheur 7 segments, les 15 caractères hexadécimaux (0 à 9 puis A, B, C, D, E et enfin F) avec une périodicité d'une seconde environ par caractère.

Matériel nécessaire :

Micro-ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 210 000
Câble de liaison USB ou à défaut câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,
Simulateur d'entrées sorties réf : EID001000

Durée : 4 heures

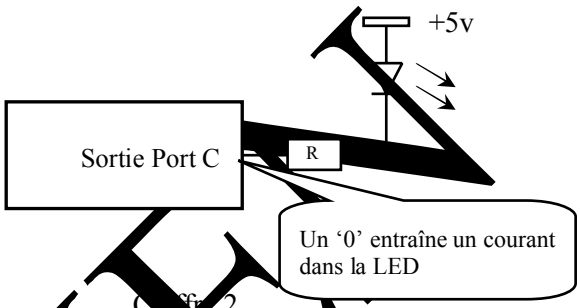
3.2 Eléments de solution

3.2.1 Activation des sorties

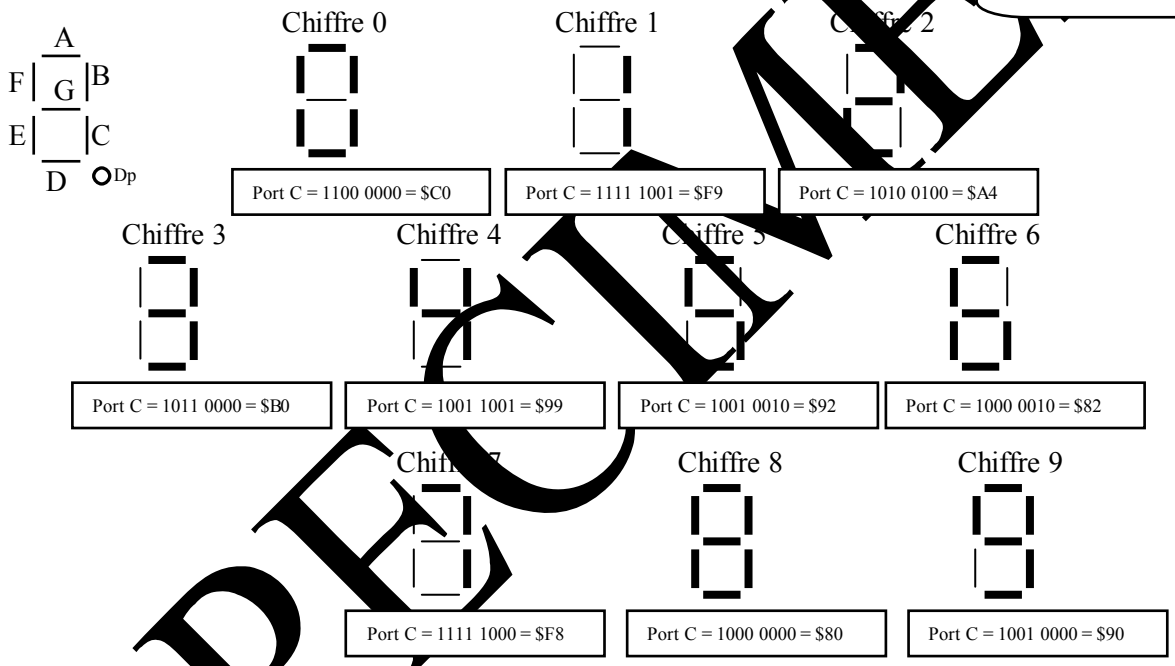
D'après le plan donné en ANNEXE, on constate que l'afficheur 7 segments est connecté sur le port C de la carte processeur, avec la correspondance donnée ci-contre :

Bits Port C :	7	6	5	4	3	2	1	0
Segment :	Dp	G	F	E	D	C	B	A

D'autre part l'afficheur est de type « ANODE COMMUNE » c'est à dire qu'il faudra écrire un 0 logique sur une sortie pour que la « LED » correspondante s'allume.



Détermination des états logiques du port C en fonction du chiffre que l'on souhaite afficher :

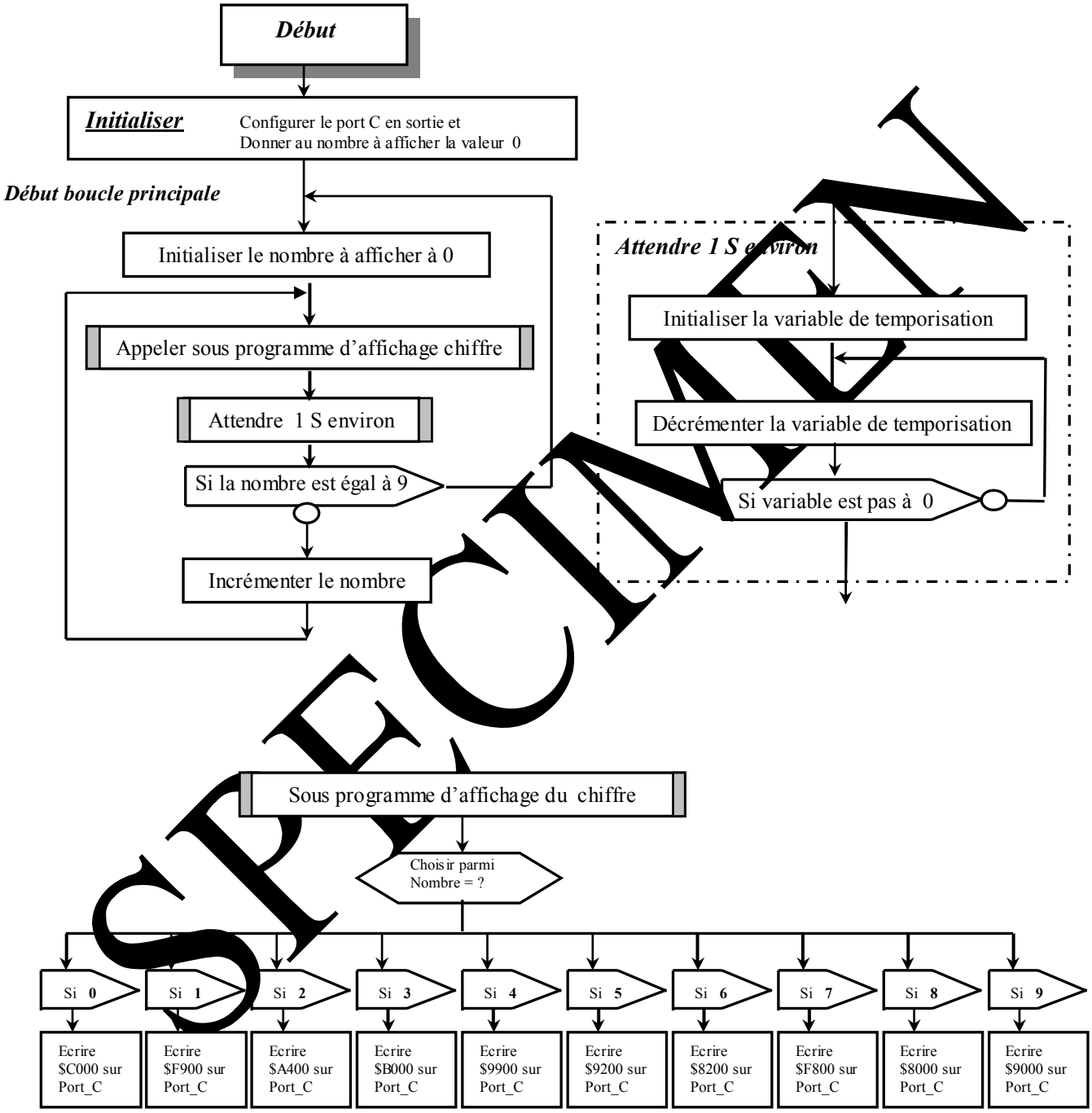


Initialisation et ecriture
Avant d'utiliser un bit du port C en sortie il faut initialiser le registre de direction associé (label DIR_Port_C dont l'adresse est définie dans le fichier de définition EID210.def qu'il faudra inclure).
Il faut écrire un '1' logique à l'emplacement correspondant dans le registre de direction pour pouvoir utiliser un bit du port C comme sortie.
Dans notre cas il faudra donc écrire $FF00 (car emplacement sur les poids fort).
Pour activer les bits de sorties, il faut écrire la valeur dans le registre de donnée associé au port C (label Port_C dont l'adresse est définie dans le fichier de définition EID210.def).

Exemple :
Si on souhaite afficher le chiffre '8' il faudra écrire l'instruction
Car la valeur doit être placée sur les poids forts du mot.

```
Move.w    #8000,Port_C
```

3.2.2 Organigrammes pour un affichage décimal (Variante 1)



3.2.3 Programme en assembleur A68xxx de l'affichage décimal (Variante 1)

```
        ************************************************************************
        *              TP  SUR EID210 AVEC SIMULATEUR D'ENTREES SORTIES        *
        ************************************************************************
        *                                                                      *
        *              Test du port C qui pilote l'afficheur 7 segments        *
        *               pour afficher un nombre décimal  ( 0 à 9)              *
        *                                                                      *
        *  Description:                                                        *
        ***************                                                       *
        *  L'afficheur affiche un nombre décimal (entre 0 et 9)                *
        *  qui s'incrémente environ toutes les secondes                        *
        *  Ce temps sera généré par une boucle "logiciel"                      *
        *                                                                      *
        *  NOM du FICHIER:  T_Port_C_1.SRC                                     *
        **********************                                              *
        ************************************************************************

************************************
*         Déclaration des variables        *
************************************
        section       var
NOMBRE ds.b       1           *  de taille 8 bits
*  Inclusion
**************
* Inclusion du fichier définissant les différents labels
        include               EID210.def


************************************
*  Début du programme exécutable        *
************************************
        section               code

*         INITIALISER
*************************
debut     move.w    #$FF00,DIR_Port_C  * Le port C est configuré en sortie
                                       * Un 1 impose un fonctionnement en sortie

*          BOUCLE PRINCIPALE
***********************************
*Incrementer nombre
Deb_BP    move.b              NOMBRE,d0           * Registre "do" est  image de NOMBRE
          cmp.b               #9,d0
          bne                 Etiq1
          move.b              #0,d0
          bra                 Etiq2
Etiq1     add.b               #1,d0
Etiq2     move.b              d0,NOMBRE
* Aller afficher le résultat
          jsr                 AFFICHER
*  Boucle d'attente d'environ 1 seconde
          move.l              #$01FFFFFF,d2
ATT       subi                #1,d2
          bne                 ATT
*  Passer au nombre suivant
          bra                 Deb_BP
* Fin de la boucle principale
*******************************
****************************
* Fin du programme principal   *
****************************

* Suite du programme page suivante
```

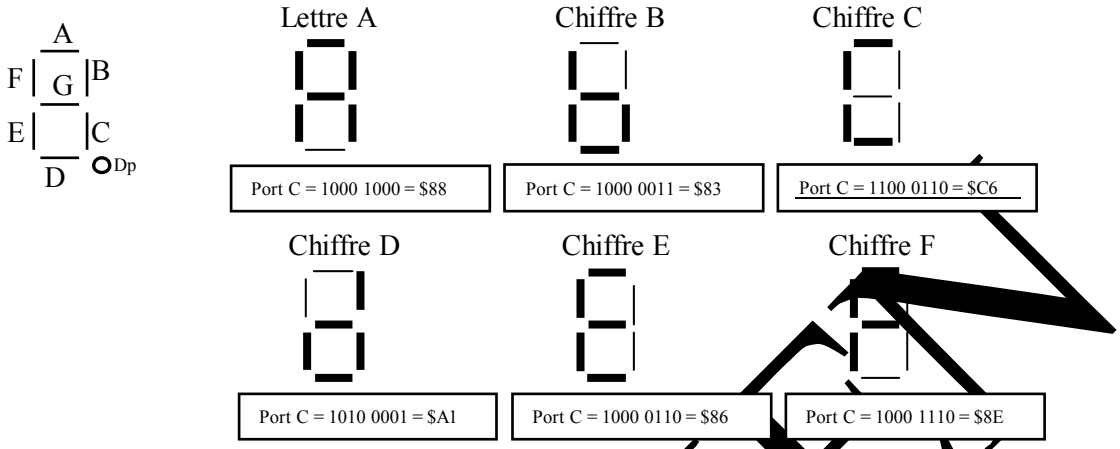
```
*******************************************
*  Sous programe de transcodage           *
*  "Décimal Codé Binaire" -> 7 Segments   *
*  et d'affichage                         *
*******************************************
AFFICHER
            cmp.b       #$00,d0
            bne         afficher1       * Sortir si c'est pas 0
            move.w      #$C000,Port_c
            rts
afficher1   cmp.b       #$01,d0
            bne         afficher2       * Sortir si c'est pas 1
            move.w      #$F900,Port_c
            rts
afficher2   cmp.b       #$02,d0
            bne         afficher3       * Sortir si c'est pas 2
            move.w      #$A400,Port_c
            rts
afficher3   cmp.b       #$03,d0
            bne         afficher4       * Sortir si c'est pas 3
            move.w      #$B000,Port_c
            rts
afficher4   cmp.b       #$04,d0
            bne         afficher5       * Sortir si c'est pas 4
            move.w      #$9900,port_c
            rts
afficher5cmp.b          #$05,d0
            bne         afficher6       * Sortir si c'est pas 5
            move.w      #$9200,port_c
            rts
afficher6   cmp.b       #$06,d0
            bne         afficher7       * Sortir si c'est pas 6
            move.w      #$8200,port_c
            rts
afficher7   cmp.b       #$07,d0
            bne         afficher8       Sortie si c'est pas 7
            move.w      #$F800,port_c
            rts
afficher  8 cmp.b       #$08,d0
            bne         afficher9       Sortir si c'est pas 8
            move.w      #$8000,port_c
            rts
afficher9   cmp.b       #$09,d0
            bne         afficher_rien   * Sortir si c'est pas 9
            move.w      #$9000,port_c
            rts
* Si c'est en dehors de 0-9 on n'affiche l'affichage
afficher_rien
            move.w      #$FF00,port_c
            rts
* Fin du sous programme de transcodage et d'affichage
*******************************************************

* Fin du listing
*******
            end
```

3.2.4 Programme en langage C

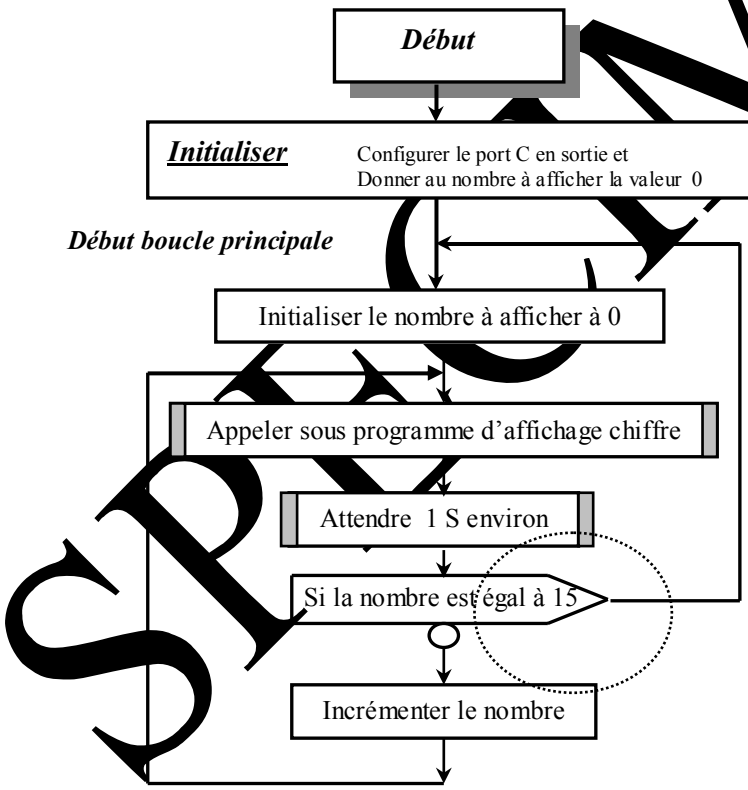
```
            /*************************************************************************
            *    TP sur  EID210 avec SIMULATEUR D'ENTREES SORTIES    *
            *************************************************************************
            *              Test du port C qui pilote l'afficheur 7 segments         *
            *               pour afficher un nombre décimal  ( 0 à 9)               *
            * CAHIER DES CHARGES  :                                                 *
            ***************************                                           *
            * L'afficheur affiche un nombre décimal (entre 0 et 9)                  *
            * qui s'incrémente environ toutes les secondes                          *
            * Ce temps sera généré par une boucle "logiciel"                        *
            *-----------------------------------------------------------------------*
            * NOM du FICHIER:  Aff_7S_D.C                                           *
            *********************                                                 *
            *************************************************************************/
#include "EID210_reg.h"
#include "Simulateur_ES.h"
#include "Structures_donnees.h"

//********************
//  DECLARATIONS
//********************
// Les prototypes des fonctions
void AFFICHER(unsigned char);
void ATTENDRE(unsigned char);
//==================================
//            FONCTION PRINCIPALE
//==================================
main()
{
//Varirables locales
unsigned char compteur;
//  Initialisations
//********************
Dir_Port_C=0xFF;     // Les 8 bits MSB sont valides
Port_C=0x00;
//Début de la boucle principale
//************************************
            do
            {           for (compteur=0;compteur<=9;compteur++)
                        {  AFFICHER(compteur);          //Appeler la fonction AFFICHER
                           ATTENDRE(1);                 //Appeler la fonction ATTENDRE
                        }
             compteur=0;                                /*Remise à 0 du compteur
            }while (1);
}
//Fin de la fonction principale
//******************************
//****************************************
//            FONCTION AFFICHER
//****************************************
void AFFICHER(unsigned char chiffre)
{switch(chiffre)
            {
            case 0 : PortC=AFF_0;break;
            case 1 : PortC=AFF_1;break;
            case 2 : PortC=AFF_2;break;
            case 3 : PortC=AFF_3;break;
            case 4 : PortC=AFF_4;break;
            case 5 : PortC=AFF_5;break;
            case 6 : PortC=AFF_6;break;
            case 7 : PortC=AFF_7;break;
            case 8 : PortC=AFF_8;break;
            case 9 : PortC=AFF_9;break;
            case 10 : PortC=AFF_10;break;
            case 11 : PortC=AFF_11;break;
            case 12 : PortC=AFF_12;break;
            case 13 : PortC=AFF_13;break;
            case 14 : PortC=AFF_14;break;
            case 15 : PortC=AFF_15;break;
            }
}

//==============================
//            FONCTION  ATTENDRE
//==============================
void ATTENDRE(unsigned char nb)
{int i;
            for(i=300000*nb;i>0;i--);           /* Décrémentation de la variable de temporisation */
}
```

3.2.5 Modifications à apporter pour satisfaire la Variante 2

Complément pour la représentation des lettres A, B C, D, E et F



Modification sur l'organigramme:



3.2.6 Programme en assembleur A68xxx de l'affichage hexadécimal (Variante 2)

```
        ************************************************************************
        *              TP  SUR EID210 AVEC SIMULATEUR D'ENTREES SORTIES        *
        ************************************************************************
        *              Test du port C qui pilote l'afficheur 7 segments        *
        *               pour afficher une grandeur hexadécimale (0 à 15)       *
        *                                                                      *
        *  Description:                                                        *
        ***************                                                       *
        *  L'afficheur affiche un nombre hexadécimal (entre 0 et 15)           *
        *  soit: 0,1,2,3,4,5,6,7,8,9,A,b,C,d E,F                               *
        *  qui s'incrémente environ toutes les secondes                        *
        *  Ce temps sera généré par une boucle "logiciel"                      *
        *                                                                      *
        *  NOM du FICHIER:  T_Port_C_2.SRC                                     *
        ******************                                                    *
        ************************************************************************

************************************
*       Déclaration des variables      *
************************************
        section     var
NOMBRE ds.b     1           *  de taille 8 bits

*  Inclusion
**************
* Inclusion du fichier définissant les différents labels
        include             EID210.def

************************************
*  Début du programme exécutable      *
************************************
        section             code

*       INITIALISER
**********************
*   Le port C
debut   move.w    #$FF00,DIR_Port_C  * Le port C est configuré en sortie
                                     *  Un 1 impose un fonctionnement en sortie

* Le nombre à afficher
        clr                 NOMBRE
        clr                 d0                  * Registre "do" est  image de NOMBRE
*       BOUCLE PRINCIPALE
****************************
Deb_BP * Début de la boucle principale
* Aller afficher le nombre courant
        jsr         AFFICHE
*  Boucle d'attente d'environ 1 seconde
        move.l              #$001F0FFF,d2
ATT     subq.l              #1,d2
        bne                 ATT

* Incrementer nombre
        cmp.b               #15,d0
        bne                 Etiq1               * Si c'est pas égal à 15 aller incrémenter
        clr.b               d0                  * Si c'est égal à 15 repartir de 0
        bra                 Etiq2
Etiq1   add.b               #1,d0
Etiq2   move.b              d0,NOMBRE

*  Passer au nombre suivant
        bra                 Deb_BP
* Fin de la boucle principale
*****************************
*****************************
* Fin du programme principal   *
*****************************
```

```
*******************************************************************************
*          Sous programe de transcodage      " Hexadécimal" -> 7 Segments   et d'affichage        *
*******************************************************************************
AFFICHER        cmp.b           #$00,d0
                bne             afficher1       * Sortir si c'est pas 0
                move.w          #$C000,Port_c
                rts
afficher1       cmp.b           #$01,d0
                bne             afficher2       * Sortir si c'est pas 1
                move.w          #$F900,Port_c
                rts
afficher2       cmp.b           #$02,d0
                bne             afficher3       * Sortir si c'est pas 2
                move.w          #$A400,Port_c
                rts
afficher3       cmp.b           #$03,d0
                bne             afficher4       * Sortir si c'est pas 3
                move.w          #$B000,Port_c
                rts
afficher4       cmp.b           #$04,d0
                bne             afficher5       * Sortir si c'est pas 4
                move.w          #$9900,port_c
                rts
afficher5       cmp.b           #$05,d0
                bne             afficher6       * Sortir si c'est pas 5
                move.w          #$9200,port_c
                rts
afficher6       cmp.b           #$06,d0
                bne             afficher7       * Sortir si c'est pas 6
                move.w          #$8200,port_c
                rts
afficher7       cmp.b           #$07,d0
                bne             afficher8       * Sortie si c'est pas 7
                move.w          #$F800,port_c
                rts
afficher8       cmp.b           #$08,d0
                bne             afficher9       * Sortir si c'est pas 8
                move.w          #$8000,port_c
                rts
afficher9       cmp.b           #$09,d0
                bne             afficher10      * Sortir si c'est pas 9
                move.w          #$9000,port_c
                rts
afficher10      cmp.b           #$0A,d0
                bne             afficher11      * Sortir si c'est pas 10 ($A)
                move.w          #$8800,port_c
                rts
afficher11      cmp.b           #$0B,d0
                bne             afficher12      * Sortir si c'est pas 11 ($B)
                move.w          #$8300,port_c
                rts
afficher12      cmp.b           #$0C,d0
                bne             afficher13      * Sortir si c'est pas 12 ($C)
                move.w          #$C600,port_c
                rts
afficher13      cmp.b           #$0D,d0
                bne             afficher14      * Sortir si c'est pas 13 ($D)
                move.w          #$A100,port_c
                rts
afficher14      cmp.b           #$0E,d0
                bne             afficher15      * Sortir si c'est pas 14 ($E)
                move.w          #$8600,port_c
                rts
afficher15      cmp.b           #$0F,d0
                bne             afficher_rien   * Sortir si c'est pas 15 ($F)
                move.w          #$8E00,port_c
                rts
afficher_rien   move.w          #$FF00,port_c   * Si c'est en dehors de $00 à $0F on éteint l'afficheur
                rts
* Fin du sous programme de transcodage et d'affichage
*********************************************
* Fin du listing
**************
                end
```

3.2.7 Programme en langage C

```
            /*************************************************************************
            *    TP sur  EID210 avec SIMULATEUR D'ENTREES SORTIES     *
            *************************************************************************
            *              Test du port C qui pilote l'afficheur 7 segments         *
            *               pour afficher un nombre décimal  ( 0 à 9)               *
            *  CAHIER DES CHARGES  :                                                *
            ***************************                                           *
            *  L'afficheur affiche un nombre décimal (entre 0 et 9)                 *
            *  qui s'incrémente environ toutes les secondes                         *
            *  Ce temps sera généré par Timer en interruption                       *
            *-----------------------------------------------------------------------*
            *  NOM du FICHIER:  Aff_7S_HI.C                                         *
            *********************                                                 *
            *************************************************************************/
#include "EID210_reg.h"
#include "Simulateur_ES.h"
#include "CPU_reg.h"
#include "Structures_Donnees.h"

//********************
//  DECLARATIONS
//********************
// Les prototypes des fonctions
void AFFICHER(unsigned char);
/* Déclaration des variables pour la temporisation */
int CPTR_mS;                                        /* déclaration des variables et compteurs de millisecondes */
unsigned char Fin_tempo,Tempo_en_cours;             /*  déclaration variables pour gestion temporisation

/* FONCTION d'interruption pour temporisation  (base de temps: toutes les mS)
================================================================*/
void irq_bt()
{           if (Tempo_en_cours==1)
                    { CPTR_mS++;
                if (CPTR_mS ==1000) Fin_tempo=1,Tempo_en_cours=0,CPTR_mS=0;
                    }
} // Fin de la définition de la fonction d'interruption pour la temporisation


//============================
//          FONCTION PRINCIPALE
//============================
main()
{
//Varirables locales
//********************
unsigned char compteur;

//  Initialisations
//*****************
// Pour le port C
Dir_Port_C=0xFF;    // Tous les bits du port C sont en sortie
Port_C=0x00;                    // On éteint tous les segments
// Pour base de temps
CPTR_mS=0;                              // initialisation compteurs millisecondes
SetVect(96,&irq_bt);                    // Mise en place de l'autovecteur
PITR = 0x0008;                          // Une interruption toutes les millisecondes
PICR = 0x8060;                          // 96 = 60H
Tempo_en_cours=0;                       // Temporisation inactive

//Début de la boucle principale
//*****************************
            do
            {           for (compteur=0;compteur<=15;compteur++)
                        {
                                    // AFFICHER
                                    Port_C=Tab_Valeurs_7Seg[compteur];
                                    //Pour attendre 1S
                                    Tempo_en_cours=1,Fin_tempo=0;           // Initialisation pour tempo
                                    do{}while(Fin_tempo==0);                // Attendre fin tempo
                        }
             /*Remise à 0 du compteur */
             compteur=0;
            }while (1);
}
//Fin de la fonction principale
//*******************************
```

TP 4 VISUALISER LA POSITION DU POTENTIOMETRE

4.1 Sujet

Objectif :	**Capacités complémentaires:** Etre capable de configurer un convertisseur Analogique -> numérique. Etre capable de détecter la fin d'une conversion Analogique -> numérique.
Cahier des charges :	Visualiser sur l'afficheur 7 segments les 4 bits de poids fort du résultat de conversion de l'entrée analogique AN sang 0 sur laquelle est connecté le point milieu du potentiomètre.

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 210 000
Câble de liaison USB ou à défaut câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,
Simulateur d'entrées sorties réf : EID001000

Durée : 2 heures

4.2 Eléments de solution

Convertir une entrée analogique:
Le convertisseur Analogique -> Numérique utilisé a pour référence MAX196. Il est capable de convertir 6 entrées analogiques. L'entrée que l'on souhaite convertir est le canal de rang '0'.
Une demande de conversion s'effectue par une écriture d'un mot de contrôle dont le format est le suivant:

Les bits A2, A1 et A0 définissent le rang de l'entrée analogique à convertir.

Les 2 bits RNG et BIP définissent la plage de variation de l'entrée analogique sélectionnée.

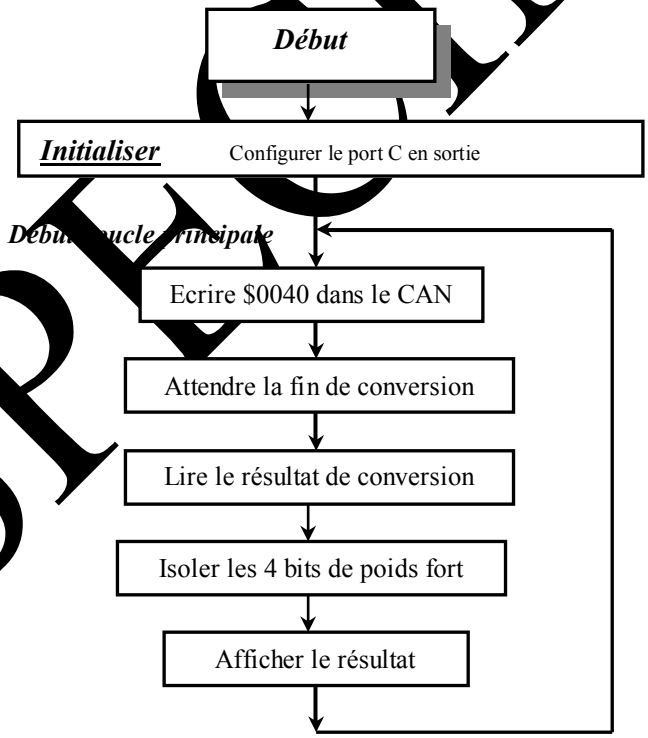
Les bits PD1,PD0, ACQ devront être initialisé à 010.

RNG	BIP	Plage
0	0	0 à 5V
0	1	0 à 10V
1	0	+-5V
1	1	+-10V

A2	A1	A0	rang entrée Analogique
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5

Il faudra donc écrire le mot 0100 0000 = $0040 à l'adresse du CAN (définie dans le fichier EID210.def, fichier qu'il faudra inclure).
La conversion n'est pas immédiate. La fin de conversion peut être connue en testant le bit de rang 14 du régime d'état.

4.2.1 Organigramme



4.2.2 Programme en assembleur A68xxx

```
        ********************************************************************************************
        *               TP  SUR EID210 AVEC SIMULATEUR D'ENTREES SORTIES                           *
        ********************************************************************************************
        *               Affichage sur 7 segments du résultat de conversion del'entrée analogique   *
        *               n° 0 (celle du potentiométre)                                              *
        *                                                                                          *
        *  Cahier des charges:                                                                     *
        ******************                                                                        *
        *  On affiche un nombre hexadécimal (entre 0 et 15)  qui correspond aux poids forts du résultat de conversion  *
        * de l'entrée analogique E0   (imposée par la position du potentiomètre)                   *
        *                                                                                          *
        *  NOM du FICHIER:  T_CAN_Port_C.SRC                                                       *
        ******************
        ********************************************************************************************

*  Inclusion
**************
* Inclusion du fichier définissant les différents labels
        include                 EID210.def

************************************
*  Début du programme exécutable           *
************************************
        section                 code

*       INITIALISER
********************
*   Le port C
debut           move.w          #$FF00,DIR_Port_C               * Le port C est configuré en sortie
                                                                  Un 1 impose un fonctionnement en sortie

*       BOUCLE PRINCIPALE
******************************
Deb_BP * Début de la boucle principale
* Lancer la conversion Analogique -> Numérique de la voie n°0
                move.w          #$0040,CAN
* Attendre la fin de conversion (test du bit fin de conversion A->N du registre d'état)
Att_F_C move.w  REG_ETAT,D0
                and.w           #$4000,D0
                bne             Att_F_C
* Lire le résultat de conversion
                move.w          CAN,d0
                and.w           #$0FFF,d0
* Ne prendre que les 4 bits de poids fort
                lsr.w           #8,D0
* Aller afficher le résultat
                jsr             AFFICHER
*  Boucle -> Recommencer
                bra             Deb_BP
* Fin de la boucle principale
********************************
*********************************
* Fin  du programme  principal   *
*********************************

*********************************************
*  Sous programe de transcodage             *
*   " Hexadécimal"  ->  7 Segments          *
*   et d'affichage                          *
*********************************************
AFFICHER        cmp.b           #$00,d0
                bne             afficher1               * Sortir si c'est pas 0
                move.w          #$C000,Port_c
                rts
```

```
afficher1       cmp.b       #$01,d0
                bne         afficher2       * Sortir si c'est pas 1
                move.w      #$F900,Port_c
                rts
afficher2       cmp.b       #$02,d0
                bne         afficher3       * Sortir si c'est pas 2
                move.w      #$A400,Port_c
                rts
afficher3cmp.b  #$03,d0
                bne         afficher4       * Sortir si c'est pas 3
                move.w      #$B000,Port_c
                rts
afficher4cmp.b  #$04,d0
                bne         afficher5       * Sortir si c'est pas 4
                move.w      #$9900,port_c
                rts
afficher5       cmp.b       #$05,d0
                bne         afficher6       * Sortir si c'est pas 5
                move.w      #$9200,port_c
                rts
afficher6       cmp.b       #$06,d0
                bne         afficher7       * Sortir si c'est pas 6
                move.w      #$8200,port_c
                rts
afficher7       cmp.b       #$07,d0
                bne         afficher8       * Sortie si c'est pas 7
                move.w      #$F800,port_c
                rts
afficher8       cmp.b       #$08,d0
                bne         afficher9       * Sorti[illegible]8
                move.w      $8000,port_c
                rts
afficher9       cmp.b       #$09,d0
                bne         afficher10      * So[illegible]i c'est pas 9
                move.w      #$9000,port_c
                rts
afficher10      cmp.b       #$0A,d0
                bne         afficher[illegible]  * Sortir si c'est p[illegible]10 ($A)
                move.w      #$8800[illegible]rt_c
                rts
afficher11      cmp.b       #$0B,d0
                bne         afficher12      * Sorti[illegible]si c'est pas 11 ($B)
                move.w      #$83[illegible]0,port_c
                rts
afficher12      cmp.b       #$0C,d0
                bne         [illegible]fficher13    * Sortir si c'est pas 12 ($C)
                m[illegible]ve.w    [illegible]600,port_c
                r[illegible]
afficher13      [illegible].b       #$0D,d0
                bne         afficher14      * Sortir si c'est pas 13 ($D)
                move.w      #$A100,port_c
                rts
afficher14      cmp.b       #$0E,d0
                [illegible]         afficher15      * Sortir si c'est pas 14 ($E)
                [illegible]ve.w     #$8600,port_c
                rts
afficher15      cmp.b       #$0F,d0
                bne         afficher_rien   * Sortir si c'est pas 15 ($F)
                [illegible]ove.w    #$8E00,port_c
                rts
* Si c'est en dehors de $00 à $0F on éteint l'afficheur
afficher_rien   move.w      #$FF00,port_c
                rts
* Fin du sous programme de transcodage et d'affichage
************************************************************

* Fin du listing
***************
                end
```

4.2.3 Programme en langage C

```
/*****************************************************************************
*    TP sur  EID210 avec SIMULATEUR D'ENTREES SORTIES     *
*****************************************************************************
*              Affichage sur 7 segments de l'entrée analogique      *
*              n° 0 (celle du potentiométre)                        *
*  Cahier des charges:                                              *
* *************************                                         *
*  On affiche un nombre hexadécimal (entre 0 et 15)                 *
*  qui correspond aux poids forts du résultat de conversion         *
*  de l'entrée analogique E0                                        *
*  (imposée par la position du potentiomètre)                       *
*  On affiche également les 8 bit MSB sur le port A                 *
*  On applique les 8 bit MSB sur la sortie analogique SA0           *
*------------------------------------------------------------------  *
*  -> Tournez le potentiométre pour voir s'afficher la valeur       *
*  ------------------------------------------------------------    *
*  NOM du FICHIER:  CAN_Port_C&PA&SA0.C                             *
*********************                                               *
*****************************************************************************/
/* Liste des fichiers à inclure */
#include "Cpu_reg.h"
#include "EID210_reg.h"
#include "Simulateur_ES.h"
#include "Structures_donnees.h"
//*************************
//    DECLARATIONS     *
//*************************
/* Declaration des prototypes des fonctions */
void Afficher_Sur_PA(unsigned char);       // Pour afficher résultat sur le port A
//***********************************************
//              FONCTION PRINCIPALE
//***********************************************
main()
{
/*Variable locale*/
unsigned short resultat_conv;
unsigned char val_aff;
//  Initialisations
//*****************
/* Initialisation du port A en sortie*/
CFSR3=0x8888;                   /*  CHA0 à CHA3 en mode "DIO"            */
CFSR2=0x8888;                   /*  CHA4 à CHA7 en mode "DIO"            */
CPR1=0xFFFF;                    /*  Tous les bits en priorité haute      */

/*Initialisation de l'afficheur*/
DirPortC=0xFF;                  /*  Le port C en sortie                  */
//Début de la boucle principale
//*********************************
 do
   {        CAN=0x0040;                        // Lancer la conversion de la voie 0
            do{}while(Fin_CAN==1);             // Attendre la fin de la conversion
            resultat_conv=CAN;                 // Lire le résultat de conversion
            //masquer les bits de poids faibles pour afficher les 4 bits de poids fort sur 7 Segments
            val_aff=(unsigned char) ((resultat_conv&0x0FFF)>>8);
            Port_C=Tab_Valeurs_7Seg[val_aff] ;
            val_aff=(unsigned char) ((resultat_conv&0x0FFF)>>4);
            Afficher_Sur_PA(val_aff);          // Sortir sur le port A
            SA0=val_aff;                       // Sortir sur la sortie analogique 0
    }while(1);
}       // Fin de la boucle principale

void Afficher_Sur_PA(unsigned char etatPA)
{           if(etatPA&0x01)PA0=un;
             else PA0=zero;
            if(etatPA&0x02)PA1=un;
             else PA1=zero;
            if(etatPA&0x04)PA2=un;
             else PA2=zero;
            if(etatPA&0x08)PA3=un;
             else PA3=zero;
            if(etatPA&0x10)PA4=un;
             else PA4=zero;
            if(etatPA&0x20)PA5=un;
             else PA5=zero;
            if(etatPA&0x40)PA6=un;
             else PA6=zero;
            if(etatPA&0x80)PA7=un;
             else PA7=zero;
            HSRR1=Image_PA;
}
```

TP 5 COMPTAGE DES COMMUTATIONS DE L'ENTREE SW2 EN INTERRUPTION

5.1 Sujet

Objectif :	**Capacités complémentaires:** Etre capable de gérer une entrée logique en interruption.
Cahier des charges :	Visualiser sur l'afficheur 7 segments les 4 bits de poids fort du résultat de compteur qui s'incrémente à chaque fois que l'on commute le commutateur repéré "SW2" sur la carte de simulation. L'incrémentation du compteur s'effectuant dans un programme d'interruption. Les 8 bits du compteur sera également affiché sur les 8 LEDs connectées sur le portA

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 210 000
Câble de liaison USB ou à défaut câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,
Simulateur d'entrées sorties réf : EID001000

Durée : 2 heures

5.2 Eléments de solution

Programme en langage C

```
/*************************************************************************
*    TP sur  EID210 avec SIMULATEUR D'ENTREES SORTIES    *
*************************************************************************
*              Comptage du nombre de commutation sur l'entrée   *
*               repérée IRQ sur la carte de simulation.         *
*               La commutation générera une interruption.       *
*              Ce nombre est affiché sur l'afficheur 7 Segments *
*           (les 4 bits LSB) et sur les 8 leds du portA.        *
* Cahier des charges:                                           *
* ********************                                          *
* A chaque commutation un compteur s'incrémente  dans un        *
*  programme d'interruption associé                             *
* La résultat de comptage est actualisé dans la fonction        *
*  principale                                                   *
* ------------------------------------------------------------- *
* NOM du FICHIER:  Compt_IRQ.SRC                                *
* ********************                                          *
*************************************************************************/
/* Liste des fichiers à inclure */
#include "Cpu_reg.h"
#include "EID210_reg.h"
#include "Simulateur_ES.h"
#include "Structures_donnees.h"
//************************
//    DECLARATIONS     *
//************************
// Du prototype de la fonction d'affichage sur le portA
void Afficher_Sur_PA(unsigned char );
// Des variables globales
unsigned char compteur;
unsigned char Flag_Aff;
// De la fonction d'interruption
void irq_IRQ()
{           compteur++;
            Flag_Aff=1;
}
//*********************************************
//            FONCTION PRINCIPALE
//*********************************************
main()
{ //  Initialisations
  /* Pour port A en sortie */
  CFSR3=0x8888;                 /* CHA0 à CHA3 en mode "DIO" */
  CFSR2=0x8888;                 /* CHA4 à CHA7 en mode "DIO" */
  CPR1=0xFFFF;                  /* Toutes les bits en priorité haute          */
  compteur=0;  // Pour le comptage des commutations
  Flag_Aff=0;
  // Mise en place du vecteur d'interruption
  SetVect(30,&irq_IRQ);                 // mise en place de l'autovecteur  avec la fonction irq_IRQ()
  // Autoriser les interruptions du bouton IRQ (connecté sur entrée IRQ4 du microcontrôleur)
  Valid_Irq4=1;
  DirPortC=0xFF;                /*            Le port C en sortie             */
//Début de la boucle principale
//*****************************
    do
     {      do{}while(Flag_Aff==0);          /*Attendre ordre d'affichage ... donc d'une commutation de l'interrupteur "IRQ" */
            Afficher_Sur_PA(compteur);    // On affiche les 8 bits sur le port A
            Port_C=Tab_Valeurs_7Seg[compteur&0x0F] ; // On affiche les 4 bits de poids faible sur 7 Segments
            Flag_Aff=0;                   // Pour attendre la prochaine commutation
     } while(1);
}           //Fin de la boucle principale
void Afficher_Sur_PA(unsigned char etatPA)
{           if(etatPA&0x01)PA0=un;
              else PA0=zero;
            if(etatPA&0x02)PA1=un;
              else PA1=zero;
            if(etatPA&0x04)PA2=un;
              else PA2=zero;
            if(etatPA&0x08)PA3=un;
              else PA3=zero;
            if(etatPA&0x10)PA4=un;
              else PA4=zero;
            if(etatPA&0x20)PA5=un;
              else PA5=zero;
            if(etatPA&0x40)PA6=un;
              else PA6=zero;
            if(etatPA&0x80)PA7=un;
              else PA7=zero;
            HSRR1=Image_PA;
}
```

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_donnees.h

```
//=====================================================================
//          STRUCTURES DE DONNEES UTILES A LA PROGRAMMATION EN  C
//=====================================================================
//          Nom du fichier:
//          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 séparant les 16 bits par ensembles de 2 bits  (Utile pour les sorties TPU)
//*************************************************************************************************/
          union word_duo
          {         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 2003
 *          Auteurs:  J. 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 0
#define CFSR1       *(short *)(0xFFFE0E)        // Permet de définir la fonction souhaitée pour chacun des
#define CFSR2       *(short *)(0xFFFE10)        // 15 canaux (lignes) d'entrée-sortie (TPU0 à TPU15)
#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)         // Control
#define CTRL_TPU1  *(short *)(0xFFFF10)         // Une zone de 8 mots d'état 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 su
#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 Anal      ue voie 0  (SA0) */
#define    SA1                 (*(BYTE*)         (CNA+0x02))       /*          Sortie Analog    voie 0  (SA0)    */
#define    SA2                 (*(BYTE*)         (CNA+0x04))       /*          Sortie Analogiq      ie 0  (SA0)  */
#define    SA3                 (*(BYTE*)         (CNA+0x06))       /*          S      Analogique      0  (SA0) */
#define    CAN                 (*(WORD*)         0xB20000)         /*  Convertis      a -> Num      CS6      */
#define    BUS                                   0xB30000          /*  Ad. de base      . 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      g ne tourne pas si RESET!  */
 #define E_Irq_CAN              REG_ETAT.bits_octet.b14     /*      t du Bit d      ion Ana -> Num        */
 #define E_Irq_USB              REG_ETAT.bits_octet.b13     /*      t du Bit d'      de la ligne interruption USB */
 #define E_Irq4_Bus             REG_ETAT.bits_octet.b12     /* E      u Bit d'état ligne IRQ4 du BUS        */
 #define E_Irq3_Bus             REG_ETAT.bits_octet.b11     /* Etat      Bit d'état ligne IRQ3 du BUS       */
 #define E_Irq2_Bus             REG_ETAT.bits_octet.b10     /* Etat d      d'état ligne IRQ2 du BUS         */
 #define E_Irq1_Bus             REG_ETAT.bits_octet.b9      /* Etat du      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 VERSI      soft PLD sur 8 bits        */


/*  Pour autoriser (valider) les interruption      */
#define VALID_IRQs  (*(union word_octet_bits*) (0x90      ))
#define Valid_IrqCtrl           VALID_IRQs.octet_bits.
#define Valid_IrqCan            V      D_IRQs.octet_bits.b2
#define Valid_Irq1              VA      IRQs.octet_bits.b3
#define Valid_Irq2              VALI      Qs.octet_bits.b
#define Valid_Irq3              VALID_I      octet_bits.b5
#define Valid_Irq4              VALID_IRQ      ctet_bits.b
#define Valid_IrqUsb            ID_IRQs.      t_bits.b7
#define Valid_unus              V      D_IRQs.oc      ts.b0

// Pour une ge      ion direct      Port C
#define Po          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
```

ANNEXE 3 : PLAn du simulatateur d'entrées sorties