EXPERIMENTS MANUAL

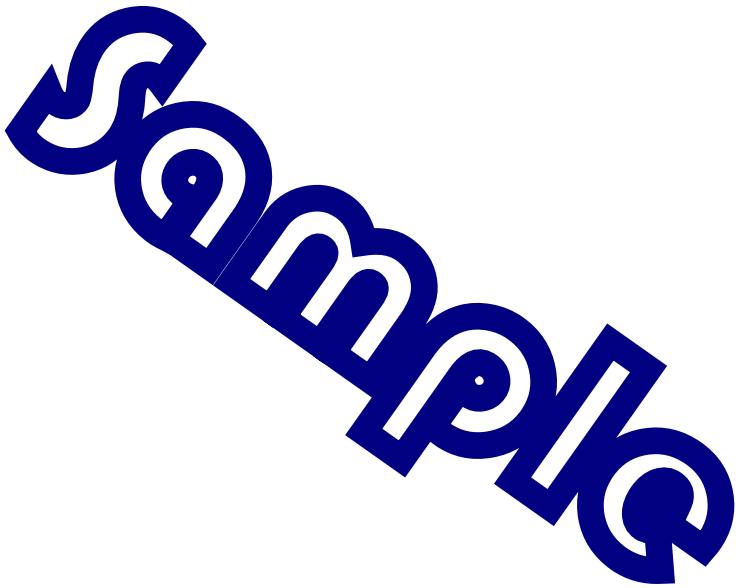
On EID210 System
+
"Inputs / Outputs Simulator" Module

DMS Didalab Matelco Stci

Z.A.C de La Clé St Pierre
5, rue du Groupe Manoukian
78990 ELANCOURT
Tél. : (33) 01 30 66 08 88
Fax : (33) 01 30 66 72 20

Issued on : 31/08/15

Réf. : EID211041


Sample

SUMMARY

Sample

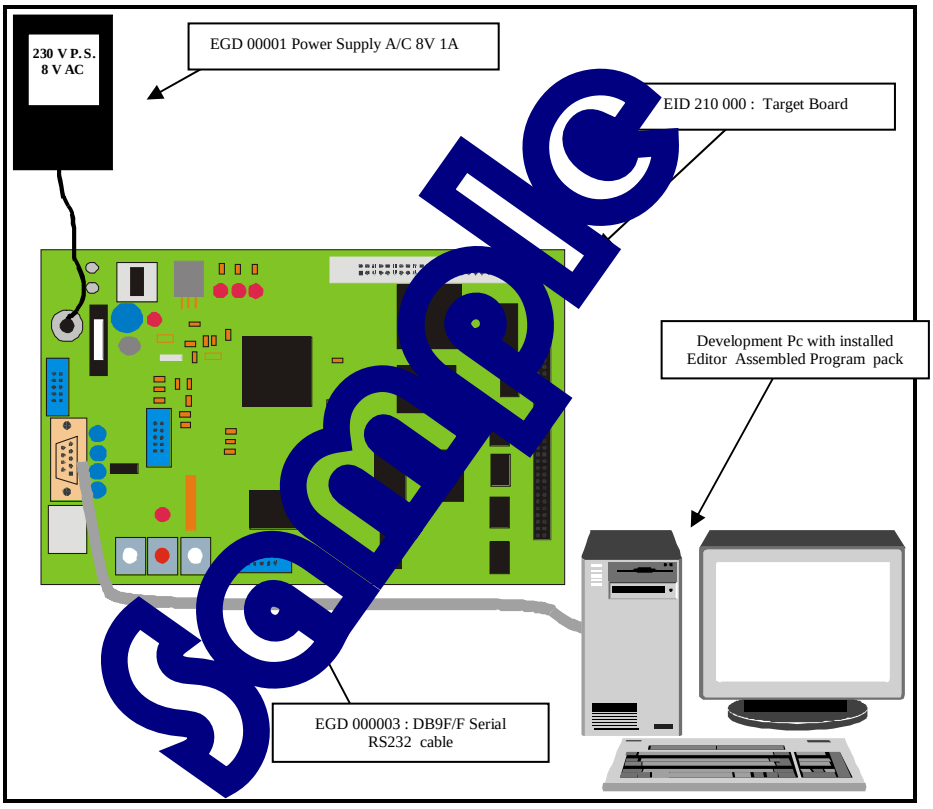
INSTALLATION OF THE EQUIPMENT

-> **Connect** the EID 210 000 board to the development PC with Assembler language (provided together with the equipment and beforehand installed in accordance with the technical manual), in using the USB cable or if unavailable, the RS232 serial cable.

-> **Plug** the Power Supply onto the EID 210 000 Board , (from 7 to12 V AC or DC),

-> **Connect** the EID 001 000 Inputs Outputs Simulator to the EID 210 000(CPU Board).

-> **Press** down the EID 210 000 Board On/Off pushbutton, the red light must go on.



Sample

EXPERIMENT N° 1- SEQUENTIAL SHIFT DEVICE WITH LEDS ONTO PORT A

1.1 Topic

Object :	Being capable of configuring one port into Output and activating these Outputs (This port is one element of the micro-controller « TPU »). Being capable of analysing imposed specifications and writing a program for complying with it. Being capable of carrying out a program time delay, then using the Micro-controller "TIMER" function. Being capable of carrying out a program in Assembler language
Specifications :	A light shift device is carried out in using the 8 Leds connected onto port A of EID210 Processor board. Which leads to the following sequence: D8 on, then wait for about 0,5S, then D7 on, then wait for 1S, etc… After D1 turning on, there is a loop, which means that we come back to D8. ***Variant n°1 :*** The delay is carried out by program waiting loop. ***Variant n°2 :*** The delay is carried out in using the micro-controller internal "TIMER".

Necessary Equipment :

PC Micro-Computer using Windows ® 95 or latter,
68332 Micro-Controller 16/32 bits Mother Board, Ref. : EID 100 000
USB link cable or if unavailable, RS232 cable, Ref. : EGD 000 003
AC/AC 8V Power Supply, 1 A, Ref. : EGD000001,
Input / Output Simulator, Ref. : EID001000

Duration : 4 hours

1.2 [2] Elements of solution

1.2.1 Outputs activation

Configure the port bits onto Output:

Following the Simulator Board layout as given in ANNEX, the LEDs are connected onto port A of the EID210 Processor Board.
Port A is connected onto Outputs from CHANNEL0 (TPU0) <-> PA0 to CHANNEL7 (TPU7) <-> PA7 of the Micro-controller.
The operating mode of a micro-controller TPU line is determined in loading a 4 bits code into a register dedicated to this purpose (Registers CFSRi with i=0,1,2,3 on 16 bits, as specified on file to be included "**EID210.def**"), following the next correspondence:

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

In this case, the requested operating mode is a simple Output mode.
In the 68332 Micro-controller Technical Data Handbook, and more particularly in Chapter dedicated to the "**TPU**", this simple Inputs/Outputs mode is called **DIO** (**D**iscrete **I**nput **O**utput).
In this case, the 4 bits binary code to be loaded into configuration registers must be: 1000 = $8

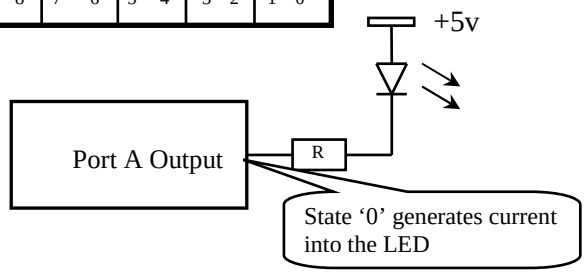
Let instructions enabling configuration of port B be in mode "DIO":

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

Activate one bit of Output port (Port A) to logic state '1' or '0'

For setting one bit of port A to logic state '1', binary binomial "0 1" must be set into the corresponding location of register HSSR1 (which address is specified into the definition file EID210.def). For setting to logic state '0', value "1 0 " must be set to the same binomial.

Led N°	D1	D2	D3	D4	D5	D6	D7	D8
Bit N° into port A	7	6	5	4	3	2	1	0
TPU "CHANNEL" N°	7	6	5	4	3	2	1	0
Bits of register HSSR1	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0



For switching a Led on, a logic state '0' must be written into the bit of the corresponding port
(As indicated on next digit)

Example :
If we want Led D8 switching on, and the others off, register HSSR1 must be loaded by the value :

N° of the concerned diode	:	D0	D2	D3	D4	D5	D6	D7	D8
Value Output port :		1	1	1	1	1	1	1	0
Value into register HSRR1 :		01	01	01	01	01	01	01	10 = $5556

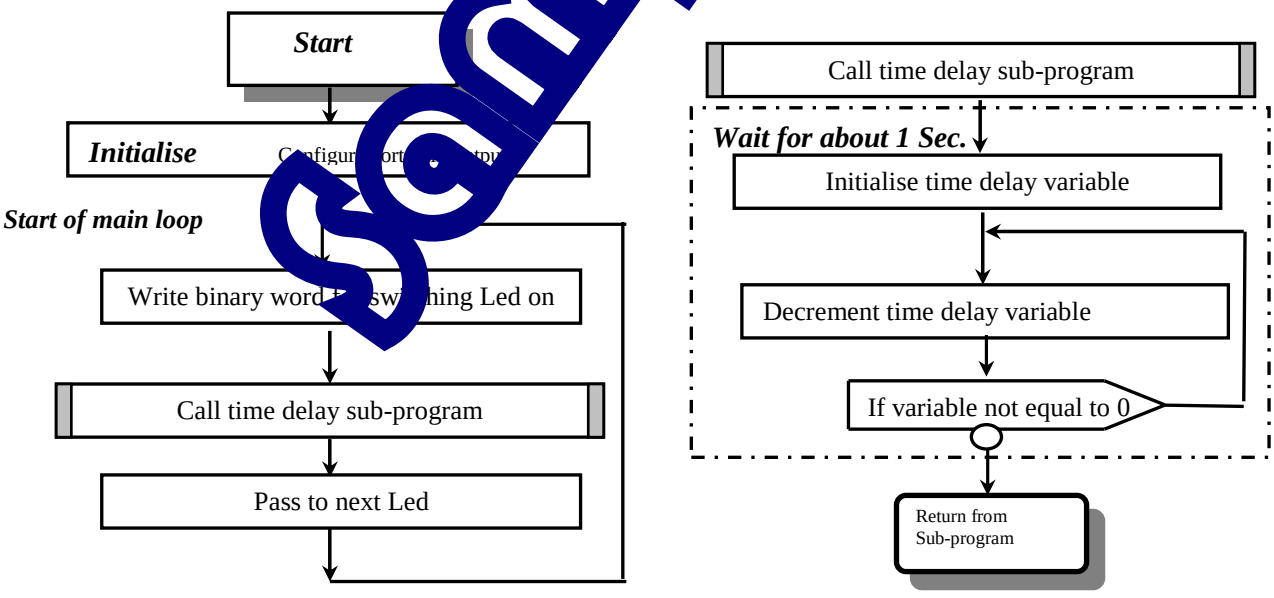
For carrying out the light shift device following the sequence requested by specifications, the following sequence must be written into register HSSR1:

N° Leds	D1	D2	D3	D4	D5	D6	D7	D8	
Logic states	1	1	1	1	1	1	1	0	
Value register HSSR1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	1 0	-> $5556
Logic states	1	1	1	1	1	1	0	1	
Value register HSSR1	0 1	0 1	0 1	0 1	0 1	0 1	1 0	0 1	-> $5559
Logic states	1	1	1	1	1	0	1	1	
Value register HSSR1	0 1	0 1	0 1	0 1	0 1	1 0	0 1	0 1	-> $5565
Logic states	1	1	1	1	0	1	1	1	
Value register HSSR1	0 1	0 1	0 1	0 1	1 0	0 1	0 1	0 1	-> $5595
Logic states	1	1	1	0	1	1	1	1	
Value register HSSR1	0 1	0 1	0 1	1 0	0 1	0 1	0 1	0 1	-> $5655
Logic states	1	1	0	1	1	1	1	1	
Value register HSSR1	0 1	0 1	1 0	0 1	0 1	0 1	0 1	0 1	-> $5955
Logic states	1	0	1	1	1	1	1	1	
Value register HSSR1	0 1	1 0	0 1	0 1	0 1	0 1	0 1	0 1	-> $6555
Logic states	0	1	1	1	1	1	1	1	
Value register HSSR1	1 0	0 1	0 1	0 1	0 1	0 1	0 1	0 1	-> $9555

Carrying out of programmable time delay:

A programmable time delay is carried out by initia[illegible] at a determined value and decrementing this value till it becomes equal to zero[illegible] decrementation loop operation duration, carried out "n" times (n = initial value of the v[illegible]) [illegible]tes the required lapse of time. On the program thereafter, the variable is included int[illegible]

1.2.2 Flowchart, solution with p[illegible]ime delay :



1.2.3 Program in A68xxx Assembler with time delay loop

```
        ********************************************************************************
        *  EXPERIMENTS ON EID210 BOARD WITH INPUTS/OUTPUTS SIMULATOR       *
        *               Carrying  out of light shift device on Leds of port A              *
        ********************************************************************************
        * Specifications :                                                               *
        ******************                                                               *
        *  Rem: These LEDs are connected to  port A                                      *
        *         68332 TPU lines: from TPU0 to TPU7  (CHA0  to CHA7)                     *
        *    Leds switch on following the cycle                                          *
        *            D8 -> D7 -> .... -> D1  then  -> D8 -> D7  ... etc                  *
        *    Leds switch on by applying a 0              PA0=0 -> PA1=0  .... etc        *
        *    Led lighting time duration is about 1S                                      *
        *    This time is generated by "program" loop                                    *
        *                                                                                *
        *  FILE NAME:  CHENI_1_PA.SRC                                                    *
        ***********************                                                         *
        ********************************************************************************
        * Inclusion of file specifying the different  labels
        include EID210.def
        section                 code
**************************************
*       MAIN PROGRAM                  *
**************************************
*       INITIALISE
*************************
* Configure port A into  "Discrete Input Output" mode -> code $8
START   move.w  #$8888,CFSR3            * CHA0 to CHA3 into "DI  de
        move.w  #$8888,CFSR2            * CHA4 to CHA7 i   DIO
        move.w  #$FFFF,CPR1             * All  bits in first  orit

*        MAIN LOOP
**************************************
Deb_BP move.w   #$5556,HSRR1            * Only D  ghts
        jsr     ATT
        move.w  #$5559,HSRR1            * On  lig
        jsr     ATT
        move.w  #$5565,HSRR1            * C  y D  u
        jsr     ATT
        move.w  #$5595,HSRR1            *  igh  p
        jsr     ATT
        move.w  #$5655,HSRR1            *  y D4  hts up
        jsr     ATT
        move.w  #$5955,HSRR1            * Only  lights up
        jsr     ATT
        move.w  #$6555,HSRR1            * Only D2 lights up
        jsr     ATT
        move.w  #$9555,HSRR1            * Only D1 lights up
        jsr     ATT
        bra     Deb_BP
*       END of main program
**********************************

**************************************
*       SUB-PROGRAMS      *
**************************************

*       Waiting sub-program of about  1 second
************************************************
ATT     move.l  #$001FFFFF,d2
ATT1    sub.l   #1,d2
        bne     ATT1
        rts                             * Return from sub-program
END of sub-program
*******************

        end
*  END of Assembler source file
**********************************
```

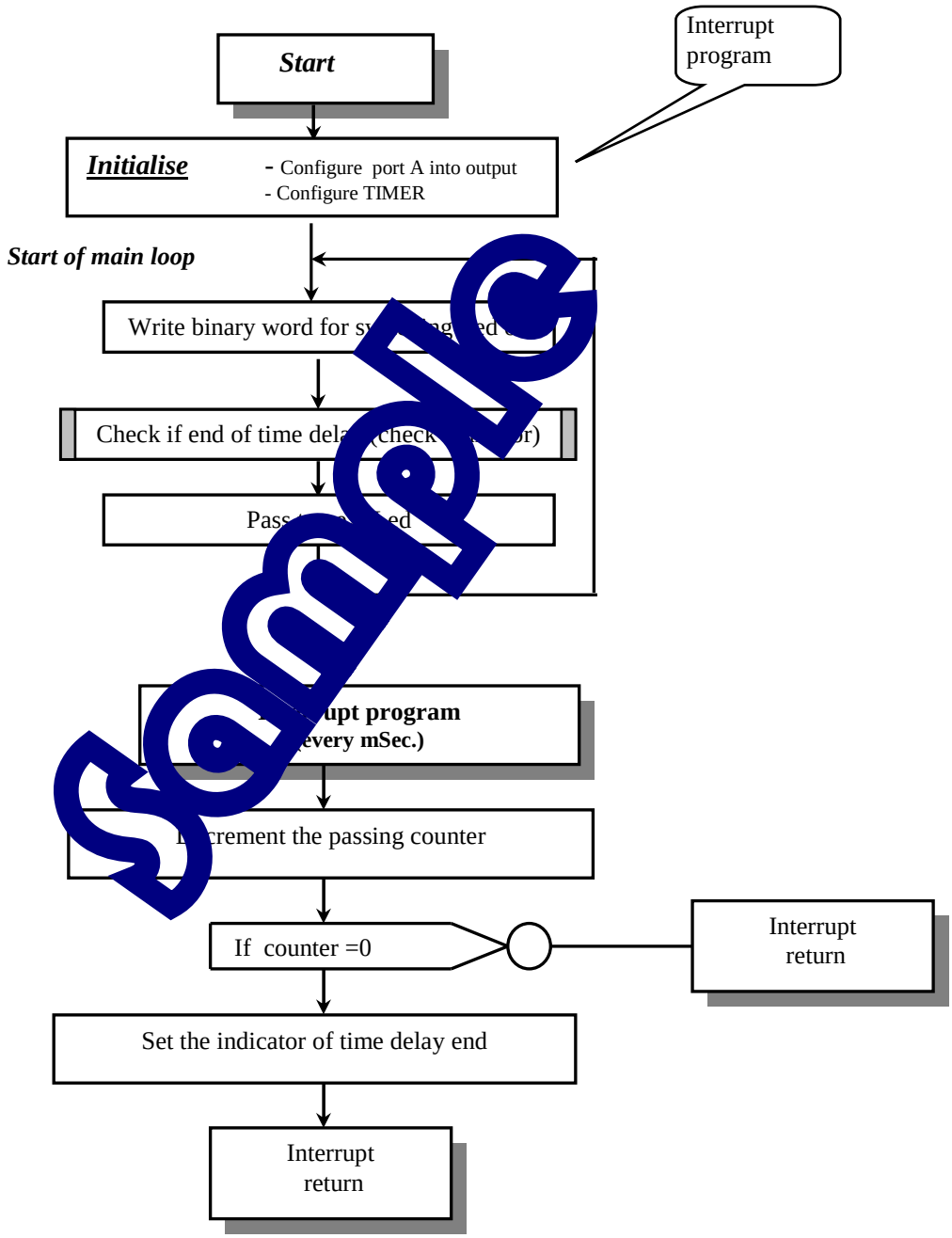
1.2.4 Flowchart, solution with time delay carried out by micro-controller "Timer" function

For having periodic interruption every 1 mS, both registers which labels have been specified in file EID210.def, must be initialised :

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

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

On the other hand, the vector table must be initialised and interrupt program must be allowed.



1.2.5 *Program in A68xxx Assembler, solution with time delay carried out by micro-controller "Timer" function*

```
*************************************************************************
*  EXPERIMENTS ON EID210 BOARD WITH INPUT/OUTPUT  SIMULATOR             *
*              Carrying out of light shift device on Leds of port A     *
*************************************************************************
* Specification  :                                                      *
******************                                                      *
*  Rem:       Lines of port A are  connected to lines of  68332 TPU:    *
*                     TPU0 to TPU7  (CHA0  to CHA7)                     *
*              TPU lines of 68332: TPU0 to TPU7  (CHA0  to CHA7)        *
*              Leds switch on by applying a  0                          *
*                     PA0=0 -> PA1=0  .... etc                          *
*              Led lighting time duration is about 1S                   *
*              This time is generated by the 68332 Timer                *
*  FILE NAME:  CHENI_2_PA.SRC  (solution N°2)                           *
******************                                                      *
*************************************************************************
* Inclusion of the file specifying the different  labels
include EID210.def

*************************************
*         Declaration of  variables         *
*************************************
        section     var
COUNTER         ds.l        1
INDICATOR       ds.b        1

*************************************
*  Start of execute program                 *
*************************************
        section             code
*       INITIALISE
**************************
* Configure port A into "Discrete Input Output" mode -> c
START   move.w          #$8888,CFSR3        *  to CHA3 in "DIO" mode
        move.w          #$8888,CFSR2        *  4 to CHA7 in "DIO" mode
        move.w          #$FFFF,CPR1         *  bits in first priority
        move.w          #$0003,CTRL_
* Configure the time base
        move.l          #96,                * 96 is the interrupt vector n°
        move.l          #it_b1              * f_it-bt is the interrupt function address
        asl.l           #2,d0
        add.l           #tab_ve             * Initialise the vectors table
        move.l          d0,a0
        move.l          a1,(a0)
        move.l          #1000,COMPTEUR      * 1000*1mS = 1S
        move.b          #$00,INDICATEUR     * end of counting
        move.w          #$0008,PITR         * 1 interrupt every 1 ms
        move.w          #$0760,PICR

*       MAIN LOOP
*******************************
Deb_BP move.w           #$5556,HSRR1        * Only  D8 lights up
ATT1    move.b          INDICATEUR,D2       * Waiting for end of time delay
        cmp.b           #01,D2
        bne             ATT1
        move.l          #1000,COMPTEUR      * 1000*1mS = 1S
        move.b          #$00,INDICATEUR     * end of counting

        move.w          #$5559,HSRR1        * Only D7 lights up
ATT2    move.b          INDICATEUR,D2       * Waiting for end of time delay
        cmp.b           #01,D2
        bne             ATT2
        move.l          #1000,COMPTEUR      * 1000*1mS = 1S
        move.b          #$00,INDICATEUR     * end of counting
        move.w          #$5565,HSRR1        * Only D6 lights up
```

```
ATT3    move.b      INDICATEUR,D2       * Waiting for end of time delay
        cmp.b       #01,D2
        bne         ATT3
        move.l      #1000,COMPTEUR      * 1000*1mS = 1S
        move.b      #$00,INDICATEUR     * end of counting

        move.w      #$5595,HSRR1        * Only D5 lights up
ATT4    move.b      INDICATEUR,D2       * Waiting for end of time delay
        cmp.b       #01,D2
        bne         ATT4
        move.l      #1000,COMPTEUR      * 1000*1mS = 1S
        move.b      #$00,INDICATEUR     * end of counting

        move.w      #$5655,HSRR1        * Only D4 lights up
ATT5    move.b      INDICATEUR,D2       * Waiting for end of time delay
        cmp.b       #01,D2
        bne         ATT5
        move.l      #1000,COMPTEUR      * 1000*1mS = 1S
        move.b      #$00,INDICATEUR     * end of counting

        move.w      #$5955,HSRR1        * Only D3 lights
ATT6    move.b      INDICATEUR,D2       * Waiting for
        cmp.b       #01,D2
        bne         ATT6
        move.l      #1000,COMPTEUR
        move.b      #$00,INDICATEUR

        move.w      #$6555,HSRR1
ATT7    move.b      INDICATEUR,D2
        cmp.b       #01,D2
        bne         ATT7
        move.l      #1000,COMPTEUR
        move.b      #$00,INDICATEUR

        move.w      #$9555,HSRR1
ATT8    move.b      INDICATEUR,D2
        cmp.b       #01,D2
        bne         ATT8
        move.l      #1000,COMPTE
        move.b      #$00,INDIC

        bra         Deb_B               * END of main loop
*       END of main program
*************************

*************************
*       INTERRUPT FUNCTION            *
*       linked to time base           *
***********************************************
it_bt   sub.l       #$00000001,COMPTEUR
        cmp.l       #$00000000,COMPTEUR
        bne         it_ret              * Return if not equal to 0
        move.b      #$01,INDICATEUR     * End of  time delay
        move.l      #1000,COMPTEUR      * Reset of time delay
it_ret  rte                             * Interrupt return
End of interrupt program
****************************

        end
*  End of source file
```

Sample

EXPERIMENT N° 2 – DUPLICATION OF ONE 8 BITS INPUT PORT ONTO 8 BITS OUTPUT PORT.

2.1 Topic

Purpose :	**Additional abilities:** Being capable of configuring one inputs / outputs port onto input port. Being capable of acquiring the state of these inputs (acquisition technique specific to the 68332 micro-co[illegible]TPU).
Specifications :	The states of the 8 "SW[illegible]CHE[illegible]connected to port B of the EID210 Board is duplicated to the 8 LE[illegible]ect[illegible]to port A.

N[illegible]ss[illegible]y Equipment :

PC Micro-Computer us[illegible]g W[illegible]d[illegible]9[illegible]r latter,
68332 Micro-Control[illegible]its Mo[illegible]r Board, Ref. : EID 100 000
USB link cable, or if nav[illegible]2 cable, Ref. : EGD 000 003
AC/AC 8V Power Sup[illegible]y 1 A, [illegible]f. : EGD000001,
Input / Output Simulator, R[illegible]: E[illegible]D001000

Duration : 4 hours

1.3 Elements of solution

1.3.1 Acquisition of "Switches" states

Configure the bits of port B into input:

As shown on the Simulator Board layout given in ANNEX, "switches" of the inputs / outputs simulator are connected onto port B of the EID210 Board.
As shown on the EID210 Board layout, bits of port B belong to the 68332 Micro-controller TPU outputs, with the following correspondence:

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

The operating mode of a micro-controller TPU line is determined in loading a 4 bits code into a register dedicated to this purpose (Registers CFSRi with i=0,1,2,3 on 16 bits, as specified in file to be included "**EID210.def**"), following the next correspondence:

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

In this case, the requested operating mode is a simple Output mode.
In the 68332 Micro-controller Technical Data Handbook and more particularly in Chapter dedicated to the "**TPU**", this simple Inputs/Outputs mode is called "**DIO**" (**D**iscrete **I**nput **O**utput).
In this case, the 4 bits binary code to be loaded in control registers must be: 1000 = $8

Let instructions enabling configuration of port B in mode "DIO":

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

Acquisition technique of the Inputs

When the input state acquisition is required, logical binomial "11" must be written to corresponding locations of HSSR0 service register.

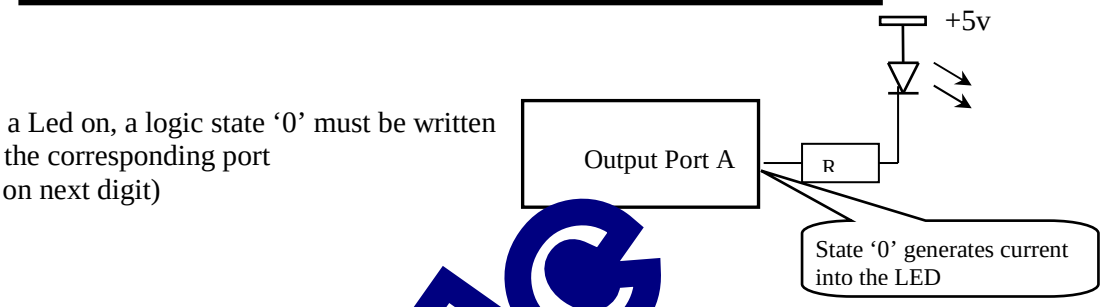
"SWITCH" N°	1	2	3	4	5	6	7	8
Bit N° into port B	7	6	5	4	3	2	1	0
TPU "CHANNEL" N°	15	14	13	12	11	10	9	8
Bits of register HSSR0	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0

When all 8 "SWITCHES" states acquisition is required, then all register HSRR0 must be loaded with binomials "11", let value be $FFFF.
When writing binary binomial "11", there is input state memorisation into16 bits memory (called state memory) carrying out a FIFO type cell (First In First Out). The last stored state is the bit of rank 15.
For knowing the input state at the moment of the last writing of binomial "11" into register HSRR0, it is sufficient to check this bit of rank 15.
The state memory address of rank i "CHANNEL" is inferred from the definition given in file EID210.def (file to be included), due to operation CH_state +CTRL_TPi

1.3.2 Activation of outputs

LEDs are connected to port A of EID210 processor Board.
Port A is connected to outputs CH0 (TPU0) <-> PA0 to CH7 (TPU7) <-> PA7 of the micro-controller.
For setting one bit of port A to logic state '1', a binary binomial "0 1" must be set into the corresponding location of register HSRR1 (which address is specified into the definition file EID210.def). For setting to logic state '0' value "1 0 " must be set to the same binomial.

Led N°	D1	D2	D3	D4	D5	D6	D7	D8
Bit N° into port A	7	6	5	4	3	2	1	0
Bits of register HSSR1	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0



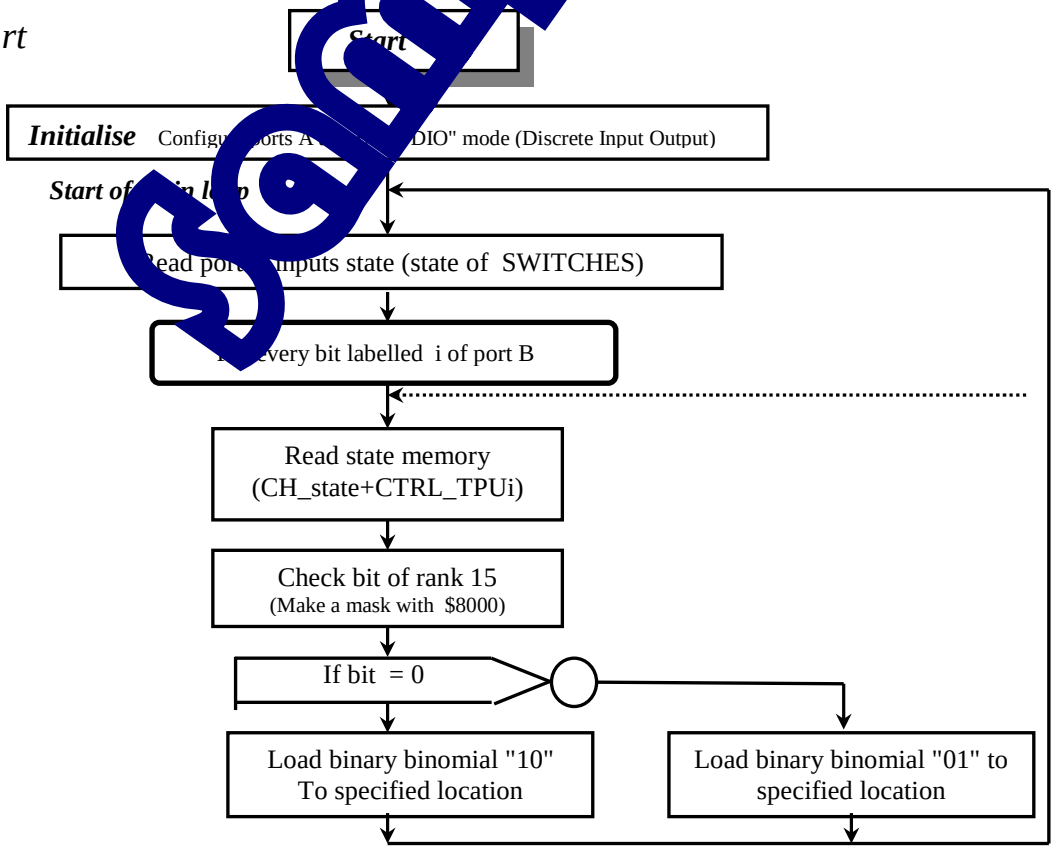
For switching a Led on, a logic state '0' must be written into the bit of the corresponding port
(As indicated on next digit)

Example : If we want Led D8 switching on, and [illegible], register HSSR1 must be loaded by the value :

N° of the concerned diode :	D0	D1	D2	D3	D4	D5	D6	D7	D8
Value Output port :	1	[illegible]				1	1	1	0
Value into register HSRR1 :	01	[illegible]				01	01	01	10

= $5556

Flowchart



1.3.3 Program in A68xxx Assembler

```
        ********************************************************************************
        *               EXPERIMENT ON EID210 WITH INPUTS/OUTPUTS SIMULATOR             *
        *               Duplicate the "Switches" state ( port B) onto Leds ( port A)   *
        ********************************************************************************
        *  Specifications                                                              *
        *********************                                                       *
        *  State of  "SWITCHES" is duplicated onto LEDs                               *
        *  Remark: These LEDs are connected to port  A                                *
        *          -> TPU lines of 68332: TPU0 to TPU7  (CHA0  to CHA7)               *
        *              "SWITCHES" are connected to  port B                            *
        *   -> TPU lines of 68332: TPU0 to TPU7  (CHA8 to CHA15)                      *
        *                                                                             *
        *                                                                             *
        *  FILE NAME:  RECOP_BsA.SRC                                                  *
        *********************                                                       *
        ********************************************************************************
        * Inclusion of the file specifying the different labels
        include EID210.def
*   Definition of constants          *
*******************************
CH_etat equ             2
****************************************
*  Start of the execute program                *
****************************************
        section                 code
*       INITIALISE
**************************
* Configure port A  in "Discrete Input Output"  (DIO) mode -> code $8
START   move.w          #$8888,CFSR3            * CHA0 t[illegible]ode
        move.w          #$8888,CFSR2            * CHA[illegible] CHA7 [illegible]mode
        move.w          #$8888,CFSR1            * CHA[illegible]to [illegible]1 i[illegible]DIO" mode
        move.w          #$8888,CFSR0            * CHA[illegible]8 to [illegible]15[illegible]"DIO" mode
* Specify  priorities
        move.w          #$FFFF,CPR1             [illegible]A in[illegible]priority
        move.w          #$FFFF,CPR0             * All bits[illegible]fi[illegible]priority

* The input states are stored into state registers
*When 11 will be written into service register
        move.w          #$AAAA,HSQR0                    [illegible]ll be stored

* All leds switched off
        move.w          #$5555,HSRR1            [illegible]1  bits PA to 1
        move.w          #$5555,[illegible]      d0 -> image of HSRR1

*       MAIN LOOP
**********************************[illegible]
Deb_BP * Acquire inputs
        move.w          #$FFFF,HS[illegible]

        * Check state of PB7 then, position PA[illegible]
        move.w          CH_etat+CTRL_TPU8,d4    * Read state register of PB7
        btst            #15,d4
        beq             b7_0
        bclr            #15,d0                  * If  1 , prepare for having PA7=1
        bset            #14,d0
        bra             b7_1
b7_0    bset            #15,d0                  * If  0 , prepare for having  PA7=0
        bclr            #14,d0
b7_1    move            d0,HSRR1                * Load onto port A

        * Check state of PB6 then, position PA6
        move.w          CH_etat+CTRL_TPU9,d4    * Read state register of PB6
        btst            #15,d4
        beq             b6_0
        bclr            #13,d0                  * If  1 , prepare for having PA6=1
        bset            #12,d0
        bra             b6_1
b6_0    bset            #13,d0                  * If  0 , prepare for having  PA6=0
        bclr            #12,d0
b6_1    move            d0,HSRR1                * Load onto port A
*   Continued next page
```

```
            * Check state of  PB5 ,then position PA5
            move.w          CH_etat+CTRL_TPU10,d4     * Read state register of PB5
            btst            #15,d4
            beq             b5_0
            bclr            #11,d0                    * If  1 , prepare for having PA5=1
            bset            #10,d0
            bra             b5_1
b5_0        bset            #11,d0                    * If  0 , prepare for having  PA5=0
            bclr            #10,d0
b5_1        move            d0,HSRR1                  * Load onto  port A

            * Check state of  PB4 ,then position PA4
            move.w          CH_etat+CTRL_TPU11,d4     * Read state register of PB4
            btst            #15,d4
            beq             b4_0
            bclr            #9,d0                     * If  1 , prepare for having PA4=1
            bset            #8,d0
            bra             b4_1
b4_0        bset            #9,d0                     * If  0 , prepare for having  PA4=0
            bclr            #8,d0
b4_1        move            d0,HSRR1                  * Load onto  port A

             * Check state of  PB3, then position PA3
            move.w          CH_etat+CTRL_TPU12,d4     * Read state register of PB3
            btst            #15,d4
            beq             b3_0
            bclr            #7,d0                     * If  1 , prepare for having PA3=1
            bset            #6,d0
            bra             b3_1
b3_0        bset            #7,d0                     * If  0 , prepare for having  PA3=0
            bclr            #6,d0
b3_1        move            d0,HSRR1                  * Load onto  port A
            Check state of  PB2, then position PA2
            move.w          CH_etat+CTRL_TPU13,d4     * Read state register of PB2
            btst            #15,d4
            beq             b2_0
            bclr            #5,d0                     * If  1 , prepare for having PA2=1
            bset            #4,d0
            bra             b2_1
b2_0        bset            #5,d0                     * If  0 , prepare for having  PA2=0
            bclr            #4,d0
b2_1        *move           d0,HSRR1                  * Load onto  port A

            * Check state of  PB1, then position PA1
            move.w          CH_etat+CTRL_TPU14,d4     * Read state register of PB1
            btst            #15,d4
            beq             b1_0
            bclr            #3,d0                     * If  1 , prepare for having PA1=1
            bset            #2,d0
            bra             b1_1
b1_0        bset            #3,d0                     * If  0 , prepare for having  PA1=0
            bclr            #2,d0
b1_1        move            d0,HSRR1                  * Load onto  port A


            * Check state of  PB0, then position PA0
            move.w          CH_etat+CTRL_TPU15,d4     * Read state register of PB0
            btst            #15,d4
            beq             b0_0
            bclr            #1,d0                     * If  1 , prepare for having PA0=1
            bset            #0,d0
            bra             b0_1
b0_0        bset            #1,d0                     * If  0 , prepare for having  PA0=0
          bclr              #0,d0
b0_1        move            d0,HSRR1                  * Load onto port A

            bra             Deb_BP                    * End of main loop  -> loop
End of main program
**************************
End of file
***************
            end
```

Sample

EXPERIMENT N° 2- CONTROL OF 7 SEGMENTS DISPLAY

1.4 Topic

Purpose :	**Additional Abilities:** Being capable of displaying a digital data in using a '7 segments' display device. Being capable of transcoding a '**B**inary **D**ecimal **C**ode'(BCD) information into a control information of a 7 segments display. Being capable of programmi[illegible] a c[illegible]puter science structure 'SELECT IN'-type.
	Variant n°1: Dec[illegible] On the 7 segme[illegible] d[illegible]y [illegible]vice, we should like to see digits from 0 to 9 displayed with a [illegible]cur[illegible] of about one second per digit. ***Variant n° 2: Hexa[illegible] display*** On th[illegible] s[illegible]ment[illegible]splay device, we should like to see the 15 hexadecimal charac[illegible]s (f[illegible] to 9 then A, B, C, D, E and F) displayed with a recurrence o[illegible] s[illegible]ond per digit.

Necessary Equipment :

PC Micro-Computer using Windows ® 95 or latter,
68332 Micro-Controller 16/32 bits Mother Board, Ref. : EID 100 000
USB link cable, or if unavailable RS232 cable, Ref. : EGD 000 003
AC/AC 8V Power Supply, 1 A, Ref. : EGD000001,
Input / Output Simulator, Ref. : EID001000

Duration : 4 hours

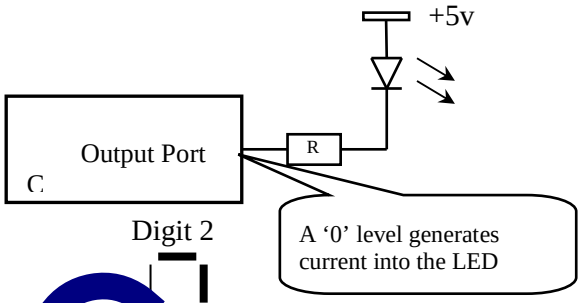
1.5 Elements of solution

1.5.1 Outputs activation

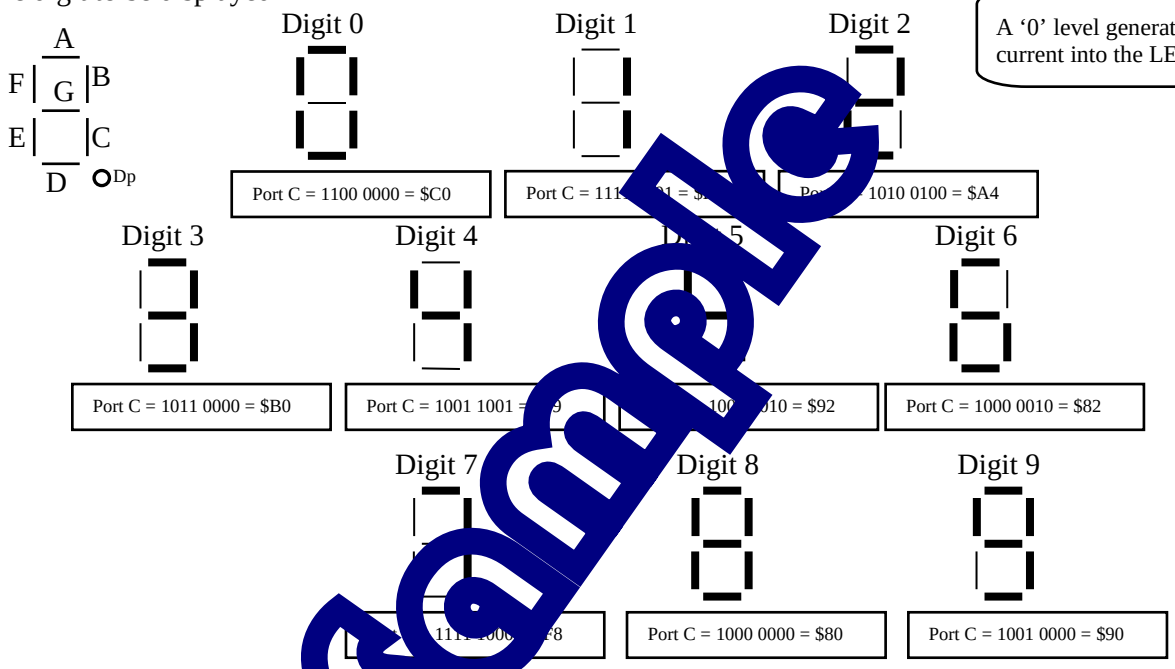
As shown on layout given in ANNEX, we see that the 7 segments display is connected to port C of the processor board, with the following correspondence :

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

On the other hand, the display device is « COMMON ANODE » - type, which means that a logic level 0 must be written onto an output for switching on the corresponding « LED ».



Determination of the logic states of port C, function of The digit to be displayed

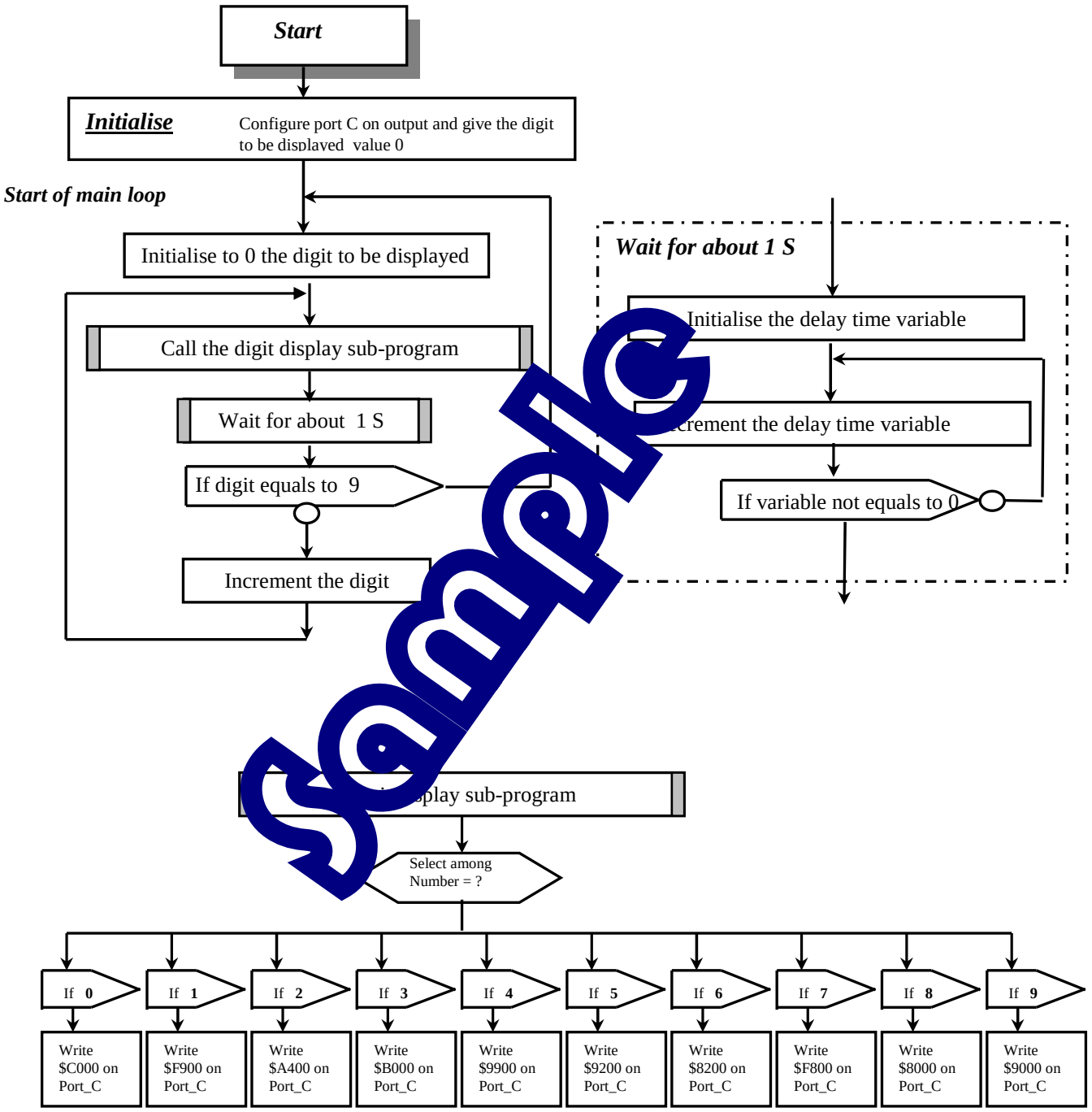


Initialisation and writing:
Before using a bit of port C in output, the associated direction register must be initialised (label DIR_Port_C, which address is specified in definition file EID210.def that must be included).
A logic level ‘1’ must be written to the corresponding place into the direction register for enabling the use of a bit of port C as an output.
In this case, we must write $FF00 (location onto MSB).
For activating the output bits, the value must be written into the data register associated to port C (label Port_C, which address is specified in definition file EID210.def).

Example :
If we require the display of digit ‘8’ then, instruction must be written Because the value must be placed onto the word MSBs.

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

1.5.2 Decimal display Flowcharts (Variant n°1)



1.5.3 Decimal display program in A68xxx Assembler (Variant n°1)

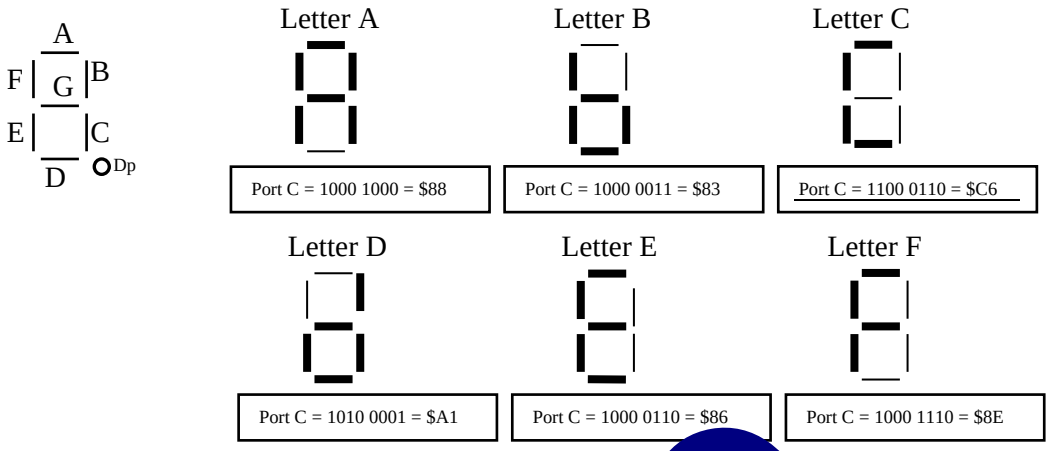
```
        ********************************************************************************
        *  EXPERIMENTS ON EID210 BOARD WITH INPUTS/OUTPUTS SIMULATOR      *
        ********************************************************************************
        *                                                                              *
        *              Check of port C controlling the  7 segments display device      *
        *              for the display of a decimal digit  ( 0 to 9)                   *
        *                                                                              *
        *  Description:                                                                *
        ***************                                                               *
        *  The display device displays a decimal digit (between  0 and 9)             *
        *  incrementing itself every second                                            *
        *  This time duration is generated by a "program" loop                         *
        *                                                                              *
        *  FILE NAME:  T_Port_C_1.SRC                                                  *
        ***********************                                                       *
        ********************************************************************************

**************************************
*          Declaration of variables              *
**************************************
          section       var
NOMBRE ds.b         1           * 8 bits size
*  Inclusion
***************
* Inclusion of the file specifying the different labels
          include                 EID210.def


**************************************
*  Start of execute program                    *
**************************************
          section                 code

*          INITIALISE
***************************
Start     move.w    #$FF00,DIR_Port_C  * Port C is con[illegible]red in ou[illegible]vel [illegible]ts output operation


*          MAIN LOOP
**************************************
*Increment digit
Deb_BP    move.b              NOMBRE,d0           Register "do" is an image of  NOMBRE
          cmp.b               #9,d0
          bne                 Etiq1
          move.b              #0,d0
          bra                 Etiq2
Etiq1     add.b               #1,d0
Etiq2     move.b              d0,NOMBRE
* Go to display the result
          jsr                 AFFICHER
*  Waiting loop of about 1 second
          move.l              #$001FFFFF,d2
ATT       sub.l               #1,d2
          bne                 ATT
*  Pass to the next digit
          bra                 Deb_BP
*End of main loop
**********************************
*****************************
* End of main program              *
*****************************
```

* Continued next page…

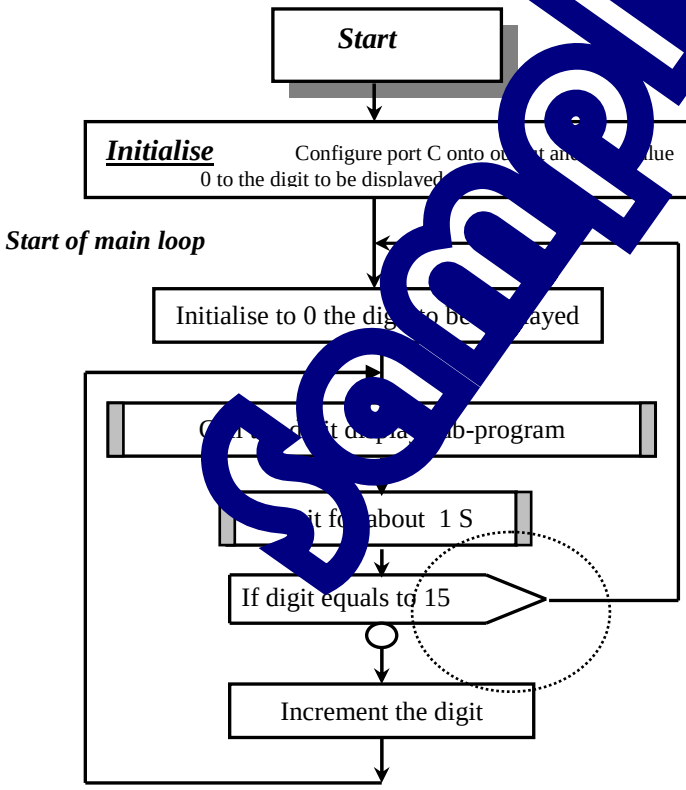
```
***********************************************
*  Transcoding sub-program                    *
*  "Decimal Coded Binary" -> 7 Segments       *
*  and display                                *
***********************************************
DISPLAY
            cmp.b       #$00,d0
            bne         display 1       * Go out if it is not 0
            move.w      #$C000,Port_c
            rts
display1    cmp.b       #$01,d0
            bne         display2        * Go out if it is not1
            move.w      #$F900,Port_c
            rts
display2    cmp.b       #$02,d0
            bne         display 3       * Go out if it is not2
            move.w      #$A400,Port_c
            rts
display3    cmp.b       #$03,d0
            bne         display 4       * Go out if it is not3
            move.w      #$B000,Port_c
            rts
display4    cmp.b       #$04,d0
            bne         display 5       * Go out if it is not 4
            move.w      #$9900,port_c
            rts
display5 cmp.b          #$05,d0
            bne         display 6       * Go out if it is not 5
            move.w      #$9200,port_c
            rts
display6    cmp.b       #$06,d0
            bne         display 7       * Go out if it is not 6
            move.w      #$8200,port_c
            rts
display7    cmp.b       #$07,d0
            bne         display 8       * Go out if it is not 7
            move.w      #$F800,port_c
            rts
display8    cmp.b       #$08,d0
            bne         display 9       * Go out if it is not 8
            move.w      #$8000,port_c
            rts
display9    cmp.b       #$09,d0
            bne         display_nothing * Go out if it is not 9
            move.w      #$9000,port_c
            rts
*If the digit is not between 0 and 9, display nothing
display _nothing
            move.w      #$FF00,port_c
            rts
* End of transcoding and displaying subprogram
************************************************************

* End of listing
*****************
            end
```

1.5.4 Modifications to be made for complying with Variant n°2

Complement for representing letters A, B C, D, E and F



Modification on the Flowchart:



1.5.5 Program in A68xxx Assembler of hexadecimal display (Variant n°2)

```
*************************************************************************
*  EXPERIMENTS ON EID210 BOARD WITH INPUTS/OUTPUTS SIMULATOR      *
*************************************************************************
*              Test of port C, controlling the 7 segments display device  *
*               for displaying an hexadecimal value (from 0 to 15)        *
*                                                                         *
*  Description:                                                           *
***************                                                          *
* The display device displays an hexadecimal digit (between  0 and 15)   *
*  let be: 0,1,2,3,4,5,6,7,8,9,A,b,C,d E,F                                *
* incrementing itself every second                                       *
* This time duration is generated by a "program" loop                    *
*                                                                         *
*  FILE NAME:  T_Port_C_2.SRC                                             *
*******************                                                      *
*************************************************************************

************************************
*          Declaration of variables              *
************************************
          section       var
NOMBRE ds.b       1           *  8 bits size

*  Inclusion
**************
* Inclusion of the file specifying the different labels
          include                 EID210.def

************************************
* Start of execute program                *
************************************
          section                 code

*          INITIALISE
************************
*   Port C
start     move.w    #$FF00,DIR_Port_C  * Port C configured as output
                                       * Level  1 sets output operation

* Digit to display
          clr                     NOMBRE
          clr                     d0                    * Register "do" is an image of  NOMBRE
*          MAIN LOOP
**********************************
Deb_BP * Start of main loop
* Go to display the current digit
          jsr          AFFICHER
*  Waiting loop of about 1 second
          move.l                  #$00FF0FF2
ATT       sub.l                   #1,d1
          bne                     ATT

* Increment digit
          cmp.b                   #15,d0
          bne                     Etiq1               * If it is not equal to 15, increment
          clr.b                   d0                  * If it is equal to 15, start again from 0
          bra                     Etiq2
Etiq1     add.b                   #1,d0
Etiq2     move.b    d0,NOMBRE

*  Pass to the next digit
          bra                     Deb_BP
* End of main loop
**********************************
*****************************
* End of main program               *
*****************************
```

```
*****************************************************************************
*          Transcoding sub-program          "  Hexadecimal"  ->  7 Segments  & display *
*****************************************************************************
AFFICHER        cmp.b           #$00,d0
                bne             display1        * Go out if it is not 0
                move.w          #$C000,Port_c
                rts
display1        cmp.b           #$01,d0
                bne             display2        * Go out if it is not 1
                move.w          #$F900,Port_c
                rts
display2        cmp.b           #$02,d0
                bne             display3        * Go out if it is not 2
                move.w          #$A400,Port_c
                rts
display3        cmp.b           #$03,d0
                bne             display4        * Go out if it is not 3
                move.w          #$B000,Port_c
                rts
display4        cmp.b           #$04,d0
                bne             display5        * Go out if it is not 4
                move.w          #$9900,port_c
                rts
display5        cmp.b           #$05,d0
                bne             display6        * Go out if it is not 5
                move.w          #$9200,port_c
                rts
display6        cmp.b           #$06,d0
                bne             display7        * Go out i[illegible]s not
                move.w          #$8200,port_c
                rts
display7        cmp.b           #$07,d0
                bne             display8        * Go o[illegible]ot 7
                move.w          #$F800,port_c
                rts
display8        cmp.b           #$08,d0
                bne             display9        Go o[illegible]ot 8
                move.w          #$8000,port_c
                rts
display9        cmp.b           #$09,d0
                bne             display10       [illegible]t is not 9
                move.w          #$9000,port_c
                rts
display10       cmp.b           #$0A
                bne             displ[illegible]1        o out if it is not 10 ($A)
                move.w          #$8[illegible]_po
                rts
display11       cmp.b           #
                bne             display[illegible]       * Go out if it is not 11 ($B)
                move.w          #00,po[illegible]_c
                rts
display12       cmp.b           #$[illegible]o
                bne             display[illegible]       * Go out if it is not 12 ($C)
                move.w          #$C600,port_c
                rts
display13       cmp.b           #$0D,d0
                bne             display14       * Go out if it is not 13 ($D)
                move.w          #$A100,port_c
                rts
display14       cmp.b           #$0E,d0
                bne             display15       * Go out if it is not 14 ($E)
                move.w          #$8600,port_c
                rts
display15       cmp.b           #$0F,d0
                bne             display_nothing * Go out if it is not 15 ($F)
                move.w          #$8E00,port_c
                rts
display1_nothing move.w         #$FF00,port_c   * If it is out from $00 to $0F, switch display device off.
                rts
* End of transcoding and display sub-program
***********************************************
* End of listing
***************
                end
```

TP 2 DISPLAY OF POTENTIOMETER POSITION

2.1 Topic

Purpose :	**Complementary abilities:** Being capable of configuring one Analogue -> Digital Converter. Being capable of detecting the end of one Analogue -> Digital Converter.
Cahier des charges :	Display on the 7 s[illegible] device the 4 most significant bits resulting from the conver[illegible] the [illegible] 0 Analogue input, on which is connected the potentiometer cu[illegible]

Necessary Equipment :

PC Micro-Computer using Windows ® 95 or latter,
68332 Micro-Controller 16/32 bits Mother Board, Ref. : EID 100 000
USB link cable, or if unavailable RS232 cable, Ref. : EGD 000 003
AC/AC 8V Power Supply, 1 A, Ref. : EGD000001,
Input / Output Simulator, Ref.. : EID001000

Duration : 2 hours

2.2 Elements of solution

Conversion of Analogue input:
The used Analogue -> Digital Converter is a MAX196. It can convert up to 6 Analogue inputs. The input to be converted is the rank '0' channel.
A conversion demand is carried out by writing a control word, which format is the following:

7	6	5	4	3	2	1	0
PD1	PD0	ACQ	RNG	BIP	A2	A1	A0

Bits A2, A1 and A0 specify the Analogue input rank to be converted.

RNG	BIP	Range
0	0	0 to 5V
0	1	0 10V
1	0	+-5V
1	1	+-10V

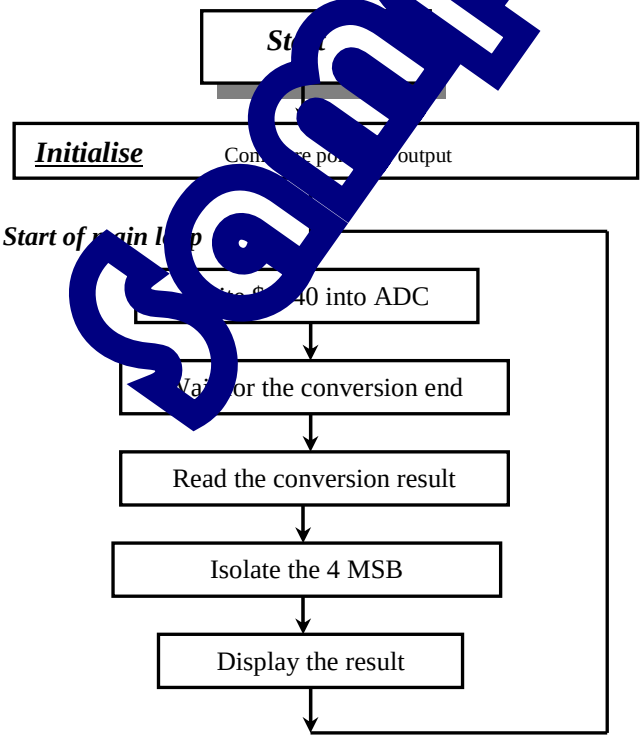
A2	A1	A0	Ana. input rank
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5

Both bits RNG and BIP specify the selected Analogue input variation range.

Bits PD1,PD0, ACQ must be initialised to 010.

Word 0100 0000 = $0040 must be written to the ADC address (n file EID210.def, file to be included).
Conversion is not immediate. The end of conversion ca ng the rank 14 bit of the state system.

2.2.1 Flowchart



2.2.2 Program in Assembler A68xxx

```
*****************************************************************************************
*               EXPERIMENTS ON EID210 BOARD WITH INPUTS/OUTPUTS  SIMULATOR              *
*****************************************************************************************
*               Display on  7 segments device of  n°0 Analogue input conversion result  *
*                (the potentiometer's one)                                              *
*                                                                                       *
*  Specifications:                                                                      *
*******************                                                                     *
*  One hexadecimal digit (between 0 and 15) is displayed, corresponding to the MSB of the E0 Analogue input  *
* conversion result (set by the potentiometer position)                                 *
*  FILE NAME:  T_CAN_Port_C.SRC                                                         *
*******************                                                                     *
*****************************************************************************************

*  Inclusion
***************
* Inclusion of the file specifying the different labels
        include                 EID210.def

***************************************
*  Start of the execute program       *
***************************************
        section                 code

*       INITIALISE
**********************
*   Port C
Start           move.w          #$FF00,DIR_Port_                * Port C is configured in output
                                                                *  Level 1 sets output operation

*        MAIN LOOP
********************************
Deb_BP * Start of main loop
* Start the Analogue -> Digital conversion of channe
                move.w          #$004 LAN
* Wait for the end of conversion (test of the " A->  end  nver n" of the state register)
Att_F_C move.w   REG_ETAT,D0
                and.w           #
                bne             tt_F_C
* Read the conversion result
                move.w          C
                and.w           #$0FF,
* Keep only the 4 MSB
                lsr.w           D0
* Go to display the result
                jsr             A FICHER
•       Loop  -> Start again
•               bra             Deb_BP
* End of main loop
**************************
*****************************
* End of main program           *
*****************************

***********************************************
*  Transcoding sub-program                    *
*   "  Hexadecimal"  ->  7 Segments            *
*   & display                                 *
***********************************************
DISPLAY         cmp.b           #$00,d0
                bne             display1        * Go out if it is not  0
                move.w          #$C000,Port_c
                rts
```

```
display1        cmp.b       #$01,d0
                bne         display2        * Go out if it is not 1
                move.w      #$F900,Port_c
                rts
display2        cmp.b       #$02,d0
                bne         display3        * Go out if it is not 2
                move.w      #$A400,Port_c
                rts
display3 cmp.b  #$03,d0
                bne         display4        * Go out if it is not 3
                move.w      #$B000,Port_c
                rts
display4 cmp.b  #$04,d0
                bne         display5        * Go out if it is not 4
                move.w      #$9900,port_c
                rts
display5        cmp.b       #$05,d0
                bne         display6        * Go out if it is not 5
                move.w      #$9200,port_c
                rts
display6        cmp.b       #$06,d0
                bne         display7        * Go out if it is not 6
                move.w      #$8200,port_c
                rts
display7        cmp.b       #$07,d0
                bne         display8        * Go out if it is not 7
                move.w      #$F800,port_c
                rts
display8        cmp.b       #$08,d0
                bne         display9        * Go out if it is
                move.w      $8000,port_c
                rts
display9        cmp.b       #$09,d0
                bne         display10       * Go out if it is
                move.w      #$9000,port_c
                rts
display10       cmp.b       #$0A,d0
                bne         display11       * Go out if it is not 10 ($A)
                move.w      #$8800,port_c
                rts
display11       cmp.b       #$0B,d0
                bne         display12       * Go out if it is not 11 ($B)
                move.w      #$830 ort_c
                rts
display12       cmp.b       #$0 0
                bne         display13       * Go out if it is not 12 ($C)
                move.w      #
                rts
display13       cmp.b       #$ ,d0
                bne         display14       * Go out if it is not 13 ($D)
                move.w      #$ t_c
                rts
display14       cmp.b       #$0E,d0
                bne         display15       * Go out if it is not 14 ($E)
                move.w      #$8600,port_c
                rts
display15       cmp.b       #$0F,d0
                bne         display_nothing * Go out if it is not 15 ($F)
                move.w      #$8E00,port_c
                rts
* If it is out from  $00 to $0F, switch display device off.
display_nothing move.w      #$FF00,port_c
                rts
* End of transcoding and display sub-program
*****************************************************************

* End of listing
*****************
                end
```

ANNEX : INPUTS/OUTPUTS SIMULATOR LAYOUT