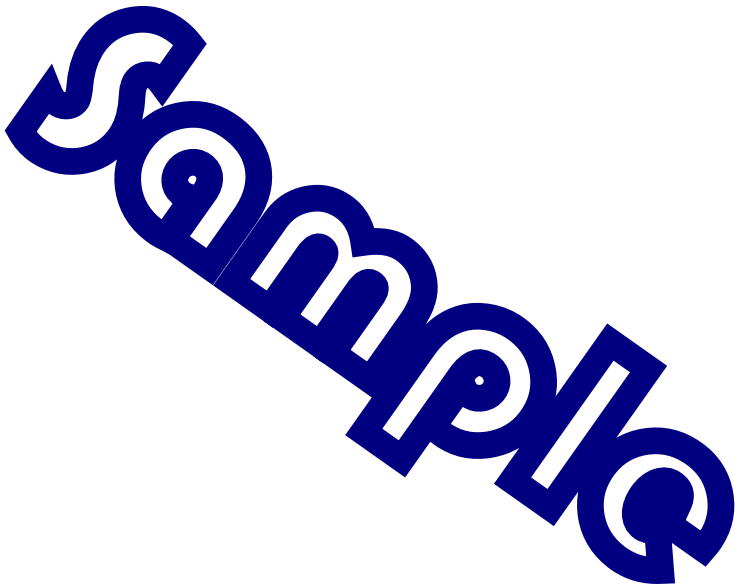
EXPERIMENTS MANUAL
on
EID210 Board
Processor Board based on 682332 Microprocessor (68000 Range)

DMS
Didalab Matelco Stci

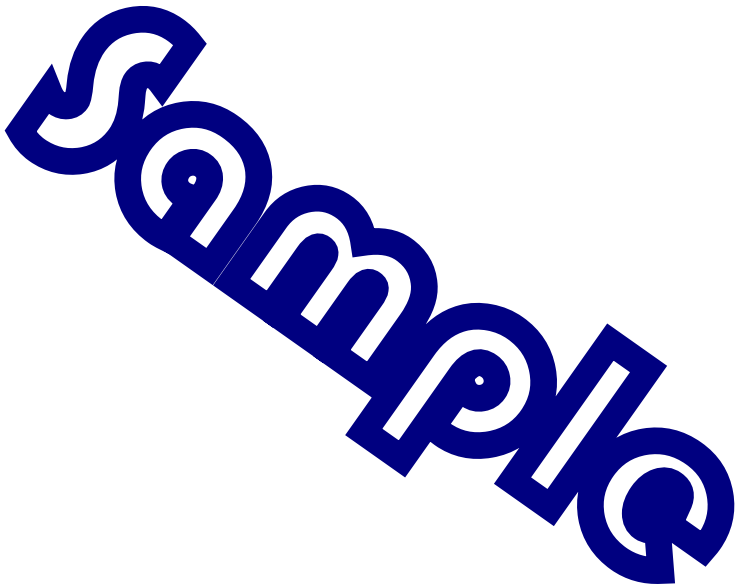
Z.A.C de La Clé St Pierre
5, rue du Groupe Manoukian
78990 ELANCOURT
Tel. : (33) 01 30 66 08 88
Fax : (33) 01 30 66 72 20

Edition of : 02/14/11

Ref. : EID210041


Sample

SUMMARY


Sample

TP 0 : DISCOVERY AND IMPLEMENTATION OF THE PROGRAM PACK

0.1 Warning

NB : The following Experiment sheet has no pedagogical purpose, its aim is to help the User to get familiar with the 68332 micro-controller EID210 study pack Unit. It is constituted of detailed successive steps on the hardware and software implementation at the first utilisation.

0.2 Exposition of the Topic

Purpose :	Start of the 68332 16/32 bits micro-controller EID 210 000 mother Board: Starting, file loading , assembli[illegible]ecking of the step by step operating mode of a loop program.
Specification :	Using a few instructi[illegible]gra[illegible]for the carrying out of the following operations : ➢ Initialisation [illegible] d1 & d2 to 0, ➢ Load the [illegible]ue 5[illegible]0 a[illegible]6 into d1, ➢ Add d0 [illegible]ith [illegible]lt (B in hexadecimal) into d0, ➢ Load [illegible] w[illegible]222 into register d2, ➢ Loop [illegible] [illegible]again.

1[illegible]essary Test Equipment :

PC Micro Computer using Wi[illegible]w® 95 or later,

68332 16/32 bits micro-controller mother Board, Ref. : EID 100 000

USB connection cable, or if not available an RS232 cable, Ref. : EGD 000 003

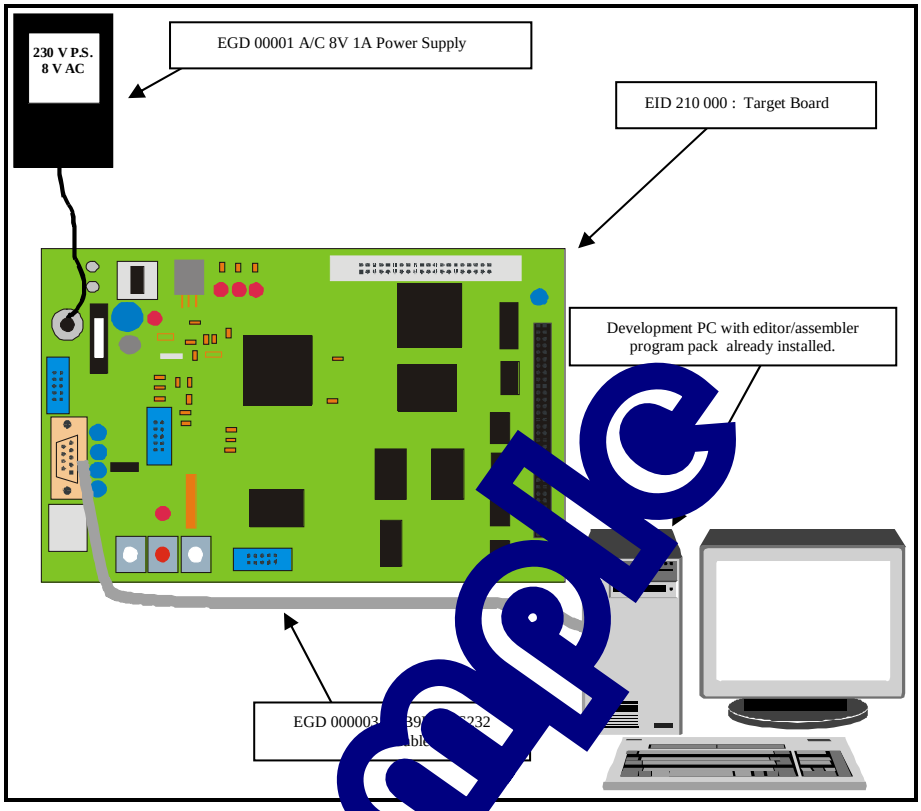
AC/AC 8V, 1 A Power Supply, Ref. : EGD000001,

Assembler source file provided : «**tst_cpu.scr**»,

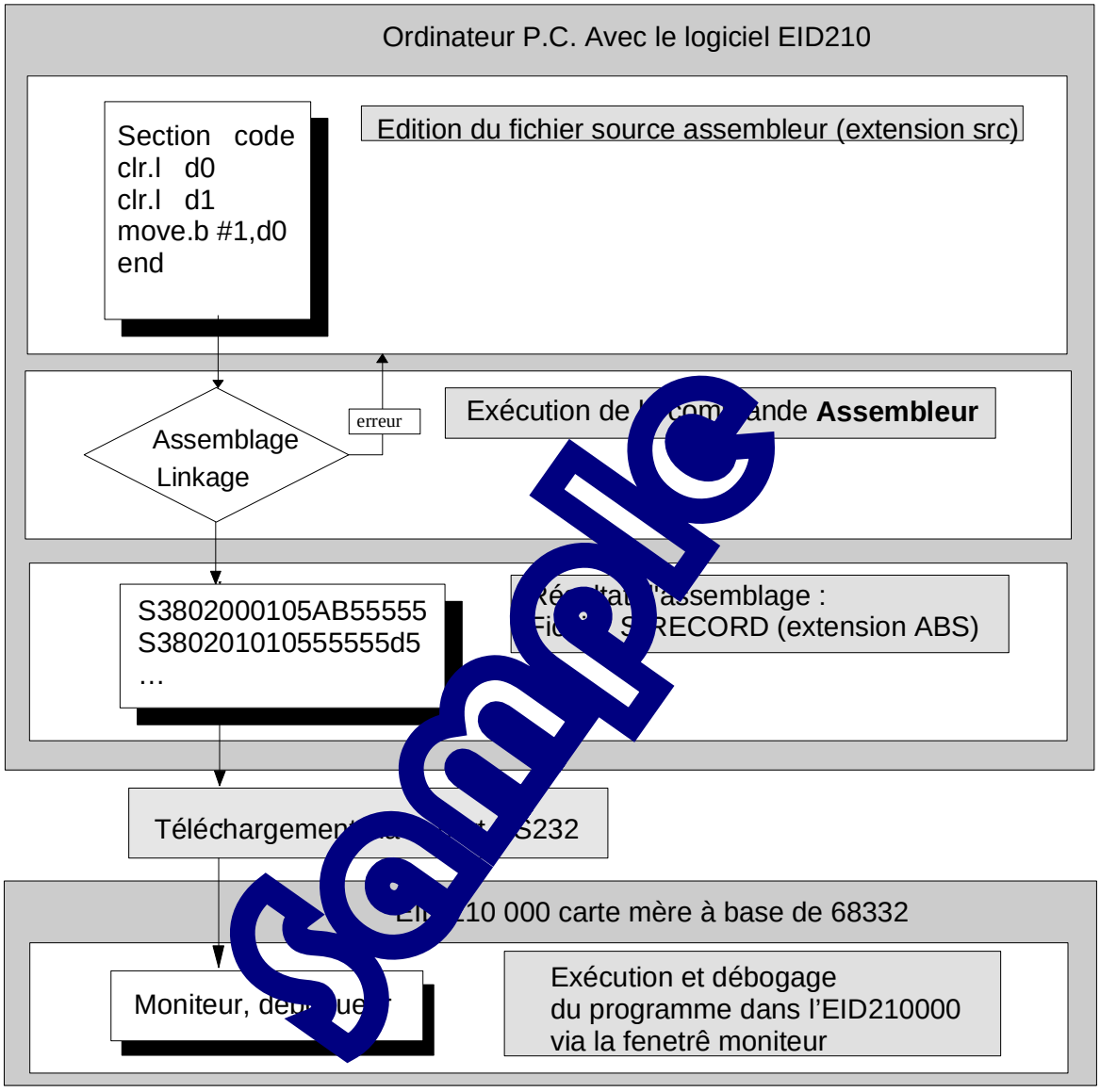
Duration : 2 hours

0.3 Installation of the equipment

- ➔ Connect the EID 210 000 Board to the development PC Computer with the Assembler program (provided together with the equipment and already installed after having followed the technical instructions) in using the USB cable or, if not available, the RS232 serial cable
- ➔ Connect the Power Supply to the EID 210 000 Board, (7 to 12 V AC or DC),
- ➔ Press the ON/OFF button on the EID 210 000 Board, the red light bulb must go on.

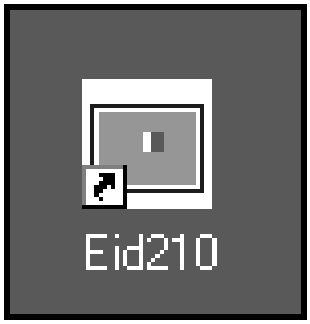


0.4 Presentation of the progress of a complete development phase in Assembler language.



0.5 Starting the Program.

➔ Click twice on icon « **Eid210 »**

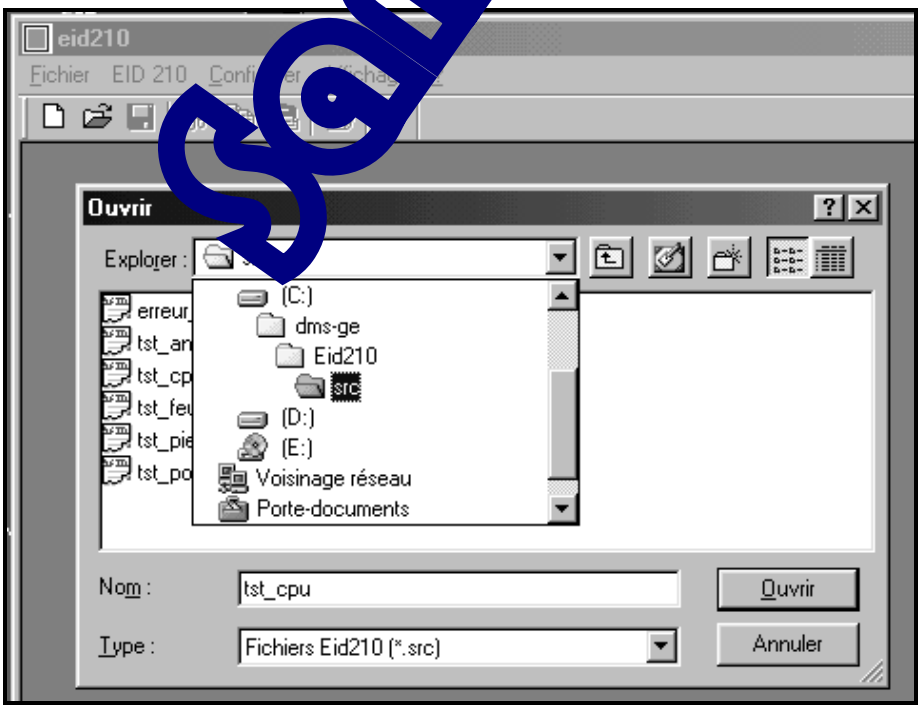


0.6 Opening of the Assembler Fil« .scr »,

➔ Click on « **File** », then « **Open** »

Using the Explorer window, go to the address file : « **Eid210\scr** »

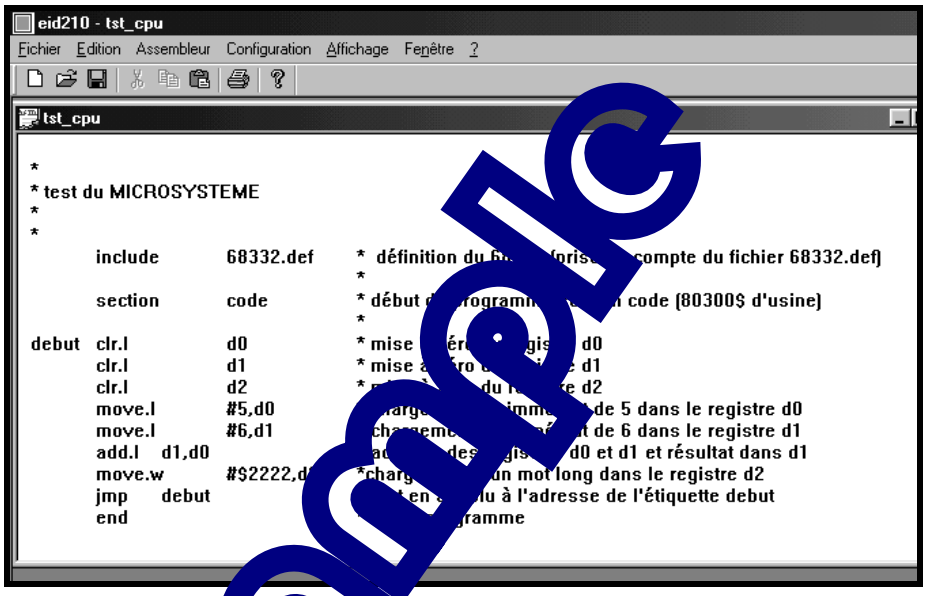
➔ Click on « **tst_cpu .scr** », then on « **Open** »



0.7 Display of file « tst_cpu.scr »

After having clicked on open (previous chapter), the file is the following.

It includes : - A first text zone, identified by «***** »=> comments,
- Function « **include** » which determines the 68332 micro-controller registers,
- The program start address, « **section code** », is automatically determined at the hexadecimal address $803 000, (see ANNEX 2)
- The Assembler program , with one « **label** » zone located on the left side of the window, one « **Instruction** » zone, one « **operand** » zone, then one « **comment** » zone identified once again by « * ».
- Detail of the instructions in the « **RESSOURCE** » file at the end of the document.



0.8 Assembli... d... file « tst_cpu.scr »

➔ Click on « **Assembler** »

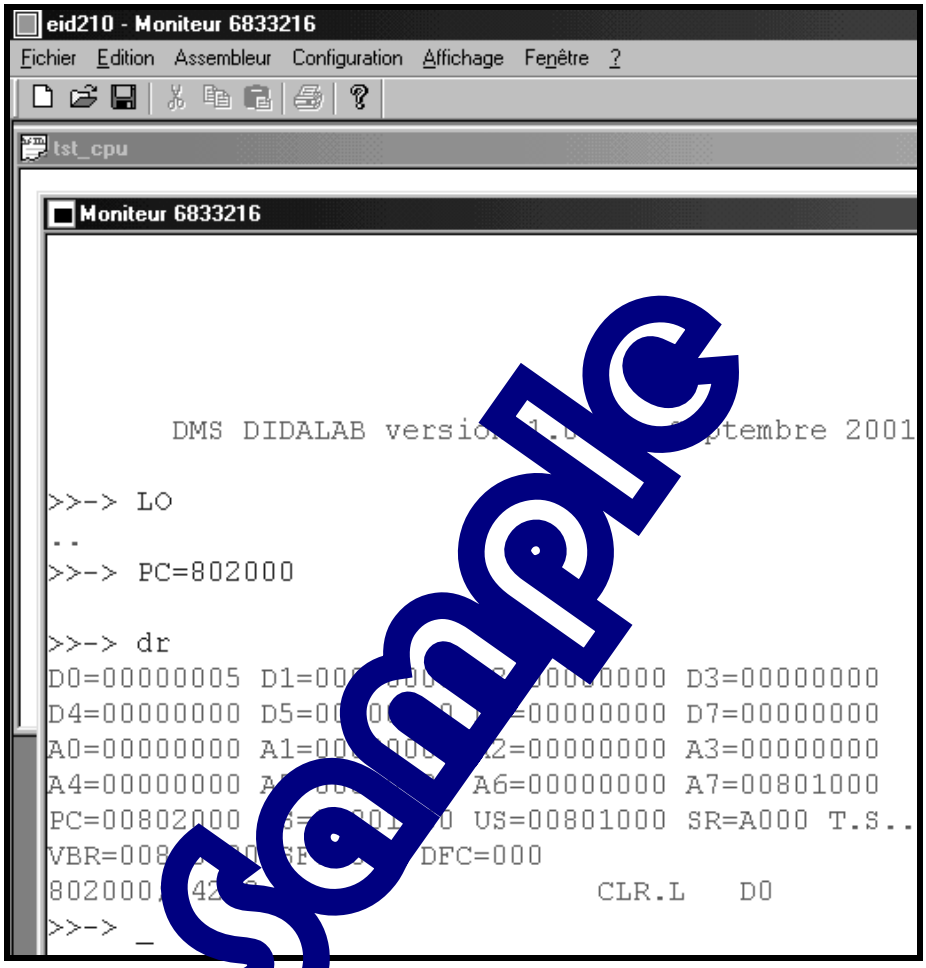
The Computer assemblies the program, then displays the assembling result, in the present case: File « tst_cpu »

Number of error(s) = « 0 »
Number of warning(s) = « 0 »,
If the Computer says: « no reply of the EID210 », refer to annex 1.

➔ Click on « **OK** »

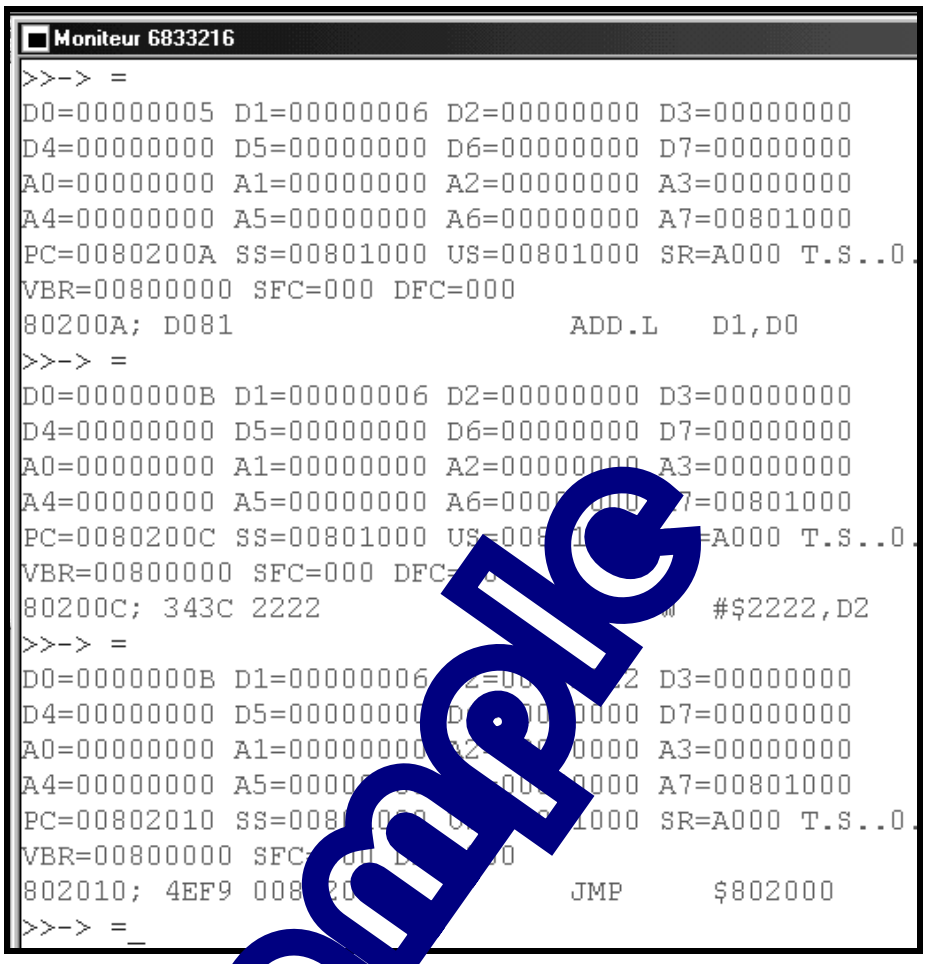
The Computer downloads the program into the target Board EID 210 000, then goes to the monitor mode.

➔ Type « **DR** », then enter for displaying the registers state, accumulators of the CPU



We can observe the CPU registers and accumulators, and mainly the ordinal Counter directed to the Address $802 000, the first instruction, operating code in hexadecimal « 4280 ».
With the de-assembling function we can read « CLR.L D0 », initialisation of D0 to zero.

➔ For executing the program in step by step mode, type « **SS** », (Single Step), then Enter.
➔ For executing an extra step, type Enter again.

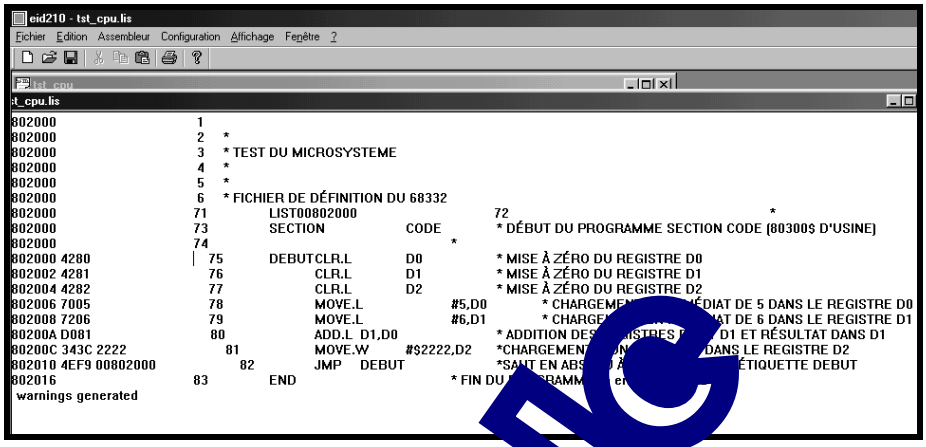


We can notice and check the progress of the program, in accordance with the Assembler source,

- First, initialisation of d7,
- Loading of 5 into d0, and 6 into d1,
- Addition of the 2 registers with result into d0,
- Loading of a long word into d2
- etc..

➔ For displaying the listing file:

- Click on « **File** »
- Click on « **Open** »
- In the window, select « **All files (*.*)** »
- Click on file « **tst_cpu.lis** »
- Click on **« Open** ».



We can notice the Assembling result listing including the addresses, operation codes, operands and comments.

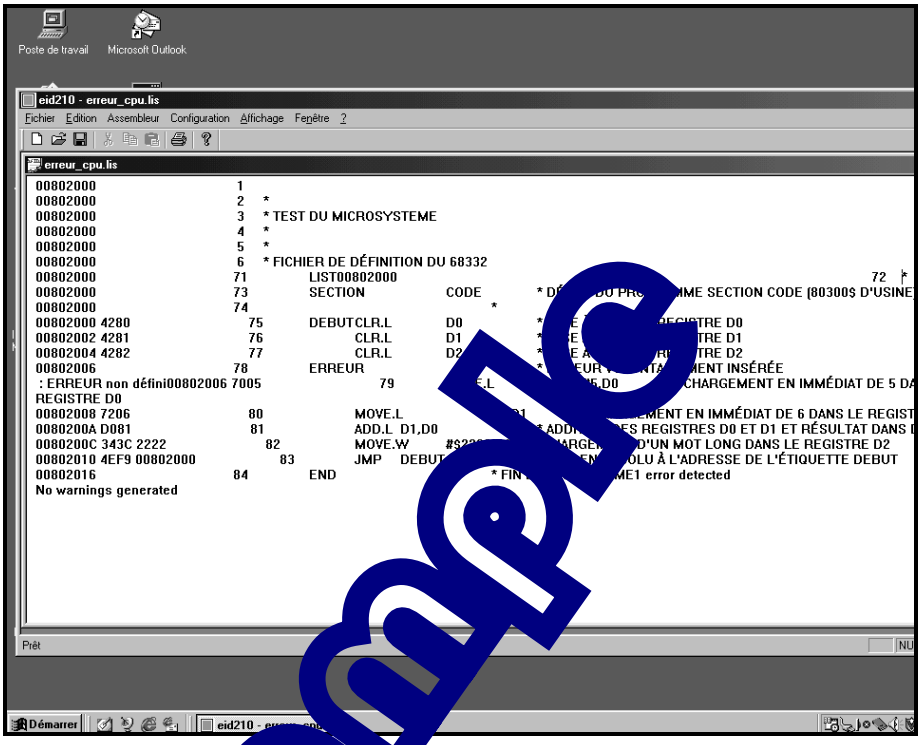
➔ Case of a file having an error :

Go back to paragraphs 0.5 to 0.7, in using the ... file « **erreur.src** » instead of the file « **tst_cpu.src** ».

During the Assembling phase, the Assembler will indicate an error and will refuse to switch to Monitor mode.

➔ For displaying the error ,

- Click on : « **File** »,
- then on « **Open** »,
- then on « **all types of files** (***.***),
- then on file « **error.lis** »,
- then on « **Open** ».



We can notice the error i[illegible]y [illegible]mbler between the code lines :

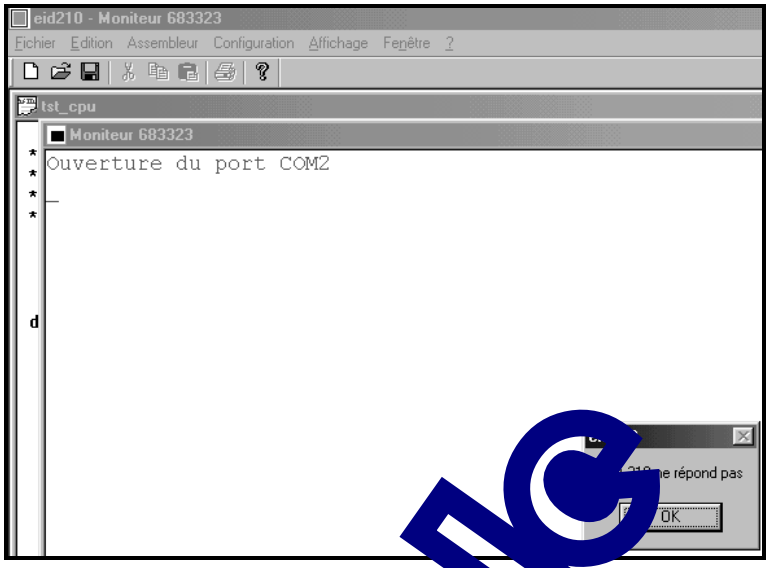
$08[illegible]

and

- $08[illegible]08, [illegible]e word « **error** » not being in any case an operating code.

ANNEX n°1 :

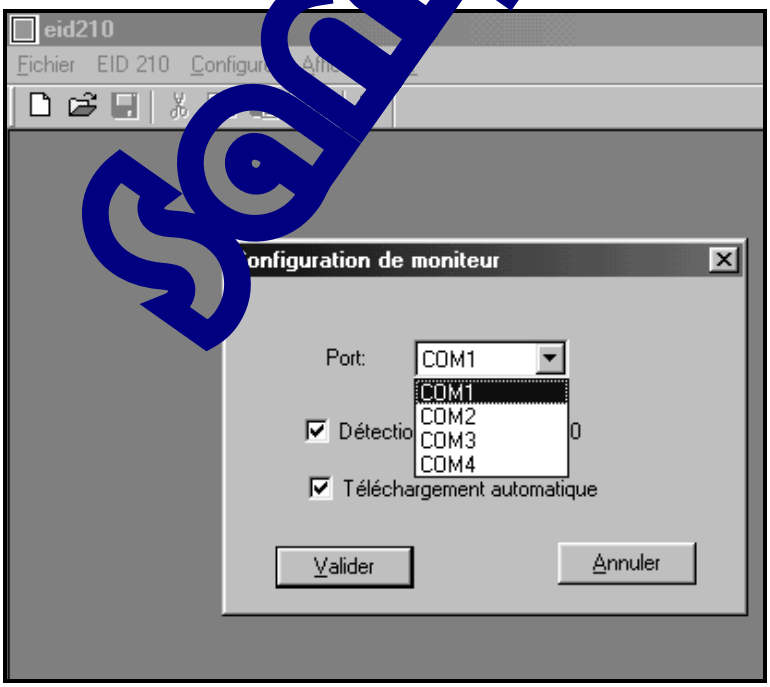
In case of communication failure between the target and the target Board EID 100 000, as indicated therebelow:



Check and parameter correctly the serial link,

When all files are closed,

Click on "**Configure**", then on " **Monitor Co gura**



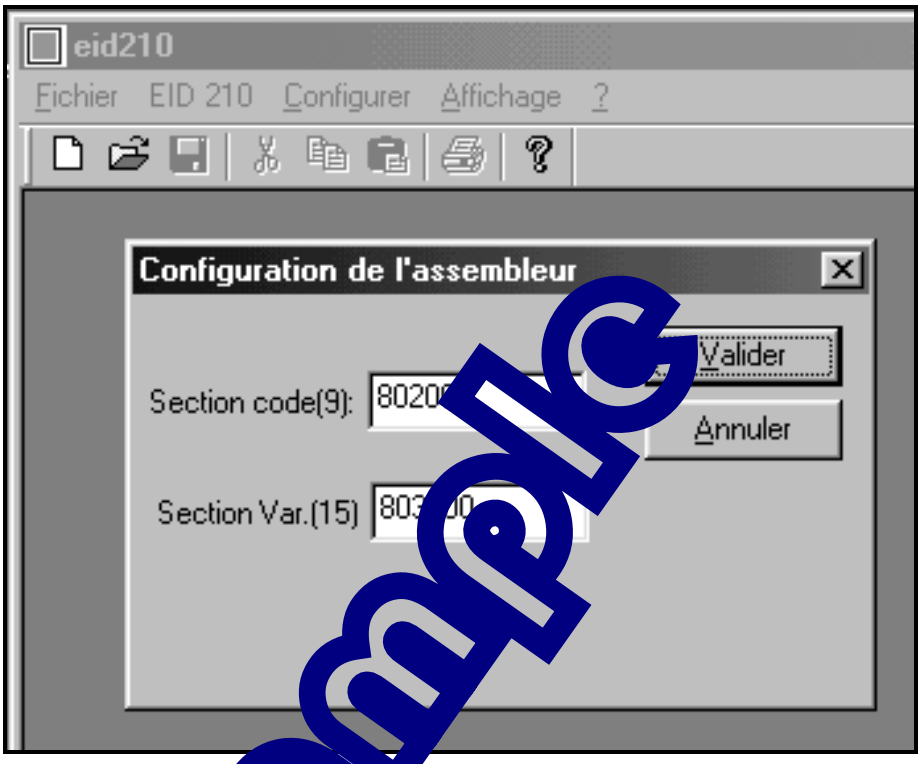
Activate the serial link in the software window corresponding to the used hardware link, and activate options " **Automatic detection** ", and " **Automatic Downloading**".

ANNEX n°2 :

Configuration of the Assembler:

Click on "**Configure**"

Click on "**Assembler**".



- Code section (9) : Address of the program starting, in live memory on the target Board ($802 000).

- Variable section (15) : Starting address of variables used in the Assembler program in live memory on the target Board ($803 000).


Samplc

EXERCISE N°1: WRITING IN A RAM ZONE

1.1Topics

Purpose :	Learning of how to handle a conditional connection instruction in an Assembler program.
Specification :	Writing a program in Assembler for placing the alphabet letters from « A to Z » in the memory zone, starting at the address $804000. ➢ Save registers used in the cell : a0, d0 ➢ Load starting address $80400[illegible]ister a0, ➢ Initialise d0 register with t[illegible]ter « A », ➢ Place the current ch[illegible]t address, ➢ Prepare the follow[illegible]rementing d0, ➢ Check if the current [illegible]», ➢ Loop if the current [illegible]», ➢ Restore the cont[illegible]0 and d0, ➢ Loop the progra[illegible], ➢ End of the prog[illegible]. ➢ As a[illegible] same program in using the dbf function as a lo[illegible] ➢

Je[illegible]y Test Equipment :

PC Micro Computer usin[illegible]Windo[illegible]® 95 or later,

68332 16/32 bits micro-contro[illegible]ther Board, Ref. : EID 100 000

USB connection cable, or if not available an RS232 cable, Ref. : EGD 000 003

AC/AC 8V, 1 A Power Supply, Ref. : EGD000001,

Duration : 2 hours

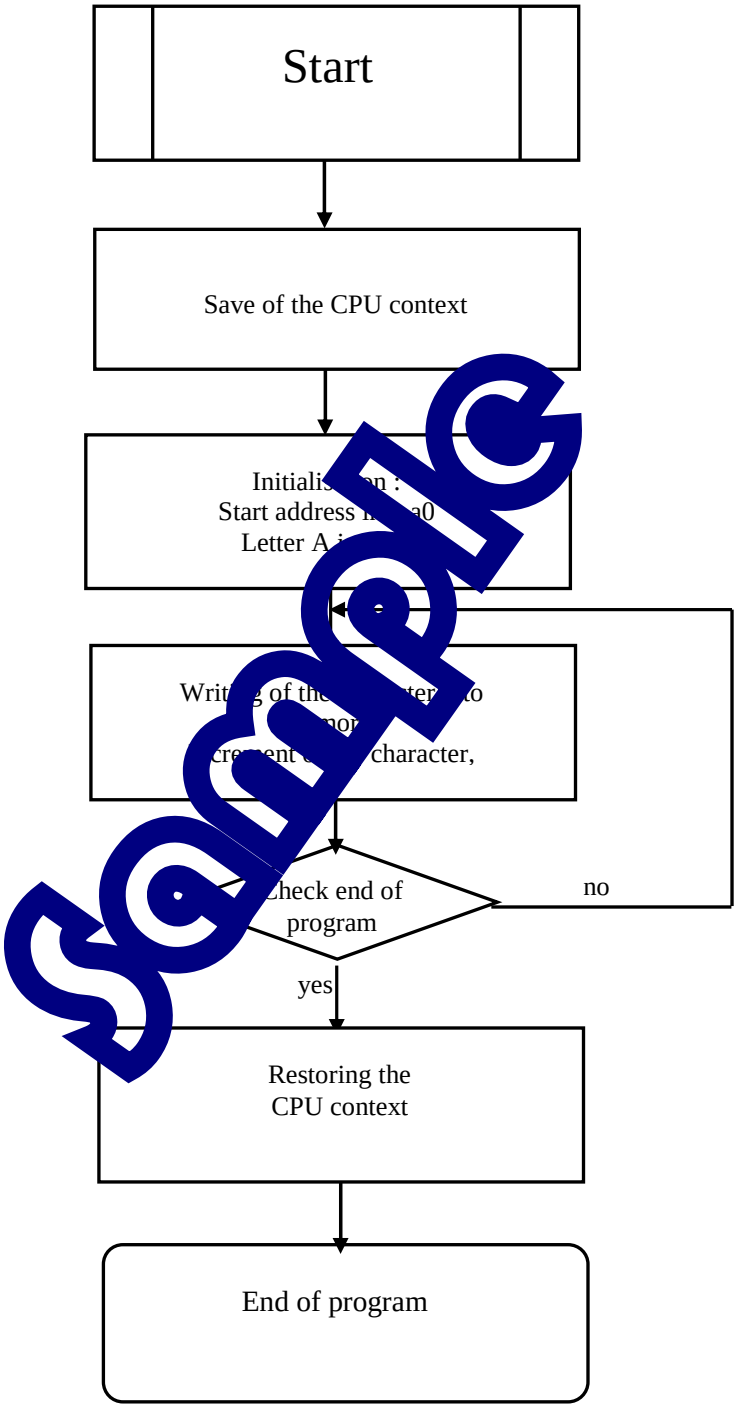
1.2 Detail of specifications :

- Save registers used in the cell : a0, d0
- Load start address $804000 into register a0,
- Initialise register d0 with the first letter « A »,
- Place the current character to the current address,
- Prepare the following character, in incrementing d0,
- Check if the current letter is « Z »,
- Loop again if the current letter is # « Z »,
- Restore the register context a0 and d0,
- Loop again the program,
- End of the program.

- As a variant, do the same program again in using the db[illegible] a loop condition.

0.9 Variant solution n°1 :

0.9.1 Variant Flowchart n°1

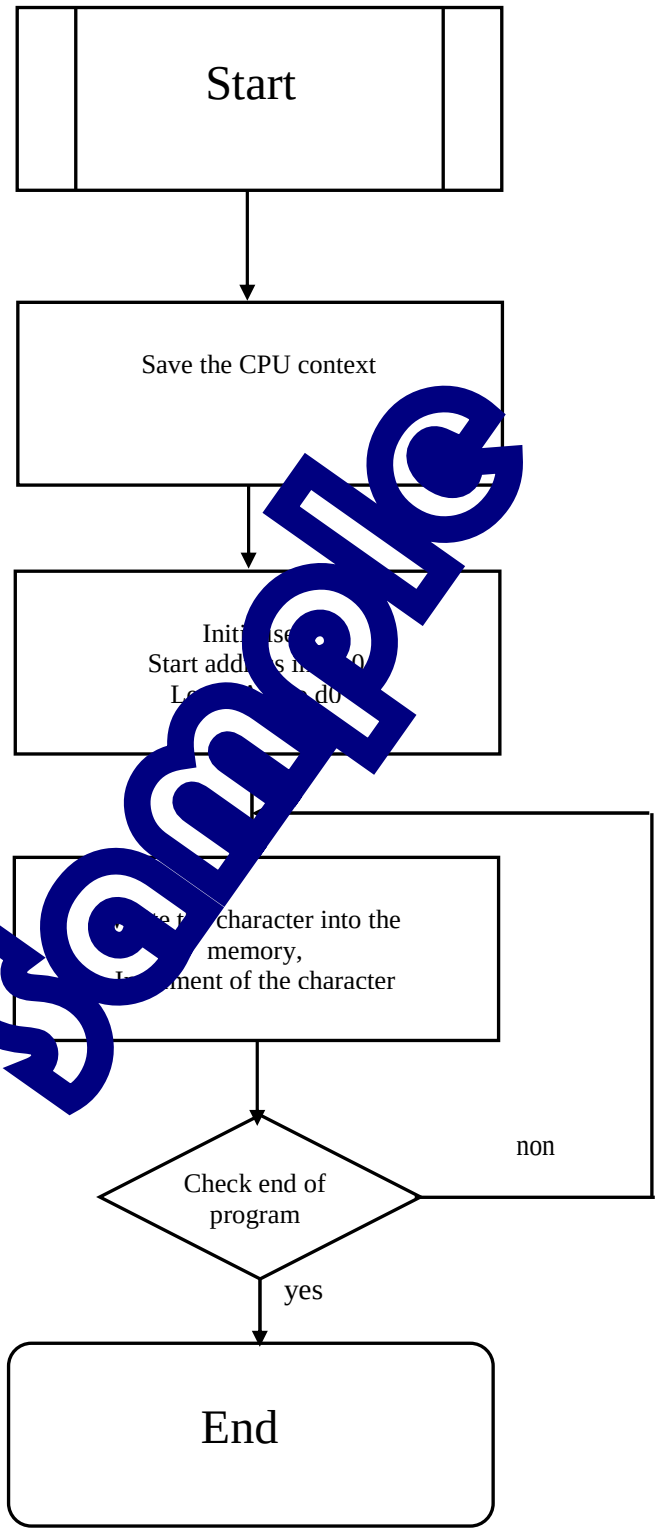


0.9.2 Variant n°1 Program in 68xxx Assembler

```
*************************************************************************
*                    EXERCISE  ON EID210 BOARD ITSELF                   *
*************************************************************************
* Title   :  Filling of the memory with the increasing letters of the alphabet.   *
* Language: 68000  Cross Assembler: System: Pack EID 100 DMS DIDALAB      *
*************************************************************************
*
*
*
*
*
        include         EID210.def              * Definitions peculiar to the processor  Board  elements
        section  code   $803000         * Start of the program  section code ($803000)


*************************************************************************
*
*                           Initialisation,
*                       The used registers are saved into the cell
*
*************************************************************************
*
        movem.l  a0/d0,-(sp)            * save of  registers int  e cel
        movea.l  #$804000,a0            * Address of the wr  g
        move.b   #'A',d0                * First letter (A) AS  capit


*************************************************************************
*
*       Remark: for changing into small  letters, it is sufficient to  tia  w  "a" value                *
*                       and check at last  "z" value

*************************************************************************
*
*       Start of the main program
*
*************************************************************************
Loop_1
        move.b   d0,(a0)+                 nt character  into the memory
        addq.b   #1,d0                  * Prep  ollowing (increment)
        cmp.b            #'Z',d0        ck t  ast character (comparison with " z ")
        bls.s            boucl          if not, go back to loop_1
        movem.l  (sp)+,a0/d0            * If ye  estore the context

        jmp              MON  EUR       * End of  program and back to the Monitor control

end                                     * End of program
```

0.10 Solution of the Variant n° 2

0.10.1 Flowchart variant n°2 :



0.10.2 Program of the variant n°2:

```
*************************************************************************
*                    EXERCISE ON THE  EID210 BOARD ITSELF                *
*************************************************************************
* Title     : Filling  memory with  alphabet  letters.                    *
* Language: 68000 Cross Assembler     : System: Pack EID 100 DMS DIDALAB   *
*************************************************************************



* Another solution is also possible in using  instruction  of
loop primitive dbcc 3 parameters:
*                   - 1 data register used as a counter,
*                   - 1 conditional (un) plugging
*                   - 1 label
*
*
*
*
          include   EID210.def                  * Definition on  Board elemen

          section   code      $803000   * Start of the program ($8030
*
*
*************************************************************
*
*                              Initialisation,
*                          Used registers are saved in the cell
*
*************************************************************
*
          movem.l   a0/d0,-(sp)         * Save of regi   s in the c
          movea.l   #$804000,a0         * Address of    iti
          move.l    #25,d1              * Counter wit   umb     ters minus 1 !!
          move.b    #'a',d0             * First le      let     alphabet


*************************************************************
*
*         Loop of the main progra
*
*************************************************************

Loop_2
          move.b    d0,(a0)+                    * Memory writing , + increment of a0
          dbf.s               d1, boucle_2      * dbf,      * Test d1=0, condition always wrong , output
                                                * When the content of d1 = -1
          jmp                 MONITEUR          * Jump under Monitor control

          end                                   * End of program
```

TP 1 : DIODE CONTROL ON MICRO-CONTROLLER"QS" PORT

1.1 Topics

Purposes:	Being capable of controlling the 3 diodes (labelled D10,D11and D12) connected to "QS" port of the 68332 Micro-controller . Being capable of detecting pressing down key labelled "CTRL". Being capable of implementing the micro-controller internal "Timer" in interrupt mode in order to carry out a time base.
Specification :	**Topic 5-1:** Writing of a program in Assembler language for carrying out a cycle with three led connected to the micro-controller QS port. Switching from one state to another is made by pressing down key "CTRL" In fact, we want to carry out the following cycle : ➢ Switching on the led labelled D10 (the 2 others are off) ➢ Switching on the led labelled D11 (the 2 others are off) ➢ Switching on the led labelled D12 (the 2 others are off) ➢ Loop **Topic 5-2:** The same cycle than the previous one must be carried out, but in automatic control (without having to press down key "CTRL"). Switching from one state to another is made after a lapse of time of about one second, carried out by a delay program. **Topic 5-3:** Identical to the previous specification, but in using micro-controller internal "Timer" in interrupt mode.

Necessary Test Equipment :

PC Micro Computer using Windows ® 95 or later,

68332 16/32 bits micro-controller mother Board, Ref. : EID 100 000

USB connection cable, or if not available an RS232 cable, Ref. : EGD 000 003

AC/AC 8V, 1 A Power Supply, Ref. : EGD000001,

Duration : 4 hours

1.2 Solution

1.2.1 Analysis

"Control" of electroluminescent diodes

These three diodes D10, D11 and D12 are connected to the micro-controller QS port

- D10 to PQS4 link
- D11 to PQS5 link
- D12 to PQS6 link

(Resource document : Structural layouts of the Board, "sheet 5 and 6")

The three bits of QS port must be configured in the output :

➔ Enter levels 1 to the corresponding positions of QS port control registers
bit n°4 to 1 ; bit n°5 to 1 ; bit n°6 to 1
reference : 7654 3210

➔ As the register is a 16 bits register we have: 0000 0000 0111 0000 -> in Hexadecimal: $0070

➔ This register address is specified in the definition file with label PQSCTR.

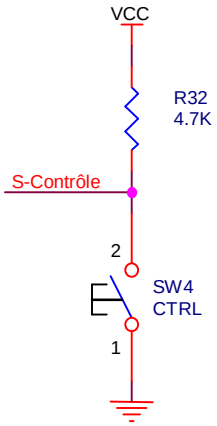
For switching a led on, we must enter 0 into the data register of the QS port :

➔ This register address is specified in the definition file with label PQSCTR

➔ For switching only led D10, we must write 0000 0000 1[illegible]0

Detection of pressing down key "CTRL" :

Following the diagram on the left, pressing down key "CTRL" leads to the logic state '0' on the "S-Control" signal.
The state of this "S-Control" signal is available in the [illegible] register on line 8 :



15 14 13 12	11 10 9	8	7 6 5 4	3 [illegible] 0

Remark :

- The state register is available in using the "REG[illegible]T" label which address is specified in the file to be included "D2[illegible].def".
- For knowing the state of the key, it is enough to read the state register and do a logic AND with a mask at : %0000 0001 0000 0000 = $0100

If the AND result gives $0[illegible]0, it is because the key is pressed down, on the other hand, if the result gives $0[illegible]0, it is because the key is released.

Carrying out of a program-type time delay :

The time delay is carried out in initialising one variable to a certain value and decrementing this value until it is equal to zero. The carrying out duration of this decrement loop constitutes the requested lapse of time. In the following program the variable is included in register d0.

Carrying out of a time delay in using micro-controller internal "Timer" :

For having a periodic interrupt 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

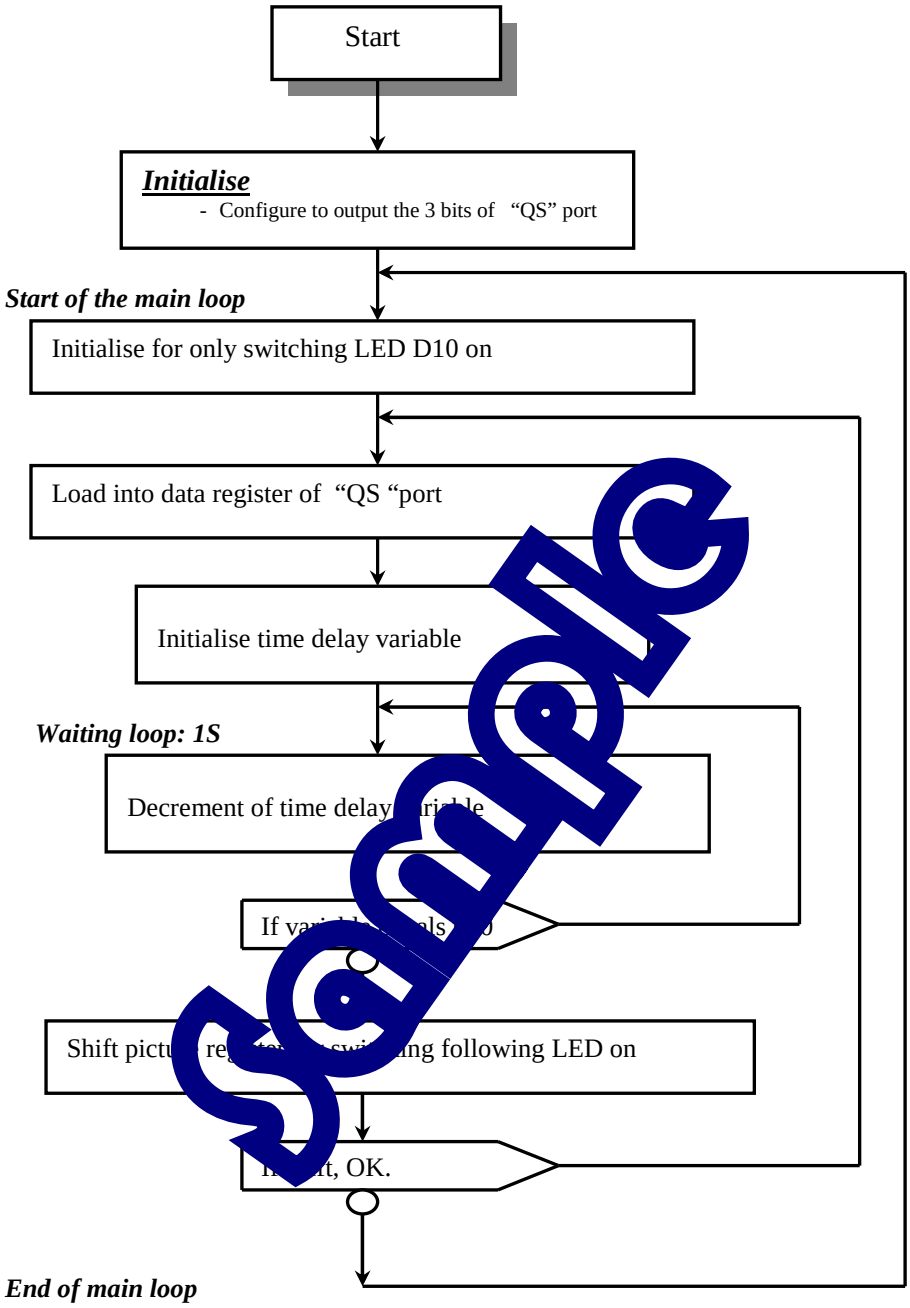
"**PITR**" (**P**eriodic **I**nterrupt **T**imer **R**egister) to $0008.

In other respects, the vector table must be initialised and the interrupt program already allowed.

1.2.2 Program for specifications 5-1

```
            ************************************************************
            *                    PRACTICALS ON EID210 BOARD ITSELF                    *
            ************************************************************
            *  Check the 3 led on  QS port  and  input "CTRL" control    *
            *                                                                        *
            *  Specifications :                                                      *
            ********************                                                    *
            *  Every  press down  button  CTRL, makes another LED switching  on,     *
            *  let  be cycle  D10 -> D11 -> D12 -> D10 ... etc                       *
            *             FILE NAME:  T_PQS.SRC                                      *
            ************************************************************

*  Inclusion of the file specifying the different  labels
            include             EID210.def
            section             code
*
************************************
*           INITIALISE                 *
************************************
* Configure on outputs, the 3 bits on QS port,  on which diodes are connected
Début       move.w              #$0070,PQSCTR     * 3 LED outputs
************************************
*           MAIN LOOP          *
************************************
* Switch on (using level  0)   the reference LED  D10 connected on bi   QS p
DebBP       move.w              #$10,d0
AFF         move.w              d0,d1
            not.w               d1                * Complemen    r swit
            and.w               #$0070,d1         * Only valid   th     d o uts
            move.w              d1,PORTQS         * Load to th   QS

*  Detect  press down key  "CTRL"

*  Wait as long as key  "CTRL" is pressed down
ATT1        move.w              REG_ETAT,d2
            and.w               #$0100,d2
            beq                 ATT1                  Loo     TRL" is pressed down

* Wait as long as key "CTRL" is released
ATT2        move.w              REG_ETA   2
            and.w               #$0100,d2
            bne                 A                     if key  "CTRL" is released

* Press on key "CTRL" has be    dete                  llowing LED
            lsl                 #   0
            btst                #7,d                  *  Check  if  passing out
            beq                 AFF                   *  If not, display
            bra                 DebBP                 * If passing out, re-initialise
*
*           END of main loop and of program
************************************************
            end          * End of file
```

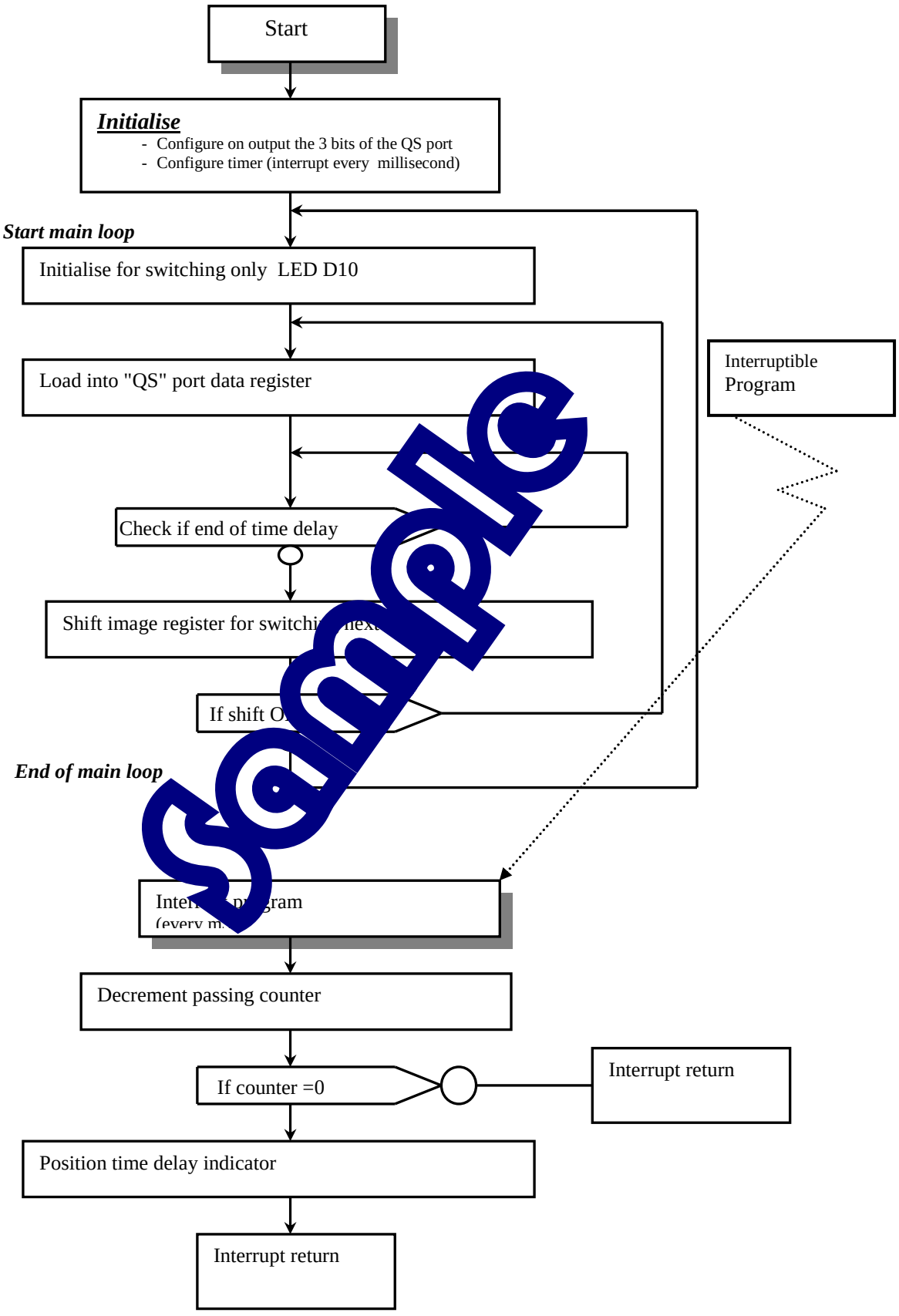
1.2.3 Flowchart for specifications 5-2



1.2.4Program for specifications 5-2

```
*************************************************************************
*                        PRACTICALS ON EID210 BOARD ITSELF                *
*************************************************************************
*  Carry out a light sequential string  with the 3 led on QS port        *
*                                                                        *
*  Specifications:                                                       *
*********************                                                   *
*  The led switch on following the cycle:                                *
*           D10 -> D11 -> D12 -> D10 ... etc                             *
*  Every led lights up during about 1 Sec.                    *
*  This time is determined by a “program” loop                           *
*  FILE NAME:  CHENI_1.SRC                                               *
**********************                                                  *
*************************************************************************
* File inclusion for specifying the different labels
include         68332.def

section         code

*          INITIALISE
***************************
* Configure on outputs, the 3 bits on QS port,  on which the diodes are connec
Début   move.w    #$0070,PQSCTR     * 3 LED outputs

*           MAIN  LOOP
************************************
* Switch on (using level  0)   reference LED  D10 connected to bit 4 (Q
DebBP   move.w    #$10,d0
ALUM    move.w    d0,d1
        not.w           d1              * Complem  fo  chi  on 0
        and.w           #$0070,d1       * Only valid  the  puts
        move.w    d1,PORTQS             * Loa

*  Wait loop of about 1 second
        move.l          #$001FFFFF,d2
ATT     sub.l           #1,d2
        bne             ATT

* Go to next LED
        lsl             #1,d0
        btst            #7,d0           Che  if  passing out
        beq             A               t, display
        bra             bBl             * assing out, re-initialise

* END of main loop and of prog
*****************************************
        end
```

1.2.5 Flowchart for Specification n° 5-3



1.2.6 Program for specification n° 5-3

```
*************************************************************
*                  PRACTICAL ON EID210 BOARD ITSELF         *
*************************************************************
* Carry out a light sequential string  with the 3 led on QS port       *
*                                                                      *
*  Specifications:                                                     *
**********************                                               *
* The led switch on following cycle:                                   *
*           D10 -> D11 -> D12 -> D10 ... etc                           *
* Every led lights up during about 1 Sec.                              *
* This time is determined by  the 68332 time base                      *
*  FILE NAME:  CHENI_2.SRC                                             *
**********************                                               *
*************************************************************
* File inclusion for specifying the different labels
            include             EID210.def
*           Declaration of variables        *
*******************************************
            section     var
COMPTEUR    ds.l        1
INDICATEUR  ds.b        1
            section     code
***********************************************
*           MAIN PROGRAM                      *
***********************************************
*           INITIALISE
**********************
* Configure on outputs, the 3 bits on QS port,  on which diodes are connected
Début       move.w    #$0070,PQSCTR     * 3 LED Outputs
* Configure time base
            move.l              #96,d0                  * Loads the interrupt vector number
            asl.l               #2,d0
            add.l               #tab_vect,d0            * Initialise the vectors table
            move.l              d0,a0
            move.l              #it_bt,a1               * Loads the address of the interrupt function
            move.l              a1,(a0)
            move.l              #1000,COMPTEUR          * 1000*1mS = 1S
            move.b              #$00,INDICATOR          * Beginning of counting
            move.w              #$0008,PITR             * 1 interrupt every  1 ms
            move.w              #$0760,PICR
*            MAIN LOOP
*********************************
* Switch on (using level  0)   reference LED D10 connected on bit 4 (QS port )
DebBP       move.w              #$0,d0
ALUM        move.w              d0,d1
            not.w               d1                      * Complement for switching on 0
            and.w               #$0070,d1               * Only validate the 3 led outputs
            move.w              d1,PQSDATA              * Load to QS port
* Wait  loop of end of time delay
ATT         move.b              INDICATEUR,D2
            cmp.b               #01,D2
            bne                 ATT
            move.b              #$00,INDICATEUR
* Go to the next LED
            lsl                 #1,d0
            btst                #7,d0                   * Check  if  passing out
            beq                 ALUM                    * If not, display
            bra                 DebBP                   * If passing out, re-initialise
* END of main loop and of program
*********************************************
*           INTERRUPT FUNCTION              *
*           linked to the time base         *
*********************************************
it_bt       sub.l               #$00000001,COMPTEUR
            cmp.l               #$00000000,COMPTEUR
            bne                 it_ret                  * Return if not equals to 0
            move.b    #$01,INDICATEUR                   * End of  time delay
            move.l              #1000,COMPTEUR          * Re-initialisation of  time delay
it_ret      rte                                         * Interrupt return
*  End of  interrupt function
*************************************
            end                                         *  End of source file
```


Samplc

EXERCISE N°2 - CARRYING OUT OF AN "ECHO" MODE FROM THE TERMINAL

1.3 Topics

Purpose :	Being capable of configuring and using the RS 232 serial communication function (internal to the 68332 micro-controller), first in "Transmission" mode ("simplex" link), then in "Transmission-Reception" mode("duplex" link). Being capable of detecting a transition (state variation) on a logic input. Being capable of defining constants (constant message in ASCII characters) and variables.
Specification :	**Subject n°2.1:** Sending of pre-defined [illegible] terminal (connected to RS232 serial link) whenever the "CTRL [illegible]n is pressed down. **Subject n°2.2:** When starting the pr[illegible]e is a pre-defined message sending (chain of characters).The [illegible]rries out the "echo" mode : If a key from [illegible]ard is pressed down, then the character is sent back (displ[illegible]een). Remark: I[illegible]e link is "half duplex" type, because transmission and recep[illegible]ultaneous.

Necessary Test Equipment :

PC Micro Computer using Windows ® 95 or later,

68332 16/32 bits micro-controller mother Board, Ref. : EID 100 000

USB connection cable, or if not available an RS232 cable, Ref. : EGD 000 003

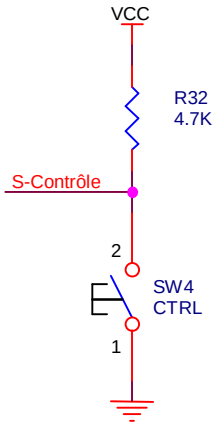
AC/AC 8V, 1 A Power Supply, Ref. : EGD000001,

Duration : 4 hours

1.4 Analysis of subject 2.1

Detection of pressing down key "CTRL" :

Following the diagram on the left, pressing down key "CTRL" leads to the logic state '0' on the "S-Control" signal.
The state of this "S-Control" signal is available in state register on line 8 :



15 14 13 12	11 10 9	8	7 6 5 4	3 2 1 0

Remark :
- The state register is available in using "REG_ETAT" label which address is specified in the file to be included "EID210.def".
- For knowing the state of the key, it is enough to read the state register and do a logic AND with a mask of value : %0000 0001 0000 0000 = $0100

If AND result gives $0000, it is because key is pressed down, on the other hand, if result gives $0100, it is because the key is release

1.4.1 Use of serial communication interface

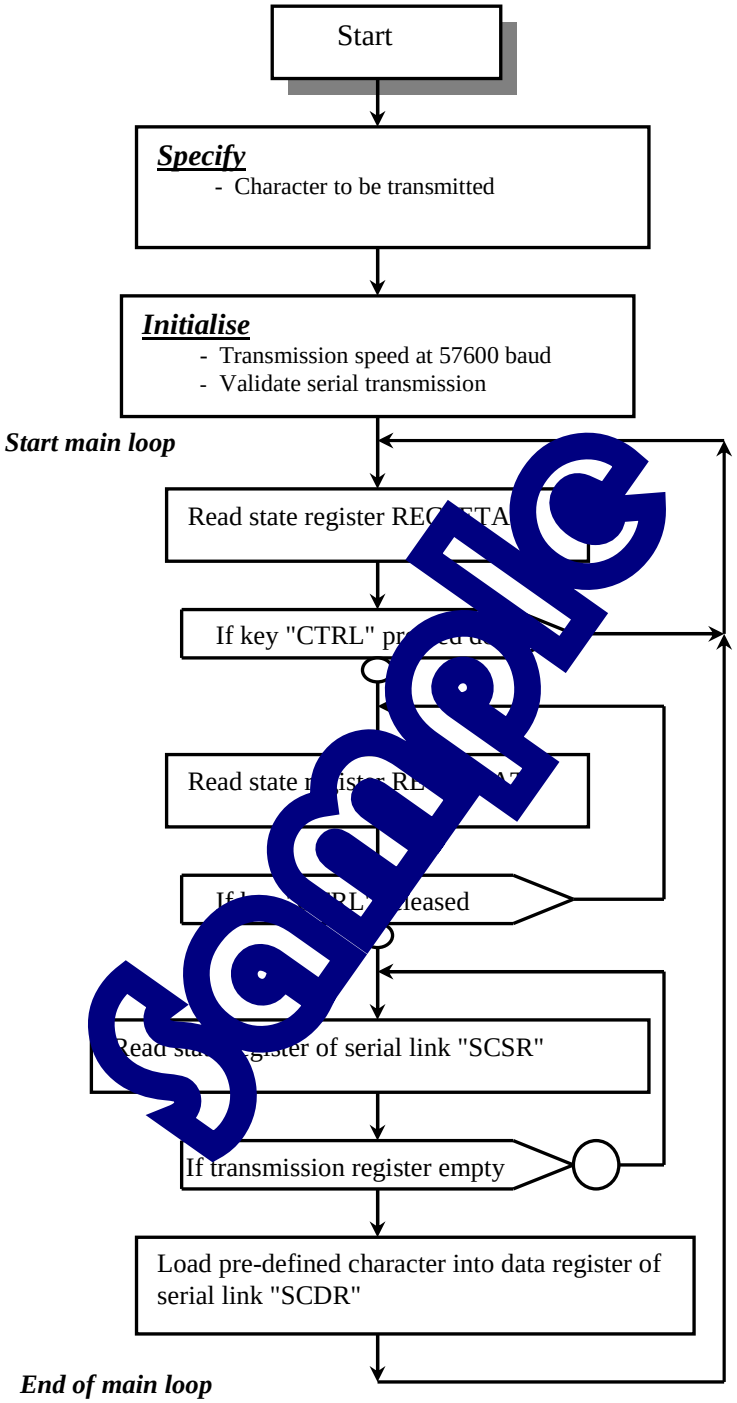
The use of serial interface is carried out by 4 *16 bits regis... whi... labels and addresses have been specified in file of definitions to be included EID210.... :

➔ Two control registers (**S**erial **C**ommunicati... **C**...ol ...egister)
"**SCCR0**" for specifying the communicat... ...ed ...wing formula :
Baud rate = System fr...uency ... da...
With "data" the value to be l... ...ec... re...ste... "SCCR0" is
" System frequency ", the i...er... ...ope...on frequency which is a multiple of the quartz frequency connected to inp...s "... ...and "EXTAL" of the micro-controller.
For complying to the M... ...m...nication velocity (57600 Baud), this register must be initialised at 9.
"**SCCR1**" for specif...g o... ra... ...o... :
bit of rank 2 : (RE Re... v Enable) must be switched to 1 for enabling reception,
bit of rank 3 : (TE Tra... ...e) must be switched to 1 for enabling transmission.
This register must be ...tialised ...o %0000 0000 0000 1100 =$000C.

➔ One data register called "S... ...R" (**S**erial **C**ommunication **D**ata **R**egister).
Under this only label, there are two registers, one is used for transmission (allowed for writing) the other one for reception (allowed for reading).
For transmitting a character via the serial link, loading ASCII code into register SCDR is sufficient (provided having checked before that it is empty). For receiving a character, reading the ASCII code in register SCDR (provided having checked before that it is full).

➔ One state register called "**SCSR**" (**S**erial **C**ommunication **S**tatus **R**egister) with :
* bit of rank 8 ("**TDRE**" **T**ransmit **D**ata **R**egister **E**mpty) is at 1 when the data register is empty, which indicates that a character can be transmitted,
* bit of rank 6 ("**RDRF**" **R**eceive **D**ata **R**egister **F**ull) is at 1 when the data register is full, which indicates that a character has been received that can be read on the data register "SCDR".
Label masks "**TDRE**" and "**RDRF**" have been specified in file EID210.def, enabling the checking of the state of these bits.

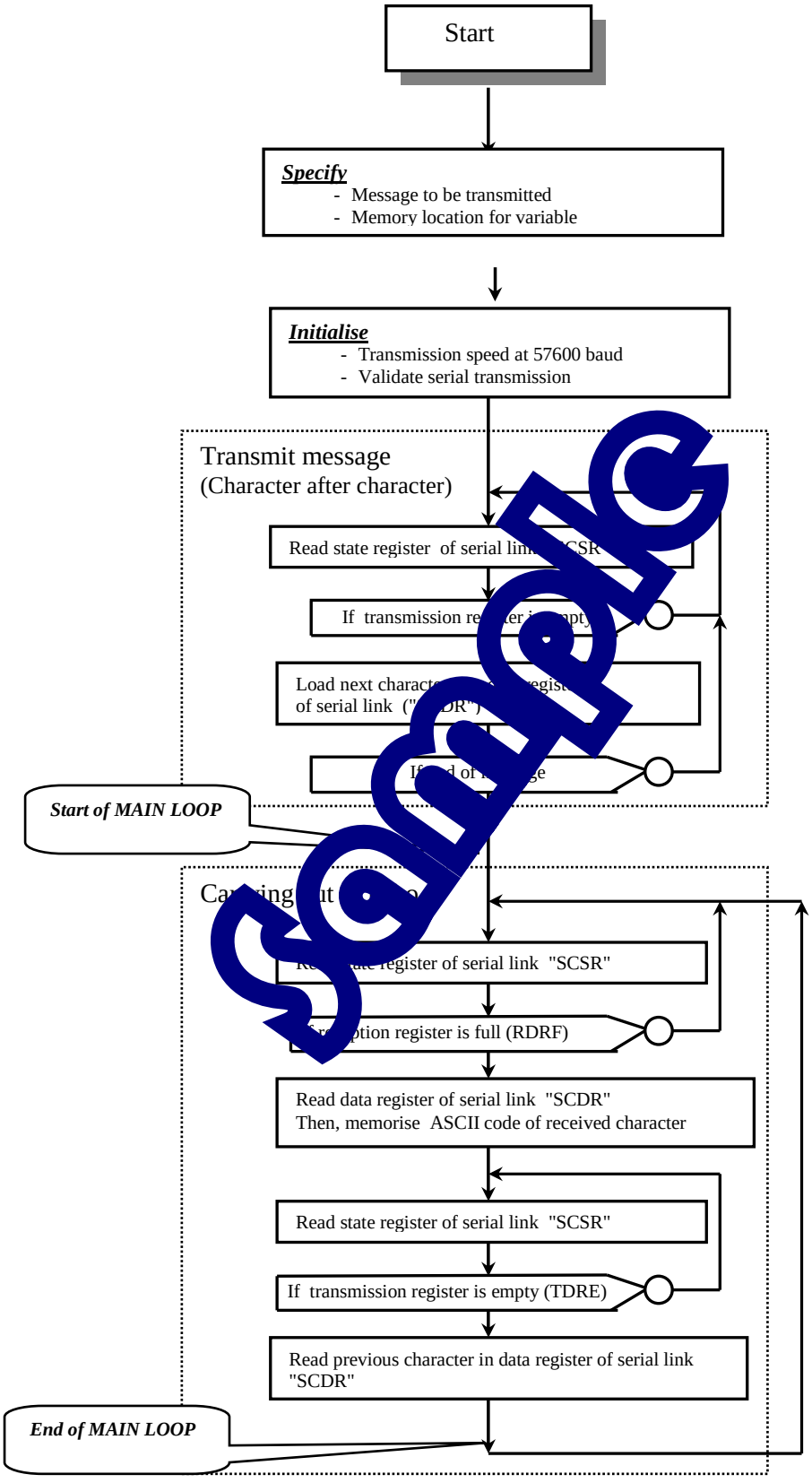
1.4.2 Flowchart topic n° 2.1



1.5 Program on topic n°2.1 in 68xxx Assembler

```
          ***************************************************************************
          *                   PRACTICALS  ON EID210 BOARD ITSELF                     *
          ***************************************************************************
          *         TRANSMIT A CHARACTER TO SERIAL COMMUNICATION PORT*
          *                                                                         *
          * Specification :                                                         *
          *******************                                                       *
          * When pressing down key  "CTRL" located on EID210, a character must be   *
          *  displayed onto the computer screen                                     *
          *                                                   *
          *                                                                         *
          * FILE NAME:  T_SERIE1.SRC                                                *
          ********************                                                      *
          ***************************************************************************
*         DEFINITION & DECLARATIONS      *
*****************************************************
* Inclusion of file specifying the different labels
          include            EID210.def
*******************************************
*         Declaration of character to be transmitted        *
*******************************************
Char                EQU      $30                     * Chara[illegible]0 wi[illegible] (A[illegible] Code : $30)

          section            code
********************************************************
*         START OF EXECUTABLE PROGRAM*
********************************************************
*         INITIALISE
************************
* The next initialisations  are inhibited because the Monitor a[illegible]ady [illegible]d th[illegible]ial port !
*  Transmission speed
Début               move.w   #9,SCCR0                * [illegible] spe[illegible] of  57600 Baud
* Validate transmission (bit "RE")
                    move.w   #$0004,SCCR1

*************************************
*          MAIN LOOP       *
*************************************
Deb_BP                                               * Start of main loop
*  Detect  pressing down key "CTR[illegible]

*  Wait as long as key  "CTRL" is p[illegible]sed down
ATT1                move.w   REG_[illegible]
                    and.w    #$0100,d2
                    beq      ATT1                    * Loop if  key "CTRL" is pressed down

* Wait as long as key  "CTRL" is released
ATT2                move.w   REG_ETAT,d2
                    and.w    #$0100,d2
                    bne      ATT2                    * Loop if key  "CTRL" is released

* Pressing down key  "CTRL" is detected
* Wait till transmission is free
ATT3                move.w   SCSR,d0                 * Acquisition of state register of serial link
                    and.w    #TDRE,d0                * Bit  indicating that transmission register is empty
                                                     * Transmit Data Register Empty
                    beq      ATT3                    * Loop if  transmission not ready
                    * Transmit character
                    move.w   #char,SCDR

                    bra      Deb_BP                  *Loop
*******************************************
* END of main loop and program                      *
*******************************************
          end       * End of Assembler source file
***************************************************
```

1.6 Analysis of topic n°2.1



1.7 Program on topic n°2.2 in 68xxx Assembler

```
*****************************************************************************************
*                          PRACTICALS ON EID210 BOARD ITSELF                            *
*****************************************************************************************
*         TRANSMIT THE RECEIVED CHARACTER BACK TO SERIAL                                *
*            COMMUNICATION PORT                                                         *
*  Specification :                                                                      *
*******************                                                                    *
*  - When starting program, there is a message sending                                 *
*  -  Then, the program carries out the "echo" mode :                                  *
* If a character is pressed down on the computer keypad,, it comes back, displayed onto the screen *
*  FILE NAME:  T_SERIE2.SRC                                                            *
*****************************************************************************************
*           DEFINITION & DECLARATIONS       *
*********************************************
* Inclusion of file specifying the different labels
          include              EID210.def
*************************************
*           Declaration of variables        *
*************************************
          section    var
Message             dc.b       ' BONJOUR!   Press a key down, the character  must  p  d
Char                ds.w       1                    * For memorising the received  ar  e
          section    code
*********************************************************
*           START OF EXECUTE PROGRAM        *
*********************************************************
*           INITIALISE
***************************
*  Transmission speed
Start               move.w     #9,SCCR0             * For having a s  ed  600  aud
* Validate transmission & reception
                    move.w     #$000C,SCCR1
                    move.l     #0,d1                * Into d1  f tra  itted characters
                    move.l     #Message,A1          * Into A  the add  ssa  start
* Sending of message    Wait free transmission
ATT1                move.w     SCSR,d0              * A  isition  egister of serial link
                    and.w      #TDRE,d0             * B  nd  g if  smission register is free
                    beq        ATT1                 * L  if t  on unready
                    move.b     (A1),d0
                    move.w     d0,SCDR
                    add.l      #1,d1                * Pass t  ext character
                    add.l      #1,A1                *  k i  essage transmit carried out
                    cmp.l      #66,d                *  66 characters in the message
                    bne        ATT
ATT2                move.w     SCS  d0                quisition of  state register of serial link
                    and.w      #TDR  0              Bit indicating  if transmit register empty
                    beq        ATT2                 Loop if  transmit unready
                    move.w     #$0D,SCDR            $0D is ASCII code of CR "Enter"
ATT3                move.w     SCSR,d0
                    and.w      #TDRE,d0
                    beq        ATT3                 * Loop if  transmit unready
                    move.w     #$0A,SCDR            * $0A is ASCII code of LF "jump line"
*           MAIN LOOP       *
*************************************
                    * Wait character reception
Deb_BP    move.w    SCSR,d0                         * Acquisition of  state register of serial link
          nd.w      #RDRF,d0                        * Bit indicating if receive register full
                                                    * Receive Data Register Full
                    beq        Deb_BP               * Loop if nothing received
                    move.w     SCDR,char            * Received character recovered
                    * Wait transmission ready
AT2                 move.w     SCSR,d0              * Acquisition of  state register of serial link
                    and.w      #TDRE,d0             * Bit  indicating  if transmit register empty
                                                    * Transmit Data Register Empty
                    beq        AT2                  * Loop if  transmit not ready
                    *Transmit received character back
                    move.w     char,SCDR
                    bra        Deb_BP               * Boucler
* END of main loop and program
*****************************************
                    end                             * End of Assembler source file
```

TP 2 : GIVE VALUE TO A REGISTER SPECIFIED BY THE USER

2.1 Exposition of the Topics

Purpose :	- Being capable of configuring and using the serial RS 232 communication function (function internal to the 68332 Micro-controller), in "Transmission-Reception" mode ("duplex" link). - Being capable of acquiring a character and checking its relevance, then executing a pre-specified action (a[illegible]ring by a pre-specified message). - Being capable of converting [illegible]inary word into ASCII 16 characters. - Being capable of str[illegible]progra[illegible]n requiring repetitively, to sub-programs (Assembler) o[illegible]nct[illegible]language).
Specifications :	Initialise to rema[illegible]valu[illegible]e data registers not used in the program, on 16 bits : D2 = $7[illegible]22 , D[illegible]233[illegible]etc … When lau[illegible]ni[illegible]the [illegible]gram, there is transmission of a pre-specified message (chain of c[illegible]arac[illegible] **NUMERO DU REGISTRE ? DE 2 à 7** ' (Register numbe[illegible]2 t[illegible]). W[illegible]n [illegible]e[illegible]nters the register number he wants to know the value, the a[illegible]am co[illegible]ols the received code, and displays the message : [illegible] **DU REGISTRE NON VALIDE** ", (not valid register number) if error. [illegible]herwise, it reads the specified register then, transmits the answer as : **d = xxxxxxxxxxxxxx** (w[illegible] as different binary states). Then, go back to Start position (asking of register number).

Necessary Test Equipment :

PC Micro Computer using Windows ® 95 or later,

68332 16/32 bits micro-controller mother Board, Ref. : EID 100 000

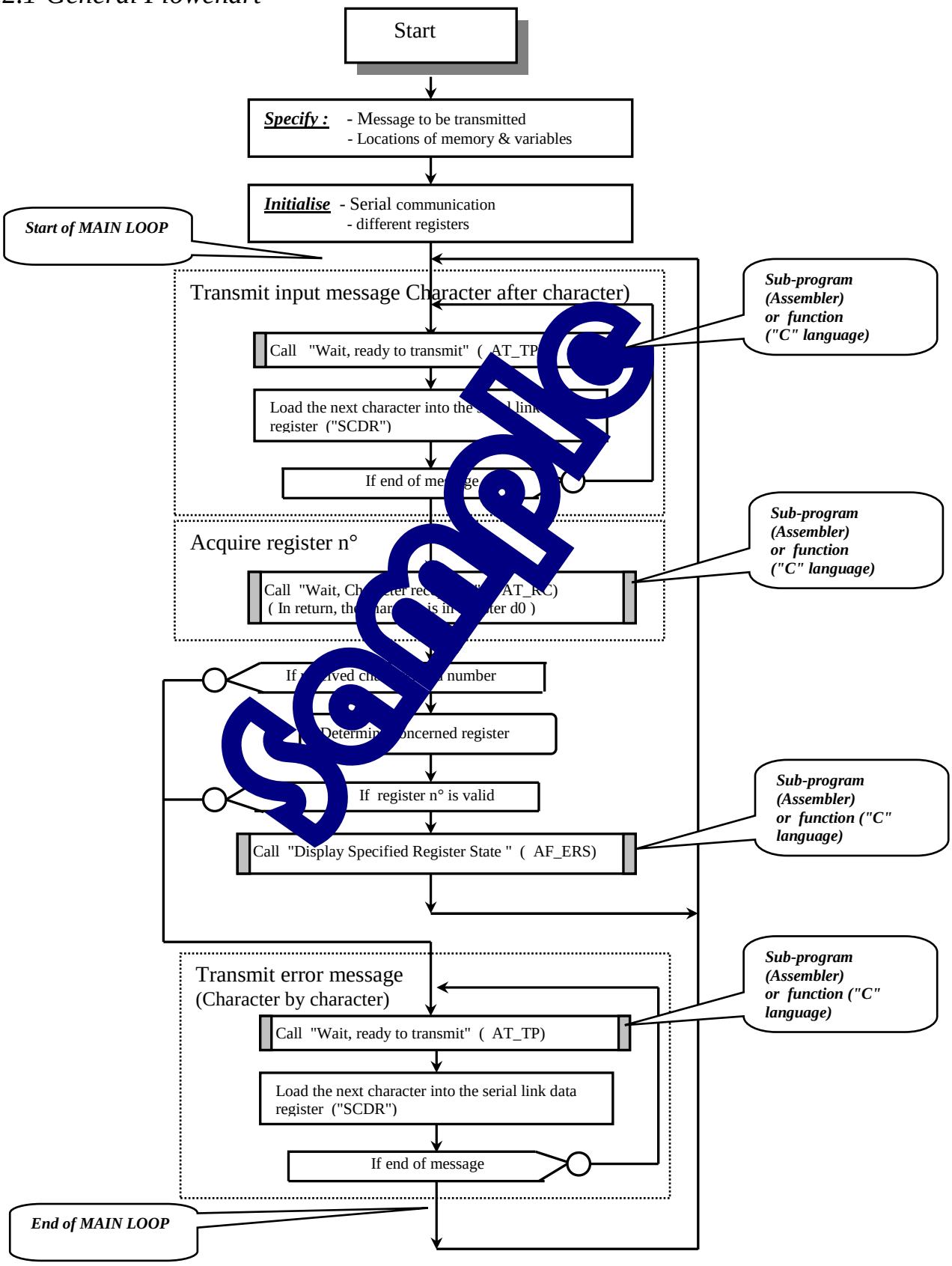
USB connection cable, or if not available an RS232 cable, Ref. : EGD 000 003

AC/AC 8V, 1 A Power Supply, Ref. : EGD000001,

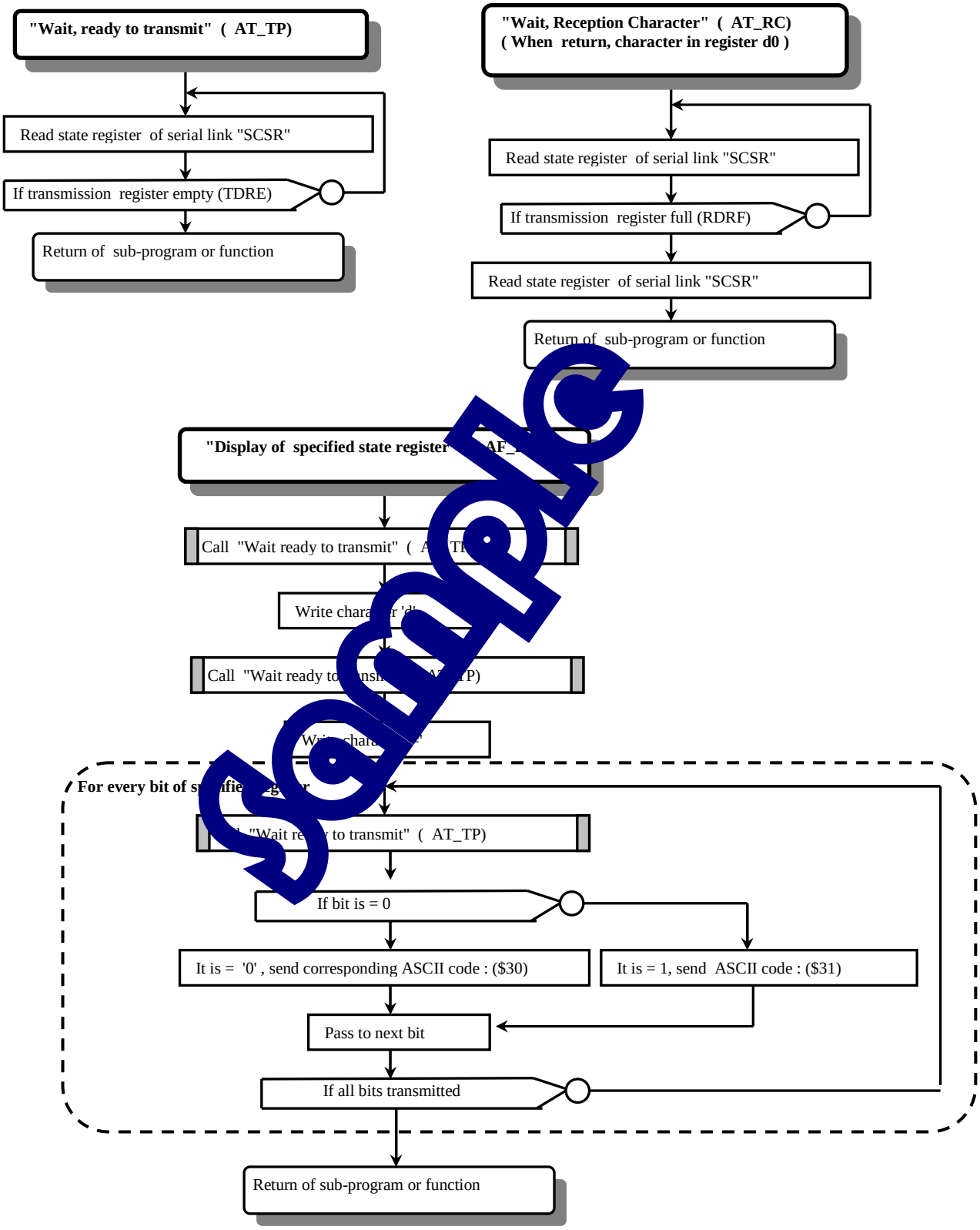
Duration : 4 hours

2.2 Analysis

2.2.1 General Flowchart



2.2.2 Flowcharts of sub-programs (or functions)



2.3 Program in 68xxx Assembler

```
          ***************************************************************************
          *                          PRACTICALS ON EID210 BOARD ITSELF              *
          ***************************************************************************
          *      DISPLAY OF REGISTER CONTENT                                        *
          *                                                                         *
          *  Specifications:                                                        *
          *******************                                                       *
          *  - When  starting  program, there is a message transmission             *
          *  - The data register  n° which value must be known, is typed            *
           ( n° between  2 and 7 inclusive)                                         *
          *                                                                         *
          *  FILE NAME:  T_SERIE3.SRC                                               *
          *********************************                                         *
          ***************************************************************************

*         DEFINITION & DECLARATIONS                    *
*****************************************************
* Inclusion of file for specifying the different labels
          include         EID210.def
*********************************************
*         Declaration of variables             *
*********************************************
          section   var

Message   dc.b      ' Numéro du registre de donnée (Data register n°)  de 2 a   2 to 7
Mes_erreur          dc.b      ' Numéro de Registre non valide (Register    t va
Char                ds.w      1                        * For memoris    he fi    ceived character
Num                 ds.b      1                        * For m                 it   to be displayed

          section             code
*****************************************************
*         START OF EXECUTE PROGRAM                     *
*****************************************************
*         INITIALISE
*************************
* The following  initialisations are inhibited, because th   onitor  a     onfigured the serial port !
*  Transmission speed
Début               move.w    #9,SCCR0                 *      ing a speed of  57600 Baud
* Validate transmission & reception
                    move.w    #$002C,SCCR1
* Initialisation of  registers
                    move.w    #$2222
                    move.w    #$33    d3
                    move.w    #$44    ,d4
                    move.w    #$55    d5
                    move.w    #$666
                    move.w    #$7777,d7
************************************
*          MAIN LOOP                  *
************************************
Deb_BP    * Sending of input message
                    move.w    #$0,d1                   * Into d1, the number of transmitted characters
                    move.l              #Message,A1    * Into A1, the address of  message start
* Passing to next line and line jump
                    bsr                 AT_TP          * For waiting if  ready to transmit
                    move.w    #$0D,SCDR                * $0D is the ASCII code of  CR "Enter"
                    bsr                 AT_TP          * For waiting if  ready to transmit
                    move.w    #$0A,SCDR                * $0A is the ASCII code of LF " line jump "
Disp_cont_mes       bsr                 AT_TP          * For waiting if  ready to transmit
                    move.b    (A1),d0
                    move.w    d0,SCDR
                    add.l               #1,d1          * Pass to next character
                    add.l               #1,A1          * Check if message sending ended
                    cmp.l               #43,d1         * There  is 43 characters in the message
                    bne                 Aff_suite_mes
* Passing to next line and line jump
                    bsr                 AT_TP          * For waiting if  ready to transmit
                    move.w    #$0D,SCDR                * $0D is the ASCII code of  CR "Enter"
                    jsr                 AT_TP          * For waiting if  ready to transmit
                    move.w    #$0A,SCDR                * $0A is the ASCII code of LF " line jump "
* Cont. next page
```

```
*  Continuation
* Reception of the register  n°, which value is unknown
************************************************************
* The received character  must be a figure between 0 and 9 (ASCII Code between $30 and $39)
                    bsr             AT_RC           * Wait character reception
                    move.w          SCDR,char       * Recovering of received character
                    move.w          char,d0
                    and.w           #$00FF,d0
                    cmp.w           #$0030,d0       * Check if received character is a figure
                    blt             EM_erreur       * The figure ASCII codes are > to $30
                    cmp.w           #$0039,d0       * Waiting for a figure
                    bgt             EM_erreur       * The figure ASCII codes are < to $39
* Display of binary specified state register
*****************************************************
                    move.w    char,d0
                    and.w           #$000F,d0
                    cmp.w           #$0002,d0
                    bne             test_si_d3      * Go out if it is not d2
                    * Display of d2 state
                    move            d2,d1
                    bsr             AF_ERS          * Toward display of specified state register
                    bra             Deb_BP          * Return to main loop start
Test  si_d3         cmp.w           #$0003,d0       * Check if d3 state requested
                    bne             test_si_d4      * Go out if it is not
                    * Display of d3 state
                    move            d3,d1
                    bsr             AF_ERS          * Tow  displ  o    te register
                    bra             Deb_BP          * Re   t    oop start
Test  si _d4        cmp.w           #$0004,d0       * Ch   f d4
                    bne             test_si_d5      * Go ou  is no
                    * Display of d4 state
                    move            d4,d1
                    bsr             AF_ERS          *  wa  disp  of specified state register
                    bra             Deb_BP          *  tu   ha  oop start
Test  si _d5        cmp.w           #$0005,d0       *  eck   e requested
                    bne             test_si_d6      *  t if  t d5
                    * Display of d5 state
                    move            d5,d1
                    bsr             AF_ERS          *  dis  y of specified state register
                    bra             Deb_BP          *  to main loop start
Test  si_d6         cmp.w           #$0006,d0       *  k if d6 state requested
                    bne             test_si_d7      *  out if it is not d6
                    * Display of d6 state
                    move            d
                    bsr             _E              * Toward display of specified state register
                    bra             b_              * Return to main loop start

Test si_d7          cmp.w                           * Check if d7 state requested
                    bne             EM  reur        * Go out if it is not d7
                    * Display of d
                    move            7,d1
                    bsr               S
                    jmp       Deb_BP                * Return to main loop start
* Register N° not valid ( between  2 and 7 inclusive)  thus sending of error message
EM_erreur           move.w    #$0,d1                * Into d1, the number of transmitted characters
                    move.l          #Mes_erreur,A1  * Into A1, the address of  message start
* Pass to next line and line jump
                    bsr             AT_TP           * For waiting if transmission ready
                    move.w    #$0D,SCDR             * $0D is the ASCII code of CR "enter"
                    bsr             AT_TP           * For waiting if transmission ready
                    move.w    #$0A,SCDR             * $0Ais the ASCII code of  LF "Jump line"
EM_erreur1          bsr             AT_TP           * For waiting if transmission ready
                    move.b    (A1),d0
                    move.w    d0,SCDR
                    add.l           #1,d1           * Pass to next character
                    add.l           #1,A1           * Check if  message sending achieved
                    cmp.l           #31,d1          * There are 31 characters into the message
                    bne             EM_error1

Loop      jmp       Deb_BP
* END of main loop and main program
************************************************************
```

```
*********************************************************
*           Waiting sub-program if ready to transmit
*********************************************************
AT_TP             move.w    SCSR,d0  * Acquire the serial link state register
                  and.w               #TDRE,d0        * Bit  indicating if transmission register  empty
                                                      * Transmit Data Register Empty
                  beq                 AT_TP           * Loop if transmission unready
                  rts                                 * Return from sub-program


*********************************************************
*           Waiting  sub- program if character reception
*********************************************************
AT_RC             move.w    SCSR,d0  * Acquire The serial link state register
                  and.w               #RDRF,d0        * Bit indicating if  reception register full
                                                      * Receive Data Register Full
                  beq                 AT_RC           * Loop if nothing received
                  rts                                 * Return from sub-program


*********************************************************
* Display Sub-program for specified register state
*********************************************************
AF_ERS            * Transmit character 'd'
                  bsr                 AT_TP           * For waiting if  tra
                  move.w    #$64,SCDR                 * $64 is the
                  * Transmit register n°
                  bsr                 AT_TP           * For wa
                  move.w    char,d0
                  move.w    d0,SCDR                   * Into d0 th
                  * Transmit characters '='
                  jsr                 AT_TP           * For
                  move.w    #$3D,SCDR                 * $3D
                  move.b    #16,num                   * d1 in
                  * 15 binary states are transmitted  follow
AF_ERS2 lsl.w               #1,d1
                  bcc                 AF_ERS0
                  * It  is 1 thus, the ASCII code of fig
                  jsr                 AT_TP           * For
                  move.w    #$0031,SCDR               * A
                  bra                 AF_ERS1
AF_ERS0           * It  is 0 thus, the ASCII cod
                  jsr                 AT_T            * waiting if  transmission ready
                  move.w    #$0030,SCDR               * The ASCII code of figure 0 is transmitted
AF_ERS1 * Pass to the next  bit
                  sub.b
                  bne
                  rts                                 * Return from sub-program
 * END of specified register state displa
*********************************************************


***********************************
* END OF SUB-PROGRAMS
***********************************
                  end                                 * End of   Assembler source file
```

PRACTICAL N° 5: WRITING OR READING TO A SPECIFIED ADDRESS

2.4 Topics

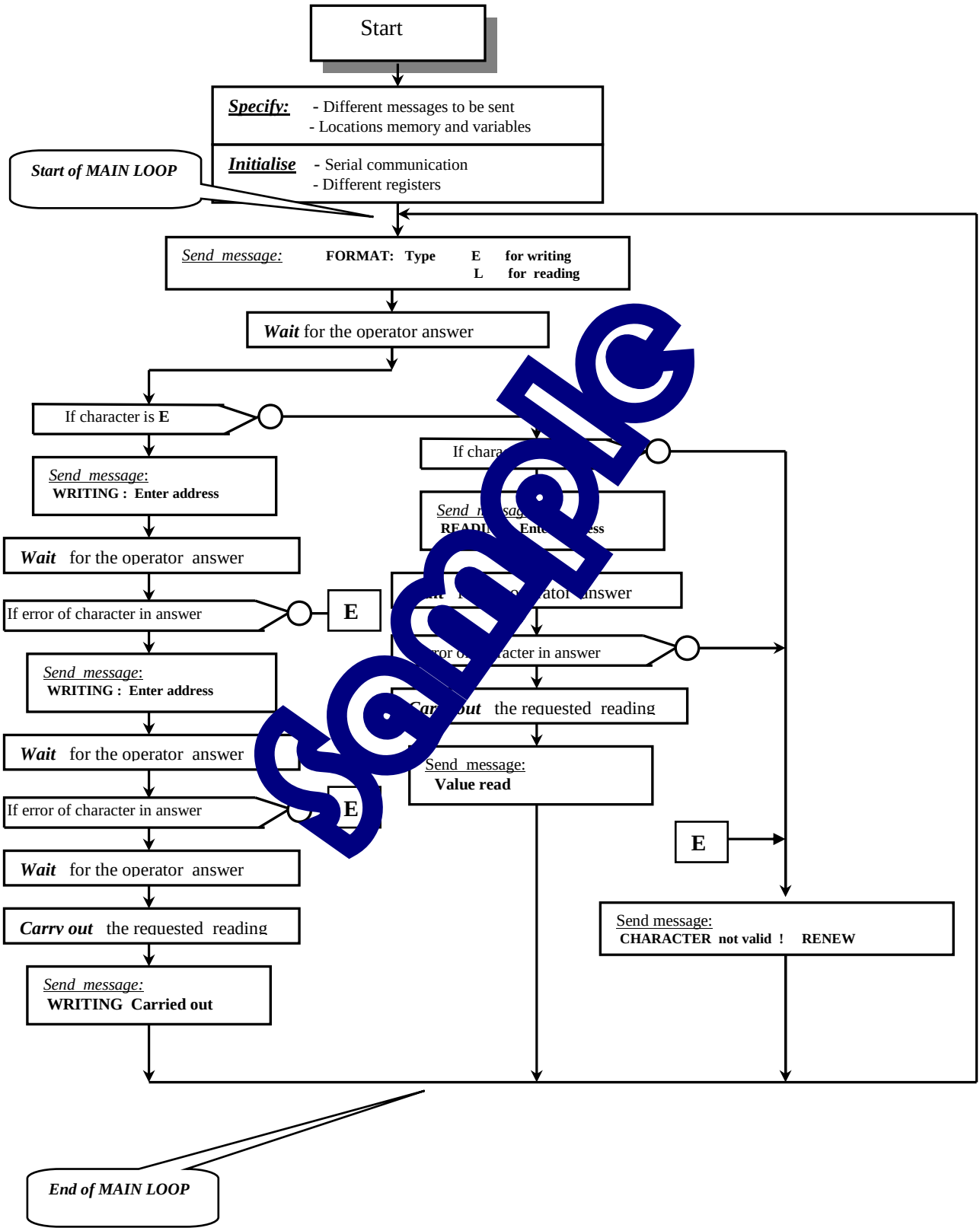
Purposes :	Being capable of configuring and using the RS232 serial communication function (internal to the 68332 Micro-controller), in "Transmission-Reception" mode ("duplex" link). Being capable of acquiring a message (chain of characters) constituting an order, of analysing this message for the detection of possible errors then, executing and answering to this message. Being capable of converting encoded ASCII information into hexadecimal and vice versa.
Specifications :	When starting the program, the user is asked pre-defined message sending (chain of characters) informing about the syntax: "**FORMAT: Type L'adresse for writing** **Type L'adresse for reading** " If the answer is 'E' the address (on 6 digits) is requested, then the data (4 digits). There is checking of received information (ASCII codes of HEXA codes).If an error is detected, there is transmission of an error message: **" CARACTERE NON VALIDE ! RECOMMENCER"** (not valid character! Renew) If there is no error detected, writing is carried out and a message is sent "**ECRITURE effectuée**" (Writing carried out) If the answer is '**L**' , the address on 6 digits is requested . There is checking of received information (ASCII codes of HEXA codes).If an error is detected, there is transmission of an error message: **" CARACTERE NON VALIDE ! RECOMMENCER"** (not valid character! Renew) If there is no error detected, writing is carried out and a message is sent: **"Valeur lue à l'adresse spécifiée**: **xxxx** " (Value read at specified address: xxxx) where xxxx is the word read at the specified address, on HEXA. encoded 16 bits.

Necessary Test Equipment :

PC Micro Computer using Windows ® 95 or later,
68332 16/32 bits micro-controller mother Board, Ref. : EID 100 000
USB connection cable, or if not available an RS232 cable, Ref. : EGD 000 003
AC/AC 8V, 1 A Power Supply, Ref. : EGD000001,

Duration : 4 hours

2.5 Analysis



2.6 Program in 68xxx Assembler

```
          ************************************************************************
          *                          PRACTICALS ON EID210 BOARD ITSELF          *
          ************************************************************************
          *      WRITING OR READING AT A MEMORY ADDRESS                           *
          *  Specifications:                                                     *
          ******************                                                     *
          *  - When starting the program, there is message sending               *
          *  - Format of a writing request at a specified address :              *
          *          Eaaaaaa=??      where  aaaaaa -> Hexadecimal address         *
          *                          ?? -> value to be written in Hexadecimal     *
          *  - Format of a writing request at a specified address :              *
          *          Laaaaaa         where  aaaaaa -> Hexadecimal address         *
          *  - The answer is:     (aaaaaa) = ??                                  *
          *                                                                      *
          *  FILE NAME:  T_SERIE4.SRC                                            *
          *********************                                                  *
          ************************************************************************
**********************************************
*         DEFINITION & DECLARATIONS          *
**********************************************
* File inclusion defining  the different  labels
          include          EID210.def

************************************
*         Declaration of  variables        *
************************************
          section     var
Mes_entree1          dc.b      ' FORMAT:   Type  [illegible] f[illegible]riti[illegible]                '
Mes_entree2          dc.b      '                 [illegible]di[illegible]                              '
Mes_entree3          dc.b      ' Caution, if writing in [illegible]M, [illegible]ions "var" and "code" '
Mes_rep_Lec          dc.b      ' READING:  E[illegible]ress ([illegible] HEXA characters)        '
Mes_rep_Ecr1         dc.b      ' WRITING: E[illegible]r one [illegible](on [illegible]XA characters )       '
Mes_rep_Ecr2         dc.b      ' WRITING: [illegible]te[illegible] da[illegible]XA characters)      '
Mes_rep_Ecr3         dc.b      ' WRITIN[illegible]                '
Mes_Val_Lue          dc.b      '  Value r[illegible] at [illegible] spe[illegible]address:         '
Mes_erreur           dc.b      ' CARAC[illegible]ER[illegible] ! RECOMMENCER          '
Char                 ds.w      1         [illegible]r memorising the 1st received character
Num                  ds.b      1         [illegible]or memorising the received character number
Nombre               ds.b      1[illegible]  For the number of characters to be displayed
AD_ASCII             ds.b[illegible]       * For memorising the address in ASCII
AD_HEXA              d[illegible]          * For memorising the address in HEXA
DATA_lue             [illegible]w  1[illegible]
DATA_HEXA            [illegible]w          * For data in HEXA
DATA_ASCII           d[illegible]  4       * For memorising the data in ASCII
          section      co[illegible]
*****************************************************
*         START OF EXECUTE PRO[illegible]                 *
*****************************************************
*         INITIALISE
************************
* The following initialisations are inhibited, because the serial port is  already configured by the monitor!
*  Transmission speed
Début          move.w    #9,SCCR0               * For  a speed of : 57600 Baud
* Validate Transmission and Reception
               move.w    #$002C,SCCR1

************************************
*         MAIN LOOP          *
************************************
Deb_BP   * Sending of input message
               move.l          #Mes_entree1,A1  * Into register A1, the address  message
               move.b    #55,nombre             * Number of characters to be displayed
               jsr       Env_Mes
               move.l          #Mes_entree2,A1
               move.b    #55,nombre             * Number of characters to be displayed
               jsr       Env_Mes
               move.l          #Mes_entree3,A1
               move.b    #70,nombre             * Number of characters to be displayed
               jsr       Env_Mes
```

```
* CONTINUATION of the program
*     Reception of the order
**********************
* The 1st received character must be  E (ASCII Code $45 )  or L  (ASCII Code $4C )
Test_RC  bsr                AT_RC                   * Waiting for character  reception
                move.w      SCDR,char               * The received character is recovered
                move.w      char,d0
                and.w               #$0045,d0
                cmp.w               #$0045,d0       * Check if the received character is  E
                bne                 Test_crL        * Check if it is L
* Writing for a specified data  to a  specified  address
*********************************************************
                move.l              #Mes_rep_Ecr1,A1
                move.b      #55,nombre              * Number of characters to be displayed
                jsr         Env_Mes                 * Send  message
* Waiting for the address on  6 hexadecimal characters
                jsr                 ATT_AD          * Toward sub-program of address reception
                cmp.w               #0,d0
                beq                 Test_RC_E       * Address reception with  error
                jsr         AT_TP
                move.w      #$20,SCDR               * $20 is the ASCII code for  "SPACING"
                move.l              #AD_ASCII,A1
                move.b      #6,nombre               * Number of characters to be displayed
                jsr         Env_Mes                 * Send  message ( address sending)
                move.b      #55,nombre              * Number of characters to be displayed
                move.l              #Mes_rep_Ecr2,A1
                jsr         Env_Mes                 * Send  message
* Waiting for the data on 4 hexadecimal characters (one "word")
                jsr                 ATT_DATA
                cmp.w               #0,d0
                beq                 Test_RC_E       * Data reception with error
                jsr         AT_TP
                move.w      #$20,SCDR               * $20 is the ASCII code for "SPACING"
                move.l              #DATA_ASCII,A1
                move.b      #4,nombre               * Number of characters to be displayed
                jsr         Env_Mes                 * Send message
* Address & data are corrects, thus carry out writing
                move.l              AD_HEXA,d0
                lsl.l               #4,d0
                lsr.l               #4,d0           * Loading address in hexadecimal
                move.l      d0,A0
                move.w      DATA_HEXA,d0            * Loading data
                move.w      d0,(A0)                 * Carrying out the reading
                * Sending  "Writing carried out"
                move.l              #Mes_rep_Ecr3,
                move.b      #30,nombre              * Number of characters to be displayed
                jsr         Env_Mes                 * Send  message
                * Jump line two times
                jsr         AT_TP
                move.w      #$0A,SCDR               * $0A is the ASCII code for LF "Jump line"
                jsr         AT_TP
                move.w      #$0A,SCDR               * $0A is the ASCII code for LF "Jump line"
* End of order :WRITING TO A SPECIFIED ADDRESS
                bra         Deb_BP                  * Loop when writing carried out
*************************************************************************************************
Test_crL  move.w    char,d0
                and.w               #$004C,d0
                cmp.w               #$004C,d0       * Check if the received character is L
                bne                 Test_RC_E       *  Send error message
* Reading to a specified address
*******************************************
                move.l              #Mes_rep_Lec,A1
                move.b      #60,nombre              * Number of characters to be displayed
                jsr         Env_Mes
* Waiting for the address on 6 hexadecimal characters
                jsr                 ATT_AD
                cmp.w               #0,d0
                beq                 Test_RC_E       * Address reception with  error
                jsr         AT_TP
                move.w      #$20,SCDR               * $20 is the ASCII code for "SPACING"
                move.l              #AD_ASCII,A1
                move.b      #6,nombre               * Number of characters to be displayed
                jsr         Env_Mes                 * Send  message ( address display)
```

```
* Continuation…
* Reading at  specified address
                    move.l          AD_HEXA,d0
                    lsl.l           #4,d0
                    lsr.l           #4,d0           * For  erasing MSB byte
                  move.l            d0,A0           * Load the address in hexadecimal
                    move.w  (A0),d0
                    move.w  d0,DATA_lue             * Read at the specified address
* Display result message
                    jsr             TRAD_ASCII      * For translating into ASCII 4 characters
                    move.b  #44,nombre              * Number of characters to be displayed
                    jsr             AT_TP
                    move.w  #$20,SCDR               * $20 is the ASCII code for  "SPACING"
                    move.l          #Mes_Val_Lue,A1
                    jsr             Env_Mes         * Send  message ("Value read at the address")
                    jsr             AT_TP
                    move.w  #$20,SCDR               * $20 is the ASCII code for  "SPACING"
                    move.b  #4,nombre               * Number of characters to be displayed
                    move.l          #DATA_ASCII,A1
                    jsr             Env_Mes         * Send message (read value)
* End of reading at a specified address
                    * Jump line twice
                    jsr             AT_TP
                    move.w  #$0A,SCDR               * $0Ais the ASCII code for LF "Jump line"
                    jsr             AT_TP
                    move.w  #$0A,SCDR               * $0Ais the ASC  ode fo  "Jump line"
                    bra             Deb_BP          * Return to ma  o

* The character is wrong, renew the order reception
*****************************************************
Test_RC_E           jsr             AT_TP
                    move.w  #$0A,SCDR               * $0Ai  I co  r LF "Jump line"
                    move.l          #Mes_erreur,A1
                    move.b  #60,nombre              * N  mb  f ch  acters to be displayed
                    jsr             Env_Mes
                    * Jump line twice
                    jsr             AT_TP
                    move.w  #$0A,SCDR               *  $  ASC  ode for LF "Jump line"
Loop        bra     Deb_BP                          * Retu  loop starting

* END of main loop and main program
*****************************************  ***  *

*****************************************  ***
*           WAITING SUB- PROGRAM  if  nsmissi
*****************************************  *******
AT_TP               move.w  S  d0                   * Acquire the serial link  state register
                    and.w           #  RE,d0        * Bit  indicating  if transmission register is empty
                                                    * Transmit Data Register Empty
                    beq             AT_             * Loop when transmission unready
                    rts                             * Return from sub- program

*****************************************  *******
*           WAITING SUB-PROGRAM if  reception of character
*****************************************************
AT_RC               move.w          SCSR,d0  * Acquire  serial link  state register
                    and.w           #RDRF,d0        * Bit  indicating  if reception register is full
                                                    * Receive Data Register Full
                    beq             AT_RC           * Loop if nothing received
                    rts                             * Return from sub-program
 * END of specified register state display sub-program
*************************************************************
* Continued next page
```

```
* CONTINUED …
************************************************
* SUB-PROGRAM of a message sending with 'jump line' and 'enter'
************************************************
Env_Mes  move.b   #$0,d1                          * Into d1, the number of transmitted characters
Aff_suite_mes     bsr          AT_TP              * For  waiting  if transmission ready
                  move.b   (A1),d0
                  move.w   d0,SCDR
                  add.l        #1,d1              * Go to next character
                  add.l        #1,A1              * Check if message sending achieved
                  cmp.b        Nombre,d1          * There are characters in the message
                  bne          Aff_suite_mes
          * Go to next line  and jump line
                  bsr          AT_TP              * Waiting  for  transmission ready
                  move.w   #$0D,SCDR              * $0D is the ASCII code for CR "Enter"
                  bsr          AT_TP              * For waiting if transmission ready
                  move.w   #$0A,SCDR              * $0A is the ASCII code for LF "Jump line"
                  rts                       * Return from sub-program
*         End of  sub-program
*****************************************************************************************
*  SUB-PROGRAM of address reception on 6 ASCII characters
*   and address make up in HEXA (3 bytes)
*****************************************************************************************
ATT_AD  move.b    #0,Num    * Num character = 0
                  move.l       #AD_ASCII,A1
                  move.l       #0,AD_HEXA
ATT_AD1           jsr          AT_RC              * Waiting for cha
                  move.w   SCDR,char          * Recovering of rece
                  move.w   char,d0
                  jsr          Test_CH    * Check if  Hexade
                  cmp.w        #0,d0              * Value sent
                  beq          ATT_AD_err         * Return wi
                  move.b   d0,(A1)            * memorise the re
                  move.b   #5,d3              * restore the add
                  sub.b        num,d3
                  and.l        #$00000F,d3        * Into
                  lsl.l        #2,d3              *
                  lsl.l        d3,d1              carry
                  or.l         d1,AD_HEXA
                  add.l        #1,A1              *
                  add.b        #1,Num
                  cmp.b        #6,Num     * C
                  bne          ATT_AD1
                  rts
ATT_AD_err        rts                 * Ret
*  END of the sub-program
*****************************************************************************************
*  SUB- PROGRAM of the data rec
*  and data composition in HEXA (
*****************************************************************************************
ATT_DATA          move.b   #0,Num
                  move.l       #
                  move.l       #0,
ATT_DATA1         jsr          AT_R               * Waiting for character reception
                  move.w   SCDR,char          * recovering of the received character
                  move.w   char,d0
                  jsr          Test_CH    * Check if  Hexadecimal character
                  cmp.w        #0,d0
                  beq          ATT_DATA_err       * Return with error
                  move.b   d0,(A1)            * memorise the HEXA received character in ASCII
                  move.b   #3,d3              * recovering of the data in HEXA
                  sub.b        num,d3
                  and.w        #$000F,d3
                  lsl.w        #2,d3
                  lsl.w        d3,d1              * carry out shifts
                  or.w         d1,DATA_HEXA
                  add.l        #1,A1              * go to next character
                  add.b        #1,Num
                  cmp.b        #4,Num     * Check if the 6 address characters are acquired s
                  bne          ATT_DATA1
                  rts
ATT_DATA_err      rts                 * Return with  error (do=0)
*  Fin du sous programme
* SUITE page suivante
```

```
* Continued…..


*********************************************************************
*  SUB- PROGRAM of check if  Hexadecimal character  in ASCII and translation into HEXA
*********************************************************************
Test_CH  and.w          #$00FF,d0
         cmp.w          #$0030,d0 * Check if the received character is a figure
         blt            Test_CH_err       * The ASCII codes of  HEXA are > to $30
         cmp.w          #$0039,d0
         bgt            Test_CH1* The ASCII codes of figures are < to  $39
         move.w  d0,d1             * Into d0 the ASCI code of the HEXA character
         andi.w         #$000F,d1* Into d1 the HEXA code from 0 to 9
         rts                       * Return  if  it  is OK
Test_CH1 cmp.w          #$0041,d0 * Check if the received character is an HEXA letter
         blt            Test_CH_err       * ASCII codes of  HEXA letters are > to $41
         cmp.w          #$0046,d0
         bgt            Test_CH_err       * ASCII codes of  HEXA letters are< to $46
         move.w  d0,d1             * Into d0 the ASCI code of the HEXA character
         andi.w         #$000F,d1*
         add.w          #9,d1             * Into d1 the HEXA code from 0 to 9
         rts                       * Return  if  it  is OK
Test_CH_err      move.w  #0000,d0  * Return with  0 if error
         rts
*  END of the sub-program
****************************


*******************************************************
*   Sub- program of data translation from HEXA into ASCII
*******************************************************
TRADE_ASCII      move.b  #0,Num    * Num character = 0
         move.l          #DATA_ASCII,A1
TRADE_ASCII1     move.w  DATA_lue,d0       * Read valu[illegible] is [illegible]
         move.b  #3,d3             * For isolati[illegible] th[illegible]te[illegible] be converted
         sub.b           num,d3    * Calculate [illegible]shi[illegible]
         and.w           #$000F,d3
         lsl.w           #2,d3
         lsr.w           d3,d0                Carr[illegible]ifts
         and.w           #$000F,d0*[illegible]date [illegible]
         * Convert into ASCII
         cmp.b           #9,d0
         bgt             TRAD_Le[illegible]> than 9, it is a letter (A to F)
         * It is a figure (0 to 9)
         or.b            #[illegible],d0   *[illegible]II code of figures are from $30 to $39
         bra             [illegible]AR[illegible]CII2
TRAD_Lettre      add.b   [illegible]5,d[illegible]         * The ASCII code of HEXA letters are from  $41 to $46 (65 to70)
TRAD_ASCII2      move.b  (A1[illegible]
         add.l
         add.b           #1,[illegible]n
         cmp.b           #4,N[illegible]   * Check if the 4 data characters are translated
         bne             RA[illegible]ASCII1       * Carry on if it is not ended
         rts                       * return if it is ended
*  END of  sub-program
*********************************


***********************************
* END OF SUB-PROGRAMS
***********************************


         end       * End of Assembler source file
```