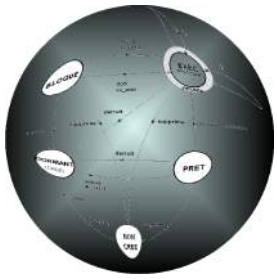
PRACTICAL WORKS MANUAL

CAN SYSTEM- V.M.D.

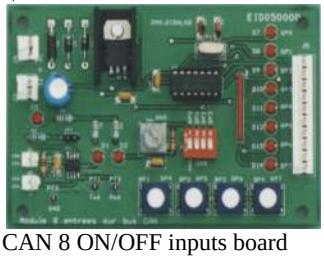
Multiplexed Didacti Vehicle
with Real Time E e



ard around 68332

CAN controller board (PC104)

CAN netwo



CAN 8 ON/OFF inputs board

CAN 4 power outputs board

CAN Servo-system board

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


Sampic

SUMMARY


Sampic

1 INTRODUCTION

1.1 Software Organization

1.1.1 Definition files

The provided programs need the following definition files:

→ "Structures_Donnees.h " defines all the useful data structures for a structural programming,

→ "Mtr86.h" contains the executive multitasking primitives' prototypes as well as the constants definitions. It must be put on top of all source files.

→ "EID210.h" defines the access address to the different material elements implanted on the processor board,

→ "vmd.h" defines the VMD application elements.

Ensure that all files are present, in the work directory, which is located in the directory containing the EID210 assembler editor.
These files are specific in the real-time executive of the EID210 board, and must remain in the work directory.

1.2 Specific functions

Specific functions accede to the CAN network via the "ATON SYSTEMES" CAN-PC104 board.
The prototypes of these functions are defined in the Mtr_CAN.h file and their definition in the Mtr_CAN.C file.
If it is not already, this file must be compiled and its Mtr_CAN.o compilation result should be included as the "Linker" file with the application files.

> ⚠ **Ensure that the** Mtr_CAN.o **file is present in the work directory which is located in the directory containing the EID210 editor-assembler.**
>
> **V Also ensure that this** Mtr_CAN.o **file is included in "linker** "..
> **Pour ce faire :**
> **In the EID210 software** *Configuration* **then** *GNU C/C++* **then** *Linker* **then** *Add*
> **then find and select the** Mtr_CAN.o **file which will be added to already presented files i.e. CTR0.o and EID210.o**

→ `OpenCAN(SJA1000 board version)` Function ;
This function initializes the SJA1000integrated circuit, th[illegible]ES CAN-PC104 board core.
The transmitted parameter is the version of your ATON-S[illegible]oard: `ATON_V1`(equivalent to aton_can.o) `or ATON_V2` (equivalent to aton_can2.[illegible]
This function is called in the "Initialization" part of th[illegible]program

→ `EcrireTrame(&Nom_Trame)` Function
This function allows sending a frame on the ne[illegible]
The last variable is a pointer on the structured [illegible] of "Frame" model (model defined in the previous chapter).
This function doesn't return any parameter.

→ `LireTrame(&Nom_Trame_re[illegible]`
This function can know whether a f[illegible]eceived from the CAN network and if yes, accede to the network. The last variabl[illegible]structured variable in the form of "Frame" model (model defined in the previ[illegible]
Warning this function is loc[illegible]:
If no message is received, the [illegible]falls [illegible]leep.
When receiving a message, the ta[illegible]s up and returns 1.

→ `AfficheTrame(&Nom_Trame)` Function
This function can display a frame on the computer screen connected to the CPU.
The last variable is a pointer on the structured address in the form of "Frame" model (model defined in the previous chapter).
This function doesn't return any parameter.

→ `MONITOR and ENDM` Functions
These functions allow confining a writing area on the monitor. They must be used in during printf or AfficheTrame function; they allow the reserve for a writing and avoid that the monitor become a critical resource.

2 EX N°1 : SWITCH THE LIGHTS FLASH OF THE OPTICAL BLOCK

2.1 Topic :

Purposes :	- Understand and use the proposed and specific data structures - Understand and use the proposed and specific functions. - Define and send a data frame to a CAN module recipient, which is accessible to a given address. - Test whether a frame has been received or not. - Display received and sent frames on the screen.
Specifications :	At first all the lamps of "Right Back lights" block are off. We would like to realize a function " chase " with 4 lamps block (after a regular time interval, we tu[illegible]f the previous lamp and then turn on the next one). → The sent or received [illegible]s through the CAN circuit are displayed. → The delay is produced by the "so[illegible]ware" (count the number [illegible]e main loop) After a minimum mo[illegible]fic[illegible]ns the program should allow to realize the same function with the [illegible]ock "Left Front lights " and then " Right Front lights" and fin[illegible]y "Le[illegible]t li[illegible]s ". Three tasks will be re[illegible]ed. The first de[illegible]s w[illegible]chase, The se[illegible]s t[illegible]frame to the destination of selected optical block, The [illegible]rd read[illegible]displays the received frame.

Necessary hardware and software :

PC Micro Computer using Windows ® 95 or later
Editor Software-Assembler-Debugger
If programming in C, GNU; C / C + + Compiler Ref: EID210101
Processor board 16/32 bit 68332 microcontroller and its software environment
(Editor-Cross Assembler-Debugger) Ref: EID210001
CAN PC/104 Network board in ATON SYSTEMS Ref NIC: EID004001
CAN network with four power outputs modules for lights Ref: EID051001
USB connection cable, or if not available RS232 cable, Ref: EGD000003
AC / AC Power source 8V 1A Ref: EGD000001
12V Power source supply for the CAN modules ("energy" network)

Time : 3 hours

2.2 Solution elements

2.2.1 Analysis

Remind

Chase for the « Right Back Lights »

If we want to define the status of 4 power outputs module, the frame of type "Input Message"(IM) has to be sent back with a "Write Register" function on its register which is accessible to the "GPLAT" output port (from the technical manual of MCP25050 circuit page 22).
Indeed, seen from the module, it receives a command frame requesting it to change the status of its output register which imposes the condition of different outputs of the interface MCP25050 circuit.

For an IM "Input Message", the included register is the RXF1 which leads to the identifier base 0F 88 xx xx for the right front lights (see identifier table in Chapter 1). The 3 least significant bits must be turned to 0 (from the technical manual of MCP25050 circuit page 22) and also let the other indifferent bits turn to 0, which ultimately leads to the identifier 0F880000 (only 29 useful bits), with the label defined in the "T_Ident_IM_FVD" of "CAN_VMD.h" file.
Definition of the command frame (find in Chapter 1 of this document the definition of the different fields of the frame for this example).

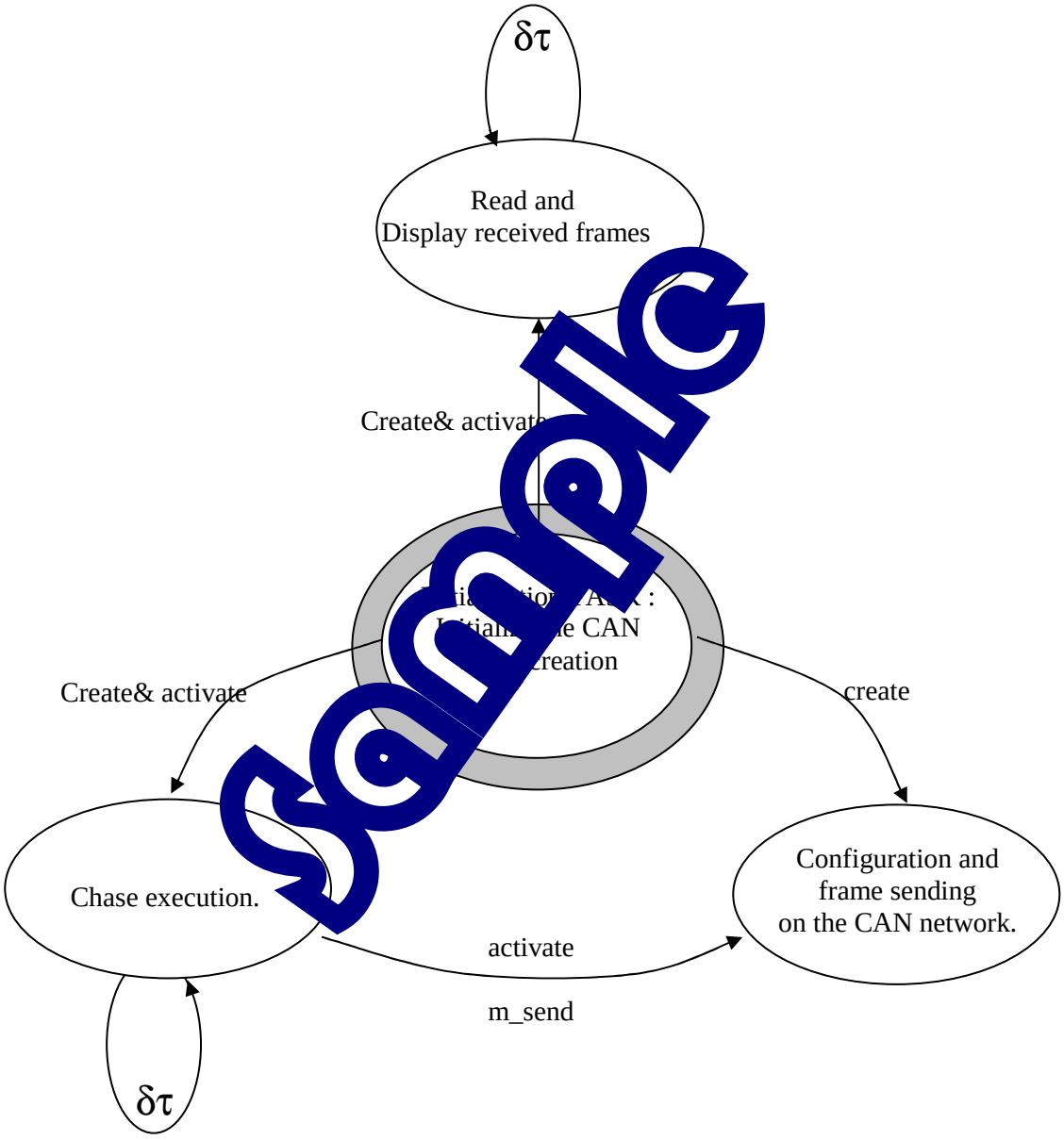
Remarks

- In the command frame, there are three parameters defined in the "data" area:
  - → Parameter "address" (in the rank 0 of "data") defines the concerned address register by writing.
    In the case before, it is GPLAT and address is 1E h (Page 15 MCP25050 manual).
    02h+shift (note1) = 02h +1Ch = 1Eh
  - → Parameter "mask" (in the rank 1 of "data") permits to restore some unchanged bits of the register if they are not to be affected by the writing operation.
    In our case the power outputs are controlled by the 4 least significant bits of the port. The 4 first bits should be masked with the mask value: 0Fh.
  - → Parameter "value" (in the rank 2 of "data") allows to define the status of the unmasked outputs.
    In our case, to achieve the chase function, it will give the successive values: 00, then 01, then 02, then 04 and finally 08.
- After an IM "Input Message" the message has been received by the recipient, it should return an acknowledgment frame module whose identifier is defined by TXID1. In the case of "Right back Lights" module, the identifier of the acknowledgment message will be 0E A0 00 00 (from table in Chapter 1).

2.2.2 Tasks management and organization

Three tasks are going to be realized. And we need to add an initialization task.
1- The initialization task will initialize the CAN board and create the following tasks.
2- The chase task will change a variable to turn on the optical block. It will activate the task which can turn on the optical block. Then it will fall asleep to let other tasks run.
3- The EcrireFeux task will send the frame on the CAN network which can turn on or turn off the operative part.
4- The LectureTrame task will be permanently active. It will wait for a frame reception in order to display on the screen.

2.2.3 State diagram



2.2.4 "C" Program

```
/*****************************************************************************************
**
       *      Experiment on EID210 / CAN Network - V.M.D (Multiplexed Didactic Vehicle)

       ****************************************************************************************
**********
       *      EXPERIMENT n 1: SWITCH THE LIGHTS FLASH OF THE OPTICAL PART

       *--------------------------------------------------------------------------------------
--------
       *   SPECIFICATIONS :

       *  ********************

       *   We want that after a regular time interval, we turn off
       *   the previously activated output to turn on the following(chase function)

       *--------------------------------------------------------------------------------------
--------

*****************************************************************************************
**/
// File to include
//*********************
#include <stdio.h>
#include "mtr86.h"
#include "Structures_donnees.h"
#include "vmd.h"
#include "eid210.h"

//Optical block selection
#define Feux_Selec Ident_T_IM_FRD

//Tasks prototype
TACHE  init  (void);
TACHE  LectureTrame(void);
TACHE   chenillard(void);
TACHE  EcrireFeux();

/*****************************************************************/
/*                          Initialization                       */
/*****************************************************************/
TACHE init(void)
{
OpenCAN(ATON_V2);     //ATON V1 for old CAN boards
                      //ATON_V2 for new ATON CAN boards

active(cree(chenillard,1,1024));  //The chase management is the most important task
cree(EcrireFeux,2,1024);          //Writing on the optical blocks is to control after the
chase
active(cree(LectureTrame,3,1024));//the display task has the lowest priority
}

/*****************************************************************/
/*                          The chase management task            */
/*****************************************************************/
TACHE chenillard(void)
{
//the variables that we pass as parameter are necessarily defined on static
static p_s=0x00;              //Value to write on the optical block
while(1)                      //The task runs continuously
       {
       switch(p_s)
                      {
                      case 0 : p_s=0x01;
                               break;
                      case 1 : p_s=0x02;
                               break;
                      case 2 : p_s=0x04;
                               break;
                      case 4 : p_s=0x08;
                               break;
                      case 8 : p_s=0;
                               break;
```

```
            }
       m_send(num_tache(EcrireFeux),&p_s);  //Activation of the writing task on the CAN network
                                            //Passing the address of the value
       dort(200);                           //the task sleeps for 2 seconds
       }
}

/*********************************************************************/
/*            Frames sending control task on the CAN network                 */
/*It will be called by the chase task                                        */
/*It will run when the chase task is asleep                        */
/*********************************************************************/
TACHE  EcrireFeux(e_s)
char e_s;      //Recovery of sent parameter
{
Trame feux;
//frame configuration
feux.trame_info.registre=0x00;
feux.trame_info.champ.extend=1;
feux.trame_info.champ.dlc=0x03;
feux.ident.extend.identificateur.ident=Feux_Selec;
feux.data[0]=0x1e;
feux.data[1]=0x0f;
feux.data[2]=e_s;
//send the frame on the CAN network
EcrireTrame(&feux);
}

/*********************************************************************/
/*                   Frames reception and display task                        */
/*This task is created and activated by the initialization                   */
/*It will run when the chase task is asleep                                  */
/*and the Ecrire feux task is finished                                       */
/*********************************************************************/
TACHE LectureTrame(void)
{
Trame tr_rx;   //definition of a frame
while(1)              //task activates perma
       {
       LireTrame(&tr_rx);    //This functi  m  th    leep when there are no  received
frames
       MONITOR               //Monitor  se  tio  r the display (critical resource)
       AfficheTrame(&tr_rx); //Display  e  frame
       ENDM                        //Re  se    monitor for other task
       }
}
/*********************************************************************/
/*                                                                   */
/*********************************************************************/

main()
{
clsscr();
printf("CAN bus EX_1 with mtr86
//Launch of the real-time executive
start_mtr(init,512);
}
```


Sample

3 EX N°2 : ACQUIRE THE STATE OF LIGHTS STALK

3.1 Topic :

Purposes :	- Define and then send a remote frame to an input module, which is accessible to a defined address. - Testing whether a frame has been received. - Extract the expected information from a response frame. - Display received and sent frames on the screen. - Display the expected data on the screen.
Specifications :	After a regular time interval, we interrogate the 8 inputs' module on which is connected the lights stalk so that we can know its condition. → The sent or received frames by t… circuit are displayed. → The delay is produced b… "…" (count the number of passa… in t… main loop) → The different comman… impos… the position of the lights stalk will be displayed individually.

Nece… …ware and software :

PC Micro Computer using W…ows … or later
Editor Software-Assembler-D… …er
If programming in C, GNU; C / C++ C…mpiler Ref: EID210101
Processor board 16/32 bit 68332 mic… controller and its software environment
(Editor-Cross Assembler-Debugger) Ref: EID210001
CAN PC/104Network board in ATON SYSTEMS Ref NIC: EID004001
CAN network with four power outputs modules for lights Ref: EID051001
USB connection cable, or if not available RS232 cable, Ref: EGD000003
AC / AC Power source 8V 1A Ref: EGD000001
12V Power source supply for the CAN modules ("energy" network)

Time : 3 hours

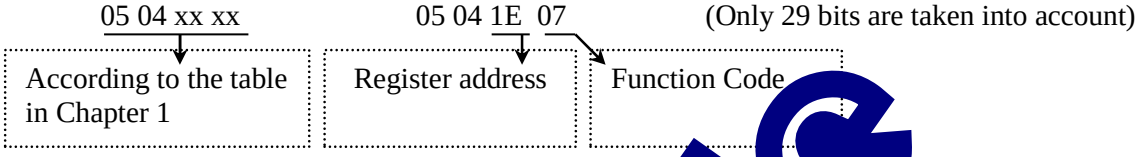
3.2 Solution elements

3.2.1 Analysis

When the semaphore is released, we interrogate the 8 inputs' module on which is connected to the lights stalk

Remind

According to the table given on page 22 of the MCP25050 manual, the identifier itself will contain the address of the read register. This address is on the identifier bits ID15 to ID8 in extended mode (bits that are received and put in the RXBEID8 register). The concerned registry is GPPIN with 1Eh address "(see in MPC25025 technical documentation page 37) ..
On the other hand, the least significant 3 bits of the identifier in extended mode must be set to 1.
The identifier defined in Chapter 1 should be completed as follows:



→ Definition of structured variables under "**Trame**" module :

```
Trame T_IRM_Commodo_Feux;
```

→ Access and definition of the different elements of the structured variable "T_IRM_Commodo"

```
T_IRM_Commodo.trame_info.register=0x00;  //All bits to be initialized to 0
T_IRM_Commodo.trame_info.champ.extend=1;  //Working in extended mode
T_IRM_Commodo.trame_info.champ.rtr=0x01;  //Frame type
T_IRM_Commodo.trame_info.champ.dlc=0x01;  //There will be 1 data byte
T_IRM_Commodo_ad.ident.extend.identifier=0x05041E07;
  //! It has 29 bits.   To set the address of lights stalk
```

Definition of the remote frame which was received in response

According to the definition of identifiers given in Chapter 1, a frame with response of IRM has the same identifier as the original remote frame.
Seen from the module (the MCP2510) the response to an IRM (Information Request Message) is an OM (Output Message).
The difference with the original frame is that this response frame contains the parameter "value" (rank 0 of the part of "data" in the frame). This parameter is the image of the input port. Thus we recover the status of various buttons.

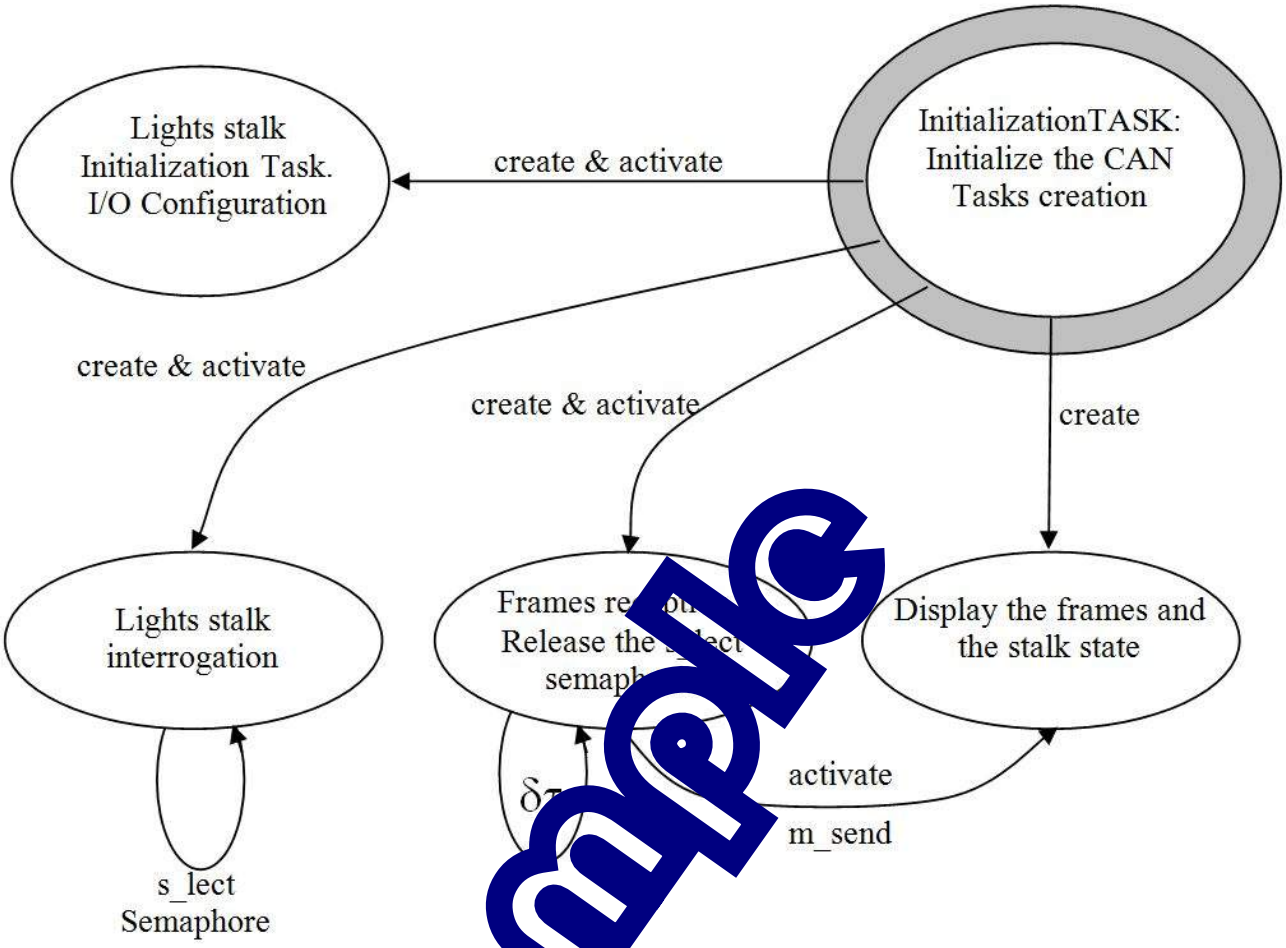
Access to the different binary status commands

The parameter "value" of the response frame, recovered from the data in rank 0 is a byte inputs image. The different bits of this byte are extracted individually because of a data structure defined in the CAN_VMD file.

3.2.2 Tasks management and organization

1- The initialization task will initialize the CAN board and create the following tasks.
2- The lights stalk initialization task; configuration of GP0 to GP7 as inputs.
3- The lights stalk interrogation task will be executed when the semaphore is available.
4- The task manager frame reception. Release the semaphore and active display task.
5- The display task. It displays the received frame. If there is a stalk response, its state will be displayed.

3.2.3 Flowchart



3.2.4 "C" Program

```
/*Light stalk reading*/
//*******************
// File to include
#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"

//Semaphore declaration
SEMAPHORE  s_lect;

//Tasks prototype
TACHE  init  (void);
TACHE  LectureTrame(void);
TACHE  init_commodo(void);
TACHE  lecture_commodo(void);
TACHE  aff();

/*************************************************************************/
/*                          Initialization task                          */
/*************************************************************************/
TACHE init(void)
{
OpenCAN(ATON_V2);     //ATON_V1 for old ATONCAN boards
                      //ATON_V2 for new ATONCAN board
s_lect=s_cree(0);     //Semaphore creation
active(cree(init_commodo,1,512));    //first priori  to  e th  I/Os configuration from
the begining
active(cree(LectureTrame,2,1024));
active(cree(lecture_commodo,3,1024));
cree(aff,4,1024);
}

/*************************************************************************/
/*                  Lights stalk initializ                               */
/*************************************************************************/
TACHE init_commodo(void)
{
Trame t_gpdr;
t_gpdr.trame_info.registre=0x00;
t_gpdr.trame_info.champ.extend=1; // Work  the xtended mode
t_gpdr.trame_info.champ.dlc=0x03; // the  data of 8 bits (3 sent bytes)
t_gpdr.ident.extend.identificateu ident=Id  _IM_Commodo_Feux;
t_gpdr.data[0]=0x1F;  // fir  da  dr  s" of concerned register
                      // G  ve a direction of I/O) address = 1Fh  page 16
t_gpdr.data[1]=0x7F;  // s con da  -> "  sk"
                      //  bits are concerned except GP0 (see doc page 16)
t_gpdr.data[2]=0x7F;  // t  d data-  "Value" -> all the bits are inputs
// Send frame to define the di  ion of the inputs and outputs
EcrireTrame(&t_gpdr);
}

/*************************************************************************/
/*                    lights stalk interrogation task                    */
/*************************************************************************/
TACHE lecture_commodo(void)
{
Trame T_IRM_Commodo_Feux;
//Frame initialization
T_IRM_Commodo_Feux.trame_info.registre=0x00;
T_IRM_Commodo_Feux.trame_info.champ.extend=1;
T_IRM_Commodo_Feux.trame_info.champ.dlc=0x01;
T_IRM_Commodo_Feux.trame_info.champ.rtr=1;
T_IRM_Commodo_Feux.ident.extend.identificateur.ident=Ident_T_IRM_Commodo_Feux;
while(1)                                              //the task runs permanently
       {
       s_wait(s_lect,0);                              //Wait until the semaphore is
released
                                                      //and make the task sleep (parameter
0)
       EcrireTrame(&T_IRM_Commodo_Feux);          //to send an IRM to lights stalk
       }
}
```

```
/*********************************************************************/
/*                    Reading received frames task                   */
/*********************************************************************/
TACHE LectureTrame(void)
{
static Trame tr_rx;
int compteur=0;

while(1)
        {
        LireTrame(&tr_rx);    //make the task sleep until reception of a frame
        if (compteur++==10)
                {
                MONITOR
                printf("Display\n");
                ENDM
                m_send(num_tache(aff),&tr_rx);
                compteur=0;
                }
        if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_Commodo_Feux)
                {
                dort(20);             //the task falls aslepp for 0,2 sescond
                s_signal(s_lect);     //Release the semaphore to autorize the IRM sending in the
Lecture_Commodo task
                }
        }
}

/*********************************************************************/
/*                              Display task                         */
/*********************************************************************/
TACHE aff(p_t)
Trame p_t;
{
MONITOR
gotoxy(2,2);
AfficheTrame(&p_t);
if (p_t.ident.extend.identificateur.ident==Ident_T_IRM_Commodo_Feux)
        { // We received stalk state and the display the states
        Etat_Commodo_Feux.valeur=~p_t.data[0];
        gotoxy(4,16);
        printf("State of different inputs (image) of the stalk:\n");
        printf("    Recovered and complemental bytes (in Hexa)
=%2.2x\n",Etat_Commodo_Feux.valeur);
        printf("    Side light= %d    Dipped light= %d ,   Head light=
%d\n",Cde_Veilleuse,Cde_Code,Cde_Phare);
        printf("    Left indicator= %d    Right indicator=
%d\n",Cde_Clign_Gauche,Cde_Clign_Droit);
        printf("    Horn= %d\n",Cde_Avertisseur);
        printf("    Stop light= %d\n",Cde_Stop);
        printf("    Warning= %d\n",Cde_Warning);
        }

ENDM
}

/*********************************************************************/
/*                              Main program                         */
/*********************************************************************/
main()
{
clsscr();
printf("CAN bus EX_2 with mtr86\n");
start_mtr(init,512);
}
```


Sample

4 EX N°3 : CHECK THE FUNCTION OF AN OPTICAL BLOCK

4.1 Topic :

Purposes :	- Analyze a diagram structure in order to define the hardware function realization by a software function. - Link up the remote frames and control frames to satisfy imposed specifications. - Test the response of received frames. - Extract the expected information from the response frame. - Analyze the received information and make a diagnosis. - Represent imposed specifications by a diagram of status. - Code and program a diagram of status. - Carry out cyclical actions, during a specific period.
Specifications :	We want to cycle the different statu[s] o[f] t[he] o[p]tical block (Nothing, Side light, Side light + Dipped light ,Si[de] ligh[t] + [...]t etc ...) and the indicator. We also realize the control [...]nc[e] - Verify that the ordered mod[ule] ret[urns] the acknowledgment frame. - Check (by software functio[n]) [...]ont[rolled] lamps are effectively lit (by current using). → We show what are the [...]lamps and the test result. We impose the lam[p]s perm[...]n p[e]riod: 3.6 S,

N[ec]ess[ary] [h]ware and software :

PC Micro Computer using W[in]dows [...]5 or later
Editor Software-Assembler-De[bu]gger
If programming in C, GNU; C / C++ [C]ompiler Ref: EID210101
Processor board 16/32 bit 68332 microcontroller and its software environment (Editor-Cross Assembler-Debugger) Ref: EID210001
CAN PC/104 Network board in ATON SYSTEMS Ref NIC: EID004001
CAN network with four power outputs modules for lights Ref: EID051001
USB connection cable, or if not available RS232 cable, Ref: EGD000003
AC / AC Power source 8V 1A Ref: EGD000001
12V Power source supply for the CAN modules ("energy" network)

Time : 4 hours

4.2 Solution elements

4.2.1 Analysis

Remind

The power circuit of the reference "VN05" which leads 4 power outputs Modules, generates a diagnostic signal marked "STAT" -> STATus (refer to the data sheet of the VN05 circuit).

In the case that we activate the power circuit, if the load current is close to 0 (bulb filament cut off for example), the signal "STATus" changes to 0. The output current threshold which sequences the setting to 0 for the "STATus" output is from 5 mA (minimum value) to 180 mA (maximum value).
In the case of 4 power outputs module, the LEDs who are intended to indicate whether a power output is at work or not, doesn't consume enough current to inhibit this diagnostic function.

According to the technical manual of CAN-VMD system and the structured diagram of 4 power outputs module, these 4 signals "STATus" are connected to the interface CAN MCP25050 circuit and therefore constitute diagnostic inputs that can be read through the CAN network:

- The "STATus" output of the power circuit driven by GP[illegible]ected to GP4,
- The "STATus" output of the power circuit driven by G[illegible]nected to GP5,
- The "STATus" output of the power circuit driv[illegible]cted to GP6,
- The "STATus" output of the power circuit dr[illegible]ected to GP7.

Activation of the lamps and diagnostic of the right f[illegible]ock

To turn on the lamps, it should send frame type IM "[illegible]e" with the "Write Register " function on its register accessible to the " GPPIN " [illegible]nd in technical manual MCP25050 circuit page 22).
In this case, according to the table given in Ch[illegible] fo[illegible]t front block, the identifier will be 0E880000, the " addr" parameter will be 1[illegible]ster address - page 37 of the "Data sheet" MCP25050), the "mask " parameter will b[illegible]its on the 4 least significant bits of the port), and the "value" parameter is defined by the [illegible]tatus.

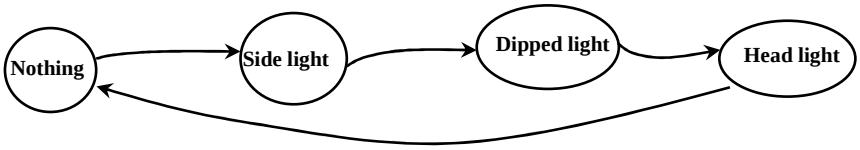
To recover the logic status im[illegible]rt, it should send the concerned module IRM frame "Information Request messa[illegible]ster" function. In this case, the identifier contains the considered address of the re[illegible]ress 1E - find in technical manual MCP25050 circuit page 37) as well as the func[illegible]ight of 07 .
Therefore, for the right front ligh[illegible]ding to the table given in Chapter 1 in this document) the identifier to give will ultimately be [illegible]41E 07 and the frame does not include a parameter in the "data" area.
The module receiver of this frame will respond with the same identifier, but with the rank 0 of the "Data" area, the status of the input port.

Lamp Swapping

We want the order of the lamp status is like following:

Nothing, then only side light, then side light + dipped light, then side light + head light.

This temporal sequence can be represented by the status diagram shown below:



We pass the previous status to the next status after the end of each "Light delay".

Data Structure

It is useful to store in the memory, an image of the lamp status of the optical block (image of the output port module status). When we want to change the status of a lamp, we deal with one of the bits in this file. Then this image becomes a block of the control frame.

The sent data in a control frame (or recovered by a remote frame) existing in 8 bits which we use the "byte_bits" data structure defined in the "Structure-Donnees.h" file.

```
/*   Union to access in one byte (BYTE) either directly or by individualizing 8 bits
//*****************************************************************************************/
        union byte_bits
        {       struct
                 {      unsigned char b7:1;
                        unsigned char b6:1;
                        unsigned char b5:1;
                        unsigned char b4:1;
                        unsigned char b3:1;
                        unsigned char b2:1;
                        unsigned char b1:1;
                        unsigned char b0:1;
                }bit;
                BYTE value;
        };
```

For light control

The image variable is defined in the vmd.h under the form

```
byte_bits Image_FVG;
```

The same as the binary variable, lamps state image:

- for a front optical block

```
#define Valeur_FVG Image_FVG.valeur       // For com...te ...ss
#define Veilleuse_FVG Image_FVG.bit.b0
#define Code_FVG Image_FVG.bit.b1
#define Phare_FVG Image_FVG.bit.b2
#define Clignot_FVG Image_FVG.bit.b3
```

To turn on a lamp, it's simple to do :

- change the status in the memory image,

```
Phare_FVG=1;  // for example
```

- transfer the memory image in the "va...e" ...ara... ...ontrol frame (IM frame),

```
T_IM_Feux.data[2]= Valeur_FVG;
```

- send the command frame,

```
Ecrire_Trame(T_IM_Feux);  // ...M ... ...mber that this is a control frame
```

For the bulbs' function control

Similarly for controlling the b...lbs s...tu...

- The image variable is defi...d

```
byte_bits Image_Et... _F...
```

Then we define binary varia...s and i...ges of the lamps status:

```
#define Etat_Veilleuse Image_...eux...t.b0
#define Etat_Code Image_Etat_Feu...it.
#define Etat_Phare Image_Etat_F...bit.2
#define Clignot Image_Etat_Feux.bi...
```

To read the status, it sends a remote frame

Ecrire_Trame(T_IRM_Feux); // IRM -> to remember that this is a remote frame

In the response frame, we recover the result of reading the port in the 0 rank parameter,

```
Image_Etat_Feux.value = Trame_recue.data[0];
```

We can then compare the logical status of the "status" bits based on the controlled outputs, for example:

- if Phare = 1 and Etat_Phare = 0 It's the bulb grilled or nothing is connected,
- if Phare = 1 and Etat_Phare = 1 It's OK.

4.2.2 Tasks management and organization

1- The initialization task will initialize the CAN board and create the following tasks.

2- The chase task will change a variable to turn on the optical block. It will activate the task which can turn on the optical block, then check its state. Then it will fall asleep to let other tasks run.

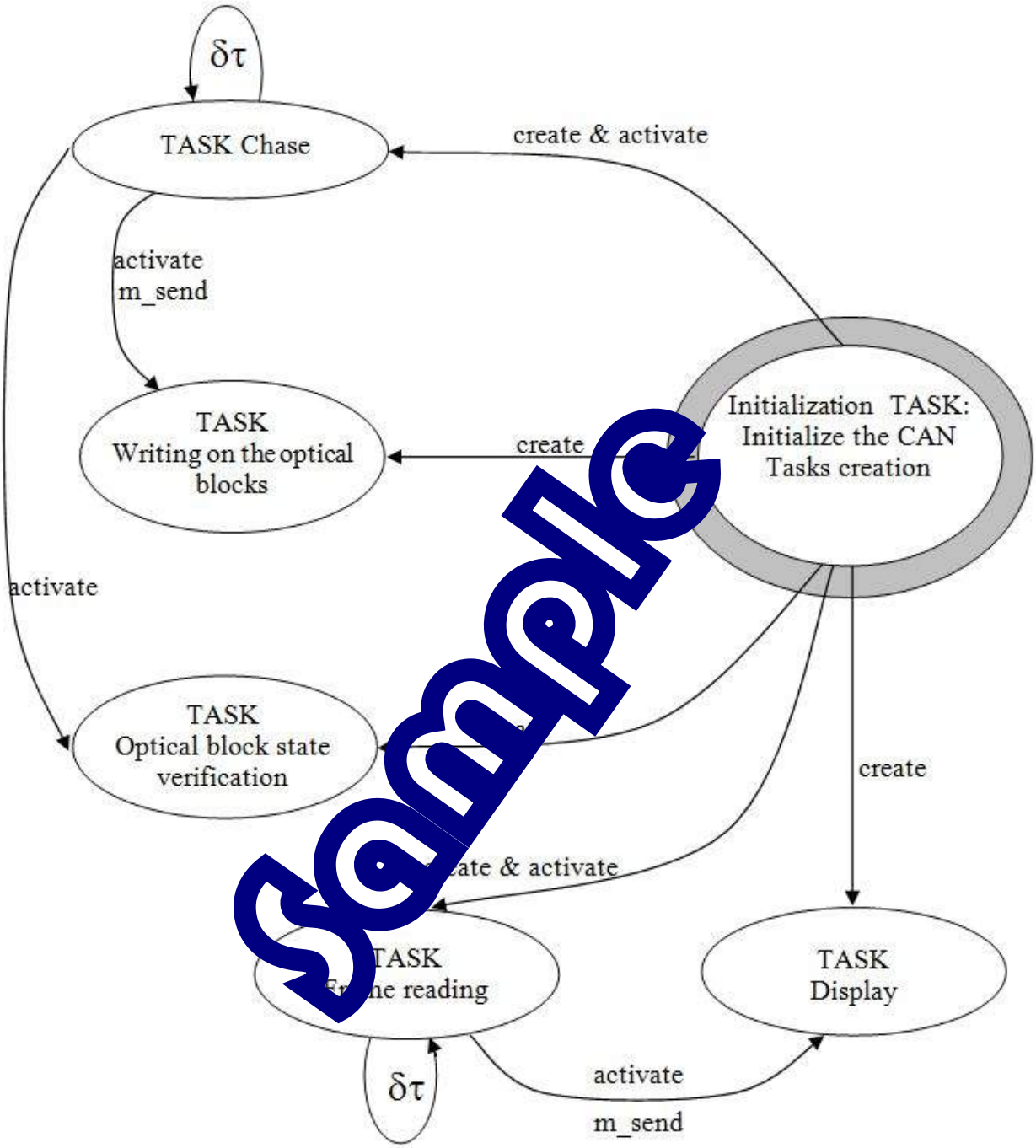
3- The EcrireFeux task will send the frame on the CAN network which can turn on or turn off the optical block bulb (IM type frame).

4-The State verification task will send the frame on the CAN network can know the real state of the optical block (IRM type frame).

5- The LectureTrame task will be permanently active. It will wait for a reception. Thenceforwards, it should distinguish the received frame type (IM or IRM) and load the variablrs (`Valeur_FVG` and `Valeur_Status_FVG`).
Then it will activate the display task.

6- The display task. It displays the received frame and build an optical block report(by Valeur_FVG and Valeur_Status_FVG).

4.2.3 State Diagram



4.2.4 "C" Program

```
/*The Left Front Lights Chase and State Verification*/

#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"

TACHE  init  (void);
TACHE  LectureTrame(void);
TACHE      chenillard(void);
TACHE  EcrireFeux();
TACHE  Verif_Etat_Feux(void);
TACHE  Affichage();

/*********************************************************************/
/*                                Initialization Task                                */
/*********************************************************************/
TACHE init(void)
{
OpenCAN(ATON_V2);   //ATON_V1 for old ATONCAN boards
                    //ATON_V2 for new ATONCAN boards
active(cree(chenillard,1,1024));       //permanently active
cree(EcrireFeux,2,1024);               //Create in the second priority to realize a writing,
cree(Verif_Etat_Feux,3,1024);          //before check the real state of the optical block
active(cree(LectureTrame,4,1024));     //frames reception
cree(Affichage,5,1024);                         //The display retains the last priority
}

/*********************************************************************/
/*                                Chase control task                                 */
/*********************************************************************/
TACHE chenillard(void)
{
static p_s=0x00;
while(1)
        {
        switch(p_s)
                        {
                        case 0 : p_s=0x01;
                                  break;
                        case 1 : p_s=0x02;
                                  break;
                        case 2 : p_s=0x04;
                                  break;
                        case 4 : p_s=0x08;
                                  break;
                        case 8 : p_s=0;
                                  break;
                }
        m_send(num_tache(EcrireFeux),&p_s);    //Writ      as      ivat
        active(num_tache(Verif_Etat_Feux)); //Verif      ion tas      ion
        dort(360); //stop 3,6 seconds
        }
}

/*********************************************************************/
/*                Frames sending control tas      the C      k                    */
/*    to turn on the optical block                                                   */
/*********************************************************************/
TACHE  EcrireFeux(e_s)
char e_s;
{
//The command frame initialization      the      hts
Trame T_IM_Feux;
T_IM_Feux.trame_info.registre=0x00;
T_IM_Feux.trame_info.champ.extend=1;
T_IM_Feux.trame_info.champ.dlc=0x03;
T_IM_Feux.ident.extend.identificateur.iden      _IM_FVG;
T_IM_Feux.data[0]=0x1e;
T_IM_Feux.data[1]=0x0f;
//Send the frame on the CAN bus to activate the lights
T_IM_Feux.data[2]=e_s;
EcrireTrame(&T_IM_Feux);
//task infinite sleeping (activation on awake or m_send)
}

/*********************************************************************/
/*                Frames sending control task on the CAN network             */
/*    to check the optical block state                                           */
/*********************************************************************/
TACHE  Verif_Etat_Feux(void)
{
//The state verification frame initialization on the Left Front Lights
Trame T_IRM_Feux;
T_IRM_Feux.trame_info.registre=0x00;
T_IRM_Feux.trame_info.champ.extend=1;
T_IRM_Feux.trame_info.champ.dlc=0x01;
T_IRM_Feux.trame_info.champ.rtr=1;
T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FVG;
//Send the remote frame on the CAN bus
EcrireTrame(&T_IRM_Feux);
//task infinite sleeping (activation on awake or m_send)
}

/*********************************************************************/
/*                Received frames reception and management                   */
/*********************************************************************/
TACHE LectureTrame(void)
{
```

```
static Trame tr_rx;
static Valeur_FVG_Mem;
while(1)
          {
          LireTrame(&tr_rx);

          if (tr_rx.ident.extend.identificateur.ident==Ident_T_AIM_FVG)
                    {
                    m_send(num_tache(Affichage),&tr_rx);
                    }
          if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FVG)
                    {
                    Valeur_Status_FVG=tr_rx.data[0];       //Recover the state in the most significant bits of data[0]
                    Valeur_FVG=tr_rx.data[0];              //Recover the asked state in the least significant bits of data[0]
                    if(Valeur_FVG_Mem!=Valeur_FVG)
                              {
                              Valeur_FVG_Mem=Valeur_FVG;
                              m_send(num_tache(Affichage),&tr_rx);
                              }
                    }
          }
}

/*********************************************************************/
/*                    Display the frames and the real states of the lamps                */
/*********************************************************************/
TACHE Affichage(tram)
Trame tram;
{
MONITOR
if(tram.ident.extend.identificateur.ident==Ident_T_AIM_FVG)
          gotoxy(1,7);
if(tram.ident.extend.identificateur.ident==Ident_T_IRM_FVG)
          {
          gotoxy(1,11);
          if(Veilleuse_FVG)   // We ask for turning on the side light
                                        //We check that the state meets the dem
                    if(S_Veilleuse_FVG) printf("Side light is OK \n");
                              else printf("Side light is out of servi
demands
          else if(Code_FVG)   //We ask for turning on the dipped ligh
                                        //We check that the state meets
                    if(S_Code_FVG) printf("Dipped light is OK \n");
                              else printf("Dipped light is out of se
          else if(Phare_FVG)  //We ask for turning on the head ligh
                                        //We check that the state
                    if(S_Phare_FVG) printf("Head light is OK \n")
                              else printf("Head light is out of se
          else if(Clignot_FVG)          //We ask for turning on th
                    if(S_Clignot_FVG) printf("Indicator is OK
                              else printf("Indicator is out
          gotoxy(1,9);
          }
AfficheTrame(&tram);
ENDM
}

/*********************************************************************/
/*                              Main progra                                */
/*********************************************************************/
main()
{
clsscr();
printf("CAN bus EX_3 with mtr86\n");
gotoxy(1,4);
printf("Received frames\n");
gotoxy(1,6);
printf("Acknowledge\n");
gotoxy(1,8);
printf("Lights' states\n");
start_mtr(init,512);
}
```


Sample

5 EX N°4 : COMMAND THE LAMPS BY LIGHTS STALK

5.1 Topic :

Purposes :	- Carry out a control application commanding a controllable system by the CAN network. - Display system status on the screen. - Carry out a precise delay. - Realize the tasks of verification and control.
Specifications :	After a regular time interval, from the enquired module on which is connected with the lights stalk, so that we can know its status. According to the status of the lights stalk activate the different lamps of front and back blocks. → The necessary delay to operate the indicators is realized by "programmable timer" (inside the microprocessor). → The different commands issued by the position of the lights stalk will be displayed individually. → It controls the blinking function of different blocks.

Necessary hardware and software :

PC Micro Computer using Windows Operating System
Editor Software-Assembler-Linker-Debugger
If programming in C, GNU; C / C++ Compiler Ref: EID210101
Processor board 16/32 bit 68332 microcontroller and its software environment
(Editor-Cross Assembler-Debugger) Ref: EID210001
CAN PC/104 Network board in ATON SYSTEMS Ref NIC: EID004001
CAN network with:
- 8 logic inputs module for the lights stalk Ref: EID050001
- 4 power outputs module for left/right back/front lights Ref: EID051001

USB connection cable, or if not available RS232 cable, Ref: EGD000003
AC / AC Power source 8V 1A Ref: EGD000001
12V Power source supply for the CAN modules ("energy" network)

Time : 2×4 hours

5.2 Solution elements

5.2.1 Analysis

Remind

Functionality

We consider a cyclic operation where the state of the lights stalk is required after a regular time interval imposed by a time base. The next state of the lights stalk is then compared to the previous state (inherited once before). If a position change has been detected, then a light conditions change begins. (We control successively 4 blocks with the new values)
→ We interrogate successively 4 blocks and check if the values of "Status" corroborate the sent commands. If a different is detected, an alarm message is displayed.
→ After sending a frame, we will verify that the received frame in response is correct (acknowledgment of module having received a control frame, or coherent response module having received a remote frame).

5.2.2 Tasks management and organization

1- The VMD initialization task can initialize the CAN b[illegible] the Input/Outputs function of different CAN modules (lights and stalk). It will activate[illegible] task.

2- The initialization task will initialize the CAN board[illegible] following tasks.

3- The Stalk Reading task will send an IRM to the li[illegible]r a regular time interval. It will falls asleep to let other tasks run and not saturate the b[illegible]

4-The Frame Reading task will be permaner[illegible] It must receive frames, manage variables updating and awaken the appropriate tasks[illegible]

5- The interrupt task will handle the in[illegible] have to be able to activate the task controlling the writing on the lights, by modifyi[illegible]ed parameter to turn off or turn on the adequate indicators.

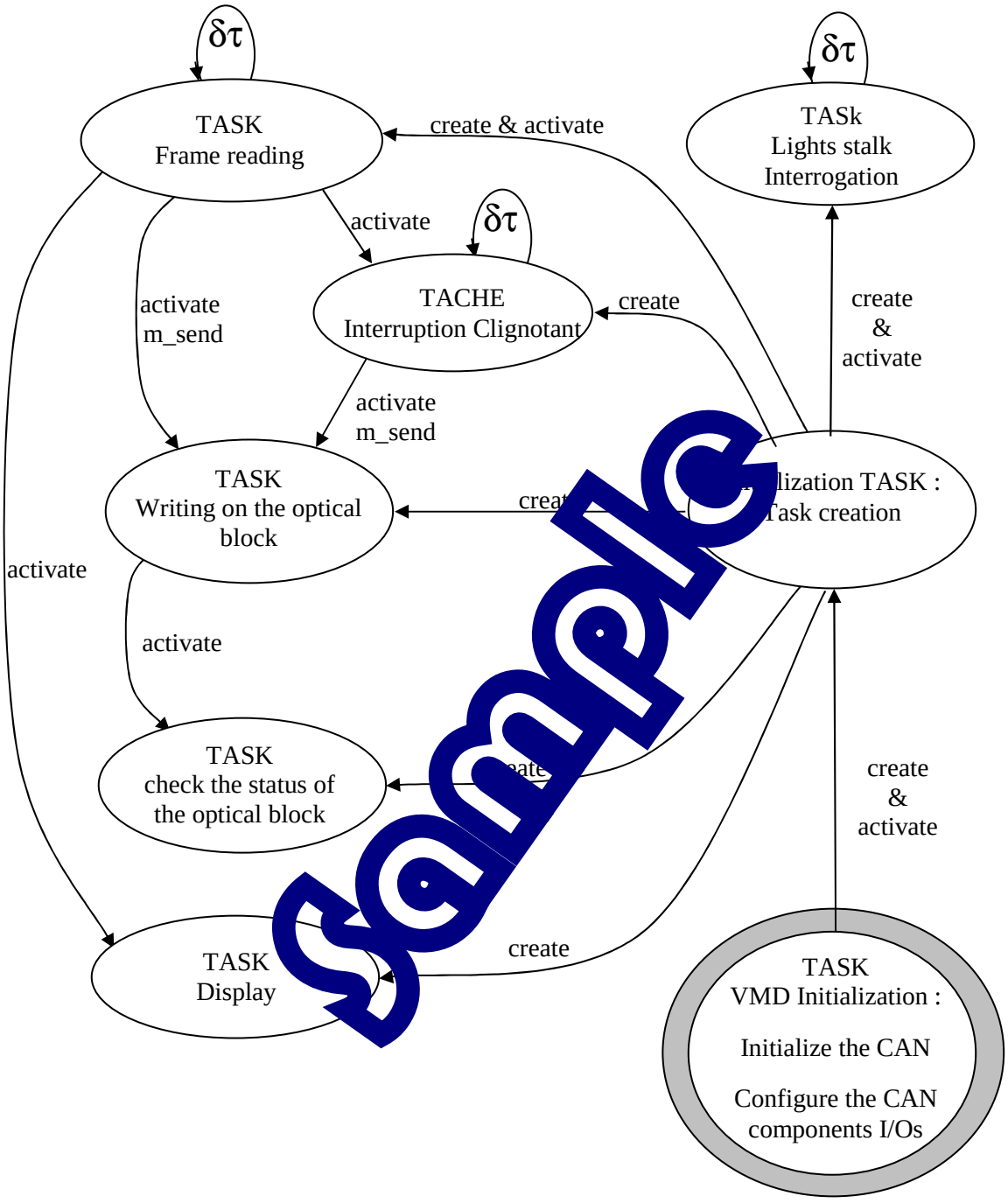
6-The EcrireFeux task will [illegible]nd the [illegible] on the CAN network which can turn on or turn off the optical block bulbs (IM type [illegible])

7-The State verification task will sen[illegible]e frame on the CAN network can know the real state of the optical block (IRM type frame).

8- The display task. It displays the received frame and build an optical block report (by Valeur_FVG and Valeur_Status_FVG).

Remark: The VMD initialization task can anticipate in the display; by putting in the text, the display task will only have to position themselves for displaying the stalk' and lights' states.

5.2.3 State diagram



5.2.4 "C" Program

```
/* VMD Management: Optical blocks and Lights stalk*/
#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"

//For the delay between 2 indicator flashing
#define Tempo_Clign 8        //unit: ms


//Tasks prototype
TACHE  Init_VMD  (void);
TACHE  init  (void);
TACHE      Actualise_Etat_Feux();
TACHE  irq_Clign(void);
TACHE  LectureTrame(void);
TACHE  Lecture_Commodo(void);
TACHE  Verif_Etat_Feux(void);
TACHE  Affichage();

/*********************************************************************/
/*                                VMD initialization Task                                */
/*********************************************************************/
TACHE Init_VMD(void)
{
//Modules conficulation
Trame T_IM_CFG,Trame_Recue;
int Temp,Cptr_TimeOut; // For the waiting delay

int cpt;//for the loops
//Different modules definition to configure under the form of table
int identificateur[]={Ident_T_IM_FRG,Ident_T_IM_FRD,Ident_T_IM_FVG,Ident_T_IM_[illegible],[illegible]ommodo_Feux};
int identificateurACK[]={Ident_T_AIM_FRG,Ident_T_AIM_FRD,Ident_T_AIM_F[illegible]dent_AI[illegible]d_T_AIM_Commodo_Feux};
int valeur_data2[]={0xF0,0xF0,0xF0,0xF0,0xFF};
char *msg[]={"Left Back Lights","Right Back Lights ","Left Front Lig[illegible]","R[illegible]t L[illegible]s"," Light Stalk"};

OpenCAN(ATON_V2);   //ATON_V1 for old ATONCAN boards
                    //ATON_V2 for new ATONCAN boards

//Frame configuration for the I/O configuration
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1; // Work in the extended mode
T_IM_CFG.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits [illegible]
// For the 4 outputs modules configuration (GP0 to GP3) and 4input[illegible]4 to GP8)
T_IM_CFG.data[0]=0x1F;        // first data -> "Address" of [illegible]GPDDR
T_IM_CFG.data[1]=0xFF;        // second data -> "Mask"-> Se[illegible]

//Configuration of modules to address and operation of th[illegible]
for(cpt=0;cpt<5;cpt++)
{
T_IM_CFG.ident.extend.identificateur.ident=identifica[illegible]  //e[illegible]lishment of the identifier
T_IM_CFG.data[2]=valeur_data2[cpt];    // third data-"Va[illegible]

MONITOR
gotoxy(2,9);        //we display a message to [illegible]te is being initialized
printf("Initialization of %s\n",msg[cpt]);

do        {
          EcrireTrame(&T_IM_CFG);//[illegible]e CAN network
          Cptr_TimeOut=0;     //The [illegible]ction [illegible]the task sleep if there aren't any  received frames.
          do{Cptr_TimeOut++;}while([illegible])==0)&&(Cptr_TimeOut<200));
          if(Trame_Recue.ident.exte[illegible]ident==identificateurACK[cpt])Cptr_TimeOut=200; // Test if the identifier
is correct
                    else {printf("Prob[illegible]
                              Cptr_TimeOut=[illegible]
          for(Temp=0;Temp<100000;Temp++);[illegible]for a while!
          }while(Cptr_TimeOut!=200);
printf("OK\n");
ENDM
}

MONITOR
clsscr();
//The display takes a lot of time, so we anticipate
gotoxy(1,1);
printf("************************EX_4 CAN bus with mtr86************************\n");
printf(" Lights Stalk State                                                \n");
printf("  GP0: Side light=      ,GP2:Head light=        ,GP3: Dipped light=                \n");
printf("  GP1: Warning=        ,GP4: Left indicator=   ,GP5: Right indicator=          \n");
printf("  GP6: Stop light=      ,GP7: Horn=                                \n");

printf(" Left Back Lights:   Action        RealState                       \n");
printf(" Side light                                                                                \n");
printf(" Stop light                                                                                \n");
printf(" Indicator                                                                                 \n");
printf(" Horn                                                                                      \n");
printf("                                                                                           \n");
printf(" Right Back Lights:   Action        RealState                       \n");
printf(" Side light                                                                                \n");
printf(" Stop light                                                                                \n");
printf(" Indicator                                                                                 \n");
printf("                                                                                           \n");
printf(" Left Front Lights:   Action        RealState                       \n");
printf(" Side light                                                                                \n");
printf(" Dipped light
        \n");
printf(" Head light                                                                                \n");
```

```
printf(" Indicator                                                                                                    \n");
printf("                                                                                                              \n");
printf(" Right Front Lights:   Action       RealState                          \n");
printf(" Side light                                                                                                   \n");
printf(" Dipped light
            \n");
printf(" Head light                                                                                                   \n");
printf(" Indicator                                                                                                    \n");
ENDM

active(cree(init,1,1024));
dort(0); //Stop indefinitely the task
}

/*********************************************************************/
/*                              Initialization task                  */
/*********************************************************************/
TACHE init(void)
{
cree(Actualise_Etat_Feux,1,1024);               //Priority n°1 because lights activation is the most important
active(cree(LectureTrame,2,1024));              //frame reading will automatically fall asleep
cree(irq_Clign,2,256);                                  //The IT priority management to have a exact timebase
active(cree(Lecture_Commodo,3,1024));  //Read the state of the light stalk
cree(Verif_Etat_Feux,5,1024);                   //Checking the status light is less important
cree(Affichage,6,1024);                                 //The display retains the lowest priority
}

/*********************************************************************/
/*                              IRQ control task                                    */
/*********************************************************************/
TACHE irq_Clign(void)
{
int actualise;
static char Flag_Fin_Tempo_Clign;

while((Cde_Clign_Gauche || Cde_Clign_Droit || Cde_Warning))
        {
        dort(10*Tempo_Clign);                           //Wait for
        Flag_Fin_Tempo_Clign^=0x01;  //Enable or disable the flag
//Pay attention to send a command, it checks at first if it is a ind      tor      the Wa      g
        if(((Cde_Clign_Gauche==0) && (Cde_Clign_Droit==0) && (Cde_W      ag==0      if command dropped->we leave
        if(Flag_Fin_Tempo_Clign==0x01) //End of the indicator delay (      Clig      mt)
                { //for extinction
                if(Cde_Clign_Droit) actualise=((Valeur_Commodo_Fe      0)); //Turn off the right indicator and Warning
                else if(Cde_Clign_Gauche) actualise=((Valeur_Co      do_Feu      FD); //Turn off the left indicator and
Warning
                else if (Cde_Warning ) actualise=Valeur_Commo      Fe      2;      Warning
                m_send(num_tache(Actualise_Etat_Feux),&actual      );
                }
        else if(Flag_Fin_Tempo_Clign==0x00) //End of the in      lay      o Clignottant)
                { //to turn on
                if (Cde_Clign_Droit) actualise=Valeur_      mmodo_Fe      //      on right indicator and turn off the
warning
                else if (Cde_Clign_Gauche) actualis      eu      do      ux&      D; //turn on left indicator and turn off
warning

                else if (Cde_Warning ) actualise=      eu      do      ux; //turn on Warning
                m_send(num_tache(Actualise_Etat_Fe      &ac      );
                }
        }

}

/*********************************      **********      **********/
/*                                      ception      */
/*   This task must also manage th      pda      d                  */
/* the waking up of appropriate ta                                  */
/*********************************      **********      **********/
TACHE LectureTrame(void)
{
static Trame tr_rx;
static Valeur_Commodo_Feux_Mem;         //      ariable as it is a memorisation
static Actu_Aff;    //Actu_Aff is defined in s      ic because it is a parameter to transmit
while(1)                                //Task actives permanently
        {

        if(LireTrame(&tr_rx))  //LireTrame stop the task when there arn't any received frame
           {
           if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_Commodo_Feux) //we received a stalk response
                {
                Valeur_Commodo_Feux=~(tr_rx.data[0]);  //Bitwise inversion to the stalk real state
                if(Valeur_Commodo_Feux_Mem!=Valeur_Commodo_Feux)
                        {//Its state changed
                        Valeur_Commodo_Feux_Mem=Valeur_Commodo_Feux; //we  update the stored value
                        //If a indicator is active, we validate the IT which will activate Actualise_Etat_Feux
                        if(Cde_Clign_Gauche || Cde_Clign_Droit || Cde_Warning) active(num_tache(irq_Clign));
                        //Otherwise, we wake up the lights state actualisation task with the stalk without indicator
value
                        else m_send(num_tache(Actualise_Etat_Feux),&Valeur_Commodo_Feux_Mem);
                        //we activate the display task
                        Actu_Aff=1;
                        m_send(num_tache(Affichage),&Actu_Aff);
                        }
                } //End of stalk handling

             if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FRG)
                {
                Valeur_Status_FRG=tr_rx.data[0];        //Recover the state in the most significant bits of data[0]
                Valeur_FRG=tr_rx.data[0];               //Recover the asked state in the least significant bits of data[0]
                //we activate the display task
                Actu_Aff=2;
                m_send(num_tache(Affichage),&Actu_Aff);
```

```
                    } //End of Left Back Lights handling

            if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FRD)
                    {
                    Valeur_Status_FRD=tr_rx.data[0];       //Recover the state in the most significant bits of data[0]
                    Valeur_FRD=tr_rx.data[0];              //Recover the asked state in the least significant bits of data[0]
                    //we activate the display task
                    Actu_Aff=3;
                    m_send(num_tache(Affichage),&Actu_Aff);
                    } //End of Right Back Lights handling

            if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FVG)
                    {
                    Valeur_Status_FVG=tr_rx.data[0];       //Recover the state in the most significant bits of data[0]
                    Valeur_FVG=tr_rx.data[0];              //Recover the asked state in the least significant bits of data[0]
                    //we activate the display task
                    Actu_Aff=4;
                    m_send(num_tache(Affichage),&Actu_Aff);
                    }  //End of Left Front Lights handling

            if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FVD)
                    {
                    Valeur_Status_FVD=tr_rx.data[0];       //Recover the state in the most significant bits of data[0]
                    Valeur_FVD=tr_rx.data[0];              //Recover the asked state in the least significant bits of data[0]
                    //we activate the display task
                    Actu_Aff=5;
                    m_send(num_tache(Affichage),&Actu_Aff);
                    } //End of Right Front Lights handling
             } //End of Received frames reading
          } //End of while
}

/***********************************************************************/
/*                                       Stalk questioning task (IRM)                              */
/***********************************************************************/
TACHE Lecture_Commodo(void)
{
Trame T_IRM_Commodo_Feux;
T_IRM_Commodo_Feux.trame_info.registre=0x00;
T_IRM_Commodo_Feux.trame_info.champ.extend=1;
T_IRM_Commodo_Feux.trame_info.champ.dlc=0x01;
T_IRM_Commodo_Feux.trame_info.champ.rtr=1;
T_IRM_Commodo_Feux.ident.extend.identificateur.ident=Ident_T_IRM_Commodo_F[illegible]
while(1)             //Task actives permanently
          {
          EcrireTrame(&T_IRM_Commodo_Feux);
          dort(10);
          }
}

/*************************************************************[illegible]*/
/*                                       Lights control tas[illegible]                              */
/*        Depending on the past parameter(image of stalk)[illegible]e [illegible] on
/*  turn off the lampes                                                                            */
/*************************************************************[illegible]*/
TACHE Actualise_Etat_Feux(Value_Commodo)           [illegible]ask c[illegible]d by reading frame
Port_8ES Value_Commodo; //the variable is under the f[illegible]n o[illegible]e of Port_8ES type
                                       //which allows [illegible]a d[illegible]cess to the bit level
{
Trame T_IM_Feux,T_recue; //Frame that will be s[illegible]e optical blocks
Valeur_FRG=0x00;    //sets optical blocks to [illegible]
Valeur_FRD=0x00;
Valeur_FVG=0x00;
Valeur_FVD=0x00;

if(Value_Commodo.bit.GP2) //Head l[illegible]t
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_P;
          Valeur_FVD|=Cde_FV_P;
          }
else if(Value_Commodo.bit.GP3) //Dipped light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_C;
          Valeur_FVD|=Cde_FV_C;
          }
else if(Value_Commodo.bit.GP0) //Side light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_V;
          Valeur_FVD|=Cde_FV_V;
          }

if(Value_Commodo.bit.GP1)     //Warning
          {
          Valeur_FRG|=Masque_Clign_AR;
          Valeur_FRD|=Masque_Clign_AR;
          Valeur_FVG|=Masque_Clign_AV;
          Valeur_FVD|=Masque_Clign_AV;
          }
          else
          {
          if(Value_Commodo.bit.GP4) //Left Indicator
                    {
                    Valeur_FRG|=Masque_Clign_AR;
                    Valeur_FVG|=Masque_Clign_AV;
                    }
          if(Value_Commodo.bit.GP5) //Right indicator
```

```
                    {
                    Valeur_FRD|=Masque_Clign_AR;
                    Valeur_FVD|=Masque_Clign_AV;
                    }
          }

if(Value_Commodo.bit.GP6)//Stop light
          {
          Valeur_FRG|=Masque_Stop;
          Valeur_FRD|=Masque_Stop;
          }

if(Value_Commodo.bit.GP7) //Horn
          Valeur_FRG|=Masque_Klaxon;

//printf("FRG=%x\nFRD=%x\nFVG=%x\nFVD=%x\n",Valeur_FRG,Valeur_FRD,Valeur_FVG,Valeur_FVD);

//initialization of the command frame on the lights
T_IM_Feux.trame_info.registre=0x00;
T_IM_Feux.trame_info.champ.extend=1;
T_IM_Feux.trame_info.champ.dlc=0x03;
T_IM_Feux.data[0]=0x1e;
T_IM_Feux.data[1]=0x0f;

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FRG;
T_IM_Feux.data[2]=Valeur_FRG;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FRD;
T_IM_Feux.data[2]=Valeur_FRD;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FVG;
T_IM_Feux.data[2]=Valeur_FVG;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FVD;
T_IM_Feux.data[2]=Valeur_FVD;
EcrireTrame(&T_IM_Feux);

//we just made a lights modification request  .
active(num_tache(Verif_Etat_Feux)); // we check the lights state
}


/*********************************************************************
/*                  Checking the actual state of the optical blo    t                */
/*        IRM sending to 4 optical blocks                                     */
/*********************************************************************
TACHE  Verif_Etat_Feux(void)
{
//Left Front Lights state verification frame initializati
Trame T_IRM_Feux;
T_IRM_Feux.trame_info.registre=0x00;
T_IRM_Feux.trame_info.champ.extend=1;
T_IRM_Feux.trame_info.champ.dlc=0x01;
T_IRM_Feux.trame_info.champ.rtr=1;
T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_I   FRG;
//Send the remote frame on the CAN bus
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificateur.ident   ent   RM_F
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identifica   r.i    =   nt_T_IRM   G;
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificat   ident=Ide   T_IRM_FVD;
EcrireTrame(&T_IRM_Feux);

//task infinite sleeping (activation on a    or   send)
}

/***********************************************************************/
/*                                               Display Task                                        */
/***********************************************************************/
TACHE Affichage(actu)
char actu;
{
if(actu==1)
          {
          MONITOR             //we update the values of the lights stalk on the Monitor
          gotoxy(18,3);
          printf("%d\n",Cde_Veilleuse);
          gotoxy(36,3);
          printf("%d\n",Cde_Phare);
          gotoxy(55,3);
          printf("%d\n",Cde_Code);
          gotoxy(16,4);
          printf("%d\n",Cde_Warning);
          gotoxy(41,4);
          printf("%d\n",Cde_Clign_Gauche);
          gotoxy(60,4);
          printf("%d\n",Cde_Clign_Droit);
          gotoxy(18,5);
          printf("%d\n",Cde_Stop);
          gotoxy(37,5);
          printf("%d\n",Cde_Klaxon);
          ENDM
          }
if(actu==2)
          {
```

```
                    MONITOR
                    if(Veilleuse_FRG)   //We ask for turning on the side light
                              {                   //We check that the state meets the demands
                              gotoxy(26,7);
                              printf("1\n");
                              gotoxy(40,7);
                              if(S_Veilleuse_FRG) printf("OK\n");    //State is 1, it's ok
                                        else printf("Out of service\n");       //State is 0, the state doesn't meet the demands
                              }
                    else
                              {
                              gotoxy(26,7);
                              printf("0\n");
                              gotoxy(40,7);
                              printf("  \n");
                              }
                    if(Stop_FRG)        //We ask for turning on the stop light
                              {                   //We check that the state meets the demands
                              gotoxy(26,8);
                              printf("1\n");
                              gotoxy(40,8);
                              if(S_Stop_FRG) printf("OK\n");
                                        else printf("Out of service\n");
                              }
                    else
                              {
                              gotoxy(26,8);
                              printf("0\n");
                              gotoxy(40,8);
                              printf("  \n");
                              }
                    if(Clignot_FRG)     //We ask for turning on the indicator
                              {                   //We check that the state meets th                ds
                              gotoxy(26,9);
                              printf("1\n");
                              gotoxy(40,9);
                              if(S_Clignot_FRG) printf("OK\n");
                                        else printf("Out of service\n"
                              }
                    else
                              {
                              gotoxy(26,9);
                              printf("0\n");
                              gotoxy(40,9);
                              printf("  \n");
                              }
                    if(Klaxon_FRG)      //We ask for turning on th hor
                              {                   //We check that  st                the demands
                              gotoxy(26,10);
                              printf("1\n");
                              gotoxy(40,10);
                              if(S_Klaxon_FRG) printf("OK\
                                        else printf("Ou                 e
                              }
                    else
                              {
                              gotoxy(26,10);
                              printf("0\n");
                              gotoxy(40,10);
                              printf("  \n");
                              }
                    ENDM
          }

if(actu==3)
{
                    MONITOR
                    if(Veilleuse_FRD)          ask      turning on the side light
                              {                        We check that the state meets the demands
                              gotoxy(26,1
                              printf("1\n");
                              gotoxy(40,13);
                              if(S_Veilleuse_FRD) printf("OK\n");    //State is 1, it's ok
                                        else printf("Out of service\n");                //State is 0, the state doesn't meet
the demands
                              }
                    else
                              {
                              gotoxy(26,13);
                              printf("0\n");
                              gotoxy(40,13);
                              printf("  \n");
                              }
                    if(Stop_FRD)        //We ask for turning on the stop light
                              {                   //We check that the state meets the demands
                              gotoxy(26,14);
                              printf("1\n");
                              gotoxy(40,14);
                              if(S_Stop_FRD) printf("OK\n");
                                        else printf("Out of service\n");
                              }
                    else
                              {
                              gotoxy(26,14);
                              printf("0\n");
                              gotoxy(40,14);
                              printf("  \n");
                              }
                    if(Clignot_FRD)     //We ask for turning on the indicator
                              {                   //We check that the state meets the demands
                              gotoxy(26,15);
```

```
                                printf("1\n");
                                gotoxy(40,15);
                                if(S_Clignot_FRD) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,15);
                                printf("0\n");
                                gotoxy(40,15);
                                printf("  \n");
                                }
                        ENDM

}

if(actu==4)
{
                        MONITOR
                        if(Veilleuse_FVG)   //We ask for turning on the side light
                                {                       //We check that the state meets the demands
                                gotoxy(26,18);
                                printf("1\n");
                                gotoxy(40,18);
                                if(S_Veilleuse_FVG) printf("OK\n");     //State is 1, it's ok
                                        else printf("Out of service\n");       //State is 0, the state doesn't meet the demands
                                }
                        else
                                {
                                gotoxy(26,18);
                                printf("0\n");
                                gotoxy(40,18);
                                printf("  \n");
                                }
                        if(Code_FVG)        //We ask for turning on the dipped light
                                {                       //We check that the state meet[illegible]
                                gotoxy(26,19);
                                printf("1\n");
                                gotoxy(40,19);
                                if(S_Code_FVG) printf("OK\n");
                                        else printf("Out of service\n
                                }
                        else
                                {
                                gotoxy(26,19);
                                printf("0\n");
                                gotoxy(40,19);
                                printf("  \n");
                                }
                        if(Phare_FVG)       //We ask for turning [illegible]
                                {                       //We chec[illegible] the demands
                                gotoxy(26,20);
                                printf("1\n");
                                gotoxy(40,20);
                                if(S_Phare_FVG) printf("
                                        else printf("
                                }
                        else
                                {
                                gotoxy(26,20);
                                printf("0\n");
                                gotoxy(40,20);
                                printf("
                                }
                        if(Clignot_FVG)
                                {                       k that the state meets the demands
                                gotox
                                printf
                                gotoxy(
                                if(S_Clignot_FRD) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,21);
                                printf("0\n");
                                gotoxy(40,21);
                                printf("  \n");
                                }
                        ENDM

}

if(actu==5)
          {
          MONITOR
          if(Veilleuse_FVD)   //We ask for turning on the side light
                    {                       //We check that the state meets the demands
                    gotoxy(26,24);
                    printf("1\n");
                    gotoxy(40,24);
                    if(S_Veilleuse_FVD) printf("OK\n");     //State is 1, it's ok
                              else printf("Out of service\n");                //State is 0, the state doesn't meet the demands
                    }
          else
                    {
                    gotoxy(26,24);
                    printf("0\n");
                    gotoxy(40,24);
                    printf("  \n");
                    }
          if(Code_FVD)        //We ask for turning on the dipped light
```

```
                    {                   //We check that the state meets the demands
                    gotoxy(26,25);
                    printf("1\n");
                    gotoxy(40,25);
                    if(S_Code_FVD) printf("OK\n");
                              else printf("Out of service\n");
                    }
          else
                    {
                    gotoxy(26,25);
                    printf("0\n");
                    gotoxy(40,25);
                    printf("  \n");
                    }
          if(Phare_FVD)       //We ask for turning on the head light
                    {                   //We check that the state meets the demands
                    gotoxy(26,26);
                    printf("1\n");
                    gotoxy(40,26);
                    if(S_Phare_FVD) printf("OK\n");
                              else printf("Out of service\n");
                    }
          else
                    {
                    gotoxy(26,26);
                    printf("0\n");
                    gotoxy(40,26);
                    printf("  \n");
                    }
          if(Clignot_FVD)     //We ask for turning on the indicator
                    {                   //We check that the state meets the demands
                    gotoxy(26,27);
                    printf("1\n");
                    gotoxy(40,27);
                    if(S_Clignot_FVD) printf("OK\n");
                              else printf("Out of service\n");
                    }
          else
                    {
                    gotoxy(26,27);
                    printf("0\n");
                    gotoxy(40,27);
                    printf("  \n");
                    }
          ENDM
          }

}

/*****************************************************************/
/*                              Main Program                     */
/*****************************************************************/
main()
{
clsscr();
start_mtr(Init_VMD,1024);
}
```

6 EX N°5 : COMMAND THE WINDSHIELD WIPER BY STALK

6.1 Topic :

Purposes :	Carry out an application of a command control system driven by CAN network. - Show the system state on the screen. - Realize accurate timers. - Carry out verification and control tasks.
Specifications :	After a regular time interval, we question the module on which is connected the windshield wiper stalk in order to know its state. Depending on the wiper stalk state, it activates the motor rotation in the appropriate direction. → The necessary delay for the operation in intermittent position is realized by a specific task. → The different commands implemented by the position of the stalk wheel are displayed individually. → We control the motor rotation condition by controlling the left and right limit switches.

Necessary hardware and software :

PC Micro Computer using Windows® Software
Editor Software-Assembler-Debugger
If programming in C, GNU; C / C + + Compiler Ref: EID210101
Processor board 16/32 bit 68332 Microcontroller and its software environment
(Editor-Cross Assembler-Debugger) Ref: EID210001
CAN PC/104 Network board in ATON SYSTEMS Ref NIC: EID004001
CAN network with:
- 8 logic inputs module for the lights stalk Ref: EID050001
- 4 power outputs module for left/right back/front lights Ref: EID051001

USB connection cable, or if not available RS232 cable, Ref: EGD000003
AC / AC Power source 8V 1A Ref: EGD000001
12V Power source supply for the CAN modules ("energy" network)

Time : 4 hours

6.2 Solution elements

6.2.1 Analysis

Remind

Windshield wiper stalk module function :

The identifier defined in Chapter 1, for an "IM" (Input Message -> Command Frame) sent to the "Wiper stalk" board is: 0x05880000

→ Definition of structured variables under the "**Trame**" model:

```
Trame T_IM_Commodo_EG;
```

→ Definition of the different elements of the "**T_IM_Asservissement**" structured variable

```
T_IM_Commodo_EG.trame_info.register=0x00;  // All bits are initialized to 0
T_IM_Commodo_EG.trame_info.champ.extend=1; // Work in extended mode
T_IM_Commodo_EG.trame_info.champ.dlc=0x03; //There will be a 3 bytes data
T_IM_Commodo_EG.ident.extend.identificateur.ident=0x05880000;
```

→ *Enable and configure the conversion from Analog to Digita*

According to the technical manual MCP25050 circuit (p[illegible])

Initialize the ADCON0 register

```
T_IM_ Commodo_EG.data[0]=0x2A; // ADCON0 regis[illegible] address
 (doc MCP25050 p15) 0E_H + shift = 0E_H + 1C_H =
T_IM_ Commodo_EG.data[1]=0xF0; // Mask : [illegible] is affected
T_IM_ Commodo_EG.data[2]=0x80; // Value: [illegible] Activation of the converter
and "prescaler rate" = 1:32
```

As well as ADCON1 register.

```
T_IM_ Commodo_EG.data[0]=0x2B; // AD[illegible] address
 (doc MCP25050 p15) 0F_H + shift = 0E[illegible]
T_IM_ Commodo_EG.data[1]=0xFF; // [illegible] bits are affected
T_IM_ Commodo_EG.data[2]=0xOE; // [illegible] MCP25050 p36)
     b7=ADCS1=0; b6=ADCS0=0 → Fre[illegible]c/2
     b5=VCFG1=0; b4=VCFG0=0 → [illegible]tage 0/+5V
     PCFG3:PCFG0=1110          → C[illegible] input 0 ( on GP0)
```

Acquisition frames (type "[illegible]est Message) of the wiper stalk state:

The identifier defined in Ch[illegible]M"; sent to the " wiper stalk " board is :

0x05[illegible]0000

→ Definition of structured variab[illegible]r the "**Trame**" model:

```
Trame T_IRM_Etat_Commodo_E[illegible]
```
// Frame appointed for enquiry of the 8E module to acquire the status of the wiper stalk.

→ Access and definition of the different elements of the " **Lecture_FC** " structured variable

```
T_IRM_ Etat_Commodo_EG.trame_info.register=0x00;  // All bits are initialized to
0
T_IRM_ Etat_Commodo_EG.trame_info.champ.extend=1; // Work in extended mode
T_IRM_ Etat_Commodo_EG.trame_info.champ.dlc=0x08; // There will be a 8 bytes data
T_IRM_ Etat_Commodo_EG.ident.extend.identificateur.ident=0x05840000;
```

In response to this query frame, we recover the logic states in the rank 1 data and the conversion result of the wheel position in the rank 2 data.

Servo-system module function:

→ Definition of structured variables under the "**Trame**" model :

```
Trame T_IM_Asservissement;
```

→ Defining identification details of the "**T_IM_Asservissement**" structured variable

```
T_IM_Asservissement.trame_info.register=0x00;  // All bits are initialized to 0
T_IM_Asservissement.trame_info.champ.extend=1; // Work in extended mode
T_IM_Asservissement.trame_info.champ.dlc=0x03; // There is 3 data of 8 bits (3
bytes)
T_IM_Asservissement.ident.extend.identificateur.ident=0x00880000;
```

In each control frames "IM", there will have to define the three associated bytes.

Definition of associated data of three bytes for:
define the input and output

```
T_IM_Asservissement.data[0]=0x1F; // GPDDR register address
T_IM_Asservissement.data[1]=0x7F; // Mask: 7 Bits not affected
T_IM_Asservissement.data[2]=0xE3; // Value: load into the addressed register
```

initialize the output GP2 by PWM1 output (variation of motor speed in the positive direction)

```
T_IM_Asservissement.data[0]=0x21; // T1CON Register address
T_IM_Asservissement.data[1]=0xB3; // Mask Register (doc MCP25050 p32)
T_IM_Asservissement.data[2]=0x80; // Value loaded into the addressed register
```

Set the frequency of the t PWM1 output:

```
T_IM_Asservissement.data[0]=0x23; // PR1Register address
T_IM_Asservissement.data[1]=0xFF; // Mask Register (doc MCP25050 p32)
T_IM_Asservissement.data[2]=0xFF; // 255 loaded into the register
```

initialize the output G32 by PWM2 output(variation of motor speed in the negative direction)

```
T_IM_Asservissement.data[0]=0x22; // Register T2CON address
T_IM_Asservissement.data[1]=0xB3; // Mask Register (doc MCP25050 p32)
T_IM_Asservissement.data[2]=0x80; // Value loaded into the addressed register
```

Set the frequency of the t PWM2 output:

```
T_IM_Asservissement.data[0]=0x24; // Register PR2 address
T_IM_Asservissement.data[1]=0xFF; // Mask register (doc MCP25050 p32)
T_IM_Asservissement.data[2]=0xFF; // 255 loaded into the register
```

change the cyclic duty of the PWM1 output (motor control in the positive direction)

```
T_IM_Asservissement.data[0]=0x[illegible]; // PWM1DCH Register address
T_IM_Asservissement.data[1]=0xFF; // Mask register (doc MCP25050 p33)
T_IM_Asservissement.data[2]=0x[illegible]; // All bits are initialized to 0 -> No Command
```

change the cyclic duty of the PWM2 output (motor control in the negative direction)

```
T_IM_Asservissement.data[0]=0x[illegible]; // PWM2DCH Register address
T_IM_Asservissement.data[1]=0xFF; // Mask register (doc MCP25050 p33)
T_IM_Asservissement.data[2]=0x00; // All bits are initialized to 0 -> No Command
```

to validate the power circuit

```
T_IM_Asservissement.data[0]=0x1E; // GPLAT Register address
T_IM_Asservissement.data[1]=0x10; // Mask register (doc MCP25050 p27)
T_IM_Asservissement.data[2]=0x10; // Set 4th register bit to 1
```

Acquire the status of the limit switch:

Definition of remote frame to know the status of the limit switch:
Just send an IRM to Servo Module.

→ Definition of structured variables under the "**Trame**" model:
Trame T_IRM_Acquisition_FC; // Frame appointed for enquiry of the servo-system module to acquire the limit switch
Remark: The structured variable **T_IRM_Acquisition_FC** only contain 5 useful bytes.

→ Access and definition of the different elements of the " **Lecture_FC** " structured variable

```
  T_IRM_Acquisition_FC.trame_info.register=0x00;  // All bits are initialized to 0
  T_IRM_Acquisition_FC.trame_info.champ.extend=1; // Work in extended mode
  T_IRM_Acquisition_FC.trame_info.champ.dlc=0x01; // There is 3 data of 8 bits (3 bytes)
  T_IRM_Acquisition _FC.ident.extend.identificateur.ident=0x00841E07;
```

Definition of structured variables images of the state of the limit switch

The received frame in response to this remote frame include in data [0], state of the limit switch. We copy the received data in a variable image.

```
union byte_bits FC;
#define Etats_FC FC.value // For the group of state of the limit switch
#define fs FC.bit.b7  // For over limit switch
#define fcg FC.bit.b6 // For left limit switch
#define fcd FC.bit.b5 // For right limit switch
```

In order to detect the modification of the sensors' states, it is pres[illegible]tates in a second structured variable ->Etat_mémorisé

```
static Valeur_FC_EG_Mem;      // Variable defined as static to pre[illegible]e its [illegible]thus do the comparison
```

If the status of an acquired variable is different from [illegible] st[illegible] v[illegible]ue, there has been a state change. Therefore we have to do something.

6.2.2 Tasks management and organization

1- The VMD initialization task can initialize the CAN board and the Inputs/Outputs function, of the PWM output and of the analog/digital converter function of different CAN modules (servo-system and stalk). It will activate the initialization task.
Remark : 1- We will be able to make the configuration through the table with a loop.
2- This task can anticipate on the display; by putting in the text, the display task will only have to position themselves for displaying the stalk's and limit switch's states

2- The initialization task will initialize the CAN board and create the following tasks.

3- The Frame Reading task will be permanently active. It should receive the frames, manage the variables updating, and wake up the appropriate tasks if there is a variables' state change.

4- The interrupt task will handle the intermittent position. It will activate or inhibit a flag indicating the end of the tempo, and activate the task manager and the motor rotation.

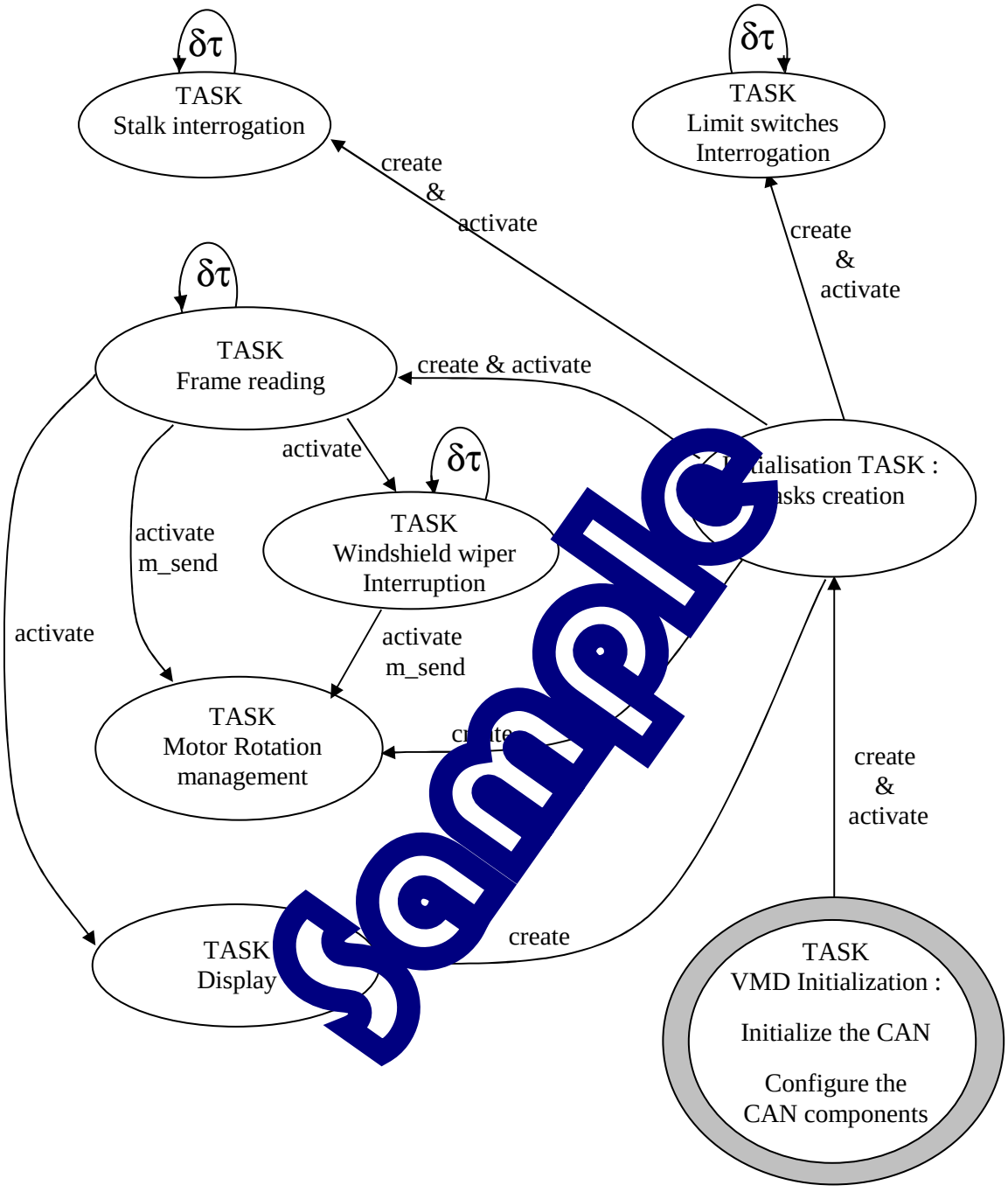
5-The servo-system task should control the motor rotation according to the stalk position, the asked speed and the limit switch.

6-The stalk interrogation task should send a remote frame (IRM) to [illegible] after a regular time interval.

7- The limit switch interrogation task should send a remote frame (RTR) to the servo-system after a regular time interval.

8-The display task will build a report for the stalk states and limit switches.

6.2.3 State diagram



6.2.4 "C" Program

```
/*VMD management: Windshield Wiper and its Stalk*/
#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"

//Global Variables
//For the Windshield Wiper delay
#define Tempo1 2    // in second
#define Tempo2 4 //For intermittent mode
#define Tempo3 6
#define Tempo4 8
#define Tempo5 10
//For the beat rate
#define Vitesse_Rapide 0x80
#define Vitesse_Lente 0x50
#define Vitesse_Lave_Glace 0x60
//Flag of Windshield Wiper delay end
char Flag_Fin_Tempo_EG;

// Tasks Prototype
TACHE  Init_VMD  (void);
TACHE  init  (void);
TACHE      Actualise_ASV();
TACHE  Lecture_FDC(void);
TACHE  LectureTrame(void);
TACHE  irq_eg();
TACHE  Lecture_Commodo_EG(void);
TACHE  Affichage(void);

TACHE Init_VMD(void)
{
//Modules Configuration
Trame T_IM_CFG,Trame_Recue,T_IRM_Etat_FC;
int Temp,Cptr_TimeOut; //For the waiting delay

int cpt;
//Table including concerned modules' identifers
int identificateur[]={Ident_T_IM_Commodo_EG,Ident_T_IM_Asservissement};
int identificateurACK[]={Ident_T_AIM_Commodo_EG,Ident_T_AIM_Asservissement
//Table including the data for I / O configuration
 int valeur_data2[]={0xFF,0xE3};
//Strings
char *msg[]={" Windshield Wiper Stalk"," Servo-system Module"};

//Table including all the data.
//The 2 first values are for the Analog/Digital Convertor of the W   st   eel.
//The following are for the Servo-system (cfg PWM, frequen   dati   .)
int datazero[]={0x2A,0x2B,0x21,0x23,0x22,0x24,0x25,0x26,0   };
int dataun[]={0xF0,0xFF,0xB3,0xFF,0xB3,0xFF,0xFF,0xFF,0x1   ;
int datadeux[]={0x80,0x0E,0x80,0xFF,0x80,0xFF,0x00,0x00   0};


OpenCAN(ATON_V2);   //ATON_V1 for old ATONCAN boards
                    //ATON_V2 for new ATONC   boa

T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1; // Work i   e ext
T_IM_CFG.trame_info.champ.dlc=0x03; // There   e 3 data   ts (3 sent bytes)
// For the 4 outputs modules configuration   0 t   and   nputs status 4 (GP4 to GP8)
T_IM_CFG.data[0]=0x1F;        // fi   ta   "A   oncerned register-> GPDDR
T_IM_CFG.data[1]=0xFF;        // s   d   "Ma   e Doc MCP25050 page 16


//Configuration of modules to addr   and ope   of their I/O
for(cpt=0;cpt<2;cpt++)
{
T_IM_CFG.ident.extend.identificateur.ident   ent   cateur[cpt]; //establishment of the identifier
if(cpt==0) T_IM_CFG.data[2]=valeur_data2[   // third data-> "Value"
if(cpt==1) T_IM_CFG.data[2]=valeur_data2[1];   // third data-> "Value"

MONITOR
gotoxy(2,9);        //we display a message to say that which module is being initialized
printf("Initialization of %s\n",msg[cpt]);

do        {
          EcrireTrame(&T_IM_CFG);// Send a first frame on the CAN network
          Cptr_TimeOut=0;     //The LireTrame function let the task sleep if there aren't any received frames.
          do{Cptr_TimeOut++;}while((LireTrame(&Trame_Recue)==0)&&(Cptr_TimeOut<200));
          if(Trame_Recue.ident.extend.identificateur.ident==identificateurACK[cpt])Cptr_TimeOut=200; // Test if the identifier
is correct
                    else {printf("Problem\n");
                              Cptr_TimeOut=0;}
          for(Temp=0;Temp<100000;Temp++); // To wait for a while!
          }while(Cptr_TimeOut!=200);
printf("OK\n");
ENDM
}

//Specific Configuration of windshield wiper stalk and of Servo-system module
for(cpt=0;cpt<9;cpt++)
{
if(cpt==0) T_IM_CFG.ident.extend.identificateur.ident=identificateur[0];
if(cpt==2) T_IM_CFG.ident.extend.identificateur.ident=identificateur[1];

T_IM_CFG.data[0]=datazero[cpt];        // Register address
T_IM_CFG.data[1]=dataun[cpt];          // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];        // Value
MONITOR
gotoxy(2,9);        //we display a message to say that which module is being configured
```

```
if(cpt<2) printf("Configuration of %s ...\n",msg[0]);
if(cpt>=2) printf("Configuration of %s ...\n",msg[1]);
ENDM
EcrireTrame(&T_IM_CFG);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the response
}

//Take the original position
MONITOR
printf(" Take the wiper POM \n");
ENDM
//Configuration of limit switch interrogation
T_IRM_Etat_FC.trame_info.registre=0x00;
T_IRM_Etat_FC.trame_info.champ.extend=1; // Work in the extended mode
T_IRM_Etat_FC.trame_info.champ.dlc=0x01; // There is 1 data of 8 bits
T_IRM_Etat_FC.trame_info.champ.rtr=1;  //in return
T_IRM_Etat_FC.ident.extend.identificateur.ident=Ident_T_IRM1_Asservissement;
EcrireTrame(&T_IRM_Etat_FC);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the response
Valeur_FC_EG=~(Trame_Recue.data[0]);

if(!(fcd==0 && fcg==1))
          {
          //Configuration of motor control frame
          T_IM_CFG.trame_info.registre=0x00;
          T_IM_CFG.trame_info.champ.extend=1; // Work in the extended mode
          T_IM_CFG.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
          // For the 4 outputs modules configuration (GP0 to GP3) and 4inputs status 4 (GP4 to GP8)
          T_IM_CFG.data[0]=0x26;
          T_IM_CFG.data[1]=0xFF;
          T_IM_CFG.data[2]=0x40;
          EcrireTrame(&T_IM_CFG);//active negative speed
          while(!(fcd==0 && fcg==1))
                    {
                    EcrireTrame(&T_IRM_Etat_FC);
                    while(LireTrame(&Trame_Recue)==0){}; //Wait for the respons
                    Valeur_FC_EG=~(Trame_Recue.data[0]); // Recover the limit
                    }
          T_IM_CFG.data[2]=0; //cancel the negative speed
          EcrireTrame(&T_IM_CFG); //disable the negative control
          } //End of Servo-system module initialization
EcrireTrame(&T_IRM_Etat_FC);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the response
Valeur_FC_EG=~(Trame_Recue.data[0]);

MONITOR
clsscr();
//The display takes a lot of time, so we anticipate
gotoxy(1,1);
printf("*************************EX_5 CAN bus with mtr86*****                    ****\n");
printf("
\n");
printf(" Windshield Wiper Stalk Status                                                \n");
printf("  GP0: Cde_EG_Av_Int=           , Value Potar                                                  \n");
printf("  GP3: Cde_EG_Av_Pos1=        , GP4: Cde_EG_A    os2=                                        \n");
printf("  GP7: Cde_Lave_Glace_Av=                                                                      \n");
printf("  GP1: Input1=                  , GP2: Inpu                                                \n");
printf("  GP5: Cde_EG_Ar=                                                                \n");
printf("  GP6: Cde_Lave_Glace_Ar=                                                                                \n");
printf("
\n");
printf("  Limit Switch State on the Servo-sy    m                                                \n");
printf("  : Left Limit Switch    =                                                                       \n");
printf("  : Over Limit Switch    =                                                                       \n");
printf("  : Right Limit Switch   =                                                                       \n");

ENDM

active(cree(init,1,1024));
dort(0); //Stop indefinitely the task
}

TACHE init(void)
{
cree(Actualise_ASV,1,1024);            //Priority n°1 because windshield wiper management and
active(cree(Lecture_FDC,1,1024));      //limit switch state are the most important
active(cree(LectureTrame,2,1024));              //frame reading will automatically fall asleep
cree(irq_eg,2,1024);                              //Delay between two beats
active(cree(Lecture_Commodo_EG,3,1024));        //Read the state of the windshield wiper stalk
cree(Affichage,4,1024);                           //The display retains the lowest priority
}

/*****************************************************************************************************************/
/*                                  Reading limit switches and managing the direction of motor rotation
                  */
/*****************************************************************************************************************/
TACHE Actualise_ASV(Vitesse_EG)                 //task called by reading frame
char Vitesse_EG;
{
Trame T_IM_Cmde_Negative,T_IM_Cmde_Positive;
//Configuration of Negative command frame
T_IM_Cmde_Negative.trame_info.registre=0x00;
T_IM_Cmde_Negative.trame_info.champ.extend=1; // Work in the extended mode
T_IM_Cmde_Negative.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
T_IM_Cmde_Negative.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
T_IM_Cmde_Negative.data[0]=0x26;        // PWM2DC register address
T_IM_Cmde_Negative.data[1]=0xFF;        // Mask -> all the bits are concerned
//Configuration of Positive command frame
T_IM_Cmde_Positive.trame_info.registre=0x00;
T_IM_Cmde_Positive.trame_info.champ.extend=1; // Work in the extended mode
T_IM_Cmde_Positive.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
T_IM_Cmde_Positive.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
```

```
T_IM_Cmde_Positive.data[0]=0x25;        // PWM2DC register address
T_IM_Cmde_Positive.data[1]=0xFF;        // Mask -> all the bits are concerned

//The windshield washer controls are reversed on the stalk(Active=0 , inactive=1)
if((Cde_EG_Av_Pos2 || Cde_EG_Av_Pos1 || (Cde_Lave_Glace_Av==0))) //a "classic" command is valid
          {
          if(fcd)                                 //If it's at right side, return to left
                    {
                    T_IM_Cmde_Negative.data[2]=Vitesse_EG;
                    T_IM_Cmde_Positive.data[2]=0x00;
                    }
          else if(fcg)                            //If it's at left side
                    {
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=Vitesse_EG;
                    }
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(Valeur_Analogique<200) //intermittent control is activated, we must manage according to the delay
          {
          if(fcd) //we are at right, it passes to the left
                    {
                    T_IM_Cmde_Negative.data[2]=Vitesse_EG;
                    T_IM_Cmde_Positive.data[2]=0x00;
                    Flag_Fin_Tempo_EG=0;                                  //we cancel the flag of delay end
                    }
          else if ((Flag_Fin_Tempo_EG)) //the delay is finished
                    {
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=Vitesse_EG; //we launch the motor
                    }
          else if ((fcg) && (Flag_Fin_Tempo_EG==0))
                    { //at left, in intermittent position, but the delay isn't fi
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=0x00;        //we stop the motor
                    }
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fcd) //it's at the right without any command, return to left
          {
          T_IM_Cmde_Negative.data[2]=0x40;
          T_IM_Cmde_Positive.data[2]=0x00;
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fcg) //it's at the left without any command, return to ri
          {
          T_IM_Cmde_Negative.data[2]=0x00;
          T_IM_Cmde_Positive.data[2]=0x00;
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
}

/*****************************************************************************************************************/
/*                                                Send          Servo-system board
                              */
/*****************************************************************************************************************/
TACHE Lecture_FDC(void)
{
Trame T_IRM_ASV;
T_IRM_ASV.trame_info.registre=0x00
T_IRM_ASV.trame_info.champ.extend=
T_IRM_ASV.trame_info.champ.dlc=0x0
T_IRM_ASV.trame_info.champ.rtr=1;
T_IRM_ASV.ident.extend.identificate   ident=Ide   T_IRM1_Asservissement;
while(1)            //Task actives pe
          {
          EcrireTrame(&T_IRM_ASV);
          dort(3);
          }
}

/*****************************************************************************************************************/
/*                                                          Reading received frames
                                                */
/*****************************************************************************************************************/
TACHE LectureTrame(void)
{
static Trame tr_rx;
static Valeur_Commodo_EG_Mem; //static variable as it is a memorization
static Valeur_FC_EG_Mem;
static Valeur_Analogique_Mem;
static Actu_Aff,Vitesse_EG;//defined in static because it is a parameter to transmit
static Tempo_EG;
while(1)                                //Task actives permanently
          {
          if(LireTrame(&tr_rx))  //LireTrame stop the task when there aren't any received frame
             {
             Actu_Aff=0;
             if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM8_Commodo_EG) //we received a stalk response
                    {
                    Valeur_Commodo_EG=~(tr_rx.data[1]);     //recover the status of windshield wiper stalk
                    Valeur_Analogique=tr_rx.data[2];        //recover the value of potar

                    //we determine the potar position to activate the delay
                    if(Valeur_Analogique!=Valeur_Analogique_Mem)
                              {
                              Actu_Aff=1;
                              Valeur_Analogique_Mem=Valeur_Analogique;
```

```
                              if(Valeur_Analogique>=200)
                                        {
                                        detruit(num_tache(irq_eg));
                                        Tempo_EG=0;
                                        }
                              else
                                        {
                                        detruit(num_tache(irq_eg));
                                        Vitesse_EG=Vitesse_Lente; //  "Intermittent" Position
                                        if(Valeur_Analogique>=150)Tempo_EG=Tempo5;// Depending on the position of the wheel
                                                  else if(Valeur_Analogique>=140)Tempo_EG=Tempo4;// the long or short delay
                                                            else if(Valeur_Analogique>=120)Tempo_EG=Tempo3;
                                                                      else if(Valeur_Analogique>=90)Tempo_EG=Tempo2;
                                                                                else if(Valeur_Analogique==0)Tempo_EG=Tempo1;
                                        m_send(num_tache(irq_eg),&Tempo_EG);
                                        }
                              }

                    if(Valeur_Commodo_EG_Mem!=Valeur_Commodo_EG)
                              {
                              Actu_Aff=1;
                              Valeur_Commodo_EG_Mem=Valeur_Commodo_EG;
                              Vitesse_EG=0;
                              //Disable the intermittent position if it was chosen
                              if(Valeur_Analogique<200) detruit(num_tache(irq_eg));
                              //...Handle the selected position...
                              if(Cde_EG_Av_Pos2) Vitesse_EG=Vitesse_Rapide;
                              else if(Cde_EG_Av_Pos1) Vitesse_EG=Vitesse_Lente;
                              else if(Cde_Lave_Glace_Av==0) Vitesse_EG=Vitesse_Lave_Glace; //inversion stalk
                              //...and reactivate it if it is still actual
                              else if(Cde_EG_Av_Int && (Valeur_Analogique<200))
                                        {
                                        Vitesse_EG=Vitesse_Lente;
                                        m_send(num_tache(irq_eg),&Tempo_EG);
                                        }
                              m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                              }
                    } //End of stalk handling

           if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM8_Asservissement) // received a servo-system response
                    {
                    Valeur_FC_EG=~(tr_rx.data[0]);

                    if(Valeur_FC_EG_Mem!=Valeur_FC_EG)
                              {
                              Actu_Aff=2;
                              Valeur_FC_EG_Mem=Valeur_FC_EG;
                              m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                              }
                    }//End of limit switch handling

          if(Actu_Aff!=0)
                    active(num_tache(Affichage));
          } //End of received frame reading
        } //End of while
}

/*****************************************************************************************************************/
/*                                                                                  Interruption

/*****************************************************************************************************************/
TACHE irq_eg(Tempo)           //task called by frame reading
char Tempo;
{
static Vitesse;
static Valeur_Ana;
Flag_Fin_Tempo_EG=0;
Vitesse=Vitesse_Lente;
while((Valeur_Analogique<200) && (Cde_EG_Av_Pos2==0) && (Cde_EG_Av_Pos1==0) && (Cde_Lave_Glace_Av==1))
          {
          if(Flag_Fin_Tempo_EG==0)
                    {
                    dort(100*Tempo);
                    Flag_Fin_Tempo_EG=1;
                    m_send(num_tache(Actualise_ASV),&Vitesse);
                    }
          }
}

/*****************************************************************************************************************/
/*                                        Send a frame to Windshield Wiper Stalk board
                              */
/*****************************************************************************************************************/
TACHE Lecture_Commodo_EG(void)
{
Trame T_IRM_Commodo_EG;
T_IRM_Commodo_EG.trame_info.registre=0x00;
T_IRM_Commodo_EG.trame_info.champ.extend=1;
T_IRM_Commodo_EG.trame_info.champ.dlc=0x08;
T_IRM_Commodo_EG.trame_info.champ.rtr=1;
T_IRM_Commodo_EG.ident.extend.identificateur.ident=Ident_T_IRM8_Commodo_EG;
while(1)            //task actives permanently
          {
          EcrireTrame(&T_IRM_Commodo_EG);
          dort(10);
          }
}

/*****************************************************************************************************************/
/*                                                            Display task
                                        */
/*****************************************************************************************************************/
```

```
TACHE Affichage(void)
{
static Valeur_Commodo_EG_Aff;
static Valeur_Analogique_Aff;
static Valeur_FC_EG_Aff;

if(Valeur_Analogique_Aff!=Valeur_Analogique)
          {
          Valeur_Analogique_Aff=Valeur_Analogique;
          MONITOR             // we update the values of stalk on the Monitor
          gotoxy(45,4);
          printf("%d   \n",Valeur_Analogique);
          ENDM
          }

if(Valeur_Commodo_EG_Aff!=Valeur_Commodo_EG)
          {
          Valeur_Commodo_EG_Aff=Valeur_Commodo_EG;
          MONITOR             //we update the values of stalk on the Monitor
          gotoxy(22,4);
          printf("%d\n",Cde_EG_Av_Int);
          gotoxy(16,7);
          printf("%d\n",Entree1);
          gotoxy(39,7);
          printf("%d\n",Entree2);
          gotoxy(23,5);
          printf("%d\n",Cde_EG_Av_Pos1);
          gotoxy(53,5);
          printf("%d\n",Cde_EG_Av_Pos2);
          gotoxy(18,8);
          printf("%d\n",Cde_EG_Ar);
          //the windshield washer commands are inversed on the stalk (Active=0 , Inactive=1)
          gotoxy(27,9);
          printf("%d\n",(Cde_Lave_Glace_Ar^0x01));
          gotoxy(27,6);
          printf("%d\n",(Cde_Lave_Glace_Av^0x01));
          ENDM
          }
if(Valeur_FC_EG_Aff!=Valeur_FC_EG)
          {                   //we update the values of limit switch  the
          Valeur_FC_EG_Aff=Valeur_FC_EG;
          MONITOR
          if(fcg)   //End of left active progress
                    {
                    gotoxy(27,12);
                    printf("1\n");
                    }
          else
                    {
                    gotoxy(27,12);
                    printf("0\n");
                    }
          if(fcd)   //End of right active progress
                    {
                    gotoxy(27,14);
                    printf("1\n");
                    }
          else
                    {
                    gotoxy(27,14);
                    printf("0\n");
                    }
          if(fs)    //End of over li
                    {
                    gotoxy(27,13);
                    printf("1\n");
                    }
          else
                    {
                    gotoxy(27,13);
                    printf("0\n");
                    }
                    ENDM
          }

}


/*****************************************************************************************************************/
/*                                                                    Main Program
                                                  */
/*****************************************************************************************************************/
main()
{
clsscr();
start_mtr(Init_VMD,1024);
}
```


Sample

7 EX N°6 : VMD CONTROL WITH REAL TIME EXECUTIVE

7.1 Topic :

Purposes :	- Develop a complete real time application, at the same time including binary (on/off) sensors and analog sensors, analog pre-actuators and incorporating a speed variable function.
Specifications :	After a regular time interval, we can know its status by the module on which is connected the wiper stalk and lamps. According to the state of the wiper s[illegible] th[illegible]otor (intermittent, position 1, position 2, etc.) is controlled. According to the state of the [illegible]hts [illegible]control the different lamps in different optical blocks (indica[illegible]side [illegible]ht, dipped light, head light, stop lights) We wish to leave correct[illegible] th[illegible]og[illegible]m (press the EID210 CRTL key to turn off the optical blocks and to[illegible] [illegible]ndshield wiper)

Necessary h[illegible]ar[illegible]val[illegible]nd software :

PC Micro Computer using Windows [illegible]
Software: Editor -Assembler-Debug[illegible]er.
If programming in C, GNU; [illegible]C ++-C[illegible]Ref: EID210101
Processor board 16/32 bit 6[illegible]32[illegible]cro[illegible]contr[illegible]er and its software environment
(Editor-Cross Assembler-[illegible]bugger)[illegible]f. EID210001
CAN PC/104 Network board [illegible]ON[illegible]YSTEMS Ref NIC: EID004001
CAN network with :
- 2 modules of 8 logic inputs for stalk Ref : EID050001
- 4 modules of 4 power outputs for the 2 front/back lamps Ref : EID051001
- 1 Servo-system module for driving the windshield wiper Ref : EID053001

USB connection cable, or if not available use RS232 cable, Ref: EGD000003
AC / AC Power source 8V 1A Ref: EGD000001
12V Power source supply for the CAN modules ("energy" network)

Time : 3 x 4 hours

7.2 Solution elements

7.2.1 Remind

The program must control two independent processors:

-the "lights system" ordered by the "lights" stalk integrating the control bulbs,

-the "wiper system" ordered by the "windshield wiper" stalk.

We must also control a 3rd processor (depending on the first two) realizing the display.

7.2.2 Tasks management and organization

1- The VMD initialization task can initialize the CAN board and the Inputs/Outputs function, of the PWM output and of the analog/digital convertor of the different CAN modules (Stalks, Servo-system, Lights,). It will activate the initialization task.

Remark : 1- We will be able to make the configuration through the table with a loop.

2- This task can anticipate on the display; by putting in the text, the display task will only have to position for displaying the stalk's and limit switch's states.

2- The initialization task will initialize the CAN board and create the following tasks.

3-The servo-system task should control the motor rotation according to the stalk position, the asked speed and the limit switch.

4- The stalk interrogation task should send a remote frame (IRM) to the stalk after a regular time interval.

5-The EcrireFeux task will send the frame on the CAN network which can turn on or turn off the optical block bulbs (IM type frame).

6-The Frame Reading task will be permanently active. It will receive frames, manage variables updating and awaken the appropriate tasks.

7-The windshield wiper interrupt task will handle the intermittent position. It will activate or inhibit a flag indicating the end of the delay, and activate the task controlling the motor rotation.

8- The lights interrupt task will handle the indicators. It will have to be able to activate the task controlling the writing on the lights, after modifying the transferred parameter to turn off or turn on the adequate indicators.

9- The lights stalk interrogation task will have to send a remote frame (IRM) to the stalk after a regular time interval. It will fall asleep to leave other tasks run and to not block the bus.

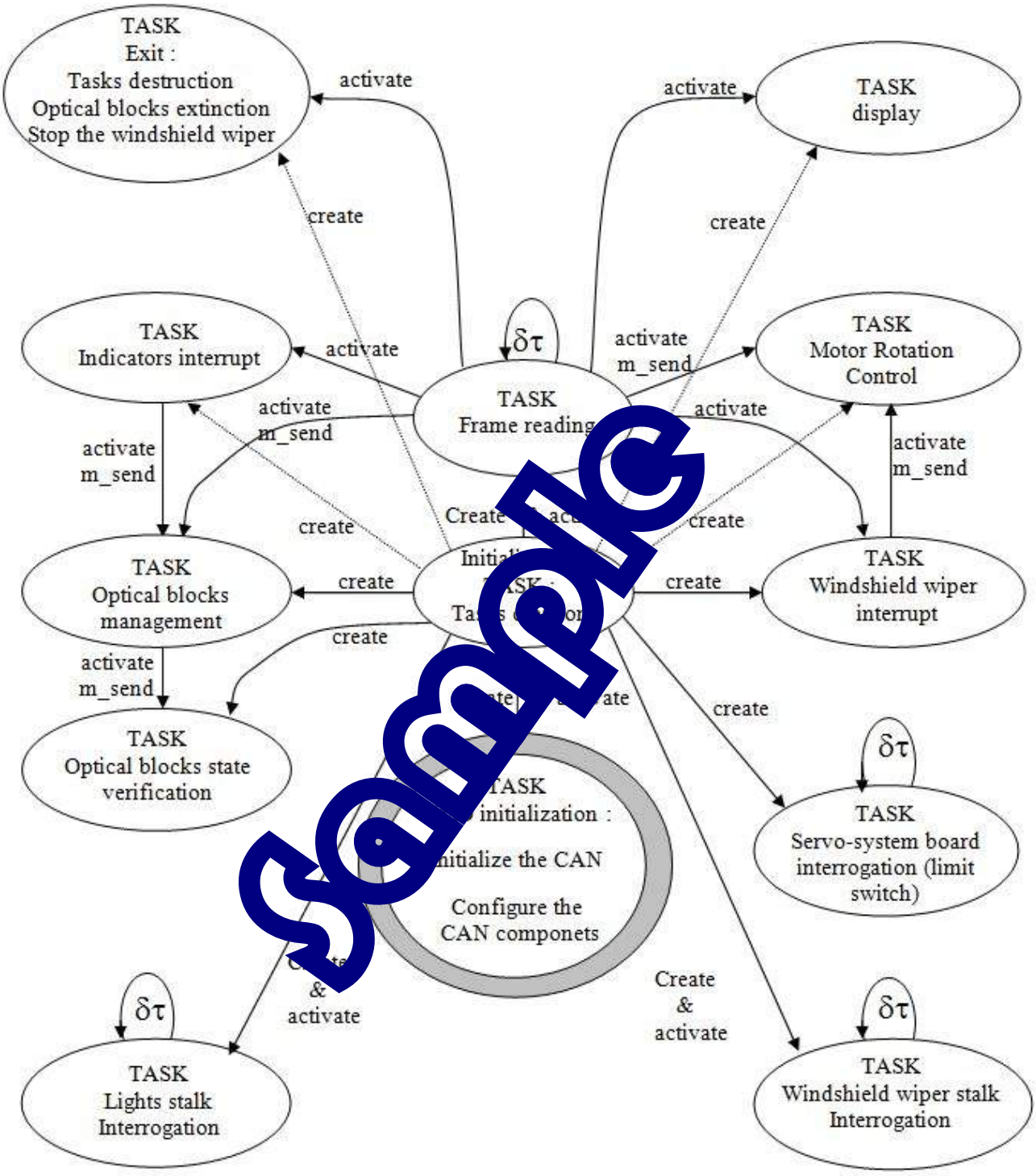
10- The windshield wiper stalk interrogation task will have to send a remote frame (IRM) to the stalk after a regular time interval. It will fall asleep to leave other tasks run and to not block the bus.

11- The State verification task will send the frame on the CAN network can know the real state of the optical block (IRM type frame).

12-The display task will build a report about the state of the stalks, the limit switches, and the optical blocks.

13- The Exit task can completely leave the program (press on the CRTL key to activate this task)

7.2.3 State diagram



7.2.4 C Program

```
/* VMD management: optical blocks, Lights Stalk, Windshield Wiper and its Stalk*/
//***************************
#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"
#include "eid210.h"

//Global Variables
//For the Windshield Wiper delay
#define Tempo1 2    // in second
#define Tempo2 4 // For intermittent mode
#define Tempo3 6
#define Tempo4 8
#define Tempo5 10
//Flag of Windshield Wiper delay end
char Flag_Fin_Tempo_EG;
//For the beat rate
#define Vitesse_Rapide 0x90
#define Vitesse_Lente 0x40
#define Vitesse_Lave_Glace 0x60
//For the delay between two indicator's flashing
#define TempoClign 6          //in ms


//Tasks Prototype
TACHE  Init_VMD(void);
TACHE  init  (void);
TACHE      Actualise_ASV();
TACHE  Interroge_FDC(void);
TACHE      Actualise_Etat_Feux();
TACHE  LectureTrame(void);
TACHE  irq_Clign(void);
TACHE  irq_eg();
TACHE  Interroge_Commodo_Feux(void);
TACHE  Interroge_Commodo_EG(void);
TACHE  Verif_Etat_Feux(void);
TACHE  Affichage(void);
TACHE  Quit(void);


TACHE Init_VMD(void)
{
//Modules Configuration
Trame T_IM_CFG,T_IRM_Etat_FC,Trame_Recue;
//For the waiting delays between two sends
int Temp,Cptr_TimeOut;
//put the identifiers, value,...
int cpt;
//Identifiers,values,data:
//identifier
int
identificateur[]={Ident_T_IM_Commodo_Feux,Ident_T_IM_                    _T_IM_Asservissement,Ident_T_IM_FRG,Ident_T_IM_FRD,Ident
_T_IM_FVG,Ident_T_IM_FVD};
//Acknowledge
int
identificateurACK[]={Ident_T_AIM_Commodo_Feu      dent_T_AI                 do_EG,Ident_T_AIM_Asservissement,Ident_T_AIM_FRG,Ident_T_AIM_F
RD,Ident_T_AIM_FVG,Ident_T_AIM_FVD};
//Table including the data for I /      figu ti
int Config_E_S[]={0xFF,0xFF,0xE3,0      ,0x      ,0x      ,

//Strings can indicate module ini    liza
char *msg[]={" Light stalk"," Wind     eld Wipe      talk "," Servo-system Module ","Left Back Lights","Right Back Lights","Left
Front Lights","Right Front Lights"};

//Table including all the data.
//The 2 first values are for the Analog/D      al  nvertor of the wiper stalk wheel.
//The following are for the Servo-system (Cr      , frequency, Power Validation,...)
int datazero[]={0x2A,0x2B,0x21,0x23,0x22,0x24,0x25,0x26,0x1E};
int dataun[]={0xF0,0xFF,0xB3,0xFF,0xB3,0xFF,0xFF,0xFF,0x10};
int datadeux[]={0x80,0x0E,0x80,0xFF,0x80,0xFF,0x00,0x00,0x10};

//CAN and MTR configuration
dsp_stk(); //Know the state of the tasks when we leave through mtr86exit(0);
OpenCAN(ATON_V2);   //ATON_V1 for old ATONCAN boards
                            //ATON_V2 for new ATONCAN boards

T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1; // Work in the extended mode
T_IM_CFG.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
// For the 4 outputs modules configuration (GP0 to GP3) and 4inputs status 4 (GP4 to GP8)
T_IM_CFG.data[0]=0x1F;        // first data -> "Address" of concerned register-> GPDDR
T_IM_CFG.data[1]=0xFF;        // second data -> "Mask"-> See Doc MCP25050 page 16


//Configuration of modules to address and operation of their I/O
for(cpt=0;cpt<7;cpt++)
{
T_IM_CFG.ident.extend.identificateur.ident=identificateur[cpt]; //establishment of the identifier
T_IM_CFG.data[2]=Config_E_S[cpt];      // third data-> "Value"

MONITOR
gotoxy(2,9);       //we display a message to say that which module is being initialized
printf("Initialization of %s\n",msg[cpt]);

do        {
          EcrireTrame(&T_IM_CFG);// Send a first frame on the CAN network
          Cptr_TimeOut=0;     //The LireTrame function let the task sleep if there aren't any received frames.
```

```
          do{Cptr_TimeOut++;}while((LireTrame(&Trame_Recue)==0)&&(Cptr_TimeOut<200));
          if(Trame_Recue.ident.extend.identificateur.ident==identificateurACK[cpt])Cptr_TimeOut=200; // Test if the identifier
is correct
                    else
                    {
                    printf("Problem\n");
                    Cptr_TimeOut=0;
                    }
          for(Temp=0;Temp<100000;Temp++); // To wait for a while!
          }while(Cptr_TimeOut!=200);
printf("OK\n");
ENDM
}

//Frame configuration
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1;
T_IM_CFG.trame_info.champ.dlc=0x03;

//configuration of Windshield Wiper stalk Ana -> Dig converter
MONITOR
gotoxy(2,9);        //we display a message to say that which module is being configured
printf("Configuration of the  A/D conversion of %s ...\n",msg[1]);
ENDM
T_IM_CFG.ident.extend.identificateur.ident=identificateur[1];
for(cpt=0;cpt<2;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];         // Register address
T_IM_CFG.data[1]=dataun[cpt];           // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];         // Value
EcrireTrame(&T_IM_CFG);
while((LireTrame(&Trame_Recue)==0));
}

//Specific Configuration of Servo-system module
MONITOR
gotoxy(2,9);        //we display a message to say that which module is being c
printf("Specific Configuration of %s ...           \n",msg[2]);
ENDM
T_IM_CFG.ident.extend.identificateur.ident=identificateur[2];
for(cpt=2;cpt<9;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];         // Register address
T_IM_CFG.data[1]=dataun[cpt];           // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];         // Value
EcrireTrame(&T_IM_CFG);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the response
}

//Take the original position
MONITOR
printf("Take the wiper POM \n");
ENDM
//Configuration of limit switch interrogation
T_IRM_Etat_FC.trame_info.registre=0x00;
T_IRM_Etat_FC.trame_info.champ.extend=1; // Work in
T_IRM_Etat_FC.trame_info.champ.dlc=0x01; // There is
T_IRM_Etat_FC.trame_info.champ.rtr=1;  //in return
T_IRM_Etat_FC.ident.extend.identificateur.ident=
EcrireTrame(&T_IRM_Etat_FC);
do{}while(LireTrame(&Trame_Recue)==0); // Wa
Valeur_FC_EG=~(Trame_Recue.data[0]);

if(!(fcd==0 && fcg==1))
          {
          //Configuration of moto
          T_IM_CFG.trame_info.regi
          T_IM_CFG.trame_info.champ
          T_IM_CFG.trame_info.champ
          // For the 4 outputs modules
          T_IM_CFG.data[0]=0x26;
          T_IM_CFG.data[1]=0xFF;
          T_IM_CFG.data[2]=0x40;
          EcrireTrame(&T_IM_CFG);//active negative speed
          while(!(fcd==0 && fcg==1))
                    {
                    EcrireTrame(&T_IRM_Etat_FC);
                    while(LireTrame(&Trame_Recue)==0){}; //Wait for the response
                    Valeur_FC_EG=~(Trame_Recue.data[0]); // Recover the limit switch state
                    }
          T_IM_CFG.data[2]=0; //cancel the negative speed
          EcrireTrame(&T_IM_CFG); //disable the negative control
          } //End of Servo-system module initialization
EcrireTrame(&T_IRM_Etat_FC);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the response
Valeur_FC_EG=~(Trame_Recue.data[0]);

MONITOR
clsscr();
//The display takes a lot of time, so we anticipate
gotoxy(1,1);
printf("************************EX_6 CAN bus with mtr86************************\n");
printf(" Windshield Wiper Stalk Status                                                                    \n");
printf("  GP0: Cde_EG_Av_Int=         , Valeur Potar=                                                    \n");
printf("  GP3: Cde_EG_Av_Pos1=        , GP4: Cde_EG_Av_Pos2=                    \n");
printf("  GP7: Cde_Lave_Glace_Av=                                                                        \n");
printf("  GP1: Input1=               , GP2: Input2=                            \n");
printf("  GP5: Cde_EG_Ar=             , GP6: Cde_Lave_Glace_Ar=                \n");
printf("
\n");
printf(" Lights stalk status                    \n");
printf("  GP0: Side light=      ,GP2: Head light=        ,GP3: Dipped light=                  \n");
```

```
printf("  GP1: Warning=        ,GP4: Left Indicator=   ,GP5: Right Indicator=                \n");
printf("  GP6: Stop light=      ,GP7: Horn=                                        \n");
printf("                                                                                    \n");
printf(" Limit Switch State on the Servo-system board                                \n");
printf("  Limit Switch: Left=            Over-limit=          Right=              \n");
printf("                                                                                    \n");
printf(" Left Back Lights:   Action        RealStatus                               \n");
printf("  Side light                                                                        \n");
printf("  Stop light                                                                      \n");
printf("  Indicator                                                                 \n");
printf("  Horn                                                                              \n");
printf(" Right Back Lights:   Action        RealStatus                               \n");
printf("  Side light                                                                        \n");
printf("  Stop light                                                                      \n");
printf("  Indicator                                                                 \n");
printf(" Left Front Lights:   Action        RealStatus                               \n");
printf("  Side light                                                                        \n");
printf("  Dipped light                                                                    \n");
printf("  Head light                                                                      \n");
printf("  Indicator                                                                 \n");
printf(" Right Front Lights:   Action        RealStatus                                  \n");
printf("  Side light                                                                        \n");
printf("  Dipped light                                                                    \n");
printf("  Head light                                                                      \n");
printf("  Indicator                                                                 \n");
ENDM

active(cree(init,1,1024)); //Tasks Initialization
}

TACHE init(void)
{
cree(Actualise_ASV,1,256);             //Priority n°1 because windshield wiper management and
active(cree(Interroge_FDC,2,256));     //limit switch state are the most importa
cree(Actualise_Etat_Feux,1,256);                //Priority n°1 because lights  evatio s important
active(cree(LectureTrame,2,256));               //frame reading will automati  sleep
cree(irq_Clign,2,256);        //The IT control for the indicators
cree(irq_eg,2,256); //The IT control for the windshield wiper
active(cree(Interroge_Commodo_Feux,3,256));     //Read the state of  h sta
active(cree(Interroge_Commodo_EG,3,256));       //Read the state o e w  wipe alk
cree(Verif_Etat_Feux,4,256);                    //Lights state veri ion  tant
cree(Affichage,5,1024);                                //The displa ains  owest priority
cree(Quit,6,256);                      //The task will disable other tas  lea  mpletely
}
/*****************************************************************************  ***************************/
/*                        Control the direction of motor rota n
                */
/*****************************************************************************  ***************************/

TACHE Actualise_ASV(Vitesse_EG)                 //task call ling
char Vitesse_EG;
{
//Limit switches' status (at t-1) to restore wiper when  o limi activated
static Valeur_FC_EG_Verif_fs;
//Frames to drive the motor rotation direction
Trame T_IM_Cmde_Negative,T_IM_Cmde_Positive;
//Configuration of Negative command frame
T_IM_Cmde_Negative.trame_info.registre=0x00;
T_IM_Cmde_Negative.trame_info.champ.extend=1; // W  e ex  ed mode
T_IM_Cmde_Negative.trame_info.champ.dlc=0x03;  of 8 bits (3 sent bytes)
T_IM_Cmde_Negative.ident.extend.identificateu ident=Id  Asservissement;
T_IM_Cmde_Negative.data[0]=0x26;       // PW DC  ster  ess
T_IM_Cmde_Negative.data[1]=0xFF;       // M  -  e  s are concerned
//Configuration of Positive comman
T_IM_Cmde_Positive.trame_info.regi e=
T_IM_Cmde_Positive.trame_info.cha ext  W  he extended mode
T_IM_Cmde_Positive.trame_info.cha dlc=0  are 3 data of 8 bits (3 sent bytes)
T_IM_Cmde_Positive.ident.extend.ide ficateur. t=Ident_T_IM_Asservissement;
T_IM_Cmde_Positive.data[0]=0x25;       // M2DC egister address
T_IM_Cmde_Positive.data[1]=0xFF;       //  k -  ll the bits are concerned

if((Cde_EG_Av_Pos2 || Cde_EG_Av_Pos1 || (Cd  Glace_Av==0))) //a "classic" command is valid
        {
        if(fcd)                                 //If it's at right side, return to left
                {
                T_IM_Cmde_Negative.data[2]=Vitesse_EG;
                T_IM_Cmde_Positive.data[2]=0x00;
                }
        else if(fcg)                            //If it's at left side, return to right
                {
                T_IM_Cmde_Negative.data[2]=0x00;
                T_IM_Cmde_Positive.data[2]=Vitesse_EG;
                }
        EcrireTrame(&T_IM_Cmde_Positive);
        EcrireTrame(&T_IM_Cmde_Negative);
        }
else if(Valeur_Analogique<200) //intermittent control is activated, we must manage according to the delay
        {
        if(fcd) //we are at right, it passes to the left
                {
                T_IM_Cmde_Negative.data[2]=Vitesse_EG;
                T_IM_Cmde_Positive.data[2]=0x00;
                Flag_Fin_Tempo_EG=0;                                    //we cancel the flag of delay end
                }
        else if ((Flag_Fin_Tempo_EG)) //the delay is finished
                {
                T_IM_Cmde_Negative.data[2]=0x00;
                T_IM_Cmde_Positive.data[2]=Vitesse_EG;  //we launch the motor
                }
        else if (fcg && Flag_Fin_Tempo_EG==0)
                { //at left, in intermittent position, but the delay isn't finished
                T_IM_Cmde_Negative.data[2]=0x00;
```

```
                    T_IM_Cmde_Positive.data[2]=0x00;       //we stop the motor
                    }
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fcd) //it's at the right without any command, return to left
          {
          T_IM_Cmde_Negative.data[2]=0x40;
          T_IM_Cmde_Positive.data[2]=0x00;
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fcg) //it's at the left without any command, return to right
          {
          T_IM_Cmde_Negative.data[2]=0x00;
          T_IM_Cmde_Positive.data[2]=0x00;
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
if(fs)//error, it's over limit
          {
          if((Valeur_FC_EG_Verif_fs&0x40)==0x40) //for the left, return until the left limit switch
                    {
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=0x40;
                    EcrireTrame(&T_IM_Cmde_Negative);
                    EcrireTrame(&T_IM_Cmde_Positive);
                    }
          if((Valeur_FC_EG_Verif_fs&0x20)==0x20)  //for the right, return until the right limit switch
                    {
                    T_IM_Cmde_Negative.data[2]=0x40;
                    T_IM_Cmde_Positive.data[2]=0x00;
                    EcrireTrame(&T_IM_Cmde_Negative);
                    EcrireTrame(&T_IM_Cmde_Positive);
                    }
          }
Valeur_FC_EG_Verif_fs=Valeur_FC_EG;
}
/**********************************************************************************************************************/
/*                              Question the Servo-system board to kn  he    tches'  ate                          */
/**********************************************************************************************************************/
TACHE Interroge_FDC(void)
{
Trame T_IRM_ASV;
T_IRM_ASV.trame_info.registre=0x00;
T_IRM_ASV.trame_info.champ.extend=1;
T_IRM_ASV.trame_info.champ.dlc=0x08;
T_IRM_ASV.trame_info.champ.rtr=1;
T_IRM_ASV.ident.extend.identificateur.ident=Ident_T_IRM1_Asservis  ent;
while(1)            //Task actives permanently
          {
          EcrireTrame(&T_IRM_ASV);
          dort(2); //For the maximal speed
          }
}
/**********************************************************************************************************************/
/*                  Turn on the lights                                                   /
/**********************************************************************************************************************/
TACHE Actualise_Etat_Feux(Value_Commodo)                  //ta  called by reading frame
Port_8ES Value_Commodo; //the variable is unde          ucture of Port_8ES type
                                        //whi  allows         ct access to the bit level
{
Trame T_IM_Feux,T_recue; //Frame tha  ill  set     ad  ss the optical blocks
Valeur_FRG=0x00;    //sets optical       to
Valeur_FRD=0x00;
Valeur_FVG=0x00;
Valeur_FVD=0x00;

if(Value_Commodo.bit.GP2) //Head lig
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_P;
          Valeur_FVD|=Cde_FV_P;
          }
else if(Value_Commodo.bit.GP3) //Dipped light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_C;
          Valeur_FVD|=Cde_FV_C;
          }
else if(Value_Commodo.bit.GP0) //Side light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_V;
          Valeur_FVD|=Cde_FV_V;
          }

if(Value_Commodo.bit.GP1)     //Warning
          {
          Valeur_FRG|=Masque_Clign_AR;
          Valeur_FRD|=Masque_Clign_AR;
          Valeur_FVG|=Masque_Clign_AV;
          Valeur_FVD|=Masque_Clign_AV;
          }
          else
          {
          if(Value_Commodo.bit.GP4) //Left Indicator
                    {
                    Valeur_FRG|=Masque_Clign_AR;
```

```
                    Valeur_FVG|=Masque_Clign_AV;
                    }
          if(Value_Commodo.bit.GP5) //Right indicator
                    {
                    Valeur_FRD|=Masque_Clign_AR;
                    Valeur_FVD|=Masque_Clign_AV;
                    }
          }

if(Value_Commodo.bit.GP6)//Stop light
          {
          Valeur_FRG|=Masque_Stop;
          Valeur_FRD|=Masque_Stop;
          }

if(Value_Commodo.bit.GP7) //Horn
          Valeur_FRG|=Masque_Klaxon;

//initialization of the command frame on the lights
T_IM_Feux.trame_info.registre=0x00;
T_IM_Feux.trame_info.champ.extend=1;
T_IM_Feux.trame_info.champ.dlc=0x03;
T_IM_Feux.data[0]=0x1e;
T_IM_Feux.data[1]=0x0f;

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FRG;
T_IM_Feux.data[2]=Valeur_FRG;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FRD;
T_IM_Feux.data[2]=Valeur_FRD;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FVG;
T_IM_Feux.data[2]=Valeur_FVG;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FVD;
T_IM_Feux.data[2]=Valeur_FVD;
EcrireTrame(&T_IM_Feux);

//we just made a lights modification request  .
active(num_tache(Verif_Etat_Feux)); // we check the lights state
}
/*****************************************************************************************************************/
/*                  Reading received frames
          */
/*****************************************************************************************************************/
TACHE LectureTrame(void)
{
static Trame tr_rx;
static Valeur_Commodo_Feux_Mem;        //static variable as it is
static Valeur_Commodo_EG_Mem; //static variable as it is
static Valeur_FC_EG_Mem;
static Actu_Aff;
static Valeur_Analogique_Mem; //defined in static because it is a parameter to transmit
static Vitesse_EG;
static Tempo_EG;
while(1)                                //Task
          {
          if(LireTrame(&tr_rx))  //LireTrame
             {
          if(CRTL==0) active(num_tach
             Actu_Aff=0;
             if (tr_rx.ident.exten
response
                    {
                    Valeur_Commodo_
                    Valeur_Analogique

                    //we determine the
                    if(Valeur_Analogique!=
                              {
                              Actu_Aff=1;
                              Valeur_Analogique_Mem=Valeur_Analogique;
                              if(Valeur_Analogique>=200)
                                        {
                                        detruit(num_tache(irq_eg));
                                        Tempo_EG=0;
                                        }
                              else
                                        {
                                        detruit(num_tache(irq_eg));
                                        Vitesse_EG=Vitesse_Lente; // "Intermittent" Position
                                        if(Valeur_Analogique>=150)Tempo_EG=Tempo5;// Depending on the position of the wheel
                                                  else if(Valeur_Analogique>=140)Tempo_EG=Tempo4;// the long or short delay
                                                            else if(Valeur_Analogique>=120)Tempo_EG=Tempo3;
                                                                      else if(Valeur_Analogique>=90)Tempo_EG=Tempo2;
                                                                                else
if(20<=Valeur_Analogique==0)Tempo_EG=Tempo1;
                                        Flag_Fin_Tempo_EG=0;
                                        m_send(num_tache(irq_eg),&Tempo_EG);
                                        }
                              }

                    if(Valeur_Commodo_EG_Mem!=Valeur_Commodo_EG)
                              {
                              Actu_Aff=1;
                              Valeur_Commodo_EG_Mem=Valeur_Commodo_EG;
                              Vitesse_EG=0;
                              //Disable the intermittent position if it was chosen
                              if(Valeur_Analogique<200) detruit(num_tache(irq_eg));
```

```
                            //...Handle the selected position...
                            if(Cde_EG_Av_Pos2) Vitesse_EG=Vitesse_Rapide;
                            else if(Cde_EG_Av_Pos1) Vitesse_EG=Vitesse_Lente;
                            else if(Cde_Lave_Glace_Av==0) Vitesse_EG=Vitesse_Lave_Glace;
                            //...and reactivate it if it is still actual
                            else if(Cde_EG_Av_Int && (Valeur_Analogique<200))
                                    {
                                    Vitesse_EG=Vitesse_Lente;
                                    m_send(num_tache(irq_eg),&Tempo_EG);
                                    }
                            m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                            }
                    } //End of stalk handling

             if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM1_Asservissement) //we received a servo-system response
                    {
                    Valeur_FC_EG=~(tr_rx.data[0]);
                            if(Valeur_FC_EG_Mem!=Valeur_FC_EG)
                            {
                            Valeur_FC_EG_Mem=Valeur_FC_EG;
                            m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                            Actu_Aff=2;
                            }
                            //else if(fs==1) m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                    }//End of limit switch handling

             if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_Commodo_Feux) //we received a stalk response
                    {
                    Valeur_Commodo_Feux=~(tr_rx.data[0]);  //Bitwise inversion to the stalk real state
                    if(Valeur_Commodo_Feux_Mem!=Valeur_Commodo_Feux)
                            {//Its state changed
                            Valeur_Commodo_Feux_Mem=Valeur_Commodo_Feux; //we  update the stored value
                            //If a indicator is active, we validate the IT which will activate Actualise_Etat_Feux
                            if(Cde_Clign_Gauche || Cde_Clign_Droit || Cde_Warnin[illegible]e(num_tache(irq_Clign));
                            //Otherwise, we wake up the lights state actualisa[illegible] tas[illegible]th the stalk without indicator
value
                                    else m_send(num_tache(Actualise_Etat_F[illegible])[illegible]modo_Feux_Mem);
                            //we activate the display task
                            Actu_Aff=3;
                            }
                    } //End of stalk handling

            if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FRG)
                    {
                    Valeur_Status_FRG=tr_rx.data[0];       //Recov[illegible]he sta[illegible] most significant bits of data[0]
                    Valeur_FRG=tr_rx.data[0];              //Reco[illegible] th[illegible]ske[illegible]ta[illegible] in the least significant bits of data[0]
                    //we activate the display task
                    Actu_Aff=4;
                    } //End of Left Back Lights handling

            if (tr_rx.ident.extend.identificateur.ident==I[illegible]
                    {
                    Valeur_Status_FRD=tr_rx.data[0];       //[illegible]ver [illegible] the most significant bits of data[0]
                    Valeur_FRD=tr_rx.data[0];              //[illegible] t[illegible]aske[illegible]state in the least significant bits of data[0]
                    //we activate the display task
                    Actu_Aff=5;
                    } //End of Right Back Lights hand[illegible]ng

            if (tr_rx.ident.extend.identificateur.id[illegible]ent_[illegible]RM_FVG)
                    {
                    Valeur_Status_FVG=tr_rx.d[illegible][0];       [illegible]over the state in the most significant bits of data[0]
                    Valeur_FVG=tr_rx.data[0][illegible]            [illegible]ecover the asked state in the least significant bits of data[0]
                    //we activate the displa[illegible]tas[illegible]
                    Actu_Aff=6;
                    }  //End of Le[illegible]Fro[illegible]i[illegible]ts handl[illegible]

            if (tr_rx.ident.extend[illegible]enti[illegible][illegible]==Ident_T_IRM_FVD)
                    {
                    Valeur_Status_FV[illegible]data[illegible];       //Recover the state in the most significant bits of data[0]
                    Valeur_FVD=tr_rx.data[illegible]                  //Recover the asked state in the least significant bits of data[0]
                    //we activate the di[illegible]ta[illegible]
                    Actu_Aff=7;
                    } //End of Right Front Li[illegible]s handling
             if(Actu_Aff!=0) active(num_tache(Affichage));
             } //End of Received frames reading
          } //End of while
}
/*****************************************************************************************************************/
/*                          Interruption
                            */
/*****************************************************************************************************************/
TACHE irq_Clign(void)
{
int actualise;
static char Flag_Fin_Tempo_Clign;

while((Cde_Clign_Gauche || Cde_Clign_Droit || Cde_Warning))
          {
          dort(10*TempoClign);                              //Wait for 0.8s
          Flag_Fin_Tempo_Clign^=0x01;   //Enable or disable the flag
//Pay attention to send a command, it checks at first if it is a indicator to show the Warning
          if(((Cde_Clign_Gauche==0) && (Cde_Clign_Droit==0) && (Cde_Warning==0))) break; //If command dropped->we leave
          if(Flag_Fin_Tempo_Clign==0x01) //End of the indicator delay
                    { //for extinction
                    if(Cde_Clign_Droit) actualise=((Valeur_Commodo_Feux^0x20)&0xFD); //Turn off the right indicator and
Warning
                    else if(Cde_Clign_Gauche) actualise=((Valeur_Commodo_Feux^0x10)&0xFD); //Turn off the left indicator and
Warning
                    else if (Cde_Warning ) actualise=Valeur_Commodo_Feux^0x02; //Warning
                    m_send(num_tache(Actualise_Etat_Feux),&actualise);
                    }
          else if(Flag_Fin_Tempo_Clign==0x00) //End of the indicator delay
```

```
                    { //to turn on
                    if (Cde_Clign_Droit) actualise=Valeur_Commodo_Feux&0xFD; //turn on right indicator and turn off the
warning
                    else if (Cde_Clign_Gauche) actualise=Valeur_Commodo_Feux&0xFD; //turn on left indicator and turn off
warning
                    else if (Cde_Warning ) actualise=Valeur_Commodo_Feux; //turn on Warning
                    m_send(num_tache(Actualise_Etat_Feux),&actualise);
                    }
          }

}

TACHE irq_eg(Tempo)          //task called by reading frame
char Tempo;
{
static Vitesse;
static Valeur_Ana;
Vitesse=Vitesse_Lente;
while((Valeur_Analogique<200) && (Cde_EG_Av_Pos2==0) && (Cde_EG_Av_Pos1==0) && (Cde_Lave_Glace_Av==1))
          {
          if(Flag_Fin_Tempo_EG==0)
                    {
                    dort(100*Tempo);
                    Flag_Fin_Tempo_EG=1;
                    m_send(num_tache(Actualise_ASV),&Vitesse);
                    }
          }
}
/*****************************************************************************************************************/
/*                                    Read the lights stalk                                                      */
/*****************************************************************************************************************/
TACHE Interroge_Commodo_Feux(void)
{
Trame T_IRM_Commodo_Feux;
T_IRM_Commodo_Feux.trame_info.registre=0x00;
T_IRM_Commodo_Feux.trame_info.champ.extend=1;
T_IRM_Commodo_Feux.trame_info.champ.dlc=0x01;
T_IRM_Commodo_Feux.trame_info.champ.rtr=1;
T_IRM_Commodo_Feux.ident.extend.identificateur.ident=Ident_T_IRM_Com
while(1)            //Task activates permanently
          {
          EcrireTrame(&T_IRM_Commodo_Feux);
          dort(10);
          }
}
/*****************************************************************************************************************/
/*                    Question the Windshield Wiper Stalk board
          */
/*****************************************************************************************************************/
TACHE Interroge_Commodo_EG(void)
{
Trame T_IRM_Commodo_EG;
T_IRM_Commodo_EG.trame_info.registre=0x00;
T_IRM_Commodo_EG.trame_info.champ.extend=1;
T_IRM_Commodo_EG.trame_info.champ.dlc=0x08;
T_IRM_Commodo_EG.trame_info.champ.rtr=1;
T_IRM_Commodo_EG.ident.extend.identificateur.ident=I
while(1)            //Task actives permanently
          {
          EcrireTrame(&T_IRM_Commodo_EG);
          dort(10);
          }
}
/*****************************************************************************************************************/
/*                              Chec                                                                             */
/*****************************************************************************************************************/
TACHE  Verif_Etat_Feux(void)
{
//Left Front Lights state verificati                ini ization
Trame T_IRM_Feux;
T_IRM_Feux.trame_info.registre=0x00;
T_IRM_Feux.trame_info.champ.extend=1;
T_IRM_Feux.trame_info.champ.dlc=0x01;
T_IRM_Feux.trame_info.champ.rtr=1;
T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_IRM_FRG;
//Send the remote frame on the CAN bus
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_IRM_FRD;
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_IRM_FVG;
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_IRM_FVD;
EcrireTrame(&T_IRM_Feux);

//task infinite sleeping (activation on awake or m_send)
}
/*****************************************************************************************************************/
/*                              Display task       */
/*****************************************************************************************************************/
TACHE Affichage(void)
{
static Valeur_FRG_Aff,Valeur_FRD_Aff,Valeur_FVG_Aff,Valeur_FVD_Aff;
static Valeur_Commodo_Feux_Aff;
static Valeur_Commodo_EG_Aff;
static Valeur_Analogique_Aff;
static Valeur_FC_EG_Aff;

if(Valeur_Analogique_Aff!=Valeur_Analogique)
          {
```

```
          Valeur_Analogique_Aff=Valeur_Analogique;
          gotoxy(45,3);
          printf("%d   \n",Valeur_Analogique);
          }

if(Valeur_Commodo_EG_Aff!=Valeur_Commodo_EG)
//we update the values of windshield wiper stalk on the Monitor
          {
          Valeur_Commodo_EG_Aff=Valeur_Commodo_EG;
          MONITOR
          gotoxy(22,3);
          printf("%d\n",Cde_EG_Av_Int);
          gotoxy(16,6);
          printf("%d\n",Entree1);
          gotoxy(46,6);
          printf("%d\n",Entree2);
          gotoxy(23,4);
          printf("%d\n",Cde_EG_Av_Pos1);
          gotoxy(53,4);
          printf("%d\n",Cde_EG_Av_Pos2);
          gotoxy(18,7);
          printf("%d\n",Cde_EG_Ar);
          gotoxy(56,7);
          printf("%d\n",(Cde_Lave_Glace_Ar^0x01));
          gotoxy(27,5);
          printf("%d\n",(Cde_Lave_Glace_Av^0x01));
          ENDM
          }

if(Valeur_FC_EG_Aff!=Valeur_FC_EG)
//we update the values of limit switch on the Monitor
          {
          Valeur_FC_EG_Aff=Valeur_FC_EG;
          MONITOR
          gotoxy(26,15);
          if(fcg)   //End of left active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(62,15);
          if(fcd)   //End of right active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(46,15);
          if(fs)    //End of over limit
                    printf("1\n");
          else
                    printf("0\n");
                    ENDM
          }

if(Valeur_Commodo_Feux_Aff!=Valeur_Commodo_Feux)
//we update the values of lights stalk on the Monito
          {
          Valeur_Commodo_Feux_Aff=Valeur_Commodo_Feux
          MONITOR
          gotoxy(18,10);
          printf("%d\n",Cde_Veilleuse);
          gotoxy(36,10);
          printf("%d\n",Cde_Phare);
          gotoxy(55,10);
          printf("%d\n",Cde_Code);
          gotoxy(16,11);
          printf("%d\n",Cde_Warnin
          gotoxy(41,11);
          printf("%d\n",Cde_Clign_Ga
          gotoxy(60,11);
          printf("%d\n",Cde_Clign_Droit);
          gotoxy(18,12);
          printf("%d\n",Cde_Stop);
          gotoxy(37,12);
          printf("%d\n",Cde_Klaxon);
          ENDM
          }

if(Valeur_FRG_Aff!=Valeur_FRG)
//Left Back Optical Block status
          {
          Valeur_FRG_Aff=Valeur_FRG;
          MONITOR
                    if(Veilleuse_FRG)   //We ask for turning on the side light
                              {                   //We check that the state meets the demands
                              gotoxy(26,18);
                              printf("1\n");
                              gotoxy(40,18);
                              if(S_Veilleuse_FRG) printf("OK\n");    //State is 1, it's ok
                                        else printf("Out of service\n");//State is 0, the state doesn't meet the demands
                              }
                    else
                              {
                              gotoxy(26,18);
                              printf("0\n");
                              gotoxy(40,18);
                              printf("  \n");
                              }
                    if(Stop_FRG)        //We ask for turning on the stop light
                              {                   //We check that the state meets the demands
                              gotoxy(26,19);
                              printf("1\n");
```

```
                                gotoxy(40,19);
                                if(S_Stop_FRG) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,19);
                                printf("0\n");
                                gotoxy(40,19);
                                printf("  \n");
                                }
                        if(Clignot_FRG)     //We ask for turning on the indicator
                                {                   //We check that the state meets the demands
                                gotoxy(26,20);
                                printf("1\n");
                                gotoxy(40,20);
                                if(S_Clignot_FRG) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,20);
                                printf("0\n");
                                gotoxy(40,20);
                                printf("  \n");
                                }
                        if(Klaxon_FRG)      //We ask for turning on the horn
                                {                   //We check that the state meets the demands
                                gotoxy(26,21);
                                printf("1\n");
                                gotoxy(40,21);
                                if(S_Klaxon_FRG) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,21);
                                printf("0\n");
                                gotoxy(40,21);
                                printf("  \n");
                                }
                ENDM
                }

if(Valeur_FRD_Aff!=Valeur_FRD)
//Right Back Optical Block status
                {
                Valeur_FRD_Aff=Valeur_FRD;
                MONITOR
                        if(Veilleuse_FRD)   //We ask for turning on the side light
                                {                   //We check that the state meets the demands
                                gotoxy(26,23);
                                printf("1\n");
                                gotoxy(40,23);
                                if(S_Veilleuse_FRD) printf("OK\n");  //State is 1, it's ok
                                        else printf("Out of service\n");                //State is 0, the state doesn't meet
the demands
                                }
                        else
                                {
                                gotoxy(26,23);
                                printf("0\n");
                                gotoxy(40,23);
                                printf("  \n");
                                }
                        if(Stop_FRD)        //We ask for turning on the stop light
                                {                   //We check that the state meets the demands
                                gotoxy(26,24);
                                printf("1\n");
                                gotoxy(40,24);
                                if(S_Stop_FRD) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,24);
                                printf("0\n");
                                gotoxy(40,24);
                                printf("  \n");
                                }
                        if(Clignot_FRD)     //We ask for turning on the indicator
                                {                   //We check that the state meets the demands
                                gotoxy(26,25);
                                printf("1\n");
                                gotoxy(40,25);
                                if(S_Clignot_FRD) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,25);
                                printf("0\n");
                                gotoxy(40,25);
                                printf("  \n");
                                }
                ENDM
                }

if(Valeur_FVG_Aff!=Valeur_FVG)
//Left Front Optical Block status
                {
                Valeur_FVG_Aff=Valeur_FVG;
```

```
          MONITOR
                    if(Veilleuse_FVG)   //We ask for turning on the side light
                              {                   //We check that the state meets the demands
                              gotoxy(26,27);
                              printf("1\n");
                              gotoxy(40,27);
                              if(S_Veilleuse_FVG) printf("OK\n");    //State is 1, it's ok
                                        else printf("Out of service\n");                //State is 0, the state doesn't meet
the demands
                              }
                    else
                              {
                              gotoxy(26,27);
                              printf("0\n");
                              gotoxy(40,27);
                              printf("  \n");
                              }
                    if(Code_FVG)        //We ask for turning on the dipped light
                              {                   //We check that the state meets the demands
                              gotoxy(26,28);
                              printf("1\n");
                              gotoxy(40,28);
                              if(S_Code_FVG) printf("OK\n");
                                        else printf("Out of service\n");
                              }
                    else
                              {
                              gotoxy(26,28);
                              printf("0\n");
                              gotoxy(40,28);
                              printf("  \n");
                              }
                    if(Phare_FVG)       //We ask for turning on the head light
                              {                   //We check that the state meets
                              gotoxy(26,29);
                              printf("1\n");
                              gotoxy(40,29);
                              if(S_Phare_FVG) printf("OK\n");
                                        else printf("Out of service\n
                              }
                    else
                              {
                              gotoxy(26,29);
                              printf("0\n");
                              gotoxy(40,29);
                              printf("  \n");
                              }
                    if(Clignot_FVG)     //We ask for turning on t
                              {                   //We check that
                              gotoxy(26,30);
                              printf("1\n");
                              gotoxy(40,30);
                              if(S_Clignot_FVG) printf("O
                                        else printf("Ou
                              }
                    else
                              {
                              gotoxy(26,30);
                              printf("0\n");
                              gotoxy(40,30);
                              printf("  \n");
                              }
          ENDM
          }

if(Valeur_FVD_Aff!=Valeur_FVD)
//Right Front Optical Block status
          {
          Valeur_FVD_Aff=Valeur_FVD;
          MONITOR
          if(Veilleuse_FVD)   //We ask f
                    {                   //W
                    gotoxy(26,32);
                    printf("1\n");
                    gotoxy(40,32);
                    if(S_Veilleuse_FVD) printf("OK\n");    //State is 1, it's ok
                              else printf("Out of service\n");                //State is 0, the state doesn't meet the
demands
                    }
          else
                    {
                    gotoxy(26,32);
                    printf("0\n");
                    gotoxy(40,32);
                    printf("  \n");
                    }
          if(Code_FVD)        //We ask for turning on the dipped light
                    {                   //We check that the state meets the demands
                    gotoxy(26,33);
                    printf("1\n");
                    gotoxy(40,33);
                    if(S_Code_FVD) printf("OK\n");
                              else printf("Out of service\n");
                    }
          else
                    {
                    gotoxy(26,33);
                    printf("0\n");
                    gotoxy(40,33);
                    printf("  \n");
                    }
          if(Phare_FVD)       //We ask for turning on the head light
```

```
                    {                   //We check that the state meets the demands
                    gotoxy(26,34);
                    printf("1\n");
                    gotoxy(40,34);
                    if(S_Phare_FVD) printf("OK\n");
                              else printf("Out of service\n");
                    }
          else
                    {
                    gotoxy(26,34);
                    printf("0\n");
                    gotoxy(40,34);
                    printf("  \n");
                    }
          if(Clignot_FVD)     //We ask for turning on the indicator
                    {                   //We check that the state meets the demands
                    gotoxy(26,35);
                    printf("1\n");
                    gotoxy(40,35);
                    if(S_Clignot_FVD) printf("OK\n");
                              else printf("Out of service\n");
                    }
          else
                    {
                    gotoxy(26,35);
                    printf("0\n");
                    gotoxy(40,35);
                    printf("  \n");
                    }
          ENDM
          }
}
/*****************************************************************************************************************/
/*                                        Leave completely                                                       */
/*****************************************************************************************************************/
TACHE Quit(void)
{
//Stop th modules
Trame T_IM_Arret;
int Tempo;
//Destroy all tasks
detruit(num_tache(Actualise_ASV));
detruit(num_tache(Interroge_FDC));
detruit(num_tache(Actualise_Etat_Feux));
detruit(num_tache(LectureTrame));
detruit(num_tache(irq_Clign));
detruit(num_tache(irq_eg));
detruit(num_tache(Interroge_Commodo_EG));
detruit(num_tache(Verif_Etat_Feux));
detruit(num_tache(Affichage));
detruit(num_tache(Interroge_Commodo_Feux));

//General Frame Configuration
T_IM_Arret.trame_info.registre=0x00;
T_IM_Arret.trame_info.champ.extend=1;
T_IM_Arret.trame_info.champ.dlc=0x03;

//For the Servo-system board
 //In positive
T_IM_Arret.ident.extend.identificateur.ident=I                    ement;
T_IM_Arret.data[0]=0x25;
T_IM_Arret.data[1]=0xFF;
T_IM_Arret.data[2]=0x00;
EcrireTrame(&T_IM_Arret);
 //In negative
T_IM_Arret.data[0]=0x26;
EcrireTrame(&T_IM_Arret);

//For the Left Back Lights
T_IM_Arret.ident.extend.identificateur.id      Ide   T_IM_FRG;
T_IM_Arret.data[0]=0x1e;
T_IM_Arret.data[1]=0x0f;
T_IM_Arret.data[2]=0x00;
EcrireTrame(&T_IM_Arret);

//For the Right Back Lights
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_FRD;
EcrireTrame(&T_IM_Arret);

//For the Left Front Lights
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_FVG;
EcrireTrame(&T_IM_Arret);

//For the Right Front Lights
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_FVD;
EcrireTrame(&T_IM_Arret);

mtr86exit(0);
}
/*****************************************************************************************************************/
/*                                                  Main Program                                                 */
/*****************************************************************************************************************/
main()
{
clsscr();
start_mtr(Init_VMD,2000);
}
```

8 EX N°7: AUTOMATIC FRAME SENDING BY THE MCP25050

8.1 Topic :

Purposes :	- MCP25050 components configuration sothat it can automatically send a message.
Specifications :	After regular time intervals, or once its state change, we want to know the state of lights stalk and windshield wiper stalk as well as the servo-system board limit switches. We don't want to question the board but that it can automatically send back its state. We will display the received frames.

Necessary hardware and software :

PC Micro Computer using Windows
Editor Software-Assembler-Debugger
If programming in C, GNU; C + + Ref: EID210101
Processor board 16/32 bit 68332 microcontroller and its software environment
(Editor-Cross Assembler-Debugger) Ref: EID210001
CAN PC/104 Network board from ICSPNET SYSTEMS Ref NIC: EID004001
CAN network with:
two 8 logic inputs modules for the lights stalk Ref: EID050001
one 4 power outputs module for left/right back/front lights Ref: EID051001
USB connection cable, or if not available RS232 cable, Ref: EGD000003
AC / AC Power source 8V 1A Ref: EGD000001
12V Power source supply for the CAN modules ("energy" network)

Time : 4 hours

8.2 Solution elements

8.2.1 Analysis

Possible solutions, and a mode determination.

Two possibilities are offered with the MCP25050 circuit:

The first is to configure the component so that it can send back the GPLAT register state after a regular time interval**.**

The second is to configure the component so that it can send back the GPLAT register state during a state change (rising or falling front).

For the first function mode; the goal will be to do a sending every 1 second for the stalk and for the servo-system board (the period for the servo-system board will be minimized during the VMD complete function).

In our case, the second mode seems to be the most appropriate. However the inputs didn't treat the debounce, and the analog channel capturing is not possible. The first method's inconvenience will be corrected due to the front detection, but the second can't be corre[illegible].

8.2.2 Solution 1

We choose a MCP25050 frame sending after a regular time interval.

Concerned register

The configuration register is STCON. It is at the 0x10 address to which we must add the offset of 1C.
Register Composition:

STEN	STMS	STBF1	STBF0	STM3	STM2	STM1	STM0

STEN : Mode validation or inhibition
STMS : number of sent data (NL=1=>DLC=8, NL=0=>DLC=0).
STBFX and STMX: calculation of the time base (cf. datasheet p24).

Then we obtain the following code:

```
// Configuration Component for automatic sending frames...
cfg_envoie_auto.trame_info.registre=0x00;
cfg_envoie_auto.trame_info.champ.extend=1;
cfg_envoie_auto.trame_info.champ.dlc=0x03;
//Lights Stalk
cfg_envoie_auto.ident.extend.identificateur.ident=Ident_T_IM_Commodo_Feux;
cfg_envoie_auto.data[0]=0x2C; // STCON=0x10 register address
          // + Offset=0x1C==>0x2C
cfg_envoie_auto.data[1]=0xFF; //All the bits are concerned
cfg_envoie_auto.data[2]=0xeF; //STEN=1=>Validate automatic sending frames
          //SMTS=1=>DLC=8
                                          //STBF1=1,STBF0=0,STM3=STM2=S
                          //=>Time base=1 second
                                          //==> it therefore a
                  //every second
EcrireTrame(&cfg_envoie_auto);
```

Value example for the timebase: 0xCF=4ms, 0xD2=12ms, 0xD3=[illegible]ms, 0xEF=1s

The sent frame determination

In this function mode, the frame (see MCP25050) is [illegible]"
The sensors' values are located in the data[1]. The po[illegible] value (of Windshield Wiper stalk) is in data[2].

The frames' identifiers received by the SJA[illegible]00 a[illegible]ensors acquisition will be like :

```
          if (tr_rx.ident.extend.identificateur.iden[illegible]0x[illegible]000)
                    {
                    Valeur_Commodo_Feux=~tr_rx.data[1]
                    }
          if (tr_rx.ident.extend.identificateu[illegible])
                    {
                    Valeur_Commodo_EG=~tr_rx.[illegible]ta
                    Valeur_Analogiqu[illegible]rx.[illegible]ta[
                    }
          if (tr_rx.ident.extend.i[illegible]ti[illegible]xte[illegible]ident==0[illegible]0900000)
                    {
                    Valeur_FC_EG=~[illegible]rx.data[
                    }
```

8.2.3 Solution 2

We choose a MCP25050 frame sending during a state changing.

Concerned register

The register of Transition validation is IOINTEN. It is located at 0x00 address to which we have to add the offset of 1C.
Register Composition:

GP7TXC	GP6TXC	GP5TXC	GP4TXC	GP3TXC	GP2TXC	GP2TXC	GP0TXC

A logic 1 can activate the automatic transition on the state changing.
A logic 0 (default value) inhibits this function.

To avoid the bounds, you can use the IOINTPO register which authorizes the sending during a transition of 1 to 0, or 0 to 1. It is located at 0x01 address to which we have to add the offset of 1C.

Register Composition:

GP7POL	GP6POL	GP5POL	GP4POL	GP3POL	GP2POL	GP2POL	GP0POL

A logic 1 validate a transition from 0 to 1.
A logic 0 validate a transition from 1 to 0.
At each change of state, we have to change the register. Ex : wh[illegible]ght control is active, we will have to validate a transition from 1 to 0.

Then we obtain the following code:

```
// Configuration Component for automatic sending frames
cfg_envoie_auto.trame_info.registre=0x00;
cfg_envoie_auto.trame_info.champ.extend=1;
cfg_envoie_auto.trame_info.champ.dlc=0x03;
//Lights Stalk
cfg_envoie_auto.ident.extend.identificateur.ident=Ident_T_IM_Comm[illegible]_Fe
cfg_envoie_auto.data[0]=0x1C; // IOTEN=0x00 register address
        //+ Offset=0x1C==>0x1C
cfg_envoie_auto.data[1]=0xFF; //All the bits are concerne[illegible]
cfg_envoie_auto.data[2]=0xFF; //GP7TXC to GP0TXC are vali[illegible]r a [illegible]
//state changing
EcrireTrame(&cfg_envoie_auto);

//Transition configuration of rising or falling fron[illegible]
Cfg_envoie_auto.trame_info.registre=0x00;
Cfg_envoie_auto.trame_info.champ.extend=1;
Cfg_envoie_auto.trame_info.champ.dlc=0x03;
Cfg_envoie_auto.ident.extend.identificateur.i[illegible]t=Ident_[illegible]mmodo_Feux;
Cfg_envoie_auto.data[0]=0x1D; // IOINTP0=0x0[illegible]egi[illegible]add[illegible]
        // + Offset=0x1C==>0x1D
Cfg_envoie_auto.data[1]=0xFF; // Al[illegible]bit[illegible]are
Cfg_envoie_auto.data[2]=Valeur_Com[illegible]do_[illegible]; //stalk [illegible]e,
                                                        // exactly correspond to
                                                // transitions to do
EcrireTrame(&cfg_envoie_auto);
```

The sent frame determination

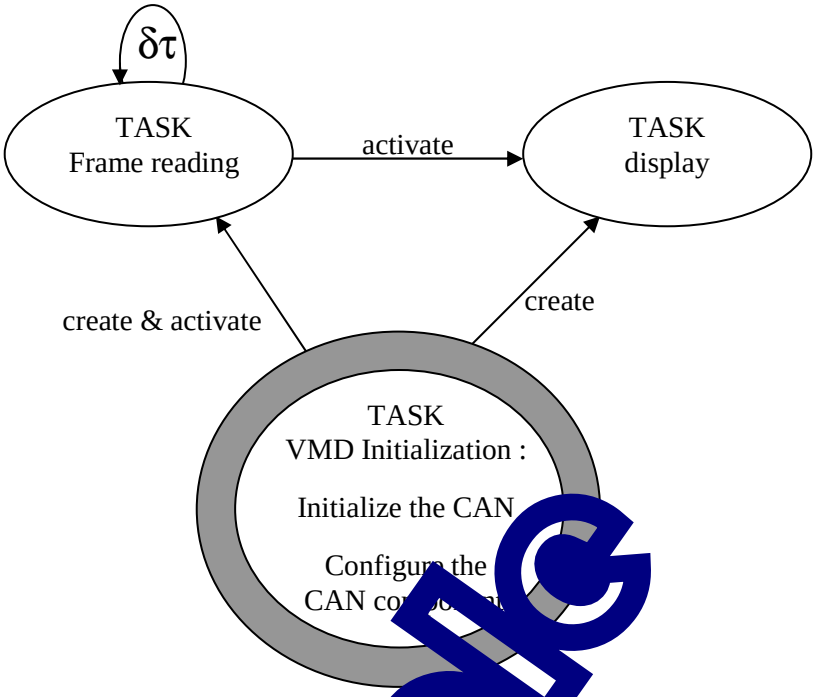
In this function mode, the frame (see MCP25050) is an OM, with an IRM type identifier at address zero. The sensors' values are located in the data[1].

The frames' identifiers received by the SJA1000, and the sensors acquisition will be like :

```
        if (tr_rx.ident.extend.identificateur.ident==0x05400000)
                {
                Valeur_Commodo_Feux=~tr_rx.data[1];
                }
        if (tr_rx.ident.extend.identificateur.ident==0x05C00000)
                {
                Valeur_Commodo_EG=~tr_rx.data[1];
                }
        if (tr_rx.ident.extend.identificateur.ident==0x00C00000)
                {
                Valeur_FC_EG=~tr_rx.data[1];
                }
```

Remark : It is now possible to manage the VMD with only automatic sending

8.2.4 State diagram



8.2.5 C Program

Frames regular sending

```
/* Frames automatical sending of lights stalk, Windshield Wiper and Servo-system board after a regular time interval */
#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"

//Tasks prototype
TACHE  init  (void);
TACHE  LectureTrame(void);
TACHE  init_commodo(void);
TACHE  Affichage(void);

/*****************************************************************************************************************/
/*                            VMD initialization task( modules configuration)                                    */
/*****************************************************************************************************************/
TACHE init(void)
{
Trame t_gpdr,cfg_envoie_auto,T_IM_CFG,Trame_Recue;
//the data
int cpt;
//Table including all the data
//The 2 first values are for the Analog/Digital Convertor of the wiper stalk wheel
//The following are for the Servo-system (cfg PWM, frequency, Power Validation,...)
int datazero[]={0x2A,0x2B,0x21,0x23,0x22,0x24,0x25,0x26,0x1E};
int dataun[]={0xF0,0xFF,0xB3,0xFF,0xB3,0xFF,0xFF,0xFF,0x10};
int datadeux[]={0x80,0x0E,0x80,0xFF,0x80,0xFF,0x00,0x00,0x10};

OpenCAN(ATON_V2);   //ATON_V1 for old ATONCAN boards
                    //ATON_V2 for new ATONCAN boards

//Inputs/Outputs configuration
t_gpdr.trame_info.registre=0x00;
t_gpdr.trame_info.champ.extend=1; // Work in the extended mode
t_gpdr.trame_info.champ.dlc=0x03; // there are 3 data of 8 bits (3 sent bytes)
//Lights stalk
t_gpdr.ident.extend.identificateur.ident=Ident_T_IM_Commodo_Feux;
t_gpdr.data[0]=0x1F;
t_gpdr.data[1]=0x7F;
t_gpdr.data[2]=0x7F; // third data -> "Value" -> all the bits are imp
EcrireTrame(&t_gpdr);
while((LireTrame(&Trame_Recue)==0));
//Windshield Wiper Stalk
t_gpdr.ident.extend.identificateur.ident=Ident_T_IM_Commo
t_gpdr.data[2]=0x7F;
EcrireTrame(&t_gpdr);
while((LireTrame(&Trame_Recue)==0));
//Servo-system board
t_gpdr.ident.extend.identificateur.ident=Ident_T_IM_
t_gpdr.data[2]=0xE3;
EcrireTrame(&t_gpdr);
while((LireTrame(&Trame_Recue)==0));

//Windshield Wiper Stalk Ana -> Dig converto
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1;
T_IM_CFG.trame_info.champ.dlc=0x03
T_IM_CFG.ident.extend.identificate
for(cpt=0;cpt<2;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];
T_IM_CFG.data[1]=dataun[cpt];
T_IM_CFG.data[2]=datadeux[cpt];
EcrireTrame(&T_IM_CFG);
while((LireTrame(&Trame_Recue)==0));
}

//Specific Configuration of Servo-system module
T_IM_CFG.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
for(cpt=0;cpt<9;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];        // Register address
T_IM_CFG.data[1]=dataun[cpt];          // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];        // Value
EcrireTrame(&T_IM_CFG);
while((LireTrame(&Trame_Recue)==0));
}

//Components configuration for the  automatical frames sending...
cfg_envoie_auto.trame_info.registre=0x00;
cfg_envoie_auto.trame_info.champ.extend=1;
cfg_envoie_auto.trame_info.champ.dlc=0x03;
//Lights stalk
cfg_envoie_auto.ident.extend.identificateur.ident=Ident_T_IM_Commodo_Feux;
cfg_envoie_auto.data[0]=0x2C; //Register address STCON=0x10 + Offset=0x1C==>0x2C
cfg_envoie_auto.data[1]=0xFF; //All the bits are concerned
cfg_envoie_auto.data[2]=0xeF; //STEN=1=>Validate automatic frame sending/SMTS=1=>DLC=8
                                        //STBF1=1,STBF0=0,STM3=STM2=STM1=STM0=1 =>Time base=1seconde
                                        //==>we authorize an automatic sending every 1 second (cf datasheet p24)
EcrireTrame(&cfg_envoie_auto);
//Windshield Wiper Stalk
cfg_envoie_auto.ident.extend.identificateur.ident=Ident_T_IM_Commodo_EG;
EcrireTrame(&cfg_envoie_auto);
//Servo-system
cfg_envoie_auto.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
```

```
EcrireTrame(&cfg_envoie_auto);

MONITOR
clsscr();
//The display takes a lot of time, so we anticipate
gotoxy(1,1);
printf("************************EX_7 CAN bus with mtr86************************\n");
printf(" Windshield Wiper Stalk State\n");
printf("  GP0: Cde_EG_Av_Int=         , Potentiomenter value=                                \n");
printf("  GP3: Cde_EG_Av_Pos1=        , GP4: Cde_EG_Av_Pos2=                      \n");
printf("  GP7: Cde_Lave_Glace_Av=                                                            \n");
printf("  GP1: Input1=               , GP2: Input2=                               \n");
printf("  GP5: Cde_EG_Ar=             , GP6: Cde_Lave_Glace_Ar=                   \n");
printf("
\n");
printf(" Lights Stalk Status\n");
printf("  GP0: Side light=      ,GP2: Head light=        ,GP3: Dipped light=                  \n");
printf("  GP1: Warning=        , GP4: Left indicator=   , GP5: Right indicator=              \n");
printf("  GP6: Stop light=      ,GP7: Horn=                                      \n");
printf("
\n");
printf(" Limit Switch State on the Servo-system board                                        \n");
printf("  Limit Switch: Left=            Over-limit=          Right=              \n");
ENDM

active(cree(LectureTrame,1,1024));
cree(Affichage,2,1024);
}

/*****************************************************************************************************************/
/*                                                 Read the received frames
                                                */
/*****************************************************************************************************************/
TACHE LectureTrame(void)
{
static Trame tr_rx;
while(1)
        {
        if(LireTrame(&tr_rx))
                {
                MONITOR
                gotoxy(1,16);
                printf("Received frames\n");
                if (tr_rx.ident.extend.identificateur.ident==0x0
                        {
                        Valeur_Commodo_Feux=~tr_rx.data[1];
                        gotoxy(1,17);
                        printf("Lights Stalk response,%x  \                 o_Feux);
                        }
                if (tr_rx.ident.extend.identificateur.ide
                        {
                        Valeur_Commodo_EG=~tr_rx.da      1];
                        Valeur_Analogique=tr_rx.dat      ]*
                        gotoxy(1,18);
                        printf("Windshield Wipe      talk res                \n",Valeur_Commodo_EG);
                        }
                if (tr_rx.ident.extend.identifica      ur.                  000000)
                        {
                        Valeur_FC_EG=~tr_rx
                        gotoxy(1,19);
                        printf("Servo-s      em respo      \n",Valeur_FC_EG);
                        }
                gotoxy(1,20);
                AfficheTrame(&t
                active(num_tac      Af      g
                ENDM
                dort(20);
                }
        }
}

/*****************************************************************************************************************/
/*                          Display task (up      ing the system state on the monitor)
        */
/*****************************************************************************************************************/
TACHE Affichage(void)
{
static Valeur_Commodo_Feux_Aff;
static Valeur_Commodo_EG_Aff;
static Valeur_FC_EG_Aff;
static Valeur_Analogique_Aff;

if(Valeur_Analogique_Aff!=Valeur_Analogique)
        {
        Valeur_Analogique_Aff=Valeur_Analogique;
        gotoxy(45,3);
        printf("%d   \n",Valeur_Analogique);
        }

if(Valeur_Commodo_EG_Aff!=Valeur_Commodo_EG)
//we update the values of windshield wiper stalk on the Monitor
        {
        Valeur_Commodo_EG_Aff=Valeur_Commodo_EG;
        MONITOR
        gotoxy(22,3);
        printf("%d\n",Cde_EG_Av_Int);
        gotoxy(16,6);
        printf("%d\n",Entree1);
        gotoxy(46,6);
        printf("%d\n",Entree2);
        gotoxy(23,4);
        printf("%d\n",Cde_EG_Av_Pos1);
```

```
          gotoxy(53,4);
          printf("%d\n",Cde_EG_Av_Pos2);
          gotoxy(18,7);
          printf("%d\n",Cde_EG_Ar);
          gotoxy(56,7);
          printf("%d\n",(Cde_Lave_Glace_Ar^0x01));
          gotoxy(27,5);
          printf("%d\n",(Cde_Lave_Glace_Av^0x01));
          ENDM
          }

if(Valeur_FC_EG_Aff!=Valeur_FC_EG)
//we update the values of limit switch on the Monitor
          {
          Valeur_FC_EG_Aff=Valeur_FC_EG;
          MONITOR
          gotoxy(26,15);
          if(fcg)   //End of left active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(62,15);
          if(fcd)   //End of right active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(46,15);
          if(fs)    //End of over limit
                    printf("1\n");
          else
                    printf("0\n");
          ENDM
          }
if(Valeur_Commodo_Feux_Aff!=Valeur_Commodo_Feux)
//we update the values of lights stalk on the Monitor
          {
          Valeur_Commodo_Feux_Aff=Valeur_Commodo_Feux;
          MONITOR
          gotoxy(18,10);
          printf("%d\n",Cde_Veilleuse);
          gotoxy(36,10);
          printf("%d\n",Cde_Phare);
          gotoxy(55,10);
          printf("%d\n",Cde_Code);
          gotoxy(16,11);
          printf("%d\n",Cde_Warning);
          gotoxy(41,11);
          printf("%d\n",Cde_Clign_Gauche);
          gotoxy(60,11);
          printf("%d\n",Cde_Clign_Droit);
          gotoxy(18,12);
          printf("%d\n",Cde_Stop);
          gotoxy(37,12);
          printf("%d\n",Cde_Klaxon);
          ENDM
          }
}

/*******************************************************************************************************************/
/*                                                           Program

/*******************************************************************************************************************/
main()
{
clsscr();
printf("EX_7 CAN bus with mtr86\n");
start_mtr(init,1024);
}
```

Frames sending for state changing

```
/* Frames automatical sending of lights stalk, Windshield Wiper and Servo-system board when status changing*/
#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"

//Tasks prototype
TACHE  init  (void);
TACHE  LectureTrame(void);
TACHE  init_commodo(void);
TACHE Affichage(void);

/*****************************************************************************************************************/
/*                                              VMD initialization task( modules configuration)
                */
/*****************************************************************************************************************/
TACHE init(void)
{
Trame t_gpdr,cfg_envoie_auto,T_IM_CFG,Trame_Recue;

//the data
int cpt;
//Table including all the data
//The 2 first values are for the Analog/Digital Convertor of the wiper stalk wheel
//The following are for the Servo-system (cfg PWM, frequency, Power Validation,...)
int datazero[]={0x2A,0x2B,0x21,0x23,0x22,0x24,0x25,0x26,0x1E};
int dataun[]={0xF0,0xFF,0xB3,0xFF,0xB3,0xFF,0xFF,0xFF,0x10};
int datadeux[]={0x80,0x0E,0x80,0xFF,0x80,0xFF,0x00,0x00,0x10};

OpenCAN(ATON_V2);   //ATON_V1 for old ATONCAN boards
                            //ATON_V2 for new ATONCAN boards

//Inputs/Outputs configuration
t_gpdr.trame_info.registre=0x00;
t_gpdr.trame_info.champ.extend=1; // Work in the extended mode
t_gpdr.trame_info.champ.dlc=0x03; // there are 3 data of 8 bits (3 s
//Lights stalk
t_gpdr.ident.extend.identificateur.ident=Ident_T_IM_Commodo_Feux;
t_gpdr.data[0]=0x1F;
t_gpdr.data[1]=0x7F;
t_gpdr.data[2]=0x7F;
EcrireTrame(&t_gpdr);
while((LireTrame(&Trame_Recue)==0));
//Windshield Wiper Stalk
t_gpdr.ident.extend.identificateur.ident=Ident_T_IM_Commodo_EG;
t_gpdr.data[2]=0x7F; // third data-> "Value" -> All the bits
EcrireTrame(&t_gpdr);
while((LireTrame(&Trame_Recue)==0));
//Servo-system board
t_gpdr.ident.extend.identificateur.ident=Ident_T_IM_As
t_gpdr.data[2]=0xE3;
EcrireTrame(&t_gpdr);
while((LireTrame(&Trame_Recue)==0));

//Windshield Wiper Stalk Ana -> Dig convertor configu
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1; // Work i
T_IM_CFG.trame_info.champ.dlc=0x03; // there
T_IM_CFG.ident.extend.identificateur.ident=I
for(cpt=0;cpt<2;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];
T_IM_CFG.data[1]=dataun[cpt];
T_IM_CFG.data[2]=datadeux[cpt];
EcrireTrame(&T_IM_CFG);
while((LireTrame(&Trame_Recue)==0));
}

//Specific Configuration of Servo-system mod
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1; // Work in the extended mode
T_IM_CFG.trame_info.champ.dlc=0x03; // there are 3 data of 8 bits (3 sent bytes)
T_IM_CFG.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
for(cpt=0;cpt<9;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];        // Register address
T_IM_CFG.data[1]=dataun[cpt];          // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];        // Value
EcrireTrame(&T_IM_CFG);
while((LireTrame(&Trame_Recue)==0));
}

//Components configuration for the  automatical frames sending...
cfg_envoie_auto.trame_info.registre=0x00;
cfg_envoie_auto.trame_info.champ.extend=1;
cfg_envoie_auto.trame_info.champ.dlc=0x03;
//Lights stalk
cfg_envoie_auto.ident.extend.identificateur.ident=Ident_T_IM_Commodo_Feux;
cfg_envoie_auto.data[0]=0x1C; //Register address IOTEN=0x00 + Offset=0x1C==>0x1C
cfg_envoie_auto.data[1]=0xFF; //All the bits are concerned
cfg_envoie_auto.data[2]=0xFF; //GP7TXC to GP0TXC are valid for a message sending when state changing
EcrireTrame(&cfg_envoie_auto);
while((LireTrame(&Trame_Recue)==0));

//Windshield Wiper Stalk
cfg_envoie_auto.ident.extend.identificateur.ident=Ident_T_IM_Commodo_EG;
EcrireTrame(&cfg_envoie_auto);
while((LireTrame(&Trame_Recue)==0));
//Servo-system
```

```
cfg_envoie_auto.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
EcrireTrame(&cfg_envoie_auto);
while((LireTrame(&Trame_Recue)==0));

MONITOR
clsscr();
//The display takes a lot of time, so we anticipate
gotoxy(1,1);
printf("************************EX_7 CAN bus with mtr86************************\n");
printf(" Windshield Wiper Stalk State                                                                    \n");
printf("  GP0: Cde_EG_Av_Int=                                                                          \n");
printf("  GP3: Cde_EG_Av_Pos1=        , GP4: Cde_EG_Av_Pos2=                      \n");
printf("  GP7: Cde_Lave_Glace_Av=                                                                      \n");
printf("  GP1: Input1=               , GP2: Input2=                               \n");
printf("  GP5: Cde_EG_Ar=             , GP6: Cde_Lave_Glace_Ar=                   \n");
printf("
\n");
printf(" Lights Stalk Status\n");
printf("  GP0: Side light=      ,GP2: Head light=        ,GP3: Dipped light=                  \n");
printf("  GP1: Warning=        , GP4: Left indicator=   , GP5: Right indicator=              \n");
printf("  GP6: Stop light=      ,GP7: Horn=                                        \n");
printf("
\n");
printf("
\n");
printf(" Limit Switch State on the Servo-system board                                          \n");
printf("  Limit Switch: Left=            Over-limit=          Right=              \n");
ENDM

active(cree(LectureTrame,1,1024));
cree(Affichage,2,1024);
}

/*****************************************************************************************************************/
/*                                                Read the received fr                                           */
/*****************************************************************************************************************/
TACHE LectureTrame(void)
{
static Trame tr_rx,Cfg_IointP0;
//This frame allows us to define the transition on a rising or falli ront
Cfg_IointP0.trame_info.registre=0x00;
Cfg_IointP0.trame_info.champ.extend=1;
Cfg_IointP0.trame_info.champ.dlc=0x03;
Cfg_IointP0.data[0]=0x1D;     //Register address IOINTP0=0x01 + Off =0x1C=
Cfg_IointP0.data[1]=0xFF; //All the bits are concerned

while(1)
          {
          if(LireTrame(&tr_rx))
                    {
                    MONITOR
                    gotoxy(1,16);
                    printf("Received frames\n");
                    if (tr_rx.ident.extend.identifica .ident= 000)
                              {
                              Valeur_Commodo_Feux=~tr .d
                              gotoxy(1,17);
                              printf("Lights Stal %x ,Valeur_Commodo_Feux);
                              //the transition g on the state of the inputs
                              // If it was th ising f change it to the falling front or inverse
                              Cfg_IointP0.id .ex ide cateur.ident=Ident_T_IM_Commodo_Feux;
                              Cfg_Io P0.da 2] do_Feux;
                              Ecrir &C Io
                              }
                    if (tr_rx.iden exte ident==0x05C00000)
                              {
                              Valeu mmodo_EG- _rx.data[1];
                              gotoxy(
                              printf("Wi ld per Stalk response,%x  \n",Valeur_Commodo_EG);
                              //the trans i anged depending on the state of the inputs
                              // If it was ing front, we change it to the falling front or inverse
                              Cfg_IointP0.ident.extend.identificateur.ident=Ident_T_IM_Commodo_EG;
                              Cfg_IointP0.data[2]=Valeur_Commodo_EG;
                              EcrireTrame(&Cfg_IointP0);
                              }
                    if (tr_rx.ident.extend.identificateur.ident==0x00C00000)
                              {
                              Valeur_FC_EG=~tr_rx.data[1];
                              gotoxy(1,19);
                              printf("Servo-system response,%x  \n",Valeur_FC_EG);
                              //the transition is changed depending on the state of the inputs
                              // If it was the rising front, we change it to the falling front or inverse
                              Cfg_IointP0.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
                              Cfg_IointP0.data[2]=Valeur_FC_EG;
                              EcrireTrame(&Cfg_IointP0);
                              }
                    gotoxy(1,20);
                    AfficheTrame(&tr_rx);
                    active(num_tache(Affichage));
                    ENDM
                    dort(20);
                    }
          }
}

/*****************************************************************************************************************/
/*                          Display task (updating the system state on the monitor)                              */
/*****************************************************************************************************************/
TACHE Affichage(void)
{
static Valeur_Commodo_Feux_Aff;
```

```
static Valeur_Commodo_EG_Aff;
static Valeur_FC_EG_Aff;

if(Valeur_Commodo_EG_Aff!=Valeur_Commodo_EG)
//we update the values of windshield wiper stalk on the Monitor
          {
          Valeur_Commodo_EG_Aff=Valeur_Commodo_EG;
          MONITOR
          gotoxy(22,3);
          printf("%d\n",Cde_EG_Av_Int);
          gotoxy(16,6);
          printf("%d\n",Entree1);
          gotoxy(46,6);
          printf("%d\n",Entree2);
          gotoxy(23,4);
          printf("%d\n",Cde_EG_Av_Pos1);
          gotoxy(53,4);
          printf("%d\n",Cde_EG_Av_Pos2);
          gotoxy(18,7);
          printf("%d\n",Cde_EG_Ar);
          gotoxy(56,7);
          printf("%d\n",(Cde_Lave_Glace_Ar^0x01));
          gotoxy(27,5);
          printf("%d\n",(Cde_Lave_Glace_Av^0x01));
          ENDM
          }

if(Valeur_FC_EG_Aff!=Valeur_FC_EG)
//we update the values of limit switch on the Monitor
          {
          Valeur_FC_EG_Aff=Valeur_FC_EG;
          MONITOR
          gotoxy(26,15);
          if(fcg)   //End of left active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(62,15);
          if(fcd)   //End of right active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(46,15);
          if(fs)    //End of over limit
                    printf("1\n");
          else
                    printf("0\n");
          ENDM
          }
if(Valeur_Commodo_Feux_Aff!=Valeur_Commodo_Feux)
//we update the values of lights stalk on the Monitor
          {
          Valeur_Commodo_Feux_Aff=Valeur_Commodo_Feux
          MONITOR
          gotoxy(18,10);
          printf("%d\n",Cde_Veilleuse);
          gotoxy(36,10);
          printf("%d\n",Cde_Phare);
          gotoxy(55,10);
          printf("%d\n",Cde_Code);
          gotoxy(16,11);
          printf("%d\n",Cde_Warning
          gotoxy(41,11);
          printf("%d\n",Cde_Clign_
          gotoxy(60,11);
          printf("%d\n",Cde_Clign_D
          gotoxy(18,12);
          printf("%d\n",Cde_Stop);
          gotoxy(37,12);
          printf("%d\n",Cde_Klaxon);
          ENDM
          }
}

/*****************************************************************************************************************/
/*                                                  Main Program
                                                */
/*****************************************************************************************************************/
main()
{
clsscr();
printf("EX_7 CAN bus with mtr86\n");
start_mtr(init,1200);
}
```


Sample

9 EX N°8 : VMD CONTROL WITH REAL TIME EXECUTIVE

9.1 Topic :

Purposes :	- Develop a complete real time application, including the ON/OFF sensors and the analog sensors at the same time, the analog pre-actuators and integrating a speed change function.
Specifications :	We want that the Lights Stalk, Windshield Wiper Stalk and the Servo-system inform us their states. At first, we want a regular sending from the components. Then, we want a regular sending on the state changing. According to the state of the wiper stalk, we control the motor (intermittent, position 1, position 2, etc.). According to the state of the [illegible] stalk, [illegible] trol the different lights from optical blocks (indicators, [illegible] lights, headlights) We wish to leave correctly [illegible] press the EID210 CRTL key to turn off the optical blocks and [illegible] the windshield wiper)

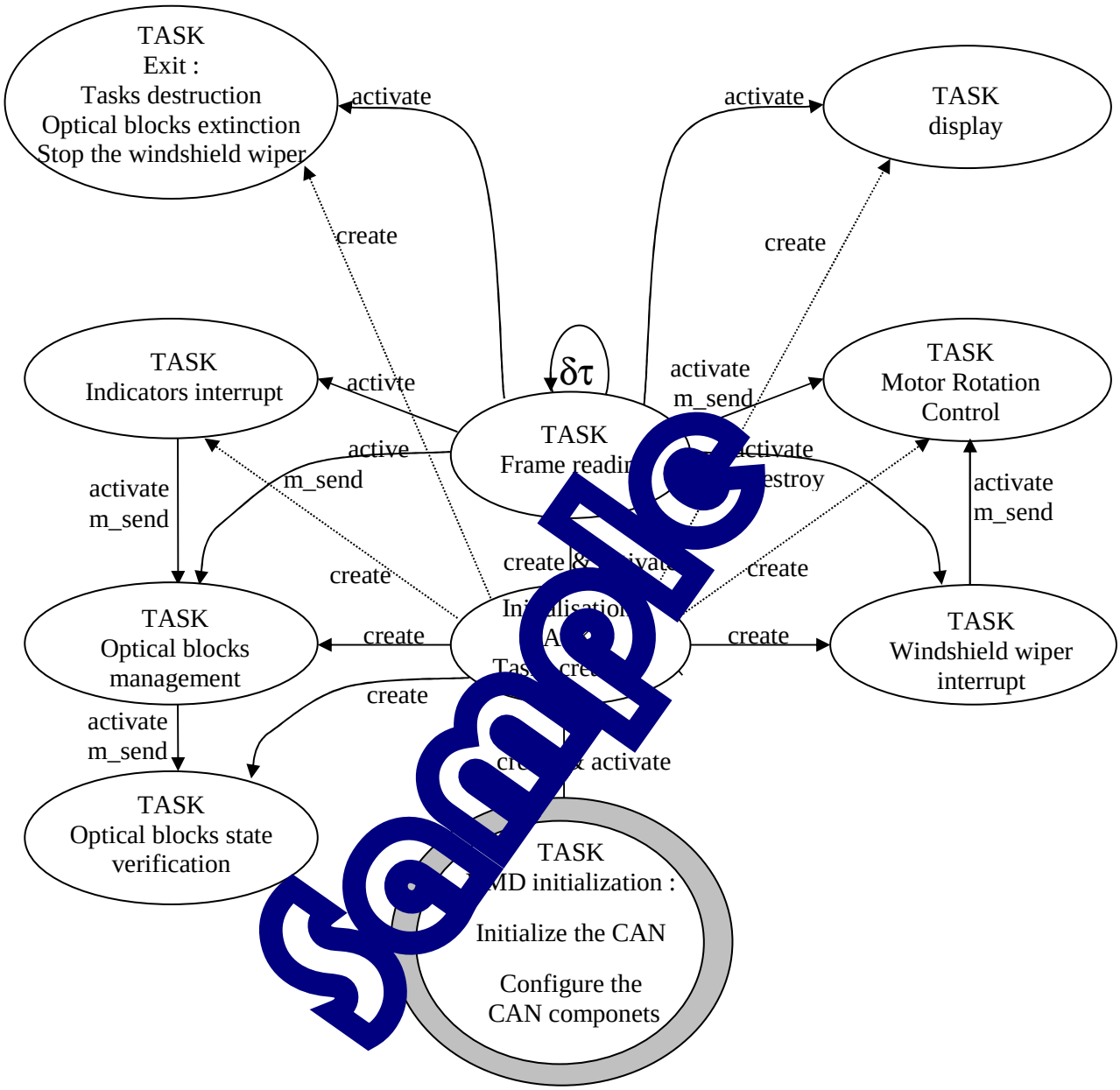
Necessary hardware and software:

PC Micro Computer using Windows ® 95 [illegible]
Editor Software-Assembler-Debugger
If programming in C, GNU; C / C + [illegible] Ref: EID210101
Processor board 16/32 bit 68[illegible] and its software environment
(Editor-Cross Assembler-[illegible] Ref: [illegible]210001
CAN PC/104 Network board in AT[illegible]EMS Ref NIC: EID004001
CAN network with:
- two 8 logic inputs module for the lights stalk Ref: EID050001
- one 4 power outputs module for left/right back/front lights Ref: EID051001
- one Servo-system module for the windshield wiper Ref : EID053001

USB connection cable, or if not available RS232 cable, Ref: EGD000003
AC / AC Power source 8V 1A Ref: EGD000001
12V Power source supply for the CAN modules ("energy" network)

Time : 3 x 4 hours

9.1.1 State diagram



9.1.2 C Program

Frames ragular sending

```
/* VMD management: optical blocks, Lights Stalk, Windshield Wiper and its Stalk */
//**************************
#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"
#include "eid210.h"

#define Ident_T_OB_Commodo_Feux 0x05100000 // "Light Stalk"  On Bus (by test manager)

//For the Windshield Wiper delay
#define Tempo1 2    // in second
#define Tempo2 4 // For intermittent mode
#define Tempo3 6
#define Tempo4 8
#define Tempo5 10
//Flag of Windshield Wiper delay end
char Flag_Fin_Tempo_EG;
//For the beat rate
#define Vitesse_Rapide 0x90
#define Vitesse_Lente 0x40
#define Vitesse_Lave_Glace 0x60
//For the delay between two indicator's flashing
#define TempoClign 6         //in ms


//Tasks Prototype
TACHE  Init_VMD(void);
TACHE  init  (void);
TACHE      Actualise_ASV();
TACHE      Actualise_Etat_Feux();
TACHE  LectureTrame(void);
TACHE  irq_Clign(void);
TACHE  irq_eg();
TACHE  Verif_Etat_Feux(void);
TACHE  Affichage(void);
TACHE  Quit(void);

/*****************************************************************************************************************/
/*                             VMD initialization task (Modules Co[illegible]gu[illegible])                                  */
/*****************************************************************************************************************/
TACHE Init_VMD(void)
{
//Modules Configuration
Trame T_IM_CFG,T_IRM_Etat_FC,Trame_Recue;
//For the waiting delays between two sends
int Temp,Cptr_TimeOut;
// the identifiers, value,...
int cpt;
//Identifiers,values,data:
//Identifier
int
identificateur[]={Ident_T_IM_Commodo_Feux,Id[illegible]_T_IM_Com[illegible],Ident_T_IM_Asservissement,Ident_T_IM_FRG,Ident_T_IM_FRD,Ident
_T_IM_FVG,Ident_T_IM_FVD};
//Acknowledge
int
identificateurACK[]={Ident_T_AIM_C[illegible]modo[illegible]eux[illegible]dent_T_A[illegible]Commodo_EG,Ident_T_AIM_Asservissement,Ident_T_AIM_FRG,Ident_T_AIM_F
RD,Ident_T_AIM_FVG,Ident_T_AIM_FV[illegible]
//Table including the data for I /[illegible] configu[illegible]
int Config_E_S[]={0xFF,0xFF,0xE3,0x[illegible]0xF0,0xF0,[illegible]F0};

//Strings can indicate module initializati[illegible]
char *msg[]={" Light stalk"," Windshield [illegible]r S[illegible]lk "," Servo-system Module ","Left Back Lights","Right Back Lights","Left
Front Lights","Right Front Lights"};

//Table including all the data..
//The 2 first values are for the Analog/Digital Convertor of the wiper stalk wheel
//The following are for the Servo-system (cfg PWM, frequency, Power Validation,...)
int datazero[]={0x2A,0x2B,0x21,0x23,0x22,0x24,0x25,0x26,0x1E};
int dataun[]={0xF0,0xFF,0xB3,0xFF,0xB3,0xFF,0xFF,0xFF,0x10};
int datadeux[]={0x80,0x0E,0x80,0xFF,0x80,0xFF,0x00,0x00,0x10};

//CAN and MTR configuration
dsp_stk(); //Know the state of the tasks when we leave through mtr86exit(0);
OpenCAN(ATON_V2);   //ATON_V1 for old ATONCAN boards
                    //ATON_V2 for new ATONCAN boards

T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1; // Work in the extended mode
T_IM_CFG.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
// For the 4 outputs modules configuration (GP0 to GP3) and 4inputs status 4 (GP4 to GP8)
T_IM_CFG.data[0]=0x1F;        // first data -> "Address" of concerned register-> GPDDR
T_IM_CFG.data[1]=0xFF;        // second data -> "Mask"-> See Doc MCP25050 page 16


//Configuration of modules to address and operation of their I/O
for(cpt=0;cpt<7;cpt++)
{
T_IM_CFG.ident.extend.identificateur.ident=identificateur[cpt]; //establishment of the identifier
T_IM_CFG.data[2]=Config_E_S[cpt];       // third data-> "Value"

MONITOR
gotoxy(2,9);        //we display a message to say that which module is being initialized
printf("Initialisation du %s\n",msg[cpt]);
```

```
do        {
          EcrireTrame(&T_IM_CFG);// Send a first frame on the CAN network
          Cptr_TimeOut=0;     //The LireTrame function let the task sleep if there aren't any received frames.
          do{Cptr_TimeOut++;}while((LireTrame(&Trame_Recue)==0)&&(Cptr_TimeOut<200));
          if(Trame_Recue.ident.extend.identificateur.ident==identificateurACK[cpt])Cptr_TimeOut=200; // Test if the identifier
is correct
                    else
                    {
                    printf("Probleme\n");
                    Cptr_TimeOut=0;
                    }
          for(Temp=0;Temp<100000;Temp++); // To wait for a while!
          }while(Cptr_TimeOut!=200);
printf("OK\n");
ENDM
}

//Frame configuration
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1;
T_IM_CFG.trame_info.champ.dlc=0x03;

//configuration of Windshield Wiper stalk Ana -> Dig converter
MONITOR
gotoxy(2,9);        //we display a message to say that which module is being configured
printf("Configuration of the  A/D conversion of %s ...\n",msg[1]);
ENDM
T_IM_CFG.ident.extend.identificateur.ident=identificateur[1];
for(cpt=0;cpt<2;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];        // Register address
T_IM_CFG.data[1]=dataun[cpt];          // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];        // Value
EcrireTrame(&T_IM_CFG);
while((LireTrame(&Trame_Recue)==0));
}

//Specific Configuration of Servo-system module
MONITOR
gotoxy(2,9);        //we display a message to say that which module i          ing
printf("Specific Configuration of %s ...           \n",msg[2]);
ENDM
T_IM_CFG.ident.extend.identificateur.ident=identificateur[2];
for(cpt=0;cpt<9;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];        // Register address
T_IM_CFG.data[1]=dataun[cpt];          // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];        // Value
EcrireTrame(&T_IM_CFG);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the res
}

//Take the original position
MONITOR
printf("Take the wiper POM \n");
ENDM
//Configuration of limit switch interrogation
T_IRM_Etat_FC.trame_info.registre=0x00;
T_IRM_Etat_FC.trame_info.champ.extend=1; // Wo                mode
T_IRM_Etat_FC.trame_info.champ.dlc=0x01; //                bits
T_IRM_Etat_FC.trame_info.champ.rtr=1;   //in
T_IRM_Etat_FC.ident.extend.identific                Asservissement;
EcrireTrame(&T_IRM_Etat_FC);
do{}while(LireTrame(&Trame_Recue)==0); //                onse
Valeur_FC_EG=~(Trame_Recue.data[0]

if(!(fcd==0 && fcg==1))
          {
          //Configuration of motor contro
          T_IM_CFG.trame_info.registre=0
          T_IM_CFG.trame_info.champ.extend=                ork in the extended mode
          T_IM_CFG.trame_info.champ.dlc=0x03;     There are 3 data of 8 bits (3 sent bytes)
          // For the 4 outputs modules configuration (GP0 to GP3) and 4inputs status 4 (GP4 to GP8)
          T_IM_CFG.data[0]=0x26;
          T_IM_CFG.data[1]=0xFF;
          T_IM_CFG.data[2]=0x40;
          EcrireTrame(&T_IM_CFG);//active negative speed
          while(!(fcd==0 && fcg==1))
                    {
                    EcrireTrame(&T_IRM_Etat_FC);
                    while(LireTrame(&Trame_Recue)==0){}; //Wait for the response
                    Valeur_FC_EG=~(Trame_Recue.data[0]); // Recover the limit switch state
                    }
          T_IM_CFG.data[2]=0; //cancel the negative speed
          EcrireTrame(&T_IM_CFG); //disable the negative control
          } //End of Servo-system module initialization
EcrireTrame(&T_IRM_Etat_FC);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the response
Valeur_FC_EG=~(Trame_Recue.data[0]);

MONITOR
clsscr();
//The display takes a lot of time, so we anticipate
gotoxy(1,1);
printf("************************EX_8 CAN bus with mtr86************************\n");
printf(" Windshield Wiper Stalk Status   \n");
printf("  GP0: Cde_EG_Av_Int=         , Potentiometer value=                                                    \n");
printf("  GP3: Cde_EG_Av_Pos1=        , GP4: Cde_EG_Av_Pos2=                        \n");
printf("  GP7: Cde_Lave_Glace_Av=                                                                  \n");
printf("  GP1: Inpout1=               , GP2: Input2=                                      \n");
printf("  GP5: Cde_EG_Ar=             , GP6: Cde_Lave_Glace_Ar=                    \n");
```

```
printf("
\n");
printf(" Lights stalk status                    \n");
printf("  GP0: Side light=      ,GP2: Head light=        ,GP3: Dipped light=                    \n");
printf("  GP1: Warning=        ,GP4: Left Indicator=   ,GP5: Right Indicator\n");
printf("  GP6: Stop light=      ,GP7: Horn=                                    \n");
printf("
\n");
printf(" Limit Switch State on the Servo-system board                                  \n");
printf("  Limit Switch: Left=            Over-limit=          Right=                    \n");
printf("
\n");
printf(" Left Back Lights:   Action        RealStatus                          \n");
printf("  Side light
\n");
printf("  Stop light
\n");
printf("  Indicator                                                                                  \n");
printf("  Horn
\n");
printf(" Right Back Lights:   Action        RealStatus                           \n");
printf("  Side light
\n");
printf("  Stop light
            \n");
printf("  Indicator                                                                                  \n");
printf(" Left Front Lights:   Action        RealStatus                           \n");
printf("  Side light
\n");
printf("  Dipped light
            \n");
printf("  Head light
            \n");
printf("  Indicator                                                                                  \n");
printf(" Right Front Lights:   Action        RealStatus                                  \n");
printf("  Side light
\n");
printf("  Dipped light
            \n");
printf("  Head light
            \n");
printf("  Indicator                                                                                  \n");
ENDM

//Configuration of lights stalk, Windshield Wiper and Servo-system board mo      frames automatical sending //after a
regular time interval
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1;
T_IM_CFG.trame_info.champ.dlc=0x03;
T_IM_CFG.ident.extend.identificateur.ident=Ident_T_IM_Commod      (Ligh      alk
//Time base calucation:(1/160000)*4096*(STBF1:STBF0)*(STM3      tashe      24)
T_IM_CFG.data[0]=0x2C;       //Register address STCON=0x      + Offse      0x2
T_IM_CFG.data[1]=0xFF; //All the bits are concerned
T_IM_CFG.data[2]=0xeF; //STEN=1=>Validate automatic fr      en      TS      =DL
                                        //ST      =1 STBF      =STM2=STM1=STM0=1 =>Timebase=1 second
                                        //=      e      rize      automatic sending every 1 second
EcrireTrame(&T_IM_CFG);

T_IM_CFG.ident.extend.identificateur.ident=Ident_T_IM_      do_E      //Windshield Wiper Stalk
EcrireTrame(&T_IM_CFG);

T_IM_CFG.ident.extend.identificateur.ident=I      t_T      Asse      sement; //Servo-system board
T_IM_CFG.data[2]=0xd5; //for a more      rt ti      i      (0      =4 ms) (0xD2=12ms) (0xD5=25ms)
EcrireTrame(&T_IM_CFG);

active(cree(init,1,1024)); //task      iti      ;
}

TACHE init(void)
{
cree(Actualise_ASV,1,512);              /      ori      n°1 because lights activation is the most important
cree(Actualise_Etat_Feux,1,512);
active(cree(LectureTrame,2,512));       //fram      eading will automatically fall asleep
cree(irq_Clign,2,256);                          //The IT control for the indicators
cree(irq_eg,2,256);                     //The IT control for the windshield wiper
cree(Verif_Etat_Feux,4,512);            //Checking the status light is less important
cree(Affichage,5,1024);                         //The display retains the lowest priority
cree(Quit,6,256);                       //The task will disable other tasks and leave completely
}
/*****************************************************************************************************************/
/*                                                                  Control the direction of motor rotation
                                                */
/*****************************************************************************************************************/

TACHE Actualise_ASV(Vitesse_EG)                 //task called by reading frame
char Vitesse_EG;
{
//Limit switches' status (at t-1) to restore wiper when the over-limit has been activated
static Valeur_FC_EG_Verif_fs;
//Frames to drive the motor rotation direction
Trame T_IM_Cmde_Negative,T_IM_Cmde_Positive;
//Configuration of Negative command frame
T_IM_Cmde_Negative.trame_info.registre=0x00;
T_IM_Cmde_Negative.trame_info.champ.extend=1; // Work in the extended mode
T_IM_Cmde_Negative.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
T_IM_Cmde_Negative.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
T_IM_Cmde_Negative.data[0]=0x26;        // PWM2DC register address
T_IM_Cmde_Negative.data[1]=0xFF;        // Mask -> all the bits are concerned
//Configuration of Positive command frame
T_IM_Cmde_Positive.trame_info.registre=0x00;
T_IM_Cmde_Positive.trame_info.champ.extend=1; // Work in the extended mode
T_IM_Cmde_Positive.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
```

```
T_IM_Cmde_Positive.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
T_IM_Cmde_Positive.data[0]=0x25;        // PWM2DC register address
T_IM_Cmde_Positive.data[1]=0xFF;        // Mask -> all the bits are concerned

//The windshield washer controls are reversed on the stalk(Active=0 , inactive=1)
if((Cde_EG_Av_Pos2 || Cde_EG_Av_Pos1 || (Cde_Lave_Glace_Av==0))) //a "classic" command is valid
          {
          if(fcd)                               //If it's at right side, return to left
                    {
                    T_IM_Cmde_Negative.data[2]=Vitesse_EG;
                    T_IM_Cmde_Positive.data[2]=0x00;
                    }
          else if(fcg)                          //If it's at left side, return to right
                    {
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=Vitesse_EG;
                    }
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(Valeur_Analogique<200) //intermittent control is activated, we must manage according to the delay
          {
          if(fcd) //we are at right, it passes to the left
                    {
                    T_IM_Cmde_Negative.data[2]=Vitesse_EG;
                    T_IM_Cmde_Positive.data[2]=0x00;
                    Flag_Fin_Tempo_EG=0;                                        //we cancel the flag of delay end
                    }
          else if ((Flag_Fin_Tempo_EG)) //the delay is finished
                    {
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=Vitesse_EG;  //we launch the motor
                    }
          else if (fcg && Flag_Fin_Tempo_EG==0)
                    { //at left, in intermittent position, but the delay isn't finished
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=0x00;        //we stop the motor
                    }
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fcd) //it's at the right without any command, return to left si
          {
          T_IM_Cmde_Negative.data[2]=0x40;
          T_IM_Cmde_Positive.data[2]=0x00;
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fcg) //it's at the left without any command, return to ri  si
          {
          T_IM_Cmde_Negative.data[2]=0x00;
          T_IM_Cmde_Positive.data[2]=0x00;
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fs)//error, it's over limit
          {
          if((Valeur_FC_EG_Verif_fs&0x40)==0x40) //fo  e l  urn until the left limit switch
                    {
                    T_IM_Cmde_Negative.data[2]=
                    T_IM_Cmde_Positive.data[2]  x40;
                    EcrireTrame(&T_IM_Cmde_N  ti
                    EcrireTrame(&T_IM_Cmde_P  ti
                    }
          if((Valeur_FC_EG_Verif_f  x2  0)  //for  right, return until the right limit switch
                    {
                    T_IM_Cmde_Nega  e.da
                    T_IM_Cmde_Posit  data[2]=
                    EcrireTrame(&T_I  egat  );
                    EcrireTrame(&T_IM_Cmd  siti  );
                    }
          }
Valeur_FC_EG_Verif_fs=Valeur_FC_EG;
}
/*****************************************************************************************************************/
/*                                                                  Turn on the lights
                                                                 */
/*****************************************************************************************************************/
TACHE Actualise_Etat_Feux(Value_Commodo)                //task called by reading frame
Port_8ES Value_Commodo; //the variable is under the form of a structure of Port_8ES type
                                  //which allows us a direct access to the bit level
{
Trame T_IM_Feux,T_recue; //Frame that will be set up to address the optical blocks
Valeur_FRG=0x00;    //sets optical blocks to 0
Valeur_FRD=0x00;
Valeur_FVG=0x00;
Valeur_FVD=0x00;

if(Value_Commodo.bit.GP2) //Head light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_P;
          Valeur_FVD|=Cde_FV_P;
          }
else if(Value_Commodo.bit.GP3) //Dipped light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_C;
          Valeur_FVD|=Cde_FV_C;
          }
```

```
else if(Value_Commodo.bit.GP0) //Side light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_V;
          Valeur_FVD|=Cde_FV_V;
          }

if(Value_Commodo.bit.GP1)     //Warning
          {
          Valeur_FRG|=Masque_Clign_AR;
          Valeur_FRD|=Masque_Clign_AR;
          Valeur_FVG|=Masque_Clign_AV;
          Valeur_FVD|=Masque_Clign_AV;
          }
          else
          {
          if(Value_Commodo.bit.GP4) //Left Indicator
                    {
                    Valeur_FRG|=Masque_Clign_AR;
                    Valeur_FVG|=Masque_Clign_AV;
                    }
          if(Value_Commodo.bit.GP5) //Right indicator
                    {
                    Valeur_FRD|=Masque_Clign_AR;
                    Valeur_FVD|=Masque_Clign_AV;
                    }
          }

if(Value_Commodo.bit.GP6)//Stop light
          {
          Valeur_FRG|=Masque_Stop;
          Valeur_FRD|=Masque_Stop;
          }

if(Value_Commodo.bit.GP7) //Horn
          Valeur_FRG|=Masque_Klaxon;

//initialization of the command frame on the lights
T_IM_Feux.trame_info.registre=0x00;
T_IM_Feux.trame_info.champ.extend=1;
T_IM_Feux.trame_info.champ.dlc=0x03;
T_IM_Feux.data[0]=0x1e;
T_IM_Feux.data[1]=0x0f;

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FRG;
T_IM_Feux.data[2]=Valeur_FRG;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FRD;
T_IM_Feux.data[2]=Valeur_FRD;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FVG;
T_IM_Feux.data[2]=Valeur_FVG;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FVD;
T_IM_Feux.data[2]=Valeur_FVD;
EcrireTrame(&T_IM_Feux);

//we just made a lights modification request...
active(num_tache(Verif_Etat_Feux));  //we check the lights state
}
/*****************************************************************************************************************/
/*                                   Reading of received frames                                          */
/*****************************************************************************************************************/
TACHE LectureTrame(void)
{
static Trame tr_rx;
static Valeur_Commodo_Feux_Mem;         //static variable as it is a memorization
static Valeur_Commodo_EG_Mem;
static Valeur_FC_EG_Mem;
static Actu_Aff;
static Valeur_Analogique_Mem; //defined in static because it is a parameter to transmit
static Vitesse_EG;
static Tempo_EG;
while(1)                                  //Task activates permanently
          {
          if(LireTrame(&tr_rx))  //LireTrame stop the task when there aren't any received frame
             {
         if(CRTL==0) active(num_tache(Quit)); //If CRTL key is pressed, we leave
             Actu_Aff=0;

             if (tr_rx.ident.extend.identificateur.ident==Ident_T_OB_Asservissement) //we received a servo-system response
                    {
                    Valeur_FC_EG=~(tr_rx.data[1]);
                              if(Valeur_FC_EG_Mem!=Valeur_FC_EG)
                              {
                              m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                              Actu_Aff=2;
                              }
                              else if(fs==1) m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                    }//End of limit switch handling

             else if (tr_rx.ident.extend.identificateur.ident==Ident_T_OB_Commodo_EG) //we received a windshield wiper stalk
response
                    {
                    Valeur_Commodo_EG=~(tr_rx.data[1]);     //recover the status of windshield wiper stalk
                    Valeur_Analogique=tr_rx.data[2];       //recover the value of potentiometer
                    //we determine the potar position to activate the delay
                    if(Valeur_Analogique!=Valeur_Analogique_Mem)
```

```
                                {
                                Actu_Aff=1;
                                Valeur_Analogique_Mem=Valeur_Analogique;
                                if(Valeur_Analogique>=200)
                                        {
                                        detruit(num_tache(irq_eg));
                                        Tempo_EG=0;
                                        }
                                else
                                        {
                                        detruit(num_tache(irq_eg));
                                        Vitesse_EG=Vitesse_Lente; // "Intermittent" Position
                                        if(Valeur_Analogique>=150)Tempo_EG=Tempo5;// Depending on the position of the wheel
                                                 else if(Valeur_Analogique>=140)Tempo_EG=Tempo4;// the long or short delay
                                                         else if(Valeur_Analogique>=120)Tempo_EG=Tempo3;
                                                                 else if(Valeur_Analogique>=90)Tempo_EG=Tempo2;
                                                                         else
if(20<=Valeur_Analogique==0)Tempo_EG=Tempo1;
                                        m_send(num_tache(irq_eg),&Tempo_EG);
                                        }
                                }

                        if(Valeur_Commodo_EG_Mem!=Valeur_Commodo_EG)
                                {
                                Actu_Aff=1;
                                Valeur_Commodo_EG_Mem=Valeur_Commodo_EG;
                                Vitesse_EG=0;
                                //Disable the intermittent position if it was chosen
                                if(Valeur_Analogique<200) detruit(num_tache(irq_eg));
                                //...Handle the selected position...
                                if(Cde_EG_Av_Pos2) Vitesse_EG=Vitesse_Rapide;
                                else if(Cde_EG_Av_Pos1) Vitesse_EG=Vitesse_Lente;
                                else if(Cde_Lave_Glace_Av==0) Vitesse_EG=Vitesse_Lave_Glace; //stalk inversion
                        //...and reactivate it if it is still actual
                                else if(Cde_EG_Av_Int && (Valeur_Analogique<200))
                                        {
                                        Vitesse_EG=Vitesse_Lente;
                                        m_send(num_tache(irq_eg),&Temp
                                        }
                                m_send(num_tache(Actualise_ASV),&Vite
                                }
                        } //End of stalk handling

                else if (tr_rx.ident.extend.identificateur.ident==Ident
                        {
                        Valeur_Commodo_Feux=~(tr_rx.data[1]);  //Bitw
                        if(Valeur_Commodo_Feux_Mem!=Valeur_Commodo_Fe
                                {//Its state changed
                                Valeur_Commodo_Feux_Mem=Valeur_Comm
                                //If a indicator is active, we
                                if(Cde_Clign_Gauche || Cde_Cl
                                //Otherwise, we wake up the
value
                                        else m_send(num
                                //we activate the displ
                                Actu_Aff=3;
                                }
                        } //End of stalk handling
                else if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FRG)
                        {
                        Valeur_Status_FRG=tr_rx.d
                        Valeur_FRG=tr_rx.data[0]
                        //we activate the displa
                        Actu_Aff=4;
                        } //End of Lef

                else if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FRD)
                        {
                        Valeur_Status_FRD=tr_rx.data[0];        //Recover the state in the most significant bits of data[0]
                        Valeur_FRD=tr_rx.data[0];               //Recover the asked state in the least significant bits of data[0]

                        //we activate the disp
                        Actu_Aff=5;
                        } //End of Right Back Lights handling

                else if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FVG)
                        {
                        Valeur_Status_FVG=tr_rx.data[0];        //Recover the state in the most significant bits of data[0]
                        Valeur_FVG=tr_rx.data[0];               //Recover the asked state in the least significant bits of data[0]
                        //we activate the display task
                        Actu_Aff=6;
                        }  //End of Left Front Lights handling

                else if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FVD)
                        {
                        Valeur_Status_FVD=tr_rx.data[0];        //Recover the state in the most significant bits of data[0]
                        Valeur_FVD=tr_rx.data[0];               //Recover the asked state in the least significant bits of data[0]
                        //we activate the display task
                        Actu_Aff=7;
                        } //End of Received frames reading
                if(Actu_Aff!=0) active(num_tache(Affichage));
                } //End of Received frames reading
          } //End of while
}
/*****************************************************************************************************************/
/*                                Interruption                                                                   */
/*****************************************************************************************************************/
TACHE irq_Clign(void)
{
int actualise;
static char Flag_Fin_Tempo_Clign;
```

```
while((Cde_Clign_Gauche || Cde_Clign_Droit || Cde_Warning))
          {
          dort(10*TempoClign);                              //Wait for 0.8s
          Flag_Fin_Tempo_Clign^=0x01;   //Enable or disable the flag
//Pay attention to send a command, it checks at first if it is a indicator to show the Warning
          if(((Cde_Clign_Gauche==0) && (Cde_Clign_Droit==0) && (Cde_Warning==0))) break; //If command dropped->we leave
          if(Flag_Fin_Tempo_Clign==0x01) //End of the indicator delay
                    { //for extinction
                    if(Cde_Clign_Droit) actualise=((Valeur_Commodo_Feux^0x20)&0xFD); //turn off the right indicator and
Warning
                    else if(Cde_Clign_Gauche) actualise=((Valeur_Commodo_Feux^0x10)&0xFD); //Turn off the left indicator and
Warning
                    else if (Cde_Warning ) actualise=Valeur_Commodo_Feux^0x02; //Warning
                    m_send(num_tache(Actualise_Etat_Feux),&actualise);
                    }
          else if(Flag_Fin_Tempo_Clign==0x00) //End of the indicator delay
                    { //to turn on
                    if (Cde_Clign_Droit) actualise=Valeur_Commodo_Feux&0xFD; //turn on right indicator and turn off the
warning
                    else if (Cde_Clign_Gauche) actualise=Valeur_Commodo_Feux&0xFD; //turn on left indicator and turn off
warning
                    else if (Cde_Warning ) actualise=Valeur_Commodo_Feux; //turn on Warning
                    m_send(num_tache(Actualise_Etat_Feux),&actualise);
                    }
          }

}

TACHE irq_eg(Tempo)           //task called by reading frame
char Tempo;
{
static Vitesse;
static Valeur_Ana;
Vitesse=Vitesse_Lente;
Flag_Fin_Tempo_EG=0;
while((Valeur_Analogique<200) && (Cde_EG_Av_Pos2==0) && (Cde_EG_Av_Pos1==0) && [illegible]de[illegible]lace_Av==1))
          {
          if(Flag_Fin_Tempo_EG==0)
                    {
                    dort(100*Tempo);
                    Flag_Fin_Tempo_EG=1;
                    m_send(num_tache(Actualise_ASV),&Vitesse);
                    }
          }
}
/*************************************************************[illegible]*********************************/
/*                                                       Check [illegible]e [illegible]igh[illegible] state
/*************************************************************[illegible]*********************************/
TACHE  Verif_Etat_Feux(void)
{
//Left Front Lights state verification frame initializatio[illegible]
Trame T_IRM_Feux;
T_IRM_Feux.trame_info.registre=0x00;
T_IRM_Feux.trame_info.champ.extend=1;
T_IRM_Feux.trame_info.champ.dlc=0x01;
T_IRM_Feux.trame_info.champ.rtr=1;
T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_[illegible]M_F[illegible]
//Send the remote frame on the CAN bus
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificateur.ident=[illegible]nt_T_IR[illegible]
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificat[illegible]ent[illegible]den[illegible];
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identifica[illegible]ur.ide[illegible]M_FVD;
EcrireTrame(&T_IRM_Feux);

//task infinite sleeping (activation on aw[illegible] or [illegible]send)
}
/******************************************[illegible]**********************************************************/
/*                           Display task                                                                  */
/*******************************************************************************************************/
TACHE Affichage(void)
{
static Valeur_FRG_Aff,Valeur_FRD_Aff,Valeur_FVG_Aff,Valeur_FVD_Aff;
static Valeur_Commodo_Feux_Aff;
static Valeur_Commodo_EG_Aff;
static Valeur_Analogique_Aff;
static Valeur_FC_EG_Aff;

if(Valeur_Analogique_Aff!=Valeur_Analogique)
          {
          Valeur_Analogique_Aff=Valeur_Analogique;
          gotoxy(45,3);
          printf("%d    \n",Valeur_Analogique);
          }

if(Valeur_Commodo_EG_Aff!=Valeur_Commodo_EG)
//we update the values of windshield wiper stalk on the Monitor
          {
          Valeur_Commodo_EG_Aff=Valeur_Commodo_EG;
          MONITOR
          gotoxy(22,3);
          printf("%d\n",Cde_EG_Av_Int);
          gotoxy(16,6);
          printf("%d\n",Entree1);
          gotoxy(46,6);
          printf("%d\n",Entree2);
          gotoxy(23,4);
          printf("%d\n",Cde_EG_Av_Pos1);
```

```
          gotoxy(53,4);
          printf("%d\n",Cde_EG_Av_Pos2);
          gotoxy(18,7);
          printf("%d\n",Cde_EG_Ar);
          //The windshield washer commands are inversed on the stalk Les (Active=0 , Inactive=1)
          gotoxy(56,7);
          printf("%d\n",(Cde_Lave_Glace_Ar^0x01));
          gotoxy(27,5);
          printf("%d\n",(Cde_Lave_Glace_Av^0x01));
          ENDM
          }

if(Valeur_FC_EG_Aff!=Valeur_FC_EG)
//we update the values of limit switch on the Monitor
          {
          Valeur_FC_EG_Aff=Valeur_FC_EG;
          MONITOR
          gotoxy(26,15);
          if(fcg)   //End of left active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(62,15);
          if(fcd)   //End of right active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(46,15);
          if(fs)    //End of over limit
                    printf("1\n");
          else
                    printf("0\n");
                    ENDM
          }

if(Valeur_Commodo_Feux_Aff!=Valeur_Commodo_Feux)
//we update the values of lights stalk on the Monitor
          {
          Valeur_Commodo_Feux_Aff=Valeur_Commodo_Feux;
          MONITOR
          gotoxy(18,10);
          printf("%d\n",Cde_Veilleuse);
          gotoxy(36,10);
          printf("%d\n",Cde_Phare);
          gotoxy(55,10);
          printf("%d\n",Cde_Code);
          gotoxy(16,11);
          printf("%d\n",Cde_Warning);
          gotoxy(41,11);
          printf("%d\n",Cde_Clign_Gauche);
          gotoxy(60,11);
          printf("%d\n",Cde_Clign_Droit);
          gotoxy(18,12);
          printf("%d\n",Cde_Stop);
          gotoxy(37,12);
          printf("%d\n",Cde_Klaxon);
          ENDM
          }

if(Valeur_FRG_Aff!=Valeur_Status_FRG)
//Left Back Optical Block status
          {
          Valeur_FRG_Aff=Valeur_FR
          MONITOR
                    if(Veilleuse_F        //                  rning on the side light
                              {                       //We check that the state meets the demands
                              gotoxy(
                              printf("1\n"
                              gotoxy(40,
                              if(S_Veilleus          printf("OK\n");    //State is 1, it's ok
                                        else p       tf("Out of service\n");                //State is 0, the state doesn't meet
the demands
                              }
                    else
                              {
                              gotoxy(26,18);
                              printf("0\n");
                              gotoxy(40,18);
                              printf("  \n");
                              }
                    if(Stop_FRG)        //We ask for turning on the stop light
                              {                   //We check that the state meets the demands
                              gotoxy(26,19);
                              printf("1\n");
                              gotoxy(40,19);
                              if(S_Stop_FRG) printf("OK\n");
                                        else printf("Out of service\n");
                              }
                    else
                              {
                              gotoxy(26,19);
                              printf("0\n");
                              gotoxy(40,19);
                              printf("  \n");
                              }
                    if(Clignot_FRG)     //We ask for turning on the indicator
                              {                   //We check that the state meets the demands
                              gotoxy(26,20);
                              printf("1\n");
                              gotoxy(40,20);
```

```
                                if(S_Clignot_FRG) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,20);
                                printf("0\n");
                                gotoxy(40,20);
                                printf("  \n");
                                }
                        if(Klaxon_FRG)      //We ask for turning on the horn
                                {                   //We check that the state meets the demands
                                gotoxy(26,21);
                                printf("1\n");
                                gotoxy(40,21);
                                if(S_Klaxon_FRG) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,21);
                                printf("0\n");
                                gotoxy(40,21);
                                printf("  \n");
                                }
                ENDM
                }

if(Valeur_FRD_Aff!=Valeur_Status_FRD)
//Right Back Optical Block status
                {
                Valeur_FRD_Aff=Valeur_FRD;
                MONITOR
                        if(Veilleuse_FRD)   //We ask for turning on the side light
                                {                   //We check that the state meet[illegible]e de[illegible]s
                                gotoxy(26,23);
                                printf("1\n");
                                gotoxy(40,23);
                                if(S_Veilleuse_FRD) printf("OK\n");
                                        else printf("Out of service[illegible]ate is 0, the state doesn't meet
the demands
                                }
                        else
                                {
                                gotoxy(26,23);
                                printf("0\n");
                                gotoxy(40,23);
                                printf("  \n");
                                }
                        if(Stop_FRD)        //We ask for turning [illegible] the demands
                                {                   //We chec[illegible]te m[illegible]
                                gotoxy(26,24);
                                printf("1\n");
                                gotoxy(40,24);
                                if(S_Stop_FRD) printf("[illegible]
                                        else printf("[illegible]
                                }
                        else
                                {
                                gotoxy(26,24);
                                printf("0\n");
                                gotoxy(40,24);
                                printf([illegible]\n");
                                }
                        if(Clignot_FRD[illegible] for tur[illegible]g on the indicator
                                {                   //We[illegible]k that the state meets the demands
                                goto[illegible]26,25);
                                print[illegible]1\n");
                                gotoxy([illegible]
                                if(S_Clignot[illegible]0) p[illegible]ntf("OK\n");
                                        e[illegible]ri[illegible]("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,25);
                                printf("0\n");
                                gotoxy(40,25);
                                printf("  \n");
                                }
                ENDM
                }

if(Valeur_FVG_Aff!=Valeur_Status_FVG)
//Left Front Optical Block status
                {
                Valeur_FVG_Aff=Valeur_FVG;
                MONITOR
                        if(Veilleuse_FVG)   //We ask for turning on the side light
                                {                   //We check that the state meets the demands
                                gotoxy(26,27);
                                printf("1\n");
                                gotoxy(40,27);
                                if(S_Veilleuse_FVG) printf("OK\n");    //State is 1, it's ok
                                        else printf("Out of service\n");                //State is 0, the state doesn't meet
the demands
                                }
                        else
                                {
                                gotoxy(26,27);
                                printf("0\n");
                                gotoxy(40,27);
                                printf("  \n");
```

```
                    }
                if(Code_FVG)        //We ask for turning on the dipped light
                        {                   //We check that the state meets the demands
                        gotoxy(26,28);
                        printf("1\n");
                        gotoxy(40,28);
                        if(S_Code_FVG) printf("OK\n");
                                else printf("Out of service\n");
                        }
                else
                        {
                        gotoxy(26,28);
                        printf("0\n");
                        gotoxy(40,28);
                        printf("  \n");
                        }
                if(Phare_FVG)       //We ask for turning on the head light
                        {                   //We check that the state meets the demands
                        gotoxy(26,29);
                        printf("1\n");
                        gotoxy(40,29);
                        if(S_Phare_FVG) printf("OK\n");
                                else printf("Out of service\n");
                        }
                else
                        {
                        gotoxy(26,29);
                        printf("0\n");
                        gotoxy(40,29);
                        printf("  \n");
                        }
                if(Clignot_FVG)     //We ask for turning on the indicator
                        {                   //We check that the state meets the demands
                        gotoxy(26,30);
                        printf("1\n");
                        gotoxy(40,30);
                        if(S_Clignot_FVG) printf("OK\n");
                                else printf("Out of service\n");
                        }
                else
                        {
                        gotoxy(26,30);
                        printf("0\n");
                        gotoxy(40,30);
                        printf("  \n");
                        }
        ENDM
        }

if(Valeur_FVD_Aff!=Valeur_Status_FVD)
//Right Front Optical Block status
        {
        Valeur_FVD_Aff=Valeur_FVD;
        MONITOR
        if(Veilleuse_FVD)   //We ask for turning on the side l
                {                   //We check th  th  te m  the demands
                gotoxy(26,32);
                printf("1\n");
                gotoxy(40,32);
                if(S_Veilleuse_FVD) printf(  e is 1, it's ok
                        else printf("Ou  servic                    //State is 0, the state doesn't meet the
demands
                }
        else
                {
                gotoxy(26,32);
                printf("0\n");
                gotoxy(40,32);
                printf("  \n");
                }
        if(Code_FVD)        //We ask f  rni  on the dipped light
                {                   //W    that the state meets the demands
                gotoxy(26,33);
                printf("1\n");
                gotoxy(40,33);
                if(S_Code_FVD) printf("OK\n");
                        else printf("Out of service\n");
                }
        else
                {
                gotoxy(26,33);
                printf("0\n");
                gotoxy(40,33);
                printf("  \n");
                }
        if(Phare_FVD)       //We ask for turning on the head light
                {                   //We check that the state meets the demands
                gotoxy(26,34);
                printf("1\n");
                gotoxy(40,34);
                if(S_Phare_FVD) printf("OK\n");
                        else printf("Out of service\n");
                }
        else
                {
                gotoxy(26,34);
                printf("0\n");
                gotoxy(40,34);
                printf("  \n");
                }
        if(Clignot_FVD)     //We ask for turning on the indicator
                {                   //We check that the state meets the demands
```

```
                    gotoxy(26,35);
                    printf("1\n");
                    gotoxy(40,35);
                    if(S_Clignot_FVD) printf("OK\n");
                              else printf("Out of service\n");
                    }
          else
                    {
                    gotoxy(26,35);
                    printf("0\n");
                    gotoxy(40,35);
                    printf("  \n");
                    }
          ENDM
          }
}
/*****************************************************************************************************************/
/*                                        Leave completely                                                       */
/*****************************************************************************************************************/
TACHE Quit(void)
{
//Stop th modules
Trame T_IM_Arret;
int Tempo;
//Destroy all tasks
detruit(num_tache(Actualise_ASV));
detruit(num_tache(Actualise_Etat_Feux));
detruit(num_tache(LectureTrame));
detruit(num_tache(irq_Clign));
detruit(num_tache(irq_eg));
detruit(num_tache(Verif_Etat_Feux));
detruit(num_tache(Affichage));

//General Frame Configuration
T_IM_Arret.trame_info.registre=0x00;
T_IM_Arret.trame_info.champ.extend=1;
T_IM_Arret.trame_info.champ.dlc=0x03;

//For the Servo-system board
 //In positive
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_Asservissemen
T_IM_Arret.data[0]=0x25;
T_IM_Arret.data[1]=0xFF;
T_IM_Arret.data[2]=0x00;
EcrireTrame(&T_IM_Arret);
 //in negative
T_IM_Arret.data[0]=0x26;
EcrireTrame(&T_IM_Arret);

//For the Left Back Lights
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_FR
T_IM_Arret.data[0]=0x1e;
T_IM_Arret.data[1]=0x0f;
T_IM_Arret.data[2]=0x00;
EcrireTrame(&T_IM_Arret);

//For the Right Back Lights
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_    ERD;
EcrireTrame(&T_IM_Arret);

//For the Left Front Lights
T_IM_Arret.ident.extend.identificateur.ident  iden    IM_FV
EcrireTrame(&T_IM_Arret);

//For the Right Front Lights
T_IM_Arret.ident.extend.identifica  ur.I                VD;
EcrireTrame(&T_IM_Arret);

//Stop the automatical sending
//Configuration of lights stalk, Windshie  ipe  talk and Servo-system board modules for frames //automatical sending after
a regular time interval
T_IM_Arret.trame_info.registre=0x00;
T_IM_Arret.trame_info.champ.extend=1;
T_IM_Arret.trame_info.champ.dlc=0x03;
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_Commodo_Feux; //lights stalk
T_IM_Arret.data[0]=0x2C;
T_IM_Arret.data[1]=0xFF;
T_IM_Arret.data[2]=0x00;
EcrireTrame(&T_IM_Arret);

T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_Commodo_EG; //Windshield Wiper Stalk
EcrireTrame(&T_IM_Arret);

T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_Asservissement; //Servo-system board
EcrireTrame(&T_IM_Arret);


mtr86exit(0);
}

/*****************************************************************************************************************/
/*                                          Main Program                                                         */
/*****************************************************************************************************************/
main()
{
clsscr();
start_mtr(Init_VMD,2000);
}
```

Frames sending for state changing

```
/* VMD management: optical blocks, Lights Stalk, Windshield Wiper and its Stalk */
//***************************
#include <stdio.h>
#include "Structures_donnees.h"
#include "mtr86.h"
#include "vmd.h"
#include "eid210.h"

#define Ident_T_OB_Commodo_Feux 0x05100000 // "Light Stalk"  On Bus (by test manager)

//For the Windshield Wiper delay
#define Tempo1 2    // in second
#define Tempo2 4 // For intermittent mode
#define Tempo3 6
#define Tempo4 8
#define Tempo5 10
//Flag of Windshield Wiper delay end
char Flag_Fin_Tempo_EG;
//For the beat rate
#define Vitesse_Rapide 0x90
#define Vitesse_Lente 0x40
#define Vitesse_Lave_Glace 0x60
//For the delay between two indicator's flashing
#define TempoClign 6         //in ms


//Tasks Prototype
TACHE  Init_VMD(void);
TACHE  init  (void);
TACHE      Actualise_ASV();
TACHE      Actualise_Etat_Feux();
TACHE  LectureTrame(void);
TACHE  irq_Clign(void);
TACHE  irq_eg();
TACHE  Verif_Etat_Feux(void);
TACHE  Affichage(void);
TACHE  Quit(void);

/*************************************************************************[illegible]*********************************/
/*                                              VMD initialization [illegible] C[illegible]guration )
                   */
/*******************************************************************[illegible]**********************************/
TACHE Init_VMD(void)
{
//Modules Configuration
Trame T_IM_CFG,T_IRM_Etat_FC,Trame_Recue;
//For the waiting delays between two sends
int Temp,Cptr_TimeOut;
// the identifiers, value,...
int cpt;
//Identifiers,values,data:
//Identifier
int
identificateur[]={Ident_T_IM_Commodo_Feux,Ident_T_IM_[illegible]t_T_IM_Asservissement,Ident_T_IM_FRG,Ident_T_IM_FRD,Ident
_T_IM_FVG,Ident_T_IM_FVD};
//Acknowledge
int
identificateurACK[]={Ident_T_AIM_Commodo_Feu[illegible]do_EG,Ident_T_AIM_Asservissement,Ident_T_AIM_FRG,Ident_T_AIM_F
RD,Ident_T_AIM_FVG,Ident_T_AIM_FVD};
//Table including the data for I /[illegible]
int Config_E_S[]={0xFF,0xFF,0xE3,[illegible]};

//Strings can indicate module init[illegible]
char *msg[]={" Light stalk"," Wind[illegible]eld Wiper[illegible]lk "," Servo-system Module ","Left Back Lights","Right Back Lights","Left
Front Lights","Right Front Lights"};

//Table including all the data..
//The 2 first values are for the Analog/Di[illegible]nvertor of the wiper stalk wheel
//The following are for the Servo-system (cf[illegible], frequency, Power Validation,...)
int datazero[]={0x2A,0x2B,0x21,0x23,0x22,0x24,0x25,0x26,0x1E};
int dataun[]={0xF0,0xFF,0xB3,0xFF,0xB3,0xFF,0xFF,0xFF,0x10};
int datadeux[]={0x80,0x0E,0x80,0xFF,0x80,0xFF,0x00,0x00,0x10};

//CAN and MTR configuration
dsp_stk(); //Know the state of the tasks when we leave through mtr86exit(0);
OpenCAN(ATON_V2);   //ATON_V1 for old ATONCAN boards
                              //ATON_V2 for new ATONCAN boards

T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1; // Work in the extended mode
T_IM_CFG.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
// For the 4 outputs modules configuration (GP0 to GP3) and 4inputs status 4 (GP4 to GP8)
T_IM_CFG.data[0]=0x1F;        // first data -> "Address" of concerned register-> GPDDR
T_IM_CFG.data[1]=0xFF;        // second data -> "Mask"-> See Doc MCP25050 page 16


//Configuration of modules to address and operation of their I/O
for(cpt=0;cpt<7;cpt++)
{
T_IM_CFG.ident.extend.identificateur.ident=identificateur[cpt]; //establishment of the identifier
T_IM_CFG.data[2]=Config_E_S[cpt];       // third data-> "Value"

MONITOR
gotoxy(2,9);        //we display a message to say that which module is being initialized
printf("Initialisation du %s\n",msg[cpt]);

do         {
          EcrireTrame(&T_IM_CFG);// Send a first frame on the CAN network
          Cptr_TimeOut=0;     //The LireTrame function let the task sleep if there aren't any received frames.
```

```
          do{Cptr_TimeOut++;}while((LireTrame(&Trame_Recue)==0)&&(Cptr_TimeOut<200));
          if(Trame_Recue.ident.extend.identificateur.ident==identificateurACK[cpt])Cptr_TimeOut=200; // Test if the identifier
is correct
                    else
                    {
                    printf("Problem\n");
                    Cptr_TimeOut=0;
                    }
          for(Temp=0;Temp<100000;Temp++); // To wait for a while!
          }while(Cptr_TimeOut!=200);
printf("OK\n");
ENDM
}

//Frame configuration
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1;
T_IM_CFG.trame_info.champ.dlc=0x03;

//configuration of Windshield Wiper stalk Ana -> Dig converter
MONITOR
gotoxy(2,9);        //we display a message to say that which module is being configured
printf("Configuration of the  A/D conversion of %s ...\n",msg[1]);
ENDM
T_IM_CFG.ident.extend.identificateur.ident=identificateur[1];
for(cpt=0;cpt<2;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];         // Register address
T_IM_CFG.data[1]=dataun[cpt];           // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];         // Value
EcrireTrame(&T_IM_CFG);
while((LireTrame(&Trame_Recue)==0));
}

//Specific Configuration of Servo-system module
MONITOR
gotoxy(2,9);        //we display a message to say that which module is being c
printf("Specific Configuration of %s ...           \n",msg[2]);
ENDM
T_IM_CFG.ident.extend.identificateur.ident=identificateur[2];
for(cpt=0;cpt<9;cpt++)
{
T_IM_CFG.data[0]=datazero[cpt];         // Register address
T_IM_CFG.data[1]=dataun[cpt];           // Mask -> concerned bits
T_IM_CFG.data[2]=datadeux[cpt];         // Value
EcrireTrame(&T_IM_CFG);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the response
}

//Take the original position
MONITOR
printf("Take the wiper POM \n");
ENDM
//Configuration of limit switch interrogation
T_IRM_Etat_FC.trame_info.registre=0x00;
T_IRM_Etat_FC.trame_info.champ.extend=1; // Work in  e  ed m
T_IRM_Etat_FC.trame_info.champ.dlc=0x01; // There is  dat  b
T_IRM_Etat_FC.trame_info.champ.rtr=1;  //in return
T_IRM_Etat_FC.ident.extend.identificateur.ident=Id  1_As  issement;
EcrireTrame(&T_IRM_Etat_FC);
do{}while(LireTrame(&Trame_Recue)==0); // Wai  or the
Valeur_FC_EG=~(Trame_Recue.data[0]);

if(!(fcd==0 && fcg==1))
          {
          //Configuration of moto  ont
          T_IM_CFG.trame_info.regi  e=0x
          T_IM_CFG.trame_info.champ  tend=1; //  ork in the extended mode
          T_IM_CFG.trame_info.champ.  //  ere are 3 data of 8 bits (3 sent bytes)
          // For the 4 outputs modules Co  ura  n (GP0 to GP3) and 4inputs status 4 (GP4 to GP8)
          T_IM_CFG.data[0]=0x26;
          T_IM_CFG.data[1]=0xFF;
          T_IM_CFG.data[2]=0x40;
          EcrireTrame(&T_IM_CFG);//active the negative speed
          while(!(fcd==0 && fcg==1))
                    {
                    EcrireTrame(&T_IRM_Etat_FC);
                    while(LireTrame(&Trame_Recue)==0){}; //Wait for the response
                    Valeur_FC_EG=~(Trame_Recue.data[0]); // Recover the limit switch state
                    }
          T_IM_CFG.data[2]=0; //cancel the negative speed
          EcrireTrame(&T_IM_CFG); //disable the negative control
          } //End of Servo-system module initialization
EcrireTrame(&T_IRM_Etat_FC);
do{}while(LireTrame(&Trame_Recue)==0); // Wait for the response
Valeur_FC_EG=~(Trame_Recue.data[0]);

MONITOR
clsscr();
//The display takes a lot of time, so we anticipate
gotoxy(1,1);
printf("************************EX_8 CAN bus with mtr86************************\n");
printf(" Windshield Wiper Stalk Status   \n");
printf("  GP0: Cde_EG_Av_Int=         , Potentiometer value=                                                    \n");
printf("  GP3: Cde_EG_Av_Pos1=        , GP4: Cde_EG_Av_Pos2=                    \n");
printf("  GP7: Cde_Lave_Glace_Av=                                                          \n");
printf("  GP1: Inpout1=               , GP2: Input2=                                \n");
printf("  GP5: Cde_EG_Ar=             , GP6: Cde_Lave_Glace_Ar=                  \n");
printf("
\n");
printf(" Lights stalk status                    \n");
printf("  GP0: Side light=      ,GP2: Head light=        ,GP3: Dipped light=                  \n");
```

```
printf("  GP1: Warning=        ,GP4: Left Indicator=   ,GP5: Right Indicator\n");
printf("  GP6: Stop light=      ,GP7: Horn=                                    \n");
printf("
\n");
printf(" Limit Switch State on the Servo-system board                                  \n");
printf("  Limit Switch: Left=            Over-limit=          Right=                 \n");
printf("
\n");
printf(" Left Back Lights:   Action        RealStatus                          \n");
printf("  Side light
\n");
printf("  Stop light
\n");
printf("  Indicator                                                                                  \n");
printf("  Horn
\n");
printf(" Right Back Lights:   Action        RealStatus                           \n");
printf("  Side light
\n");
printf("  Stop light
        \n");
printf("  Indicator                                                                                  \n");
printf(" Left Front Lights:   Action        RealStatus                          \n");
printf("  Side light
\n");
printf("  Dipped light
        \n");
printf("  Head light
        \n");
printf("  Indicator                                                                                  \n");
printf(" Right Front Lights:   Action        RealStatus                            \n");
printf("  Side light
\n");
printf("  Dipped light
        \n");
printf("  Head light
        \n");
printf("  Indicator                                                                                  \n");
ENDM

//Configuration of lights stalk, Windshield Wiper with its stalk and o-sy dules for frames automatical
sending after a regular time interval
T_IM_CFG.trame_info.registre=0x00;
T_IM_CFG.trame_info.champ.extend=1;
T_IM_CFG.trame_info.champ.dlc=0x03;
T_IM_CFG.ident.extend.identificateur.ident=Ident_T_IM_Commodo_Feu  //l ts  lk
//Time base calucation:(1/160000)*4096*(STBF1:STBF0)*(STM3toSTM0 cf  nee  024)
T_IM_CFG.data[0]=0x2C;        //Register address STCON=0x10 + Off  =0  0x
T_IM_CFG.data[1]=0xFF; //All the bits are concerned
T_IM_CFG.data[2]=0xeF; //STEN=1=>Validate automatic frame se  =1=>
                                          //STBF1=1  STM2  =STM0=1 =>Timebase=1 second
                                          //==>we  horize  ati ending every 1 second
EcrireTrame(&T_IM_CFG);

T_IM_CFG.ident.extend.identificateur.ident=Ident_T_IM  mmodo_Ec  shield Wiper Stalk
EcrireTrame(&T_IM_CFG);

T_IM_CFG.ident.extend.identificateur.ident=Ident_T_IM  servi  ; //Servo-system board
T_IM_CFG.data[2]=0xd5; //for a more short time int  CF=4  (0xD2=12ms) (0xD5=25ms)
EcrireTrame(&T_IM_CFG);

active(cree(init,1,1024)); //task Initializ  on
}

TACHE init(void)
{
cree(Actualise_ASV,1,512);          //  because lights activation is the most important
cree(Actualise_Etat_Feux,1,512);
active(cree(LectureTrame,2,512));       me  ding will automatically fall asleep
cree(irq_Clign,2,256);                  The IT control for the indicators
cree(irq_eg,2,256);                 //  IT  ntrol for the windshield wiper
cree(Verif_Etat_Feux,4,512);        //C    the status light is less important
cree(Affichage,5,1024);                 //The display retains the lowest priority
cree(Quit,6,256);                   //The task will disable other tasks and leave completely
}
/*****************************************************************************************************************/
/*                              Control the direction of motor rotation
                            */
/*****************************************************************************************************************/

TACHE Actualise_ASV(Vitesse_EG)                 //task called by reading frame
char Vitesse_EG;
{
//Limit switches' status (at t-1) to restore wiper when the over-limit has been activated
static Valeur_FC_EG_Verif_fs;
//Frames to drive the motor rotation direction
Trame T_IM_Cmde_Negative,T_IM_Cmde_Positive;
//Configuration of Negative command frame
T_IM_Cmde_Negative.trame_info.registre=0x00;
T_IM_Cmde_Negative.trame_info.champ.extend=1; // Work in the extended mode
T_IM_Cmde_Negative.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
T_IM_Cmde_Negative.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
T_IM_Cmde_Negative.data[0]=0x26;        // PWM2DC register address
T_IM_Cmde_Negative.data[1]=0xFF;        // Mask -> all the bits are concerned
//Configuration of Positive command frame
T_IM_Cmde_Positive.trame_info.registre=0x00;
T_IM_Cmde_Positive.trame_info.champ.extend=1; // Work in the extended mode
T_IM_Cmde_Positive.trame_info.champ.dlc=0x03; // There are 3 data of 8 bits (3 sent bytes)
T_IM_Cmde_Positive.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
T_IM_Cmde_Positive.data[0]=0x25;        // PWM2DC register address
T_IM_Cmde_Positive.data[1]=0xFF;        // Mask -> all the bits are concerned
```

```
//The windshield washer controls are reversed on the stalk(Active=0 , inactive=1)
if((Cde_EG_Av_Pos2 || Cde_EG_Av_Pos1 || (Cde_Lave_Glace_Av==0))) //a "classic" command is valid
          {
          if(fcd)                                 //If it's at right side, return to left
                    {
                    T_IM_Cmde_Negative.data[2]=Vitesse_EG;
                    T_IM_Cmde_Positive.data[2]=0x00;
                    }
          else if(fcg)                            //If it's at left side, return to right
                    {
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=Vitesse_EG;
                    }
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(Valeur_Analogique<200) //intermittent control is activated, we must manage according to the delay
          {
          if(fcd) //we are at right, it passes to the left
                    {
                    T_IM_Cmde_Negative.data[2]=Vitesse_EG;
                    T_IM_Cmde_Positive.data[2]=0x00;
                    Flag_Fin_Tempo_EG=0;                                        //we cancel the flag of delay end
                    }
          else if ((Flag_Fin_Tempo_EG)) //the delay is finished
                    {
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=Vitesse_EG;  //we launch the motor
                    }
          else if (fcg && Flag_Fin_Tempo_EG==0)
                    { //at left, in intermittent position, but the delay isn't finished
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=0x00;       //we stop the motor
                    }
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fcd) //it's at the right without any command, return to left
          {
          T_IM_Cmde_Negative.data[2]=0x40;
          T_IM_Cmde_Positive.data[2]=0x00;
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fcg) //it's at the left without any command, return to ri
          {
          T_IM_Cmde_Negative.data[2]=0x00;
          T_IM_Cmde_Positive.data[2]=0x00;
          EcrireTrame(&T_IM_Cmde_Positive);
          EcrireTrame(&T_IM_Cmde_Negative);
          }
else if(fs)//error, it's over limit
          {
          if((Valeur_FC_EG_Verif_fs&0x40)==0x40) //for      e      ur until the left limit switch
                    {
                    T_IM_Cmde_Negative.data[2]=0x00;
                    T_IM_Cmde_Positive.data[2]=0x40;
                    EcrireTrame(&T_IM_Cmde_Negative);
                    EcrireTrame(&T_IM_Cmde_Positive);
                    }
          if((Valeur_FC_EG_Verif_fs&0x20)==0x      //for      t, return until the right limit switch
                    {
                    T_IM_Cmde_Negati data[ =0x
                    T_IM_Cmde_Posit a[ =0x
                    EcrireTrame(&T _C e tive);
                    EcrireTrame(&T M_C e);
                    }
          }
Valeur_FC_EG_Verif_fs=Valeur_FC_EG;
}
/*****************************************      ***************************************************************************/
/*                  Turn on the lights
*/
/*****************************************************************************************************************************/
TACHE Actualise_Etat_Feux(Value_Commodo)                  //task called by reading frame
Port_8ES Value_Commodo; //the variable is under the form of a structure of Port_8ES type
                                    //which allows us a direct access to the bit level
{
Trame T_IM_Feux,T_recue; //Frame that will be set up to address the optical blocks
Valeur_FRG=0x00;    //sets optical blocks to 0
Valeur_FRD=0x00;
Valeur_FVG=0x00;
Valeur_FVD=0x00;

if(Value_Commodo.bit.GP2) //Head light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_P;
          Valeur_FVD|=Cde_FV_P;
          }
else if(Value_Commodo.bit.GP3) //Dipped light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
          Valeur_FVG|=Cde_FV_C;
          Valeur_FVD|=Cde_FV_C;
          }
else if(Value_Commodo.bit.GP0) //Side light
          {
          Valeur_FRG|=Cde_FR_V;
          Valeur_FRD|=Cde_FR_V;
```

```
          Valeur_FVG|=Cde_FV_V;
          Valeur_FVD|=Cde_FV_V;
          }

if(Value_Commodo.bit.GP1)     //Warning
          {
          Valeur_FRG|=Masque_Clign_AR;
          Valeur_FRD|=Masque_Clign_AR;
          Valeur_FVG|=Masque_Clign_AV;
          Valeur_FVD|=Masque_Clign_AV;
          }
          else
          {
          if(Value_Commodo.bit.GP4) //Left Indicator
                    {
                    Valeur_FRG|=Masque_Clign_AR;
                    Valeur_FVG|=Masque_Clign_AV;
                    }
          if(Value_Commodo.bit.GP5) //Right indicator
                    {
                    Valeur_FRD|=Masque_Clign_AR;
                    Valeur_FVD|=Masque_Clign_AV;
                    }
          }

if(Value_Commodo.bit.GP6)//Stop light
          {
          Valeur_FRG|=Masque_Stop;
          Valeur_FRD|=Masque_Stop;
          }

if(Value_Commodo.bit.GP7) //Horn
          Valeur_FRG|=Masque_Klaxon;

//initialization of the command frame on the lights
T_IM_Feux.trame_info.registre=0x00;
T_IM_Feux.trame_info.champ.extend=1;
T_IM_Feux.trame_info.champ.dlc=0x03;
T_IM_Feux.data[0]=0x1e;
T_IM_Feux.data[1]=0x0f;

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FRG;
T_IM_Feux.data[2]=Valeur_FRG;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FRD;
T_IM_Feux.data[2]=Valeur_FRD;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM_FVG;
T_IM_Feux.data[2]=Valeur_FVG;
EcrireTrame(&T_IM_Feux);

T_IM_Feux.ident.extend.identificateur.ident=Ident_T_IM
T_IM_Feux.data[2]=Valeur_FVD;
EcrireTrame(&T_IM_Feux);

//we just made a lights modification request  .
active(num_tache(Verif_Etat_Feux)); // we check the li   stat
}
/*************************************************************************************************************/
/*                                Readi   re   ed fr
                              */
/*************************************************************************************************************/
TACHE LectureTrame(void)
{
static Trame tr_rx;
static Valeur_Commodo_Feux_Mem;         //static   riable as it is a memorization
static Valeur_Commodo_EG_Mem;
static Valeur_FC_EG_Mem;
static Actu_Aff;
static Valeur_Analogique_Mem; //defined in       because it is a parameter to transmit
static Vitesse_EG;
static Tempo_EG;
//This frame allows us to define the transition on a rising or falling front
Cfg_IointP0.trame_info.registre=0x00;
Cfg_IointP0.trame_info.champ.extend=1;
Cfg_IointP0.trame_info.champ.dlc=0x03;
Cfg_IointP0.data[0]=0x1D;     //Register address IOINTP0=0x01 + Offset=0x1C==>0x1D
Cfg_IointP0.data[1]=0xFF; //All the bits are concerned

while(1)                                //Task activates permanently
          {
          if(LireTrame(&tr_rx))  //LireTrame stop the task when there aren't any received frame
             {
         if(CRTL==0) active(num_tache(Quit)); //If CRTL key is pressed, we leave
             Actu_Aff=0;

             if (tr_rx.ident.extend.identificateur.ident==0x00C00000) //we received a servo-system response
                    {
                    Valeur_FC_EG=~(tr_rx.data[1]);
                              if(Valeur_FC_EG_Mem!=Valeur_FC_EG)
                              {
                              m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                              Actu_Aff=2;
                               //the transition is changed depending on the state of the inputs
                              // If it was the rising front, we change it to the falling front or inverse
                              Cfg_IointP0.ident.extend.identificateur.ident=Ident_T_IM_Asservissement;
                              Cfg_IointP0.data[2]=Valeur_FC_EG;
                              EcrireTrame(&Cfg_IointP0);
                              }
                              else if(fs==1) m_send(num_tache(Actualise_ASV),&Vitesse_EG);
```

```
                    }//End of limit switch handling

            if (tr_rx.ident.extend.identificateur.ident==0x05C00000) //we received a windshield wiper stalk response
                    {
                    Valeur_Commodo_EG=~(tr_rx.data[1]);     //recover the status of windshield wiper stalk
                    Valeur_Analogique=tr_rx.data[2];        //recover the value of potentiometer
                    //we determine the potar position to activate the delay
                    if(Valeur_Analogique!=Valeur_Analogique_Mem)
                            {
                            Actu_Aff=1;
                            Valeur_Commodo_EG_Mem=Valeur_Commodo_EG;
                            Vitesse_EG=0;
                            //Disable the intermittent position if it was selected
                            if(Valeur_Analogique<200) detruit(num_tache(irq_eg));
                            //...Handle the selected position...
                            if(Cde_EG_Av_Pos2) Vitesse_EG=Vitesse_Rapide;
                            else if(Cde_EG_Av_Pos1) Vitesse_EG=Vitesse_Lente;
                            else if(Cde_Lave_Glace_Av==0) Vitesse_EG=Vitesse_Lave_Glace; //stalk inversion
                            //...and the reactivate if it is always actual
                            else if(Cde_EG_Av_Int && (Valeur_Analogique<200))
                                    {
                                    Vitesse_EG=Vitesse_Lente;
                                    m_send(num_tache(irq_eg),&Tempo_EG);
                                    }
                            // the transition is changed depending on the state of the inputs
                            // If it was the rising front, we change it to the falling front or inverse
                            Cfg_IointP0.ident.extend.identificateur.ident=Ident_T_IM_Commodo_EG;
                            Cfg_IointP0.data[2]=Valeur_Commodo_EG;
                            EcrireTrame(&Cfg_IointP0);
                            m_send(num_tache(Actualise_ASV),&Vitesse_EG);
                            }
                    } //End of windshield wiper stalk handling


            if (tr_rx.ident.extend.identificateur.ident==0x05400000) //we receive[illegible]alk response
                    {
                    Valeur_Commodo_Feux=~(tr_rx.data[1]);  // Bitwise inversi[illegible]eal state
                    if(Valeur_Commodo_Feux_Mem!=Valeur_Commodo_Feux)
                            {// Its state changed
                            Valeur_Commodo_Feux_Mem=Valeur_Commodo[illegible]ux;  [illegible]pdate [illegible]stored value
                            // If a indicator is active, we validat[illegible] If [illegible]tivate Actualise_Etat_Feux
                            if(Cde_Clign_Gauche || Cde_Clign_Droit ||[illegible]Warn[illegible]ctive(num_tache(irq_Clign));
                            // Otherwise, we wake up the lights state a[illegible]sati[illegible]sk with the stalk without indicator
value
                                    else m_send(num_tache(Actu[illegible]e_Etat_[illegible]aleur_Commodo_Feux_Mem);
                            // we activate the display task
                            Actu_Aff=3;
                            // the transition is changed depend[illegible] o[illegible]st[illegible] of the inputs
                            // If it was the rising front, we c[illegible]e [illegible] falling front or inverse)
                            Cfg_IointP0.ident.extend.identif[illegible]ident[illegible]t_T_IM_Commodo_Feux;
                            Cfg_IointP0.data[2]=Valeur_C[illegible]
                            EcrireTrame(&Cfg_IointP0);
                            }
                    } // End of lights stalk handling
              if (tr_rx.ident.extend.identificateur.ide[illegible]=Ident_[illegible])
                    {
                    Valeur_Status_FRG=tr_rx.data[0];       [illegible]the state in the most significant bits of data[0]
                    Valeur_FRG=tr_rx.data[0];              [illegible]the asked state in the least significant bits of data[0]
                    //we activate the display task
                    Actu_Aff=4;
                    } //End of Left Back Ligh[illegible]handling

             if (tr_rx.ident.extend.identifi[illegible]teu[illegible]=[illegible]nt_T_IRM_FRD)
                    {
                    Valeur_Status_[illegible]=tr[illegible]d[illegible][0];       //Recover the state in the most significant bits of data[0]
                    Valeur_FRD=tr_[illegible]dat[illegible]                       //Recover the asked state in the least significant bits of data[0]

                    //we activate t[illegible]display t[illegible]
                    Actu_Aff=5;
                    } //End of Right Back [illegible]hts [illegible]ndling

             if (tr_rx.ident.extend.identi[illegible].ident==Ident_T_IRM_FVG)
                    {
                    Valeur_Status_FVG=tr_rx.data[0];       //Recover the state in the most significant bits of data[0]
                    Valeur_FVG=tr_rx.data[0];              //Recover the asked state in the least significant bits of data[0]
                    //we activate the display task
                    Actu_Aff=6;
                    }  //End of Left Front Lights handling

              if (tr_rx.ident.extend.identificateur.ident==Ident_T_IRM_FVD)
                    {
                    Valeur_Status_FVD=tr_rx.data[0];       //Recover the state in the most significant bits of data[0]
                    Valeur_FVD=tr_rx.data[0];              //Recover the asked state in the least significant bits of data[0]
                    //we activate the display task
                    Actu_Aff=7;
                    } //End of Received frames reading
             if(Actu_Aff!=0) active(num_tache(Affichage));
             } //End of Received frames reading
          } //End of while
}
/*****************************************************************************************************************/
/*                              Interruption
                            */
/*****************************************************************************************************************/
TACHE irq_Clign(void)
{
int actualise;
static char Flag_Fin_Tempo_Clign;

while((Cde_Clign_Gauche || Cde_Clign_Droit || Cde_Warning))
          {
          dort(10*TempoClign);                          //Wait for 0.8s
```

```
          Flag_Fin_Tempo_Clign^=0x01;   //Enable or disable the flag
//Pay attention to send a command, it checks at first if it is a indicator to show the Warning
          if(((Cde_Clign_Gauche==0) && (Cde_Clign_Droit==0) && (Cde_Warning==0))) break; //If command dropped->we leave
          if(Flag_Fin_Tempo_Clign==0x01) //End of the indicator delay
                    { //for extinction
                    if(Cde_Clign_Droit) actualise=((Valeur_Commodo_Feux^0x20)&0xFD); //turn off the right indicator and
Warning
                    else if(Cde_Clign_Gauche) actualise=((Valeur_Commodo_Feux^0x10)&0xFD); //Turn off the left indicator and
Warning
                    else if (Cde_Warning ) actualise=Valeur_Commodo_Feux^0x02; //Warning
                    m_send(num_tache(Actualise_Etat_Feux),&actualise);
                    }
          else if(Flag_Fin_Tempo_Clign==0x00) //End of the indicator delay
                    { //to turn on
                    if (Cde_Clign_Droit) actualise=Valeur_Commodo_Feux&0xFD; //turn on right indicator and turn off the
warning
                    else if (Cde_Clign_Gauche) actualise=Valeur_Commodo_Feux&0xFD; //turn on left indicator and turn off
warning
                    else if (Cde_Warning ) actualise=Valeur_Commodo_Feux; //turn on Warning
                    m_send(num_tache(Actualise_Etat_Feux),&actualise);
                    }
          }

}

TACHE irq_eg(Tempo)          //task called by fram reading
char Tempo;
{
static Vitesse;
static Valeur_Ana;
Vitesse=Vitesse_Lente;
Flag_Fin_Tempo_EG=0;
while((Valeur_Analogique<200) && (Cde_EG_Av_Pos2==0) && (Cde_EG_Av_Pos1==0) && (Cde_Lave_Glace_Av==1))
          {
          if(Flag_Fin_Tempo_EG==0)
                    {
                    dort(100*Tempo);
                    Flag_Fin_Tempo_EG=1;
                    m_send(num_tache(Actualise_ASV),&Vitesse);
                    }
          }
}
/*****************************************************************************************************************/
/*                                                Check the real lig                                             */
/*****************************************************************************************************************/
TACHE  Verif_Etat_Feux(void)
{
//Left Front Lights state verification frame initialization
Trame T_IRM_Feux;
T_IRM_Feux.trame_info.registre=0x00;
T_IRM_Feux.trame_info.champ.extend=1;
T_IRM_Feux.trame_info.champ.dlc=0x01;
T_IRM_Feux.trame_info.champ.rtr=1;
T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_I      G;
//Send the remote frame on the CAN bus
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificateur.ident=Ident_T_      FRD;
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificateur.ident=      nt_T_IR
EcrireTrame(&T_IRM_Feux);

T_IRM_Feux.ident.extend.identificat      ent_iden      ;
EcrireTrame(&T_IRM_Feux);

//task infinite sleeping (activati      on a      )
}
/*****************************************************************************************************************/
/*                                                Display task                                                   */
                                                  */
/*****************************************************************************************************************/
TACHE Affichage(void)
{
static Valeur_FRG_Aff,Valeur_FRD_Aff,Valeur_FVG_Aff,Valeur_FVD_Aff;
static Valeur_Commodo_Feux_Aff;
static Valeur_Commodo_EG_Aff;
static Valeur_Analogique_Aff;
static Valeur_FC_EG_Aff;

if(Valeur_Analogique_Aff!=Valeur_Analogique)
          {
          Valeur_Analogique_Aff=Valeur_Analogique;
          gotoxy(45,3);
          printf("%d   \n",Valeur_Analogique);
          }

if(Valeur_Commodo_EG_Aff!=Valeur_Commodo_EG)
//we update the values of windshield wiper stalk on the Monitor
          {
          Valeur_Commodo_EG_Aff=Valeur_Commodo_EG;
          MONITOR
          gotoxy(22,3);
          printf("%d\n",Cde_EG_Av_Int);
          gotoxy(16,6);
          printf("%d\n",Entree1);
          gotoxy(46,6);
          printf("%d\n",Entree2);
          gotoxy(23,4);
          printf("%d\n",Cde_EG_Av_Pos1);
          gotoxy(53,4);
          printf("%d\n",Cde_EG_Av_Pos2);
```

```
          gotoxy(18,7);
          printf("%d\n",Cde_EG_Ar);
          //The windshield washer commands are inversed on the stalk Les (Active=0 , Inactive=1)
          gotoxy(56,7);
          printf("%d\n",(Cde_Lave_Glace_Ar^0x01));
          gotoxy(27,5);
          printf("%d\n",(Cde_Lave_Glace_Av^0x01));
          ENDM
          }

if(Valeur_FC_EG_Aff!=Valeur_FC_EG)
//we update the values of limit switch on the Monitor
          {
          Valeur_FC_EG_Aff=Valeur_FC_EG;
          MONITOR
          gotoxy(26,15);
          if(fcg)   //End of left active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(62,15);
          if(fcd)   //End of right active progress
                    printf("1\n");
          else
                    printf("0\n");

          gotoxy(46,15);
          if(fs)    //End of over limit
                    printf("1\n");
          else
                    printf("0\n");
                    ENDM
          }

if(Valeur_Commodo_Feux_Aff!=Valeur_Commodo_Feux)
//we update the values of lights stalk on the Monitor
          {
          Valeur_Commodo_Feux_Aff=Valeur_Commodo_Feux;
          MONITOR
          gotoxy(18,10);
          printf("%d\n",Cde_Veilleuse);
          gotoxy(36,10);
          printf("%d\n",Cde_Phare);
          gotoxy(55,10);
          printf("%d\n",Cde_Code);
          gotoxy(16,11);
          printf("%d\n",Cde_Warning);
          gotoxy(41,11);
          printf("%d\n",Cde_Clign_Gauche);
          gotoxy(60,11);
          printf("%d\n",Cde_Clign_Droit);
          gotoxy(18,12);
          printf("%d\n",Cde_Stop);
          gotoxy(37,12);
          printf("%d\n",Cde_Klaxon);
          ENDM
          }

if(Valeur_FRG_Aff!=Valeur_Status_FRG)
//Left Back Optical Block status
          {
          Valeur_FRG_Aff=Valeur_FRG;
          MONITOR
                    if(Veilleuse_F                    for tur    g on the side light
                              {                                  //We  ck that the state meets the demands
                              goto   26,18);
                              print    1\n");
                              gotoxy(
                              if(S_Veilleu     FRG) printf("OK\n");    //State is 1, it's ok
                                        e    print ("Out of service\n");                //State is 0, the state doesn't meet
the demands
                              }
                    else
                              {
                              gotoxy(26,18);
                              printf("0\n");
                              gotoxy(40,18);
                              printf("  \n");
                              }
                    if(Stop_FRG)        //We ask for turning on the stop light
                              {                   //We check that the state meets the demands
                              gotoxy(26,19);
                              printf("1\n");
                              gotoxy(40,19);
                              if(S_Stop_FRG) printf("OK\n");
                                        else printf("Out of service\n");
                              }
                    else
                              {
                              gotoxy(26,19);
                              printf("0\n");
                              gotoxy(40,19);
                              printf("  \n");
                              }
                    if(Clignot_FRG)     //We ask for turning on the indicator
                              {                   //We check that the state meets the demands
                              gotoxy(26,20);
                              printf("1\n");
                              gotoxy(40,20);
                              if(S_Clignot_FRG) printf("OK\n");
                                        else printf("Out of service\n");
```

```
                        }
                else
                        {
                        gotoxy(26,20);
                        printf("0\n");
                        gotoxy(40,20);
                        printf("  \n");
                        }
                if(Klaxon_FRG)      //We ask for turning on the horn
                        {                   //We check that the state meets the demands
                        gotoxy(26,21);
                        printf("1\n");
                        gotoxy(40,21);
                        if(S_Klaxon_FRG) printf("OK\n");
                                else printf("Out of service\n");
                        }
                else
                        {
                        gotoxy(26,21);
                        printf("0\n");
                        gotoxy(40,21);
                        printf("  \n");
                        }
        ENDM
        }

if(Valeur_FRD_Aff!=Valeur_Status_FRD)
//Right Back Optical Block status
        {
        Valeur_FRD_Aff=Valeur_FRD;
        MONITOR
                if(Veilleuse_FRD)   //We ask for turning on the side light
                        {                   //We check that the state meets the demands
                        gotoxy(26,23);
                        printf("1\n");
                        gotoxy(40,23);
                        if(S_Veilleuse_FRD) printf("OK\n");     //State i[illegible]
                                else printf("Out of service\n");[illegible]te is 0, the state doesn't meet
the demands
                        }
                else
                        {
                        gotoxy(26,23);
                        printf("0\n");
                        gotoxy(40,23);
                        printf("  \n");
                        }
                if(Stop_FRD)        //We ask for turning on t[illegible]
                        {                   //We check that [illegible] the demands
                        gotoxy(26,24);
                        printf("1\n");
                        gotoxy(40,24);
                        if(S_Stop_FRD) printf("OK\n
                                else printf("O[illegible]
                        }
                else
                        {
                        gotoxy(26,24);
                        printf("0\n");
                        gotoxy(40,24);
                        printf("  \n");
                        }
                if(Clignot_FRD)     //We [illegible] the indicator
                        {                   [illegible] that the state meets the demands
                        goto[illegible]26,
                        prin[illegible]"1\
                        goto[illegible]40,
                        if(S_[illegible]not_FRD)[illegible]ntf("OK\n");
                                pri[illegible]"Out of service\n");
                        }
                else
                        {
                        gotoxy(26,25);
                        printf("0\n");
                        gotoxy(40,25);
                        printf("  \n");
                        }
        ENDM
        }

if(Valeur_FVG_Aff!=Valeur_Status_FVG)
//Left Front Optical Block status
        {
        Valeur_FVG_Aff=Valeur_FVG;
        MONITOR
                if(Veilleuse_FVG)   //We ask for turning on the side light
                        {                   //We check that the state meets the demands
                        gotoxy(26,27);
                        printf("1\n");
                        gotoxy(40,27);
                        if(S_Veilleuse_FVG) printf("OK\n");     //State is 1, it's ok
                                else printf("Out of service\n");                //State is 0, the state doesn't meet
the demands
                        }
                else
                        {
                        gotoxy(26,27);
                        printf("0\n");
                        gotoxy(40,27);
                        printf("  \n");
                        }
                if(Code_FVG)        //We ask for turning on the dipped light
```

```
                                {                       //We check that the state meets the demands
                                gotoxy(26,28);
                                printf("1\n");
                                gotoxy(40,28);
                                if(S_Code_FVG) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,28);
                                printf("0\n");
                                gotoxy(40,28);
                                printf("  \n");
                                }
                        if(Phare_FVG)       //We ask for turning on the head light
                                {                       //We check that the state meets the demands
                                gotoxy(26,29);
                                printf("1\n");
                                gotoxy(40,29);
                                if(S_Phare_FVG) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,29);
                                printf("0\n");
                                gotoxy(40,29);
                                printf("  \n");
                                }
                        if(Clignot_FVG)     //We ask for turning on the indicator
                                {                       //We check that the state meets the demands
                                gotoxy(26,30);
                                printf("1\n");
                                gotoxy(40,30);
                                if(S_Clignot_FVG) printf("OK\n");
                                        else printf("Out of service\n");
                                }
                        else
                                {
                                gotoxy(26,30);
                                printf("0\n");
                                gotoxy(40,30);
                                printf("  \n");
                                }
                ENDM
                }

if(Valeur_FVD_Aff!=Valeur_Status_FVD)
//Right Front Optical Block status
                {
                Valeur_FVD_Aff=Valeur_FVD;
                MONITOR
                if(Veilleuse_FVD)   //We ask for turning on the side light
                        {                       //We check that the state meets the demands
                        gotoxy(26,32);
                        printf("1\n");
                        gotoxy(40,32);
                        if(S_Veilleuse_FVD) printf("OK\n");     //State is 1, it's ok
                                else printf("Out of service\n");                       //State is 0, the state doesn't meet the
demands
                        }
                else
                        {
                        gotoxy(26,32);
                        printf("0\n");
                        gotoxy(40,32);
                        printf("  \n");
                        }
                if(Code_FVD)        //We ask for turning on the dipped light
                        {                       //We check that the state meets the demands
                        gotoxy(26,33);
                        printf("1\n");
                        gotoxy(40,33);
                        if(S_Code_FVD) printf("OK\n");
                                else printf("Out of service\n");
                        }
                else
                        {
                        gotoxy(26,33);
                        printf("0\n");
                        gotoxy(40,33);
                        printf("  \n");
                        }
                if(Phare_FVD)       //We ask for turning on the head light
                        {                       //We check that the state meets the demands
                        gotoxy(26,34);
                        printf("1\n");
                        gotoxy(40,34);
                        if(S_Phare_FVD) printf("OK\n");
                                else printf("Out of service\n");
                        }
                else
                        {
                        gotoxy(26,34);
                        printf("0\n");
                        gotoxy(40,34);
                        printf("  \n");
                        }
                if(Clignot_FVD)     //We ask for turning on the indicator
                        {                       //We check that the state meets the demands
                        gotoxy(26,35);
                        printf("1\n");
```

```
                    gotoxy(40,35);
                    if(S_Clignot_FVD) printf("OK\n");
                              else printf("Out of service\n");
                    }
          else
                    {
                    gotoxy(26,35);
                    printf("0\n");
                    gotoxy(40,35);
                    printf("  \n");
                    }
          ENDM
          }
}
/*****************************************************************************************************************/
/*                                    Leave completely
                    */
/*****************************************************************************************************************/
TACHE Quit(void)
{
//Stop th modules
Trame T_IM_Arret;
int Tempo;
//Destroy all tasks
detruit(num_tache(Actualise_ASV));
detruit(num_tache(Actualise_Etat_Feux));
detruit(num_tache(LectureTrame));
detruit(num_tache(irq_Clign));
detruit(num_tache(irq_eg));
detruit(num_tache(Verif_Etat_Feux));
detruit(num_tache(Affichage));

//General Frame Configuration
T_IM_Arret.trame_info.registre=0x00;
T_IM_Arret.trame_info.champ.extend=1;
T_IM_Arret.trame_info.champ.dlc=0x03;

//For the Servo-system board
 //In positive
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_Asservissem
T_IM_Arret.data[0]=0x25;
T_IM_Arret.data[1]=0xFF;
T_IM_Arret.data[2]=0x00;
EcrireTrame(&T_IM_Arret);
 //in negative
T_IM_Arret.data[0]=0x26;
EcrireTrame(&T_IM_Arret);

//For the Left Back Lights
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_FRG;
T_IM_Arret.data[0]=0x1e;
T_IM_Arret.data[1]=0x0f;
T_IM_Arret.data[2]=0x00;
EcrireTrame(&T_IM_Arret);

//For the Right Back Lights
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_
EcrireTrame(&T_IM_Arret);

//For the Left Front Lights
T_IM_Arret.ident.extend.identificateur.ident=
EcrireTrame(&T_IM_Arret);

//For the Right Front Lights
T_IM_Arret.ident.extend.identificateur.
EcrireTrame(&T_IM_Arret);

//Stop the automatical sending
//Configuration of lights stalk, Wind
a regular time interval
T_IM_Arret.trame_info.registre=0x00;
T_IM_Arret.trame_info.champ.extend=1;
T_IM_Arret.trame_info.champ.dlc=0x03;
T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_Commodo_Feux; //lights stalk
T_IM_Arret.data[0]=0x1C;
T_IM_Arret.data[1]=0xFF;
T_IM_Arret.data[2]=0x00;
EcrireTrame(&T_IM_Arret);

T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_Commodo_EG; //Windshield Wiper Stalk
EcrireTrame(&T_IM_Arret);

T_IM_Arret.ident.extend.identificateur.ident=Ident_T_IM_Asservissement; //Servo-system board
EcrireTrame(&T_IM_Arret);


mtr86exit(0);
}

/*****************************************************************************************************************/
/*                                    Main Program
                    */
/*****************************************************************************************************************/
main()
{
clsscr();
start_mtr(Init_VMD,2000);
}
```