LAB – 1 SIU.C FILE

```
#include "siu.h"

void Siu_Init() {

        /*********************************************************************
         *                  Task 1                    *
         *********************************************************************/
  /* Plate configuration LED */
        SIU.PCR[9].R  = 0x0200; //P
        SIU.PCR[36].R  = 0x0200; //Rx
        SIU.PCR[37].R  = 0x0200; //Tx
        SIU.PCR[8].R  = 0x0200; //U3
        SIU.PCR[7].R  = 0x0200;//U2
        SIU.PCR[6].R  = 0x0200;//U1

/*************************************************************************
*                   Task 2                    *
*              Analog inputs configuration              *
*************************************************************************/

        /* Analog inputs */

        /* LDR */
        SIU.PCR[32].R  = 0x02500;

  /* Potentiometer */
        SIU.PCR[66].R  = 0x02500;

/*************************************************************************
*              Button and Switches configuration            *
*************************************************************************/
  /* SW 1-4 and BT 5-6 */

        SIU.PCR[33].R  = 0x0100;
        SIU.PCR[34].R  = 0x0100;
        SIU.PCR[73].R  = 0x0100;
        SIU.PCR[74].R  = 0x0100;

        SIU.PCR[78].R  = 0x0100;
        SIU.PCR[83].R  = 0x0100;

}
```

LAB – 1 MAIN.C FILE

```
#include "me_init.h"

/*****************************************************************
 *          Global variables and function declarations          *
 *****************************************************************/
void showData(int value);
int SW1,SW2,SW3,SW4,BT5,BT6,value;

/*****************************************************************
 *                  Application entry point                      *
 *****************************************************************/
int main(void) {

  /* Board and modules initialization */
  ME_Init();

  /* Configure and start timmer channels */
          PIT_ConfigureTimer(1,1000);

  /* Application main loop that runs forever*/
for ( ; ; ) {

          if(SIU.GPDI[74].R == 1)
          {
           value = ADC1.CDR[66].B.CDATA;
                    showDataA(value);
          }
          else
          {
                    value = ADC0.CDR[32].B.CDATA;
                              showDataB(value);
          }

          if (SIU.GPDI[33].R == 1)
          {
                    PIT_StartTimer(1);
          }
          else
          {
                    PIT_StopTimer(1);
          }

  /* Operating System Delay*/
  osalThreadDelayMilliseconds(250UL);
```

```
 }
}
/*********************************************************************
 *                  Task 2                          *
 *********************************************************************/
void showDataA(int value)
{
/*********************************************************************
 *                                                   *
 *              _____ ___  ___  ___                  *
 *             |_   _|/ _ \|   \/ _ \                *
 *               | | | (_) | |) | (_) |               *
 *               |_|  \___/|___/ \___/                *
 *                                                   *
 *********************************************************************/
 if(value <= 256)
 {
   /* U1 - on, U2 - off, U3 - off, P - off */

                              SIU.GDPO[6].R=1;
                              SIU.GDPO[7].R=0;
                              SIU.GDPO[8].R=0;
                              SIU.GDPO[9].R=0;
 }
 else if(value > 256 && value <= 512)
 {
   /* U1 - on, U2 - on, U3 - off, P - off */

            SIU.GDPO[6].R=1;
                                    SIU.GDPO[7].R=1;
                                    SIU.GDPO[8].R=0;
                                    SIU.GDPO[9].R=0;

 }
 else if(value > 512 && value <= 768)
 {
   /* U1 - on, U2 - on, U3 - on, P - off */

            SIU.GDPO[6].R=1;
                                    SIU.GDPO[7].R=1;
                                    SIU.GDPO[8].R=1;
                                    SIU.GDPO[9].R=0;
 }
 else
 {
```

```
   /* U1 - on, U2 - on, U3 - on, P - on */

            SIU.GDPO[6].R=1;
                                    SIU.GDPO[7].R=1;
                                    SIU.GDPO[8].R=1;
                                    SIU.GDPO[9].R=1;
  }
}

void showDataB(int value)
{
/*********************************************************************
 *                  _____ ___  ___  ___                  *
 *                 |_   _|/ _ \|   \/ _ \                *
 *                   | | | (_) | |) | (_) |               *
 *                   |_|  \___/|___/ \___/                *
 *                                      *
 *********************************************************************/
  if(value <= 205)
  {
   /* U1 - on, U2 - on, U3 - on, Tx - on, P - on */

                            SIU.GDPO[6].R=1;
                            SIU.GDPO[7].R=1;
                            SIU.GDPO[8].R=1;
                            SIU.GDPO[37].R=1;
                            SIU.GDPO[9].R=1;
  }
  else if(value > 205 && value <= 410)
  {
            /* U1 - on, U2 - on, U3 - on, Tx - off, P - off */

            SIU.GDPO[6].R=1;
                                    SIU.GDPO[7].R=1;
                                    SIU.GDPO[8].R=1;
                                    SIU.GDPO[37].R=1;
                                    SIU.GDPO[9].R=0;

  }
  else if(value > 410 && value <= 615)
  {
            /* U1 - on, U2 - on, U3 - on, Tx - off, P - off */

            SIU.GDPO[6].R=1;
```

```
                                        SIU.GDPO[7].R=1;
                                        SIU.GDPO[8].R=1;
                                        SIU.GDPO[37].R=0;
                                        SIU.GDPO[9].R=0;
  }
  else if(value > 615 && value <= 820)
  {
            /* U1 - on, U2 - on, U3 - off, Tx - off, P - off */

            SIU.GDPO[6].R=1;
                                        SIU.GDPO[7].R=1;
                                        SIU.GDPO[8].R=0;
                                        SIU.GDPO[37].R=0;
                                        SIU.GDPO[9].R=0;
  }
  else
  {
            /* U1 - on, U2 - off, U3 - off, Tx - off, P - off */

            SIU.GDPO[6].R=1;
                                        SIU.GDPO[7].R=0;
                                        SIU.GDPO[8].R=0;
                                        SIU.GDPO[37].R=0;
                                        SIU.GDPO[9].R=0;
  }
}
/*********************************************************************
 *              Interrupt Handlers for PIT Channel 1-3       *
 *********************************************************************/
void Pit_Channel_1()
{
PIT.CHANNEL[1].TFLG.R = 1;
SIU.GPDO[36].R = ~SIU.GPDO[36].R;
}

void Pit_Channel_2()
{
PIT.CHANNEL[2].TFLG.R = 1;
}

void Pit_Channel_3()
{
PIT.CHANNEL[3].TFLG.R = 1;
}
```

LAB – 2 SIU.C FILE

```
#include "siu.h"

void Siu_Init() {
/*****************************************************************************
*                    LED Configuration                                       *
*****************************************************************************/
        SIU.PCR[45].R = 0x0200;     /* LED 0 */
        SIU.PCR[46].R = 0x0200;    /*  LED 1 */
        SIU.PCR[47].R = 0x0200;    /*  LED 2 */

  /* Plate configuration LED */
        SIU.PCR[6].R  = 0x0200;
        SIU.PCR[7].R = 0x0200;
        SIU.PCR[8].R = 0x0200;
        SIU.PCR[37].R = 0x0200;
        SIU.PCR[9].R = 0x0200;
        SIU.PCR[36].R = 0x0200;

/*****************************************************************************
*                  Analog inputs configuration                  *
*****************************************************************************/

        /* Analog inputs */
        SIU.PCR[32].R = 0x2500;      /* LDR */
        SIU.PCR[66].R = 0x2500;      /* Potentiometer */

/*****************************************************************************
*                CAN pin configuration                  *
*****************************************************************************/
          /* Setup FlexCAN 1 pins */
          /* TX */
        SIU.PCR[16].B.PA  = 1;
        SIU.PCR[16].B.OBE = 1;
        SIU.PCR[16].B.IBE = 0;

                    /* RX */
        SIU.PCR[17].B.PA = 1;
        SIU.PCR[17].B.OBE = 0;
        SIU.PCR[17].B.IBE = 1;

  SIU.PSMI[0].B.PADSEL = 0x1;
  SIU.PSMI[33].B.PADSEL = 0x1;
}
```

LAB – 2 CAN.C FILE

```
#include "components.h"
#include "can_lld_cfg.h"
#include "me_init.h"
#include "can.h"

// The same configuration can also be done in the main file.
// Please only use Buffers 8-11.
void CANMsgBufInit(void)
{
/*********************************************************************
*           CAN message buffer configuration              *
*                 _____ ___ ___  ___                    *
*                |_   _|/ _ \|   \/ _ \                 *
*                  | | | (_) | |) | (_) |               *
*                  |_|  \___/|___/ \___/                *
*                                    *
*********************************************************************/
/* MB Code */
CAN_0.BUF[8].CS.B.CODE = 8;
/* Standard format */
CAN_0.BUF[8].MSG_CS.B.IDE = 0;
/* SRR */
CAN_0.BUF[8].MSG_CS.B.SRR = 0;
/* Data Frame */
CAN_0.BUF[8].MSG_CS.B.RTR = 0;
/* Data Length */
CAN_0.BUF[8].CS.B.LENGTH = 1;
/* STD_ID */
CAN_0.BUF[8].MSG_ID.B.STD_ID = 0x105;
*********************************************************************

/* MB Code */
CAN_0.BUF[9].CS.B.CODE = 8;
/* Standard format */
CAN_0.BUF[9].MSG_CS.B.IDE = 0;
/* SRR */
CAN_0.BUF[9].MSG_CS.B.SRR = 0;
/* Data Frame */
CAN_0.BUF[9].MSG_CS.B.RTR = 0;
/* Data Length */
CAN_0.BUF[9].CS.B.LENGTH = 2;
/* STD_ID */
CAN_0.BUF[9].MSG_ID.B.STD_ID = 0x301;
```

```
************************************************************************
/* MB Code */
CAN_0.BUF[10].CS.B.CODE = 8;
/* Standard format */
CAN_0.BUF[10].MSG_CS.B.IDE = 0;
/* SRR */
CAN_0.BUF[10].MSG_CS.B.SRR = 0;
/* Data Frame */
CAN_0.BUF[10].MSG_CS.B.RTR = 0;
/* Data Length */
CAN_0.BUF[10].CS.B.LENGTH = 2;
/* STD_ID */
CAN_0.BUF[10].MSG_ID.B.STD_ID = 0x301;


//CAN_0.RXFIFO.IDTABLE[0].R = 0; //0x08000000
//CAN_0.RXIMR[0].R = 0; //0x1fcfffff
}

/************************************************************************
 *               Don't touch anything below!                *
 ************************************************************************/

void cfg0_errorcb(CANDriver *canp, uint32_t esr,uint8_t rx_err_counter, uint8_t tx_err_counter){
  /* Put error management code Here */
  (void)canp;
  (void)esr;
  (void)rx_err_counter;
  (void)tx_err_counter;
}
```

LAB – 2 IRQ_CFG.H FILE

```
#ifndef _IRQ_CFG_H_
#define _IRQ_CFG_H_

#if !defined(FALSE) || defined(__DOXYGEN__)
#define FALSE                   0U
#endif

#if !defined(TRUE) || defined(__DOXYGEN__)
#define TRUE                    1U
#endif
/**********************************************************************
 *                  TASK 1                      *
 *                                          *
 **********************************************************************/
// #define <vectorxx>    <function_Name>
#define vector60 Pit_Channel_1

#define vector61 Pit_Channel_2

#define vector127 Pit_Channel_3

#if !defined(_FROM_ASM_)
#ifdef __cplusplus
extern "C" {
#endif
/*****************************************************************************
 *                              TASK 1                          *
 *            Define the PIT interrupt processing function prototype          *
 *****************************************************************************/
// void <function_Name>(void);   this function is now an external function and can defined in main
void Pit_Channel_1(void);

void Pit_Channel_2(void);

void Pit_Channel_3(void);

  void irq_cfg_init(void);

#ifdef __cplusplus
}
#endif
#endif /* !defined(_FROM_ASM_) */
#endif /* _IRQ_CFG_H_ */
/** @} */
```

LAB – 2 MAIN.C FILE

```
#include "me_init.h"

/*********************************************************************
 *          Global variables and function declarations          *
 *********************************************************************/
void updateInputs(void);
void checkSW1(void);
void sendAlive(void);
void sendFuel(void);
void sendSpeed(void);
void calculateSpeed(void);

int fuel_level;
int rpm;
int acc = 1;
int s_value = 0;

/* Switches and buttons variable to be used to receive signals from board */
int SW1 = 0;
int SW2 = 0;
int SW3 = 0;
int SW4= 0;
int BT5= 0;
int BT6= 0;

/*********************************************************************
 *                  Application entry point                *
 *********************************************************************/
int main(void) {

  /* Board and modules initialization */
  ME_Init();
  PIT_ConfigureTimer(1,200);
  PIT_ConfigureTimer(2,200);

  /* Application main loop that runs forever*/
for ( ; ; ) {
/*********************************************************************
 *                    Main Loop                        *
 *                  _____  ___  ___   ___                  *
 *                 |_   _|/ _ \|   \ / _ \                 *
 *                   | | | (_) | |) | (_) |                *
 *                   |_|  \___/|___/ \___/                 *
 *                                        *
```

```
 *              Write down your logic here.                 *
 ****************************************************************/
        fuel_level = POT;
  /* Operating System Delay*/
  osalThreadDelayMilliseconds(250UL);
 }
}
/*****************************************************************
 *              Can Reception Function                  *
 ****************************************************************/
// The below function is an interrupt function that is invoked every time a CAN message is received
// Since the CAN controller is configured to have a hardware queue for message reception,
// Buffers 0-7 are used for to implement this. Buffer 0's memory is used for reading from the queue
// The interrupt flag of Buffer 5 is used to check if there is message in the queue
// When the id from buffer 0 is read, the data then is lost and next message is pushed to its memory
// The below function is implemented to receive only one message. This message should always be
received
// to be able to receive button updates. Please do not delete it
// When reading multiple messages, make sure to use either a switch case or to store the data and ID
// before comparing them
void can_receive() {
        if(CAN_0.IFRL.B.BUF5I == 1)
        {
                Switch(CAN_0.BUF[0].ID.B.STD_ID) {
                case 0x88:
                {
                        updateInputs();
                        checkSW1();
                } break;

                case 0xFF:
                        U3 = ~U3;
                        break;

                case 0x01:
                        PIT_StartTimer(2);
                        break;
                }

        }
}
```

```
// The following function checks for SW1 switch status
// The switch status is received over CAN Bus from the display's CAN node
void checkSW1(){
	if(SW1 == 1){
		PIT_StartTimer(1);
	}
	else{
		PIT_StopTimer(1);
	}
}
void calculateSpeed(){
	if(acc == 1)
	{
		s_value += 5;
		if(s_value == 300)
		acc = 0;
	}
	else
	{
		s_value -= 5;
		if(s_value == 0)
		acc = 1;
	}
}
// The following function is an example for sending sensor data to the Virtual cockpit for fuel level indication
// Each data byte contains bits of the 10 bit ADC data register
// In this example potentiometer is used as a sensor for data
// At the receiving end, the virtual cockpit combines this split 10 bit data in the same order to be able to read
void sendFuel()
{
	CAN_0.BUF[9].DATA.B[0] = fuel_level & 0xFF;
	CAN_0.BUF[9].DATA.B[1] = (fuel_level>>8) & 0x03;
	CAN_0.BUF[9].CS.B.CODE = 12;
}
void sendAlive(){
	CAN_0.BUF[8].CS.B.CODE = 12;
}
void sendSpeed()
{
	CAN_0.BUF[10].DATA.B[0] = s_value & 0xFF;
	CAN_0.BUF[10].DATA.B[1] = (s_value>>8) & 0x01;
	CAN_0.BUF[10].CS.B.CODE = 12;
```

```
}
/*********************************************************************
 *              Interrupt Functions                  *
 *                 _____  ___  ___   ___             *
 *                |_   _|/ _ \|   \ / _ \            *
 *                  | | | (_) | |) | (_) |           *
 *                  |_|  \___/|___/ \___/            *
 *                                    *
 *          Interrupts can be handled below.          *
 *********************************************************************
 *             Interrupt Handlers for PIT             *
 *********************************************************************/
// This is an example interrupt function that sends alive messages periodically
void Pit_Channel_1()
{
sendAlive();
Tx = ~Tx;
PIT.CHANNEL[1].TFLG.R = 1;
}
void Pit_Channel_2()
{
        U2 = 1;
        sendSpeed();
        PIT.CHANNEL[2].TFLG.R = 1;
        }
/*********************************************************************
*             Interrupt Handlers for CAN Message Buffer      *
*********************************************************************/
// This is the interrupt handler function which is already configured
// It handles the flags generated upon successful transmission
// If any buffer apart from 8 is being used, the respective flag must be cleared
IRQ_HANDLER(SPC5_FLEXCAN0_BUF_08_11_HANDLER) {
        CAN_0.IFRL.B.BUF8I  = 1;
        CAN_0.IFRL.B.BUF9I  = 1;
        CAN_0.IFRL.B.BUF10I  = 1;
}
*********************************************************************
*             Function to receive the data of the display         *
*********************************************************************/
// This function updates the values switches and buttons that are present on the display
// An example here can also be seen for reading and storing CAN message data bytes
void updateInputs()
{
        SW1 = CAN_0.BUF[0].DATA.B[0];
```

```
        SW2 = CAN_0.BUF[0].DATA.B[1];
        SW3 = CAN_0.BUF[0].DATA.B[2];
        SW4 = CAN_0.BUF[0].DATA.B[3];
        BT5 = CAN_0.BUF[0].DATA.B[4];
        BT6 = CAN_0.BUF[0].DATA.B[5];

}

//Task4

// use values form update input function and generate messages to send. Another ECU will receive this message
//buffer 11 is already configured in can.c. Can use buffer 12 as well, but have to configure it
void sendMessageTask4() {
        CAN_0.BUF[11].DATA.B[1] = BT5;//use buffer 11 to send message
        CAN_0.BUF[11].DATA.B[2] = BT6;

        if (BT5 == 1 && BT6 == 1) {
                CAN_0.BUF[11].DATA.B[3] = 1;//if both indicators are on then hazard light is on
        }

        CAN_0.BUF[11].DATA.B[4] = SW1;
        CAN_0.BUF[11].DATA.B[5] = SW3;
        CAN_0.BUF[11].DATA.B[6] = SW4;

        CAN_0.BUF[11].CS.B.CODE = 12;//enable buffer to send
}
```

```
//Additional task (2.1)
void sendMessageAddTask21() {
        int LIBit0;                                     //left indicator
        int RIBit1;                                     //Right indicator
        int HLBit2;                                     //Hazard light
        int HBBit3;                                     //High beam
        int LBBit4;                                     //low beam
        int PLBit5;                                     //Parking light

        //set bit values based on condition
        LIBit0 = (SW2 == 1) ? 0 : 1;                    //if SW2 - ON, then left indicator is on. Otherwise off
        RIBit1 = (SW3 == 1) ? 0 : 1;                    //if SW3 - ON, then right indicator is on. Otherwise off
        HLBit2 = (SW2 == 1 && SW3 == 1) ? 0 : 1;        //if both switches are on then hazard light is on
        HBBit3 = (SW4 == 1) ? 0 : 1;                    //if SW4 - ON, then high beam is on
        LBBit4 = (SW4 == 0) ? 0 : 1;                    //if SW4 - OFF, then low beam is on
        PLBit5 = (speedInt < 20) ? 0 : 1;               //if speed<20, parking light is on. Otherwise off

        //Pack all bits into one
        int messageToSend = LIBit0 | RIBit1*2 | HLBit2*4 | HBBit3*8 | LBBit4*16 | PLBit5*32;

        CAN_0.BUF[12].DATA.B[0] = messageToSend & 0xFF;
        //0 means 8 bit first of mesage 1 byte
        CAN_0.BUF[12].CS.B.CODE = 12;//enable buffer to send
}
```

LAB – 2 CAN.H FILE

```
#ifndef CAN_H_
#define CAN_H_
#include "lldconf.h"

/*****************************************************************************
*                        Task 3                          *
*                    Masking Configuration                    *
*****************************************************************************/
#define MASK_REGISTER 0xF1U // 1111 0001
#define ACCEPTANCE_REGISTER 0x81U // 1000 XXX1 (1000 0001)
#endif /* CAN_H_ */
/*
0x81 1000 0001
0x83 1000 0011
0x85 1000 0101
0x87 1000 0111
0x89 1000 1001
0x8B 1000 1011
0x8D 1000 1101
0x8F 1000 1111

mask 1111 0001
Acc. 1000 XXX1

incoming 1000 1001
mask    1111 0001
       1000 0001

mask    1111 0001
filter   1000 xxx1
       1000 0001
*/
```

LAB – 4

```
void LiAdpAut_Adaptor()
{
	// These functions can be used to access the data elements
	boolean parameter0 = Rte_IRead_Adaptor_RP_SignalAutoIn_Boolean();
	boolean parameter1 = Rte_IRead_Adaptor_RP_SignalLBPIn_Boolean();
	uint8 parameter2 = Rte_IRead_Adaptor_RP_SignalLVIn_Uint();
	//Rte_IWrite_Adaptor_PP_ActSignalLB_Boolean(boolean parameter3);
	//Rte_IWrite_Adaptor_PP_ActSignalRL_Boolean(boolean parameter4);

	if(parameter0 == 1){

		if(parameter2 <= 100){

		Rte_IWrite_Adaptor_PP_ActSignalLB_Boolean(1);
		Rte_IWrite_Adaptor_PP_ActSignalRL_Boolean(0);
			}
		else if(parameter2 > 100){

		Rte_IWrite_Adaptor_PP_ActSignalLB_Boolean(0);
		Rte_IWrite_Adaptor_PP_ActSignalRL_Boolean(1);
			}
		}
	else if(parameter1 == 1)
	{
		Rte_IWrite_Adaptor_PP_ActSignalLB_Boolean(1);
		Rte_IWrite_Adaptor_PP_ActSignalRL_Boolean(0);
	}

	else {

		Rte_IWrite_Adaptor_PP_ActSignalLB_Boolean(0);
		Rte_IWrite_Adaptor_PP_ActSignalRL_Boolean(0);
	}


}
```

LAB – 5

// manual test

Start

```
#define seatbelt_plugged 3
#define seat_occupied 2
#define current_speed 1
#define play_sound 4
#define led_on 9

Case 1:

   Dialog("seatbelt not plugged, seat not occupied, slower than 15, no led, no sound")

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 0 && seatOccupied == 0 && currentSpeed < 15 && playSound == 0 && ledOn ==
0)
   then
      SUCCESS
   else
      FAIL

Case 2:

   Dialog("seatbelt not plugged, seat not occupied, faster than 15, no led, no sound")

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 0 && seatOccupied == 0 && currentSpeed >= 15 && playSound == 0 && ledOn
== 0)
   then
```

```
    SUCCESS
  else
    FAIL

Case 3:

  Dialog("seatbelt plugged, seat not occupied, slower than 15, no led, no sound")

  seatbeltPlugged = can.read(seatbelt_plugged)
  seatOccupied = can.read(seat_occupied)
  currentSpeed = can.read(current_speed)
  playSound = can.read(play_sound)
  ledOn = can.read(led_on)

  if (seatbeltPlugged == 1 && seatOccupied == 0 && currentSpeed < 15 && playSound == 0 && ledOn ==
0)
  then
    SUCCESS
  else
    FAIL

Case 4:

  Dialog("seatbelt plugged, seat not occupied, faster than 15, no led, no sound")

  seatbeltPlugged = can.read(seatbelt_plugged)
  seatOccupied = can.read(seat_occupied)
  currentSpeed = can.read(current_speed)
  playSound = can.read(play_sound)
  ledOn = can.read(led_on)

  if (seatbeltPlugged == 1 && seatOccupied == 0 && currentSpeed >= 15 && playSound == 0 && ledOn
== 0)
  then
    SUCCESS
  else
    FAIL

Case 5:

  Dialog("seatbelt not plugged, seat occupied, slower than 15, led on, no sound")

  seatbeltPlugged = can.read(seatbelt_plugged)
  seatOccupied = can.read(seat_occupied)
```

```
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 0 && seatOccupied == 1 && currentSpeed < 15 && playSound == 0 && ledOn ==
1)
   then
      SUCCESS
   else
      FAIL
```

Case 6:

```
   Dialog("seatbelt not plugged, seat occupied, faster than 15, led on, sound on")

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 0 && seatOccupied == 1 && currentSpeed >= 15 && playSound == 1 && ledOn
== 1)
   then
      SUCCESS
   else
      FAIL
```

Case 7:

```
   Dialog("seatbelt plugged, seat occupied, slower than 15, no led, no sound")

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 1 && seatOccupied == 1 && currentSpeed < 15 && playSound == 0 && ledOn ==
0)
   then
      SUCCESS
   else
      FAIL
```

Case 8:

```
   Dialog("seatbelt plugged, seat occupied, faster than 15, no led, no sound")

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 1 && seatOccupied == 1 && currentSpeed >= 15 && playSound == 0 && ledOn
== 0)
   then
      SUCCESS
   else
      FAIL
Stop



// automatic test

Start

#define seatbelt_plugged 3
#define seat_occupied 2
#define current_speed 1
#define play_sound 4
#define led_on 9

wait(250)

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 0 && seatOccupied == 0 && currentSpeed < 15 && playSound == 0 && ledOn ==
0)
   then
      SUCCESS
   else
```

```
      FAIL

wait(500)

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 0 && seatOccupied == 0 && currentSpeed >= 15 && playSound == 0 && ledOn
== 0)
   then
      SUCCESS
   else
      FAIL

wait(500)

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 1 && seatOccupied == 0 && currentSpeed < 15 && playSound == 0 && ledOn ==
0)
   then
      SUCCESS
   else
      FAIL

wait(500)

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 1 && seatOccupied == 0 && currentSpeed >= 15 && playSound == 0 && ledOn
== 0)
   then
      SUCCESS
```

```
   else
      FAIL

wait(500)

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 0 && seatOccupied == 1 && currentSpeed < 15 && playSound == 0 && ledOn ==
1)
   then
      SUCCESS
   else
      FAIL

wait(500)

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 0 && seatOccupied == 1 && currentSpeed >= 15 && playSound == 1 && ledOn
== 1)
   then
      SUCCESS
   else
      FAIL

wait(500)

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 1 && seatOccupied == 1 && currentSpeed < 15 && playSound == 0 && ledOn ==
0)
   then
```

```
      SUCCESS
   else
      FAIL

wait(500)

   seatbeltPlugged = can.read(seatbelt_plugged)
   seatOccupied = can.read(seat_occupied)
   currentSpeed = can.read(current_speed)
   playSound = can.read(play_sound)
   ledOn = can.read(led_on)

   if (seatbeltPlugged == 1 && seatOccupied == 1 && currentSpeed >= 15 && playSound == 0 && ledOn
== 0)
   then
      SUCCESS
   else
      FAIL


wait(250)

Stop
```