Automotive Technology

VE/TI Lab

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Experiment 3

Name	NADZIRAH BINTI BACHO
Group Number	
Date	

3. CAN-LIN Gateway

In this experiment, we want to analyze the activation of two connected bus systems by a control unit used for development purposes, once more issued by Vector itself.

In the experiment, the hazard light function in the instrument cluster, which is already known from the previous experiment, should be realized via the CAN bus. This function should be controlled by a LIN keypad.

The experiment is divided into two parts. In the first part, the LIN keypad will be put into operation, in the second part, the same will be done for the instrument cluster.

Sending and Receiving LIN messages with a development ECU

3.1. Introduction

- Revise the chapter dealing with the LIN bus in your lecture documents.

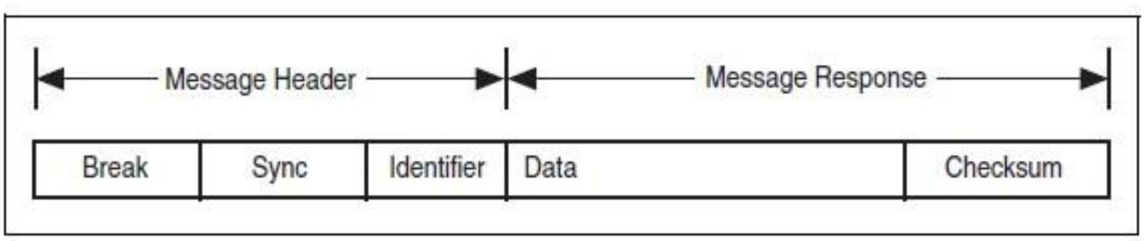
  a) How is a LIN message set up?

= The LIN bus message frame consists of a header and a response. Typically, the LIN master transmits a header to the LIN bus. This triggers a slave, which sends up to 8 data bytes in response.

  b) What does start bit and stop bit mean and what is their polarity?

= The start and stop bits are used in asynchronous communication as a means of timing or synchronizing the data characters being transmitted. Without the use of these bits, the sending and receiving systems will not know where one character ends and another begins.
Another bit used to separate data characters during transmission is the mark (or idle) RS bit. This bit, a binary 1, is transmitted when the communication line is idle and no characters are being sent or received.
When a start bit (binary 0) is received by the system, it is understood that a fixed number character bit (determined by the bits per character parameter), and even a parity bit (determined by the parity parameter), follows the start bit. Then a stop bit (binary 1) is received by the system. In the following example, the parity bit is present, and the bit per character is 7.

c) What are the roles of the master and the slave in the LIN network?



= The LIN bus message frame consists of a header and a response. Typically, the LIN master transmits a header to the LIN bus. This triggers a slave, which sends up to 8 data bytes in response.

d) What are the signals' levels that are defined in the LIN bus?

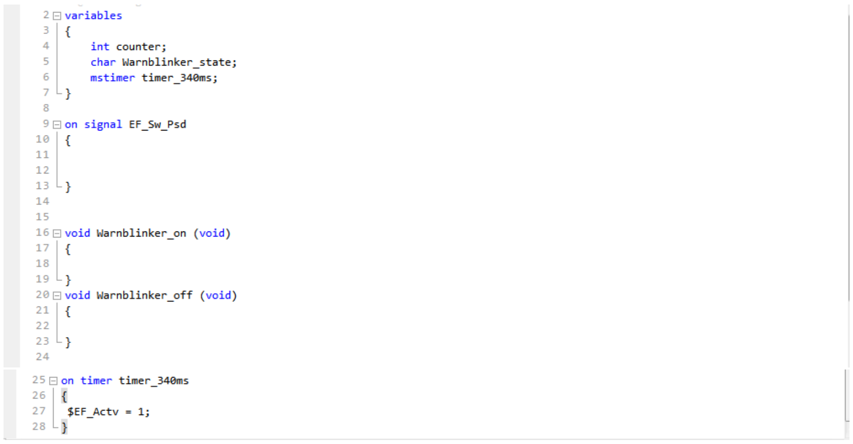
= In looking at LIN Bus signals on a scope, the lower level voltage (logic zero) should be less than 20% of battery voltage (typically 1 V) and the upper level voltage (logic one) should be more than 80% of battery voltage

e) What are the electrical connections needed for the LIN bus?

= A standard LIN bus consists of a master node and up to 15 slave nodes connected to a single network. The physical LIN network is a three-wire configuration consisting of power (vehicle battery), ground and the LIN bus communication line. A pull-up resistor, RLIN, typically 1kΩ, is required on the master's LIN bus line.

3.2. Introduction to CAPL

The behaviour of simulated network nodes (LIN, CAN) in CANoe on the bus can be programmed directly in CANoe by the programming language CAPL (Communication Access Programming Language) that is similar to C. Follow the Video "Experiment 3 – Introduction to CAPL" by composing your own short examples of the functions explained. Use the help function (index search) of CANoe for the following functions:



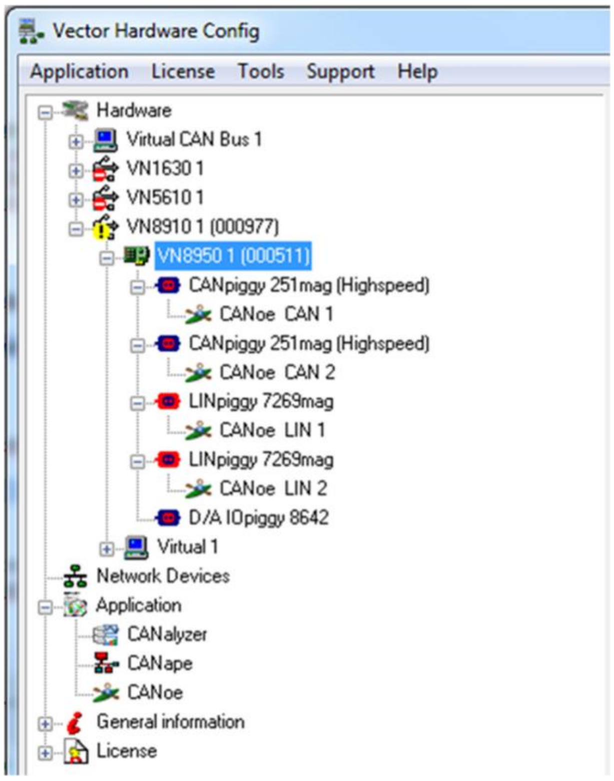
```
variables
{
    int counter;
    char Warnblinker_state;
    mstimer timer_340ms;
}

on signal EF_Sw_Psd
{


}


void Warnblinker_on (void)
{

}
void Warnblinker_off (void)
{

}

on timer timer_340ms
{
 $EF_Actv = 1;
}
```

- on start
- on signal
- on timer
  (Creating a timer, initializing a timer, starting a timer, stopping a timer)
- What are the different timers that can be used?
- How can you get access to signals contained in databases by using CAPL?
- How can you get access to system variables by using CAPL?

3.3. Vector driver configuration (channel settings Hardware/software)

For the hardware setup, watch the video “Experiment 3 – Hardware setup of LIN keypad”. As far as configuration in CANoe is concerned, you can skip it because you do not have the hardware necessary. If ever you want to set it up, you can do it by opening the driver configuration of the Vector hardware in CANoe clicking on *Hardware -> Bus hardware -> drivers…*

Now you can define the channels by right-hand clicking according to the following figure 3.1.



```
Hardware
    Virtual CAN Bus 1
    VN1630 1
    VN5610 1
    VN8910 1 (000977)
        VN8950 1 (000511)
            CANpiggy 251mag (Highspeed)
                CANoe  CAN 1
            CANpiggy 251mag (Highspeed)
                CANoe  CAN 2
            LINpiggy 7269mag
                CANoe  LIN 1
            LINpiggy 7269mag
                CANoe  LIN 2
            D/A IOpiggy 8642
        Virtual 1
Network Devices
Application
    CANalyzer
    CANape
    CANoe
General information
License
```

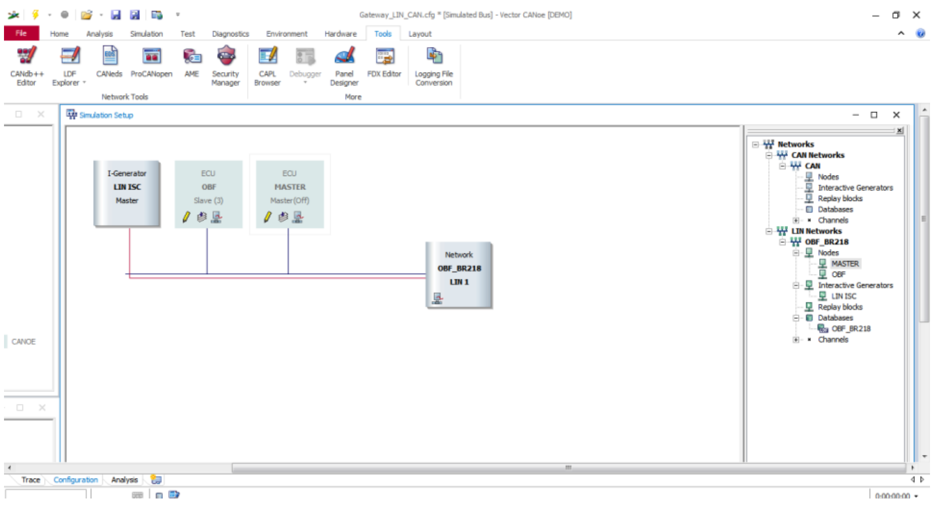
Figure 1: Vector hardware configuration

3.4. The LIN node

The LIN keypad is a component taken out of a Mercedes Benz CLS, it is manufactured by the company Marquardt. An extract of the documentation is available in the file *Kurzbeschreibung OBF.pdf*.

3.5. Visualizing LIN signals

- Copy the lab files necessary on your desktop.
- For hardware configuration, please refer to the video.
- Open the configuration file *Gateway_LIN_CAN.cfg* in CANoe.
- Open the simulation setup window by clicking on *View -> simulation setup* and add the database *OBF_BR218.ldf* to the LIN network. After this, deactivate the nodes *LIN master* and *OBF* by right-hand clicking on the nodes.



- Open the panel designer by clicking on *Tools -> Panel Designer* and create a panel that visualizes the pushing of the button on the real hazard light button.
  Find the signal *EF_SW_Psd* (edge recognition of the hazard lights button) in the database and connect it to a corresponding element.
- Save the panel as *"Tasterleiste.xvp"* in your repository.

3.6. Connecting the hazard lights function/hazard lights button

To make sure that the hazard lights button will shine when pushing it, the signal *EF_Actv* must have the value 1.

- Reopen the panel designer and add a button to the panel. Label it “*Warnblinkleuchte EIN*” in the panel.
- Assign the signal *EF_Actv* (Activation of the hazard light button’ LED) of the database to this button.

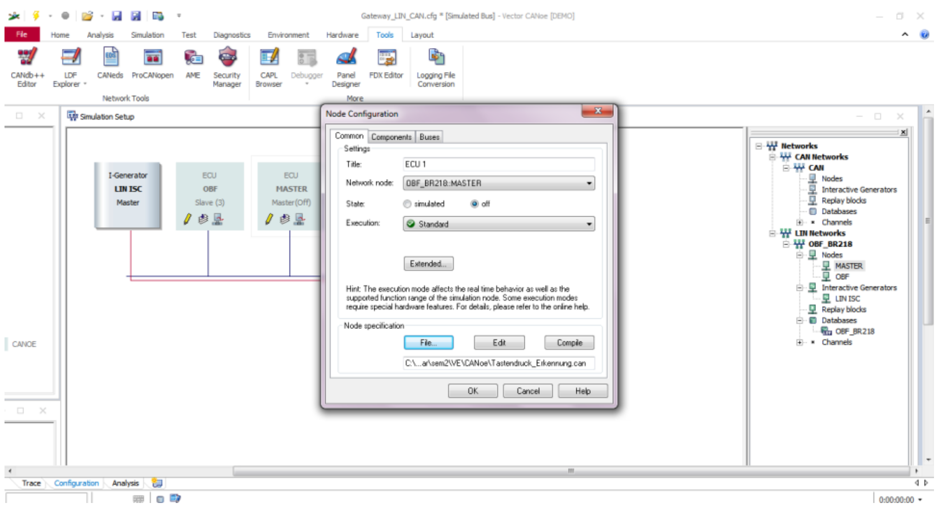
Now, when the button in the panel is pushed, the real hazard light button should flash. The software of the LIN keypad will reset the light individually after some milliseconds. As you do not have the real hardware, just put your files in the online repository. We will test them for you later.

3.6.1. Programming the hazard light button

In the following step, the buttons' blinking behaviour will be realized when pushing the button on the real keypad.

- By means of a flowchart, show how you would realize the connection between the button and its lighting. Take into consideration, when creating the logic, that the signal *EF_SW_Psd* jumps up on value 1 (rising and falling edge) for a short moment whenever you push or release the button!

- In the simulation setup window, add the CAPL code to the master node by right-hand clicking on the *node* and then -> *configuration*, in the tab *common* -> *node specification* -> *file* and then adding the *Tastendruck.Erkennung.can* that you can find online.



Node Configuration

Common | Components | Buses

Settings

Title: ECU 1

Network node: OBF_BR218::MASTER

State: ○ simulated ● off

Execution: Standard

Extended...

Hint: The execution mode affects the real time behavior as well as the supported function range of the simulation node. Some execution modes require special hardware features. For details, please refer to the online help.

Node specification

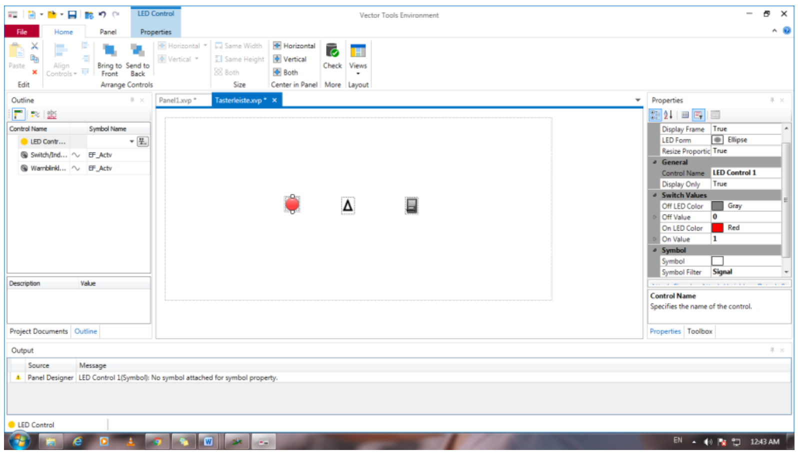
File... | Edit | Compile

C:\...ar\sem2\VE\CANoe\Tastendruck_Erkennung.can

OK | Cancel | Help

- Realize the flashing of the hazard light button when pushing it, by the means of the function framework and your flowchart.

- Extend your already existing panel by a hazard light symbol that flashes when pushing the button. Therefore, use the *switch/indicator* element you can find in the library. By clicking on *Choose image* in the element's properties, you can add the corresponding symbol. You can download the graphic file online.



Control Name	Symbol Name
LED Contr...	
Switch/Ind...	EF_Actv
Warnblinkl...	EF_Actv

Description	Value

Display Frame	True
LED Form	Ellipse
Resize Proportic	True
General	
Control Name	LED Control 1
Display Only	True
Switch Values	
Off LED Color	Gray
Off Value	0
On LED Color	Red
On Value	1
Symbol	
Symbol	
Symbol Filter	Signal

Control Name
Specifies the name of the control.

Output

Source	Message
Panel Designer	LED Control 1(Symbol): No symbol attached for symbol property.

LED Control

- Extend your flowchart by the real blinking function of the hazard light. It should be activated when pushing the button and the hazard light button should blink by a frequency of 2.94 Hz. When pushing the button one more time, the blinking should stop.

- Transfer your extended flowchart into CAPL code and realize the function. Always be careful on using the predefined function framework.

3.7. Finishing your work

The lab folder will be needed one more time in the next experiment so do not delete it!