Lab n° 3
Real-Time systems [ELEC-H-410]
CAN network overview

2017–2018

Purpose

This lab aims at giving a short overview of the CAN network. You will first study the CAN features then write an application sending and receiving messages.

Useful documents are stored on the network share:

`\\labo\ELEC-H-410\Useful Documents\`

- `dsPIC30F-33F Programmer's Reference.pdf`
- `dsPIC33 Data Sheet.pdf section 21`
- `Introduction to MPLAB.pdf`
- `Explorer 16 User Guide 51589a.pdf`
- `MPLAB C30 C Compiler User's guide.pdf`
- `uCOSII_RefMan.pdf`
- `Enhanced Controller Area Network.pdf`
- `Introduction to language C for microcontrollers.pdf`
- `Table of ASCII codes.`
- `Troubleshooting µC/OS-II`

1 CAN and dsPIC33

Plug the CAN network yellow twisted pair in the DB9 connector of the Explorer 16 extension Board.

You will program the CAN peripheral of the dsPIC to send a message, then to receive messages transmitted continuously by a PC of the laboratory.

1.1 Read the manual...

Open the datasheet of the `Enhanced Controller Area Network.pdf`, read:

- CAN module overview : p2–5
- Message transmission: p35 and following
- Message reception: p41 and following

Exercise 1. Describe a CAN bus message.

As you can see on figure 21-1, the dsPIC contains 32 buffers for transmission/reception, which can be configured either in reception, or in transmission. These buffers are not directly in the CAN peripheral.

Exercise 2. Explain which mechanism enables to access to the CAN peripheral and what are the advantages of this mechanism.

The buffers for reception use a particular mechanism to filter the identifiers of the messages.

Exercise 3. Explain this mechanism. What are the advantages and disadvantages?

1.2 . . . and use your knowledge

Open the project `OSCan`.

A library of functions has been developed to facilitate the use of the CAN peripheral (see files `CanDspic.c` and `CanDspic.h`).

Examine the file `CanDspic.h`. Observe the structure that has been set for the CAN buffers. Refer to section 21.4 of the pdf for more details.

Exercise 4. Which configuration was chosen for the CAN buffers?

In the function `main()`, configure the CAN peripheral for a bit rate of 500kbps. An initializing function allows you to configure automatically the first 7 buffers in reception and the 8th buffer in transmission.

Exercise 5. Based on the `IntroCan` project, write a task that continuously send a CAN message on the network (at a rate of 1s). This message must contain an ASCII string of maximum 8 characters of your choice. Use as ID (0x180 + y), with y your group number. To check that your message is correctly sent, one of the nodes of the lab has been programmed to monitor and displays all the messages passing on the bus.

You shall now receive and process messages transmitted on the network. A PC node has been programmed to send permanently 4 messages on the network. Your goal is to find what are these messages and their contents. The reception of the messages should be done by interruption. You will find a skeleton of the interrupt service routine in `CANRxInterrupt.c`.

Exercise 6. Configure the reception of CAN messages and activate the interruptions in `main()`

Exercise 7. Write the interrupt service allowing to find a first message (ID: 153). An ISR should be as short as possible and should only be used for basic data processing and to transfer data to tasks (see previous lab for communication structures). By using a mailbox write a task that will print the content of the message on the LCD screen. You might need to make a copy of the buffer using the following code.

Listing 1: Buffer copy

```
char message[8];
...
Str_Copy_N(message, receiveBuffers[0].DATA, receiveBuffers[0].DLC);
...
```

In the case of interrupt nesting, `OSIntEnter()` and `OSIntExit()` should be used but that's not necessary with the current OS configuration (see reference manual pp. 29 – 31).

Exercise 8. Modify your code to receive the other messages (ID: 154, 155, 156). You can implement a "software only" filter if you want. Once you have received the 4 messages, decode them using the ASCII table and follow the instructions they contain.