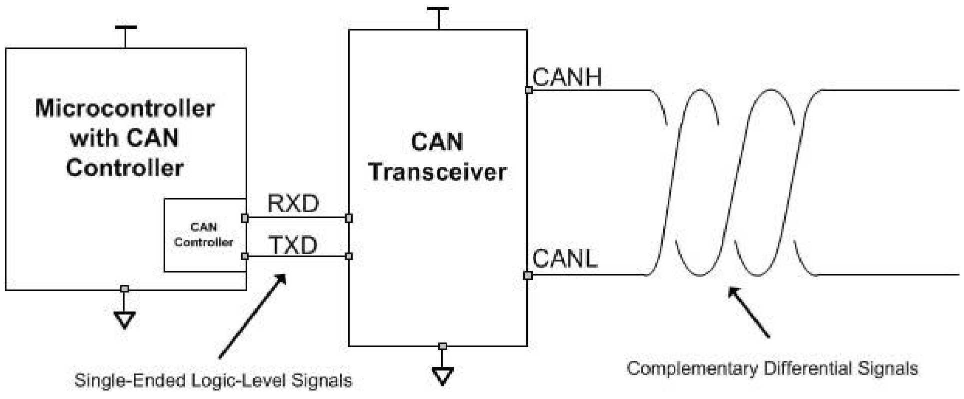
Redes Automotivas 2018.2
Project

Objective

Design and build a CAN ECU that must be able to exchange CAN frames with others ECUs. Your ECU should be composed of: a microcontroller, a CAN controller and a CAN transceiver. The microcontroller must be programmed in C/C++ and can be an arduino or other platform. The CAN controller must be implemented in software using the microcontroller. The CAN transceiver will be provided, such that the ECU can be connected to a CAN bus. The CAN controller must be connected to a CAN bus through the CAN transceiver using the CAN RX and CAN TX pins.


Microcontroller with CAN Controller
CAN Controller
RXD
TXD
CAN Transceiver
CANH
CANL
Single-Ended Logic-Level Signals
Complementary Differential Signals

The controller has two main features: encode and decode CAN frames. The controller designed must be able to send and receive CAN2.0A and CAN2.0B frames. The correct functioning of your controller will be validated using a CAN bus shield. The CAN bus shield will send CAN frames that your controller must decode and your controller must send CAN frames that the shield can decode. To design the CAN encoder and decoder you must comprehend the purpose of each bit and field of a CAN frame, what error conditions can occur, validate the CRC, transmit the ack bit, send an error frame if necessary, and synchronize to the bus. It is essential to spend some time studying the CAN protocol to fully understand it. In a nutshell, your controller must be able to:

1) Send and receive data and remote frames checking the CRC and sending the ACK bit;
2) Identify the occurrence of error/overload frames;
3) Work at a 500 Kbps baud rate according to the following Bit Timing parameters:
    a) fosc = 16MHz
    b) Tq = 2/fosc
    c) SJW = 1Tq
    d) Prop Seg = 1Tq
    e) Phase Seg 1 = 7Tq
    f) Phase Seg 2 = 7Tq

Finally, each team will be designated a few CAN frames corresponding to specific signals that should be transmitted or received and interpreted in a particular way. Then, all

ECUs designed will be connected to the same bus to exchange CAN frames among themselves and operate as they were the ECUs of an actual vehicle.

Platform and programming language
Arduino IDE or other equivalent and Processing IDE.
The programming language must be C/C++. The use of Linux or any other High Level OS is not allowed. Only microcontrollers and RTOS for microcontrollers are allowed.

Teams
8 teams of 2 people
1 team of 3 people

Deadlines and deliverables

Deadline 1: **17/10/2018**
Grade percentage: **10%**
Deliverables 1: Flowcharts, block diagrams and state diagrams for the Bit Timing module needed for the CAN controller design.

Deadline 2: **24/10/2018**
Grade percentage: **15%**
Deliverables 2: Flowcharts, block diagrams and state diagrams for the CAN encoder/decoder modules needed for the CAN controller design.

Deadline 3: **12/11/2018**
Grade percentage: **25%**
Deliverables 3: Implementation for the Bit Timing module.

Deadline 4: **28/11/2018**
Grade percentage: **30%**
Deliverables 3: Implementation for the encoder/decoder modules. The designed ECU must be working and will be tested in this day both individually and connected with the ECUs designed by the other teams.

Deadline 5: **05/12/2018**
Grade percentage: **20%**
Deliverables 3: Powerpoint presentation in class, report, public exhibition and final project zip.