VANIER COLLEGE - Computer Engineering Technology
247-609 Networked Embedded Systems

Lab 6 : CAN Remote Frame

Leonardo Fusser (1946995)

Purpose:

a) To understand the functionality of remote frame in CAN bus.
b) To familiarize and verify the operation of remote frames using BBB and SocketCAN.

To be submitted before the deadline, via MS Teams Assignment:

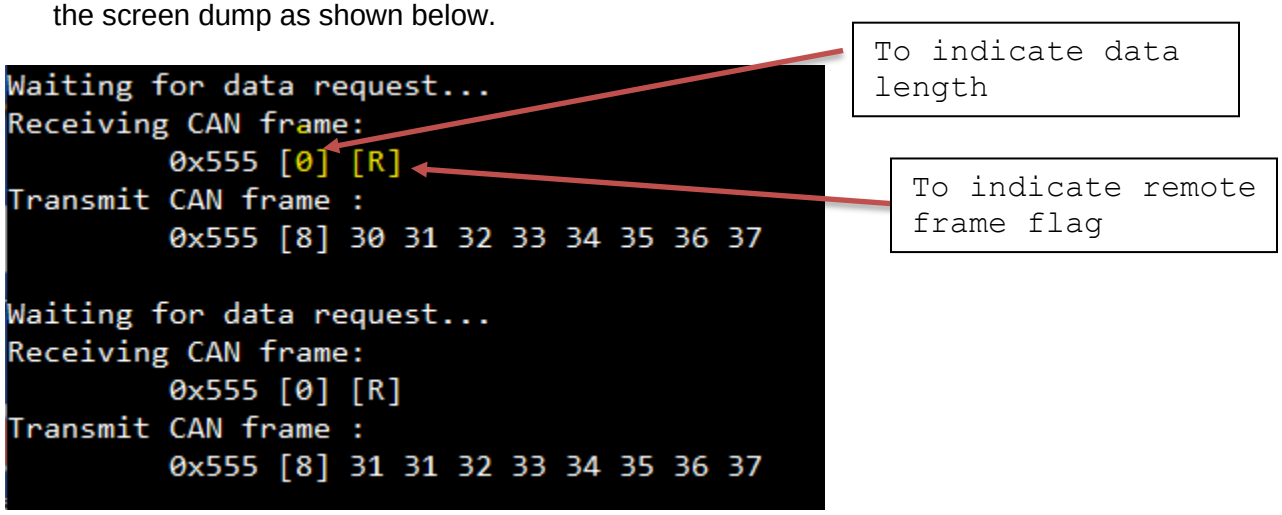
1. **No formal report required**. Report must be in a single MS Word document, and includes the following:
   a. Answer questions, screen shots (clearly labelled, with analysis/explanation).
   b. Include a final discussion and conclusion session.
2. Well commented, final C code.

Lab Work:

Part A : Remote frame operation (pair up in a team of at least 2)

1. Setup your CAN operation as per usual. No loopback!

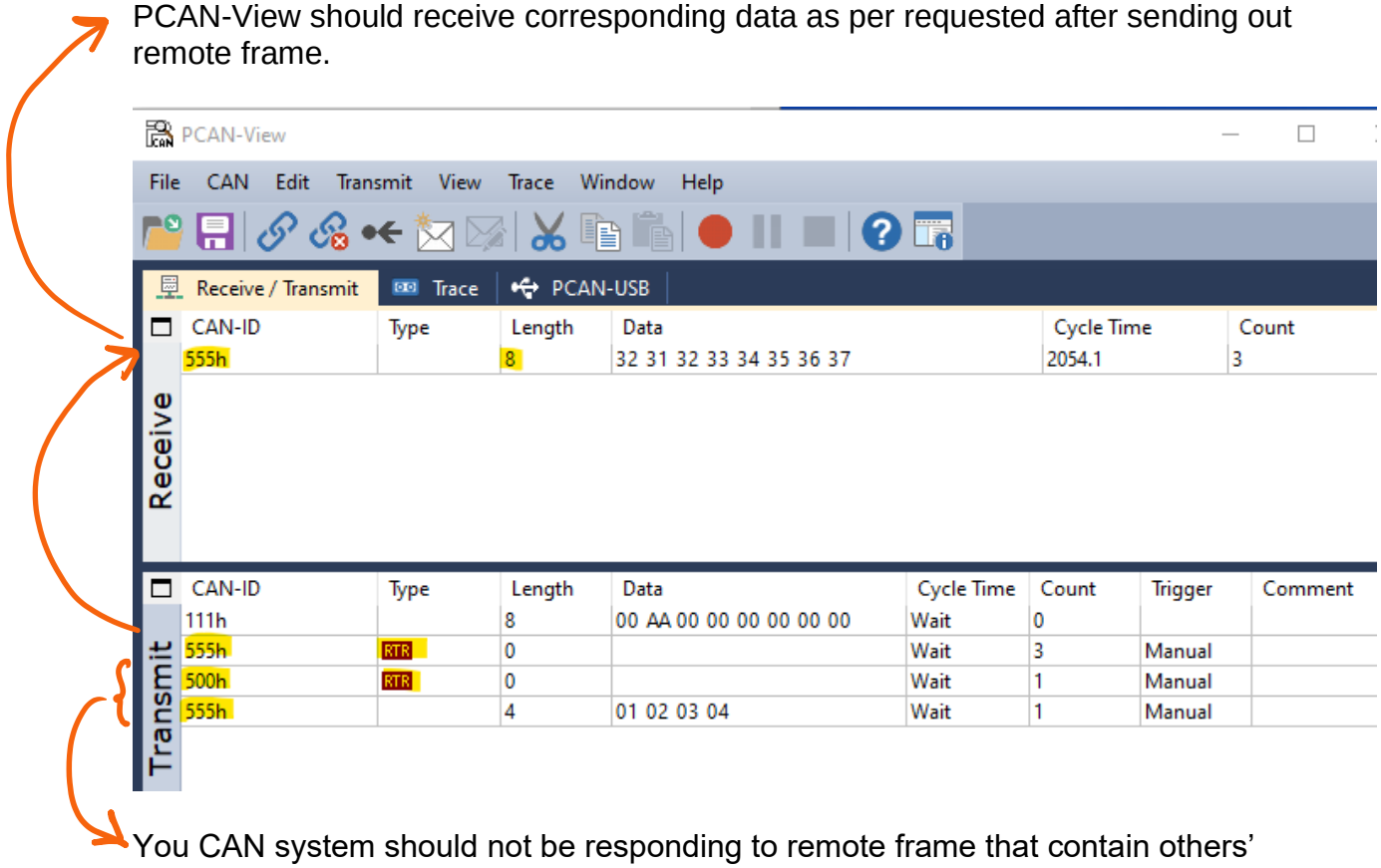
2. Based on your basic code framework from lab 5, modify your code for the following:
   - Same as in previous lab, set up your bench number as your message.
   - Upon receiving the request that match the ID of you message, send out a data message as per requested. The format of the data message should be the same as previous lab.
   - Print out your CAN action/status/received message related information. Example of the screen dump as shown below.



Once your code is ready, connect up the system. You may use PEAK PCAN-USB to generate the remote frames. Test your system to ensure that the system is working as expected, by sending different remote frames and data frames to ensure that your CAN system works as expected.

Example :

PCAN-View should receive corresponding data as per requested after sending out remote frame.



You CAN system should not be responding to remote frame that contain others' message id, or data frame with your id.

3. Screen shot (with explanation and descriptions) from protocol analyzer and tera term are required to illustrate the operation of your system.

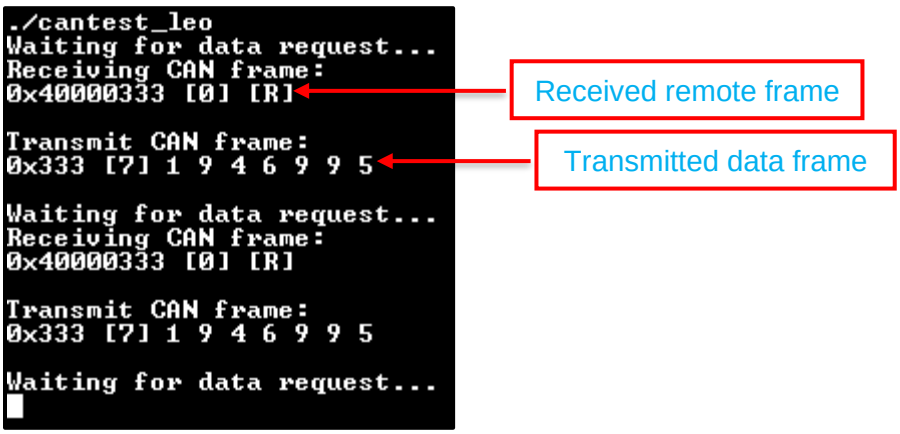


Figure 1. Screenshot above shows the newly modified code responding to remote frames received. The program waits until a specific remote frame has been received before continuing on to transmitting a data frame. The program waits until a remote frame with a CAN ID of 0x333 has been received before continuing on.

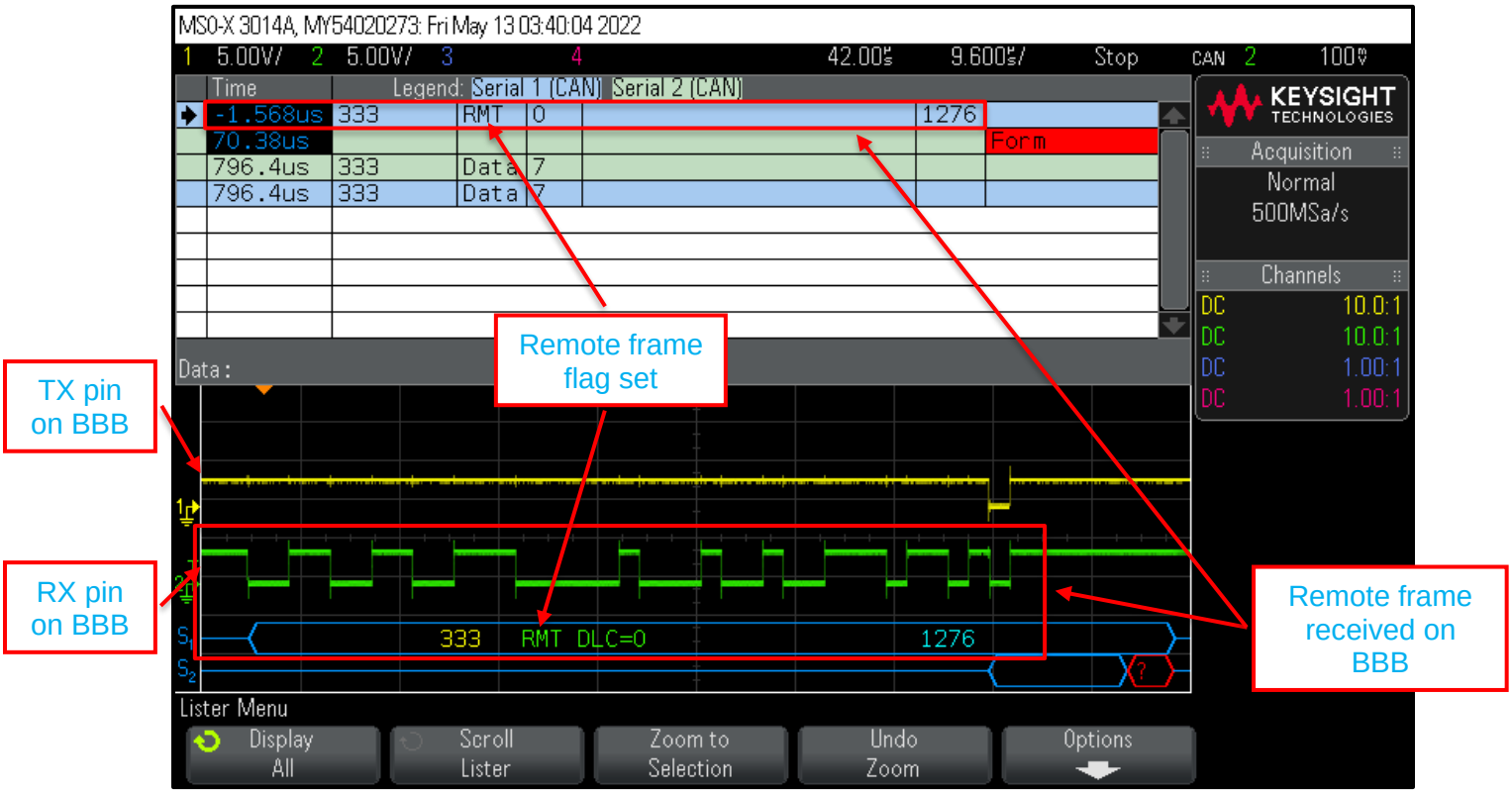


Figure 2. Screenshot shows remote frame received on BBB on the protocol analyzer. Green signal is probed to RX pin and yellow signal is probed to the TX pin on the BBB. The transmitted data frame is shown in the next screenshot on the next page.

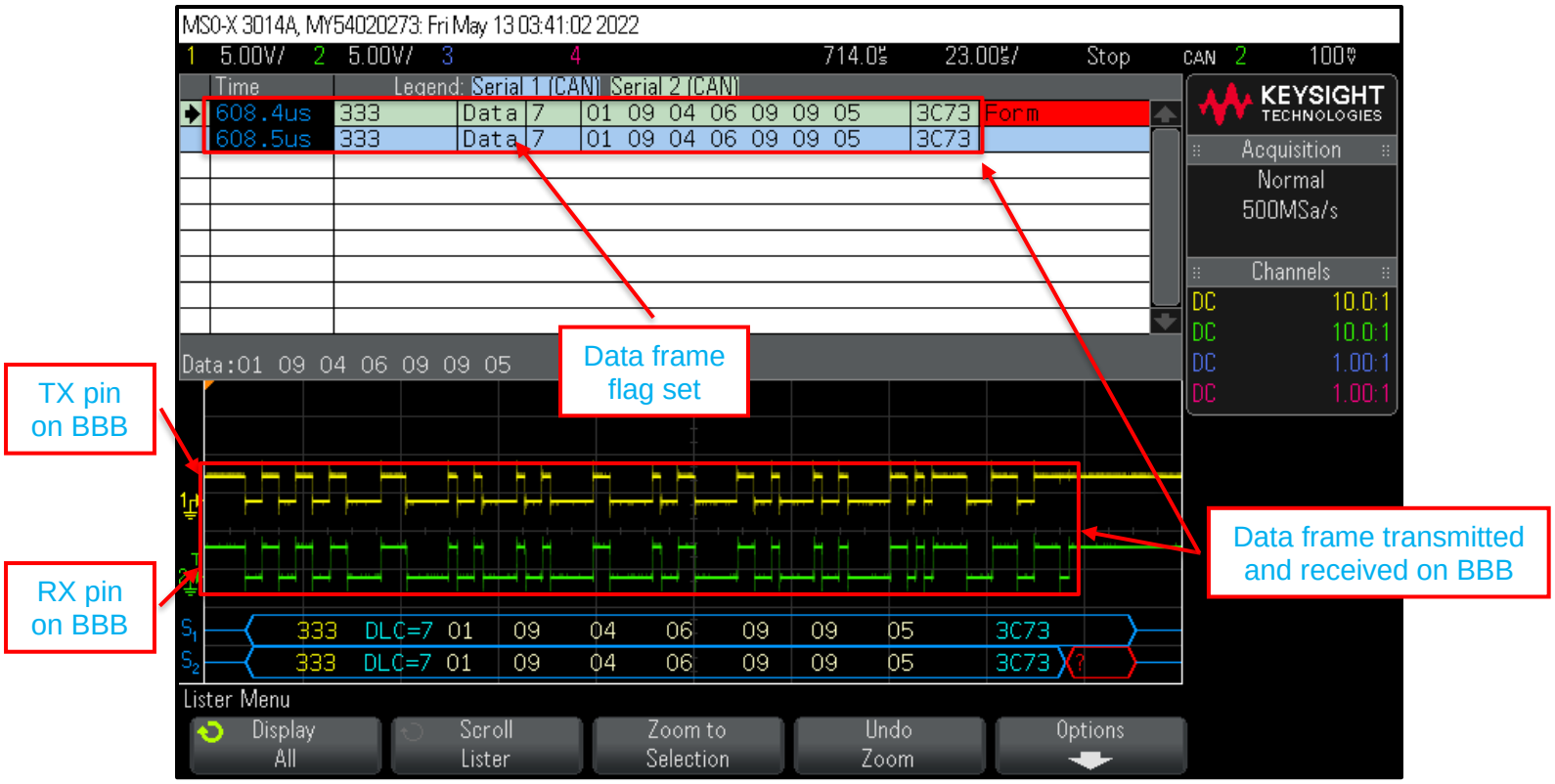


Figure 3. Screenshot shows data frame transmitted and received on BBB on the protocol analyzer. Green signal is probed to RX pin and yellow signal is probed to the TX pin on the BBB. Two identical data frames can be seen since the source is coming from the BBB.

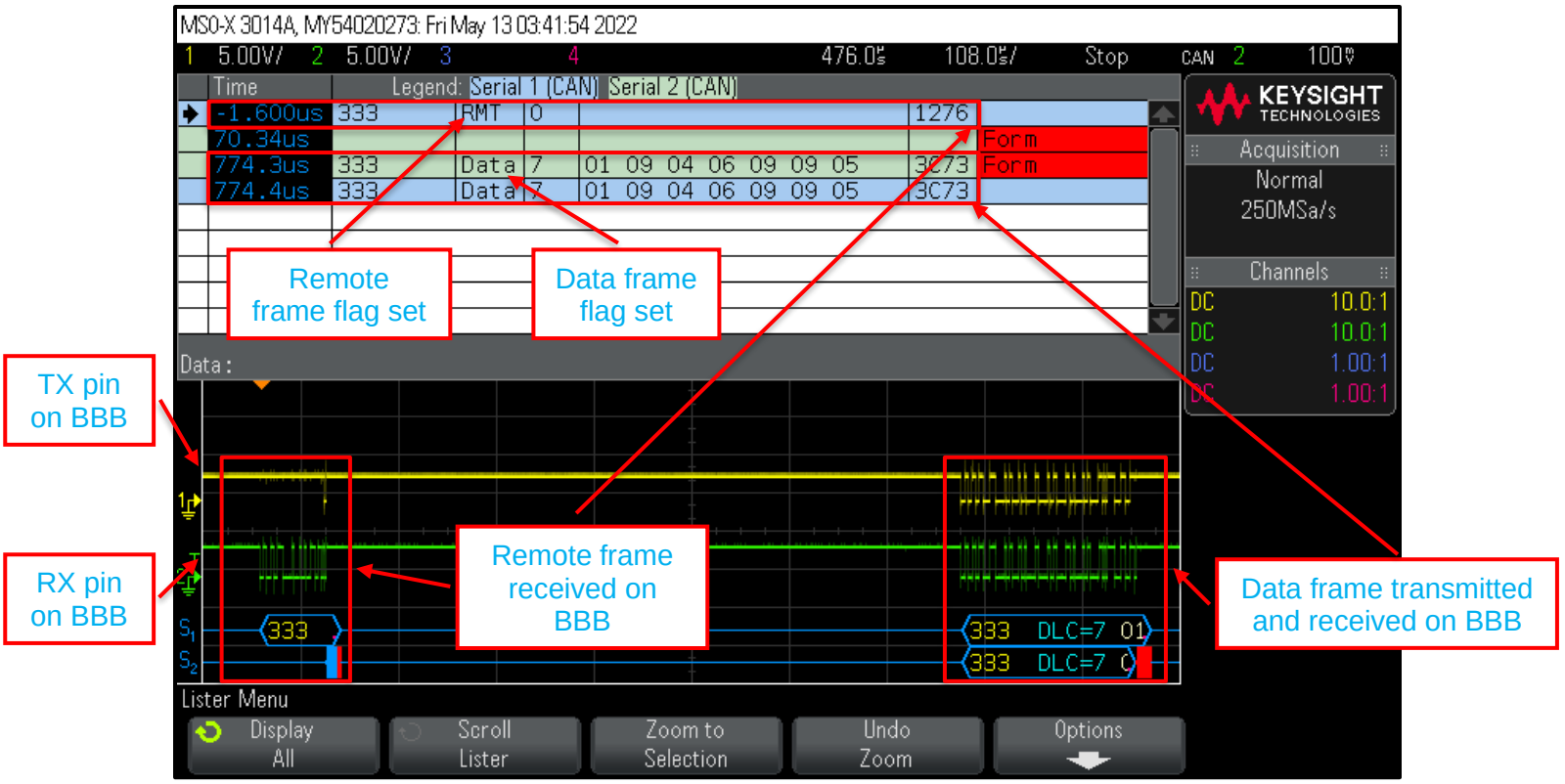


Figure 4. Screenshot shows received remote frame and transmitted and received data frame on BBB on the protocol analyzer. Green signal is probed to RX pin and yellow signal is probed to the TX pin on the BBB. The far left shows the captured remote frame and the far left shows the transmitted and received data frame. Two identical data frames can be seen since the source is coming from the BBB.

Discussion:

- For the first part of the lab, the CAN code in C was modified to handle remote frames. More specifically, the code was modified to let only certain remote frames through, and others being rejected. A filter was used to achieve this (only remote frames with a CAN ID of 0x333 were let through). Furthermore, even regular data frames with a matching CAN ID would be blocked on a regular basis.

  Once the code was modified, screenshots were taken using the protocol analyzer on the oscilloscopes. The received remote frame on the BBB was captured on the protocol analyzer. The lister function was used to facilitate in identifying the captured remote and data frames. The transmitted and received data frame on the BBB was also captured and the lister function was used like before.

  The overall lab was a success.

Conclusion:

- Successfully understood the functionality of remote frames in a CAN bus network.
- Successfully familiarized and verified the operation of remote frames using the BBB and SocketCAN C library.