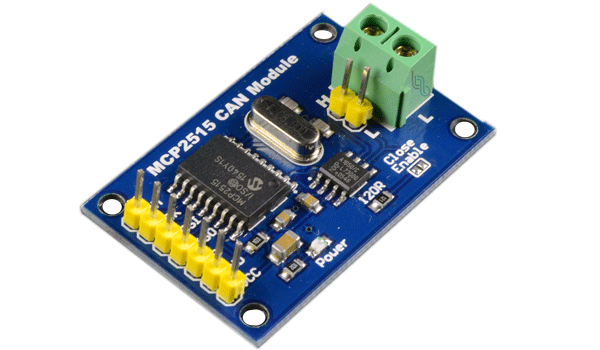
CAN



Semester 3 Communication

CAN Assignment

Dec 07, 2022.

Students: Andre Sanao & Nhat Minh Nguyen

Course: Technology

Table of Contents

[**Acronyms** 3](#_Toc121701112)

[Introduction 4](#_Toc121701113)

[Part 1 5](#_Toc121701114)

[Introduction 5](#_Toc121701115)

[Implementation 5](#_Toc121701116)

[(table 1, sent and measured data compairation) 6](#_Toc121701117)

[Conclusion 6](#_Toc121701118)

[Part 2 7](#_Toc121701119)

[Introduction 7](#_Toc121701120)

[Implementation 7](#_Toc121701121)

[Conclusion 7](#_Toc121701122)

[Part 3 8](#_Toc121701123)

[Introduction 8](#_Toc121701124)

[Implementation 8](#_Toc121701125)

[Conclusion 8](#_Toc121701126)

[Part 4 8](#_Toc121701127)

[Introduction 8](#_Toc121701128)

[Implementation 8](#_Toc121701129)

[Conclusion 8](#_Toc121701130)

**Acronyms**

*Table 1 – List of acronyms used throughout the report*

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| CAN | *à*  *Controller Area Network* |
|  |  |

# Introduction

The assignment on which this document presents is the approach and implementation of which has an end goal of merging the knowledge of the previous assignments from this subject.

Tasks in this assignment requires us to know how communication protocol works between 2 hardware devices. Therefore, we are using an Arduino to connect to a CAN module and put our knowledge that we have received during the lecture. The assignment is split into 4 parts, each regarding one concept of the CAN module.

# Part 1

## Introduction

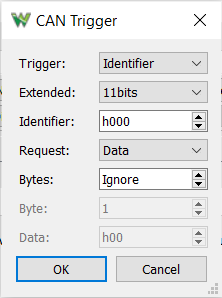
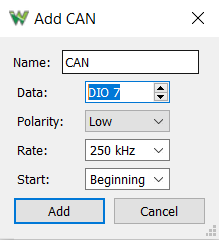
In this part, we are required to use a logic analyzer (Analog Discovery 2) to measure the signal of an integer array send by the CAN device.

## Implementation

For hardware configurations, we connect the CANL pin to pin 7 of Analog Discovery 2. We connect CAN and Analog Discovery 2 to the same ground, this is not entirely sure whether it is necessary, but in our case if they are not connect to a common ground, we received a lot of error.

To use the fuction from <SPI.h> and <mcp2515\_can.h>, we need to install the "CAN\_BUS\_Shield" library from seeed studio. We send an array of 8 integers from 0 to 7 using this fuction: "CAN.sendMsgBuf(0x00, 0, 8, stmp);" in this case we choose our Arbitration ID to be 0, and our control field byte is 8.

In Waveform, we tried out a lot of different values for the parameters. After a lot of fails and some hardware issues, we successfully ended up with a bit rate of 250 kHz the capture of data is triggerd by an 11 bit indentifier (figure 1.1 and 1.2).



(figures 1.1 and 1.2, choices of configurations Waveform)

We captured the signal of each data we sent (figure 2), the next step is to calculate the bit of the Arbitration ID, Control Field, and for every data.



(figure 2, data read by the Analog Discovery 2)

The Sample rate we chosen was 256 ki, and the bit was counted in the order of Arbitration ID, Control Field, and data bytes. At the end, we compare the measured data from the logic analyzed to the actual data that we sent (table 1).

|  |  |  |
| --- | --- | --- |
| Sent data | Sent data in bit | Measured data |
| 0x00 | 00000000000 | 00010001000 + RTR |
| 8 | 001000 | 011000 |
| 0 | 00000000 | 01000010 |
| 1 | 00000001 | 00010001 |
| 2 | 00000010 | 00001010 |
| 3 | 00000011 | 00010011 |
| 4 | 00000100 | 00001100 |
| 5 | 00000101 | 00100101 |
| 6 | 00000110 | 00001110 |
| 7 | 00000111 | 00010111 |

## (table 1, sent and measured data compairation)

## Conclusion

In the end, we were able to read the data from the sender CAN module via Analog Discovery 2, as you can see from table 1, some 1 are added due to bit stuffing but these bits are ignored by the receiver.

# Part 2

## Introduction

In part 2, we must measure the bit time for 1 bit. This can be seen by looking at the previous assignment measurements with the logic analyzer.

## Implementation

For the time to be measured, we must measure 1 bit in the logic analyzer. A bitrate of 500 kbps is used for the CAN module. Both values are expected to be different from each other because the module has a different bit time. In the library of mcp2515\_can, there is an example where we can see the information of the init function. The default clock rate of the init function is 16Mhz, the comment for this function says that for MCP2515, it should be 8Mhz. 8Mhz difference would make the bit time half of the bit time.

Diagram

Description automatically generated with medium confidence

Figure measurement of 1 bit

## Conclusion

This concludes part 3, it is not so difficult to measure the bit time of the one bit and finding the default clock rate. It is a short theory part of the assignment in which how to understand the library, so we don’t make any mistake if using it for bigger projects.

# Part 3

## Introduction

In part 3, we are to test the connection between 2 CAN module. CAN module A will be the sender and CAN module B will be the receiver. We will be sending data to test the connection.

## Implementation

To implement the assignment requirements, there is a link provided as an example on how to send data to the receiver. The example sends an array of bytes to the sender and if the receiver receives a certain data, then it will execute a given task. To test this, we used the serial monitor to send ON and OFF commands to the receiver by putting each character in an array and use the send function provided by the library. In the receiver, it will iterate character by character and if it reads ON then it will turn on the LED and if it is OFF then it will turn off. A video is provided for a visual demonstration.

## Conclusion

To conclude this part of the assignment, we needed to change a small part of the example to be able to achieve the requirements of the assignment. It was easy to implement due to previous knowledge of the serial communication.

# Part 4

## Introduction

For this part of the assignment, we are going to use 2 CAN module and a joystick. The joystick will send data and the CAN module will responds to the movement direction of the joystick.

## Implementation

To implement the assignment requirements, we must know the ID of both CAN modules. We used a function to get the ID of each CAN module. These ID would be used to send tasks to a specific CAN module.

We did a little research on how to use the joystick and implement it to the assignment. The left and right direction of the joystick will send to CAN module A which blinks the LED depending on which side it is. For CAN module B, it will act like a high beam of a car. If the joystick is in up direction, it will turn on the LED and if it is down direction, it will turn the LED off. There is a visual demonstration provided.

## Conclusion

To conclude the final part of the assignment, there was a lot of confusion on how to connect the hardware part of the assignment. After so many ideas on how to tackle the assignment, we decided to just use 2 codes to demonstrate CAN module A and B separately.