EMERTXE TRAINING PROJECT DOCUMENTATION FRAMEWORK

# REQUIREMENTS & DESIGN DOCUMENT

## Module – Microcontroller

# CAN Node

# Contents

# 1 Abstract

A Controller Area Network (CAN bus) is a vehicle bus standard designed to allow Microcontrollers and devices to communicate with each other in applications without a host computer. It is a message-based protocol, designed originally for multiplex electrical wiring within auto-mobiles, but is also used in many other contexts. CAN is an electronic communication bus defined by the ISO 11898 standards. Those standards define how communication happens, how wiring is configured and how messages are constructed, among other things. Collectively, this system is referred to as a CAN bus.

The CAN is a broadcast bus, which means that all nodes can 'hear' all transmissions. There is no way to send a message to just a specific node connected via the bus. All nodes will pick up all the data transmitted in the bus. The CAN hardware, however, provides local filtering so that each node may react only on the interesting messages. CAN uses short messages with maximum utility load is 94 bits. There is no explicit address in the messages. Instead, the messages can be said to be contents-addressed, that is, their contents implicitly determines their address. There are four type of messages sent in form of frames, namely:

- Data Frame - The Data Frame is the most common message type for data exchange
- Remote Frame - It is used to solicit the transmission of corresponding Data Frame
- Error Frame - The Error Frame is a special message that violates the framing rules
- Overload Frame - Overload frame is transmitted by a node that becomes too busy

Along with this the CAN protocol has multiple other features like Bus Arbitration and Message Priority, Message Addressing and Identification and Error Handling.

The goal of this project is to implement CAN protocol based data exchange between two PIC based boards, which are acting as "nodes". In this implementation automotive use-case is assumed by exchanging data like engine temperature, speed etc. that are displayed using LCD panels.

# 2 Requirements

- Default Screen
  - The system should display the current system parameters.
- Mode Selection
  - By pressing the MODE key the display should toggle between Operation Mode and Config Mode
- Config Mode
  - Set Node ID
    - Press CF key to enter this screen
    - The default Node ID should be displayed
    - UP / DOWN key is used to change the node ID
    - ACK key updates the Node ID and stores it in persistent memory
    - Return to the Config Screen
  - Set Thresholds
    - Press CF key to enter this screen
    - UP / DOWN key to scroll the menu
    - ACK key should be used to select the parameter to be changed
    - UP / DOWN key is used to change the parameter
    - ACK key updates the changes in persistent memory
    - Return to the Config Screen
- Operation Mode
  - On request from the server, the system parameters should be transferred to the server
  - On exceeding the set parameter threshold, a message should be sent to the server

# 3 Prerequisites

- Embedded C programming
- CAN, I2C protocols

# 4 Design

- Block Diagrams

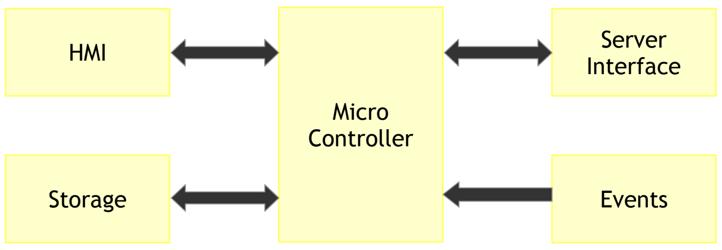


*Fig 4 1: Block Diagram - Level 0*

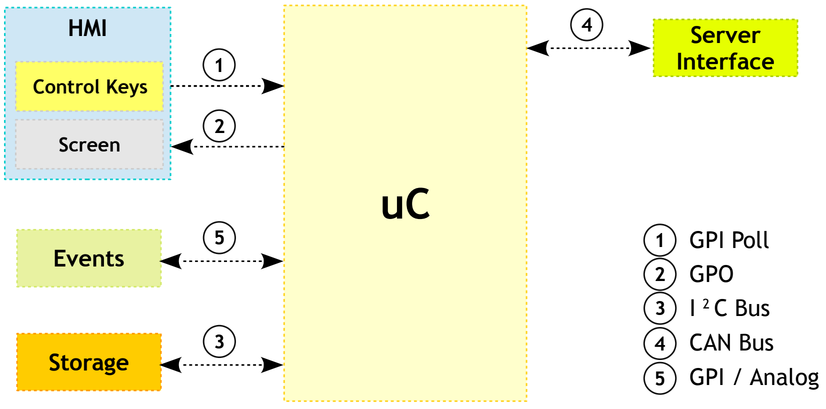


*Fig 4 2: Block Diagram - Level 1*

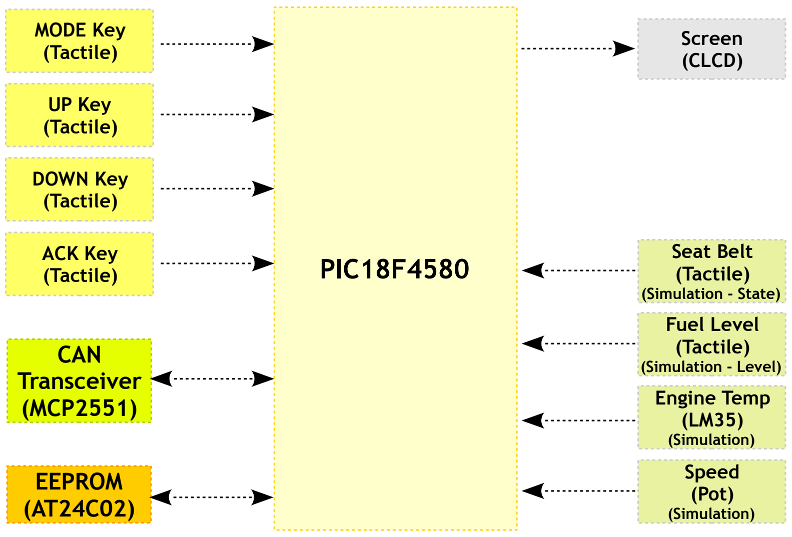
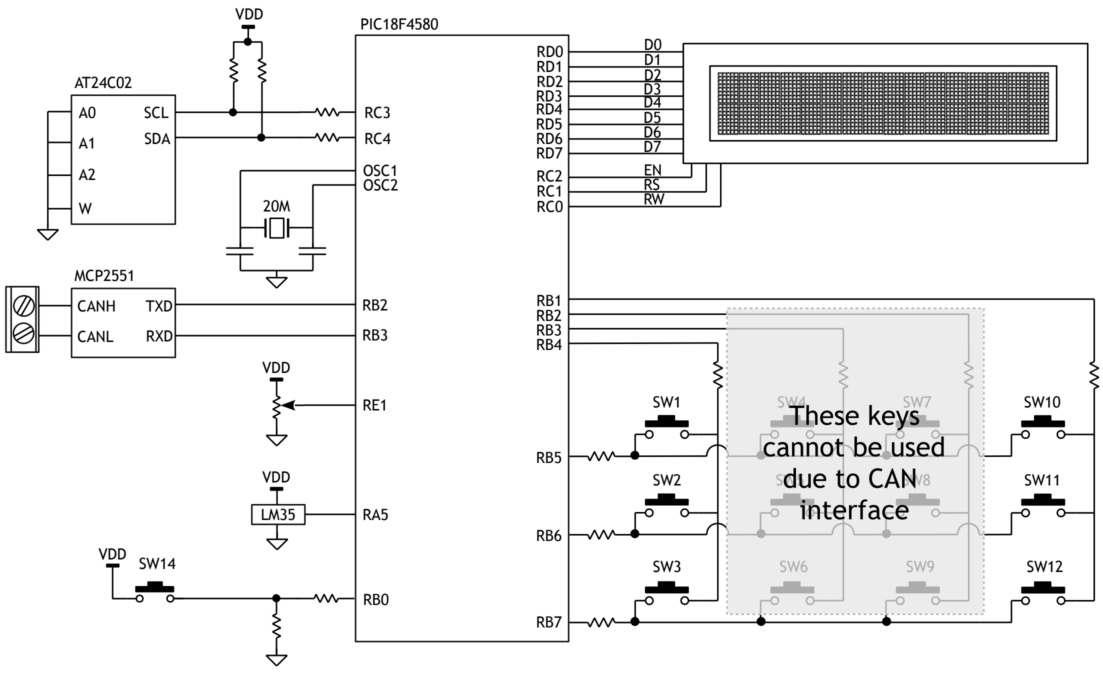

MODE Key (Tactile)
UP Key (Tactile)
DOWN Key (Tactile)
ACK Key (Tactile)
CAN Transceiver (MCP2551)
EEPROM (AT24C02)
PIC18F4580
Screen (CLCD)
Seat Belt (Tactile) (Simulation - State)
Fuel Level (Tactile) (Simulation - Level)
Engine Temp (LM35) (Simulation)
Speed (Pot) (Simulation)

*Fig 4 3: Block Diagram - Level 2*

- Schematic


VDD
PIC18F4580
AT24C02
A0
A1
A2
W
SCL
SDA
RC3
RC4
OSC1
OSC2
20M
MCP2551
CANH
CANL
TXD
RXD
RB2
RB3
RE1
LM35
RA5
SW14
RB0
RD0
RD1
RD2
RD3
RD4
RD5
RD6
RD7
D0
D1
D2
D3
D4
D5
D6
D7
RC2
RC1
RC0
EN
RS
RW
RB1
RB2
RB3
RB4
RB5
RB6
RB7
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
These keys cannot be used due to CAN interface

# 5 Sample Output

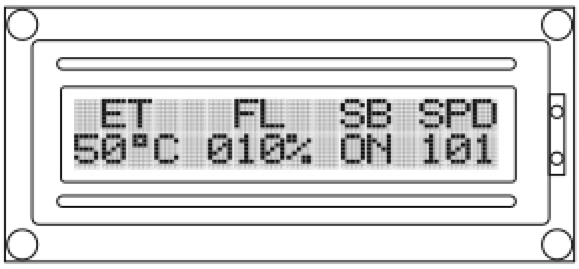


*Fig 5 1: Default Screen while in Operation Mode. By pressing the Mode Key it should take you Config Screen*

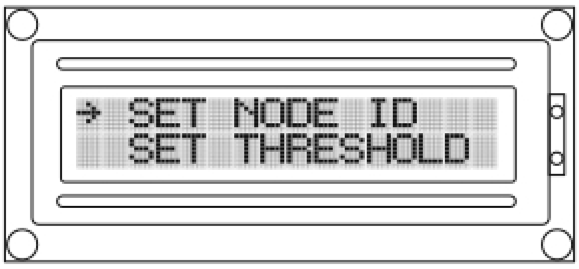


*Fig 5 2: Config Screen. Pressing Mode Key should take you back to Default Screen. By pressing CF key selected item should take you the corresponding function*

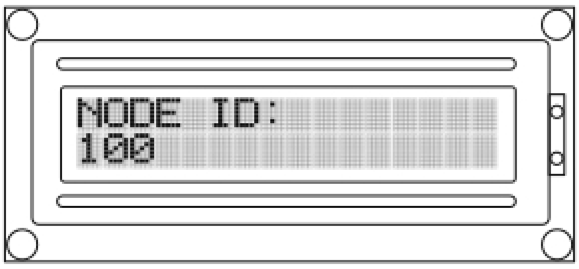


*Fig 5 3: Config Screen. Set Node ID screen. Pressing ACK key should save the data in to memory and go back to config screen*

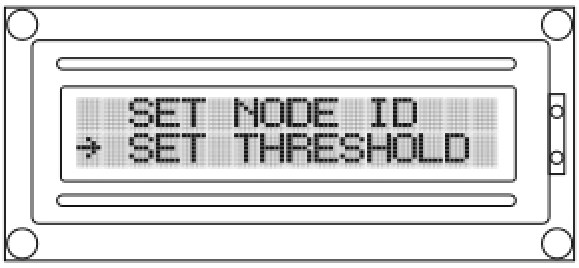


*Fig 5 4: Config Screen. Pressing Mode Key should take you back to Default Screen. By pressing CF key selected item should take you the corresponding function*

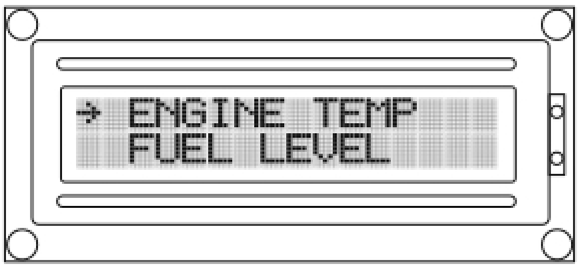


*Fig 5 5: Config Screen. Set Threshold Menu. Pressing CF key should take you to selected option*

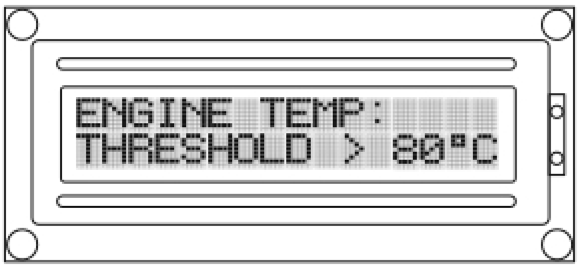


*Fig 5 6: Config Screen. Set Threshold Screen. Engine temperature. Enter the required threshold and press ACK key to go back*

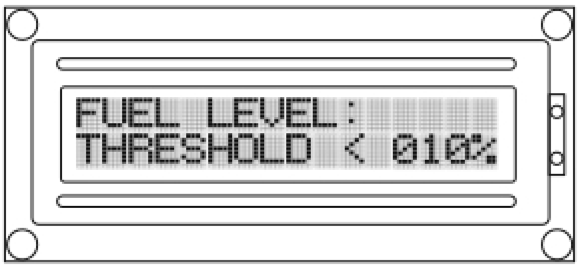


*Fig 5 7: Config Screen. Set Threshold Screen. Fuel level. Enter the required threshold and press ACK key to go back*

# 6 Artifacts

## 6.1 Skeleton Code

The skeleton code is a very interesting concept used in Emertxe. By looking into the skeleton code, you will get a clear picture into converting the given requirement into a working solution. This will also take care of important aspects like modularity, clean coding practices, re-usability etc.

- TBD

## 6.2 References

- https://en.wikipedia.org/wiki/CAN_bus