Expertise
and insight
for the future

Basanta Bhattarai

External Memory Module in Train Automation

Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Program in Electronics

Bachelor's Thesis

3rd June 2020

Author	Basanta Bhattarai
Title	External Memory Module in Train Automation
Number of Pages Date	32 pages 3rd June 2020
Degree	Bachelor of Engineering
Degree Program	Degree Program in Electronics
Instructors	Janne Mäntykoski, Senior Lecturer Altti Helläkoski, Software Engineer Manager

The goal of this project is related to study and integration of new memory device with higher capacity, as an option to existing memory device. It deals with reviewing the related tools and application/firmware support.

The memory device is housed in metal frame and plugged in coaches. The key element of a memory device is EEPROM. It is used in Train Communication Network (TCN) for automatic configuration. It contains several train related data like train length, number for coaches, coach ID, Train ID and so on. The train computer reads configuration data for identification or location of the current computer in network or coach. It also reads several other configuration parameters related to communication, like node ID is decided by reading the attached memory device. The memory used in this project is integral part of train communication.

The project was started with learning and understanding the basics of communication network and the train communication standard IEC61375. IEC61375 enlists several communication protocols used in communication throughout the train. It also provides various data transaction structure and semantics for compatibility for communication within same network between different vendors in a train. This standard provides interoperability between different devices in the network. The newer memory was chosen from same vendor and with same communication protocol as older, to support resources. The 1-wire protocol was the communication protocol which uses the least amount of connection between reader and the device among all available protocol. The memory device is used with application running on top of Embedded Linux OS, which is lightweight and popular among embedded devices.

The project target was achieved by introducing the higher memory size device (EEPROM) of 20 Kilobit as an option for older memory size of 4 Kilobit which has same communication protocol (1-wire). The testing tool was updated to support access of newer memory device. The firmware change for train computer was done to support newer memory while maintaining legacy compatibility to older memory devices. Testing and debugging took major time in the process of development. Software configuration management was followed for managing, organizing, and tracking changes in the software development using tools like, bitbucket, Jira, Git and many other devices that support integrity of software development.

Keywords	Linux, 1-wire, EEPROM, memory devices, TCN, IEC61375

Table of Contents

List of Figure

List of Tables

List of abbreviations and definitions

BCU	Brake Control Unit
CAN	Controller Area Network
CPG	Central Processing Unit with Graphics
CPS	Central Processing Unit with Serial
DIO	Digital Input Output
DCU	Door Control Unit
DRAM	Dynamic RAM
EPROM	Erasable PROM
EEPROM	Electrically Erasable PROM
ETB	Ethernet Train Backbone
ECN	Ethernet Consists Network
ESN	Electronic Serial Number
IEC	International Electrotechnical Commission
I^2C	Inter-Integrated Circuit
MHz	Megahertz
MVB	Multifunction Vehicle Bus
MAC	Media Access Control
PROM	Programmable ROM
RAM	Random Access Memory
ROM	Read Only Memory
SRAM	Static RAM
SPI	Serial Peripheral Interface
TCMS	Train Control Management System
TCN	Train Communication Network (also, Traction Control Unit)
VIU	Vehicle Identification Unit
WTB	Wire Train Bus

1 Introduction

Memory devices have been integral part of electronic measurement and communication systems. Digital data storage facilitates automatic loading of configuration, thus replacing need to configure device manually. In computer cluster in the network, a computer accessible/readable storage medium can help initialize the configuration, like its node ID in network. Similarly, in train communication system, there can be cluster of computers working together from different coaches, over the same network monitoring the condition or state of train. Static data storage facility with such computer cluster in a network can help with node/location identification and thus help to automatically initialize the configuration for that coach. Digital data storage can also protect data integrity and maintain accuracy in contrast to manual configuration of the system. Electronic measurement system might need calibration data in course of time or during startup for reference. Storing calibration data in external memory can maintain accuracy and precision with measurement devices. Memory devices can retain data over multiple decades and the data can be read/written multiple times which also decreases repair overhead costs. Ability to retain data over decades can also ensure safety and smooth running of services for the application with longer life-cycle.

The project is done in premise of EKE-Electronics Ltd with resources and tools provided by the company. Train automation and management system is the area of expertise of the company. The train automation system uses computer to monitor the state and condition throughout the train. The train communication system uses digital data storage device to aid in communication processes. The purpose of the project is to study and integrate the expansion of existing memory device size used in digital data storage in train communication network, while maintaining legacy compatibility.

Furthermore, the accessing applications and tools must also be reviewed to support new memory with higher capacity. Memory devices, in this project are used to store train configuration data which facilitates train configuration and identification. Several proprietary modules related to memory device from EKE Electronics Ltd have been mentioned in this document. Software or tools used to achieve this will be described in here. The roadmap of study, method and implementation is also described in upcoming sections.

2 Train Communication Network

Train Communication Network (TCN) is architecture of information flow throughout the train for condition monitoring. The standard for railway communication, IEC61375 defines it as hierarchy of two network levels, a train backbone level and a consists network level. Train backbone level contains bus connecting the vehicles (coaches) of a train which conforms to the TCN protocol and is dynamic network as it changes its topology each time there is a change in train composition. It can be implemented by WTB or ETB for train wide communication. [17, 1]

Consist network is a communication network established between devices in single vehicles or a group of vehicles which are not separated during normal operation as shown in Figure 1. Intra-vehicle communication or consist network is implemented with bus technologies like MVB, Serial Links, CAN or switched technology like ECN (Ethernet Consist Network) and are static and preconfigured network. There can be vehicle specific configuration on top of consists network configuration. The different vehicles in a consist can contain different end devices like Brake Control Unit (BCU), Traction Control Unit (TCU), displays or controller devices. Thus, there is a necessity of vehicle or consists configuration and identification system [1].

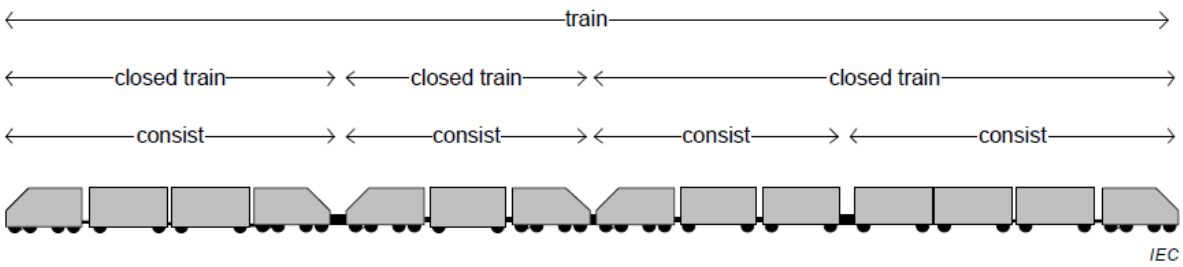

Figure 1: Train structure as defined in IEC 61375-1 [4,3].

2.1 Train Backbone

Train backbone refers to the topology of network that connects multiple train backbone nodes. Node or Train Backbone Node (TBN) in this context refers to the devices in one vehicle/coach network that connects/monitors the end devices like traction control unit,

temperature probes etc. as shown in the Figure 2. The train backbone facilitates train wide communication of nodes. Some of the train computer modules acting as train backbone node will be discussed in this document. End Devices (ED) as shown in the figure below refers to several monitoring devices connected to TBN like, sensors actuators, power converters and so on.

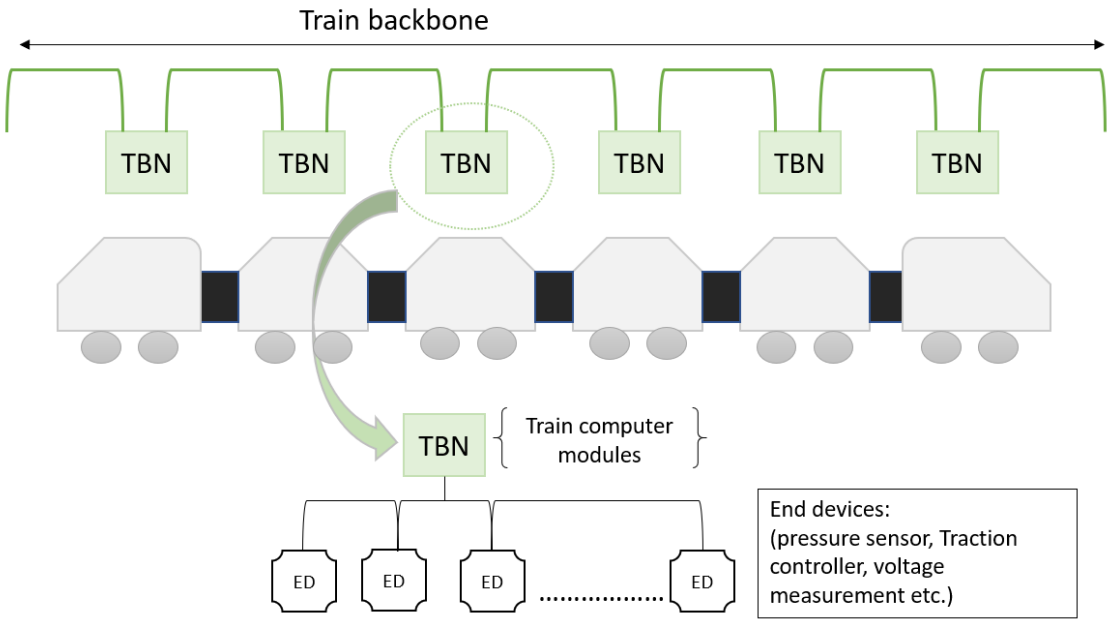

Figure 2: Train backbone and nodes [1]

2.1.1 Wire Train Bus (WTB)

It is the bus type for establishing train backbone. It is a serial data communication bus designed for interconnection of train backbone nodes. The data rate is 1,0 Mbit/s which translates to 1MHz of frequency as per Manchester encoding [2].

2.1.2 Ethernet Train Backbone (ETB)

ETB is the train communication network standardized by International Electrotechnical Commission (IEC) with IEC61375-2-5: Ethernet train backbone. It is based on ethernet technology. The standard mentioned earlier covers the description of OSI layers for connected devices. There are seven layers. They are: Application, Presentation, Session, Transport, Network, Data, Physical. It is faster as compared to WTB with Gigabit ethernet connection.

The Table 1 shows mandatory requirement for OSI layers as defined in IEC61375-2-5.

Table 1: Mandatory requirement for OSI layer implementation defined in IEC61375-2-5 [3]

OSI Layers	Mandatory requirement	Description
Application	Not mentioned in IEC61375	
Presentation	Not mentioned in IEC61375	
Session	Not mentioned in IEC61375	
Transport	ICMP, RFC 792	
	IGMP, RFC 2236	ED shall support IPv4 multicast.
	UDP, RFC 768	
	TCP, RFC 793	
Network	ARP, IETF RFC 826	
	IPv4 Internet Protocol IETF RFC 791	
	Hostname	Name of ED. shall be unique in its owner Consist whatever the ETBN is connected to .

		Should be statically defined: read from a local permanent memory, an external coding key, etc.
	IPv4 address	In range 10.128/9 (IEC61375-2-5: 6.4.2) , Dynamically defined following Inauguration (IEC61375-2-5: 6.5.2)
	IPv4 mask	255.255.192.0
Data Link	Mac services and addressing, IEEE 802.3	
	Frame Relaying, IEEE 802.1D PICS A. 7	Frame reception/transmission. Forwarding process which comprises: Queuing, QoS Priority mapping, FCS calculation, etc.
	Frame Filtering (layer 2 filtering) IEEE 802.1D, Clause 7, PICS A.8	Learning process, Filtering data base (Mac addresses, ports, VLAN association), static/dynamic entries
	Frame Queuing IEEE 802.1D, 7.7.3, 77.4 PICS A16- Annex G	Multiple traffic classes (TC) for relaying frames; assign ingress frames a defined priority
	Frame tagging/untagging IEEE 802.3, 3.5 IEEE 802.1Q (VLAN)	Ethernet frames can be tagged during switch port ingress. The tag can then remain within the frame or can be removed during port egress.
	VLAN services IEEE 802.1Q (VLAN), PICS A.21	Helps subdividing the physical LAN in different virtual LANs.

	LLDP protocol Link Layer Discovery Protocol, IEEE 802.1AB	Used by Train Topology Discovery Protocol between trains
	Management and Remote Mgt IEEE 802,1D, Clause 14 PICS A.14, A.15	Configuration of the switch, Fault management, Performance management
Physical (Inter car and inter consists)	100 BASE TX Physical Layer Coding, Medium attachment, and Medium Dependent Access (PCS, PMA, PMD) for copper cables	Conformance to IEEE 802.3.2012, Clause 24, and 25
	Full Duplex Mode	Conformance to IEEE 802.3.2012, Clause 25
	Physical Layer crossover	Conformance to IEEE 802.3.2012, 25.4.8
	Cables CAT5e	ISO/IEC 11801, IEC 61156
	Segment performance	Conformance to IEC 62236-3-2 Ethernet Certification compliant to ISO/IEC 11801:2002

2.2 Consists Network

A consists is a concept in TCN network that is defined by IEC61375 standard as train set, rake of coaches or single vehicle or group of vehicles which are not separated during normal operation and containing no, one or several consists networks.

Consists Network facilitates inter train backbone node communication. It can be implemented using bus topology (MVB, CAN open) or switched topology (ethernet) [1].

3 The Open System Interconnect (OSI) Model

The OSI Layer was invented in 1980s to standardize the pattern for computer networking. The OSI layer defines network into various division or fragments, working together in communication. It covers low level hardware to high level application access for transmitting data between devices throughout the network. The Figure 3 shows the OSI communication layers.

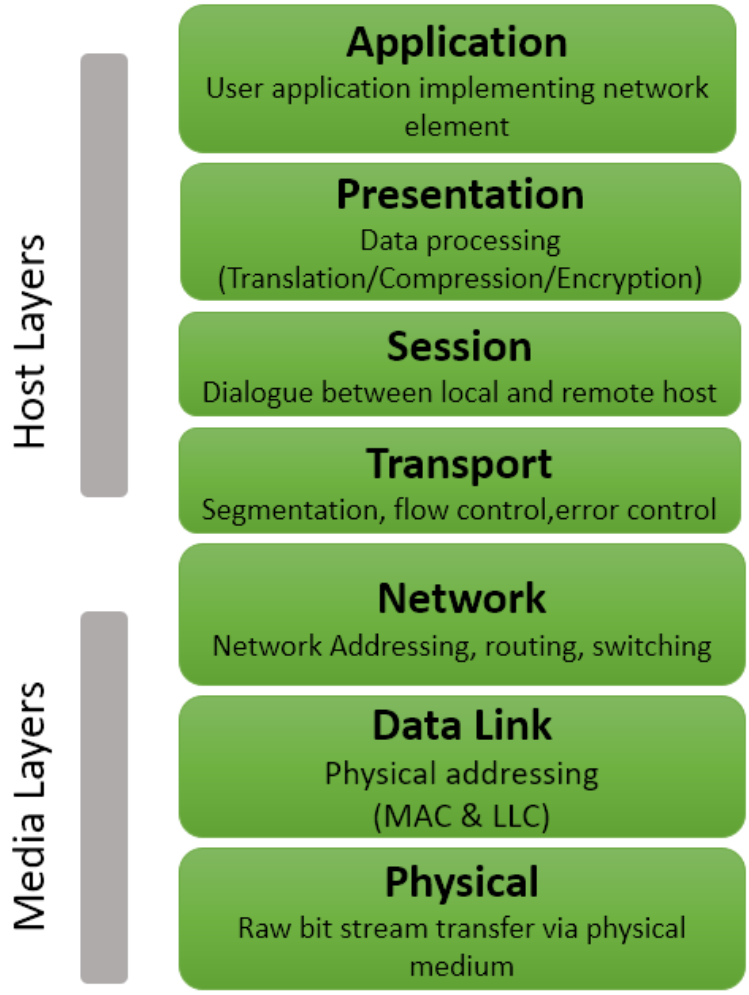

Figure 3: OSI Layers [5,30]

3.1 Application Layer

The applications that access the network for communication reside in this layer. The user interacts directly with this layer via application. This layer contains the protocol that are used by user application like HTTP which is used as a protocol for accessing webpages/websites. This layer also deals with electronic mail (IMAP), File transfers (FTP, SFTP), session management (SSH, TELNET) etc.

3.2 Presentation Layer

The data passed from application layer may or may not be understood to the destination user application. There can be data structure mismatch between source and destination. The presentation layer deals with data translation, compression, encryption to send. It is related to structure and semantics of information/data. This layer also does reverse operation of decompression, translation, and decryption of data upon arrival from destination. The encryption of data is done by Secure Socket Layer (SSL) for data encryption,

3.3 Session Layer

The application in the network requires to open a session or connection with each other for communication. It also includes authentication for example: user credentials (login and password). The authorization, like access to certain file, is also handled by this layer.

3.4 Transport Layer

The transport layer breaks data into segments (for TCP) or datagrams (for UDP). Each fragmented data is padded with header that contains source/destination port number and many other information. This layer also deals with flow control and error control. There are two protocols handling transport of data.

Segmentation, flow control (transmission rate) ,an error control (checksum added in segment for transmission redundancy).

3.4.1 Transmission Control Protocol (TCP)

TCP is connection-oriented transmission. The application requiring guaranteed delivery uses TCP as delivery mechanism. The checksum is also part of TCP segment as an error control mechanism. Some of the examples include: Email, File transfer, World Wide Web uses this method of delivery. Flow control is a feature of this protocol to make sure the transmitter does not overwhelm receiver by sending data faster than it receives or consumes. The various features and composition of TCP header can be seen in Figure 4

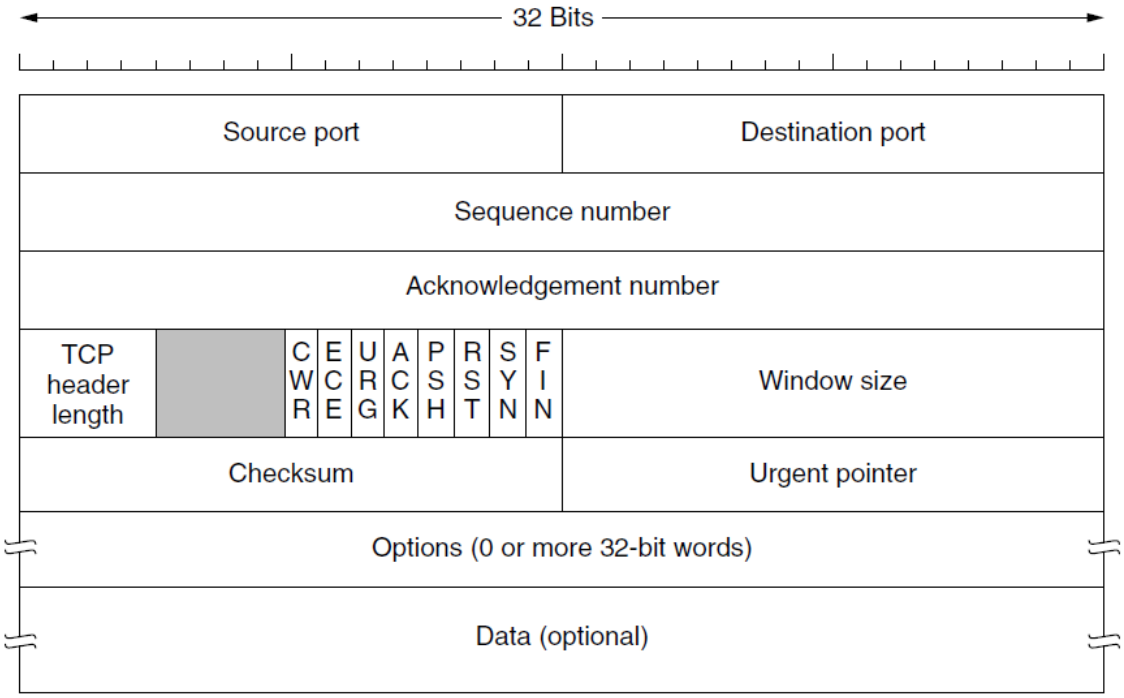

Figure 4 : The TCP header [6,557]

3.4.2 User Datagram Protocol (UDP)

UDP is faster as compared to TCP as there is absence of features which are present in TCP as seen in the Figure 5 which is UDP header. It is connectionless transmission with no flow control (full speed), no error correction and many other operations. This method of transmission is beneficial for online video streams, VOIP, etc.

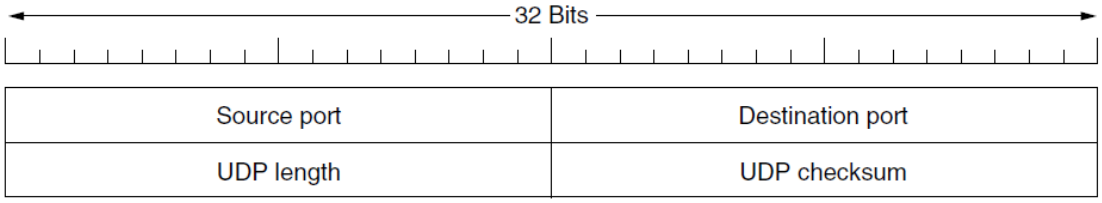

Figure 5 : The UDP header [7,542]

3.5 Network Layer

Routing and addressing are the dealt within this layer. This layer decides the path and optimizes it to take shortest route to destination via different Local Area Networks (LANs) or Wide Area Networks (WANs). The data arriving here from transport layer are padded with header and are called packets. The common example of this layer is Internet Protocol (IP). The IP (v4) header format is shown in Figure 6.

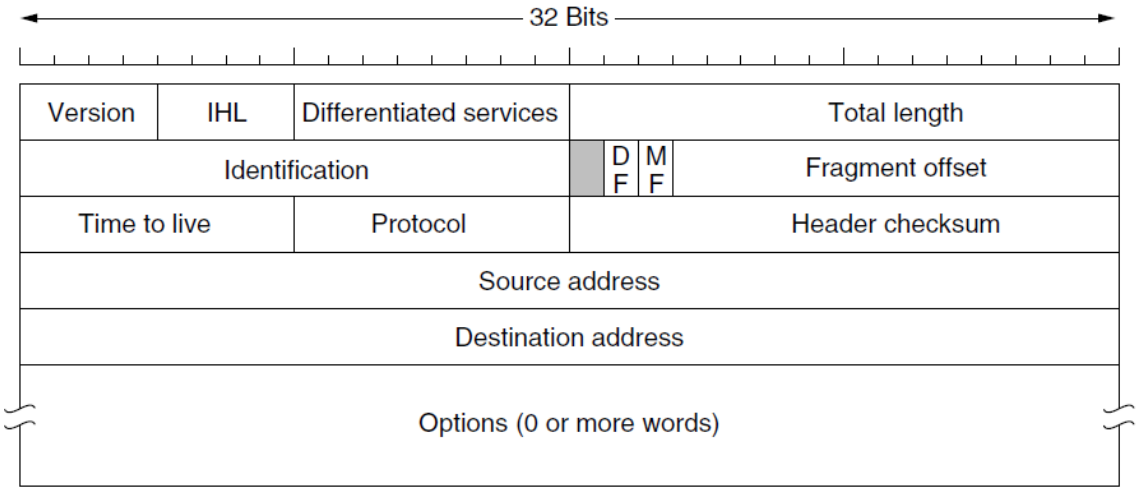

Figure 6: IP (Ipv4) header format [8,439]

3.6 Data Link Layer

Physical addressing is done at this layer. This layer deals with how the data is placed and received from the media. Data in this layer is called frame. Media Access Control (MAC) controls the hardware that interacts with physical medium like wire, optical fibers, or wireless transmission medium. There can be multiple devices connected to the same media. Data link layer verifies the traffic availability before transmitting via shared transmission medium, this is called Carrier Sense Multiple Access (CSMA). The CSMA avoids, detects & resolves the frame collision. Network switches and Bridges operate at this layer. The flow control, control bit sequencing and error checking is handles in this layer by Link Logic control Layer (LLC) which is present on top of MAC layer.

3.7 Physical Layer

Physical Layer defines hardware level implementation of network communication. It specifies the mechanical, electrical, and functional aspect of physical layer. It deals with conversion of raw data bits into electrical signal to send or receive via a transmission medium . It deals with cables, patch panels, physical connection socket etc.

4 Memory Device Types

Memory devices are available as per various parameters like volatility of data storage, size, speed, internal or external. Memory device types as per volatility of data storage is the scope of this document. This document is focused on EEPROM memory and its usage. However, some of the popularly used (in past) or in current use has been mentioned below.

4.1 Random Access Memory (RAM)

This type of memory devices can be accessed (read/write) in any order as required by the processor. Data is stored and read many times from this type of memory. It is principally used by operating system and application for temporary storage. It is also called volatile memory as data is lost when it loses power. There are various types of RAM and some of them are listed below.

- Static RAM (SRAM)
- Dynamic Ram (DRAM)

4.2 Read Only Memory (ROM)

ROM is non-volatile memory, meaning data is not lost when power is removed. This type of memory device is primarily used for storing data for permanent use like BIOS of a computer is stored in ROM. Appliances firmware are stored in ROM. They are primarily designed for reading data. There are various types of ROM depending upon mainly programming features [9].

4.2.1 Programable ROM (PROM)

PROM consists of an array of fusible links and bit patterns (data) are formed by burning fuses during the programming process and as a result it cannot be erased after programming [9].

4.2.2 Erasable PROM (EPROM)

These are special type of programmable ROM which can be programmed as well as erased. Internally they contain field-effect transistor, which has insulating layer or oxide over the channel and then conductive gate electrode of silicon (floating gate) and further thick layer of oxide is grown over it followed by one more layer of silicon layer (as gate) as shown below in Figure 7 [9].

They are programmed by applying high voltage over the terminals resulting in electrons passing through insulating oxide layer and accumulated or trapped on the gate electrodes. The presence of these charge turns-on the channel or creates the conductive path which usually refers to logic “0” . They are erased by applying Ultraviolet light over a quartz window, that ejects the charge out of gate and thus the channels are non-conductive referring to logic “1” [9].

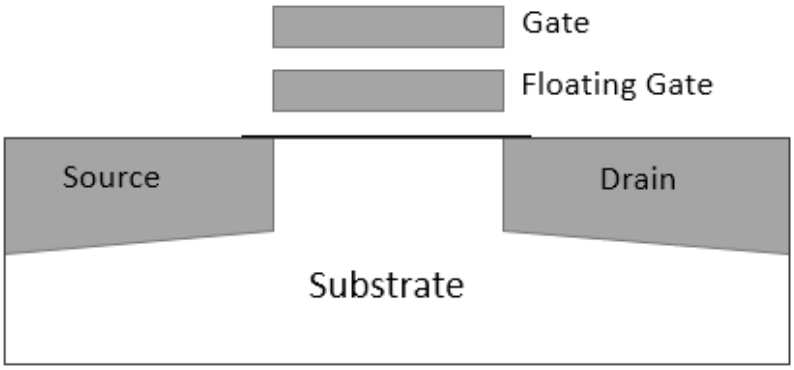

Figure 7: Internal structure of EPROM [10]

4.2.3 Electrically Erasable PROM (EEPROM)

One of the many advantages of EEPROM over its predecessor EPROM is ability to erase the device electrically. They have similar internal structure to EPROM but often EEPROM memory cell is built with two field effect transistors where one of these acts as storage transistor [9].

EEPROM can have serial or parallel electrical interface. The communication protocols used to access these devices are discussed in upcoming section. They are popular form of permanent data storage in different communication systems or appliances [9].

5 Electrical Interfaces of EEPROM

The electrical interface of the memory devices can be classified into two basic classes: serial and parallel interfaces. Parallel interfaces are faster than serial interface when comes to access of device, but with compromise of higher pin count. Serial interfaces are most popular form of interfaces compared to parallel interfaces. Some of the serial interface are SPI, I^2C, Microwire, UNI/O and 1-wire etc.

5.1 Serial Peripheral Interface (SPI)

SPI is a synchronous communication utilizing master-slave architecture. It can communicate in full duplex mode. It has four signal wires: Serial Clock (SCLK), Master Output Slave Input (MOSI), Master Input Slave Output (MISO), Slave Select (SS). The Figure 8 shows a master and a slave configuration. However, it can also be configured with one master and multiple slave configuration.

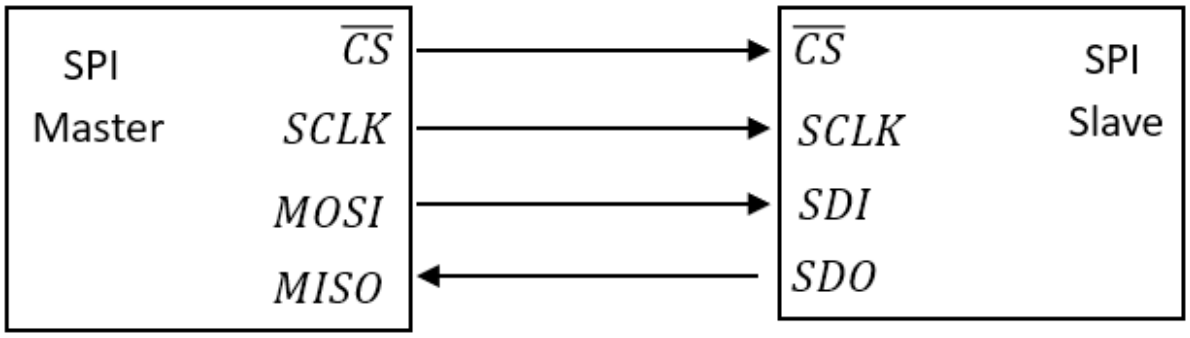

Figure 8: SPI configuration with and a slave [11]

To start the communication, master is configured with certain frequency (typically few MHz). Master pulls the Slave Select (SS) line to low (or ground) and sends data on MOSI line, while the slave sends the output or reply for each bit in MISO line, thus forming full-duplex communication.

5.2 Inter-Integrated Circuit (I^2C)

I2C is multimaster two wire synchronous communication interface. It has Serial Data (SDA) and Serial Clock (SCL) in two lines which are pulled up by a resistor to power supply. Communication is initiated by the master using slave address. Slave address is of 7 bits with maximum 127 slave attached to same line. The model connection is shown in Figure 1. The data rate of I2C bus can vary from standard mode of 100kbit/s, full speed of 400kbit/s until high speed of 3,2Mbit/s [11].

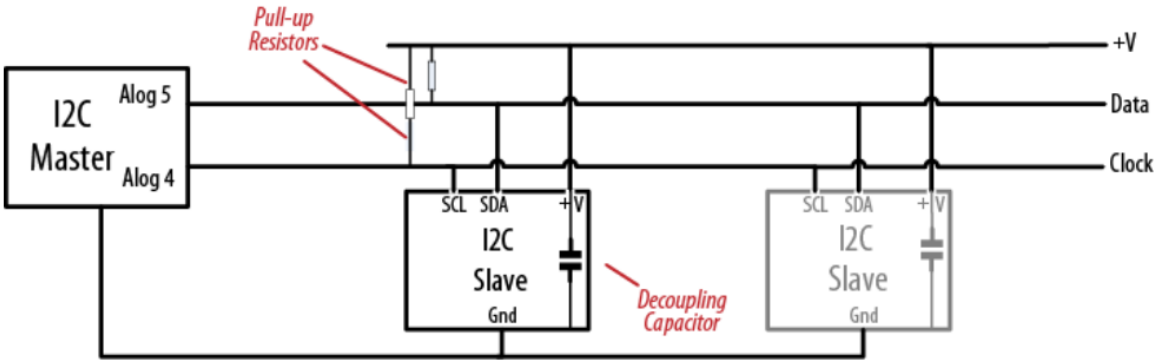

Figure 9: One master connected to multiple slaves in single bus [12]

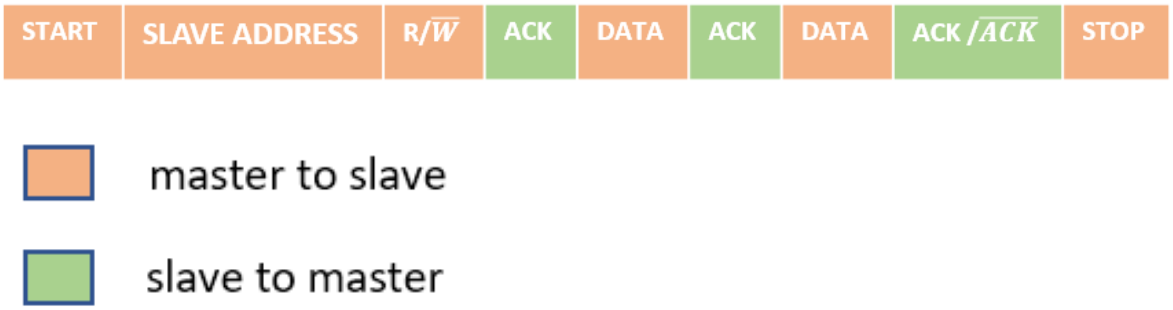

Figure 10: Master writing to slave via I2C interface [13]

Master initiates the communication by pulling SDA to low, indicating start condition. After the start condition has been initiated, it can only be used by that master. Start condition is followed by 7-bit slave address byte and 1 bit of data direction (read / write) . Writing to the slave device require the bit 0 of address byte to be set 0 and reading slave device requires the bit 0 of address byte to be set to 1. Data in 8 bits is transferred and finally stop condition is raised by releasing the SDA to high. Each byte (8-bit) transfer results

in transfer of acknowledgement (ACK) bit to be transferred in other direction as shown in Figure 10. Figure this shows the master and multiple slave configuration in a bus.

5.3 1-wire communication

1-wire communication is voltage-based system that works with two contacts, data, and ground, for half-duplex bidirectional communication. It has master-slave communication architecture and master can have multiple slaves in the same line listening as shown in Figure 11. Data line is used for both way communication and is pulled up by a resistor upon requirement. The legacy standard speed of 1-wire support 16.3 Kbps and overdrive speed being 125 Kbps. However, there can be differences in exact baud rate among different family device of 1-wire. [14.]

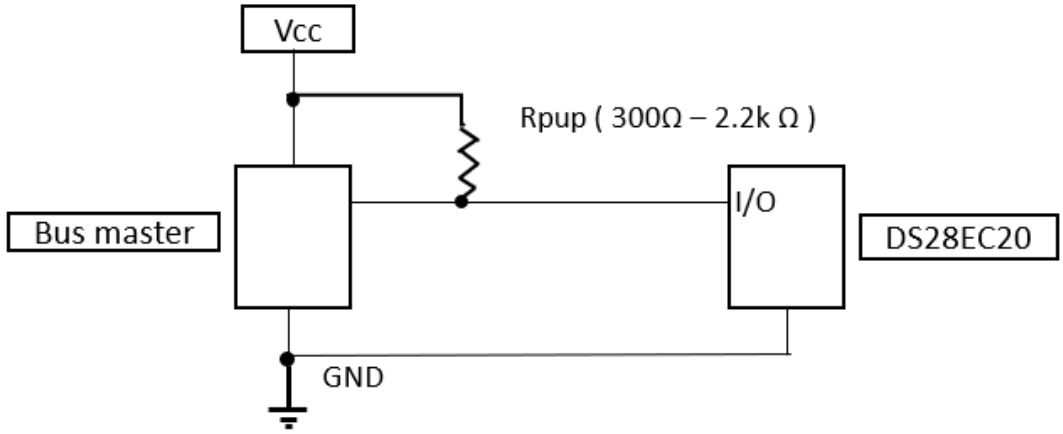

Figure 11: Typical operating circuit [16]

The Figure 12 shows typical communication procedure using 1-wire communication using standard speed of 16.3 Kbps. The communication starts with a reset-/presence-detect cycle. Master starts the communication by pulling the data line low for at least 480 µs, then releases it and goes into receive mode. This action resets the slave devices available in the line and transmit the “presence” pulse by holding the line low for at least 60 µs depending upon the number of slaves present in the line. After the reset-/presence-detect cycle is completed, a 1-Wire master/slave is ready for communication using read/write time slots. Master initiates write cycle to slave device with binary 1 or logic high by pulling the data line low for 1-15 µs and releases the line (50 µs) and wait until it reaches at least 65 µs total (this defines the bit rate). Binary 0 is represented by holding line low for 15 µs and keeping it low for at least 65 µs in total hold period. When master

wants to read from slave device, master pulls the line low for 1-15 µs and releases it and samples the data line somewhere in between the bit period. If the line is still high after master pulls and releases it after 15 µs then first bit is 1 whereas if the line is low even if master released after 15 µs then it is logic low or bit 0 as slave is pulling line low. Again after 65 µs the same procedure is done until 8 bits are read. In short, the sequence is reset-/presence-detect cycle followed by command of 8-bit and then receive data in 8-bit. [14.]

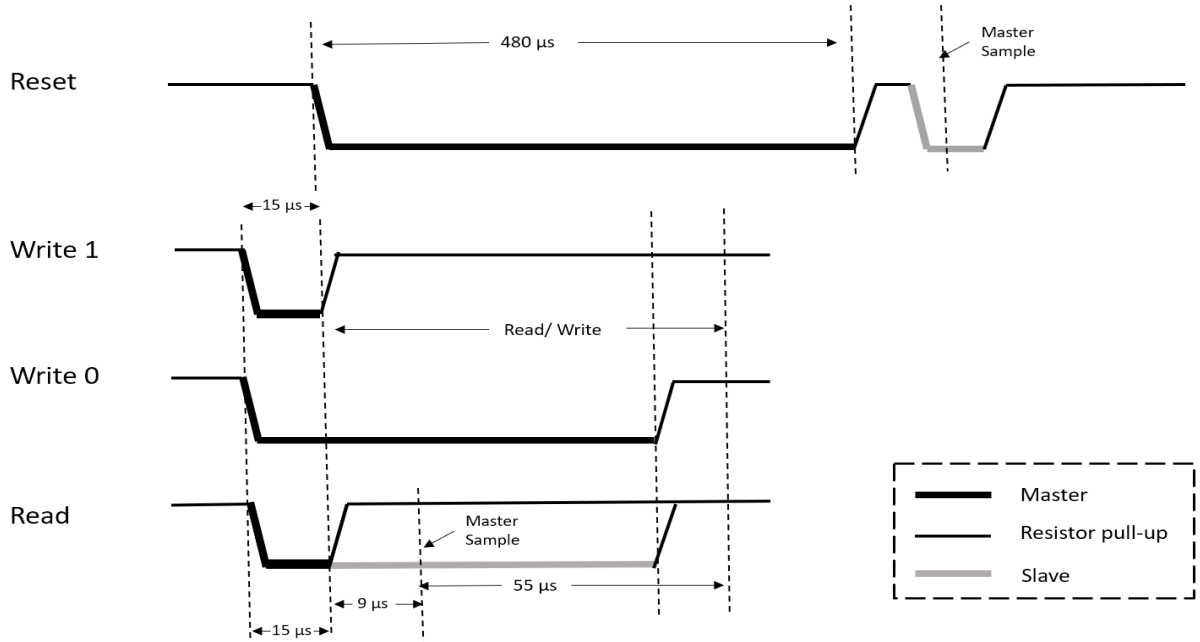

Figure 12: 1-wire communication architecture [15]

6 Embedded Linux Operating System

It is open source operating system customizable as per project or user basis. The computer/modules in this project were powered by embedded Linux OS. Embedded Linux operating system is stripped down version of full Linux OS, that has Graphic User Interface (GUI) and various peripheral support. The unwanted features from it are removed to have optimal efficiency in resource constraint environment. The difference in traditional embedded operating system and Linux is isolation of kernel and user applications resulting in more secure and robust system. [19.]

6.1 The Shell

The Linux Shell is the combination of command line interpreter and environment of pre-defined functions and variables. Users input is received and translated into predefined action by interpreter. It can also receive file as an input called shell script and command the computer to execute the instructions. Beside that user is also able to set-up their own environment setting for current session like instructions, commands shortcut (aliases) and values stored in variables. The Figure 13 shows the elements of Linux shell. The shell in Linux can be accessed via terminal connection [18.].

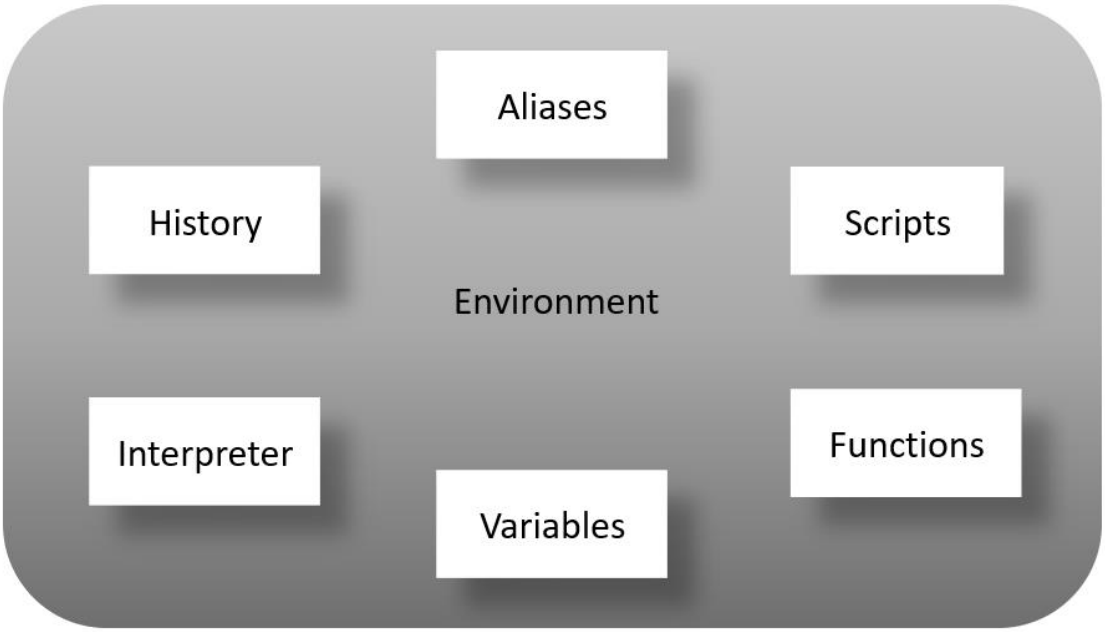

Figure 13: Components of a Linux shell [18]

There are many types of command line interpreter such as Bourne shell (sh), Bourne-Again shell (bash), C shell (csh) etc.

6.2 Linux Boot Process

Embedded Linux operating system ‘s boot process starts when power to the hard-ware/board is turned on. The bootloader executes when CPU comes of reset. It initializes various minimum required processes before loading and passing the control over to

Linux kernel. Linux kernel loads drivers, mounts the root filesystem, executes the *init* process and other user processes to complete the bootup process.

6.3 Device File

Device file, also known as special file in Linux, is mapping of the interface between the user's interaction using a set of standardized calls that are independent of specific driver, with device-specific operations performed on real hardware which is done by device driver. In Linux, they are traditionally accessed under */dev* directory. There are two fundamental device file or special file type. They are character and block devices.

6.3.1 Character Devices

Character Devices are accessed like a file. Character device driver takes care of implementing the "*file like*" behavior like *open, close, read, write* etc. system calls. In this project the memory data is stored in char device after reading.

6.3.2 Block Devices

Block devices handle buffered data unlike character devices. They are mostly used with flash, hard disk etc. It permits any number of byte transfer at a time [20].

7 Hardware Architecture for Memory Device Access

7.1 Introduction to DS28EC20

The DS28EC20 is a Nonvolatile EEPROM partitioned into 80 memory pages of 256-Bit each. Additional volatile page of 32-byte (256-bit) scratchpad is present that acts as a buffer to write data to the EEPROM to ensure data integrity. Data written to this buffer can be read back for verification before uploading to memory. There is also a page for control functions such as permanent write protection. The communication to device happens over single data line via 1-wire protocol. This device family supports both standard speed of 15.4kbps and overdrive speed of 90kbps maximum. It has fixed globally unique 64-bit ROM registration number for item traceability. It can be used in multidrop

operation, which means multiple devices attached to the line can listen to master simultaneously and the same data line can be used to communicate with any slave. The Figure 14 shows the internal block diagram of the device. [16.]

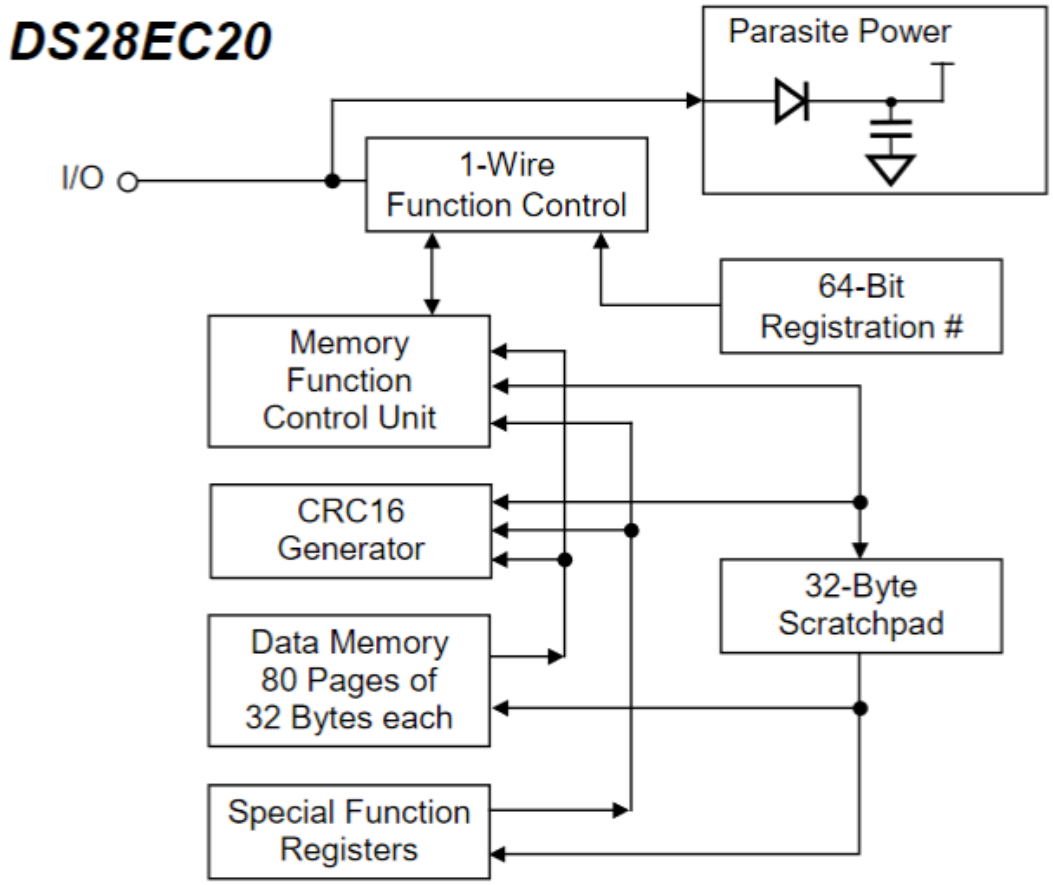

Figure 14: Block diagram of DS28EC20 [16]

7.1.1 Data Transaction Protocol

The Figure 15 shows the data transaction sequence for DS28EC20. The read/write operation of the device occurs in the following steps.

1. Initialization sequence:
   It consists of reset pulse (write zero) transmitted by the bus master and slave replies with presence pulse (write one) and is ready to progress with executing ROM Function Command.

2. ROM Function Command:
   There are seven ROM command bus master can execute. The ROM Function Commands are of 8 bits long. It is mandatory to execute one of the ROM commands to proceed further to Memory Function Command.

3. Memory Function Command:
   Figure 15 shows memory access commands. Writing to memory must start with writing to 32-bit scratch pad and then followed by copying scratchpad to memory.

4. Transaction/Data:
   After executing Memory Function Command, the slave starts the data transaction with the bus master based on Memory Function Command.

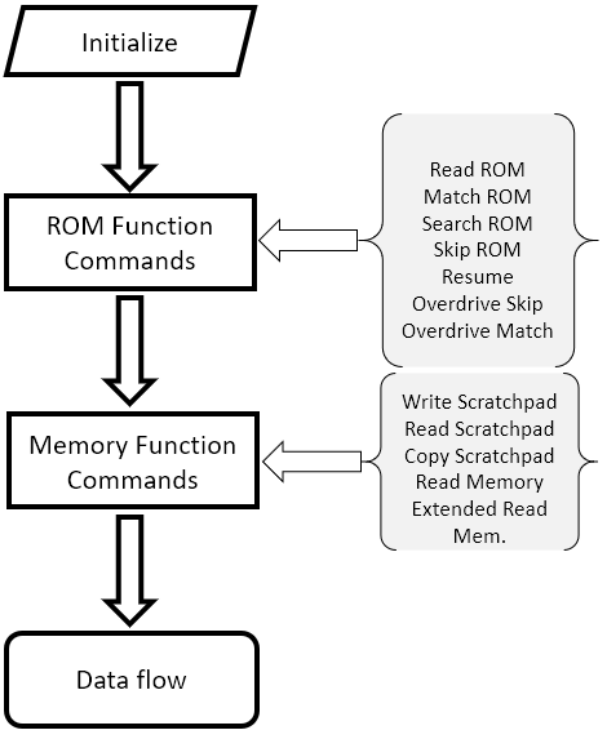

Figure 15: Device access method [16]

7.2 Electronic Serial Number (ESN)

Electronic Serial Number (ESN) is a module that encloses EEPROM memory like DS28EC20, with bare minimum circuit to make it easy to be integrated into the rack module (VIU) that accesses the device. The Figure 16 shows ESN and its internal physical layout.

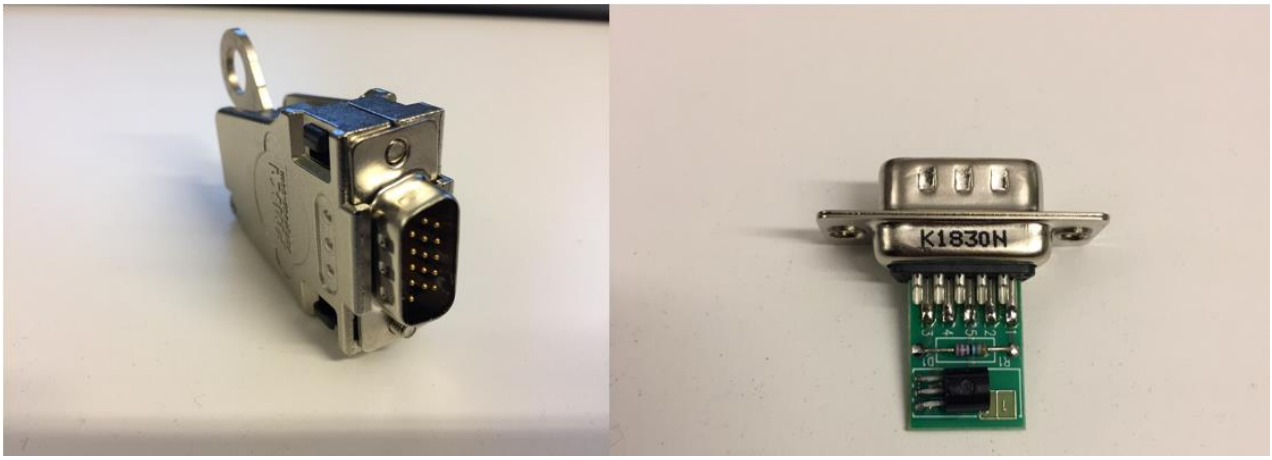

Figure 16: ESN module and its internal physical layout

7.3 Hardware Modules

The hardware modules seen in Figure 17 is designed by and is property of EKE-Electronics Ltd. The hardware modules come with standard VME connector on its back. There are hardware modules reading ESN locally attached to it or via I/O bus through backplane. Some of the modules and their purpose are described in this document.

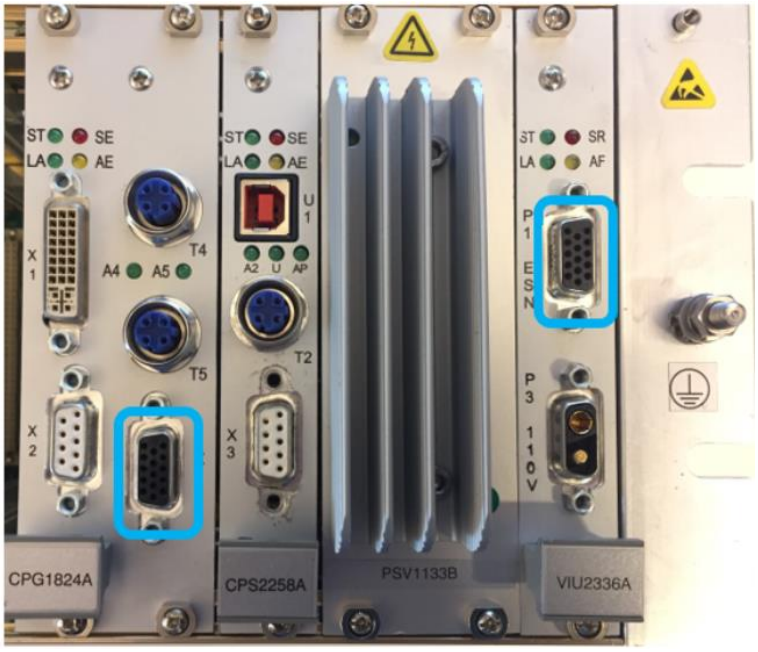

Figure 17: VIU module (rightmost) with connector for ESN and train CPU (two leftmost)

7.3.1 Vehicle Identification Unit (VIU)

VIU is host for connecting ESN module as shown in Figure 17. It also has power supply socket which is input power. The power converter outputs various voltage level to back-plane from where other modules are connected to various voltage supply. VIU module has Field Programmable Graphic Array (FPGA) chip which accesses the memory via 1-wire communication. The computer module accesses 1-wire by RS-485 communication via I/O bus in backplane of the module rack. The Figure 17 with the blue highlight shows the connector in front panel for hosting memory connection in the VIU module and CPG module (computer) inside the rack slot.

7.3.2 Central Processing Unit with Serial Links (CPS)

CPS is used for train control automation, data processing or diagnostics. It runs on Linux operating system. In Figure 17 there is CPS module (in middle) with model number CPS2258A. It has Ethernet, Serial, USB port for various connectivity to subsystems for monitoring and diagnostics. It reads the ESN module via VME backplane and stores it in character device.

7.3.3 Central Processing Unit with Graphical Display Controller (CPG)

CPG also shares similar features with CPS. In addition, it includes DVI interface to drive display applications for data visualization and monitoring the status. In Figure 17 the CPG1824A lies on left most side. It also acts as host for ESN module. During bootup it looks for ESN module connected in front panel to read, if unavailable it attempts to read ESN via backplane to initialize various configuration. [17.]

7.4 USB to 1-wire Adapter

The data storage in memory can be done by external tool from Maxim Integrated, as shown in the Figure 18. It has USB to 1-wire converter adapter and RJ45 jack type connecter to connect HD15 female connector on the other end to be connected to memory device module (ESN). The application used to program the memory device will be discussed in later sections.

Figure 18: USB to 1-wire adapter (DS9490R) with cable and HD15 connecter

8 Software Architecture for Memory device Access

8.1 Memory Device Access and Store Architecture

In this project, EEPROM is used as vehicle identification for facilitating automatic inauguration of train configuration for various applications. The semantics of accessing data can be seen in the Figure 19, it shows the memory device read process upon system bootup. If there is ESN connection available in train computer front panel then it is read during bootup process also, like CPG module seen in Figure 17 and if unavailable, it is read via backplane from VIU module. The content is saved into kernel memory space for the access by other application. This shows that the data can be accessed locally or remotely via shared bus.

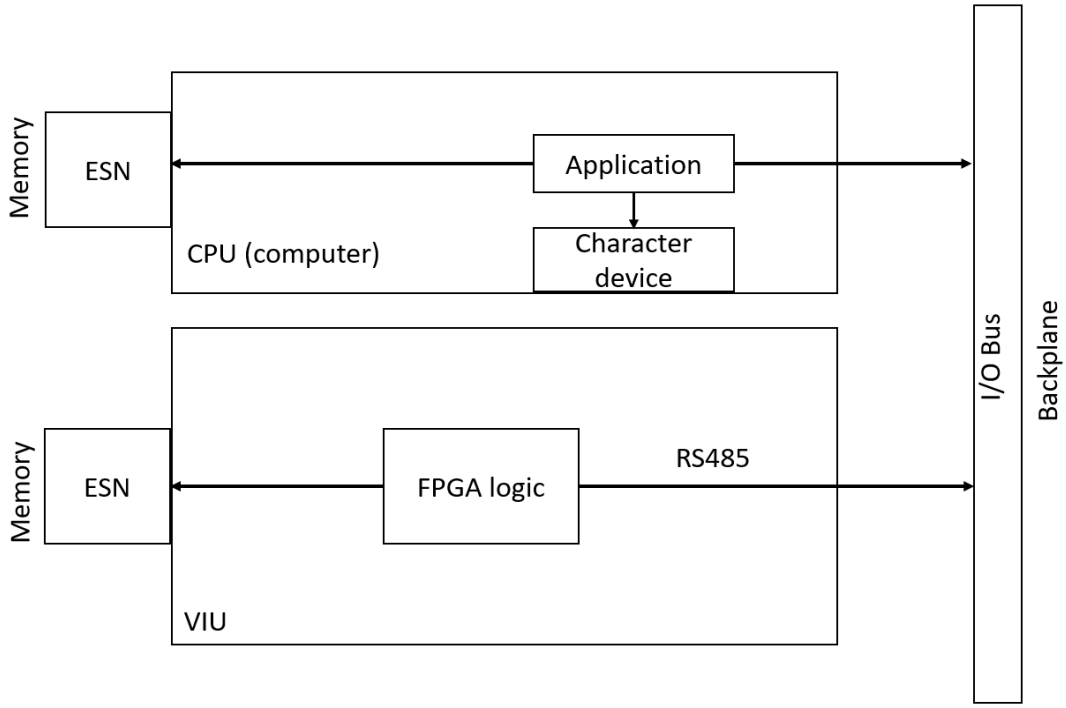

Figure 19: Architecture for accessing memory device [21]

8.2 Windows Software for Accessing Data

The accessing of memory device can be done using Microsoft Windows Graphic User Interface (GUI) for writing and reading of data. The additional hardware as shown in Figure 18 was required in conjunction with GUI software shown in Figure 20. The Figure

20 shows the Windows-based application modified from existing application to support reading of memory devices with higher capacity (DS28EC20) for test purpose.

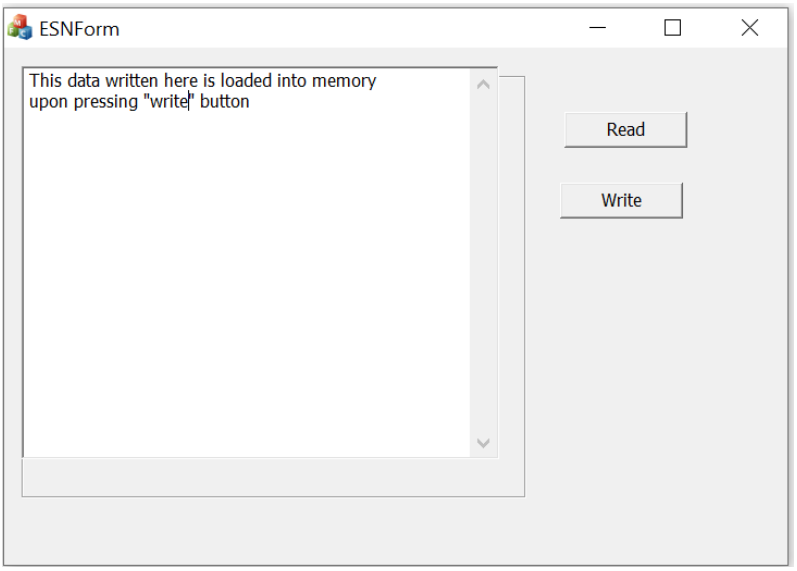

Figure 20: Windows application for testing

9 Implementation

9.1 EEPROM Vs Flash

The Table 2 below shows the reasons for choosing EEPROM as main element in external memory device. The memory devices are read upon train computer startup and requires faster data read process to ensure shorter startup time.

Table 2: Differences between EEPROM and Flash

EEPROM	**Flash**
Faster to read data.	Relatively slower to read data.
Byte-wise erasable.	Erasing takes place block by block.
It is usually used to store configuration data.	It is used when data is frequently re-written.

9.2 Memory Device Recognition and Access

The EEPROM used in this project has globally unique ROM ID that is 64 bits long as shown in Figure 21. The first 8 bits are the family code for 1-wire devices. It is the major differentiator for EEPROM type/size identification. Table 3 shows Device family code for both existing and its newer upgrade with higher size memory device type.

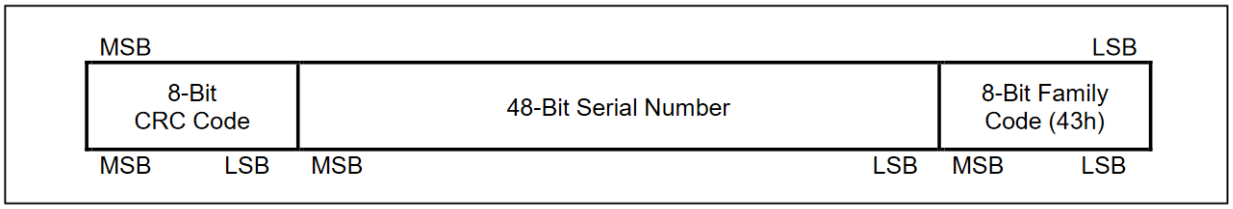

Figure 21: 64-Bit ROM ID with family code for DS28EC20 [16]

Table 3: Memory device Family code in ROM ID [22]

Device type	Device Family Code (hex)	Size (byte)
DS2433	0x23	512
DS28EC20	0x43	2560

9.3 Automation of build process

This project deals with C/C++ programming language as a language of firmware implementation and Windows GUI memory access tool for testing. During software

development, developers need to compile the code repository multiple times for test and verification purpose. Automating compilation process facilitates developer and organization for continuous delivery. Some examples of build tools are: Make, Apache ant, MS Build etc.

In this project the compilation process was dealt with *Make*. It is a build automation tool for tracking dependencies of program and libraries and compiling them. Make reads the set of directives from file called *Makefile* to produce targeted program.

9.4 Software Configuration Management

The necessity was evaluated for the requirement of higher memory sizes. The discussion of feature is done for the target definition. There are many collaboration tools for issue tracking or software change requirement management like, Jira, Basecamp, Asana etc.

The software repository that was updated in this project has online collaboration and contained other developers working simultaneously. Coordination is key factory in working environment of multiple developers. The firmware repository had revision control system to control the change to ensure integrity and traceability of the software life cycle. The version control system allows the user or group to manage the change. There are many types of version control system for e.g. Microsoft Visual Source Safe(VSS), Concurrent version system(CVS), Subversion(SVN), ClearCase, Source Forge, Git, Mercurial, Bazaar.

To merge the change, it must go through testing and change review process. The testing was done with relevant change and the change was forwarded for review process for concerned developers for approval. The code repository was managed for development flow. There are many software tools that can host multiple repository that uses version control system (e.g. Git or SVN). Some of them are: Gitbucket Gitlab, Bitbucket, Deveo, sourcehut, Gerrit etc. Such tools simplify the maintenance and management of multiple code repository.

10 Conclusion

The goal of the thesis project was to integrate reading of newer high capacity memory modules while maintaining legacy compatibility with older memory module. It was achieved by making changes in firmware of computer modules which facilitates reading of memory. The reading of memory during startup is normal process to initialize various configuration. The System on Chip (SoC) that runs Operating System is itself connected to various memory ICs internally on the same board for various purposes. The external connection of memory devices seems to be very helpful for cluster of modules connected to same network as seen in this thesis. Portable memory devices can be programmed independent of computer currently reading it, as during failure of that computer it is easily replaced with another computer to initialize the same configuration with the same memory module. This also provides redundancy to the system.

The workflow on integrating memory with higher capacity started with requirement definition and followed by making changes in memory programmer/writer for testing purpose. Firmware change came with many learning majorly about elements of Linux system. Testing and debugging took major chunk of time during this project. The new memory size of 20 Kilobit (DS24EC20) was introduced as an existing option to older memory 4 Kilobit (DS2433) in capacity. This change facilitates holding more data for configuration or identification, for many customer projects that are limited by size of memory, and many other projects to come in future.

References

1 IEC 61375-1:2012, Ed.3, Electronic railway equipment – Train communication network (TCN) – Part 1: General architecture, International Electrotechnical Commission (IEC)

2 IEC 61375-2-1, ED. 1, :2015, Electronic railway equipment – Train communication network (TCN) – Part 2-1: Wire Train Bus (WTB), International Electrotechnical Commission (IEC)

3 IEC 61375-2-5, ED. 1, :2014, Electronic railway equipment – Train communication network (TCN) – Part 2-5: Ethernet Train Backbone (ETB), International Electrotechnical Commission (IEC)

4 IEC 61375-2-3:2015, Electronic railway equipment – Train communication network (TCN), page 31, International Electrotechnical Commission (IEC)

5 S. Tanenbaum, Wetherall. Ed. 5, 2014, chapter 1, page 30, *COMPUTER NETWORKS,* PRENTICE HALL, [Accessed 16 May 2020].

6 S. Tanenbaum, Wetherall. Ed. 5, 2014, chapter 6, page 557, *COMPUTER NETWORKS,* PRENTICE HALL, [Accessed 11 May 2020].

7 S. Tanenbaum, Wetherall. Ed. 5, 2014, chapter 6, page 542, *COMPUTER NETWORKS,* PRENTICE HALL, [Accessed 11 May 2020].

8 S. Tanenbaum, Wetherall. Ed. 5, 2014, chapter 5, page 439, *COMPUTER NETWORKS,* PRENTICE HALL, [Accessed 11 May 2020].

9 Semiconductor Memory Types & Technologies, Electronics Notes, [online] Available at: <https://www.electronics-notes.com/articles/electronic_components/semiconductor-ic-memory/memory-types-technologies.php> [Accessed 1 Feb 2020].

10 Gutmann, Peter 2015, *Data Remanence in Semiconductor Devices*. [online] IBM T.J Watson Research Center. Available at: < https://www.usenix.org/legacy/publications/library/proceedings/sec01/full_papers/gutmann/gutmann_html/index.html> [Accessed 5 May 2020].

11 E. Frenzel, L., 2015. *Handbook of Serial Communications Interfaces*. [online] O'Reilly Online Learning. Available at: < https://learning.oreilly.com/library/view/handbook-of-serial/9780128006719/xhtml/chp035.xhtml#chp035titl > [Accessed 5 May 2020].

12 Michael Margolis, 2011. *Arduino Cookbook*. [online] O'Reilly Online Learning. Available at: < https://learning.oreilly.com/library/view/arduino-cookbook/9781449399368/ch13.html> [Accessed 7 May 2020].

13 E. Frenzel, L., 2015. *Handbook of Serial Communications Interfaces*. [online] O'Reilly Online Learning. Available at: <https://learning.oreilly.com/library/view/handbook-of-serial/9780128006719/xhtml/chp013.xhtml#chp013titl> [Accessed 5 May 2020].

14 APPLICATION NOTE 74, READING AND WRITING 1-WIRE® DEVICES THROUGH SERIAL INTERFACES, Maxim Integrated, [online] Available at: < https://www.maximintegrated.com/en/design/technical-documents/app-notes/7/74.html > [Accessed 8 Jan 2020].

15 APPLICATION NOTE 126, 1-WIRE COMMUNICATION THROUGH SOFTWARE, Maxim Integrated, [online] Available at: < https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/126.html> [Accessed 8 Jan 2020].

16 DS28EC20, Datasheet, Maxim Integrated, [online] Available at: < https://datasheets.maximintegrated.com/en/ds/DS28EC20.pdf> [Accessed 8 Jan 2020].

17 EKE-Electronics Ltd. Brochure. Available at: < https://www.eke-electronics.com/images/eke/Brochure/EKE_Trainnet_Brochure.pdf> [Accessed 5 May 2020].

18 Fox Richard, 2014.*Linux with Operating System Concepts*. [online] O'Reilly Online Learning. Available at: < https://learning.oreilly.com/library/view/linux-with-operating/9781482235890/K23087_C001.xhtml#_idParaDest-13> [Accessed 8 May 2020].

19 Sally Gene, 2009.*Pro Linux Embedded System*. [online] O'Reilly Online Learning. Available at: < https://learning.oreilly.com/library/view/pro-linux-embedded/9781430272274/ch01.html> [Accessed 5 May 2020].

20 Corbert Jonathan,K.Hartman Greg, Rubini Alessandro,2009.*Linux Device Drivers,3rd Edition*. [online] O'Reilly Online Learning. Available at: < https://learning.oreilly.com/library/view/linux-device-drivers/0596005903/ch01.html> [Accessed 5 May 2020].

21 VIU2336_1_00.pdf/ ELE-Electronics Ltd /manual

22 APPLICATION NOTE 155, 1-WIRE SOFTWARE RESOURCE GUIDE DEVICE DESCRIPTION, Maxim Integrated, [online] Available at: < https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/155.html > [Accessed 8 Jan 2020]