

**DESIGN AND INVESTIGATION OF A NEW PROTOCOL FOR A SINGLE WIRE LOW
SPEED BI-DIRECTIONAL DATA BUS FOR EMBEDDED SYSTEMS**

BY

Hasib Rahman
181-31-253

This Report Presented in Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Electronics and Telecommunication Engineering

Supervised By

Md. Taslim Arefin
Associate Professor & Head
Department of ETE
Daffodil International University



DAFFODIL INTERNATIONAL UNIVERSITY

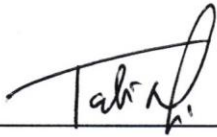
DHAKA, BANGLADESH

OCTOBAR 2019

APPROVAL

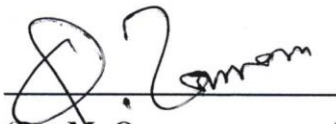
This Project titled “**Design and investigation of a new protocol for a single wire low speed bi-directional data bus for embedded systems**”, submitted by Hasib Rahman to the Department of Electronics and Telecommunication Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of M.Sc. in Electronics and Telecommunication Engineering and approved as to its style and contents. The presentation was held on 27 October, 2019.

BOARD OF EXAMINERS



(Md. Taslim Arefin)
Associate Professor & Head
Department of ETE
Faculty of Engineering
Daffodil International University

Chairman



(Dr. M. Quamruzzaman)
Professor
Department of ETE
Faculty of Engineering
Daffodil International University

Internal Examiner



(Dr. Md. Fayzur Rahman)
Professor & Chairperson
Department of EEE
Green University

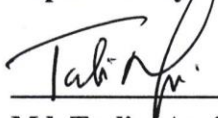
External Examiner

DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Md. Taslim Arefin, Associate Professor & Head, Department of ETE**, Daffodil International University.

We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:



Md. Taslim Arefin
Associate Professor & Head
Department of ETE
Daffodil International University

Submitted by:



Hasib Rahman
ID: 181-31-253
Department of ETE
Daffodil International University

ACKNOWLEDGEMENT

The very first words of acknowledgement deserve to be directed toward my thesis supervisor, Md. Taslim Arefin (Associate Professor & ETE dept. Head) for his valuable advice, guidance, constructive suggestions and making time from his busy schedule to advice and oversee my research work. I would like to offer him my profound indebtedness and sincere thanks along with my heartfelt gratitude.

I would like to thank and offer respect to all the authors of the application notes, research papers, journals, books and literature that were used throughout this research. The aforementioned materials not only helped my research but also broadened my reservoir of knowledge as well as my intellectual capacity.

And last but absolutely not least, I would like to offer a strong sense of gratitude to this intensely powerful supercomputer that I was able to use throughout, not only this research, but literally my whole life. Honestly, without it, it would have been literally impossible to complete this research, to achieve my M.Sc. in ETE, my B.Sc. in EEE, to learn, understand and apply science in my daily life and the list goes on. Therefore, a fairly large piece of this acknowledgement note is reserved for this approx. 1400 grams of various types of biological cells that comes together to form this supercomputer which is physically suspended in cerebrospinal fluid and physically resides inside my head.

ABSTRACT

A new protocol for a low speed single wire half duplex serial digital data BUS utilizing Master-Slave configuration is proposed in this thesis. Industry standard single wire data BUS already exists in the market from multiple large name brands. These existing single wire protocols work by either using delay based digital logic state definition or Manchester encoding or some other encoding technique to multiplex both data and clock in the same transmitted bit. Existing data buses generally have a standard speed of around 15Kbps and around 100kbps at high speed or overdrive mode. The data is kept synchronous by syncing with falling or rising edge of the transmitted signal.

The proposed data BUS tries to multiplex data and clock in an unconventional way. An existing standard for digital logic state definition has been slightly modified to suit the requirements. In the research, attempts are made to combine both a digital data and an analog Clock signal on the transmitter side and separate the clock and digital data on the receiver side. The hardware for the combining and separating signals are designed to be as simple as possible. The whole system is tested out on a breadboard prototype using off the shelf components. Prototype consists of the individual transmitting and receiving modules and their respective Microcontroller Unit(s) running respective behavior model programs to simulate a functional transmitter and receiver, that are communicating predefined data at real time. Prior to construction of the prototype, detailed computer simulation has been generated to observe to viability of the idea, to test the designed circuits and to find and fix the bugs and signal glitches. Computer simulation is kept limited between voltage sweep and interactive simulation depending on the circuit and the virtual test instruments made available by the said simulator.

Although signal degradation occurred in the prototype, which was not present in the computer simulation, the BUS behaved exactly as predicted. Results from both the simulations and hardware prototype shows that it is possible to communicate in the proposed manner. The result from the proposed Single Wire data BUS is compared to the existing protocols to reveal that it is possible to theoretically exceed (21Kbps to 27Kbps) the standard speed of the existing protocols even with the breadboard implementation of the proposed data BUS.

TABLE OF CONTENT

CONTENTS	Page No
Board of examiners	ii
Declaration	iii
Acknowledgements	iv
Abstract	v
Table of Content	vi – vii
List of Figures	viii – xi
List of tables	xii
 CHAPTER	
Chapter : 1 Introduction	1
1.1 Objective of this thesis work	2
1.2 Possible Outcome	2
1.3 Organization of the thesis	3
 Chapter : 2 Digital data Communication BUS	 5
2.1 Parallel data BUS	5
2.2 Serial Data BUS	6
2.3 Comparison between Serial and Parallel BUS	7
2.4 Effects of noise and attenuation	8
 Chapter : 3 Common protocols for 8-bit serial communication	 11
3.1 I2C/IIC	11
3.2 Working principle of IIC	12
3.3 Serial Protocol (EIA/TIA 232)	14
3.4 Requirement for single wire BUS	16
 Chapter : 4 Existing single wire BUS protocols	 18
4.1 1-Wire Protocol	18
4.2 UNI/O	23

Chapter : 5	Proposed Single Wire BUS	26
5.1	Working principle	26
5.2	Data frame format, BUS modes and communication protocol	29
5.2.1	Data frame format	30
5.2.2	BUS modes	30
5.2.3	Communication protocol	31
5.3	M-Ary modulation and buffering	32
5.4	System and sub-system block diagrams	33
5.4.1	Multiplexer	34
5.4.2	Timer	34
5.4.3	Isolator	36
5.4.4	Error handler	38
5.5	Complete protocol	40
Chapter : 6	Hardware Design	41
6.1	Signal Multiplexer	41
6.2	Multiplexer working principle	42
6.3	Isolator	46
6.4	Isolator working principle	48
6.5	Test Setup of complete module	51
6.6	Transmitter and receiver behavior model	57
6.6.1	Transmitter	57
6.6.2	Receiver	59
Chapter : 7	Experimental Hardware Emulation and Results	61
7.1	Experimental Setup result	63
Chapter : 8	Conclusion	71
8.1	Thesis Summary	72
8.2	Present problems solved by the proposed data BUS	73
8.3	Probable future development(s)	74
References		76
Appendix		78
A	List of abbreviations	78
B	List of Designators used in the Schematic	79
C	Schematic for hardware emulation	80
D	Emulation hardware setup	81

LIST OF FIGURES

FIGURES

- Figure 3.1 IIC timing diagram for sending Address followed by data
- Figure 3.2 Devices and pull-up resistor connection of a typical IIC BUS
- Figure 3.3 Single ended transmission with line noise and ground noise model
- Figure 3.4 Differential transmission with line noise and ground noise model
- Figure 3.5 Data frame format of TIA/EIA 232 standard
- Figure 4.1 Typical device connections of 1-wire protocol
- Figure 4.2 Typical communication flow of 1-wire protocol
- Figure 4.3 1-wire protocol reset with at least 1 slave
- Figure 4.4 1-wire reset with no slave
- Figure 4.5 1-wire Write 0 (digital LOW)
- Figure 4.6 1-wire Write 1 (digital HIGH)
- Figure 4.7 1-wire Read 1 (digital HIGH)
- Figure 4.8 1-wire Read 0 (digital LOW)
- Figure 4.9 UNI/O BUS example
- Figure 4.10 Manchester encoding
- Figure 4.11 UNI/O one-byte transmission using Manchester Encoding
- Figure 4.12 UNI/O complete one-byte transmission with both Master and Slave acknowledgement
- Figure 4.13 Complete timing diagram for read operation from UNI/O compatible EEPROM
- Figure 5.1 Figure 5.1 Voltage Levels for the logic states
(a) Conventional 5v TTL standard, (b) Modified standard for the proposed single wire BUS

Figure 5.2	Digital signal waveform for conventional TTL standard
Figure 5.3	Digital signal waveform for proposed single wire standard
Figure 5.4	Data frame for the proposed single wire standard
Figure 5.5	M-ary modulation of binary data
Figure 5.6	Block diagram of data and clock multiplexing circuit
Figure 5.7	Block diagram of clock isolating circuit (isolator)
Figure 5.8	Figure 5.7 Voltage transition (a) Digital signal, (b) Actual continuous representation of voltage transition
Figure 5.9	Block diagram of clock detection circuit (subsystem of isolator)
Figure 5.10	Block diagram of combined multiplexer and demultiplexer to form a single module
Figure 5.11	Block diagram of BUS latch up prevention circuit (error handler)
Figure 5.12	Complete block diagram of combined multiplexer and demultiplexer to form a single module
Figure 6.1	Resistive voltage divider (a) Clock voltage level generator, (b) Data voltage level generator
Figure 6.2	Clock generating circuit
Figure 6.3	Data generating circuit
Figure 6.4	Complete multiplexing circuit
Figure 6.5	Timing diagram and output of the multiplexing circuit From bottom to top: CLK (blue), Data (green), Multiplexed signal (red)
Figure 6.6	Complete isolator circuit
Figure 6.7	DC sweep of the isolator
Figure 6.8	DC sweep of low threshold comparator
Figure 6.9	DC sweep of high threshold comparator

Figure 6.10	DC sweep of comparator XOR gate
Figure 6.11	DC sweep of all components of the isolator
Figure 6.12	Complete circuit combining TX RX module
Figure 6.13	Waveform of the combined module From bottom to top: CLK, Data, Multiplexed signal and extracted data from multiplexed signal
Figure 6.14	Waveform of the isolator output From bottom to top: Output of high and low Comparator, XOR gate U1A and extracted CLK from multiplexed signal
Figure 6.15	Glitch in the isolator due to slow slew rate of the op-amps
Figure 6.16	Updated circuit with dedicated comparator op-amps
Figure 6.17	Waveform of the updated circuit From bottom to top: Output of high and low Comparator, XOR gate U1A and extracted CLK from multiplexed signal
Figure 6.18	Transmitter behavioral model flow diagram
Figure 6.19	Receiver behavioral model flow diagram
Figure 7.1	Initialization of setup mode followed by transmission of slave address and one data packet (BUS & CLK, blue and yellow respectively)
Figure 7.2	Initialization of setup mode followed by transmission of slave address and one data packet (BUS & data, blue and yellow respectively)
Figure 7.3	Transmission of address and header (BUS & CLK, blue and yellow respectively)
Figure 7.4	Transmission of data and header (BUS & CLK, blue and yellow respectively)
Figure 7.5	Transmission of data and header (BUS & data, blue and yellow respectively)
Figure 7.6	Transmission of data and header (BUS & data, blue and yellow respectively)
Figure 7.7	Transistor switching pattern for initialization of setup mode followed by transmission of slave address and one data packet (data & CLK, yellow and blue respectively)

- Figure 7.8 Transistor switching pattern for transmission of slave address and command header (data & CLK, yellow and blue respectively)
- Figure 7.9 Transistor switching pattern for transmission of data and command header (data & CLK, yellow and blue respectively)
- Figure 8.1 Signal level transition

LIST OF TABLES

Table 2.1	Pinout of Parallel-ATA (PATA) BUS
Table 2.2	Pinout of Serial-ATA (SATA) BUS
Table 2.3	Comparison between Serial and Parallel BUS
Table 3.1	comparison between IIC and TIA/EIA 232
Table 5.1	Command headers and their respective mode descriptions
Table 6.1	Component list for multiplexing circuit
Table 6.2	Component list for isolator circuit
Table 6.3	Component list for complete module circuit
Table 7.1	Component list for protoboard hardware emulator
Table 7.2	List of instruments, simulation package and CAD packages used in the research
Table 7.3	Comparison between existing industry standard single wire BUS against proposed single wire BUS

Chapter – 1

Introduction

Modern civilization is heavily dependent on electronic technologies. With improvement of electronic technologies, devices are becoming more capable, efficient, smarter and smaller. An individual standalone electronic system is now capable of working in conjunction with other individual standalone system(s) to offer a better and more efficient way of solving any particular task no matter the complexity. To work in conjunction with one another, the devices/systems need to communicate between each other. In order for the standalone systems to communicate between each other, a common communication medium and a common communication protocol is necessary. There exist multiple different kinds of communication protocols each with its own characteristics, advantages, disadvantages, operating conditions and limitations. Communication between standalone systems can be achieved via wired, wireless or optical medium. This thesis paper focuses on communication BUS and protocols used by embedded systems using wired medium.

In the thesis paper, a new single wire digital data BUS is proposed. The data BUS is implemented using a new/experimental signaling protocol that multiplexes digital data and clock signal into a single signal. The signaling protocol can be thought of as a cross between the traditional 5v TTL standard and an analog pulsating dc signal. In the traditional 5v TTL standard, the threshold for logical HIGH is set for voltages that are greater than or equal to 2.4v. The logical LOW is set for less than or equal to 0.4v. The middle section between 0.4v to 2.4v is left unused. Any electrical pulse in this region is regarded by a digital logic circuit as undefined (metastable state) and is discarded. In this thesis, circuits and protocols are generated that allow usage of the undefined region to carry the clock signal without disturbing/affecting the digital data. Circuits are designed to combine both the digital data and digital clock signal, transmit them and at the receiving end, receive and separate them to retrieve the sent digital data.

There are multiple commercially available single wire BUS in the market, however, none of them uses this specific signaling method and protocol researched in this thesis work. The proposed BUS uses single ended signaling to carry data and CLK.

1.1 Objective of this thesis work

The objective of this thesis work is to

- Research a new way to communicate data between multiple chip/devices/systems using only single physical wire utilizing the proposed signaling protocol
- Design and simulate the hardware for the BUS on a PC based simulator
- Build a hardware prototype to verify the designed BUS in real time in real life scenario
- Compare and contrast to existing technology(s)

1.2 Possible Outcome

The main section of the proposed single wire data BUS is its ability to multiplex digital data and clock signal. The following outcomes are expected

- The multiplexing/encoding circuit will generate a completely new signal waveform with provided clock and data
- The demultiplexing/decoding circuit will extract both the clock and data signal from the newly generated signal, transmitted through a single physical wire
- The protocol is expected to have a lossless exchange of synchronous data at bit rate of at least 10 Kbits/s (theoretically)

1.3 Organization of the thesis

This thesis work deals with serial data BUS, and therefore, a brief overview of serial and parallel data BUS is provided to justify the popularity of serial BUS. The proposed BUS is intended to be used with embedded systems. To justify the necessity for a single wire BUS, some very popular multi wire data BUS are discussed briefly. The result of the proposed single wire digital data BUS is compared to existing single wire BUS later in the thesis report and therefore, existing single wire data BUS is also explained briefly. The thesis report is organized as follows

Chapter 1 – The introductory chapter that set the aim and motivation behind this thesis research. It also defines the objective and predicts the outcome of the research.

Chapter 2 – Provides a brief overview of two different ways digital data is exchanged via wired medium, compares them from different practical aspects and points out why serial protocols are preferred over parallel protocols in modern embedded systems.

Chapter 3 – Briefly describes the working principle and limitations of two of the most widely used serial data BUS for embedded systems and tries to justify the requirement for single wire data BUS.

Chapter 4 – Gives a light overview of the existing industry standard single wire data BUS, describes their working principle, applications, advantages, disadvantages and limitations. This chapter is presented here so that it can be later referred to when comparing the proposed single wire data BUS to the existing ones.

Chapter 5 – Describes the proposed single wire data BUS, its working principle, block diagram and the functionality of the blocks.

Chapter 6 – Deals with the design and working principle of the circuits to multiplex and demultiplex the digital data and clock signal, show and discuss the results of the simulation

waveforms and discuss the behavior models for a dummy transmitter and receiver to communicate data using the proposed single wire BUS in real time.

Chapter 7 – Presents and discusses the results of bread board prototype.

Chapter 8 – Concluding remark of this thesis work along with summary and probable research topics to update this proposed data BUS in future.

Reference – Chapter 8 is followed by reference containing detail information of all the documents, books and literatures used in the research and in this thesis paper.

Appendix – Consists of two sections namely A and B. Section A contains list of abbreviations used throughout this literature and section B contains list of designators used in the schematic of the actual electronic circuit generated by CAD tool.

Chapter – 2

Digital Data Communication BUS

Any digital system can transfer data either serially or in parallel using serial communication or parallel communication method respectively. In both case for higher degree of accuracy and efficiency, data synchronization signal is needed. The synchronization signal is generally referred to as Clock signal and the frequency of the Clock signal is termed as Clock speed. Higher the Clock speed of the data BUS, faster is the data transmission rate. Note that this thesis is inclined toward microcontroller based systems and therefore only the BUS that are commonly used by 8-bit MCU are considered here. A very brief overview of both parallel and serial communication is provided in section 2.1 and section 2.2.

2.1 Parallel data BUS

Parallel data BUS uses multiple data line to communicate data. Ideally, the number of parallel data lines equals number of bits on the data e.g. a simple ideal 8-bit parallel data BUS will have 8 data lines. In reality the number of parallel data lines will be higher as Clock signal, various control signals, status and/or acknowledgement signals are also communicated through the BUS. To transmit data in simplest form (without control, status and acknowledgment signals) through an 8-bit parallel BUS, the 8-bit data is placed on 8-bit data BUS (1 bit per line making sure to place LSB and MSB in their respective position on the BUS) and the Clock signal is transmitted by the transmitter. Once the receiver receives the Clock signal, it takes in the bits that are in the parallel data lines. The process is repeated each time data needs to be transferred. Parallel data BUS therefore, physically, consists of a large number individual wires on the data BUS and requires a large number of physical pins on the IC/system handling the data, both on transmitter and receiver, that is using the said BUS.

Table 2.1 shows the pinout of Parallel ATA (PATA) BUS conventionally used by computers to connect motherboard to external devices such as hard disk, floppy drive, CD-ROM drive etc. A 40

pin connector, connected to 40 pin ribbon cable, would be used to physically connect the 40 pin BUS to the motherboard and other devices. Note that of the 40 pin connector, only 16 pins are used to convey data. Rest are different control signals address, chip select, ACK etc.

Signal	Pin number	Pin number	Signal
GND	2	1	RST
DD8	4	3	DD7
DD9	6	5	DD6
DD10	8	7	DD5
DD11	10	9	DD4
DD12	12	11	DD3
DD13	14	13	DD2
DD14	16	15	DD1
DD15	18	17	DD0
VCC in	20	19	GND
GND	22	21	DDRQ
GND	24	23	I/O W
GND	26	25	I/O R
CS	28	27	IOCHRDY
GND	30	29	DDACK
N/C	32	31	IRQ
DMA66 detect	34	33	ADDR 1
ADDR 2	36	35	ADDR 0
Chip select 3P	38	37	Chip select 1P
GND	40	39	Activity

Table 2.1 Pinout of Parallel-ATA (PATA) BUS

2.2 Serial Data BUS

Serial data BUS uses much lower number of wires to communicate data. Common serial data BUS like IIC or SPI uses one or two wire to send/receive data and a single dedicated wire for transmitting Clock signal. In a simple ideal 8-bit serial data BUS, the 8-bit data is first placed serially on the data wire/line and then Clock signal is applied. Each CLK sends one-bit serial data. Therefore, for an 8bit data, 8 CLK are needed to completely transmit it. However unlike parallel BUS, where the number of wires can increase depending on control, status and acknowledge signals, the number of wires on serial BUS can stay same. The control, status and acknowledge

bits can be serially sent using the same wires that are used to send data, however, more CLK are then needed. E.g. if there are 1 control bit, 1 status bit and 1 acknowledge bit on an 8bit serial data BUS, a total of $8+3 = 11$ CLK are needed to transmit the whole data. Therefore, serial data BUS requires significantly less number of physical pins on the IC/system handling the data, both on transmitter and receiver, that is using the said BUS. This makes serial BUS very popular in digital systems.

Table 2.2 shows the pinout of a Serial ATA (SATA) data BUS. Note the reduction of pin numbers from 40 pin to 7 pins. Unlike PATA, SATA uses differential pairs to convey signals, one pair for the receiver (B+ & B-) and another pair for the transmitter (A+ & A-).

Pin number	Signal
1	Ground
2	A+ (transmit)
3	A- (transmit)
4	Ground
5	B- (receive)
6	B+ (receive)
7	Ground
8	Coding notch (nonconductive, built on the connectors)

Table 2.2 Pinout of Serial-ATA (SATA) BUS

2.3 Comparison between Serial and Parallel BUS

In parallel data BUS, a lot of physical wires are used to transfer one or more bytes at a time. A lot of data can be transferred in a very short time in this way. A serial data BUS requires significantly lower number of physical wires, however, requires a significantly higher number of Clock signals. This is a tradeoff between physical wires vs communication speed. In the early days of computer, the technology to generate, control high clock speed and handle high data transmission was not available (even if the technology existed, it was too expensive and was financially impractical) and so parallel BUS was very popular e.g. Intel i51, i85, i86 and Pentium series processors used 8/16/32/64-bit parallel BUS for data and address. As technology improved, high speed data handling capability has also been developed (and fabrication price is reduced significantly) resulting in serial data BUS that are orders of magnitude faster than conventional

parallel data BUS. It is true that using the same technology, parallel systems can also be made faster, however, parallel BUS requires a large number of physical connections which are not practical by today's standard, at least in case of system design (VLSI design frequently uses combination of serial and parallel BUS). Furthermore, effect of noise and signal attenuation must also be considered which are explained in the next section.

2.4 Effects of noise and attenuation

Every piece of current carrying wire, by nature, has properties to acts like an antenna. In principle, an antenna can both receive and transmit signals. Therefore, every physical wires, carrying digital/analog signal, will always act like an antenna and will both radiate and receive signals in the appropriate range of the EM spectrum.

Fourier analysis of radiating signals will point out that the emitted/radiated frequency includes the fundamental frequency (frequency of original signal carried by the conductor) and several harmonics of the fundamental frequency. All of these emitted signals are not an intended part of the original design and are therefore considered to be noise. Furthermore, signals emitted by one wire can be picked up by another wire carrying different signal as all wires can act like antenna. These picked up stray signals or noise can interact with the original signals and, in worst case, may cause change in logic state of the original signal. The noise can get out of the actual system that generates it and spread out on the surrounding environment. If the system is generating the noise, the noise is labeled as Electromagnetic Interference (EMI). If the system is picking up the noise, the noise is known as Electromagnetic Crosstalk (EMC). EMI/EMC can destabilize sensitive digital systems (e.g. high speed telecom or RF instrument, computers, audio-video systems etc.), make systems such as analog acquisition circuits to acquire wrong data and may cause false state switching of CMOS circuit(s). Any form of electromagnetic radiation can also affect biological and ecological systems (affect depends on frequency and power of the signal). Electromagnetic radiation (radiated emission) can also escape its source via power line (conducted emission) and may destabilize electronic system (causing fire in worst case) connected to the same power BUS. This is the reason why developed countries has a strong set of guidelines for allowed EMI, RFI level for all the electronic products that enters the said country legally, for example, USA has FCC

standard, EU has CE standard, Japan has VCCI and all developed countries, will most probably, accept UL (Universal Listing) standard(s). There are separate guidelines to deal with EMI and EMC for industries such as telecom equipment manufacturing or automotive industry ^[25] ^[26]. All properly designed and tested electronic products (commercial or industrial) are tested for EMI/EMC by professional compliance test facility like UL prior to release in the large scale mass market ^[21] ^[22].

On a different note, every signals will face attenuation no matter the medium they are passing through. In case of electronics, the medium is generally copper which is a very good conductor of electricity. Even though copper is a very good conductor of electricity, it has certain amount of resistivity i.e. resistance. The resistance will cause the signal to get attenuated. Given that the signal travels a distance long enough, it could be attenuated such that the digital HIGH of 2.4v (considering 5v TTL ^[9]) may drop to unaccepted (metastable) region which will most definitely cause signal error in the receiver. Attenuation is applicable for both signal and power distribution. This is the reason why all professional PCB CAD packages has a dedicated Power Integrity Analysis and Signal Integrity Analysis e.g. leading IC design package from Cadence has a dedicated tool for Signal and Power integrity analysis called 'Sigrity'.

By combining all of the above stated real life impeding conditions and applying it to Parallel Data BUS, from engineering perspective, it can be observed that if transmission distance is too long there is a possibility that the data will be attenuated too much and/or distorted by noise or EMI/EMC. From financial point of view, carrying a large number of wires over extended distance is costly and wasteful. In such situations it is convenient to transfer data in series. Serial transmission(s) requires fewer number of wires compared to parallel communication and data can be transmitted across long distance using proper transmission protocols like differential pairing. This makes serial transmission cheaper and more convenient from engineering point of view and economical from financial point of view.

A very brief comparison of Serial versus Parallel communication BUS is depicted on table 2.1.

<u>Attributes</u>	<u>Name of data BUS</u>	
	Serial	Parallel
Number of Physical wires	1 wire can carry multiple bits and so, the number of wires can be drastically reduced	Multiple wires, 1 wire per bit data
EMI generation	Higher than Parallel BUS (as operating frequency is higher)	Relatively lower than Serial BUS
EMI reduction	EMI can be suppressed using differential signal and using properly terminated shielded cables	Wires can be shielded to decrease EMI emission
Noise susceptibility	High if proper shielded cable and/or differential signaling is not used	Wires can be shielded to increase noise susceptibility
Bit rate	Clock has to be much faster to be same rate as parallel counterpart	Single clock can transmit multiple bits, therefore orders of magnitude faster than serial counterpart
Latency	High	Significantly lower than Serial counterpart
Range	High, in the range of Km using differential signaling	Low, large distance parallel cable is possible but impractical
Cost	Much lower than parallel, only practical way to carry data through wired medium	Impractical, incurs high cost in cable, maintenance and troubleshooting

Table 2.2 Comparison between Serial and Parallel BUS

Chapter - 3

Common protocols for 8-bit serial communication

This section briefly explains the working principle and usage of two most commonly used serial data communication protocols. Both of these protocols are industry standard and are used in numerous industrial, commercial and consumer electronics. Please note that only the serial communication protocols that uses up to two physical wires for data exchange are discussed in this chapter. The brief explanation of these serial communication protocols will later be used to justify the need for a single wire communication BUS.

3.1 I2C/IIC ^[17]

Invented in the late 1980's by Philips Semiconductor (now NXP), IIC is an 8-bit data communication BUS that allows multiple embedded systems, SOC or MCU to communicate between each other. IIC is the original name that was given by Philips, however, other corporations such as Atmel (now Microchip Technology Inc.) uses the license from NXP to use IIC with their own products while renaming IIC to TWI ^[19] (TWI is specific branding of Microchip Technology Inc.). IIC (also known as I2C and I²C) stands for Inter Integrated Circuit. Two Wire Interface (TWI) is the name given by Atmel/Microchip to avoid violating NXP's trademark but apart from name both are same in all aspects. IIC is a half-duplex, low speed, synchronous data BUS. Presently, this protocol for communication is widely used by system engineers and hardware developers to glue down several embedded systems together. Its wide use in industry has forced a common industry standard to be created. IIC is used in modern systems like voltage regulator and Clock generator chip on computer, displays on lab instruments, HDMI port for monitors and on active probes for oscilloscope etc.

The basic advantage of IIC is that it requires only two wires for bidirectional transmission namely SCL & SDA ^[17]. SCL is system clock and SDA is system data. As a system clock is involved, it is a synchronous system. Two wire IIC works great except for its limitation that the capacitance between SCL & SDA, relative to ground, must be less than 400pf (may depend on the chip

generating CLK). Therefore, IIC is not suitable for long distance communication through wire (buffers are available to increase range to some extent). The clock frequency is user defined. Higher the clock frequency, the greater is the capacitance between SCL & SDA and with respect to ground. Thus a middle ground between range and speed needs to be established during the BUS implementation.

3.2 Working principle of IIC ^[13] ^[17] ^[19]

All IIC compatible devices must be connected in parallel to each other as daisy chain. The SCL and SDA of all compatible devices must be connected to the SCL and SDA of the BUS respectively. The IIC works in Master Slave fashion. At any time, there can be several Slaves but only one Master. Each Slave is assigned a unique address. As IIC is 8-bit data BUS and has 7-bit address, maximum 127 devices (address of 0x00 is reserved as General call) can be connected to it at a time. The Master device is responsible for generating the clock signal and all of the control signals. When the Master device needs to communicate with any of the Slave device, it generates START condition, places the 7-bit/10-bit (depends on manufactures) slave address along with 1-bit read/write bit. If the Slave returns ACK, the Master reads or writes from/to the Slave. If MASTER doesn't need to use the BUS anymore, it generates STOP condition. If the Master needs to use the BUS but don't want to release the BUS, it generates REPEATED START condition. START & STOP can be generated only once but REPEATED START can be generated as many times as needed.

Sometimes there may be multiple Master in the system. At any time only one Master can control the BUS. During that time all other devices are treated as Slave. If multiple Master tries to take control of the BUS at the same time, the condition is called ARBRITRATION. If ARBRITRATION arises, the BUS becomes first come first serve (procedure must be present in algorithm to handle ARBRITRATION). The Master that reached first takes control of the BUS and the other Master(s) revert back to Slave status and wait for the BUS to be free up.

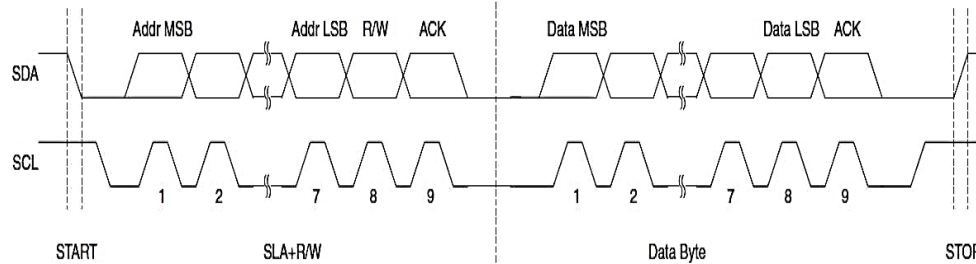


Figure 3.1 IIC timing diagram for sending Address followed by data ^[19]

IIC uses the classic open drain configuration. Therefore, in order to use IIC, both the SCL and SDA pins of the MCU must be physically pulled up by the user. Pulling up can be done via single resistor on each IC pins. The value of the pull up resistor depends the BUS speed and BUS length. This is to be done as IIC can only pull down and can't pull up as IIC has an internal open drain connection. Therefore, any break in the pull up circuit will make the whole system unstable as IIC is used as the central data BUS. Only one pull up circuit is enough for a whole system.

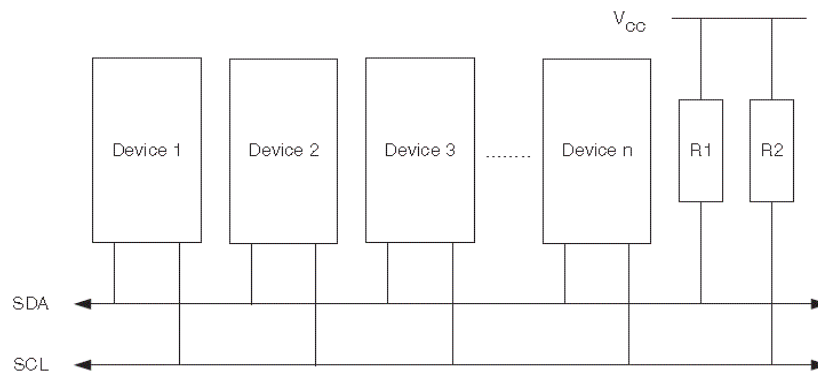


Figure 3.2 Devices and pull-up resistor connection of a typical IIC BUS ^[19]

Pulling up is conventionally done on the slave side. The resistors used for pulling up are determined by the speed of the BUS. The resistance can range from 1K to 10k. The theoretical value of resistance can be calculated from the specification formula. A good rule of thumb is to use 4.7 K Ω . It should be noted that both resistors must be of same value and with very tight tolerance and temperature coefficient.

Complete IIC user manual is presented in the reference [17].

3.3 Serial Protocol (EIA/TIA 232) ^{[1] [4] [7]}

The most commonly used serial I/O standard for computers and electronic instrumentations is TIA/EIA-232 which is most commonly known as RS232. Serial protocol for MCU can be either USART or UART. USART stands for Universal Synchronous Asynchronous Receiver Transmitter. UART stands for Universal Asynchronous Receiver Transmitter. USART means data can be transferred both synchronously and asynchronously while UART is asynchronous communication only. The data is received by another device with RS232 standard serial port. In synchronous mode, one character or one 8-bit block of data is send at a time. In asynchronous mode, a whole byte is send. If synchronous mode is used, another physical wire must be used to send Clock signal making total number of wires three.

Serial protocol has three different standards namely RS232, RS422 and RS485. These standards were set by Electronics Industries Association (EIA) to maintain communication compatibility among devices made by different manufactures. It is today most widely used I/O interfacing standard used in PC, industrial and lab equipment and in other electronics where serial communication is needed. The USB port that we frequently use on computer is actually a RS232 port with a USB to Serial converter.

It should be noted that these standards were set in the 1960s and updated for newer technology on the 1980s. Digital technology was still at its infancy on 1960s. That was the era of analog electronics. These standards were set to entertain the 1960s standards and requirements. Only RS232 is used on MCU as RS422 and RS485 requires differential signaling leaving only RS232 as single ended. Figure 3.3 and 4.3 shows both single ended and differential signaling respectively.

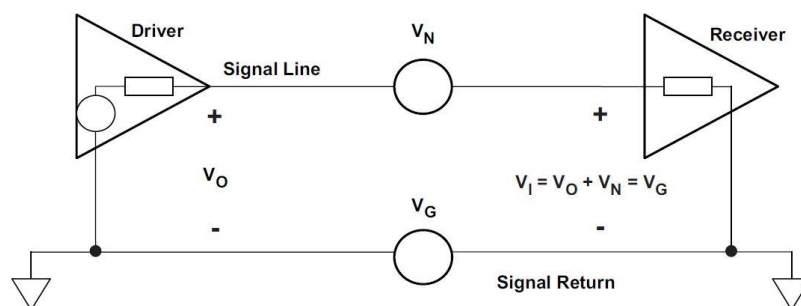


Figure 3.3 Single ended transmission with line noise and ground noise model ^[1]

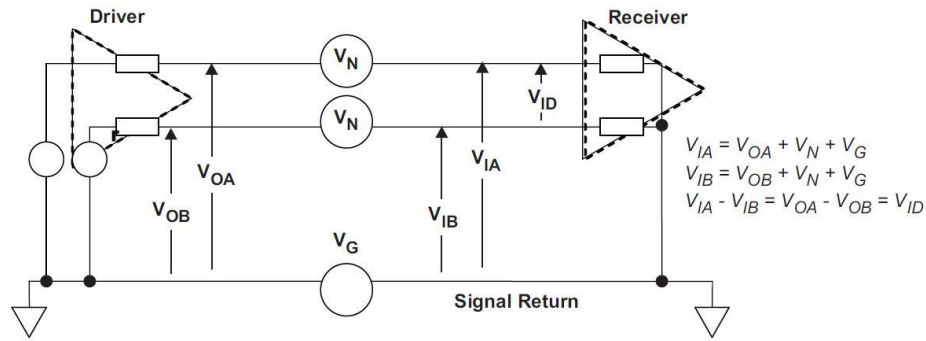


Figure 3.4 Differential transmission with line noise and ground noise model ^[1]

Differential pairs are capable of conveying signals a much longer distance than its single ended counterpart. A differential pair transmits the same signal via two closely placed parallel physical wires. One of the wires carries the actual signal and the other one carries the inverted image of the original. The parallel wires are close enough to form capacitive coupling to each other. This setup allows a very high immunity to noise/EMI and attention which effectively increases the range of the BUS. Modern microcontrollers generally have serial protocol using TIA/EIA-232 single ended form. If differential signaling is required, external circuitry needs to be added.

In TIA/EIA232 standard, digital HIGH (1) is represented by -3v to -15v and digital LOW (0) by +3v to +15v. The gap of -3v to +3v is undefined. To convert it to today's TTL standard, a converter needs to be used. Maxim/Dallas MAX232 is such a converter. It is bidirectional and can be used to both receive and transmit. In RS232 there is one pin RX for receiving and one pin TX for transmitting along with other control and status pins however only TX and RX pins can be used to communicate between devices with very little error. If VCC and GND are considered, a total of four wires are needed for complete asynchronous bidirectional serial transmission.

Figure 3.5 shows the data frame format of TIA/EIA 232.

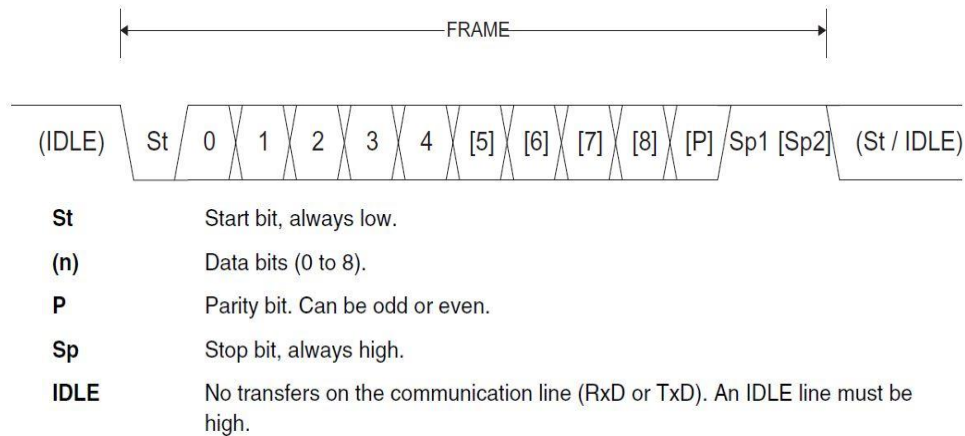


Figure 3.5 Data frame format of TIA/EIA 232 standard ^[19]

3.4 Requirement for single wire BUS

A common problem often faced by hardware developers is the fact that the numbers of physical I/O pins on the microcontroller may run out before the task at hand is completed. Modern microcontrollers are designed with a large number of features built into a chip of relatively small physical footprint. To ensure all features can easily be accessed while minimizing the physical footprint of the microcontroller, a single I/O pin may be used to serve more than one function. Thus it is very easy to have high pin usage count in the applications that uses a lot of the microcontroller's features. In this type of situation, it is often wasteful to dedicate multiple I/O pins for whatever serial communication protocol that is being used. Also often times the application demands that data is exchanged between multiple different types of standalone embedded systems with possibility of hot swap option. Removing system(s) from a data BUS without some sort of notification can crash the larger embedded system.

A bidirectional data BUS using only one I/O pin can reduce wastage of I/O pins while facilitate addition of standalone application chips such as EEPROM, ADC, Port expander, LCD display driver, PWM controller etc. Situations demanding minimum number of physical wires for bidirectional data exchange, single wire data BUS are without rival.

A brief comparison between IIC and RS-232 standard is presented in table 3.1.

Attributes	Name of Data BUS	
	IIC	TIA/EIA 232 *1
Number of wires	2	<ul style="list-style-type: none"> • 2 for asynchronous • 3 for synchronous
BUS Protocol	Defined as common industry standard	RS-232 (industry standard)
Clock Speed	100 to 400 KHz typical 1000 KHz maximum	2400 KHz minimum 115 KHz maximum (typical)
Signal type	Digital	<ul style="list-style-type: none"> • Designed for Analog communication systems • Can be converted to modern digital standard by using converters
BUS type	Serial	Serial
Signaling	Single ended only	RS232 – Single ended RS 422 – Differential *2 RS 485 – Differential *2
Range	Wire length is restricted by line capacitance of maximum 400pf	RS 232 – 20m RS 422 – 1200m *2 RS 485 – 1200m *2
BUS direction	Half duplex	Full duplex
Device configuration	Master Slave system	TX and RX can operate independently (RS 422 & RS 485 are Multidrop)

Table 3.1 comparison between IIC and TIA/EIA 232

Notes:

*1. Assume the RS-232 standard for embedded systems. Some devices have built in UART modules that require 3 pins for synchronous communication e.g. Atmega32A MCU.

*2. Different standard, presented here for reference and comparison purpose only.

Chapter – 4

Existing single wire BUS protocols

Presently multiple single wire BUS protocols exist in the market. These protocols are widely used in commercial products where absolutely minimum number of wires are needed to implement low speed BUS carrying non critical data. All of these single wire BUS are patented/copyrighted and use their own protocol for data transmission and signal generation. Two most popular ones are 1-Wire protocol from Maxim Integrated and UNI/O from Microchip Technology Inc. Both of these BUS are explained briefly in this section.

4.1 1-Wire Protocol ^[12] ^[14]

Maxim Integrated owns a proprietary single wire BUS that is currently being used widely in commercial applications. Maxim 1-Wire Protocol was initially designed to form a data BUS that can communicate between one Master and multiple slaves sharing the same data BUS located on a PCB. As the usage of devices using 1-Wire protocol increased, Maxim Integrated found a way to extend the range of 1-Wire protocol beyond the PCB and converted the BUS into a network which has been named M^IcroLAN. Present application of 1-Wire protocol includes but not limited to accessories, clone prevention, peripherals, security feature, intellectual property protection, PCB identification and authentication etc. ^[28].

1-Wire protocol utilizes Master-Slave configuration. The Master device is responsible for generating the required waveforms. 1-wire protocol is designed to be used in low speed embedded systems. The BUS or network uses open drain environment i.e. to use the BUS or network, a resistor must be used to pull-up the BUS to VCC. The value of the pull-up resistor depends on the BUS length, network topology, data rate or device power source ^[14].

One of the most interesting feature of 1-Wire Protocol is Parasitic Power which means that the Master device is able to provide the slave device power through the BUS itself. This feature allows 1-Wire Protocol to operate using only two physical wire namely BUS and GND. Therefore, the slave device can be designed to have no external power supply (not even battery) which makes the resulting product to have a very small physical dimension. However, this feature also impacts the BUS speed and thereby reduces data transfer bit rate.

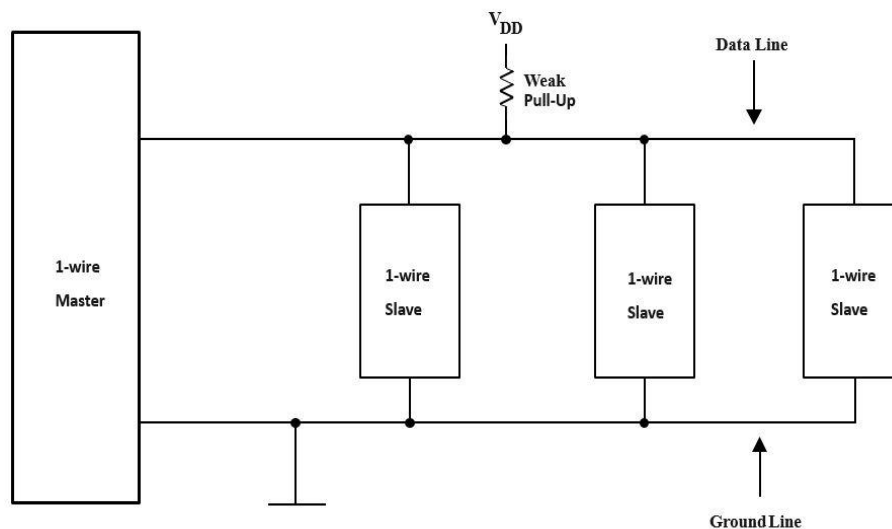


Figure 4.1 Typical device connections of 1-wire protocol ^[12]

Bit definition on 1-Wire is done by pulling down the BUS for specific time. 1-Wire has 4 different operations namely

- Write '1' bit
- Write '0' bit
- Read
- Reset

Each of these operations are implemented using different amount of delay to pull down the BUS. The following table 4.1 shows the operations and the conditions to implement these operations.

Operation	Description	Implementation
Read (any bit)	Read 1 bit from slaves	<ol style="list-style-type: none"> 1. Drive BUS low for 6 μs 2. Keep BUS released for 9 μs 3. Sample BUS for next 55 μs to receive one bit from slave
Write ('0' bit)	Write (send) '0' bit to the slaves	<ol style="list-style-type: none"> 1. Drive bus low for 60 μs 2. Keep BUS released for 10 μs
Write ('1' bit)	Write (send) '1' bit to the slaves	<ol style="list-style-type: none"> 1. Drive BUS low for 6 μs 2. Keep BUS released for 64 μs
Reset	Reset all 1-wire slave devices and get them ready to receive a command	<ol style="list-style-type: none"> 1. Drive BUS low for 480 μs Keep BUS released for 70 μs 2. Sample BUS for next 410 μs to receive ATR 3. '0' = at least one slave device is present 4. '1' = no slave device is present

Table 4.1 Commands and their respective implementation timing for 1-wire protocol ^[12] ^[16]

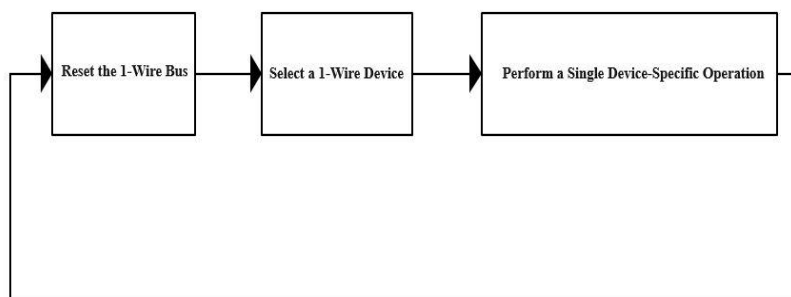


Figure 4.2 Typical communication flow of 1-wire protocol ^[12]

Timing diagram of 1-Wire is as follows for all of 1-wire protocol operations ^[12].

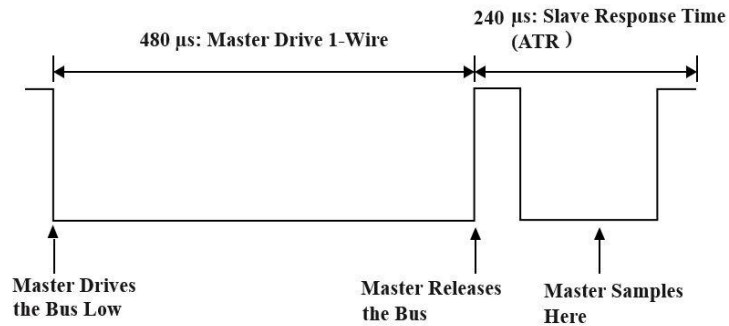


Figure 4.3 1-wire protocol **Reset** with at least 1 slave

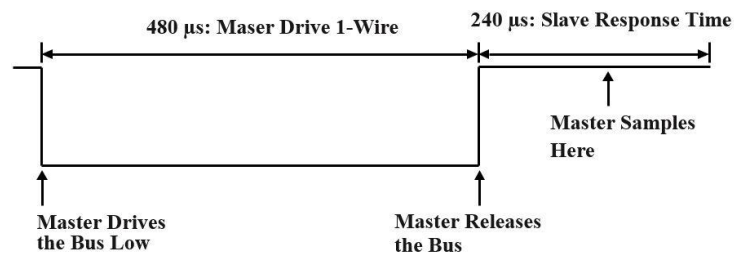


Figure 4.4 1-wire **Reset** with no slave

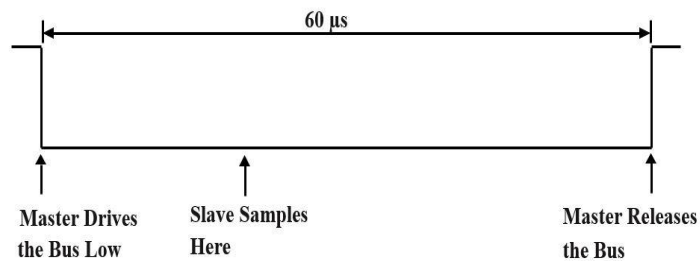


Figure 4.5 1-wire **Write 0** (digital LOW)

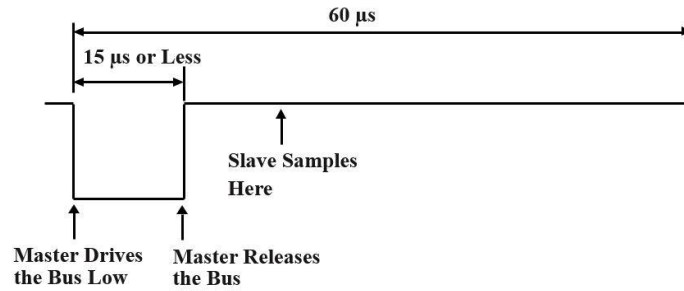


Figure 4.6 1-wire **Write 1** (digital HIGH)

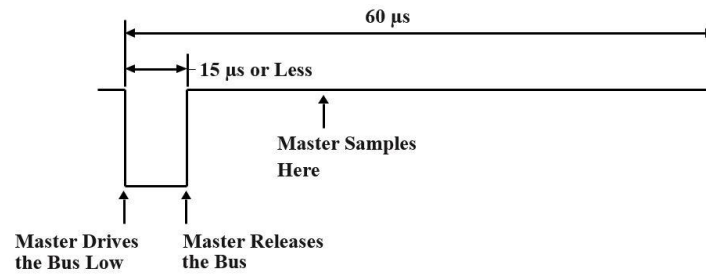


Figure 4.7 1-wire **Read 1** (digital HIGH)

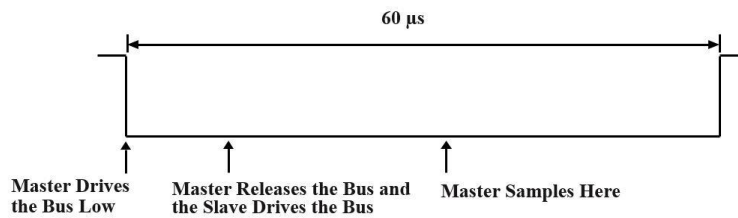


Figure 4.8 1-wire **Read 0** (digital LOW)

With all of these operations being setup perfectly, the 1-Wire can operate at two different bit rates of

- Standard at 16.3 Kbit/s [12] [16]
- Overdrive at approx. 110 Kbit/s [12] [16]

Complete details of 1-Wire can be accessed from the reference [12] [14] & [16].

4.2 UNI/O ^[2]

UNI/O is the single wire BUS designed, owned and patented by Microchip Technologies Inc. This is also a low speed BUS developed to carry data to multiple devices/components while using as small number of physical connections as possible to carry the data. UNI/O uses one I/O pin to communicate with all connected devices. Typical application of UNI/O of UNI/O includes but not limited to ADC, EEPROM, I/O Port Expander and wide range of sensors ^[2].

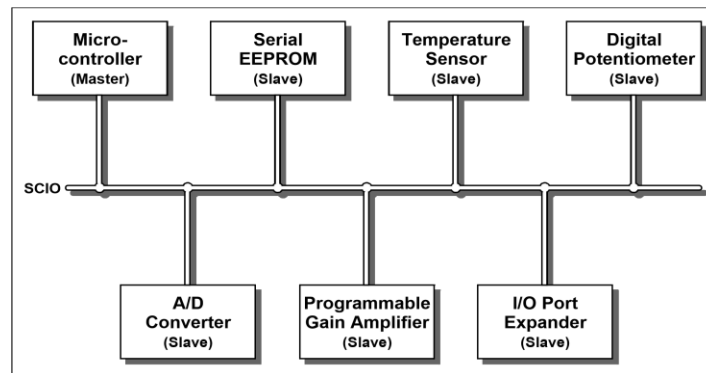


Figure 4.9 UNI/O BUS example ^[2]

UNI/O also uses Master-Slave configuration to communicate. The Master is responsible for determining the clock period, generating the data signals, controlling the BUS access and initiate all operations and modes. Both Master and Slave can be configured as transmitter and receiver however, Master is responsible for determining the active mode.

Both the clock and data is encoded together using Manchester Encoding. In Manchester encoding, each data bit is transmitted within a specified bit period. The bit period is defined by the Master at the starting of the communication. The bit period is sent to the Slave device by the Master device by transmitting a value of 0x55 (0b01010101) at the start header of the command. Each bit period includes an edge transition in the middle. The direction of the edge transition determines the logical state of the bit. A rising edge signifies a logical HIGH and a falling edge signifies a logical LOW.

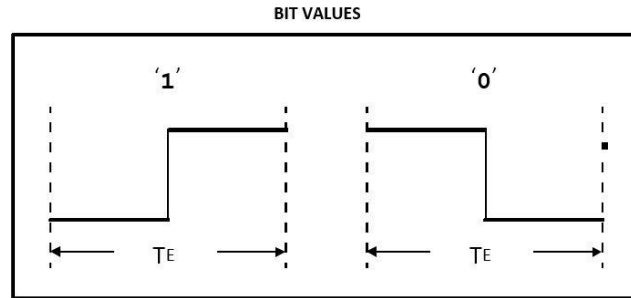


Figure 4.10 Manchester encoding ^[2]

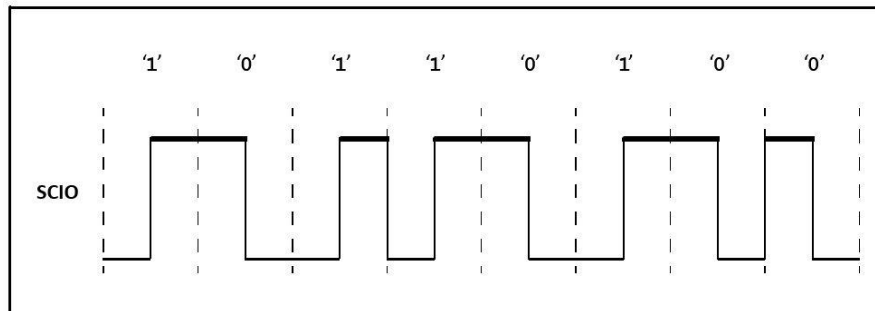


Figure 4.11 UNI/O one-byte transmission using Manchester Encoding ^[2]

When Master needs to communicate with any of the Slaves, it selects the Slave by transmitting 8-bit or 12-bit address. After transmission of device address, the Master must send a command byte to declare its intended operation. The types of operations are predefined on the devices. A complete operation can consist of a single byte to theoretically infinite number of bytes. Acknowledge and Not-Acknowledge bits are transmitted by both Master and Slave where they are needed. Acknowledgement bytes are named as follows

- MAK – Master Acknowledge
- NoMAK – Master Not Acknowledge
- SAK – Slave Acknowledge
- NoSAK – Slave Not Acknowledge

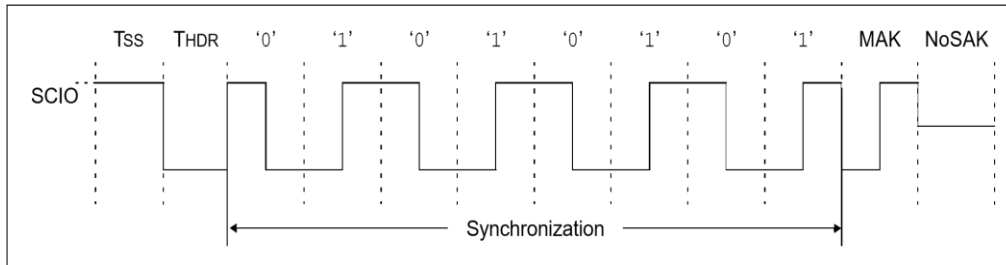


Figure 4.12 UNI/O complete one-byte transmission with both Master and Slave Acknowledgement [2]

A full operation always consists of the following steps

- Standby Pulse
- Start Header
- Device Address
- Command Byte
- Other necessary bytes required for the operation

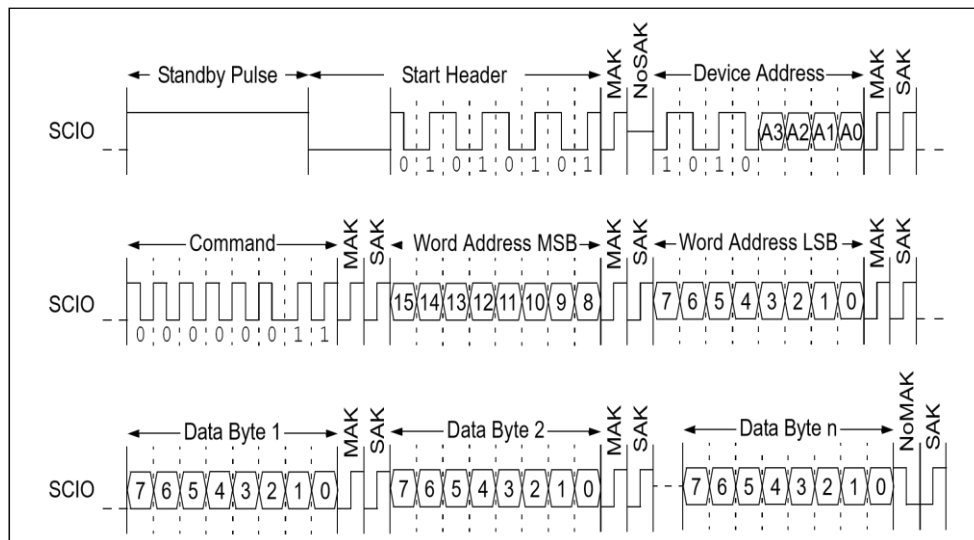


Figure 4.13 Complete timing diagram for read operation from UNI/O compatible EEPROM [2]

A complete BUS specification for the UNI/O can be accessed from the reference [2].

Chapter – 5

Proposed Single Wire BUS

Although low speed single wire data BUS for embedded systems exist in the market and are used widely in commercial and industrial applications, they use either delay in the signal or Manchester encoding for logical bit definition. The motivation of this research is to find another way multiplex both clock and data and send them down the same physical wire while making an attempt to have a higher communication bit rate than that are offered by standard transmission rate of the existing protocols. Attempts are to be made to keep the hardware for the BUS as simple as possible so that it can be simulated in real life with ease and, if commercially available, can be produced with a low cost. With all these requirements in mind, the specifications for the proposed single wire data BUS is defined as follows

- One Wire deployment
- Synchronous
- Half duplex
- Carries 8-bit data with 2-bit command header at 10-bit total
- Master-Slave configuration
- ACK/NACK/BUSY status transmission by both Master and Slave
- Data rate of 15 Kbits/sec
- Communication range is within the PCB only i.e. short distance
- Single ended signaling

5.1 Working principle

The biggest problem of sending data and clock together through a single wire is to differentiate between the data and clock at the receiver (discussed more at the concluding chapter). Algorithm can be generated to sample the data BUS at specific intervals until specific number of samples are taken and then based on predefined rules, clock and data is separated. However, this approach

beats the idea of a synchronous BUS as the samples are taken at specific interval rather than on arrival of a clock signal. As the sampling circuit is not triggered by an external clock signal, there is no point in sending a clock signal mixed with data bits to begin with. Also this approach runs the risk of inaccurately labeling the clock signal as data and vice versa in case there is any mismatch in timing between sending of the data and sampling it. The aforementioned single wire protocols address these problems by either

- using Manchester encoding on UNI/O from Microchip Technology Inc. ^[2]
- holding the line down for specific delay and synchronizing the data on falling edge of the signal on 1-Wire protocol from Maxim Integrated ^[12]

The proposed data BUS multiplexes the clock and data for transmission using same physical wire by assigning different voltage levels for data and clock. The conventional 5v TTL standard is slightly modified to suit this purpose. In the conventional setup, any signal with voltage level of greater than or equal to 2.4v is regarded as digital HIGH (logical 1) and signals with voltage levels less than or equal to 0.4v is regarded as digital LOW (logical 0). The mid-section between HIGH and LOW (from 0.4v to 2.4v), there is an unacceptable region. According to the 5v TTL standard, signals with voltage in this region is neither 1 nor 0 and is regarded as unacceptable. In the proposed BUS, while data signal is assigned the common 5v TTL standard for logical 1 (HIGH) and 0 (LOW), clock is assigned a voltage value midway between HIGH threshold and LOW threshold. The middle section of HIGH threshold and LOW threshold is unaccepted (metastable) region and is not used by the TTL standard. The clock signal is assigned a small region of 600mV inside the unaccepted 2.0v region. The region for clock signal starts at 1.0v and continues up to 1.6v. Figure 5.1 shows the conventional 5vTTL standard and the modified TTL standard for the proposed BUS.

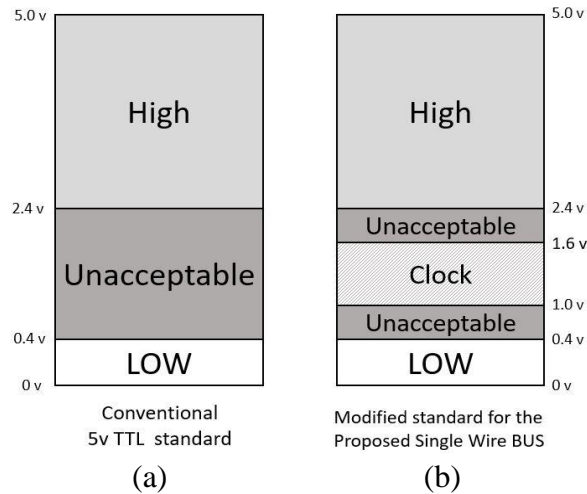


Figure 5.1 Voltage Levels for the logic states

(a) Conventional 5v TTL standard ^[9], (b) Modified standard for the proposed single wire BUS

The waveform for digital signal using conventional 5v TTL standard is shown in figure 5.2. In this case a 6-bit number of 0b101100 is being sent.

Data = 1 0 1 1 0 0

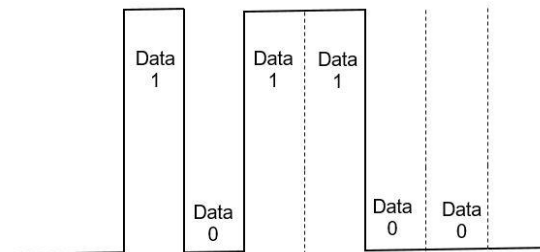


Figure 5.2 Digital signal waveform for conventional TTL standard

The same 6-bit number of 0b101100 when sent by the proposed modified TTL standard will assume the shape shown in figure 5.3. The clock signal and data signals are sent sequentially. Therefore, there is always a data signal next to the clock signal. Clock signal comes first followed by the data bit. The clock signal signifies the start of a new data bit. The receiver gets itself ready for receiving 1-bit data on arrival of a clock signal.

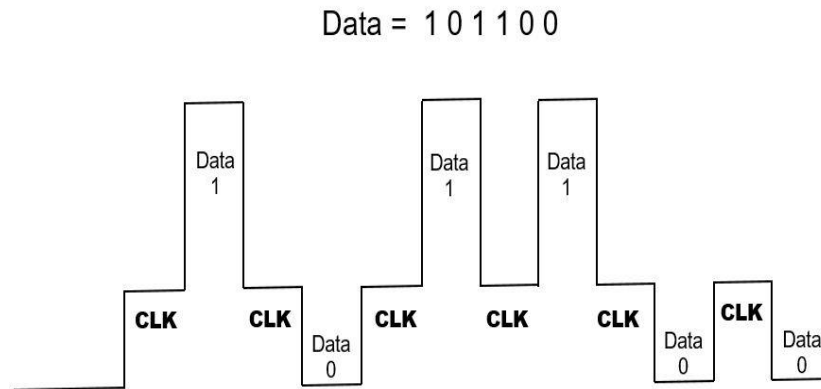


Figure 5.3 Digital signal waveform for proposed single wire standard

The transmitter is connected to the signal multiplexer and the receiver is connected to the isolator and data buffer (explained in detail in the next chapter). Both signal multiplexer and isolator consists of multiple blocks, each performing different functions. Throughout this whole literature, circuit that combines CLK and digital data is referred to as multiplexer and circuit that separates CLK and digital data from the multiplexed signal is referred to as isolator. Both the signal multiplexer and isolator has one physical connection to the data BUS, however, there are multiple physical wire connections between the signal multiplexer, isolator and the transmitter and receiver respectively. Both transmitter and receiver shares a common ground voltage reference (GND). The signals going into and out of the multiplexer and demultiplexer respectively are omnidirectional.

The multiplexing circuit is designed to have two stages, one for generating clock signals (1.0v to 1.6v) and another to generate data signals (less than 0.5v or greater than 2.3v). The multiplexing circuit has two inputs for driving it. The transmitter is responsible for generating the ON/OFF pattern for the different encoder blocks based on the given data to be transmitted. The output of clock driver and Data driver connects directly to the BUS.

5.2 Data frame format, BUS modes and communication protocol

As this thesis research is dealing with a non-existent data BUS, the complete communication protocol must be built from scratch in order to make the data BUS communicate during the real

life hardware emulation. This section sets the outline for the communication protocol, frame format and different modes used by the BUS to configure and/or communicate data between multiple standalone devices. Complete protocol will be generated in the next chapter(s) where the physical hardware is developed and simulated.

5.2.1 Data frame format

The proposed BUS communicates by sending/receiving data packet of predefined length. Data packet for the proposed BUS is defined in the research to be 10-bit long. The least two significant bits are the command headers and the rest 8-bits are the data. The most significant bit of data is also the most significant bit of the packet. That makes the least significant bit of data the 2nd bit of the packet. Data frame format for the proposed BUS is shown in figure 5.4.

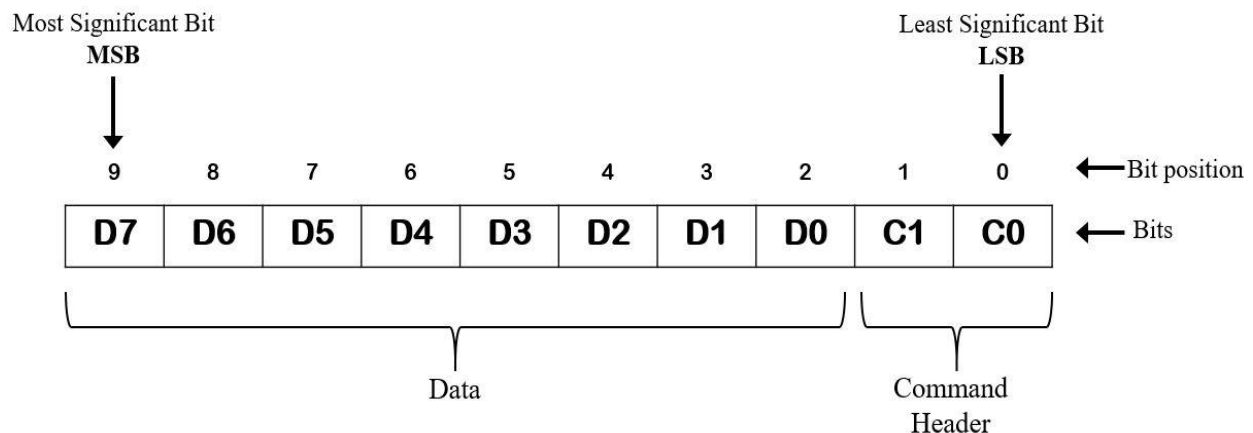


Figure 5.4 Data frame for the proposed single wire standard

5.2.2 BUS modes

The proposed data BUS has two different modes namely setup mode and general mode. During setup mode, all communicated data is used to configure the devices attached to the BUS. During general mode or general purpose mode, all communicated data are considered to be common traffic that will in no way modify any of the device configuration. All data communicated during general mode are directly conveyed to the user. Refer to table 5.1 for complete description of command

header combinations. Most of the combination in the command header are not used in this research and is reserved for future development.

Mode	Bits		Description
	C1	C0	
Setup	0	0	Unused
	0	1	Initialization of a slave
	1	0	Unused
	1	1	Unused
General	0	0	Unused
	0	1	Master intends to Read from slave
	1	0	Unused
	1	1	Master intends to write to slave

Table 5.1 Command headers and their respective mode descriptions

5.2.3 Communication protocol

The proposed BUS is designed to have a single master and multiple slaves at a time. The address is of 8 bits and therefore, theoretically, there can be maximum of 256 ($2^8 = 256$) devices on the BUS at a time including master and slaves. As the BUS is designed to have only one master, maximum of 255 slave devices can be controlled by one master. The master device possesses complete control of the BUS. The master can communicate with any slave it wants but the slaves cannot communicate between themselves. The slave device cannot even communicate with the master unless it has been specifically instructed to do so by the master.

As stated earlier, the BUS has two different modes and a data packet size of 10 bits and two different modes of operation with which it can communicate with the slave device. All this aforementioned information is compiled together to form a complete functional single wire data

BUS. The complete protocol for errorless communication of data between a master device and a slave device is stated on section 5.5.

5.3 M-Ary signaling and buffering

One can raise the argument that the signal shape of the proposed single wire BUS resembles that of the M-ary modulation method widely used in the telecommunication industry. However, that is far from the truth. M-Ary modulation consists of multiple symbols each symbol specified by a specific voltage level. Each symbol represents a certain combination of binary numbers. In the proposed BUS, the voltage levels represent either digital HIGH/LOW or a CLK signal. Figure 5.3 shows the example of a m-ary modulation of 14-bit binary data of 11101100010011. Each symbol in figure 5.3 represents a specific 2-bit combination.

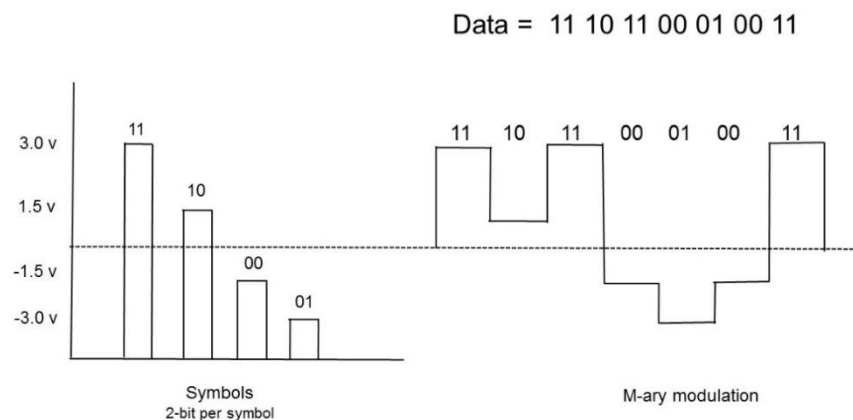


Figure 5.5 M-ary modulation of binary data ^[19]

M-ary modulation becomes increasingly susceptible to noise as the number of combinations per symbol increases. However, theoretically speaking, the proposed BUS will have a higher susceptibility to noise, relative to M-ary modulation. The receiver or buffer/repeater will know specifically which signal it is amplifying. After a CLK signal, there will only be room for a digital signal. After the digital signal, only expectation is CLK signal. Therefore, it is virtually impossible for the receiver or buffer/repeater to mix-up between CLK and data and amplify the wrong type. The buffer/repeater will have to have two separate stages. One will reinforce digital signals and other the CLK signal. It will need some control system to determine the correct stage and switch

between two stages. The buffer/repeater need not be a single standalone system as it can be embedded on the receiver. This research will not be going forward with designing buffer/repeater.

5.4 System and sub-system block diagrams

Before the system hardware is designed, block diagram for the whole system is generated to simplify the design stage of the actual circuits. The whole system is divided into multiple large blocks and their respective sub-blocks. The large blocks accomplish multiple complicated tasks. The sub-blocks are designed to accomplish only one task. Multiple sub-blocks, each performing a single task, can combine to together to form one large block that can perform multiple complex tasks.

There are only 4 large block in the system namely

- Multiplexer
- Timer
- Isolator
- Error-handler

This section explains the working principle of all the large blocks and these theory of operation will be used in the next chapter to generate the actual circuit and the respective simulation parameters to test the respective circuits.

5.4.1 Multiplexer

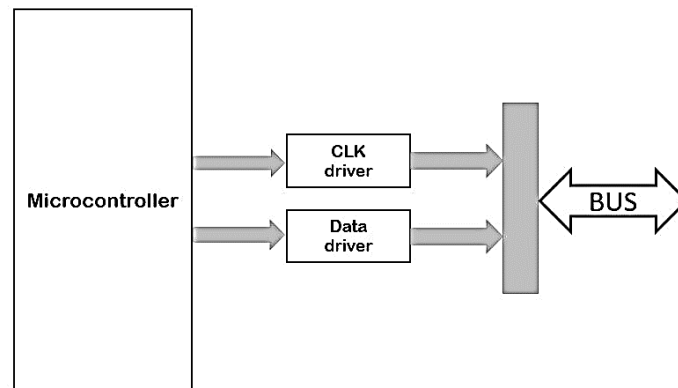


Figure 5.6 Block diagram of data and clock multiplexing circuit

The multiplexer combines CLK and data to form the multiplexed signal and then transmits the signal up the BUS. The multiplexer has two different sub-blocks (circuits) for driving the BUS to CLK level and to switch between digital 1 & 0 for data transmission. Note that only one circuit block can be activated at a time. Refer to figure 5.6 for the multiplexer block diagram. Following the arrows going from the microcontroller to the sub blocks, it can be noted that the sub-blocks and, by extension, the multiplexer takes in no input from the BUS.

5.4.2 Timer

Timer is a large block that has other three sub-blocks namely

- Time-out counter
- 1-bit memory
- Logic state comparator

Time-out counter is a basic count up timer that, once triggered, starts counting time up from zero until a preset threshold is reached. Logic state comparator is a basic digital logic comparator. 1-bit memory holds a single bit of status information about the current triggered state of the timer.

The timer takes in two inputs and has one output. Both inputs are from the BUS and are electrically shorted to each other. Figure 5.7 shows the internal architecture of a timer.

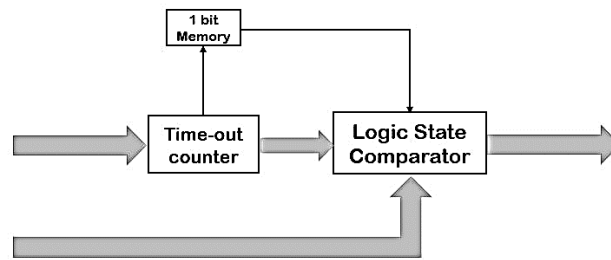


Figure 5.7 Timer internal architecture

The Timer serves two purpose. One of them is to detect if any specific mode request is sent by the BUS. The other one is to differentiate between data and CLK. Even though we are dealing with digital signal, any digital signal is nothing more than an analog voltage that has been defined in such a way that if a specific signal has its voltage level within a predefined threshold, the said signal occupies a certain digital state. In order to change digital state, the voltage level has to transit from one voltage threshold to another. This transition of voltage is a continuous function as shown in figure 5.8. Voltage cannot jump from 3.5v to 0.2v without being in all the different voltages in between 3.5v and 0.2v (figure 5.8). Therefore, whenever a data of digital 0101 is being transmitted, actually the voltage level of the carrier wire is jumping between 5.0v and 0.0v. If this analogy is applied to the proposed single wire BUS, every time data is transmitted through the BUS, provided the former data bit is compliment to the present data bit, there will be a transition in voltage level. As voltage transition is a continuous function, during transition the voltage level of CLK (1.0v to 1.6v) will be encountered and the circuits up the BUS will falsely think that a CLK has been received. This false CLK will most likely cause the whole system to malfunction.



Figure 5.8 Voltage transition

(a) Digital signal, (b) Actual continuous representation of voltage transition

To prevent false CLK triggering during data transmission, the timer block is used. To get a definite CLK, the transmitter needs to hold the BUS at CLK level for a predefined amount of time. The clock detecting timer (Timer-1) will be triggered immediately the BUS voltage is on CLK level. If the BUS voltage is still on CLK after the timer (Timer-1) times out, the signal is a definite CLK. Else the signal is data. The 1-bit memory serves to save the current triggering state of the timer so that a timer cannot retrigger while it is already triggered. All timers reset itself after each trigger timeout cycle to get itself ready for next run.

The system contains two timers as in Timer-1 and Timer-2. Both timers have same internal construction but different threshold for Time-out counter and both of them has different trigger voltage level.

5.4.3 Isolator

The isolator circuit is also designed to have two levels, one for isolating clock signal and another for isolating data signals. In terms of simplicity, the multiplexing circuit is much simpler than the isolator circuit. The isolator circuit has multiple blocks and sub-blocks. The isolator circuit consists of four main components i.e. one analog comparator, two timers and one data buffer. Block diagram of isolator is shown in figure 5.9.

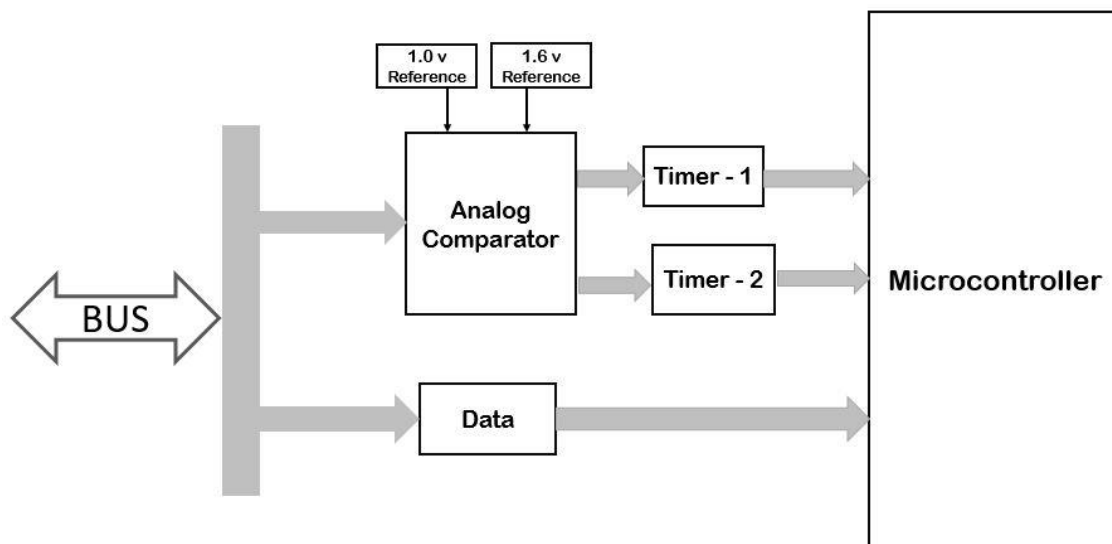


Figure 5.9 Block diagram of clock isolating circuit (isolator)

The analog comparator and Timer-1 is used to detect clock signal and Timer-2 is used to detect initiation of Setup mode. Data buffer is used to differentiate between data and clock signals (CLK is in metastable region) and buffer the data signal.

The clock detection block works in two steps. First the analog comparator detects the voltage level that is assigned to the clock signal. There are two voltage references provided for the analog comparator ^[6] ^[20] that sets the boundary of the comparator. If the analog comparator detects any voltage that is within its boundary, it set the output to HIGH (initially the output is LOW). In theory, the microcontroller would take the digital HIGH as presence of clock signal and would ready itself for data signal.

Setting the output of analog comparator HIGH causes Timer-1 to trigger. The time-out counter inside Timer-1 will raise a flag on its 1-bit memory, keep time-out counter running for a predefined time and on timeout triggers the Logic State Comparator. The logic state comparator checks if the signal on the input has changed since the time-out counter was triggered. If they are same, the output of “Logic State Comparator” is HIGH which shows a definite clock signal has been received. The 1-bit memory serves the purpose of stopping the Timer-1 from retriggering for the same event. The system resets itself once the comparator output is LOW.

Timer-2 serves the purpose of detecting the setup mode. It triggers itself whenever the BUS is pulled down to CLK state. Timer-2 has similar setup inside it as Timer-1. Timer-1 and Timer-2 has different predefined delay (time-out counter) setting. Normally the output of Timer-2 is LOW. If setup mode is detected, the output changes to HIGH. Whole setup of Timer-2 resets itself once the BUS is pulled up from VEE. Once triggered, the output stays HIGH as long as the BUS stays pulled down to CLK state.

Each single wire module acts both as a transmitter and receiver. Therefore, each single wire module must have both a signal multiplexer and a signal isolator (figure 5.10). In all of the blocks, a microcontroller is used as a signal receiver and transmitter however, in real life application, the microcontroller can/will be replaced by a dedicated control logic (or ASIC in case of mass

production) that can take in data from external systems, send it along the single wire BUS and also receive the data from single wire BUS and send it to the external systems connected to the module.

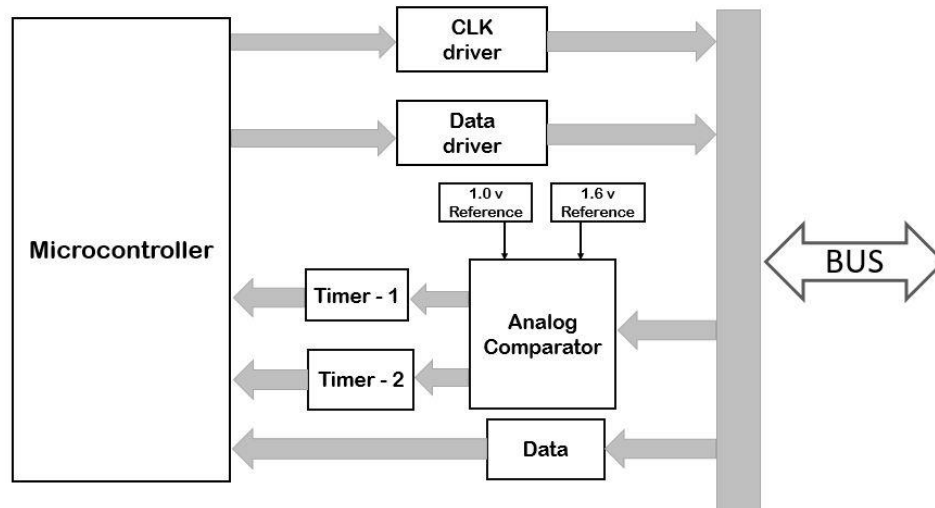


Figure 5.10 Block diagram of combined multiplexer and demultiplexer to form a single module

5.4.4 Error handler

The shown block diagram of the BUS works perfectly in ideal situations, however, BUS latch up is inevitable in real life applications. The definition of BUS latch-up in this literature is when the BUS is pulled down to VEE/GND and kept there indefinitely. The block diagram of error handler is presented in the figure 5.11.

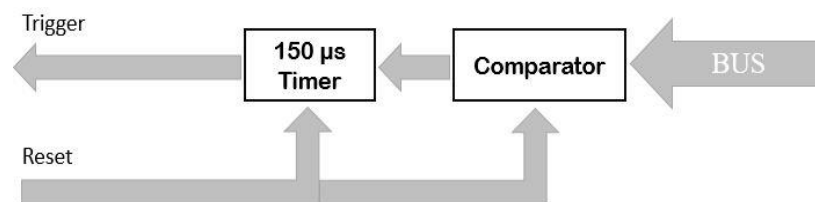


Figure 5.11 Block diagram of BUS latch up prevention circuit (error handler)

The characteristics of the error handler is as below.

- As long as the Reset is set at HIGH, the error handling module stays inactive. Therefore, it is possible for the microcontroller to enable/disable timer at its own discretion. In this case, the error handling module is enabled by the microcontroller whenever slave device is showing BUSY status after receiving 10-bit data.
- Whenever the BUS is pulled to VEE/GND, the comparator triggers the timer by setting timer trigger input HIGH. The timer stays disabled as long as the trigger input stays LOW. Therefore, the comparator can stop the timer and disable it by setting the trigger input to LOW.
- In case of BUS latch up by any of the slave devices, the timer causes all the other respective devices to release the BUS and end the current procedure.

The complete block diagram of the module including the error handler is shown in the figure 5.12.

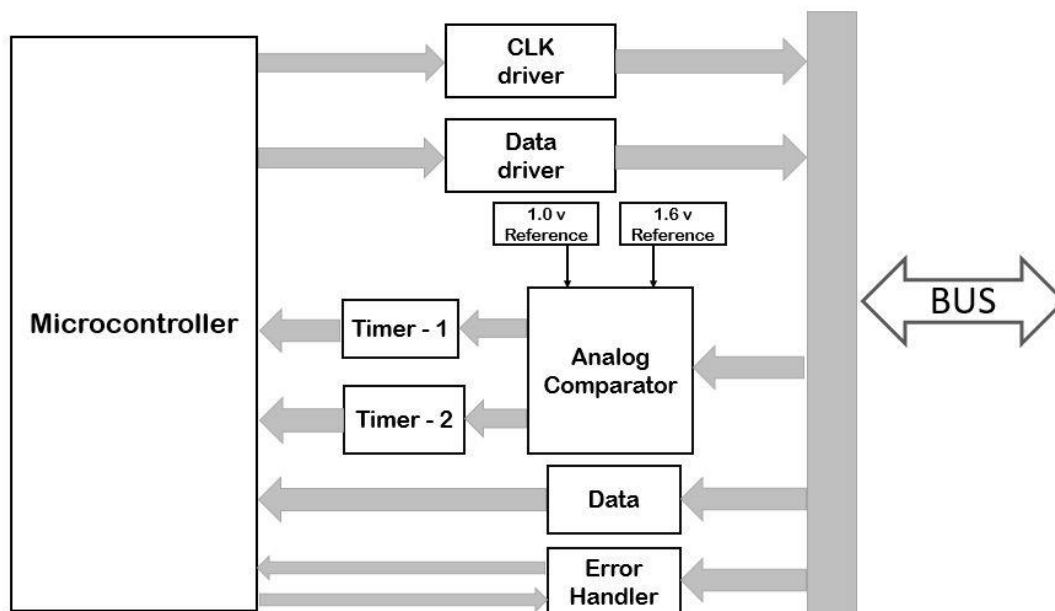


Figure 5.12 Complete block diagram of combined multiplexer and demultiplexer to form a single module

5.5 Complete protocol

The microcontroller is required to follow the following steps to complete one errorless communication of 10-bit data using the proposed single wire data BUS utilizing the given block diagram between a master device and a slave device.

1. The master checks if the BUS is free.
2. The master pulls the BUS to CLK state for 500 μ s to enter the setup mode.
3. On releasing the BUS to idle, the slave devices send ACK by pulling the BUS down to VEE/GND for at least 500 μ s to notify the master device that there is at least one slave device present on the BUS.
4. On receiving ACK, the master transmitter sends in 8-bit slave address with the write bit followed by enabling its error handler module.
5. The slave receives the 8-bit address, and pulls the BUS down to VEE/GND to send both ACK and BUSY status if the address matches.
6. The master knows that the slave of this particular address exists in the BUS as BUSY status would be missing without a positive address match. The master therefore awaits while the BUSY status is enabled.
7. The slave removes the BUSY status as it is ready to receive data.
8. The master checks the BUS error status after detecting that the BUSY status has been removed.
9. The master sends 10-bit data (8-bit data and 2-bit command header) down the proposed single wire BUS.
10. Upon receiving the data, the slave enables BUSY status while the data is processed and then the BUSY status is removed to free the BUS.
11. The master can now decide to send in more data or move on to attend to other tasks. The slave will continue to wait until more data is send. In case of multiple master system, the current master can send the proper command to release the current slave and make the BUS free for other master to use.

Chapter – 6

Hardware Design

A complete working principle of the proposed single wire BUS along with different blocks for handling different tasks has been described in the former chapter. This chapter deals with actual hardware design of the blocks. During the design phase, attempts were made to generate the hardware from off the shelf components. Before attempting to prototype on the breadboard, the circuits are simulated in computer based simulator. This thesis work uses National Instrument's "Multisim 14.0" simulator's both SPICE and Interactive simulator. Types of simulation are switched between dc sweep and transient analysis for SPICE.

6.1 Signal Multiplexer

The signal multiplexer is responsible for multiplexing the clock and data together to generate the multiplexed signal according to the modified TTL standard required for the proposed single wire data BUS. The proposed data BUS is designed to be an open collector system. To use the BUS, a single pull-up resistor must be used to pull the BUS to VCC. Therefore, the idle state of data BUS is always digital HIGH. When working, the BUS can assume any of three available states namely, digital HIGH (1), digital LOW (0) and CLK (clock). The available states and their corresponding voltage levels can be summarized as

1. Idle state – Always same voltage as VCC (5v in this case).
2. Data digital HIGH (1) – Voltage greater than or equal to 2.4v
3. Data digital LOW (0) – Voltage less than or equal to 0.4v
4. CLK (clock) – Voltage greater than or equal to 1.0v and less than or equal to 1.6 v.

As the states of the BUS are different than the conventional data BUS, off the shelf digital IC's cannot be used to produce all of these states. Therefore, custom made circuits are made from discrete transistors and resistors to accommodate the requirement.

6.2 Multiplexer working principle

The receiver assumes any signal having voltage between 1.0v to 1.6v as clock. For the purpose of hardware simulation in real life condition, a midpoint of 1.30v is chosen to send up the data BUS as clock. To generate a voltage of 1.3v from a 5v supply is a simple task that can be accomplished by using a simple resistive voltage divider. To generate a voltage of less than 0.4v or something very close to VEE can be achieved by shorting the BUS to VEE. As the data BUS is already pulled up to VCC by a resistor, the current flow during the short circuit is limited. The current flow can be minimized by using high value of resistance. Refer to figure 6.1.

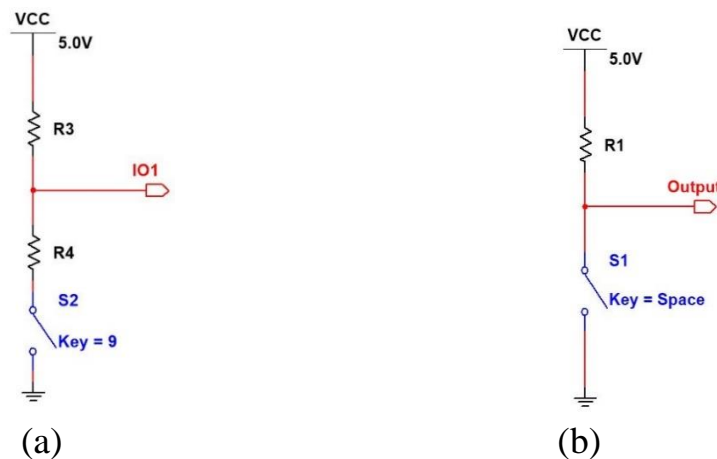


Figure 6.1 Resistive voltage divider
(a) Clock voltage level generator, (b) Data voltage level generator

Neither the resistive voltage divider or the short to VEE can be permanently placed in the BUS as they not always needed and are used one at a time. Therefore, next problem is to figure out a way to selectively apply these voltages on the BUS. The simplest way in this case is to attach individual switch/switching network to the individual resistive network and shorting circuit and turn the switches on/off to apply the effect of respective circuits controlled by the respective switch/switching network. Turning the switch ON will connect the corresponding network on the BUS however, turning the switch OFF will cause the network to be isolated from the BUS.

As the whole system is digital in nature, transistors are used as electronic switches. The current handling capability of neither the BUS nor the employed switches or resistors need to be high as they are not dealing with high power exchange. The switches however need to have a bandwidth that is compatible to the BUS frequency. Figure 6.2 shows the for clock/CLK generator and figure 6.3 shows the data level generator circuit. Both these circuits are attached to a digital pattern generator and an oscilloscope to generate the switching signals and to view them in real time respectively.

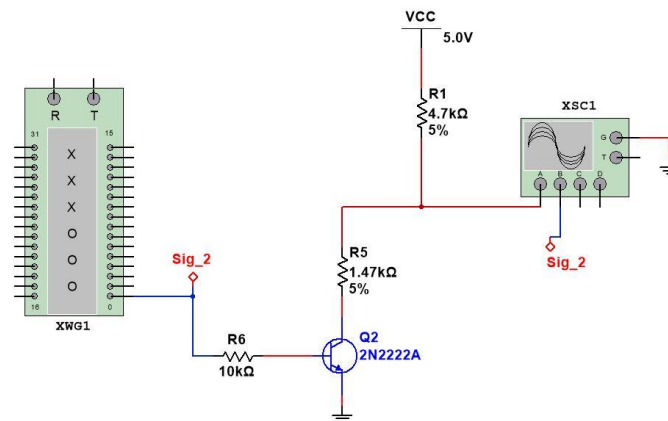


Figure 6.2 Clock generating circuit

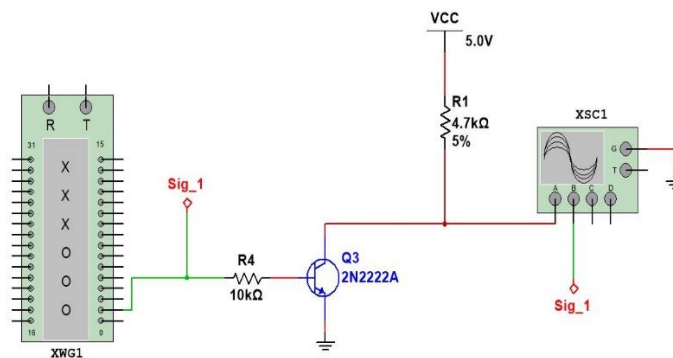


Figure 6.3 Data generating circuit

The complete multiplexer circuit containing both data and clock handler is presented in the figure 6.4. The simulation uses pattern generator to generate switching sequence to imitate transmission of a data and clock. R5 from figure 6.2 is a precision trimmer potentiometer to adjust the CLK signal voltage to its appropriate range.

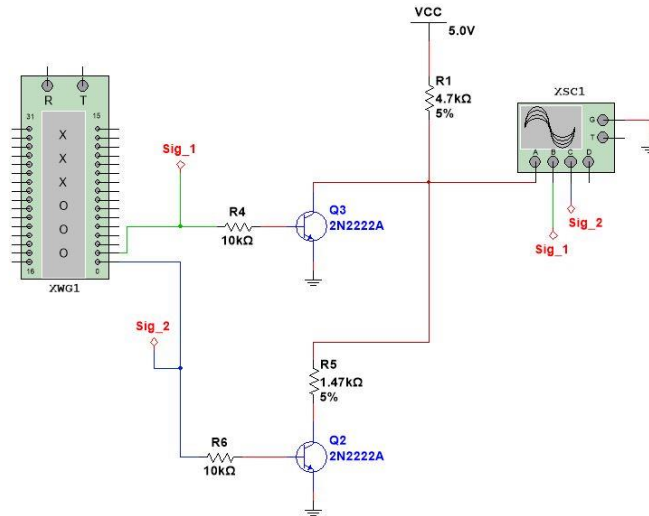


Figure 6.4 Complete multiplexing circuit

From figure 6.4, value of the pull-up resistor (R1) is defined arbitrarily using educated guess, however, the value of CLK level generator resistor (R5) has to be calculated. As R1 and R2 forms a resistive voltage divider, the value of R5 always depends on R1. Change in R1 value will affect the voltage generated by the said voltage divider. The following table 6.1 shows the component list for the complete multiplexing circuit.

# SL.	Component Designator	Component Name/Value	Component type	Description
1.	Q3	2N2222A	npn transistor	Switch for Data signal
2.	Q2	2N2222A	npn transistor	Switch for Clock signal
3.	R1	4.7 K Ω	Resistor	Pull up resistor
4.	R4	10 K Ω	Resistor	Current limiting resistor
5.	R5	1.47 K Ω	Resistor	Partial pull down resistor
6.	R6	10 K Ω	Resistor	Current limiting resistor
7.	XWG1		32-bit pattern generator	Simulate the signals generated by the transmitter
8.	XSC1		4 channel analog oscilloscope	Captures the signals on the BUS and generated patterns

Table 6.1 Component list for multiplexing circuit

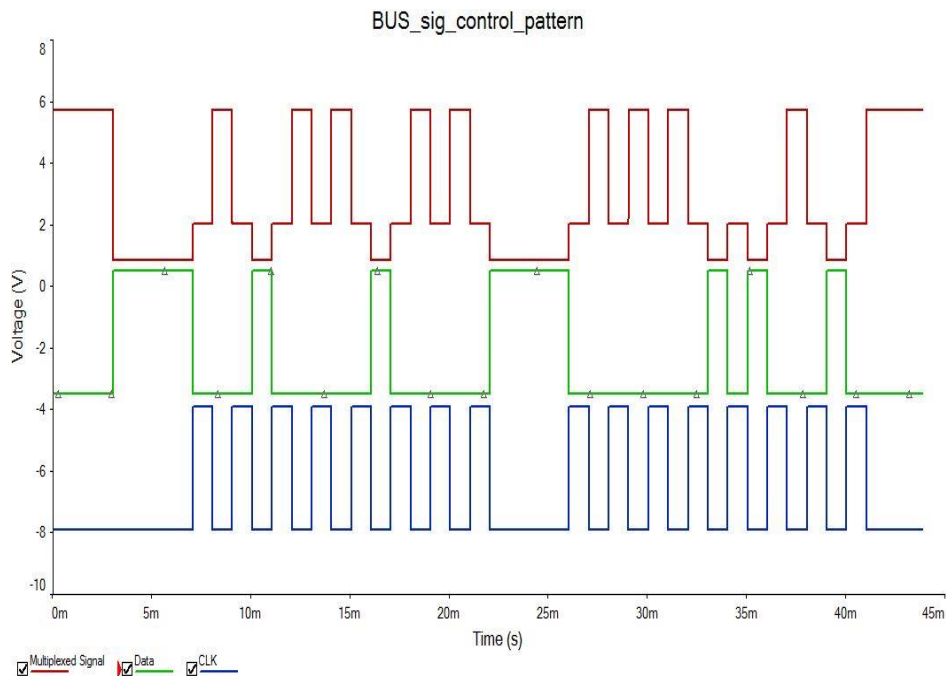


Figure 6.5 Timing diagram and output of the multiplexing circuit
From bottom to top: CLK (blue), Data (green), Multiplexed signal (red)

Referring to figure 6.5, the bottom trace is the clock, the middle trace is data and the top trace is the clock and data multiplexed using the proposed single wire protocol. Both the data and clock signal is generated as a digital signal. The multiplexing circuit converts it into its multiplexed form. From figure 6.5, it should be apparent that the data trace is the inverted version of its actual form. This is necessary to drive an open collector system.

The data can be read from the multiplexed signal very easily. Everytime the clock signal appears, it has to be followed by a data signal. After the appearance of clock signal, if the BUS is going LOW, the data is digital 0. If the BUS goes HIGH, the data is digital 1. From figure 6.5, the data is, in two 7-bit packets, 1011011 followed by 1110010.

6.3 Isolator

It is the isolator's task to isolate the clock signal from the encoded signal. The voltage range of clock signal is from 1.0v to 1.6v. The isolator, therefore, needs to respond to any signal with voltage that

are within this range. The isolator can be compared to a band-pass filter. In a band-pass filter, a threshold is set for low frequency cutoff point and another threshold for high frequency cutoff point. Similarly, to detect specific voltage within a given voltage range, the isolator uses two different analog comparators attached to two threshold voltage reference.

One comparator is used to set the low voltage threshold and another is used to set the high voltage threshold. The isolator is designed to output either zero volt or the voltage of VCC. If the input voltage is not within the isolator's boundary voltage, the output is zero volt. If the input voltage is within the isolator's boundary voltage, the output is complete voltage of VCC.

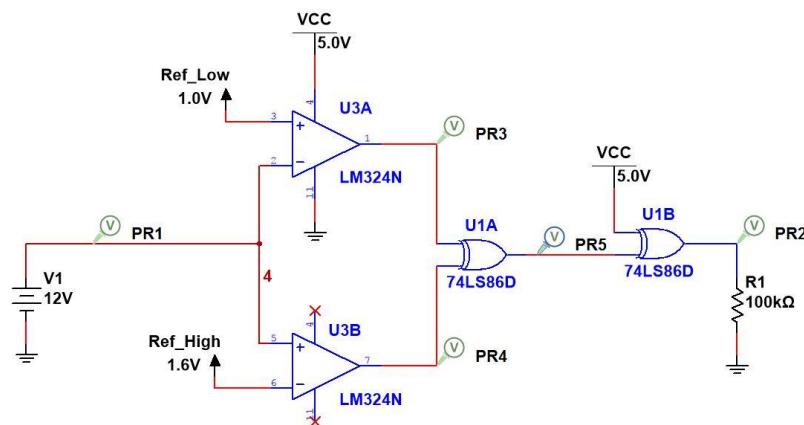


Figure 6.6 Complete isolator circuit

The simulation result of the DC sweep is shown in figure 6.7. As expected, the isolator circuit output is HIGH only when the input voltage is between 1.0v and 1.6v.

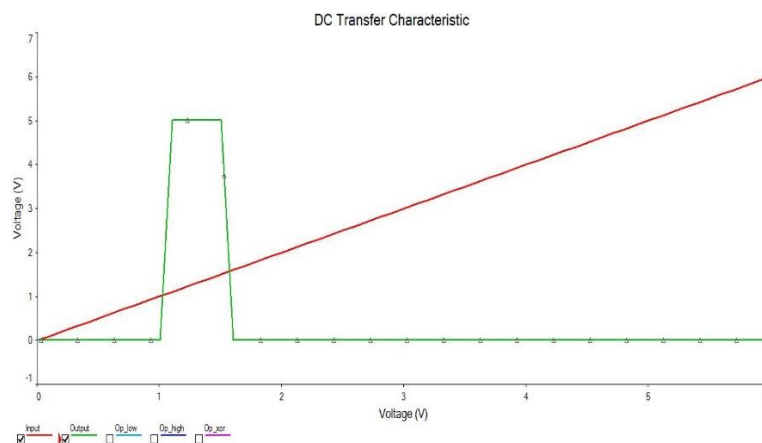


Figure 6.7 DC sweep of the isolator

Table 6.2 presents the list of component used to build the simulator model of the isolator circuit.

# SL.	Component Designator	Component Name/Value	Component type	Description
1.	U1A	74LS86	XOR gate	Digital comparator
2.	U1B	74LS86	XOR gate	Digital buffer and inverter
3.	U3A	LM324N	General purpose op-amp	Comparator for low threshold
4.	U3B	LM324N	General purpose op-amp	Comparator for high threshold
5.	R1	100 K Ω	Resistor	Resistive load, as required for simulation (Multisim cannot determine voltage probe output at no load condition)
6.	V1	12 volts	Voltage source simulation model	DC voltage supply model for voltage sweep

Table 6.2 Component list for isolator circuit

6.4 Isolator working principle

Figure 6.8 shows the state of the low threshold comparator when the input is within the boundary voltage. Figure 6.9 shows the state of high threshold comparator when the input is within the boundary voltage (note the triggering voltage from both figures). From figure 6.6, observe that the output of the comparators of the isolator is connected to the input of the XOR gate U1A. The XOR gate detects differences on its input. If there is a difference in the logic state of the input pins, the XOR gate output is HIGH, else it is LOW. From figure 6.8 and figure 6.9, observe that the output of the comparators is always at different voltage (voltage at VCC on figure 6.8 and VEE on figure 6.9) unless the input voltage is within the boundary region. Therefore, when the input voltage is within boundary region, both inputs of XOR gate U1A becomes similar and the output of U1A becomes LOW as shown in figure 6.10. The isolator is designed to output HIGH when the input voltage is within its boundary voltage. To make the LOW of U1A HIGH, an inverter is connected at the output of U1A (refer to schematic on figure 6.6). To minimize the number of chips used in

the circuit, the inverter is made out of another XOR gate, because chip with 4 XOR gates is already being used, namely U1B. This way the output of isolator becomes HIGH only when the input voltage is within its boundary voltage. Figure 6.11 shows all the outputs of the individual components of the isolator superimposed on a single graph.

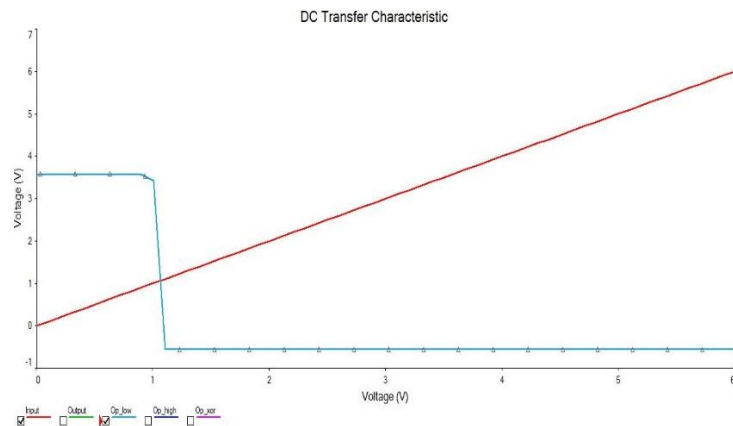


Figure 6.8 DC sweep of low threshold comparator

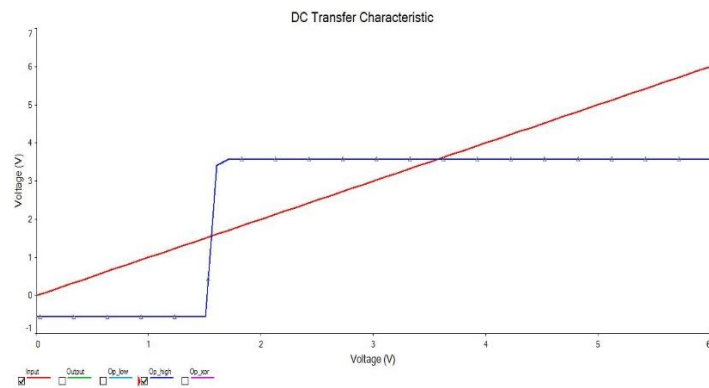


Figure 6.9 DC sweep of high threshold comparator

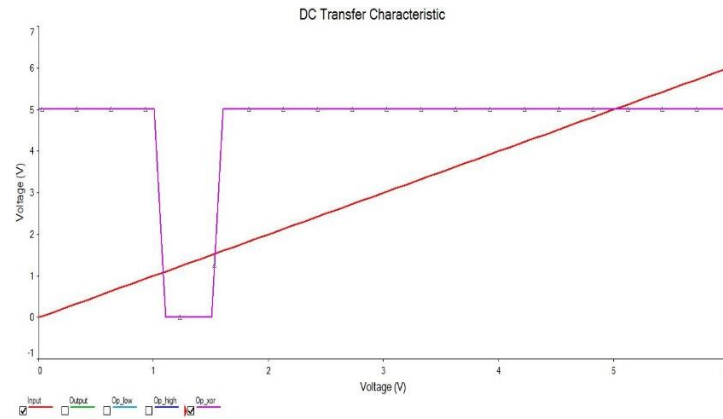


Figure 6.10 DC sweep of comparator XOR gate

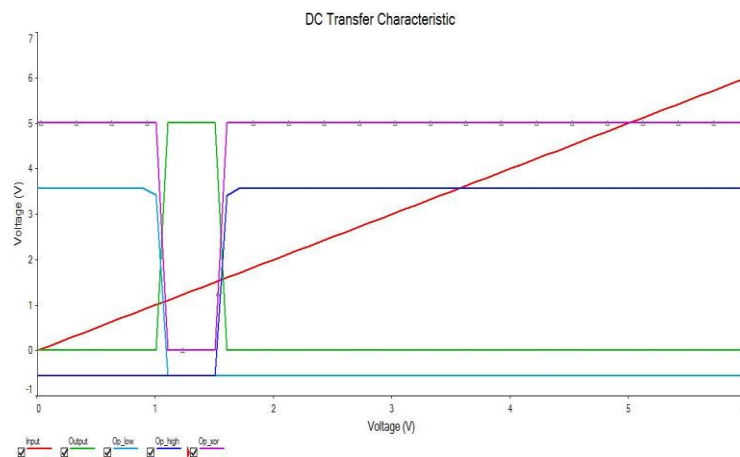


Figure 6.11 DC sweep of all components of the isolator

Input (red), Output (green), Low threshold comparator (cyan), High threshold comparator (blue), XOR of both comparator (pink)

Note that the output of comparators can become lower than zero in the simulation. This is because the comparators are made from analog operational amplifier. The slight negative voltage output is due to the semiconductor architecture of the operational amplifier. The output of XOR gates are always either VCC (5v) or VEE (0v) as negative voltages are unacceptable in digital electronics and therefore digital electronics' components are designed as such.

6.5 Test Setup of complete module

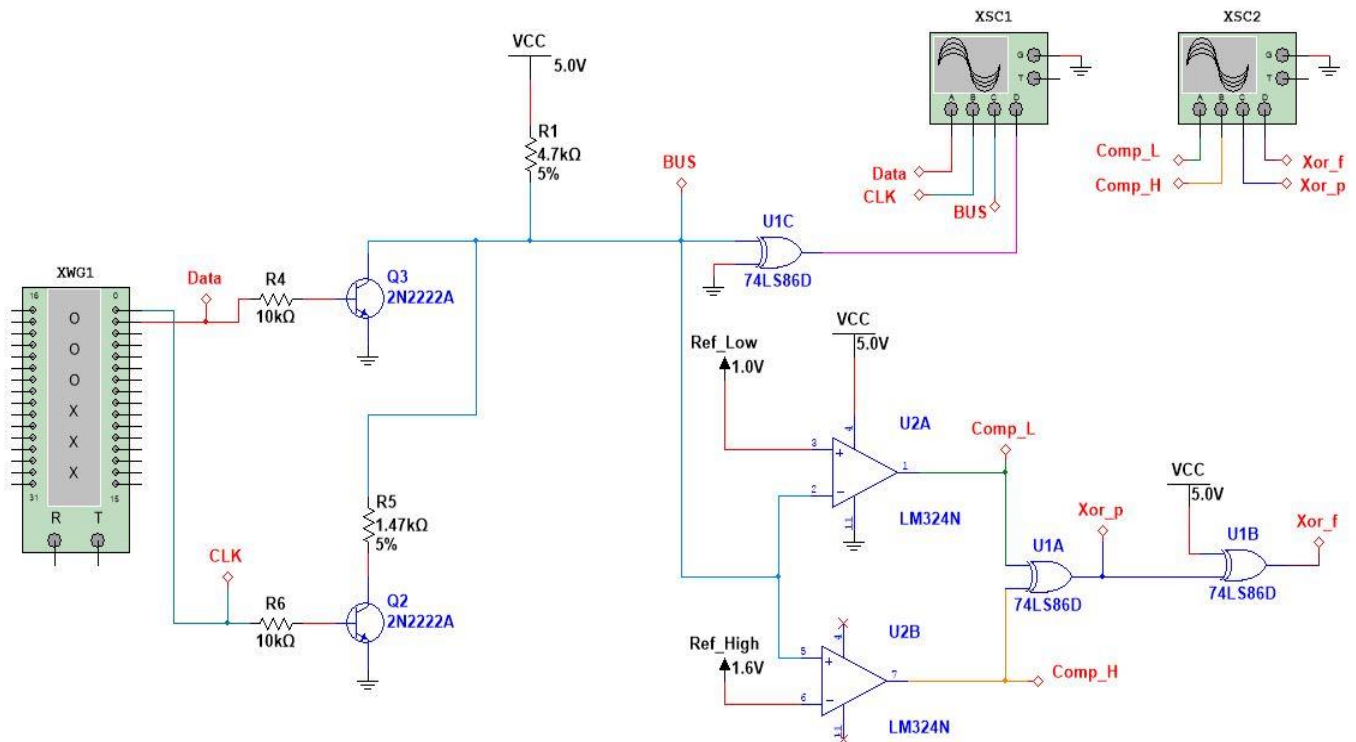


Figure 6.12 Complete module combining TX RX module

Figure 6.12 shows the test setup combining both the multiplexer and the isolator. The multiplexer multiplexes the CLK and data on one side and the isolator separates CLK and data on the other side and shows them on the oscilloscope. A multiplexer and an isolator forms the complete module needed to communicate using the proposed single wire data BUS. Each device using this BUS (be it a master or slave) will have one complete module built in.

The pattern generator generates the multiplexed signal following the proposed single wire BUS protocol. The multiplexed signal is then passed on to the isolator circuit which extracts the clock from the signal and the data is extracted by the digital buffer U1C (figure 6.12). Please note that the component designator numbers in figure 6.6 and figure 6.12 may not be identical. Figure 6.13 shows the test data, CLK (clock signal), multiplexed signal and the extracted data signal from the multiplexed signal. Figure 6.14 shows the various stages of clock extraction process.

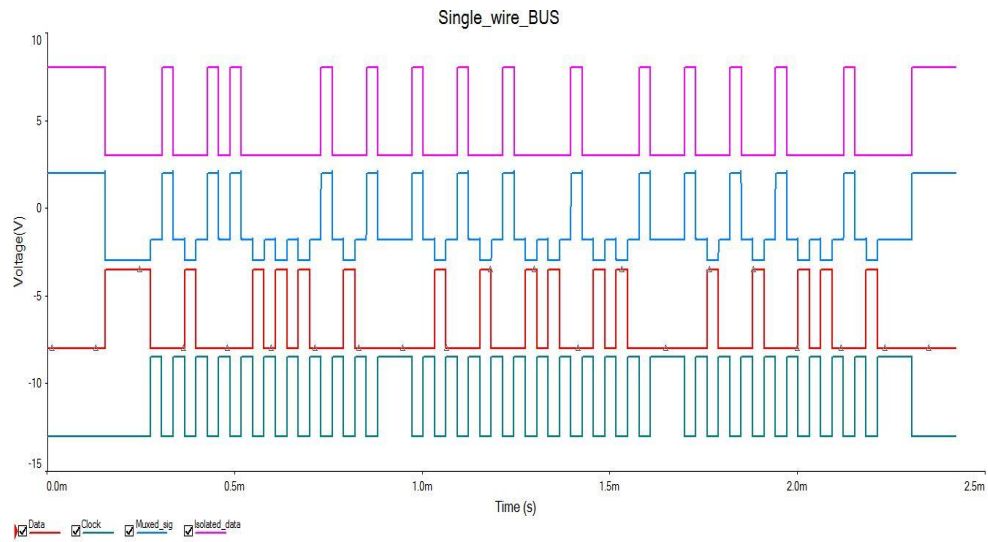


Figure 6.13 Waveform of the combined module

From bottom to top: CLK (cyan), Data (red), Multiplexed signal (light blue) and extracted data from multiplexed signal (pink)

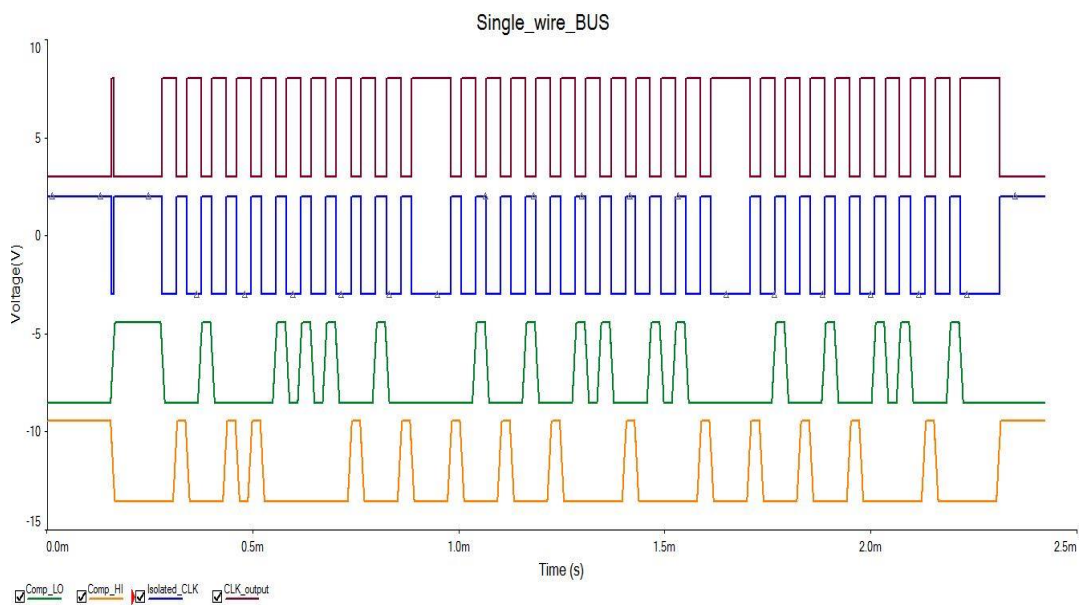


Figure 6.14 Waveform of the isolator output

From bottom to top: Output of high (yellow) and low (green) Comparator, XOR gate U1A (violet) and extracted CLK from multiplexed signal (dark red)

Notice from figure 6.14 that there is a very short duration spike at the starting of the clock signal is extracted by the isolator from multiplexed signal. That spike should not be there, and therefore, should be treated as a glitch. Figure 6.15 shows the magnified section of the glitch.

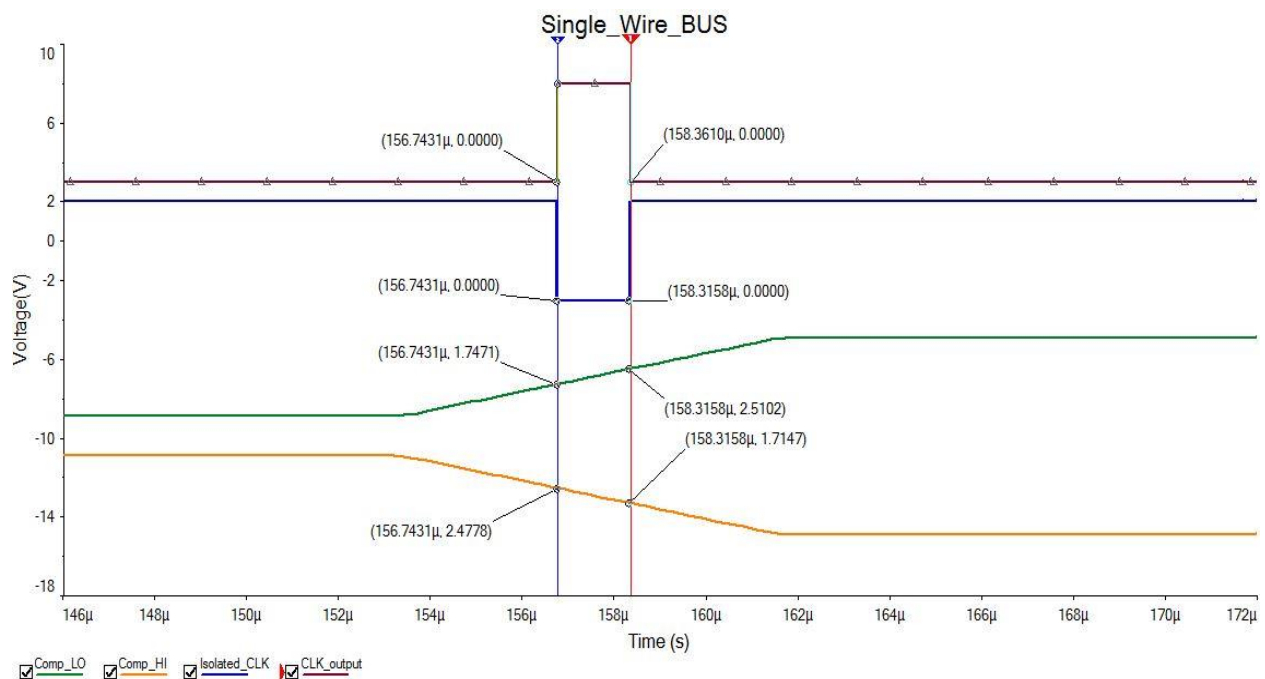


Figure 6.15 Glitch in the isolator due to slow slew rate of the op-amps

From bottom to top: High threshold comparator (yellow), Low threshold comparator (green), Isolated CLK (blue), CLK output (red)

From figure 6.15, it can be observed that the LM324 op-amp has a very slow slew rate with respect to the 74LS86 XOR gates. The slow slew rate of the comparator op-amp causes the XOR gates to generate a false positive signal. The transient response of the XOR gates are much faster than the LM324 op-amp. This means the LM324 has slower rise time. The slow rise time of the op-amp causes the input voltages of the XOR gates to be in the unaccepted region (metastable state) for a brief period of time (156.7431 μs to 158.3158 μs from cursor text in figure 6.15). This is equivalent to the XOR gate inputs being floating. If the gate inputs are floating, depending on the silicon architecture and noises in the surrounding environment, the XOR gate can see the inputs to be of any arbitrary state. In this case, the XOR gate is assuming both its inputs to be of same state, as in both HIGH or both LOW, and thereby falsely signaling the rest of the circuitry that a CLK signal has been received. To remove the glitch, the analog comparator op-amp needs to have, at least,

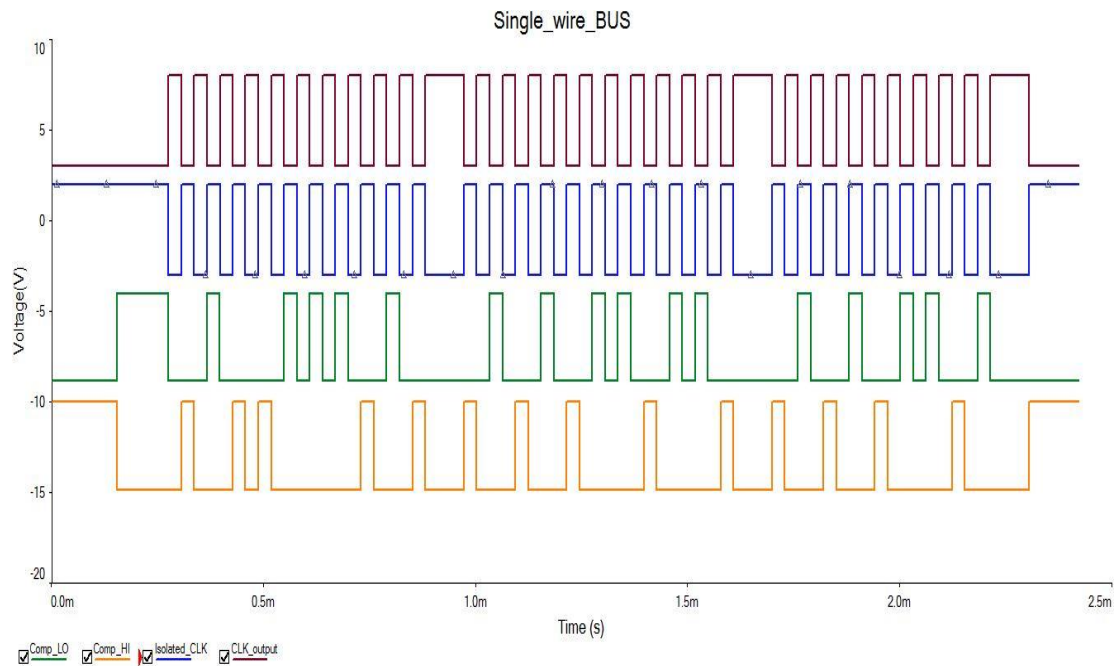


Figure 6.17 Waveform of the updated circuit
 From bottom to top: Output of high (yellow) and low (green) Comparator,
 XOR gate U1A (violet) and extracted CLK from multiplexed signal (dark red)

Table 6.3 shows the complete list of components and instruments used in the simulation.

# SL.	Component Designator	Component Name/Value	Component type	Function
1.	U1A	74LS86	XOR gate	Digital Comparator
2.	U1B	74LS86		Digital buffer and inverter
3.	U1C	74LS86		Digital buffer for data (placed on receiver side)
4.	U3A	LM339N	Special purpose op-amp (comparator)	Comparator for low threshold
5.	U3B	LM339N	General purpose op-amp (comparator)	Comparator for high threshold
6.	Q3	2N2222A	npn transistor	Switch for Data signal
7.	Q2	2N2222A	npn transistor	Switch for Clock signal
8.	R1	4.7 K Ω	Resistor	Pull-up resistor
9.	R2	2.2 K Ω		
10.	R3	2.2 K Ω		
11.	R4	10 K Ω	Resistor	BJT base current limiting resistor
12.	R5	1.47 K Ω	Resistor	CLK level pull-down resistor
13.	R6	10 K Ω	Resistor	BJT base current limiting resistor
14.	XWG1		32-bit pattern generator	Emulates the signals generated by the transmitter
15.	XSC1		4 channel digital oscilloscope	Captures the signals on the circuit
16.	XSC2		4 channel digital oscilloscope	

Table 6.3 Component list for complete module circuit

6.6 Transmitter and receiver behavior model

In the simulation, the test signals for data and clock was generated by using pattern generator and the outputs are shown in oscilloscope. To test the setup in real life condition, a transmitter and a receiver was built using microcontrollers running behavior models to imitate a transmitter and receiver. This section describes the behavior models, their working principles and their shortcomings.

6.6.1 Transmitter

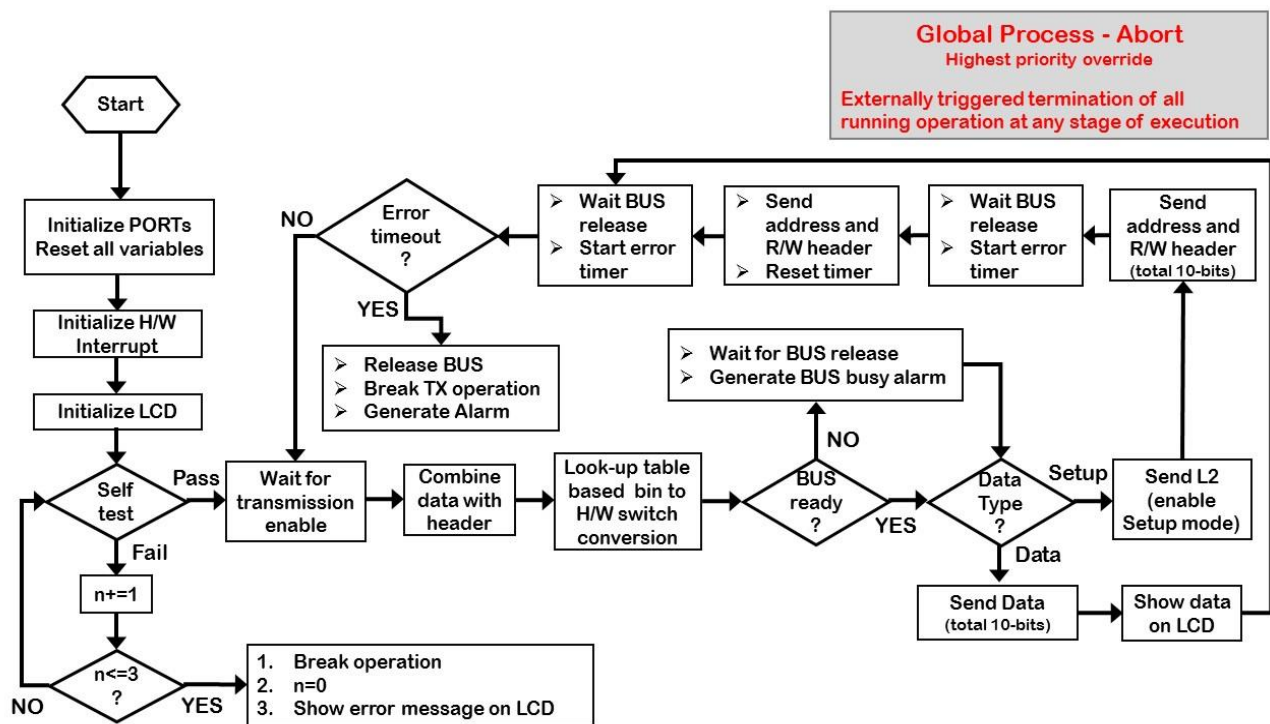


Figure 6.18 Transmitter behavioral model flow diagram

Figure 6.18 shows the flow diagram of the procedure followed by the transmitter. At the beginning, the input and output ports are initialized by programming the direction registers of the microcontroller I/O port to specific values. Next, the built in hardware interrupt of the microcontroller is initialized. To enable a user interface, LCD is used and it is initialized next. To ensure the hardware are all working as it should, a brief self-test function, following a predefined look-up table, is initiated. If the self-test is not passed within total three tries, the whole transmitter

operation is aborted and a fatal error message is shown in the UI. If the self-test is passed, the system waits for arrival of data and signal to enable transmission of corresponding data.

The system also expects the type of to be transmitted data i.e. if the Master is transmitting slave address or Master/Slave transmitting data etc. The transmitter generates the command headers by itself based on the data type. The transmitter itself combines the 8-bit data with the 2-bit command header to form the predefined 10-bit packet.

The data BUS uses signaling protocol that requires two different port pins, one for data and another for clock. The transmitter is therefore required to convert the binary data into port switching control signal and define the number of clock pulses required to transmit the whole 10-bit data. Before transmission, the transmitter checks if the data BUS is in use. If in use, it would wait for BUS to free up and notify about it in the UI. If the BUS is free, the data is transmitted. Before transmission, the transmitter checks the command header to check if it is sending general data or setup data. If setup data is to be sent, first the transmitter initiates the setup mode on all connected devices by pulling down the BUS to CLK for the predefined amount of time to make the slave device(s) enter setup mode. After setup mode is initialized, setup data is sent along with proper command headers.

If, after transmission of any data, the BUSY status is initiated by the slave device, the transmitter considers this as a ACK signal and knows that the slave with corresponding address is present in the BUS and it has received the data correctly. If no ACK is received, the transmitter considers this as error and reports it to the UI. On receiving BUSY status, the transmitter starts its error timer and waits for the BUS to be released. If the error timer times out, the transmitter will consider this as an occurrence of an error on the BUS and therefore will notify the error on UI, clear the memory and break the current operation. If BUS is released before timer times out, it transmits the 10-bit data and repeat the error detection subroutine. This process can go on indefinitely until all data is sent.

Once a slave is addressed, it stays in the specific Master's control until the corresponding Master releases it by sending the corresponding slave address again or try to reach a different slave device.

As long as specific Master device has control over a specific slave, no other device can use the data BUS as Master i.e. one Master device at a time.

6.6.2 Receiver

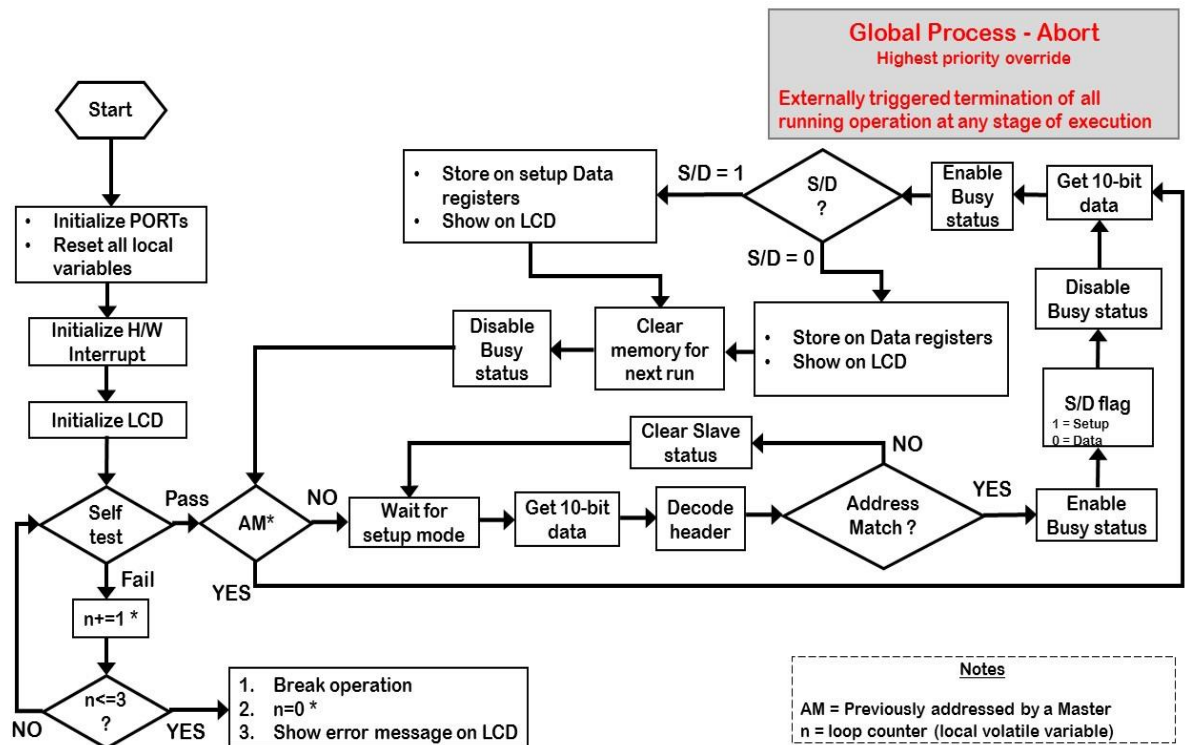


Figure 6.19 Receiver behavioral model flow diagram^{*3}

Figure 6.19 shows the block diagram of the receiver module. Both the transmitter and the receiver module uses the same initialization and self-test procedures. After self-test is passed, the receiver checks if the receiver was formerly addressed any Master device. If it was not addressed before, it will wait for a Master device to guide it into setup mode. After it goes through setup mode, it will expect setup data namely address in this case. From the 10-bit setup data, the receiver extracts the type of setup data from the command header.

Notes:

*3. The flow diagram of the receiver does not include the algorithm used to detect if the master is releasing the receiver either by sending its own address again or by selecting another slave device

After receiving address, the address is matched with the slave's own address. If it is not a match, the slave device clears its memory and reverts back to waiting for setup mode to be initiated. If it is a match, the receiver pulls the data BUS to VEE to initiate BUSY status followed by setting up the flag (S/D flag from figure 6.19) for setup data or normal data. The data BUS is then released and the receiver expects 10-bit of data from the master transmitter. After receiving the 10-bit data, the data BUS is again pulled down to VEE to signal BUSY status.

Based on the S/D flag setup before, the receiver stores the data either on setup data memory or on general data memory. The setup data memory can be used to configure the receiver itself. The general data memory is directly given to the digital system connected to the receiver (i.e. the user of the slave device). All memory is local (i.e. contained within the receiver) and volatile. Before next run, all local memory is cleared. If the device was addressed before, the receiver sends BUSY status, directly stores the data on general data memory that can be accessed by the digital system connected to the receiver, clear memory for next run and remove the BUSY status showing transmitter that it is ready for next run.

Chapter – 7

Experimental Hardware Emulation and Results

In order to verify the viability of the proposed data BUS, circuits were designed and simulated on computer based simulator. The computer simulator worked exactly as was expected and exceeded expectation by at least 50 times by giving out flawless result up to a CLK frequency of 800 KHz. However, computer simulations assume the ideal world conditions and do not take into account any of the numerous variable present in the real world situations that can affect the simulation result in numerous different ways. To verify the proposed data BUS and all the related circuits, a protoboard model of all the circuits is constructed the resulting signals are viewed on a digital storage oscilloscope.

The test circuit is identical to the circuit used in the simulation. The hardware emulation is setup to emulate a completely bi-directional half duplex single wire BUS. Therefore, a number of changes have been made in the actual hardware setup. On the transmitter side, pattern generator is replaced with a microcontroller that has the algorithm for generating the same switching signals that is generated by the pattern generator during the simulation. The transmitter algorithm is explained in section 6.6.1. On the receiver side, in the simulation, signals were captured to be seen on oscilloscope. On the hardware emulation, a receiver is made using microcontroller running the algorithm explained in the section 6.6.2.

A complete proposed single wire module consists of a transmitter, a receiver and a microcontroller having the algorithm to enable communication as both a transmitter and a receiver. In order to emulate a bi-direction data communication, at least two modules are needed, one acting as slave and another being the master. The schematic for a single module is shown in the Appendix-C.

The complete component list for the protoboard model for both transmitter and receiver is shown in table 7.1. All the instruments, simulation and CAD packages used throughout the research is tabulated in the table 7.2.

# SL.	Component Name/Value	Component type	Function
1.	AtMega32A	8-bit microcontroller	Holds and executes behavior models to emulate a transmitter and a receiver
2.	74LS86	XOR gate	<ul style="list-style-type: none"> • Digital buffer • Digital Comparator
3.	LM339N	General purpose op-amp (comparator)	Analog comparator
4.	2N2222A	npn transistor	Electronic switch
5.	5% 250mW carbon film resistors of different values	Fixed resistor	<ul style="list-style-type: none"> • BUS Pull up • Current limiting • Digital State
6.	1% 250mW wire wound 10 turn variable resistor	Variable resistor	<ul style="list-style-type: none"> • Variable precision voltage reference via resistive divider • Resistance adjustment
7.	HD 44780 based LCD	20x4 Alphanumeric LCD	Part of user interface
8.	Push button	Momentary Switch	Part of user interface

Table 7.1 Component list for protoboard hardware emulator

# SL.	Name	Vendor	Type	Description
1.	Multisim 14.0	National Instruments	PC based Simulator	Performs all signal simulation
2.	CodeVisionAVR	HP InfoTech S.R.L	AVR code Compiler	Compiles C language AVR program to .Hex for AVR MCU
3.	Altium Designer	Altium Limited	CAD tool	Schematic design tool
4.	AVR development board	Generic	AVR MCU based development environment	Development board contains the AVR MCU along with all support components needed by the MCU so that the MCU can be used with least amount of setup time and effort
5.	UTD2052CL	Uni-Trend	2 channel DSO	Captures the signals from the prototype BUS
6.	DC battery	Generic	DC 5v supply	Provides DC 5v to the prototype circuits

Table 7.2 List of instruments, simulation package and CAD packages used in the research

7.1 Experimental Setup result

Proposed data BUS signal from the hardware setup is captured in the digital storage oscilloscope and the result is presented below. Figure 7.1 to figure 7.9 shows different signaling stages for the proposed data BUS in real time. All the signals are capture by the single shot trigger of the DSO with trigger voltage of 0.2v. All the figures (7.1 to 7.9) are screenshots captured and exported on a USB flash drive by the DSO itself. No external camera or optical device has been used to capture any of the screenshots shown in the figures from figure 7.1 to figure 7.9.

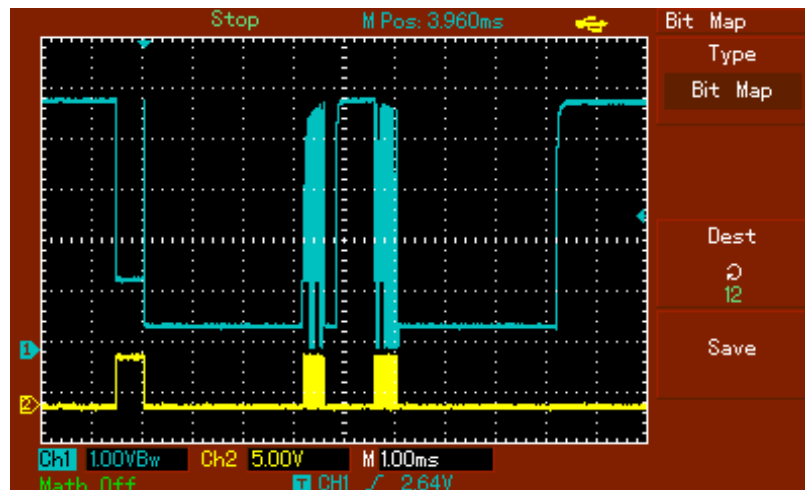


Figure 7.1 Initialization of setup mode followed by transmission of slave address and one data packet (BUS & CLK, blue and yellow respectively)

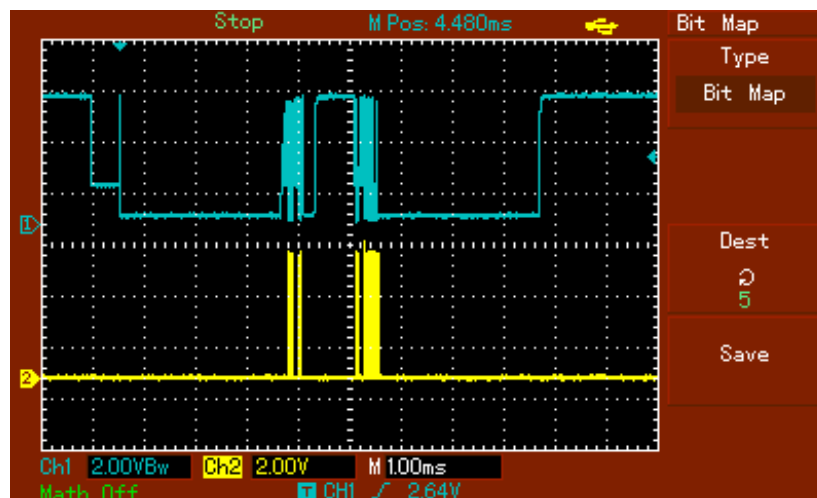


Figure 7.2 Initialization of setup mode followed by transmission of slave address and one data packet (BUS & data, blue and yellow respectively)

Referring to figure 7.1 and 7.2, both figure shows the initialization of the slave by setup mode followed by 8-bit address and 2-bit header. A 10 bit general data is sent after the slave has been initialized. The multiplexed signal is shown is blue trace (upper trace) and the transistor input for CLK (figure 7.1) and data (figure 7.2) is shown in blue (lower trace). From figure 7.1 and/or 7.2, notice that the initial state of the BUS is pull-up to VCC (5v). The clock line is activated (figure 7.1, yellow trace) by master for a specific time to signal setup mode. The clock line release is denoted by a spike immediately after the CLK level. The spike is caused by the brief duration the

BUS stays pulled up before being pulled down to VEE/GND by the receiver. From figure 7.1 and 7.2, notice both data and CLK transistors are off. The pulling down signifies that there is at least one slave device present on the BUS. After the BUS is released by the slave/receiver, 8-bit address is sent by master. The BUS is again pulled down to VEE/GND by the slave/receiver to signify that the device of that specific address is present on the BUS. After the BUS is released, the master device is free to send as much 10-bit packet as it wishes. In this case, the master waits for about 1ms before sending in 10-bit packet of general data.

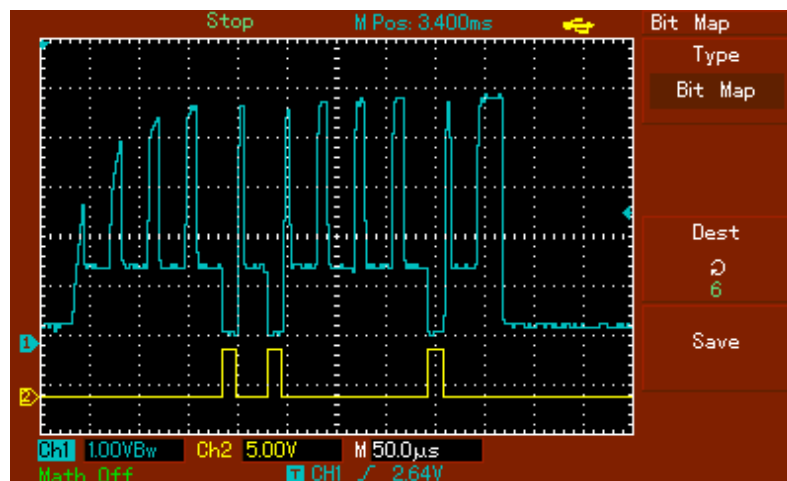


Figure 7.3 Transmission of address and header (BUS & CLK, blue and yellow respectively)

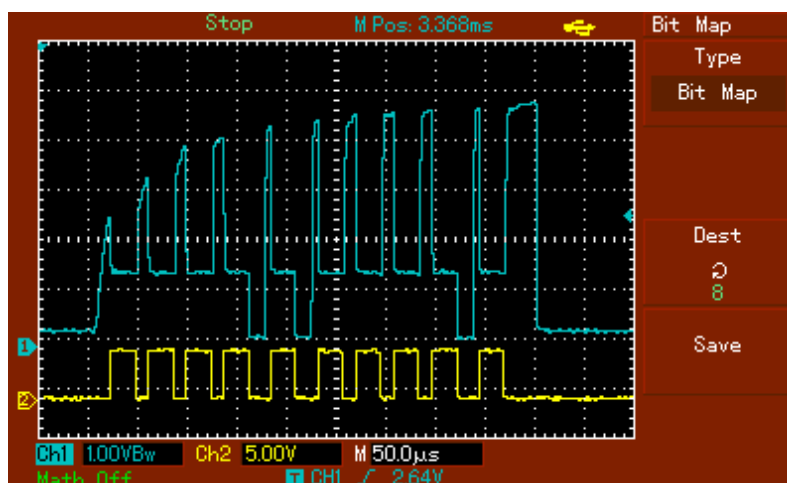


Figure 7.4 Transmission of data and header (BUS & CLK, blue and yellow respectively)

Figure 7.3 and 7.4 shows the detailed signals for 8-bit address and 2-bit header. The master is sending 11100111 address and 01 command header. From the left, the clock is transmitted by pulling the BUS at CLK level and immediately after that, the BUS is pulled up to VCC to signify digital 1 (HIGH). After 4th spike, the BUS transmits a digital 0 (LOW) by pulling BUS to VEE right after the CLK. At this point, a limitation of the hardware emulator circuit takes hold. Notice the voltage spike going up to VCC before the next clock signal is transmitted. This voltage spike is caused by the fact that the BUS needs to be released for a small time after transmitting LOW before next CLK can be sent. Failure to release the BUS like this causes the receiver to misbehave. As, unlike HIGH, LOW needs the BUS to be released for a specific amount of time before next CLK, the setup time of LOW becomes higher than the HIGH causing the emulator circuit to have an asymmetric setup time for LOW and HIGH. This also slows down the BUS. This phenomenon dictates the best case and the worst case of the hardware emulator. The best case is when all the bits, including command header, are 1 as no extra time is needed for transmitting HIGH. The worst case, therefore, is when all the bits are 0, including command header, as LOW requires extra time for receiver recovery.

Figure 7.5 and 7.6 shows the transmission of general data of 00110000 and 00 as header, a complete 10-bit data of 0011000000. The figure 7.5 and 7.6 are not explained further as at this point it is self-explanatory.

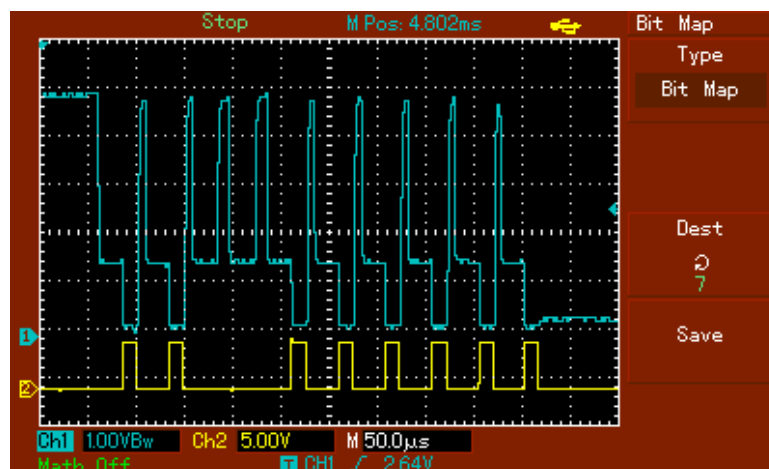


Figure 7.5 Transmission of data and header (BUS & data, blue and yellow respectively)

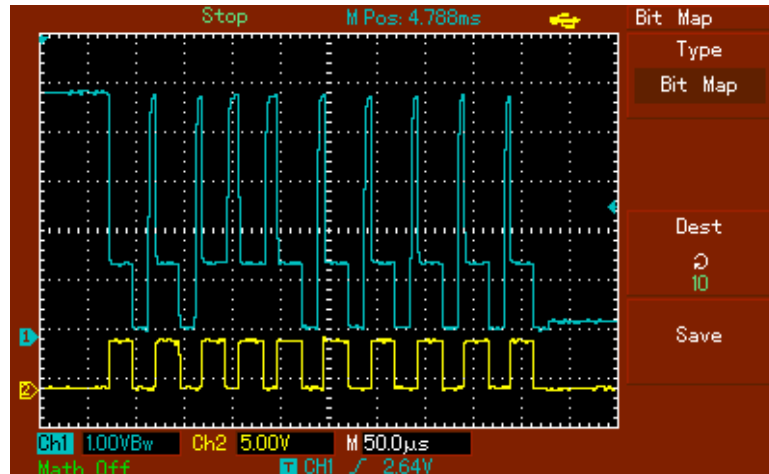


Figure 7.6 Transmission of data and header (BUS & data, blue and yellow respectively)

Figure 7.7, 7.8 and 7.9 shows the state of data (yellow, upper trace) and CLK (blue, lower trace) driver transistor for the whole operation (BUS is omitted). The BUS is transmitting the same bits as earlier for both address and general data. Figure 7.8 shows the whole operation and others show detailed information on each packets.

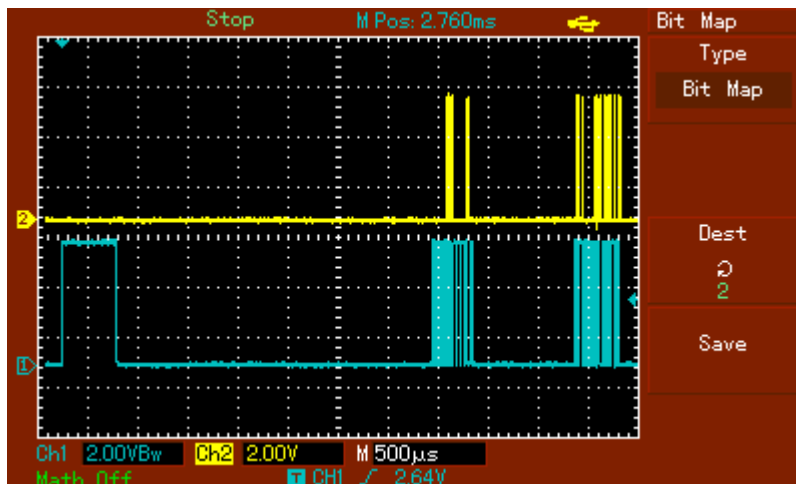


Figure 7.7 Transistor switching pattern for initialization of setup mode followed by transmission of slave address and one data packet (data & CLK, yellow and blue respectively)

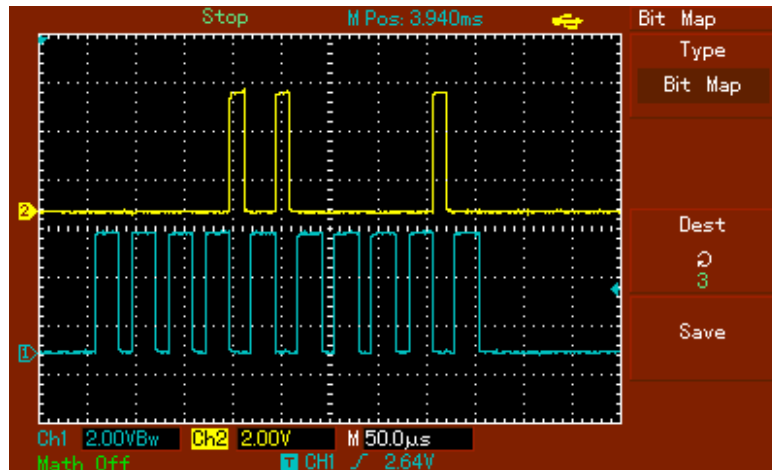


Figure 7.8 Transistor switching pattern for transmission of slave address and command header (data & CLK, yellow and blue respectively)

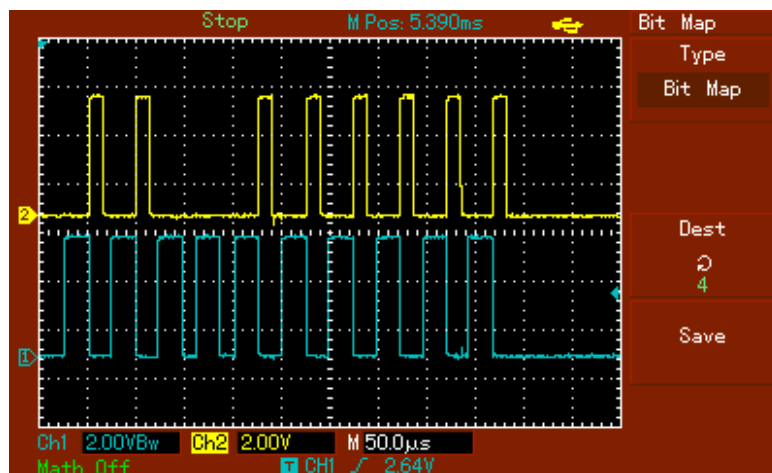


Figure 7.9 Transistor switching pattern for transmission of data and command header (data & CLK, yellow and blue respectively)

If only data transmission is considered (leaving the setup mode out of calculation), the final speed of the proposed single wire BUS has been found to be exceeding the set goal. Due to limitations of the microcontroller and possibly unintended imperfections of the C code for the behavioral mode, the system has been found to have asymmetric setup time for digital 1 (HIGH) and digital 0 (LOW). Combining the complete time required to successfully transmit clock and HIGH is 37 microseconds and for clock and digital LOW, the figure increases to 46 microseconds. Therefore, in order to transmit a 10-bit packet, total required time can range anything from 370 microseconds to 460 microseconds. 370 microseconds are for the packet containing all 1's and 460 microseconds is for all 0's.

As mentioned earlier, the setup time for transmitting HIGH and LOW, along with clock, is asymmetrical with LOW having higher setup time than HIGH. Therefore, it would be logical to set the definition for BUS worse case as the timer when the BUS has to carry a 10-bit data of all of all zeros (0b0000000000). Logic also dictates that the best case would therefore would be when the BUS is carrying all one (0b1111111111). Following this definition, the worst case for this data BUS is 460 microseconds and the best case is 370 microseconds.

For hardware emulation, the data BUS uses 7 I/O pins on the microcontroller to complete its intended operations. In the current test setup, 7 I/O pins beats the purpose of original motivation behind this thesis work i.e. reduce the number of I/O pin usage. The data BUS can become true single wire transmitter and receiver only when the individual modules are designed in silicon and integrated with the microcontroller or other processor, FPGA, ASIC or SOC. If integrated, this data BUS can be enabled/disabled by the discretion of the user of the microcontroller and will provide great flexibility in terms of performance, reduced number of pins on the IC and development time of the application the MCU is being used in. In VLSI design stage, the number of transistors are always minimized to reduce real state usage on the silicon wafer. As the related circuits for the proposed data BUS are simple to begin with, a complete module for TX and RX will probably take very little area on silicon wafer which will ensure maximize performance at low price.

The following Table 7.2 performs a side by side technical comparison of the proposed single wire BUS with two of the most widely used industry single wire data BUS available in the market namely 1-wire protocol by Maxim Integrated and UNI/O by Maxim Integrated. It must be noted that although all these BUS are similar in their purpose, each use different methodology to get the job done. Therefore, the comparison between some parameters might be inconclusive/incomplete and the total comparison leans toward comparison of bit rates and other relevant electrical specifications and performances.

<u>Attributes</u>	<u>Name of Data BUS</u>		
	1-wire ^{[12] [16]}	UNI/O ^[2]	Proposed data bus
Technology vendor	Maxim Integrated	Microchip Technology Incorporate	Early stage of academic research
Signaling Protocol	Delay based state definition	Manchester encoding	Modified 5v TTL standard
BUS type	Serial	Serial	Serial
Standard bit rate	16 Kbits/sec (approx.)	16 Kbits/sec (approx.)	21 Kbits/sec to 27 Kbits/sec (approx.)
Signal type	Digital	Digital	Digital
Physical signaling	Single ended only	Single ended only	Single ended only
Range	Can be extended to form a network outside of PCB	Short	Short
BUS direction	Half duplex (1 way at a time)	Half duplex (1 way at a time)	Half duplex (1 way at a time)
Device configuration	Master Slave system	Master Slave system	Master Slave system
Parasitic power	Yes	No	Further research needed
Bits per frame		10	10
Maximum number of devices on BUS		Depends on device family (range from 8 to 12bits)	256

Table 7.3 Comparison between existing industry standard single wire BUS against proposed single wire BUS

Chapter – 8

Conclusion

Unless the whole system is contained within specifically designed ASIC, SOC or FPGA, usage of data BUS is inevitable in any embedded systems no matter size or complexity. Majority of the electronic systems in the market are designed in modular form. An example could be a modern laboratory programmable benchtop power supply in which the DAC of the regulator board of a programmable power supply generally use SPI^[27]. Generally, the CPU connects to the DAC using SPI BUS. The display driver, the UI handling subsystem or other components/subsystems can all be connected to the same SPI BUS. As everything is designed in modular form, in case of failure of any module, only the failed module needs to be replaced in order to make the power supply active again. This makes repair very easy, less expensive and less time consuming. However, the SPI BUS needs to be connected to all the modules. The number of copper traces on PCB or the number of wires on the ribbon cable can be reduced if the BUS uses less physical connections. Furthermore, lower the number of physical connections, lower is the chance of probable physical fault points and lower is effect and generation EMI and EMC, meaning, less shielding can be used. The proposed single wire BUS can connect to 256 devices, requires only one wire and in experiment has reached a data bit rate of at least 21Kbits/s, which is good enough to configure DAC or read from ADC or update any front panel display at regular interval in any commercial grade consumer application. This BUS can offer electronics design engineers flexible options, which are not limited to

- Reduction of the number of copper trace on the PCB, allowing easier design of PCB
- Reduction of source of EMI/EMC from the produce as only one wire is carrying signal
- Reduction of the size or number of wires on board to board interconnects or ribbon cable respectively and thereby reducing the number of probable fault point
- Usage of shielded cables at a lower cost while lowering effect and generation of EMI and EMC respectively and increasing noise susceptibility
- Reduction of final cost of product as interconnect/ribbon cable size is decreased which also enables the PCB to be smaller

8.1 Thesis Summary

A new single wire data BUS for embedded systems is researched in this thesis work. The whole research is based on a new method for multiplexing data and clock on a single signal to form a non-conventional signal shape. In the research, the shape of the physical wave is first defined followed by the decision to make the BUS an open collector system. Electronics to multiplex both the clock and data to form the intended wave shape is designed and each part of the hardware is tested in the simulator. Next the hardware for demultiplexing clock and data from the compound signal is designed and the circuit is extensively tested. The target of the thesis was to get a transfer 10Kbits/second, however, simulation model exceeded the target by approximately 2.5 times without degrading the signal. Although a glitch in the clock demultiplexing circuit (isolator) was faced, the problem was fixed by using components of higher slew rate and bandwidth.

The computer simulation model was later assembled in the protoboard for a real world simulation. The system was found to have a best case of data transfer rate of approx. 27Kbits/s (ignoring setup mode). The protoboard model, however, faced the problem of having asymmetrical setup time for data HIGH and LOW. This asymmetry was later found to be caused by the limitation of the microcontroller used to emulate the transmitter and receiver. The limitation of the microcontroller also limited the maximum data transfer rate to approx. 27Kbits/s.

Considering the specifications set for the proposed data BUS, it is observed that both the computer simulation and the real life hardware simulation has fulfilled the set goal(s) and in some case exceeded the goal(s) and expectation.

In the end, it must be pointed out that this research only explores the possibility of using a new non-conventional signal waveform shape and a new protocol to successfully communicate data between multiple digital systems. In no way it is claimed that this new method is complete as it is and is fit to be used in any industrial and/or commercial purposes. Before deploying in large scale market of mass production, the rough edges must be sanded down and more research must be conducted to assess the performance and viability of the proposed data BUS in wide range of probable environment and application.

8.2 Present problems solved by the proposed data BUS

This thesis research compares the proposed data BUS with two industry standard single wire data BUS which have been in the market for an extended period of time. These data BUS, by no mean, are free from various limiting constraints. The proposed data BUS tries to solve some of the limitation. The limitations and their solution in the proposed BUS are explained in this section.

1-wire protocol from Maxim Integrated uses delay to define various digital states and transmission of control and acknowledgement signals. As delay is used to communicate between devices, it becomes difficult to make the BUS faster. Even if the delays are small, there should be a reasonable gap between the delays for specific commands to have a satisfactory margin of error.

On the other hand, UNI/O from Microchip Technology Incorporate communicates data using Manchester encoding. This particular encoding method requires a fixed time frame and the direction of transition of signal in the middle of a specific frame shows the specific digital state of the signal of that specific time frame. To specify the timing of the time frame, the transmitter has to send a specific code at the beginning of transmission. Using Manchester encoding requires specialized electronic circuits that can detect the direction of signal transition at the middle of the time frame. This poses as a limitation as, even though high speed electronics are now available, nothing can be faster than the receiver receiving only 1/0 without worrying about the transition at the middle of a time frame.

A synchronous BUS will most definitely have some way of defining CLK and hence the name synchronous. In case of both these BUS (1-Wire & UNI/O), the CLK is defined by either the rising or falling edge of the signal. In multi wired BUS, CLK has a complete physical wire dedicated to it. In single wire data BUS, unless there is some type of level transition in the signal, it is impossible to multiplex a CLK signal with data as both of these aforementioned data BUS use signal level transition as CLK. If the same digital state persists for multiple sequential bits, the receiver will not receive a CLK and therefore will not sample the data and the total data will either be corrupt or will have missing bits. Figure 8.1 summarizes the idea.

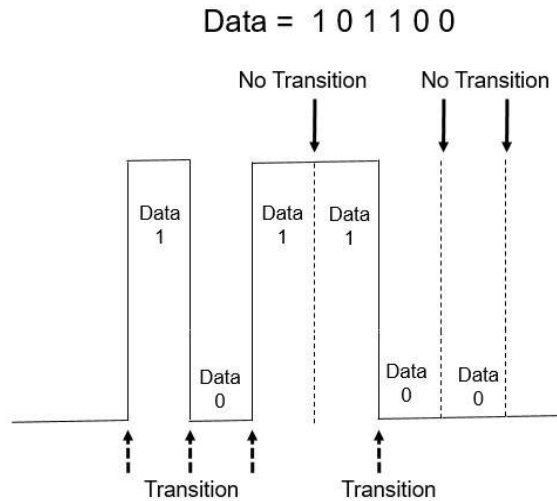


Figure 8.1 Signal level transition

The proposed data BUS introduces a new state to transmit CLK. CLK is transmitted after every data bit. That means even if the data consists of same digital state for multiple sequential bits, they are sampled correctly by the receiver as CLK has its own state and is not defined by data signal transition. As specific length of time frame is not required like Manchester encoding in UNI/O, the multiplexing and demultiplexing circuits can be simpler. Furthermore, the proposed data BUS can be made faster as the BUS is not using delay based state definition like 1-wire protocol or fixed time frame like UNI/O.

Current silicon fabrication technology allows fabrication of very fast transistors which are also very small in area. This allows design and fabrication of small but fast signal transmitter and receiver. The multiplexing and demultiplexing circuit for the proposed BUS consists of simple circuits and therefore can be easily integrated into MCU and other monolithic chip or SOC for maximum efficiency.

8.3 Probable future development(s)

Although the prototype exceeded the target of this thesis, the data BUS has the possibility to do even better. The future research on this BUS protocol can be directed toward achieving the following goal(s).

1. 8-bit microcontroller has been used to emulate transmitter and receiver module. The microcontroller can be replaced with a FPGA to get significantly data bit rate ^{[8] [15]}. Also the codes for microcontrollers are written in C language. A hardware description language ^{[8] [15]} like Verilog can offer an orders of magnitude increase in efficiency and degree of control.
2. The hardware is tested with only jellybean off the shelf components. The circuit could be remade with proper specialized components for increased performance.
3. Both the transmitter and receiver module can be redesigned using discrete components only. That way total setup time can be possibly reduced as the device is not being processed by a CPU of complex architecture (i.e. in case of Timer-1/2 & TX/RX MCU).
4. The transmitter, the receiver and all of the related components can be designed to form a silicon chip. A design based completely on silicon can be significantly smaller in size and will reduce the component (transistor) count. The resulting device will be faster and more efficient as well.
5. Present setup requires both transmitter and receiver to be powered by external source while sharing a common electrical signal return reference. Attempts can be made to power the slave device from the Master via the data BUS itself (parasitic power) as done in 1-Wire protocol.
6. Effect of EMI, EMC ^[22], attenuation ^[5] and noise requires to be researched if the data BUS is to be used in commercial/industrial setting as signal corruption is a very real threat and more so when gaps between consecutive states are small.
7. Building on 5th point, emission (EMC and EMI) by the data BUS is also a critical topic to be researched as excessive emission may cause finished product to fail either/both near field or far field compliance test ^{[3] [12] [25] [26]}.

References

- [1] Falk Aliche, Matthias Feulner, Frank Dehmelt, Ankur Verma, and G. Becke
Comparing Bus Solutions, Application Report SLLA067C, March 2000–Revised
August 2017, Texas Instruments Incorporated
- [2] “UNI/O Bus Specification”, 2009, Microchip Technology Inc.
- [3] “Electromagnetic Interference/Compatibility (EMI/EMC) Control Test and
Measurement Facility, User Test Planning Guide”, National Aeronautics and Space
Administration
- [4] M. Rafiquzzaman, “Microprocessors Theory and Application Intel and Motorola”,
Revised edition 2012, PHI Learning Privet Limited
- [5] “Basics in EMC / EMI and Power Quality, introduction, annotations, applications”,
2013, Schaffner Group
- [6] Thomas L. Floyd, “Electronic Devices”, 7th edition, 2009, Pearson Education
- [7] “RS-232, RS-422, RS-485 Serial Communication General Concepts”, White paper
11390, Apr 17th 2018, National Instruments
- [8] Nazeih M. Botros, “HDL Programming VHDL and Verilog”, 2009, Dreamtech Press
- [9] Thomas L. Floyd, “Digital fundamentals”, 10th edition, 2011, Pearson Education
- [10] Thomas Kugelstadt, “The RS-485 Design Guide”, Application Report SLLA272C,
February 2008–Revised October 2016, Texas Instruments Incorporated
- [11] Manny Soltero, Jing Zhang, Chris Cockril, Kevin Zhang, Clark Kinnaird, and
Thomas Kugelstadt, “RS-422 and RS-485 Standards Overview and System
Configurations”, Application Report SLLA070D, June 2002–Revised May 2010,
Texas Instruments Incorporated
- [12] “1-Wire Enumeration”, Application Report SPMA057C, August 2013–Revised
January 2018, Texas Instruments Incorporated
- [13] “I2C Guide 2013”, Application Report SSZC003a, 2013, Texas Instruments
Incorporated
- [14] Bernhard Linke, “Choosing the Right 1-Wire Master for Embedded Applications”,
Reference Design 4206, Mar 27, 2008, Maxim Integrated
- [15] M. Morris Mano, Michael D. Ciletti, “Digital Design”, 4th edition, 2012, Pearson
Education

- [16] Sashavalli Maniyar, “1-Wire Communication with PIC Microcontroller”, Application note AN1199, 2008, Microchip Technology Inc.
- [17] Jean-Marc Irazabal, Steve Blozis “I²C MANUAL” Application note AN10216-01, March 24th 2003, Philips Semiconductors
- [18] Albert Malvino, David J Bates, “Electronic Principles”, 7th edition, 2010, Tata McGraw Hill Private Limited
- [19] ATmega32A datasheet, Microchip Technology Inc.
- [20] Ramakant A. Gayakwad, “Op-Amps and Linear Integrated Circuits”, 4th edition, 2012, PHI Learning Privet Limited
- [21] “Making Conducted and Radiated Emissions Measurements”, Keysight Technologies
- [22] “Electromagnetic Compliance: Troubleshooting with Near-Field and Current Probes”, October 2017, Siglent Technologies
- [23] Muhammad Ali Mazidi, Sarmad Naimi, Sephr Naimi, “The AVR microcontroller and embedded systems using assembly and C”, 2013, Pearson Education
- [24] “Guidance for users of the CISPR Standards”, International Electrotechnical Commission
- [25] Timothy Hegarty, “An overview of radiated EMI specifications for power supplies”, Application Report SLYY142, June 2018, Texas Instruments Incorporated
- [26] Timothy Hegarty, “An overview of conducted EMI specifications for power supplies”, Application Report SLYY136, February 2018, Texas Instruments Incorporated
- [27] “KeyStone Architecture Serial Peripheral Interface (SPI) User Guide”, Literature Number SPRUGP2A, March 2012, Texas Instruments Incorporated
- [28] “Guidelines for reliable long 1-wire network”, Application note 148, 2014, Maxim Integrated
- [29] B.P. Lathi, Zhi Ding, “Modern digital and analog communication systems”, International revised 4th edition, Oxford University Press

Appendix

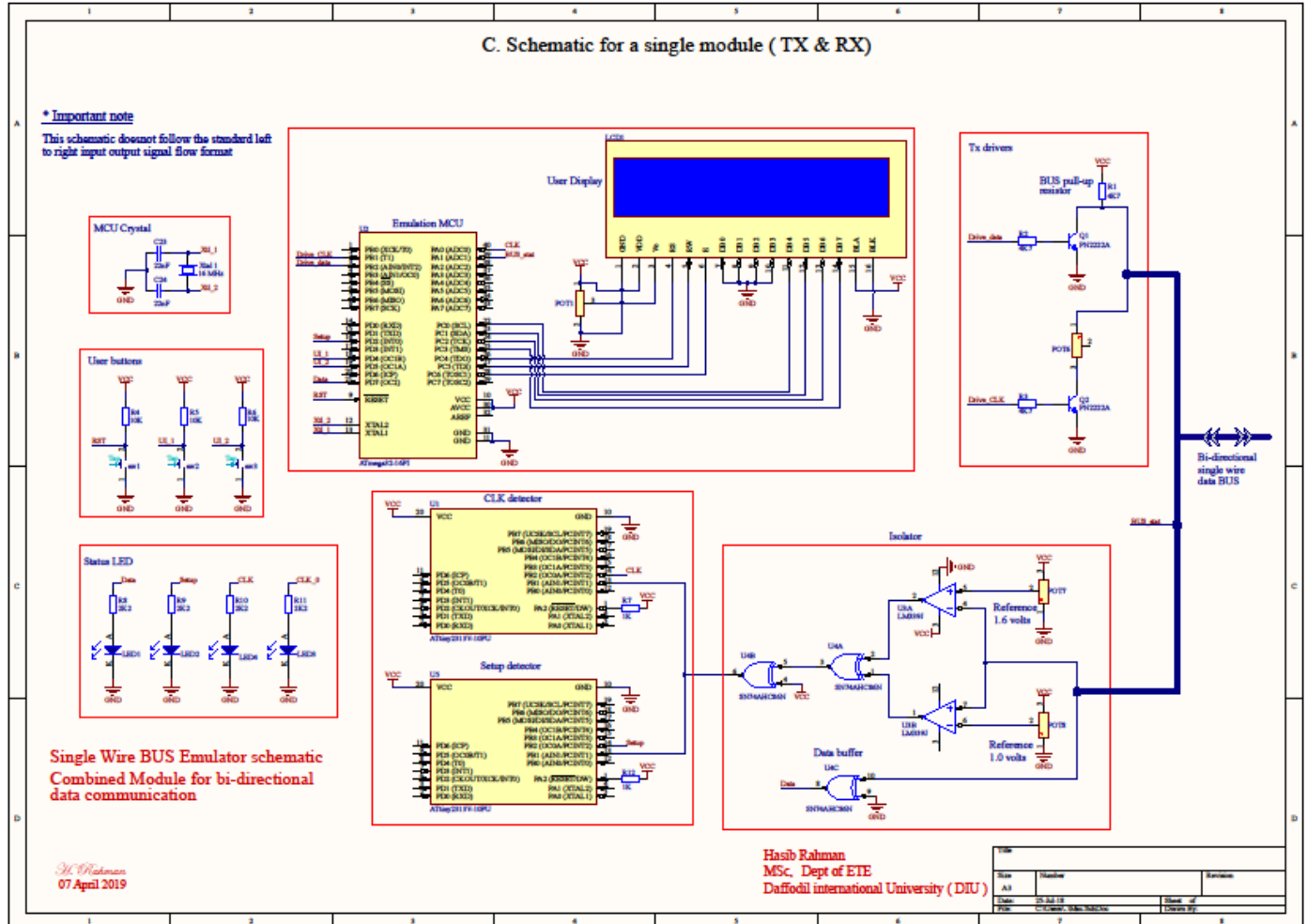
A. LIST OF ABBREVIATIONS

Abbreviation	Full form
ASIC	Application Specific Integrated Circuit
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
CMOS	Complementary Metal Oxide Semiconductor
TTL	Transistor Transistor Logic
LED	Light Emitting Diode
CLK	Clock
CPU	Central Processing Unit
LSB	Least Significant Bit
MSB	Most Significant Bit
MCU	Microcontroller Unit
FPGA	Field Programmable Gate Array
SOC	System On-chip
USART	Universal Synchronous Asynchronous Receiver Transmitter
UART	Universal Asynchronous Receiver Transmitter
IIC/I2C	Inter Integrated Circuit
SPI	Serial Peripheral Interface
EMI	Electromagnetic Interference
EMC	Electromagnetic Crosstalk
SCL	Serial Clock
SDA	Serial Data
HDMI	High Definition Multimedia Interface
USB	Universal Serial BUS
PC	Personal Computer
LCD	Liquid Crystal Display
EEPROM	Electrically Erasable Programmable Read Only Memory
PWM	Pulse Width Modulation
EIA/TIA	Electronic Industries Alliance/Telecommunication Industries Alliance
PCB	Printed Circuit Board
I/O	Input/Output
ACK	Acknowledge
NACK	Not Acknowledge
HDL	Hardware Description Language
XOR	Exclusive OR
DSO	Digital Storage Oscilloscope
CISPR	The Comité International Spécial des Perturbation Radioélectriques
VLSI	Very Large Scale Integration

B. LIST OF DESIGNATORS USED IN THE SCHEMATIC

Designator	Meaning
U	Integrated Circuit (IC)
Q	Transistor
R	Resistor
C	Capacitor
V	Voltage Source
LS	Low power Schottky (a family of digital IC)
PR	Probe Point
Ref_low/Ref_high	Reference voltages for analog comparator
BUS	Common highway for data communication
HIGH	Digital state where input voltage is higher than set threshold
LOW	Digital state where input voltage is lower than set threshold
VDD /VCC	Power Supply (+ 5 volts)
VEE /GND	Power Supply (Common reference voltage)

C. SCHEMATIC FOR HARDWARE EMULATION



D. EMULATION HARDWARE SETUP

