1.1 BACKGROUND

Successful communication can only be achieved if it follows certain set of predefined rules called as protocol. They work together to help people communicate successfully. The need for protocols also applies to computing systems. Engineers have written rules for communication that must be strictly followed for successful host-to-host communication. These rules apply to different layers of sophistication such as which physical connections to use, how hosts listen, how to interrupt, how to terminate communications, which language to use and many others.

In recent years, the evolution of embedded systems has been shown dominating trends toward application- specific, single chip solutions of practical systems like communication network/s. The ARM processor is one of the leading RISC processor architectures in the embedded domain. The work involves study and evaluating the functionality and performance of ARM7 for multiprotocol transmitter-receiver. ARM (RISC) architecture has a performance advantage that can be utilized in implementing various communication network/s.

1.2 RELEVANCE

Many systems exist in the world, often with different hardware and software specifications. User having a system often needs to communicate with another system supporting different protocol/s. The 'system' is considered in generic sense; it may be an embedded system like digital camera or general purpose system like PC.

Consider a system supporting I2C protocol needs to communicate with another system supporting SPI protocol. In such a situation there must be a transmitter-receiver system which will accept data from I2C protocol and transmit the same over SPI protocol. Or there can be a need of system which accepts real time analog input, convert into digital format and send it over different protocols, then to receive the digitized information over particular protocol and convert back into analog format.

Thus, "Multiprotocol Transmitter-Receiver" plays important role in building a network of systems supporting different protocols. It will also help bridging the gap where analog information is required to send over communication protocol/s and retrieved back into original format in real time.

1.3 PROPOSED SYSTEM AND IT'S FEATURES

Fig 1.1 shows block diagram of proposed system. Information source can be analog such as electrical signal from microphone or digital such as binary data stream from computer. The transmitter would convert analog information into digital format. Then protocol would be selected based upon user´s selection. Information will be transmitted over selected protocol. Receiver will receive the information over particular protocol. Digitized information will then be converted back into analog format. Keypad will be used at transmitter end to select the protocol. User interface is provided via LCD showing the protocol being used.

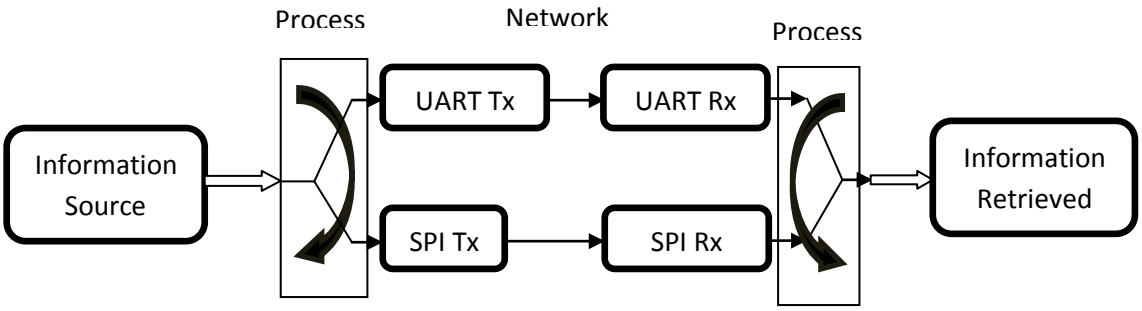

Fig 1.1 Basic Block Diagram of Proposed System

Following are the key features of Multiprotocol Transmitter-Receiver

1. Two systems with different protocols can communicate without hardware and/or software changes.
2. Multicasting and/or broadcasting of information over multiple protocols is possible.
3. Protocol selection facility provided to the user.
4. System can access information easily from variety of sources (both analog and digital).
5. ARM family provides high performance ensuring communication compatibility with almost all existing embedded systems.
6. All commonly used protocols are supported. – UART, SPI, I2C
7. User friendly system with small size, low weight.
8. Easy provision of hardware plug-ins for each protocol.
9. Optimal user interfacing viz. LCD, Keyboard etc.

10. Low Power Consumption achieved by using LPC 2148.

11. Two possible modes: - Battery operated and AC Mains driven.

1.4 ORGANISATION OF REPORT

After first chapter, this section includes brief introduction of each chapter.

Chapter 2 (Literature Survey) contains a brief theory regarding various protocols used in the project along with some key features of ARM7TDMI family. It also includes current market review.

Chapter 3 (Experimentation) contains detailed description about system, such as system block diagram, its components with features, technical specifications and design aspects along with the software algorithms.

Chapter 4 (Conclusion and References) contains conclusions obtained after developing the idea of project. It also contains list of various references.

The related literature of the project must be reviewed before proceeding towards implementation of project as it provides an optimal direction to the approach of system implementation.

2.1 INTRODUCTION

This chapter provides some useful background theory of various protocols along with information about ARM7TDMI family of microprocessors. The reader will get enough background information which will help to understand the further chapters in this report.

2.2 COMMUNICATION PROTOCOLS

2.2.1 I2C

As per [1], I²C (Inter-Integrated Circuit) is a multi-master serial single-ended computer bus invented by Philips that is used to attach low-speed peripherals to a motherboard, embedded system, or cell phone. It's a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer.

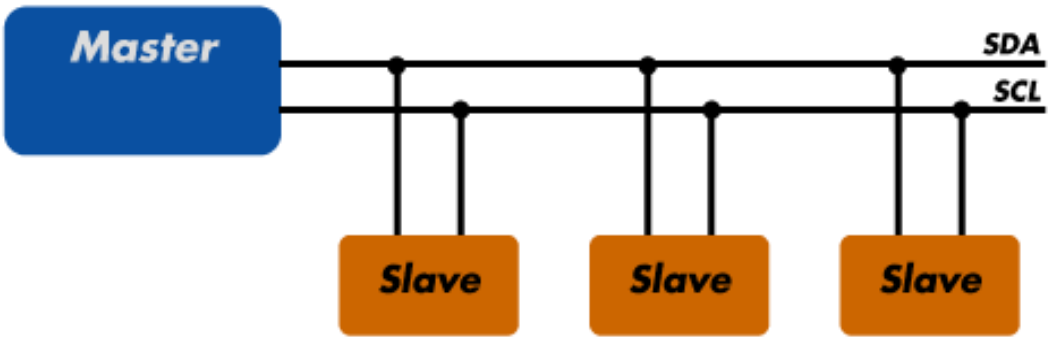

Fig 2.1 Block diagram of an I2C configuration

I²C uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V although systems with other voltages are permitted. The I²C reference design has a 7-bit address space with 16 reserved addresses, so a maximum of 112 nodes can communicate on the same bus. Common I²C bus speeds are the 100 kb/s standard

mode and the 10 kb/s low-speed mode, but arbitrarily low clock frequencies are also allowed.

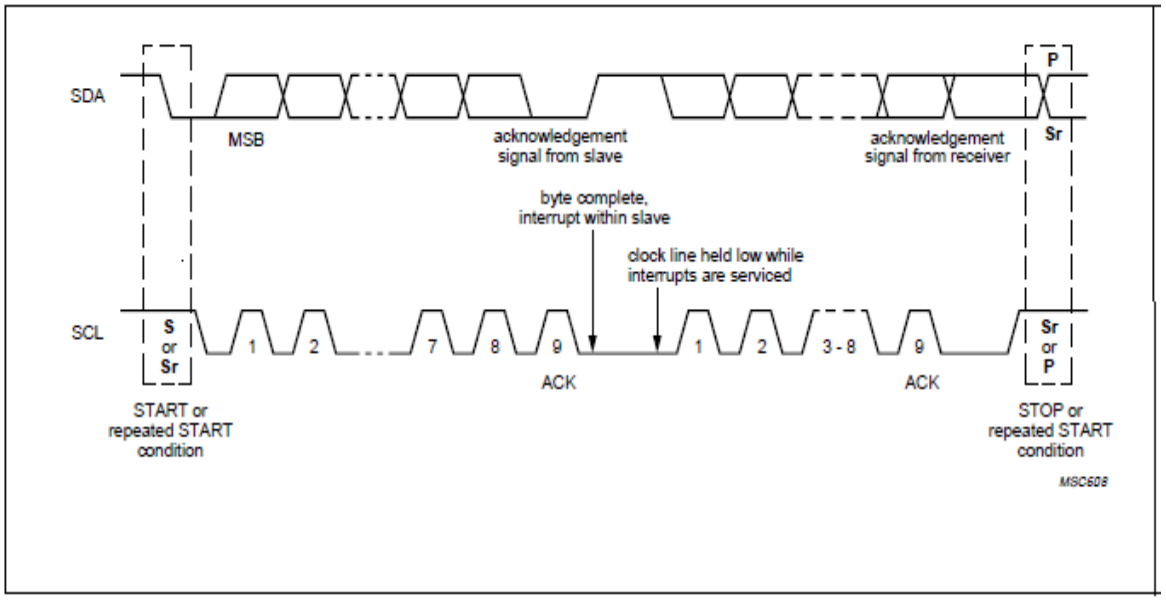

Fig 2.2 Timing diagram of I2C Protocol

Recent revisions of I²C can host more nodes and run faster (400 kb/s Fast mode, 1 Mb/s Fast mode plus or FM+, and 3.4 Mb/s High Speed mode); those speeds are more widely used on embedded systems than on PCs. There are other features, such as 16-bit addressing [1].

2.2.2 SPI

From [1], The Serial Peripheral Interface Bus (SPI) bus is a synchronous serial data link standard named by Motorola that operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines. Sometimes SPI is called a "four wire" serial bus, contrasting with three, two, and one wire serial buses.

The SPI bus specifies four logic signals.

- SCLK — Serial Clock (output from master)
- MOSI/SIMO — Master Output, Slave Input (output from master)

- MISO/SOMI — Master Input, Slave Output (output from slave)
- SS — Slave Select (active low; output from master)

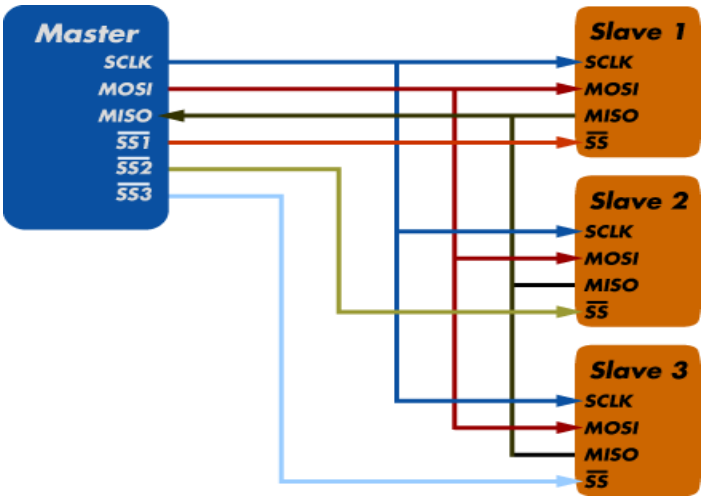

Fig 2.3 Block diagram of SPI configuration

SPI is simple and efficient because less overhead than I^2C due to lack of addressing, plus SPI is full duplex.

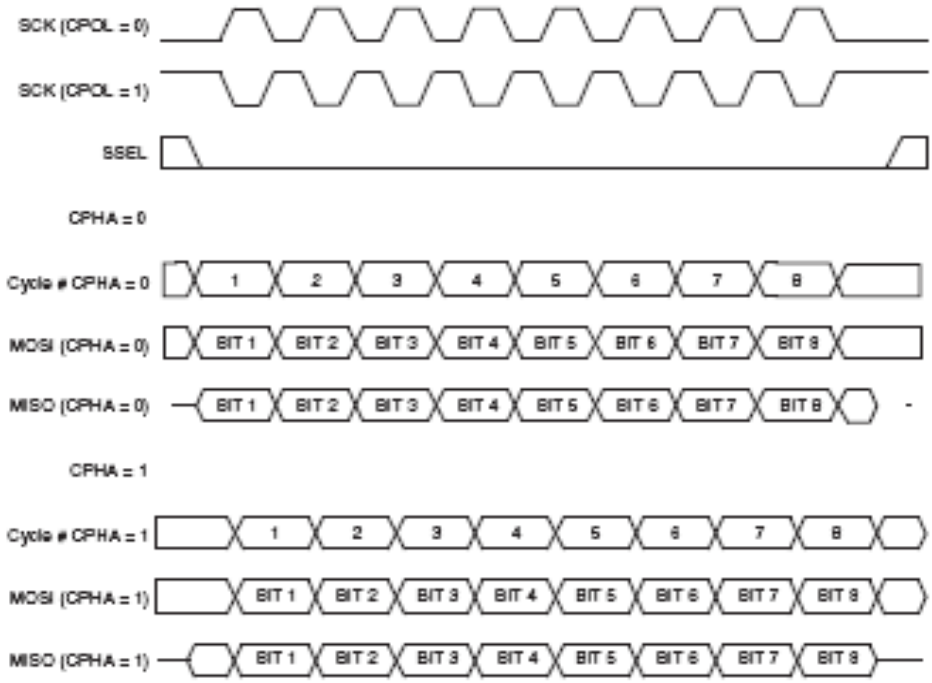

Fig 2.4 SPI data transfer format (CPHA = 0 and CPHA = 1)

But the drawback of SPI is that for multiple slaves, each slave needs separate slave select signal more effort and more hardware than I^2C. There is no specified flow controls any acknowledgement mechanism to confirm receipt of data. SPI is faster, but gets complicated when there is more than one slave involved [1].

2.2.3 UART

From [1], UART performs serial to parallel conversion on data characters received from a peripheral device or a modem and parallel to serial conversion on data characters received from CPU.

Fig 2.5 Block diagram of UART Protocol

A universal asynchronous receiver/transmitter, abbreviated UART is a type of "asynchronous receiver/transmitter", a piece if computer hardware that translates data between parallel and serial forms. UARTs are commonly used in conjunction with communication standards such as RS-232, RS-422 or RS-485. The universal designation indicates that the data format and transmission speeds are configurable and that the actual electrical signaling levels and methods (such as differential signaling etc.) typically are handled by a special driver circuit external to the UART.

A UART is usually an individual (or a part of an) integrated circuit used for serial communications over a computer or peripheral device serial port. UARTs are now commonly included in microcontrollers. Many modern ICs now come with a UART that can also communicate synchronously; these devices are called as USART (universal synchronous/asynchronous receiver/transmitter).

Transmission operation is simpler since it is under control of the transmitting system. As soon as data is deposited in the shift register after completion if the previous character, the UART hardware generates a start bit, shifts the required number of data

bits out to the line, generates and appends the parity bit (if used), and appends the stop bit. Since transmission of a single character may take a long time relative to CPU speed, the UART will maintain a flag showing busy status so that the host system does not deposit a new character for the transmission until the previous one has been completed; this may also be done with an interrupt.

2.3 ARM CORE

2.3.1 Introduction to ARM7TDMI based LPC2148

As per [7], the LPC2141/42/44/46/48 microcontrollers are based on a 16-bit/32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combine microcontroller with embedded high speed flash memory ranging from 32kB to 512 kB. A 128-bit wide memory interface and a unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30%.

2.3.2 Selection Criteria of ARM7TDMI based LPC2148

From [7], due to their tiny size and low power consumption, LPC2148 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. Serial communications interfaces ranging from a USB 2.0 Full-speed device, multiple UARTs, SPI, SSP to I2C-bus and on-chip SRAM of 8kB up to 40kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power.

2.3.3 Main features of LPC2148 which satisfy the system requirements

System requires the interfacings of protocols are UART, I2C and SPI. The system also requires A-D and D-A converter. In the [7], the technical terms and specifications of these protocols and ADC, DAC are elaborated as follows.

[a] UARTs

The LPC214148 contain two UARTs. In addition to standard transmit and receive data lines, the LPC2148 UART1 also provide a full modem control handshake

interface. UARTs in LPC2148 introduce a fractional baud rate generator for both UARTs, enabling these microcontrollers to achieve standard baud rates such as 115200 with any crystal frequency above 2MHz. In addition, auto-CTS/RTS flow-control functions are fully implemented in hardware.

[b] I2C-bus serial I/O controller

The LPC214148 contains two I2C-bus controllers. The I2C-bus implemented in LPC214148 supports bit rates up to 400kbit/s (Fast I2C-bus).

[c] SPI serial I/O controller

The LPC2141 contain one SPI controller.

After getting acquainted with required terminologies and previous work related to the system by surveying the literature, the next stage is to design the system on paper.

[d] Analog to Digital converter

Basic clocking for the A/D converters is provided by the APB clock. A programmable divider is included in each converter, to scale this clock to the 4.5 MHz (max) clock needed by the successive approximation process. A fully accurate conversion requires 11 of these clocks.

- 10 bit successive approximation analog to digital converter (two in LPC2144/6/8).
- Input multiplexing among 6 or 8 pins (ADC0 and ADC1).
- Measurement range 0 V to VREF (typically 3 V; not to exceed VDDA voltage level).

[e] Digital to Analog converter

- 10 bit digital to analog converter
- Resistor string architecture
- Buffered output, low settling time.

2.4 CURRENT MARKET REVIEW

Protocol converter applications vary from industry to industry. Most widely used protocol converters are Philips I2C to UART bridges, Cypress UART to SPI bridges.

These are one to one/many protocol converters. But they do not accept real time analog inputs. Transmission protocol selection facility is not available widely. The system proposed in this project can be used to accept both analog/digital information and transmit data over user selected protocol from list of supported protocol. The analog signal is retrieved back into original format at the receiver as well.

After getting acquainted with required terminologies and previous work related to the system by surveying the literature, the next stage is to design the system on paper and implementation of the same. It also includes testing of the implemented system.

3.1 INTRODUCTION

This chapter includes the complete design details along with some encoding algorithm/s. Technical specifications related to use of protocols by LPC2148 is also stated in detail. Protocol initialization algorithms are included along with details of related registers of LPC 2148.

3.2 SYSTEM BLOCK DIAGRAM

Block diagram of "Multiprotocol Transmitter" system is as shown below.

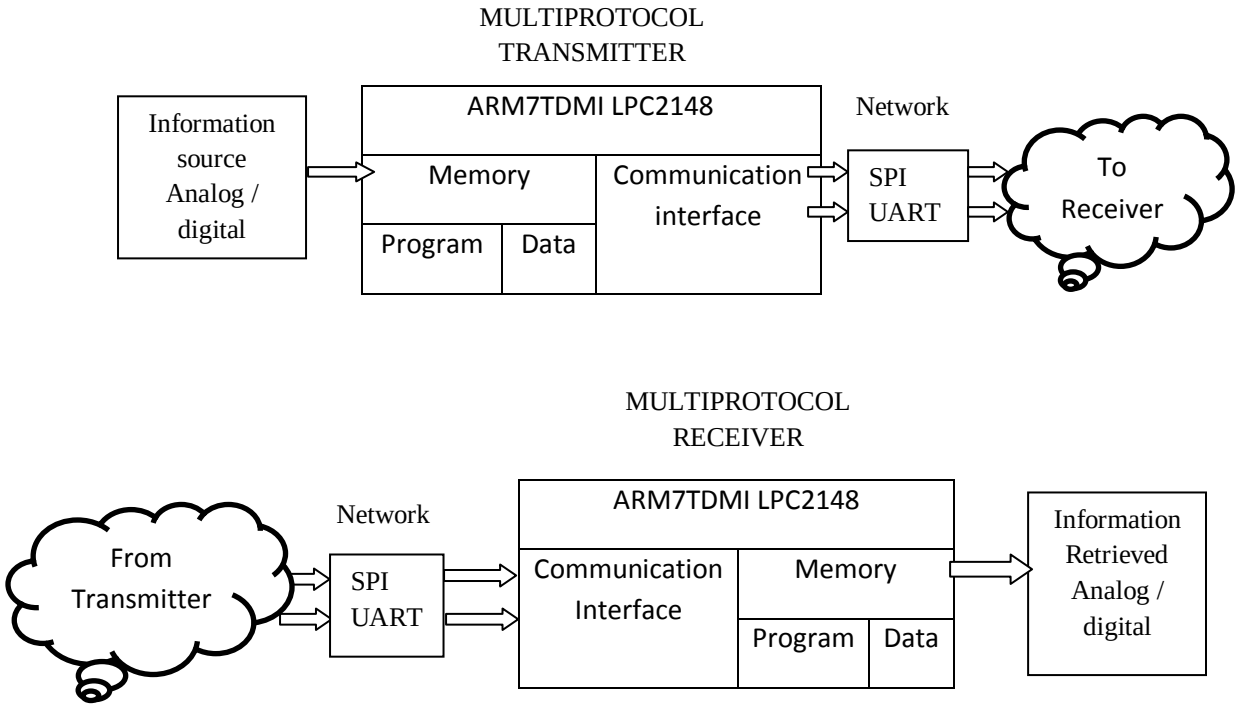

Fig 3.1 System block diagram of Multiprotocol Transmitter

3.3 FUNCTIONAL DESCRIPTION

The multiprotocol transmitter - receiver system will accept input from external source. The input is conditioned according to requirements and the digital data is provided to processor. ARM7TDMI core based LPC 2148 is used as the central processing unit.

User will select the required communication protocol (UART, I2C and SPI) through matrix keypad. LPC 2148 will carry out all the necessary framing of data according to selected protocol and send it to respected communication port and display name of the selected protocol on LCD. Communication network will transmit this information to desired node on the network.

At the receiver end, data will be received on the particular protocol (UART, I2C and SPI). LPC 2148 will carry out all the necessary de-framing of data accordingly. Name of protocol on which data is received is displayed on LCD. Received data stream will be converted back into analog form using DAC.

The system block diagram shown above is much generalized and purposely do not specify any interconnection details. This section does not put tight constraints on hardware specifications. Following sections will explore each block in detail and interconnections are specified at the end of this chapter.

3.3.1 Information Source

The multiprotocol transmitter would be able to accept any data from any information source. The sources can be broadly divided into two categories – analog and digital.
The inbuilt Analog to Digital Convertor will be useful to accept real world data in analog voltage form. Other analog information like voice may be applied to system through CODEC ICs in digital format. Image information can also be provided for transmission through digital camera.
Digital information from various sources like Personal Computer/s or other embedded system/s on network can be readily accepted using respected protocols.
In this way information is provided to CPU in digital form.

3.3.2 ARM7TDMI LPC 2148

The information is ultimately processed in digital format by CPU of the system i.e. LPC 2148. This digital data is transmitted to another node/s on network using protocol selected by the user.

Following are the key features of LPC2148

- 16-bit/32-bit ARM7TDMI-S microcontroller.

- 8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory.
- USB 2.0 Full-speed compliant device controller with 2 kB of endpoint RAM.
- Multiple serial interfaces including two UARTs, two Fast I2C bus, SPI with buffering and variable data length capabilities.
- Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.

From [7], the following figure shows pin configuration of LPC 2148

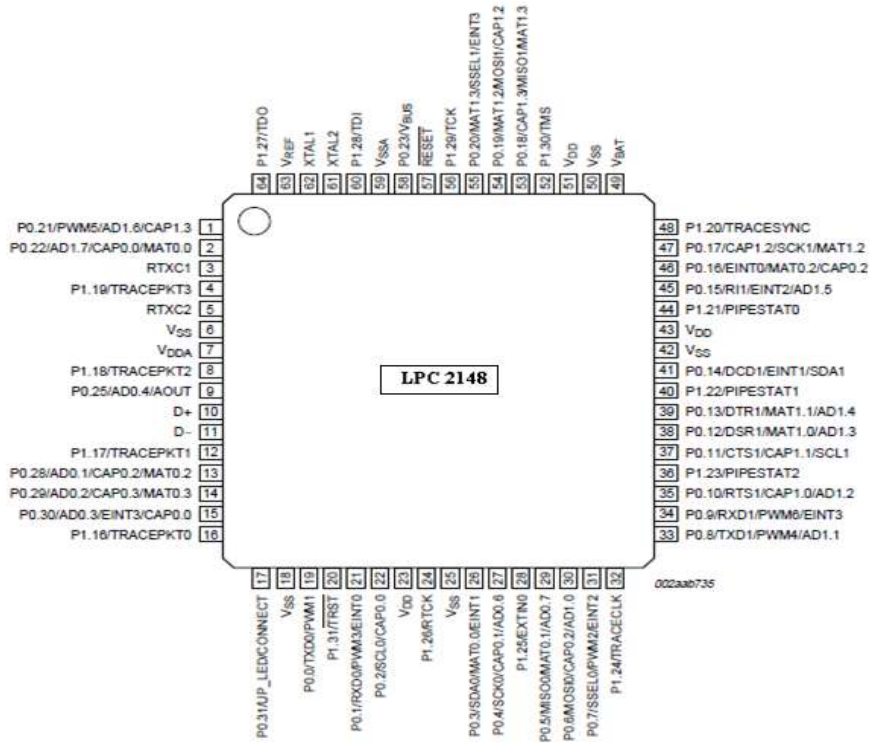

Fig 3.2 Pin Configuration of ARM Processor

Thus this block will perform critical operation of network communication. It is explained as below.

A] COMMUNICATION INTERFACE

This section provides details about actual procedure of data transmission using particular protocol supported by LPC 2148. Hardware specific features of LPC 2148 regarding these protocols are provided along with internal register description wherever necessary. Protocol initialization algorithms are also included with every

possible mode of operation. Embedded 'C' codes are not included as these cannot be separated from hardware specifications of peripheral systems which are not the part of discussion of this report.

1) UARTx

As per [8], the LPC2148 contain two UARTs. In addition to standard transmit and receive data lines, the LPC2148 UART1 also provide a full modem control handshake interface.

Compared to previous LPC2000 microcontrollers, UARTs in LPC2148 introduce a fractional baud rate generator for both UARTs, enabling these microcontrollers to achieve standard baud rates such as 115200 with any crystal frequency above 2 MHz. In addition, auto-CTS/RTS flow-control functions are fully implemented in hardware.

Following are the key Features of UARTx

- 16 B Receive and Transmit FIFOs
- Receiver FIFO trigger points at 1 B, 4 B, 8 B, and 14 B
- Built-in fractional baud rate generator covering wide range of baud rates without a need for external crystals of particular values
- Transmission FIFO control enables implementation of software (XON/XOFF) flow control on both UARTs
- LPC2148 UART1 equipped with standard modem interface signals, this module
- also provides full support for hardware flow control (auto-CTS/RTS).

Following are the LPC 2148 Pins Related to UART0

P0.0/TXD0 - Transmitter output for UART0

P0.1/RXD0 - Receiver input for UART0

Following are the key Registers related to UART in LPC 2148

UART0 Receiver Buffer register (U0RBR - 0xE000 C000) - The U0RBR is the top byte of the UART0 Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents

the "oldest" received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

UART0 Transmit Holding Register (U0THR - 0xE000 C000) -The U0THR is the top byte of the UART0 TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit. The Divisor Latch Access Bit (DLAB) in U0LCR must be zero in order to access the U0THR. The U0THR is always Write Only.

UART0 Divisor Latch registers (U0DLL-0xE000C000 and U0DLM-0xE000C004) - The UART0 Divisor Latch is part of the UART0 Baud Rate Generator and holds the value used to divide the clock in order to produce the baud rate clock, which must be 16x the desired baud rate.

The U0DLL and U0DLM registers together form a 16 bit divisor where U0DLL contains the lower 8 bits of the divisor and U0DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed. The Divisor Latch Access Bit (DLAB) in U0LCR must be one in order to access the UART0 Divisor Latches.

UARTnbaudrate = PCLK/{16 ×(256 × UnDLM + UnDLL)}

2) I2C-bus serial I/O controller

As per [8], the LPC2148 contain two I2C-bus controllers. The I2C-bus is bidirectional, for inter-IC control using only two wires: a serial clock line (SCL), and a serial data line (SDA).

Each device is recognized by a unique address and can operate as either a receiver-only device (e.g., an LCD driver or a transmitter with the capability to both receive and send information (such as memory)).

Transmitters and/or receivers can operate in either master or slave mode, depending on whether the chip has to initiate a data transfer or is only addressed. The I2C-bus is a multi-master bus, it can be controlled by more than one bus master connected to it. The I2C-bus implemented in LPC2141/42/44/46/48 supports bit rates up to 400 k bit/s (Fast I2C-bus).

Following are the key Features of I2C-bus serial I/O controller

- Compliant with standard I2C-bus interface
- Easy to configure as master, slave, or master/slave
- Programmable clocks allow versatile rate control
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- The I2C-bus can be used for test and diagnostic purposes
- Programmable clock to allow adjustment of I2C transfer rates.

Following are the LPC 2148 Pins Related to I2C

P0.2/SCL0 - I2C0 clock input/output, open-drain output

P0.3/SDA0 - I2C0 data input/output, open-drain output

P0.11/CTS1/SCL1 - I2C1 clock input/output, open-drain output

P0.14/DCD1/SDA1 - I2C1 data input/output, open-drain output.

All above pads of LPC 2148 are open-drain 5 V tolerant digital I/O I2C-bus 400 kHz specification compatible pads. It requires external pull-up to provide output functionality.

Following are the key Registers related to I2C in LPC 2148

I2CONSET - I2C Control Set Register. When a one is written to a bit of this register, the corresponding bit in the I2C control register is set. Writing a zero has no effect on the corresponding bit in the I2C control register.

I2STAT - I2C Status Register. During I2C operation, this register provides detailed status codes that allow software to determine the next action needed.

I2DAT - I2C Data Register. During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.

I2ADR - I2C Slave Address Register. Contains the 7-bit slave address for operation of the I2C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the general call address.

I2SCLH - SCH Duty Cycle Register High Half Word. It determines the high time of the I2C clock.

I2SCLL - SCL Duty Cycle Register Low Half Word. It determines the low time of the I2C clock. I2SCLL and I2SCLH together determine the clock frequency generated by an I2C master and certain times used in slave mode.

I2CONCLR - I2C Control Clear Register. When a one is written to a bit of this register, the corresponding bit in the I2C control register is cleared. Writing a zero has no effect on the corresponding bit in the I2C control register.

Initialization routine for I2C in LPC 2148 is as follows

Example to initialize I2C Interface as a Slave and/or Master.

1. Load I2ADR with own Slave Address, enable general call recognition if needed.
2. Enable I2C interrupt.
3. Write 0x44 to I2CONSET to set the I2EN and AA bits, enabling Slave functions. For Master only functions, write 0x40 to I2CONSET.

Start Master Transmit function

Begin a Master Transmit operation by setting up the buffer, pointer, and data count, then initiating a Start.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Write bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up data to be transmitted in Master Transmit buffer.
5. Initialize the Master data counter to match the length of the message being sent.
6. Exit

Start Master Receive function

Begin a Master Receive operation by setting up the buffer, pointer, and data count, then initiating a Start.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Read bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up the Master Receive buffer.
5. Initialize the Master data counter to match the length of the message to be received.
6. Exit

3) SPI serial I/O controller

As per [8], the LPC2148 contain one SPI controller. The SPI is a full duplex serial interface, designed to handle multiple masters and slaves connected to a given bus.

Only a single master and a single slave can communicate on the interface during a given data transfer. During a data transfer the master always sends a byte of data to the slave, and the slave always sends a byte of data to the master.

Following are the key Features of SPI serial I/O controller

- Compliant with SPI specification
- Synchronous, Serial, Full Duplex, Communication
- Combined SPI master and slave
- Maximum data bit rate of one eighth of the input clock rate

Following are the LPC 2148 Pins Related to SPI

P0.4/SCK0 - Serial clock for SPI0, SPI clock output from master or input to slave.

P0.5/MISO0 - Master In Slave OUT for SPI0, data input to SPI master or data output from SPI slave

P0.6/MOSI0 - Master Out Slave In for SPI0, data output from SPI master or data input to SPI slave

P0.7/SSEL0 - Slave Select for SPI0, selects the SPI interface as a slave

P0.18/MISO1 - Master In Slave Out for SSP, data input to SPI master or data output from SSP slave.

Following are the key Registers related to SPI in LPC 2148

SPCR - SPI Control Register. This register controls the operation of the SPI.

SPSR - SPI Status Register. This register shows the status of the SPI.

SPDR - SPI Data Register. This bi-directional register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI by writing to this register. Data received by the SPI can be read from this register.

SPCCR - SPI Clock Counter Register. This register controls the frequency of a master's SCK.

SPINT - SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.

Master operation

The following sequence describes how to process a data transfer with the SPI block when it is set up as the master. This process assumes that any prior data transfer has already completed.

1. Set the SPI clock counter register to the desired clock rate.
2. Set the SPI control register to the desired settings.
3. Write the data to transmit to the SPI data register. This write starts the SPI data transfer.

4. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last cycle of the SPI data transfer.
5. Read the SPI status register.
6. Read the received data from the SPI data register (optional).
7. Go to step 3 if more data is required to transmit.

Slave operation

The following sequence describes how to process a data transfer with the SPI block when it is set up as slave. This process assumes that any prior data transfer has already completed. It is required that the system clock driving the SPI logic be at least 8X faster than the SPI.

1. Set the SPI control register to the desired settings.
2. Write the data to be transmitted to the SPI data register (optional). Note that this can only be done when a slave SPI transfer is not in progress.
3. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last sampling clock edge of the SPI data transfer.
4. Read the SPI status register.
5. Read the received data from the SPI data register (optional).
6. Go to step 2 if more data is required to transmit.

3.3.3 Network

The network will consist of various node/s which may be embedded system/s or general purpose system like Personal Computer. These nodes can vary according to user application and may support different protocols. So these nodes can be virtually replaced by the protocol/s they support. Following are the system supported protocols.

USB2.0: Universal Serial Bus is a set of connectivity specifications which allows high-speed, easy connection of peripherals with a design data rate of up to 480 Mbps [1].

UART: UART performs serial to parallel conversion on data characters received from a peripheral device or a modem and parallel to serial conversion on data characters received from CPU [1].

I2C: It's a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer [1].

SPI: SPI is a synchronous protocol that allows a master device to initiate communication with a slave device [1].

Reverse operations will take place at receiver end. Network section will receive the information; ARM7TDMI LPC 2148 will de-frame the information received over particular protocol. Protocol name would be displayed on LCD. De-framed stream of date will be fade to DAC/Decoder to bring it back into analog format.

3.3.4 TESTING RESULTS

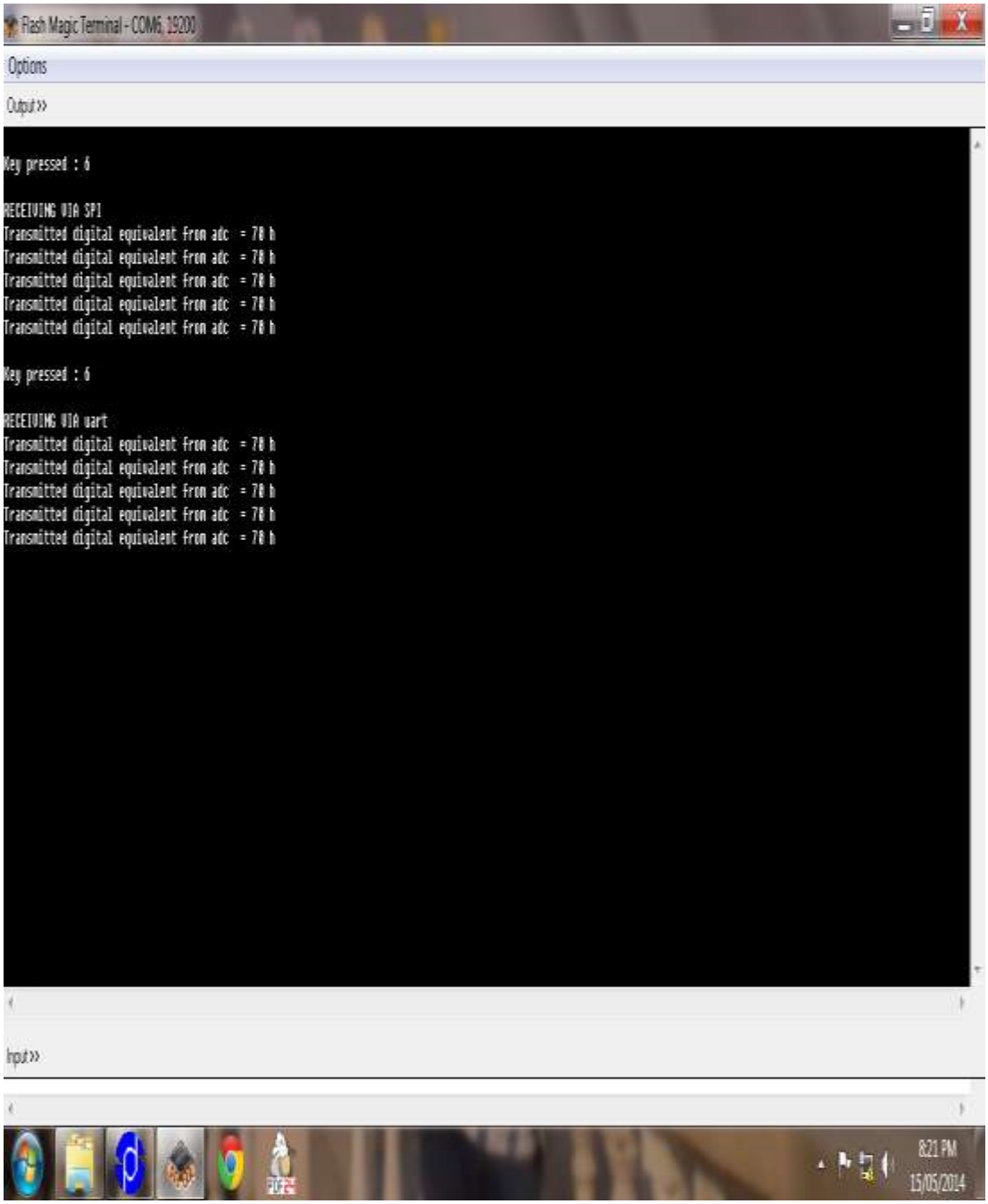

Fig 3.3 Transmitter Results

Fig 3.4 Receiver Results

Due to tiny size and low power consumption, LPC2148 is ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. Serial communications interfaces ranging from a USB 2.0 Full-speed device, multiple UARTs, SPI, SSP to I2C-bus and on-chip SRAM of 8kB up to 40kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers suitable for industrial control. This chapter will deal with various features of selected processor and draw a conclusion whether it is suitable to use the same for proposed system.

4.1 CONCLUSION

ARM's approach has been to design RISC processor architectures with instruction sets that provide efficient support for various applications, space with an optimal balance between hardware and software implementations. The proposed system needs this key feature of handling the hardware and software implementation of various protocols.

Thus we select the ARM7TDMI Family of microprocessor and LPC 2148 microcontroller by NXP based on this processor family as it satisfies all requirements of the system efficiently.

LPC2148 supports the required protocols with high data rate that suffice real time data transmission. Built in protocol controllers along with respective special function registers reduces programming burden.

Although LPC2148 works on 3.3V, most IO pads and pads used by protocols are TTL compatible. This makes the interfacing of system with other systems easier.

Single cycle instruction execution provides very high data processing rate which ensures efficient sampling of analog data. 10 bit successive approximation analog to

digital converter provides conversion time in the range of microseconds which helps the execution of real time application.

At the time of actual data transmission noise or glitches are introduced in channel which can be taken care of by software control. 512 kB of flash memory is sufficient for most of embedded applications. This system consumes flash memory around 80 kB.

REFERENCES

[1] www.wikepedia.com

[2] www.alldatasheet.com

[3] www.nxp.com

[4] www.efy.com

[5] www.google.com

[6] ARM System Developer's Guide Designing and Optimizing, System Software by, Andrew N. Sloss, Dominic Symes, Chris Wright

[7] LPC2141/42/44/46/48 Product Data Sheet

[8] UM10139LPC214x User manual Rev. 4 — 23 April 2012