A BRIEF INTRODUCTION TO I^{2}C PROTOCOL & IT`S APPLICATION IN AT-MEGA 128

Susmit Paul[1] Rishav Choudhary[2]

[1,2]*Department of Electronics & Communication Engineering,*
Tula's Institute/ Uttarakhand Technical University, (India)

ABSTRACT

One of the most common communication buses in the world, and hence one that should be well understood by prospective engineers, is the I2C bus. With the help of I^{2}C communication is supported with various slow, on-board peripheral devices that are accessed intermittently, while being extremely modest in its hardware resource needs. In this paper we are discussing on I^{2}C protocol which is a very important part of microprocessors. Embedded system is one of the most important parts for any communicating electronics devices. Inter integrated circuit has very low bandwidth which are operated up to the speed 400kbps. I^{2}C are used in parallel as well as serial communication, but it is very important for parallel communication and very useful in communication channel. In this project we discuss the basics of I^{2}C protocols. We have made alarm clock with the help of I2C protocol using temperature sensor and light sensor. Because of many advantages of I^{2}C protocol, it is one of the most popular serial peripheral interfaces to connect integrated circuit. Here we use the AT MEGA128 microcontroller for the microcontroller based alarm system. I^{2}C devices are also found in embedded system including real time clock and thermal sensor. I^{2}C bus is the most common communication type in the world and is easily understand by the engineer.

Keywords: I^{2}C Protocol, SDA, SCL, Microcontroller, AT MEGA, SPI Protocol

I. INTRODUCTION

Communication is one of the integral parts of science that has to be developed day by day. For the communication process we always need protocols. Like any mobile communication, the embedded devices also require the protocols for the communication with the peripherals. Inter Integrated Circuit protocol (I^{2}C) is one of the most important for the microcontroller devices. With the help of I^{2}C communication, we can support various slow, on-board peripheral devices that are accessed intermittently, while being extremely modest in its hardware resource needs. Basically this protocol has two serial lines, Serial Data line (SDA) and Serial Clock line (SCL). These two lines basically work as the read and write operations respectively. Here in this paper, we will describe the brief introduction of I^{2}C protocols and after that; we will differentiate it with the Serial Peripheral Interface (SPI) protocols.

In the other hand, the continuous development of microcontrollers gives us different series of controllers. The AVR series is one of the recent popular microcontrollers in this telecom industry. Here we use the AT MEGA 128 for the performance of various applications. AT MEGA 128 is the advanced controller with 128 KB flash memory, 64 pin IC which can be operated with only 3v3. Firstly we performed the applications of alarm clock, digital watch, led glowing, switches etc. Then we will make an alarm clock using I^{2}C protocol.

II. CONCEPT OF I²C PROTOCOL

To communicate across circuit-board distances we often used Inter-Ic bus. Here's a primer on the protocol. At the low end of the spectrum of communication options for "inside the box" communication is I²C ("eye-squared-see"). The name I²C is shorthand for a standard Inter-IC (integrated circuit) bus.

With the help of I²C communication is supported with various slow, on-board peripheral devices that are accessed intermittently, while being extremely modest in its hardware resource needs. I²C is a simple protocol having low-bandwidth and short-distance.I²C devices which are mostly available operate at speeds up to 400Kbps, with some venturing up into the low megahertz range. To connect multiple devices together we can easily use I2C since it has a built-in addressing scheme.

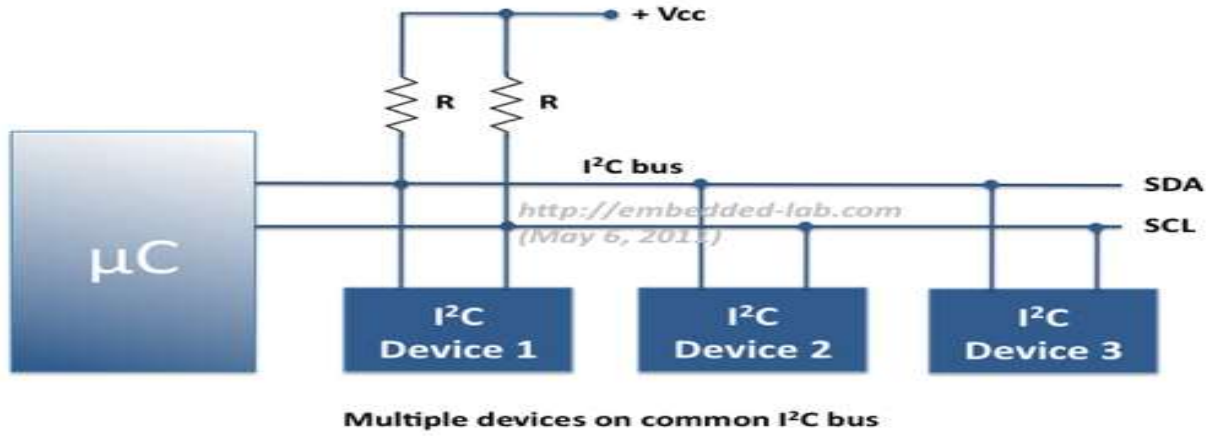

Fig: 1 Basic Concept of I²C Protocols

Philips originally developed I²C for communication devices inside of a TV set. Examples of simple I²C-compatible devices found in embedded systems include EEPROMs, thermal sensors, and real-time clocks. I²C is also used as a control interface to signal processing devices that have separate, application-specific data interfaces. For instance, it's commonly used in multimedia applications, where typical devices include RF tuners, video decoders and encoders, and audio processors. In all, Philips, National Semiconductor, Xi-cor, Siemens, and other manufacturers offer hundreds of I²C-compatible devices.

III. PHYSICAL LAYER OF I²C BUS

At the physical layer, SCL and SDA are two wires. SCL is the clock line and gives a flow control mechanism. It is used to synchronize all data transfers over the I2C bus. SDA is the data line. In I²C SCL and SDA lines are connected to all devices. There we need third wire use as the ground or 0 volts. There may also be a 5volt wire power is being distributed to the devices. Both SCL and SDA lines are "open drain" drivers. What this means is that the chip can drive its output low, but it cannot drive it high. If you want to make the line high you must provide pull-up resistors to the 5v supply. There should be a two resistor one from the SCL line to the 5v line and another from the SDA line to the 5v line. There is no critical value of resistor which we used. I have seen anything from 1k8 (1800 ohms) to 47k (47000 ohms) used. 1k8, 4k7 and 10k are common values, but anything in this range should work OK. I recommend that you should use 1k8 for the best performance. If you cannot use the resistor , the both line SCL and SDA lines will always be low - nearly 0 volt then the I2C bus will not working.

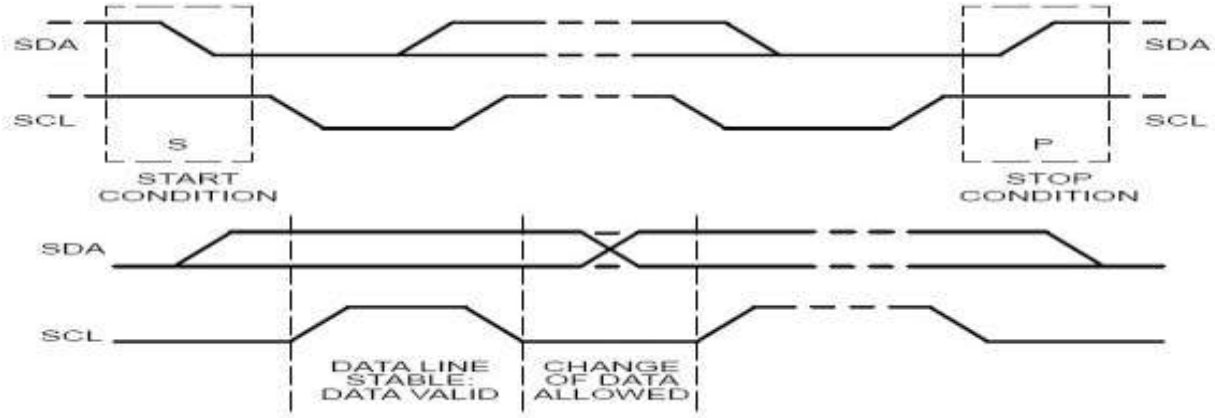

Fig: 2 Physical Layer of I^{2}C Bus

The devices on the I2C bus are either masters or slaves. The SCL clock line is derived from the masters. The slaves are the devices that respond to the master. A master can initiate a transfer over the I2C bus, but slaves cannot do that. On the I2C bus usually have multiple slaves but there is normally only one master. It is possible to have multiple masters. A transfer can never initiate by Slaves. Data can transfer over the I2C bus by the both master and slaves, but that transfer is always controlled by the master.

When the master (your for example) it begins by issuing a start sequence on the I2C. There is two special sequences defined for the I2C bus, start the stop sequence. The start sequence and stop sequence are special in that these are the only places where the SDA (data line) is allowed to change while the SCL (clock line) is high .SCL is high while SDA must remain stable and not change when the data is being transmitted . The beginning and end of a transaction with the slave device mark by the start and stop sequence.

Data is transferred in sequences of 8 bits. The bits are placed on the SDA line starting with the MSB (Most Significant Bit). The SCL line is then pulsed high, then low. Remember that the chip cannot really drive the line high, it simply "let's go" of it and the resistor actually pulls it high. The device receiving the data sends back an acknowledge bit for every 8 bit transferred , so there are actually 9 SCL clock pulses to transfer each 8 bit byte of data. If the receiving device sends back a low ACK bit, then it has received the data and is ready to accept another byte. If it sends back a high then it is indicating it cannot accept any further data and the master should terminate the transfer by sending a stop sequence.

IV. I^{2}C PROTOCOL-HOW IT WORKS, AND WHAT TO WATCH OUT FOR

One of the most common communication buses in the world, and hence one that should be well understood by prospective engineers, is the I2C bus. It may not be as well known as USB or Ethernet, but much of our world of electronic devices is completely dependent on it. It also has some siblings known as SMBus and TWI, which are essentially identical from a protocol standpoint.

Clock and data signal are used byI2c protocol. SCL or SCK (for Serial Clock) called Clock and SDA (for Serial Data) is called data. There are only two sequence (start and stop) which make the I2S unique is the use of special combination and changes.

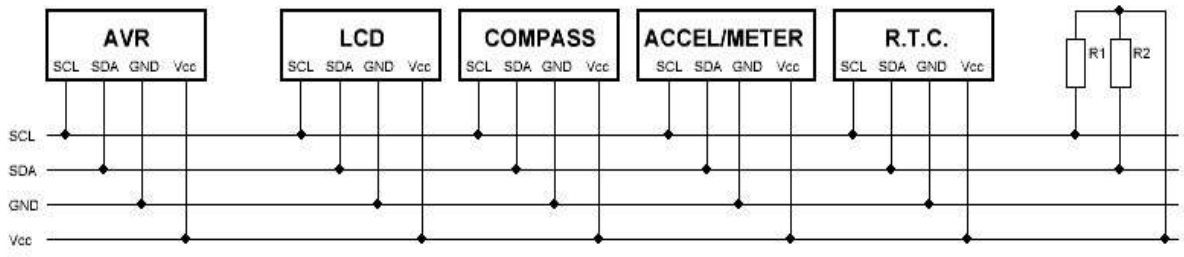

Fig: 3 Working of I²C Protocols with SDA & SCL Line

You rarely see multiple masters in the real world but in I2S bus can potentially have multiple masters and many 'slave' devices sharing the same bus to avoiding the Signal contention we used an open-collector drive scheme, in this only low signal are drives and no high signal are drives. The bus is pulled up by a resistor to drive a high signal. An addressing scheme allows a device to determine if it is being queried by the master. When the slave responds to master address by sending an Acknowledgement bit (driving SDA low) then master sees that the slaves is present, and when there is no addressed SDA line is pulled high, and this is interpreted by the Master as a Not-Acknowledgement.

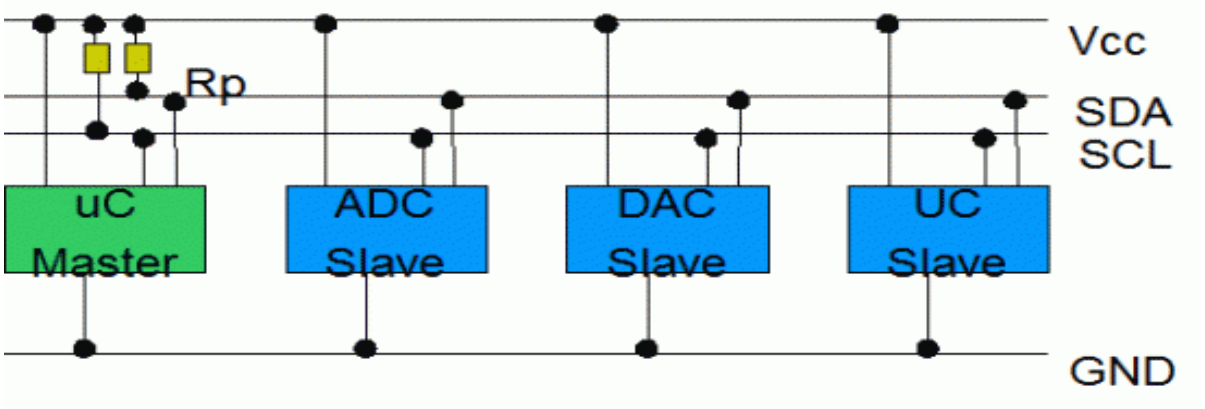

Fig: 4 Master-Slave Composition of I²C Protocol

An IDLE bus condition is obtained when both SDA and SCL lines are high. (This condition can exist within a bus transaction, but generally it is transitory.) A'START" condition is generated by the Master, followed by 7 bits of address, then a Read/Write bit. When the address match detect by a slave device it will send an ACK by driving SDA low during the next clock cycle; when it cannot get a address match then the SDA line will be left alone to be pulled up high. Following a successful ACK, data will be either sent to the slave device or read from the slave device (depending on what was indicated by the Read/Write bit). Therefore, each byte is 9 bits: 7 address plus one R/W plus one ACK/NAK, or 8 data plus one ACK/NAK. . The last data byte of a transaction should generally be followed by a NAK, to indicate that it is intended to be the final byte. After this master controls the slaves issued STOP or a re-start.

Bus errors can be introduced in several places. For example, data could follow a MAK of an address; this can potentially have multiple masters and many 'slave' devices sharing the same bus should ideally never happen. It is possible that clocking could begin from a Bus-Idle condition without a proper START. When using a dedicated I2C peripheral on the Master bus errors are rarely introduced, but errors are commonly find when used I2C protocol in a software-only solution (bit- banging). With the help of oscilloscope and digit scope it is difficult to identifying the source of these errors. The best solution is to use an I2C protocol analyzer.

V. I²C SOFTWARE PROTOCOL

A conclusion Wi-Fi, The first thing that will happen is that the master will send out a start sequence. This will alert all the slave devices on the bus that a transaction is starting and they should listen in case it is for them. Next the master will send out the device address. The slave that matches this address will continue with the transaction, any others will ignore the rest of this transaction and wait for the next. Having addressed the slave device the master must now send out the internal location or register number inside the slave that it wishes to write to or read from. This number is obviously dependant on what the slave actually is and how many internal registers it has. Some very simple devices do not have any, but most do, including all of our modules. Our CMPS03 has 16 locations numbered 0-15. The SRF08 has 36. Having sent the I2C address and the internal register address the master can now send the data byte (or bytes, it doesn't have to be just one). The master can continue to send data bytes to the slave and these will normally be placed in the following registers because the slave will automatically increment the internal register address after each byte.

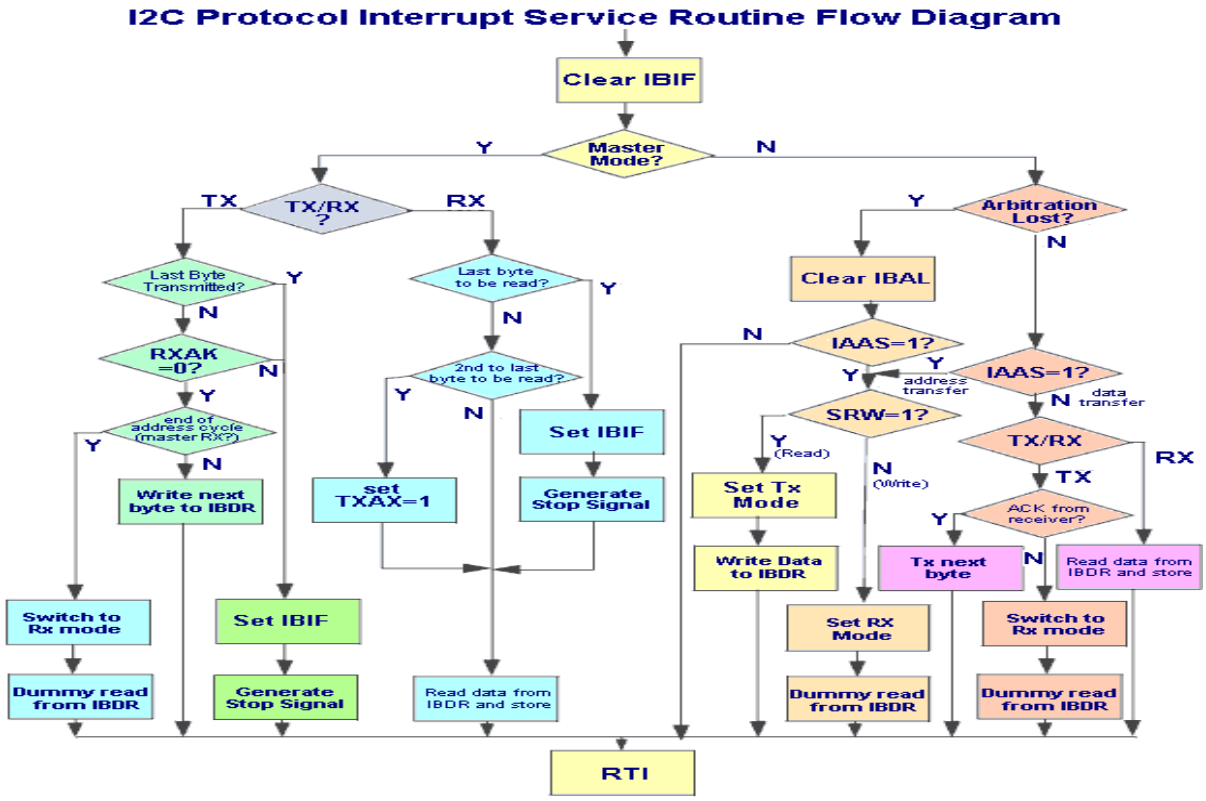

Fig: 5 Algorithm or Flow Chart of I^2C Protocols

When the master has finished writing all data to the slave, it sends a stop sequence which completes the transaction. So to write to a slave device:

1. Send a start sequence.
2. Send the I2C address of the slave with the R/W bit low (even address).
3. Send the internal register number you want to write to.
4. Send the data byte.
5. Send the stop sequences.

As an example, you have an SRF08 at the factory default address of 0xE0. To start the SRF08 ranging you would write 0x51 to the command register at 0x00 like this:

1. Send a start sequence.
2. Send 0xE0 (I2C address of the SRF08 with the R/W bit low (even address).

3. Send 0x00 (Internal address of the command register).
4. Send 0x51 (The command to start the SRF08 ranging).
5. Send the stop sequence.

5.1 Reading from Slave

This is a little more complicated - but not too much more. Before reading data from the slave device, you must tell it which of its internal addresses you want to read. So a read of the slave actually starts off by writing to it. This is the same as when you want to write to it: You send the start sequence, the I2C address of the slave with the R/W bit low (even address) and the internal register number you want to write to. Now you send another start sequence (sometimes called a restart) and the I2C address again - this time with the read bit set. You then read as many data bytes as you wish and terminate the transaction with a stop sequence. So to read the compass bearing as a byte from the CMPS03 module:

1. Send a start sequence
2. Send 0xC0 (I2C address of the CMPS03 with the R/W bit low (even address)
3. Send 0x01 (Internal address of the bearing register)
4. Send a start sequence again (repeated start)
5. Send 0xC1 (I2C address of the CMPS03 with the R/W bit high (odd address)
6. Read data byte from CMPS03
7. Send the stop sequence.

VI. COMPARISON: I2C VS. SPI

SPI is a single-master communication protocol. This means that one central device initiates all the communications with the slaves. When the SPI master wishes to send data to a slave and/or request information from it, it selects slave by pulling the corresponding SS line low and it activates the clock signal at a clock frequency usable by the master and the slave. The master generates information onto MOSI line while it samples the MISO line. A master/slave pair must use the same set of parameters – SCLK frequency, CPOL, and CPHA for a communication to be possible. If multiple slaves are used, that are fixed in different configurations, the master will have to reconfigure itself each time it needs to communicate with a different slave.

Although both SPI and I2C provide good support for communication with slow peripheral devices that are accessed intermittently, each of the way of communication have its own advantages towards each other.

SPI is better suited than I2C for applications that are naturally thought of as data streams (as opposed to reading and writing addressed locations in a slave device). An example of a "stream" application is data communication between microprocessors or digital signal processors. Another is data transfer from analog-to-digital converters.

SPI can also achieve significantly higher data rates than I2C which is limited to 400 KHz in most cases. SPI-compatible interfaces often range into the tens of megahertz. SPI really gains efficiency in applications that take advantage of its duplex capability, such as the communication between a "codec" (coder-decoder) and a digital signal processor, which consists of simultaneously sending samples in and out.

Due to SPI lack of built-in device addressing, it requires more effort and more hardware resources than I2C when more than one slave is involved. The disadvantage here lies that it is a three-wire interface and if you are having more than 1 device, then you have to provide each device with separate Chip Select pins (CS).

I2C is also a true multi-master bus because it has collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer. Furthermore, I2C also preserve data integrity by filtering rejects spikes on the bus data line.

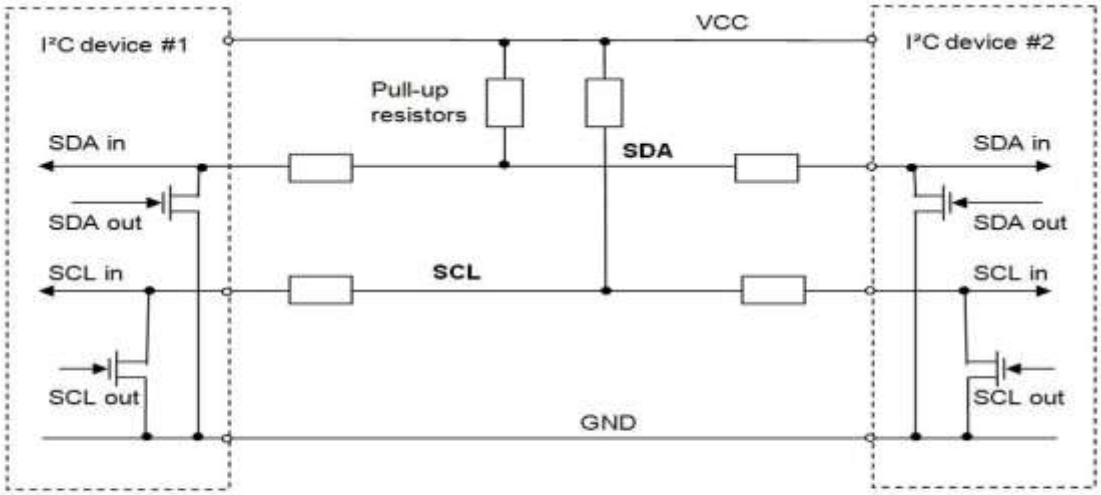

Figure 4: I²C bus with 2 devices connected. SDA and SCL are connected to VCC through pull-up resistors. Each device controls the bus lines outputs with open drain buffers.

Fig: 6 typical I2C protocols

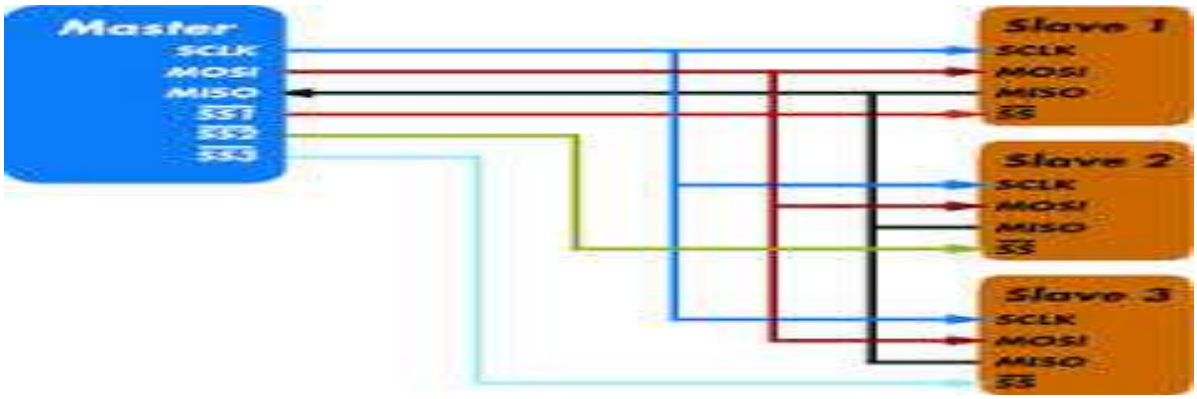

Fig: 7 SPI Protocol with MISO & MOSI line

SPI Protocol	I^2C Protocol
Three bus lines are required; a data input line (SI_1), a data output line (SO_1) and a serial clock line (SCK_1) [Plus 1 Chip Select (CS)].	Two bus lines are required; a serial data line (SDA) and a serial clock line (SCL)
No official specification (component dependent)	With official specification (I2C protocol created by Philips)
Higher data rates (up to 10	Support transfer speeds of

MHz or more)	around 100kHz (original standard, or 400kHz using the most recent standard)
More efficient in point-to-point (single master, single slave) applications	More efficient in multi-master, multi-slave applications
Lack of built-in device addressing	Built-in addressing scheme and straightforward
Does not have an acknowledgement mechanism to confirm receipt of data.	Have an acknowledgement mechanism to confirm receipt of data.
Suited better for communication with on-board devices that are accessed on an occasional basis.	More overhead when handling point-to-point application
Suited better for applications that are naturally thought of as data streams .	Suited better for communication with on-board devices that are accessed on an occasional basis.

VII. APPLICATION OF I2C PROTOCOL

I²C is appropriate for peripherals where simplicity and low manufacturing cost are more important than speed. Common applications of the I²C bus are:

- Reading configuration data from SPD EEPROMs on SDRAM, DDR SDRAM, DDR2 SDRAM memory sticks (DIMM) and other stacked PC boards.
- Supporting systems management for PCI cards, through a SMBus 2.0 connection.
- Accessing NVRAM chips that keep user settings.
- Accessing low speed DACs and ADCs.
- Changing contrast, hue, and color balance settings in monitors (Display Data Channel).
- Changing sound volume in intelligent speakers.
- Controlling OLED/LCD displays, like in a cell phone.
- Reading hardware monitors and diagnostic sensors, like a CPU thermostat [and] fan speed.

- Reading real-time clocks.
- Turning on and turning off the power supply of system components.

VIII. INTRODUCTION TO AVR AT MEGA 128 MICROCONTROLLER

The AVR is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time.

The AVR is a modified Harvard architecture machine with program and data stored in separate physical memory systems that appear in different address spaces, but having the ability to read data items from program memory using special instructions.

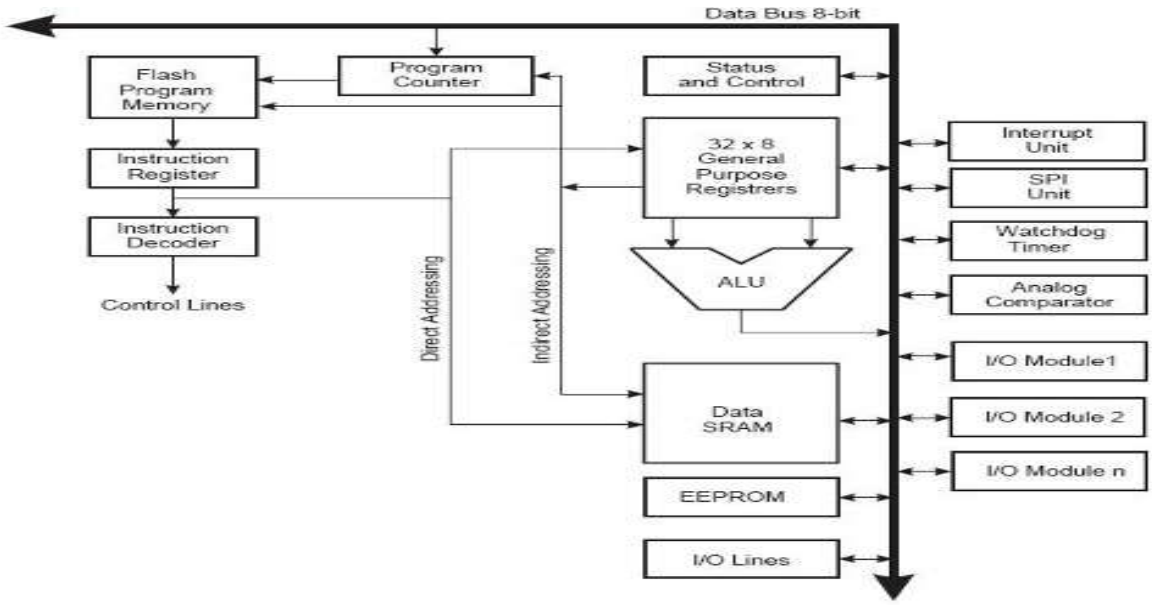

Fig: 8 AT MEGA 128 BUSES

8.1 Features of AT MEGA 128

• High-performance, Low-power AVR® 8-bit Microcontroller. Advanced RISC Architecture. • Nonvolatile Program and Data Memories – 128K Bytes of In-System Reprogrammable Flash. Endurance: 10,000 Write/Erase Cycles – Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation – 4K Bytes EEPROM Endurance: 100,000 Write/Erase Cycles – 4K Bytes Internal SRAM – Up to 64K Bytes Optional External Memory Space – Programming Lock for Software Security – SPI Interface for In-System Programming • JTAG (IEEE std. 1149.1 Compliant) Interface – Boundary-scan Capabilities According to the JTAG Standard – Extensive On-chip Debug Support – Programming of Flash, EEPROM, Fuses and Lock Bits

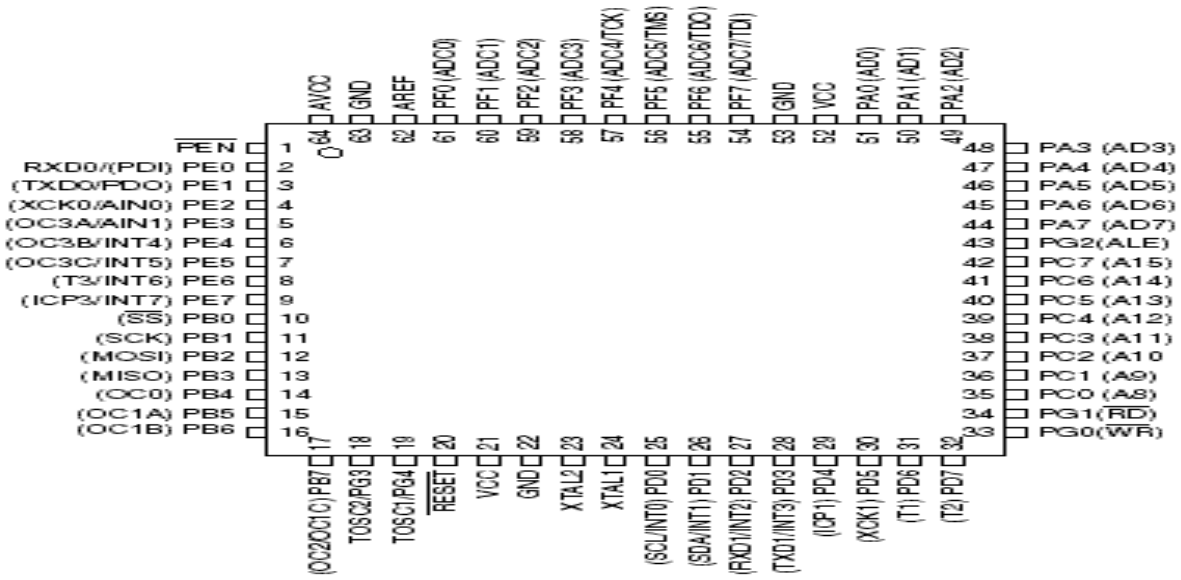

Fig: 9 Pin Configuration of AT MEGA 128

through the JTAG Interface • Peripheral Features – 53 Programmable I/O Lines – Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes – Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode – Real Time Counter with Separate Oscillator – Two 8-bit PWM Channels – 6 PWM Channels with Programmable Resolution from 2 to 16 Bits – Output Compare Modulator – 8-channel, 10-bit ADC 8 Single-ended Channels 7 Differential Channels 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x – Byte-oriented Two-wire Serial Interface – Dual Programmable Serial USARTs – Master/Slave SPI Serial Interface – Programmable Watchdog Timer with On-chip Oscillator – On-chip Analog Comparator • Special Microcontroller Features – Power-on Reset and Programmable Brown-out Detection – Internal Calibrated RC Oscillator – External and Internal Interrupt Sources – Software Selectable Clock Frequency – Global Pull-up Disable.

Fig: our AT MEGA 128 kit

IX. TEMPERATURE SENSOR VIA I2C BUS

There are four main contact temperature-sensing devices available, divided in three families: thermocouples (self-generating sensors), resistance temperature detectors and thermostats (resistive sensors), and temperature-transuding ICs (PN or Semi conductive). These sensors translate the temperature into a reference voltage, resistance or current, which is then measured and processed and a numerical temperature value is computed.

A temperature sensor produces an analogue or digital output whose strength depends on the temperature of the sensor. Heat is conducted to the sensing element through the sensor's package and its metal leads. In general, a sensor in a metal package will have a dominant thermal path through the package. For sensors in plastic packages, the leads provide the dominant thermal path. Therefore a board-mounted IC sensor will do a fine job of measuring the temperature of the circuit board.

If it's needed to measure the temperature of something other than the circuit board, it should be ensured that the sensor and its leads are at the same temperature as the object you wish to measure. This usually involves making a good mechanical (and thermal) contact by attaching the sensor and its leads to the object being measured with thermally-conductive epoxy.

If a liquid's temperature is to be measured, the sensor can be mounted inside a sealed-end metal tube and dipped into a bath, or screwed into a threaded hole in a tank. Temperature sensors and any accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion.

Any linear circuit connected to wires in an uncongenial environment can have its performance adversely affected by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, etc, as its wiring can act as an aerial and the internal junctions can act as rectifiers. In such cases, a small bypass capacitor from the power supply pin to ground rail helps clean up power supply noise.

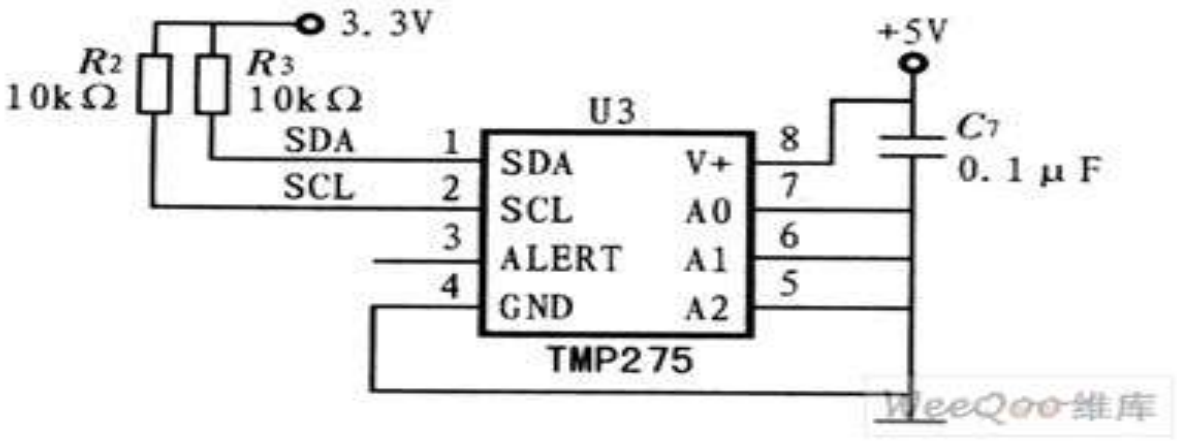

Fig: 10 TMP-275 Temperature Sensors

TMP275 is a temperature sensor from TI. It has a resolution of 0.5 degree centigrade. It is a digital output sensor interfaced with microcontroller via I2C. The TMP275 is a 0.5°C accurate, Two-Wire, serial output temperature sensor available in an MSOP-8 or an SO-8 package. The TMP275 is capable of reading temperatures with a resolution of 0.0625°C. The TMP275 is SMBus-compatible and allows up to eight devices on one bus.

X. LIGHT SENSOR VIA I²C BUS

Light sensors or sensors of light are often referred as types of or photo sensors. There are several varieties of these sensors. Light sensors detect light density but do not record images. Most common light density sensors are: Photodiodes (LDR Light Dependent Resistors) and photo resistors (LDR Light Dependent Resistors). Photo diode is capable of converting light into either current or voltage. Photo resistor is a resistor whose resistance decreases with increasing light intensity.

The light sensor enables a robot to detect light. Robots can be programmed to have a specific reaction if a certain amount of light is detected. The light sensor uses a cadmium sulfosolenide (CdS) photoconductive photocell. The CdS photocell is a photo resistor, meaning that its resistance value changes based on the amount of incident light. A typical system for detecting infrared radiation using infrared sensors includes the infrared source such as blackbody radiators, tungsten lamps, and silicon carbide. In case of active IR sensors, the sources are infrared lasers and LEDs of specific IR wavelengths. Next is the transmission medium used for infrared transmission, which includes vacuum, the atmosphere, and optical fibres.

Thirdly, optical components such as optical lenses made from quartz, CaF_2, Ge and Si, polyethylene Fresnel lenses, and Al or Au mirrors, are used to converge or focus infrared radiation. Likewise, to limit spectral response, band-pass filters are ideal.

Finally, the infrared detector completes the system for detecting infrared radiation. The output from the detector is usually very small, and hence pre-amplifiers coupled with circuitry are added to further process the received signals.

Light sensor is APDS-9300 from Avago Technologies. The APDS-9300 is a low-voltage Digital Ambient Light Photo Sensor that converts light intensity to digital signal output capable of direct I2C interface. Each device consists of one broadband photodiode (visible plus infrared) and one infrared photodiode.

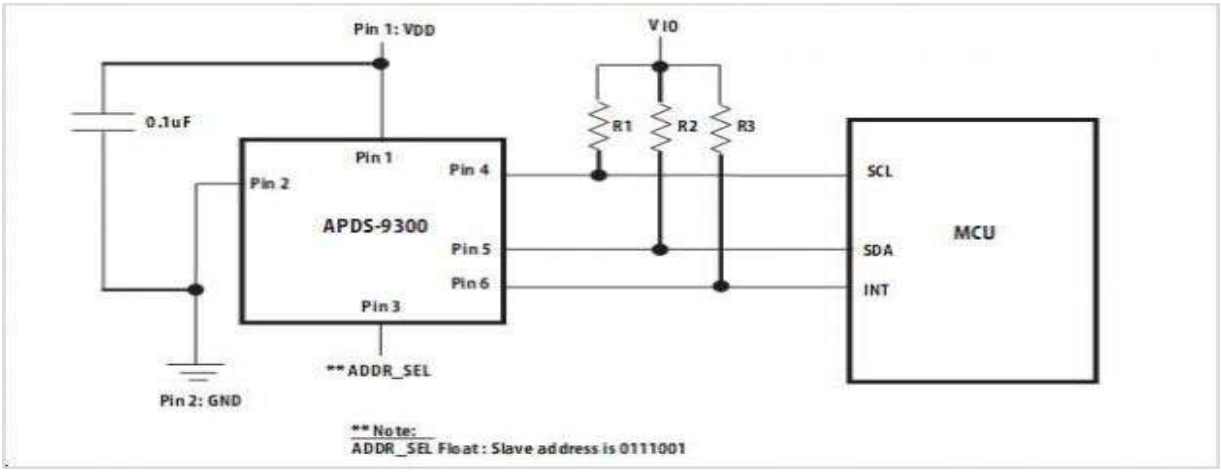

Fig: 11 Light sensors APDS-9300

XI. LITERATURE REVIEW

Today, at the low end of the communication protocols, we find I²C (for 'Inter-Integrated Circuit', protocol) and SPI (for 'Serial Peripheral Interface'). Both protocols are well-suited for communications between integrated circuits, for slow communication with on-board peripherals. At the roots of these two popular protocols we find

two major companies Philips for I²C and Motorola for SPI – and two different histories about why, when and how the protocols were created [1] [2]

Philips Semiconductors (now NXP Semiconductors) developed a simple bidirectional 2-wire bus for efficient inter-IC control. This bus is called the Inter-IC or I2C-bus. Only two bus lines are required: a serial data line (SDA) and a serial clock line (SCL). Serial, 8-bit oriented, bidirectional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to400 kbit/s in the Fast-mode, up to 1 Mbit/s in the Fast-mode Plus (Fm+), orup to 3.4 Mbit/s in the High-speed mode. The Ultra-Fast-mode is anti-directional mode with data transfers of up to 5 Mbit/s [3] [4]

Inter-Integrated Circuit, abbreviated as I2C is a serial bus short distance protocol developed by Philips Semiconductor about two decades ago to enhance communication between the core on the board and various other ICs involved around the core. This application note intends to describe the functionality of various serial buses with emphasis on I2C, and how I2C is different from other serial buses. This application note also intends to explain the functionality and working of I2C, as well as some sample code that explains how I2C is implemented [5]

A microcontroller (sometimes abbreviated µC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Neither program memory in the form of NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications [6]

XII. OUR WORK: I2C PROTOCOL BASED ALARM CLOCK

```
#include<stdlib.h>
#include<avr/io.h>
#include<util/delay.h>
#include<lcd.h>
Int main(void)
{
DDRA=0xFF;
DDRA=0xAF;
PORTA=0xAF;
PORTD=0x5F;
}
While(5);
{
Alarm_minute++;
Tone(100);
_delay_ms(100);
Tone(0);
If(alarm_minute>=60)
{
```

```
Alarm_minute=0;
}
Eeprom_update_byte(& minute_ce, alarm_minute);
Break;
}
Alarm_hour=eeprom_read_byte(&hour_ce);
Alarm_minute=eeprom_read_byte(&minute_ce);
If((hour==alarm_hour)&&(minute++alarm_minute)&&(second<=60))
{
Lcd_cmd(0x01);
Lcd_string("ALARM!!!!");
Lcd_cmd(0xC0);
Lcd_string("TASK BEGIN");
Tone(14);
_delay_ms(500);
Tone();
}
```

Fig: 12 Alarm clocks

XIII. FUTURE WORK

Fortunately or Unfortunately, the power is going to be the key differentiator and driver for the deep-submicron products in the future. There are a plethora of power management techniques available to be used. However the most effective power management approach seems to be targeting the architecture itself and make the device low power by design [7]

For the result, those protocols will be valid for the future, who consumes low power and energy. For example here the internal clock at the top level which originally has to be generated internally inside the I2C master through state machine is considered. Only for this reason, the I2C protocols already started to replace the SPI protocols in the microcontroller working purpose [9]

The results of simulation and desired behavior of the I2C bus controller agree. The interfacing of microcontroller (master) and serial EEPROM (slave) has done by using I2C bus which elaborates that I2C components are working as our desired conditions. The design of I2C controller using Verilog HDL, simplifies the design process. The designer can write his design description without choosing any specific fabrication technology. If a new technology emerges, designers do not need to redesign the circuit. He simply input the design program to the logic synthesis tool and creates a new gate level net list using the new fabrication technology. The logic synthesis tools will optimize the circuit in area and timing for the new technology [8]
Not only the protocols, can we also improve the alarm clock which can be operated by developed protocols.

XIV. CONCLUSION

In the information technology, a protocol, which is taken from the Greek protocol on, is the special set of rules that end points in a telecommunication connection use when they communicate. So the protocol is very important for any telecommunication systems. The I2C protocol is very important for parallel communication. This protocol uses only serial clock line and the serial data line. In the world of communication protocols, I²C is often considered as 'little' communication protocols compared to Ethernet, USB, SATA, PCI-Express and others, that present throughput in the x100 megabit per second range if not gigabit per second. Though, one must not forget what each protocol is meant for. Ethernet, USB, SATA are meant for 'outside the box communications' and data exchanges between whole systems. When there is a need to implement a communication between integrated circuit such as a microcontroller and a set of relatively slow peripheral, there is no point at using any excessively complex protocols. In the AT MEGA 128 microcontroller, the I2C protocol is also for the parallel communication. In the PORT D the SCL & SDA line are available here. Any microcontroller can communicate with I2C devices even if it has no special I2C interface. I2C specifications are flexible – I2C bus can communicate with slow devices and can also use high speed modes to transfer large amounts of data. Because of many advantages, I2C bus will remain as one of the most popular serial interfaces to connect integrated circuits on the board.

REFERENCES

[1] http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols
[2] http://www.dauniv.ac.in/downloads/EmbsysRevEd_PPTs/Chap_3Lesson18EmsysNew.pdf
[3] http://www.nxp.com/documents/user_manual/UM10204.pdf
[4] http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html
[5] http://www.egr.msu.edu/classes/ece480/capstone/fall09/group03/AN_hemmanur.pdf
[6] http://en.wikipedia.org/wiki/Microcontroller
[7] International Journal of Advanced Research in Computer and Communication EngineeringVol.2, Issue10, October 2013"Design of SPI to I2C Protocol Converter and Implementation of Low Power Techniques"
[8] www.mentor.com, Concise Manual for the Questasim simulator.
[9] Debasis Behera "Design and Modeling of I2C Bus Controller.