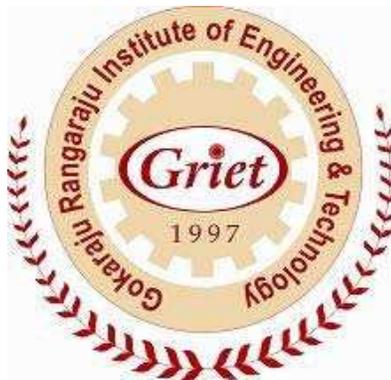


MICROCONTROLLER BASED HOME SECURITY SYSTEM USING WIRELESS ALERTS



DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

BY:

M. Laxmi rohini

(07241A0430)

CONTENTS

| | | |
|--|-----------|----|
| CHAPTER 1 | | |
| INTRODUCTION | | 1 |
| 1.1 Problem definition | | |
| 1.2 Required hardware | | |
| 1.3 Block Diagram | | 2 |
| CHAPTER 2 | | |
| DESIGN FILES | | 3 |
| 2.1 Schematic diagram | CHAPTER 3 | |
| MICROCONTROLLER | | 5 |
| 3.1 AVR ATMEGA 8515 microcontroller | | |
| 3.2 Features | | |
| 3.3 Description | | 6 |
| 3.4 ATmega8515 Processor architecture | | |
| 3.5 Pin diagram | | 7 |
| 3.6 Pin description | | |
| 3.7 Architectural Overview | | 9 |
| 3.8 Register Set | | |
| CHAPTER 4 | | |
| COMPONENTS USED AT THE TRANSMITTER MODULE | | 14 |
| 4.1 Passive infra red sensor | | |
| 4.2 Infrared sensor | | |
| 4.3 Wireless Magnetic Door / Window Contact Sensor | | 18 |
| 4.5 Transmitter (433 MHz RF transmitter STT-433)—RST 433TX | | 19 |
| CHAPTER 5 | | |
| COMPONENTS USED AT THE RECEIVER MODULE | | 23 |
| 5.1 Receiver (433 MHz RF Receiver STT-433) | | |
| 5.2 LCD interfacing with microcontroller | | 25 |
| 5.3 Buzzer | | 32 |
| CHAPTER 4 | | |
| GETTING STARTED WITH ‘C’ PROGRAMMING FOR THE “ATMEL AVR MICROCONTROLLER “ | | 34 |
| 6.1 Installing tools for C programming | | |
| 6.2 Using AVR Studio for C programming | | |
| 6.3 Creating an AVR Studio project | | |
| 6.4 Compiling C code to HEX file | | 37 |
| CHAPTER 5 | | |

| | |
|--|----|
| PROGRAMMING | 40 |
| 7.1 Program at the transmitter Microcontroller | |
| 7.2 Program at the receiver Microcontroller | 43 |
| CHAPTER 8 | |
| CONCLUSION | 51 |

1. INTRODUCTION

1.1 PROBLEM DEFINITION

Security is a prime concern in our day-today life. Everyone wants to be as much secure as possible. Knowing your home is protected provides peace of mind both when you are away and when you are home. Security is important even if you have outstanding public safety agencies (police, fire, highway patrol, etc.) in your area. There are far more homes than there are police officers, not to mention a plethora of skilled thieves. So we would like to implement our project to do everything possible to make your house secure rather than just relying on others.

With the increasing busy schedules people rely on machines to support them in this modern world. It led to the need for intelligence to these machines in every area. Here the case is with home security.

Our project involves an access control for doors and windows forming a vital link in a security chain implemented using two fully controlled 8 bit microcontrollers 8515. The Microcontroller based Home Security System can be adopted at Home, it has various types of Sensors. The other microcontroller can be carried along with the user or can be placed at his working place, it has a buzzer and an LCD .In our Project we have Passive infra red sensor (PIR sensor), Infra red sensor (IR sensor), Magnetic sensor switch sensor (Reed sensor switch).

The Microcontroller at the transmitter end will continuously monitors all the Sensors and if any security problem is found then the Microcontroller at the receiver end will switch on the Buzzer (Alarm) and the type of problem is displayed on the LCD. This equipment uses low power and operates in real time.

Hence this project “Home security system using wireless alerts” came to existence. Detail study of the project is given in the following chapters.

1.2 REQUIRED HARDWARE

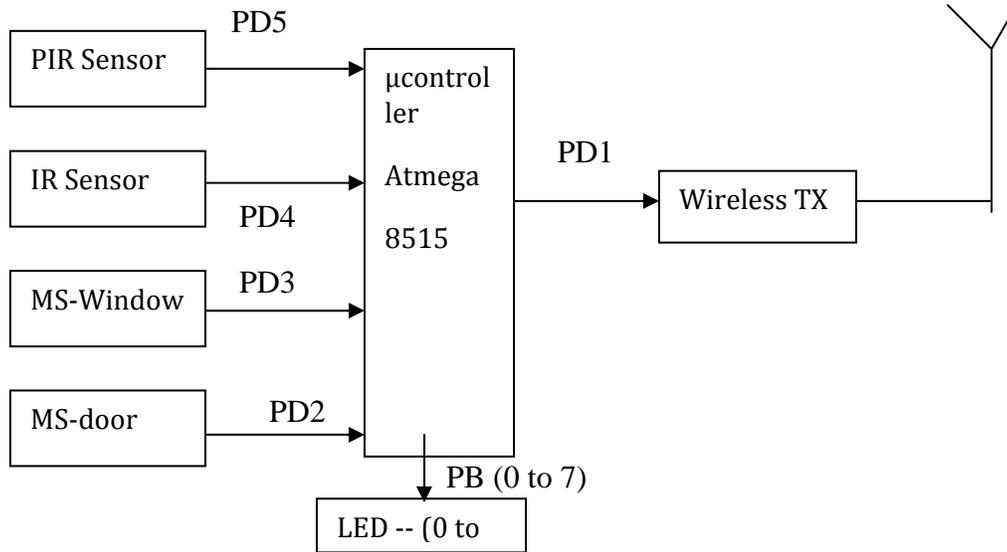
List of hardware

1. Passive infra red sensor
2. Infra red sensor
3. Magnetic sensor switch (Reed switch) →2 in number
4. AVR Atmega8515 microcontroller →2 in number
5. RST 433 TX
6. RST 433 RX
7. LCD

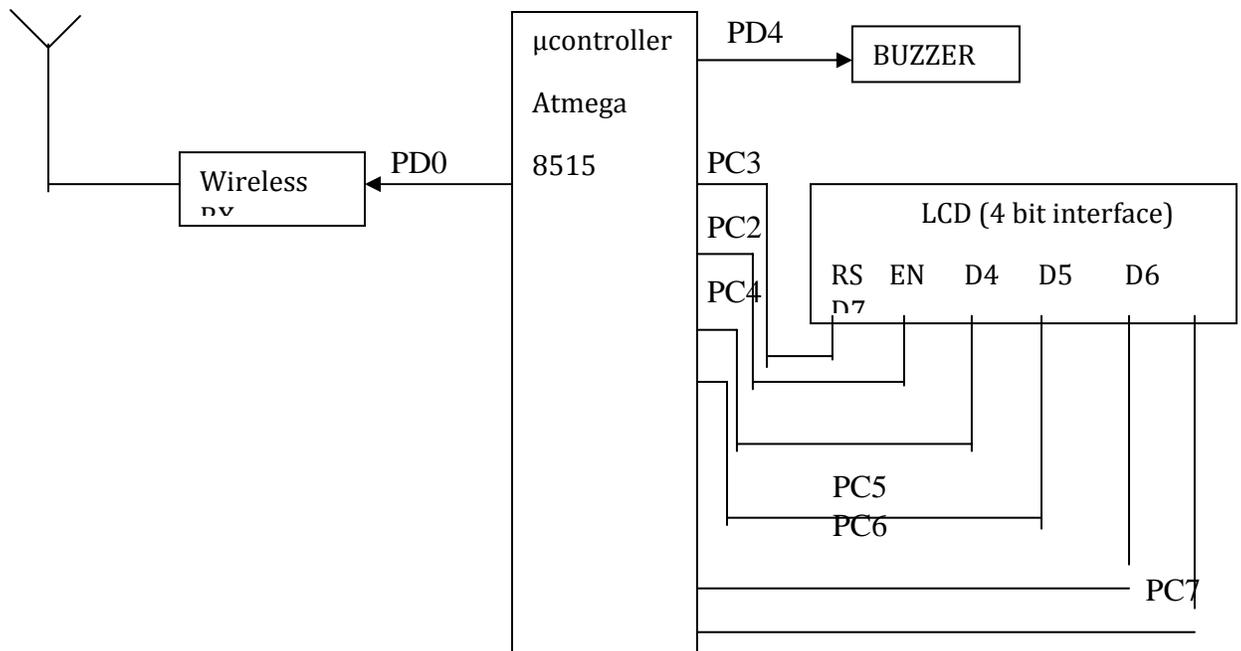
- 8. AVR STUDIO software
- 9. LED
- 10. BUZZER

1.3 Block Diagram

❖ **Transmitter**



❖ **Receiver**

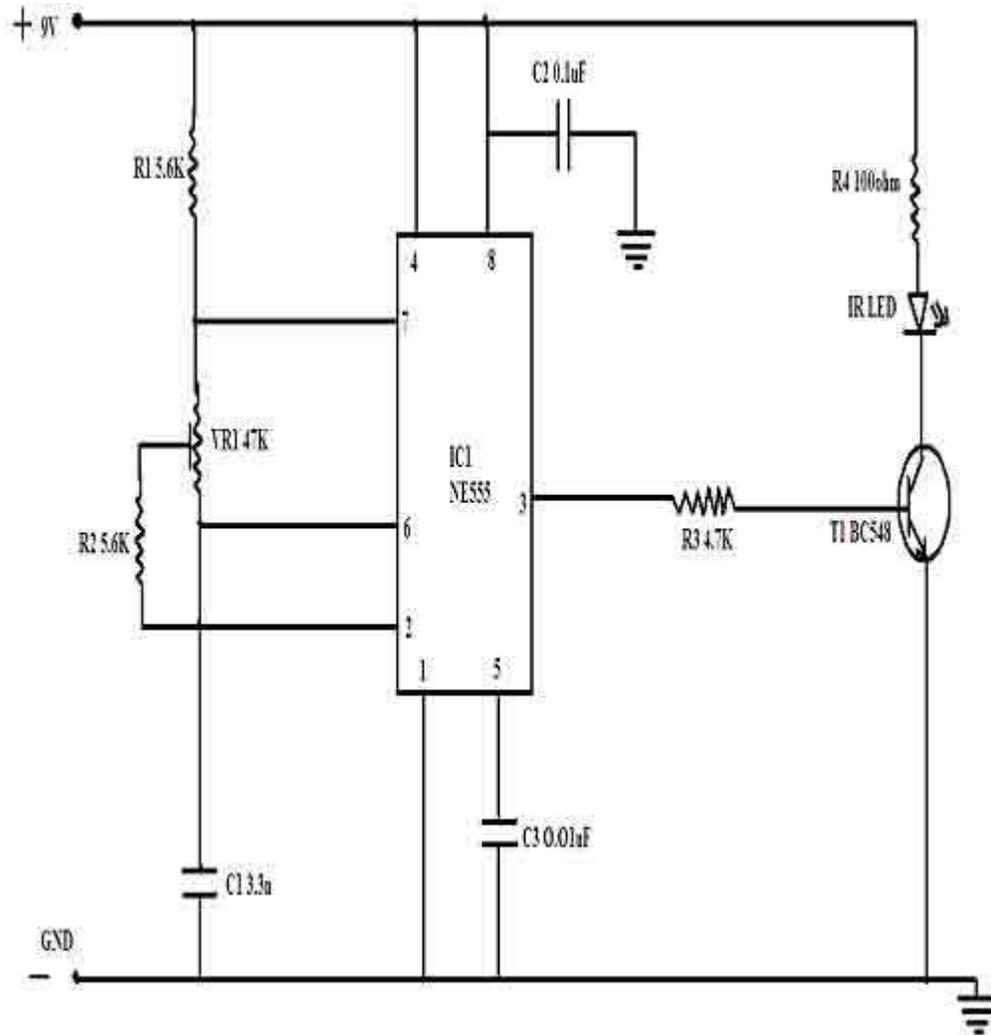


2 .DESIGN FILES

2.1 SCHEMATIC DIAGRAM

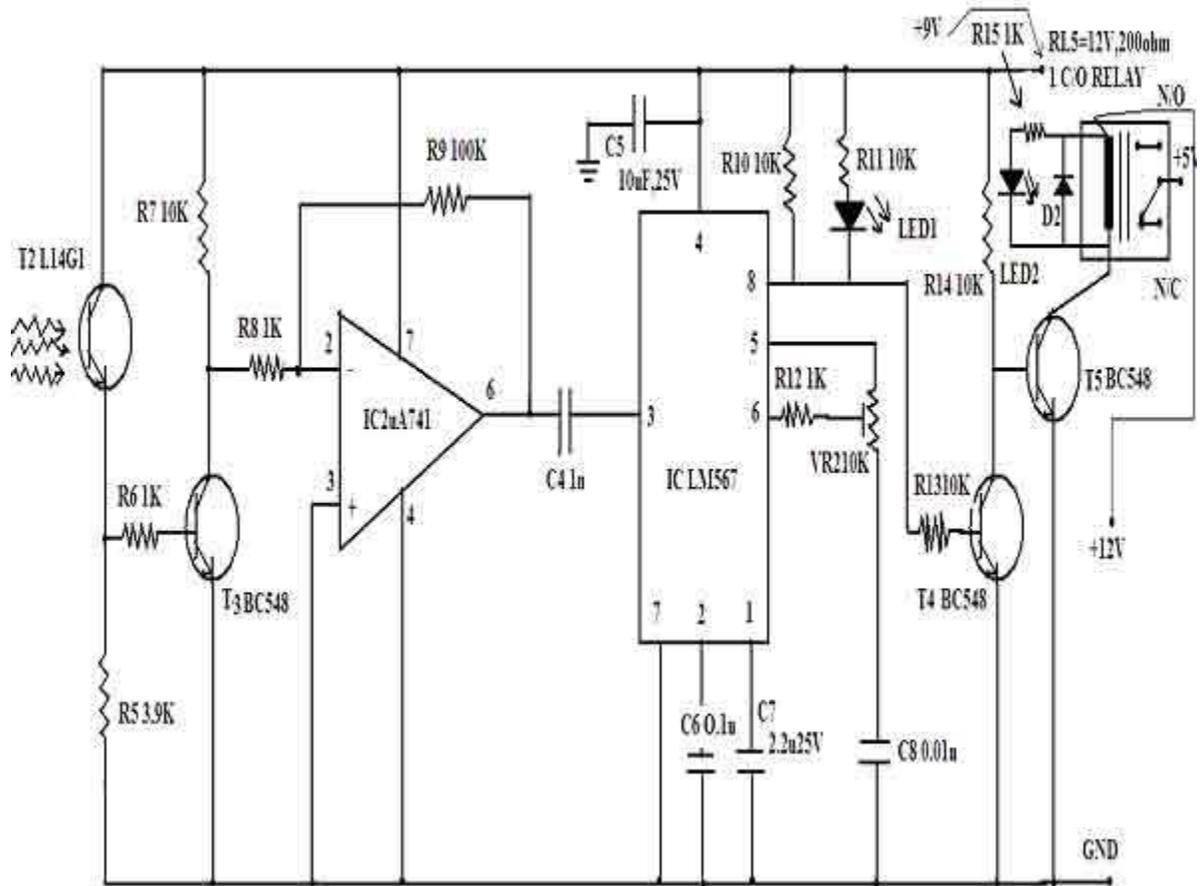


Transmitter





Receiver



3.MICROCONTROLLER

A Microcontroller is a single chip micro-computer that contains all the components such as the CPU, RAM, some form of ROM, I/O ports and timers. Unlike a general-purpose computer, which also includes all of these components a microcontroller is designed for a very specific task--to control a particular system. Micro controllers are sometimes called "*embedded microcontrollers*", which just means that they are part of an embedded system-- that is, one part of a large devices or system.

A Microprocessor is a general purpose digital computer with central processing unit(CPU),which contains arithmetic and logical unit (ALU) ,a program counter (PC),a stack pointer(SP),some working registers ,a clock timing circuit, and interrupt circuit. The main disadvantage of micro processor is that it has no on-chip memory, so we are going for micro controller since it has on-board programmable ROM and I/O that can be programmed for various control functions.

3.1 AVR ATMEGA 8515 microcontroller

3.2Features

- ❖ High-performance, Low-power AVR® 8-bit Microcontroller
- ❖ RISC Architecture
 - 130 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16 MHz
 - On-chip 2-cycle Multiplier
- ❖ Nonvolatile Program and Data Memories
 - 8K Bytes of In-System Self-programmable 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
 - 512 Bytes EEPROM
- ❖ Endurance: 100,000 Write/Erase Cycles
 - 512 Bytes Internal SRAM
 - Up to 64K Bytes Optional External Memory Space
 - Programming Lock for Software Security
- ❖ Peripheral Features
 - One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture
- ❖ Mode
 - Three PWM Channels
 - Programmable Serial USART
 - Master/Slave SPI Serial Interface
 - Programmable Watchdog Timer with Separate 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
 - Three Sleep Modes: Idle, Power-down and Standby
- ❖ I/O and Packages
 - 35 Programmable I/O Lines
 - 40-pin PDIP, 44-lead TQFP, 44-lead PLCC, and 44-pad QFN/MLF
- ❖ Operating Voltages
 - 2.7 - 5.5V for ATmega8515L
 - 4.5 - 5.5V for ATmega8515
- ❖ Speed Grades
 - 0 - 8 MHz for ATmega8515L
 - 0 - 16 MHz for ATmega8515

3.3 Description

The ATmega8515 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8515 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

3.4 ATmega8515 Processor architecture

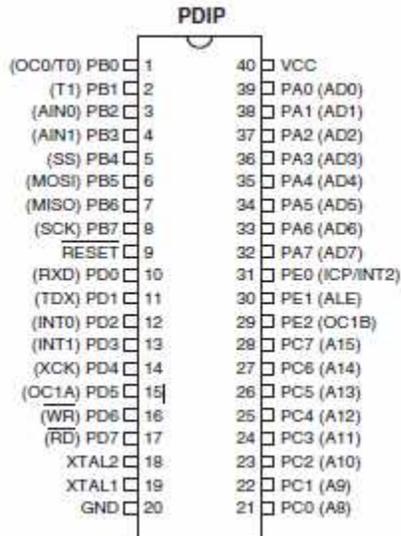
The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega8515 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes EEPROM, 512 bytes SRAM, an External memory interface, 35 general purpose I/O lines, 32 general purpose working registers, two flexible Timer/Counters with compare modes, Internal and External interrupts, a Serial Programmable USART, a programmable Watchdog Timer with internal Oscillator, SPI serial port and three software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and Interrupt system to continue functioning. The Power-down mode saves the Register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the Program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-programmable Flash on a monolithic

chip, the Atmel ATmega8515 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega8515 is supported with a full suite of program and system development tools including: C Compilers, Macro assemblers, Program debugger/simulators, In-circuit Emulators, and Evaluation kits.

3.5 Pin diagram



3.6 Pin description

VCC: Digital supply voltage.

GND : Ground.

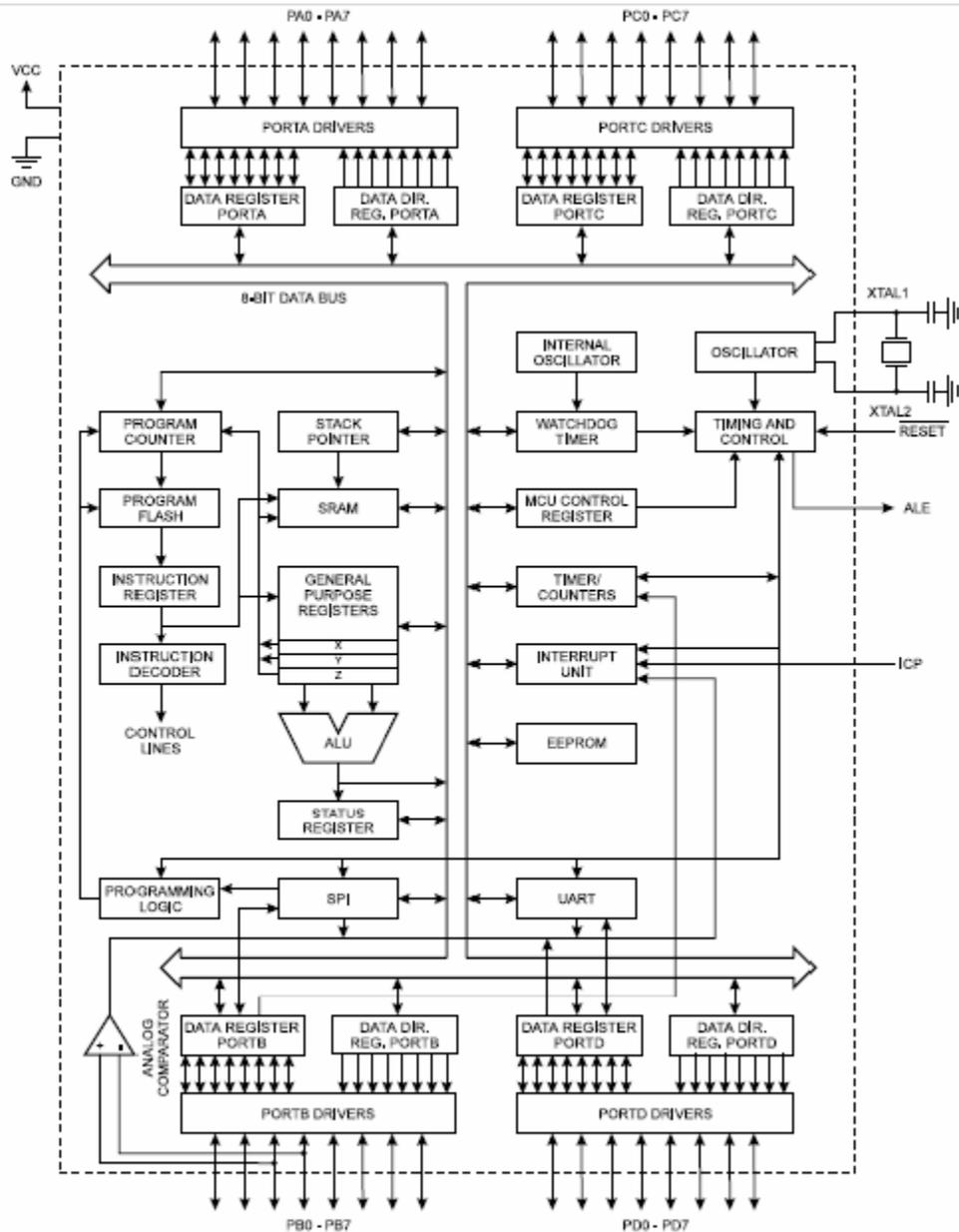
Port A (PA7..PA0): Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The PortA pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0) :Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C (PC7..PC0) :Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D (PD7..PD0) :Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E (PE2..PE0) Port E is a 3-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.



RESET Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running.

XTAL1 Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2 Output from the inverting Oscillator amplifier

| | | | | | | | | |
|---------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| PORTB | PORT B7 | PORT B6 | PORT B5 | PORT B4 | PORT B3 | PORT B2 | PORT B1 | PORT B0 |
| Read/Write | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

PORTB is initialized to-F0H

General Interrupt Control Register—GICR

| | | | | | | | | |
|---------------|-------------|-------------|-------------|---|---|---|--------------|-------------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GICR | INT1 | INT0 | INT2 | - | - | - | IVSEL | IVCE |
| Read/Write | R/W | R/W | R/W | R | R | R | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

GICR is initialized to C0H-int0 and 1 are enabled

MCU Control Register

| | | | | | | | | |
|---------------|------------|--------------|-----------|------------|--------------|--------------|--------------|--------------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MCUCR | SRE | SRW10 | SE | SM1 | ISC11 | ISC10 | ISC01 | ISC00 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

For falling edge 0x0A :: for rising edge 0x0F

USART Baud Rate Register –UBRRL and UBRRH

| | | | | | | | | |
|---------------|------------------|-----|-----|-----|-------------------|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| UBRRH | URSEL | - | - | - | UBRR[11:8] | | | |
| UBRRL | UBRR[7:0] | | | | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Read/write | R/W | R | R | R | R/W | R/W | R/W | R/W |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

UBRRH-1

UBRRL-159H for baud rate of 1200 UBRR value =415=256+159 .The UBRRH Register shares the same I/O location as the UCSRC Register .See the “Accessing UBRRH/UCSRC Registers” which describes how to access this register.

- **Bit 15 –URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

- **Bit 14:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0:USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRL contains the eight least significant bits of the USART Baud rate. Ongoing transmissions by the Transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRL will trigger as an immediate update of the baud rate prescaler.

USART Control and Status Register B—UCSRB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------------|--------------|--------------|--------------|-------------|-------------|--------------|-------------|-------------|
| UCSRB | RXCIE | TXCIE | UDRIE | RXEN | TXEN | UCSZ2 | RXB8 | TXB8 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

UCSRB is initialized to 08H-only tx is enabled

- **Bit 7 – RXCIE :RX Complete Interrupt Enable**

Writing this bit to one, enables interrupt on the RXC flag. A USART receive Complete Interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

- **Bit 6—TXCIE :TX Complete Interrupt Enable**

Writing this bit to one enable interrupt on the TXC flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCRA is set.

- **Bit 4—RXEN : Receiver Enable**

Writing this bit to one enable the USART receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receiver buffer invalidating the FE, DOR and PE flags.

- **Bit 3—TXEN :Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed. For example, when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD port.

- **Bit 2 – UCSZ2 : Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in the UCSRC sets the number of data bits (character size) in a frame the receiver and transmitter use.

- **Bit 1 – RXB8 : receive data Bit 8**

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

- **Bit 0 – TXB8 : Transmit Data Bit 8**

TXB8 is the ninth data bit in the character to be transmitted when operating with serial Frames with 9 data bits. Must be written before writing the low bits to UDR.

USART Control and Status Register C – UCSRC

| | | | | | | | | |
|---------------|--------------|--------------|-------------|-------------|-------------|--------------|--------------|--------------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UCSRC | URSEL | UMSEL | UPM1 | UPM0 | USBS | UCSZ1 | UCSZ0 | UCPOL |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

UCSRC is initialized to 86H - * bit data.

Port D Data Direction Register – DDRD

| | | | | | | | | |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DDRD | DDD7 | DDD6 | DDD5 | DDD4 | DDD3 | DDD2 | DDD1 | DDD0 |
| Read/Write | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

USART I/O Data Register – UDR

| | | | | | | | | |
|---------------|-----------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UDR(Read) | RXB[7:0] | | | | | | | |
| UDR(Write) | TXB[7:0] | | | | | | | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The USART Transmit Data Buffer register and USART Receive Data Buffer Register share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive data Buffer Register (RXB). For 5-, 6- or 7-bit character the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

Timer/Counter Control Register – TCCR0

| | | | | | | | | |
|-----|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---------------|-------------|--------------|--------------|--------------|--------------|-------------|-------------|-------------|
| TCCR0 | FOC0 | WGM00 | COM01 | COM00 | WGM01 | CS02 | CS01 | CS00 |
| Read/Write | W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Timer/Counter Interrupt Mask Register – TIMSK

| | | | | | | | | |
|---------------|--------------|---------------|---------------|-----|---------------|-----|--------------|--------------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMSK | TOIE1 | OCIE1A | OCIE1B | - | TICIE1 | - | TOIE0 | OCIE0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Bit 1 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 0 – OCIE0 :Timer/Counter0 Output Compare Match Interrupt Enable**

When the OCIE0 bit is written to one, and the I-bit in the status register is set (one), the Timer/Counter0 compare Match interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCFO bit is set in the Timer/Counter Interrupt Flag register – TIFR.

Timer/Counter Register – TCNT0

| | | | | | | | | |
|---------------|-------------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TCNT0 | TCNT0[7:0] | | | | | | | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

4. COMPONENTS USED AT THE TRANSMITTER MODULE

4.1 Passive infra red sensor

The PIR Sensor detects motion up to 20 feet away by using a Fresnel lens and infrared-sensitive element to detect changing patterns of passive infrared emitted by

objects in its vicinity. Inexpensive and easy to use, it's ideal for alarm systems, motion-activated lighting, and holiday props. The PIR Sensor is compatible with all Parallax microcontrollers.



4.1.1 FEATURES

- ❖ Detection range up to 20 feet away
- ❖ Single bit output
- ❖ Jumper selects single or continuous trigger output mode
- ❖ 3-pin SIP header ready for breadboard or through-hole projects
- ❖ Small size makes it easy to conceal
- ❖ Compatible with BASIC Stamp, Propeller and many other microcontrollers

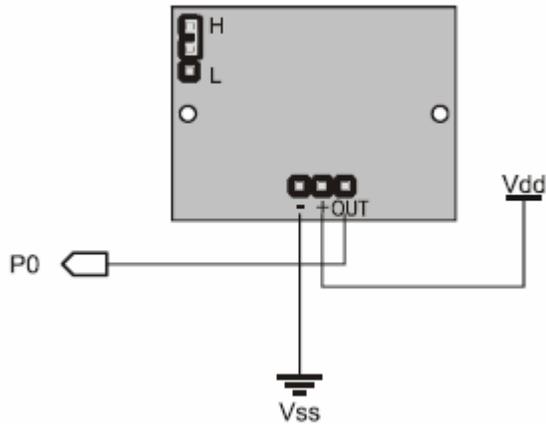
4.1.2 APPLICATIONS

- ❖ Alarm Systems
- ❖ Holiday animated Props
- ❖ Motion-activated lighting

4.1.3 KEY SPECIFICATIONS

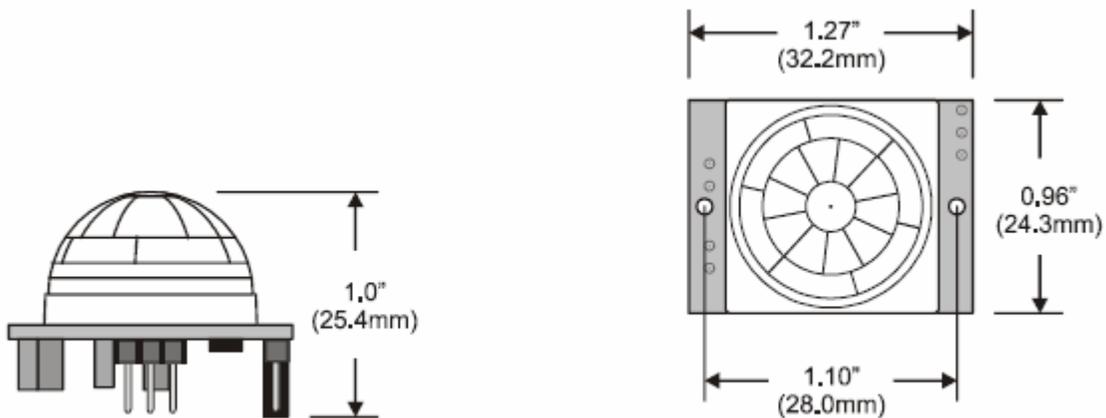
- ❖ Power requirements: 3.3 to 5 VDC
- ❖ Communication: single bit high/low output
- ❖ Dimensions: 1.27 x 0.96 x 1.0 in (32.2 x 24.3 x 25.4 mm)
- ❖ Operating temp range: +32 to +121 °F (0 to +50 °C)

❖ 4.1.4 Quick Start Circuit



Note: The sensor is active high when the jumper (shown in the upper left) is in either position.

4.1.5 Module Dimensions



4.1.6 Theory of Operation

Pyroelectric devices, such as the PIR sensor, have elements made of a crystalline material that generates an electric charge when exposed to infrared radiation. The changes in the amount of infrared striking the element change the voltages generated, which are measured by an on-board amplifier. The device contains a special filter called a Fresnel lens, which focuses the infrared signals onto the element. As the ambient infrared signals change rapidly, the on-board amplifier trips the output to indicate motion.

4.1.7 Pin Definitions and Ratings

| Pin | Name | Function |
|-----|--------|--|
| - | GND | Connects to Ground or Vss |
| + | V+ | Connects to +5 VDC or Vdd |
| OUT | Output | Connects to an I/O pin set to INPUT mode |

4.1.8 Connecting and Testing

Connect the 3-pin header to your circuit so that the minus (-) pin connects to ground or V_{ss} , the plus (+) pin connects to +5 volts or V_{dd} and the OUT pin connects to your microcontroller's I/O pin. If you use the Board Of Education, be sure the servo voltage jumper (located between the 2 servo header blocks) is in the V_{dd} position, not V_{in} . If you do not have this jumper on your board you should manually connect to V_{dd} through the breadboard.

You may also plug the sensor directly into the edge of the breadboard and connect the signals from there. Remember the position of the pins when you plug the sensor into the breadboard.

Once the sensor warms up (settles) the output will remain low until there is motion, at which time the output will swing high for a couple of seconds, then return low. If motion continues the output will cycle in this manner until the sensors line of sight of still again.

4.1.9 Calibration

The PIR Sensor requires a 'warm-up' time in order to function properly. This is due to the settling time involved in 'learning' its environment. This could be anywhere from 10-60 seconds. During this time there should be as little motion as possible in the sensors field of view.

4.1.10 Sensitivity

The PIR Sensor has a range of approximately 20 feet. This can vary with environmental conditions. The sensor is designed to adjust to slowly changing conditions that would happen normally as the day progresses and the environmental conditions change, but responds by toggling its output when sudden changes occur, such as when there is motion.

4.2 Infrared sensor



4.2.1 FEATURES

- ❖ Easy to build and easy to calibrate
- ❖ Provides a detection range of 35cm
- ❖ Doesn't provide ambient light immunity
- ❖ To detect very small voltage changes accurately an Op-Amp(Operational Amplifier) is used.

4.2.2 APPLICATIONS

- ❖ Indoor applications(no imp ambient light is present)

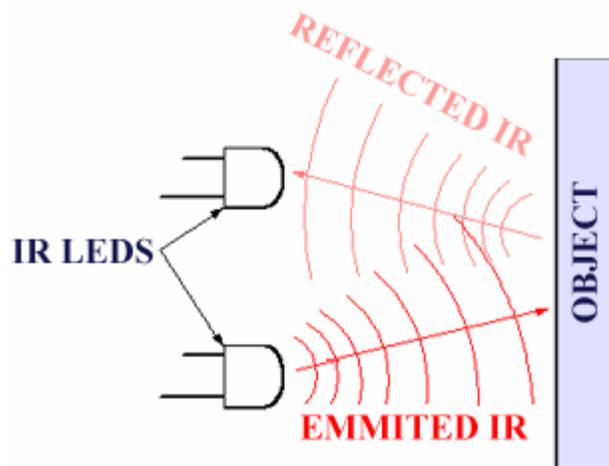
- ❖ To measure speed of object(moving at a very high speed)
- ❖ Industry
- ❖ Tachometers

4.2.3 Object detection using IR light

It is the same principle in ALL Infra-Red proximity sensors. The basic idea is to send infra red light through IR-LEDs, which is then reflected by any object in front of the sensor.

Then all you have to do is to pick-up the reflected IR light. For detecting the reflected IR light, we are going to use a very original technique: we are going to use another IR-LED, to detect the IR light that was emitted from another led of the exact same type!

This is an electrical property of Light Emitting Diodes (LEDs) which is the fact that a led Produce a voltage difference across its leads when it is subjected to light. As if it was a photo-cell, but with much lower output current. In other words, the voltage generated by the led's can't be - in any way - used to generate electrical power from light, It can barely be detected.



4.2.4 The electronic circuit

Two different designs are proposed, each one of them is more suitable for different applications. The main difference between the 2 designs is the way infra-red (IR) light is sent on the object. The receiver part of the circuit is exactly the same in both designs.

Note: Both the sender and the receiver are constructed on the same board. They are separated in the schematics for simplification.

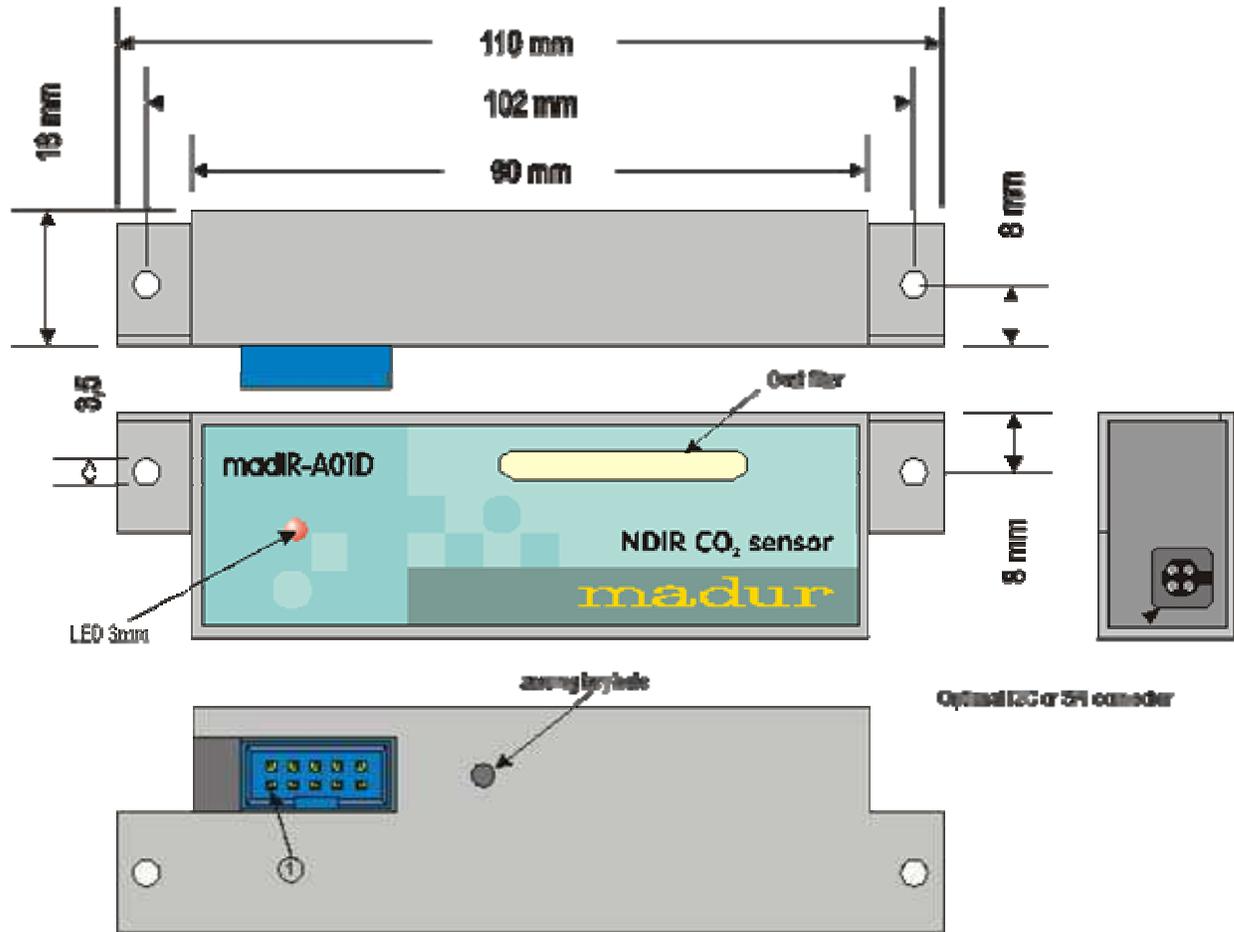
4.2.5 Specifications

- ❖ Sensor type = Reflective IR
- ❖ IR detector = Panasonic PNA4602M
- ❖ IR LED type = Narrow focus 10°
- ❖ I/O required = Three digital lines, 2 outputs, 1 input
- ❖ Minimum range = Approximately 4"
- ❖ Maximum range = Approximately 26"
- ❖ Input voltage = 5vdc regulated
- ❖ Current requirements = 8mA
- ❖ PC board size = 2.3" x .75"

4.2.6 Power requirements

| Version | DC supply range | AC supply range | Vout range [max] | Average supply current / power [max]l |
|------------------------------|------------------------|------------------------|-------------------------|--|
| Standard | 13.5..18V | N/A | 0..10V | 45mA*VDC |
| Low Voltage | 7.5..18V | N/A | 0..5V | 35mA*VDC |
| Extended Supply, Standard | 13..32V | 9..24VAC | 0..10V | 1W |
| Extended Supply, Low Voltage | 7..32V | 6..24VAC | 0..5V | 0.7W |

4.2.7 External view and dimensions



4.3 Wireless Magnetic Door / Window Contact Sensor

The Wireless Magnetic Door / Window Contact Sensor is a protective device that can send notification when the opening and closure of doors and windows occurs. It is designed to send a wireless signal to a compatible home security system when the contact between the transmitter and corresponding magnetic sensor is broken.



- ❖ Magnetic contact used to monitor normally closed entry points such as doors and windows
- ❖ Wirelessly communicates with a compatible home security system up to 300 feet away
- ❖ Adjustable radio frequency settings and built-in technology prevent missed signals and transmission errors.

4.3.1 Reed Switch features

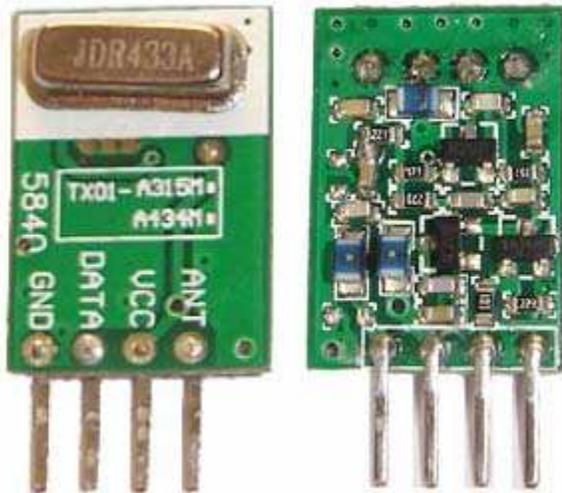
- ❖ Ability to switch up to 10,000 Volts
- ❖ Ability to switch currents up to 5 Amps
- ❖ Ability to switch or carry as low as 10 nanoVolts without signal loss
- ❖ Ability to switch or carry as low as 1 femptoAmp without signal loss
- ❖ Ability to switch or carry up to 7 GigaHertz with minimal signal loss
- ❖ Isolation across the contacts up to 1015 W
- ❖ Contact resistance (on resistance) typical 50 milliOhms (mW)
- ❖ In its off state it requires no power or circuitry
- ❖ Ability to offer a latching feature
- ❖ Operate time in the 100 ms to 300 ms range
- ❖ Ability to operate over extreme temperature ranges from -55°C to 200°C
- ❖ Ability to operate in all types of environments including air, water, vacuum, oil, fuels, and dust laden atmospheres
- ❖ Ability to withstand shocks up to 200 Gs
- ❖ Ability to withstand vibration environments of 50 Hz to 2000 Hz at up to 30 Gs
- ❖ Long life. With no wearing parts, load switching under 5 Volts at 10 mA, will operate well into the billions of operations

4.3.2 Specifications

| Specifications | |
|-------------------------------|---|
| Manufacturer: | Visonic |
| Manufacturer Product No.: | MCT-302 |
| Dimensions: | 3 3/16" x 7/8" x 15/16" (81mm x 22mm x 23.5mm) |
| Weight: | 1.2 oz. (34g), excluding battery |
| Battery Weight: | 0.45 oz. (13g) |
| Operating Frequency: | 315 MHz |
| Transmitter's ID Code: | 24-bit digital word, over 16 million combinations, pulse width modulation |
| Overall Message Length: | 36 bits |
| Alarm Inputs: | 2: one internal, one external, each with a separate 24-bit transmitter |
| Auxiliary Input Circuit Type: | N.C. / E.O.L., selected with DIP switch |
| E.O.L. Resister Required: | 47 kilo ohms |
| Message Repetition: | Once every 3 minutes or one-shot, selectable via on-board DIP switch |

| | |
|--------------------------|---|
| Supervision: | Signaling at 60-minute intervals via reed switch or aux. input, including battery supervision |
| Tamper Reporting: | Every 3 minutes until restored |
| Power Source: | 3V Lithium CR-2-type battery, Panasonic or Sanyo only (not included) |
| Battery Life Expectancy: | 3 years (typical use) |
| Operating Temperature: | 32° F to 120° F (0° C to 49° C) |
| Standards: | FCC compliant |

4.5 Transmitter (433 MHz RF transmitter STT-433)—RST 433TX



4.5.1 Introduction

The STT-433 is ideal for remote control applications where low cost and longer range is required. The transmitter operates from a 1.5-12V supply, making it ideal for battery-powered applications. The transmitter employs a SAW-stabilized oscillator, ensuring accurate frequency control for best range performance. Output power and harmonic emissions are easy to control, making FCC and ETSI compliance easy. The manufacturing-friendly SIP style package and low-cost make the STT-433 suitable for high volume applications.

4.5.2 Features

- ❖ 433.92 MHz Frequency
- ❖ Low Cost
- ❖ 1.5-12V operation
- ❖ 11mA current consumption at 3V
- ❖ Small size
- ❖ 4 dBm output power at 3V

4.5.3 Applications

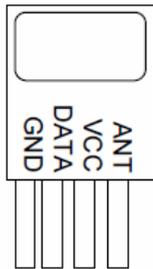
- ❖ Remote Keyless Entry (RKE)
- ❖ Remote Lighting Controls
- ❖ On-Site Paging

- ❖ Asset Tracking
- ❖ Wireless Alarm and Security Systems
- ❖ Long Range RFID
- ❖ Automated Resource Management

4.5.4 Specifications

| Parameter | Symbol | Min | Typ. | Max | Unit |
|---------------------------------|--------|-----|--------------------------------------|-----|----------|
| Operating Voltage | Vcc | 1.5 | 3.0 | 12 | Volts DC |
| Operating Current Data = VCC | Icc | - | 11mA @3V 59mA @5V | - | mA |
| Operating Current Data = GND | Icc | - | 100 | - | uA |
| Frequency Accuracy | TOL fc | -75 | 0 | +75 | Khz |
| Center Frequency | Fc | - | 433 | - | Mhz |
| RF Output Power | | - | 4 dBm@3V (2 mW) 16 dBm@5V (39 mW) | | dBm / mW |
| Data Rate | | 200 | 1K | 3K | BPS |
| Temperature | | -20 | | +60 | Deg. C |
| Power up delay | | | 20 | | ms |

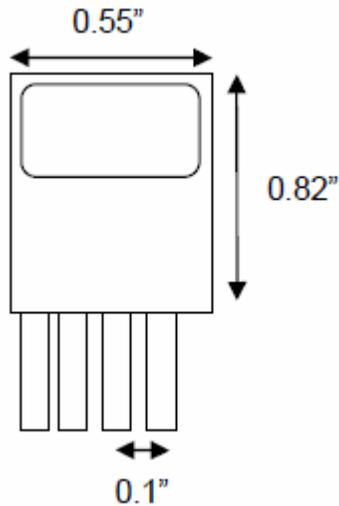
4.5.5 Pin description



| Pin Name | Description |
|----------|---|
| ANT | 50 ohm antenna output. The antenna port impedance affects output power and harmonic emissions. An L-C low-pass filter may be needed to sufficiently filter harmonic emissions. Antenna can be single core wire of approximately 17cm length or PCB trace antenna. |
| VCC | Operating voltage for the transmitter. VCC should be bypassed with a .01uF ceramic capacitor and filtered with a 4.7uF tantalum capacitor. Noise on the power supply will degrade transmitter noise performance. |
| DATA | Digital data input. This input is CMOS compatible and should be driven with CMOS level inputs. |
| GND | Transmitter ground. Connect to ground plane. |

4.5.6. Mechanical drawing

{Dimensions in inches}



4.5.7 Operation

OOK(On Off Keying) modulation is a binary form of amplitude modulation. When a logical 0 (data line low) is being sent, the transmitter is off, fully suppressing the carrier. In this state, the transmitter current is very low, less than 1mA. When a logical 1 is being sent, the carrier is fully on. In this state, the module current consumption is at its highest, about 11mA with a 3V power supply.

OOK is the modulation method of choice for remote control applications where power consumption and cost are the primary factors. Because OOK transmitters draw no power when they transmit a 0, they exhibit significantly better power consumption than FSK transmitters.

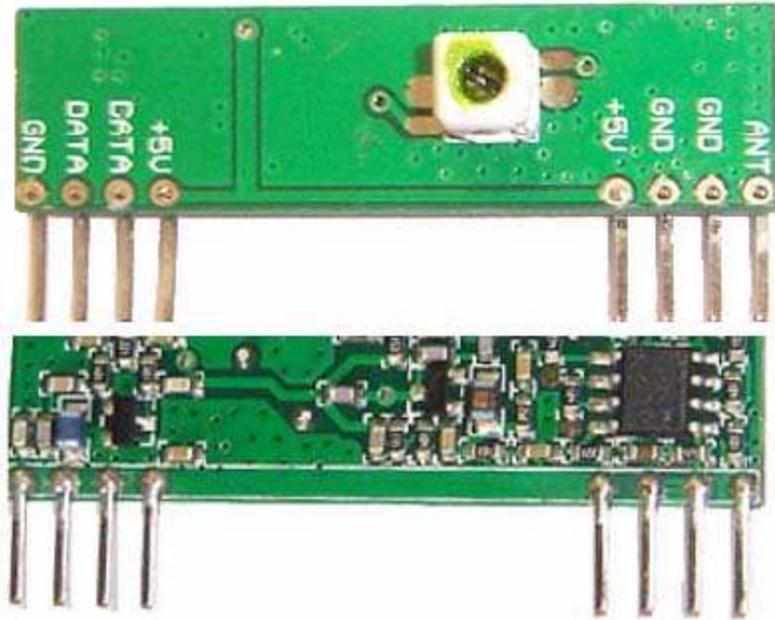
OOK data rate is limited by the start-up time of the oscillator. High-Q oscillators which have very stable center frequencies take longer to start-up than low-Q oscillators. The start-up time of the oscillator determines the maximum data rate that the transmitter can send.

4.5.8 Data Rate

The oscillator start-up time is on the order of 40uSec, which limits the maximum data rate to 4.8 kbit/sec.

5.COMPONENTS USED AT THE RECEIVER MODULE

5.1 RECEIVER (433 MHz RF Receiver STT-433)



5.1.1 Introduction

The STR-433 is ideal for short-range remote control applications where cost is a primary concern. The receiver module requires no external RF components except for the antenna. It generates virtually no emissions, making FCC and ETSI approvals easy. The super-regenerative design exhibits exceptional sensitivity at a very low cost. The manufacturing-friendly SIP style package and low-cost make the STR-433 suitable for high volume applications.

5.1.2 Features

- ❖ Low Cost
- ❖ 5V operation
- ❖ 3.5mA current drain
- ❖ No External Parts are required
- ❖ Receiver Frequency: 433.92 MHZ
- ❖ Typical sensitivity: -105dBm
- ❖ IF Frequency: 1MHz

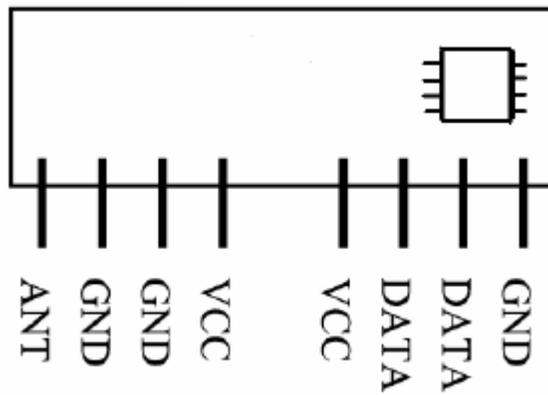
5.1.3 Applications

- ❖ Car security system
- ❖ Sensor reporting
- ❖ Automation system
- ❖ Remote Keyless Entry (RKE)
- ❖ Remote Lighting Controls
- ❖ On-Site Paging
- ❖ Asset Tracking
- ❖ Wireless Alarm and Security Systems
- ❖ Long Range RFID
- ❖ Automated Resource Management

5.1.4 Specifications

| Parameter | Symbol | Min | Typ. | Max | Unit |
|-----------------------|--------|-----|--------|-----|--------|
| Operating Voltage | Vcc | 4.5 | 5.0 | 5.5 | VDC |
| Operating Current | Icc | - | 3.5 | 4.5 | mA |
| Reception Bandwidth | BW rx | - | 1.0 | - | MHz |
| Center Frequency | Fc | - | 433.92 | - | MHz |
| Sensitivity | - | - | -105 | - | dBm |
| Max Data Rate | - | 300 | 1k | 3K | Kbit/s |
| Turn On Time | - | - | 25 | - | ms |
| Operating Temperature | T op | -10 | - | +60 | °C |

5.1.5 Pin outs



| Pin Name | Description |
|----------|--|
| ANT | Antenna input. |
| GND | Receiver Ground. Connect to ground plane. |
| VCC(5V) | VCC pins are electrically connected and provide operating voltage for the receiver. VCC can be applied to either or both. VCC should be bypassed with a .1µF ceramic capacitor. Noise on the power supply will degrade receiver sensitivity. |
| DATA | Digital data output. This output is capable of driving one TTL or CMOS load. It is a CMOS compatible output. |

5.1.6 Antenna Input

It will support most antenna types, including printed antennas integrated directly onto the PCB and simple single core wire of about 17cm. The performance of the different antennas varies. Any time a trace is longer than 1/8th the wavelength of the frequency it is carrying, it should be a 50 ohm micro strip.

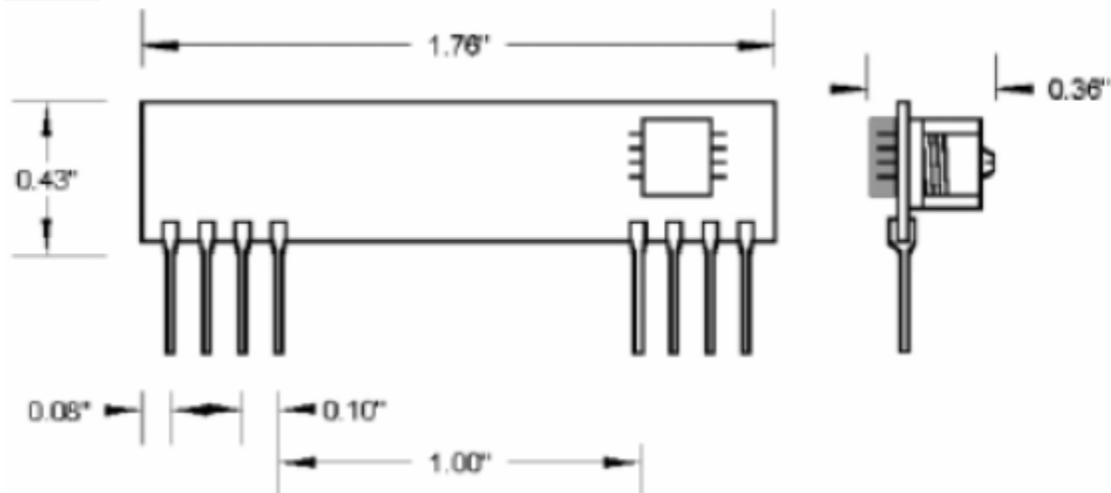
5.1.7 Operation

The STR-433 uses a super-regenerative AM detector to demodulate the incoming AM carrier. A super-regenerative detector is a gain stage with positive feedback greater than unity so that it oscillates. An RC-time constant is included in the gain stage so that when the gain stage oscillates, the gain will be lowered over time proportional to the RC time constant until the oscillation eventually dies. When the oscillation dies, the current draw

of the gain stage decreases, charging the RC circuit, increasing the gain, and ultimately the oscillation starts again. In this way, the oscillation of the gain stage is turned on and off at a rate set by the RC time constant. This rate is chosen to be super-audible but much lower than the main oscillation rate. Detection is accomplished by measuring the emitter current of the gain stage. Any RF input signal at the frequency of the main oscillation will aid the main oscillation in restarting. If the amplitude of the RF input increases, the main oscillation will stay on for a longer period of time, and the emitter current will be higher. Therefore, we can detect the original base-band signal by simply low-pass filtering the emitter current.

The average emitter current is not very linear as a function of the RF input level. It exhibits a $1/\ln$ response because of the exponentially rising nature of oscillator start-up. The steep slope of a logarithm near zero results in high sensitivity to small input signals.

5.1.8 Mechanical drawing (Dimensions in inches)

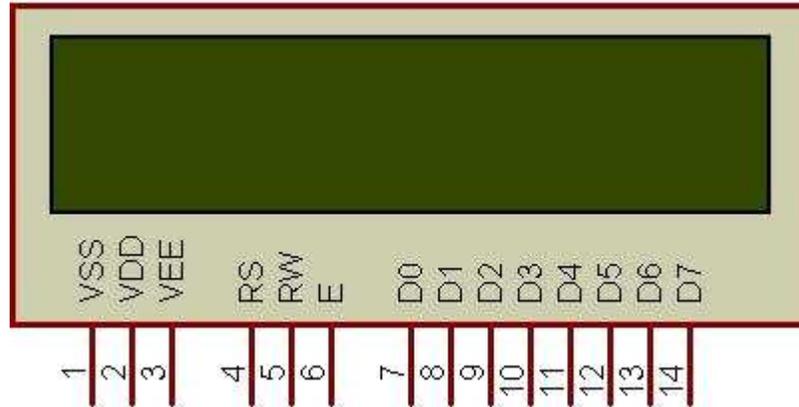


5.2 LCD INTERFACING WITH MICROCONTROLLER

5.2.1 Pin Description

The most commonly used LCDs found in the market today are 1 Line, 2 Line or 4 Line LCDs which have only 1 controller and support at most of 80 characters, whereas LCDs supporting more than 80 characters make use of 2 HD44780 controllers.

Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections). Pin description is



shown in the table

Figure 1: Character LCD type HD44780 Pin diagram

| Pin No. | Name | Description |
|------------|------------|---|
| Pin no. 1 | VSS | Power supply (GND) |
| Pin no. 2 | VCC | Power supply (+5V) |
| Pin no. 3 | VEE | Contrast adjust |
| Pin no. 4 | RS | 0 = Instruction input 1 = Data input |
| Pin no. 5 | R/W | 0 = Write to LCD module 1 = Read from LCD module |
| Pin no. 6 | EN | Enable signal |
| Pin no. 7 | D0 | Data bus line 0 (LSB) |
| Pin no. 8 | D1 | Data bus line 1 |
| Pin no. 9 | D2 | Data bus line 2 |
| Pin no. 10 | D3 | Data bus line 3 |
| Pin no. 11 | D4 | Data bus line 4 |
| Pin no. 12 | D5 | Data bus line 5 |
| Pin no. 13 | D6 | Data bus line 6 |
| Pin no. 14 | D7 | Data bus line 7 (MSB) |

Table 1: Character LCD pins with 1 Controller

5.2.2 Instruction Register (IR) and Data Register (DR)

There are two 8-bit registers in HD44780 controller Instruction and Data register. Instruction register corresponds to the register where you send commands to LCD e.g LCD shift command, LCD clear, LCD address etc. and Data register is used for storing data which is to be displayed on LCD. When send the enable signal of the LCD is asserted, the data on the pins is latched in to the data register and data is then moved automatically to the DDRAM and hence is displayed on the LCD.

Data Register is not only used for sending data to DDRAM but also for CGRAM, the address where you want to send the data, is decided by the instruction you send to LCD.

5.2.3 LCD Commands and Instruction set

Only the instruction register (IR) and the data register (DR) of the LCD can be controlled by the MCU. Before starting the internal operation of the LCD, control

information is temporarily stored into these registers to allow interfacing with various MCUs, which operate at different speeds, or various peripheral control devices. The internal operation of the LCD is determined by signals sent from the MCU. These signals, which include register selection signal (RS), read/write signal (R/W), and the data bus (DB0 to DB7), make up the LCD instructions.

There are four categories of instructions that:

- ❖ Designate LCD functions, such as display format, data length, etc
- ❖ Set internal RAM addresses
- ❖ Perform data transfer with internal RAM
- ❖ Perform miscellaneous functions

| No. | Instruction | Hex | Decimal |
|-----|--|------------|----------|
| 1 | Function Set: 8-bit, 1 Line, 5x7 Dots | 0x30 | 48 |
| 2 | Function Set: 8-bit, 2 Line, 5x7 Dots | 0x38 | 56 |
| 3 | Function Set: 4-bit, 1 Line, 5x7 Dots | 0x20 | 32 |
| 4 | Function Set: 4-bit, 2 Line, 5x7 Dots | 0x28 | 40 |
| 5 | Entry Mode | 0x06 | 6 |
| 6 | Display off Cursor off (clearing display without clearing DDRAM content) | 0x08 | 8 |
| 7 | Display on Cursor on | 0x0E | 14 |
| 8 | Display on Cursor off | 0x0C | 12 |
| 9 | Display on Cursor blinking | 0x0F | 15 |
| 10 | Shift entire display left | 0x18 | 24 |
| 12 | Shift entire display right | 0x1C | 30 |
| 13 | Move cursor left by one character | 0x10 | 16 |
| 14 | Move cursor right by one character | 0x14 | 20 |
| 15 | Clear Display (also clear DDRAM content) | 0x01 | 1 |
| 16 | Set DDRAM address or cursor position on display | 0x80+add* | 128+add* |
| 17 | Set CGRAM address or set pointer to CGRAM location | 0x40+add** | 64+add** |

Table 4: Frequently used commands and instructions for LCD

| Command | Code | | | | | | | | | | Description | Execution Time | |
|-----------------------------|---|-----|------------|-----------------|-----------------|-----|-----|-----|-----|------------------------------------|--|---|-------------|
| | RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 | | | |
| Clear Display | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | Clears the display and returns the cursor to the home position (address 0). | 82µs~1.64ms |
| Return Home | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | * | Returns the cursor to the home position (address 0). Also returns a shifted display to the home position. DD RAM contents remain unchanged. | 40µs~1.64ms |
| Entry Mode Set | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S | | Sets the cursor move direction and enables/disables the display. | 40µs |
| Display ON/OFF Control | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B | | Turns the display ON/OFF (D), or the cursor ON/OFF (C), and blink of the character at the cursor position (B). | 40µs |
| Cursor & Display Shift | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | * | * | | Moves the cursor and shifts the display without changing the DD RAM contents. | 40µs |
| Function Set | 0 | 0 | 0 | 0 | 1 | DL | N\$ | F | * | # | | Sets the data width (DL), the number of lines in the display (L), and the character font (F). | 40µs |
| Set CG RAM Address | 0 | 0 | 0 | 1 | A _{CG} | | | | | | | Sets the CG RAM address. CG RAM data can be read or altered after making this setting. | 40µs |
| Set DD RAM Address | 0 | 0 | 1 | A _{DD} | | | | | | | Sets the DD RAM address. Data may be written or read after making this setting. | 40µs | |
| Read Busy Flag & Address | 0 | 1 | BF | AC | | | | | | | Reads the BUSY flag (BF) indicating that an internal operation is being performed and reads the address counter contents. | 1µs | |
| Write Data to CG or DD RAM | 1 | 0 | Write Data | | | | | | | Writes data into DD RAM or CG RAM. | 46µs | | |
| Read Data from CG or DD RAM | 1 | 1 | Read Data | | | | | | | Reads data from DD RAM or CG RAM. | 46µs | | |
| | I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift. S/C = 1: Display shift S/C = 0: cursor move R/L = 1: Shift to the right. R/L = 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5x10 dots F = 0: 5 x 7 dots BF = 1: Busy BF = 0: Can accept data # Set to 1 on 24x4 modules \$ With KS0072 is Address Mode. | | | | | | | | | | DD RAM: Display data RAM CG RAM: Character generator RAM A _{CG} : CG RAM Address A _{DD} : DD RAM Address Corresponds to cursor address. AC: Address counter Used for both DD and CG RAM address. | Execution times are typical. If transfers are timed by software and the busy flag is not used, add 10% to the above times. | |

Table 3: Command and Instruction set for LCD type HD44780

5.2.4 LCD Initialization

Before using the LCD for display purpose, LCD has to be initialized either by the internal reset circuit or sending set of commands to initialize the LCD. It is the user who has to decide whether an LCD has to be initialized by instructions or by internal reset circuit.

5.2.4.1 Initialization by instructions

Initializing LCD with instructions is really simple. Given below is a flowchart that describes the step to follow, to initialize the LCD.

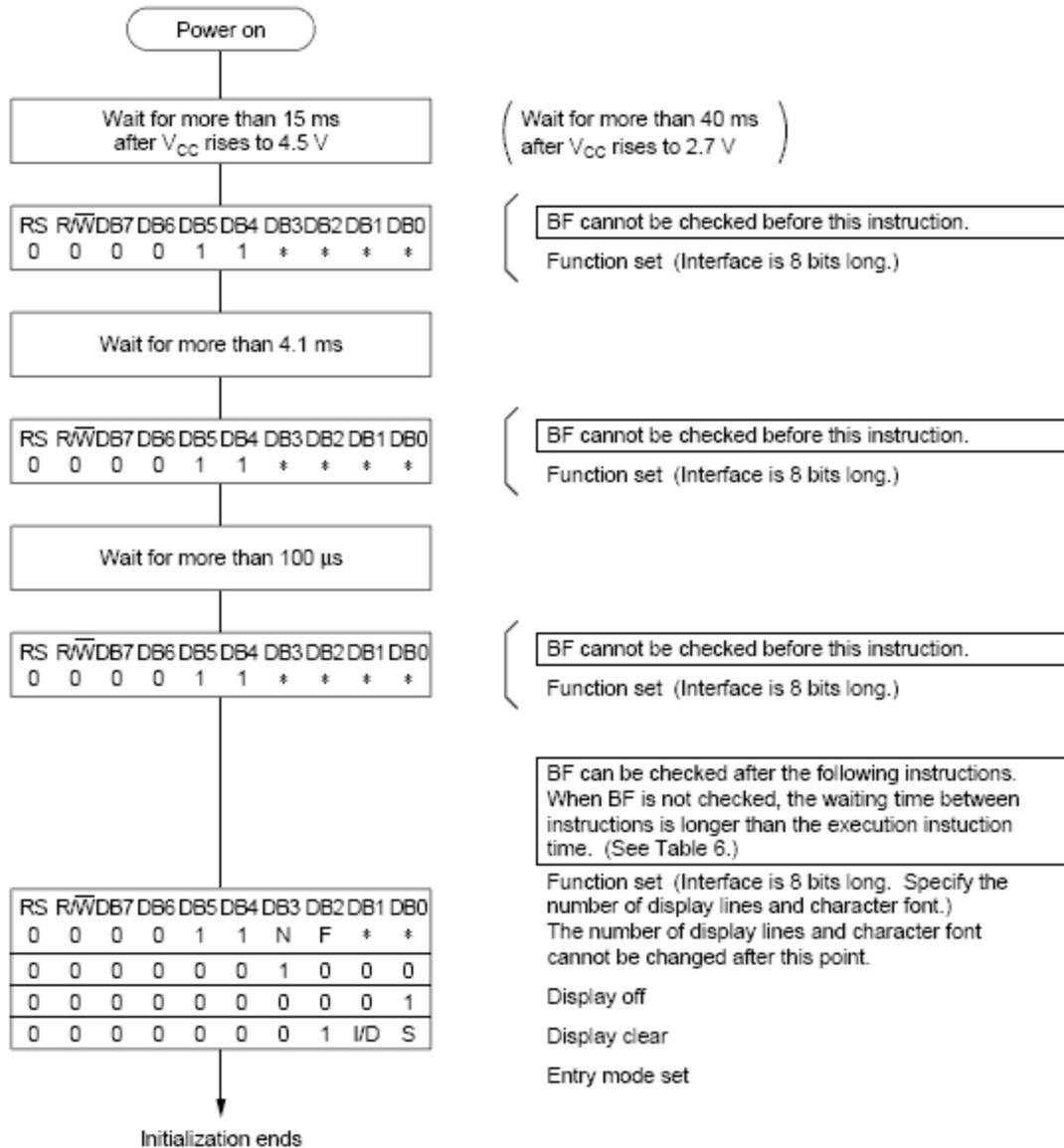


Figure 8: Flow chart for LCD initialization

As you can see from the flow chart, the LCD is initialized in the following sequence...

- 1) Send command 0x30 - Using 8-bit interface
- 2) Delay 20ms
- 3) Send command 0x30 - 8-bit interface
- 4) Delay 20ms
- 5) Send command 0x30 - 8-bit interface
- 6) Delay 20ms
- 7) Send Function set - see Table 4 for more information
- 8) Display Clear command
- 9) Set entry mode command - explained below

The first 3 commands are usually not required but are recommended when you are using

4-bit interface. So you can program the LCD starting from step 7 when working with 8-bit interface.

5.2.5 LCD Entry mode

Two bits decide the entry mode for LCD, these bits are:

- a) I/D - Increment/Decrement bit
- b) S - Display shift.

With these two bits we get four combinations of entry mode which are 0x04, 0x05, 0x06, 0x07. So we get different results with these different entry modes. Normally entry mode 0x06 is used which is No shift and auto increment.

5.2.6 Sending Commands to LCD

To send commands simply select the command register. Everything is same as in the initialization routine. Following are the steps that are to be kept in a single subroutine:

- Move data to LCD port
- Select command register
- Select write operation
- Send enable signal
- Wait for LCD to process the command.

5.2.7 Setting cursor position on LCD

To set the cursor position on LCD, we need to send the DDRAM address...

| | | | | | | | | |
|------|---|-----|-----|-----|-----|-----|-----|-----|
| BIT | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CODE | 1 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |

The seventh bit is always 1, and bit 0 to 7 are DDRAM address. To put the cursor on first position the address will be '0000000B' in binary and 7th bit is 1. So address will be 0x80, so for DDRAM all address starts from 0x80.

For 2 line and 16 character LCD: The address from 0x80 to 0x8F are visible on first line and 0xC0 to 0xCF are visible on second line, rest of the DDRAM area is still available but is not visible on the LCD.

5.2.8 Sending Data to LCD

To send data simply select the data register. Everything is same as the command routine. Following are the steps:

- ❖ Move data to LCD port
- ❖ Select data register
- ❖ Select write operation
- ❖ Send enable signal
- ❖ Wait for LCD to process the data.

5.2.9 LCD interfacing with Microcontrollers:4-bit Mode (The basic reason to use LCD in 4-bit mode instead of 8-bit mode is lesser number of pins are needed to interface LCD)

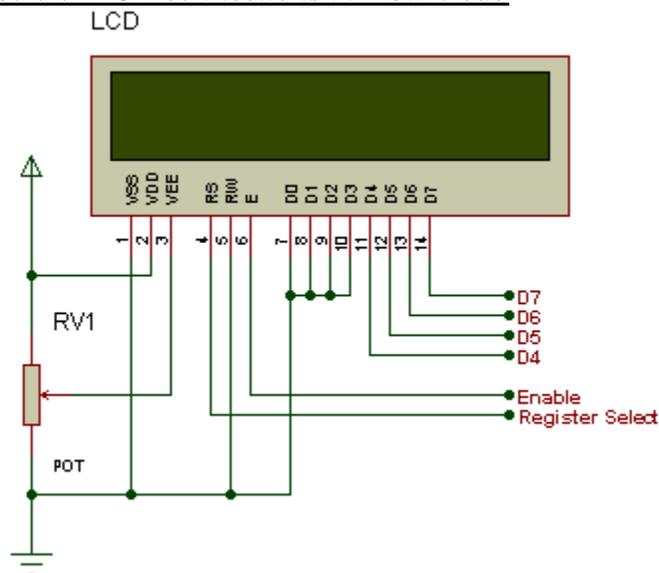
5.2.9.1 Introduction

In 4-bit mode the data is sent in nibbles, first we send the higher nibble and then the lower nibble. To enable the 4-bit mode of LCD, we need to follow special sequence of initialization that tells the LCD controller that user has selected 4-bit mode of operation. We call this special sequence as resetting the LCD. Following is the reset sequence of LCD.

- ❖ Wait for about 20mS
- ❖ Send the first init value (0x30)
- ❖ Wait for about 10mS
- ❖ Send second init value (0x30)
- ❖ Wait for about 1mS
- ❖ Send third init value (0x30)
- ❖ Wait for 1mS
- ❖ Select bus width (0x30 - for 8-bit and 0x20 for 4-bit)
- ❖ Wait for 1mS

The busy flag will only be valid after the above reset sequence. Usually we do not use busy flag in 4-bit mode as we have to write code for reading two nibbles from the LCD. Instead we simply put a certain amount of delay usually 300 to 600uS. This delay might vary depending on the LCD you are using, as you might have a different crystal frequency on which LCD controller is running. So it actually depends on the LCD module you are using

5.2.9.2 LCD connections in 4-bit Mode



Above is the connection diagram of

LCD in 4-bit mode, where we only need 6 pins to interface an LCD. D4-D7 are the data pins connection and Enable and Register select are for LCD control pins. We are not using Read/Write (RW) Pin of the LCD, as we are only writing on the LCD so we have made it grounded permanently. If you want to use it then you may connect it on your controller but that will only increase another pin and does not make any big difference. Potentiometer RV1 is used to control the LCD contrast. The unwanted data pins of LCD i.e.D0-D3 are connected to ground.

5.2.9.3 Sending data/command in 4-bit Mode

In 4-bit mode the higher 4-bits and lower 4-bits of both command and data sending function are to be separated.

The common steps are:

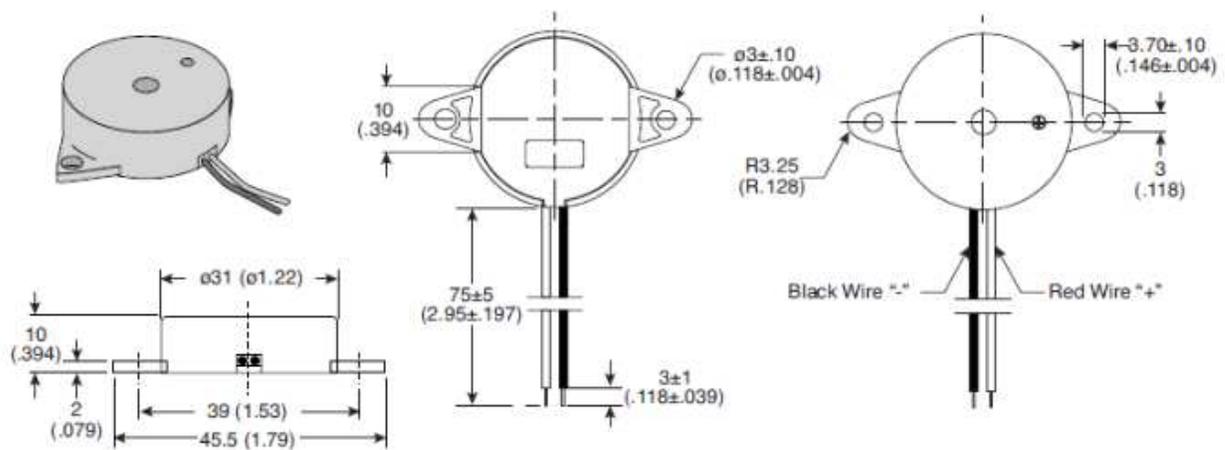
- Mask lower 4-bits
- Send to the LCD port
- Send enable signal
- Mask higher 4-bits
- Send to LCD port
- Send enable signal

5.3 BUZZER

5.3.1 Piezoelectric Buzzer

(Built-In Drive)
(In.)

Dimensions: mm

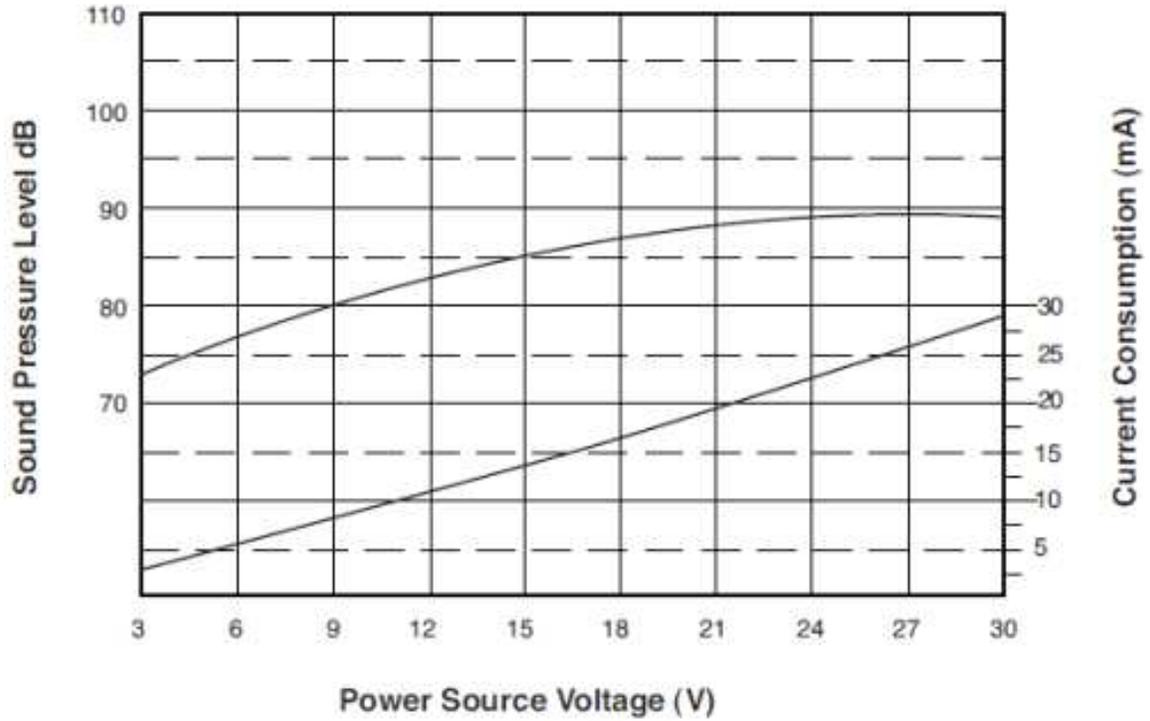


5.3.2 Electrical Specifications

- ❖ Sound Pressure Level: 80dB min./30cm./9VDC
- ❖ Oscillating Frequency: 2.5 ± 0.5 KHz
- ❖ Current Consumption: 8mA max./9VDC
- ❖ Operating Voltage: 3 to 30VDC

5.3.3 Mechanical Specifications

- ❖ Operating Temperature: -30°C to +70°C
- ❖ Storage Temperature: -40°C to +85°C



5.3.4 Materials

- ❖ Tone: Continuous
- ❖ Case: PBT
- ❖ Lead Wire: UL1095 28AWG
- ❖ Weight: 7 grams

6. GETTING STARTED WITH 'C' PROGRAMMING FOR THE "ATMEL AVR MICROCONTROLLER "

6.1 Installing tools for C programming

To work with the Atmel AVR microcontroller using the C programming language, two tools are required: *AVR Studio* and *Win AVR*.

- AVR Studio is an integrated development environment that includes an editor, the assembler, HEX file downloader and a microcontroller emulator.
- Win AVR is for a GCC-based compiler for AVR. It appears in AVR Studio as a plug-in. Win AVR also includes a program called Programmer's Notepad that can be used to edit and compile C programs, independently of AVR Studio. Installing these tools is easy: just download and run the setup files, and accept the default installation options. Remember to install AVR Studio first before Win AVR.

6.2 Using AVR Studio for C programming

After creating a simple C program for the Atmel AVR you will be guided through four major stages:

- creating an AVR Studio project,
- compiling C code to HEX file,
- debugging C program using the simulator,
- downloading HEX file to the STK500 development board and running it

6.3 Creating an AVR Studio project

Perform the following steps to create a simple AVR Studio project.

- Start the AVR Studio program by selecting **Start | Programs | Atmel AVR Tools | AVR Studio**.
- Select menu **Project | New Project**. In the dialog box that appears (see Figure 1), select AVR GCC as project type, and specify the project name and project location. If options 'Create initial file' and 'Create folder' are selected, an empty C file and containing folder will be created for you. In this case, we create a project called 'led'. Click button **Next** when you are ready.

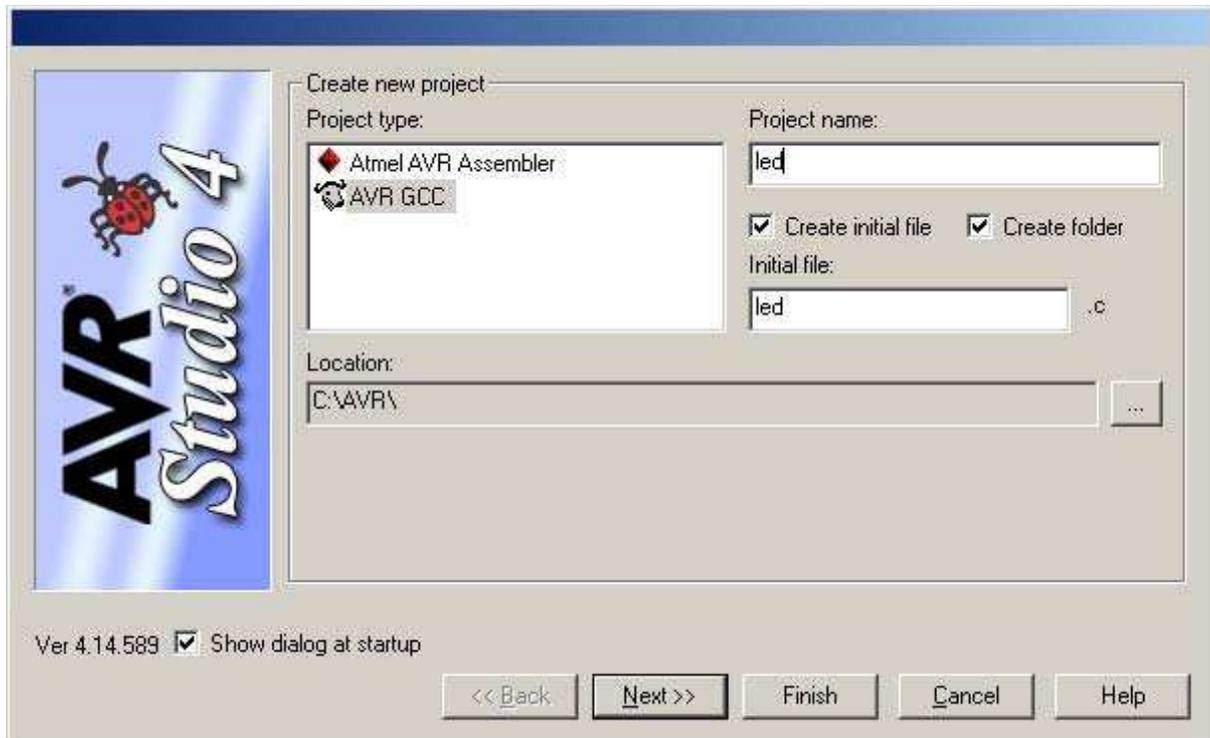


Figure 1: Entering project type, name and location.

- In the ‘Select debug platform and device’ dialog that appears (see Figure 2), choose ‘AVR Simulator’ as the debug platform and ‘ATMEGA16’ as the device. Click button **Finish**.

Note: If you want to use other AVR chips such as ATMAGE8515, select it at this step. In this tutorial, we will use ATMEGA16 for both software simulation and hardware testing.

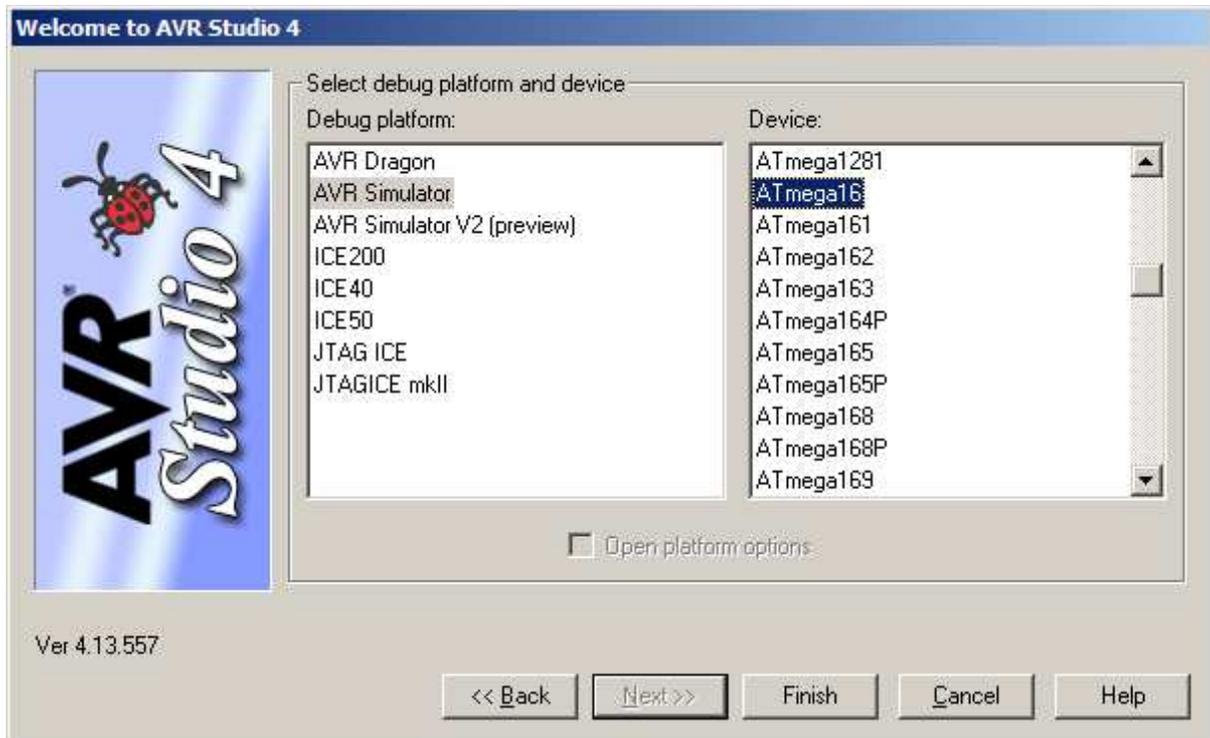


Figure 2: Selecting debug platform and device.

- A project file will be created and AVR Studio displays an empty file led.c (see Figure 3).

Enter the C code shown in Figure 4. It is not important to understand the code at this stage, but you can do that by reading the C comments.

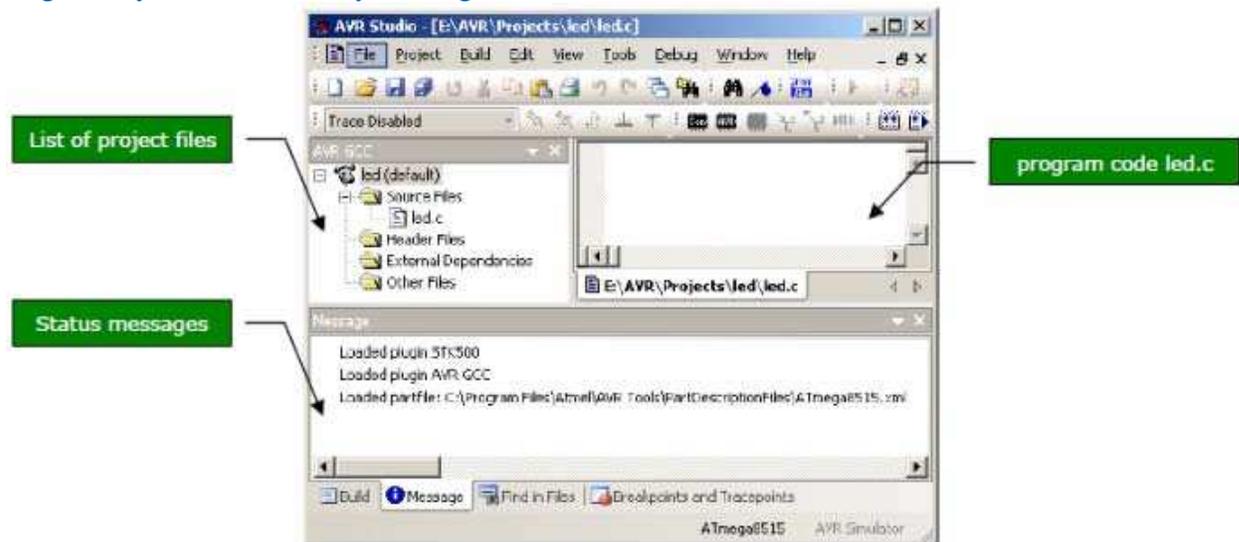


Figure 3: The AVR Studio with a project file open.

- Click menu **Project | Save Project** to save the project file and the C program. AVR Studio project files have extension 'aps'.

6.4 Compiling C code to HEX file

- Click menu **Build | Rebuild All** to compile the C code.
- If there is no error message, a file called `led.hex` will be produced (see Figure 5). This file contains the machine code that is ready to be downloaded to the ATMEGA16 microcontroller. The file is stored in sub-folder ‘default’ of your project.
- If there are error messages, check your C code. Most often, they are caused by some typos or syntax errors.

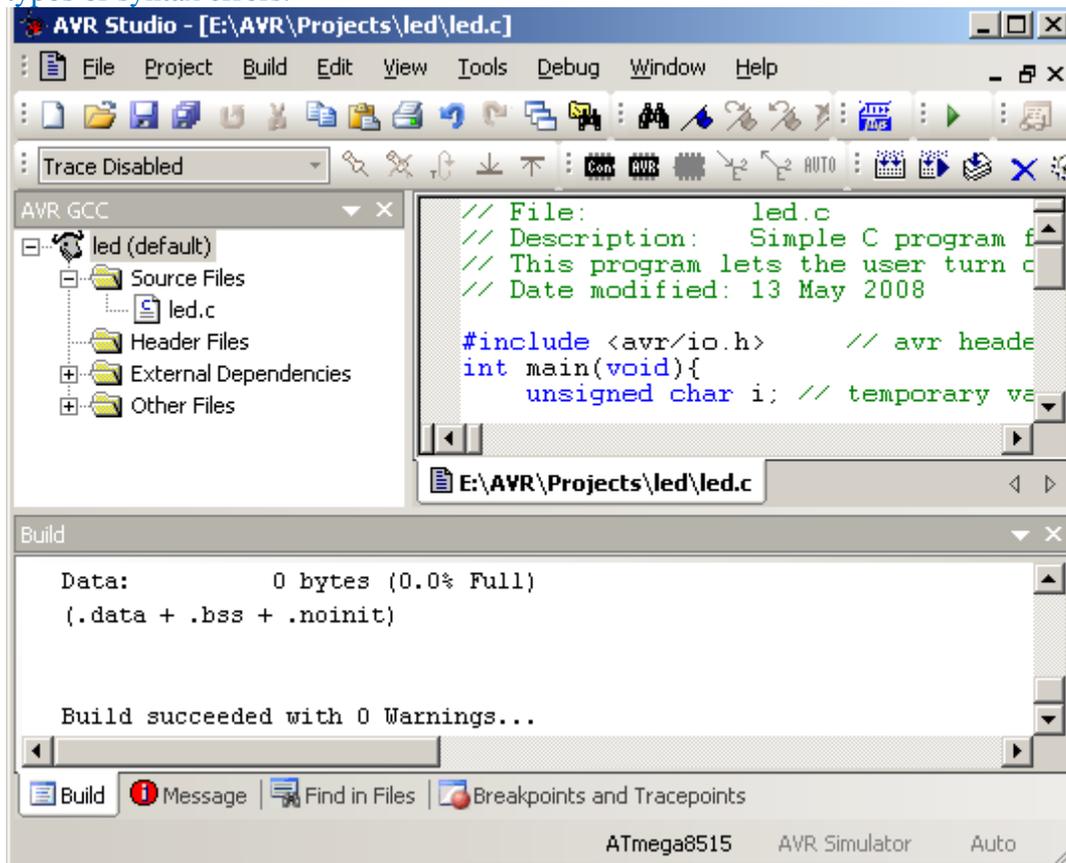


Figure 5: Selecting menu Build | Rebuild All to create HEX file.

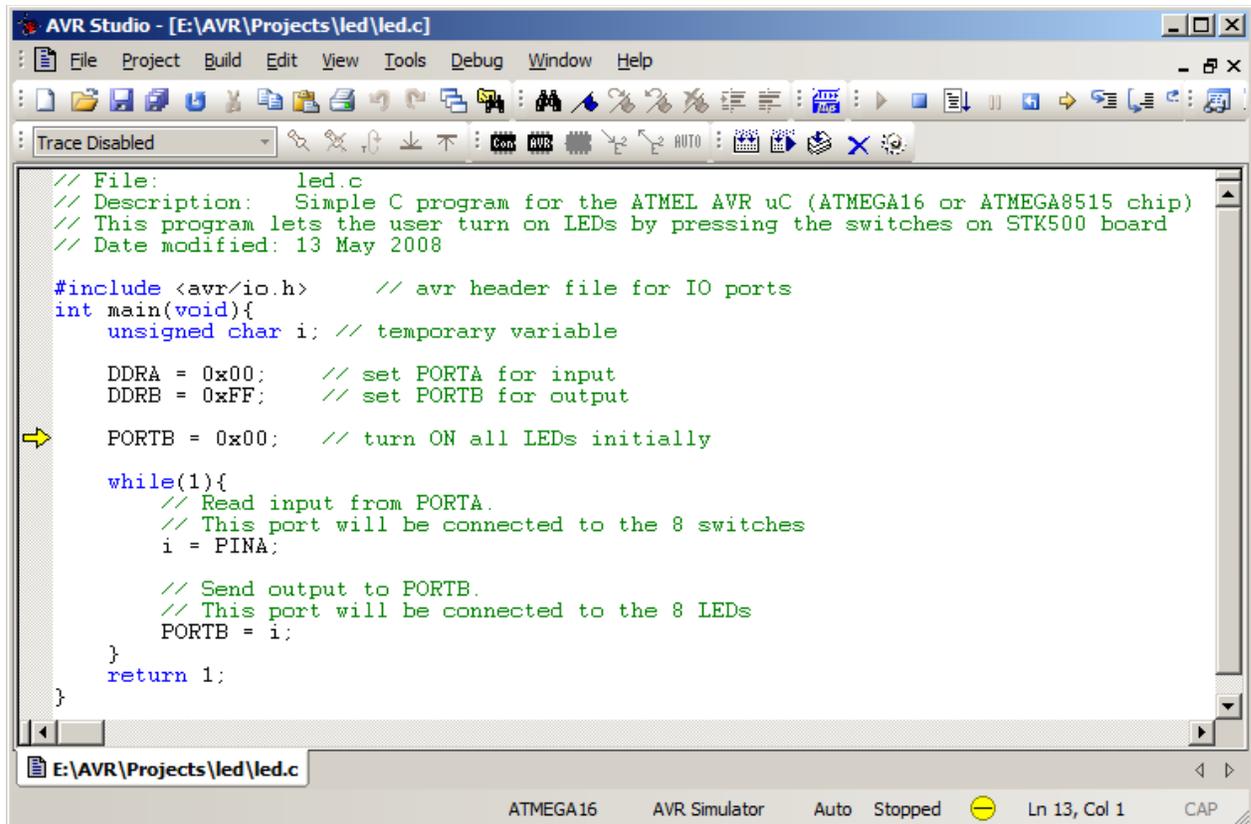
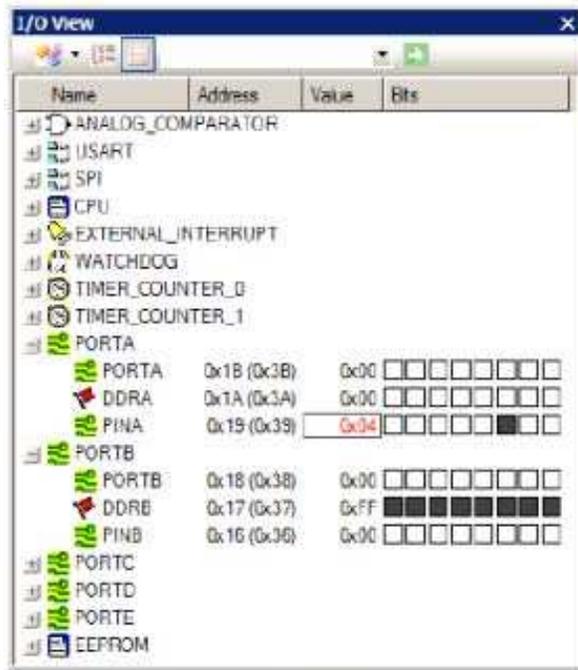


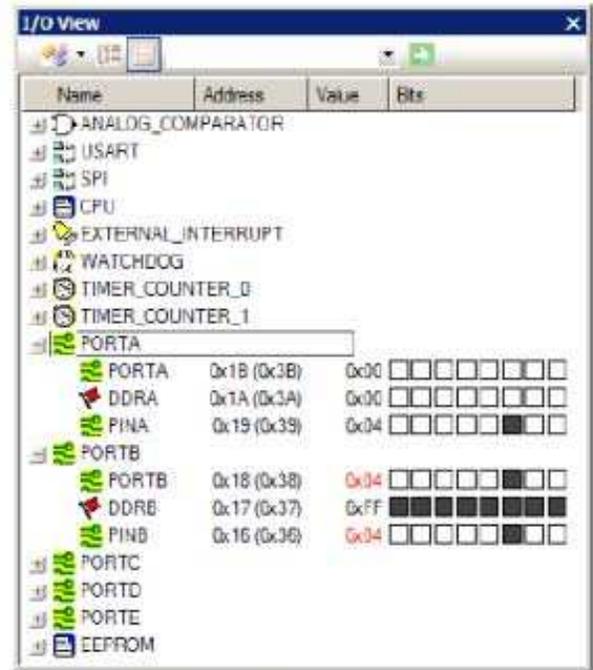
Figure 6: Stepping through a C program in debugging mode.

- While debugging the C program, you can change the contents of a register. For example, to change Port A Input Pins register (PINA), click on the value column of PINA and enter a new value (Figure 7a). This change takes effect immediately. Subsequently, the contents of PORTB will be 0x04 (see Figure 7b) after running the two C instructions:
`i = PINA;`

`PORTB = i;`



(a) changing PINA register to 0x04



(b) effects on PORTB after running `i = PINA; PORTB = i;`

Figure 7: Modifying registers manually.

- To monitor a C variable, select the variable name in the code window and click menu **Debug | Quick Watch**. The variable will be added to a watch window, as in Figure 8.

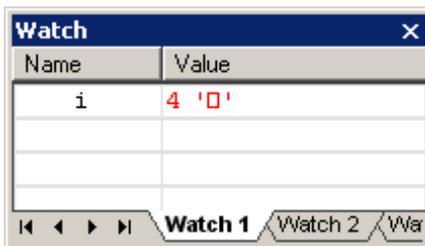


Figure 8: Watch window for C variables.

7. PROGRAMMING

7.1 Program at the transmitter Microcontroller

```
#include<avr/io.h>

#include<avr/interrupt.h>

void uart_init(void);

void delay(unsigned int);

volatile unsigned char start,address,data,chksum;

int main(void)

{

    DDRB=0XFF; //output for LED

    PORTB=0X11; //initial testing condition for LED

    DDRD=0X00; //input from sensors

    PORTD=0XFF; //enabling the pull up transistors

    uart_init();

    GICR=0XC0;          //to enable int1 and int0

    MCUCR=0X0F;        //to enable both the interrupts at the raising edge of the clk
pulse

    sei();              //set enable interrupt

    while(1)

    {

        start= 0XAA;

        address= 0X11;

        delay(10);

        if((PIND&0X20)==0X20) //to monitor the PIR sensor using the polling
```

concept

```
{
    PORTB=0X66;

    data='P';
}

if((PIND&0X10)==00) //to monitor the IR sensor using the polling concept
{
    PORTB=0XAA;

    data= 'T';
}

checksum= address+data;

while((UCSRA&0X20)==0); //transmit complete until the UDR is empty

UDR= start;

while((UCSRA&0X20)==0);

UDR= address;

while((UCSRA&0X20)==0);

UDR= data;

while((UCSRA&0X20)==0);

UDR= checksum;
}
}

void uart_init(void)
{
```

```
    UBRRL=0X3F; //1200 baud rate and 8MHz crystal frequency

    UBRRH=0X02;

    UCSRB=0X08; //transmit enable

    UCSRC=0X86; //8 bit length

}

ISR( INT0_vect)

{

cli();

PORTB=0X55;

data= 'D';

sei();

}

ISR( INT1_vect)

{

cli();

PORTB=0XAA;

data='W';

sei();

}

void delay(unsigned int t)

{

    int i,j;

    for(i=0;i<t;i++)
```

```
    for(j=0;j<500;j++);  
}
```

7.2 Program at the receiver Microcontroller

```
// LCD CONTROL PORTC  
  
// BUZZER - PORTD 4TH PIN  
  
// LCD DATA LINES PORTC  
  
#include<avr/io.h>  
  
#include<avr/pgmspace.h>  
  
#include<util/delay.h>  
  
#define sbi(PORT,BIT)PORT|=(1<<BIT)  
#define cbi(PORT,BIT)PORT&= ~(1<<BIT)  
  
void delay(unsigned int);  
  
void lcdcmd(unsigned char);  
  
void lcddata(unsigned char);  
  
void lcdinit();  
  
int main(void)  
{  
  
volatile unsigned char start,address,data,chksum,test; // global variables  
  
unsigned char s,*msg1,*msg2,*msg3,*msg4; // declaring four messages  
  
msg1 = PSTR("found movement");  
  
msg2 = PSTR("found obstacle");  
  
msg3 = PSTR("door broken");  
  
msg4 = PSTR(" window broken");
```

```

UBRRL=160; //baud rate OF 1200 @ 8MHZ

UBRRH=1;

UCSRB=0X10; // receive enable

UCSRC=0X86;

DDRD = 0XFF; // OUTPUT FOR BUZZER

PORTD = 0X00; // CLEARING PORTD ,4 LINE FOR BUZZER PIN

DDRC = 0XFF; //for lcd commands

DDRC= 0XFF; // for lcd data

lcdinit();

//while(!(UCSRA&0X20)); // checking start bit

// UDR='x';

while(1)

{

    UDR=0;

while(!(UCSRA&0X80)); // checking start bit

start=UDR;

//lcddata(start);

if(start==0XAA)

//if(start=='A')

{

    UDR=0;

while(!(UCSRA&0X80)); // receive address

address=UDR;

```

```

UDR=0;

while(!(UCSRA&0X80)); // receive data

data=UDR;

UDR=0;

while(!(UCSRA&0X80)); // receive chksum

chksum=UDR;

if(address==0X11)

    // if(address=='B')

    {

        //if(chksum==address+data) // checking chksum

        // if(chksum== 'Z')

        {

            if(data== 'P')

            {

                sbi(PORTD,4);// buzzer on

                lcdinit();

                for(s=0;pgm_read_byte(&msg1[s])!='\0';s++)

                {

                    lcddata(pgm_read_byte(&msg1[s])); // display message1 for PIR

                    delay(10);

                }

            }

        }

    }

} //end of P

else

```

```
    cbi(PORTD,4);// buzzer off

if(data== 'I')

{

    sbi(PORTD,4);// buzzer on

    lcdinit();

    for(s=0;pgm_read_byte(&msg2[s])!='\0';s++)

    {

        lcddata(pgm_read_byte(&msg2[s]));// display message2 for IR

        delay(10);

    }

} // end of I

else

    cbi(PORTD,4);// buzzer off

if(data== 'D')

{

    sbi(PORTD,4); // buzzer on

    lcdinit();

    for(s=0;pgm_read_byte(&msg3[s])!='\0';s++)

    {

        lcddata(pgm_read_byte(&msg3[s]));// display message3 for door

        delay(10);

    }

} // end of D
```

```

else

cbi(PORTD,4);// buzzer off

if(data== 'W')

{

sbi(PORTD,4); // buzzer on

lcdinit();

for(s=0;pgm_read_byte(&msg4[s])!='\0';s++)

{

lcddata(pgm_read_byte(&msg4[s])); // display mesage4 for window

delay(10);

}

} // end of W

else

cbi(PORTD,4);// buzzer off

} // end of chksum

} // end of address

} // end of start

} // end of while

} // end of main

void lcdcmd(unsigned char t)

{

unsigned char temp;

PORTC= t&0XF0;

```

```
cbi(PORTC,2);  
  
//cbi(PORTE,1);  
  
sbi(PORTC,3);// RS=0;RW=0;EN=1;  
  
delay(10);  
  
cbi(PORTC,3);  
  
temp=t&0X0F;  
  
temp=(temp<<4);  
  
PORTC=temp;  
  
delay(10);  
  
cbi(PORTC,2);  
  
sbi(PORTC,3);  
  
delay(10);  
  
cbi(PORTC,3);// RS=0;RW=0;EN=0;  
  
_delay_ms(100);  
  
return;  
  
} // end of lcd cmd  
  
void lcddata(unsigned char t)  
  
{  
  
    unsigned char temp;  
  
    PORTC= t&0XF0;  
  
    sbi(PORTC,2);  
  
    //cbi(PORTE,1);  
  
    sbi(PORTC,3);// RS=0;RW=0;EN=1;
```

```

delay(10);

cbi(PORTC,3);

temp=t&0X0F;

temp=(temp<<4);

PORTC=temp;

delay(10);

sbi(PORTC,2);

sbi(PORTC,3);

delay(10);

cbi(PORTC,3);// RS=0;RW=0;EN=0;

_delay_ms(100);

return;

} // end of lcd data

void delay(unsigned int v)

{

    _delay_ms(20);

} // end of

void lcdinit()

{

    lcdcmd(0X03);//set byte interface

    lcdcmd(0X03);// cursor blink on the first position

    lcdcmd(0X03);// clear the display

    lcdcmd(0X02);// increment cursor

```

```
lcdcmd(0X28);// first line first position  
lcdcmd(0X08);//set byte interface  
lcdcmd(0X01);// cursor blink on the first position  
lcdcmd(0X0f);// clear the display  
}
```

8. CONCLUSION

In this project we implemented Microcontroller based Home security system using Wireless alerts using RF signals as communication standards. The hardware equipment is tested and result is obtained.

This project is cost effective. Implementation of this project in present day will effectively provide a peace of mind to the residents. This project can be implemented both in apartments and General stores as well.

With a slight modification this project can be implemented in a near by area police station, by connecting to several homes, locating where the robbery has occurred or by directly sending an SMS to them.

