Visvesvaraya Technological University
Belgaum, Karnataka-590 018

A Project Report on

"AN INTELLIGENT SUVEILLANCE SYSTEM BASED ON MOTION DETECTION"

Project Report submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering
In
Electrical & Electronics Engineering

Submitted by

Santosh Kumar N A(1CR16EE415)
Devanshu kumar(1CR15EE026)
ABHISHEK KUMAR RAY(1CR14EE001)
ARMAN KUMAR(USN- 1CR15EE013)

Under the Guidance of

Ms. Lokasree B S

Assistant professor, Department of Electrical & Electronics Engineering
CMR Institute of Technology

CMR Institute of Technology, Bengaluru-560 037

Department of Electrical & Electronics Engineering

CMR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

AECS Layout, Bengaluru-560 037

Certificate

Certified that the project work entitled **"AN INTELLIGENT SUVEILLANCE SYSTEM BASED ON MOTIon DETECTION"** carried out by **Mr Arman kumar (1CR15EE013)** and **Mr DEVANSHU Kumar(1CR15EE026) Mr ABHISHEK KUMAR RAY(1cr14ee001) SANTHOSH KUMAR N A(1cr16ee415)** are bonafied students of CMR Institute of Technology, Bengaluru, in partial fulfillment for the award of Bachelor of Engineering in Electrical & Electronics Engineering of the Visvesvaraya Technological University, Belgaum, during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in escribe

Signature of the Guide	*Signature of the HOD*	*Signature of the Principal*
Ms.lokasree B S, Assistant Professor EEE Department CMRIT, Bengaluru	Dr. K. Chitra Professor & HOD EEE Department CMRIT, Bengaluru	Dr. Sanjay Jain Principal, CMRIT, Bengaluru

External Viva

Name of the Examiners	Signature & Date
1.	
2.	

CMR INSTITUTE OF TECHNOLOY

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

AECS Layout, Bengaluru-560 037

DECLARATION

We, [Mr. Arman kumar (1CR15EE013) and Mr. DEVANSHU Kumar(1CR15EE026, Mr ABHISHEK KUMAR RAY(1cr14ee001) SANTHOSH KUMAR N A(1cr16ee415) hereby declare that the report entitled **"AN INTELLIGENT SUVEILLANCE SYSTEM BASED ON MOTION DETECTION"** has been carried out by us under the guidance of **Ms. Lokasree B S,** Assistant Professor, Department of Electrical & Electronics Engineering, CMR Institute of Technology, Bengaluru, in partial fulfillment of the requirement for the degree of **BACHELOR OF ENGINEERING in ELECTRICAL & ELECTRONICS ENGINEERING**, of Visveswaraya Technological University, Belagaum during the academic year 2019-20. The work done in this report is original and it has not been submitted for any other degree in any university.

Place: Bengaluru

Date:

Abstract

Surveillance is a common technology nowadays. By general it means a mechanism to monitor the behavior, activities, or another changing information. Most technology used by implementing a CCTV device in watched area.

A CCTV consist of video devices, PC to monitor the video real time, and human as person who monitor the area. The weakness occurs from that kind of mechanism, it needs a human to watch every time. And the cost is very expensive for building it.

Thus, an infrared movement detection sensor called PIR (passive infrared) Sensor can be utilized.

Passive infrared (PIR) sensors are very popular as they are mass-produced at very low cost. Most commonly they have been used for in-building and homes applications to detect presence and movement in order to directly switch lights and appliance. Passive infrared (PIR) sensors are very popular as they are mass-produced at very low cost. Most commonly they have been used for in-building and homes applications to detect presence and movement in order to directly switch lights and appliances. In contrast to other sensors such as ultrasound rangers or infrared light distance sensors, PIRs have very low power consumption.

.

-

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people, who are responsible for the completion of the project and who made it possible, because success is outcome of hard work and perseverance, but stead fast of all is encouraging guidance. So with gratitude we acknowledge all those whose guidance and encouragement served us to motivate towards the success of the project work.

We take great pleasure in expressing our sincere thanks to ***Dr. Sanjay Jain****,* ***Principal, CMR Institute of Technology, Bengaluru*** *for providing an excellent academic environment in the college and for his continuous motivation towards a dynamic career. We would like to profoundly thank* ***Dr. B Narasimha Murthy****, Vice-principal of CMR Institute of Technology and the whole* ***Management*** *for providing such a healthy environment for the successful completion of the project work.*

We would like to convey our sincere gratitude to ***Dr. K Chitra****,* ***Head of Electrical and Electronics Engineering Department, CMR Institute of Technology, Bengaluru*** *for her invaluable guidance and encouragement and for providing good facilities to carry out this project work.*

We would like to express our deep sense of gratitude to ***Lokasree B S, Assistant Professor, Electrical and Electronics Engineering, CMR Institute of Technology, Bengaluru*** *for his/her exemplary guidance, valuable suggestions, expert advice and encouragement to pursue this project work.*

We are thankful to all the faculties and laboratory staffs of ***Electrical and Electronics Engineering Department, CMR Institute of Technology, Bengaluru*** *for helping us in all possible manners during the entire period.*

Finally, we acknowledge the people who mean a lot to us, our parents, for their inspiration, unconditional love, support, and faith for carrying out this work to the finishing line. We want to give special thanks to all our friends who went through hard times together, cheered us on, helped us a lot, and celebrated each accomplishment.

Lastly, to the ***Almighty****, for showering His Blessings and to many more, whom we*

didn't mention here.

v

CONTENTS

vi

LIST OF ABBREVIATIONS AND SYMBOLS

Internet of Things

The Internet of Things is a collection of interrelated computing devices, mec hanical and digital machines, objects, animals or individuals that have uniqu e identifiers and the ability to transfer data over a network without needing h uman-to-human or computer-to-computer interaction.

PIR

A **passive Infrared sensor** (**PIR sensor**) is an electronic that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors

MC AVR

An embedded system is a system which is going to do a predefined specified task is the embedded system and is even defined as combination of both software and hardware. A general-purpose definition of embedded systems is that they are devices used to control, monitor or assist the operation of equipment, machinery or plant. "Embedded" reflects the fact that they are an integral part of the system. At the other extreme a general-purpose computer may be used to control the operation of a large complex processing plant, and its presence will be obvious .All embedded systems are including computers or microprocessors. Some of these computers are however very simple systems as compared with a personal computer .The very simplest embedded systems are capable of performing only a single function or set of functions to meet a single predetermined purpose.

CHAPTER 1

INTRODUCTION

Surveillance is a common technology nowadays. By general it means a mechanism to monitor the behavior, activities, or another changing information. Most technology used by implementing a CCTV device in watched area. A CCTV consist of video devices, PC to monitor the video real time, and human as person who monitor the area. The weakness occurs from that kind of mechanism, it needs a human to watch every time. And the cost is very expensive for building it. Thus, an infrared movement detection sensor called PIR (passive infrared) Sensor can be utilized.

Passive infrared (PIR) sensors are very popular as they are mass-produced at very low cost. Most commonly they have been used for in-building and homes applications to detect presence and movement in order to directly switch lights and appliances. In contrast to other sensors such as ultrasound rangers or infrared light distance sensors, PIRs have very low power consumption.

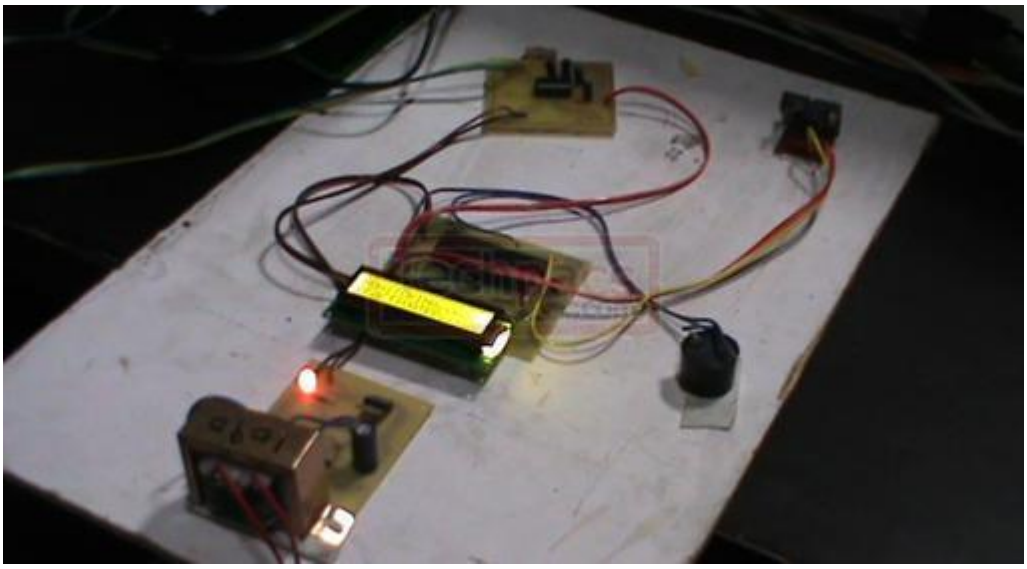


Fig 1.1 **AN INTELLIGENT SUVEILLANCE SYSTEM BASED ON MOTION DETECTION**

CHAPTER 2

LITERATURE SURVEY

Bashar Alathari et al. [1] considered the problem of automated position estimation using the electronic circuit of inexpensive binary motion sensors. They present simulation and experiments with Passive Infrared (PIR) motion sensors that suggest our current estimator. Fritzing software simulator is used to test and draw the circuits of the system. The proposed design worked efficiently during the experiments and shown high performance with 360 degrees of det

Sanjana Prasad et al. [2] The proposed home security system captures information and transmits it via a 3G Dongle to a Smart phone using web application. Raspberry pi operates and controls motion detectors and video cameras for remote sensing and surveillance, streams live video and records it for future playback. It can also find the number of persons located with the help of the Infrared sensor..

R Ismail et al. [3] used PIR and IR and designed a robot.The hardware was integrated in one application board as embedded system design. The software was developed using C++ and compiled by Arduino IDE 1.6.5. The main objective of this project is to provide simple guidelines to the polytechnic students and beginners who are interested in this type of research.

Cheng huang tesai et al. [4] present a way to reduce the standby power consumption of a PIR-sensor-based lighting device. Generally, although a PIR-sensor-based lighting device will turn on when motion is detected and turn off when the motion is no longer present, this device still consumes 1-3 W of power when the lamp is off. In this design the device consumes 0.004 W when the light is turned off, and it is not only easy to set up but also inexpensive.

K. N. Ha et al. [5] presents a novel non-terminal-based approach using an array of pyroelectric infrared sensors (PIR sensors) that can detect residents. The feasibility of the system is evaluated experimentally on a test bed

Automatic system

Four motors are used for the purpose like movement of robot, water pump.

Relays are used to drive water pumps and motor cleaners. LM293D IC is used for motor wheel drive. All the details is shown on LCD. Such criteri a are compatible with some of the behaviors expected to be programmed i nto the robot. This consists of four dedicated wipers fixed to the board. One of the wipers is cylindrical, and the others are geometrically square. The flat wipers are mounted symmetrically at the bottom of the Platform,

positioned in' V' shape to ensure efficient cleaning and dust collection. The roller wipers are mounted with proper ties and driver motor at the end of the platform. Using wet wiping method, cleaning is rendered effective. This device uses a small bottle which carries water within it. That ensures a full surface cleaning. Only the front wipers are rendered watery. It means that the rear wiper removes the water from the surface.

The robot works in both autonomous and manual mode, along with additio nal features such as precise time scheduling and bagless dirt containers wit h auto-
dirt disposal function. This work can be very useful in the development of human life style.

The proposed design is in dual mode operation. In one of the modes, the r obot is fully autonomous and makes decisions based on the outputs of infr arot proximity sensors, ultrasonic sensors and tactile sensors after Arduino (mega) controller is processed and the actuators are controlled (2 DC encoder motors) by the H-bridge driving circuitry.

The robot can also be used to clean a particular area of a room by manuall y monitoring it from a laptop using the Visual Studio Graphical User Inte rface (GUI). (C# programming language) via Bluetooth connectivity.

The following figure shows the working model of CLEAR:

The base of the body comprises of acrylic sheet, two encoder motors along with Teflon tires having O-rings on them for avoiding friction, two ball casters of adjustable height having frictionless steel balls, aluminium angular brackets and aluminium holders for two lead acid batteries of 12V and 1.2Ah rating.

A DC geared motor, sprockets for moving chain from geared motor to spi nning brush and two aluminum rods for supporting vacuum cleaner mech anism and dirt compartment are included in the cleaning system.

Components are installed at the bottom of the acrylic sheet, so that the ce nter of gravity is lower and the robot is stable

In some of the vacuum cleaners, AT89S52 Micro-Controller manages both hardware and software operations .

RF modules were used for remote (manual) and robot wireless communica tion with a range of 50 m. This robot is equipped for obstacle detection an d automated water sprayer pump with IR sensor. It uses four motors, two f or cleaning, one for water pumping and one for wheels. Dual relay circuit used to drive the motors one for water pump and another for cleaner. All the operations themselves are managed in the automatic mode robot and the lane changes in case of hurdle detection and moves back. The keypad is used in manual mode to accomplish the intended task and to run robot. The RF module was used to transmit and receive information between remote and robot, and to display information on the LCD relevant to hurdle detection.

Unlike other floor cleaner robots this is not a vacuum cleaner robot; it performs sweeping and mopping operation. Detachable mop is used for mopping. Robot performs all of the operations itself in automatic mode. First, the robot begins, moves forward and carries out cleaning operation. These have been used to track obstacles and to clear hurdle IR sensors.

If any obstacle is detected then the robot will automatically change the lan e,will not stop and will start cleaning. This assumes zig-zag trajectory.

For the convenience of the user,automatic water sprayer is attached whic h automatically sprays water for mopping, so no need to reattach wet clot

h for mop-
ing.Four motors have been used to perform valued operations such as mo ving the device, water pumping and cleaning.

Relays are used to operate water pumps and engine cleaners. LM293D IC is used for motor wheel driving.

All the details is displayed on LCD. In manual mode, the robot is controll ed by the user itself. RF module have been used to transmit and receive the signal

In the manual mode, if any hurdle is detected, then signal of hurdle detection is displayed on the LCD of remote via RF module**.**

All the information displayed on LCD. In the manual mode, user itself operates the robot. RF module have been used to transmit and receive the signal to operate the robot through remote. In the manual mode, if any hurdle detected, then signal of hurdle detection displayed on the LCD of remote via RF module.
pathway. Direction does it take depends on where the bumper makes contact. For example, if a vacuum enters an object with its bumper's left side, it will generally turn right because the object has been determined to be to its left. And maneuvering around items can often leave uncleaned swaths of concrete.
Some manufacturers take different approaches to obstacles, literally, to minimize this. For example, an iRobot Roomba can slow as its encounters an obstacle. "The beauty of Roomba is that we handle things softly,because what we find is that you can move through soft items such as curtains and bed skirts," said Ken Bazydola, product management director for iRobot.This gives us better coverage .

dict whether our vacuum cleaner has completely cleaned our room or not”

.

Whereas in our proposed work we can conform ourselves that our floor has been cleaned by vacuum cleaner completely because in our project we will be using range of interest which will help our vacuum cleaner in determining the obstacles and selecting proper path to move.

PIR SENSORS

A **passive Infrared sensor** (**PIR sensor**) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.

Operating Principle

All objects above absolute zero emit heat energy in the form of infrared radiation (infrared light). Usually infrared light is invisible to the human eye, but it can be detected by electronic devices designed for such a purpose.

The term *passive* in this instance refers to the fact that PIR devices do not generate or radiate any energy for detection purposes. They work entirely by detecting the energy given off by other objects.

Construction

Infrared radiation enters through the front of the sensor, known as the *sensor face*. At the core of a PIR sensor is a solid state *sensor* or set of sensors, made from pyroelectric materials -- materials which generate energy when exposed to heat. Typically, the sensors are approximately 1/4 inch square, and take the form of a thin film. Materials commonly used in PIR sensors include gallium nitride (GaN), caesium nitrate ($CsNO_3$), polyvinyl fluorides, derivatives of phenylpyrazine, and cobalt phthalocyanine. The sensor is often manufactured as part of an integrated circuit.

PIR based motion detector

A PIR-based motion detector is used to sense movement of people, animals, or other objects. They are commonly used in burglar alarms and automatically-activated lighting systems. They are commonly called simply "PIR", or sometimes "PID", for *passive infrared detector*.

Operation

Strictly speaking, individual PIR sensors do not detect motion; rather, they detect abrupt changes in temperature at a given point. As an object, such as a human, passes in front of the background, such as a wall, the temperature at that point will rise from room temperature to body temperature, and then back again. This quick change triggers the detection. Moving objects of identical temperature will not trigger detection.

PIDs can be equipped with more than one internal sensing element so that, with the appropriate electronics, it can detect the apparent direction of movement. As an object passes in front of adjacent sensors in turn, this implies the direction of movement. This may be used by on-board electronics to reduce false alarms, i.e., by requiring adjacent sensors to trip in succession. It may also be used to signal the direction of movement to a monitoring apparatus.

PIDs come in many configurations for a wide variety of applications. The most common models have numerous Fresnel lenses or mirror segments, an effective range of about thirty feet, and a field of view less than 180 degrees. Models with wider fields of view, including 360 degrees, are available -- typically designed to mount on a ceiling. Some larger PIDs are made with single segment mirrors and can sense changes in infrared energy over one hundred feet away from the PID. There are also PIDs designed with reversible orientation mirrors which allow either broad coverage (110° wide) or very narrow "curtain" coverage, or with individually selectable segments to "shape" the coverage.

Differential detection

Pairs of sensor elements may be wired as opposite inputs to a differential amplifier. In such a configuration, the PIR measurements cancel each other so that

the average temperature of the field of view is removed from the electrical signal; an increase of IR energy across the entire sensor is self-cancelling and will not trigger the device. This allows the device to resist false indications of change in the event of being exposed to brief flashes of light or field-wide illumination. (Continuous high energy exposure may still be able to saturate the sensor materials and render the sensor unable to register further information.) At the same time, this differential arrangement minimizes common-mode interference, allowing the device to resist triggering due to nearby electric fields. However, a differential pair of sensors cannot measure temperature in this configuration, and therefore is only useful for motion detection.

Product design

The PIR sensor is typically mounted on a printed circuit board containing the necessary electronics required to interpret the signals from the sensor itself. The complete assembly is usually contained within a housing, mounted in a location where the sensor can cover area to be monitored.

The housing will usually have a plastic "window" through which the infrared energy can enter. Despite often being only translucent to visible light, infrared energy is able to reach the sensor through the window because the plastic used is transparent to infrared radiation. The plastic window reduces the chance of foreign objects (dust, insects, etc.) from obscuring the sensor's field of view, damaging the mechanism, and/or causing false alarms. The window may be used as a filter, to limit the wavelengths to 8-14 micrometres, which is closest to the infrared radiation emitted by humans. It may also serve as a focusing mechanism; see below.

Focusing

Different mechanisms can be used to focus the distant infrared energy onto the sensor surface.

Lenses

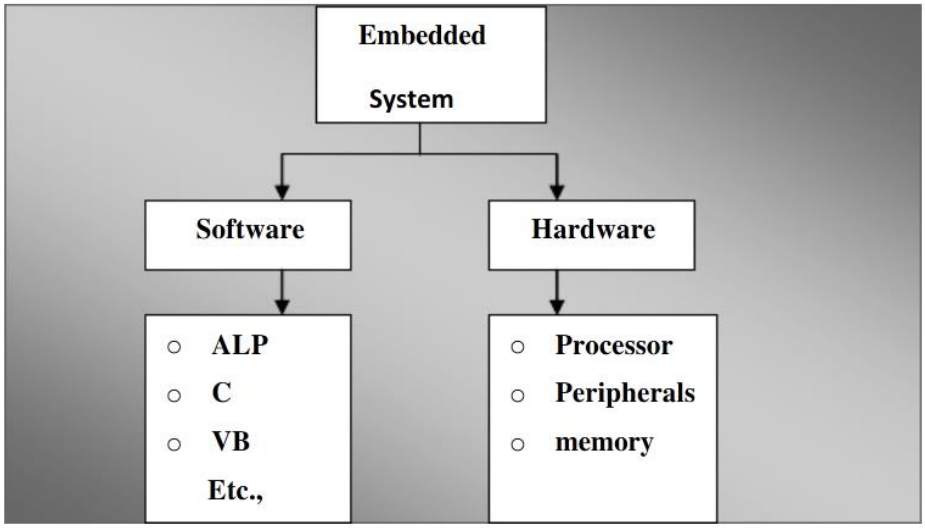
The plastic window covering may have multiple facets molded into it, to focus the infrared energy onto the sensor. Each individual facet is a Fresnel lens.

MC AVR

Embedded System

An embedded system is a system which is going to do a predefined specified task is the embedded system and is even defined as combination of both software and hardware. A general-purpose definition of embedded systems is that they are devices used to control, monitor or assist the operation of equipment, machinery or plant. "Embedded" reflects the fact that they are an integral part of the system. At the other extreme a general-purpose computer may be used to control the operation of a large complex processing plant, and its presence will be obvious .All embedded systems are including computers or microprocessors. Some of these computers are however very simple systems as compared with a personal computer .The very simplest embedded systems are capable of performing only a single function or set of functions to meet a single predetermined purpose. In more complex systems an application program that enables the embedded system to be used for a particular purpose in a specific application determines the functioning of the embedded system. The ability to have programs means that the same embedded system can be used for a variety of different purposes. In some cases a microprocessor may be designed in such a way that application software for a particular purpose can be added to the basic software in a second process, after which it is not possible to make further changes. The applications software on such processors is sometimes referred to as firmware .The simplest devices consist of a single microprocessor (often called a "chip‖), which may itself be packaged with other chips in a hybrid system or Application Specific Integrated Circuit (ASIC). Its input comes from a detector or sensor and its output goes to a switch or activator which (for example) may start or stop the operation of a machine or, by operating a valve, may control the flow of fuel to an engine.

As the embedded system is the combination of both software and hardware



Block diagram of Embedded System

Software deals with the languages like ALP, C, and VB etc., and Hardware deals with Processors, Peripherals, and Memory.

Memory: It is used to store data or address.

Peripherals: These are the external devices connected

Processor: It is an IC which is used to perform some task

Applications of embedded systems

- Manufacturing and process control
- Construction industry
- Transport
- Buildings and premises
- Domestic service
- Communications
- Office systems and mobile equipment
- Banking, finance and commercial
- Medical diagnostics, monitoring and life support
- Testing, monitoring and diagnostic systems

Processors are classified into four types like:

- Micro Processor (µp)
- Micro controller (µc)

- Digital Signal Processor (DSP)
- Application Specific Integrated Circuits (ASIC)

Micro Processor (µp):

A silicon chip that contains a CPU is called MPU. In the world of personal computer the terms microprocessor and CPU are used interchangeably. At the heart of all personal computers and most workstations sits a microprocessor. Microprocessors also control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles.

Three basic characteristics differentiate microprocessors:

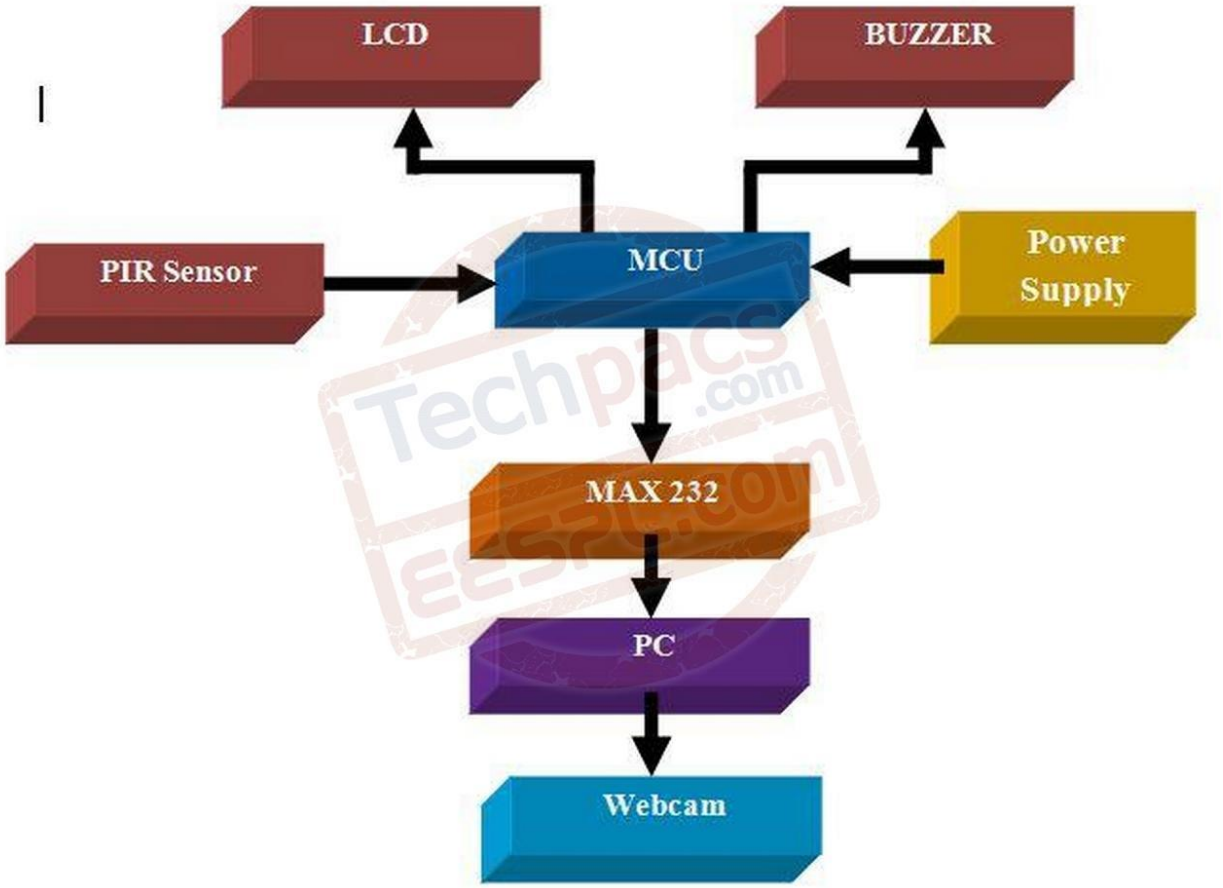
- Instruction set:-The set of instructions that the microprocessor can execute.
- Bandwidth :-The number of bits processed in a single instruction.
- Clock speed:-Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute. In both cases, the higher the value, the more powerful the CPU. For example, a 32-bit microprocessor that runs at 50MHz is more powerful than a 16-bit microprocessor that runs at 25MHz. In addition to bandwidth and clock speed, microprocessors are classified as being either RISC (reduced instruction set computer) or CISC (complex instruction set computer).

A microprocessor has three basic elements, as shown above. The ALU performs all arithmetic computations, such as addition, subtraction and logic operations (AND, OR, etc). It is controlled by the Control Unit and receives its data from the Register Array. The Register Array is a set of registers used for storing data. These registers can be accessed by the ALU very quickly. Some registers have specific functions - we will deal with these later. The Control Unit controls the entire process. It provides the timing and a control signal for getting data into and out of the registers and the ALU and it

CHAPTER 3

METHODOLOGY

BLOCK DIAGRAM:



- In the remote sensing Model, microcontroller is responsible for performing all the operations as it sends signals to the required devices.
- The microcontroller is connected to the power supply giving 220 V. As, microcontroller requires less voltage, this supply is converted into 5V by using step down transformer.
- Besides this, In order to convert the alternating current into Direct current the power supply is connected to the full wave rectifier. The

converted voltage is regulated by using the voltage regulator.

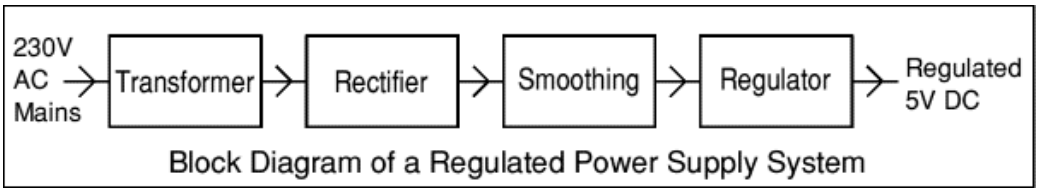
- Eventually, the microcontroller will receive regulated voltage and it is connected to LCD and Buzzer. LCD is used in this model to display the presence of any object and the buzzer will alert the user by producing alarming sound if anything is detected.
- The main role in sensing the persons is played by the PIR sensor which is connected to the microcontroller. The sensor works for the If within the range, any object in motion is detected by the sensor it will transmit the signal to microcontroller.
- After receiving the signal, microcontroller will pass the signal to the PC using serial communication. Simultaneously, it will send the signal to LCD and buzzer that will result in displaying the message of a object's detection on LCD and beeping sound by the buzzer.
- Afterwards, PC will retrieve the signal and transmits it to the MATLAB software wherein A GUI for this Model will be present. This GUI bears the responsibility of transmitting the signal to the webcam.
- As soon as the camera will get the signal, it will start recording the video for the set time period in the code. The captured video is stored in the data base and camera will get back to its original state i.e. it will wait for the next signal to capture any motion performed by the object. In this way the project will work and the person will be detected.

HARDWARE METHODOLOGY

Regulated Power Supply

Power supplies are designed to convert high voltage AC mains to a suitable low voltage supply for electronic circuits and other devices. A power supply can be broken down into a series of blocks, each of which performs a particular function.

For example a 5V regulated supply:



Block Diagram of a Regulated Power Supply System

Each of the blocks has its own function as described below

1. Transformer – steps down high voltage AC mains to low voltage AC.
2. Rectifier – converts AC to DC, but the DC output is varying.
3. Smoothing – smoothes the DC from varying greatly to a small ripple.
4. Regulator – eliminates ripple by setting DC output to a fixed voltage.

Transformer

Transformers convert AC electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity is AC. The two types of transformers

- Step-up transformers increase voltage,
- Step-down transformers reduce voltage.

Most power supplies use a step-down transformer to

reduce the dangerously high mains voltage (230V in UK) to a safer low voltage. The input coil is called the primary and the output coil is called the secondary. There is no electrical connection between the two coils, instead they are linked by an alternating magnetic field created in the soft-iron core of the transformer. The two lines in the middle of the circuit symbol represent the core.

Transformers waste very little power so the power out is (almost) equal to the power in. Note that as voltage is stepped down current is stepped up. The ratio of the number of turns on each coil, called the turn ratio, determines the ratio of the voltages. A step-down transformer has a large number of turns on its primary (input) coil which is connected to the high voltage mains supply, and a small number of turns on its secondary (output) coil to give a low output voltage.

Turns Ratio = $\frac{Vp}{Vs} = \frac{Np}{Ns}$

And Power Out = Power In

$Vs \times Is = Vp \times Ip$

Where

Vp = primary (input) voltage

Np = number of turns on primary coil

Ip = primary (input) current

Ns = number of turns on secondary coil

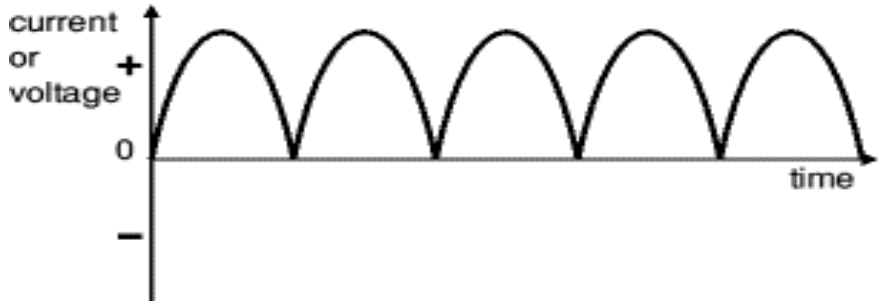
Is = secondary (output) current

Vs = secondary (output) voltage

Bridge Rectifier

A bridge rectifier can be made using four individual diodes, but it is also available

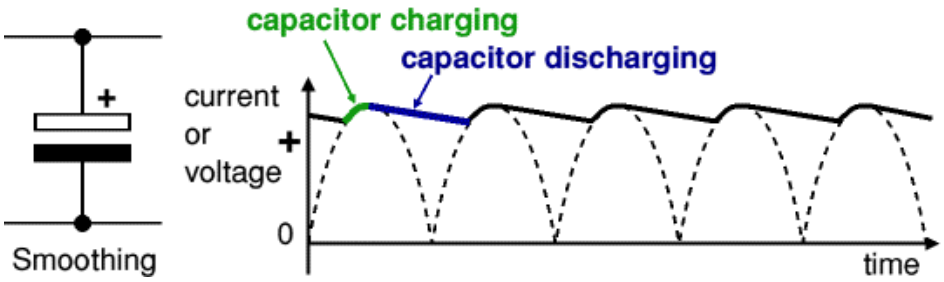
in special packages containing the four diodes required. It is called a full-wave rectifier because it uses all AC wave (both positive and negative sections). 1.4V is used up in the bridge rectifier because each diode uses 0.7V when conducting and there are always two diodes conducting, as shown in the diagram below. Bridge rectifiers are rated by the maximum current they can pass and the maximum reverse voltage they can withstand (this must be at least three times the supply RMS voltage so the rectifier can withstand the peak voltages). In this alternate pairs of diodes conduct, changing over the connections so the alternating directions of AC are converted to the one direction of DC.



OUTPUT – Full-wave Varying DC

SMOOTHING

Smoothing is performed by a large value electrolytic capacitor connected across the DC supply to act as a reservoir, supplying current to the output when the varying DC voltage from the rectifier is falling. The diagram shows the unsmoothed varying DC (dotted line) and the smoothed DC (solid line). The capacitor charges quickly near the peak of the varying DC, and then discharges as it supplies current to the output.



Note that smoothing significantly increases the average DC voltage to almost the peak value (1.4 × RMS value). For example 6V RMS AC is rectified to full wave DC of about 4.6V RMS (1.4V is lost in the bridge rectifier), with smoothing this increases to almost the peak value giving 1.4 × 4.6 = 6.4V smooth DC.

Smoothing is not perfect due to the capacitor voltage falling a little as it discharges, giving a small ripple voltage. For many circuits a ripple which is 10% of the supply voltage is satisfactory and the equation below gives the required value for the smoothing capacitor. A larger capacitor will give fewer ripples. The capacitor value must be doubled when smoothing half-wave DC.

Smoothing capacitor for 10% ripple, $C = 5 \times Io$

$$\mathbf{Vs \times f}$$

Where

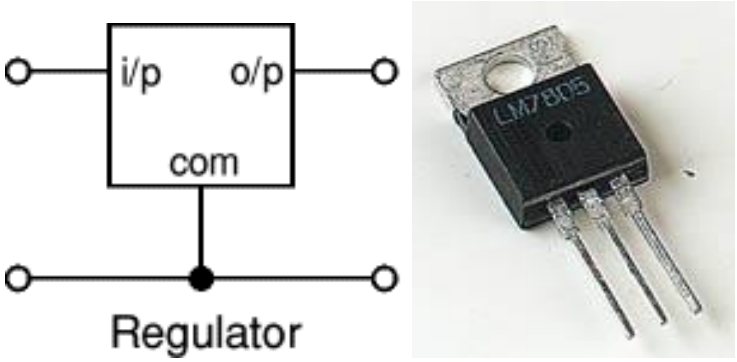
C = smoothing capacitance in farads (F)

Io = output current from the supply in amps (A)

Vs = supply voltage in volts (V), this is the peak value of the unsmoothed DC

f = frequency of the AC supply in hertz (Hz), 50Hz in the UK

REGULATOR

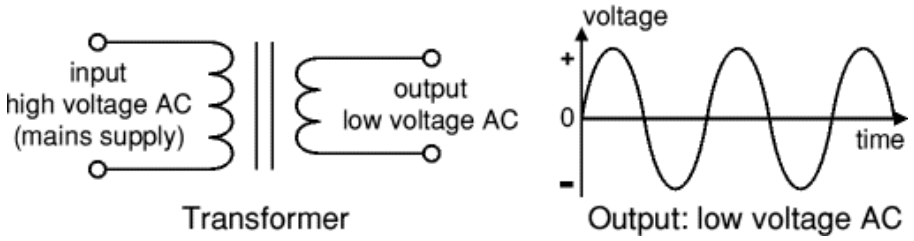


Voltage regulator ICs are available with fixed (typically 5, 12 and 15V) or variable

output voltages. They are also rated by the maximum current they can pass. Negative voltage regulators are available, mainly for use in dual supplies. Most regulators include some automatic protection from excessive current (‘overload protection') and overheating (‘thermal protection'). Many of the fixed voltage regulator ICs has 3 leads and look like power transistors, such as the 7805 +5V 1A regulator shown on the right. They include a hole for attaching a heat sink if necessary.

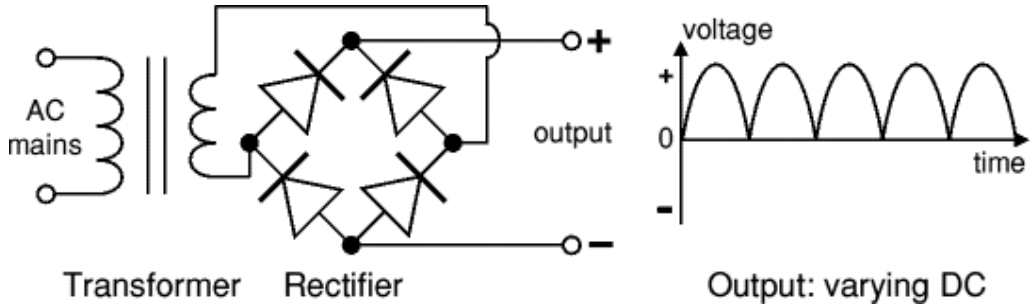
Working of Power Supply

- **Transformer**



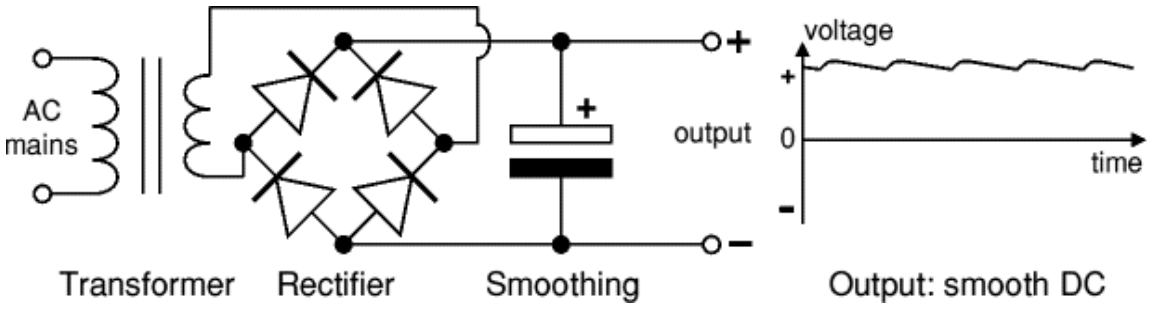
The low voltage AC output is suitable for lamps, heaters and special AC motors. It is not suitable for electronic circuits unless they include a rectifier and a smoothing capacitor.

- **Transformer + Rectifier**



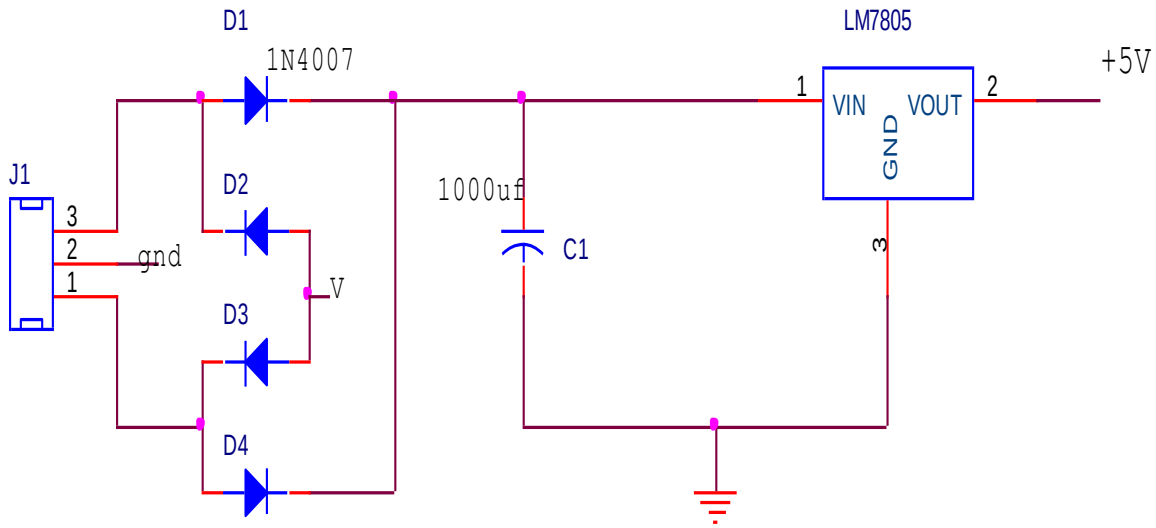
The varying DC output is suitable for lamps, heaters and standard motors. It is not suitable for electronic circuits unless they include a smoothing capacitor.

- **Transformer + Rectifier + Smoothing**



The smooth DC output has a small ripple. It is suitable for most electronic circuits.

- **Transformer + Rectifier + Smoothing + Regulator**



(Liquid Crystal Display)

Liquid crystal displays (LCD) are widely used in recent years as compares to LEDs. This is due to the declining prices of LCD, the ability to display numbers, characters and graphics, incorporation of a refreshing controller into the LCD, their by relieving the CPU of the task of refreshing the LCD and also the ease of programming for characters and graphics. HD 44780 based LCDs are most commonly used.

The LCD, which is used as a display in the system, is LMB162A. The main features of this LCD are: 16 X 2 display, intelligent LCD, used for alphanumeric characters & based on ASCII codes. This LCD contains 16 pins, in which 8 pins are used as 8-bit data I/O, which are extended ASCII. Three pins are used as control lines these are Read/Write pin, Enable pin and Register select pin. Two pins are used for Backlight and LCD voltage, another two pins are for Backlight & LCD ground and one pin is used for contrast change.

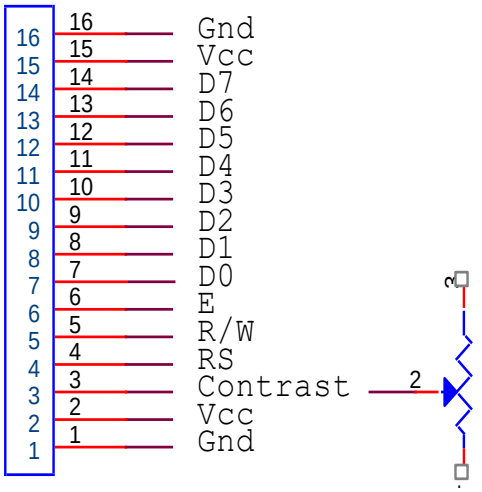
LCD pin description

Pin	Symbol	I/O	Description
1	VSS	-	Ground
2	VCC	-	+5V power supply
3	VEE	-	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register.
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8 bit data bus
8	DB1	I/O	The 8 bit data bus
9	DB2	I/O	The 8 bit data bus
10	DB3	I/O	The 8 bit data bus

11	DB4	I/O	The 8 bit data bus
12	DB5	I/O	The 8 bit data bus
13	DB6	I/O	The 8 bit data bus
14	DB7	I/O	The 8 bit data bus

LCD pin description

The LCD discuss in this section has the most common connector used for the Hitachi 44780 based LCD is 14 pins in a row and modes of operation and how to program and interface with microcontroller is describes in this section.



LCD Pin Description Diagram

V_{CC}, V_{SS}, V_{EE}

The voltage V_{CC} and V_{SS} provided by +5V and ground respectively while V_{EE} is used for controlling LCD contrast. Variable voltage between Ground and V_{cc} is used to specify the contrast (or "darkness") of the characters on the LCD screen.

RS (register select)

There are two important registers inside the LCD. The RS pin is used for their selection as follows. If RS=0, the instruction command code register is selected, then allowing to user to send a command such as clear display, cursor at home etc..

If RS=1, the data register is selected, allowing the user to send data to be displayed on the LCD.

R/W (read/write)

The R/W (read/write) input allowing the user to write information from it. R/W=1, when it read and R/W=0, when it writing.

EN (enable)

The enable pin is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high power, a high-to-low pulse must be applied to this pin in order to for the LCD to latch in the data presented at the data pins.

D0-D7 (data lines)

The 8-bit data pins, D0-D7, are used to send information to the LCD or read the contents of the LCD's internal registers. To displays the letters and numbers, we send ASCII codes for the letters A-Z, a-z, and numbers 0-9 to these pins while making RS =1. There are also command codes that can be sent to clear the display or force the cursor to the home position or blink the cursor.

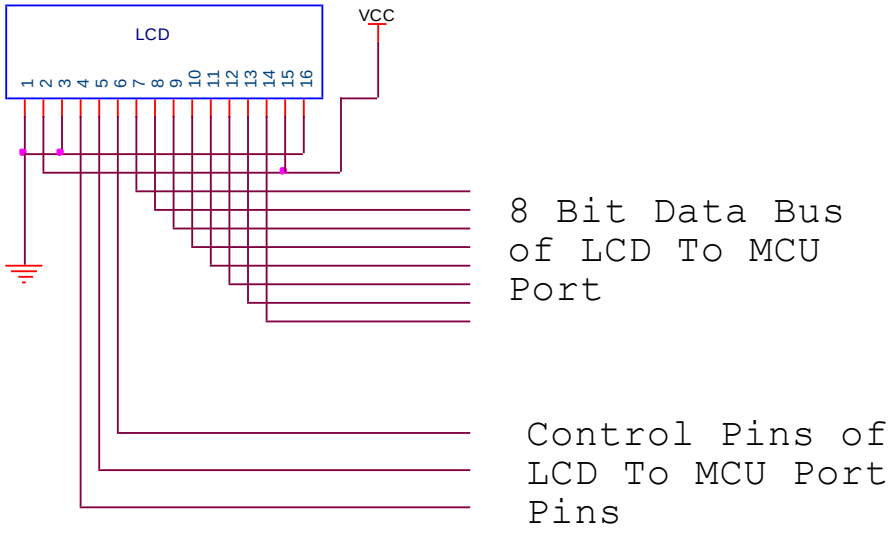
We also use RS =0 to check the busy flag bit to see if the LCD is ready to receive the information. The busy flag is D7 and can be read when R/W =1 and RS =0, as follows: if R/W =1 and RS =0, when D7 =1(busy flag =1), the LCD is busy taking care of internal operations and will not accept any information. When D7 =0, the LCD is ready to receive new information.

Interfacing of micro controller with LCD display

In most applications, the "R/W" line is grounded. This simplifies the application because when data is read back, the microcontroller I/O pins have to be alternated between input and output modes.

In this case, "R/W" to ground and just wait the maximum amount of time for each instruction (4.1ms for clearing the display or moving the cursor/display to the "home position", 160µs for all other commands) and also the application software is simpler, it also frees up a microcontroller pin for other uses. Different LCD

execute instructions at different rates and to avoid problems later on (such as if the LCD is changed to a slower unit). Before sending commands or data to the LCD module, the Module must be initialized. Once the initialization is complete, the LCD can be written to with data or instructions as required. Each character to display is written like the control bytes, except that the "RS" line is set. During initialization, by setting the "S/C" bit during the "Move Cursor/Shift Display" command, after each character is sent to the LCD, the cursor built into the LCD will increment to the next position (either right or left). Normally, the "S/C" bit is set (equal to "1")



Interfacing of Microcontroller with LCD

LCD Command Code

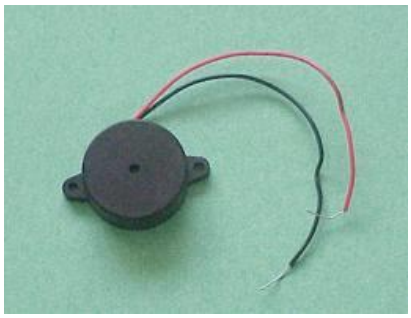
Code (HEX)	Command to LCD Instruction Register
1	Clear the display screen
2	Return home
4	Decrement cursor(shift cursor to left)
6	Increment cursor(shift cursor to right)
7	Shift display right
8	Shift display left

The original model became far more popular than expected, selling uses such as robotics outside of its target market.

9	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to left
1C	Shift the entire display to right
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	2 line and 5×7 matrix

Buzzer for Beep Source

A **buzzer** or **beeper** is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.



Buzzer

Mechanical Buzzer

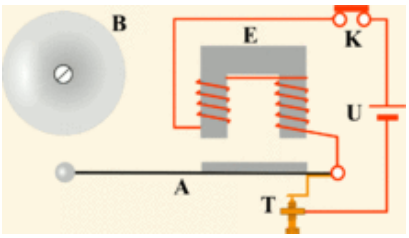
A joy buzzer is an example of a purely mechanical buzzer.

A joy buzzer (also called a hand buzzer) is a practical joke device that consists of a coiled spring inside a disc worn in the palm of the hand. When the wearer shakes hands with another person, a button on the disc releases the spring, which rapidly unwinds creating a vibration that feels somewhat like an electric shock to someone not expecting it.

Electromechanical Buzzer

Early devices were based on an electromechanical system identical to an electric bell without the metal gong. Similarly, a relay may be connected to interrupt its own actuating current, causing the contacts to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.

An electric bell is a mechanical bell that functions by means of an electromagnet. When an electric current is applied, it produces a repetitive buzzing or clanging sound. Electric bells have been widely used at railroad crossings, in telephones, fire and burglar alarms, as school bells, doorbells, and alarms in industrial plants, but they are now being widely replaced with electronic sounders.



An electric buzzer uses a similar mechanism to an interrupter bell, but without the resonant bell. They are quieter than bells, but adequate for a warning tone over a small distance, such as across a desktop.

With the development of low cost electronics from the 1970s onwards, most buzzers have now been replaced by electronic 'sounders'. These replace the electromechanical striker of a bell with an electronic oscillator and a loudspeaker, often a piezo transducer.

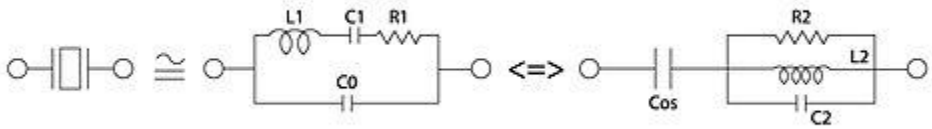
Piezoelectric buzzer

A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

Oscillation is the repetitive variation, typically in time, of some measure about a central value (often a point of equilibrium) or between two or more different states.

Familiar examples include a swinging pendulum and AC power. The term vibration is sometimes used more narrowly to mean a mechanical oscillation but sometimes is used to be synonymous with "oscillation". Oscillations occur not only in physical systems but also in biological systems and in human society.

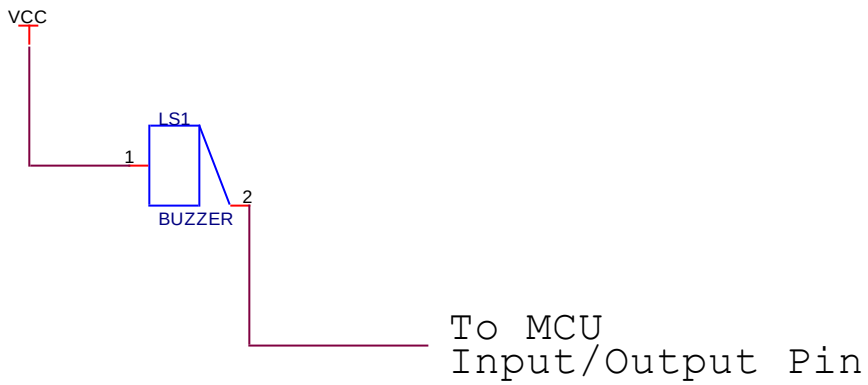
A piezoelectric audio amplifier (PAA) is a single integrated circuit or a PCB developed to amplify small audio signals to drive piezoelectric audio loudspeaker elements. A piezo audio amplifier can amplify a small signal sine wave of 1 volt peak-to-peak to a signal of about 30 or 60 Volts.



Uses

- Annunciate panels
- Electronic metronomes
- Game shows
- Microwave ovens and other household appliances
- Sporting events such as basketball games
- Electrical alarms
- Buzzers

Interfacing Circuit to MCU



Passive Infra Red Sensor

A **passive Infrared sensor** (**PIR sensor**) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.

Operating Principle

All objects above absolute zero emit heat energy in the form of infrared radiation (infrared light). Usually infrared light is invisible to the human eye, but it can be detected by electronic devices designed for such a purpose.

The term *passive* in this instance refers to the fact that PIR devices do not generate or radiate any energy for detection purposes. They work entirely by detecting the energy given off by other objects.

Construction

Infrared radiation enters through the front of the sensor, known as the *sensor face*. At the core of a PIR sensor is a solid state *sensor* or set of sensors, made from pyroelectric materials -- materials which generate energy when exposed to heat. Typically, the sensors are approximately 1/4 inch square, and take the form of a thin film. Materials commonly used in PIR sensors include gallium nitride (GaN), caesium nitrate ($CsNO_3$), polyvinyl fluorides, derivatives of phenylpyrazine, and cobalt phthalocyanine. The sensor is often manufactured as part of an integrated circuit.

PIR based motion detector

A PIR-based motion detector is used to sense movement of people, animals, or other objects. They are commonly used in burglar alarms and automatically-activated lighting systems. They are commonly called simply "PIR", or sometimes "PID", for *passive infrared detector*.

Operation

Strictly speaking, individual PIR sensors do not detect motion; rather, they detect abrupt changes in temperature at a given point. As an object, such as a human, passes in front of the background, such as a wall, the temperature at that point will rise from room temperature to body temperature, and then back again. This quick change triggers the detection. Moving objects of identical temperature will not trigger detection.

PIDs can be equipped with more than one internal sensing element so that, with the appropriate electronics, it can detect the apparent direction of movement. As an object passes in front of adjacent sensors in turn, this implies the direction of movement. This may be used by on-board electronics to reduce false alarms, i.e., by requiring adjacent sensors to trip in succession. It may also be used to signal the direction of movement to a monitoring apparatus.

PIDs come in many configurations for a wide variety of applications. The most common models have numerous Fresnel lenses or mirror segments, an effective range of about thirty feet, and a field of view less than 180 degrees. Models with wider fields of view, including 360 degrees, are available -- typically designed to mount on a ceiling. Some larger PIDs are made with single segment mirrors and can sense changes in infrared energy over one hundred feet away from the PID. There are also PIDs designed with reversible orientation mirrors which allow either broad coverage (110° wide) or very narrow "curtain" coverage, or with individually selectable segments to "shape" the coverage.

Differential detection

Pairs of sensor elements may be wired as opposite inputs to a differential amplifier. In such a configuration, the PIR measurements cancel each other so that the average temperature of the field of view is removed from the electrical signal; an increase of IR energy across the entire sensor is self-cancelling and will not trigger the device. This allows the device to resist false indications of change in the event of being exposed to brief flashes of light or field-wide illumination. (Continuous high energy exposure may still be able to saturate the sensor materials

and render the sensor unable to register further information.) At the same time, this differential arrangement minimizes common-mode interference, allowing the device to resist triggering due to nearby electric fields. However, a differential pair of sensors cannot measure temperature in this configuration, and therefore is only useful for motion detection.

Product design

The PIR sensor is typically mounted on a printed circuit board containing the necessary electronics required to interpret the signals from the sensor itself. The complete assembly is usually contained within a housing, mounted in a location where the sensor can cover area to be monitored.

The housing will usually have a plastic "window" through which the infrared energy can enter. Despite often being only translucent to visible light, infrared energy is able to reach the sensor through the window because the plastic used is transparent to infrared radiation. The plastic window reduces the chance of foreign objects (dust, insects, etc.) from obscuring the sensor's field of view, damaging the mechanism, and/or causing false alarms. The window may be used as a filter, to limit the wavelengths to 8-14 micrometres, which is closest to the infrared radiation emitted by humans. It may also serve as a focusing mechanism; see below.

Focusing

Different mechanisms can be used to focus the distant infrared energy onto the sensor surface.

Lenses

The plastic window covering may have multiple facets molded into it, to focus the infrared energy onto the sensor. Each individual facet is a Fresnel lens.

Mirrors

Some PIDs are manufactured with internal, segmented parabolic mirrors to focus the infrared energy. Where mirrors are used, the plastic window cover generally has no Fresnel lenses molded into it.

Motion detector module uses a motion detector IC and PCB mounted Fresnel lens (Item: SB0061)

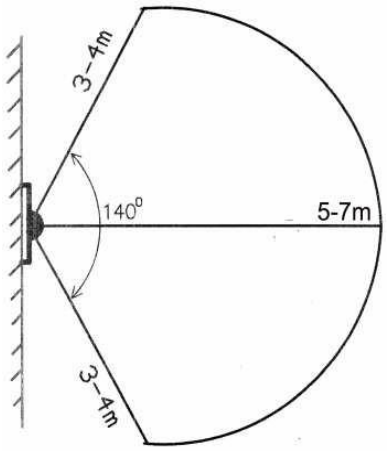
General

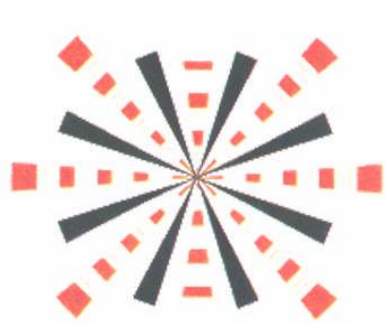
SB0061 is a pyroelectric sensor module which developed for human body detection. A PIR detector combined with a fresnel lens are mounted on a compact size PCB together with an analog IC, SB0061, and limited components to form the module. High level output of variable width is provided.

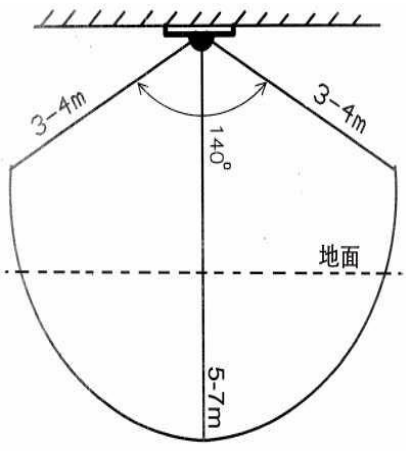
Features and Electrical Specification

- Compact size (28 x 38 mm)
- Supply current: DC5V-20V(can design DC3V-24V)
- Current drain :< 50uA (Other choice: DC0.8V-4.5V; Current drain: 1.5mA-0.1mA)
- Voltage Output: High/Low level signal ：3.3V (Other choice: Open-Collector Output)
- TTL output
- High sensitivity
- Delay time：5s-18 minute
- Blockade time：0.5s-50s (acquiescently 0 seconds)
- Operation Temperature: -15oC -70Oc
- Infrared sensor: dual element, low noise, high sensitivity
- Light sensor: CdS photocell (can be add as customer requirement)

Lens information







Application Note

1. Power anode

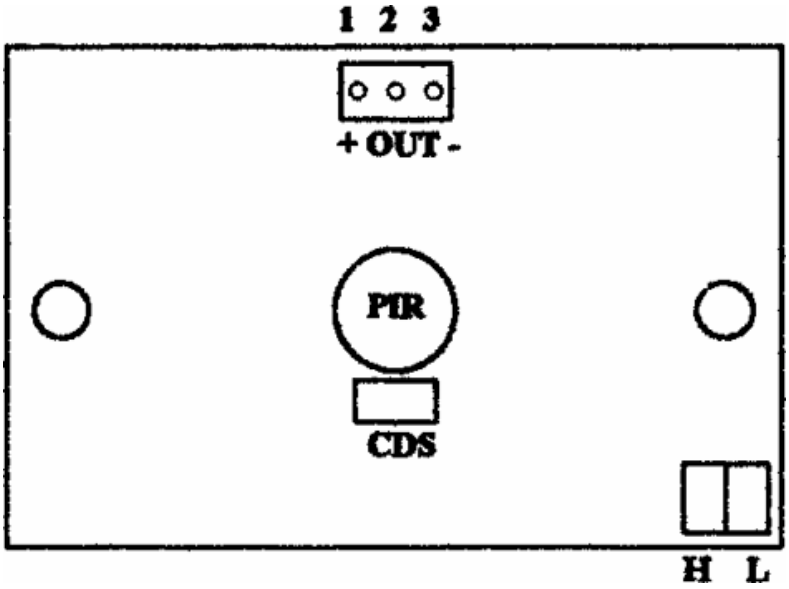
2. Output: High level signal

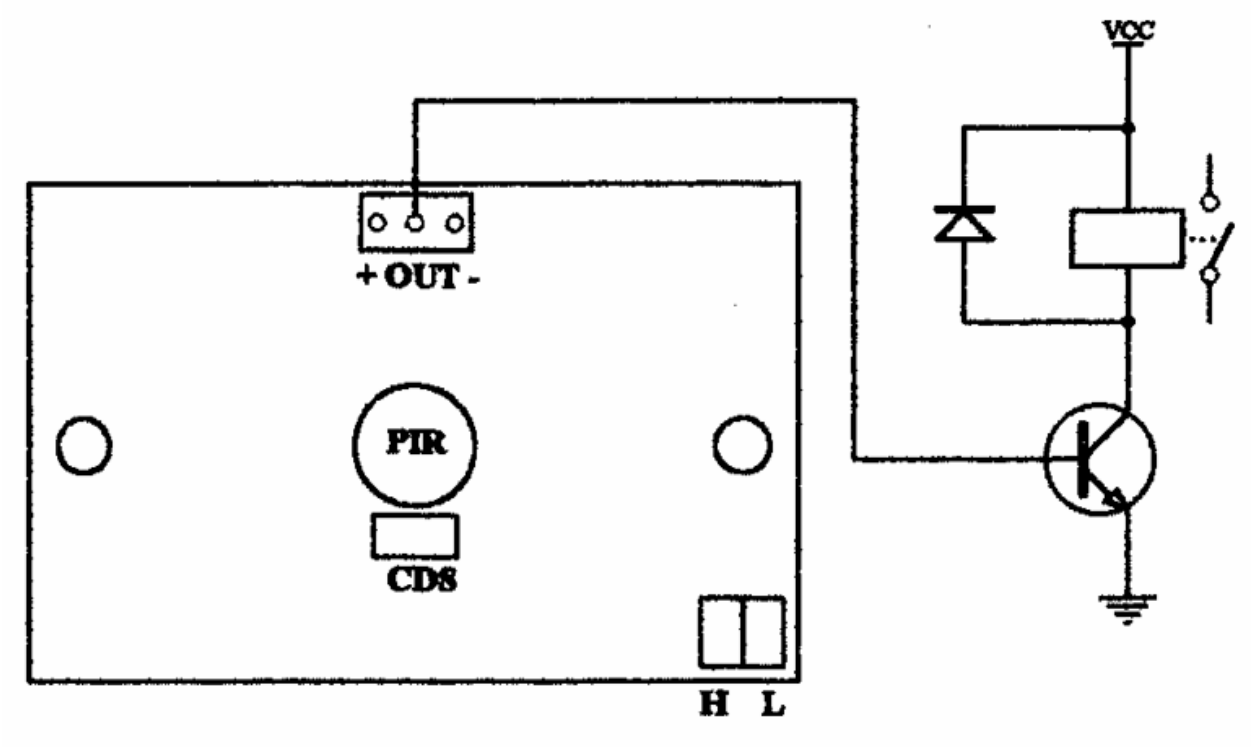
3. Power cathode

H: Can be spring repeatedly

L: Can not be spring repeatedly

CDS: Photocell





Note

Due to the high sensitivity of PIR sensor device, it is not recommended to use the module in the following or similar condition.

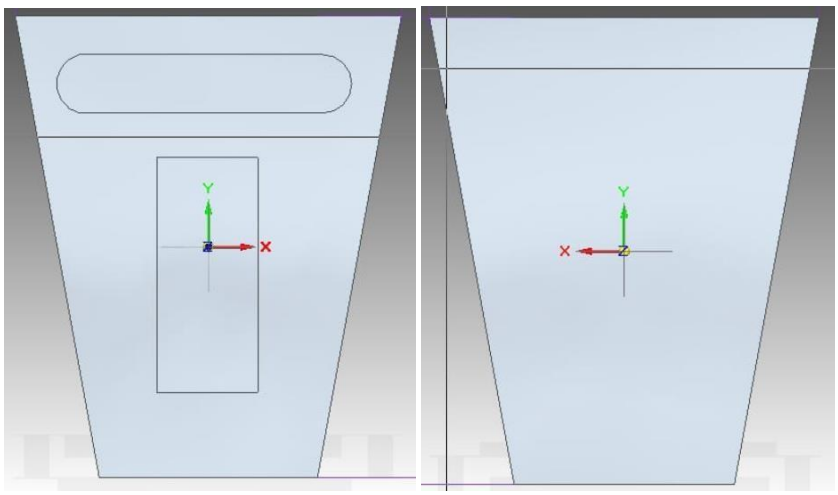
A) in rapid environmental changes

B) in strong shock or vibration

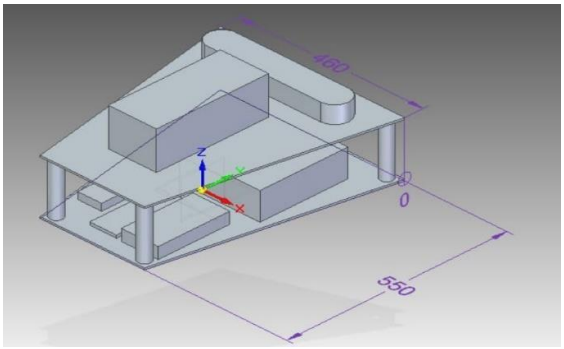
C) in a place where there are obstructing material (eg. glass) through which IR cannot pass within detection area.

D) exposed to direct sun light

E) exposed to direct wind from a heater or air condition



(e)

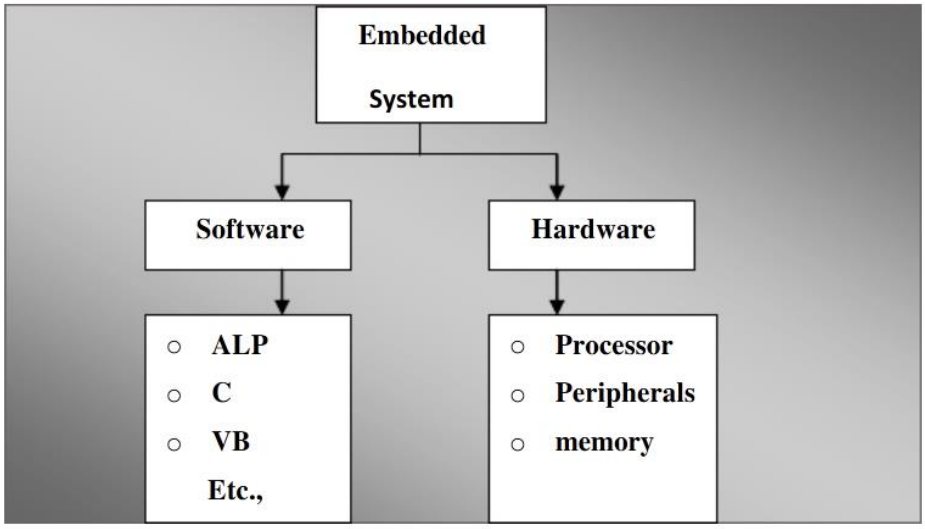


Microcontroller AVR Mega Series

Embedded System

An embedded system is a system which is going to do a predefined specified task is the embedded system and is even defined as combination of both software and hardware. A general-purpose definition of embedded systems is that they are devices used to control, monitor or assist the operation of equipment, machinery or plant. "Embedded" reflects the fact that they are an integral part of the system. At the other extreme a general-purpose computer may be used to control the operation of a large complex processing plant, and its presence will be obvious .All embedded systems are including computers or microprocessors. Some of these computers are however very simple systems as compared with a personal computer .The very simplest embedded systems are capable of performing only a single function or set of functions to meet a single predetermined purpose. In more complex systems an application program that enables the embedded system to be used for a particular purpose in a specific application determines the functioning of the embedded system. The ability to have programs means that the same embedded system can be used for a variety of different purposes. In some cases a microprocessor may be designed in such a way that application software for a particular purpose can be added to the basic software in a second process, after which it is not possible to make further changes. The applications software on such processors is sometimes referred to as firmware .The simplest devices consist of a single microprocessor (often called a "chip‖), which may itself be packaged with other chips in a hybrid system or Application Specific Integrated Circuit (ASIC). Its input comes from a detector or sensor and its output goes to a switch or activator which (for example) may start or stop the operation of a machine or, by operating a valve, may control the flow of fuel to an engine.

As the embedded system is the combination of both software and hardware



Block diagram of Embedded System

Software deals with the languages like ALP, C, and VB etc., and Hardware deals with Processors, Peripherals, and Memory.

Memory: It is used to store data or address.

Peripherals: These are the external devices connected

Processor: It is an IC which is used to perform some task

Applications of embedded systems

- Manufacturing and process control
- Construction industry
- Transport
- Buildings and premises
- Domestic service
- Communications
- Office systems and mobile equipment
- Banking, finance and commercial
- Medical diagnostics, monitoring and life support
- Testing, monitoring and diagnostic systems

Processors are classified into four types like:

- Micro Processor (µp)

- Micro controller (µc)
- Digital Signal Processor (DSP)
- Application Specific Integrated Circuits (ASIC)

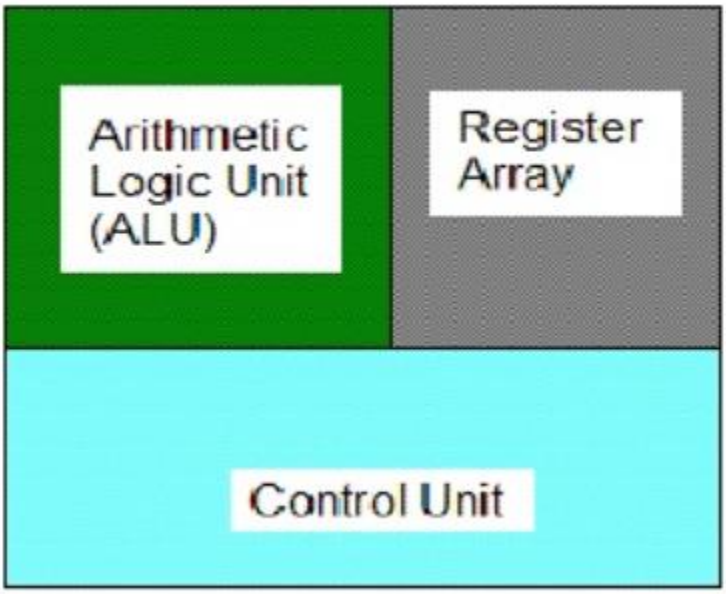
Micro Processor (µp):

A silicon chip that contains a CPU is called MPU. In the world of personal computer the terms microprocessor and CPU are used interchangeably. At the heart of all personal computers and most workstations sits a microprocessor. Microprocessors also control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles.

Three basic characteristics differentiate microprocessors:

- Instruction set:-The set of instructions that the microprocessor can execute.
- Bandwidth :-The number of bits processed in a single instruction.
- Clock speed:-Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute. In both cases, the higher the value, the more powerful the CPU. For example, a 32-bit microprocessor that runs at 50MHz is more powerful than a 16-bit microprocessor that runs at 25MHz. In addition to bandwidth and clock speed, microprocessors are classified as being either RISC (reduced instruction set computer) or CISC (complex instruction set computer).

A microprocessor has three basic elements, as shown above. The ALU performs all arithmetic computations, such as addition, subtraction and logic operations (AND, OR, etc). It is controlled by the Control Unit and receives its data from the Register Array. The Register Array is a set of registers used for storing data. These registers can be accessed by the ALU very quickly. Some registers have specific functions - we will deal with these later. The Control Unit controls the entire process. It provides the timing and a control signal for getting data into and out of the registers and the ALU and it synchronizes the execution of instructions (we will deal with instruction execution at a later date)



Three Basic Elements of a Microprocessor

Micro Controller (µc)

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. 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.

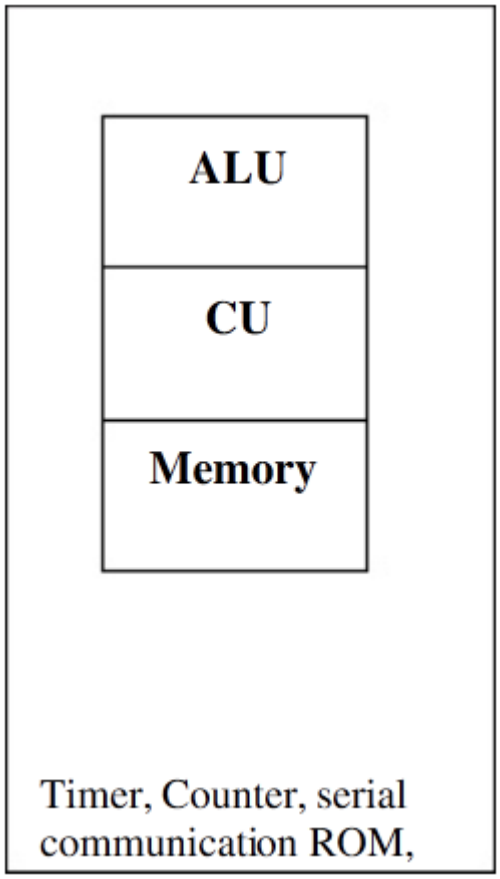


Figure: Block Diagram of Micro Controller (µc)

Digital Signal Processors (DSPs)

Digital Signal Processors is one which performs scientific and mathematical operation. Digital Signal Processor chips - specialized microprocessors with architectures designed specifically for

The types of operations required in digital signal processing. Like a general-purpose microprocessor, a DSP is a programmable device, with its own native instruction code. DSP chips are capable of carrying out millions of floating point operations per second, and like their better-known general-purpose cousins, faster and more powerful versions are continually being introduced. DSPs can also be embedded within complex "system-on-chip" devices, often containing both analog and digital circuitry.

Application Specific Integrated Circuit (ASIC)

ASIC is a combination of digital and analog circuits packed into an IC to achieve the desired control/computation function

- CPU cores for computation and control

- Peripherals to control timing critical functions
- Memories to store data and program
- Analog circuits to provide clocks and interface to the real world which is analog in nature
- I/Os to connect to external components like LEDs, memories, monitors etc.

Computer Instruction Set

There are two different types of computer instruction set there are:

1. RISC (Reduced Instruction Set Computer)
2. CISC (Complex Instruction Set computer)

Reduced Instruction Set Computer (RISC)

- Reduced Instruction Set Computer
- As compared to Complex Instruction Set Computers, i.e. x86
- Assumption: Simpler instructions execute faster
- Optimized most used instructions
- Other RISC machines: ARM, PowerPC, SPARC

A RISC (reduced instruction set computer) is a microprocessor that is designed to perform a smaller number of types of computer instruction so that it can operate at a higher speed (perform more million instructions per second, or millions of instructions per second). Since each instruction type that a computer must perform requires additional transistors and circuitry, a larger list or set of computer instructions tends to make the microprocessor more complicated and slower in operation. Besides performance improvement, some advantages of RISC and related design improvements are

- A new microprocessor can be developed and tested more quickly if one of its aims is to be less complicated.
- Operating system and application programmers who use the microprocessor's instructions will find it easier to develop code with a smaller instruction set.
- The simplicity of RISC allows more freedom to choose how to use the space on a microprocessor. Higher-level language compilers produce more efficient code than

formerly because they have always tended to use the smaller set of instructions to be found in a RISC computer.

RISC characteristics

Simple instruction set

In a RISC machine, the instruction set contains simple, basic instructions, from which more. Complex instructions can be composed.

Same length instructions

Each instruction is the same length, so that it may be fetched in a single operation. Most instructions complete in one machine cycle, which allows the processor to handle several instructions at the same time. This pipelining is a key technique used to speed up RISC machines.

Complex Instruction Set Computer (CISC)

CISC, which stands for Complex Instruction Set Computer, is a philosophy for designing chips that are easy to program and which make efficient use of memory. Each instruction in a CISC instruction set might perform a series of operations inside the processor. This reduces the number of instructions required to implement a given program, and allows the programmer to learn a small but flexible set of instructions.

The advantages of CISC

At the time of their initial development, CISC machines used available technologies to optimize computer performance.

- Microprogramming is as easy as assembly language to implement, and much less expensive than hardwiring a control unit.
- The ease of micro-coding new instructions allowed designers to make CISC machines upwardly compatible: a new computer could run the same programs as earlier computers because the new computer would contain a superset of the instructions of the earlier computers.
- As each instruction became more capable, fewer instructions could be used to implement a given task. This made more efficient use of the relatively slow main memory.

- Because micro program instruction sets can be written to match the constructs of high-level languages, the compiler does not have to be as complicated.

Disadvantages of CISC

Still, designers soon realized that the CISC philosophy had its own problems, including:

- Earlier generations of a processor family generally were contained as a subset in every new version --- so instruction set & chip hardware become more complex with each generation of computers.
- So that as many instructions as possible could be stored in memory with the least possible wasted space, individual instructions could be of almost any length---this means that different instructions will take different amounts of clock time to execute, slowing down the overall performance of the machine.
- Many specialized instructions aren't used frequently enough to justify their existence --- approximately 20% of the available instructions are used in a typical program.
- CISC instructions typically set the condition codes as a side effect of the instruction. Not only does setting the condition codes take time, but programmers have to remember to examine the condition code bits before a subsequent instruction changes them.

Memory Architecture

There two different type's memory architectures there are:

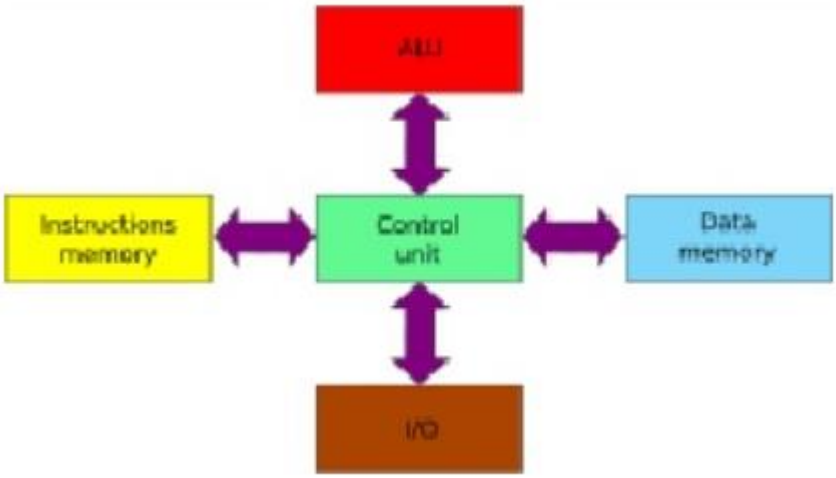
- Harvard Architecture
- Von-Neumann Architecture

Harvard Architecture

Computers have separate memory areas for program instructions and data. There are two or more internal data buses, which allow simultaneous access to both instructions and data. The CPU fetches program instructions on the program memory bus.

The **Harvard architecture** is computer architecture with physically separate storage and signal pathways for instructions and data. The term originated from the Harvard Mark Irelay-based computer, which stored instructions on punched tape (24 bits wide) and data in electro-mechanical counters. These early machines had limited data storage, entirely contained within the central processing unit and provided no access to the instruction storage as data. Programs

needed to be loaded by an operator, the processor could not boot itself.



Harvard Architecture Modern uses of the Harvard architecture

Modern uses of the Harvard architecture

The principal advantage of the pure Harvard architecture - simultaneous access to more than one memory system - has been reduced by modified Harvard processors using modern CPU cache systems. Relatively pure Harvard architecture machines are used mostly in applications where tradeoffs, such as the cost and power savings from omitting caches, outweigh the programming penalties from having distinct code and data address spaces.

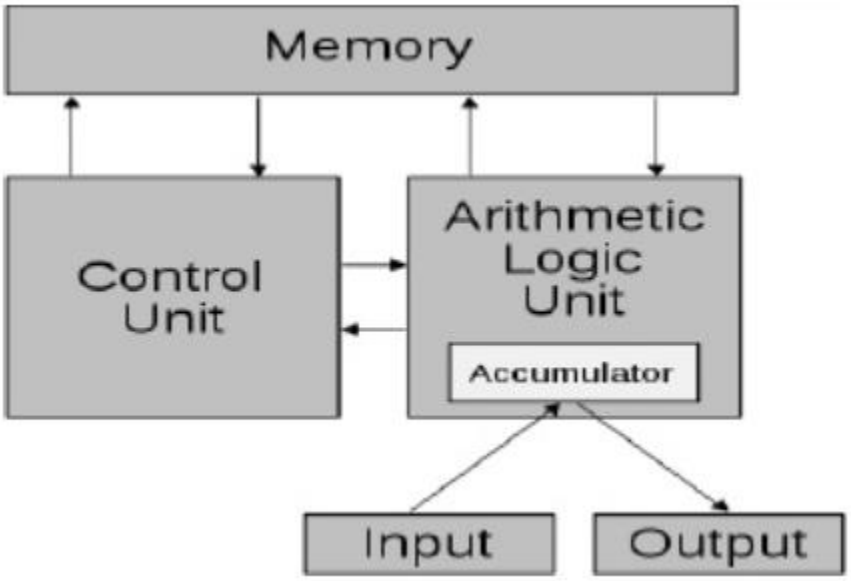
- Digital signal processors (DSPs) generally execute small, highly-optimized audio or video processing algorithms. They avoid caches because their behavior must be extremely reproducible. The difficulties of coping with multiple address spaces are of secondary concern to speed of execution. As a result, some DSPs have multiple data memories in distinct address spaces to facilitate SIMD and VLIW processing. Texas Instruments TMS320 C55x processors, as one example, have multiple parallel data busses (two write, three read) and one instruction bus.
- Microcontrollers are characterized by having small amounts of program (flash memory) and data (SRAM) memory, with no cache, and take advantage of the Harvard architecture to speed processing by concurrent instruction and data access. The separate storage means the program and data memories can have different bit depths, for example using 16-bit wide instructions and 8-bit wide data. They also mean that

instruction pre-fetch can be performed in parallel with other activities. Examples include, the AVR by Atmel Corp, the PIC by Microchip Technology, Inc. and the ARM Cortex-M3 processor (not all ARM chips have Harvard architecture).Even in these cases, it is common to have special instructions to access program memory as data for read-only tables, or for reprogramming.

Von-Neumann Architecture

A computer has a single, common memory space in which both program instructions and data are stored. There is a single internal data bus that fetches both instructions and data. They cannot be performed at the same time The von Neumann architecture is a design model for a stored-program digital computer that uses a central processing unit (CPU) and a single separate storage structure ("memory") to hold both instructions and data .It is named after the mathematician and early computer scientist John von Neumann. Such computers implement a universal Turing machine and have a sequential architecture.

A stored-program digital computer is one that keeps its programmed instructions, as well as its data, in read-write, random-access memory (RAM). Stored-program computers were advancement over the program-controlled computers of the 1940s, such as the Colossus and the ENIAC, which were programmed by setting switches and inserting patch leads to route data and to control signals between various functional units. In the vast majority of modern computers, the same memory is used for both data and program instructions. The mechanisms for transferring the data and instructions between the CPU and memory are, however, considerably more complex than the original von Neumann architecture. The terms "von Neumann architecture" and "stored-program computer" are generally used interchangeably, and that usage is followed in this article.



Schematic of the Von-Neumann Architecture

Basic Difference between Harvard and Von-Neumann Architecture

- The primary difference between Harvard architecture and the Von Neumann architecture is in the Von Neumann architecture data and programs are stored in the same memory and managed by the same information handling system.
- Whereas the Harvard architecture stores data and programs in separate memory devices and they are handled by different subsystems.

- In a computer using the Von-Neumann architecture without cache; the central processing unit (CPU) can either be reading and instruction or writing/reading data to/from the memory. Both of these operations cannot occur simultaneously as the data and instructions use the same system bus.

- In a computer using the Harvard architecture the CPU can both read an instruction and access data memory at the same time without cache. This means that a computer with Harvard architecture can potentially be faster for a given circuit complexity because data access and instruction fetches do not contend for use of a single memory pathway.
- Today, the vast majority of computers are designed and built using the Von Neumann architecture template primarily because of the dynamic capabilities and efficiencies gained in designing, implementing, operating one memory system as opposed to two.

Von Neumann architecture may be somewhat slower than the contrasting Harvard Architecture for certain

- Specific tasks, but it is much more flexible and allows for many concepts unavailable to Harvard architecture such as self programming, word processing and so on.
- Harvard architectures are typically only used in either specialized systems or for very specific uses. It is used in specialized digital signal processing (DSP), typically for video and audio processing products. It is also used in many small microcontrollers used in electronics applications such as Advanced RISK Machine (ARM) based products for many vendors

What is AVR?

- Modified Harvard architecture 8-bit RISC single chip microcontroller
- Complete System-on-a-chip
    - On Board Memory (FLASH, SRAM & EEPROM)
    - On Board Peripherals
- Advanced (for 8 bit processors) technology
- Developed by Atmel in 1996
- First In-house CPU design by Atmel

AVR Family

- 8 Bit tinyAVR

  Small package – as small as 6 pins
- 8 Bit megaAVR

  Wide variety of configurations and packages
- 8 / 16 Bit AVR XMEGA

  Second Generation Technology
- 32 Bit AVR UC3

  Higher computational throughput

Overview: The ATmega8535 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing instructions in a single clock cycle, the ATmega8535 achieves throughputs approaching 1 MIPS per MHz

allowing the system designed to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All 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 ATmega8535 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes EEPROM, 512 bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain in TQFP package, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six 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 Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. 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. In Extended Standby mode, both the main Oscillator and the asynchronous timer continue to run.

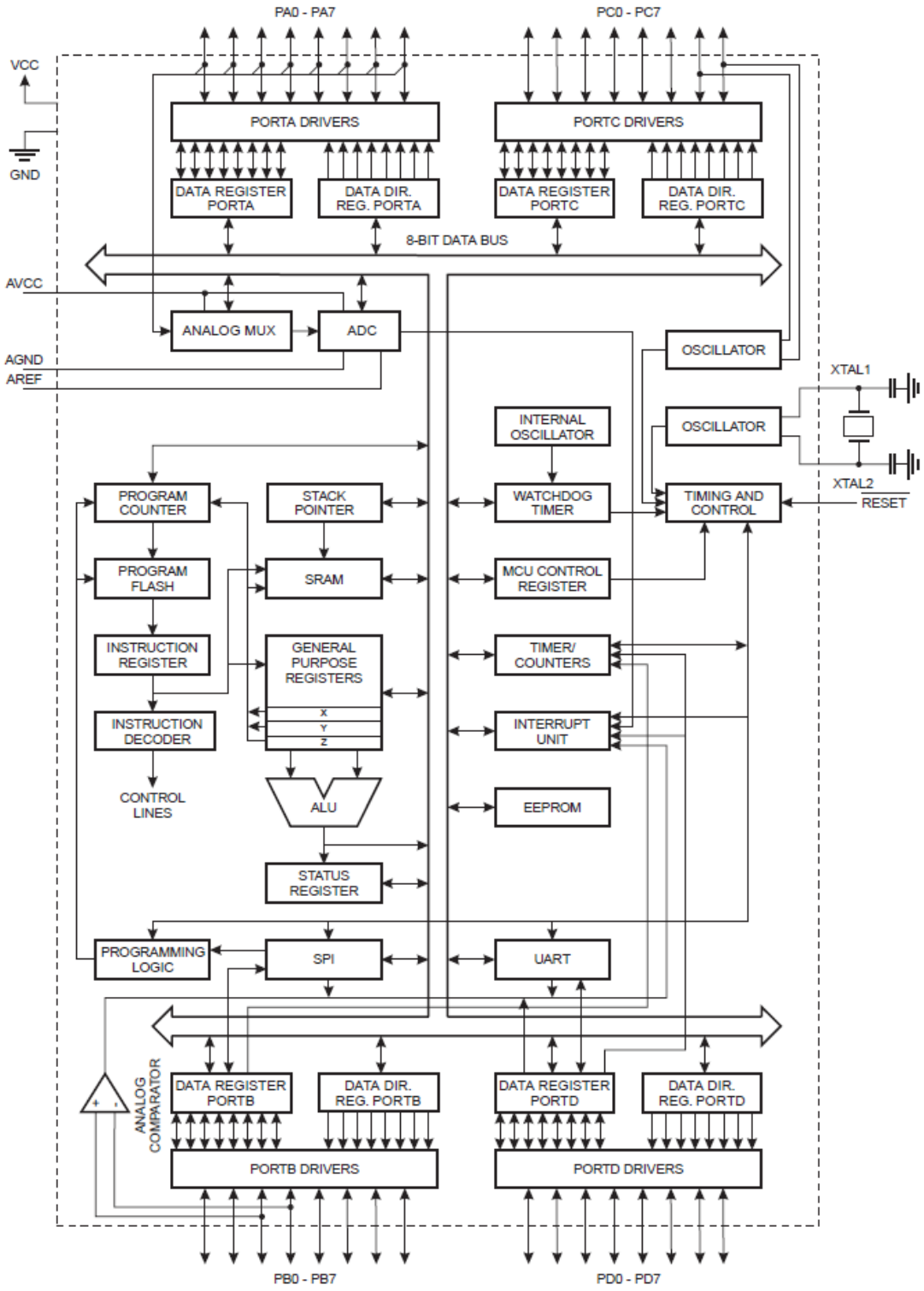
The device is manufactured using Atmel’s high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be

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 ATmega8535 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

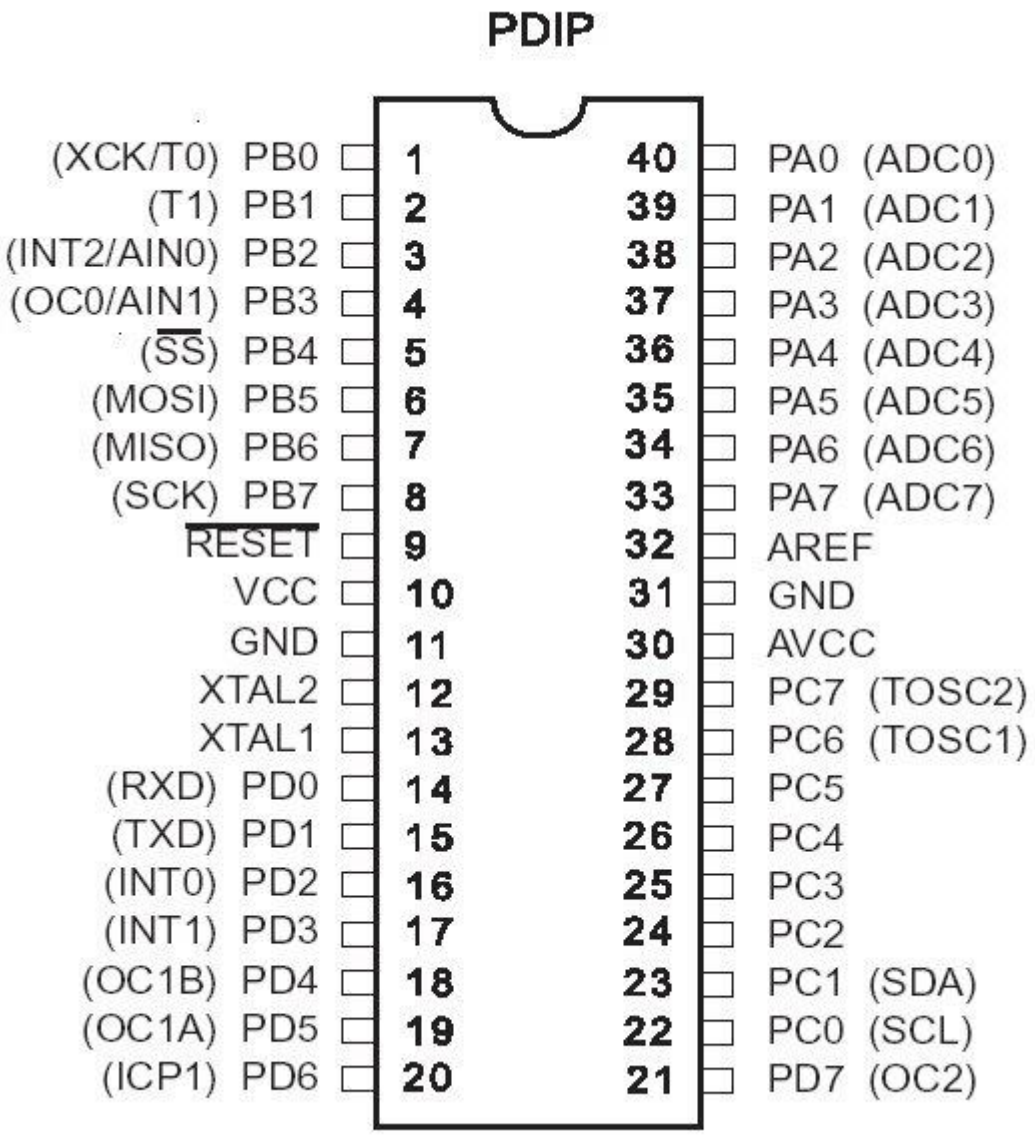
The ATmega8535 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In Circuit Emulators, and evaluation kits.

AT90S8535 Compatibility The ATmega8535 provides all the features of the AT90S8535. In addition, several new features are added. The ATmega8535 is backward compatible with AT90S8535 in most cases. However, some incompatibilities between the two microcontrollers exist. To solve this problem, an AT90S8535 compatibility mode can be selected by programming the S8535C fuse. ATmega8535 is pin compatible with AT90S8535, and can replace the AT90S8535 on current Printed Circuit Boards. However, the location of fuse bits and the electrical characteristics differs between the two devices.

Block Diagram



Pin Description



Pin Descriptions

VCC: Digital supply voltage

GND: Ground

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.
Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide 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 Port A 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.

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. Shorter pulses are not guaranteed to generate a reset.

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

XTAL2 Output from the inverting Oscillator amplifier.

AVCC: AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.

AREF: AREF is the analog reference pin for the A/D Converter.

About Code

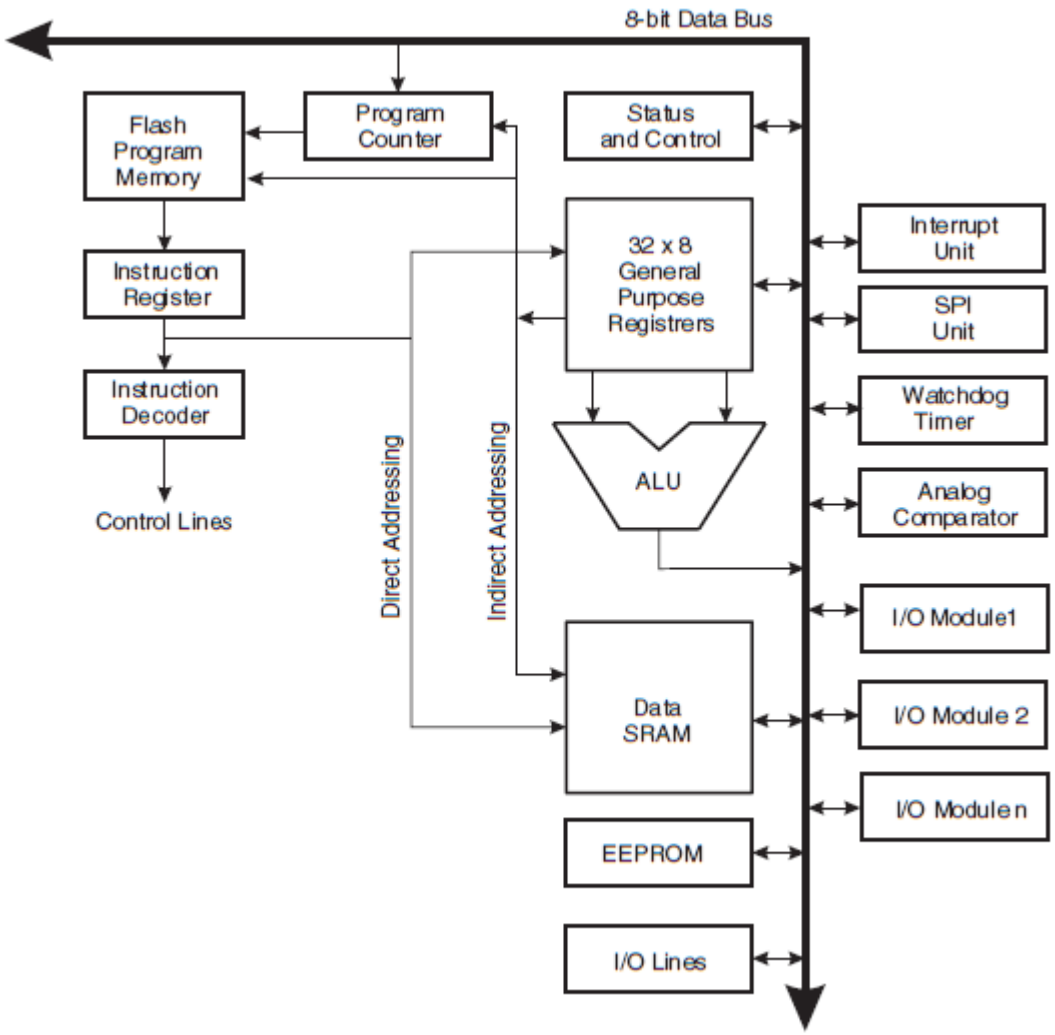
Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C Compiler documentation for more details.

AVR CPU Core

Introduction This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Architectural Overview Figure: Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In- System Re-Programmable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU)

Operation: In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect addresses register pointers for Data Space addressing – enabling efficient address calculations. One of these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-registers, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is Stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps. A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector

address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F.

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions.

Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the "Instruction Set" section for a detailed description.

Status Register The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will, in many cases, remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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 7 – I: Global Interrupt Enable

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are

enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The Ibit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

• Bit 6 – T: Bit Copy Storage

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

• Bit 5 – H: Half Carry Flag

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half carry is useful in BCD arithmetic

• Bit 4 – S: Sign Bit, S = N ⊕ V

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V

• Bit 3 – V: Two's Complement Overflow Flag

The Two's Complement Overflow Flag V supports two's complement arithmetics.

• Bit 2 – N: Negative Flag

The Negative Flag N indicates a negative result in an arithmetic or logic operation.

• Bit 1 – Z: Zero Flag

The Zero Flag Z indicates a zero result in an arithmetic or logic operation

• Bit 0 – C: Carry Flag

The Carry Flag C indicates a carry in an arithmetic or logic operation

General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

• One 8-bit output operand and one 8-bit result input

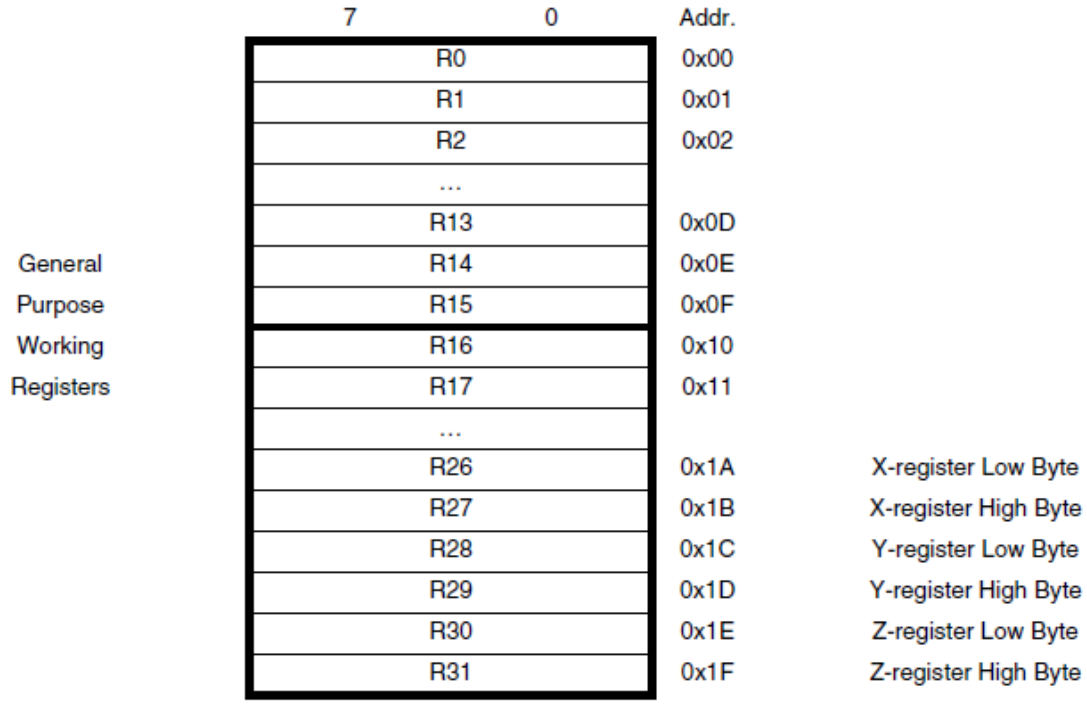
• Two 8-bit output operands and one 8-bit result input

• Two 8-bit output operands and one 16-bit result input

• One 16-bit output operand and one 16-bit result input

Figure shows the structure of the 32 general purpose working registers in the CPU.

AVR CPU General Purpose Working Registers



Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.

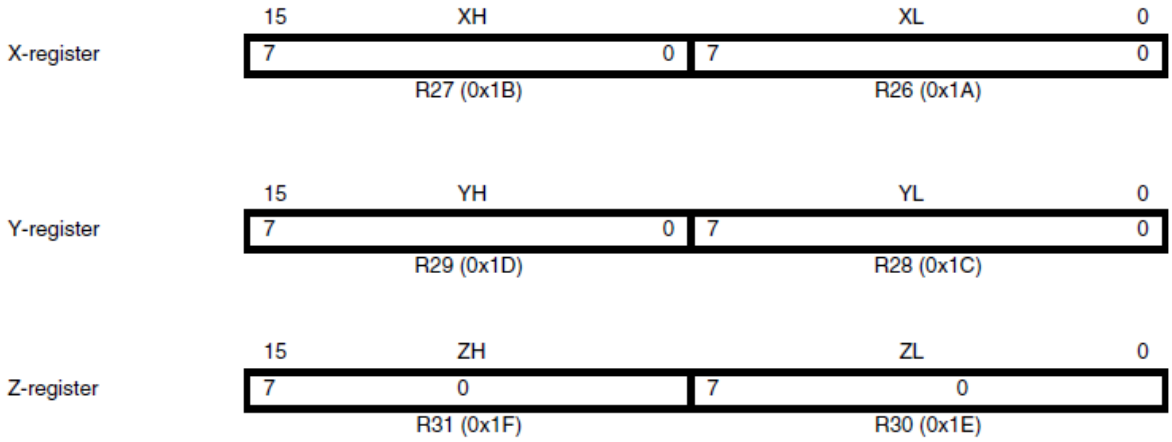
The X-register, Y-register, and Z-register

The registers R26..R31 have some added functions to their general purpose usage.

These registers are 16-bit address pointers for indirect addressing of the Data Space.

The three indirect address registers X, Y, and Z are defined as described in Figure.

The X-, Y-, and Z-registers



In the different addressing modes, these address registers have functions as fixed displacement, automatic increment, and automatic decrement.

Stack Pointer The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt.

The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	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	

Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk-CPU, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

The Parallel Instruction Fetches and Instruction Executions

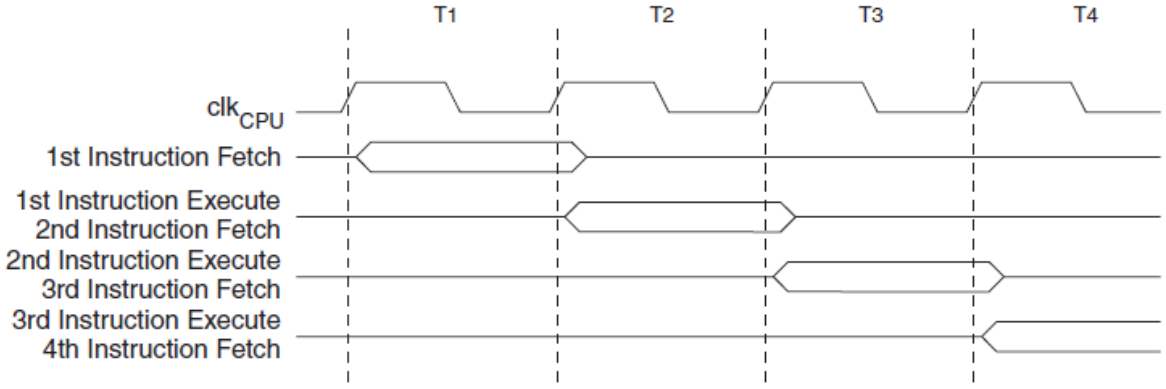


Figure shows the internal timing concept for the Register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Single Cycle ALU Operation

Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate Program Vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status

Register in order to enable the interrupt.

Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security.

The lowest addresses in the program memory space are, by default, defined as the Reset and Interrupt Vectors The list also determines the priority levels of the different interrupts. The lower the address, the higher the priority level is. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR).

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts.

All enabled interrupts can then interrupt the current interrupt routine. The I-bit is Automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be

triggered. When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in   r16, SREG      ; store SREG value
cli                 ; disable interrupts during timed sequence
sbi  EECR, EEMWE    ; start EEPROM write
sbi  EECR, EEWE
out  SREG, r16      ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei     ; set global interrupt enable
sleep   ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
_SEI(); /* set global interrupt enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

Interrupt Response Time The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles, the Program Vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The Vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

AVR ATmega8535 Memories

This section describes the different memories in the ATmega8535. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space.

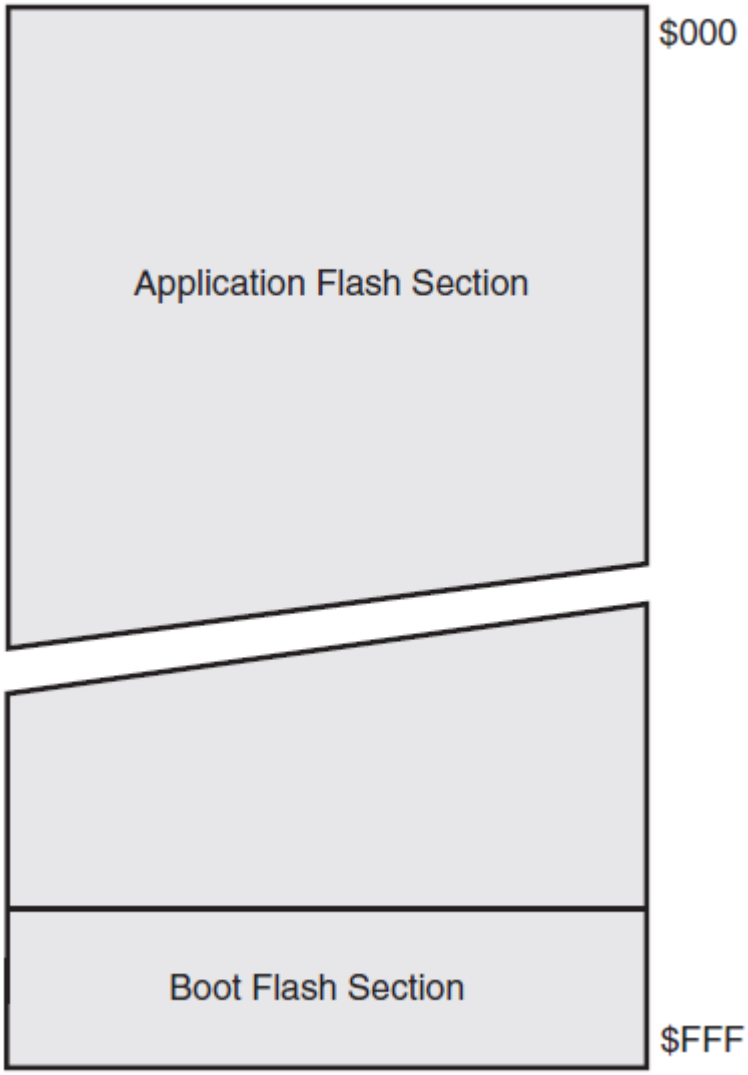
In addition, the ATmega8535 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory

The ATmega8535 contains 8K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 4K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega8535 Program Counter (PC) is 12 bits wide, thus addressing the 4K program memory locations. Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction

description).



Program Memory Map

SRAM Data Memory Figure shows how the ATmega8535 SRAM Memory is organized. The 608 Data Memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File and I/O Memory, and the next 512 locations address the internal data SRAM.

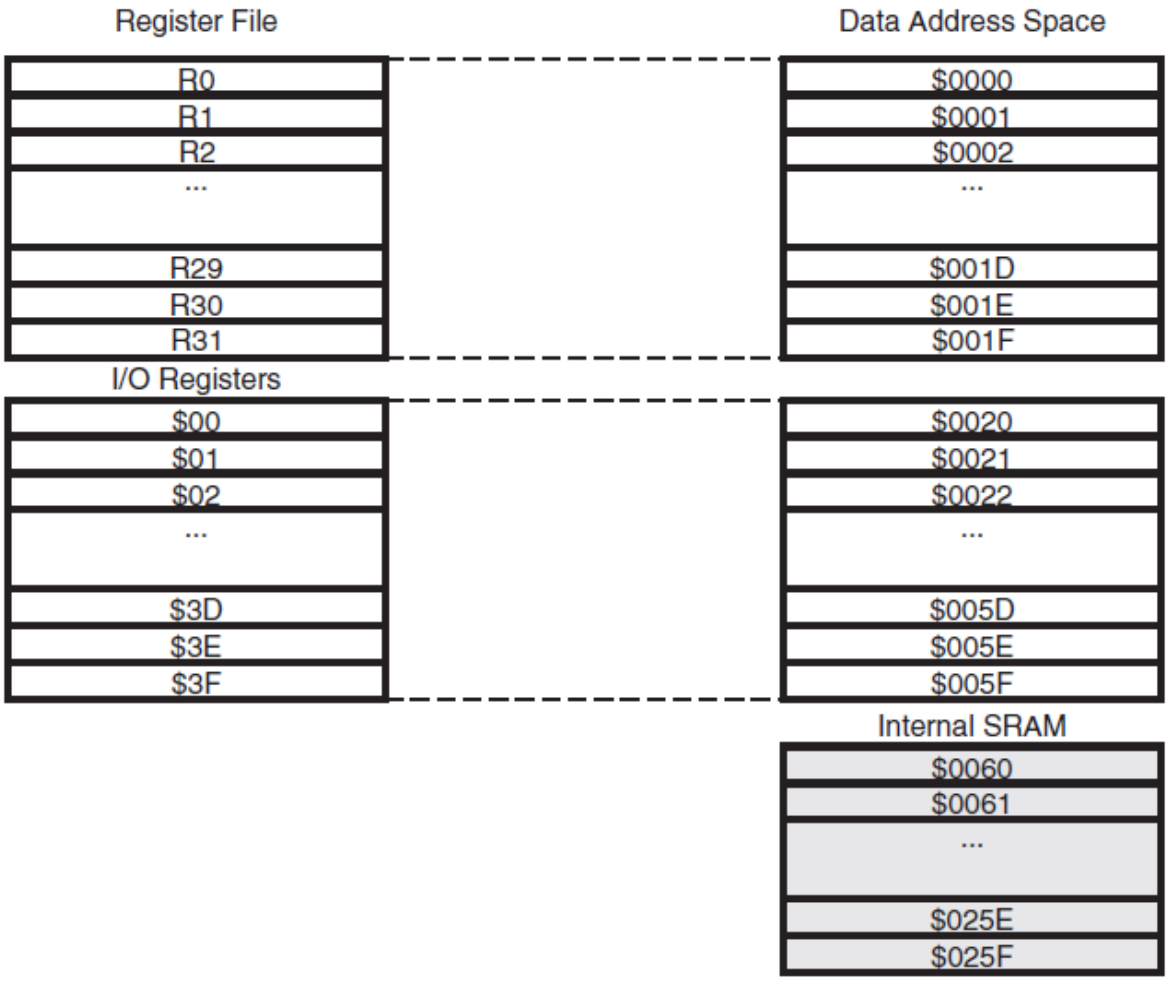
The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers. The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base

address given by the Y- or Z-register.

When using registers indirect addressing modes with automatic pre-decrement and post increment, the address registers X, Y, and Z are decremented or incremented.

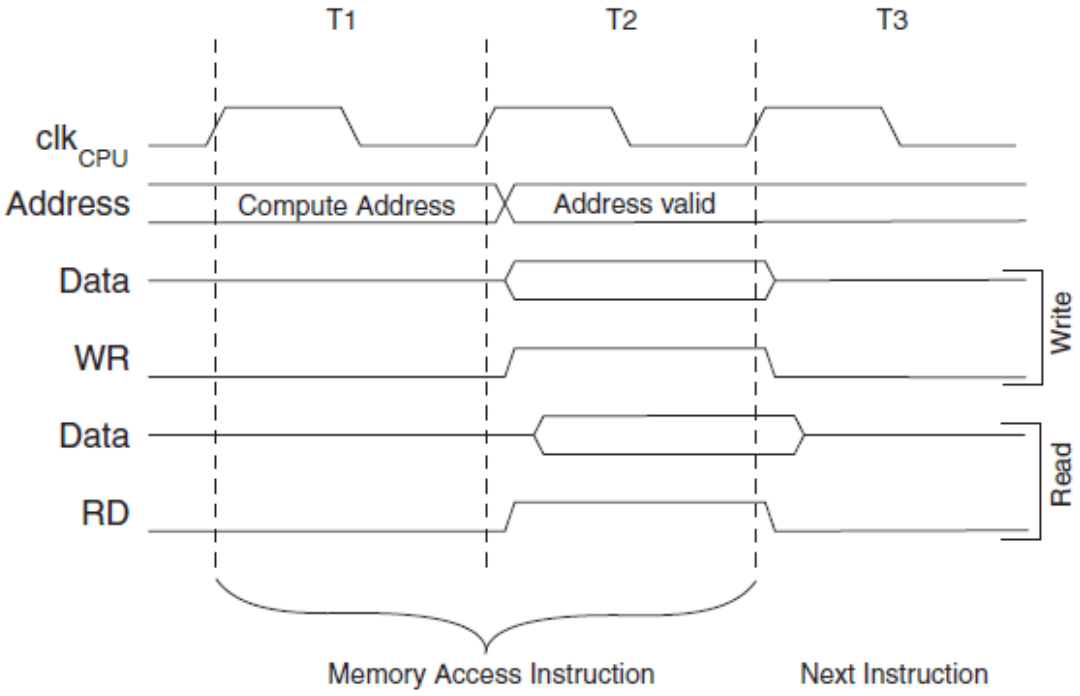
The 32 general purpose working registers, 64 I/O Registers, and the 512 bytes of internal data SRAM in the ATmega8535 are all accessible through all these addressing modes.



Data Memory Access Times This section describes the general access timing concepts for internal memory access.

The internal data SRAM access is performed in two clk-CPU cycles as described in Figure.

On-chip Data SRAM Access Cycles



EEPROM Data Memory The ATmega8535 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

EEPROM Read/Write Access The EEPROM Access Registers are accessible in the I/O space. The write access time for the EEPROM is given in Table. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, VCC is likely to rise or fall slowly on Power-up/down. This causes the device, for some period of time, to run at a voltage lower than specified as minimum for the clock frequency used.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed.

Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next Instruction is executed. When the EEPROM is written, the CPU is halted for two

clock cycles before the next instruction is executed.

The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	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	X	
	X	X	X	X	X	X	X	X	

• Bits 15..9 – Res: Reserved Bits

These bits are reserved bits in the ATmega8535 and will always read as zero.

• Bits 8..0 – EEAR8..0: EEPROM Address

The EEPROM Address Registers – EEARH and EEARL specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

The EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
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	

• Bits 7..0 – EEDR7..0: EEPROM Data

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

The EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

• Bits 7..4 – Res: Reserved Bits

These bits are reserved bits in the ATmega8535 and will always read as zero.

• Bit 3 – EERIE: EEPROM Ready Interrupt Enable

Writing EERIE to one enables the EEPROM Ready Interrupt if the I-bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

• Bit 2 – EEMWE: EEPROM Master Write Enable

The EEMWE bit determines whether setting EEWE to one cause the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

• Bit 1 – EEWE: EEPROM Write Enable

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEWE becomes zero.
2. Wait until SPMEN in SPMCR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

The EEPROM cannot be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never updated by the CPU, step 2 can be omitted.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

• Bit 0 – EERE: EEPROM Read Enable

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 1 lists the typical programming time for EEPROM access from the CPU.

Note: Uses 1 MHz clock, independent of CKSEL Fuse settings.

EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles(1)	Typ Programming Time
EEPROM Write (from CPU)	8448	8.4 ms

Note: Uses 1 MHz clock, independent of CKSEL Fuse settings.

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of

these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_write
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Write data (r16) to Data Register
  out  EEDR,r16
  ; Write logical one to EEMWE
  sbi  EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi  EECR,EEWE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up Address and Data Registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMWE */
  EECR |= (1<<EEMWE);
  /* Start eeprom write by setting EEWE */
  EECR |= (1<<EEWE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_read
  ; Set up address (r18:r17) in Address Register
  out  EEARH, r18
  out  EEARL, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from Data Register
  in   r16,EEDR
  ret
```

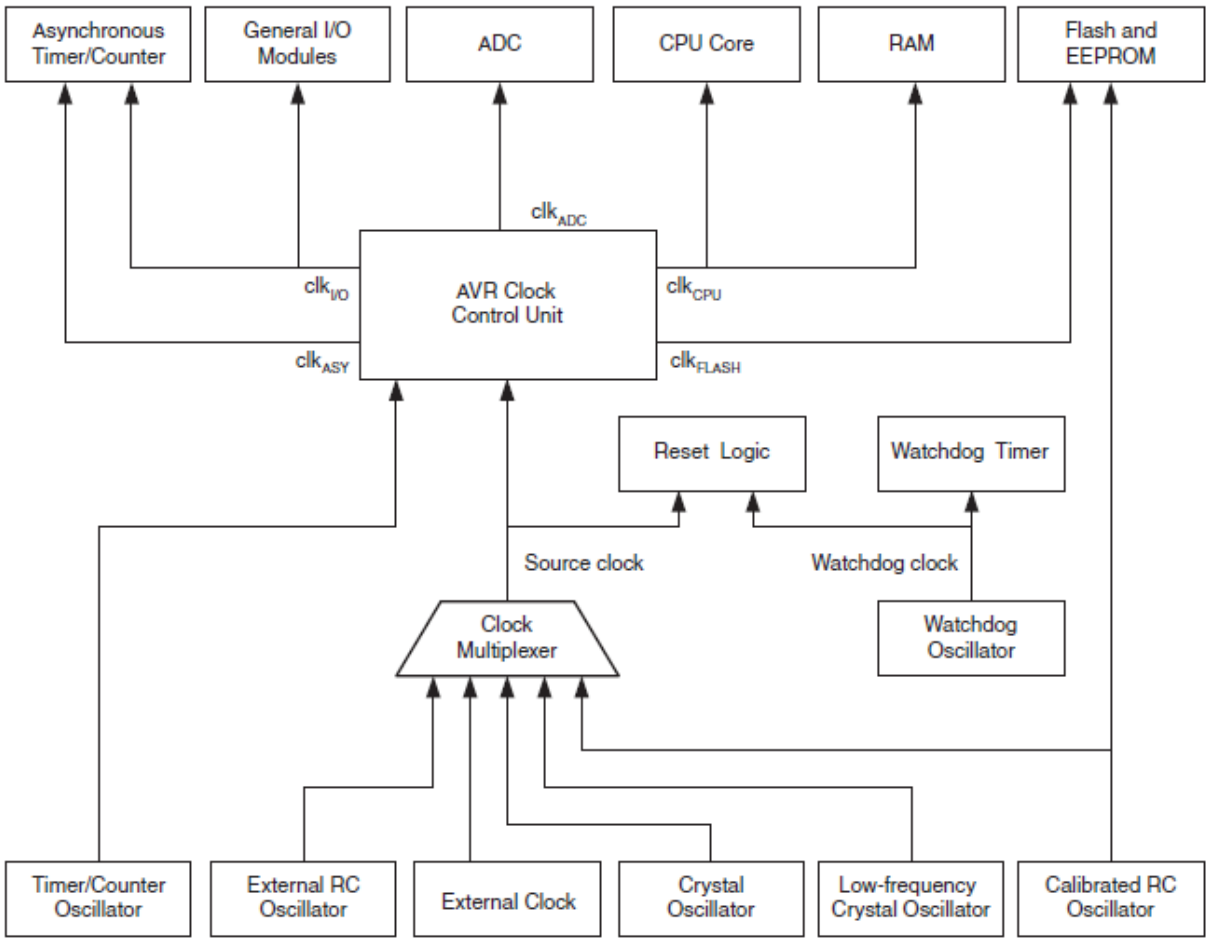
C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up Address Register */
  EEAR = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from Data Register */
  return EEDR;
}
```

System Clock and Clock Options Clock Systems and their Distribution

Figure presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes.

Clock Distribution



CPU Clock – clkCPU The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

I/O Clock – clkI/O The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted. Also note that address recognition in the TWI module is carried out asynchronously when clkI/O is halted, enabling TWI address reception in all sleep modes.

Flash Clock – clkFLASH The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

Asynchronous Timer Clock – clkASY

The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.

ADC Clock – clkADC The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

Clock Sources The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Note:. For all fuses “1” means un-programmed while “0” means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from Reset, there is as an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table.

Default Clock Source The device is shipped with CKSEL = “0001” and SUT = “10”. The default clock source setting is therefore the Internal RC Oscillator with longest startup time. This default setting ensures that all users can make their desired clock source setting using an In- System or Parallel Programmer.

Crystal Oscillator XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different oscillator amplifier modes. When

CKOPT is programmed, the Oscillator output will oscillate will a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is un-programmed, the Oscillator has a smaller output swing. This reduces power consumption considerably.

Device Clocking Options Select(1)

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

Note:. For all fuses “1” means unprogrammed while “0” means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts.

When the CPU starts from Reset, there is as an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table.

Number of Watchdog Oscillator Cycles

Typ Time-out (V_{CC} = 5.0V)	Typ Time-out (V_{CC} = 3.0V)	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

Default Clock Source The device is shipped with CKSEL = “0001” and SUT = “10”. The default clock source setting is therefore the Internal RC Oscillator with longest startup time. This default setting ensures that all users can make their desired clock source setting using an In- System or Parallel Programmer.

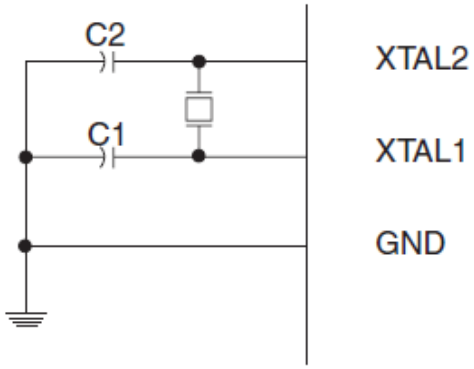
Crystal Oscillator XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as

shown in Figure. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate will a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is un-programmed, the Oscillator has a smaller output swing. This reduces power consumption considerably.

This mode has a limited frequency range and it cannot be used to drive other clock buffers. For resonators, the maximum frequency is 8 MHz with CKOPT un-programmed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment.

Some initial guidelines for choosing capacitors for use with crystals are given in

For ceramic resonators, the capacitor values given by the manufacturer should be used.

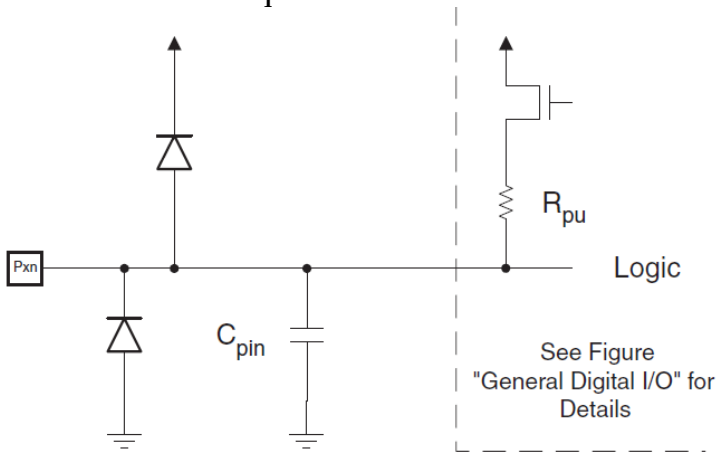


I/O-Ports

Introduction All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays

directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both VCC and Ground as indicated in Figure.

I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in “Register Description for I/O-Ports” Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. In addition, the Pull-up Disable – PUD bit in SFIOR disables the pull-up function for all pins in all ports when set.

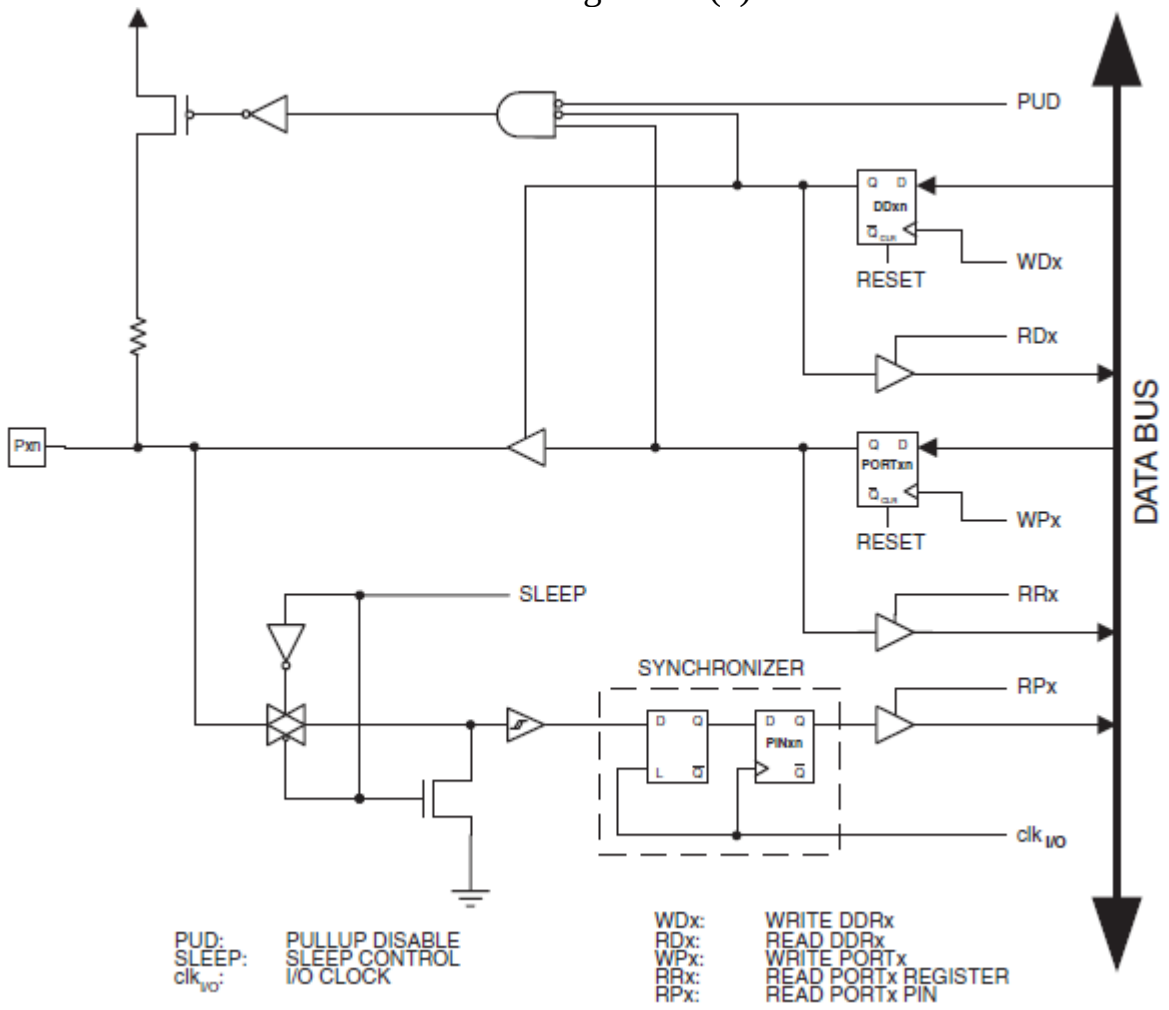
Using the I/O port as General Digital I/O is described in “Ports as General Digital I/O” . Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in “Alternate Port Functions” . Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure shows a functional description of one I/O-port pin, here generically called Pxn.

General Digital I/O(1)



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. clkI/O, SLEEP, and PUD are common to all ports.

Configuring the Pin Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written a logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be

written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

If PORTxn is written a logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written a logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} =0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the SFIOR Register can be set to disable all pull-ups in all ports. Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b10) as an intermediate step. Table summarizes the control signals for the pin value.

Port Pin Configurations

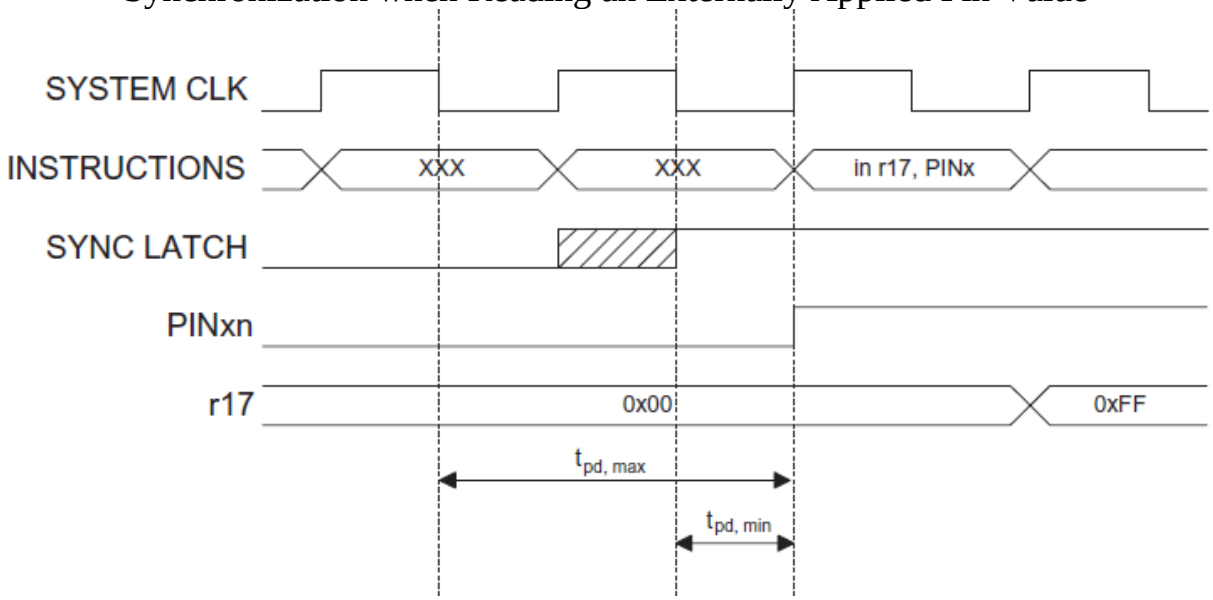
DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Reading the Pin Value Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and

minimum propagation delays are denoted tpd, max and tpd, min respectively.

Consider the clock period starting shortly *after* the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

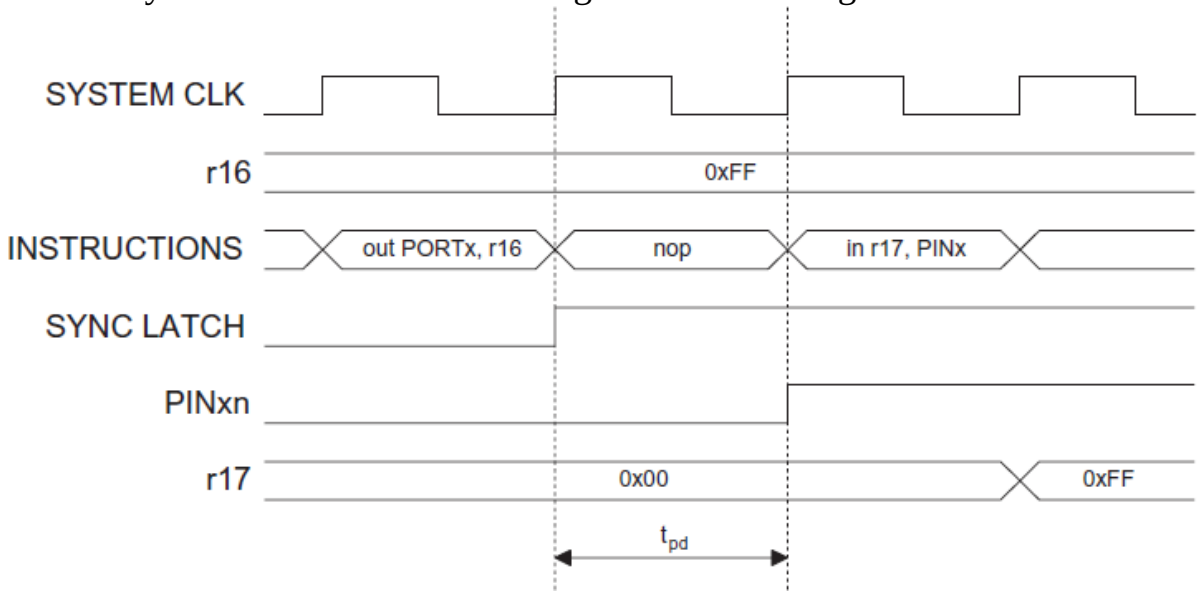
Synchronization when Reading an Externally Applied Pin Value



Consider the clock period starting shortly *after* the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a *nop* instruction must be inserted as indicated in Figure. The *out* instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is one system clock period.

Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a *nop* instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi     r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi     r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out     PORTB,r16
out     DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in      r16,PINB
...
```

C Code Example

```
unsigned char i;
  ...
  /* Define pull-ups and set outputs high */
  /* Define directions for port pins */
  PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
  DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
  /* Insert nop for synchronization*/
  _NOP();
  /* Read port pins */
  i = PINB;
  ...
```

Note: For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bits 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

Analog-to-Digital Converter

Features

- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **±2 LSB Absolute Accuracy**
- **65 - 260 µs Conversion Time**
- **Up to 15 kSPS at Maximum Resolution**
- **8 Multiplexed Single Ended Input Channels**
- **7 Differential Input Channels**
- **2 Differential Input Channels with Optional Gain of 10x and 200x**(1)
- **Optional Left Adjustment for ADC Result Readout**
- **0 - VCC ADC Input Voltage Range**
- **Selectable 2.56V ADC Reference Voltage**
- **Free Running or Single Conversion Mode**
- **ADC Start Conversion by Auto Triggering on Interrupt Sources**
- **Interrupt on ADC Conversion Complete**

• **Sleep Mode Noise Canceller**

Note: The differential input channel is not tested for devices in PDIP and PLCC Package. This feature is only guaranteed to work for devices in TQFP and MLF Packages.

The ATmega8535 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).

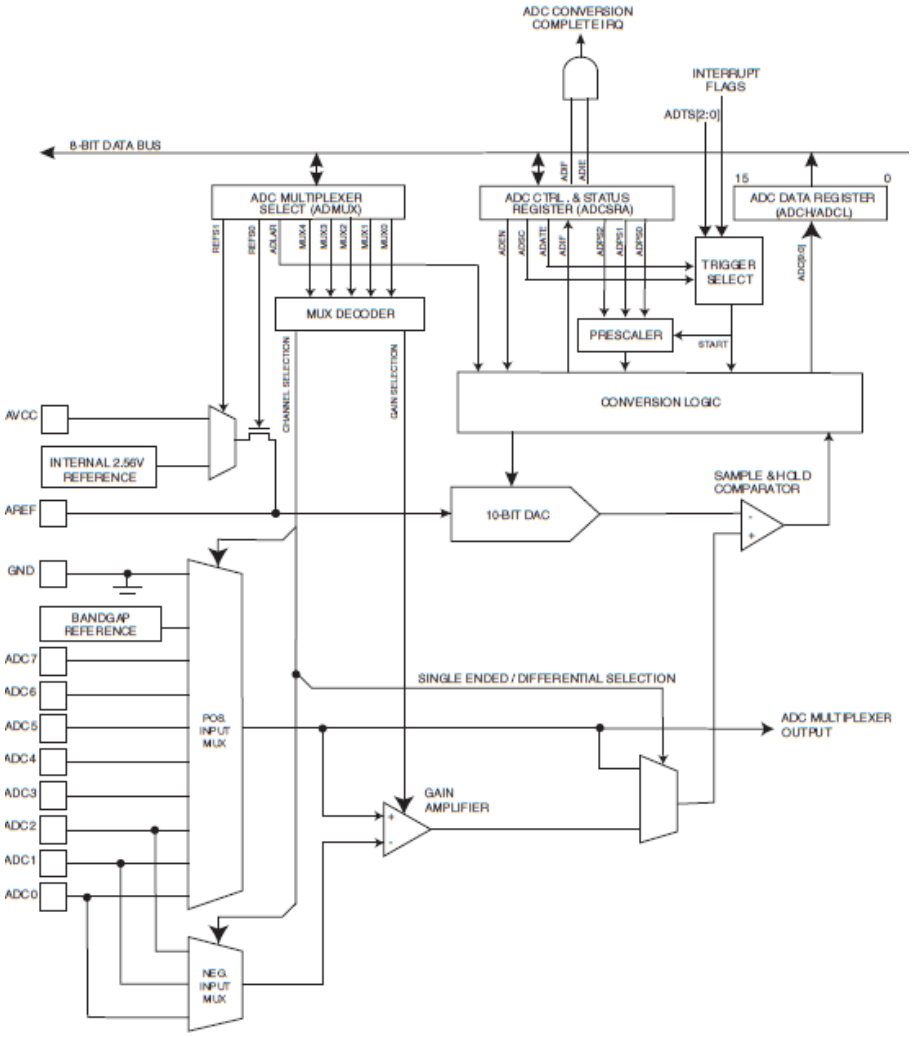
The device also supports 16 differential voltage input combinations. Two of the differential inputs (ADC1, ADC0 and ADC3, ADC2) are equipped with a programmable gain stage, providing amplification steps of 0 dB (1x), 20 dB (10x), or 46 dB (200x) on the differential input voltage before the A/D conversion. Seven differential analog input channels share a common negative terminal (ADC1), while any other ADC input can be selected as the positive input terminal. If 1x or 10x gain is used, 8-bit resolution can be expected. If 200 x gains are used, 7-bit resolution can be expected.

The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion.

The ADC has a separate analog supply voltage pin, AVCC. AVCC must not differ more than ±0.3V from VCC.

Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

Analog-to-Digital Converter Block Schematic



Operation The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AVCC or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel and differential gain are selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. A selection of ADC input pins can be selected as positive and negative inputs to the differential gain amplifier. If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input channel pair by

the selected gain factor. This amplified value then becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

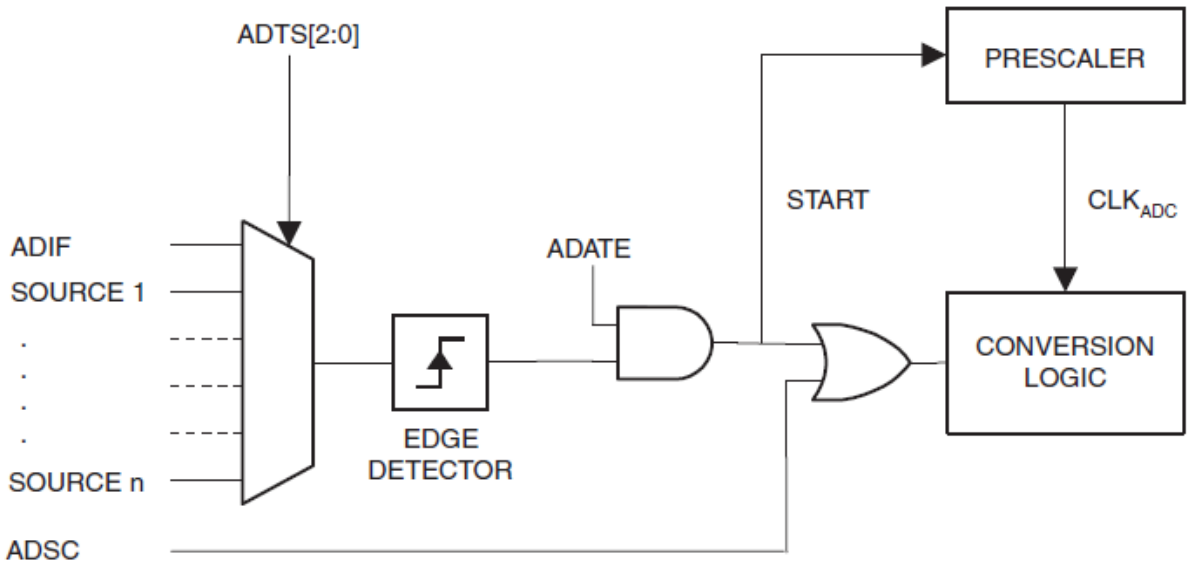
If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, and then ADCH, to ensure that the content of the data registers belongs to the same conversion. Once ADCL is read, ADC access to data registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

Starting a Conversion A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR (See description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC

prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an interrupt flag will be set even if the specific interrupt is disabled or the global interrupt enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the interrupt flag must be cleared in order to trigger a new conversion at the next interrupt event.

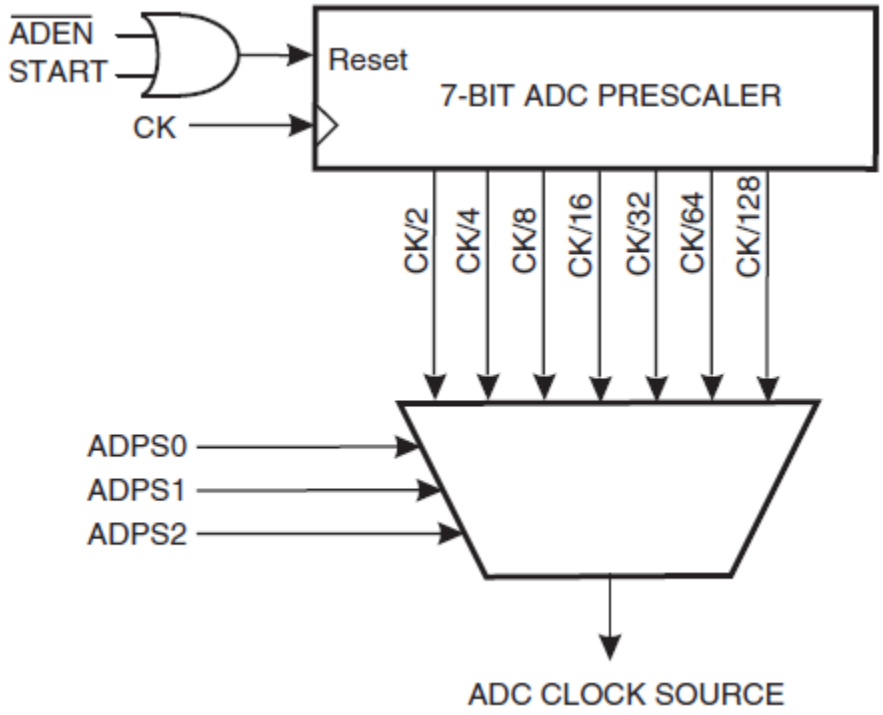


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not.

If Auto Triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

Prescaling and Conversion Timing

ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200 kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

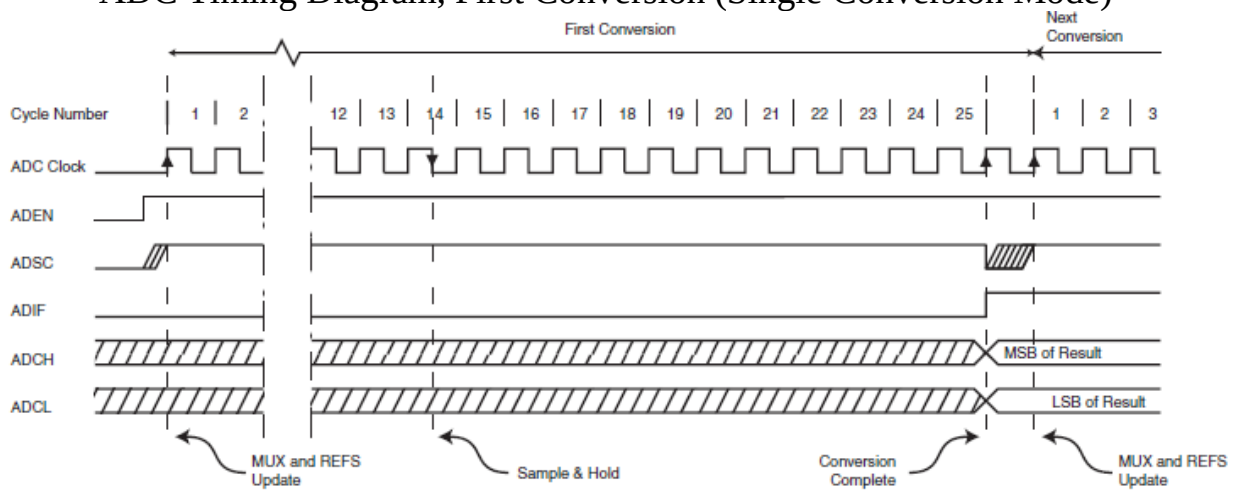
A normal conversion takes 13 ADC clock cycles. The first conversion after the

ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

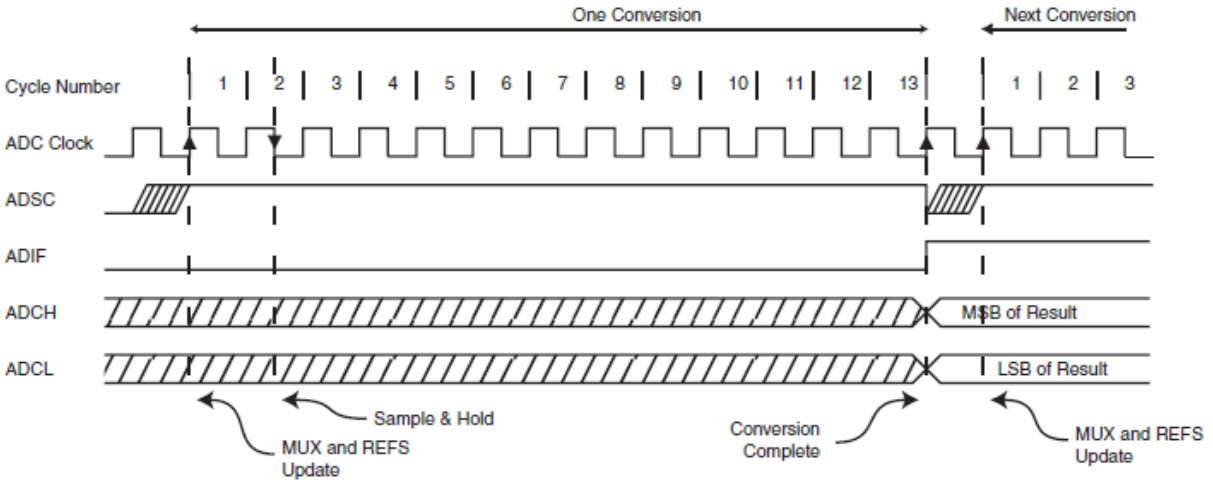
The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an first conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge. When Auto Triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times.

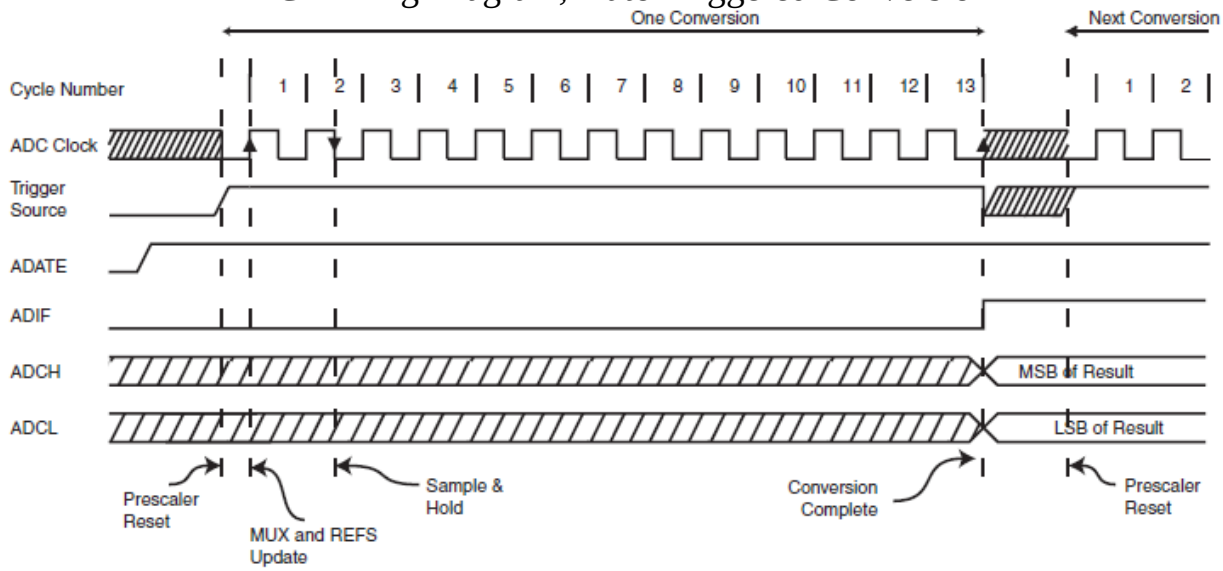
ADC Timing Diagram, First Conversion (Single Conversion Mode)



ADC Timing Diagram, Single Conversion



ADC Timing Diagram, Auto Triggered Conversion



ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	14.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5(1)	13/14(1)

Note: Depending on the state of CKADC2.

Differential Gain Channels When using differential gain channels, certain

aspects of the conversion need to be taken into consideration.

Differential conversions are synchronized to the internal clock CKADC2 equal to half the ADC clock. This synchronization is done automatically by the ADC interface in such a way that the sample-and-hold occurs at a specific phase of CKADC2. A conversion initiated by the user (i.e., all single conversions, and the first free running conversion) when CKADC2 is low will take the same amount of time as a single ended conversion (13 ADC clock cycles from the next prescaled clock cycle). A conversion initiated by the user when CKADC2 is high will take 14 ADC clock cycles due to the synchronization mechanism. In free running mode, a new conversion is initiated immediately after the previous conversion completes and since CKADC2 is high at this time, all automatically started (i.e., all but the first) free running conversions will take 14 ADC clock cycles. The gain stage is optimized for a bandwidth of 4 kHz at all gain settings. Higher frequencies may be subjected to non-linear amplification. An external low-pass filter should be used if the input signal contains higher frequency components than the gain stage bandwidth. Note that the ADC clock frequency is independent of the gain stage bandwidth limitation. For example, the ADC clock period may be 6 μs, allowing a channel to be sampled at 12 kSPS, regardless of the bandwidth of this channel.

If differential gain channels are used and conversions are started by Auto Triggering, the ADC must be switched off between conversions. When Auto Triggering is used, the ADC prescaler is reset before the conversion is started. Since the gain stage is dependent of a stable ADC clock prior to the conversion, this conversion will not be valid. By disabling and then re-enabling the ADC between each conversion (writing ADEN in ADCSRA to “0” then to “1”), only extended conversions are performed. The result from the extended conversions will be valid.

CHAPTER 4

RESULTS & DISCUSSIONS

The proposed model is applicable to many areas. One of the foremost application is that it can be used to detect the burglary events.
Home security.The model is cost effective and reliable in terms of detecting the person/ object in the movement Compared to surveillance cameras, PIR sensors need minimal maintenance. This sensor is reacts to receive stimulation in the form of infrared rays. Essentially every objects emit infrared rays, PIR sensor catches every infrared that emitted, then analyze and make it an input based on frequency of analysis that accepted.

CHAPTER 5

REFERENCE

- Bashar Alathari, Mohammed Falih Kadhim, Salam Al-Khammasi, Nabeel Salih Ali, “A framework implementation of surveillance tracking system based on pir motion sensors ”, Indonesian Journal of Electrical Engineering and Computer Science , Vol. 13, No. 1, January 2019, pp. 235~242
- Sanjana Prasad, P.Mahalakshmi, A.John Clement Sunder,R.Swathi, “Smart Surveillance Monitoring System Using Raspberry PI and PIR Sensor, IJCSIT, Vol. 5 (6) , 2014, 7107-7109
- R Ismail[1], Z Omar[1] and S Suaibu n, “Obstacle-avoiding robot with IR and PIR motion sensors”, IOP Conference Series: Materials Science and Engineering , 52(2016) 012064, pp:16
- C. Tsai, Y. Bai, C. Chu, C. Chung and M. Lin, "PIR-sensor-based lighting device with ultra-low standby power consumption," in IEEE Transactions on Consumer Electronics, vol. 57, no. 3, pp. 1157-1164, August 2011, doi: 10.1109/TCE.2011.6018869.
- K. N. Ha, K. C. Lee and S. Lee, "Development of PIR sensor based indoor location detection system for smart home," 2006 SICE-ICASE International Joint Conference, Busan, 2006, pp. 2162-2167, doi: 10.1109/SICE.2006.315642.