BABY PRESENCE DETECTOR & WARNING SYSTEM IN CAR

**AIM:**

The main aim the project is to design “A CHILDLEFT BEHIND ALERTING SYSTEM BASED ON CAPACITIVE SENSING PRINCIPLE”.

**ABSTRACT**

The paper presents an efficient technique to protect a child when it is forgotten in car unknowingly by parents. There are several such cases reported worldwide and recently the survey by nonprofit organization called kids-and-cars says that around 30 children died in U.S. only due to heatstroke when they are left unattended in car. It shows severity of problem and need to develop a system for the same. Here we have developed a system that will sense the presence of child and will generate an alarm. If no one attends the child, the system will make a call to parents and emergency person.

**EXISTING METHOD**:

Once a car is turned-off and parked, keeping its window glass closed, the temperature inside the car increases rapidly even on a day with atmospheric temperature of about 21degree centigrade. As the thermoregulatory system of the child’s not well developed, this condition may lead to heatstroke which can be fatal. As we know, the child entire depends on elders but, unknowingly, in a busy schedule, the driver or passengers may forget to take the child in the front seat, usually kept in a back seat of the car. Such incidents can be prevented by sensing the presence of a child soon after a car is turned-off & the generating/sending a suitable warning signal to the driver or parents.

**PROPOSED METHOD:**

Here we proposed a simple and compact capacitive sensor that can be placed in front seat to detect presence of a child. The proposed system also has a vehicle ignition monitor to confirm presence of driver inside a car. It also has a temperature inside the car. A GSM modem is used to alert driver/parents/guardians as soon as a child left in the car in front seat is detected and the car is found to be turned-off. Principle of operation of the capacitive sensor, measurement scheme employed details of the prototype sensor and warning system (through SMS or Voice announcement) developed and tested.

**BLOCK DIAGRAM:**

**POWER SUPPLY**

**LCD DISPLAY**

**(16\*2 LINES)**

**MICRO CONTROLLER**

**(8051)**

**GSM**

**PIR SENSOR**

**IGNITION SWITCH**

**IR SENSOR**

**SOFTWARE :** Embedded ‘C’

**TOOLS :** Keil IDE, ISP

**TARGET DEVICE :** 8051

**APPLICATIONS :** Industrial areas, Coal mine environments, Leakage environments.

**ADVANTAGES :** low cost, long distance networks

CHAPTER 2

INTRODUCTION TO EMBEDDED SYSTEMS

Many embedded systems have substantially different design constraints than desktop computing applications. No single characterization applies to the diverse spectrum of embedded systems. However, some combination of cost pressure, long life-cycle, real¬-time requirements, reliability requirements, and design culture dysfunction can make it difficult to be successful applying traditional computer design methodologies and tools to embedded applications. Embedded systems in many cases must be optimized for life-cycle and business-driven factors rather than for maximum computing throughput. There is currently little tool support for expanding embedded computer design to the scope of holistic embedded system design. However, knowing the strengths and weaknesses of current approaches can set expectations appropriately, identify risk areas to tool adopters, and suggest ways in which tool builders can meet industrial needs. If we look around us, today we see numerous appliances which we use daily, be it our refrigerator, the microwave oven, cars, PDAs etc. Most appliances today are powered by something beneath the sheath that makes them do what they do. These are tiny microprocessors, which respond to various keystrokes or inputs. These tiny microprocessors, working on basic assembly languages, are the heart of the appliances. We call them embedded systems. Of all the semiconductor industries, the embedded systems market place is the most conservative, and engineering decisions here usually lean towards established, low risk solutions. Welcome to the world of embedded systems, of computers that will not look like computers and won’t function like anything we are familiar with.

2.1 CLASSIFICATION

Embedded systems are divided into autonomous, realtime, networked & mobile categories.

Autonomous systems

They function in standalone mode. Many embedded systems used for process control in manufacturing units& automobiles fall under this category.

Real-time embedded systems

These are required to carry out specific tasks in a specified amount of time. These systems are extensively used to carry out time critical tasks in process control.

Networked embedded systems

They monitor plant parameters such as temperature, pressure and humidity and send the data over the network to a centralized system for on line monitoring.

Mobile gadgets

Mobile gadgets need to store databases locally in their memory. These gadgets imbibe powerful computing & communication capabilities to perform realtime as well as nonrealtime tasks and handle multimedia applications. The embedded system is a combination of computer hardware, software, firmware and perhaps additional mechanical parts, designed to perform a specific function. A good example is an automatic washing machine or a microwave oven. Such a system is in direct contrast to a personal computer, which is not designed to do only a specific task. But an embedded system is designed to do a specific task with in a given timeframe, repeatedly, endlessly, with or without human interaction.

Hardware

Good software design in embedded systems stems from a good understanding of the hardware behind it. All embedded systems need a microprocessor, and the kinds of microprocessors used in them are quite varied. A list of some of the common microprocessors families are: ARM family, The Zilog Z8 family, Intel 8051/X86 family, Motorola 68K family and the power PC family. For processing of information and execution of programs, embedded system incorporates microprocessor or micro- controller. In an embedded system the microprocessor is a part of final product and is not available for reprogramming to the end user. An embedded system also needs memory for two purposes, to store its program and to store its data. Unlike normal desktops in which data and programs are stored at the same place, embedded systems store data and programs in different memories. This is simply because the embedded system does not have a hard drive and the program must be stored in memory even when the power is turned off. This type of memory is called ROM. Embedded applications commonly employ a special type of ROM that can be programmed or reprogrammed with the help of special devices.

2.2 OTHER COMMON PARTS FOUND ON MANY EMBEDDED SYSTEMS

• UART& RS232

• PLD

• ASIC’s& FPGA’s

• Watch dog timer etc.

2.3 DESIGN PROCESS

Embedded system design is a quantitative job. The pillars of the system design methodology are the separation between function and architecture, is an essential step from conception to implementation. In recent past, the search and industrial community has paid significant attention to the topic of hardware-software (HW/SW) codesign and has tackled the problem of coordinating the design of the parts to be implemented as software and the parts to be implemented as hardware avoiding the HW/SW integration problem marred the electronics system industry so long. In any large scale embedded systems design methodology, concurrency must be considered as a first class citizen at all levels of abstraction and in both hardware and software. Formal models & transformations in system design are used so that verification and synthesis can be applied to advantage in the design methodology. Simulation tools are used for exploring the design space for validating the functional and timing behaviors of embedded systems. Hardware can be simulated at different levels such as electrical circuits, logic gates, RTL e.t.c. using VHDL description. In some environments software development tools can be coupled with hardware simulators, while in others the software is executed on the simulated hardware. The later approach is feasible only for small parts of embedded systems. Design of an embedded system using Intel’s 80C188EB chip is shown in the figure. Inorder to reduce complexity, the design process is divided in four major steps: specification, system synthesis, implementation synthesis and performance evaluation of the prototype.

2.3.1 SPECIFICATION

During this part of the design process, the informal requirements of the analysis are transformed to formal specification using SDL.

2.3.2 SYSTEM-SYNTHESIS

For performing an automatic HW/SW partitioning, the system synthesis step translates the SDL specification to an internal system model switch contains problem graph& architecture graph. After system synthesis, the resulting system model is translated back to SDL.

2.3.3 IMPLEMENTATION-SYNTHESIS

SDL specification is then translated into conventional implementation languages such as VHDL for hardware modules and C for software parts of the system.

2.3.4 PROTOTYPING

On a prototyping platform, the implementation of the system under development is executed with the software parts running on multiprocessor unit and the hardware part running on a FPGA board known as phoenix, prototype hardware for Embedded Network Interconnect Accelerators.

2.3.5 APPLICATIONS

Embedded systems are finding their way into robotic toys and electronic pets, intelligent cars and remote controllable home appliances. All the major toy makers across the world have been coming out with advanced interactive toys that can become our friends for life. ‘Furby’ and ‘AIBO’ are good examples at this kind. Furbies have a distinct life cycle just like human beings, starting from being a baby and growing to an adult one. In AIBO first two letters stands for Artificial Intelligence. Next two letters represents robot. The AIBO is robotic dog. Embedded systems in cars also known as Telematic Systems are used to provide navigational security communication & entertinment services using GPS, satellite. Home appliances are going the embedded way. LG electronics digital DIOS refrigerator can be used for surfing the net, checking e-mail, making video phone calls and watching TV.IBM is developing an air conditioner that we can control over the net. Embedded systems cover such a broad range of products that generalization is difficult. Here are some broad categories.

Fig 2.1: Embedded Development Life Cycle

• Aerospace and defence electronics: Fire control, radar, robotics/sensors, sonar.

• Automotive: Autobody electronics, auto power train, auto safety, car information systems.

• Broadcast & entertainment: Analog and digital sound products, camaras, DVDs, Set top boxes, virtual reality systems, graphic products.

• Consumer/internet appliances: Business handheld computers, business network computers/terminals, electronic books, internet smart handheld devices, PDAs.

• Data communications: Analog modems, ATM switches, cable modems, XDSL modems, Ethernet switches, concentrators.

• Digital imaging: Copiers, digital still cameras, Fax machines, printers, scanners.

• Industrial measurement and control: Hydro electric utility research & management traffic management systems, train marine vessel management systems.

• Medical electronics: Diagnostic devices, real time medical imaging systems, surgical devices, critical care systems.

• Server I/O: Embedded servers, enterprise PC servers, PCI LAN/NIC controllers, RAID devices, SCSI devices.

• Telecommunications: ATM communication products, base stations, networking switches, SONET/SDH cross connect, multiplexer.

• Mobile data infrastructures: Mobile data terminals, pagers, VSATs, Wireless LANs, Wireless phones.

CHAPTER 3

ARUDINO:

The Arduino is a family of microcontroller boards to simplify electronic design, prototyping and experimenting for artists, hackers, hobbyists, but also many professionals. People use it as brains for their robots, to build new digital music instruments, or to build a system that lets your house plants tweet you when they’re dry. Arduinos (we use the standard Arduino Uno) are built around an ATmega microcontroller — essentially a complete computer with CPU, RAM, Flash memory, and input/output pins, all on a single chip. Unlike, say, a Raspberry Pi, it’s designed to attach all kinds of sensors, LEDs, small motors and speakers, servos, etc. directly to these pins, which can read in or output digital or analog voltages between 0 and 5 volts. The Arduino connects to your computer via USB, where you program it in a simple language (C/C++, similar to Java) from inside the free Arduino IDE by uploading your compiled code to the board. Once programmed, the Arduino can run with the USB link back to your computer, or stand-alone without it — no keyboard or screen needed, just power.

Figure 2.2 Structure of Arduino Board

Looking at the board from the top down, this is an outline of what you will see (parts of the board you might interact with in the course of normal use are highlighted)

Figure 2.3 Arduino Board

Starting clockwise from the top center:

 Analog Reference pin (orange)

 Digital Ground (light green)

 Digital Pins 2-13 (green)

 Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (Digital Read and Digital Write) if you are also using serial communication (e.g. Serial.begin).

 Reset Button - S1 (dark blue)

 In-circuit Serial Programmer (blue-green)

 Analog In Pins 0-5 (light blue)

 Power and Ground Pins (power: orange, grounds: light orange)

 External Power Supply In (9-12VDC) - X1 (pink)

 Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)

 USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

DIGITAL PINS

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pin Mode(), Digital Read(), and Digital Write() commands. Each pin has an internal pull-up resistor which can be turned on and off using digital Write() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40mA.

 Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).

 External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach Interrupt() function for details.

 PWM: 3, 5, 6, 9, 10, and 11 Provide 8-bit PWM output with the analog Write() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.

 BT Reset: 7. (Arduino BT-only) Connected to the reset line of the bluetooth module.

 SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

 LED: 13. On the Diecimila and LilyPad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

ANALOG PINS

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analog Read() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

 I2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

POWER PINS

 VIN (sometimes labeled "9V"): The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Also note that the Lily Pad has no VIN pin and accepts only a regulated input.

 5V: The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

 3V3 (Diecimila-only) : A 3.3 volt supply generated by the on-board FTDI chip.

 GND: Ground pins.

OTHER PINS

 AREF: Reference voltage for the analog inputs. Used with analog Reference().

 Reset: (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

ATMEGA328

Pin diagram

Figure 2.4 Pin Configuration of Atmega328

Pin Description

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 B also serves the functions of various special features of the ATmega32.

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. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. The TD0 pin is tri-stated unless TAP states that shift out data are entered. Port C also serves the functions of the JTAG interface.

Port D (PD7-PD0):

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega32.

Reset (Reset Input):

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

FEATURES

 1.8-5.5V operating range

 Up to 20MHz

 Part: ATMEGA328P-AU

 32kB Flash program memory

 1kB EEPROM

 2kB Internal SRAM

 2 8-bit Timer/Counters

 16-bit Timer/Counter

 RTC with separate oscillator

 6 PWM Channels

 8 Channel 10-bit ADC

 Serial USART

 Master/Slave SPI interface

 2-wire (I2C) interface

 Watchdog timer

 Analog comparator

 23 IO lines

 Data retention: 20 years at 85C/ 100 years at 25C

 Digital I/O Pins are 14 (out of which 6 provide PWM output)

 Analog Input Pins are 6.

 DC Current per I/O is 40 mA

 DC Current for 3.3V Pin is 50mA

AVR CPU CORE

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

Figure 2.5 Block Diagram

OVERVIEW

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.

Figure AVR core architecture

In order to maximize performance and parallelism, the AVR uses a 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 Reprogrammable 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 address 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-register, 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.

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:

Figure 2.7 AVR status register

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 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 I-bit 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 The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

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. See the “Instruction Set Description” for detailed information.

Bit 3 – V: Two’s Complement Overflow Flag

The Two’s Complement Overflow Flag V supports two’s complement arithmetic.

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.

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. Note that the Stack is implemented as growing from higher to lower memorylocations. The Stack Pointer Register always points to the top of the Stack. The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer.

The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM

Table 2.1 Stack Pointer instructions

The AVR ATmega128A 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.SPH and SPL - Stack Pointer High and Low Register.

Figure 2.10 SPH and SPL - Stack Pointer High and Low Register

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.

AVR Memories

This section describes the different memories in the ATmega328. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, theATmega328 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory:

The ATmega328 contains 4/8/16/32Kbytes 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 2/4/8/16K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Loader Section and Application Program Section. The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega328 Program Counter (PC) is 11/12/13/14 bits wide, thus addressing the 2/4/8/16K program memory locations.

SRAM Data Memory:

ATmega328 is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The lower 768/1280/1280/2303 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 512/1024/1024/2048 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 register 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, 160 Extended I/O Registers, and the 512/1024/1024/2048 bytes of internal data SRAM in the ATmega328 are all accessible through all these addressing modes.

Figure 2.11 Data Memory Map

Interrupts

This section describes the specifics of the interrupt handling as performed in the Atmega328. In Atmega328Each Interrupt Vector occupies two instruction words and the Reset Vector is affected by the BOOTRST fuse, and the Interrupt Vector start address is affected by the IVSEL bit in MCUCR.

Table 2.2 Reset and Interrupt Vectors in ATMEGA 328 and ATMEGA 328P

Vector No. Program Address Source Interrupt Definition

1 0x0000 RESET External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset

2 0x0002 INT0 External Interrupt Request 0

3 0x0004 INT1 External Interrupt Request 0

4 0x0006 PCINTO Pin Change Interrupt Request 0

5 0x0008 PCINT1 Pin Change Interrupt Request 1

6 0x000A PCINT2 Pin Change Interrupt Request 2

7 0x000C WDT Watchdog Time-out Interrupt

8 0x000E TIMER2 COMPA Timer/Counter2 Compare Match A

9 0x0010 TIMER2 COMPB Timer/Counter2 Compare Match B

10 0x0012 TIMER2 OVF Timer/Counter 2 Overflow

11 0x0014 TIMER1 CAPT Timer/Counter 2 Capture Event

12 0x0016 TIMER1 COMPA Timer/Counter1 Compare Match A

13 0x0018 TIMER1 COMPB Timer/Counter1 Compare Match B

14 0x001A TIMER 1 OVF Timer/Counter1 Overflow

15 0x001C TIMER0 COMPA Timer/Counter0 Compare Match A

16 0x001E TIMER0 COMPB Timer/Counter0 Compare Match B

17 0x0020 TIME0 OVF Timer/Counter0 Overflow

18 0x0022 SPI, STC SPI Serial Transfer Complete

19 0x0024 USART, RX USART RX Complete

20 0x0026 USART, UDRE USART, Data Register Empty

21 0x0028 USART, TX USART, TX Complete

22 0x002A ADC ADC Conversion Complete

23 0x002C EE READY EEPROM Ready

24 0x002E ANALOG COMP Analog Comparator

25 0x0030 TWI 2-wire Serial Interface

26 0x0032 SPM READY Store Program Memory Ready

When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.Table below shows reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 2.3 Reset and Interrupt Vectors Placement in ATmega328 and ATmega328P

Arduino with ATmega328

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to versionR2) programmed as a USB-to-serial converter.

 Pin out: Added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino. Due that operates with 3.3V. The second one is a not connected pin that is reserved for future purposes.

 Stronger RESET circuit.

 Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Arduino Characteristics

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

 VIN: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

 GND. Ground pins.

 IOREF. This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Memory:

The ATmega328 has 32 KB (with 0.5 KB used for the boot loader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Serial Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus. For SPI communication, use the SPI library.

CHAPTER 4

HARDWARE COMPONENTS

LCD (Liquid Cristal Display)

Introduction:

A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. Each pixel consists of a column of liquid crystal molecules suspended between two transparent electrodes, and two polarizing filters, the axes of polarity of which are perpendicular to each other. Without the liquid crystals between them, light passing through one would be blocked by the other. The liquid crystal twists the polarization of light entering one filter to allow it to pass through the other.

A program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an controller is an LCD display. Some of the most common LCDs connected to the contollers are 16X1, 16x2 and 20x2 displays. This means 16 characters per line by 1 line 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Shapes and S

available. Line lengths of 8, 16, 20, 24, 32 and 40 characters are all standard, in one, two

Many microcontroller devices use 'smart LCD' displays to output visual information. LCD displays designed around LCD NT-C1611 module, are inexpensive, easy to use, and it is even possible to produce a readout using the 5X7 dots plus cursor of the display. They have a standard ASCII set of characters and mathematical symbols. For an 8-bit data bus, the display requires a +5V supply plus 10 I/O lines (RS RW D7 D6 D5 D4 D3 D2 D1 D0). For a 4-bit data bus it only requires the supply lines plus 6 extra lines(RS RW D7 D6 D5 D4). When the LCD display is not enabled, data lines are tri-state and they do not interfere with the operation of the microcontroller.

Features:

(1) Interface with either 4-bit or 8-bit microprocessor.

(2) Display data RAM

(3) 80x8 bits (80 characters).

(4) Character generator ROM

(5). 160 different 5 7 dot-matrix character patterns.

(6). Character generator RAM

(7) 8 different user programmed 5 7 dot-matrix patterns.

(8).Display data RAM and character generator RAM may be

Accessed by the microprocessor.

(9) Numerous instructions

(10) .Clear Display, Cursor Home, Display ON/OFF, Cursor ON/OFF,

Blink Character, Cursor Shift, Display Shift.

(11). Built-in reset circuit is triggered at power ON.

(12). Built-in oscillator.

Data can be placed at any location on the LCD. For 16×1 LCD, the address locations are:

Fig : Address locations for a 1x16 line LCD

Shapes and sizes:

Even limited to character based modules,there is still a wide variety of shapes and sizes available. Line lenghs of 8,16,20,24,32 and 40 charecters are all standard, in one, two and four line versions.

Several different LC technologies exists. “supertwist” types, for example, offer Improved contrast and viewing angle over the older “twisted nematic” types. Some modules are available with back lighting, so so that they can be viewed in dimly-lit conditions. The back lighting may be either “electro-luminescent”, requiring a high voltage inverter circuit, or simple LED illumination.

Electrical blockdiagram:

Power supply for lcd driving:

PIN DESCRIPTION:

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

Fig: pin diagram of 1x16 lines lcd

CONTROL LINES:

EN:

Line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

RS:

Line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which sould be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

RW:

Line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands, so RW will almost always be low.

Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

Logic status on control lines:

• E - 0 Access to LCD disabled

- 1 Access to LCD enabled

• R/W - 0 Writing data to LCD

- 1 Reading data from LCD

• RS - 0 Instructions

- 1 Character

Writing data to the LCD:

1) Set R/W bit to low

2) Set RS bit to logic 0 or 1 (instruction or character)

3) Set data to data lines (if it is writing)

4) Set E line to high

5) Set E line to low

Read data from data lines (if it is reading)on LCD:

1) Set R/W bit to high

2) Set RS bit to logic 0 or 1 (instruction or character)

3) Set data to data lines (if it is writing)

4) Set E line to high

5) Set E line to low

Entering Text:

First, a little tip: it is manually a lot easier to enter characters and commands in hexadecimal rather than binary (although, of course, you will need to translate commands from binary couple of sub-miniature hexadecimal rotary switches is a simple matter, although a little bit into hex so that you know which bits you are setting). Replacing the d.i.l. switch pack with a of re-wiring is necessary.

The switches must be the type where On = 0, so that when they are turned to the zero position, all four outputs are shorted to the common pin, and in position “F”, all four outputs are open circuit.

All the available characters that are built into the module are shown in Table 3. Studying the table, you will see that codes associated with the characters are quoted in binary and hexadecimal, most significant bits (“left-hand” four bits) across the top, and least significant bits (“right-hand” four bits) down the left.

Most of the characters conform to the ASCII standard, although the Japanese and Greek characters (and a few other things) are obvious exceptions. Since these intelligent modules were designed in the “Land of the Rising Sun,” it seems only fair that their Katakana phonetic symbols should also be incorporated. The more extensive Kanji character set, which the Japanese share with the Chinese, consisting of several thousand different characters, is not included!

Using the switches, of whatever type, and referring to Table 3, enter a few characters onto the display, both letters and numbers. The RS switch (S10) must be “up” (logic 1) when sending the characters, and switch E (S9) must be pressed for each of them. Thus the operational order is: set RS high, enter character, trigger E, leave RS high, enter another character, trigger E, and so on.

The first 16 codes in Table 3, 00000000 to 00001111, ($00 to $0F) refer to the CGRAM. This is the Character Generator RAM (random access memory), which can be used to hold user-defined graphics characters. This is where these modules really start to show their potential, offering such capabilities as bar graphs, flashing symbols, even animated characters. Before the user-defined characters are set up, these codes will just bring up strange looking symbols.

Codes 00010000 to 00011111 ($10 to $1F) are not used and just display blank characters. ASCII codes “proper” start at 00100000 ($20) and end with 01111111 ($7F). Codes 10000000 to 10011111 ($80 to $9F) are not used, and 10100000 to 11011111 ($A0 to $DF) are the Japanese characters.

Initialization by Instructions:

If the power conditions for the normal operation of the internal reset circuit are not satisfied, then executing a series of instructions must initialize LCD unit. The procedure for this initialization process is as above show.

REGULATED POWER SUPPLY:

Introduction:

Power supply is a supply of electrical power. A device or system that supplies electrical or other types of energy to an output load or group of loads is called a power supply unit or PSU. The term is most commonly applied to electrical energy supplies, less often to mechanical ones, and rarely to others.

A power supply may include a power distribution system as well as primary or secondary sources of energy such as

• Conversion of one form of electrical power to another desired form and voltage, typically involving converting AC line voltage to a well-regulated lower-voltage DC for electronic devices. Low voltage, low power DC power supply units are commonly integrated with the devices they supply, such as computers and household electronics.

• Batteries.

• Chemical fuel cells and other forms of energy storage systems.

• Solar power.

• Generators or alternators.

Block Diagram:

Fig .Regulated Power Supply

The basic circuit diagram of a regulated power supply (DC O/P) with led connected as load is shown in fig:

Fig Circuit diagram of Regulated Power Supply with Led connection

The components mainly used in above figure are

• 230V AC MAINS

• TRANSFORMER

• BRIDGE RECTIFIER(DIODES)

• CAPACITOR

• VOLTAGE REGULATOR(IC 7805)

• RESISTOR

• LED(LIGHT EMITTING DIODE)

The detailed explanation of each and every component mentioned above is as follows:

Step 1: Transformation: The process of transforming energy from one device to another is called transformation. For transforming energy we use transformers.

Transformers:

A transformer is a device that transfers electrical energy from one circuit to another through inductively coupled conductors without changing its frequency. A varying current in the first or primary winding creates a varying magnetic flux in the transformer's core, and thus a varying magnetic field through the secondary winding. This varying magnetic field induces a varying electromotive force (EMF) or "voltage" in the secondary winding. This effect is called mutual induction.

If a load is connected to the secondary, an electric current will flow in the secondary winding and electrical energy will be transferred from the primary circuit through the transformer to the load. This field is made up from lines of force and has the same shape as a bar magnet.

If the current is increased, the lines of force move outwards from the coil. If the current is reduced, the lines of force move inwards.

If another coil is placed adjacent to the first coil then, as the field moves out or in, the moving lines of force will "cut" the turns of the second coil. As it does this, a voltage is induced in the second coil. With the 50 Hz AC mains supply, this will happen 50 times a second. This is called MUTUAL INDUCTION and forms the basis of the transformer.

The input coil is called the PRIMARY WINDING; the output coil is the SECONDARY WINDING. Fig: 3.3.4 shows step-down transformer.

Fig 3.3.3: Step-Down Transformer

The voltage induced in the secondary is determined by the TURNS RATIO.

For example, if the secondary has half the primary turns; the secondary will have half the primary voltage.

Another example is if the primary has 5000 turns and the secondary has 500 turns, then the turn’s ratio is 10:1.

If the primary voltage is 240 volts then the secondary voltage will be x 10 smaller = 24 volts. Assuming a perfect transformer, the power provided by the primary must equal the power taken by a load on the secondary. If a 24-watt lamp is connected across a 24 volt secondary, then the primary must supply 24 watts.

To aid magnetic coupling between primary and secondary, the coils are wound on a metal CORE. Since the primary would induce power, called EDDY CURRENTS, into this core, the core is LAMINATED. This means that it is made up from metal sheets insulated from each other. Transformers to work at higher frequencies have an iron dust core or no core at all.

Note that the transformer only works on AC, which has a constantly changing current and moving field. DC has a steady current and therefore a steady field and there would be no induction.

Some transformers have an electrostatic screen between primary and secondary. This is to prevent some types of interference being fed from the equipment down into the mains supply, or in the other direction. Transformers are sometimes used for IMPEDANCE MATCHING.

We can use the transformers as step up or step down.

Step Up transformer:

In case of step up transformer, primary windings are every less compared to secondary winding.

Because of having more turns secondary winding accepts more energy, and it releases more voltage at the output side.

Step down transformer:

Incase of step down transformer, Primary winding induces more flux than the secondary winding, and secondary winding is having less number of turns because of that it accepts less number of flux, and releases less amount of voltage.

Battery power supply:

A battery is a type of linear power supply that offers benefits that traditional line-operated power supplies lack: mobility, portability and reliability. A battery consists of multiple electrochemical cells connected to provide the voltage desired. Fig: 3.3.4 shows Hi-Watt 9V battery

Fig : Hi-Watt 9V Battery

The most commonly used dry-cell battery is the carbon-zinc dry cell battery. Dry-cell batteries are made by stacking a carbon plate, a layer of electrolyte paste, and a zinc plate alternately until the desired total voltage is achieved. The most common dry-cell batteries have one of the following voltages: 1.5, 3, 6, 9, 22.5, 45, and 90. During the discharge of a carbon-zinc battery, the zinc metal is converted to a zinc salt in the electrolyte, and magnesium dioxide is reduced at the carbon electrode. These actions establish a voltage of approximately 1.5 V.

The lead-acid storage battery may be used. This battery is rechargeable; it consists of lead and lead/dioxide electrodes which are immersed in sulfuric acid. When fully charged, this type of battery has a 2.06-2.14 V potential (A 12 volt car battery uses 6 cells in series). During discharge, the lead is converted to lead sulfate and the sulfuric acid is converted to water. When the battery is charging, the lead sulfate is converted back to lead and lead dioxide A nickel-cadmium battery has become more popular in recent years. This battery cell is completely sealed and rechargeable. The electrolyte is not involved in the electrode reaction, making the voltage constant over the span of the batteries long service life. During the charging process, nickel oxide is oxidized to its higher oxidation state and cadmium oxide is reduced. The nickel-cadmium batteries have many benefits. They can be stored both charged and uncharged. They have a long service life, high current availabilities, constant voltage, and the ability to be recharged. Fig: 3.3.5 shows pencil battery of 1.5V.

Fig : Pencil Battery of 1.5V

Step 2: Rectification

The process of converting an alternating current to a pulsating direct current is called as rectification. For rectification purpose we use rectifiers.

Rectifiers:

A rectifier is an electrical device that converts alternating current (AC) to direct current (DC), a process known as rectification. Rectifiers have many uses including as components of power supplies and as detectors of radio signals. Rectifiers may be made of solid-state diodes, vacuum tube diodes, mercury arc valves, and other components.

A device that it can perform the opposite function (converting DC to AC) is known as an inverter.

When only one diode is used to rectify AC (by blocking the negative or positive portion of the waveform), the difference between the term diode and the term rectifier is merely one of usage, i.e., the term rectifier describes a diode that is being used to convert AC to DC. Almost all rectifiers comprise a number of diodes in a specific arrangement for more efficiently converting AC to DC than is possible with only one diode. Before the development of silicon semiconductor rectifiers, vacuum tube diodes and copper (I) oxide or selenium rectifier stacks were used.

Bridge full wave rectifier:

The Bridge rectifier circuit is shown in figure, which converts an ac voltage to dc voltage using both half cycles of the input ac voltage. The Bridge rectifier circuit is shown in the figure. The circuit has four diodes connected to form a bridge. The ac input voltage is applied to the diagonally opposite ends of the bridge. The load resistance is connected between the other two ends of the bridge.

For the positive half cycle of the input ac voltage, diodes D1 and D3 conduct, whereas diodes D2 and D4 remain in the OFF state. The conducting diodes will be in series with the load resistance RL and hence the load current flows through RL.

For the negative half cycle of the input ac voltage, diodes D2 and D4 conduct whereas, D1 and D3 remain OFF. The conducting diodes D2 and D4 will be in series with the load resistance RL and hence the current flows through RL in the same direction as in the previous half cycle. Thus a bi-directional wave is converted into a unidirectional wave.

Input Output

Fig : Bridge rectifier: a full-wave rectifier using 4 diodes

DB107:

Now -a -days Bridge rectifier is available in IC with a number of DB107. In our project we are using an IC in place of bridge rectifier.

Features:

• Good for automation insertion

• Surge overload rating - 30 amperes peak

• Ideal for printed circuit board

• Reliable low cost construction utilizing molded

• Glass passivated device

• Polarity symbols molded on body

• Mounting position: Any

• Weight: 1.0 gram

Fig : DB107

Step 3: Filtration

The process of converting a pulsating direct current to a pure direct current using filters is called as filtration.

Filters:

Electronic filters are electronic circuits, which perform signal-processing functions, specifically to remove unwanted frequency components from the signal, to enhance wanted ones.

Introduction to Capacitors:

The Capacitor or sometimes referred to as a Condenser is a passive device, and one which stores energy in the form of an electrostatic field which produces a potential (static voltage) across its plates. In its basic form a capacitor consists of two parallel conductive plates that are not connected but are electrically separated either by air or by an insulating material called the Dielectric. When a voltage is applied to these plates, a current flows charging up the plates with electrons giving one plate a positive charge and the other plate an equal and opposite negative charge. This flow of electrons to the plates is known as the Charging Current and continues to flow until the voltage across the plates (and hence the capacitor) is equal to the applied voltage Vcc. At this point the capacitor is said to be fully charged and this is illustrated below.

Fig:Construction Of a Capacitor Fig :Electrolytic Capaticor

Units of Capacitance:

Microfarad (μF) 1μF = 1/1,000,000 = 0.000001 = 10-6 F

Nanofarad (nF) 1nF = 1/1,000,000,000 = 0.000000001 = 10-9 F

Pico farad (pF) 1pF = 1/1,000,000,000,000 = 0.000000000001 = 10-12 F

Operation of Capacitor:

Think of water flowing through a pipe. If we imagine a capacitor as being a storage tank with an inlet and an outlet pipe, it is possible to show approximately how an electronic capacitor works.

First, let's consider the case of a "coupling capacitor" where the capacitor is used to connect a signal from one part of a circuit to another but without allowing any direct current to flow.

If the current flow is alternating between zero and a maximum, our "storage tank" capacitor will allow the current waves to pass through.

However, if there is a steady current, only the initial short burst will flow until the "floating ball valve" closes and stops further flow.

So a coupling capacitor allows "alternating current" to pass through because the ball valve doesn't get a chance to close as the waves go up and down. However, a steady current quickly fills the tank so that all flow stops.

A capacitor will pass alternating current but (apart from an initial surge) it will not pass d.c.

Where a capacitor is used to decouple a circuit, the effect is to "smooth out ripples". Any ripples, waves or pulses of current are passed to ground while d.c. Flows smoothly.

Step 4: Regulation

The process of converting a varying voltage to a constant regulated voltage is called as regulation. For the process of regulation we use voltage regulators.

Voltage Regulator:

A voltage regulator (also called a ‘regulator’) with only three terminals appears to be a simple device, but it is in fact a very complex integrated circuit. It converts a varying input voltage into a constant ‘regulated’ output voltage. Voltage Regulators are available in a variety of outputs like 5V, 6V, 9V, 12V and 15V. The LM78XX series of voltage regulators are designed for positive input. For applications requiring negative input, the LM79XX series is used. Using a pair of ‘voltage-divider’ resistors can increase the output voltage of a regulator circuit.

It is not possible to obtain a voltage lower than the stated rating. You cannot use a 12V regulator to make a 5V power supply. Voltage regulators are very robust. These can withstand over-current draw due to short circuits and also over-heating. In both cases, the regulator will cut off before any damage occurs. The only way to destroy a regulator is to apply reverse voltage to its input. Reverse polarity destroys the regulator almost instantly. Fig: 3.3.10 shows voltage regulator.

Fig : Voltage Regulator

Resistors:

A resistor is a two-terminal electronic component that produces a voltage across its terminals that is proportional to the electric current passing through it in accordance with Ohm's law:

V = IR

Resistors are elements of electrical networks and electronic circuits and are ubiquitous in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).

The primary characteristics of a resistor are the resistance, the tolerance, maximum working voltage and the power rating. Other characteristics include temperature coefficient, noise, and inductance. Less well-known is critical resistance, the value below which power dissipation limits the maximum permitted current flow, and above which the limit is applied voltage. Critical resistance is determined by the design, materials and dimensions of the resistor.

Resistors can be made to control the flow of current, to work as Voltage dividers, to dissipate power and it can shape electrical waves when used in combination of other components. Basic unit is ohms.

Theory of operation:

Ohm's law:

The behavior of an ideal resistor is dictated by the relationship specified in Ohm's law:

V = IR

Ohm's law states that the voltage (V) across a resistor is proportional to the current (I) through it where the constant of proportionality is the resistance (R).

Power dissipation:

The power dissipated by a resistor (or the equivalent resistance of a resistor network) is calculated using the following:

Fig : Resistor Fig : Color Bands In Resistor

LED:

A light-emitting diode (LED) is a semiconductor light source. LEDs are used as indicator lamps in many devices, and are increasingly used for lighting. Introduced as a practical electronic component in 1962, early LEDs emitted low-intensity red light, but modern versions are available across the visible, ultraviolet and infrared wavelengths, with very high brightness. The internal structure and parts of a led are shown below.

Fig : Inside a LED Fig : Parts of a LED

Working:

The structure of the LED light is completely different than that of the light bulb. Amazingly, the LED has a simple and strong structure. The light-emitting semiconductor material is what determines the LED's color. The LED is based on the semiconductor diode.

When a diode is forward biased (switched on), electrons are able to recombine with holes within the device, releasing energy in the form of photons. This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy gap of the semiconductor. An LED is usually small in area (less than 1 mm2), and integrated optical components are used to shape its radiation pattern and assist in reflection. LEDs present many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved robustness, smaller size, faster switching, and greater durability and reliability. However, they are relatively expensive and require more precise current and heat management than traditional light sources. Current LED products for general lighting are more expensive to buy than fluorescent lamp sources of comparable output. They also enjoy use in applications as diverse as replacements for traditional light sources in automotive lighting (particularly indicators) and in traffic signals. The compact size of LEDs has allowed new text and video displays and sensors to be developed, while their high switching rates are useful in advanced communications technology. The electrical symbol and polarities of led are shown in fig:

Fig : Electrical Symbol & Polarities of LED

LED lights have a variety of advantages over other light sources:

• High-levels of brightness and intensity

• High-efficiency

• Low-voltage and current requirements

• Low radiated heat

• High reliability (resistant to shock and vibration)

• No UV Rays

• Long source life

• Can be easily controlled and programmed

Applications of LED fall into three major categories:

• Visual signal application where the light goes more or less directly from the LED to the human eye, to convey a message or meaning.

• Illumination where LED light is reflected from object to give visual response of these objects.

• Generate light for measuring and interacting with processes that do not involve the human visual system.

SOFTWARE DESCRIPTION

ARDUINO SOFTWARE:

The Arduino is a family of microcontroller boards to simplify electronic design, prototyping and experimenting for artists, hackers, hobbyists, but also many professionals. People use it as brains for their robots, to build new digital music instruments, or to build a system that lets your house plants tweet you when they’re dry. Arduinos (we use the standard Arduino Uno) are built around an ATmega microcontroller — essentially a complete computer with CPU, RAM, Flash memory, and input/output

What you will need:

 A computer (Windows, Mac, or Linux)

 An Arduino-compatible microcontroller (anything from this guide should work)

 A USB A-to-B cable, or another appropriate way to connect your Arduino-compatible microcontroller to your computer (check out this USB buying guide if you’re not sure which cable to get).

 An Arduino Uno

 Windows 7, Vista, and XP

 Installing the Drivers for the Arduino Uno (from Arduino.cc)

 Plug in your board and wait for Windows to begin it’s driver installation process After a few moments, the process will fail, despite its best efforts

 Click on the Start Menu, and open up the Control Panel

 While in the Control Panel, navigate to System and Security. Next, click on System Once the System window is up, open the Device Manager

 Look under Ports (COM & LPT). You should see an open port named “Arduino UNO (COMxx)”.

 If there is no COM & LPT section, look under ‘Other Devices’ for ‘Unknown Device’

 Right click on the “Arduino UNO (COMxx)” or “Unknown Device” port and choose the “Update Driver Software” opti Next, choose the “Browse my computer for Driver software” option

 Finally, navigate to and select the Uno’s driver file, named “ArduinoUNO.inf”, located in the “Drivers” folder of the Arduino Software download (not the “FTDI USB Drivers” sub-directory). If you cannot see the .inf file, it is probably just hidden. You can select the ‘drivers’ folder with the ‘search sub-folders’ option selected instead. Windows will finish up the driver installation

LAUNCH AND BLINK!

After following the appropriate steps for your software install, we are now ready to test your first program with your Arduino board!

 Launch the Arduino application

 If you disconnected your board, plug it back in

 Open the Blink example sketch by going to: File > Examples > 1.Basics > Blink

 Select the type of Arduino board you’re using: Tools > Board > your board type

 Select the serial/COM port that your Arduino is attached to: Tools > Port > COMxx

If you’re not sure which serial device is your Arduino, take a look at the available ports, then unplug your Arduino and look again. The one that disappeared is your Arduino.

With your Arduino board connected, and the Blink sketch open, press the ‘Upload’ button

After a second, you should see some LEDs flashing on your Arduino, followed by the message ‘Done Uploading’ in the status bar of the Blink sketch.

If everything worked, the onboard LED on your Arduino should now be blinking! You just programmed your first Arduino!