VEHICLE HEALTH MONITORING SYSTEM

Team 5

Vikas Reddy - 12M61A0489
Rama Sai Mallam - 12M61A0492
Sai Manideep - 13M65A0401
Venkatesh Gujjarlapudi - 12M61A0479

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S.NO	DESCRIPTION	
1	CAN	Controller Area Network
2	LCD	liquid-crystal display
3	ARM	Advanced RISC Machines
4	ADC	Analog to Digital Converter
5	SAR	Successive Approximation Register
6	EOC	End Of Conversion
7	UART	Universal Asynchronous Receiver/Transmitter
8	RFID	Radio Frequency Identification
9	AIDC	Automatic Identification and Data Capture

ABSTRACT

This project deals with developing an embedded system for detecting the vehicle condition by monitoring the internal parameters that are used in evaluating the vehicle‘s current health condition. Traveler information plays a critical role in supporting safety, security, mobility, and in improving the reliability of travel. This traveler information can be a continuous data on performance of the vehicle and the status of its internal components. In this project, an in-vehicle embedded system is being developed to generate a vehicle health report (VHR) whenever needed by the user. It also acts as an eco friendly vehicle by monitoring the emissions from the vehicle which in turn helps in regulating (by taking proper actions to reduce the emissions as per the faults indicated in the VHR) the environmental pollution. It predicts the future errors so that the driver can have an uninterrupted journey and can avoid accidents. Thus, it alerts the driver about future errors and assists him for a safe drive. The data required for generating the health report consists of parameter values (outputs of in-built sensors) of different systems inside the vehicle. A real time evaluation system is being defined that can be used for rapid condition screening and provide reliable information about the vehicle conditions. This real time evaluation system can be called Vehicle Health Monitoring System.

CHAPTER -1

INTRODUCTION TO VEHICLE HEALTH MONITORING SYSTEM

INTRODUCTION TO VEHICLE HEALTH MONITORING SYSTEM

1.1 INTRODUCTION:

Automobile is one of the most widely distributed cyber-physical systems. Over the last few years, the electronic explosion in automotive vehicles has significantly increased the complexity, heterogeneity and interconnectedness of embedded systems. Although designed to sustain long life, systems degrade in performance due to gradual development of anomalies eventually leading to faults. In addition, system usage and operating conditions may lead to different failure modes that can affect the performance of vehicles. Advanced diagnosis and prognosis technologies are needed to quickly detect and isolate faults in network-embedded automotive systems so that proactive corrective maintenance actions can be taken to avoid failures and improve vehicle availability.

In this project we will develop a system which will monitor the health i.e. level of fuel in the Tank, Temperature at the engine, Air pressure in the tires, Voltage of the battery; provide security (RFID Tag), identify different zones (School Zone, Hospital Zone) using RF and line Crossing (Line Follower) alert system in a CAR. Two ARM Controllers will communicate using CAN Protocol. Finally, we will display readings which were taken from the Sensors on a Graphical LCD. When there are more electrical control devices in the modern cars, such as power train management system, antilock braking system (ABS), and acceleration skid control (ASC) system, etc, the functionality and wiring of these electric control units (!XU) gets more complicated. Therefore, it is of great concern to upgrade the traditional wire harness to a smart &car network. In 198Os, a Germany car component provider Robert Bosch Co. introduced an in-car network the controller area network (CAN) bus, to replace the complex and expensive traditional in-car wiring. Here, a high-level protocol CAN open is adopted to interconnect those CAN nodes with reliable communications among sensors.

1.2 ORGANIZATION OF THESIS:

The project tutorial is a comprehensive tutorial on design of vehicle monitoring system. In view of the proposed thesis work explanation of theoretical aspects and algorithms used in this work are presented as per the sequence described below.

Chapter-1 starts with an introductory chapter that gives some basic idea about the project.

Chapter-2 is about the aim and objectives of the project.

Chapter-3 literature survey regarding the project is provided, and its hardware requirement.

Chapter-4 is about design implementation and operation of block diagram.

Chapter-5 The construction and description of various modules used for the application are described in detail.

Chapter-6 Includes different types of sensors used in the project.

Chapter-7 consists of firmware and software implementation of the project.

Chapter-8 consists of results & application & sample output.

Chapter-9 conclusion & future enhancement

Chapter-10 Bibliography of the project.

CHAPTER-2

AIM AND OBJECTIVES

AIM & OBJECTIVES OF VEHICLE HEALTH MONITORING SYSTEM DESIGN

2.1 AIM & OBJECTIVE:

The main aim of our project is to develop a vehicle health monitoring and alert System in an Automobile**.**

The main objective of the project is to design a vehicle health monitoring system that has the potential to run on low energy with different sensors. It is very common in designing any of application to find out the high-quality products that cost a little and work for a period of long time. In this project for designing of a vehicle health monitoring system with sensors, the most important factors to be achieved in the design are the cost and simplicity of the circuit. The cost will be minimized by choosing low cost component with better use.

2.2 BACKGROUND OF THE PROJECT:

The total car sales rose to 2.6 million units in India alone by the year 2014. With the increase in number of cars along with other modes of transport such as public transport system, vehicles for supply chains and two wheelers on road, the issues like safety, fuel consumption, pollution check are of utmost importance which depends on vehicle condition, road infrastructure and driver behavior. So to improve the safety of the vehicles we design a vehicle health monitoring system which unlocks the door only if the supported card is kept. If the code doesn't matches the existing code the door will not open. It has several sensors included in it like temperature, pressure sensor and the battery level indicator, fuel level. The buzzer will rang if any dangerous position is reached in the car and the temperature of the car is noted every second so that a certain precautions could be taken. The health of the vehicle is monitored frequently so that the vehicles health is maintained in safe state.

CHAPTER-3

LITERATURE SURVEY & COMPONENTS

3.1 LITERATURE SURVEY:

This system is based on the widely used CAN bus technology, to extract the vehicle's status or fault information. Therefore, vehicle interior network came into existence. CAN (Controller Area Network, CAN) relying on its stability performance, low price and high reliability and real-time, has now been widely used in automotive internal network. The increasing complexity of automotive electronic control system makes automotive fault diagnosis and maintenance work more difficult. Transfer of large amount of data and exchange of different signals between electronic control systems on the bus vehicle is essential. This project focuses on vehicle health monitoring system based on CAN bus, which can be used to improve the efficiency of monitoring, to maintain the system security, to lower the maintenance costs as well as the operating costs. This system is based on the widely used CAN bus technology. CAN bus is used to extract the vehicle's status or fault information. In automotive electronics the important parameters such as battery level detect, brake fluid level detects, speed sensing system, fuel level detect system etc. are obtained using CAN

3.2. HARDWARE REQUIREMENTS:

- LPC2148 Micro controller.
- Sensors (Pressure Sensor, Temperature, seat belt detector, Fuel Level Indicator, battery level,)
- Graphical LCD
- L293D for driving motors.
- 60 rpm motor.
- Ignition Key
- Regulated +5v and +3.3v Power supply.

CHAPTER 4

DESIGN IMPLEMENTATION

4.1 FLOW CHART:

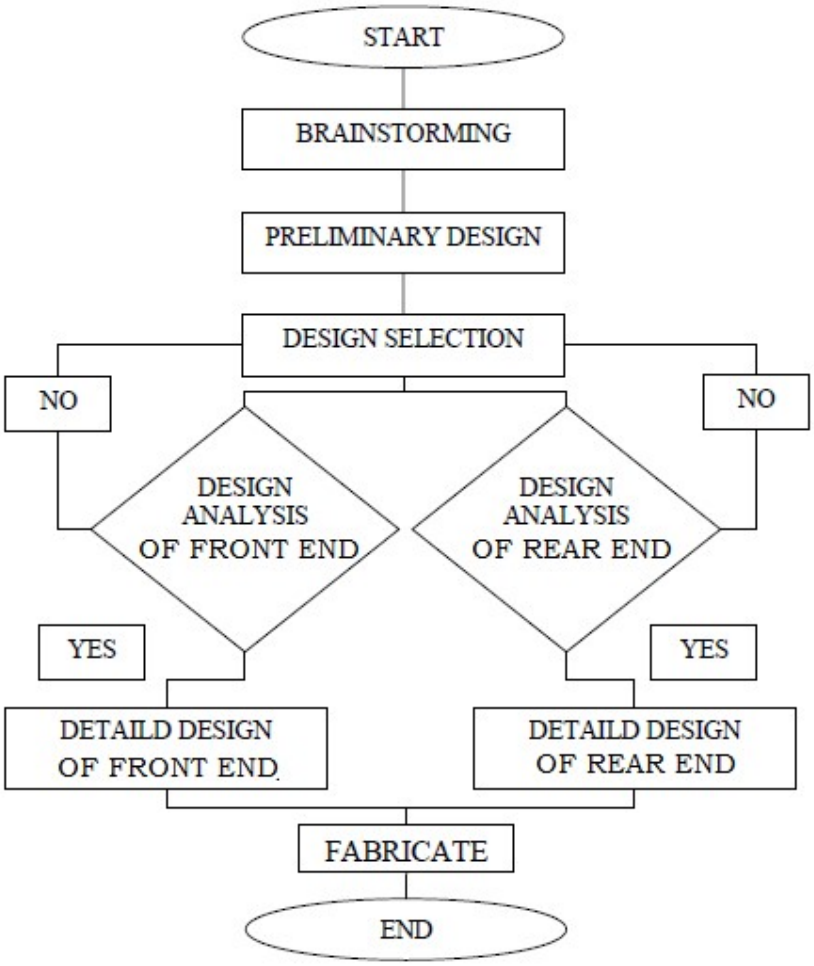


Fig 4.1 Flow chart

- Brainstorming is the rough idea about the chassis and body designed is described by sketching the chassis and body.
- Preliminary design process is the evaluation leading up to the selection of the best overall design. It includes the overall system configuration, basic schematics and layout.
- The basic design must follow the regulations. A decision then must be made on one of the designs as the preliminary design. These designs must go through to next step of design process.

4.2 BLOCK DIAGRAM:

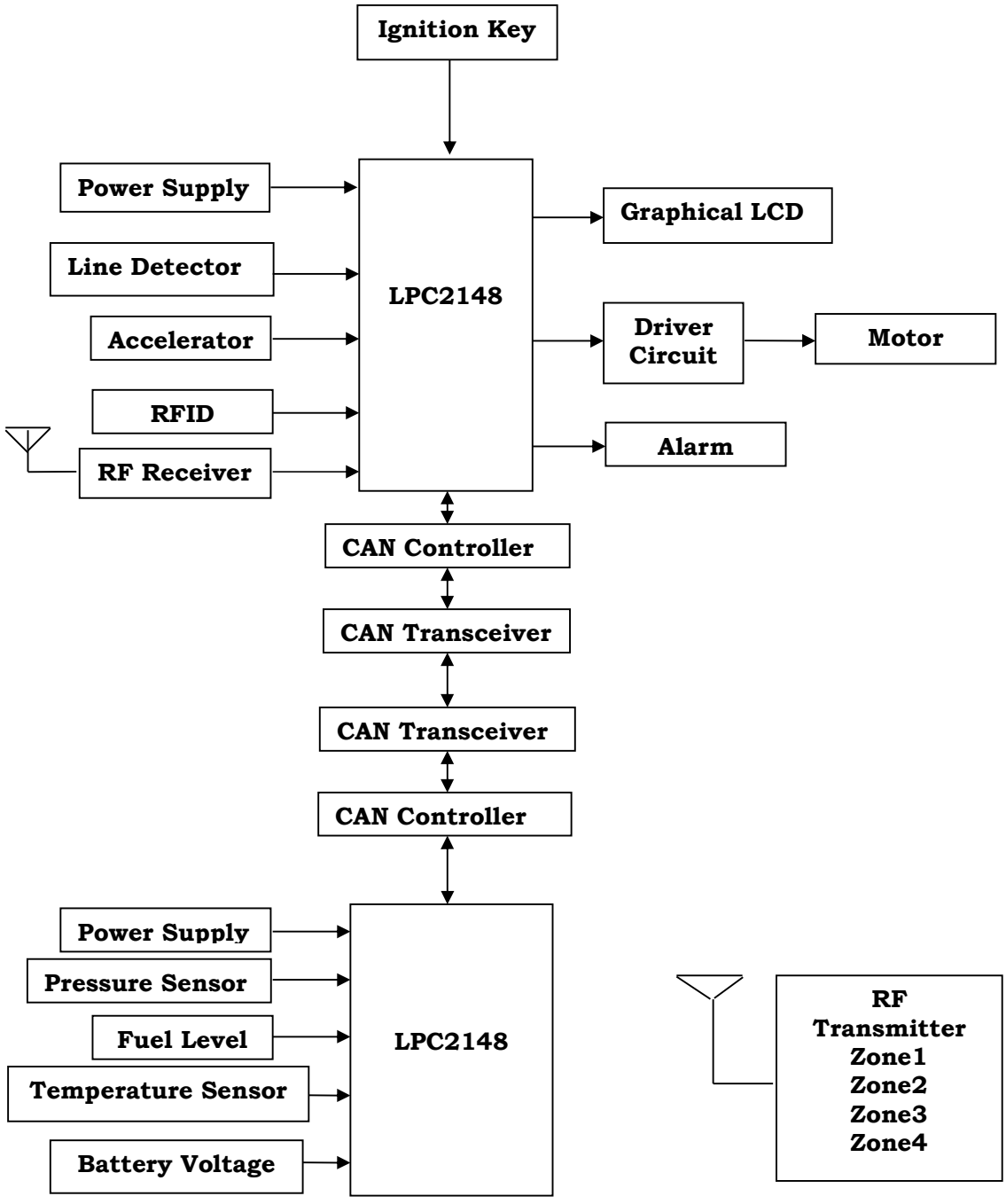


Fig 4.2: Block diagram

4.3 THE FRONT-END SUB-SYSTEM:

This subsystem for generating warning signals for the front-end collision avoidance is constructed by measuring the distance with SICK laser radar (LMS221-30206). The collision avoidance of the front-end car usually operated under a relatively high speed. Therefore, the laser radar is required to detect the front car in a relative long distance as far as 50-80 *m* with a high resolution as 1 cm**.** The warning signal is for deriver's attention to avoid the collision by the braking action actively.

4.4 THE REAR-END SUB-SYSTEM:

In the other hand, the real-end collision avoidance would be inherently in shorter distance with a slow approaching speed. Besides, only passively action, which a warning signal can be generated for the approaching car drivers, can be taken. Therefore, rear-end collision avoidance warning sub-system is constructed with the available ultrasonic sensors which have been widely implemented on commercial vehicles.

4.5 LCD:

A liquid-crystal display (LCD) is a flat-panel display or other electronic visual display that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly. LCDs are available to display arbitrary images or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and 7-segment displays as in a digital clock. They use the same basic technology, except that arbitrary images are made up of a large number of small pixels, while other displays have larger elements.

Figure 4.5: LCD

The LCD screen is more energy-efficient and can be disposed of more safely than a CRT. Its low electrical power consumption enables it to be used in battery-powered electronic equipment more efficiently than CRTs. Each pixel of an LCD typically consists of a layer of molecules aligned between two transparent electrodes, and two polarizing filters the axes of transmission of which are perpendicular to each other. Without the liquid crystal between the polarizing filters, light passing through the first filter would be blocked by the second polarizer.

4.6 BUZZERS:

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.

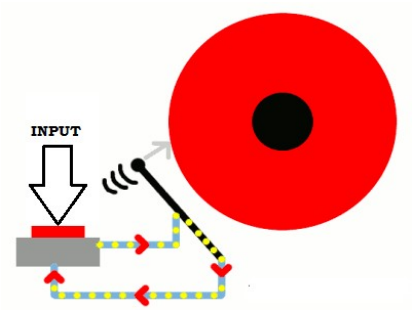


Figure 4.6: Buzzer

4.7 ARM7TDMI PROCESSOR:

The ARM7TDMI core is a member of the ARM family of general-purpose 32-bitMicroprocessors. The ARM family offers high performance for very low power Consumption and small size. The RISC instruction set and related decode mechanism are much simple than those of Complex Instruction Set Computer (CISC) designs.

This simplicity gives:

• A high instruction throughput

• An excellent real-time interrupt response

• A small, cost-effective, processor macro cell.

4.8 THE INSTRUCTION PIPELINE:

The ARM7TDMI core uses a pipeline to increase the speed of the flow of instructions to the processor. This enables several operations to take place simultaneously, and the processing and memory systems to operate continuously. A three-stage pipeline is used as shown in fig 4.4, so instructions are executed in three stages:

• Fetch

• Decode

• Execute.

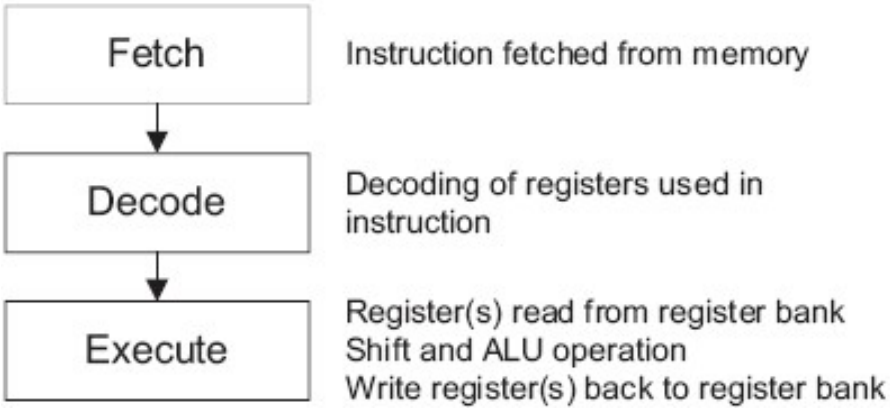


Figure 4.8: The instruction pipeline

During normal operation, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory. The program counter points to the instruction being fetched rather than to the instruction being executed. This is important because it means that the Program Counter (PC) value used in an executing instruction is always two instructions ahead of the address.

4.9 MEMORY ACCESS:

The ARM7TDMI core has Von Neumann architecture, with a single 32-bit data bus carrying both instructions and data. Only load, store, and swap instructions can access data from memory.

Data can be:

- 8-bit (bytes)
- 16-bit (half words)
- 32-bit (words).

Words must be aligned to 4-byte boundaries. Half words must be aligned to 2-byte boundaries.

4.10 MEMORY INTERFACE:

The ARM7TDMI processor memory interface has been designed to allow performance potential to be realized, while minimizing the use of memory. Speed-critical control signals are pipelined to enable system control functions to be implemented in standard low-power logic. These control signals facilitate the exploitation of the fast-burst access modes supported by many on-chip and off-chip memory technologies.
The ARM7TDMI core has four basic types of memory cycle:

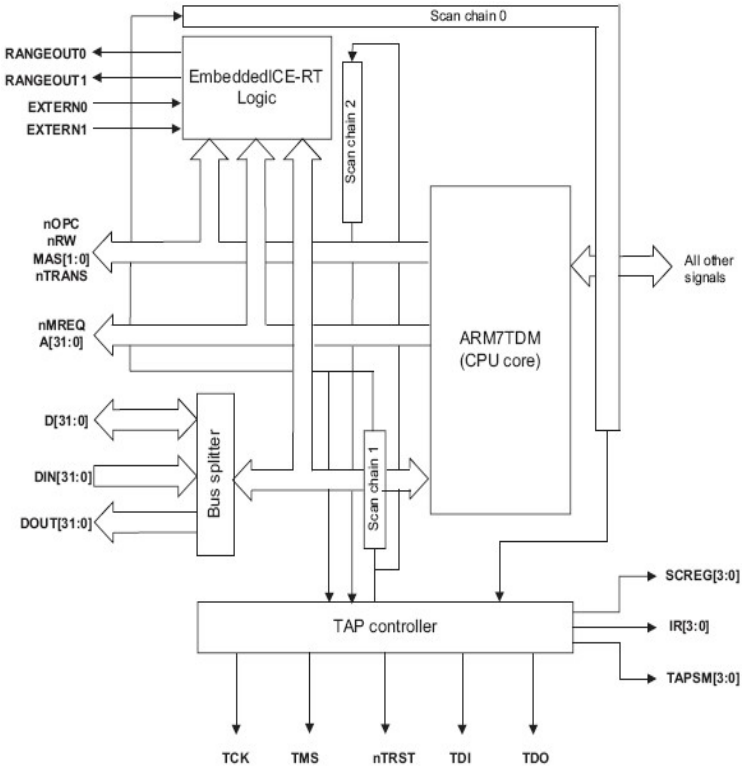


Figure 4.10: ARM7TDMI processor block diagram

4.11 ARCHITECTURE:

The ARM7TDMI processor has two instruction sets:

• The 32-bit ARM instruction set

• The 16-bit Thumb instruction set.

The ARM7TDMI processor is an implementation of the ARMv4T Architecture.

4.12 INSTRUCTION COMPRESSION:

Microprocessor architectures traditionally have the same width for instructions and data. In comparison with 16-bit architectures, 32-bit architectures exhibit higher performance when manipulating 32-bit data and can address a large address space much more efficiently.16-bit architectures typically have higher code density than 32-bit architectures, but approximately half the performance. Thumb implements a 16-bit instruction set on a 32-bit architecture to provide:

• Higher performance than a 16-bit architecture

• Higher code density than a 32-bit architecture.

4.13 THE THUMB INSTRUCTION SET:

The Thumb instruction set is a subset of the most commonly used 32-bit ARM instructions. Thumb instructions are each 16 bits long and have a corresponding 32-bit ARM instruction that has the same effect on the processor model. Thumb instructions operate with the standard ARM register configuration, allowing excellent interoperability between ARM and Thumb states. On execution, 16-bit Thumb instructions are transparently decompressed to full 32-bitARM instructions in real time without performance loss.

Thumb has all the advantages of a 32-bit core:

• 32-bit address space

• 32-bit registers

• 32-bit shifter, and Arithmetic Logic Unit (ALU)

• 32-bit memory transfer.

Thumb therefore offers a Long Branch range, powerful arithmetic operations, and a large address space. Thumb code is typically 65% of the size of ARM code and provides 160% of the performance of ARM code when running from a 16-bit memory system. Thumb, therefore, makes the ARM7TDMI core ideally suited to embedded applications with restricted memory bandwidth, where code density and footprint is important. The availability of both 16-bit Thumb and 32-bit ARM instruction sets gives designers the flexibility to emphasize performance or code size on a subroutine level, according to the requirements of their applications. For example, critical loops for applications such as fast interrupts and DSP algorithms can be coded using the full ARM instruction set then linked with Thumb code.

4.14 CONTROLLER AREA NETWORK PROTOCOL:

Controller Area Network (CAN) is an advanced serial bus system that efficiently supports distributed control system with a very high level of security Robert Bosch Germany initially developed it for the use in motor vehicles in the late 1980's. It's domain of application ranges from high-speed network to low cost multiplex wiring. To improve the behavior of the vehicle, it was necessary for the different control system (and their sensor) to exchange information. This was usually done by discrete interconnection of the different system (i.e. point -to - point wiring). The requirement for the information exchange has then grown to such an extent that a cable network with a length up to several miles and many connectors were required. This leads to growing problems concerning material cost, production time and reliability.

The solution to this problem was the connection of the Control system via a serial bus system. With the use of CAN, point - to - point wiring is replaced by one serial bus connecting to all control systems. This is accomplished by adding some CAN specific hardware to each

control unit that provides the “rules" or the protocol for transmitting and receiving information via the bus.

The CAN protocol uses the data link layer and the physical layer in the ISO_OSI model.

4.15 OVERVIEW OF CAN PROTOCOL:

CAN is a multi - master bus with an open, linear structure with one logic bus line. The number of nodes is not limited by the protocol. In CAN protocol, two versions are available. They are version 2.0A CAN and version 2.0B CAN. Version 2.0A is original CAN specifications specify an 11bit identifier which allows 2^11(=2048) different message identifiers and is known as standard CAN. Version 2.0B CAN contain 29 bit identifiers which allows 2^29 (over 536 million) message identifiers.

CAN have the following properties:

- Prioritization of messages.
- Guarantee of latency times.
- Configuration flexibility.
- Multicast reception with time synchronization.
- System wide data consistency.
- Error detection and error signaling.

The CAN protocol handle bus accesses according to the concept called “Carrier Sense Multiple Access with arbitration on message priority”. This arbitration Concept avoids collisions of messages whose transmission was started by more than one node simultaneously and makes sure the most important message is sent first without time loss. If two or more bus nodes start their transmission at the same time after having found the bus to be idle, collision of the messages is avoided by bitwise arbitration. Each node sends the bits of its message identifier and monitors the bus level.

4.16 BASIC CONCEPTS:

Some of the basic concepts which are necessary to understand the CAN operation. They are as follows.

4.16.1 MESSAGES:

Information on the bus is sent in fixed format messages of different but limited length. When the bus is free any connected unit may start to transmit a new message.

4.16.2 INFORMATION ROUTING:

In CAN systems a can node does not make use of any information about the system configuration (e.g. station addresses).this has several important consequences.

4.16.3 SYSTEM FLEXIBILITY:

Nodes can be added to the CAN network without requiring any change in the software or hardware of any node and application layer.

4.16.4 MESSAGE ROUTING:

The content of a message is named by an IDENTIFIER. The identifier does not indicate the destination of the message but describes the meaning of the data, so that all nodes in the network are able to decide by message filtering whether the data is to be acted upon by them or not.

4.16.5 MULTICASTING:

As a consequence of the concept of message filtering any number of nodes can receive and simultaneously act upon the same message.

4.16.6 DATA CONSISTENCY:

Within a CAN network it is guaranteed that a message is simultaneously accepted either by all nodes or by no node. Thus data

consistency of a system is achieved by the concepts of multicast and by error handling.

4.16.7 BIT RATE:

The speed of CAN may be different in different systems. However, in a given system the bit rate is uniform and fixed.

4.16.8 PRIORITIES:

The identifier defines a static message priority during bus access.

4.16.9 REMOTE DATA REQUEST:

By sending a remote frame a node requiring data may request another node to send the corresponding data frame. The data frame and the corresponding remote frame named by the same identifier.

4.16.10 MULTI MASTER:

When the bus is free any may start to transmit a message. The unit with the message of highest priority to be transmitted gains bus access.

4.16.11 ARBITRATION:

Whenever the bus is free, any unit may start to transmit a message. If two or more units start transmitting messages at the same time, the bus access conflict is resolved by bit wise arbitration using the identifier. The mechanism of arbitration grantees that neither information nor time is lost. If a data frame and a remote frame with the same identifier are initiated at the same time, the data frame prevails over the remote frame. During arbitration every transmitter compares the level of the bit transmitted with the level that is monitored on the bus. If these levels are equal the unit may continue to send. When a 'recessive' level is sent and a 'dominant' level is monitored, the unit has lost arbitration and must withdraw without sending one more bit.

4.17 BUS CHARACTERISTICS:

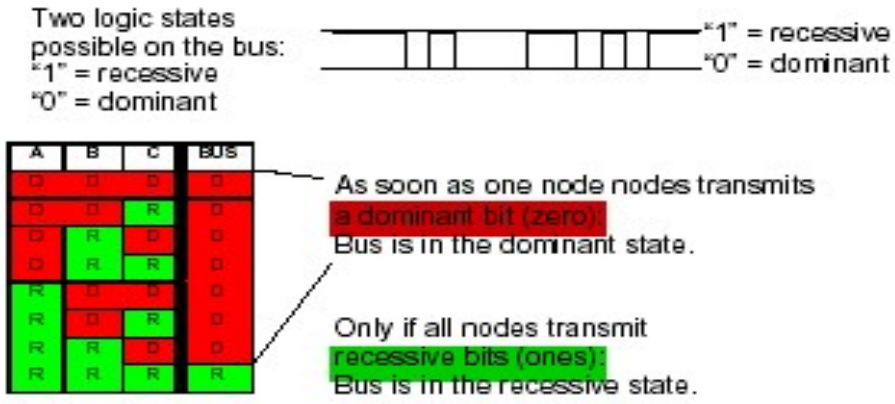


Figure 4.17: Bus characteristics

There are two bus states, called "dominant" and "recessive". The bus logic uses a "Wired-AND" mechanism, that is, "dominant bits" (equivalent to the logic level "Zero") overwrite the "recessive" bits (equivalent to the logic level "One").

4.17.1 BUS ACCESS AND ARBITRATION:

The CAN protocol handles bus accesses according to the concept called "Carrier Sense Multiple Access with Arbitration on Message Priority". This arbitration concept avoids collisions of messages whose transmission was started by more than one node simultaneously and makes sure the most important message is sent first without time loss.

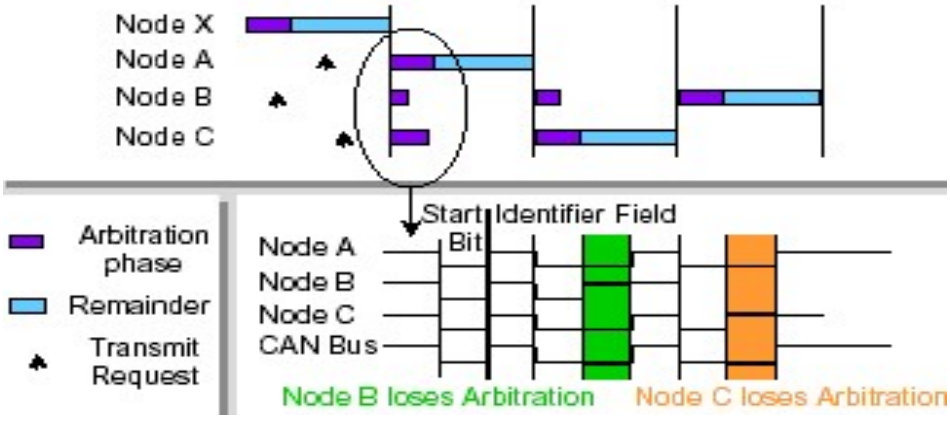


Figure 4.17.1: Bus Access and Arbitration

In the picture above you can see the trace of the transmit pins of three bus nodes called A, B and C, and the resulting bus state according to the wired-AND principle. If two or more bus nodes start their transmission at the same time after having found the bus to be idle, collision of the messages is avoided by bitwise arbitration. Each node sends the bits of its message identifier and monitors the bus level. At a certain time nodes A and C send a dominant identifier bit. Node B sends a recessive identifier bit but reads back a dominant one. Node B loses bus arbitration and switches to receive mode.

Some bits later node C loses arbitration against node A. This means that the message identifier of node A has a lower binary value and therefore a higher priority than the messages of nodes B and C. In this way, the bus node with the highest priority message wins arbitration without losing time by having to repeat the message. Nodes B and C automatically try to repeat their transmission once the bus returns to the idle state. Node B loses against node C, so the message of node C is transmitted next, followed by node B's message.

It is not permitted for different nodes to send messages with the same identifier as arbitration could fail leading to collisions and errors

4.18 MESSAGE TRANSFER:

Message transfer is manifested and controlled by two frame types.

4.18.1 A DATA FRAME:

As shown in fig 4.18.1 carries data from a transmitter to the receivers. It is composed of seven different bit fields. Start of frame, Arbitration field, control field, data field, crc field, Ack field, End of frame.

4.18.2 A REMOTE FRAME:

As shown in fig4.18.2 is transmitted by a bus unit to request the transmission of the data frame with the same identifier. It is composed of six different bit fields. Start of frame, Arbitration field, control field, crc field, Ack field, End of frame.

A "Data Frame" is generated by a CAN node when the node wishes to transmit data. The Standard CAN Data Frame is shown above. The frame begins with a dominant Start of Frame bit for hard synchronization of all nodes.

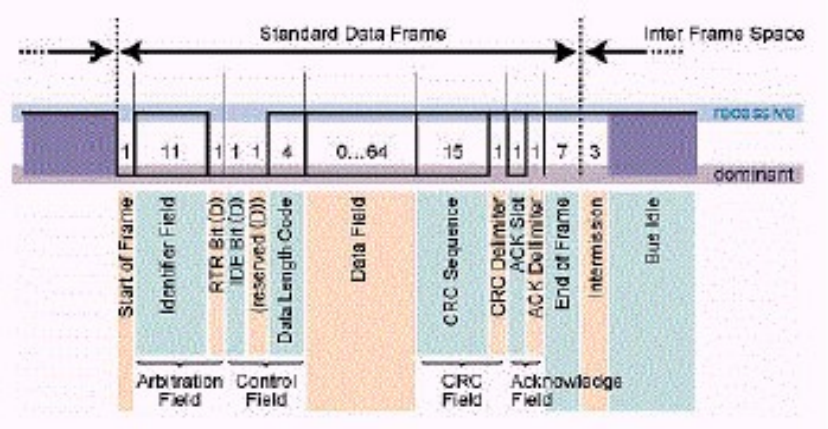


Figure 4.18.1: Data Frame

The Start of Frame bit is followed by the Arbitration Field consisting of 12 bits. The 11-bit Identifier, which reflects the contents and priority of the message, and the Remote Transmission Request bit. The Remote transmission request bit is used to distinguish a Data Frame (RTR = dominant) from a Remote Frame (RTR = recessive).The next field is the Control Field, consisting of 6 bits. The first bit of this field is called the IDE bit (Identifier Extension) and is at dominant state to specify that the frame is a Standard Frame. The following bit is reserved and defined as a dominant bit. The remaining 4 bits of the Control Field are the Data Length Code (DLC) and specify the number of bytes of data contained in the message (0 - 8 bytes).

The data being sent follows in the Data Field which is of the length defined by the DLC above (0, 8, 16, 56 or 64 bits).The Cyclic Redundancy Field (CRC field) follows and is used to detect possible transmission errors. The CRC Field consists of a 15-bit CRC sequence, completed by the recessive CRC Delimiter bit. The next field is the Acknowledge Field. During the ACK Slot bit the transmitting node sends out a recessive bit. Any node that has received an error free frame acknowledges the correct reception of the frame by sending back a dominant bit (regardless of whether the node is configured to accept that specific message or not). From this it can be seen that CAN belongs to the "in-bit-response" group of protocols. The recessive Acknowledge Delimiter completes the Acknowledge Slot and may not be overwritten by a dominant bit. Seven recessive bits (End of Frame) end the Data Frame.

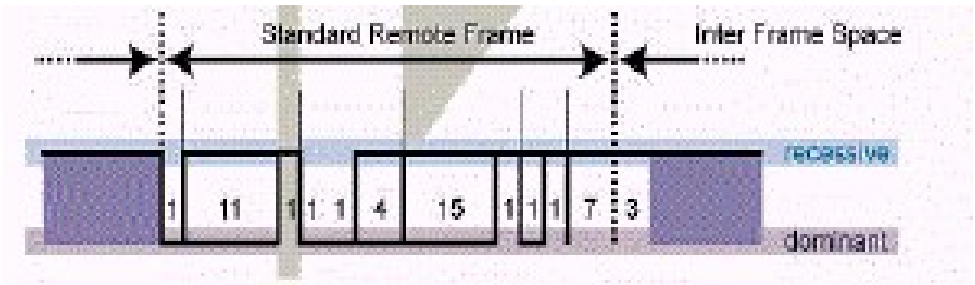


Figure 4.18.2: Remote Frame

Generally data transmission is performed on an autonomous basis with the data source node (e.g. a sensor) sending out a Data Frame. It is also possible, however, for a destination node to request the data from the source by sending a Remote Frame.

There are 2 differences between a Data Frame and a Remote Frame. Firstly the RTR-bit is transmitted as a dominant bit in the Data Frame and secondly in the Remote Frame there is no Data Field. In the very unlikely event of a Data Frame and a Remote Frame with the same identifier being transmitted at the same time. The Data Frame wins arbitration due to the dominant RTR bit following the identifier. In this

way, the node that transmitted the Remote Frame receives the desired data immediately.

4.19 CAN CONTROLLER OPERATION:

The CAN controller having error detection and handling capacity in efficient way, it also makes sleep mode of operation and produces interrupts

4.19.1 ERROR HANDLING:

The CAN Controllers count and handle transmit and receive errors as specified in CAN Spec 2.0B. The Transmit and Receive Error Counters are incremented for each detected error and are decremented when operation is error-free. If the Transmit Error counter contains 255 and another error occurs, the CAN Controller is forced into a state called Bus-Off. In this state, the following register bits are set: BS in CANSR, BEI and EI in CANIR if these are enabled, and RM in CANMOD. RM resets and disables much of the CAN Controller. Also, at this time the Transmit Error Counter is set to 127 and the Receive Error Counter is cleared. Software must next clear the RM bit. Thereafter the Transmit Error Counter will count down 128 occurrences of the Bus Free condition (11 consecutive recessive bits). Software can monitor this countdown by reading the Tx Error Counter. When this countdown is complete, the CAN Controller clears BS and ES in CANSR, and sets EI in CANSR if EIE in IER is 1.The Tx and Rx error counters can be written if RM in CANMOD is 1. Writing 255 to the Tx Error Counter forces the CAN Controller to Bus-Off state. If Bus-Off (BS in CANSR) is 1, writing any value 0 through 254 to the Tx Error Counter clears Bus-Off. When software clears RM in CANMOD thereafter, only one Bus Free condition (11 consecutive recessive bits) is needed before operation resumes.

4.19.2 SLEEP MODE:

The CAN Controller will enter sleep mode if the SM bit in the CAN Mode register is 1, no CAN interrupt is pending, and there is no activity on the CAN bus. Software can only set SM when RM in the CAN Mode

register is 0; it can also set the WUIE bit in the CAN Interrupt Enable register to enable an interrupt on any wake-up condition.

The CAN Controller wakes up (and sets WUI in the CAN Interrupt register if WUIE in the CAN Interrupt Enable register is 1) in response to a) a dominant bit on the CAN bus, or b) software clearing SM in the CAN Mode register. A sleeping CAN Controller that wakes up in response to bus activity, is not able to receive an initial message, until after it detects Bus free (11 consecutive recessive bits). If an interrupt is pending or the CAN bus is active when software sets SM, the wakeup is immediate.

4.19.3 INTERRUPTS:

Each CAN Controller produces 3 interrupt requests, Receive, Transmit, and “other status”. The Transmit interrupt is the OR of the Transmit interrupts from the three Tx Buffers. Each Receive and Transmit interrupt request from each controller is assigned its own channel in the Vectored Interrupt Controller (VIC), and can have its own interrupt service routine. The “other status” interrupts from all of the CAN controllers, and the Acceptance Filter LUTerr condition, are OR-ed into one VIC channel.

4.19.4 TRANSMIT PRIORITY:

If the TPM bit in the CANMOD register is 0, multiple enabled Tx Buffers contend for the right to send their messages based on the value of their CAN Identifier (TID). If TPM is 1, they contend based on the PRIO fields in bits 7:0 of their CANTFS registers. In both cases the smallest binary value has priority. If two (or three) transmit-enabled buffers have the same smallest value, the lowest-numbered buffer sends first. The CAN controller selects among multiple enabled Tx Buffers dynamically, just before it sends each message.

4.20 ANALOG TO DIGITAL CONVERTER:

The ADC0808, ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-

channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter shown in fig 4.16 uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended analog signals. The device eliminates the need for external zero and full scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATEÉ outputs. The design of the ADC0808, ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0808, ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power.

4.20.1 Block diagram

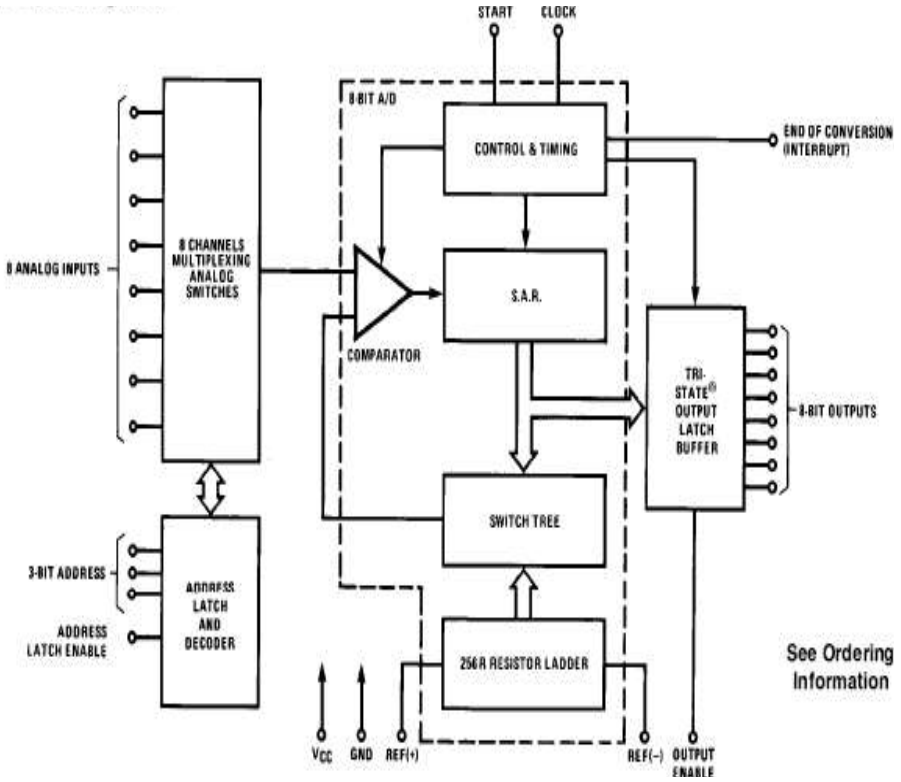


Figure 4.20.1: Block diagram of ADC

4.20.2 FUNCTIONAL DESCRIPTION:

The some of the blocks in ADC and their functional description is given below.

4.20.3 MULTIPLEXER:

The device contains an 8-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. Table I shows the input states for the address lines to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

SELECTED	ADDRESS LINE		
ANALOG CHANNEL	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

Table 4.20.3 Multiplexer

4.20.4 THE CONVERTER:

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network [5], the successive approximation register, and the comparator. The converter's digital outputs are positive true. The 256R ladder network approach (Figure 4.2) was chosen over the conventional R/2R ladder because of its inherent monotonic, which guarantees no missing digital codes. Monotonic is particularly important in closed loop feedback control systems. A non-monotonic relationship can cause oscillations that will be catastrophic for the system. Additionally, the 256R network does not cause load variations on the reference voltage. The bottom resistor and the top resistor of the ladder network are not the same

value as the remainder of the network. The difference in these resistors causes the output characteristic to be symmetrical with the zero and full-scale points of the transfer curve. The first output transition occurs when the analog signal has reached +1/2 LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

The successive approximation register (SAR) performs 8 iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter. Figure shows a typical example of a 3-bit converter. In the ADC0808, ADC0809, the approximation technique is extended to 8 bits using the 256R network.

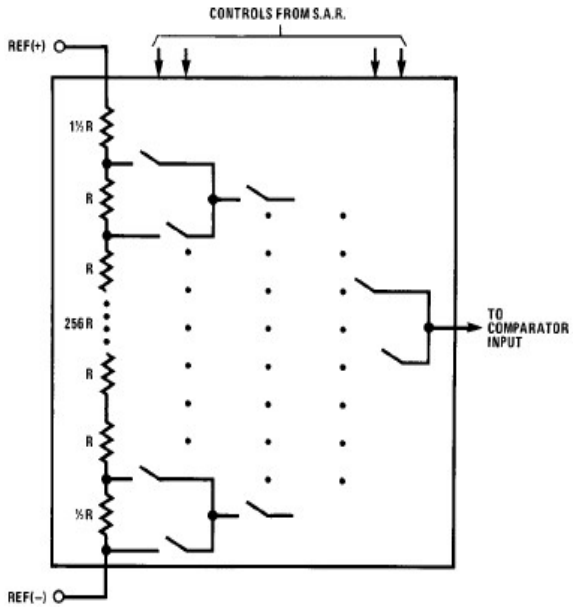


Figure 4.20.4: Resistor ladder and switch tree

The A/D converter's successive approximation register (SAR) is reset on the positive edge of the start conversion (SC) pulse. The conversion is begun on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion may be accomplished by tying the end of conversion (EOC) output to the SC input. If used in this mode, an external start conversion pulse should be applied after power up. End-of-conversion will go low between and 8 clock pulses after the rising

edge of start conversion. The most important section of the A/D converter is the comparator. It is this section which is responsible for the ultimate accuracy of the entire converter. It is also the comparator drift which has the greatest influence on the repeatability of the device. A chopper-stabilized comparator provides the most effective method of satisfying all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is then fed through a high gain AC amplifier and has the DC level restored. This technique limits the drift component of the amplifier since the drift is a DC component which is not passed by the AC amplifier. This makes the entire A/D converter extremely insensitive to temperature, long term drift and input offset errors.

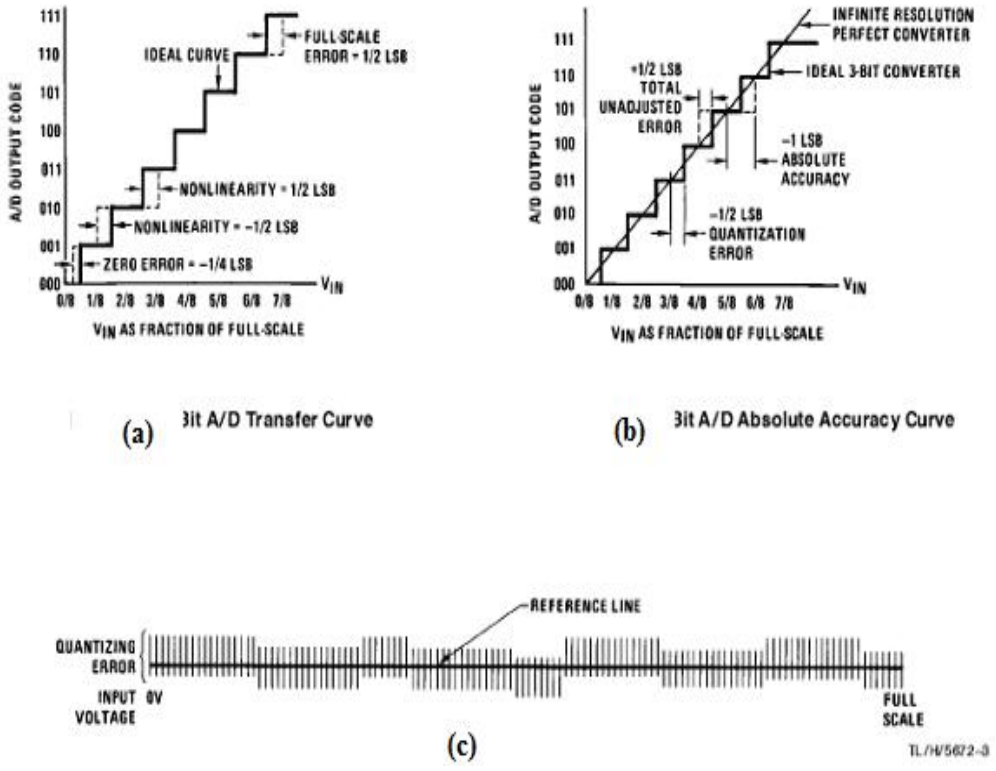


Figure 4.20.4.1 ADC analysis

4.20.5 ANALOG TO DIGITAL CONVERTER IN ARM:

The A/D converter present on some LPC2000 variants is a 10-bit successive approximation converter, with a conversion time of 2.44 microseconds or just shy of 410 KSps. The A/D converter has either 4 or 8 multiplexed inputs depending on the variant

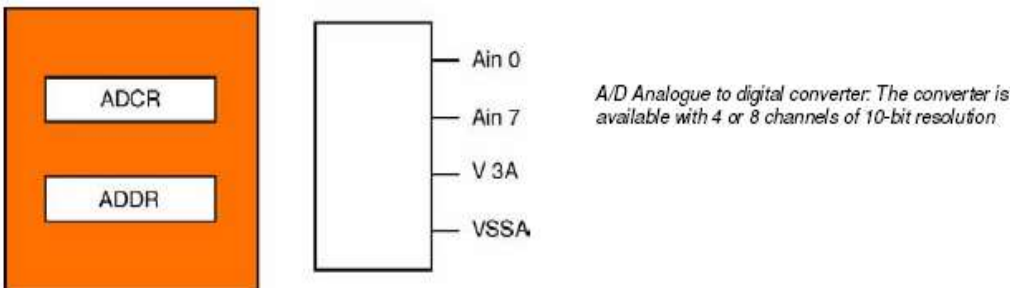


Figure 4.20.5: A/D 4 or 8 channel of 10 bit resolution

The A/D control register establishes the configuration of the converter & controls the start of conversion. The first step in configuring the converter is to set up the peripheral clock. As with all the other peripherals, the A/D clock is derived from the PCLK. This PCLK must be divided down to equal 4.5MHz. This is a maximum value and if PCLK cannot be divided down to equal 4.5MHz then the nearest value below 4.5MHz which can be achieved should be selected.

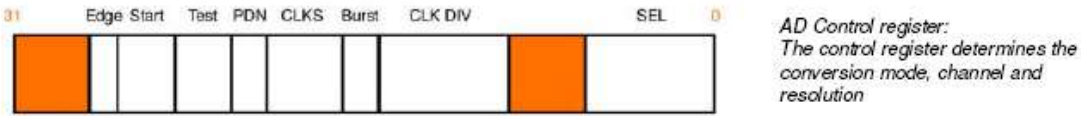


Figure 4.20.5.1: A/D control register

PCLK is divided by the value stored in the CLKDIV field plus one. Hence the equation for the A/D clock is as follows:

CLKDIV = (PCLK/Adclk) – 1

As well as being able to stop the clock to the A/D converter in the peripheral power down register, the A/D has the ability to fully power down. This reduces the overall power consumption and the on-chip

noise created by the A/D. On reset, the A/D is in power down mode, so as well as setting the clock rate the A/D must be switched on. This is controlled by the PDN bit in ADCR.

Logic one in this field enables the converter. Unlike other peripherals the A/D converter can make measurements of the external pins when they are configured as GPIO pins.

However, by using the pin select block to make the external pins dedicated to the A/D converter the overall conversion accuracy is increased Prior to a conversion the resolution of the result may be defined by programming the CLKS field. The A/D has a maximum resolution of 10 bits but can be programmed to give any resolution down to 3 bits. The conversion resolution is equal to the number of clock cycles per conversion minus one. Hence for a 10-bit result the A/D requires 11 ADCLK cycles and four for a 3-bit result. Once you have configured the A/D resolution, a conversion can be made. The A/D has two conversion modes, hardware and software. The hardware mode allows you to select a number of channels and then set the A/D running. In this mode a conversion is made for each channel in turn until the converter is stopped. At the end of each conversion the result is available in the A/D data register.

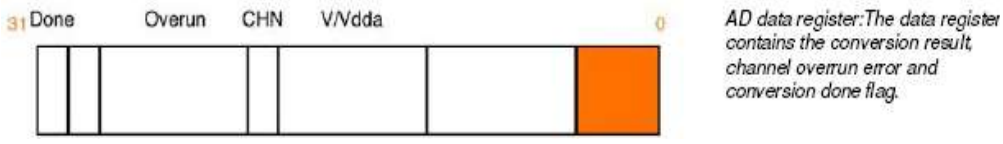


Figure 4.20.5.2 A/D data register

At the end of a conversion the done bit is set and an interrupt may also be generated. The conversion result is stored in the V/Vdd a field as a ratio of the voltage on the analogue channel divided by the voltage on the analogue power supply pin. The number of the channel for which the conversion was made is also stored alongside the result. This value is stored in the CHN field. Finally, if the result of a conversion is not read

before the next result is due, it will be overwritten by the fresh result and the OVERUN bit is set to one.

Start Bits	Conversion Trigger
0 0 0	Halted
0 0 1	Start Now
0 1 0	PO - 16
0 1 1	PO - 22
1 0 0	MAT 0-1
1 0 1	MAT 0-3
1 1 0	MATT 1-0
1 1 1	MAT 1-1

Table 4.20.5.2: The A/D may be started by a software event or it may be started by several hardware triggers

The A/D has a second software conversion mode. In this case, a channel is selected for conversion using the SEL bits and the conversion is started under software control by writing 0x01 to the START field. This causes the A/D to perform single conversion and store the results in the ADDR in the same fashion as the hardware mode. The end of conversion can be signaled by an interrupt, or by polling the done bit in the ADDR. In the software conversion mode it is possible to start a conversion when a match event occurs on timer zero or timer one. Or when a selected edge occurs on P0.16 or P0.22, the edge can be rising or falling, as selected by the EDGE field in the ADCR.

4.21 UART IN ARM:

The LPC2xxx devices currently have two on-chip UARTS. They are both identical to use, except UART1 has additional modem support. Both peripherals conform to the “550 industry standard” specification. Both have a built-in baud rate generator and 16 byte transmit and receive FIFOs.

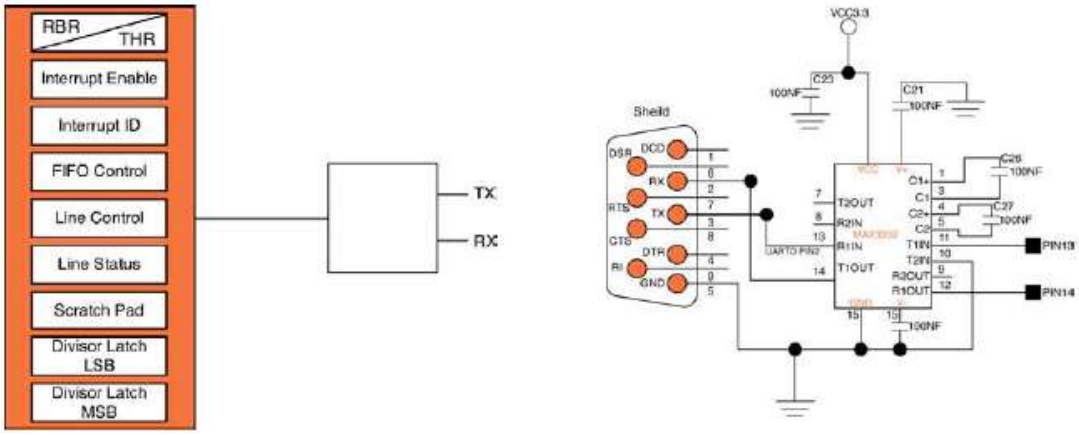


Figure 4.21: max232

First the pin select block must be programmed to switch the processor pins from GPIO to the UART functions. Next the UART line control register is used to configure the format of the transmitter data character.

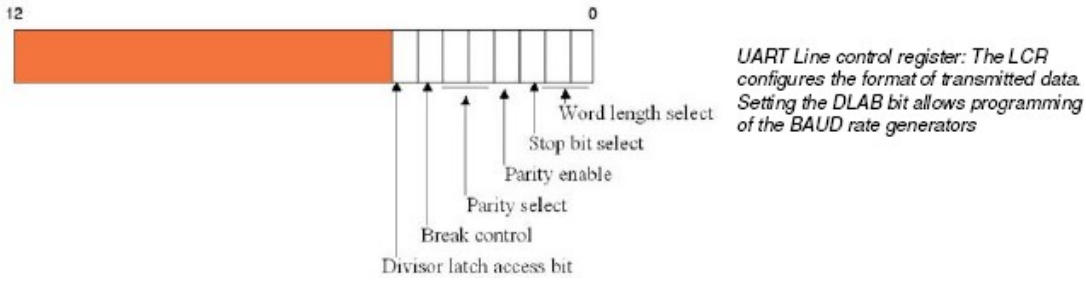


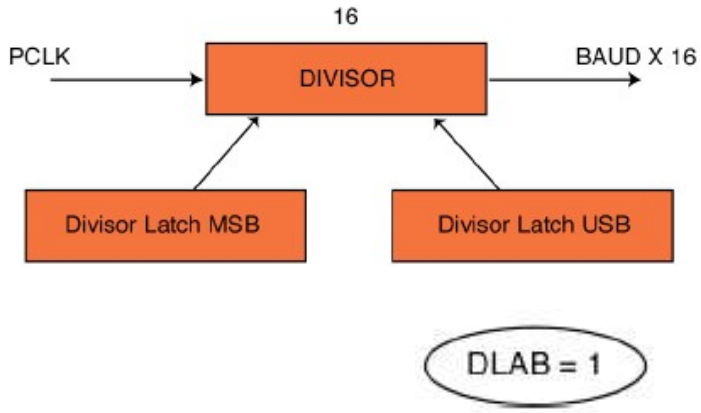
Figure 4.21.1: UART line control register

The character format is set to 8 bits, no parity and one stop bit. In the LCR there is an additional bit called DLAB, which is the divisor latch access bit. In order to be able to program the baud rate generator this bit must be set.

The baud rate generator is a 16-bit pre scalar which divides down Pclk to generate the UART clock which must run at 16 times the baud rate. Hence the formula used to calculate the UART baud rate is:
Divisor = Pclk/16 x BAUDIn our case at 15MHz:

Divisor = 15,000,000/16 x 9600 = (approx) 97 or 0x62

This gives a true baud rate of 9665. Often it is not possible to get an exact baud rate for the UARTs however they will work with up to around a 5% error in the bit timing. So, you have some leeway with the UART timings if you need to adjust the Pclk to get exact timings on other peripherals such as the CAN bit timings. The divisor value is held in two registers, Divisor latch MSB (DLM) and Divisor latch LSB (DLL). The first eight bits of both registers holds each half of the divisor as shown below. Finally, the DLAB bit in the LCR register must be set back to zero to protect the contents of the divisor registers.



UART baud rate: The UART clock frequency must be 16 times the required BAUD rate. This is derived by dividing Pclk by a 16-bit divisor register.

Figure 4.21.1.1: UART baud rate

4.21.1 HARDWARE TOOLS:

The Hardware includes LPC2129microcontroller, LV-Max sonar ultrasonic sensor, MCP2551.The detailed study of these components is given below.

4.21.2 LPC 2129:

DEVICE DESCRIPTION:

The LPC2119/2129/2194/2292/2294 is based on a 16/32 bit ARM7TDMI-STM CPU with real-time emulation and embedded trace support, together with 128/256 kilobytes (kB) of embedded high speed flash memory. A 128-bit wide internal memory interface and unique accelerator architecture enable 32-bit code execution at maximum clock rate.

For critical code size applications, the alternative 16-bit Thumb Mode reduces code by more than 30% with minimal performance penalty. With their compact 64 and 144 pin packages, low power consumption, various 32-bit timers, combination of 4-channel 10-bit ADC and 2/4 advanced CAN channels or 8-channel 10-bit ADC and 2/4 advanced CAN channels (64 and 144 pin packages respectively), and up to 9 external interrupt pins these microcontrollers are particularly suitable for industrial control, medical systems, access control and point-of-sale. Number of available GPIOs goes up to 46 in 64 pin package. In 144 pin packages number of available GPIOs tops 76 (with external memory in use) through 112 (single-chip application). Being equipped wide range of serial communications interfaces, they are also very well suited for communication gateways, protocol converters and embedded soft modems as well as many other general-purpose applications.

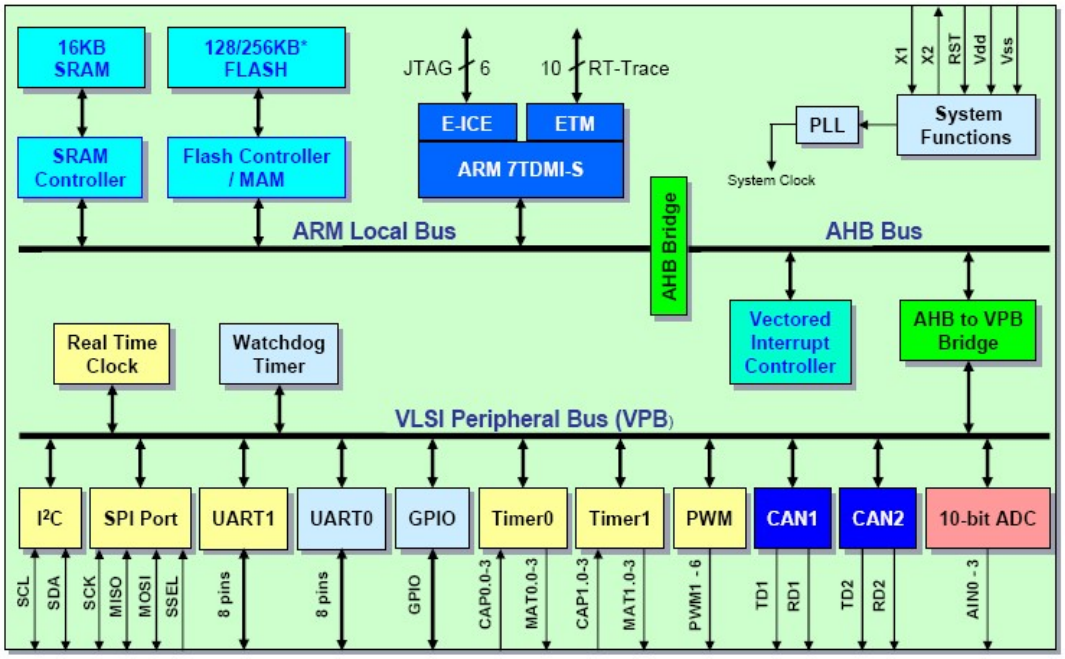


Figure 4.21.2: LPC2129 internal Block diagram

Number of Components Used: Two KNOWX ARM BOARDS

4.21.3 MCP2551

DEVICE DESCRIPTION:

The MCP2551 is a high-speed CAN, fault-tolerant device that serves as the interface between a CAN protocol controller and the physical bus. The MCP2551 provides differential transmit and receive capability for the CAN protocol controller and is fully compatible with the ISO-11898 standard, including 24V requirements. It will operate at speeds of up to 1Mb/s. typically; each node in a CAN system must have a device to convert the digital signals generated by a CAN controller to signals suitable for transmission over the bus cabling. It also provides a buffer between the CAN controller and the high-voltage spikes that can be generated on the CAN bus by outside sources (EMI, ESD, electrical transients, etc.).

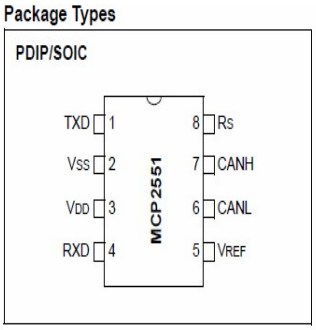


Figure (a)

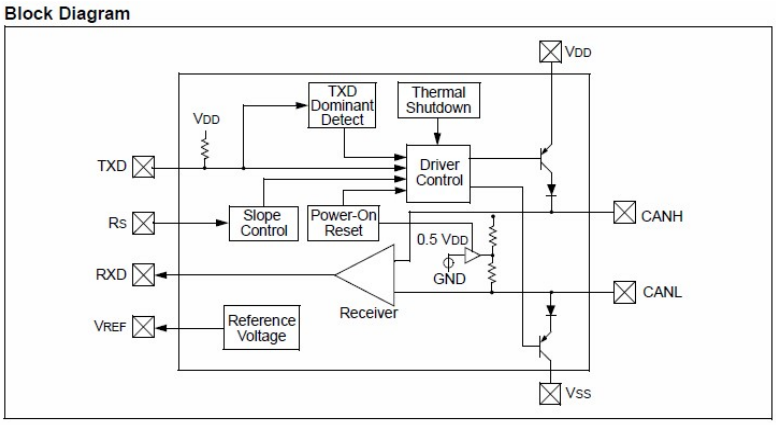


Figure (b)

Figure 4.21.3: (a) IC package and (b) block diagram

TRANSMITTER FUNCTION:

The CAN bus has two states: Dominant and Recessive. A dominant state occurs when the differential voltage between CANH and CANL is greater than a defined voltage (e.g., 1.2V). A recessive state occurs when the differential voltage is less than a defined voltage (typically 0V). The dominant and recessive states correspond to the low and high state of the TXD input pin, respectively. However, a dominant state initiated by another CAN node will override a recessive state on the CAN bus.

MAXIMUM NUMBER OF NODES:

The MCP2551 CAN outputs will drive a minimum load of 45Ω, allowing a maximum of 112 nodes to be connected (given a minimum differential input resistance of 20 kΩ and a nominal termination resistor value of 120Ω).

RECEIVER FUNCTION:

The RXD output pin reflects the differential bus voltage between CANH and CANL. The low and high states of the RXD output pin correspond to the dominant and recessive states of the CAN bus, respectively.

4.21.4 KALMAN FILTER:

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

In statistics, the Kalman filter is a mathematical method named after Rudolf E. Kalman. Its purpose is to use measurements that are observed over time that contain noise (random variations) and other inaccuracies, and produce values that tend to be closer to the true

values of the measurements and their associated calculated values. The Kalman filter has many applications in technology and is an essential part of the development of space and military technology. Perhaps the most commonly used type of very simple Kalman filter is the phase-locked loop, which is now ubiquitous in FM radios and most electronic communications equipment. Extensions and generalizations to the method have also been developed.

The Kalman filter produces estimates of the true values of measurements and their associated calculated values by predicting a value, estimating the uncertainty of the predicted value, and computing a weighted average of the predicted value and the measured value. The most weight is given to the value with the least uncertainty. The estimates produced by the method tend to be closer to the true values than the original measurements because the weighted average has a better estimated uncertainty than either of the values that went into the weighted average.

From a theoretical standpoint, the Kalman filter is an algorithm for efficiently doing exact inference in a linear dynamical system, which is a Bayesian model similar to a hidden Markov model but where the state space of the latent variables is continuous and where all latent and observed variables have a Gaussian distribution (often a multivariate Gaussian distribution).

The Kalman filter uses a system's dynamics model (i.e., physical laws of motion), known control inputs to that system, and measurements (such as from sensors) to form an estimate of the system's varying quantities (its state) that is better than the estimate obtained by using any one measurement alone. As such, it is a common sensor fusion algorithm.

All measurements and calculations based on models are estimates to some degree. Noisy sensor data, approximations in the equations that describe how a system changes and external factors that are not

accounted for introduce some uncertainty about the inferred values for a system's state. The Kalman filter averages a prediction of a system's state with a new measurement using a weighted average. The purpose of the weights is that values with better (i.e., smaller) estimated uncertainty is "trusted" more. The weights are calculated from the covariance, a measure of the estimated uncertainty of the prediction of the system's state. The result of the weighted average is a new state estimate that lies in between the predicted and measured state, and has a better estimated uncertainty than either alone. This process is repeated every time step, with the new estimate and its covariance informing the prediction used in the following iteration. This means that the Kalman filter works recursively and requires only the last "best guess" - not the entire history - of a system's state to calculate a new state.

When performing the actual calculations for the filter (as discussed below), the state estimate and covariance's are coded into matrices to handle the multiple dimensions involved in a single set of calculations. This allows for representation of linear relationships between different state variables (such as position, velocity, and acceleration) in any of the transition models or covariances.

The Kalman filter is used in sensor fusion and data fusion. Typically real time systems produce multiple sequential measurements rather than making a single measurement to obtain the state of the system. These multiple measurements are then combined mathematically to generate the system's state at that time instant.

As an example application, consider the problem of determining the precise location of a truck. The truck can be equipped with a GPS unit that provides an estimate of the position within a few meters. The GPS estimate is likely to be noisy; readings 'jump around' rapidly, though always remaining within a few meters of the real position. The truck's position can also be estimated by integrating its speed and direction over time, determined by keeping track of the amount the

accelerator is depressed and how much the steering wheel is turned. This is a technique known as dead reckoning. Typically, dead reckoning will provide a very smooth estimate of the truck's position, but it will drift over time as small errors accumulate. Additionally, the truck is expected to follow the laws of physics, so its position should be expected to change proportionally to its velocity.

In this example, the Kalman filter can be thought of as operating in two distinct phases: predict and update. In the prediction phase, the truck's old position will be modified according to the physical laws of motion (the dynamic or "state transition" model) plus any changes produced by the accelerator pedal and steering wheel. Not only will a new position estimate be calculated, but a new covariance will be calculated as well. Perhaps the covariance is proportional to the speed of the truck because we are more uncertain about the accuracy of the dead reckoning estimate at high speeds but very certain about the position when moving slowly. Next, in the update phase, a measurement of the truck's position is taken from the GPS unit. Along with this measurement come some amount of uncertainty, and its covariance relative to that of the prediction from the previous phase determines how much the new measurement will affect the updated prediction. Ideally, if the dead reckoning estimates tend to drift away from the real position, the GPS measurement should pull the position estimate back towards the real position but not disturb it to the point of becoming rapidly changing and noisy.

The iterative predictor-corrector nature of the Kalman filter can be helpful, because at each time instance only one constraint on the state variable need be considered. This process is repeated, considering a different constraint at every time instance. All the measured data are accumulated over time and help in predicting the state.

Video can also be pre-processed, perhaps using a segmentation technique, to reduce the computation and hence latency.

The Kalman filter is a recursive estimator. This means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state. In contrast to batch estimation techniques, no history of observations and/or estimates is required in what follows, the notation \hat{\textbf{x}}_{n|m} represents the estimate of \textbf{x} at time n given observations up to, and including at time m.

The state of the filter is represented by two variables:

* \hat{\textbf{x}}_{k|k}, the a posteriori state estimate at time k given observations up to and including at time k;

* \textbf{P}_{k|k}, the a posteriori error covariance matrix (a measure of the estimated accuracy of the state estimate).

The Kalman filter can be written as a single equation, however it is most often conceptualized as two distinct phases: Predict and Update. The predict phase uses the state estimate from the previous timestep to produce an estimate of the state at the current timestep. This predicted state estimate is also known as the a priori state estimate because, although it is an estimate of the state at the current time step, it does not include observation information from the current time step. In the update phase, the current a priori prediction is combined with current observation information to refine the state estimate. This improved estimate is termed the a posteriori state estimate.

Typically, the two phases alternate, with the prediction advancing the state until the next scheduled observation, and the update incorporating the observation. However, this is not necessary; if an observation is unavailable for some reason, the update may be skipped and multiple prediction steps performed. Likewise, if multiple independent observations are available at the same time, multiple update steps may be performed (typically with different observation matrices Hk).

Predict

Predicted (a priori) state estimate

$$\hat{\textbf{x}}_{k|k-1} = \textbf{F}_{k}\hat{\textbf{x}}_{k-1|k-1} + \textbf{B}_{k} \textbf{u}_{k}$$

Predicted (a priori) estimate covariance

$$\textbf{P}_{k|k-1} = \textbf{F}_{k} \textbf{P}_{k-1|k-1} \textbf{F}_{k}^{\text{T}} + \textbf{Q}_{k}$$

[edit] Update

Innovation or measurement residual

$$\tilde{\textbf{y}}_k = \textbf{z}_k - \textbf{H}_k\hat{\textbf{x}}_{k|k-1}$$

Innovation (or residual) covariance $\textbf{S}_k = \textbf{H}_k \textbf{P}_{k|k-1} \textbf{H}_k^\text{T} + \textbf{R}_k$

Optimal Kalman gain $\textbf{K}_k = \textbf{P}_{k|k-1}\textbf{H}_k^\text{T}\textbf{S}_k^{-1}$

Updated (a posteriori) state estimate $\hat{\textbf{x}}_{k|k} = \hat{\textbf{x}}_{k|k-1} + \textbf{K}_k\tilde{\textbf{y}}_k$

Updated (a posteriori) estimate covariance $\textbf{P}_{k|k} = (I - \textbf{K}_k \textbf{H}_k) \textbf{P}_{k|k-1}$

The formula for the updated estimate and covariance above is only valid for the optimal Kalman gain. Usage of other gain values requires a more complex formula found in the derivations section.

4.22 DC MOTOR:

A DC motor is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or

electronic; to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line. DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances.

Figure 4.22: DC motor

4.23 RELAY:

Relays can be used for switching as well as protection application. A relay is used to switch a circuit such that current through it can be diverted from present circuit to another. Manual operation for switching a relay is performed through push buttons and other conventional switches. Protective relays are used to ensure the smooth operation of any power system such that they isolate the particular circuit or generate the alarm whenever parameters like voltage or current exceeds their limits. Therefore the principal function of the relay is to make or break the circuit in switching and protection applications. A variety class of relays is found in several applications.

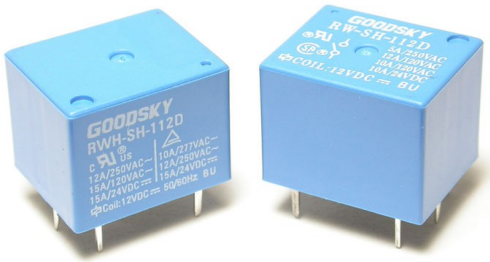


Figure 4.23: Relay

CHAPTER 5

POWER SUPPLY

POWER SUPPLY

5.1 POWER SUPPLY:

The power supplies are designed to convert high voltage AC mains electricity to a suitable low voltage supply for electronic circuits and other devices. A power supply can by broken down into a series of blocks, each of which performs a particular function. The input to the circuit is applied from the regulated power supply. The a.c input i.e., 230V from the mains supply is step down by the transformer to 12V and is fed to a rectifier. The output obtained from the rectifier is a pulsating d.c voltage. So in order to get a pure d.c voltage, the output voltage from the rectifier is fed to a filter to remove any a.c components present even after rectification. Now, this voltage is given to a voltage regulator to obtain a pure constant dc voltage

A d.c power supply which maintains the output voltage constant irrespective of an a.c mains fluctuations or load variations is known as "Regulated D.C Power Supply". For example a 5V regulated power supply system as shown below.

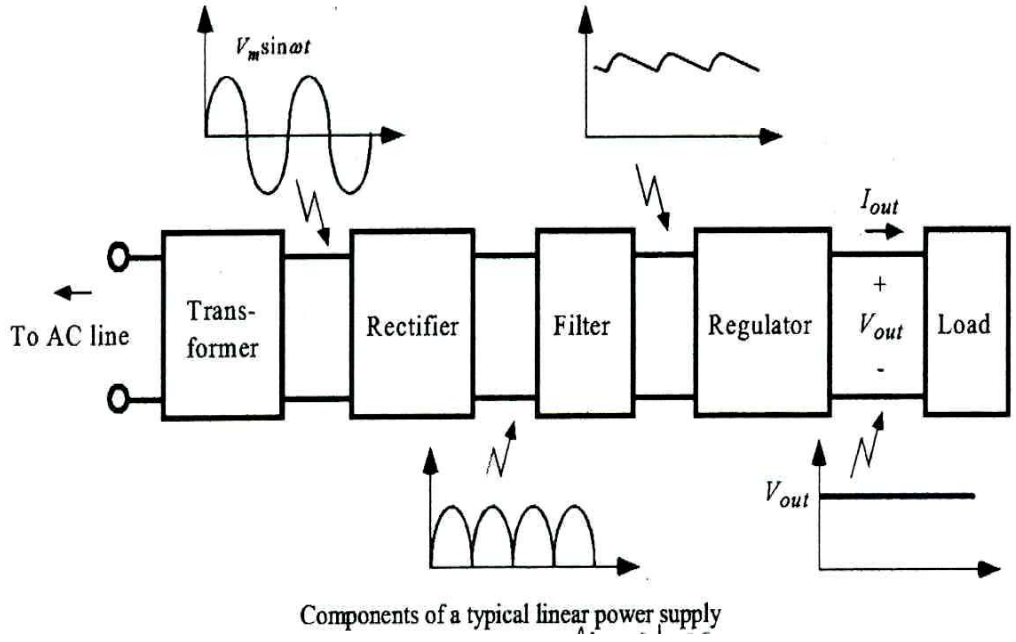


Fig 5.1: linear power supply

5.2 TRANSFORMER:

A transformer is an electrical device which is used to convert electrical power from one electrical circuit to another without change in frequency. 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. Step-up transformers increase in output voltage, step-down transformers decrease in output voltage. Most power supplies use a step-down transformer to reduce the dangerously high mains voltage 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's 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.

Figure 5.2: An Electrical Transformer

5.3 RECTIFIER:

A circuit which is used to convert AC to DC is known as RECTIFIER. The process of conversion AC to DC is called “rectification”. Rectifiers have many uses, but are often found serving as components of DC power supplies and high-voltage direct current power transmission systems.

5.4 FULL-WAVE RECTIFIER:

From the above comparison we came to know that full wave bridge rectifier as more advantages than the other two rectifiers. So, in our project we are using full wave bridge rectifier circuit.

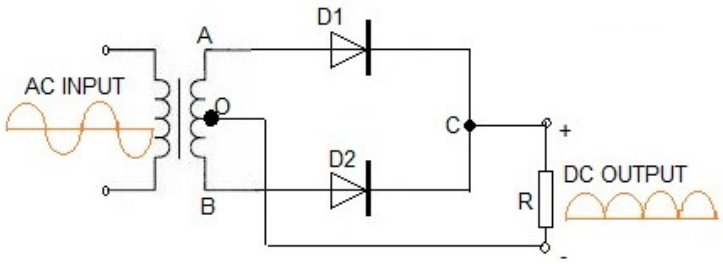


Figure 5.4: Full wave rectifier

5.4.1 BRIDGE RECTIFIER:

A bridge rectifier makes use of four diodes in a bridge arrangement to achieve full-wave rectification. This is a widely used configuration, both with individual diodes wired as shown and with single component bridges where the diode bridge is wired internally.

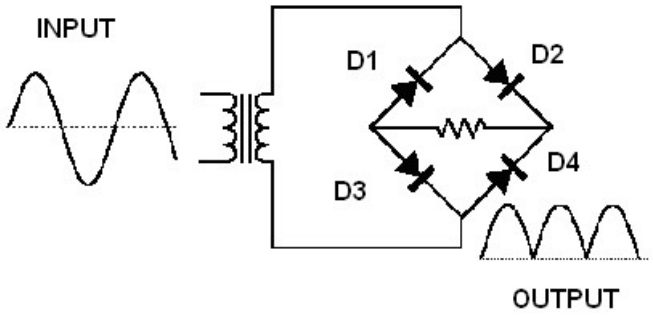


Figure 5.4.1: Bridge rectifier

A bridge rectifier makes use of four diodes in a bridge arrangement as shown in above figure to achieve full-wave rectification. This is a widely used configuration, both with individual diodes wired as shown and with single component bridges where the diode bridge is wired internally.

OPERATION:

During positive half cycle of secondary, the diodes D2 and D3 are in forward biased while D1 and D4 are in reverse biased as shown in the fig(b). The current flow direction is shown in the fig with dotted arrows.

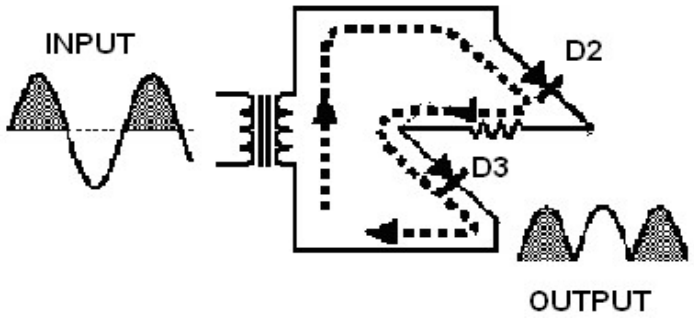


Figure 5.4.1.2: positive half cycle of bridge rectifier

During negative half cycle of secondary voltage, the diodes D1 and D4 are in forward biased while D2 and D3 are in reverse biased as shown in the fig(c).

The current flow direction is shown in the fig (c) with dotted arrows.

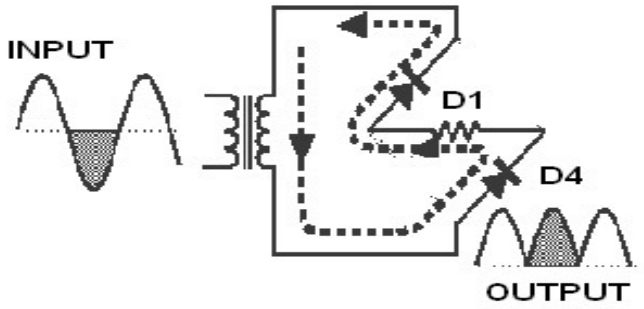


Figure 5.4.1.3: Negative half cycle of bridge rectifier

5.5 FILTER:

A Filter is a device which removes the AC component of rectifier output but allows the DC component to reach the load. Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the DC. Output received from this filter is constant until the mains voltage and load is maintained constant. However, if either of the two is varied DC. voltage received at this point changes. Therefore a regulator is applied at the output stage.

5.6 REGULATOR:

As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. Voltage regulator ICs is available with fixed (typically 5, 12 and 15V) or variable output voltages. The maximum current they can pass also rates them. 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. The LM7805 is simple to use. You simply connect the positive lead of your unregulated DC power supply (anything from 9VDC to 24VDC) to the Input pin, connect the negative lead to the Common pin and then when you turn on the power, you get a 5 volt supply from the output pin.

Figure 5.6: Voltage regulator

CHAPTER 6

TYPES OF SENSORS

TYPES OF SENSORS

6.1 INTRODUCTION OF SENSORS:

A sensor is a device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure, or any one of a great number of other environmental phenomena. The purpose of a sensor is to respond to some kind of an input physical property and to convert it into an electrical signal that is compatible with electronic circuits. We may say that a sensor is a translator of a generally nonelectrical value into an electrical value. The electrical value means a signal, which can be channeled, amplified, and modified by electronic devices.

The sensor's output signal may be in the form of voltage, current, or charge. These may be further described in terms of amplitude, polarity, frequency, phase, or digital code. This set of characteristics is called the output signal format. Therefore, a sensor has input properties (of any kind) and electrical output properties.

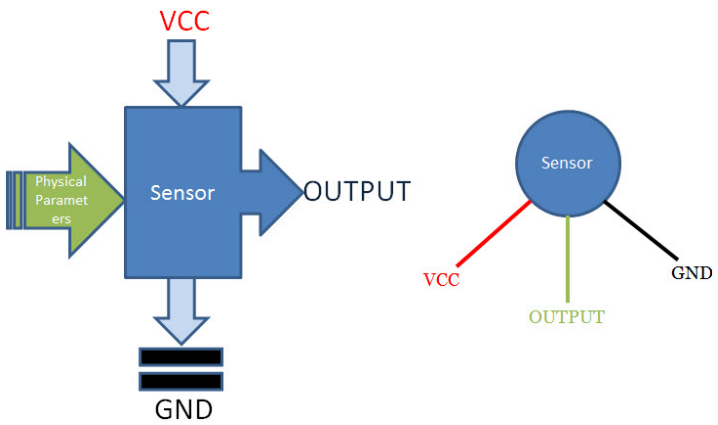


Figure 6.1: Sensor

Any sensor is an energy converter. No matter what you try to measure, you always deal with energy transfer from the object of measurement to the sensor. The process of sensing is a particular case of information transfer, and any transmission of information requires transmission of energy. Wireless sensor networks have been applied to

many applications since emerging. Among them, one of the most important applications is sensor data collection, where sensed data are continuously collected at all or some of the sensor nodes and forwarded through wireless communications, each sensor node is powered by a battery and uses wireless communications. This results in the small size of a sensor node and makes it easy to be attached at any location with little disturbances to the surrounding environment. Sensors detect the presence of energy, changes in or the transfer of energy. Sensors detect by receiving a signal from a device such as a transducer, then responding to that signal by converting it into an output that can easily be read and understood. Typically sensors convert a recognized signal into an electrical analog or digital – output that is readable. In other words, a transducer converts one form of energy into another while the sensor that the transducer is parts of converts the output of the transducer to a readable format.

6.2 RADIO FREQUENCY IDENTIFICATION (RFID):

RFID uses electromagnetic fields to automatically identify and track tags attached to objects. The tags contain electronically stored information. Passive tags collect energy from a nearby RFID reader's interrogating radio waves. Active tags have a local power source such as a battery and may operate at hundreds of meters from the RFID reader. Unlike a barcode, the tag need not be within the line of sight of the reader, so it may be embedded in the tracked object. RFID is one method for Automatic Identification and Data Capture (AIDC).

Furthermore, it is important to observe the role allocation of initiator and target. The initiator is the one who wishes to communicate and starts the communication. The target receives the initiator’s communication request and sends back a reply. This concept prevents the target from sending any data without first receiving a message. Regarding the passive communication mode, the passive device acts always as NFC target. Here the active device is the initiator, responsible for generating the radio field.

Figure 6.2: RFID cards and chip

6.3 TEMPERATURE SENSOR:

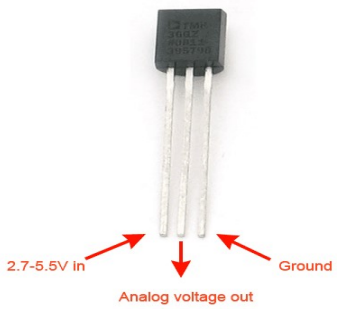


Figure 6.3: Temperature sensor

The LM35 Linear Temperature Sensor can be used to detect ambient air temperature. Sensitivity is 10mV per degree Celsius. It includes thermocouples, platinum resistance, thermal resistance and temperature semiconductor chips measurement thermocouples

6.4 SEAT BELT DETECTION:

The basic concept of IR (infrared) seat belt detection is to transmit the IR signal (radiation) in a direction and a signal is received at the IR receiver when the IR radiation bounces back from a surface of the object.

Here in the figure the object we used is seat belt, the IR LED transmits the IR signal on to the belt and the signal is reflected back from the surface of the object. The reflected signals are received by an IR receiver. The IR receiver can be a photodiode / phototransistor or a readymade module which decodes the signal.

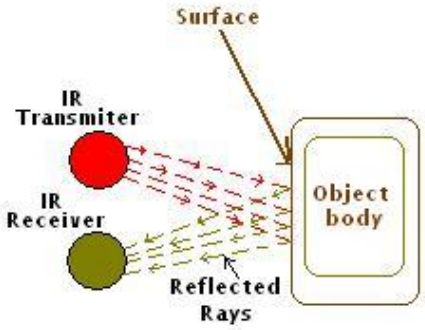


Figure 6.4: Seat belt detection:

6.5 FUEL LEVEL SENSOR:

Fuel Level Sensor is designed to measure fuel level in the vehicle's fuel tank and fuel storages. The sensor is connected to any AVL-device reporting the measured fuel value. It is also possible to monitor and control fuel data using special software which allows you to create different reports on transport fuel consumption, as well as notify in case of violations.The sensor measures fuel level in the tank and generates an output signal to forward it to a vehicle tracking device. Tracking device records and processes the sensor data for further transmission to the LCD.

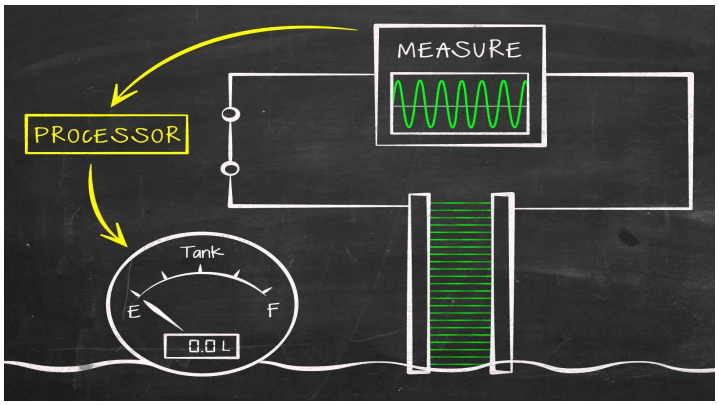


Figure 6.5: Fuel level sensor

6.6 BATTERY LEVEL INDICATOR:

Battery level indicator indicates the status of the battery by displaying the level of the voltage. The indicator is connected to the

vehicle battery so that the battery level is measured. If the level of the battery is less than 30% then a buzzer is used to alert the driver.

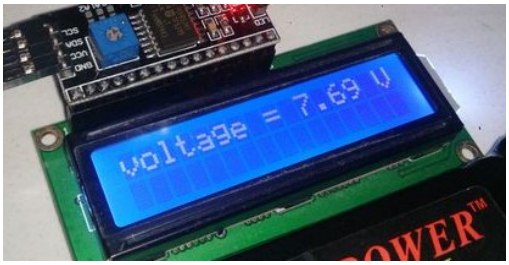


Figure 6.6: Battery level indicator

6.7 PRESSURE SENSOR:

Pressure sensor measures the air of the tyre. Two conductors are placed in the tyre so that if the air in the tyre is high the conductors are separated .If the air is being reduced then the two conductors get in contact and the current passes through it so by that the warning sign is showed in the LCD.

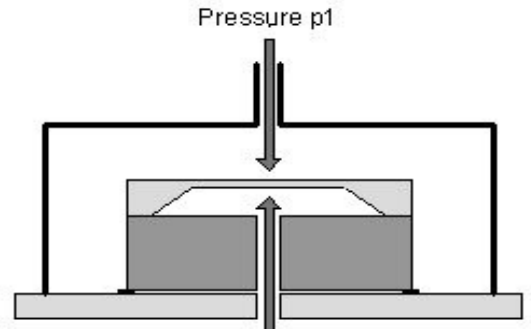


Figure 6.7: Pressure sensor

CHAPTER 7

FIRMWARE AND SOFTWARE IMPLEMENTATION

FIRMWARE AND SOFTWARE IMPLEMENTATION

This chapter briefly explains about the firmware implementation of the project and the code. The required software tools are discussed below.

7.1 SOURCE CODE

```
#include <LPC21xx.H>
void CAN1_TX(unsigned int);
void CAN2_RX(void);

void Delay(unsigned int itime)          //Here we are generating        ms
delay
{
        unsigned int i,j;
        for(i=0;i<itime;i++)
                for(j=0;j<1500;j++);
}
//=============================================

void lcd_init()
{
          lcd_cmd(0x20);
          lcd_cmd(0x28);
          lcd_cmd(0x0C);          //Display on/off control
          lcd_cmd(0x01);          //Clear Display
          lcd_cmd(0x06);          //Entry Mode set
          lcd_cmd(0x80);          //sets the display position to starting

}

void lcd_cmd(unsigned char cmd)
{
  unsigned char temp;
  temp = cmd;
  cmd &= 0xF0;
  cmd = cmd << 6;                       //left shifted by 6 times
  IOCLR0 = 0x00003C00;                          //Clear the Data Pins
  IOSET0 = cmd;
```

```
  IOCLR1 = 0x03000000;                  //Register Select Clear For
Command
  IOSET1 = 0x02000000;                  //Enable Set
  Delay(50);
  IOCLR1 = 0x02000000;                  //Enable Clear
  Delay(50);
  cmd  = temp;
  cmd &= 0x0F;
  cmd = cmd << 10;                    //left shifted by 10 times
  IOCLR0 = 0x00003C00;                  //Clear the Data Pins
  IOSET0 = cmd;
  IOCLR1 = 0x03000000;                  //Register Select Clear For
Command
  IOSET1 = 0x02000000;                  //Enable Set
  Delay(50);
  IOCLR1 = 0x02000000;                  //Enable Clear
  Delay(50);
}
//=============================================
//THIS IS THE FUNCTION FOR WRITE THE DATAS IN DATA REG
void lcd_char(unsigned int ch)
{
  unsigned char temp;
  temp = ch;
  ch  &= 0xF0;
  ch = ch << 6;                         //left shifted by 6 times
  IOCLR0 = 0x00003C00;                  //Clear the Data Pins
  IOSET0 = ch;
  IOCLR1 = 0x03000000;                  //Register Select Set For
Data
  IOSET1 = 0x03000000;                  //Enable Set
  Delay(5);
  IOCLR1 = 0x02000000;                  //Enable Clear
  Delay(5);
  ch   = temp;
  ch  &= 0x0F;
  ch   = ch << 10;                    //left shifted by 10 times
  IOCLR0 = 0x00003C00;                  //Clear the Data Pins
  IOSET0 = ch;
  IOCLR1 = 0x03000000;                  //Register Select Set For
Data
  IOSET1 = 0x03000000;                  //Enable Set
  Delay(5);
```

```
  IOCLR1 = 0x02000000;                        //Enable Clear
  Delay(5);
}
//=============================================
//DISPLAY STRING OF DATA ON LCD
void lcd_print(unsigned char *str)
{
        while(*str)
          lcd_char(*str++);                 //Display String Data On LCD
}
/*****************************************************/

void Delay1(unsigned int itime)
{
  unsigned int i,k=0;
  for(i=0;i<itime;i++)
        for(k=0;k<1000;k++);
}

/*****************************************************/


int main(void)
{
 unsigned int ch;

 IODIR0=0x00003C00;
 IODIR1=0x03000000;
  PINSEL1=0x00054000;

lcd_init();
 lcd_print("CAN Health ");
 lcd_cmd(0xC0);
 lcd_print(" monitoring SYSTEM");
 Delay1(2000);
 Delay1(2000);
 Delay1(2000);
 Delay1(2000);
 lcd_cmd(0x01);
  /*****************CAN1 intialization***************/

    C1MOD=1;/*Reset CAN1 controller*/
    C1BTR=0x001C001D;/*Set baud Rate for CAN*/
```

```
   C1MOD=0;/*Enable CAN1 controller*/

        lcd_cmd(0x01);
         lcd_print("CAN1 ENABLED");
         Delay1(200);
         Delay1(200);
         Delay1(200);

/******************CAN2 intialization***************/

   C2MOD=1;/*Reset CAN2 controller*/
   C2BTR=0x001C001D;/*Set baud Rate for CAN2*/
       C2MOD=0;/*Enable CAN2 controller*/
       lcd_cmd(0x01);

         lcd_print("CAN2 ENABLED");


         Delay1(2000);
         Delay1(2000);
         Delay1(2000);
         lcd_cmd(0x01);
        while(1)
         {
                if(IOPIN0 & 0x20000000 )                //po.29
                ch=0x01;
       else
           ch=0x02;
                 if(ch==0x01 || ch==0x02)
                 {
                 CAN1_TX(ch);
        CAN2_RX();
                  }
                 }
        }

                  void CAN1_TX(unsigned int ch)
   {
        if(C1SR & 0x00000004)
        {
        C1TFI1=0x00040000;/*Set TX data length*/
    C1TID1=C1RID;/*Set TX Identifier*/
```

```
C1TDA1 =ch;   /*Load the Data to TX1 buffer*/
C1CMR=0x21; /*send the Data*/
    Delay1(2000);

    }
}
          void CAN2_RX()
    {
    int value;
    C2CMR=0X04;
    lcd_cmd(0x01);
    value=C2RDA;
    if(value==0x01)
    {
             lcd_cmd(0x01);
              lcd_cmd(0x80);
              lcd_print(" Fuel is Empty");
                  lcd_cmd(0xC0);
              lcd_print("Press ok button   ");

                  Delay1(2000);
                  Delay1(2000);
                  Delay1(2000);

             }
             if(value==0x02)

  {
              lcd_cmd(0x01);
                  lcd_cmd(0x80);
              lcd_print(" Line clear");
                  lcd_cmd(0xC0);
              lcd_print("Drunken press ok  ");

                  Delay1(2000);
                  Delay1(2000);
                  Delay1(2000);
   }
   }
```

7.2 SOFTWARE DEVELOPMENT:

In this module we will be using an Integrated Development Environment from Keil Electronic. This IDE is called u VISION (pronounced "Micro Vision") and versions already exist for other popular microcontrollers including the 8051 and the Infineon C16X family. u VISION successfully integrates project management, editor, compiler and debugger in one seamless front-end.

7.2.1 THE COMPILER:

The u VISION development environment can be used with several different compiler tools. These include the ARM ADS compiler, the GNU compiler and Keil's own ARM compiler.

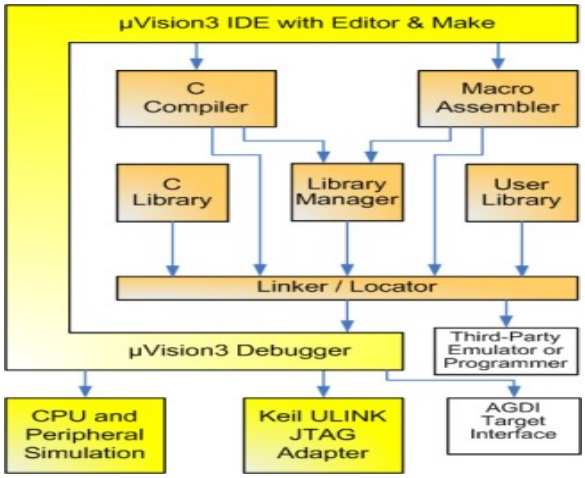
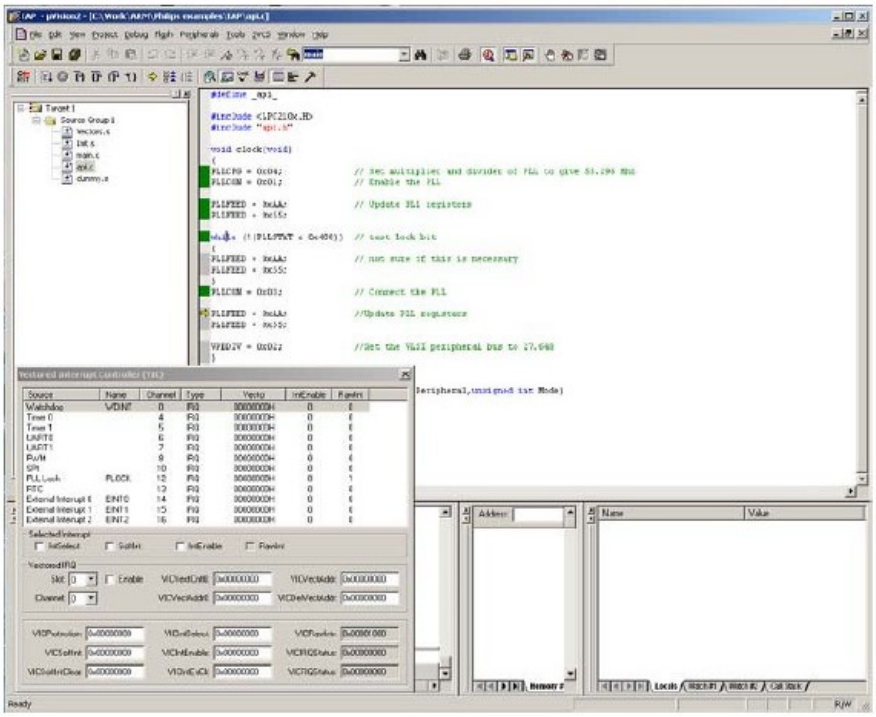


Figure 7.2.1: keil software internal stages

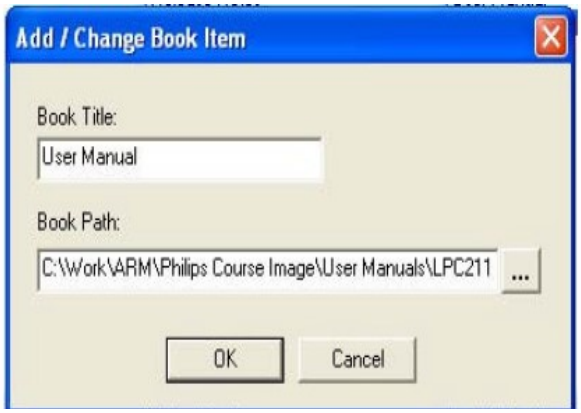
The Keil u VISION ("u VISION") IDE is designed to support several compilers, the Gnu C compiler, The ARM development suite and the Keil ARM compiler. Before compiling make sure you have the GNU compiler selected. This is done by activating the project workspace; right clicking and selecting manage components. In this dialog select the Folders/extensions tab and make sure the "Keil ARM tools" box is selected

KEIL TOOLSET

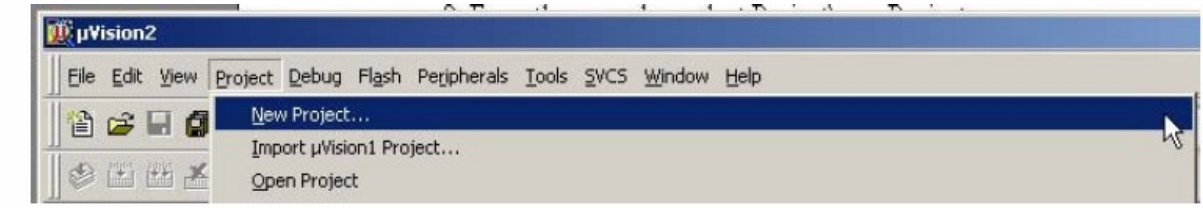
STEP1: Open Kiel UVISION3.

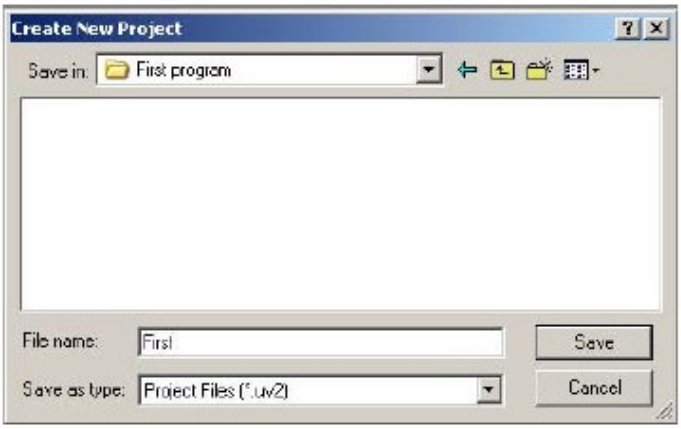


Step2: Add user manual.

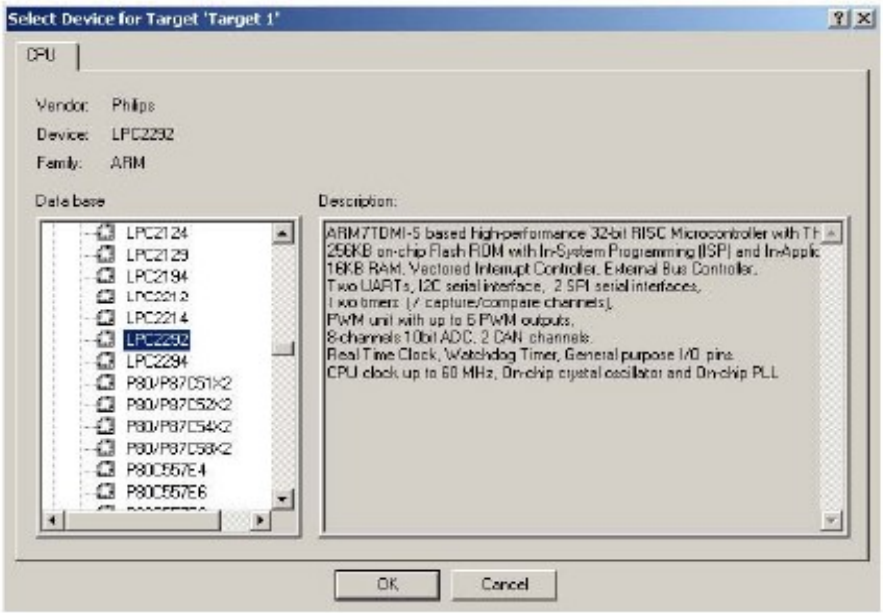


STEP3: Create new file

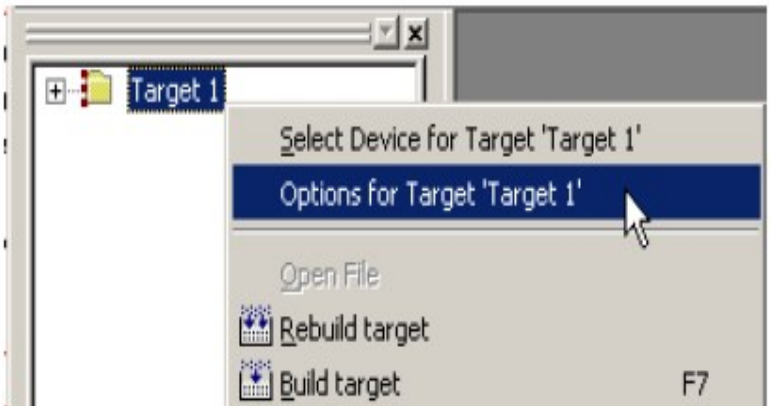




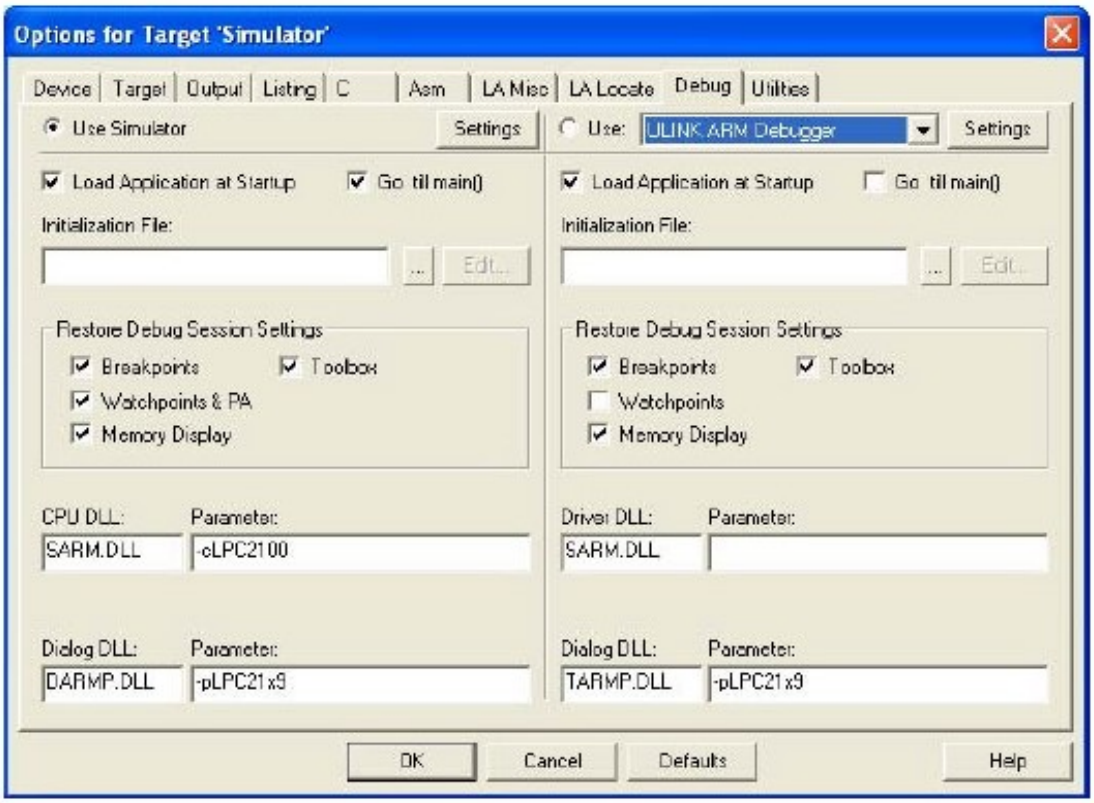
STEP4: Select LPC2129

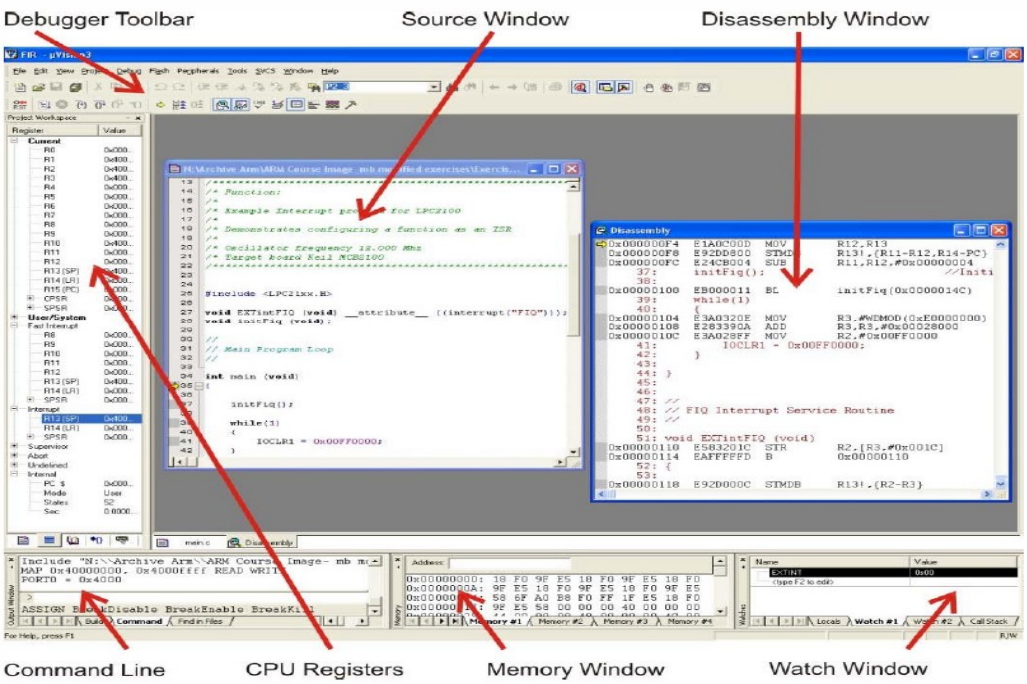


STEP5: Add code to source file and build target

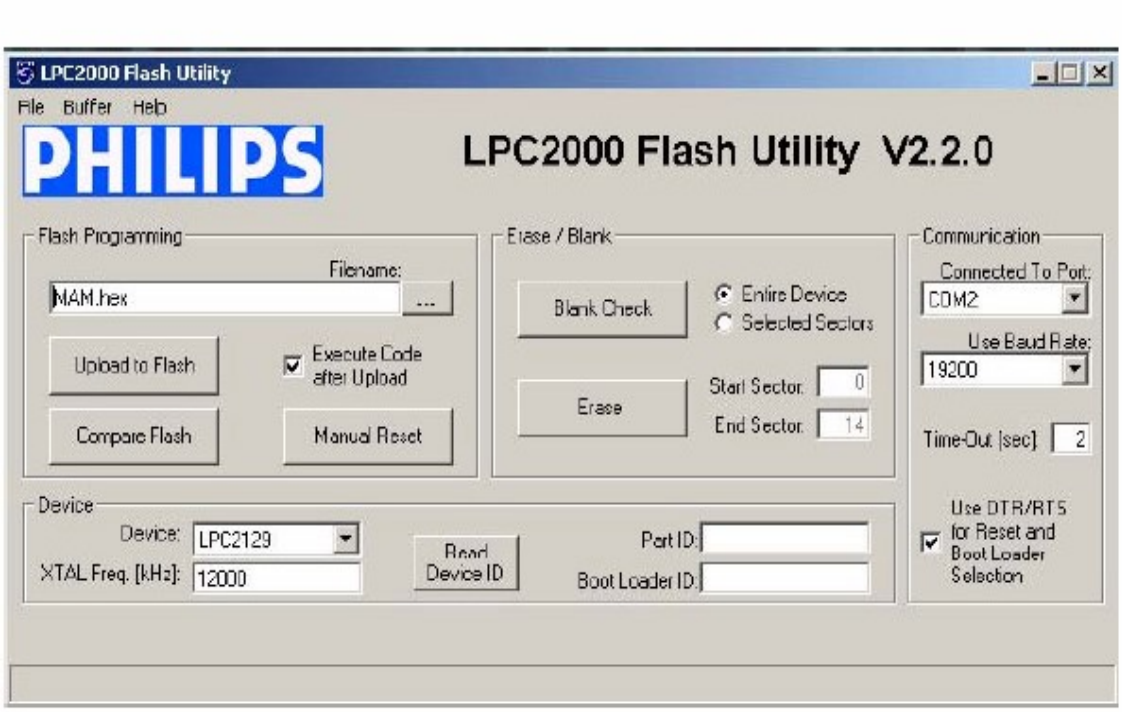


STEP6: Using simulator.





STEP7: Using PHILIPS LPC2000 Flash Utility dump the hex file of the code.



CHAPTER 8

APPLICATIONS AND RESULTS & SAMPLE OUTPUT

RESULTS & CONCLUSION

8.1 TRANSMITTER PART:

- Infra red sensors are connected to ADC.
- Communication is through CAN1.

8.2 RECEIVER PART:

- Communication is through CAN2.
- After receiving the CAN values, it will display on LCD.
- If exceeds certain limit, warnings will be given.

8.3 EXECUTION MODEL:

- First select the pins of ADC, CAN and UART channels using PINSEL0 and PINSEL1 registers in LPC2129.
- Configure 9600 baud rate for serial ports with 8 bit communication mode and 125 Kbps for CAN channel
- If time is required then we can use internal RTC by setting control registers.
- ADC in ARM7 can be configured by using ADCR register and data can be converted in to ADDR register.
- Now connect two ultrasonic sensors to the ADC of ARM7 controllers and place them at front and rear ends.
- Whatever the data is coming from these sensors monitor it and send continuously using CAN1 channel to CAN2 through bus.
- Based on the data coming to CAN2 channel through the bus, the controller will decide what operation should perform.
- If any object is very near to the vehicle then the vehicle will be stopped automatically.
- By using Kalman algorithm we can find out the relative speed between the nearby vehicles and monitor it continuously.
- All these information we are displaying on the lcd and at the same time we can send to a PC.

8.4 APPLICATIONS:

8.4.1 REAL TIME APPLICATION:

- Used as a Warning System to avoid Collision in National Highways.
- Used by Police to Track the speed of the approaching vehicles.
- Used to detect an object in Extreme conditions like Fog and misty areas.
- Can be implemented in Robotic Applications.
- Can be used in large vehicles like Trucks and buses
- Can be implemented in Aircraft and aerospace electronics
- Can be used in Passenger and cargo trains can be implemented in Maritime electronics.

8.5 SAMPLE OUTPUT:

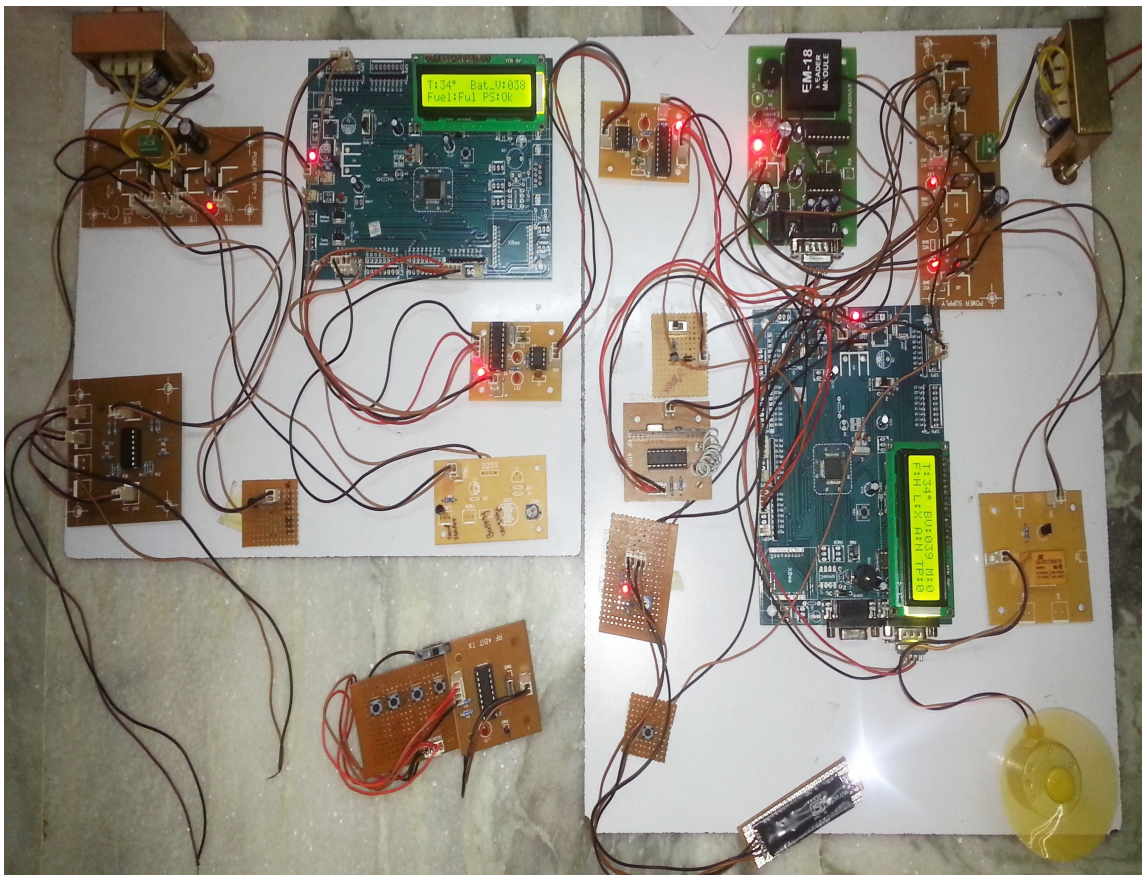


Figure 8.5: Sample Output

CHAPTER 9

CONCLUSION & FUTURE ENHANCEMENT

9.1 CONCLUSION:

This project An Integrated Health Management Process for Automotive Cyber-Physical Systems is intended for secure and smooth journey. The car/ vehicle itself is aware of its movement. If the driver himself is not concentrating on driving or any other parameters, which may cause damage to vehicle as well a life, this intelligent car/ vehicle warn the driver regarding the danger ahead. As the value of a human life is countless times more than the cost of this project, we are proud to be behind the success of this project.

9.2 FUTURE ENHANCEMENT:

The goal of the Integrated Vehicle Health Management (IVHM) project is to develop validated tools, technologies, and techniques for automated detection, diagnosis and prognosis that enable mitigation of adverse events during travel. Adverse events include those that arise from system, subsystem, or component faults or failures due to damage, degradation, or environmental hazards that occur during travel. As for the rear end, side-end end collision avoidance subsystem can be adopted with the use of the currently available ultrasonic sensors for vehicles. Further, the relative speed between two cars can be estimated by applying a one-dimension Kalman filter with great efficiency. Because software health management is a field in its infancy, this project will perform the foundational research needed to develop technologies for automated detection, diagnosis, prognostics, and mitigation of adverse events. Much effort from past programs has been placed on understanding safety issues that arise from hardware issues. From experimental data, a D/V curve can be further obtained to reliably generate a warning signal in advance of the accidental collision. As the car in traffic with pretty low speed or in a waiting state at the intersection, the warning signals should be terminated after a certain time since no collision warning is required under such circumstances.

CHAPTER 10

BIBLIOGRAPHY

10.1 REFERENCES

10.1.1 BOOKS:

[1]. ARM7TDMI datasheet ARM

[2]. LPC2119/2129/2194/2292/2294 User Manual Philips

[3]. ARM System on chip architecture Steve Furber

[4]. Architecture Reference Manual David Seal

[5]. ARM System developers guide Andrew N. Sloss,

Domonic Symes,

[6]. Chris Wright

[7]. Micro C/OS-II Jean J. Labrosse

GCC The complete reference Arthur Griffith

10.1.2 WEBSITES:

[1]. http://www.arm.com

[2]. http://www.philips.com

[3]. http://www.lpc2000.com

[4]. http://www.semiconductors.philips.com/

[5]. http://ieeexplore.ieee.org

[6]. http://ww.hitex.co.uk

[7]. http://www.keil.co.uk

[8]. http://www.ucos-ii.com