**ADVANCED DRIVER SAFETY AWARENESS AND ASSISTANCE SYSTEM FOR COGNITIVE VEHICLE CONTROL**

**A PROJECT REPORT**

***Submitted by***

R.ARCHANA

N.DHIVYA

P.GUNA SUNDARI

***in partial fulfilment for the award of the degree***

***of***

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERINg



SRI RAMANUJAR ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY: CHENNAI 600 048

APRIL 2013

**ANNA UNIVERSITY : CHENNAI-600048**

**BONAFIDE CERTIFICATE**

Certified that this project report “**ADVANCED DRIVER SAFETY AWARENESS AND ASSISTANCE SYSTEM FOR COGNITIVE VEHICLE CONTROL**” is the bonafide work of

R.ARCHANA (42609106005)

N.DHIVYA (42609106011)

P.GUNA SUNDARI (42609106016)

Who carried out their project work under my supervision

**SIGNATURE Signature**

**Prof.B.Gowrisankaran Mrs.C.Georgiana**

**HEAD OF THE DEPARTMENT SUPERVISOR**

**DEPT OF ECE DEPT OF ECE**

**Sri Ramanujar Engineering College Sri Ramanujar Engineering College**

**Nedunkundram, Vandalur, Nedunkundram, Vandalur,**

**Chennai-600 048 Chennai-600 048**

**CERTIFICATE OF EVALUATION**

**College Name: Sri Ramanujar Engineering College**

**Branch and Semester: ECE &VIII Sem**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Name of the students who have done the project | Title of the project | Name of the Supervisor with Designation |
| **1** | **R.ARCHANA** | “ADVANCED DRIVER SAFETY AWARENESS AND ASSISTANCE SYSTEM FOR COGNITIVE VEHICLE CONTROL” | M**rs.C.GEORGIANA**  LECTURER |
| **2** | **N.DHIVYA** |
| **3** | **P.GUNA SUNDARI** |

The report of the project work submitted by the above students in partial fulfilment for the award of **Bachelor of Engineering Degree** in **Electronics and Communication Engineering** of **Anna University** were evaluated and confirmed to be reports of the works done by above students.

Submitted for the Project and Viva-Voce Examination held on .

**INTERNAL EXAMINER EXTERNAL EXAMINER**

***ABSTRACT***

Nowadays, safety has become one of the most important attributes of a vehicle. A modern vehicle has a plethora of active driver assistance systems such as cruise control, collision avoidance, automatic parking and stability control etc… These systems are heavily focused on vehicular control aspect. But the root cause of majority of accidents is due to the behavior of the one who drives the vehicle, the driver himself. Presently, it is important to develop cost-effective technological solutions that can accurately identify the driving behavior of drivers and to assist them. The project proposed here aims to design an advanced driver safety awareness and assistance system that will monitor the driver and command the vehicle to take those vital safety measures in order to prevent mobile phone usage while driving, which is a major cause of accident these days, identify driver fatigue condition which is the primary reason that results in less concentration on road, one of which is sleepy driving case, knowledge of whether a driver is drunken alcohol or not, way to find if a collision is about to happen or not. That would have allowed the vehicle to take those early safety measures to minimize the injuries to the driver, autonomous way to know the accident spot in order to get early help that might save life.

**CHAPTER 1**

**INTRODUCTION**

**1.1 EMBEDDED SYSTEM**

Embedded systems have a keypad and an LCD display, they are rarely capable of using many different types of input or output. An example of an embedded system with I/O capability is a security alarm with an LCD status display, and a keypad for entering a password.

In general, an Embedded System:

* Is a system built to perform its duty, completely or partially independent of human intervention.
* Is specially designed to perform a few tasks in the most efficient way.
* Interacts with physical elements in our environment, viz. controlling and driving a motor, sensing temperature, etc.

An embedded system can be defined as a control system or computer system designed to perform a specific task. Common examples of embedded systems include MP3 players, navigation systems on aircraft and intruder alarm systems. An embedded system can also be defined as a single purpose computer.

Most embedded systems are time critical applications meaning that the embedded system is working in an environment where timing is very important and the results of an operation are only relevant if they take place in a specific time frame. An autopilot in an aircraft is a time critical embedded system. If the autopilot detects that the plane for some reason is going into a stall then it should take steps to correct this within milliseconds or there would be catastrophic results.

**1.2 USES OF EMBEDDED SYSTEMS**

The uses of embedded systems are virtually limitless, because every day new products are introduced to the market that utilizes embedded computers in novel ways. The hardware such as microprocessors, microcontrollers and FPGA chips become much cheaper. So when implementing a new form of control, it's wiser to just buy the generic chip and write your own custom software for it. Producing a custom-made chip to handle a particular task or set of tasks costs far more time and money. Many embedded computers even come with extensive libraries, so that "writing your own software" becomes a very trivial task indeed.

From an implementation viewpoint, there is a major difference between a computer and an embedded system. Embedded systems are often required to provide **Real-Time response**.

A **Real-Time system** is defined as a system whose correctness depends on the timeliness of its response. Examples of such systems are flight control systems of an aircraft, sensor systems in nuclear reactors and power plants. For these systems, delay in response is a fatal error. **Real-Time Systems** can be classified as:

* **Hard Real-Time Systems** - systems with severe constraints on the timeliness of the response.
* **Soft Real-Time Systems** - systems which tolerate small variations in response times.
* **Hybrid Real-Time Systems** - systems which exhibit both hard and soft constraints on its performance.

**1.3. CHARACTERISTICS**

Have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

* Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose. For Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real time performance constraints that must be met, for reasons such as safety and usability; others may example, the Gibson Robot Guitar features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music. Similarly, an embedded system in an automobile provides a specific function as a subsystem of the car itself.
* The program instructions written for embedded systems are referred to as firmware and are stored in Read-Only Memory or Flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard and/or screen.

**1.4. IMPORTANCE OF EMBEDDED SYSTEMS**

Embedded systems are playing the very important roles in our lives everyday life. Some of the embedded systems we use every day control the menu system on television, the timer in a microwave oven, a mobile phone, an MP3 player or any other device with some amount of intelligence built-in. Embedded systems are a rapidly growing industry with numerous opportunities.

**CHAPTER 2**

**LITERATURE REVIEW**

**TITLE :** DRIVER FATIGUE DETECTION USING SENSOR NETWORK

**AUTHOR :** Mr. Swapnil V. Deshmukh , Ms. Dipeeka P.Radake ,

Mr. Kapil N.Hande

**COMMENTS :**

This paper proposed the driver’s fatigue approach for real- time detection of driver fatigue for real detection of driver fatigue . The system consists of a sensors directly pointed towards the driver’s face. The input to the system is a continuous stream of signals from the sensors. The system monitors the driver’s eye to detect micro-sleeps , monitors the driver’s jaw to detect jaw movement and monitors to detect driver pulse from finger using LED and LDR assembling. The system can analyze the eye lid movement & jaw movement, variation in pulse rate from driver compute it as well as compare signal. Accordingly, the drivers fatigue level can be obtained & alert driver.

In our project the driver fatigue or drowsiness level can be obtained by measuring the heart rate variability and steering wheel grip pressure. The heart rate variability measures through the flow of blood using heart rate sensor and the steering wheel grip pressure is measured using flexi force pressure sensor. If the system finds any abnormality in either one measure, it first alerts the driver through buzzer and gets slowed down to halt by pressing brake pedal automatically. Additionally the system has periodic wake up call. If the driver feel drowsy, then they can set a wakeup call using keypad , after that time it alerts the driver. And also the driver can indicate their location to outside world through GSM modem.

**TITLE:** DRIVER FATIGUE DETECTION SYSTEM

**AUTHOR:** E.Rogado, J.L.Garcia, R.Barea ,L.M.Bergasa

**IEEE: 2008**

**COMMENTS:**

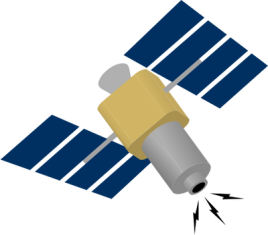
This paper presents a method for detecting the early signs of fatigue during driving. It analyzed some biological and environmental variables such as heart rate variability, steering wheel grip pressure & temperature difference between the inside & outside of the vehicle. It analyze all aspects and accordingly the system will respond.

Our project aims to design the system which prevents accident due to behavior of driver which includes mobile phone usage, driver fatigue, drunken drive etc. It not only focus on driver fatigue condition and also other factors which lead to accident. Here the fatigue level can be measured only by measuring the heart variability and steering wheel grip pressure. If any one factor changes , the system respond accordingly . It also has several sensors which monitors the driver’s behavior so that accident can be prevented. So that it alerts the driver through buzzer. Collision can be prevented by using SONAR which measures the distance between the vehicle & obstacle.

**CHAPTER 3**

**FUNCTIONAL DESCRIPTION**

**3.1 BLOCK DIAGRAM**

****

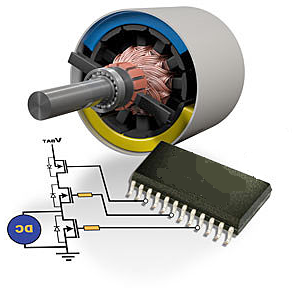
**GPS Satellites**

**Dashboard Graphics LCD**

****

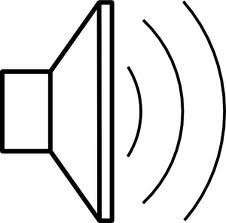
**GPS Receiver**

****

****

**DC Motor & Motor Driver Circuit**

**Buzzer**

****

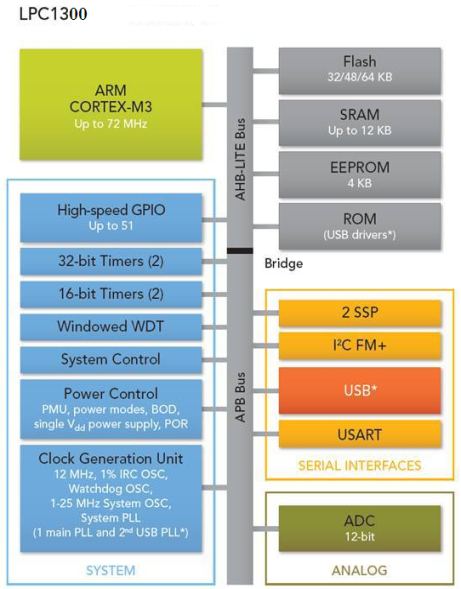
**PWM**

**UART-1**

**SPI-1**

****

**GPIO**

****

**Matrix Keypad**

**GPIO**

**MEMS Accelerometer**

**I²C**

****

**Brake Pedal Sensor**

****

**Mobile Baseband Sensor**

**Accelerator Pedal Sensor**

**ADC**

**GPIO**

****

**GPIO**

**ADC**

****

****

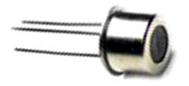
**PWM**

**ADC**

**UART-2**

**GPIO**

**SONAR**

****

**FlexiForce Pressure Sensor**

****

**Alcohol Sensor**

**Seat Belt Control Servo Motor**

**GSM Cellular Modem**

**Heart Rate Sensor**

**MEMS Accelerometer**

**I²C**

**ARM Cortex-M3**

**SPI-1**

**Dashboard Graphics LCD**

**GPS Receiver**

**UART-1**

**DC Motor & Motor Driver Circuit**

**PWM**

**Buzzer**

**GPIO**

**Brake Pedal Sensor**

**ADC**

**Matrix Keypad**

**GPIO**

**Accelerator Pedal Sensor**

**Mobile Baseband Sensor**

**GPIO**

**SONAR**

**GPIO**

**ADC**

**Flexi Force Pressure Sensor**

**GPIO**

**Heart Rate Sensor**

**Alcohol Sensor**

**ADC**

**UART-2**

**PWM**

**GSM Cellular Modem**

**Seat Belt Control Servo Motor**

**3.2 DESCRIPTION**

The project has the following features and modules

**3.2.1 MOBILE BASEBAND MONITOR**

Any activity in driver’s mobile phone such as attending incoming calls , making outgoing calls and SMS texting will be actively monitored using the built in Mobile Baseband Sensor circuit and if it finds any activity while the vehicle is running , it will be slowed down to a halt by applying the brakes automatically . The driver can regain the vehicle control by simply pressing the brake pedal.

Mobile baseband sensor senses the activity in driver’s mobile phone while driving and sends the information in the form of signal to the microcontroller through GPIO . Then in turn microcontroller sends the signal to the DC motor driver circuit to slow down the speed of the vehicle.

In order to regain the speed of vehicle , the driver has to apply the brake manually . This will be senses by the brake pedal sensor and accelerator pedal sensor. These sensors acts as a rheostat to increase the speed of the vehicle. Since the signal from the pedal sensors is analog , Analog to Digital Converter is used to convert the signal in the form of digital.

**3.2.2 HANDS FREE AUTO REPLY SMS**

The system has a GSM modem with SIM card. The driver needs to activate call divert to this number before entering this mode. In this mode , upon receiving an incoming call while driving , the GSM modem automatically sends an SMS to the calling number with a fixed message inditicating that the person has been driving . The system has a dashboard Graphics LCD that can show the calling number. This makes the driver 100% hands free. The mode can be enabled using matrix keypad.

**3.2.3 DRIVER FATIGUE WARNING**

This feature provides a method for detecting the early signs of fatigue or drowsiness during driving. By analyzing some biological and environmental variables , it is possible to detect the loss of alertness prior to the driver falling asleep. The system will then determine if the subject is able to drive. Heart rate variability and steering wheel grip pressure are used to estimate the driver’s fatigue level. A digital pulse output Heart rate sensor measures HRV and an analog output Flexiforce pressure sensor measures the steer wheel grip force and the warning is issued with Buzzer and LED lights. If the driver ignores this warning and continues to drive then the system will apply brakes automatically to slow down and halt the vehicle. Additionally the system can also be set for periodic wake up call using a keypad buttons and the dashboard graphics LCD.

The signal senses by the heart rate sensor and steering wheel grip pressure sensor are analog and hence ADC is used to convert the signal from analog to digital. Then microcontroller receives the signal, in turn it send signal to buzzer and to LED lights to indicate the warning. If the system finds abnormality in any one of the sensor that is in either HRV or steering wheel grip pressure , it gives warning. If the warning is ignored by the driver, then microcontroller sends the command to break pedal sensor to halt the vehicle.

**3.2.4 DRUNKEN DRIVE PREVENTION**

The system has a built in alcohol testing feature which would instruct the driver to blow air into the sensor unit and checks the alcohol content present in the driver’s breathe. If the value has crossed a certain limit the vehicle ignition will be locked which prevents a drunken driver from starting the vehicle. An Alcohol sensor unit is integrated into the system for this purpose. The sensor sends the signal to microcontroller through ADC.

**3.2.5 COLLISION PRESAFE ACTIVATION**

Collision presafe is an automobile safety system designed to reduce the severity of an accident. It uses pulse output SONAR to detect an imminent crash and has three warning stages. The first warning stage includes audible and visual warnings to brake. If ignored, and if the system predicts a collision is unavoidable, tightens the seatbelt using the inbuilt servo motor mechanism providing seat belt protection, and automatic application of the brakes to lessen the severity of the predicted crash.

**3.2.6 GPS AND GSM BASED ACCIDENT/PANIC ALERT**

During an emergency situation the driver can indicate his location to outside world using a simple panic button. By pressing this button , the driver can send an SMS about his current location information to a prestored number. Also in the event of a crash , the system senses that using 3-axis digital MEMS accelerometer sensor and will automatically generate a similar SMS to a prestored number about the crash location information.

If the MEMS senses the crash ,it sends the information through I2c protocol and then using GPS exact location can be determined. Using the protocol UART the microcontroller communicates with GPS and GSM.

**CHAPTER 4**

**HARDWARE DESCRIPTION**

**4.1 MICROCONTROLLER**

A **microcontroller** (sometimes abbreviated **µC**, **uC** or **MCU**) is a small computer on a single [integrated circuit](http://en.wikipedia.org/wiki/Integrated_circuit) containing a processor core, memory, and programmable [input/output](http://en.wikipedia.org/wiki/Input/output) peripherals. Program memory in the form of [NOR flash](http://en.wikipedia.org/wiki/NOR_flash) or [OTP ROM](http://en.wikipedia.org/wiki/Programmable_read-only_memory) is also often included on chip, as well as a typically small amount of [RAM](http://en.wikipedia.org/wiki/Random-access_memory). Microcontrollers are designed for embedded applications, in contrast to the [microprocessors](http://en.wikipedia.org/wiki/Microprocessor) used in [personal computers](http://en.wikipedia.org/wiki/Personal_computer) or other general purpose applications.

The LPC1313 are ARM Cortex-M3 based microcontrollers for embedded applications featuring a high level of integration and low power consumption. The ARM Cortex-M3 is a next generation core that offers system enhancements such as enhanced debug features and a higher level of support block integration. The LPC1313 operate at CPU frequencies of up to 72 MHz. The ARM Cortex-M3 CPU incorporates a 3-stage pipeline and uses Harvard architecture with separate local instruction and data buses as well as a third bus for peripherals. The ARM Cortex-M3 CPU also includes an internal prefetch unit that supports speculative branching. The peripheral complement of the LPC1313 includes up to 32 kB of flash memory, up to 8 kB of data memory, USB Device (LPC1342/43 only), one Fast-mode Plus I2C-bus interface, one UART, four general purpose timers, and up to 42 general purpose I/O pins

**FEATURES**

* ARM Cortex-M3 processor, running at frequencies of up to 72 MHz.
* ARM Cortex-M3 built-in Nested Vectored Interrupt Controller (NVIC).
* 32 kB (LPC1343/13)/16 kB (LPC1342)/8 kB (LPC1311) on-chip flash programming memory.
* 8 kB (LPC1343/13)/4 kB (LPC1342/11) SRAM.
* In-System Programming (ISP) and In-Application Programming (IAP) via on-chip bootloader software.

**4.2 GLOBAL POSITIONING SYSTEM (GPS)**

The Global Positioning System (GPS) is a location system based on a constellation of about 24 satellites orbiting the earth at altitudes of approximately 11,000 miles. GPS was developed by the United States Department of Defense (DOD), for its tremendous application as a military locating utility. The DOD's investment in GPS is immense. GPS has proven to be a useful tool in non-military mapping applications as well.

GPS satellites are orbited high enough to avoid the problems associated with land based systems, yet can provide accurate positioning 24 hours a day, anywhere in the world. Uncorrected positions determined from GPS satellite signals produce accuracies in the range of 50 to 100 meters. When using a technique called differential correction, users can get positions accurate to within 5 meters or less.

**4.2.1 DETERMINATION OF A LOCATION USING GPS**

In a nutshell, GPS is based on satellite ranging - calculating the distances between the receiver and the position of 3 or more satellites (4 or more if elevation is desired) and then applying some good old mathematics. Assuming the positions of the satellites are known, the location of the receiver can be calculated by determining the distance from each of the satellites to the receiver. GPS takes these 3 or more known references and measured distances and "triangulates" an additional position.

GPS satellites are orbiting the Earth at an altitude of 11,000 miles. The DOD can predict the paths of the satellites vs. time with great accuracy. Furthermore, the satellites can be periodically adjusted by huge land-based radar systems. Therefore, the orbits, and thus the locations of the satellites, are known in advance. Today's GPS receivers store this orbit information for all of the GPS satellites in what is known as an almanac.

Think of the almanac as a "bus schedule" advising you of where each satellite will be at a particular time. Each GPS satellite continually broadcasts the almanac. Your GPS receiver will automatically collect this information and store it for future reference.

**4.3 GSM MODEM**

A GSM modem is a wireless modem that works with a GSM wireless network. A wireless modem behaves like a dial-up modem. The main difference between them is that a dial-up modem sends and receives data through a fixed telephone line while a wireless modem sends and receives data through radio waves.

A GSM modem can be an external device or a PC Card / PCMCIA Card. Typically, an external GSM modem is connected to a computer through a serial cable or a USB cable. A GSM modem in the form of a PC Card / PCMCIA Card is designed for use with a laptop computer. It should be inserted into one of the PC Card / PCMCIA Card slots of a laptop computer.

Like a GSM mobile phone, a GSM modem requires a SIM card from a wireless carrier in order to operate. Both GSM modems and dial-up modems support a common set of standard AT commands. You can use a GSM modem just like a dial-up modem.In addition to the standard AT commands, GSM modems support an extended set of AT commands. These extended AT commands are defined in the GSM standards. With the extended AT commands, you can do things like:

* Reading, writing and deleting SMS messages.
* Sending SMS messages.
* Monitoring the signal strength.
* Monitoring the charging status and charge level of the battery.
* Reading, writing and searching phone book entries.

The number of SMS messages that can be processed by a GSM modem per minute is very low -- only about six to ten SMS messages per minute.

**4.3.1 GSM MODEM CHARACTERISTICS**

* Triband GSM GPRS modem (EGSM 900/1800 / 1900 MHz )
* Designed for GPRS, data, fax, SMS and voice applications
* GPRS multi-slot class 10
* GPRS mobile station class B
* Designed for GPRS, data, fax, SMS and voice applications
* Fully compliant with GSM Phase 2/2+ specifications
* Built-in TCP/IP Protocol
* Built-in RTC in the module.
* AT Command based

**4.3.2 SPECIFICATIONS FOR DATA**

* GPRS class 10: max 85.6 kbps (downlink)
* PBCCH Support
* Coding schemes CS 1,2,3,4
* CDS up to 14.4 kbps
* USSD
* Non transparent mode
* PPP - stack

**4.3.3 POWER SUPPLY**

Use AC ­ DC Power Adaptor with following ratings

* Input AC Voltage : 230V
* Output DC Voltage : 12V
* Output DC Current : 2A
* Polarity : Centre +ve & Outside ­ve

**4.3.4 GENERAL CHARACTERISTICS** :

* Input voltage : 9V-12V
* Input current : 3mA in idle mode (only module)
* Temperature range : Operating -20 to +55 degree Celsius; Storage -40 to +80 degree Celsius

**4.3.5 Interfaces**

* RS-232 through D-TYPE 9 pin connector
* Serial port baud rate 1200 to 115200 bps
* RJ11 voice connector
* Power supply through DC jacket
* SMA antenna connector
* Toggle spring/Flap Opening type SIM holder
* LED status of GSM / GPRS module

**4.4 SONAR**

**4.4.1 SRF02 SONAR SENSOR**

The SRF02 is a single transducer ultrasonic rangefinder in a small footprint CB. It features both I2C and a Serial interfaces. The serial interface is a standard TTL level UART format at 9600 baud,1 start, 2 stop and no parity bits, and may be connected directly to the serial ports on any microcontroller.

Up to 16 SRF02's may be connected together on a single bus, either I2C or Serial. New commands in the SRF02 include the ability to send an ultrasonic burst on its own without a reception cycle, and the ability to perform a reception cycle without the preceding burst. This has been as requested feature on our sonar's and the SRF02 is the first to see its implementation.

Because the SRF02 uses a single transducer for both transmission and reception, the minimum range is higher than our other dual transducer rangers. The minimum measurement range is around 15cm (6 inches). Like all our rangefinders, the SRF02 can measure in us, cm or inches.

**4.4.2Technical Specification SRF02**

* Ultrasonic range finder
* Single transducer
* Frequency 40kHz
* Detection angle 55°
* Power 5V/4mA typ.
* Range 15cm - 6m
* Analogue gain automatic 64 step gain control
* Interface 1 standard I2C bus
* Interface 2 serial bus
* Both interfaces can address up to 16 devices
* Full automatic tuning - no calibration, just power up and go
* Transmit-only and receive-only modes also available
* Weight - 4.6g (0.16oz)
* Size - 24mm x 20mm x 17mm height (0.94" x 0.79" x 0.67")

**4.4.3 OPERATING MODES**

There are two operating modes for the SRF02. I2C mode and Serial Mode. This is set with the Mode pin, connected to 0v Ground for Serial Mode and left unconnected (or tied to +5v Vcc) for I2C Mode.

**4.5 MEMS ACCELEROMETER**

An accelerometer is a device for measuring acceleration and gravity induced reaction forces. Single- and multi-axis models are available to detect magnitude and direction of the acceleration as a vector quantity. Accelerometers can be used to sense inclination, vibration, and shock. They are increasingly present in portable electronic devices.

Modern accelerometers are often small micro electro-mechanical systems (MEMS), and are indeed the simplest MEMS devices possible, consisting of little more than a cantilever beam with a proof mass (also known as seismic mass). Mechanically the accelerometer behaves as a mass-damper-spring system; the damping results from the residual gas sealed in the device. As long as the Q-factor is not too low, damping does not result in a lower sensitivity.

Under the influence of gravity or acceleration the proof mass deflects from its neutral position. This deflection is measured in an analog or digital manner. Most commonly the capacitance between a set of fixed beams and a set of beams attached to the proof mass is measured. This method is simple and reliable; it also does not require additional process steps making it inexpensive. Integrating piezoresistors in the springs to detect spring deformation, and thus deflection, is a good alternative, although a few more process is needed.

For very high sensitivities quantum tunneling is also used; this requires specific fabrication steps making it more expensive. Optical measurement has been demonstrated on laboratory scale.

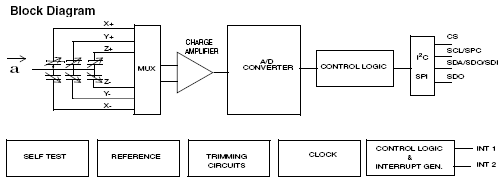
Another, far less common, type of MEMS-based accelerometer contains a small heater at the bottom of a very small dome, which heats the air inside the dome to cause it to rise. A thermocouple on the dome determines where the heated air reaches the dome and the deflection off the center is a measure of the acceleration applied to the sensor.

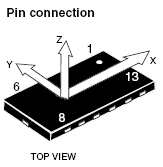
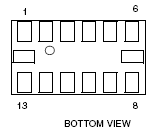
Most micromechanical accelerometers operate in-plane, that is, they are designed to be sensitive only to a direction in the plane of the die. By integrating two devices perpendicularly on a single die a two-axis accelerometer can be made. By adding an additional out-of-plane device three axes can be measured. Such a combination always has a much lower misalignment error than three discrete models combined after packaging.

Micromechanical accelerometers are available in a wide variety of measuring ranges, reaching up to thousands of g's. The designer must make a compromise between sensitivity and the maximal acceleration that can be measured.

**4.5.1 FEATURES:**

* Three axes
* SPI/I2C digital interface
* Innovative embedded functionalities
* Two highly flexible and programmable interrupt request outputs
* Programmable thresholds and timing of interrupt signals
* 2.16V to 3.6V supply voltage
* 1.8V compatible I/Os
* <1mW power consumption
* Temperature range -40 to +85ºC
* ±2/±8g selectable full scale range
* Embedded high pass filter
* Embedded self-test
* 10,000g high shock survivability
* LGA package 3x5x0.9mm3
* ECOPACK® – RoHS Directive and green compliant



**Pin description**

**Pin Name Function**

1 Vdd\_IO Power supply for I/O pins

2 GND 0V supply

3 Reserved Connect to Vdd

4 GND 0V supply

5 GND 0V supply

6 Vdd Power supply

7 CS SPI enable

I2C/SPI mode selection (1: I2C mode; 0: SPI enabled)

8 INT 1 Inertial interrupt 1

9 INT 2 Inertial interrupt 2

10 GND 0V supply

11 Reserved Connect to Gnd

12 SDO SPI Serial Data Output

I2C less significant bit of the device address

13 SDA I2C Serial Data (SDA)

SDI SPI Serial Data Input (SDI)

SDO 3-wire Interface Serial Data Output (SDO)

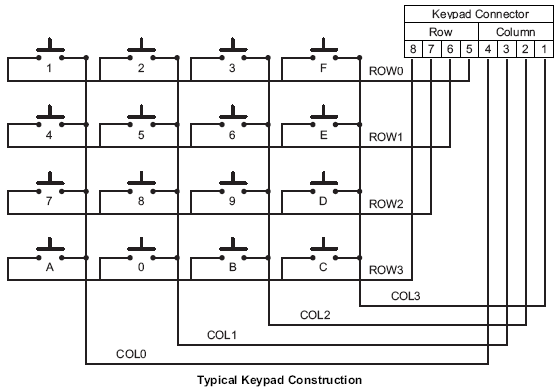
14 SCL I2C Serial Clock (SCL)SPC

SPI Serial Port Clock (SPC)

**4.6 MATRIX KEYPAD**

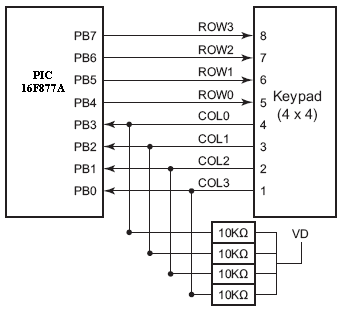
4.6.1 4 X 4 MATRIX KEYPAD **INTERFACING**

The 4 x4 matrix keypad is a general-purpose keypad. It consists of 16 switches arranged in 4 rows and 4 columns. It can connect to the MCU 8-bit port directly.



The keypad is marked with numeric keys (0–9) and with function keys (A–F), as shown in Figure.

In this application, all keys are used as data keys. The keypad is connected to Port B. Port B is connected to the keypad, and scans the keys continuously. The columns of the keypad are pulled up with 10KΩ resistance to set them normally High. Key-pad and Port B connection detail is illustrated in Figure.

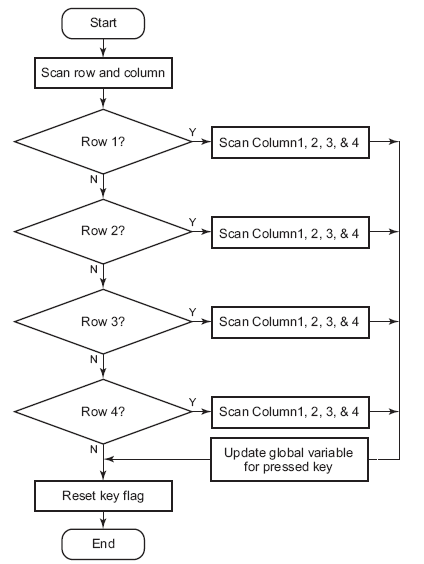


**4.5.2 Scanning a Matrix Keypad**

There are many methods depending on how you connect your keypad with your controller, but the basic logic is same. We make the columns as i/p and we drive the rows making them o/p, this whole procedure of reading the keyboard is called scanning.

In order to detect which key is pressed from the matrix, we make row lines low one by one and read the columns. Let’s say we first make Row1 low, then read the columns. If any of the key in row1 is pressed will make the corresponding column as low i.e. if second key is pressed in Row1, then column2 will give low. So we come to know that key 2 of Row1 is pressed.

So to scan the keypad completely, we need to make rows low one by one and read the columns. If any of the buttons is pressed in a row, it will take the corresponding column to a low state which tells us that a key is pressed in that row. If button 1 of a row is pressed then Column 1 will become low, if button 2 then column2 and so on...

**4.6.3FlowChart**

**4.7 SEAT BELT CONTROL SERVO MOTOR**

In any electric motor, operation is based on simple electromagnetism. A [current](http://encyclobeamia.solarbotics.net/articles/current.html)-carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the [current](http://encyclobeamia.solarbotics.net/articles/current.html) in the conductor, and to the strength of the external magnetic field. As you are well aware of from playing with magnets as a kid, opposite (North and South) polarities attract, while like polarities (North and North, South and South) repel. The internal configuration of a [DC](http://encyclobeamia.solarbotics.net/articles/dc.html) motor is designed to harness the magnetic interaction between a [current](http://encyclobeamia.solarbotics.net/articles/current.html)-carrying conductor and an external magnetic field to generate rotational motion.

[DC](http://encyclobeamia.solarbotics.net/articles/dc.html) motor has six basic parts -- axle, rotor (a.k.a., armature), stator, commutator, field magnet(s), and brushes. In most common DC motors (and all that [BEAM](http://encyclobeamia.solarbotics.net/articles/beam.html)ers will see), the external magnetic field is produced by high-strength permanent magnets1. The stator is the stationary part of the motor; this includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor (together with the axle and attached commutator) rotate with respect to the stator. The rotor consists of windings (generally on a core), the windings being electrically connected to the commutator.

**4.8 SENSORS**

A **sensor** (also called **detector**) is a [converter](http://en.wikipedia.org/wiki/Energy_conversion) that measures a [physical quantity](http://en.wikipedia.org/wiki/Physical_quantity) and converts it into a signal which can be read by an observer or by an (today mostly [electronic](http://en.wikipedia.org/wiki/Electronics)) instrument. For example, a [mercury-in-glass thermometer](http://en.wikipedia.org/wiki/Mercury-in-glass_thermometer) converts the measured temperature into expansion and contraction of a liquid which can be read on a calibrated glass tube. A [thermocouple](http://en.wikipedia.org/wiki/Thermocouple) converts temperature to an output voltage which can be read by a [voltmeter](http://en.wikipedia.org/wiki/Voltmeter). For accuracy, most sensors are [calibrated](http://en.wikipedia.org/wiki/Calibration) against known [standards](http://en.wikipedia.org/wiki/Standard_(metrology)).

Sensors are used in everyday objects such as touch-sensitive elevator buttons ([tactile sensor](http://en.wikipedia.org/wiki/Tactile_sensor)) and lamps which dim or brighten by touching the base. There are also innumerable applications for sensors of which most people are never aware. Applications include cars, machines, aerospace, medicine, manufacturing and robotics.

The sensors which are used in our project are as follows

* Mobile Baseband Sensor

It senses the activity of driver’s mobile phone while driving

* Heart Rate Sensor

It is used to sense the heart rate variability through the flow of blood

* Flexi Force Pressure Sensor

It is used for the purpose of sensing the grip pressure

* Alcohol Sensor

By blowing air into this sensor,it senses the amount of alcohol content in driver’s breathe

**4.9 INTERFACE**

**4.9.1 UART (Universal Asynchronous Receiver Transmitter)**

UART stands for the Universal Asynchronous Receiver/Transmitter. In asynchronous transmitting, teletype-style UARTs send a "start" bit, five to eight data bits, least-significant-bit first, an optional "parity" bit, and then one, one and a half, or two "stop" bits. The start bit is the opposite polarity of the data-line's idle state. The stop bit is the data-line's idle state, and provides a delay before the next character can start. (This is called asynchronous start-stop transmission). In mechanical teletypes, the "stop" bit was often stretched to two bit times to give the mechanism more time to finish printing a character. A stretched "stop" bit also helps resynchronization. The parity bit can either makes the number of "one" bits between any start/stop pair odd, or even, or it can be omitted. Odd parity is more reliable because it assures that there will always be at least one data transition, and this permits many UARTs to resynchronize.

In synchronous transmission, the clock data is recovered separately from the data stream and no start/stop bits are used. This improves the efficiency of transmission on suitable channels since more of the bits sent are usable data and not character framing. An asynchronous transmission sends nothing over the interconnection when the transmitting device has nothing to send; but a synchronous interface must send "pad" characters to maintain synchronism between the receiver and transmitter. The usual filler is the ASCII "SYN" character. This may be done automatically by the transmitting device. USART chips have both synchronous and asynchronous modes. start|< five to eight data bits >| stop bit(s)

0 ---- - - - - - - - - - - Space (logic low)

| | | | | | | | | | | |

| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | S | S |

| | | | | | | | | | | |

1 - - - - - - - - - - - -------- Mark (logic high)

Asynchronous Code Format

The right-most bit is always transmitted first. If parity is present, the parity bit comes after the data bits but before the stop bit(s).

**4.9.1.1 TRANSMITTER (USART)**

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI). The USART can be configured in the following modes:

• Asynchronous (full duplex)

• Synchronous – Master (half duplex)

• Synchronous – Slave (half duplex)

**Steps to follow when setting up an Asynchronous Transmission:**

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH.

2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.

3. If interrupts are desired, then set enable bit TXIE.

4. If 9-bit transmission is desired, then set transmit bit TX9.

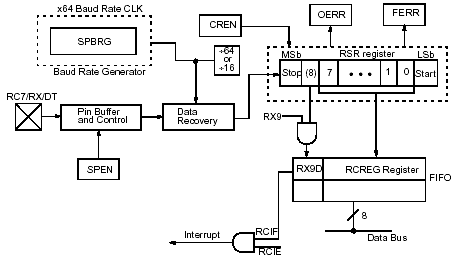
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.

6. If 9-bit transmission is selected; the ninth bit should be loaded in bit TX9D.

7. Load data to the TXREG register (starts transmission).

**4.9.1.2 USART ASYNCHRONOUS RECEIVER**

The receiver block diagram is shown in Fig. The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. The heart of the receiver is the receive (serial) shift register (RSR). After sampling the STOP bit, the received data in the RSR is transferred to the RCREG register (if it is empty). If the transfer is complete, flag bit RCIF (PIR1<5>) is set. The actual interrupt can be enabled/ disabled by setting/clearing enable bit RCIE (PIE1<5>).



USART Receiver block diagram

**Steps to follow when setting up an Asynchronous Reception:**

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH.

2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.

3. If interrupts are desired, then set enable bit RCIE.

4. If 9-bit reception is desired, then set bit RX9.

5. Enable the reception by setting bit CREN.

6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE is set.

7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.

8. Read the 8-bit received data by reading the RCREG register.

9. If any error occurred, clear the error by clearing enable bit CREN.

**4.9.2 I²C PROTOCOL**

I²C is a multi-master serial computer bus invented by Philips that is used to attach low-speed peripherals to a motherboard, embedded system, or cell phone. The name stands for Inter-Integrated Circuit and is pronounced I-squared-C and also, incorrectly, I-two-C.

I²C uses only two bidirectional open-drain lines, Serial Data (SDA) and Serial Clock (SCL), pulled up with resistors. The I²C reference design has a 7-bit address space with 16 reserved addresses, so a maximum of 112 nodes can communicate on the same bus. The maximum number of nodes is obviously limited by the address space, and also by the total bus capacitance of 400 pF.

To maximize hardware efficiency and circuit simplicity, Philips developed a simple bi-directional 2-wire bus for efficient inter-IC control. This bus is called the Inter IC or I²C-bus. All I²C-bus compatible devices incorporate an on-chip interface which allows them to communicate directly with each other via the I²C-bus. This design concept solves the many interfacing problems encountered when designing digital control circuits. Here are some of the features of the I²C-bus:

* Only two bus lines are required; a serial data line (SDA) and a serial clock line (SCL)
* Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exist at all times; masters can operate as master-transmitters or as master-receivers
* It’s a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer.
* Serial, 8-bit oriented, bi-directional data transfers can be made at up to 100 Kbit/s in the Standard-mode, up to 400 Kbit/s in the Fast-mode, or up to 3.4 Mbit/s in the High-speed mode
* On-chip filtering rejects spikes on the bus data line to preserve data integrity
* The number of ICs that can be connected to the same bus is limited only by a maximum bus capacitance of 400 pF.

For 8-bit oriented digital control applications, such as those requiring microcontrollers, certain design criteria can be established:

* + A complete system usually consists of at least one microcontroller and other peripheral devices such as memories and I/O expanders
  + The cost of connecting the various devices within the system must be minimized
  + A system that performs a control function doesn’t require high-speed data transfer
  + Overall efficiency depends on the devices chosen and the nature of the interconnecting bus structure.

To produce a system to satisfy these criteria, a serial bus structure is needed. Although serial buses don’t have the throughput capability of parallel buses, they do require less wiring and fewer IC connecting pins. However, a bus is not merely an interconnecting wire, it embodies all the formats and procedures for communication within the system**.**

Devices communicating with each other on a serial bus must have some form of protocol which avoids all possibilities of confusion, data loss and blockage of information. Fast devices must be able to communicate with slow devices.

The system must not be dependent on the devices connected to it, otherwise modifications or improvements would be impossible. A procedure has also to be devised to decide which device will be in control of the bus and when. And, if different devices with different clock speeds are connected to the bus, the bus clock source must be defined. All these criteria are involved in the specification of the I²C-bus.

#### 4.9.2.1 THE I²C BUS PROTOCOL

The I²C bus physically consists of 2 active wires and a ground connection. The active wires, called SDA and SCL, are both bi-directional. SDA is the Serial Data line, and SCL is the Serial Clock line.

Every device hooked up to the bus has its own unique address, no matter whether it is an MCU, LCD driver, memory, or ASIC. Each of these chips can act as a receiver and/or transmitter, depending on the functionality. Obviously, an LCD driver is only a receiver, while a memory or I/O chip can be both transmitter and receiver.

The I²C bus is a multi-master bus. This means that more than one IC capable of initiating a data transfer can be connected to it. The I²C protocol specification states that the IC that initiates a data transfer on the bus is considered the Bus Master. Consequently, at that time, all the other ICs are regarded to be Bus Slaves.

**4.9.3 SERIAL PERIPHERAL INTERFACE**

Serial Peripheral Interface is a simple interface which enables to communicate microcontroller and peripheral chips or intercommunicate between two or more microcontrollers. Serial Peripheral Interface bus sometimes called four wire interfaces may be used to interface such chips or devices like: LCD, sensors, memories, ADC, RTC. The range of usage is huge.

SPI Bus uses synchronous protocol, where transmitting and receiving is guided by clock signal generated by master microcontroller. SPI interface allows connecting several SPI devices while master selects each of them with CS (Chip Select) signal – (Underline means that active is LOW).

SPI bus consists of four signal wires:

Master Out Slave in (MOSI),

Master in Slave Out (MISO),

Serial Clock (SCLK or SCK)

Chip Select (CS) for the peripheral.

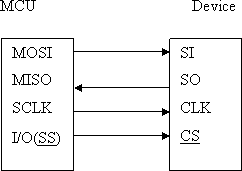
Some microcontrollers have a dedicated chip select for SPI interfacing called Slave Select (SS).

MOSI signal is generated by master – recipient is Slave. MOSI may also be labeled as SI or SDI.

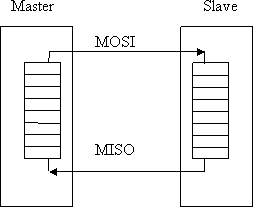
MISO signals are generated by slave. In some chips you might find labels SO or SDO.

SCLK or SCK are generated by master to synchronize data transfers.

CS (SS) signal is also generated by master to select slave chip or device.



Data transfer is organized by using Shift register in both: master and slave. While master shifts register value out through MOSI line then slave shifts data in to its shift register. If there is full duplex used, then send and receive is performed at the same time:

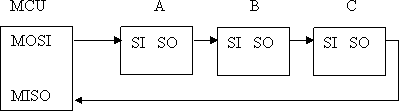


There also is multiple byte stream modes available with SPI bus interface. In this mode master can shift bytes continuously. Using this type of transfer slave select (SS) must remain low until all stream process continues. For example you can write to memory by sending address bytes and then data in the same stream operation. In this way kilobytes or more can be sent using multiple byte transfer mode.

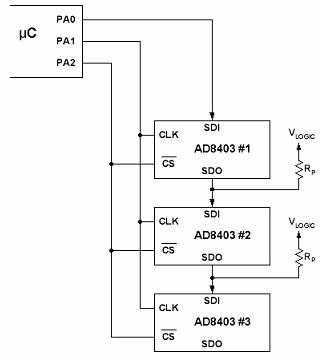
**4.9.3.1 DAISY-CHAINED MODE**

Some chips have ability when you can connect multiple SPI devices in series and data can be transferred to all devices though other.

For instance if you connect three devices to microcontroller in series MCU-A-B-C, then MCU will have to send three bytes of information. The first byte goes to C device, second to B and third byte to A:



Daisy-chain won’t work with devices which support or require multiple bytes operation (memory chips).For instance AD8403 - 4-Channel Digital Potentiometer can be connected in daisy-chain circuit:



Depending on Clock (SKC) polarity, there may be four operation modes of SPI:

Low clock polarity – when clock is low in idle and toggles in high;

High clock polarity – when clock is high in idle and toggles in low stage;

Clock phase zero – MOSI and MISO outputs are valid on rising edge of clock signal (from low to high);

Clock phase one – MOSI and MISO outputs are valid on falling edge of SCK signal (high to low)

**4.9.3.2 SPI MODE**

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

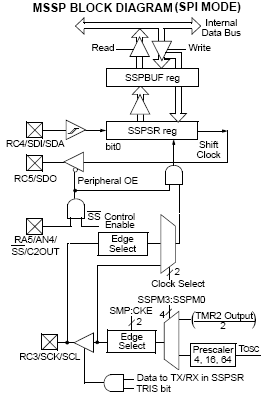
• Serial Data Out (SDO) – RC5/SDO

• Serial Data In (SDI) – RC4/SDI/SDA

• Serial Clock (SCK) – RC3/SCK/SCL

Additionally, a fourth pin may be used when in a Slave mode of operation:

• Slave Select (SS) – RA5/AN4/SS/C2OUT



**4.9.5 ANALOG TO DIGITAL CONVERTER**

An analog-to-digital converter (abbreviated ADC, A/D or A to D) is an electronic integrated circuit, which converts continuous signals to discrete digital numbers. The reverse operation is performed by a digital-to-analog converter (DAC).

**4.9.5.1 SUCCESSIVE APPROXIMATION ADC**

A successive-approximation ADC uses a comparator to reject ranges of voltages, eventually settling on a final voltage range. Successive approximation works by constantly comparing the input voltage to the output of an internal digital to analog converter (DAC, fed by the current value of the approximation) until the best approximation is achieved.

At each step in this process, a binary value of the approximation is stored in a successive approximation register (SAR). The SAR uses a reference voltage (which is the largest signal the ADC is to convert) for comparisons. For example if the input voltage is 60 V and the reference voltage is 100 V, in the 1st clock cycle, 60 V is compared to 50 V (the reference, divided by two. This is the voltage at the output of the internal DAC when the input is a '1' followed by zeros), and the voltage from the comparator is positive (or '1') (because 60 V is greater than 50 V).

At this point the first binary digit (MSB) is set to a '1'. In the 2nd clock cycle the input voltage is compared to 75 V (being halfway between 100 and 50 V: This is the output of the internal DAC when its input is '11' followed by zeros) because 60 V is less than 75 V, the comparator output is now negative (or '0'). The second binary digit is therefore set to a '0'. In the 3rd clock cycle, the input voltage is compared with 62.5 V (halfway between 50 V and 75 V: This is the output of the internal DAC when its input is '101' followed by zeros).

The output of the comparator is negative or '0' (because 60 V is less than 62.5 V) so the third binary digit is set to a 0. The fourth clock cycle similarly results in the fourth digit being a '1' (60 V is greater than 56.25 V, the DAC output for '1001' followed by zeros). The result of this would be in the binary form 1001. This is also called bit-weighting conversion, and is similar to a binary search.

The analogue value is rounded to the nearest binary value below, meaning this converter type is mid-rise (see above). Because the approximations are successive (not simultaneous), the conversion takes one clock-cycle for each bit of resolution desired. The clock frequency must be equal to the sampling frequency multiplied by the number of bits of resolution desired. For example, to sample audio at 44.1 kHz with 32 bit resolution, a clock frequency of over 1.4 MHz would be required. ADCs of this type have good resolutions and quite wide ranges.

They are more complex than some other designs.  Many PIC's have the ability to perform analogue to digital conversions, PIC16F87XA doesn't take any extra components, although a pull down resistor is always good to keep the conversions as accurate as possible, though not needed in some cases.

**4.9.5.2 CONFIGURING ANALOG PORT PINS**

The ADCON1 and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted. The A/D operation is independent of the state of the CHS2:CHS0 bits and the TRIS bits.

**4.9.5.3 A/D CONVERSIONS**

Clearing the GO/DONE bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. That is, the ADRESH: ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH: ADRESL registers). After the A/D conversion

**4.9.5.4 A/D RESULT REGISTERS**

The ADRESH: ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16 bits wide.

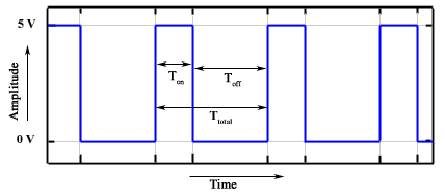
The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D is aborted; the next acquisition on the selected channel is automatically started. The GO/DONE bit can then be set to start the conversion. After the GO bit is set, the first time segment has a minimum of TCY and a maximum of TAD. Format Select bit (ADFM) controls this justification. The extra bits are loaded with ‘0’s. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

**4.9.6 PULSE WIDTH MODULATION**

Pulse width Modulation or PWM is one of the powerful techniques used in control systems today. They are not only employed in wide range of control application which includes: speed control, power control, measurement and communication.

**4.9.6.1BASIC PRINCIPAL OF PWM**

Pulse-width Modulation is archived with the help of a square wave whose duty cycle is changed to get a varying voltage output as a result of average value of waveform. A mathematical explanation of this is given below.



Consider a square wave shown in the figure above.

Ton is the time for which the output is high and Toff is time for which output is low. Let Ttotalbe time period of the wave such that,



Duty cycle of a square wave is defined as



The output voltage varies with duty cycle as...





So you can see from the final equation the output voltage can be directly varied by varying the Ton value.

If Ton is 0, Vout is also 0.

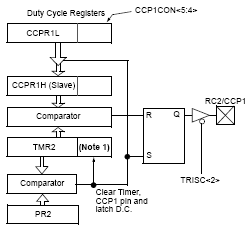
if Ton is Ttotal then Vout is Vin or say maximum.

This was all about theory behind PWM. Now let’s take a look at the practical implementation of PWM on microcontrollers.

**4.9.6.2 PWM MODE (PWM)**

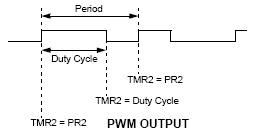
In Pulse Width Modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch,

the TRISC<2> bit must be cleared to make the CCP1pin an output. Figure shows a simplified block diagram of the CCP module in PWM mode. For a step-by-step procedure on how to set up the CCP module for PWM operation.



BLOCK DIAGRAM

A PWM output has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).



**4.9.6.3 PWM PERIOD**

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

PWM Period = [(PR2) + 1] • 4 • TOSC •

(TMR2 Prescale Value)

PWM frequency is defined as 1/ [PWM period]. When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

• TMR2 is cleared

• The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)

• The PWM duty cycle is latched from CCPR1L into CCPR1H

**4.9.6.4 PWM DUTY CYCLE**

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>.

The following equation is used to calculate the PWM duty cycle in time:

PWM Duty Cycle = (CCPR1L:CCP1CON<5:4>) •

TOSC • (TMR2 Prescale Value)

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2

Occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register. The CCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitch-free PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the following formula.



**4.9.6.5 SETUP FOR PWM OPERATION**

The following steps should be taken when configuring the CCP module for PWM operation:

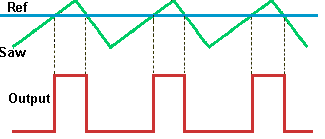
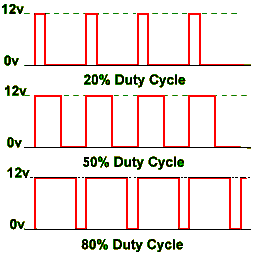
1. Set the PWM period by writing to the PR2 register.

2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.

3. Make the CCP1 pin an output by clearing the TRISC<2> bit.

4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.

5. Configure the CCP1 module for PWM operation.



PWM Duty Cycle SAW Tooth Wave

**CHAPTER 5**

**SOFTWARE DESCRIPTION**

**5.1 INTRODUCTION**

LPCXpresso is a new, low-cost development platform available from NXP. The software consists of an enhanced, Eclipse-based IDE, a GNU C compiler, linker, libraries, and an enhanced GDB debugger. The hardware consists of the LPCXpresso development board which has an LPC-Link debug interface and an NXP LPC ARM-based microcontroller target. LPCXpresso is an end-to-end solution enabling embedded engineers to develop their applications from initial evaluation to final production.

The LPCXpresso IDE, powered by Code Red Technologies, is based on the popular Eclipse development platform and includes several LPC-specific enhancements. It is an industry-standard GNU toolchain with an optimized C library that gives engineers all the tools necessary to develop high-quality software solutions quickly and cost-effectively. The C programming environment includes professional-level features. There is syntax coloring, source formatting, function folding, on- and offline help, and extensive project management automation.

5.2 LPCXpresso IDE

LPCXpresso’s IDE is a highly integrated software development environment for NXP’s LPC Microcontrollers, which includes all the tools necessary to develop high quality software solutions in a timely and cost effective fashion. LPCXpresso is based on Eclipse with many LPC specific enhancements. It also features the latest version of the industry standard GNU tool chain with a proprietary optimized C library providing professional quality tools at low cost. The LPCXpresso IDE can build an executable of any size with full code optimization and it supports a download limit of 128 kB after registration. LPCXpresso supports the full embedded product design cycle by moving beyond chip evaluation boards and supporting development on external target boards.

****

**Fig : LPCXpresso IDE**

**5.3 LPCXpresso Development Board**



5.3.1 **LPC-LINK JTAG/SWD debugger**

The LPCXpresso board contains a JTAG/SWD debugger called the “LPC-Link” and a target MCU. LPC-Link is equipped with a 10-pin JTAG header (highlighted on the above image) and it seamlessly connects to the target via USB (the USB interface and other debug features are provided by NXP’s ARM9 based LPC3154 MCU). Cutting the tracks between the LPC-link and the target will make the LPC-Link a stand-alone JTAG debugger. This enables the LPCXpresso platform to be connected to an external target and used to develop for a wide variety of NXP’s Cortex-M0, Cortex-M3, and ARM7/9 based applications. Currently supported microcontroller products include LPC1700, LPC1300, LPC1200, and LPC1100 series and specific members of the LPC2000 and LPC3000 families.

**5.3.2** **Integrated evaluation target**

The target includes a small prototyping area and easily accessible connections for expansion. The LPCXpresso board with target can be used

* On its own for software development and benchmarking
* Connected to an off-the-shelf baseboard for rapid proof-of-concepts
* Connected to customer-designed board for a full prototype

**5.4. SYSTEM REQUIREMENTS**

|  |  |
| --- | --- |
| Operating System | Microsoft® Windows - XP 32-bit or 64-bit (SP2 or greater) Microsoft® Windows - Vista 32-bit or 64-bit Microsoft® Windows - Windows 7 32-bit or 64-bit  Linux - Ubuntu 9 and 10  Linux - Fedora Core 12 and 13 |
| System RAM | 512 MB minimum (1 GB recommended) |
| Hard Disk | 225 MB of available space. |
| Screen/Display Adaptor | 1024x768 minimum recommended |
| Internet Connection | High-speed internet is recommended to download and register the software |

**5.5 ACTIVATION**

To activate your product from LPCXpresso, choose Help->Product activation->Create serial number and register. Once the wizard is open, click “Copy to clipboard” to copy the LPCXpresso serial number into the clipboard. This serial number is based on your machine’s hardware and operating system configuration, but contains no personally identifiable information. Now click the button to open the registration activation page. This should display a web form. After completing the form, you will receive an activation code via email within a few minutes. Highlight the activation code in your email program, and select Copy to place it into the Windows clipboard. Now, choose select Help->Product activation->Enter activation code from within LPCXpresso. Paste the product activation code into the Product activation dialog by right clicking in the Activation code field and choosing “Paste.” Then click the “OK” button. You should receive a dialog confirming acceptance of the activation code. It is also possible to complete LPCXpresso activation on a PC that is offline as long as another PC has access to the Internet.