**CHAPTER 1**

**INTRODUCTION**

Today embedded systems are inventions that have made the world to think and bring out different innovations and developments in the field of automobiles. Embedded System is basically a system of hardware and software designed for control and access of data. As a system it includes a controller as the brain. This controller chip can be MSP430. Today embedded systems applications are popular and their development has become a boon to automobile industry. Applications in the form of home appliances, automobiles, entertainments, consumer electronics, medical and telecommunication etc., are every day in the mode of development. Mechanical systems in automobiles are largely replaced by electronic systems. Today Automobile industry is making great use of embedded systems. Ranging from wiper controls to complex anti-lock brake controls and air bags, embedded systems have gained the overall control of recent automobiles. The automobiles that are built around using MSP430 controllers, digital signal processors or using both the processors are commonly called as Electronic Control Units. Some of the current trends of embedded systems in automobiles include airbag controllers, navigation systems, satellite radio, adaptive cruise control, drive by wire, heads up displays etc.

With rapidly changing computer and information technology and much of the technology finding way into vehicles. They are undergoing dramatic changes in their capabilities and how they interact with the drivers. Although some vehicles have provisions for deciding to either generate warnings for the human driver or controlling the vehicle autonomously, they usually must make these decisions in real time with only incomplete information. So, it is important that human drivers still have some control over the vehicle. Advanced in-vehicle information systems provide vehicles with different types and levels of intelligence to assist the driver. The introduction into the vehicle design has allowed an almost symbiotic relationship between the driver and vehicle by providing a sophisticated and intelligent driver-vehicle interface through an intelligent information network. This project aims to develop such a control framework for the vehicle which is called as digital-driving behavior, consists of a joint mechanism between the driver and vehicle for perception, decision making and control.

A vehicle was generally built with an analog driver vehicle interface for indicating various parameters of vehicle status like temperature, pressure and speed etc. To improve the driver vehicle interface, an interactive digital system is designed. A MSP430 controller based data acquisition system that uses ADC to bring all control data from analog to digital format is used. Since the in-vehicle information systems are spread out all over the body of a practical vehicle, a communication module that supports to implement a step control of the vehicle through the master controller of the digital driving system.

* 1. **MSP430G2553:**

The MSP430 is a [mixed-signal](https://en.wikipedia.org/wiki/Mixed-signal_integrated_circuit) [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) family from [Texas Instruments](https://en.wikipedia.org/wiki/Texas_Instruments). Built around a [16-bit](https://en.wikipedia.org/wiki/16-bit) [CPU](https://en.wikipedia.org/wiki/CPU), the MSP430 is designed for low cost and specifically, low power consumption embedded applications.

The MSP430 can be used for low powered [embedded devices](https://en.wikipedia.org/wiki/Embedded_devices). The [current](https://en.wikipedia.org/wiki/Electric_current) drawn in idle mode can be less than 1 µA. The top CPU speed is 25 MHz. It can be throttled back for lower power consumption. The MSP430 also uses six different low-power modes, which can disable unneeded clocks and CPU. Additionally, the MSP430 is capable of wake-up times below 1 µs, allowing the microcontroller to stay in sleep mode longer, minimizing its average current consumption.

The device comes in a variety of configurations featuring the usual peripherals: internal [oscillator](https://en.wikipedia.org/wiki/Oscillator), [timer](https://en.wikipedia.org/wiki/Timer) including [PWM](https://en.wikipedia.org/wiki/Pulse-width_modulation), [watchdog](https://en.wikipedia.org/wiki/Watchdog_timer), [USART](https://en.wikipedia.org/wiki/UART), [SPI](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus), [I²C](https://en.wikipedia.org/wiki/I%C2%B2C), 10/12/14/16/24-bit [ADCs](https://en.wikipedia.org/wiki/Analog-to-digital_converter), and [brownout](https://en.wikipedia.org/wiki/Brownout_(electricity)) reset [circuitry](https://en.wikipedia.org/wiki/Circuitry). Some less usual peripheral options include [comparators](https://en.wikipedia.org/wiki/Comparator) (that can be used with the timers to do simple ADC), on-chip [op-amps](https://en.wikipedia.org/wiki/Operational_amplifier) for [signal conditioning](https://en.wikipedia.org/wiki/Signal_conditioning), 12-bit [DAC](https://en.wikipedia.org/wiki/Digital-to-analog_converter), [LCD](https://en.wikipedia.org/wiki/Liquid_crystal_display) driver, [hardware multiplier](https://en.wikipedia.org/wiki/Hardware_multiplier), [USB](https://en.wikipedia.org/wiki/USB), and [DMA](https://en.wikipedia.org/wiki/Direct_memory_access) for ADC results. Apart from some older [EPROM](https://en.wikipedia.org/wiki/EPROM) (MSP430E3xx) and high volume [mask ROM](https://en.wikipedia.org/wiki/Mask_ROM) (MSP430Cxxx) versions, all of the devices are [in-system programmable](https://en.wikipedia.org/wiki/In-System_Programming) via [JTAG](https://en.wikipedia.org/wiki/JTAG) (full four-wire or [Spy-Bi-Wire](https://en.wikipedia.org/wiki/Spy-Bi-Wire)) or a built in [bootstrap](https://en.wikipedia.org/wiki/Bootstrap) loader (BSL) using [UART](https://en.wikipedia.org/wiki/UART) such as [RS232](https://en.wikipedia.org/wiki/RS232), or [USB](https://en.wikipedia.org/wiki/USB) on devices with USB support.

However, limitations that preclude its use in more complex embedded systems. The MSP430 does not have an external [memory bus](https://en.wikipedia.org/wiki/Memory_bus), so it is limited to on-chip memory (up to 512 KB [flash memory](https://en.wikipedia.org/wiki/Flash_memory) and 66 KB [RAM](https://en.wikipedia.org/wiki/Random-access_memory)) which may be too small for applications that require large buffers or data tables. Also, although it has a DMA controller, it is very difficult to use it to move data off the chip due to a lack of a DMA output strobe.

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 ANALYSIS ON EXISTING METHOD**

[Pradhan Suvendu Kedareswar](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Pradhan%20Suvendu%20Kedareswar.QT.&newsearch=true) (2015) proposed to take decision and automatically control the motion of the following vehicle, a collision avoidance system consisting of Cortex MO microcontroller is implemented that warns the drier using a buzzer and messages the active and passive safety mechanisms to be activated using CAN protocol and takes control decisions according to an algorithm designed to handle the situation.

Vijayalakshmi.S (2013) proposed the communication module used in this project is embedded networking by CAN which has efficient data transfer .It also takes feedback of vehicle conditions like Vehicle speed, Engine temperature etc., and controlled by main controller.

Theophilus Wellem (2010) proposed in this paper, we designed and implemented a microcontroller-based room temperature monitoring system using Atmel ATmega8535 microcontroller and National Semiconductor’s LM35 temperature sensor.

William Prodanov (2009) proposed a behavioral model of a CAN bus transceiver is proposed and experimentally verified. Moreover, we developed a methodology to efficiently manage the trade-off concerning accuracy, simulation speed, and convergence issues which are usually involved in the simulation of large CAN bus networks.

Alberto Sangiovanni-Vincentelli (2007) described about to optimize the system design and allow for plug and play of subsystems, automotive electronic system architecture evaluation and development must be supported with a robust design flow based on virtual platforms.

Tindell.K (2006) proposed, One of the perceived drawbacks to CAN has been the inability to bound accurately the worst-case response time of a given message (i.e. the longest time between queuing the message and the message arriving at the destination processors). This paper presents analysis to bound such response times, including the costs of error handling and re-transmission.

Nicolas Navet (2005) described about the specific requirements of the different car domains have led to the development of a large number of auto-motive networks such as Local Interconnect Network, J1850, CAN,TTP/C, Flex Ray, media-oriented system transport, IDB1394, etc. This paper first introduces the context of in-vehicle embedded systems and the requirements imposed on the communication systems. Then, a comprehensive review of the most widely used automotive networks, as well as the emerging ones, is given. Next, the current efforts of the automotive industry on middleware technologies, which may be of great help in mastering the heterogeneity, are reviewed.

Keith Pazul (1999) proposed systematic approach enables interoperability between products from different manufacturers. A standard was created by the International Standards Organization (ISO) as a term-plate to follow for this layered approach. It is called the ISO Open Systems Interconnection (OSI) Network Layering Reference Model.

Cook.J.A (1993) proposed a CAN message may contain up to 8 bytes of data. A message identifier describes the data content and is used by receiving nodes to determine the destination on the network. Bitrates up to 1Mbit/s are possible in short networks (1≤40 m). Longer network distances reduce the available bit rate (125kbit/s at 500 m). “High speed” CAN is considered to be 500kbit/s.

John Wiley & sons (2003) proposed the design process of digital automatic control system with PID (Proportional Integral Derivative) controller is considered. The solution of problems related with implementation of PID control algorithm into general purpose 8-bit (1byte) microcontroller is discussed. Simulation results demonstrating performance of system.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING METHOD**

This system is designed to test automotive CAN control devices or entire network. Users can configure several monitoring modes and CAN channel features of the system with PC application. Further more the USB technology adopted in the system make it a more practical and convenient CAN bus testing system.The MCP2551 is used as the main controller of USB-CAN smart card. Its 8-bit data bus interface PORT A is used to communicate with PDIUSBD12 as shown in Figure 3.1. It consists of so many peripherals such as power module, peripheral electric circuit, USB interface module and CAN interface module. Without the help of PC, it cannot monitor the vehicle and it does not work. It requires a separate CAN transceiver for each and every nodes and separate power module is required.

In this project by using a PIC based microcontroller that is interfaced via different controllers via CAN Protocol. And also by using different analog sensors that will continuously monitor the vehicle and sending the values to the microcontroller via ADC interface. There are three analog sensors. Analog sensor used is LM35 that is used to monitor the temperature of the vehicle. And MQ-3 is used to check the CO in the exhaust and the speed of the vehicle respectively.

LDR is used to verify the light intensity of the vehicle. CAN bus is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer. CAN bus is a message-based protocol, designed specifically for automotive applications but now also used in other areas such as aerospace, industrial automation and medical equipment. These sensors used in this project are very reliable and easily available. The interfacing of this sensor to the central unit helps the user to interact with the vehicle more easily.

Personal computer USB-CAN smart card CAN bus CAN nodes

CAN node1

MC9S12DP256

Monitoring system application

PDI

USBD12

Port A

CAN node2

DC-DC converter

DC-DC converter

MS CAN module

USB root hub

USB

port

CAN interface

photo coupler

CAN transceiver

CAN interface

photo coupler

CAN transceiver

Power module

CAN node3

Figure 3.1 Block Diagram Of Existing Method

Digital control of the vehicle is an important criterion of modern technology. With the rapid development of embedded technology, high performance embedded processor is penetrated into the auto industry, which is low cost, high reliability and other features to meet the needs of the modern automobile industry. The proposed high speed CAN bus system solves the problem of automotive system applications, also has a certain practical value and significance .With ARM as the main controller and it makes full use of the high performance of ARM, high speed reduction of CAN bus communication control.

The ARM based data acquisition system that uses ADC to bring all control data from analog to digital format. The communication module used in this project is embedded networking by CAN which has efficient data transfer. The CAN Protocol is necessary for the different control systems (and their sensors) to exchange information.

This is usually done by discrete interconnection of the different systems (i.e. point to point wiring). The requirement for information exchange has then grown to such an extent that a cable network with a length of up to several miles and many connectors was required. The benefits of CAN is effectively implemented in vehicle it is used for achieving automation, over other tradition schemes it will offer increase flexibility and expandability for future technology. Generally a vehicle was built with an analog driver vehicle interface for indicating various vehicle statuses like Speed, Fuel level, Engine temperature etc.

The CAN is provide a high speed and the capacity is high it is capable for handling a large number of parameter with more efficiently .The parameters like engine temperature if the temperature increase above the 600°C the automatically cooling system apply due to this temperature is not exceed, speed measure using RPM sensor if revolution increase up to 70 per minute controller act and to avoid the maximum revolution and to check the fuel level continuously and display in the percentage if fuel level below 20 percent the controller gives buzzer to the driver and fuel level and temperature continuously display on the LCD.

**3.2 PROPOSED METHOD**

Controller Area Network uses in wide variety of applications in automobiles, industries etc.., The CAN module is a serial interface, useful for communicating with other MSP430 controller devices. The protocol is designed for communication within noisy environments.

Unlike a traditional network such as USB or Ethernet, CAN does not send large number of blocks of data. It provides the acknowledgement, whether the data is received or not. It will send the short messages from transmitter to receiver. If the data rate is very low, it automatically discards the data.

MSP430 is a low power consuming microcontroller which has been used in this project for higher efficiency in Battery Life Period. This controller with low supplement can carry out all data analysis and monitoring process in automotive.

**CHAPTER 4**

**DESIGN AND IMPLEMENTATION**

**4.1 PROJECT DESCRIPTION**

In vehicle control system implementation using can protocol is implemented, by using MSP430 microcontroller interfacing can be with different controllers via CAN Protocol. In this project by using different analog sensors that will continuously monitor the vehicle and sending the values to the MSP430 controller via ADC interface as shown in Figure 4.1. There are three analog sensors. Analog sensor used is LM35 that is used to monitor the temperature of the vehicle.

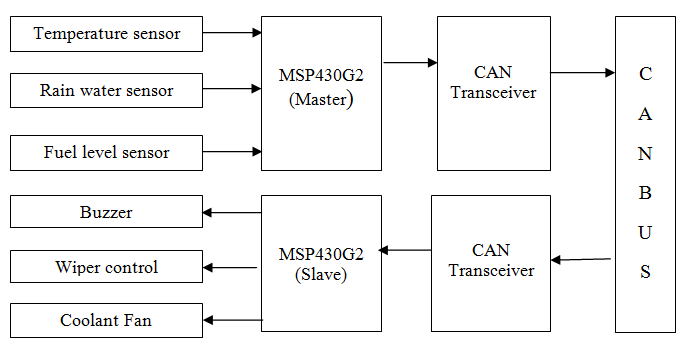
****

Figure 4.1 Block Diagram Of Proposing Method

CAN bus is vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer. CAN bus is a message-based protocol, designed specifically for automotive applications but now also used in other areas such as aerospace, industrial automation and medical equipment. These sensors used in this project are very reliable and easily available. The interfacing of this sensor to the central unit helps the user to interact with the vehicle more easily.

**4.2 CIRCUIT OPERATION**

In transmitter side, 12V supply is given to the voltage regulator and it gives the regulated voltage as 5V. It is used to run the MSP430 microcontroller. Pin 1 of the controller is connected to high supply voltage in order to clear the controller. Every time, when we start it will clear the microcontroller. Colpitts oscillator is used in micro controller in order to maintain the constant frequency. A 10 kΩ resistor is connected to pin 1in transmitter side and it acts as a pull up resistor. 22 µF is used in the controller to reduce the spikes in the transmitter side. Gas sensor is connected to the 4 pin of the controller ,temperature sensor is connected to the 1st pin and LDR is connected to the 1.1st pin of the controller.

In LCD, VSS pin is connected to the supply and VDD is connected to ground. The control signal is RS pin connected to the 2.1th pin of the controller. RW is connected to the 2.2th pin and EN is connected to the 2.3th pin of the controller. The data pins D1 to D4 of the LCD connected to the 2.3th to 2.6th of the controller. And 1.5th pin is connected to the transmitter of mcp2551 transceiver.

In receiver side, similar to transmitter it has a microcontroller. A 12V supply is given to the voltage regulator and it gives the regulated voltage as 5V. It is used to run the MSP430 microcontroller. Pin 1 of the controller is connected to high supply voltage in order to clear the controller. Every time, when we start it will clear the microcontroller. Colpitts oscillator is used in micro controller in order to maintain the constant frequency. A 10 kΩ resistor is connected to pin 1 in transmitter side and it acts as a pull up resistor.

22 µF is used in the controller to reduce the spikes in the receiver side.

1.6th pin is connected to the receiver of the mcp2551 transceiver in the receiver side. And LCD is interface with MSP430 microcontroller as, VSS pin is connected to the supply and VDD is connected to ground. The control signals RS pin is connected to the 2.1th pin of the controller, RW is connected to the 2.2th  pin and EN is connected to the 2.3th of the controller. The data pin D1 to D4 of the LCD is connected to the 27th  to 30th pin of the controller.

**4.3 CAN PROTOCOL**

MSP430 microcontroller Embedded with CAN controller

CAN bus

CAN transceiver

Figure 4.2 CAN Architecture

The CAN Protocol was developed by BOSCH as a multi-master, message broadcast system that specifies a maximum signaling rate of 1 megabit per second (mbps). CAN is an International Standardization Organization (ISO) defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The specification calls for high immunity to electrical interference and the ability to self-diagnose and repair data errors. These features have led to CAN’s popularity in a variety of industries including building automation, medical and manufacturing.

CAN controller

CAN controller

CAN transceiver

CAN transceiver

CAN L CAN H CAN L CAN H

CAN BUS

CAN Bus

Figure 4.3 Conceptual Architecture

The CANL output drives the Low side of the CAN differential bus. From Figure 4.3 the pin is also tied internally to receive the input comparator. The CANH output drives the high-side of the CAN differential bus. This pin is also tied internally to receive the input comparator.

**4.3.1 CAN Bus Module Features**

* Complies with ISO CAN Conformance Test.
* Message bit rates up to 1 Mbps.
* Conforms to CAN 2.0B Active Spec with:
  + 29-bit Identifier Fields.
  + 8-byte message length.
  + 3 Transmit Message Buffers with prioritization.
  + 2 Receive Message Buffers.
  + 6 full, 29-bit Acceptance Filters.
  + Prioritization of Acceptance Filters.
  + Multiple Receive Buffers for High Priority.
* Messages to prevent loss due to overflow.
* Advanced Error Management Features.

**4.4 CAN MODULE OVERVIEW**

The Controller Area Network(CAN) module is a serial interface, useful for communicating with other peripherals or microcontroller devices. This interface protocol was designed to allow communications within noisy environments. The CAN bus module consists of a protocol engine and message buffering and control. The CAN protocol engine handles all functions for receiving and transmitting messages on the CAN bus.

Messages are transmitted by first loading the appropriate data registers. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against filters to see if it should be received and stored in one of the two receive registers.

The CAN module supports the following frame types:

* Standard Data Frame.
* Extended Data Frame.
* Remote Frame.
* Overloaded Frame.
* Error Frame.
* Interframe Space.

**4.4.1 Standard Data Frame**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S  O  F | 11 Bit Identifier | R  T  R | I  D  E | r0 | D  L  C | 0…8 Bytes Data | CRC | ACK | E  O  F | I  F  S |

Figure 4.4 Frame Format For CAN

The various standard Data Frames are explained as follows as shown in Figure 4.4,

* **SOF**–The single dominant Start Of Frame (SOF) bit marks the start of a message,and is used to synchronize the nodes on a bus after being idle.
* **Identifier**-The Standard CAN 11-bit Identifier establishes the priority of the message. The lower the binary value, the higher its priority.
* **RTR**–The single Remote Transmission Request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way, all data being used in a system is uniform.
* **IDE**–A dominant single IDentifier Extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.
* **r0** – Reserved bit (for possible use by future standard amendment).
* **DLC**–The 4-bit Data Length Code (DLC) contains the number of bytes of data being transmitted.
* **ACK**–Every node receiving an accurate message overwrites this recessive bit in the original message with a dominate bit, indicating an error-free message has been sent. If the receiving node detects an error and leaves this bit recessive, it discards the message and the sending node repeats the message after rearbitration. In this way, each node acknowledges (ACK) the integrity of its data. ACK is two bits, one is the acknowledgment bit and the second is a delimiter.
* **EOF**–This End-Of-Frame (EOF), 7-bit field marks the end of a CAN frame (message) and disables bit-stuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is stuffed into the data.
* **IFS**–This 7-bit Inter Frame Space (IFS) contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area.

**4.4.2 Remote Frame**

The purpose of remote frame is used to request transmission of a corresponding DATA FRAME from a remote node. The remote frame is similar to data frame, with two important differences. This type of message is explicitly marked as a remote frame by a recessive RTR bit in the arbitration field, and there will be no data.

**4.4.3 Error Frame**

The error frame is a special message that violates the formatting rules of a CAN message. It is transmitted when a node detects an error in a message, and causes all other nodes in the network to send an error frame as well. The original transmitter then automatically retransmits the message.

**4.4.4 Overload Frame**

The overload frame is similar to the error frame with regard to the format, and it is transmitted by a node that becomes too busy. It is primarily used to provide for an extra delay between messages.

**4.4.5 Inter Frame Space**

Data frame and remote frame are separated by an interframe space. The time required by the controller to move a correctly received frame to its proper position in a message buffer area.

**4.5 CAN Bus Arbitration**

A fundamental CAN characteristic shown is the opposite logic state between the bus, and the driver input and receiver output. Normally, a logic-high is associated with a one, and a logic-low is associated with a zero but not so on a CAN bus. This is why TI CAN transceivers have the driver input and receiver output pins passively pulled high internally, so that in the absence of any input, the device automatically defaults to a recessive bus state on all input and output pins.

Dominant

VDIFF

CANH

Voltage Level (V)

Recessive Recessive

CANL

Time (t)

Figure 4.5 CAN Bus Arbitration

Bus access is event-driven and takes place randomly. If two nodes try to occupy the bus simultaneously, access is implemented with a nondestructive, bit-wise arbitration. Nondestructive means that the node winning arbitration just continues on with the message, without the message being destroyed or corrupted by another node. The allocation of priority to messages in the identifier is a feature of CAN that makes it particularly attractive for use within a real-time control environment.

The lower the binary message identifier number, the higher its priority. An identifier consisting entirely of zeros is the highest priority message on a network because it holds the bus dominant the longest as shown in Figure 4.5. Therefore, if two nodes begin to transmit simultaneously, the node that sends a last identifier bit as a zero (dominant) while the other nodes send a one (recessive) retains control of the CAN bus and goes on to complete its message. A dominant bit always overwrites a recessive bit on a CAN bus.

C wins B wins

Arbitration Arbitration

Node C

Transmits

Node B

Transmits

CAN Bus

Figure 4.6 CAN Bus Arbitration in nodes

Note that a transmitting node constantly monitors each bit of its own transmission. This is the reason for the transceiver configuration is in which the CANH and CANL output pins of the driver are internally tied to the receiver's input. The propagation delay of a signal in the internal loop from the driver input to the receiver output is typically used as a qualitative measure of a CAN transceiver. This propagation delay is referred to as the loop time (tloop in a TI data sheet), but takes on varied nomenclature from vendor to vendor.

It displays the CAN arbitration process that is handled automatically by a CAN controller. Because each node continuously monitors its own transmissions, as node B's recessive bit is overwritten by node C’s higher priority dominant bit, B detects that the bus state does not match the bit that it transmitted.

Therefore, node B halts transmission while node C continues on with its message. Another attempt to transmit the message is made by node B once the bus is released by node C as shown in Figure 4.6. This functionality is part of the ISO 11898 physical signaling layer, which means that it is contained entirely within the CAN controller and is completely transparent to a CAN user.

The allocation of message priority is up to a system designer, but industry groups mutually agree on the significance of certain messages. For example, a manufacturer of motor drives may specify that message 0010 is a winding current feedback signal from a motor on a CAN network and that 0011 is the tachometer speed. Because 0010 has the lowest binary identifier, messages relating to current values always have a higher priority on the bus than those concerned with tachometer readings.

**4.6 CAN Error Management**

The main advantage of CAN protocol is its error detection and error handling mechanisms. The following errors can be detected

* CRC error.
* Acknowledge error.
* Form error.
* Bit error.
* Stuff bit error.

**4.6.1 CRC Error**

With the Cyclic Redundancy Check (CRC), the transmitter calculates special Check bits for the bit sequence, from the start of a frame until the end of the data field. This CRC sequence is transmitted in the CRC field. The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence. If a mismatch is detected, the CRC error has occurred and an error frame is generated. The message is repeated.

**4.6.2 Acknowledge Error**

In the Acknowledge field of a message, the transmitter checks if the Acknowledge slot (which was sent out as a recessive bit) contains a dominant bit. If not, no other node has received the frame correctly. An Acknowledge error has occurred, an error frame is generated and the message will have to be repeated.

**4.6.3 Form Error**

If a node detects a dominant bit in one of the four segments, including end of frame,interframe space,Acknowledge delimiter or CRC delimiter,then a form error has occurred and an error frame is generated. The message is repeated.

**4.6.4 Bit Error**

A Bit Error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit, when monitoring the actual bus level and comparing it to the just transmitted bit. In the case where the transmitter sends a recessive bit and a dominant bit is detected during the arbitration field and the Acknowledge slot, no Bit Error is generated because normal arbitration is occurring.

**4.6.5 Stuff Bit Error**

If, between the start of frame and the CRC delimiter, six consecutive bits with the same polarity are detected,the bit stuffing rule has been violated. A Stuff Bit Error occurs and an error frame is generated. The message is repeated.

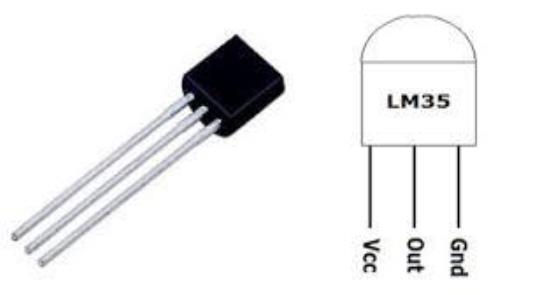
**4.7 TEMPERATURE SENSOR**

Figure 4.7 LM35 Temperature sensor

The LM35 is an integrated circuit sensor that can be used to measure temperature with an electrical output proportional to the temperature (in °C). It measures temperature more accurately than a using a thermistor. The sensor circuitry is sealed and not subject to oxidation, etc. The LM35 generates a higher output voltage than thermocouples and may not require that the output voltage be amplified. It has an output voltage that is proportional to the Celsius temperature. The scale factor is .01V/°C.

The LM35 does not require any external calibration or trimming and maintains an accuracy of +/-0.4°C at room temperature and +/- 0.8°C over a range of 0°C to +100°C. Another important characteristic of the LM35 is that it draws only 60µA from its supply and possesses a low self-heating capability. The sensor self-heating causes less than 0.1 °C temperature rise in still air. The sensor has a sensitivity of 10mV / °C.

**4.7.1 Applications**

* The LM35 is applied easily in the same way as other integrated-circuit temperature sensors. Glue or cement the device to a surface and the temperature should be within about 0.01°C of the surface temperature.
* To minimize this problem, ensure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest.
* The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the temperature of the LM35 is not affected by the air temperature.

**4.8 LDR**

A photo resistor or light-dependent resistor (LDR)or photocell is a light controlled variable resistor works on the principle of photo conductivity. Photo conductivity is an optical phenomenon in which the materials conductivity reduces when light is absorbed by the material. When light falls i.e., photons fall on the device, the electrons in the valence band of the semiconductor material are excited to the conduction band.

These photons in the incident light should have energy greater than the band gap of the semiconductor material to make the electrons jump from the valance band to the conduction band. Hence when light having enough energy is incident on the device more and more electrons are excited to the conduction band which results in large number of charge carriers.

**4.9 LCD**

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special and even custom characters (unlike in seven segments), animations and so on.

A 16x2 LCDmeans it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

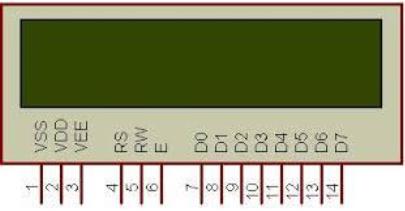


Figure 4.8 LCD Pin Diagram

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

**4.10 ENERGIA IDE –An Integrated Development Environment**

Energia is an [open-source](https://github.com/energia/energia) electronics prototyping platform started by Robert Wessels in January of 2012 with the goal to bring the Wiring and Arduino framework to the Texas Instruments MSP430 based LaunchPad. The Energia IDE is cross platform and supported on Mac OS, Windows, and Linux. Energia uses the [mspgcc](http://mspgcc.sourceforge.net/)  compiler by [Peter Bigot](http://sourceforge.net/users/pabigot) and is based on the [Wiring](http://wiring.org.co/) and [Arduino](http://arduino.cc/)  framework. Energia includes an integrated development environment (IDE) that is based on [Processing](http://www.processing.org/).  Energia is also a portable framework/abstraction layer that can be used in other popular IDEs. Utilize a web browser based environment with CCS Cloud at [dev.ti.com](http://dev.ti.com). Community maintained Energia plug-ins and integrations are available for [Xcode, Visual Studio, and Code Composer Studio](http://energia.nu/download/#communitycontributedalternatives).

The foundation of [Energia](http://energia.nu/) and [Arduino](http://arduino.cc/) is the [Wiring](http://wiring.org.co/) framework that is developed by Hernando Barragan. The framework is thoughtfully created with designers and artists in mind to encourage a community where both beginners and experts from around the world share ideas, knowledge and their collective experience. The Energia team adopts the philosophy of learning by doing and strives to make it easy to work directly with the hardware. Professional engineers, entrepreneurs, makers and students can all benefit from the ease of use Energia brings to the microcontroller.

Energia started out to bring the Wiring and Arduino framework to the Texas Instruments MSP430 LaunchPad. Texas Instruments offers a [MSP430](http://www.ti.com/ww/en/launchpad/msp430_head.html), [MSP432](http://www.ti.com/ww/en/launchpad/msp430_head.html), [TM4C](http://www.ti.com/ww/en/launchpad/launchpads-connected.html), [C2000](http://www.ti.com/ww/en/launchpad/c2000_head.html) and [CC3200](http://www.ti.com/ww/en/launchpad/launchpads-connected.html) LaunchPad. The LaunchPad is a low-cost microcontroller board that is made by Texas Instruments. The [latest](http://energia.nu/download) release of Energia supports the majority of the LaunchPad product offerings. Additional community kits from [RedBearLab](http://redbearlab.com) are also supported.

Together with Energia, LaunchPad can be used to develop interactive objects, taking inputs from a variety of switches or sensors and controlling a variety of lights, motors and other physical outputs. LaunchPad projects can be stand-alone (only run on the Target Board, i.e. your LaunchPad) or they can communicate with software running on your computer (Host PC). You can also add wireless modules to enable communication over various types of RF including Wi-Fi, NFC, Bluetooth, Zigbee, cellular and more.

**4.11 SYSTEM IMPLEMENTATION**

Initially, the supply from the transformer is given to the microcontroller and the relay circuit. The 230V from the transformer is converted to 5V using rectifier and voltage regulator, this 5V given to the controller. The voltage regulator 7805 is used to give 5V constant DC supply. The capacitor of about 1000 µF is used as a filter. The relay circuit is used to isolate low power circuit from high power circuit. The diode IN4007 is used in relay circuit to protect controller from back emf.

There are master and slave modules in this vehicle control system implementation. In master module, sensor has been interfaced. Analog sensors like Temperature sensor and Rain water sensor which are connected to ADC ports, where the Analog values are converted into digital. MSP430G2553 has a 10 bit ADC, so the maximum precision is 1023 as digital value. Digital sensor like Fuel level sensor is used.

These values from different sensors are transmitted through CAN frames using CAN transceiver MCP2551 which supports 1Mbps operation. One CAN transceiver at the transmitter end, which is on Master controller and other CAN transceiver at the receiver end, which is on Slave controller. These CAN transceivers are interconnected by using CAN bus.

In slave module, the MSP430G2553 controller is interfaced with MAX232 which is a dual transmitter / dual receiver that converts signals from a [TIA-232](https://en.wikipedia.org/wiki/RS-232) (RS-232) serial port to signals suitable for use in [TTL](https://en.wikipedia.org/wiki/Transistor-transistor_logic)-compatible digital logic circuits. It is an alternate serial port output of the sensor values.

CAN converts the data’s from sensors into frames and transmit through a wired channel which is CAN bus. The serial monitor in ENERGIA can display the digital values of the sensors, provided COM port of the USB and device should be similar, Baud rate of the controller terminal should be 9600Hz and tool selection should be made. Through receiver which is slave module the data’s are displayed in serial monitor.

**CHAPTER 5**

**RESULT AND DISCUSSION**

The vehicle control system implementation using CAN protocol with MCP430G2553 full hardware setup is shown below Figure 5.1 without power supply (i.e. power OFF condition).

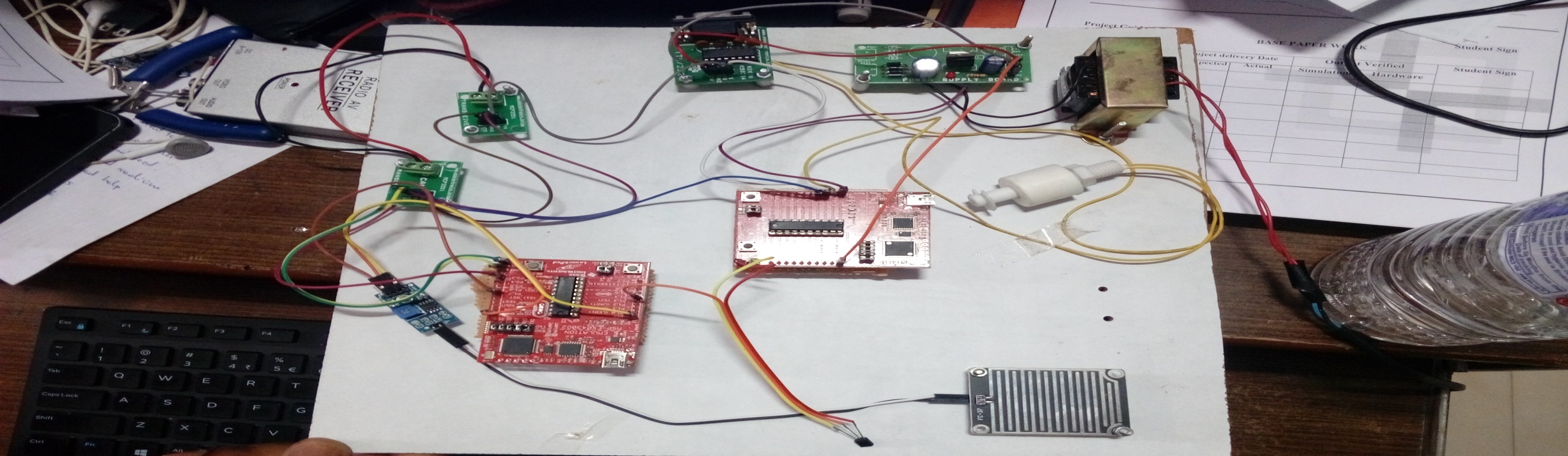
****

Figure 5.1 Complete Hardware Setup (power OFF condition)

The complete hardware setup of the vehicle control system is shown in the Figure 5.2 with power supply (i.e. power ON condition). The vehicle can be monitored by using the sensors (Temperature sensor, Fuel level sensor, Rain water sensor).

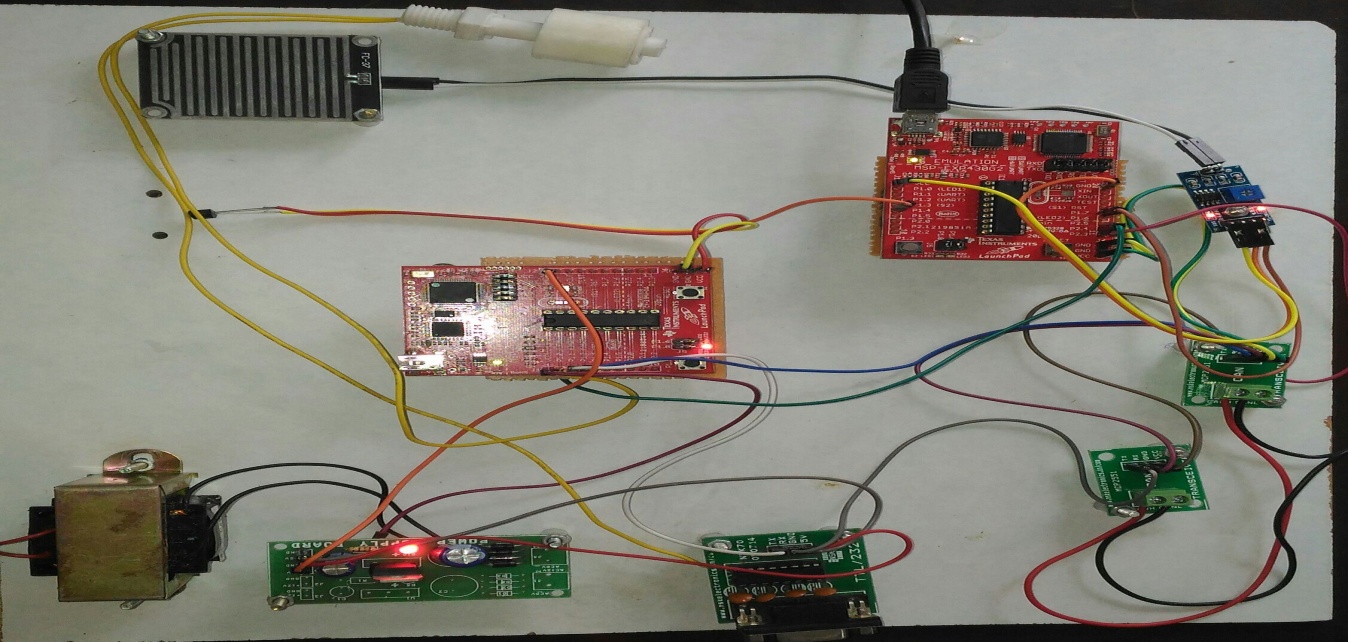


Figure 5.2 Complete Hardware Setup (power ON condition)

The Figure 5.3 shown in below is the serial monitor output of the Energia IDE which displays the sensor values and its responses using can transceivers in the rate of 1 Mbps.

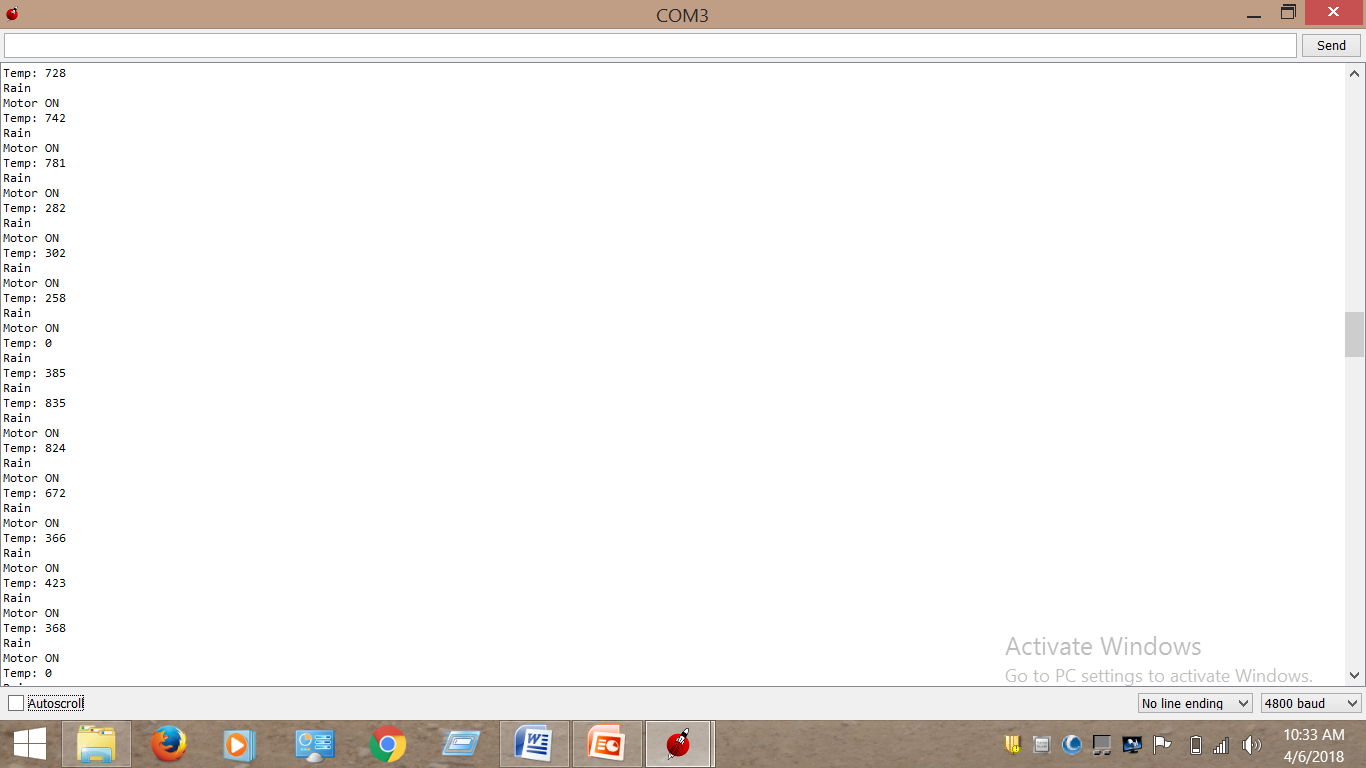


Figure.5.3 Serial Monitor Output Screen Shot.

**CHAPTER 6**

**CONCLUSION**

Project on Vehicle Control System Implementation Using Can Protocol describes about the implementation of CAN bus in automobile industries. Instead of using multiple wires, two wires are used. Thus the requirement of wires and complexity are reduced. Because of wide usage of vehicles and rapid development of embedded technology, CAN protocol is used for data transmission and it is known for its low cost, high reliability and better error handling mechanism. Vehicle conditions are monitored by CO sensor, LM35 for temperature measurement, 12V battery and LDR to measure the light intensity. The main advantage of using this CAN protocol is its high speed data rate. The programming for the microcontroller interfacing with CAN protocol is verified using a general purpose board. The temperature of engine, battery voltage, presence of head light, and CO level in the exhaust are transferred from engine side to dashboard side via CAN protocol and these readings are displayed through LCD on the dashboard.

**APPENDIX**

**A.1 MSP430G2553**

**A.1.1 MSP430G2553 PIN DIAGRAM:**

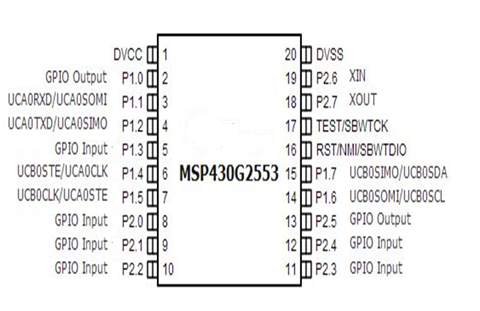
****

Figure A.1 MSP430G2553 Pin Diagram

**A.1.2 FEATURES OF MSP430G2:**

• Low Supply-Voltage Range: 1.8 V to 3.6 V

• Universal Serial Communication Interface (USCI)

– Enhanced UART Supporting Auto Baud rate Detection (LIN)

– IrDA Encoder and Decoder

– Synchronous SPI

• Ultra-Low Power Consumption

– Active Mode: 230 µA at 1 MHz, 2.2 V

– Standby Mode: 0.5 µA

– Off Mode (RAM Retention): 0.1 µA

• Five Power-Saving Modes

• Ultra-Fast Wake-Up from Standby Mode in C™ Less than 1 µs

• On-Chip Comparator for Analog Signal Compare Function or Slope Analog-to-Digital

• 16-Bit RISC Architecture, 62.5-ns Instruction (A/D) Conversion Cycle Time

• 10-Bit 200-ksps Analog-to-Digital (A/D)

• Basic Clock Module Configurations Converter with Internal Reference, Sample

– Internal Frequencies up to 16 MHz With and-Hold and Auto scan Four Calibrated Frequency

– Internal Very-Low-Power Low-Frequency

– 32-kHz Crystal

– External Digital Clock Source Fuse

• Brownout Detector

• Serial Onboard Programming, (LF) Oscillator No External Programming Voltage Needed, Programmable Code Protection by Security

• Two 16-Bit Timer A with Three

• On-Chip Emulation Logic with Spy-Bi-Wire Capture/Compare Registers Interface

• Up to 24 Capacitive-Touch enabled I/O Pins

**A.1.3 PIN CONFIGURATION MSP430G2553 LAUNCHPAD:**

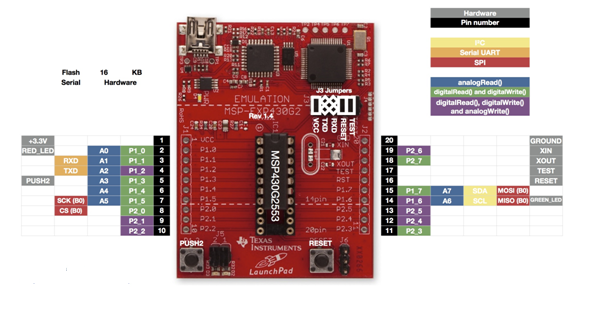


Figure A.2 MSP430G2553 Pin Configuration

**A.2 APPLICATIONS OF MSP-EXP430G2553**

The MSP430 can be used for low powered [embedded devices](https://en.wikipedia.org/wiki/Embedded_devices). The [current](https://en.wikipedia.org/wiki/Electric_current) drawn in idle mode can be less than 1 µA. The top CPU speed is 25 MHz. It can be throttled back for lower power consumption. The MSP430 also uses six different low-power modes, which can disable unneeded clocks and CPU. Additionally, the MSP430 is capable of wake-up times below 1µs, allowing the microcontroller to stay in sleep mode longer, minimizing its average current consumption.

The device comes in a variety of configurations featuring the usual peripherals:internal [oscillator](https://en.wikipedia.org/wiki/Oscillator), [timer](https://en.wikipedia.org/wiki/Timer)  including  [PWM](https://en.wikipedia.org/wiki/Pulse-width_modulation), [watchdog](https://en.wikipedia.org/wiki/Watchdog_timer), [USART](https://en.wikipedia.org/wiki/UART),  [SPI](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus), [I²C](https://en.wikipedia.org/wiki/I%C2%B2C), 10/12/14/16/24-bit [ADCs](https://en.wikipedia.org/wiki/Analog-to-digital_converter) and [brown out](https://en.wikipedia.org/wiki/Brownout_(electricity)) reset [circuitry](https://en.wikipedia.org/wiki/Circuitry). Some less usual peripheral options include [comparators](https://en.wikipedia.org/wiki/Comparator) (that can be used with the timers to do simple ADC), on-chip [op-amps](https://en.wikipedia.org/wiki/Operational_amplifier) for [signal conditioning](https://en.wikipedia.org/wiki/Signal_conditioning), 12-bit [DAC](https://en.wikipedia.org/wiki/Digital-to-analog_converter), [LCD](https://en.wikipedia.org/wiki/Liquid_crystal_display) driver, [hardware multiplier](https://en.wikipedia.org/wiki/Hardware_multiplier), [USB](https://en.wikipedia.org/wiki/USB) and [DMA](https://en.wikipedia.org/wiki/Direct_memory_access) for ADC results. Apart from some older  [EPROM](https://en.wikipedia.org/wiki/EPROM) (MSP430E3xx) and high volume [mask ROM](https://en.wikipedia.org/wiki/Mask_ROM) (MSP430Cxxx) versions, all of the devices are [in-system programmable](https://en.wikipedia.org/wiki/In-System_Programming) via [JTAG](https://en.wikipedia.org/wiki/JTAG) (full four-wire or [Spy-Bi-Wire](https://en.wikipedia.org/wiki/Spy-Bi-Wire)) or a built in [bootstrap](https://en.wikipedia.org/wiki/Bootstrap) loader (BSL) using [UART](https://en.wikipedia.org/wiki/UART) such as [RS232](https://en.wikipedia.org/wiki/RS232), or [USB](https://en.wikipedia.org/wiki/USB) on devices with USB support.

However, limitations that preclude its use in more complex embedded systems. The MSP430 does not have an external [memory bus](https://en.wikipedia.org/wiki/Memory_bus), so it is limited to on-chip memory (up to 512 KB [flash memory](https://en.wikipedia.org/wiki/Flash_memory) and 66 KB [RAM](https://en.wikipedia.org/wiki/Random-access_memory)) which may be too small for applications that require large buffers or data tables. Also, although it has a DMA controller, it is very difficult to use it to move data off the chip due to a lack of a DMA output strobe.

**A.3 VOLTAGE REGULATOR**

A voltage stabilizer is any device that keeps the voltage of a circuit at a specified level. There are many different types of voltage stabilizers but integrated circuit (IC) voltage stabilizers are among the most common. You will frequently need a voltage stabilizer for components that require regulated power. You can demonstrate the use of a voltage stabilizer in a circuit with a few components from an electronics parts store.

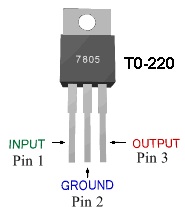


Figure A.3 7805 Voltage Regulator

Identify the parts of the voltage regulator. Place the voltage regulator so that you can read the printing on it. The digits "78" indicate a positive voltage regulator and the digits "05" indicate a 5-volt regulator. For a positive voltage regulator like a 7805, the left lead is the input, the middle lead is the ground and the right lead is the output. Mount the voltage regulator on the mounting board. Each of the voltage regulator's three leads should be inserted into a different hole in the mounting board so that the three holes are in the same column but different rows. Mount the light bulb on the mounting board. Insert the lead for the light bulb's positive terminal into a hole on the same row as the voltage regulator's output lead. Insert the light bulb's negative lead into a hole on the same row as the voltage regulator's ground lead.

Insert the positive lead of the battery holder into a hole on the same row as the voltage regulator's input. Insert the negative lead of the battery holder into a hole on the same row as the voltage regulator's ground and the light bulb's negative lead. Place the battery in the battery holder. The light bulb now receives a stable voltage of 5 volts even though the power supply is a 9-volt battery. This type of voltage regulator will dump the excess voltage as heat.

Table A.1 7805 Pin Description

|  |  |  |
| --- | --- | --- |
| **PIN No.** | **PIN** | **DESCRIPTION** |
| 1 | INPUT | In this pin of the IC positive unregulated voltage is given in regulation. |
| 2 | GROUND | In this pin where the ground is given. This pin is neutral for equally the input and output. |
| 3 | OUTPUT | The output of the regulated 5V volt is taken out at this pin of the IC regulator. |

**A.4 ARCHITECTURE OF MSP430G2553**

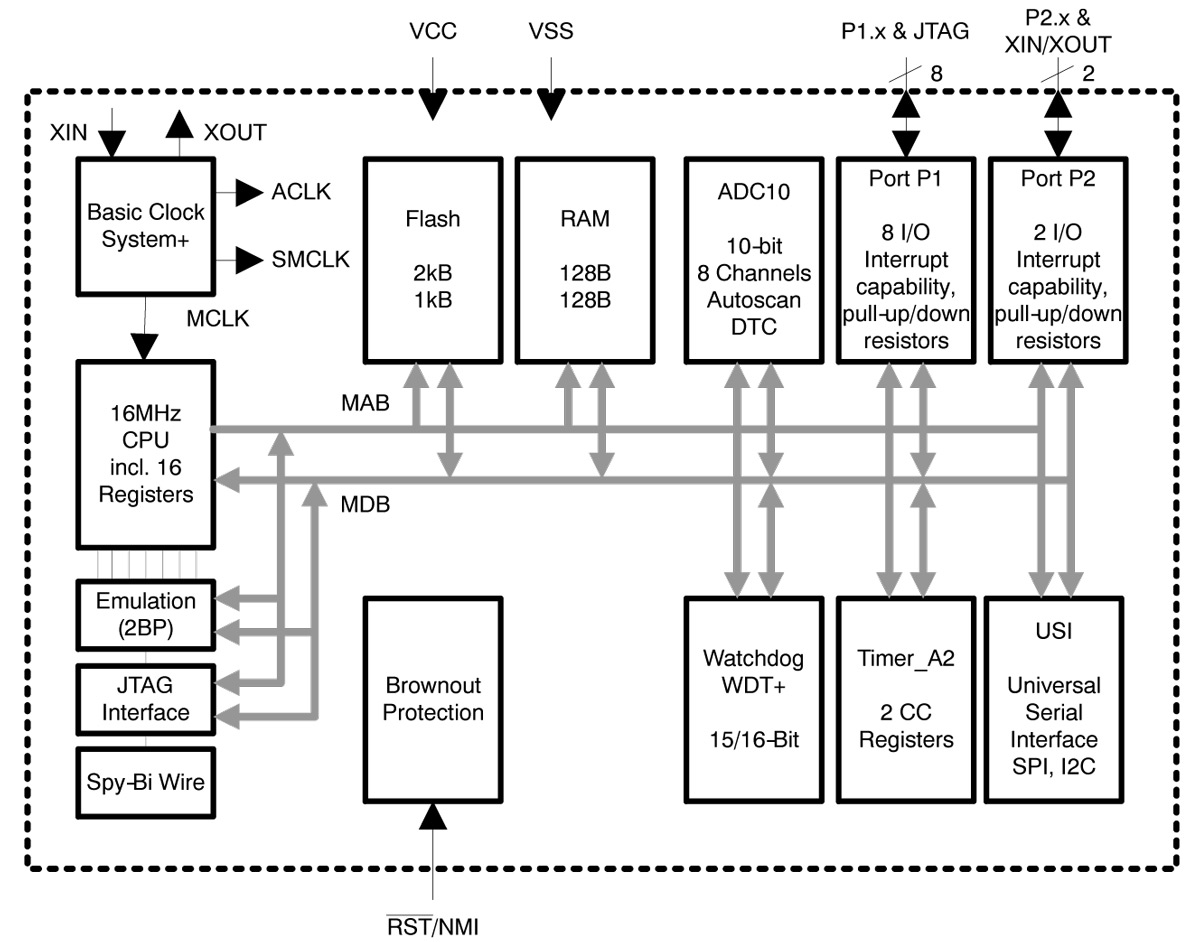


Figure A.4 Architecture of MSP430G2553

**A.5 CAN TRANSCEIVER**

**A.5.1 Features**

* Supports 1 Mb/s operation.
* Implements ISO-11898 standard physical layer.
* Suitable for 12V and 24V systems.
* Externally-controlled slope for reduced RFI emissions.
* Detection of ground fault (permanent Dominant) on TXD input.
* Power-on Reset and voltage brown-out protection.
* An unpowered node or brown-out event will not disturb the CAN bus.
* Low current standby operation.
* Protection against damage due to short-circuit conditions (positive or negative battery voltage).
* Protection against high-voltage transients.
* Automatic thermal shutdown protection.
* Up to 112 nodes can be connected.
* High-noise immunity due to differential bus implementation.
* Temperature ranges:

1. Industrial (I): -40°C to +85°C

2. Extended (E): -40°C to +125°C

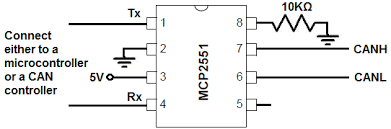


Figure A.5 Pin Diagram For MCP2551

**A.5.2 CAN Transceiver Overview**

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 device 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 1 Mb/s. 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 (differential output). 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.).

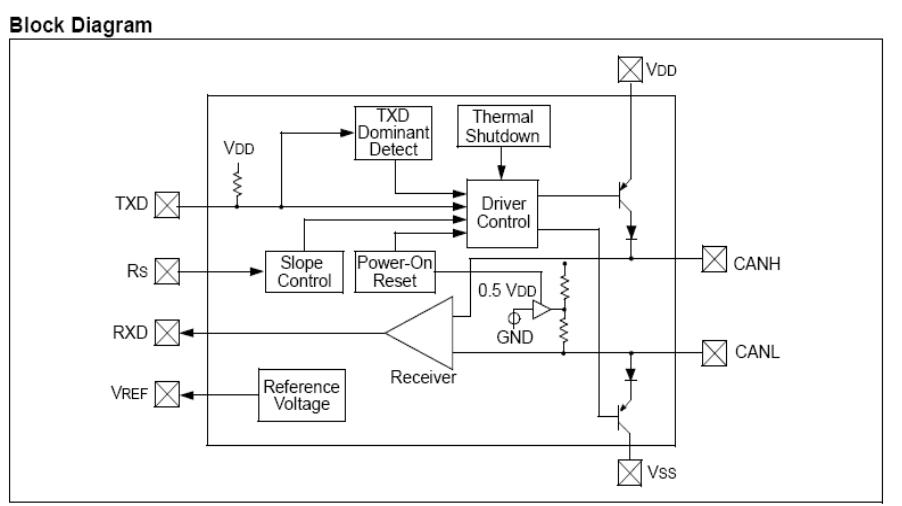
**A.5.3 CAN Transceiver Block Diagram**

Figure A.6 Block Diagram For CAN Transceiver

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.

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.

**A.6 OPERATING MODES**

The RS pin allows three modes of operation to be selected:

* High-Speed.
* Slope-Control.
* Standby.

When in High-Speed or Slope-Control mode, the drivers for the CANH and CANL signals are internally regulated to provide controlled symmetry in order to minimize EMI emissions. Additionally, the slope of the signal transitions on CANH and CANL can be controlled with a resistor connected from pin 8 (RS) to ground. The slope must be proportional to the current output at RS, which will further reduce EMI emissions.

**High-Speed:**High-Speed mode is selected by connecting the RS pin to VSS. In this mode, the transmitter output drivers have fast output rise and fall times to support high speed CAN bus rates.

**Slope-Control:**Slope-Control mode further reduces EMI by limiting the rise and fall times of CANH and CANL. The slope, or slew rate(SR), is controlled by connecting an external resistor (REXT) between RS and VOL (usually ground). The slope is proportional to the current output at the RS pin. Since the current is primarily determined by the slope-control resistance value REXT,a certain slew rate is achieved by applying a specific resistance.

**Standby Mode:**The device may be placed in Standby or SLEEP mode by applying a high-level to the RS pin. In SLEEP mode, the transmitter is switched off and the receiver operates at a lower current. The receive pin on the controller side (RXD) is still functional, but will operate at a slower rate. The attached microcontroller can monitor RXD for CAN bus activity and place the transceiver into normal operation via the RS pin at higher bus rates, the first CAN message may be lost.

**A.6 PIN DETAILS**

**Transmitter Data Input (TXD):**TXD is a TTL-compatible input pin. The data on this pin is driven out on the CANH and CANL differential output pins. It is usually connected to the transmitter data output of the CAN controller device. When TXD is Low, CANH and CANL are in the Dominant state. When TXD is High, CANH and CANL are in the Recessive state, provided that another CAN node is not driving the CAN bus with a Dominant state. TXD has an internal pull-up resistor (nominal 25 kΩ to VDD).

**Ground Supply (VSS), Supply Voltage (VDD):**Ground supply pin, Positive supply voltage pin.

**Receiver Data Output (RXD):**RXD is a CMOS-compatible output that drives High or Low depending on the differential signals on the CANH and CANL pins and is usually connected to the receiver data input of the CAN controller device. RXD is High when the CAN bus is Recessive and Low in the Dominant state.

**Reference Voltage (VREF):**Reference Voltage Output (defined as VDD/2).

**CAN Low (CANL):**The CANL output drives the Low side of the CAN differential bus. This pin is also tied internally to receive the input comparator.

**CAN High (CANH)**:The CANH output drives the high-side of the CAN differential bus. This pin is also tied internally to receive the input comparator.

Table no A.2 Pin Details Of CAN Transceiver

|  |  |  |
| --- | --- | --- |
| **PIN NUMBER** | **PIN NAME** | **PIN FUNCTION** |
| 1 | TXD | Transmitter data input |
| 2 | VSS | Ground |
| 3 | VDD | Supply Voltage |
| 4 | RXD | Receive data output |
| 5 | VREF | Reference output voltage |
| 6 | CANH | CAN High level voltage I/O |
| 7 | CANL | CAN Low level voltage I/O |
| 8 | RS | Slope-control output |

**REFERENCES**

1. Benjamin C Kuo.M (2003), Farid Golnaraghi, Automatic Control systems, 8th edition, John Wiley & sons. Inc.
2. Gmbh.B (1991), “CAN specifications” Volume 1 Version 2.0.
3. ISO (1993). Road Vehicles: Interchange of Digital Information: “Controller Area Network (CAN) for High Speed Communication”.
4. Kumar.M, Verma.A, and Srividya.A (2009), Response-Time “Modeling of Controller Area Network (CAN). Distributed Computing and Networking, Lecture Notes in Computer Science Volume 5408.

1. Li.M (2009), Design of Embedded Remote Temperature Monitoring System based on Advanced RISC Machine.
2. Pazul (1999), “Controller Area Network (CAN) Basics”, Microchip technology Inc.
3. Prodanov.W, Valle.M, and Buzas.R (2009), “A controller area network bus transceiver behavioral model for network design and simulation”.

1. Tindell .K, Burns.A, and Wellings.A.J (2005), “Calculating controller area network (CAN) message response times”.

1. Vijayalakshmi.S (2013), “Vehicle control system implementation Using CAN protocol”. IJAREEIE, Volume 2.
2. Wilfried Voss (2005-2008), “A comprehensive guide to controller area network”, Copperhill Media Corporation, 2005-2008.