Robust Tire Pressure Monitoring Using CAN Protocol

[1]Ruchi Shukla, [2]Mr. Sunil Sharma
[1]M.Tech Scholar, [2]Assistant Professor,
Department of Electronics & Communication Engineering, Pacific University, Udaipur

---

Abstract - The Controller Area Network (CAN) is a serial, asynchronous, multi-master communication protocol for connecting electronic control modules in automotive and industrial applications. CAN was designed for automotive applications needing high levels of data integrity and data rates of up to 1 Mbit/s. In this project Controller Area Network protocol is implemented using P89v51rd2 Philips variant. The application we have taken up is "Robust Tire pressure monitoring using Controller Area Network architecture". The system is constituted of two CAN nodes, each CAN node is formed by a transceiver MCP2515 or MCP2551 and 16-bit microcontroller P89v51rd2.The first stage consist of the conventional sensor of pressure (BMP) that converts the pressure into voltage signal, and then this signal is conditioned and then the signal is transmitted to the input of the A/D converter of the microcontroller. The A/D converted data is transmitted to the CAN node 2 using the CAN architecture. At node 2 the received pressure readings are displayed using an alphanumeric LCD display of size 16X2.In this thesis we have concerned with design techniques for implementation of CAN nodes for data monitoring and taking appropriate decision based on data in the control system. Implementation of CAN for tire pressure monitoring the device is successful. Keywords

Keywords - P89V51rd2 microcontroller, MCP2515 can controller, MCP2551 can controller, SPI interface, keil software, Flash Magic.

INTRODUCTION

An embedded system is a microprocessor based system that is built to control a function or range of functions and is not designed to be programmed by the end user in the same way that PC is. With a PC, this is exactly what a user can do; one minute the PC is word processor and next it's a games machine simply by changing the software. An embedded system is designed to perform on particular task albeit with choices and different options. The last point is important because it differentiates itself from the world of the PC where the end user does reprogram it whenever a different software package is bought and run. However, PCs have provided an easily accessible source of hardware and software for embedded systems and it should be no surprise that they form the basis of many embedded systems.

Inside the embedded system

Processor: The main criterion for the processor is: can it provide the processing power needed to perform the tasks within the system. While processor performance is essential and forms the first gating criterion, there are others such as cost; this should be system cost and not just the cost of the processor in isolation, power consumption, software tools and component availability and so on.

Memory: Memory is an important part of any embedded system design and is heavily influenced by the software design, and in turn may dictate how the software is designed, written and developed. It provides storage for the software that it will run. With RAM being more expensive than ROM and non-volatile, many embedded systems and in particular, microcontrollers, have small amounts of RAM compared to the ROM that is available for the program.

Peripherals: An embedded system has to communicate with the outside world and this is done by peripherals. Input peripherals are usually associated with sensors that measure the external environment and thus effectively control the output operations that the embedded system performs. Our work is confined in this area of embedded systems. Specially, there are different data communication protocols used to transfer the data from one device to another device in the network of embedded systems and devices. In this thesis work we studied three different protocols and developed hardware and software for the protocols on microcontroller (p89rd51) platform.

Software: The software component within an embedded system often encompasses the technology that adds value to the system and defines what it does and how well it does it embedded software consists of different components: initialization and configuration, operating system or run time environment, application software, error handling and debug.

Algorithms: Algorithms are the key constituents of the software that makes an embedded system behave in the way that it does. They can range from mathematical processing through to models of the external environment which are used to interpret information from external sensors and thus generate control signals. With the digital technology in use today such as MP3 and DVD players, the algorithms that digitally encode the analogue data defined by standard bodies.

Networks in Embedded systems

There are several reasons to build network based embedded systems. When the processing tasks are physically distributed, it may be necessary to put some of the computing power near where the events occur. Data reduction is another important reason for distributed processing. It may be possible to perform some initial signal processing on captured data to reduce its volume. Reducing the data on separate processor may significantly reduce the load on the processor that makes use of that data. Modularity is another motivation for network based design. When a large system is assembled out of existing components, those components may use a network port as clean interface that does not interface with the internal operation of the component in ways that using the microprocessors in one part of the network can be used to probe components in another part of network. Distributed embedded system design is another example of hardware/software co- design, since we must design the network topology as well as the software running on the network nodes.

Necessity of Networks in Embedded Systems

Building an embedded system with several processing elements talking over a network is definitely more complicated than using single large microprocessors to perform the same tasks. Distributed systems are necessary because the devices that the processing elements communicate with are physically separated. If the deadlines for processing the data are short, it may be cost –effective to put the processing elements where the data are located rather than build a higher speed network to carry the data to distant, fast processing elements.

CONTROLLER AREA NETWORK

The Controller Area Network (CAN) is a serial communications protocol that efficiently supports distributed real-time control applications with a very high level of data integrity. CAN was originally developed by the German company Robert Bosch for use in the car industry to provide a cost-effective communications bus for in-car electronics and as alternative to expensive and cumbersome wiring looms. The car industry continues to use CAN for an increasing number of applications, but because of its proven reliability and robustness, CAN is now also being used in many other industrial control applications. CAN fulfils the communication needs of a wide range of applications, from high-speed networks to low-cost multiplex wiring. For example, in automotive electronics, engine control units, sensors and anti-skid systems may be connected using CAN, with bit-rates up to 1 Mbit/s. At the same time, it is cost effective to build CAN into vehicle body electronics, such as lamp clusters and electric windows, to replace the wiring harness otherwise required. The intention of the CAN specification is to achieve compatibility between any two CAN implementations. Compatibility, however, has different aspects with respect to, for example, electrical features and the interpretation of data to be transferred.

Layered structure of a CAN node

The LLC sub layer is concerned with message filtering, overload notification and recovery management. The MAC sub layer represents the kernel of the CAN protocol. It presents messages received from the LLC sub layer and accepts messages to be transmitted by the LLC sub layer. The MAC sub layer is responsible for message framing, arbitration, acknowledgement, and error detection and signaling. The MAC sub layer is supervised by a self checking mechanism, called fault confinement, which distinguishes short disturbances from permanent failures. The Physical Layer defines how signals are actually transmitted, dealing with the descriptions of bit timing, bit encoding and synchronization. The Physical Layer is not defined here, as it will vary according to the requirements of individual applications (for example, transmission medium and signal level implementations).

Messages

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

Information routing

In CAN systems a node does not make use of any information about the system configuration (e.g. node addresses). This has several important consequences, which are described below.

Bit-rate

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

Arbitration

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

Error detection

To detect errors the following measures have been taken:

Monitoring (each transmitter compares the bit levels detected on the bus with the bit levels being transmitted)

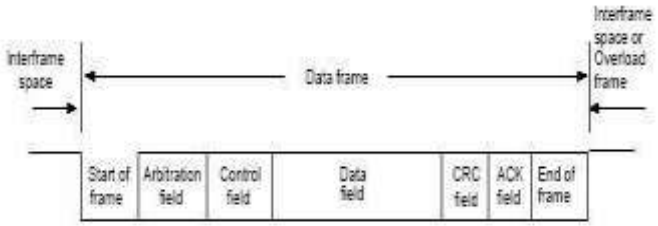
Cyclic Redundancy Check (CRC)

Bit-Stuffing

Message Frame Check

Data frame

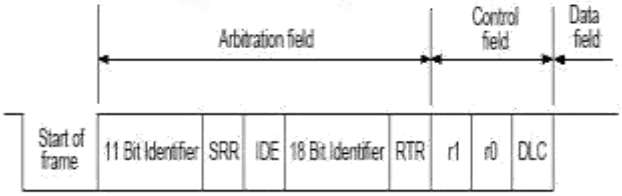
A Data frame is composed of seven different bit fields:



Start of frame, Arbitration field, Control field, Data field, CRC field, ACK field, End of frame. The Data field can be of length zero.

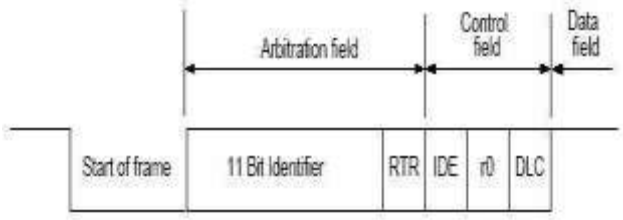
Arbitration field

The format of the Arbitration field is different for Standard Format and Extended Format frames.



Arbitration Field; Standard Format

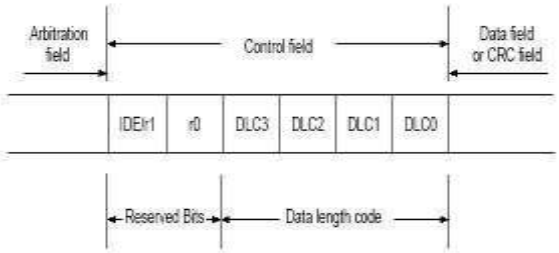
In Standard Format the Arbitration field consists of the 11 bit Identifier and the RTR-Bit. The Identifier bits are denoted ID-28 ... ID-18.In Extended Format the Arbitration field consists of the 29 bit Identifier, the SRR-Bit, the IDE-Bit, and the RTR-Bit. The Identifier bits are denoted ID-28 ... ID-0.



Arbitration Field; Extended Format

Control field

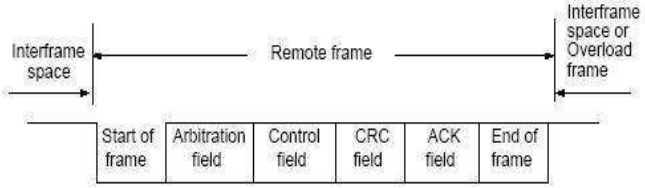
The Control field consists of six bits. The format of the Control field is different for Standard Format and Extended Format. Frames in Standard Format include the Data length code, the IDE bit, which is transmitted dominant, and the reserved bit r0. Frames in Extended Format include the Data length code and two reserved bits, r0 and r1. The reserved bits must be sent dominant, but the receivers accept dominant and recessive bits in all combinations.



The number of bytes in the Data field is indicated by the Data length code. This Data length code is 4 bits wide and is transmitted within the Control field. The DLC bits can code data lengths from 0 to 8 bytes and other values are not permitted.

Remote frame

A node acting as a Receiver for certain data can initiate the transmission of the respective data by its source node by sending a Remote frame. A Remote frame is composed of six different bit fields: Start of frame, Arbitration field, Control field, CRC field, ACK field, End of frame. The RTR bit of a Remote frame is always recessive. There is no Data field in a Remote frame, irrespective of the value of the Data length code which is that of the corresponding Data frame and may be assigned any value within the admissible range 0... 8.

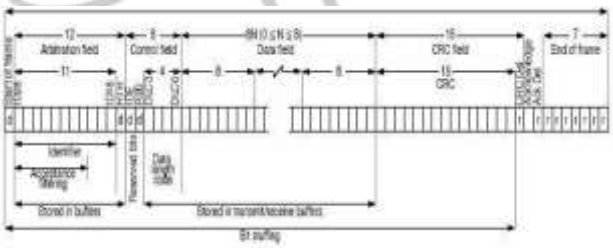


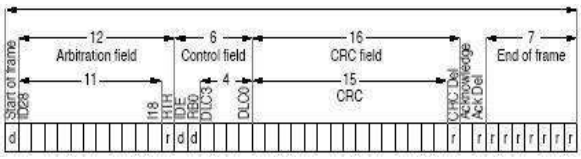
Remote Frame

The polarity of the RTR bit indicates whether a transmitted frame is a Data frame (RTR bit dominant) or a Remote frame (RTR bit recessive

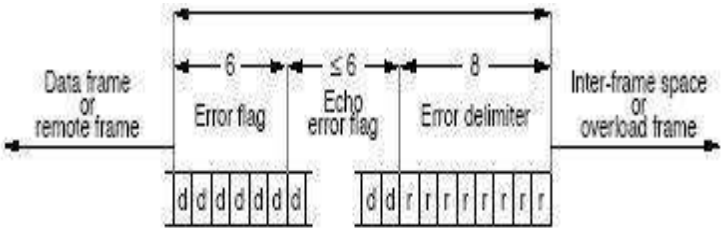
CAN Frame Format-Standard Format

Data Frame



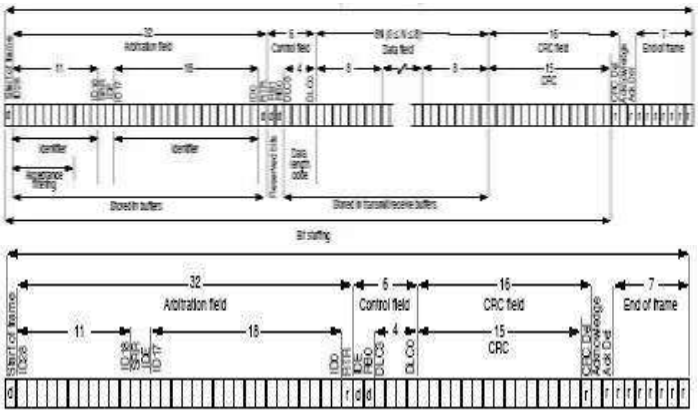


Remote Frame



CAN Frame Format-Extended Format

Data Frame



Remote Frame

Fault Confinement

CAN node status

With respect to fault confinement, a node may be in one of three states:

Error-active

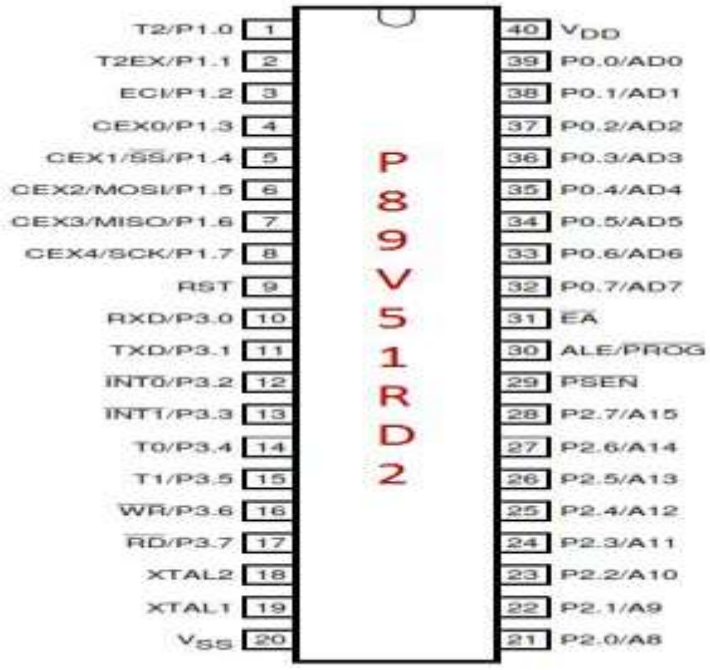
Error-passive

Bus off

An error active node can normally take part in bus communication and sends an Active Error Flag when an error has been detected. An error-passive node must not send an Active Error Flag. It takes part in bus communication, but when an error has been detected only a Passive Error Flag is sent. Also after a transmission, an error-passive node will wait before initiating a further transmission. A bus-off node is not allowed to have any influence on the bus (e.g. output drivers switched off).

Philips P89V51RD2 Microcontroller

8-bit 80C51 5 V low power 16/32/64 kb flash microcontroller with 1 kb RAM The P89V51RB2/RC2/RD2 are 80C51 microcontrollers with 16/32/64 kb flash and 1024 B of A key feature of the P89V51RB2/RC2/RD2 is its X2 mode option. The design engineer can choose to run the application with the conventional 80C51 clock rate (12 clocks per machine cycle) or select the X2 mode (six clocks per machine cycle) to achieve twice the throughput at the same clock frequency. Another way to benefit from this feature is to keep the same The flash program memory supports both parallel programming and in serial ISP. Parallel programming mode offers gang-programming at high speed, reducing programming costs and time to market. ISP allows a device to be reprogrammed in the end product under software control. The capability to field/update the application firmware makes a wide range of applications possible.



PIN DIAGRAM OF P89V51RD2

The P89V51RB2/RC2/RD2 is also capable of IAP, allowing the flash program memory to be reconfigured even while the application is running.

The P89V51RB2/RC2/RD2 are 80C51 microcontrollers with 16/32/64 kB flash and B of data RAM. A key feature of the P89V51RB2/RC2/RD2 is its X2 mode option. The design engineer can choose to run the application with the conventional 80C51 clock rate (12 clocks per machine cycle) or select the X2 mode (six clocks per machine cycle) to achieve twice the throughput at the same clock frequency. Another way to benefit from this feature is to keep the same performance by reducing the clock frequency by half, thus dramatically reducing the EMI. The flash program memory supports both parallel programming and in serial ISP. Parallel programming mode offers gang-programming at high speed, reducing programming costs and time to market. ISP allows a device to be reprogrammed in the end product under software control. The capability to field/update the application firmware makes a wide range of applications possible. The P89V51RB2/RC2/RD2 is also capable of IAP, allowing the flash program memory to be reconfigured even while the application is running.

80C51 CPU 5 V operating voltage from 0 MHz to 40 MHz kB of on-chip flash user code memory with ISP and IAP Supports 12-clock (default) or 6-clock mode selection via software or ISP SPI and enhanced UART PCA with PWM and capture/compare functions Four 8-bit I/O ports with three high-current port 1 pins (16 mA each) Three 16-bit timers/counters Programmable watchdog timer Eight interrupt sources with four priority levels Second DPTR register Low EMI mode (ALE inhibit) TTL- and CMOS-compatible logic levels I Brownout detection I Low power modes N Power-down mode with external interrupt wake-up N Idle mode DIP40, PLCC44 and TQFP44 packages Ordering information Package Name TQFP44 DIP40 Description plastic leaded chip carrier; 44 leads plastic dual in-line package; 40 leads (600 mil) plastic thin quad flat package; 44 leads; body mm plastic leaded chip carrier; 44 leads plastic thin quad flat package; 44 leads; body mm plastic dual in-line package; 40 leads (600 mil) plastic leaded chip carrier; 44 leads plastic thin quad flat package; 44 leads; body mm plastic dual in-line package; 40 leads (600 mil) plastic dual in-line package; 40 leads (600 mil) Version SOT376-1 SOT129-1 Type number

Ordering options Flash memory 64 kB Temperature range +85 °C Frequency 0 MHz to 40 MHz Type number P89V51RD2BN P89V51RD2FN Philips P89V51RD2 as 8051 Microcontroller Unit (MCU). And I have been developing my code with Open source C Compiler SDCC. Please visit my Tools page for software preparation guides. The P89V51RD2 is a 80C51 microcontroller with 64 kB Flash and 1024 bytes of data RAM. A key feature of the P89V51RD2 is its X2 mode option. The design engineer can choose to run the application with the conventional 80C51 clock rate (12 clocks per machine cycle) or select the X2 mode (6 clocks per machine cycle) to achieve twice the throughput at the same clock frequency. Another way to benefit from this feature is to keep the same performance by reducing the clock frequency by half, thus dramatically reducing the EMI. The Flash program memory supports both parallel programming and in serial In-System Programming (ISP). Parallel programming mode offers gang-programming at high speed, reducing programming costs and time to market. ISP allows a device to be reprogrammed in the end product under software control. The capability to field/update the application firmware makes a wide range of applications possible. The P89V51RD2 is also In-Application Programmable (IAP), allowing the Flash program memory to be reconfigured even while the application is running.

Features

80C51 Central Processing Unit **.**

5 V Operating voltages from 0 MHz to 40 MHz

64 B of on-chip Flash user code memory with ISP (In-System Programming) and IAP (In- Application Programming)**.**

Supports 12-clock (default) or 6-clock mode selection via software or ISP.SPI (Serial Peripheral Interface) and enhanced UARTPCA (Programmable Counter Array) with PWM and Capture/Compare functions

Four 8-bit I/O ports with three high-current Port 1 pins (16 mA each)

Three 16-bit timers/counters
Programmable watchdog timer
Eight interrupt sources with four priority levels
Second DPTR register
Low EMI mode (ALE inhibit)
TTL- and CMOS-compatible logic levels
Brown-out detection

Monitoring of pressure using Pressure Sensor Based on CAN Architecture

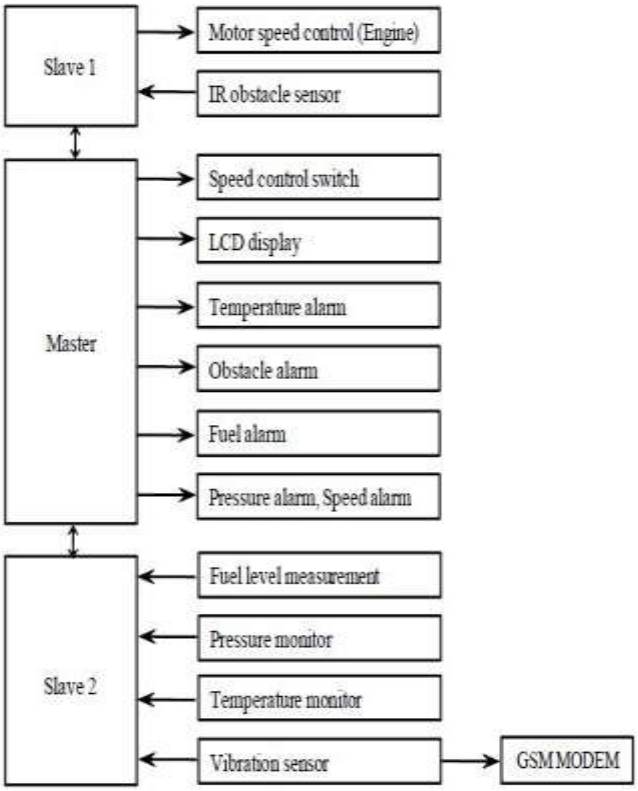


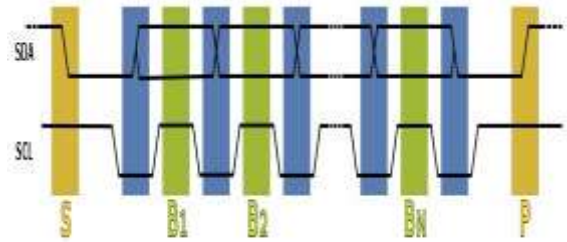
Fig 3. Block diagram of proposed system

Pressure is the most often-measured environmental quantity. This might be expected since most physical, electronic, chemical, mechanical and biological systems are affected by pressure. Some processes work well only within a narrow range of pressure. Certain chemical reactions, biological processes, and even electronic circuits perform best within limited pressure ranges. When these processes need to be optimized, control systems that keep pressure within specified limits are often used. Pressure sensors provide inputs to those control systems. When pressure limits are exceeded, action must be taken to protect the system. In these systems, pressure sensing helps enhance reliability.

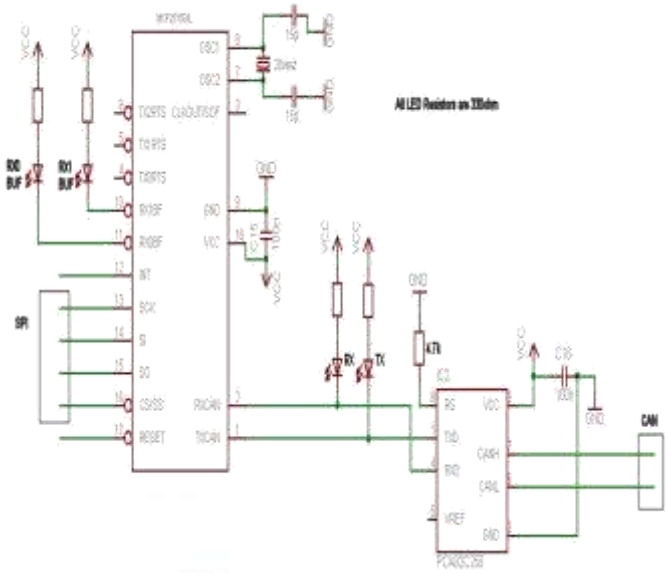
Pressure sensor (BMP SENSOR):

Bosch's BMP085 is a rock-solid barometric pressure sensor. It features a measuring range of anywhere between 30,000 and 110,000 Pa. 'Pa' meaning the Pascal unit, which you'll probably more often see converted to hPa (hector Pascal), equal to 100 Pa, or kPa (kilopascal), which is 1000 Pa. As a bonus the BMP085 also provides a **temperature** measurement, anywhere from 0 to 65 °C.

The BMP085 has a **digital** interface, I^2C to be specific. This means there may is a bit more overhead to get it talking to your microcontroller, but in return you get data that is much less susceptible to noise and other factors that may hamper an analog signal. I^2C is a synchronous two-wire interface, the first wire, SDA, transmits data, while a second wire, SCL, transmits a clock, which is used to keep track of the data. If you're using an Adriano to talk to the BMP085, the library will conveniently take care of most of the work in communicating with the sensor.



As with most Spark Fun products, before you start toying around with something, it's always best to RTFM read the fancy datasheet. The BMP085 datasheet covers everything you'd ever need to know about that little sensor on your breakout board.



Interfacing diagram of controller with can transceiver

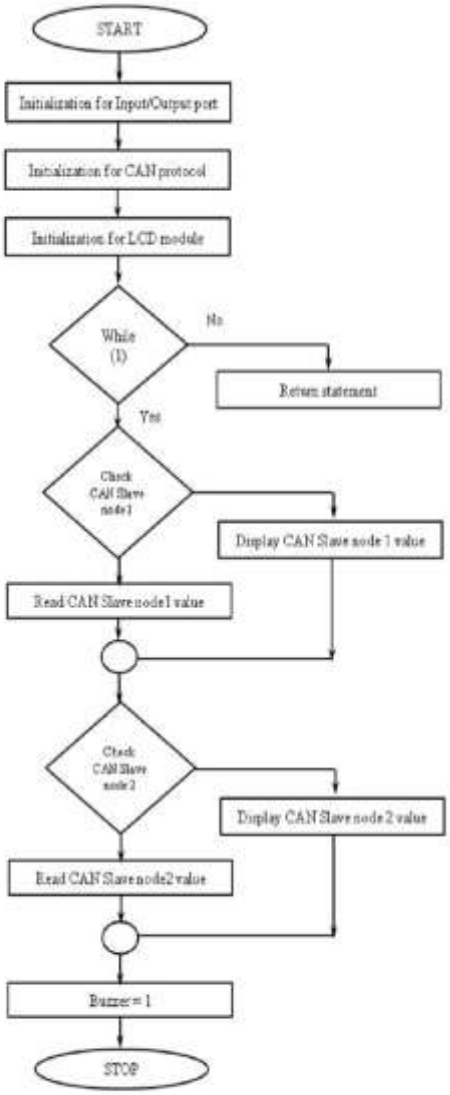


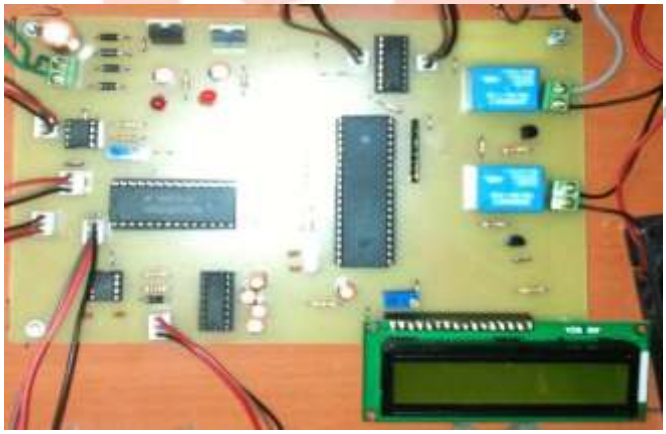
Fig 4. Flowchart for program

Fig 4 shows the flowchart for Embedded C program for vehicle control system using CAN protoc

COMPARISON OF CAN PROTOCOL WITH OTHER PROTOCOL

Reliability: - differential signaling
Priority:-easily prioritize messages
Low wire count
Node independence
Can add / remove nodes
Node breakdown doesn't bring down network.

Parameters	CAN	RS232	SPI	I^2C
Async/Sync	Async	Async	Sync	Sync
Type	Multi-Master	Peer-to- peer	Multi- Master	Multi-Master
Communication (Duplex)	Half	Full	Full	Half
Max Devices	112	2	Based on bus capacitance and bit rate	Based on bus capacitance and bit rate
Max speed	1Mbps	20 Kbps	>1Mbps	3.4 Mbps



set up of can controller

Result on LCD

It consists of one master node and two slave nodes.ARM as the master controller (Engine Control Module) which controls the vehicle status with various sensors. Two PIC ICs are used as slave nodes to receive the inputs of vehicle status. The communication between these sensors is done by using CAN controller. Slave controller receives the signals from vehicles like pressure send to master controller with high speed rate. Master controls the status of vehicle and sends the feedback to operator panel by providing digital information's via LCD display and alarms. Here Operator interface is digital type. By this operator can easily see the signals and able to control the vehicle. Bmp sensor helps in identifying the obstacles presence around the vehicle.

Paper name	Vehicle control system implementation Using CAN protocol	Robust Tire Pressure Monitoring Using CAN Protocol
HARDWARE	CAN bus controller, 32-Bit ARM LPC2129 as the main control module, LCD display to provide Digital interface, GSM for mobile communication and other accessories.	CAN bus controller, CAN node is formed by a transceiver MCP2515 or MCP2551 and 16-bit microcontroller P89v51rd2. LCD display to provide Digital interface.
PRINCIPAL	Reduced Instruction Set Computer (RISC)	Complex Instruction Set Computer (CISC)
SOFTWARE	The vehicle control system is programmed using the Embedded C and debugged with **MPLAB X IDE.**	Embedded C and debugged with **KEIL 5 VISION IDE TOOL and PROTEUS 8**
PURPOSE	Temperature controlling and obstacle detecting	Pressure sensing
POWER CONSUMPTION	<71.33 µA (Power mode)	<40 µA(Power down mode)
TIME DELAY	0.499µsec(5 MHz)	1.85 µsec(11.85 MHz)
NO. OF MODE	4 MODE	2 MODE(FIRST & FOURTH MODE)

CONCLUSION AND FUTURE WORK

This thesis is concerned with design techniques for implementation of CAN nodes for data monitoring and taking appropriate decision based on data in the control system. Implementation of CAN for tire pressure monitoring is successful and the same idea can be applicable to monitor Temperature monitoring system, Adaptive Cruise control, power window and Engine management systems in Automotives. This leads to decentralization of control system in vehicles. This can be extended to industrial control vertical, mainly in decentralizing the PLC (programmable Logic Controllers) control mechanisms. In chapter I we discussed about definition of embedded systems, networks for embedded systems and features of communication protocols for embedded systems. Chapter II deals with Controller Area Network (CAN) specifications and standards, different types of CAN, its properties, its efficiency, its advantages and disadvantages. Finally we will discuss about Philips CAN protocol. Chapter III discusses about architecture, features and tools required forP89v51rd2microcontroller. It also discusses about peripherals its supports in brief background debug mode for debugging, hardware design of circuit board used, Finally, the chapter IV discusses about implementation of tire pressure monitoring using CAN architecture.

Future work

There is potential future work implementing CAN for Industrial control systems like PLC's etc. This thesis is concerned to pressure monitoring, this can be extended to four nodes, eight nodes and 16 nodes CAN network for different automotive applications. For further decentralization of control systems, sub-network for CAN called LIN (Local Interconnect Network) can be implemented. It is intended in future works, to increase the number of CAN nodes on the bus, using CAN IP's based on FPGA, Signal processors and other microcontrollers. We can use it further in hospitals to manage complete operating system, Home automation and for security purpose.

REFRENCE

[1] Kumar, M. A.Verma, and A. Srividya, Response-Time "Modeling of Controller Area Network (CAN). Distributed Computing and Networking, Lecture Notes in Computer Science Volume 5408, p 163-174, 2009.

[2] Tindell, K., A. Burns, and A.J. Wellings, Calculating controller area network (CAN) message response times. Control Engineering Practice, 3(8):p. 1163-1169, 2005.

[3] Li, M., Design of Embedded Remote Temperature Monitoring System based on Advanced RISC Machine. Electrotechnics Electric, 06, p. 273, 2009.
[4] Prodanov, W.M.Valle, and R. Buzas, A controller area network bus transceiver behavioral model for network design and simulation. IEEE Transactions on Industrial Electronics, 56(9): p. 3762-377, 2009.
[5] ISO (1993). Road Vehicles: Interchange of Digital Information: Controller Area Network (CAN) for High Speed Communication. ISO 11898:1993.
[6] B.Gmbh, "CAN specification" vol 1 Version 2.0, 1991.
[7] Pazul, "Controller Area Network (CAN) Basics", Microchip technology Inc., AN713, May 1999.
[8] Wilfried Voss, A comprehensive guide to controller area network, Copperhill Media Corporation, 2005-2008.
[9] Benjamin C Kuo, M. Farid Golnaraghi, Automatic Control systems, Eight edition, John wiley & sons., Inc 2003.
[10] National Semiconductor, "Temperature Sensor Handbook", National Semiconductor Corporation, 2000.
[11] Jonathan W. Valvano. "Embedded Microcomputer Systems". Singapore: Thomson Brooks / Cole, 2002.
[12] tion", t N Company Limited, 2005. Wayne Wolf. "Components as Components". New Delhi: Morgan Kaufmann Publishers, 2001. Steve heath. "Embedded Systems Design". New
[13] "CAN Applications fields", http://www.can-cia.org/applications/, Dec.2005 "Controller Area Network – CAN Informa http://hem.bredband.net/stafni/developer /frames.htm , 2007.
[14] Industrial Controller Area Network (CAN) Applications. http://www.freescale.com/webapp/sps/site/application.jsp?nodeId=02430ZNtdx4J11