INTELLIGENT ROUTE DIVERTER

By

K.K.PREMNATH Reg No.:0710107069
VIJAY JAGANNATH Reg No.:0710107112
J.BALACHANDRASEKAR Reg No.: 0710107301

of

KUMARAGURU COLLEGE OF TECHNOLOGY
(An Autonomous Institution affiliated to Anna University of Technology, Coimbatore)
COIMBATORE - 641049

A PROJECT REPORT

Submitted to the

FACULTY OF ELECTRONICS AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements
for the award of the degree
of
BACHELOR OF ENGINEERING
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

APRIL 2011

BONAFIDE CERTIFICATE

Certified that this project report entitled "**INTELLIGENT ROUTE DIVERTER**" is the bonafide work of **Mr.K.K.PREMNATH, Mr.VIJAY JAGANNATH, Mr.J.BALACHANDRASEKAR** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Project Guide
S.N.S 15/4
Ms.S.N.Shivappriya

Head of the Department

Dr. Rajeswari Mariappan

The candidates with university Register no. 0710107069, 0710107112, 0710107301 are examined by us in the project viva-voce examination held on 19/4/2011

J. Jaikumar 19/4/11
Internal Examiner

External Examiner

ACKNOWLEDGEMENT

A project of this nature needs co-operation and support from many for successful completion. In this regards, we would like to express my thanks and appreciation to the many people who have contributed to the successful completion of this project.

We express my profound gratitude to our beloved Director **Dr.J. Shanmugam,** Kumaraguru College of Technology for his kind support and necessary facilities to carry out the work.

We would like to thank **Dr.S.Ramachandran.,** Principal for providing us an opportunity to carry out this project work.

We express my gratitude to **Dr.Rajeswari Mariappan,** Head of the Department, Electronics and Communication Engineering, who gave her continual support for me throughout the course of this project.

Our heartfelt thanks to **Ms.K.Kavitha M.E.,** Assistant Professor and Project Coordinator, for her contribution and innovative ideas at various stages of the project and for her help to successful completion of this project work.

I express my gratitude to **Mr. S.N.Shivappriya M.E.,** Assistant Professor and my internal project guide, for his valuable guidance, innovative ideas and constant encouragement throughout this project.

I express my sincere gratitude to my family members, friends and to all my staff members of Electronics and Communication Engineering department for their support throughout the course of my project.

Last but not the least I would like to thank the God almighty without whose grace nothing would have been possible so far.

ABSTRACT

With increasing population and other advances in various technologies, the number of vehicles used is alarmingly increasing. A serious outcome of this issue is insufficient road space to accommodate these vehicles. People get annoyed due to traffic as their valuable time is getting wasted. Closed level crossings further add to their frustration.

What would happen if they are given a helping hand? If they are given prior information about the closure of level crossings and the arrival of trains, they won't hesitate to divert their route and prefer a better route. The above choice will be of immense help to the emergency service vehicles like ambulances and fire engines. This idea is made a reality by our project.

This project is about building a setup that predicts the arrival of trains to the level crossings and communicates this to the public. The hardware consists of a Microcontroller, RFID receiver with two tags attached to the train and a GSM module for intimating the user with the appropriate information. The information is transmitted through the GSM to a pre-stored number. The receiver (mobile phone) will receive a SMS from the setup about the status and position of the train.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION:

In the present era, people have no time to wait. To cope up with their speed, all walks of life have been computerized. This project makes an attempt to add intelligence to the route diverter system in level crossing system thereby saving precious time and life of people. Our project is intended to help the vehicles and people in emergency.

This project has taken into consideration the various complexities to be encountered in finding a solution for the inconvenience caused by closed level crossings and appropriate components have been zeroed in on. The project has taken keen interest to keep it very economical and make it accessible to all the classes of people in the society.

Radio-Frequency Identification (RFID) is a technology which is gaining immense importance nowadays in such a way that it has become an inseparable part in many day-to-day applications. Global Systems for Mobile Communications (GSM) is the most popular mobile telephone standard (around 80% of the mobile users use this standard). The RFID technology is used to find the speed of the train under consideration. The internal calculations give the time in which the train reaches the level crossing. This time is communicated to the mobile user.

2. BLOCK DIAGRAM

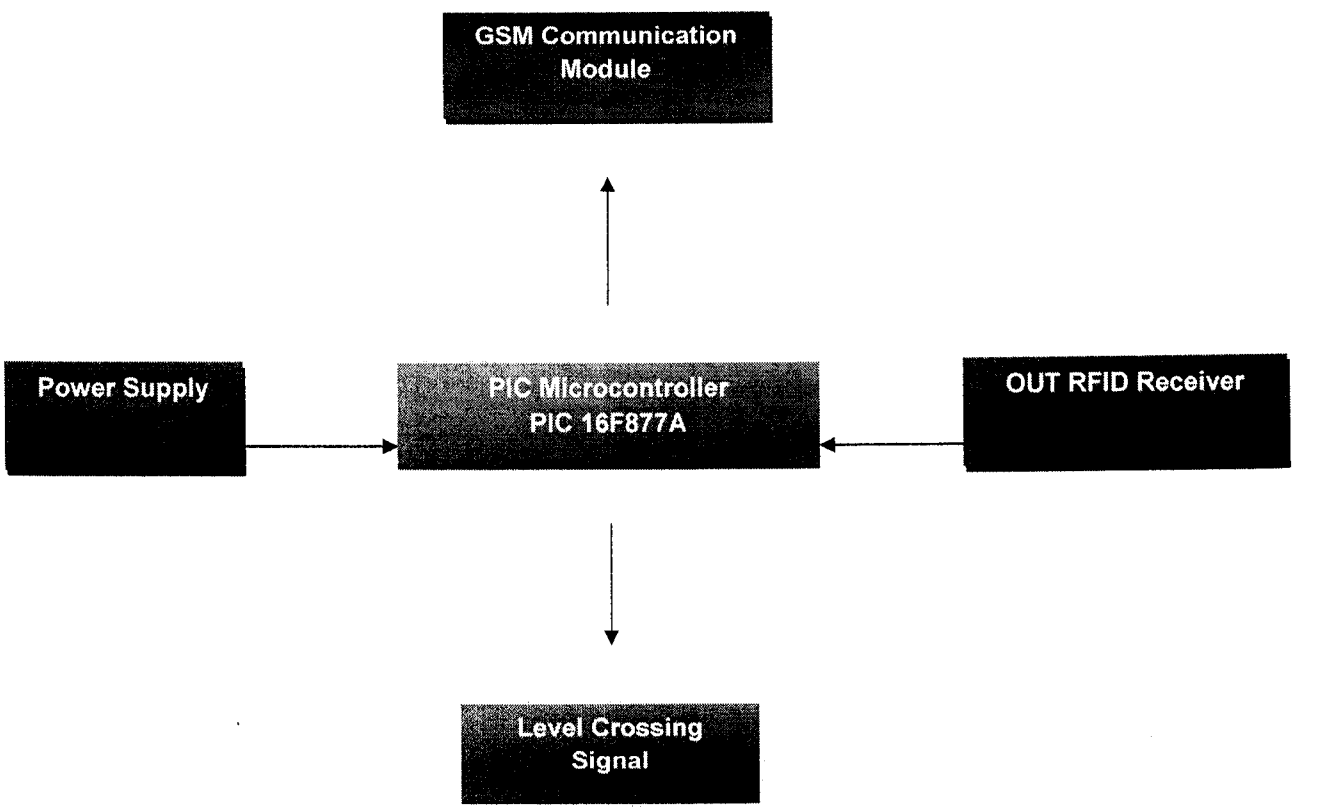


FIG1: Block diagram

3. INTRODUCTION TO MICROCONTROLLERS

Circumstances that we find ourselves in today in the field of microcontrollers had their beginnings in the development of technology of integrated circuits. This development has made it possible to store hundreds of thousands of transistors into one chip. That was a prerequisite for production of microprocessors, and the first computers were made by adding external peripherals such as memory, input-output lines, timers and other. Further increasing of the volume of the package resulted in creation of integrated circuits. These integrated circuits contained both processor and peripherals. That is how the first chip containing a microcomputer, or what would later be known as a microcontroller came about.

3.1: Memory unit

Memory is part of the microcontroller whose function is to store data. For a certain input the contents of a certain addressed memory location is got. Two new concepts are brought to us: addressing and memory location. Memory consists of all memory locations, and addressing is nothing but selecting one of them.

Besides reading from a memory location, memory must also provide for writing onto it. This is done by supplying an additional line called control line. We will designate this line as R/W (read/write). Control line is used in the following way: if r/w=1, reading is done, and if opposite is true then writing is done on the memory location.

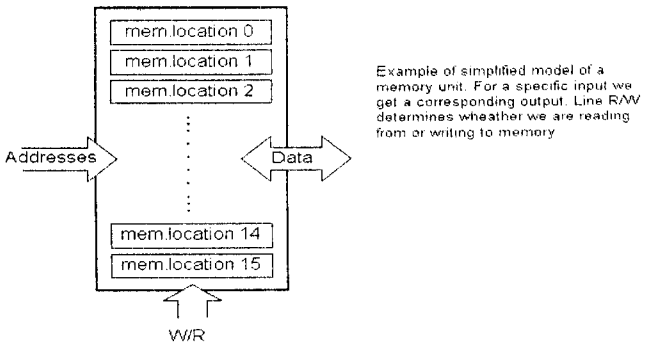


FIG 2: Block diagram of memory unit

3.2: CENTRAL PROCESSING UNIT

Let add 3 more memory locations to a specific block that will have a built in capability to multiply, divide, subtract, and move its contents from one memory location onto another. The part added in is called "central processing unit" (CPU). Its memory locations are called registers.

Registers are therefore memory locations whose role is to help with performing various mathematical operations or any other operations with data wherever data can be found. Look at the current situation. There are two independent entities (memory and CPU) which are interconnected, and thus any exchange of data is hindered, as well as its functionality.

3.3: BUS

That "way" is called "bus". Physically, it represents a group of 8, 16, or more wires. There are two types of buses: address and data bus. The first one consists of as many lines as the amount of memory we wish to address and the other one is as wide as data, in our case 8 bits or the connection line. First one serves to transmit

address from CPU memory, and the second to connect all blocks inside the microcontroller.

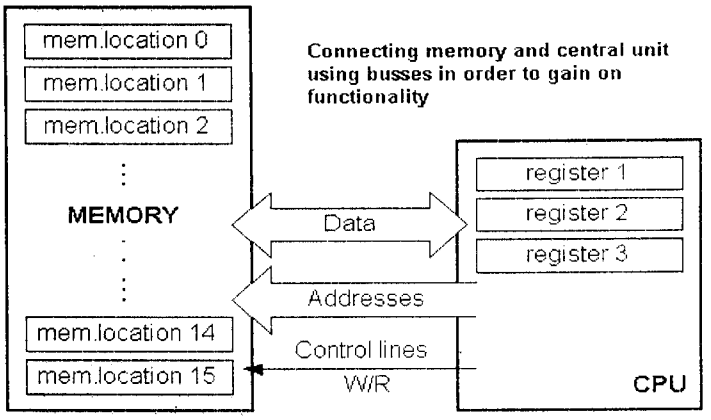


FIG 3: Representation of Bus

3.4: INPUT-OUTPUT UNIT

Those locations to be added are called "ports". There are several types of ports: input, output or bidirectional ports. When working with ports, first of all it is necessary to choose which port is needed to work with, and then to send data to, or take it from the port. When working with it the port acts like a memory location. Something is simply being written into or read from it, and it could be noticed on the pins of the microcontroller.

3.5: SERIAL COMMUNICATION

As there are separate lines for receiving and sending, it is possible to receive and send data (info.) at the same time. So called full-duplex mode block which enables this way of communication is called a serial communication block. Unlike the parallel transmission, data moves here bit by bit, or in a series of bits what defines the term serial communication comes from.

After the reception of data, it has to be read from the receiving location and store in memory as opposed to sending where the process is reversed. In order for this to work, we need to set the rules of exchange of data. These rules are called protocol. Data goes from memory through the bus to the sending location, and then to the receiving unit according to the protocol.

3.6: TIMER UNIT

The timer block this can give us information about time, duration, protocol etc. The basic unit of the timer is a free-run counter which is in fact a register whose numeric value increments by one in even intervals, so that by taking its value during periods T1 and T2 and on the basis of their difference we can determine how much time has elapsed. This is a very important part of the microcontroller whose understanding requires most of our time.

3.7: WATCHDOG:

One more thing is requiring our attention is a flawless functioning of the microcontroller during its run-time.

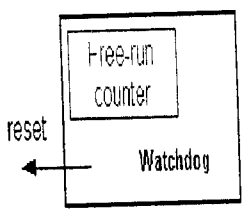


FIG 4: Watch dog timer

Suppose that as a result of some interference (which often does occur in industry) our microcontroller stops executing the program, or worse, it starts working incorrectly. Of course, when this happens with a computer, we simply reset it and it will keep working. However, there is no reset button on the

microcontroller and in order to solve our problem. To overcome this obstacle, one more block called watchdog is introduced.

This block is in fact another free-run counter where our program needs to write a zero in every time it executes correctly. In case that program gets "stuck", zero will not be written in, and counter alone will reset the microcontroller upon achieving its maximum value. This will result in executing the program again, and correctly this time around.

3.8: ANALOG TO DIGITAL CONVERTER:

As the peripheral signals usually are substantially different from the ones that microcontroller can understand (zero and one), they have to be converted into a pattern which can be comprehended by a microcontroller.

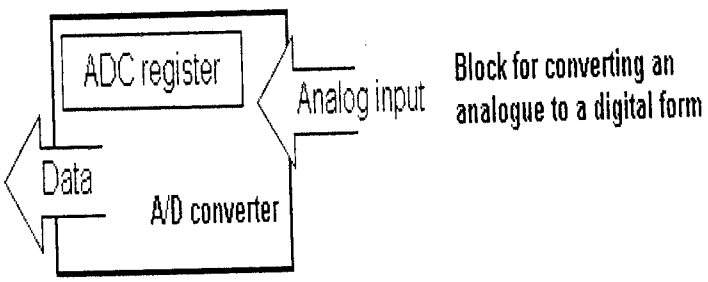


FIG 5: Analog to Digital converter

This task is performed by a block for analog to digital conversion or by an ADC. This block is responsible for converting an information about some analog value to a binary number and for follow it through to a CPU block so that CPU block can further process it.

3.9: MICROCONTROLLER WITH ITS BASIC ELEMENTS AND INTERNAL CONNECTIONS

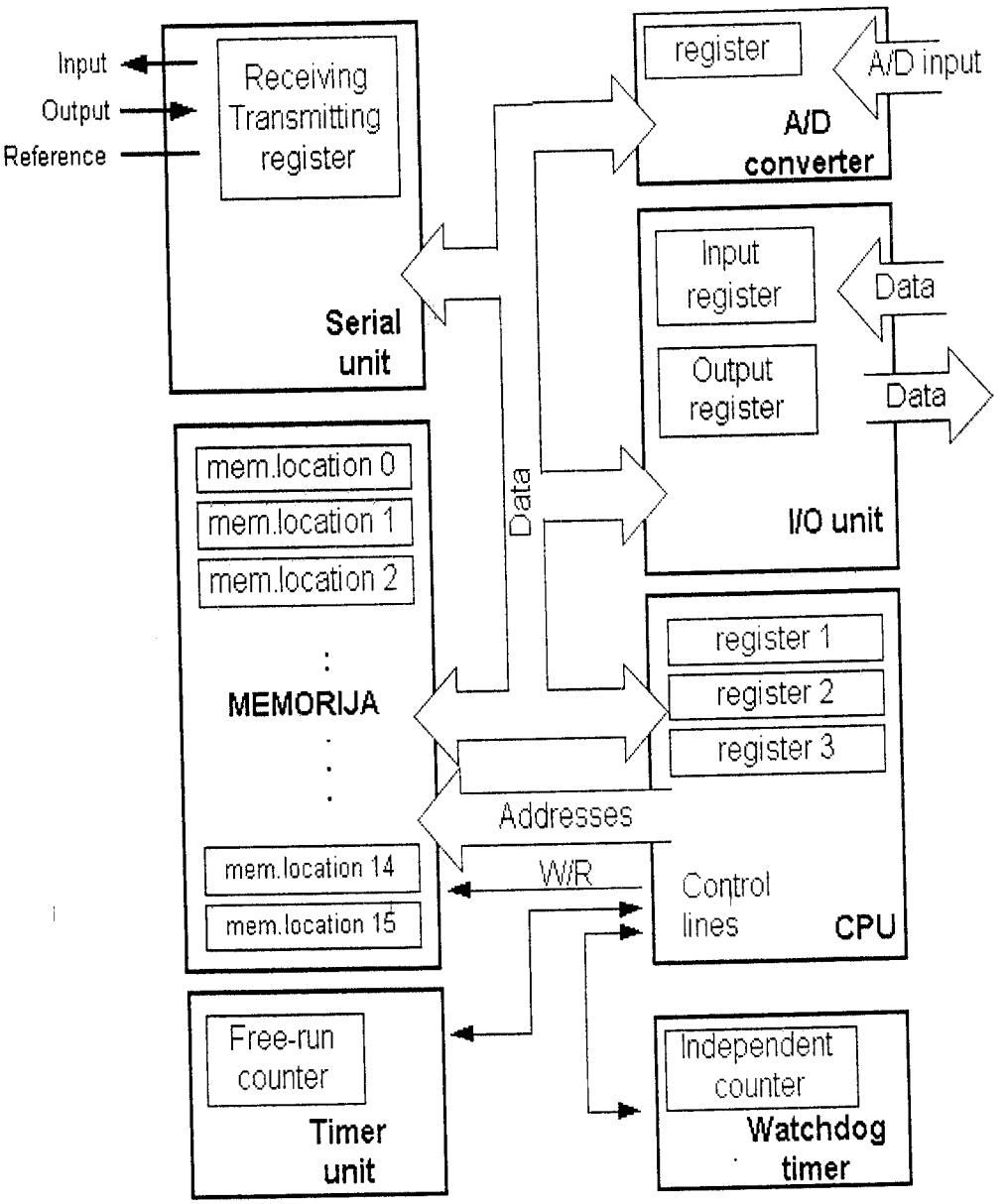


FIG 6: Block representation of the microcontroller

3.10: CISC and RISC:

Harvard architecture is a newer concept than Von-Neumann's. It rose out of the need to speed up the work of a microcontroller. In Harvard architecture, data bus and address bus are separate. Thus a greater flow of data is possible through the central processing unit, and of course, a greater speed of work. Separating a program from data memory makes it further possible for instructions not to have to be 8-bit words. It is also typical for Harvard architecture to have fewer instructions than von-Neumann's, and to have instructions usually executed in one cycle.

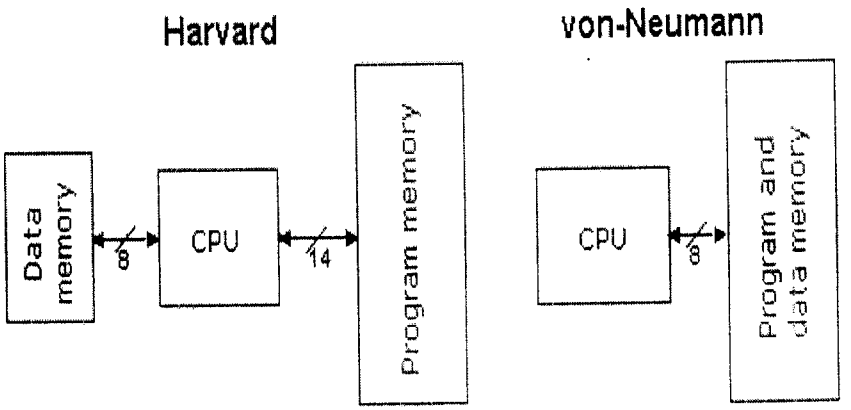


FIG7: CISC and RISC

Microcontrollers with Harvard architecture are also called "RISC microcontrollers". RISC stands for Reduced Instruction Set Computer. Microcontrollers with von-Neumann's architecture are called 'CISC microcontrollers'. Title CISC stands for Complex Instruction Set Computer. Since PIC16F877 is a RISC microcontroller, that means that it has a reduced set of instructions, more precisely 35 instructions. All of these instructions are executed in one cycle except for jump and branch instructions. PIC16F87 usually reaches

results of 2:1 in code compression and 4:1 in speed in relation to other 8-bit microcontrollers in its class.

3.11: PIC MICROCONTROLLER (PIC16F87X) MICROCONTROLLER CORE FEATURES:

- High-performance RISC CPU
- Only 35 single word instructions
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
- Up to 368 x 8 bytes of Data Memory (RAM)
- Up to 256 x 8 bytes of EEPROM data memory
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation

- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges
- Low-power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 20 mA typical @ 3V, 32 kHz

3.12: PERIPHERAL FEATURES:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during sleep via
  - ✓ External crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - ✓ Capture is 16-bit, max. Resolution is 12.5 ns
  - ✓ Compare is 16-bit, max. Resolution is 200 ns
  - ✓ PWM max. Resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI (Master Mode) and I2C (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)

3.13: MEMORY ORGANIZATION:

There are three memory blocks in each of these PICmicro MCUs. The Program Memory and Data Memory have separate buses so that concurrent access can occur.

3.13.1: PROGRAM MEMORY ORGANIZATION:

The PIC16F87X devices have a 13-bit program counter capable of addressing an 8K x 14 program memory space. The PIC16F877/876 devices have 8K x 14 words of FLASH program memory and the PIC16F873/874 devices have 4K x 14. Accessing a location above the physically implemented address will cause a wraparound. The reset vector is at 0000h and the interrupt vector is at 0004h.

3.13.2: DATA MEMORY ORGANIZATION:

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 (STATUS<6>) and RP0 (STATUS<5>) are the bank select bits.

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM.

3.14: I/O PORTS:

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

3.14.1PORTA AND THE TRISA REGISTER:

PORTA is a 6-bit wide bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (=1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a hi-impedance mode). Clearing a TRISA bit (=0) will make the corresponding PORTA pin an output (i.e.,

put the contents of the output latch on the selected pin). Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other PORTA pins have TTL input levels and full CMOS output drivers. Other PORTA pins are multiplexed with analog inputs and analog VREF input. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

3.14.2: PORTB and the TRISB Register:

PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (=1) will make the corresponding PORTB pin an input. Clearing a TRISB bit (=0) will make the corresponding PORTB pin an output. Three pins of PORTB are multiplexed with the Low Voltage Programming function; RB3/PGM, RB6/PGC and RB7/PGD. Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit RBPU (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset. Four of PORTB's pins, RB7:RB4, have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur. The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

3.14.3: PORTC and the TRISC Register:

PORTC is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (=1) will make the corresponding PORTC pin an input. Clearing a TRISC bit (=0) will make the corresponding PORTC pin an output. When the I2C module is enabled, the PORTC (3:4) pins can be configured with normal I2C levels or with SMBUS levels by using the CKE bit (SSPSTAT <6>).

3.14.4: PORTD and TRISD Registers:

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. PORTD can be configured as an 8-bit wide microprocessor port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

3.14.5: PORTE and TRISE Register:

PORTE has three pins, RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7, which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. I/O PORTE becomes control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the input buffers are TTL. PORTE pins are multiplexed with analog inputs. When selected as an analog input, these pins will read as '0's. TRISE controls the direction of the RE pins, even when they are being used as analog inputs.

3.15: TIMER0 MODULE:

The Timer0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Timer mode is selected by clearing bit T0CS (OPTION_REG<5>). In timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles.

The user can work around this by writing an adjusted value to the TMR0 register. Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In counter mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI.

3.16: Master Synchronous Serial Port (MSSP) Module:

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI).

- Inter-Integrated Circuit (I2C).

3.16.1: 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)
- Serial Data in (SDI)
- Serial Clock (SCK)

Additionally, a fourth pin may be used when in a slave mode of operation: Slave Select (SS) to enable the serial port, MSSP Enable bit, SSPEN (SSPCON<5>) must be set. To reset or reconfigure SPI mode, clear bit SSPEN, re-initialize the SSPCON registers, and then set bit SSPEN.

Figure **shows** the block diagram of the MSSP module when in SPI mode. This configures the SDI, SDO, SCK and SS pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed. That is:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> cleared
- SCK (Master mode) must have TRISC<3> cleared
- SCK (Slave mode) must have TRISC<3> set
- SS must have TRISA<5> se

3.16.2: MSSP I2C Operation:

The MSSP module in I2C mode fully implements all master and slave functions (including general call support) and provides interrupts-on-start and stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing

Two pins are used for data transfer. These are the SCL pin, which is the clock, and the SDA pin, which is the data. The SDA and SCL pins are automatically configured when the I2C mode is enabled. The SSP module functions are enabled by setting SSP Enable bit SSPEN (SSPCON<5>). The SSPCON register allows control of the I2C operation. Four mode selection bits (SSPCON<3:0>) allow one of the following I2C modes to be selected:

- I2C Slave mode (7-bit address)
- I2C Slave mode (10-bit address)
- I2C Master mode, clock = OSC/4 (SSPADD +1)

The SSPSTAT register gives the status of the data transfer. SSPBUF is the register to which the transfer data is written to or read from. In receive operations; the SSPBUF and SSPSR create a doubled buffered receiver. This allows reception of the next byte to begin before reading the last byte of received data. When the complete byte is received, it is transferred to the SSPBUF register and flag bit SSPIF is set.

3.17: PIN DIAGRAM:

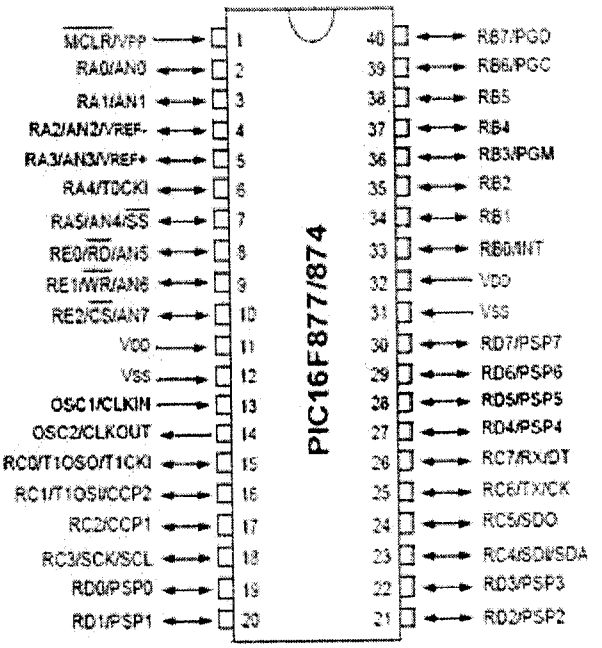


FIG 8: Pin diagram

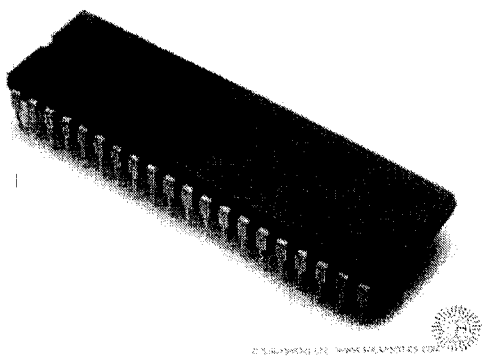


FIG 9: MC-DIP

4. GSM COMMUNICATION

4.1 INTRODUCTION

Global system for mobile communication (GSM) is a globally accepted standard for digital cellular communication. GSM is the name of a standardization group established in 1982 to create a common European mobile telephone standard that would formulate specifications for a pan-European mobile cellular radio system operating at 900 MHz, It is estimated that many countries outside of Europe will join the GSM partnership.

Cellular is one of the fastest growing and most demanding telecommunications applications. Throughout the evolution of cellular telecommunications, various systems have been developed without the benefit of standardized specifications. This presented many problems directly related to compatibility, especially with the development of digital radio technology. The GSM standard is intended to address these problems.

From 1982 to 1985 discussions were held to decide between building an analog or digital system. After multiple field tests, a digital system was adopted for GSM. The next task was to decide between a narrow or broadband solution. In May 1987, the narrowband time division multiple access (TDMA) solution was chosen.

GSM provides recommendations, not requirements. The GSM specifications define the functions and interface requirements in detail but do not address the hardware. The reason for this is to limit the designers as little as possible but still to make it possible for the operators to buy equipment from different suppliers. The GSM network is divided into three major systems: the switching system (SS), the base station system (BSS), and the operation and support system (OSS).

4.2 GSM ARCHITECTURE:

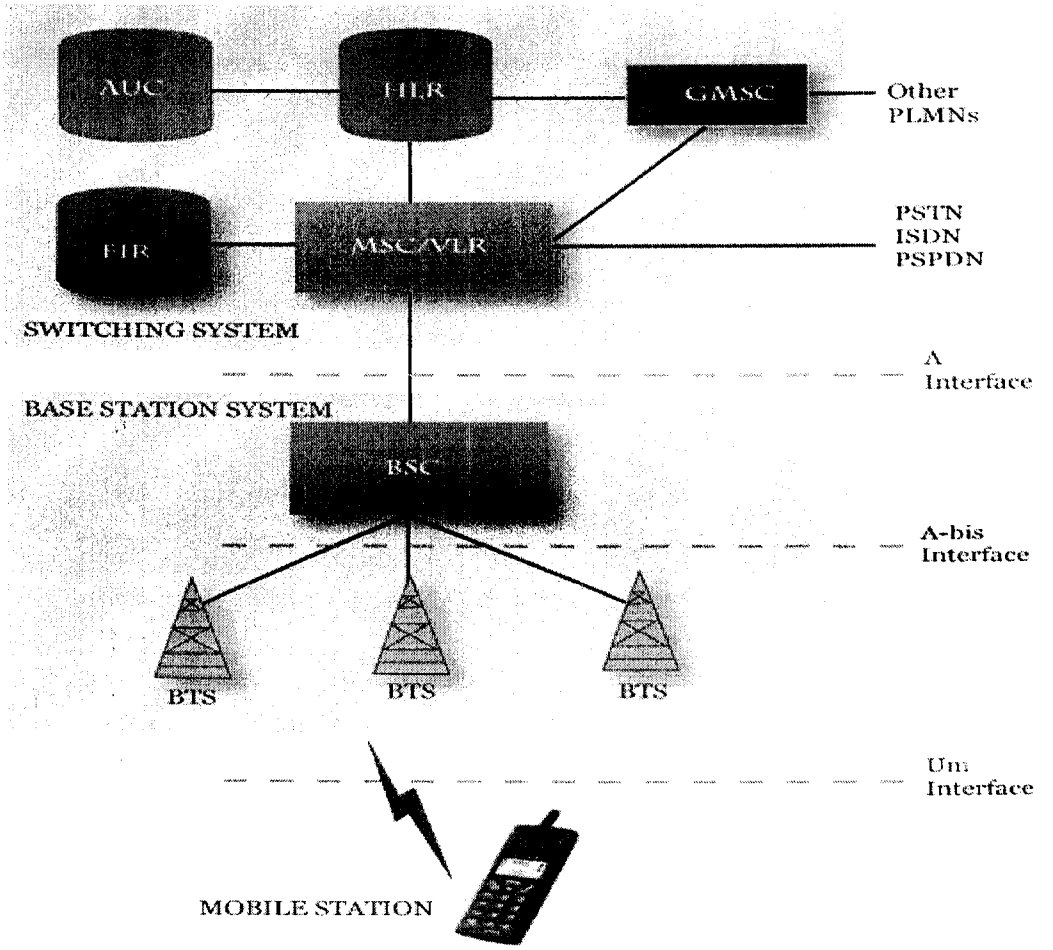


FIG 10: GSM Architecture

The Switching System:

The switching system (SS) is responsible for performing call processing and subscriber-related functions. The switching system includes the following functional units.

Home Location Register (HLR) —The HLR is a database used for storage and management of subscriptions. The HLR is considered the most important database, as it stores permanent data about subscribers, including a subscriber's service

profile, location information, and activity status. When an individual buys a subscription from one of the PCS operators, he or she is registered in the HLR of that operator.

Mobile Services Switching Center (MSC) —The MSC performs the telephony switching functions of the system. It controls calls to and from other telephone and data systems. It also performs such functions as toll ticketing, network interfacing, common channel signaling, and others.

Visitor Location Register (VLR) —The VLR is a database that contains temporary information about subscribers that is needed by the MSC in order to service visiting subscribers. The VLR is always integrated with the MSC. When a mobile station roams into a new MSC area, the VLR connected to that MSC will request data about the mobile station from the HLR. Later, if the mobile station makes a call, the VLR will have the information needed for call setup without having to interrogate the HLR each time.

Authentication Center (AUC) —A unit called the AUC provides authentication and encryption parameters that verify the user's identity and ensure the confidentiality of each call. The AUC protects network operators from different types of fraud found in today's cellular world.

Equipment Identity Register (EIR) —The EIR is a database that contains information about the identity of mobile equipment that prevents calls from stolen, unauthorized, or defective mobile stations. The AUC and EIR are implemented as stand-alone nodes or as a combined AUC/EIR node.

The Base Station System (BSS):

All radio-related functions are performed in the BSS, which consists of base station controllers (BSCs) and the base transceiver stations (BTSs).

BSC —The BSC provides all the control functions and physical links between the MSC and BTS. It is a high-capacity switch that provides functions such as handover, cell configuration data, and control of radio frequency (RF) power levels in base transceiver stations. A number of BSCs are served by an MSC.

BTS —The BTS handles the radio interface to the mobile station. The BTS is the radio equipment (transceivers and antennas) needed to service each cell in the network. A group of BTSs are controlled by a BSC.

The Operation and Support System:

The operations and maintenance center (OMC) is connected to all equipment in the switching system and to the BSC. The implementation of OMC is called the operation and support system (OSS). The OSS is the functional entity from which the network operator monitors and controls the system. The purpose of OSS is to offer the customer cost-effective support for centralized, regional, and local operational and maintenance activities that are required for a GSM network. An important function of OSS is to provide a network overview and support the maintenance activities of different operation and maintenance organizations.

4.3 ADDITIONAL FUNCTIONAL ELEMENTS:

Message Center (MXE) —The MXE is a node that provides integrated voice, fax, and data messaging. Specifically, the MXE handles short message service, cell broadcast, voice mail, fax mail, e-mail, and notification.

Mobile Service Node (MSN) —The MSN is the node that handles the mobile intelligent network (IN) services.

Gateway Mobile Services Switching Center (GMSC) —A gateway is a node used to interconnect two networks. The gateway is often implemented in an MSC. The MSC is then referred to as the GMSC.

GSM Interworking Unit (GIWU) —The GIWU consists of both hardware and software that provides an interface to various networks for data communications. Through the GIWU, users can alternate between speech and data during the same call. The GIWU hardware equipment is physically located at the MSC/VLR

4.4 GSM CELLULAR NETWORK:

GSM is a cellular network, which means that mobile phones connect to it by searching for cells in the immediate vicinity. GSM networks operate in four different frequency ranges. Most GSM networks operate in the 900 MHz or 1800 MHz bands. Some countries in the Americas (including Canada and the United States) use the 850 MHz and 1900 MHz bands because the 900 and 1800 MHz frequency bands were already allocated. The rarer 400 and 450 MHz frequency bands are assigned in some countries where these frequencies were previously used for first-generation systems.

GSM-900 uses 890–915 MHz to send information from the mobile station to the base station (uplink) and 935–960 MHz for the other direction (downlink), providing 124 RF channels (channel numbers 1 to 124) spaced at 200 kHz. Duplex spacing of 45 MHz is used. In some countries the GSM-900 band has been extended to cover a larger frequency range. This 'extended GSM', E-GSM, uses 880–915 MHz (uplink) and 925–960 MHz (downlink), adding 50 channels (channel numbers 975 to 1023 and 0) to the original GSM-900 band. Time division

multiplexing is used to allow eight full-rate or sixteen half-rate speech channels per radio frequency channel. There are eight radio timeslots (giving eight burst periods) grouped into what is called a TDMA frame. Half rate channels use alternate frames in the same timeslot. The channel data rate for all 8 channels is 270.833 kbit/s, and the frame duration is 4.615 ms.

GSM has used a variety of voice codec's to squeeze 3.1 kHz audio into between 5.6 and 13 kbit/s. Originally, two codecs, named after the types of data channel they were allocated, were used, called Half Rate (5.6 kbit/s) and Full Rate (13 kbit/s). These used a system based upon linear predictive coding (LPC). In addition to being efficient with bitrates, these codecs also made it easier to identify more important parts of the audio, allowing the air interface layer to prioritize and better protect these parts of the signal

4.5 GSM NETWORK CLASSIFICATION:

There are five different cell sizes in a GSM network—macro, micro, Pico, femto and umbrella cells.

The coverage area of each cell varies according to the implementation environment. Macro cells can be regarded as cells where the base station antenna is installed on a mast or a building above average roof top level. Micro cells are cells whose antenna height is under average roof top level; they are typically used in urban areas. Pico cells are small cells whose coverage diameter is a few dozen meters; they are mainly used indoors. Femto cells are cells designed for use in residential or small business environments and connect to the service provider's network via a broadband internet connection. Umbrella cells are used to cover shadowed regions of smaller cells and fill in gaps in coverage between those cells.

Cell horizontal radius varies depending on antenna height, antenna gain and propagation conditions from a couple of hundred meters to several tens of kilometers. The longest distance the GSM specification supports in practical use is 35 kilometers (22 mi). There are also several implementations of the concept of an extended cell, where the cell radius could be double or even more, depending on the antenna system, the type of terrain and the timing advance.

Indoor coverage is also supported by GSM and may be achieved by using an indoor pico cell base station, or an indoor repeater with distributed indoor antennas fed through power splitters, to deliver the radio signals from an antenna outdoors to the separate indoor distributed antenna system. These are typically deployed when a lot of call capacity is needed indoors, for example in shopping centers or airports. However, this is not a prerequisite, since indoor coverage is also provided by in-building penetration of the radio signals from nearby cells.

The modulation used in GSM is Gaussian minimum-shift keying (GMSK), a kind of continuous-phase frequency shift keying. In GMSK, the signal to be modulated onto the carrier is first smoothed with a Gaussian low-pass filter prior to being fed to a frequency modulator, which greatly reduces the interference to neighboring

GSM TRANSMITTER:

One of the key features of GSM is the Subscriber Identity Module (SIM), commonly known as a **SIM card**. The SIM is a detachable smart card containing the user's subscription information and phone book. This allows the user to retain his or her information after switching handsets. Alternatively, the user can also change operators while retaining the handset simply by changing the SIM. Some operators will block this by allowing the phone to use only a single SIM, or only a

SIM issued by them; this practice is known as SIM locking, and is illegal in some countries.

GSM SECURITY:

GSM was designed with a moderate level of security. The system was designed to authenticate the subscriber using a pre-shared key and challenge-response. Communications between the subscriber and the base station can be encrypted. The development of UMTS introduces an optional USIM, that uses a longer authentication key to give greater security, as well as mutually authenticating the network and the user - whereas GSM only authenticates the user to the network (and not vice versa). The security model therefore offers confidentiality and authentication, but limited authorization capabilities, and no non-repudiation. GSM uses several cryptographic algorithms for security. The A5/1 and A5/2 stream ciphers are used for ensuring over-the-air voice privacy. A5/1 was developed first and is a stronger algorithm used within Europe and the United States; A5/2 is weaker and used in other countries. Serious weaknesses have been found in both algorithms: it is possible to break A5/2 in real-time with a cipher text-only attack, and in February 2008, Pico Computing, Inc revealed its ability and plans to commercialize FPGAs that allow A5/1 to be broken with a rainbow table attack. The system supports multiple algorithms so operators may replace that cipher with a stronger one.

GSM MODEMS AND MODULES

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.

4.6 SIM300 GSM MODULE (GSM / GPRS: SIM300)

Detailed Modem Description:

The Sim300 is a Tri-Brand GSM GPRS solution in a compact plug-in module.

Featuring an industry-standard interface, the sim300 delivers GSM GPRS 900 1800 1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. The leading features of Sim300 make it ideal for virtually unlimited application, such as WLL applications (Fixed Cellular Terminal), M2M application, handheld devices and much more.

1) Sim300 is a Tri-band GSM GPRS module with a size of 40x33x2. 85mm

2) Customized MMI and keypad LCD support

3) An embedded Powerful TCP IP protocol stack

4) Based upon mature and field-proven platform, backed up by our support service, from definition to design and production.

5. RADIO-FREQUENCY IDENTIFICATION (RFID):

Radio-frequency identification (RFID) is a technology that uses communication through the use of radio waves to exchange data between a reader and an electronic tag attached to an object, for the purpose of identification and tracking.

It is possible in the near future, RFID technology will continue to proliferate in our daily lives the way that bar code technology did over the forty years leading up to the turn of the 21st century bringing unobtrusive but remarkable changes when it was new.

RFID makes it possible to give each product in a grocery store its own unique identifying number, to provide assets, people, work in process, medical devices etc. all with individual unique identifiers - like the license plate on a car but for every item in the world. This is a vast improvement over paper and pencil tracking or bar code tracking that has been used since the 1970s. With bar codes, it is only possible to identify the brand and type of package in a grocery store, for instance. Furthermore, passive RFID tags (those without a battery) can be read if passed within close enough proximity to an RFID reader. It is not necessary to "show" them to it, as with a bar code. In other words it does not require line of sight to "see" an RFID tag, the tag can be read inside a case, carton, box or other container, and unlike barcodes RFID tags can be read hundreds at a time. Bar codes can only read one at a time.

Some RFID tags can be read from several meters away and beyond the line of sight of the reader. The application of bulk reading enables an almost-parallel reading of tags.

Radio-frequency identification involves the hardware known as interrogators (also known as readers), and tags (also known as labels), as well as RFID software or RFID middleware.

Most RFID tags contain at least two parts: one is an integrated circuit for storing and processing information, modulating and demodulating a radio-frequency (RF) signal, and other specialized functions; the other is an antenna for receiving and transmitting the signal.

RFID can be either passive (using no battery), active (with an on-board battery that always broadcasts or beacons its signal) or battery assisted passive "BAP" which has a small battery on board that is activated when in the presence of an RFID reader. Passive tags in 2011 start at $ .05 each and for special tags meant to be mounted on metal, or withstand gamma sterilization go up to $5. Active tags for tracking containers, medical assets, or monitoring environmental conditions in data centers all start at $50 and can go up over $100 each. BAP tags are in the $3–10 range and also have sensor capability like temperature and humidity.

The term RFID refers to the technology. The tags should properly be called "RFID tags" not "RFIDs".

Fixed RFID and Mobile RFID: Depending on mobility, RFID readers are classified into two different types: fixed RFID and mobile RFID. If the reader reads tags in a stationary position, it is called fixed RFID. These fixed readers are set up specific interrogation zones and create a "bubble" of RF energy that can be tightly controlled if the physics is well engineered. This allows a very definitive reading area for when tags go in and out of the interrogation zone. On the other hand, if the reader is mobile when the reader reads tags, it is called mobile RFID.

Mobile readers include hand helds, carts and vehicle mounted RFID readers from manufacturers such as Motorola, Intermec, Impinj, Sirit, etc.

There are three types of RFID tags: passive RFID tags, which have no power source and require an external electromagnetic field to initiate a signal transmission, active RFID tags, which contain a battery and can transmit signals once an external source ('Interrogator') has been successfully identified, and battery assisted passive (BAP) RFID tags, which require an external source to wake up but have significant higher forward link capability providing greater range.

There are a variety of groups defining standards and regulating the use of RFID, including the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC), ASTM International, the DASH7 Alliance and EPCglobal. (Refer to Regulation and standardization below.)There are also several specific industries that have set guidelines including the Financial Services Technology Consortium (FSTC) has set a standard for tracking IT Assets with RFID, the Computer Technology Industry Association CompTIA has set a standard for certifying RFID engineers and the International Airlines Transport Association IATA set tagging guidelines for luggage in airport.

6. MAX232

The MAX232 from Maxim was the first IC which in one package contains the necessary drivers (two) and receivers (also two), to adapt the RS-232 signal voltage levels to TTL logic. It became popular, because it just needs one voltage (+5V) and generates the necessary RS-232 voltage levels (approx. -10V and +10V) internally. This greatly simplified the design of circuitry. Circuitry designers no longer need to design and build a power supply with three voltages (e.g. -12V, +5V, and +12V), but could just provide one +5V power supply, e.g. with the help of a simple 78x05 voltage converter.

The MAX232 has a successor, the MAX232**A**. The ICs are almost identical, however, the MAX232A is much more often used (and easier to get) than the original MAX232, and the MAX232A only needs external capacitors 1/10th the capacity of what the original MAX232 needs.

It should be noted that the MAX232(A) is just a driver/receiver. It does not generate the necessary RS-232 sequence of marks and spaces with the right timing, it does not decode the RS-232 signal, it does not provide a serial/parallel conversion. **All it does is to convert signal voltage levels**. Generating serial data with the right timing and decoding serial data has to be done by additional circuitry, e.g. by a 16550 UART or one of these small micro controllers (e.g. Atmel AVR, Microchip PIC) getting more and more popular.

The MAX232 and MAX232A were once rather expensive ICs, but today they are cheap. It has also helped that many companies now produce clones (ie. Sipex). These clones sometimes need different external circuitry, e.g. the capacities of the external capacitors vary. It is recommended to check the data sheet of the particular manufacturer of an IC instead of relying on Maxim's original data sheet.

The original manufacturer (and now some clone manufacturers, too) offers a large series of similar ICs, with different numbers of receivers and drivers, voltages, built-in or external capacitors, etc. E.g. The MAX232 and MAX232A need external capacitors for the internal voltage pump, while the MAX233 has these capacitors built-in. The MAX233 is also between three and ten times more expensive in electronic shops than the MAX232A because of its internal capacitors. It is also more difficult to get the MAX233 than the garden variety MAX232A.

A similar IC, the MAX3232 is nowadays available for low-power 3V logic.

6.1. MAX232 DIP Package

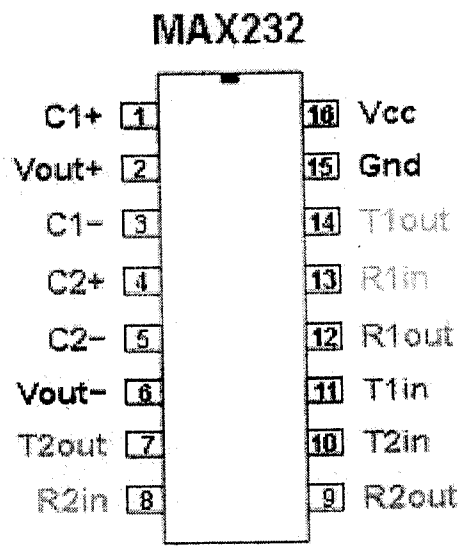


FIG 11: Pin diagram of MAX232

6.2.FUNDAMENTAL TTL GATE

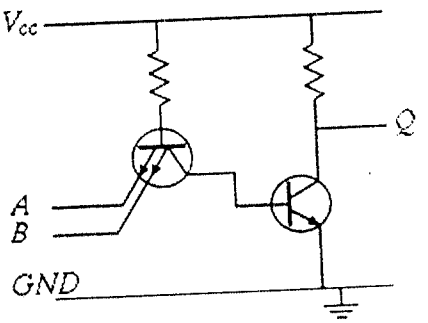


FIG 12: Fundamental TTL gate

Two-input TTL NAND gate with a simple output stage (simplified).TTL is a natural successor of DTL since it is based on the same fundamental concept – implementing the logic gate function by using the base–emitter junctions of a multiple-emitter transistor as switching elements like DTL input diodes. This IC structure is functionally equivalent to multiple transistors where the bases and collectors are tied together. The output of the simple TTL gate is buffered, like DTL, by a common emitter amplifier.

Input logical ones. When all the inputs are held at high voltage, the base–emitter junctions of the multiple-emitter transistor are backward-biased. In contrast with DTL, small (about 10 μA) "collector" currents are drawn by the inputs since the transistor is in a reverse-active mode (with swapped collector and emitter). The base resistor in combination with the supply voltage acts as a substantially constant current source. It passes current through the base–collector junction of the multiple-emitter transistor and the base–emitter junction of the output transistor thus turning it on; the output voltage becomes low (logical zero).

Input logical zero. If one input voltage becomes zero, the corresponding [illegible] transistor connects in parallel to the

two connected in series junctions (the base–collector junction of the multiple-emitter transistor and the base–emitter junction of the second transistor). The input base–emitter junction steers all the base current of the output transistor to the input source (the ground). The base of the output transistor is deprived of current causing it to go into cut-off and the output voltage becomes high (logical one). During the transition the input transistor is briefly in its active region; so it draws a large current away from the base of the output transistor and thus quickly discharges its base. This is a critical advantage of TTL over DTL that speeds up the transition over a diode input structure.

The main disadvantage of TTL with a simple output stage is the relatively high output resistance at output logical "1" that is completely determined by the output collector resistor. It limits the number of inputs that can be connected (the fanout). Some advantage of the simple output stage is the high voltage level (up to V_{CC}) of the output logical "1" when the output is not loaded.

Logic of this type is most frequently encountered with the collector resistor of the output transistor omitted, making an open collector output. This allows the designer to fabricate logic by connecting the open collector outputs of several logic gates together and providing a single external pull-up resistor. If any of the logic gates becomes logic low (transistor conducting), the combined output will be low. Examples of this type of gate are the 7401 and 7403 series.

7.DISPLAY UNIT:

This section describes the operation modes of LCDs, then describes how to program and interface an LCD to PIC Microcontroller.

7.1: LCD OPERATION:

In recent years the LCD is finding widespread use replacing LEDs (seven-segment LEDs or other multi segment LEDs). This is due to the following reasons:

- The declining prices of LCDs.
- The ability to display numbers, characters, and graphics. This is in contrast to LEDs, which are limited to numbers and a few characters.
- Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU (or in some other way) to keep displaying the data.
- Ease of programming for characters and graphics.

7.2: LCD PIN DESCRIPTIONS:

The LCD discussed in this section has 14 pins. The function of each pin is given in the table below.

V_{CC},V_{SS}, AND V_{EE}:

While V_{CC} and V_{SS} provide +5V and ground, respectively, V_{EE} is used for controlling LCD contrast.

RS, REGISTER SELECT :

There are two very important registers inside the LCD. The RS pin is used for their selection as follows. If RS=0, the instruction command code register is selected, allowing the user to send a command such as clear display, cursor at home, etc. If RS=1 the data register is selected, allowing the user to send data to be displayed on the LCD.

R/W, READ/WRITE:

R/W input allows the user to write information to the LCD or read information from it. R/W=1 when reading; R/W=0 when writing.

E, Enable :

The enable pin is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high-to-low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins. This pulse must be a minimum of 450ns wide.

D0-D7 :

The 8-bit data pins, D0-D7, are used to send information to the LCD or read the contents of the LCD's internal registers.

To display letters and numbers, we send ASCII codes for the letters A-Z, a-z, and numbers 0-9 to these pins while making RS=1.

There are also instruction command codes that can be sent to the LCD to clear the display or force the cursor to the home position or blink the cursor. Table lists the instruction command codes.

We also use RS=0 to check the busy flag bit to see if the LCD is ready to receive information. The busy flag is D7 and can be read when R/W=1 and RS=0, as follows: if R/W=1, RS=0. When D7=1 (busy flag=1), the LCD is busy taking care of internal operations and will not accept any new information.

Code(Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display off, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 matrix

TABLE 1: HEX code for LCD display

Pin No	Symbol	Details
1	GND	Ground
2	Vcc	Supply Voltage +5V
3	Vo	Contrast adjustment
4	RS	0->Control input, 1-> Data input
5	R/W	Read/ Write
6	E	Enable
7 to 14	D0 to D7	Data
15	VB1	Backlight +5V
16	VB0	Backlight ground

TABLE 2: PIN details of LCD

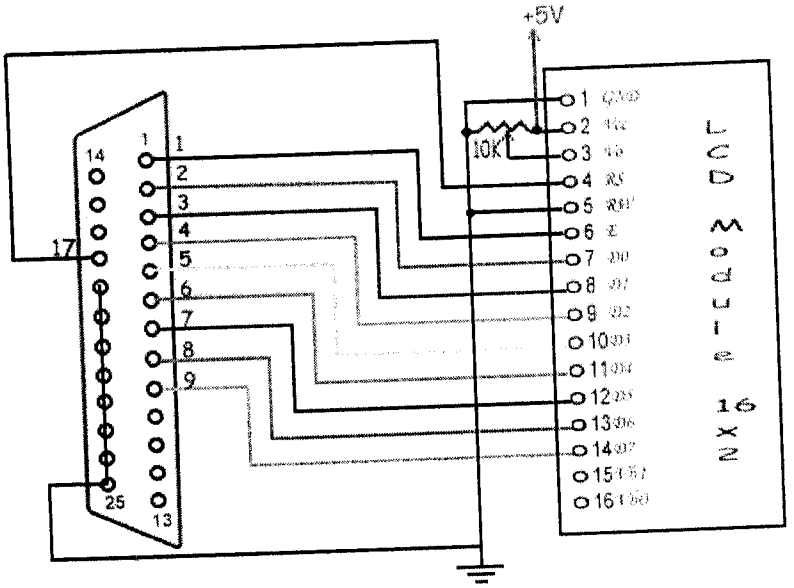


FIG 13: LCD interface

7.3: Steps to Interface LCD with PIC Microcontroller

STEP 1: Identify:

Determine what you want LCD are available in many flavors which are specified as follows 16x1 , 16x2 , 20x2 in the format AxB where A is the number of columns (chatters) and B is the number of Rows (lines) An LCD might also be Back lit .

STEP 2 : Connect : Most of the LCD's follow the standard Hitachi Pin out which is simple.

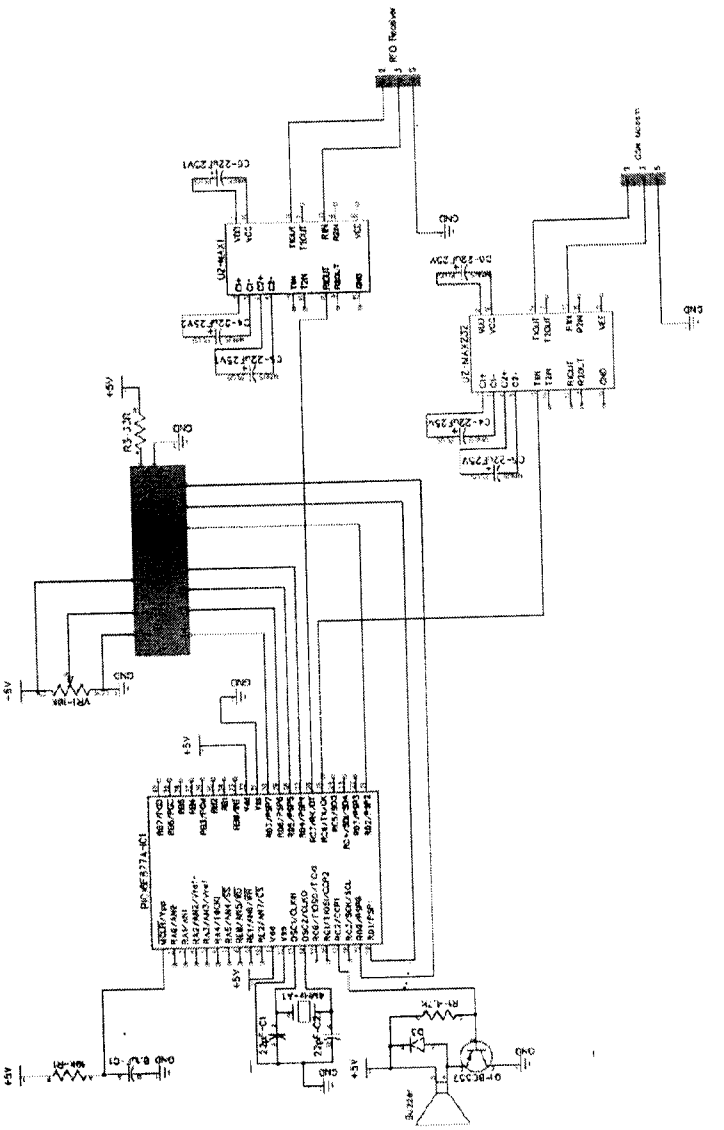
STEP 3 : Interface :

Now connect pins RS, RW, E, D0 - D7 to pins on the micro controller Lets suppose I connect Data bus on port A and the RS , RW , E on port B . (you can

save pins by using LCD in Nibble Mode (4 data pins) and permanently grounding the RW line (always in write mode) . Now well see how to go from simple switching it on to graphics on the LCD .

An Intelligent LCD Need Only a few Commands and data to function Command Set for the LCD.

CIRCUIT DIAGRAM OF INTELLIGENT ROUTE DIVERTER :



8.1 WORKING PRINCIPLE:

The main unit of the system is the PIC 16F877A it is checked for the basic MCLR connection and the crystal oscillator connection and the necessary power supply and ground connections. The port c 2^{nd} pin is connected to the buzzer of each vehicle, when the information is obtained through the GSM and serial communication that the route has been changed then the information will be sent to the microcontroller and the information will be displayed in the LCD unit and if there is any change in the direction there will be a alert buzzer to alert the driver.

Radio-frequency identification (RFID) is a technology that uses communication via radio waves to exchange data between a reader and an electronic tag attached to an object, for the purpose of identification and tracking.

This RFID tag is used here for the convenience of obtaining the correct data to the correct user, two GSM modems for transmission an reception is connected to the PORT C and D. The RFID tag is connected to the receiving end and it is used for obtaining the data from the main unit. Thus for various users the GSM module is used to transmit and receive the data. The DC motor is connected to the 40^{th} pin to control the vehicle.

8.2 POWER SUPPLY:

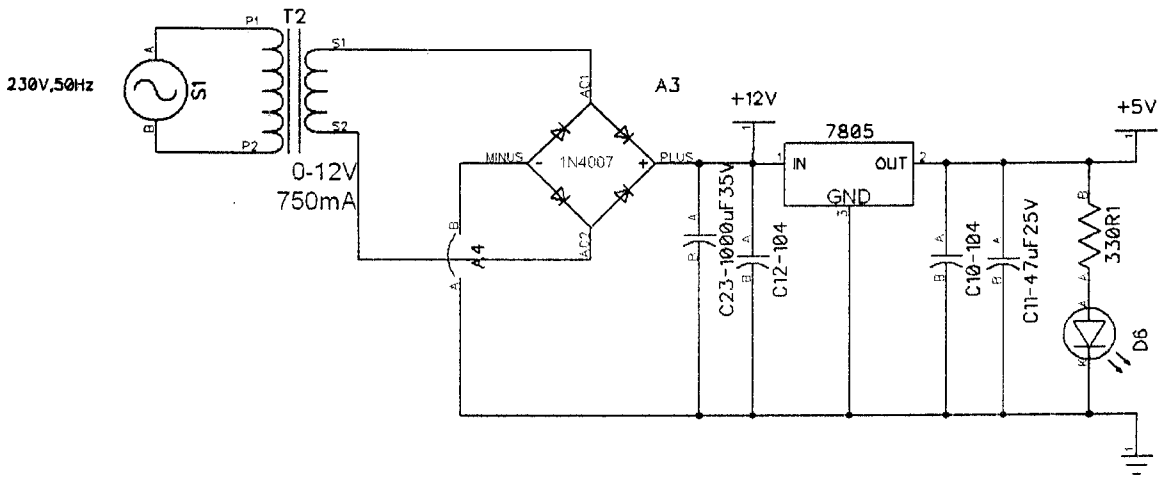


FIG 15. Power supply circuit diagram

8.3 POWER SUPPLY DESCRIPTION:

In the transformer section AC 230v is given. The output of the transformer is 12V. The Bridge circuit is connected across the transformer. The Rectified output is filtered with a 1000uf capacitor. Another capacitor is connected across the Filter for charging and discharging. The Regulator IC 7805 regulates the voltage to 5v. The first pin of the IC 7805 is connected to input, the second pin is grounded and the output is taken from the third pin.

9. SOFTWARE REQUIREMENTS

9.1 SOFTWARE TOOLS

- MPLAB
- Protel
- Propic
- HI-Tech PIC C Compiler

9.2 MPLAB INTEGRATION

MPLAB Integrated Development Environment (IDE) is a free, integrated toolset for the development of embedded applications employing Microchip's PIC micro and dsPIC microcontrollers. MPLAB IDE runs as a 32-bit application on MS Windows, is easy to use and includes a host of free software components for fast application development and super-charged debugging. MPLAB IDE also serves as a single, unified graphical user interface for additional Microchip and third party software and hardware development tools. Moving between tools is a snap, and upgrading from the free simulator to MPLAB ICD 2 or the MPLAB ICE emulator is done in a flash because MPLAB IDE has the same user interface for all tools.

Choose MPLAB C18, the highly optimized compiler for the PIC18 series microcontrollers, or try the newest Microchip's language tools compiler, MPLAB C30, targeted at the high performance PIC24 and dsPIC digital signal controllers. Or, use one of the many products from third party language tools vendors. They integrate into MPLAB IDE to function transparently from the MPLAB project

9.3 INTRODUCTION TO EMBEDDED 'C':

Ex: Hitec – c, Keil – c

HI-TECH Software makes industrial-strength software development tools and C compilers that help software developers write compact, efficient embedded processor code.

For over two decades HI-TECH Software has delivered the industry's most reliable embedded software development tools and compilers for writing efficient and compact code to run on the most popular embedded processors. Used by tens of thousands of customers including General Motors, Whirlpool, Qualcomm, John Deere and many others, HI-TECH's reliable development tools and C compilers, combined with world-class support have helped serious embedded software programmers to create hundreds of breakthrough new solutions.

Whichever embedded processor family you are targeting with your software, whether it is the ARM, PIC or 8051 series, HI-TECH tools and C compilers can help you write better code and bring it to market faster.

HI-TECH PICC is a high-performance C compiler for the Microchip PIC micro 10/12/14/16/17 series of microcontrollers. HI-TECH PICC is an industrial-strength ANSI C compiler - not a subset implementation like some other PIC compilers. The PICC compiler implements full ISO/ANSI C, with the exception of recursion. All data types are supported including 24 and 32 bit IEEE standard floating point. HI-TECH PICC makes full use of specific PIC features and using an intelligent optimizer, can generate high-quality code easily rivaling hand-written assembler. Automatic handling of page and bank selection frees the programmer from the trivial details of assembler code.

9.4 EMBEDDED "C" COMPILER

- ANSI C - full featured and portable
- Reliable - mature, field-proven technology
- Multiple C optimization levels
- An optimizing assembler
- Full linker, with overlaying of local variables to minimize RAM usage
- Comprehensive C library with all source code provided
- Includes support for 24-bit and 32-bit IEEE floating point and 32-bit long data types
- Mixed C and assembler programming
- Unlimited number of source files
- Listings showing generated assembler
- Compatible - integrates into the MPLAB IDE, MPLAB ICD and most 3rd-party development tools
- Runs on multiple platforms: Windows, Linux, UNIX, Mac OS X, Solaris

9.5 EMBEDDED DEVELOPMENT ENVIRONMENT

This environment allows you to manage all of your PIC projects. You can compile, assemble and link your embedded application with a single step.

Optionally, the compiler may be run directly from the command line, allowing you to compile, assemble and link using one command. This enables the compiler to be integrated into third party development environments, such as Microchip's MPLAB IDE.

9.6 EMBEDDED SYSTEM TOOLS

9.6.1 ASSEMBLER

An assembler is a computer program for translating assembly language — essentially, a mnemonic representation of machine language — into object code. A cross assembler (see cross compiler) produces code for one type of processor, but runs on another. The computational step where an assembler is run is known as assembly time. Translating assembly instruction mnemonics into opcodes, assemblers provide the ability to use symbolic names for memory locations (saving tedious calculations and manually updating addresses when a program is slightly modified), and macro facilities for performing textual substitution — typically used to encode common short sequences of instructions to run inline instead of in a subroutine. Assemblers are far simpler to write than compilers for high-level languages.

9.6.2 ASSEMBLY LANGUAGE HAS SEVERAL BENEFITS

Speed: Assembly language programs are generally the fastest programs around.

Space: Assembly language programs are often the smallest.

Capability: You can do things in assembly which are difficult or impossible in High level languages.

Knowledge: Your knowledge of assembly language will help you write better programs, even when using High level languages. An example of an assembler we use in our project is RAD 51.

9.6.3 SIMULATOR

Simulator is a machine that simulates an environment for the purpose of training or research. We use a UMPS simulator for this purpose in our project.

9.6.4 UMPS

Universal microprocessor program simulator simulates a microcontroller with its external environment. UMPS is able to simulate external components connected to the microcontroller. Then, debug step is dramatically reduced. UMPS is not dedicated to only one microcontroller family, it can simulate all kind of microcontrollers. The main limitation is to have less than 64K-Bytes of RAM and ROM space and the good microcontroller library. UMPS provide all the facilities other low-cost simulator does not have. It offers the user to see the "real effect" of a program and a way to change the microcontroller family without changing IDE. UMPS provide a low-cost solution to the problems. UMPS is really the best solution to your evaluation.

9.6.5 UMPS KEY FEATURES

- The speed, UMPS can run as fast as 1/5 the real microcontroller speed. No need to wait 2 days to see the result of a LCD routine access. All the microcontroller parts are simulated, interrupts, communication protocol, parallel handshake, timer and so on.
- UMPS have an integrated assembler/disassembler and debugger. It is able to accept an external assembler or compiler. It has a text editor which is not limited to 64K-bytes and shows keyword with color. It

can also communicate with an external compiler to integrate all the debug facilities you need.

- UMPS is universal, it can easily be extended to other microcontroller with a library. Ask us for toolkit development.
- External resource simulation is not limited. It can be extended to your proper needs by writing your own DLL.
- UMPS allows you to evaluate at the lowest cost the possibility to build a microcontroller project without any cable. - UMPS include a complete documentation on each microcontroller which describe special registers and each instruction

9.6.6 COMPILER

A compiler is a program that reads a program in one language, the source language and translates into an equivalent program in another language, the target language. The translation process should also report the presence of errors in the source program.

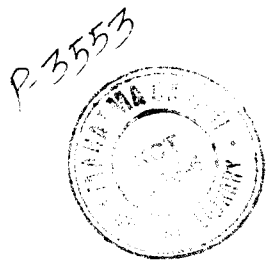
9.6.7 COUSINS OF THE COMPILER

1. Preprocessor.
2. Assembler.
3. Loader and Link-editor.

A naive approach to that front end might run the phases serially.

1. Lexical analyzer takes the source program as an input and produces a long string of tokens.
2. Syntax Analyzer takes an out of lexical analyzer and produces a large tree.

Semantic analyzer takes the output of syntax analyzer and produces another tree. Similarly, intermediate code generator takes a tree as an input produced by semantic analyzer and produces intermediate code



9.6.8 FABRICATION DETAILS

The fabrication of one demonstration unit is carried out in the following sequence.

- Finalizing the total circuit diagram, listing out the components and sources of procurement.
- Procuring the components, testing the components and screening the components.
- Making layout, repairing the interconnection diagram as per the circuit diagram.
- Assembling the components as per the component layout and circuit diagram and soldering components.
- Integrating the total unit, interwiring the unit and final testing the unit.

9.7 DESIGN OF EMBEDDED SYSTEM

Like every other system development design cycle embedded system too have a design cycle. The flow of the system will be like as given below. For any design cycle these will be the implementation steps. From the initial state of the project to the final fabrication the design considerations will be taken like the software consideration and the hardware components, sensor, input and output. The electronics usually uses either a microprocessor or a microcontroller. Some large or old systems use general-purpose mainframe computers or minicomputers.

9.8 USER INTERFACES

User interfaces for embedded systems vary widely, and thus deserve some special comment. User interface is the ultimate aim for an embedded module as to the user to check the output with complete convenience. One standard interface, widely used in embedded systems, uses two buttons (the absolute minimum) to

control a menu system (just to be clear, one button should be "next menu entry" the other button should be "select this menu entry").

Another basic trick is to minimize and simplify the type of output. Designs sometimes use a status light for each interface plug, or failure condition, to tell what failed. A cheap variation is to have two light bars with a printed matrix of errors that they select- the user can glue on the labels for the language that he speaks. For example, most small computer printers use lights labeled with stick-on labels that can be printed in any language. In some markets, these are delivered with several sets of labels, so customers can pick the most comfortable language.

In many organizations, one person approves the user interface. Often this is a customer, the major distributor or someone directly responsible for selling the system.

9.9 PLATFORM

There are many different CPU architectures used in embedded designs such as ARM, MIPS, Coldfire/68k, PowerPC, X86, PIC, 8051, Atmel AVR, H8, SH, V850, FR-V, M32R etc.

This in contrast to the desktop computer market, which as of this writing (2003) is limited to just a few competing architectures, mainly the Intel/AMD x86, and the Apple/Motorola/IBM PowerPC, used in the Apple Macintosh. With the growing acceptance of Java in this field, there is a tendency to even further eliminate the dependency on specific CPU/hardware (and OS) requirements.

Standard PC/104 is a typical base for small, low-volume embedded and ruggedized system design. These often use DOS, Linux or an embedded real-time operating system such as QNX or Inferno.

A common configuration for very-high-volume embedded systems is the system on a chip, an application-specific integrated circuit, for which the CPU was purchased as intellectual property to add to the IC's design. A related common scheme is to use a field-programmable gate array, and program it with all the logic, including the CPU. Most modern FPGAs are designed for this purpose.

9.10 TOOLS

Like typical computer programmers, embedded system designers use compilers, assemblers, and debuggers to develop embedded system software. However, they also use a few tools that are unfamiliar to most programmers.

Software tools can come from several sources:

- Software companies that specialize in the embedded market.
- Ported from the GNU software development tools.

Sometimes, development tools for a personal computer can be used if the embedded processor is a close relative to a common PC processor. Embedded system designers also use a few software tools rarely used by typical computer programmers.

One common tool is an "in-circuit emulator" (ICE) or, in more modern designs, an embedded debugger. This debugging tool is the fundamental trick used to develop embedded code. It replaces or plugs into the microprocessor, and provides facilities to quickly load and debug experimental code in the system. A small pod usually provides the special electronics to plug into the system. Often a personal computer with special software attaches to the pod to provide the debugging interface.

Another common tool is a utility program (often home-grown) to add a checksum or CRC to a program, so it can check its program data before executing it.

An embedded programmer that develops software for digital signal processing often has a math workbench such as MathCad or Mathematical to simulate the mathematics.

Less common are utility programs to turn data files into code, so one can include any kind of data in a program. A few projects use Synchronous programming languages for extra reliability or digital signal processing.

9.11 DEBUGGING

Debugging is usually performed with an in-circuit emulator, or some type of debugger that can interrupt the microcontroller's internal microcode. The microcode interrupt lets the debugger operate in hardware in which only the CPU works. The CPU-based debugger can be used to test and debug the electronics of the computer from the viewpoint of the CPU. This feature was pioneered on the PDP-11.

As the complexity of embedded systems grows, higher level tools and operating systems are migrating into machinery where it makes sense. For example, cell phones, personal digital assistants and other consumer computers often need significant software that is purchased or provided by a person other than the manufacturer of the electronics. In these systems, an open programming environment such as Linux, OSGi or Embedded Java is required so that the third-party software provider can sell to a large market.

9.12 OPERATING SYSTEM

Embedded systems often have no operating system, or a specialized embedded operating system (often a real-time operating system), or the programmer is assigned to port one of these to the new system.

9.13 BUILT- IN SELF- TEST

Most embedded systems have some degree or amount of built-in self-test.

There are several basic types.

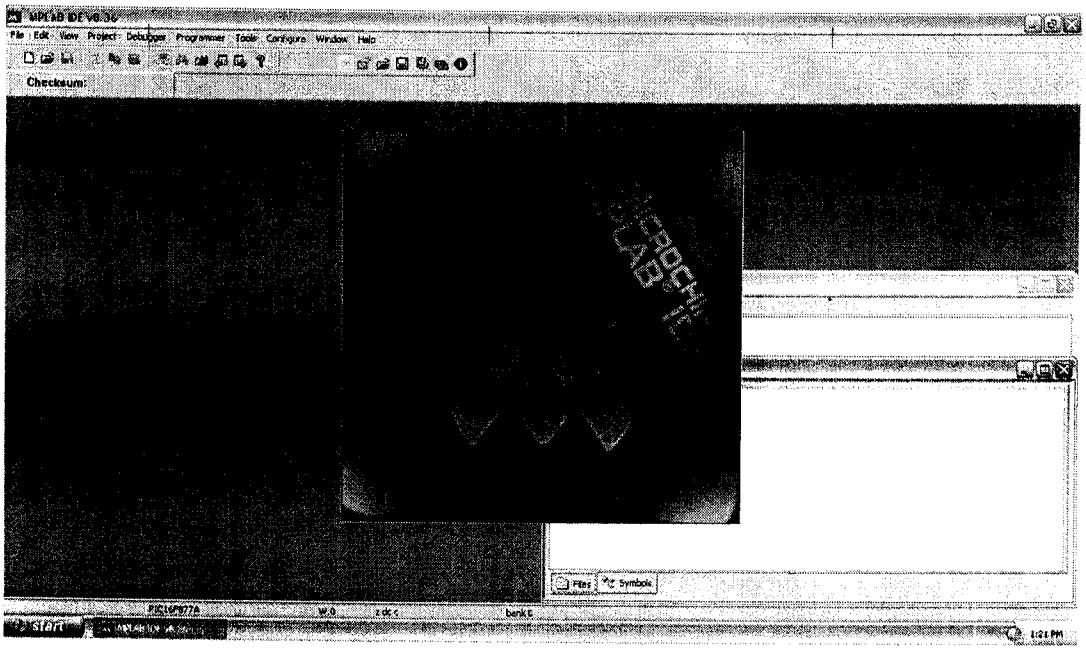
1. Testing the computer.
2. Test of peripherals.
3. Tests of power.
4. Communication tests.
5. Cabling tests.
6. Rigging tests.
7. Consumables test.
8. Operational test.
9. Safety test.

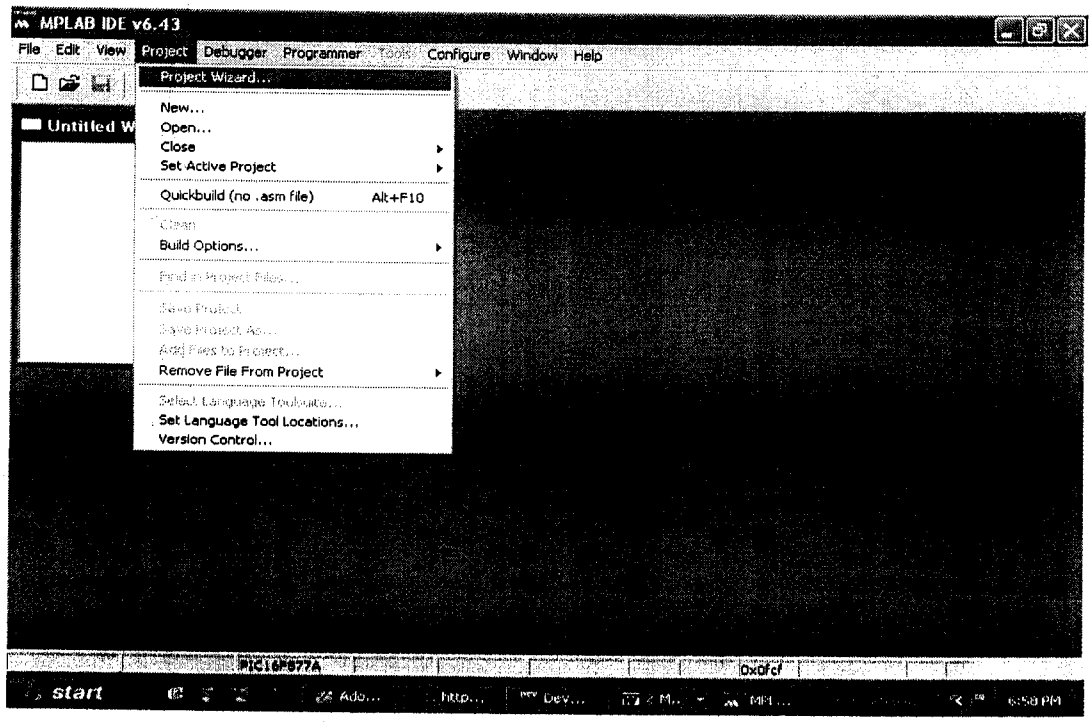
9.14 START UP

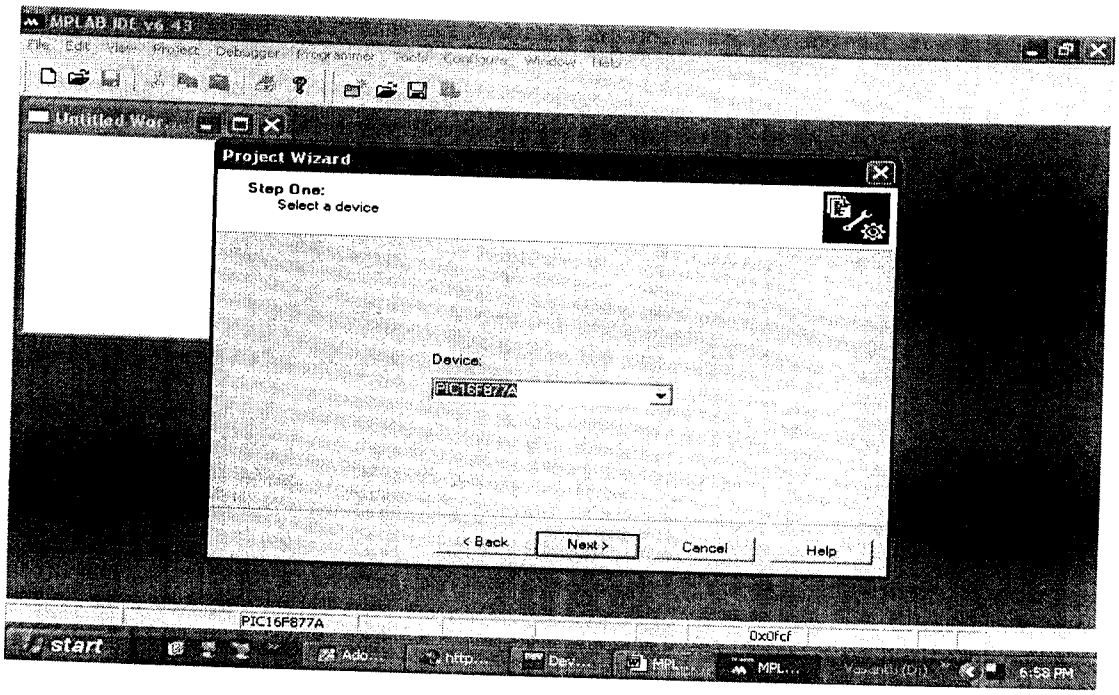
All embedded systems have start-up code. Usually it disables interrupts, sets up the electronics, tests the computer (RAM, CPU and software), and then starts the application code. Many embedded systems recover from short-term power failures by restarting (without recent self-tests). Restart times under a tenth of a second are common.

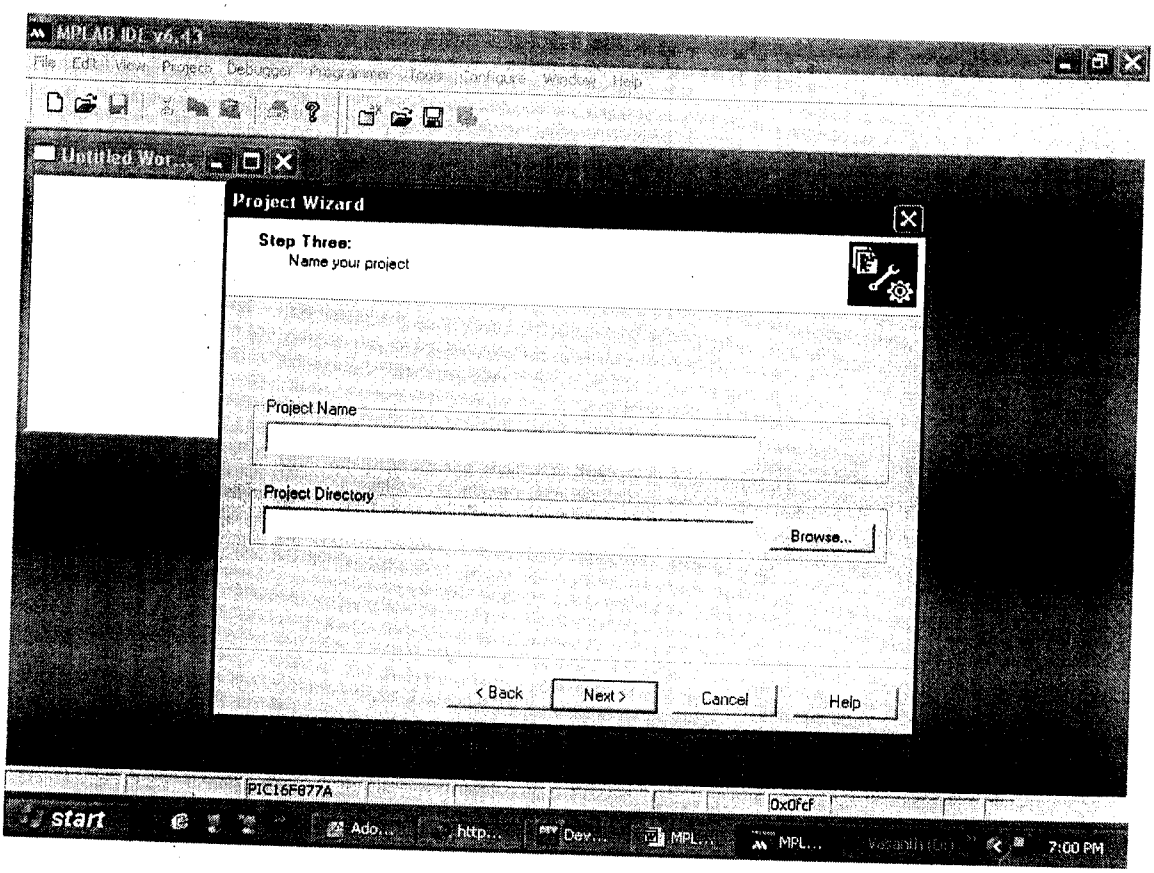
Many designers have found a few LEDs useful to indicate errors (they help troubleshooting). A common scheme is to have the electronics turn on all of the LED(s) at reset (thereby proving that power is applied and the LEDs themselves work), whereupon the software changes the LED pattern as the Power-On Self Test executes. After that, the software may blink the LED(s) or set up light patterns during normal operation to indicate program execution progress or errors. This serves to reassure most technicians/engineers and some users. An interesting exception is that on electric power meters and other items on the street, blinking lights are known to attract attention and vandalism.

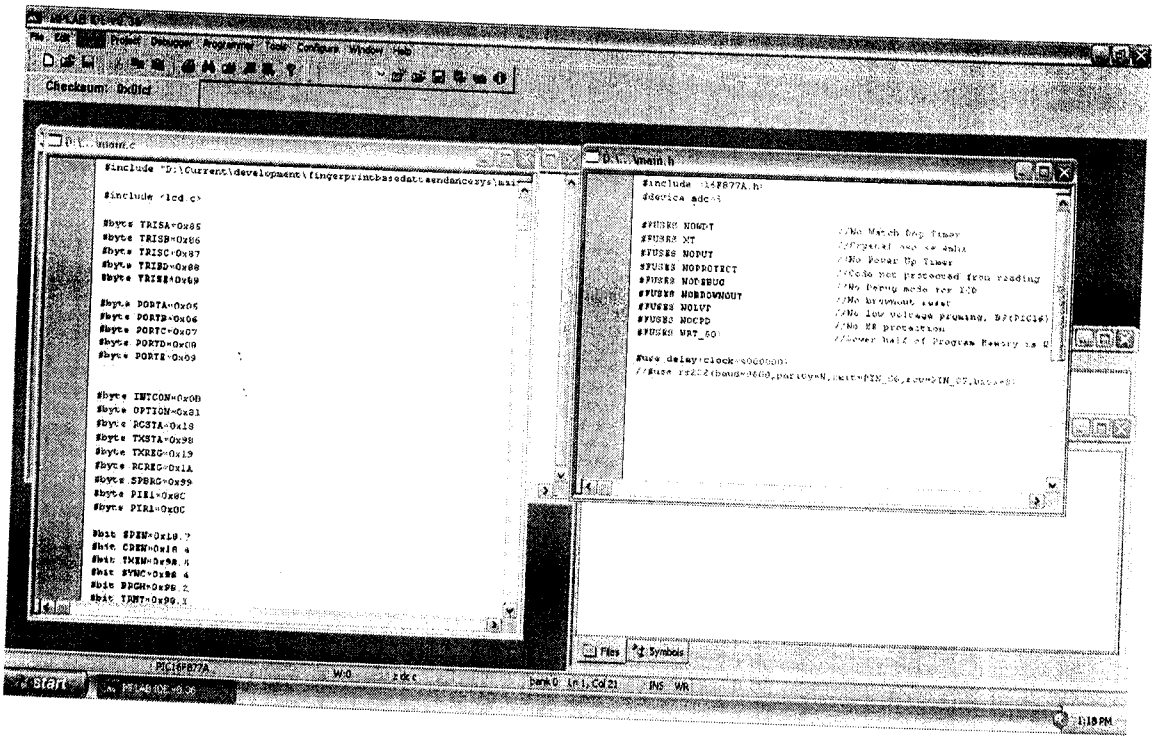
9.15 PHASES OF COMPILER

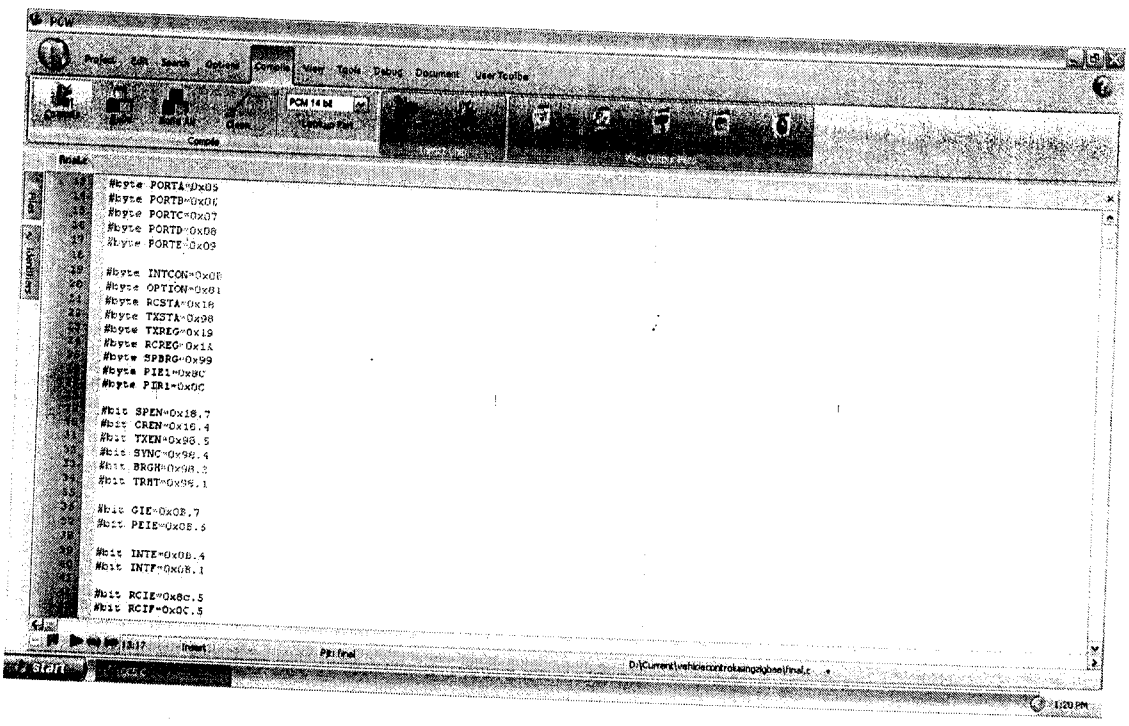


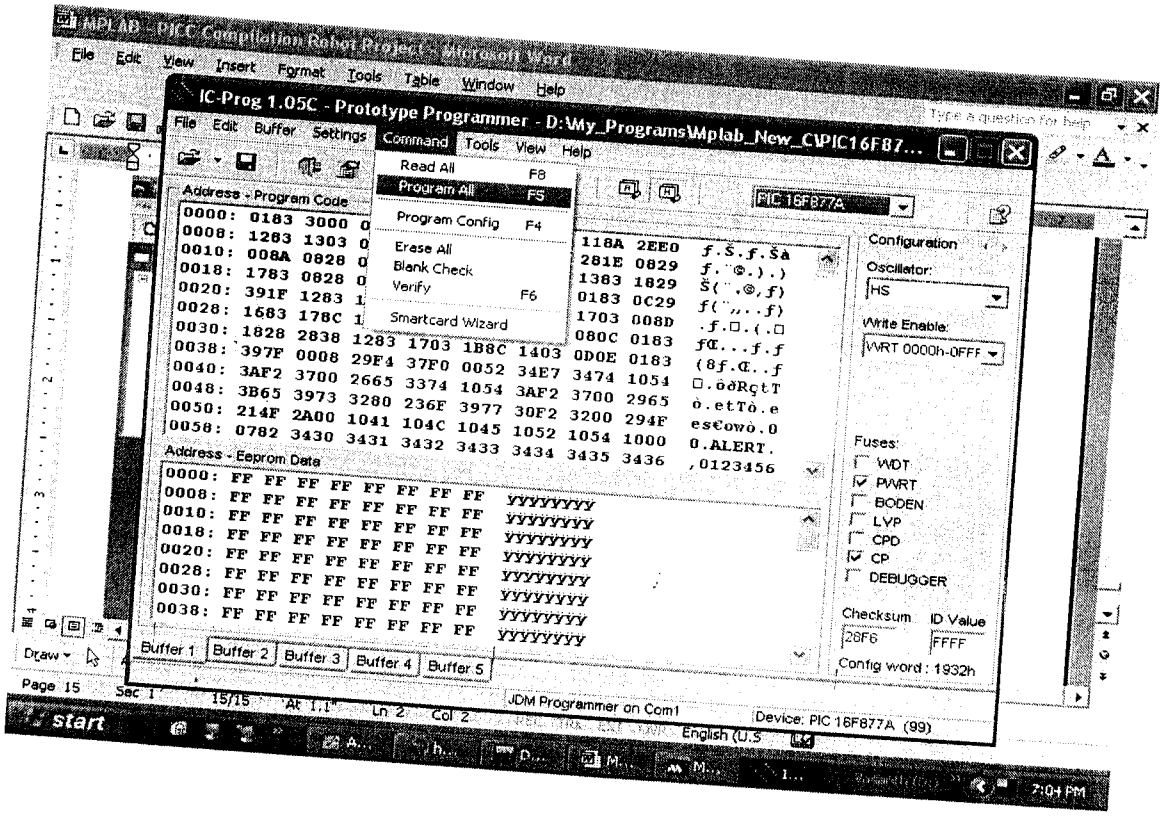


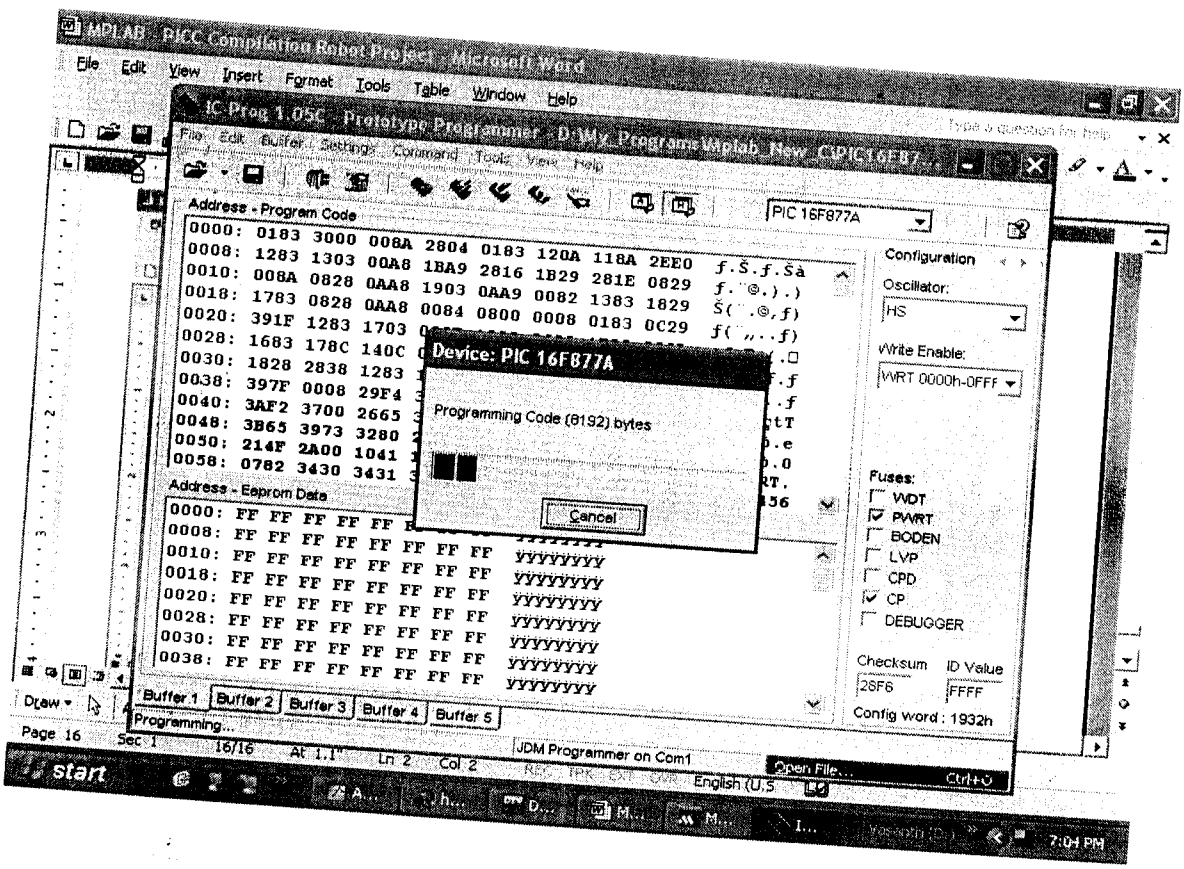

MPLAB IDE v6.43
Untitled Wor...
Project Wizard
Step One:
Select a device
Device:
PIC16F877A
< Back
Next >
Cancel
Help
PIC16F877A
0x0fcf
start
6:58 PM


MPLAB IDE v6.43
Untitled Wor...
Project Wizard
Step Three:
Name your project
Project Name
Project Directory
Browse...
< Back
Next >
Cancel
Help
PIC16F877A
0x0fcf
start
7:00 PM


#include "D:\Current\development\fingerprintbasedatmegandarduino\main
#include <lcd.c>
#byte TRISA=0x85
#byte TRISB=0x86
#byte TRISC=0x87
#byte TRISD=0x88
#byte TRISE=0x89
#byte PORTA=0x05
#byte PORTB=0x06
#byte PORTC=0x07
#byte PORTD=0x08
#byte PORTE=0x09
#byte INTCON=0x0B
#byte OPTION=0x81
#byte RCSTA=0x18
#byte TXSTA=0x98
#byte TXREG=0x19
#byte RCREG=0x1A
#byte SPBRG=0x99
#byte PIE1=0x8C
#byte PIR1=0x0C
#bit SPEN=0x18.7
#bit CREN=0x18.4
#bit TXEN=0x98.5
#bit SYNC=0x98.4
#bit BRGH=0x98.2
#bit TRMT=0x98.1
#include "16F877A.h"
#device adc=8
#FUSES NOWDT
#FUSES XT
#FUSES NOPUT
#FUSES NOPROTECT
#FUSES NODEBUG
#FUSES NOBROWNOUT
#FUSES NOLVP
#FUSES NOCPD
#FUSES WRT_60
#use delay(clock=4000000)
Files
Symbols
PIC16F877A
Start


#byte PORTA=0x05
#byte PORTB=0x06
#byte PORTC=0x07
#byte PORTD=0x08
#byte PORTE=0x09
#byte INTCON=0x0B
#byte OPTION=0x81
#byte RCSTA=0x18
#byte TXSTA=0x98
#byte TXREG=0x19
#byte RCREG=0x1A
#byte SPBRG=0x99
#byte PIE1=0x8C
#byte PIR1=0x0C
#bit SPEN=0x18.7
#bit CREN=0x18.4
#bit TXEN=0x98.5
#bit SYNC=0x98.4
#bit BRGH=0x98.2
#bit TRMT=0x98.1
#bit GIE=0x0B.7
#bit PEIE=0x0B.6
#bit INTE=0x0B.4
#bit INTF=0x0B.1
#bit RCIE=0x8C.5
#bit RCIF=0x0C.5
PCM 14 bit
Compile
Build All
Clean
Start


IC-Prog 1.05C - Prototype Programmer - D:\My_Programs\Mplab_New_C\PIC16F87...
File Edit Buffer Settings Command Tools View Help
Read All F8
Program All F5
Program Config F4
Erase All
Blank Check
Verify F6
Smartcard Wizard
PIC 16F877A
Address - Program Code
Address - Eeprom Data
Configuration
Oscillator:
HS
Write Enable:
WRT 0000h-0FFF
Fuses:
WDT
PWRT
BODEN
LVP
CPD
CP
DEBUGGER
Checksum
ID Value
28F6
FFFF
Config word : 1932h
Buffer 1
Buffer 2
Buffer 3
Buffer 4
Buffer 5
JDM Programmer on Com1
Device: PIC 16F877A (99)


IC-Prog 1.05C - Prototype Programmer - D:\My_Programs\Mplab_New_C\PIC16F87...
Device: PIC 16F877A
Programming Code (8192) bytes
Cancel
PIC 16F877A
Address - Program Code
Address - Eeprom Data
Configuration
Oscillator:
HS
Write Enable:
WRT 0000h-0FFF
Fuses:
WDT
PWRT
BODEN
LVP
CPD
CP
DEBUGGER
Checksum
ID Value
28F6
FFFF
Config word : 1932h
Buffer 1
Buffer 2
Buffer 3
Buffer 4
Buffer 5
Programming...
JDM Programmer on Com1

10. ADVANTAGES:

- The Project uses inexpensive, easily deployable components.
- A simple setup that could be erected anywhere.
- There is always a provision for continuous improvement.

11. ENHANCEMENTS:

- The train speed and thereby the arrival time could be continuously monitored and intimated by using more sensors
- In the future the same project could be implemented using GPS technology

12. CONCLUSION:

Although the project has been designed taking a single train into consideration, slight modifications would result in a setup that could handle trains of varying lengths and speed. Another scope for improvisation in the future exists where a server could be setup that could monitor and handle all level crossings at different locations.

13. REFERENCES:

www.howstuffworks.com

www.wikipedia.com

www.gsminformation.org/dale/nmea.htm

www.kh-gsm.de/nmea-faq.htm

Ying-Wen Bai, Cheng-Yu Hsu, Yu-Nien Yang, Chih-Tao Lu, and Chien-Yung C of heng, "Design and Implementation of a Remote Electronic Measurement System"

http://www.hp.com/go/HPVEE