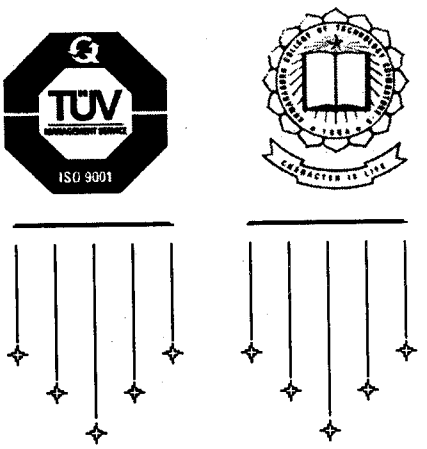
PREPAID EB CARDS

Project Report

Submitted by

Gokul Krishnaswamy
Mihir. C. Patel
Rajan Sundaran



Guided by

K.R. Baskaran B.E., M.S.
Assistant. Professor
Department of Information Technology

P-892

In partial fulfillment of the requirements
for the award of the degree of

Bachelor of Engineering in Information Technology

of the Bharathiar University, Coimbatore.

Department of Information Technology

KUMARAGURU COLLEGE OF TECHNOLOGY

Coimbatore – 641 006.

i-Tech Solutions

130/1, Designare Complex 1st Floor,
Dr. Nanjappa Road, Coimbatore.
TamilNadu- 641018. India

Phone : +91-422-305585
e.mail : tkumar@eth.net

TO WHOMSOEVER IT MAY CONCERN

Sir/Madam,

This is to certify that the following students

1) Mr.Gokul Krishnaswamy (9927S0051)
2) Mr.Mihir C. Patel (9927S0058)
3) Mr.Rajan Sundaran (9927S0069)

of final year B.E (Information Technology),Kumaraguru College of Technology, have done a project for us in embedded systems entitled **"Pre-paid EB cards".** The project was done in a span of three months (from November 2002 to March 2003) and has been completed successfully. We found their performance and conduct during the period as excellent.

M. Senthil Kumar
For,
Mr.Sivaraj
(Project Co-ordinator)

Date : March 18, 2003.

Place : COIMBATORE.

I-TECH SOLUTIONS
130/1, Dr. Nanjappa Road,
COIMBATORE-641 018.
PH: 305585

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved principal **Mr. K.K. Padmanaban** B.Sc. (engg), M.Tech., Ph.D., for all the facilities provided in carrying out the project work.

We express our gratitude and indebtedness to Dr. S. Thangasamy B.E.(Hons)., Ph.D. Head of the Department of Computer Science and Engineering, for his full fledged technical guidance, constant encouragement and suggestions in carrying out this project.

We also thank our project guide Mr. K.R. Baskaran B.E., M.S. assistant professor, Department of Information Technology for his valuable guidance and encouragement at all stages of our project.

We also like to think all the teaching and non teaching staff without whom our project would not have been a success.

SYNOPSIS

"Prepaid EB cards" aims at modifying the existing energy monitoring and billing mechanism. The core motive is to automise the entire system by supplementing a microcontroller to the energy meter thereby minimizing the modification required and hence increasing the practical viability of the entire project. The reason behind the work done was to propose a scheme that would appear lucrative to all strata of society

The currently existing method for monitoring the electricity consumption involves a lengthy process of assigning a person with the duty of noting the consumption reading on the meter and submitting the same to the billing department. The bill is generated accordingly. The consumer goes to the electricity board's office and stands in lengthy queues in order to pay the bill.

This cumbersome process is avoided by using a prepaid card that may be beneficial to the Government. The current process has a drawback where in there is a danger of delay in payments. Our system of prepayment would mean elimination of these payment arrears. This system has advantages on the consumers end as the consumer does not pay the caution deposit, thereby lowering his initial investment. This system has a provision for a secondary card so as to avoid total power failure when the balance in the card goes to zero. It also has the added advantage of alarming the consumer when the balance reaches a threshold value so that he will have sufficient time for replacement.

CONTENTS

1. INTRODUCTION

1.1 Current Status of the Problem taken up

"Necessity is the mother of invention" goes a famous saying and the necessity to surge ahead in life gave rise to the invention of power. Electricity is a major driving force especially in a country like India because agriculture and power have worked in tandem to help us reach where we are today. Electricity will remain to be of relevance if we are to realize vision 2020.

The existing system for the metering of electricity bills poses certain inconveniences to the end user. In the existing system the electricity board deputes officials to manually note the meter reading. This process not only involves a huge labor force but also the cumbersome process of door to door monitoring. In India the payment dues are accepted only on weekdays during working hours. The consumer thus has to take time off his busy schedule and personally make payments at the electricity office.

Another relevant problem is that a sizeable percentage of student communities living in urban areas occupy rented rooms. The caution deposits that the landlord demands make life uncomfortable, especially to those who are from economically poor background. A system wherein the caution deposit could be reduced would be beneficial to both the student as well as the landlord.

Wastage of power is inevitable in large industries. This is because the existing system provides no check on the consumption and therefore there is a tendency to be careless. If there was a system that allotted a fixed ration of power for a period then the workers would be more careful thereby minimizing the electricity loss.

In India 45% of the people live below the poverty line. With this economic constraint many people find the initial caution deposit unaffordable. This is the main reason that electricity has not made its way into the remote places even today. A new system that can be afforded by all strata of society is therefore essential.

In India a scheme for temporary connection was introduced in the year 1984 under the name of *'Tatkal scheme'*. Though this scheme was initially a success it failed to meet its ultimate goal.

Taking into consideration the need to further simplify the existing system, we decided to take up a project that would simplify the entire process of electricity bill payment. Since it is not possible to change the entire system we adopted a means to alter the existing system with simple modifications. Once implemented this project would save expenses of the EB department, which are otherwise unavoidable.

The field of embedded engineering has made inroads into almost all walks of life. Today we find embedded systems in washing machine, microwave ovens and other such home appliances. Our project aims at bridging the past and the future. By using micro controllers we aim to change the system that has been in existence for the past 50 years to a completely automised system with minimum human interaction involved.

1.2 RELEVANCE AND IMPORTANCE

In the previous section we discussed the various problems in the prevailing system. In this section let us see how our project would help tackling each of those problems.

In a completely automised system there is minimal human interaction as a result of which the EB department can cut down on the expenditure incurred in the form of salary. The user can plan his electricity budget before hand.

Once implemented, the students could buy their own cards and carry it with them so that they could consume current as long as they wish to. This portability is also helpful to people who are always on the move. For eg: a card bought in one state in India could be used in other states as well.

In big industries wastage of power is inevitable. If we split the distribution of energy department wise and allot a fixed quota to each then the wastage of power would be minimal. This is exactly what our project would provide. In the event of a strike or stoppage of work the simple removal of the prepaid card would result in complete power shutdown, there by elimination any possibility of power wastage.

Finally to ease the burden on the economically backward people, this project would come as a major relief because there is no question of caution deposit. The simplicity of the entire project makes it all the more practically viable because it does not involve any major alteration to the existing system.

As was discussed in the beginning of this introduction, agriculture is well and truly the backbone of Indian economy. Even the remotest portions of India thrive on agriculture and this is directly or indirectly related to power because agriculture involves pumping and distribution of water over large areas of land.

This project also has the additional facility of warning the user before the card drains out completely. The project also has a LCD display to make it convenient for the user to know his current status. The greatest advantage of using this project is that it involves no major modification to the existing system. A single sensor interfaced to the existing meter and a micro controller is enough to automise the entire system.

We have simulated the entire system with a desktop model using a commercially available meter and the working of the various components involved are discussed in the subsequent chapters.

2. LITERATURE SURVEY

In a recent survey that was conducted by ICAR, 68% of the co-operative societies preferred a change in the existing system of power connection. The major area of complaint was the considerable high investment involved in the initial stages. Another problem was the lack of proximity between the rural user and the EB office. This meant that in order to pay his monthly bills he had to make visits to the far off EB office.

On a trial basis the Japanese government implemented a new scheme in July 1998. Their work involved a case study in a village on the island of Kyoto. Kyoto has a large percentage of agriculturists whose lives greatly benefited from this project. The over whelming response obtained from the natives of Kyoto has prompted the Japanese government to implement the same on a larger scale. The implementation of this project is still in the pipeline because of the fear of piracy of the pre paid cards. Another problem is the reprogramming/replacement of the micro controller involved. The following has been taken care of by sealing the equipment with a hologram which when meddled with will result in prosecution. The menace of piracy has been settled by using the power cable to set up a network with the EB office so as to prove the authenticity of the card. The huge investment involved in creating a wired network has been nullified by using the power cable for data transmission.

The reason to pursue this project was simple. If this project was such a huge success in the small island of Kyoto, there is no reason why it would fail in an agriculture dependent nation like India. The Indian Government would most definitely face the same threats that the Japanese government faced, if implemented. But then again, the same solutions could be adopted.

3. SOFTWARE REQUIREMENTS

This is a embedded system project and the components involved here are as follows

- **Microcontroller:** This is the heart of our project. A microcontroller is an integration of a microprocessor, ROM, and RAM. It also has I/O ports, all in one chip. This is thus directly compatible to input and output devices and does not need any interfacing devices like in the case of connecting a microprocessor to an I/O device. Selecting the microcontroller that suits our purpose from the myriad of the available ones was a difficult task. The microcontroller we have used here is 89c51 by Atmel. The features of the microcontroller are
  - 40 pin controller
  - 4 I/O ports of 8 pins each
  - 4k bytes of flash memory
  - 128 byte internal RAM
  - Two 16 bit timers
  - The clock speed can reach up to 24MHz.

- **EEPROM:** EEPROM stands for Electrically Erasable Programmable Read Only Memory. This is a kind of ROM where we can erase the contents and reprogram. The difference in this and the flash memory of the microcontroller is that here we can be specific on the bits we erase and with a flash memory we will have to erase the entire ROM and write on to it afresh. The EEPROM also has varieties with it like hardware addressable and software addressable. The one that suits our purpose is the software addressable EEPROM. In addition we have to take care checking if the EEPROM is a serial or parallel one. We have selected a serial EEPROM by Atmel, the AT24C32. This is the card we interface with the microcontroller. It has the details of the amount of electricity that can be consumed before cut off. There are two cards of which one acts as the primary and the other as a secondary back up. The features of this EEPORM are
  - 8 pin IC

- 32K bytes of memory
- 2 Wire serial interface
- Write protect pin for Hardware Data Protection.

- **LCD:** LCD stands for Liquid Crystal Display. This is an output device with a limited viewing angle. The choice of LCD as an output device was because of its cost, ease of use and is better with alphabets when compared with a 7-segment LED display. We have so many kinds of LCD's today and our application requires a LCD with 2 lines and 16 characters per line. This gets data from the microcontroller and displays the same. It has 8 data lines, 3 control lines, a supply voltage Vcc (+5 V) and a GND. This makes the whole device user friendly by showing the balance left in the card. This also shows the card that is currently being used.

- **Sensor:** A sensor is a device that understands interruptions and gives an output accordingly. There are various kinds of sensors and the one that suits our application is an IR sensor. This sensor has two parts, the transmitter and the receiver. The transmitter has two inputs and send out infra red frequencies. The inputs are the supply voltage Vcc (+5 V) and the ground GND. The receiver is one that senses these infra red frequencies. It has two pins, one is the ground GND and the other is the output pin. It has a pulsed output, i.e., the pin is always in one state and when the receiver detects the IR frequency it gives a spike of a higher voltage. This is the device that is interfaced with the existing analog meter to detect the number of rotations that it makes. The output of the sensor is given to the microcontroller that keeps track of the number of rotations and decrements the card accordingly.

- **Buzzer:** A buzzer is a device that produces a sound in the audible frequency. It has two inputs the supply voltage Vcc (+5 V) and the ground GND. A buzzer is sounded when there is very little balance (10 units) left in the card that is currently being used. This makes the user alert and gets him to replace the card in

the other slot with a new one. Once this card goes dry the microcontroller automatically shifts to the other card. Thus the buzzer reminds the user of replacing the card in the secondary slot and ensures that the supply is not cut off.

- **Relay:** A relay is like a switch that is in one of the states either on or off. It has 5 pins, one input from the microcontroller, one is the ground GND, one is the supply voltage, and the other two are the outputs. The outputs are the normally closed and normally open states of the relay. When in the normally closed state the relay conducts and the supply is got through this pin. When the relay is in normally open state the output is thus not got and no supply is possible. A relay circuit provided thus cuts off the power supply when both the cards go out of balance. Thus this makes sure no power is drawn free of charge.

The entire programming was simulated in an environment using the software RIDE by Raisonance and once the program was successfully compiled it was transferred onto the controller. After this transfer was done, the PCB was fabricated and tested. A suitable slab for the metering has been implemented. This has been done because it is difficult to implement the existing metering slab followed owing to loads, which are not practically feasible in a simulated environment.

The entire system has been so designed that at any point the user may get an idea about his current usage and the remaining left in the card. This is accomplished by making an LED glow right next to the EEPROM that is currently in use. In addition to this the LCD display also has the same written so as to avoid confusion in the event that the LED stops working.

4. PROPOSED APPROACH TO THE PROBLEM

The right approach towards a problem is very important. An egregious approach could prove to be drastic and can cause a large number of inconveniences. Taking this into account a large number of approaches were thought of for solving our problem taking into account factors such as complexity, cost, and effectiveness and so on...

The present day electricity metering is done manually with the help of an energy meter. Changes in this would cause a large number of overheads. So in our approach we have decided to supplement it with other components. The sensor senses the rotations of the disc in the energy meter and the pulsed output is given to the microcontroller. Every time a pulse is obtained, the micro controller takes the value stored in the EEPROM that is currently in use and decrements it by one. This denotes the consumption of a unit (Kilowatt) and this system of metering has been done for simulation purposes.

Here we have decided to use 2 cards (EEPROM). The reason being, once the value of the EEPROM becomes zero, the total power supply will be cut off. This would be very inconvenient and not practical for the user. So here we have given a provision where we make use of 2 EEPROM's. Once the value in one card goes to zero control is transferred and the value of the second card is verified. If this value is not zero power is supplied. If the values in both the cards are zero total power supply is cut off. As a precaution we have made use of a buzzer where in a alarm is sounded once the value in the card goes below a threshold value, to intimate the user that it is time to replace the card.

On commercial implementation, a slight modification to the logic would be required; where in the completion of a rotation causes the increment of a counter which is checked against another value. For ex: If 200 rotations of the disk indicate the consumption of 1 kilowatt, then each time the disk completes a rotation a counter

value is incremented and this value is checked with the value 200. If both these values are equal then one unit is decremented from the EEPROM that is currently in use. This decremented value is now written back on to the EEPROM. The count that is maintained in the EEPROM is explicitly made to zero. This process is repeated till the value in the cards run dry. The balance that is read from the EEPROM is regularly updated on the LCD screen. The display on the LCD tells us about the card number that is currently in use and the balance on that card in terms of units.

A relay circuit has been designed to cut off the power supply in the event of the cards running dry. The entire project has been shown on a single board but for practical purposes the relay circuit will be a part of the electricity post from which energy is distributed to the consumer. This is done to avoid power theft. This project could be propped with a few encryption algorithms and anti- piracy strategies which have been discussed in the subsequent chapters.

The design has been done bearing two factors in mind:

1) The cost efficiency and
2) Simplicity.

This means that while the project will be cost efficient it will also not involve any major alterations to the existing system thereby increasing the overall viability of the project.

5. DETAILS OF THE DESIGN

The project was divided into four main modules and each module was integrated finally to get the desired output. Each module was decided based on the research and the study required. The modules were designed such that finally all that was required was a "plug and play" format. At each stage we saw that the tiniest of components were studied and their working completely understood. These modules are discussed in this section and the final integration is also discussed in brief.

MODULE 1

Identification of the Components

The first phase comprised of identifying the various components required for the project. Since this project has not been implemented anywhere thus far, we had to analyze, design the circuit and then purchase the components. The field of electronics has a lot of competition and keeping in mind the economic viability in a country like India, we had to chose the components with utmost care. After a lot of research the following components were chosen.

- **Micro Controller:** There were two controllers that we had to choose from. These were products of Atmel and Pic who are renowned the world over for their electronic products. Pic is a RISC processor with 13 I/O pins and is priced higher than Atmel. The Atmel controller, which we have chosen, is a 40 PIN CISC processor. It has 32 I/O pins through 4 ports of 8 each. The most important feature is that in an ATMEL microcontroller each port has the bit set reset facility which enables us to have 32 independent port lines. It is priced much lower than the Pic controllers. The ATMEL microcontroller has several other features like power down mode and 3-level memory lock which would be useful in enhancing the security of the final product.

- **Sensor:** An IR sensor was chosen because of its ability to send pulsed output for every detected rotation. The size of the sensor is small and so interfacing it to the existing meter is trivial. Another consideration was a proximity sensor that requires a flawless surface with absolutely nil wobbling. The IR sensor does not pose these constraints and hence was chosen to fit the purpose. The construction of the IR Sensor is simple and has been elucidated in the later chapters.

- **EEPROM:** The EEPROM chosen provides 32K bytes of memory. This is a serial EEPROM with a common 2-wire bus. The device is optimized for use in many industrial and commercial applications where low power and low voltage operations are essential. The two EEPROMS used in the project are used as a surrogate to the surface mounted IC chips. There are 2 kinds of EEPROM's that are available in the market. One is the software programmable EEPROM and the other is the hardware programmable EEPROM. We have chosen the software programmable EEPROM to economize on the number of port lines required.

- **LCD Display:** This is from where the user understands his usage of electricity. It is a 2-line display with 16 characters per line. This has limited viewing angle. It has 8 data pins, 3 control pins, a supply voltage (+5 V) and a ground. This is one of the most important components in our project because it is this display that informs the user of his consumption rate and the various other details.

- **Buzzer:** This is used to alarm the user of the card going dry. It uses only one of the 32 I/O pins of the microcontroller. It uses simple binary logic where in a logical1 will make the buzzer ring else it wont.

- **Relay:** This is used to cut off the power supply when both the cards go dry. This also uses only one I/O pin of the microcontroller. The construction of the relay is simple. Once the relay circuit is driven a coil gets charged and so the switch is pulled towards it resulting in an open non-conducting circuit.

Once the various components were purchased their data sheets were carefully studied in order to get an in-depth idea of their usage.

MODULE 2

Programming the Micro Controller

This phase involved programming the Micro Controller. The logic involved is obtaining the value from EEPROM and performing manipulations based on the number of rotations made by the energy meter. The manipulated reading is then re-written into the EEPROM. The display has to be continuously shown on the LCD display for the convenience of the user. The microcontroller should also drive the buzzer and relay at the appropriate time. The entire programming was done in Kiel Mju Vision. This was later transferred to the Micro Controller. The values are preset into the EEPROMS using a keypad. Once the program was written and compiled it was converted into hexadecimal code and then written into the micro controller.

MODULE 3

Designing and Fabrication of the PCB

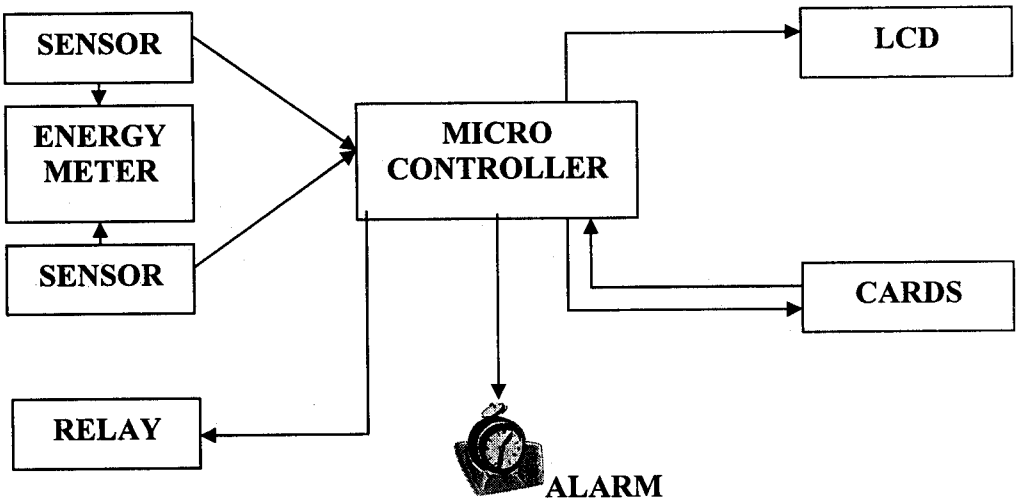
The design of the PCB involves a lot of sub divisions. First the 230v AC input must be stepped down to a 5v DC supply. A transformer is used for the voltage step down. This is then passed for an AC to DC conversion. The obtained DC voltage is used for all the components. Various active and passive devices are connected so that a surge or fall in voltage does not affect the performance of the PCB. The micro controller, EEPROM's and LCD are integrated onto the board in such a way that they work in tandem with the active and passive components to achieve their prescribed rating. The entire project was first tried on a bread board and when it was found to be successful we implemented the same on a printed circuit board.

MODULE 4

Creating a desktop model

An energy meter that is commercially being used was purchased and to a small hole was drilled on the disk so that the sensor detects the completion of a rotation. The output wire from the sensor is given as input to the micro controller. For the purpose of simulation we connect few loads and demonstrate.

DATA FLOW DIAGRAM



The diagram above shows how the various modules are integrated and the data flow between the components. The two sensors shown above and below the energy meter show the transmitter and receiver

6. IMPLEMENATION DETAILS

The details of the various components used are enlisted below.

- **ATMEL 89C51**

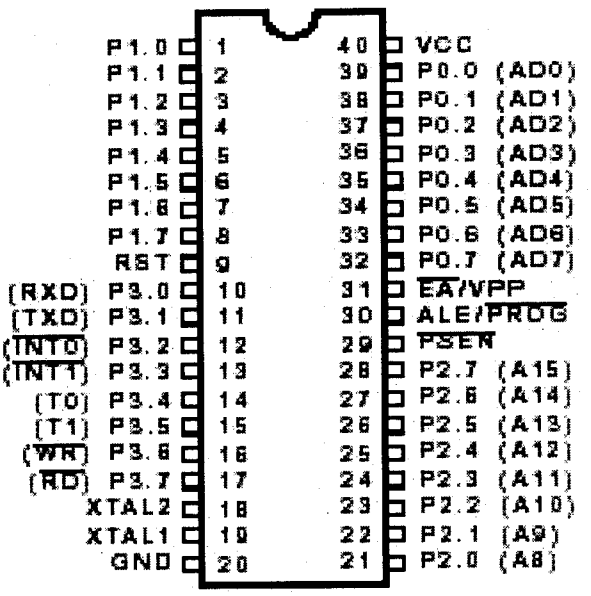
FEATURES

- 4K Bytes of In-System Reprogrammable Flash Memory
- Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

DESCRIPTION

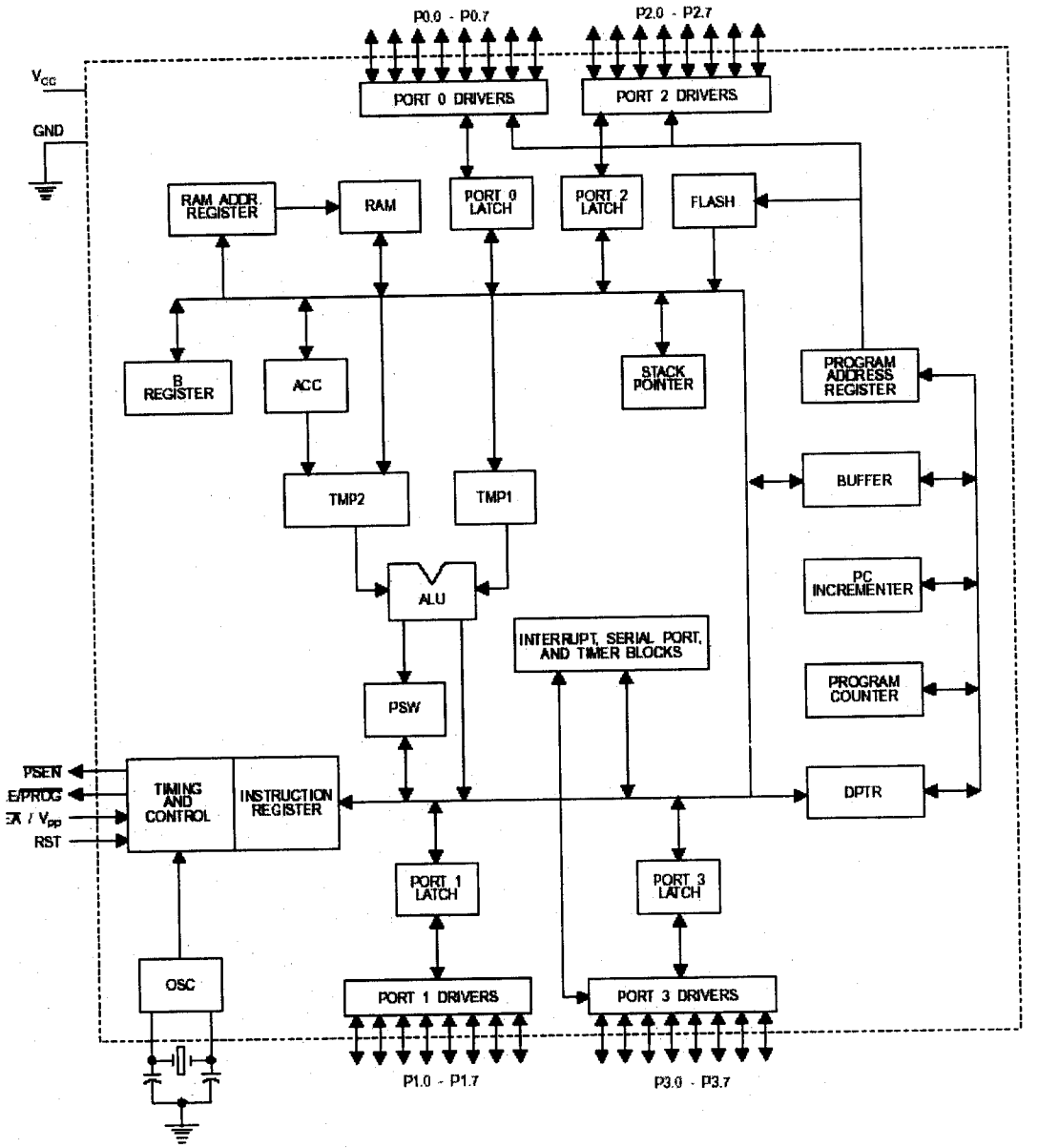
The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4Kbytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer that provides a highly flexible and cost effective solution to many embedded control applications.

PIN CONFIGURATION



PLCC

BLOCK DIAGRAM



PIN DESCRIPTION

VCC

Supply voltage.

GND

Ground.

PORT 0

Port 0 is an 8-bit open drain bi-directional I/O port. As an output port each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high impedance inputs. Port 0 may also be configured to be the multiplexed low order address/data bus during accesses to external program and data memory. In this mode P0 has internal pull-ups. Port 0 also receives the code bytes during Flash programming, and outputs the code bytes during program verification. External pull-ups are required during program verification.

PORT 1

Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. Port 1 also receives the low-order address bytes during Flash programming and verification.

PORT 2

Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins they are

pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to

External data memory that uses 16-bit addresses (MOVX @ DPTR). In this application it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

PORT 3

Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL) because of the pull-ups. Port 3 also serves the functions of various special features of the AT89C51i.e, Port 3 also receives some control signals for Flash programming and verification. Port 3 also serves the functions of various special features of the AT89C51 as listed below:

PORT PIN ALTERNATE FUNCTIONS

P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/PROG

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming. In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory. If desired, setting bit 0 of SFR location 8EH can disable ALE operation. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{\text{PSEN}}$

Program Store Enable is the read strobe to external program memory. When the AT89C51 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

EA/VPP

External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions. This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming, for parts that require 12-volt VPP.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be Observed.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all the onchip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by hardware reset. It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Status of External Pins During Idle and Power Down Modes

Mode	Program Memory	ALE	$\overline{\text{PSEN}}$	Port 0	port 1	port 2	port 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power down	Internal	0	0	Data	Data	Data	Data
Power down	External	0	0	Float	Data	Data	Data

POWER DOWN MODE

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before VCC is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

PROGRAM MEMORY LOCK BITS

On the chip are three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below: When lock bit 1 is programmed, the logic level at the EA pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of EA be in agreement with the current logic level at that pin in order for the device to function properly.

LOCK BIT PROTECTION MODES

Program Lock Bits				Protection Type
LB1	LB2	LB3	LB4	
1	U	U	U	No program lock features.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the Flash is disabled.
3	P	P	U	Same as mode 2, also verify is disabled.
4	P	P	P	Same as mode 3, also external execution is disabled.

PROGRAMMING THE FLASH

The AT89C51 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage (VCC) program enable signal. The low voltage programming mode provides a convenient way to program the AT89C51 inside the user's system, while the high-voltage programming mode is compatible with conventional third party Flash or EPROM programmers. The AT89C51 is shipped with either the high-voltage or low-voltage programming mode enabled. The AT89C51 code memory array is programmed byte-by byte in either programming mode. To program any nonblank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.

PROGRAMMING ALGORITHM

Before programming the AT89C51, the address, data and control signals should be set up according to the Flash programming mode table and Figures 3 and 4. To program the AT89C51, take the following steps.

1. Input the desired memory location on the address lines.

2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise EA/VPP to 12V for the high-voltage programming mode.
5. Pulse ALE/PROG once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

DATA POLLING

The AT89C51 features Data Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on PO.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. Data Polling may begin any time after a write cycle has been initiated.

READY/BUSY

The progress of byte programming can also be monitored by the RDY/BSY output signal. P3.4 is pulled low after ALE goes high during programming to indicate BUSY. P3.4 is pulled high again when programming is done to indicate READY.

PROGRAM VERIFY

If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

CHIP ERASE

The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/PROG low for 10 ms. The code array is written with all "1"s. The chip erase operation must be executed before the code memory can be re-programmed.

READING THE SIGNATURE BYTES

The signature bytes are read by the same procedure as a normal verification of locations 030H, 031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel

(031H) = 51H indicates 89C51

(032H) = FFH indicates 12V programming

(032H) = 05H indicates 5V programming.

PROGRAMMING INTERFACE

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self timed and once initiated, will automatically time itself to completion. All major programming vendors offer worldwide support for
the Atmel micro controller series. Please contact your local programming vendor for the appropriate software revision.

ABSOLUTE MAXIMUM RATINGS

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin-	
with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	15.0 mA

- **AT24C32**

FEATURES

- 8 pin IC
- Low-Voltage Operation
- Internally Organized 4096 x 8
- 2-Wire Serial Interface
- 100 kHz (1.8V, 2.5V, 2.7V) and 400 kHz (5V) Clock Rate
- Write Protect Pin for Hardware Data Protection
- 32-Byte Page Write Mode (Partial Page Writes Allowed)
- Self-Timed Write Cycle (10 ms max)
- High Reliability

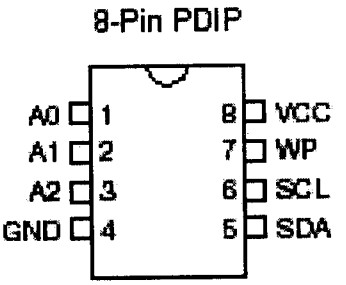
DESCRIPTION

The AT24C32 provides 32,768 bits of serial electrically erasable and programmable read only memory (EEPROM) organized as 4096 words of 8 bits each. The device's cascadable feature allows up to 8 devices to share a common 2-wire bus. The device is optimized for use in many industrial and commercial applications where low power and low voltage operation are essential. The AT24C32 is accessed via a 2-wire serial interface. In addition, the entire family is available in 2.7V (2.7V to 5.5V) and 1.8V (1.8V to 5.5V) versions.

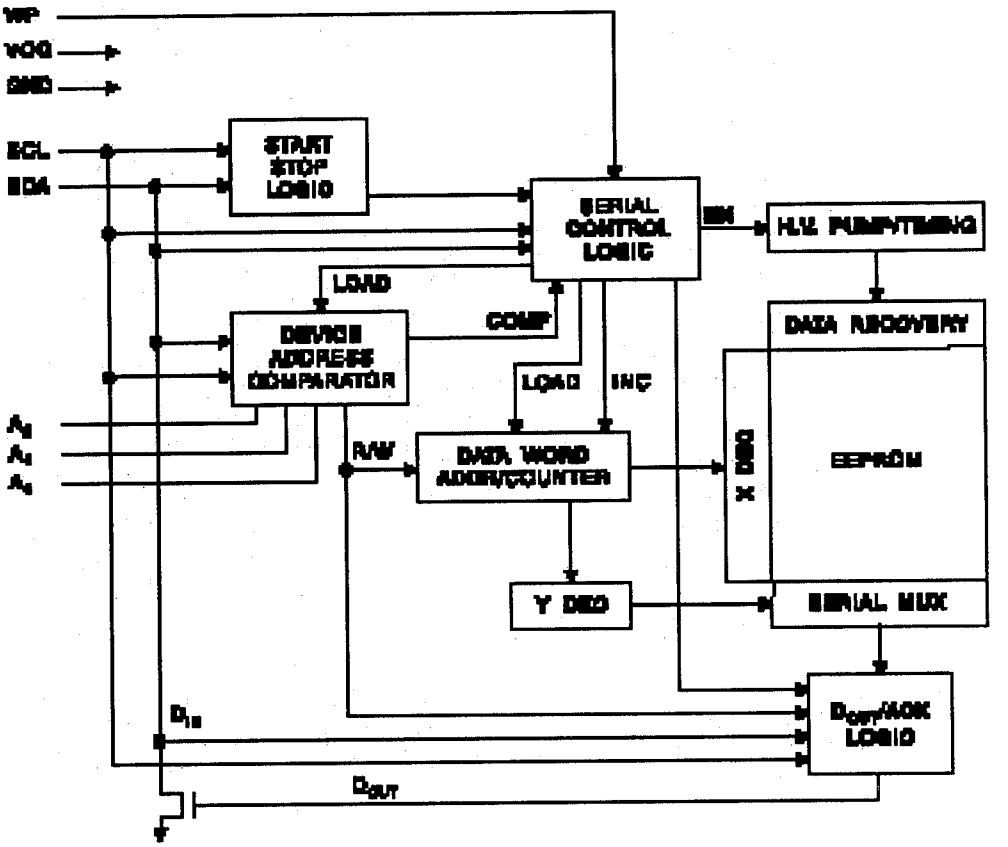
PIN CONFIGURATION

Pin Name	Function
A0 – A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect

PIN DIAGRAM



BLOCK DIAGRAM



PIN DESCRIPTION

Serial Clock (SCL)

The SCL input is used to positive edge clock data into each EEPROM device and negative edge clock data out of each device.

Serial Data (SDA)

The SDA pin is bidirectional for serial data transfer. This pin is open-drain driven and may be wire-ORed with any number of other open-drain or open collector devices.

Device/Page Addresses (A2, A1, A0)

The A2, A1 and A0 pins are device address inputs that are hard wired or left not connected for hardware compatibility with AT24C16. When the pins are hardwired, as many as eight 32K/64K devices may be addressed on a single bus system (device addressing is discussed in detail under the Device Addressing section). When the pins are not hardwired, the default A2, A1, and A0 are zero.

Write Protect (WP)

The write protect input, when tied to GND, allows normal write operations. When WP is tied high to VCC, all write operations to the upper quandrant (8/16K bits) of memory are inhibited. If left unconnected, WP is internally pulled down to GND.

Memory Organization

The 32K/64K is internally organized as 256 pages of 32 bytes each. Random word addressing requires a 12/13 bit data word address.

DEVICE OPERATION

Clock and Data Transitions

The SDA pin is normally pulled high with an external device. Data on the SDA pin may change only during SCL low time periods (refer to Data Validity timing diagram). Data changes during SCL high periods will indicate a start or stop condition as defined below.

Start Condition

A high-to-low transition of SDA with SCL high is a start condition which must precede any other command (refer to Start and Stop Definition timing diagram).

Stop Condition

A low-to-high transition of SDA with SCL high is a stop condition. After a read sequence, the stop command will place the EEPROM in a standby power mode (refer to Start and Stop Definition timing diagram).

Acknowledge

All addresses and data words are serially transmitted to and from the EEPROM in 8-bit words. The EEPROM sends a zero during the ninth clock cycle to acknowledge that it has received each word.

Standby Mode

The AT24C32/64 features a low power standby mode which is enabled: a) upon power-up and b) after the receipt of the STOP bit and the completion of any internal operations.

Memory Reset

After an interruption in protocol, power loss or system reset, any 2- wire part can be reset by following these steps: (a) Clock up to 9 cycles, (b) look for SDA high in each cycle while SCL is high and then (c) create a start condition as SDA is high.

DEVICE ADDRESSING

The 32K EEPROM requires an 8-bit device address word following a start condition to enable the chip for a read or write operation. The device address word consists of a mandatory one, zero sequence for the first four most significant bits. This is common to all 2-wire EEPROM devices. The 32K uses the three device address bits A2, A1, A0 to allow as many as eight devices on the same bus. These bits must compare to their corresponding hardwired input pins. The A2, A1, and A0 pins use an internal proprietary circuit that biases them to a logic low condition if the pins are allowed to float. The eighth bit of the device address is the read/write operation select bit. A read operation is initiated if this bit is high and a write operation is initiated if this bit is low. Upon a compare of the device address, the EEPROM will output a zero. If compare is not made, the device will return to standby state.

Noise Protection

Special internal circuitry placed on the SDA and SCL pins prevent small noise spikes from activating the device. A low-VCC detector resets the device to prevent data corruption in a noisy environment.

Data Security

The AT24C32 has a hardware data protection scheme that allows the user to write protect the upper quadrant of memory when the WP pin is at VCC.

WRITE OPERATIONS

Byte Write

A write operation requires two 8-bit data word addresses following the device address word and acknowledgment. Upon receipt of this address, the EEPROM will again respond with a zero and then clock in the first 8-bit data word. Following receipt of

the 8-bit data word, the EEPROM will output a zero and the addressing device, such as a microcontroller, must terminate the write sequence with a stop condition. At this time the EEPROM enters an internally-timed write cycle, t_{WR}, to the nonvolatile memory. All inputs are disabled during this write cycle and the EEPROM will not respond until the write is complete.

Page Write

The 32K EEPROM is capable of 32-byte page writes. A page write is initiated the same way as a byte write, but the microcontroller does not send a stop condition after the first data word is clocked in. Instead, after the EEPROM acknowledges receipt of the first data word, the microcontroller can transmit up to 31 more data words. The EEPROM will respond with a zero after each data word received. The microcontroller must terminate the page write sequence with a stop condition. The data word address lower 5 bits are internally incremented following the receipt of each data word. The higher data word address bits are not incremented, retaining the memory page row location. When the word address, internally generated, reaches the page boundary, the following byte is placed at the beginning of the same page. If more than 32 data words are transmitted to the EEPROM, the data word address will "roll over" and previous data will be overwritten.

Acknowledge Polling

Once the internally timed write cycle has started and the EEPROM inputs are disabled, acknowledge polling can be initiated. This involves sending a start condition followed by the device address word. The read/write bit is representative of the operation desired. Only if the internal write cycle has completed will the EEPROM respond with a zero, allowing the read or write sequence to continue.

READ OPERATIONS

Read operations are initiated the same way as write operations with the exception that the read/write select bit in the device address word is set to one. There are three read operations: current address read, random address read and sequential read.

Current Address Read

The internal data word address counter maintains the last address accessed during the last read or write operation, incremented by one. This address stays valid between operations as long as the chip power is maintained. The address "roll over" during read is from the last byte of the last memory page, to the first byte of the first page. The address "roll over" during write is from the last byte of the current page to the first byte of the same page. Once the device address with the read/write select bit set to one is clocked in and acknowledged by the EEPROM, the current address data word is serially clocked out. The microcontroller does not respond with an input zero but does generate a following stop condition

Random Read

A random read requires a "dummy" byte write sequence to load in the data word address. Once the device address word and data word address are clocked in and Acknowledged by the EEPROM, the microcontroller must generate another start condition. The microcontroller now initiates a current address read by sending a device address with the read/write select bit high. The EEPROM acknowledges the device address and serially clocks out the data word. The microcontroller does not respond with a zero but does generate a following stop condition.

Sequential Read

Sequential reads are initiated by either a current address read or a random address read. After the microcontroller receives a data word, it responds with an acknowledge. As long as the EEPROM receives an acknowledge, it will continue to increment the data word address and serially clock out sequential data words. When the memory address limit is reached, the data word address will "roll over" and the sequential read will continue. The sequential read operation is terminated when the microcontroller does not respond with a zero but does generate a following stop condition.

- **IR SENSOR**

A sensor is a device that senses interruptions. The purpose of a sensor here is to detect one full rotation of the disk in the energy meter. Every time the disk makes a complete rotation a pulse is sent out. It has two parts the transmitter and the receiver. The transmitter is so designed that it sends out IR frequencies. The input to the transmitter is a supply voltage of +5 volts and a ground supply. This supply drives the transmitter and it continuously sends out the corresponding frequency. The receiver is so designed that it picks up only the frequencies generated by the transmitter. The disk in between the transmitter and the receiver does not let the receiver sense the frequency that the transmitter sends. A hole made in the disk lets the frequencies go pass and hit the receiver. This triggers the receiver and it in turn sends out a pulse thus indicating the completion of a rotation. The two pins of the receiver are the ground and the pulse output pin.

- **LCD**

16 character * 2 line LCD's have 14 connections for their display either on top or bottom of the display. For orientation, the numeral one is next to connection one. If the LCD has a back light it will be powered by pins 15 and 16 or by two connections to the side of the display.

PIN FUNCTIONS

No.	Pin No.	Name	Function
1	17	V_{SS}	GND.
2	18	V_{DD}	Power Supply Voltage +5 volts.
3	19	V_{LC}	Liquid Crystal Dirving Voltage.
4	20	RS	L: Instruction Code Input. H: Data Input.
5	21	R/W	L: Data write from MPU to LCM. H: Data read from LCM to MPU.
6	22	E	Enable.
7	23	DB0	Data Bus Line.
8	24	DB1	Data Bus Line.
9	25	DB2	Data Bus Line.
10	26	DB3	Data Bus Line.
11	27	DB4	Data Bus Line.
12	28	DB5	Data Bus Line.
13	29	DB6	Data Bus Line.
14	30	DB7	Data Bus Line.
15	31	V_L	Anode.
16	32	V_C	Cathode.

FEATURES

- Display Format - 16 Characters * 2 Lines
- Display type - TN
- Display mode - Positive
- Viewing Direction - 6 O'clock
- Driving method - 1/16 Duty
- Interface with a 8-bit or a 4 bit MPU is available
- 192 kinds of alphabets, numerals, symbols and special characters can be displayed by built in character generator(RAM)

Various functions of instruction are available by programming:

- Clear display
- Cursor at home
- On/off cursor
- Blink character
- Shift display
- Shift cursor
- Read/write display

PHYSICAL DATA

Module size	-	80.0W * 36.0H * 13.0T mm
Min. view area	-	66.6W * 13.8H mm
Character Construction	-	5 * 7 dots
Character size	-	2.95W * 3.8H mm
Character pitch	-	3.65 mm
Dot size	-	0.55W * 0.5H mm
Weight	-	30 g (approx.)

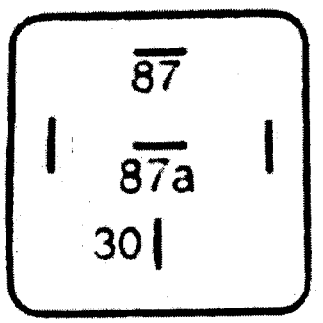
- **RELAY**

Relays are one of the oldest, simplest, and yet, easiest and most useful devices. Before the advent of the mass produced transistor, computers were made from either relays or vacuum tubes, or both.

A relay, quite simply, is a small machine consisting of an electromagnet (coil), a switch, and a spring. The spring holds the switch in one position, until a current is passed through the coil. The coil generates a magnetic field which moves the switch. It's that simple. You can use a very small amount of current to activate a relay, and the switch can often handle a lot of current.

The relay we are going to look at is the Bosch 5 pin relay. Bosch is a German manufacturing conglomerate (who also happen to own Bosch Telekom and Blaupunkt), but they are not the only manufacturer of this relay. There are several other companies such as Siemens and Potter & Brumfield. The Bosch 5 pin relay is the most widely used and versatile relay, and it can handle up to 30 amps, which is more than suitable for most applications.

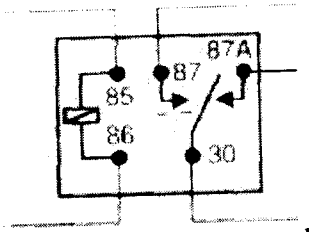
Looking at the diagram the below we see the pinout of the relay. Note that each pin is numbered, 85, 86, 87, 87a, and 30. The 30 pin is set perpendicular to the other pins to let you know where each pin is.



85 and 86 are the coil pins. Normally, it doesn't matter which way you pass the current, because if you hook it up backwards, the coil will still activate the relay. However, relays sometimes have an odd tendency to turn themselves back on briefly. To counter this, a diode (a one way switch) is placed between 85 and 86. This is referred to as a tamping diode. A diode wall has a very high resistance in one direction, and a very low resistance in the opposite direction. When a tamping diode is used, it is important that you hook the coil up according to polarity. If a tamping diode is used, and you hook it up backwards, you will essentially be shorting a wire out, which sucks, because you can and will burn something up.

30, 87, and 87a are the other three pins. 87 and 87a are the two contacts to which 30 will connect. If the coil is not activated, 30 will always be connected to 87a. Think of that pin as "87, always connected". When current is applied to the coil, 30 is

connected to 87. 87 and 87a are never connected to each other. Here, polarity does not ever matter. You can connect 30 up to positive or negative, and that is what you will get out of 87 or 87a. Refer to the picture below, and perhaps it will make the relay a tad simpler.



As you can see, the coil is in no way connected to the switch part of the relay. This can allow you to completely isolate one circuit from another. You can even use a separate power supply to control the relay.

Now let's talk about applications. One common use of a relay is for multiple amplifiers. Normally, if you run any more than three audio components off your decks remote output, it is recommended that you use a relay to provide higher currents. Also, if you wanted to power something like a fan, or neon lighting, you need a relay to keep up with the higher demands these devices place on your remote. Take a minute and guess if you can see how it is done. All you have to do is connect 85 to your decks remote output wire, and 86 to ground. Then, connect 30 to a fused 12 volts source, and 87 to your outgoing remote line. Remember that the relay can carry up to 30 amps, but if your wire can only handle 10 amps, you should use a 10 amp fuse. 87a is not used, but I recommend taping up that pin, or pulling the wire out of the base, so that it does not short out.

Testing

7. TESTING

In a domestic environment, there are several appliances running at the same time. These may vary from electric geysers to simple electric bulbs and according to the load that is applied the consumption increases and thereby the speed of rotation of the disc is also increased. In a simulated environment it is not possible to connect many loads for the purpose of demonstration. Instead we fix our own parameters similar to the original parameters and demonstrate the same. In our project we have made use of three electric bulbs of 200 watts each that will be used as the load.

In a commercially available energy meter 200 rotations of the disc will signify the consumption of one kilowatt-hour. That means, if a load of 1 kilowatt runs for one hour then the meter will rotate 200 times in that hour. Since it is difficult to connect a 1 kw load for demonstration purposes and because it is time consuming we use a similar rule. With 3 x 200w bulbs, the consumption will be 600 watts which is .6 kw. Hence we will get only 60% of the rotations made by the disc if 1 kw was connected. 60% of 200 is 120 rotations per hour and on further simplification we get 2 rotations per minute. Hence for the purpose of demonstration we have kept 2 rotations a minute to signify the consumption of 1 unit. This has been done only for our convenience and is no benchmark to go by.

The entire system was tested under various conditions of load and it was found to function normally under all the loads. The results were highly satisfying and it presented a completely automised system with cost efficiency as well. This was the core aim of the project and it has been successfully adhered to. Once it has been completely tested along with encryption algorithms and other small modifications (if required by the market), this project can be made into a commercial product and can be implemented for the benefit of the people.

8. CONCLUSION AND FUTURE OUTLOOK

The main aim of our project was to ease the existing complexity in the metering and bill processing of electricity. It is quite impossible to bring a complete change to the existing system and so the best alternative would be to alter the existing system such that it catered to all strata of society with a special inclination towards the economically backward.

The field of embedded engineering has made rapid strides and is today an integral part of the smallest home appliance. Though this field has been around for well over a decade, it is only now that it has sprung into significance. The microcontroller that we have used (manufactured by the company ATMEL) is used to make simple modifications to the existing system.

The experiment of using prepaid cards has been extremely successful with cellular phone subscribers. Today people prefer smart cards for everything including buying fuel which is why we have petro cards today. The success of smart cards could spread to the consumption of electricity as well because such cards are a major relief to those with little requirement and to those who need temporary connection.

The system that we have designed has been done with the intention to make the end-user feel minimum strain. In the event of a complete drainage of one card, the microcontroller automatically shifts to another card that is kept as standby. This is done to give the user enough time to replace his card without interrupting power supply. It is only if card 2 drains out completely, does the relay circuit in the device cut off supply from the mains.

A few enhancements that could be done in future according to the then prevailing market are

- A display of the balance in terms of currency in addition to the balance shown in terms of units will give the user a better idea of usage.

- By incorporating an additional timer and by providing scrolling keys the user could view his usage over periods of time. He could view his usage per day, per week or per month.

- A complicated encryption algorithm that would use the power line to establish contact with the EB to verify the authenticity of the card could be done.

9. REFERENCES

1. John Catsoulis, "Designing Embedded hardware", University of Delaware, Dec 1998.

2. Michael Barr, "Programming embedded systems in C and C++", TMH 1990

3. Baker and Taylor, "Systems Modelling and Requirements", Dorset Publishing.

4. You Won, "analysis and design of complex reactive embedded systems", York house Publishing.

5. "Cybernetics and Systems", A Taylor & Francis Journal

6. Niall D Murphy, "Reviews of embedded systems in C", ACCU Mar 2000

7. www.embedded.com

8. www.tmssales.com

9. www.microsoft.com/windows/embedded.

10. www.webopedia.com

PROGRAM CODE

```
# include "reg51.h"
# include <stdio.h>
# include "dra_lcd.c"
# include  "eeprom24.c"

#define TV0  0xfc  //TIMER O INTERRUPT CALL
#define TVL0 0x17

   void fill_blank();
   void put_mass_msg2();
   void put_demand_msg2();
   void put_disp1_msg();
   void motor_on_sw();
   void put_card1_msg();
   void put_card2_msg();
   void key();
   void skey();
   void set();

   void move2();
   void maxi_dema();

   void delay();
   void delay1();

   void put_dispname_msg();
   void put_rech_msg();
   void put_id_msg();
   void put_mass_msg2();
   void put_mass_msg3();
   void put_mass_msg4();
   void put_mass_msg5();
   void put_mass_msg6();
   void put_prg1_msg();
   void put_prg_msg();
   void time_factor();
   unsigned int  bcd_bin();
   void  inc1(unsigned char z1);
   void binb_bcd6(unsigned int g1);
   void sepr();
   void call_unit();
   void sepr2();
   void call_unit2();
```

```
unsigned int bcd_bin6();

        sbit card=P12;//check
        sbit card2=P13;

        sbit buzzer=P15;
        sbit motor_sw=P14;
        sbit mov_sw=P34;
        sbit set_sw=P33;
        sbit inc_sw=P35;
        sbit relay=P37;

        bit menu_flag;
        bit set_flag;
        bit flag_fun_over;
        bit inc_flag;
        bit motor_flag;
        bit first_flag;
        bit second_flag;
        bit card_two_flag;
        bit buzzer_flag;


unsigned int dmaxidema,maxidema,lmaxidema;
unsigned char
key_buffer,i9,pass[5],s1,flagt,flagr,rec_reg,y,digit1,digit2,digit3,digit4,l_units1,h_uni
ts1;
unsigned int idata misec,g2,y1,y2,y3,y4,y5,y6,y7,y8;
void main()
        {
        card_two_flag=0;
                motor_flag=0;
                first_flag=0;
                second_flag=0;
                buzzer_flag=0;
            relay=1;
                buzzer=0;

          init_lcd();
       put_mass_msg();
       put_lcd_download_1line();
       fill_blank();
       put_mass_msg1();
       put_lcd_download_2line();
       skey();
```

```
dly1sec();
 skey();
dly1sec();
dly1sec();
 skey();
dly1sec();
dly1sec();

 fill_blank();
 put_lcd_download_1line();
  put_lcd_download_2line();
 put_mass_msg4();
 put_lcd_download_1line();

 dly1sec();
 dly1sec();
 dly1sec();
 dly1sec();
 dly1sec();
 dly1sec();

      do{;} while(card==1);

              ADDR_LO=0;
              F_ADDR=0xa0;
  RF_ADDR=0xa1;
 E_DATA=2;
 read_eeprom();
 maxidema=e_reg[2]<<8;
 maxidema=maxidema|e_reg[1];
 if(maxidema==0)
   {
     card_two_flag=1;
   }
  else
    card_two_flag=0;

       do{;} while(card2==1);
         ADDR_LO=0;
               F_ADDR=0xa4;
   RF_ADDR=0xa5;
  E_DATA=2;
  read_eeprom();
  lmaxidema=e_reg[2]<<8;
  lmaxidema=lmaxidema|e_reg[1];
```

```
  if(card_two_flag==0)
   {
     put_card1_msg();
     put_lcd_download_1line();
     sepr();
     call_unit();
    }
   else
     {
      put_card2_msg();
      put_lcd_download_1line();
      sepr2();
      call_unit2();
     }

  TCON=0x10;//
  TMOD=0x01;
  TL0=TVL0;//  interrupt call for every 1 millisecond (RTC)
  TH0=TV0;
  //TR0=0;
   TR0=1;
  IE=0x82;
  EA=1;

 while(1)
 {
  motor_on_sw();

if(motor_flag==1)
   {

   if(card_two_flag==0)
    {
      if(maxidema==0)
        maxidema=0;
      else
       maxidema--;

        if(maxidema==10)
       {
        key_buffer++;
        if(key_buffer==1)
         {
         buzzer_flag=1;
          }
        if(key_buffer==5)
```

```
        key_buffer=2;
    }

 sepr();
 e_reg[1]=l_units1;
 e_reg[2]=h_units1;
 ADDR_LO=0x00;
 E_DATA=2;
 F_ADDR=0xa0;
 RF_ADDR=0xa1;
 EA=0;
write_eeprom();
  EA=1;

     motor_flag=0;
 call_unit();
  if(maxidema==0)
   { card_two_flag=1;
    key_buffer=0;
   }
 }
                                  // fill_blank();
 else
   {
   if(lmaxidema==0)
      {
      lmaxidema=0;
       card_two_flag=0;
      }
     else
         lmaxidema--;

   if(lmaxidema<=10)
      {
        key_buffer++;
       if(key_buffer==1)
        {
        buzzer_flag=1;
         }
        if(key_buffer==5)
           key_buffer=2;

      }

    sepr2();
    e_reg[1]=l_units1;
```

```
e_reg[2]=h_units1;
ADDR_LO=0x00;
E_DATA=2;
F_ADDR=0xa4;
RF_ADDR=0xa5;
EA=0;
write_eeprom();
 EA=1;

motor_flag=0;
call_unit2();
 if(lmaxidema==0)
      {
      card_two_flag=0;
      fill_blank();
       put_lcd_download_1line();
        put_lcd_download_2line();

      put_rech_msg();
      put_lcd_download_1line();
      relay=0;
      while(1)
      {;}
      }
  }
  }
  }
  }

void timer0() interrupt 1
{
                        EA=0;
                        TR0=0;
                        TH0=TV0;
                        TL0=TVL0;

                        if(buzzer_flag==1)
                         {
                            buzzer=1;
                            misec++;
                            if(misec>=5000)
                               {
                                buzzer=0;
                                misec=0;
                                buzzer_flag=0;
```

```
                        }
                    }

            //      key();
                    TR0=1;
                    EA=1;
}

void delay1()
{unsigned int  p;
for(p=1;p<=10000;)
    {p++;}
}

void put_mass_msg()
{   disp[1]=('P');
    disp[2]=('R');
    disp[3]=('E');
    disp[4]=('-');
    disp[5]=('P');
    disp[6]=('A');
    disp[7]=('I');
    disp[8]=('D');
    disp[9]=(' ');
    disp[10]=('E');
    disp[11]=('N');
    disp[12]=('E');
    disp[13]=('R');
    disp[14]=('G');
    disp[15]=('Y');
    disp[16]=(' ');
 }

 void put_mass_msg1()
 {
     disp[1]=('C');
     disp[2]=('A');
     disp[3]=('R');
     disp[4]=('D');
     disp[5]=(' ');
     disp[6]=('S');
     disp[7]=('Y');
     disp[8]=('S');
     disp[9]=('T');
     disp[10]=('E');
     disp[11]=('M');
```

```
    disp[12]=(' ');
    disp[13]=(' ');
    disp[14]=(' ');
    disp[15]=(' ');
    disp[16]=(' ');
}

 void put_disp1_msg()
 {   disp[1]=('N');
     disp[2]=('O');
     disp[3]=(' ');
     disp[4]=('O');
     disp[5]=('F');
     disp[6]=(' ');
     disp[7]=('U');
     disp[8]=('N');
     disp[9]=('I');
     disp[10]=('T');
     disp[11]=('S');
     disp[12]=(':');
    //disp[13]=(' ');
    //disp[14]=(' ');
    //disp[15]=(' ');
    //disp[16]=(' ');
   }


   void put_card1_msg()
   {
      disp[1]=(' ');
      disp[2]=(' ');
      disp[3]=(' ');
      disp[4]=(' ');
      disp[5]=('C');
      disp[6]=('A');
      disp[7]=('R ');
      disp[8]=('D ');
      disp[9]=('-');
      disp[10]=('1');
      disp[11]=(' ');
      disp[12]=(' ');
      disp[13]=(' ');
      disp[14]=(' ');
      disp[15]=(' ');
      disp[16]=(' ');
    }
```

```
void put_card2_msg()
{
   disp[1]=(' ');
   disp[2]=(' ');
   disp[3]=(' ');
   disp[4]=(' ');
   disp[5]=('C');
   disp[6]=('A');
   disp[7]=('R ');
   disp[8]=('D ');
   disp[9]=('-');
   disp[10]=('2');
   disp[11]=(' ');
   disp[12]=(' ');
   disp[13]=(' ');
   disp[14]=(' ');
   disp[15]=(' ');
   disp[16]=(' ');
 }
  void put_mass_msg3()
  {
     disp[1]=(' ');
     disp[2]=(' ');
     disp[3]=('1');
     disp[4]=(' ');
     disp[5]=(' ');
     disp[6]=(' ');
     disp[7]=(' ');
     disp[8]=(' ');
     disp[9]=(' ');
     disp[10]=(' ');
     disp[11]=(' ');
     disp[12]=(' ');
     disp[13]=(' ');
     disp[14]=('2');
     disp[15]=(' ');
     disp[16]=(' ');

   }

  void put_mass_msg4()
  {
      disp[1]=('E');
      disp[2]=('n');
      disp[3]=('t');
```

```
    disp[4]=('e');
    disp[5]=('r');
    disp[6]=(' ');
    disp[7]=('t');
    disp[8]=('w');
    disp[9]=('o');
    disp[10]=(' ');
    disp[11]=('c');
    disp[12]=('a');
    disp[13]=('r');
    disp[14]=('d');
    disp[15]=(' ');
    disp[16]=(' ');
}

void put_mass_msg5()
{
     disp[1]=('I');
     disp[2]=('D');
     disp[3]=('=');
    //disp[4]=(' ');
    //disp[5]=(' ');
    //disp[6]=(' ');
    //disp[7]=(' ');
    //disp[8]=(' ');
      disp[9]=(' ');
      disp[10]=(' ');
      disp[11]=(' ');
      disp[12]=(' ');
      disp[13]=(' ');
      disp[14]=(' ');
      disp[15]=(' ');
      disp[16]=(' ');
  }

  void put_mass_msg6()
  {
       disp[1]=('E');
       disp[2]=('n');
       disp[3]=('t');
       disp[4]=('e');
       disp[5]=('r');
       disp[6]=(' ');
       disp[7]=('n');
       disp[8]=('e');
       disp[9]=('w');
```

```
    disp[10]=(' ');
    disp[11]=(' ');
    disp[12]=('c');
    disp[13]=('a');
    disp[14]=('r');
    disp[15]=('d');
    disp[16]=(' ');
}
void put_mass_msg7()
{
    disp[1]=('p');
    disp[2]=('r');
    disp[3]=('o');
    disp[4]=('g');
    disp[5]=('r');
    disp[6]=('a ');
    disp[7]=('m');
    disp[8]=('m');
    disp[9]=('i');
    disp[10]=('n');
    disp[11]=('g');
    disp[12]=(' ');
    disp[13]=('d');
    disp[14]=('o');
    disp[15]=('n');
    disp[16]=('e');
}

void put_prg1_msg()
{
    disp[1]=('S');
    disp[2]=('E');
    disp[3]=('L');
    disp[4]=('E');
    disp[5]=('C');
    disp[6]=('T ');
    disp[7]=(' ');
    disp[8]=('T');
    disp[9]=('H');
    disp[10]=('E');
    disp[11]=(' ');
    disp[12]=('C');
    disp[13]=('A');
    disp[14]=('R');
    disp[15]=('D');
    disp[16]=(' ');
```

```
}
void put_prg_msg()
{
    disp[1]=('P ');
    disp[2]=('R');
    disp[3]=('O');
    disp[4]=('G');
    disp[5]=('R');
    disp[6]=('A ');
    disp[7]=('M');
    disp[8]=('M');
    disp[9]=('I');
    disp[10]=('N');
    disp[11]=('G');
    disp[12]=('.');
    disp[13]=('.');
    disp[14]=('.');
    disp[15]=('.');
    disp[16]=('.');
 }

 /*
 void delay()
 {unsigned char p;
 for(p=1;p<=10;)
 {p++;}
 } */
 void skey()    //P1 is used.(switches connected at P3)
 {set();
         if(set_flag==1)
         {
           set_flag=0;
           fill_blank();
           put_mass_msg6();
           put_lcd_download_1line();
            do{;}while(card==1);

           put_prg1_msg();
           put_lcd_download_1line();
           // do{;}while(card==1);
             enable_cursor();
                  cursor_pos=19;
                  send_cursor();
                   put_mass_msg3();
            put_lcd_download_2line();
             enable_cursor();
```

```
            cursor_pos=19;
            send_cursor();

do{
//put_mass_msg3();
      //put_lcd_download_2line();

  move2();
              delay();
              set();

  // maxi_dema();
   // set();

            }while(set_flag==0);

  set_flag=0;
  if(cursor_pos==19)
   {
   first_flag=1;
   ADDR_LO=0;
   E_DATA=2;
   F_ADDR=0xa0;
   RF_ADDR=0xa1;
   EA=0;
    read_eeprom();

    maxi_dema();

   }
   else
     {
     second_flag=1;
     first_flag=0;
     ADDR_LO=0;
    E_DATA=2;
    F_ADDR=0xa4;
    RF_ADDR=0xa5;
    EA=0;
     read_eeprom();

            maxi_dema();
     }
   set_flag=0;
      }
}
```

```
void delay()      //delay for 100millisec
 {int ia;
 for(ia=1;ia<=80;)//800
 {ia++;}
 }

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

        INCREMENT SWITCH FUNCTION
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

 void incr()

 {
 inc_flag=0;
  if(inc_sw==0) // set switch=p3.0
  { delay();
    delay();
     delay();

         if(inc_sw==0)   //key press is verified
   {
    delay();
  //  key_ena=0;
    enable_cursor();

              inc_flag=1;
              inc1(cursor_pos);
              put_demand_msg2();
              put_lcd_download_1line();

          send_cursor();
          delay();
         // key_ena=1;
          //data_change();
          }
   }
  }

  /*
  void  inc(unsigned char z)
```

```
{

if(inc_flag==1)
        {
       switch(z)              //to choose the digit to increment

    {
     case 11:              //increment
      {

            delay();

            if(digit1==0)     //if data value reaches 9 it is made 0
                  {
                   digit1=1;
                   digit2=0;
                   digit3=0;
                   digit4=0;
                   digit5=0;
                  }
          else if(digit1==1)
              digit1=0;
                disp[z]=(digit1+0x30);
                disp[13]=(digit2+0x30);
                disp[14]=(digit3+0x30);
                disp[15]=(digit4+0x30);
                disp[16]=(digit5+0x30);

 break;
 }
 case 13:
  {
   if(digit1<1)
   digit2++;        //increment
   delay();
   if(digit2==10)          //if data value reaches 5 it is made 0
   digit2=0;
   }
    else
   {
    digit1=0;
    digit2==1;
   }
    disp[z]=(digit2+0x30);
    disp[11]=(digit1+0x30);
```

```
break;
}
case 14:
{
 if(digit1<1)
  {
   digit3++;                    //increment
   delay();
   if(digit3==10)               //if data value reaches 9 it is made 0
   digit3=0;
  }
 else
  {
   digit1=0;
   digit3=1;
  }
  disp[z]=(digit3+0x30);
  disp[11]=(digit1+0x30);

 break;
 }
 case 15:
  {
   if(digit1<1)
    {
     digit4++;              //increment
     delay();
     if(digit4==10)             //if data value reaches 5 it is made 0
     digit4=0;
    }
    else
     {
      digit1=0;
      digit4=1;
     }
    disp[z]=(digit4+0x30);
    disp[11]=(digit1+0x30);
  break;
  }
  case 16:
   {
    if(digit1<1)
    {
     digit5++;              //increment
     delay();       //hour should not be greater than 23hrs
     if(digit5 ==9)     //in case of digit 6 is 2
```

```
 digit5=0;
 }
else
{
 digit1=0;
 digit5=1;
}
disp[z]=(digit5+0x30);
disp[11]=(digit1+0x30);
break;}
}
inc_flag=0;
send_cursor();
}
*/
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

        SET SWITCH FUNCTION
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
void set()
{
//set_flag=0;
 if(set_sw==0)              // check for key press
 { delay();
   delay();

         if(set_sw==0)           //verify for key press
          { do{delay();}
            while(set_sw==0);    //wait until key is released

            set_flag=1;          // set the flag
            }
  }
 }
 /*
 void menu()
 {
 menu_flag=0;
  if(menu_sw==0)             // check for key press
  { delay();
    delay();
```

```
        if(menu_sw==0)          //verify for key press
         { do{delay();}
          while(menu_sw==0);   //wait until key is released

         menu_flag=1;        // set the flag
         }
}
}

  */
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

        BCD TO BINARY CONVERSION
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

 void put_demand_msg2()
 {
 disp[1]=('U');
 disp[2]=('N');
 disp[3]=('I');
 disp[4]=('T');
 disp[5]=('S');
 disp[6]=(' ');
 disp[7]=(' ');
 disp[8]=(' ');
 disp[9]=(':');
 //disp[10]=(' ');
 //disp[11]=(' ');
 //disp[12]=('.');
 //disp[13]=(' ');
 disp[14]=(' ');
 disp[15]=(' ');
 disp[16]=(' ');
 }

 void put_rech_msg()
 {
  disp[1]=('R');
  disp[2]=('E');
  disp[3]=('C');
  disp[4]=('H');
  disp[5]=('A');
```

```
disp[6]=('R');
disp[7]=('G');
disp[8]=('E');
disp[9]=(' ');
disp[10]=(' ');
disp[11]=(' ');
disp[12]=('.');
disp[13]=(' ');
disp[14]=(' ');
disp[15]=(' ');
disp[16]=(' ');
}

void move1()
{

 if(mov_sw==0)  //move switch=p3.1
  {
    delay();
                delay();

                if(mov_sw==0)
        {
                        do{
                         delay();
              }while(mov_sw==0);     //wait until key is released
        //  key_ena=0;

                         enable_cursor();
                         cursor_pos++;
                         if(cursor_pos==12)
                         cursor_pos=13;
                         if(cursor_pos==15)
                         cursor_pos=11;
                         send_cursor();
                        // key_ena=1;
                        // data_change();

                }
    }

 }

 void move2()
 {
```

```
if(mov_sw==0)  //move switch=p3.1
 {  delay();
                delay();

                if(mov_sw==0)
        {
                        do{delay();}      //wait until key is released
                        while(mov_sw==0);
                 // key_ena=0;
                        enable_cursor();

                        if(cursor_pos==19)
                          cursor_pos=30;
                        else
                          if(cursor_pos==30)
                          cursor_pos=19;
                        send_cursor();
                        // data_change();

                }
      }
}

void move3()
{
 if(mov_sw==0)                  //move switch=p3.1
   {  delay();
                delay();

                if(mov_sw==0)
         {
                        do{delay();}            //wait until key is released
                        while(mov_sw==0);
                // key_ena=0;
                        enable_cursor();
                          cursor_pos++;
                        if(cursor_pos==14)
                          cursor_pos=10;
                       //  else
                         //   if(cursor_pos==15)
                           // cursor_pos=1;
                         send_cursor();
                        // key_ena=1;
                        //  data_change();

                }
```

```
      }
}

/*
void bin_bcd(double g)
{
     y1=g*10000;
     y5=y1/10000;
     y6=y1%10000;
     y2=y6/1000;
     y3=y6 % 1000;
     y4= y3/100;
       digit1=y5;
     digit2=y2;
     digit3=y4;

 }

 /* void   maxi_dema()
  {
   //  ADDR_LO=0;
    // E_DATA=2;
    // EA=0;
    // read_eeprom();
    // EA=1;
    fill_blank();
      put_lcd_download_1line();
            put_lcd_download_2line();

     maxidema=e_reg[2]<<8;
     maxidema=maxidema|e_reg[1];
     binb_bcd6(maxidema);
         disp[10]=(digit1+0x30);

          disp[11]=(digit2+0x30);

          disp[12]=(digit3+0x30);

          disp[13]=(digit4+0x30);

            put_demand_msg2();
                  put_lcd_download_1line();

     //  maxi_flag=1;
       menu_flag=0;
       set_flag=0;
```

```
        enable_cursor();
cursor_pos=10;
              send_cursor();

    do
{
 enable_cursor();
 set();                          //set switch is checked
 delay();
  move3();                       //move switch is checked
 delay();
 incr();                         //increment switch is checked
delay();
 delay();
     }
while(set_flag==0);          //checks whether set switch is pressed
cursor_pos=2;
fill_blank();
put_lcd_download_1line();
put_lcd_download_2line();
disp[10]=(digit1+0x30);
disp[11]=(digit2+0x30);
maxidema= bcd_bin6();
disp[12]=(digit3+0x30);
disp[13]=(digit4+0x30);
put_demand_msg2();
put_lcd_download_1line();
dmaxidema=maxidema;
 dmaxidema=0xff & dmaxidema;
 e_reg[1]=dmaxidema;
dmaxidema=maxidema;
dmaxidema=0xff00 & dmaxidema;
 dmaxidema=dmaxidema>>8;
 e_reg[2]=dmaxidema;
     if(first_flag==1)
    {

  F_ADDR=0xa0;
  RF_ADDR=0xa1;

     }
  else
      {
      F_ADDR=0xa4;
      RF_ADDR=0xa5;
      }
```

```
        ADDR_LO=0x00;
  E_DATA=2;
  //EA=0;
  write_eeprom();
 // EA=1;
 fill_blank();
  put_lcd_download_2line();
   put_lcd_download_1line();

 put_mass_msg7();
 put_lcd_download_2line();
 fill_blank();

 do{;}while(card==0);

}

/* void  lmaxi_dema()
 {
    ADDR_LO=2;
    E_DATA=2;
    EA=0;
    read_eeprom();
    EA=1;
    lmaxidema=e_reg[2]<<8;
    lmaxidema=lmaxidema|e_reg[1];
    binb_bcd6(lmaxidema);
        disp[10]=(digit1+0x30);

         disp[11]=(digit2+0x30);

         disp[12]=(digit3+0x30);

         disp[13]=(digit4+0x30);

           put_demand_msg3();
                put_lcd_download_1line();

       //maxi_flag=1;
        pf_flag=1;
       menu_flag=0;
       set_flag=0;
             enable_cursor();
       cursor_pos=10;
                   send_cursor();
```

```
   do{
    enable_cursor();
    key_ena=1;
    data_change();
    set();                          //set switch is checked
    delay();
    move3();                        //move switch is checked
    delay();
    incr();                         //increment switch is checked
    delay();
    delay();


    }
    while(set_flag==0);        //checks whether set switch is pressed
    //maxi_flag=0;
     pf_flag=0;
      // set_flag=0;
       cursor_pos=2;
             fill_blank();
put_lcd_download_1line();
       put_lcd_download_2line();

             lmaxidema= bcd_bin6();

     disp[10]=(digit1+0x30);

     disp[11]=(digit2+0x30);

     disp[12]=(digit3+0x30);

     disp[13]=(digit4+0x30);

       put_demand_msg3();
             put_lcd_download_2line();

            dmaxidema=lmaxidema;
            dmaxidema=0xff & dmaxidema;
            e_reg[1]=dmaxidema;
            dmaxidema=lmaxidema;
            dmaxidema=0xff00 & dmaxidema;
dmaxidema=dmaxidema>>8;
 e_reg[2]=dmaxidema;
ADDR_LO=0x02;
E_DATA=2;
```

```
   EA=0;
   write_eeprom();
   EA=1;

}
void delay5()
{
unsigned char p;
for(p=1;p<=10;)
{p++;}
} */

void  inc1(unsigned char z1)
 {
   if(inc_flag==1)
   {
    switch(z1)//to choose the digit to increment
    {
        case 10:{
                    digit1++;
                     if(digit1>9)
                    digit1=0;

disp[10]=(digit1+0x30);//increment
 break;
  }
         case 11: {
                    digit2++;
                    if(digit2>9)
                   digit2=0;
                   disp[11]=(digit2+0x30);
                    break;
                   }
         case 12:  {
                    digit3++;
                    if(digit3>9)
                    digit3=0;
                   disp[12]=(digit3+0x30);
                    break;
                   }
          case 13: {
                    digit4++;
                     if(digit4>9)
                     digit4=0;
                     disp[13]=(digit4+x30);
                    break;
```

```
}

  }
   inc_flag=0;
  send_cursor();
  }

 }

void binb_bcd6(unsigned int g1)
{
    g2=g1;
    y1=g2/10000;
    y2=g2%10000;
    y3=y2/1000;
    y4= y2%1000;
    y5=y4/100;
    y6=y4%100;
    y7=y6/10;
    y8=y6%10;

    digit1=y3;
    digit2=y5;
    digit3=y7;
    digit4=y8;

  /*else
    {
      digit1=y1;
      digit2=y3;
      digit3= y5;
    }*/
}

unsigned int   bcd_bin6()                    //converts BCD-binary
{unsigned int mm1,mm2,mm3,mm4;
    mm1=digit1*1000;
    mm2=digit2*100;
    mm3=digit3*10;
    mm4=mm1+mm2+mm3+digit4;
    //mm5=m4*.0001;
    return mm4;                              //return the result to particular register

}

void sepr()
```

```
{
dmaxidema=maxidema & 0xff;
 l_units1=dmaxidema;
 dmaxidema=maxidema & 0xff00;
 dmaxidema=dmaxidema>>8;
 h_units1=dmaxidema;
}

void sepr2()
{
dmaxidema=lmaxidema & 0xff;
 l_units1=dmaxidema;
 dmaxidema=lmaxidema & 0xff00;
 dmaxidema=dmaxidema>>8;
 h_units1=dmaxidema;
}

void call_unit()
 {
     //EA=0;
     //bin_bcd(maxidema);
      binb_bcd6(maxidema);
     //EA=1;

   // disp[12]=( digit1+0x30);
     disp[13]= (digit1+0x30);
     disp[14]= (digit2+0x30);
     disp[15]= (digit3+0x30);
     disp[16]= (digit4+0x30);

   put_disp1_msg();
   put_lcd_download_2line();
    fill_blank();

     put_card1_msg();
     put_lcd_download_1line();
     fill_blank();
 }

 void call_unit2()
 {
     //EA=0;
     //bin_bcd(maxidema);
      binb_bcd6(lmaxidema);
     //EA=1;
```

```
  // disp[12]=( digit1+0x30);
     disp[13]= (digit1+0x30);
     disp[14]= (digit2+0x30);
     disp[15]= (digit3+0x30);
     disp[16]= (digit4+0x30);

  put_disp1_msg();
   put_lcd_download_2line();

   fill_blank();
    put_card2_msg();
      put_lcd_download_1line();
      fill_blank();
}

void motor_on_sw()
{
//set_flag=0;
 if(motor_sw==0)                        // check for key press
 { delay();
         if(motor_sw==0)                //verify for key press
          { do{;}while(motor_sw==0);    //wait until key is released

          motor_flag=1;                 // set the flag
          }
  }
}
```