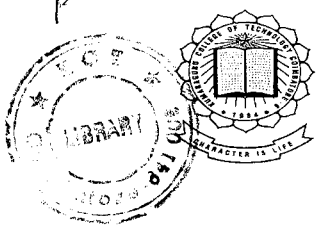


TRACKING OF BAGGAGE IN AIRPORTS USING RADIO FREQUNCY IDENTIFICATION

A PROJECT REPORT

Submitted by

POORNIMA DEVI.M	**(71201105029)**
S.G.SIVAGURU	**(71201105055)**
S.VEENA	**(71201105070)**
VINITH.M	**(71201105071)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRICAL & ELECTRONICS ENGINEERING

Under the guidance of
Mr.S.Titus

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE - 641006

ANNA UNIVERSITY:: CHENNAI - 600 025

APRIL – 2005

ANNA UNIVERSITY: CHENNAI-600 025

BONAFIDE CERTIFICATE

Certified that this project report titled " **RADIO FREQUENCY IDENTIFICATION AND TRACKING OF BAGGAGE IN AIRPORTS** " is the bonafide work of

POORNIMA DEVI.M	-	**Register No. 71201105029**
S.G.SIVAGURU	-	**Register No. 71201105055**
VEENA.S	-	**Register No. 71201105070**
VINITH.M	-	**Register No. 71201105071**

who carried out the project work under my supervision.

Signature of the Head of the Department

Signature of the guide

Prof.K.Regupathy Subramanian, B.E., M.Sc.
DEAN/EEE,
Kumaraguru college of technology

Mr.S.Titus, M.E
Lecturer, EEE
Kumaraguru college of technology

CERTIFICATE OF EVALUATION

College : KUMARAGURU COLLEGE OF TECHNOLOGY

Branch : Electrical & Electronics Engineering

Semester : Eighth Semester

S.No.	Name of the Students	Title of the Project	Name of the Supervisor with Designation
01	M.Poornima Devi	**"Tracking of Baggage in Airports using Radio-Frequency Identification"**	Mr.S.Titus M.E. Lecturer
02	S.G.SivaGuru		
03	S.Veena		
04	M.Vinith		

The report of the project work submitted by the above students in partial fulfillment for the award of Bachelor of Engineering degree in Electrical & Electronics Engineering of Anna University were evaluated and confirmed to be report of the work done by the above students and then evaluated.

(INTERNAL EXAMINER) **(EXTERNAL EXAMINER)**

ACKNOWLEDGEMENT

The successful completion of our project can be attributed to the combined efforts made by us and the contribution made in one form or the other, by the individuals we hereby acknowledge.

We are highly privileged to thank **Dr.K.K.Padmanabhan**, B.Sc., (Engg)., M.Tech., Ph.D, MISTE and F.I.E principal, Kumaraguru College of Technology for providing the necessary facilities for successful completion of the project.

We express our heartfelt gratitude and thanks to the Dean (academics) of Electrical and Electronics Department, **Dr.T.M.Kameshwaran**, B.E., M.Sc (Engg), Ph.D., F.I.E., for encouraging us to choose this project and for being with us right from the beginning of the project and guiding us at every step.

We wish to express our deep sense of gratitude and indebtedness to the Dean (R&D) and Head of Electrical and Electronics Department, **Dr. K. Regupathy Subramaniam**, B.E (Hons), M.Sc., for his enthusiasm and encouragement, which has been instrumental in the success of the project.

We wish to place on record our deep sense of gratitude and profound thanks to our guide **Mr.S.Titus**, M.E., lecturer, Electrical and Electronics department, for her valuable guidance, constant encouragement, continuous support and co-operation rendered throughout the project.

We are also thankful to all teaching and non teaching staffs of electrical and electronics engineering Department for their kind help and encouragement in making our project successful.

Last but not least we extend our sincere thanks to all our parents and friends who have contributed their ideas and encouraged us for completing the project successfully.

DEDICATED TO OUR BELOVED PARENTS

ABSTRACT

We are living in a world of technology where automation and new trends seem to be on the rise. Technology has become more like horse with no reins. It is left to us to channel these new trends to achieve modern day systems. Radio Frequency Identification (RFID) is one such trend, which has been sweeping across the industrial landscape for quiet a while. It is set to revolutionize the modern day living.

The RFID system has been on the rise since the late 1990's.We based our project in this field. The handling of baggage in International airports has come under a tight scrutiny nowadays. It is estimated that several billions of rupees is lost annually in searching for misplaced or lost baggage. This project could save substantial amount of time and money if implemented. It is the automatic tracking of baggage in airports.

The whole concept of RFID revolves around two things .The **"TAG"** and the **"READER".** It is a generic concept that uses radio waves to automatically identify people or objects from a distance. The **READER** is placed at a distance. It emits radio signals and hit them with enough power to retransmit data back.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

RFID SYSTEM

Radio Frequency Identification has become a hot topic around the world. RFID system delivers what bar codes cannot: real time information on the precise location and status of goods in a process flow. It does it through wireless transfer of data between the RFID reader and the RFID tags. The purpose of RFID system is to enable data to be transmitted by a portable device called a tag, which is read by an RFID reader and processed according to the needs of a particular application.

The read/write capability of an active RFID system is also a significant advantage in interactive applications such as work-in-process or maintenance tracking. Though it is a costlier technology (compared with barcode), RFID has become indispensable for a wide range of automated data collection and identification applications that would not be possible otherwise.

Their frequency ranges also distinguish RFID systems. Low-frequency (30 KHz to 500 KHz) systems have short reading ranges and lower system costs. They are most commonly used in security access, asset tracking, and animal identification applications. High-frequency (850 MHz to 950 MHz and 2.4 GHz to 2.5 GHz) systems, offering long read ranges (greater than 90 feet) and high reading speeds, are used for such applications as railroad car tracking and automated toll collection. The significant advantage of all types of RFID systems is the non-contact, non-line-of-sight nature of the technology.

CHAPTER 2: BLOCK DIAGRAM

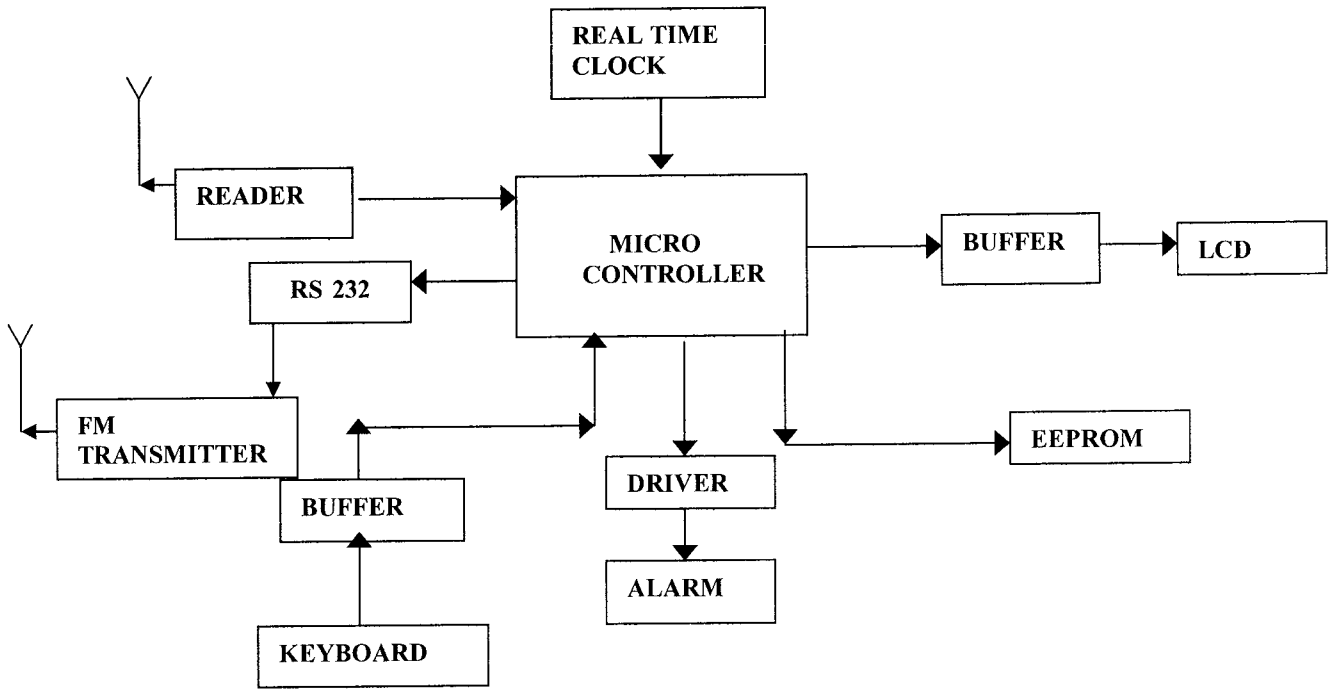


Fig 2.1: BLOCK DIAGRAM

2.2 DESCRIPTION:

The Transponder or tag is fixed on to the baggage to be tracked in the airport. When this tag comes within the range of the reader or interrogator, the tag is energized. Now, this tag transmits the data to the reader.

This data is automatically sent to the micro-controller for further processing. The time at which the tag is sensed is sent to the micro-controller from the RTC (Real Time Clock).

These details are displayed on LCD (Liquid Crystal Display) and also sent to the PC. The same is sent to the EEPROM (Electrically Erasable and Programmable Read Only Memory), which is used as a backup.

The information is sent to the PC via FM transmitter receiver. This data is stored in a database, which in turn can be stored, processed and retrieved.

CHAPTER 3: HARDWARE

MAIN COMPONENTS

3.1 Passive Tag and Reader:

Passive tags are those energized by the reader itself, they contain no power source, typically have very long lifetimes (near indefinite) a drawback over active tags is the read range, typically 2cm (1in) to 1.5m (4.5 ft), a strong positive is individual tag cost. RFID Passive tag is composed of a integrated electronic chip and a antenna coil that includes basic modulation circuitry and non-volatile memory.

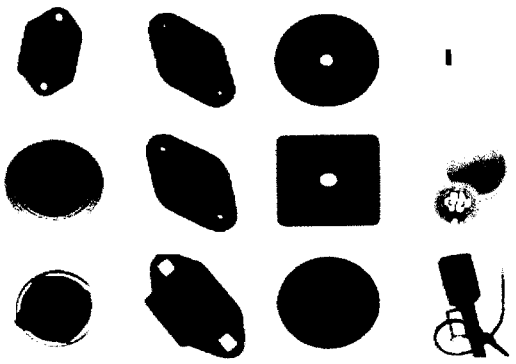


Fig 3.1.1 Different types of tags

For most general applications passive tags are usually the most cost effective. These are made in a wide variety of sizes and materials: there are durable plastic tags for discouraging retail theft, wafer thin tags for use within "smart" paper labels, tiny tracking tags which are inserted beneath an animal's skin and credit card sized tags for access control. In most cases the amount of data storage on a passive tag is fairly limited - capacity often being measured in bits as opposed to bytes.

However for most applications only a relatively small amount of data usually needs to be codified and stored on the tag, so the limited capacity does not normally pose a major limitation. Most tags also carry an unalterable unique electronic serial number, which makes RFID tags potentially very useful in applications where item tracking is needed or where security aspects are important.

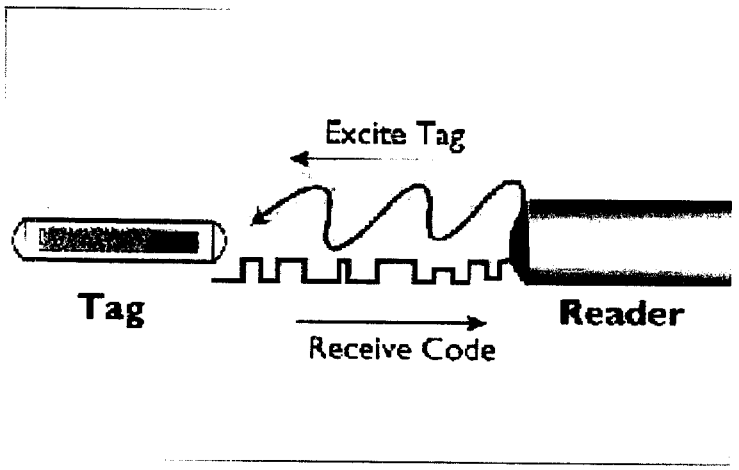


Fig 3.1.2 Interaction between tag and reader

The reader powers the tag (transponder), by emitting a radio frequency wave. The tag then responds by modulating the energizing field. This modulation can be decoded to yield the tags unique code, inherent in the tag. The resultant data can be the passed to a computer from processing. Tags have various salient features apart from their physical size: Other available features are: Read Only, Read Write, Anti-Collision.

Operating principles of RFID systems

There are a huge variety of different operating principles for RFID systems. The most important principle is inductive coupling, which is described in detail below.

Inductive coupling

An inductively coupled transponder comprises of an electronic data-carrying device, usually a single microchip and a large area coil that functions as an antenna.

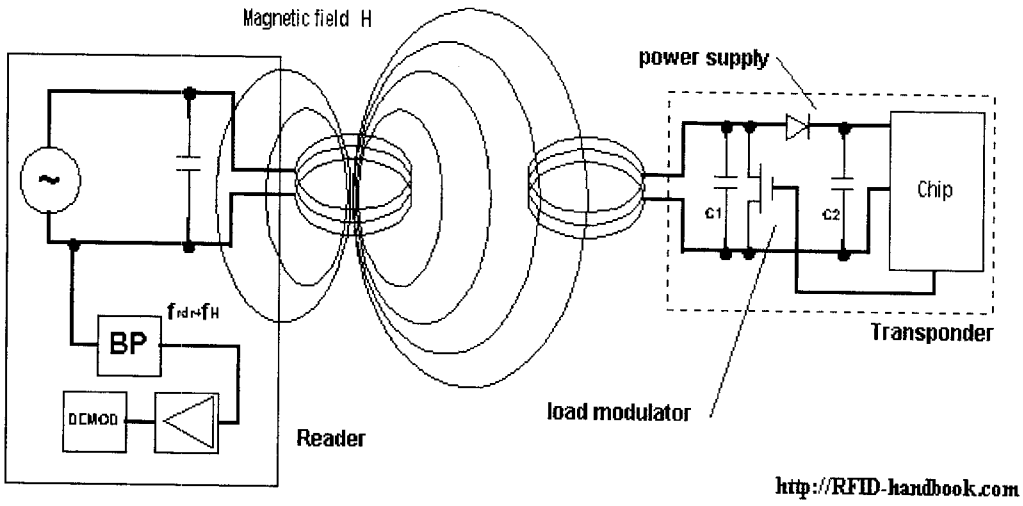


Fig3.1.3 Inductive Coupling

Inductively coupled transponders are almost always operated passively. This means that all the energy needed for the operation of the microchip has to be provided by the reader. For this purpose, the reader's antenna coil generates a strong, high frequency electro-magnetic field, which penetrates the cross-section of the coil area and the area around the coil. Because the wavelength of the frequency range used (< 135 kHz: 2400 m, 13.56 MHz: 22.1 m) is several times greater than the distance between the reader's antenna and the transponder, the electro-magnetic field may be treated as a simple magnetic alternating field with regard to the distance between transponder and antenna.

A small part of the emitted field penetrates the antenna coil of the transponder, which is some distance away from the coil of the reader. By induction, a voltage Vi is generated in the transponder's antenna coil. This voltage is rectified and serves as the power supply for the data-carrying device (microchip). A capacitor C1 is connected in parallel with the reader's antenna coil, the capacitance of which is selected such that it combines with the coil inductance of the antenna coil to form a parallel resonant circuit, with a resonant frequency that corresponds with the transmission frequency of the reader. Very high currents are generated in the antenna coil of the reader by resonance step-up in the parallel resonant circuit, which can be used to generate the required field strengths for the operation of the remote transponder.

The antenna coil of the transponder and the capacitor C1 to form a resonant circuit tuned to the transmission frequency of the reader. The voltage V at the transponder coil reaches a maximum due to resonance step-up in the parallel resonant circuit.

As described above, inductively coupled systems are based upon a transformer-type coupling between the primary coil in the reader and the secondary coil in the transponder. This is true when the distance between the coils does not exceed 0.16 times the wavelength, so that the transponder is located in the near field of the transmitter antenna

If a resonant transponder (i.e. the self-resonant frequency of the transponder corresponds with the transmission frequency of the reader) is placed within the magnetic alternating field of the reader's antenna, then this draws energy from the magnetic field. This additional power consumption can be measured as voltage drop at the internal resistance in the reader

antennae through the supply current to the reader's antenna. The switching on and off of a load resistance at the transponder's antenna therefore effects voltage changes at the reader's antenna and thus has the effect of an amplitude modulation of the antenna voltage by the remote transponder. If the switching on and off of the load resistor is controlled by data, then this data can be transferred from the transponder to the reader. This type of data transfer is called load modulation.

To reclaim the data in the reader, the voltage measured at the reader's antenna is rectified. This represents the demodulation of an amplitude-modulated signal.

Technical Specifications:

Frequency:	125 KHz / 13.56 MHz / 915 MHz / 2.45 GHz Read/Write
Distance:	Up to 6m (with mounted antenna)
Dimensions	Varies, as small as 0.8mm diameter
Weight:	6-54g
Memory:	Up to 16 Kbits
Data durability:	10 Years

The **advantages** of a passive tag are:

The tag functions without a battery; these tags have a useful life of twenty years or more.

- The tag is typically much less expensive to manufacture

- The tag is much smaller (some tags are the size of a grain of rice). These tags have almost unlimited applications in consumer goods and other areas.
- Tags can be read through a variety of substances such as snow, fog, ice, paint, crusted grime, and other visually and environmentally challenging conditions, where barcodes or other optically read technologies would be useless.
- RFID tags can also be read in challenging circumstances at remarkable speeds, in most cases responding in less than 100 millisecond

Antenna:

A reader reads identifiers from tags on pallets conveyed past the reader. The reader includes two interleaved linear arrays of antennas with circularly polarized fields. Each antenna is composed of a pair of crossed rods phased to have adjacent antennas of an array generate circularly polarized fields of opposite rotation.

Fig 3.1.3 Antenna

The vector components of the polarization in the direction across the width of the conveyor have peaks and nulls, and the interleaved arrays are arranged such that the nulls of one array's fields are covered with the peaks of the other array's fields. This arrangement allows the reader to the identifier from the tag when the tag is at any orientation. A tag at the side of the reader is aligned in the direction of travel by rails on the conveyor. The reader has antennas aligned in the direction of travel to read such tags.

3.2 8-Bit Micro controller – AT89S52

Features:

- Compatible with MCS-51 ® Products
- 8K Bytes of In-System Programmable (ISP) Flash Memory
- Endurance: 1000 Write/Erase Cycles
- 4.0V to 5.5V Operating Range
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Full Duplex UART Serial Channel
- Low-power Idle and Power-down Modes
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power off flag

DESCRIPTION:

The AT89S52 is a low-power, high-performance CMOS 8-bit micro controller with 8K Bytes of in-system programmable Flash memory. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the Industry-standard 80C51 instruction set and pin out. 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 in-system programmable flash on a monolithic chip, the Atmel AT89S52 is a powerful micro controller, which provides a highly flexible and cost-effective solution to many embedded control applications. The AT89S52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, Watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry.

In addition, the AT89S52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM con-tents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR). Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect. User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers:

Control and status bits are contained in registers T2CON and T2MOD for Timer 2. The register pair (RCAP2H, RCAP2L) is the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

Interrupt Registers:

The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the six-interrupt sources in the IP register.

T2CON – Timer/Counter 2 Control Register

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
7	6	5	4	3	2	1	0

Table 3.2.1 Interrupt registers

Symbol	Function
TF2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/T2	Timer or counter select for Timer 2. C/T2 = 0 for timer function.

	C/T2 = 1 for external event counter (falling edge triggered).
CP/RL2	Capture/Reload select. CP/RL2 = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/RL2 = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Table 3.2.2 T2CON Timer/Counter 2 Control Register

Dual Data Pointer Registers:

To facilitate accessing both internal and external data memory, two banks of 16-bit Data Pointer Registers are provided: DP0 at SFR address locations 82H-83H and DP1 at 84H-85H. Bit DPS = 0 in SFR AUXR1 selects DP0 and DPS = 1 selects DP1.

The user should always initialize the DPS bit to the appropriate value before accessing the respective Data Pointer Register.

Power Off Flag:

The Power Off Flag (POF) is located at bit 4 (PCON.4) in the PCON SFR. POF is set to "1" during power up. It can be set and rest under software control and is not affected by reset.

Memory Organization

MCS-51 devices have a separate address space for Program and Data Memory. Up to 64K bytes each of external Program and Data Memory can be addressed.

Program Memory

If the EA pin is connected to GND, all program fetches are directed to external memory. On the AT89S52, if EA is connected to VCC, program

fetches to addresses 0000H through 1FFFH are directed to internal memory and fetches to addresses 2000H through FFFFH are to external memory.

Data Memory

The AT89S52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. This means that the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space. When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions, which use direct addressing access of the SFR space.

For example, the following direct addressing instruction accesses the SFR at location 0A0H (which is P2). MOV 0A0H, #data Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H). MOV @R0, #data Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

Watchdog Timer

(One-time Enabled with Reset-out)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upsets. The WDT consists of a 13-bit counter and the Watchdog Timer Reset (WDTRST) SFR. The WDT is defaulted to disable from exiting reset. To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, it will increment every machine cycle while the oscillator is running. The WDT timeout period is dependent on the external clock frequency. There is no way to disable the WDT except through reset

(either hardware reset or WDT over-flow reset). When WDT overflows, it will drive an output RESET HIGH pulse at the RST pin.

Using the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT over-flow. The 13-bit counter overflows when it reaches 8191 (1FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the WDT at least every 8191-machine cycles. To reset the WDT the user must write 01EH and 0E1H to WDTRST.WDTRST is a write-only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is 96xTOSC, where TOSC=1/FOSC. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT .

WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt, which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S52 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT

is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down Mode. To ensure that the WDT does not overflow within a few states of exiting Power-down, it is best to reset the WDT just before entering Power-down mode. Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled.

The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S52 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and reenter IDLE mode. With WDIDLE bit enabled, the WDT will stop to count in IDLE mode and resumes the count upon exit from IDLE.

Timer 0 and 1

Timer 0 and Timer 1 in the AT89S52 operate the same way as Timer 0 and Timer 1 in the AT89C51 and AT89C52.

Timer 2

Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/T2 in the SFR T2CON. Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON. Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

RCLK +TCLK	CP/RL2	TR2	MODE
0	0	1	16-bit Auto-reload

0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

Table 3.2.3 Timer2 Operating Modes

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled During S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held.

Capture Mode

In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1- to-0 transition at external input T2EX also causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt.

Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 4). Upon reset,

the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

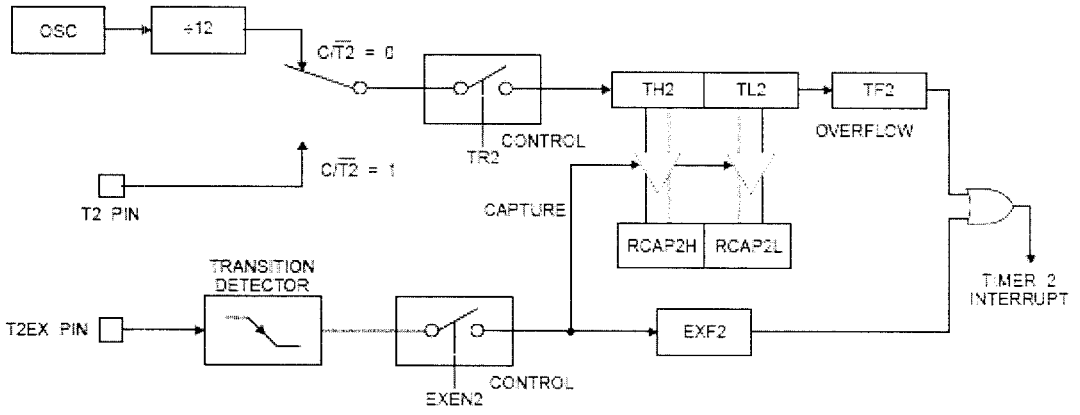


Fig 3.2.1 Timer2 in capture mode

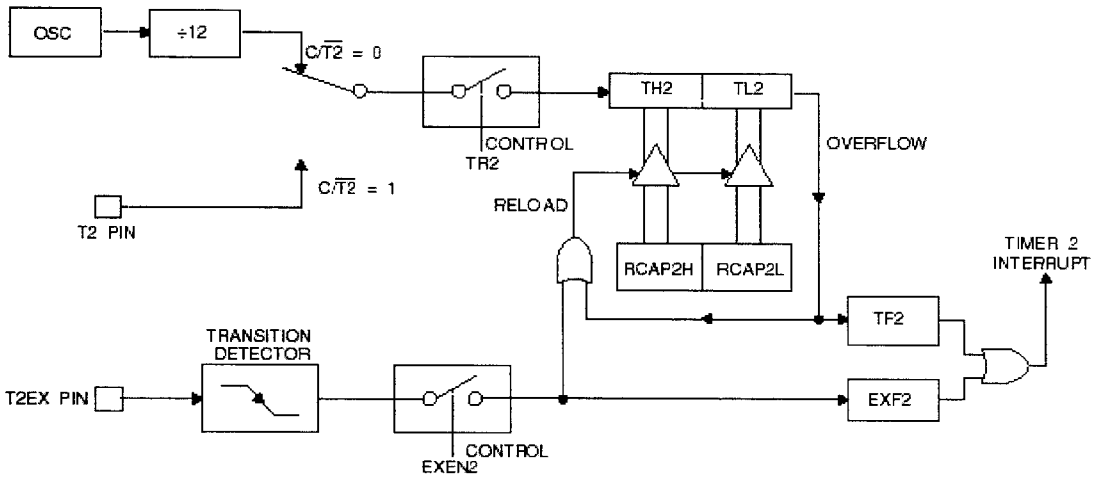


Fig 3.2.2 Timer 2 Auto Reload mode

Figure 3.2.2 shows Timer 2 automatically counting up when DCEN=0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The

overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture ModeRCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX.

This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled. Setting the DCEN bit enables Timer 2 to count up or down. In this mode, the T2EX pin controls the direction of the count. Logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit.

This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively. Logic 0 at T2EX makes Timer 2 count down. The timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers. The EXF2 bit toggles whenever Timer 2 overflows or underflows and can be used as a 17th bit of resolution. In this operating mode, EXF2 does not flag an interrupt.

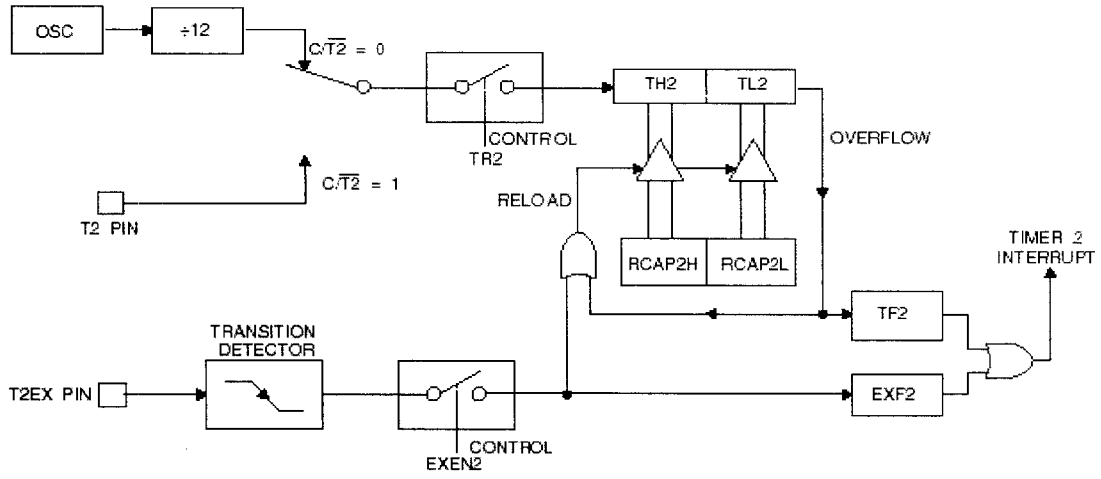


Fig 3.2.3 Timer 2 Auto reload mode (DCEN=1)

Baud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON.Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode. The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software. The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate according to the following equation. Mode 1 and 3 Baud Rates = Timer 2 Overflow Rate/16 The Timer can be configured for either timer or counter operation. In most applications, it is configured for timer operation (CP/T2 = 0). The timer operation is different for Timer 2 when it is used as a baud rate generator. Normally, as a timer, it increments every machine cycle (at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (at 1/2 the oscillator frequency).

The baud rate formula is given below.

(Mode 1 and 3)/Baud Rate = Oscillator Frequency/(32*[65536-

RCAP2H, RCAP2L]

)

Where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus, when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt. Note that when Timer 2 is running (TR2 = 1) as a timer in the baud rate generator mode, TH2 or TL2 should not be read from or written to. Under these conditions, the Timer is incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

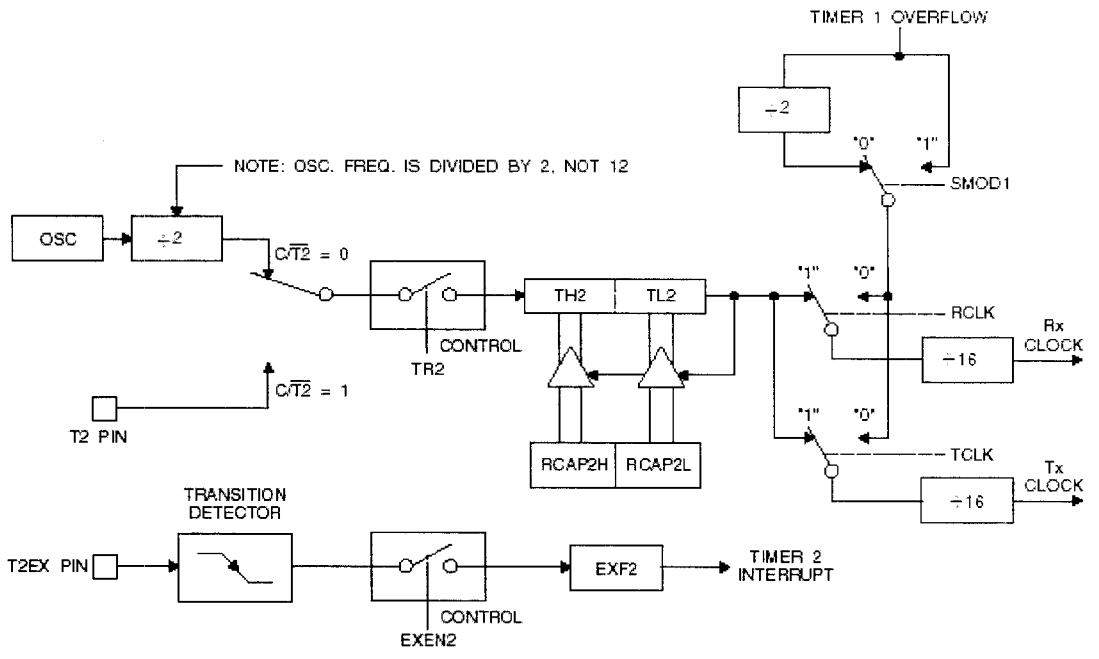


Fig 3.2.4 Timer 2 in Baud Rate Generator Mode

Interrupts

The AT89S52 has a total of six interrupt vectors: two external interrupts (INT0 and INT1), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once. In the AT89S52, bit position IE.5 is also unimplemented.

User software should not write 1s to these bit positions, since they may be used in future AT89 products.

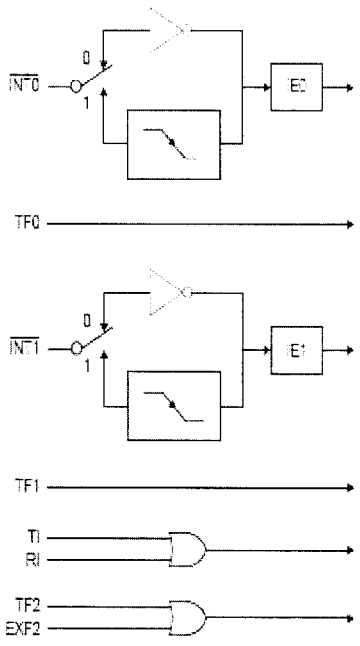


Fig 3.2.6 Interrupt Sources

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in registers T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and that bit will have to be cleared in software. The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

3.3 REAL TIME CLOCK

DESCRIPTION:

The DS12887 real-time clock (RTC) plus RAM is designed to be a direct replacement for the DS1287.The DS12887 is identical in form, fit, and function to the DS1287, and has an additional 64 bytes of general-purpose RAM. The logic level presented on AD6 during the address portion of an

access cycle determines access to this additional RAM space. A lithium energy source, quartz crystal, and write-protection circuitry are contained within a 24-pin dual in-line package. As such, the DS12887 is a complete subsystem replacing 16 components in a typical application. The functions include a nonvolatile time-of-day clock, an alarm, a 100-year calendar, programmable interrupt, square-wave generator, and 114 bytes of NV SRAM. The RTC is unique in that time-of-day and memory are maintained even in the absence of power.

Features:

- Drop-in replacement for IBM AT computer clock/calendar
- Pin-compatible with the MC146818B and DS1287
- Totally nonvolatile with over 10 years of operation in the absence of power
- Self-contained subsystem includes lithium, quartz, and support circuitry
- Counts seconds, minutes, hours, days, day of the week, date, month, and year with leap-year compensation valid up to 2100
- Binary or BCD representation of time, calendar, and alarm
- 12-hour or 24-hour clock with AM and PM in 12-hour mode
- Daylight Savings Time option
- Selectable between Motorola and Intel bus timing
- Multiplex bus for pin efficiency
- Interfaced with software as 128 RAM locations
- 14 bytes of clock and control registers
- 114 bytes of general-purpose RAM
- Programmable square-wave output signal

- Bus-compatible interrupt signals (IRQ)
- Three interrupts are separately software-maskable and testable
- Time-of-day alarm once/second to once/day
- Periodic rates from 122ms to 500ms
- End-of-clock update cycle
- Underwriters Laboratory (UL) recognized

3.4 EEPROM:

EEPROM (Electrically Erasable Programmable Read Only Memory) is a storage device of sorts. The 28c64 is right out of the Atmel Workshop. The 28c64 is accessed like a static RAM for the read or write cycles without the need for external components. During a "byte write" the address and data are latched internally, feeing the microprocessor address and the data bus for various other operations.

Following the initiation of the write cycle, the device will go to a busy state and automatically clear and write the latch data using an internal control timer. To determine when the write cycle is complete, the user has a chance of monitoring the Ready/Busy output or using data polling. The Ready/Busy pin is an open drain output, which allows easy configuration in wired-or systems. Alternatively data polling allows the user to read the location last written to when the write operation is complete. CMOS design and processing enables this part to be used in systems where reduced power consumption and reliability are required. The device has 4 standard modes of operation, which are

1) Read mode
2) Standby mode
3) Write inhibit

4) Byte write.

A complete family of packages is offered to provide utmost flexibility in operation.

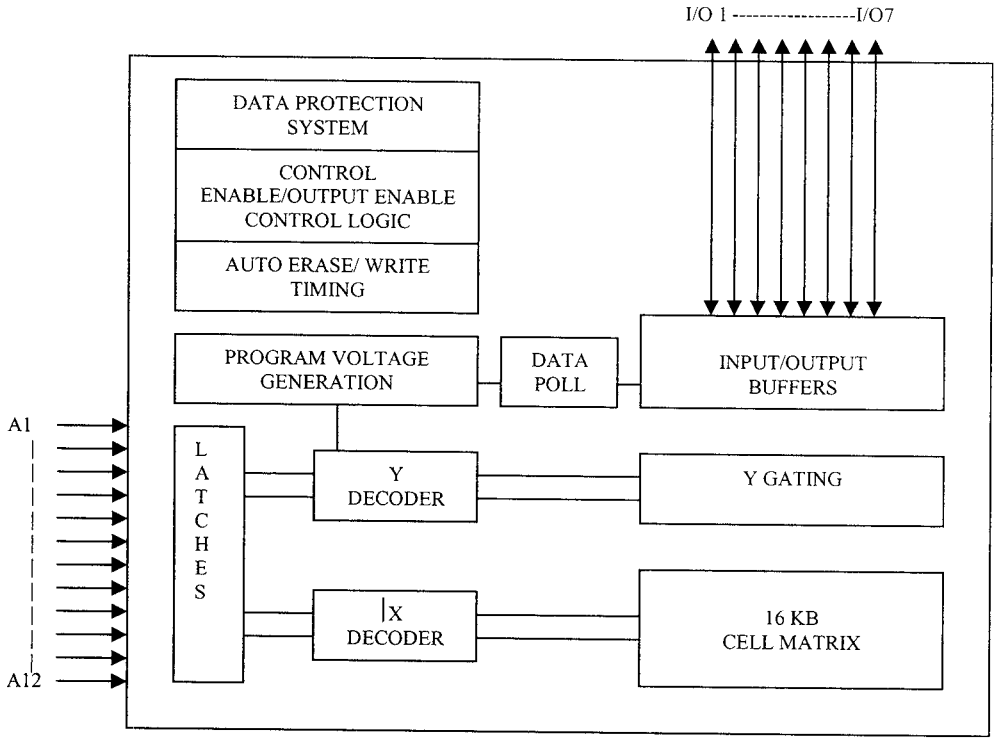


Fig 3.4.1.BLOCK DIAGRAM: EEPROM

3.5 RELAY

The relay activates the LED and BUZZER, which are connected to it only when the tag is sensed.

In this circuit transistor BC547 is used as a switch. The control signal is given to the base terminal of the transistor. The collector is attached to the relay coil. Relays are electromechanical devices. There are two types of relays.

1. Normally closed
2. Normally opened

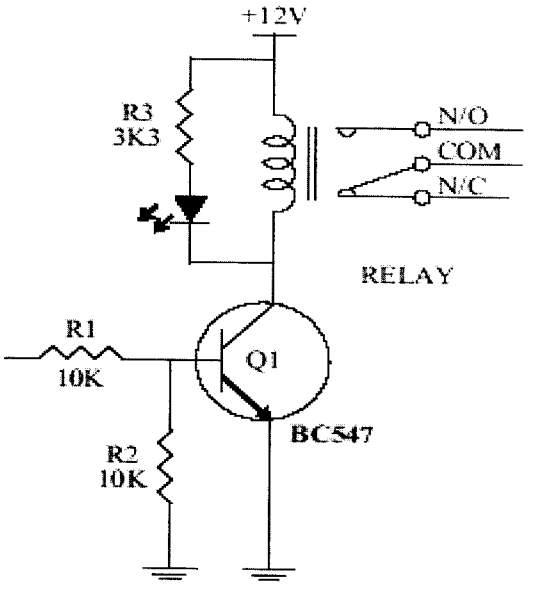


Fig 3.5.1. Relay circuit

We are using normally opened type relay. When the controller output is high the transistor will be in the ON state, so relay is energized. When the controller output from is low the transistor will be in the OFF state, so relay is de-energized. So according to the controller output the relay can be switched ON or OFF, thus giving the required output.

3.6 FM TRANSMITTER

General Description:

The 3^{18}encoders are a series of CMOS LSIs for remote control system applications. They are capable of encoding 18 bits of information, which consists of N address bits, and 18_N data bits. Each address/data input is externally trinary programmable if bonded out. It is otherwise set floating internally. Various packages of the 3^{18} encoders offer flexible combinations of programmable address/data to meet various application needs. The

programmable address/ data is transmitted together with the header bits via an RF or an infrared transmission medium upon receipt of a trigger signal. The capability to select a TE trigger type or a DATA trigger type further enhances the application flexibility of the 3^{18} series of encoders.

Functional Description:

Operation:

The 3^{18}series of encoders begins a three-word transmission cycle upon receipt of a transmission enable (TE for the HT600 / HT640 / HT680 or D12~D17 for the HT6187 /HT6207 /HT6247, active high). This cycle will repeat itself as long as the transmission enable (TE or D12~D17) is held high. Once the transmission enable falls low, the encoder output completes its final cycle and then stops as shown below.

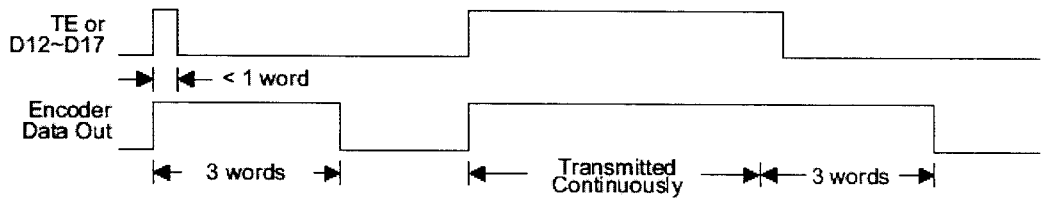
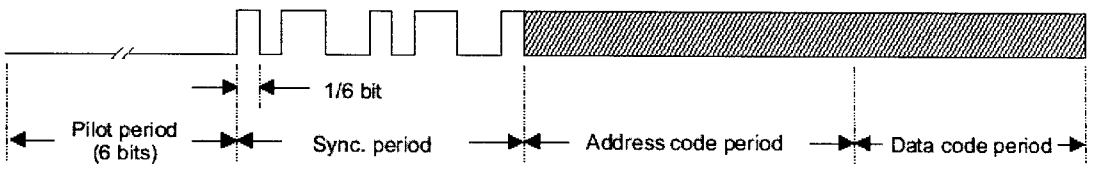


Fig 3.6.1 Transmission timing

Information word:

An information word consists of 4 periods as shown:



Composition of information

Fig 3.6.2 Information Word

3.7 FM Receiver

General Description:

The 3^{18} decoders are a series of CMOS LSIs for remote control system applications. They are paired with the 3^{18} series of encoders. For proper operation a pair of encoder/decoder pair with the same number of address and data format should be selected (refer to the encoder / decoder cross reference tables).

The 3^{18} series of decoders receives serial address and data from that series of encoders that are transmitted by a carrier using an RF or an IR transmission medium. It then compares the serial input data twice continuously with its local address. If no errors or unmatched codes are encountered, the input data codes are decoded and then transferred to the output pins. The VT pin also goes high to indicate a valid transmission. The 3^{18} decoders are capable of decoding 18 bits of information that consists of N bits of address and 18–N bits of data. To meet various applications they are arranged to provide a number of data pins whose range is from 0 to 8 and an address pin whose range is from 8 to 18. In addition, the 3^{18} decoders provide various combinations of address/data number in different packages.

Functional Description:

Operation:

The 3^{18} series of decoders provides various combinations of address and data pins in different packages. It is paired with the 3^{18}series of encoders. The decoders receive data transmitted by the encoders and interpret the first N bits of the code period as address and the last 18–N bits as data (where N is the address code number).

A signal on the DIN pin then activates the oscillator which in turns decodes the incoming address and data. The decoders will check the received

address twice continuously. If all the received address codes match the contents of the decoder's local address, the 18–N bits of data are decoded to activate the output pins, and the VT pin is set high to indicate a valid transmission. That will last until the address code is incorrect or no signal has been received. The output of the VT pin is high only when the transmission is valid. Otherwise it is low always.

Output type:

There are 2 types of output to select from:

(1) Momentary type

The data outputs follow the encoder during a valid transmission and then reset.

(2) Latch type

The data outputs follow the encoder during a valid transmission, and are then latched in this state until the next transmission occurs

Features:

- Operating voltage: 2.4V~12V
- Low power and high noise immunity CMOS technology
- Low standby current
- Three words transmission
- Built-in oscillator needs only 5% resistor
- Easy interface with an RF or infrared transmission media
- Minimal external components

Applications:

- Burglar alarm system
- Smoke and fire alarm system
- Garage door controllers
- Car door controllers

- Car alarm system
- Security system
- Cordless telephones
- Other remote control systems

3.8 LIQUID CRYSTAL DISPLAY (LCD)

Liquid crystal displays (LCDs) have materials, which combine the properties of both liquids and crystals. Rather than having a melting point, they have a temperature range within which the molecules are almost as mobile as they would be in a liquid, but are grouped together in an ordered form similar to a crystal.

An LCD consists of two glass panels, with the liquid crystal material sand witched in between them. The inner surface of the glass plates are coated with transparent electrodes which define the character, symbols or patterns to be displayed polymeric layers are present in between the electrodes and the liquid crystal, which makes the liquid crystal molecules to maintain a defined orientation angle.

One each polarizes are pasted outside the two glass panels. These polarizes would rotate the light rays passing through them to a definite angle, in a particular direction. When the LCD is in the off state, light rays are rotated by the two polarizes and the liquid crystal, such that the light rays come out of the LCD without any orientation, and hence the LCD appears transparent.

When sufficient voltage is applied to the electrodes, the liquid crystal molecules would be aligned in a specific direction. The light rays passing through the LCD would be rotated by the polarizes, which would result in activating / highlighting the desired characters. The LCD's are lightweight

with only a few millimeters thickness. Since the LCD's consume less power, they are compatible with low power electronic circuits, and can be powered for long durations.

The LCD does not generate light and so light is needed to read the display. By using backlighting, reading is possible in the dark. The LCD's have long life and a wide operating temperature range. Changing the display size or the layout size is relatively simple which makes the LCD's more customers friendly.

The LCDs used exclusively in watches, calculators and measuring instruments are the simple seven-segment displays, having a limited amount of numeric data. The recent advances in technology have resulted in better legibility, more information displaying capability and a wider temperature range. These have resulted in the LCDs being extensively used in telecommunications and entertainment electronics. The LCDs have even started replacing the cathode ray tubes (CRTs) used for the display of text and graphics, and also in small TV applications.

Crystalonics dot–matrix (alphanumeric) liquid crystal displays are available in TN, STN types, with or without backlight. The use of C-MOS LCD controller and driver ICs result in low power consumption. These modules can be interfaced with a 4-bit or 8-bit microprocessor /Micro controller.

- The built-in controller IC has the following features:
- Correspond to high speed MPU interface (2MHz)
- 80 x 8 bit display RAM (80 Characters max)
- 9,920-bit character generator ROM for a total of 240 character fonts. 208 character fonts (5 x 8 dots) 32 character fonts (5 x 10 dots)

- 64 x 8 bit character generator RAM 8 character generator RAM 8 character fonts (5 x 8 dots) 4 characters fonts (5 x 10 dots)
- Programmable duty cycles
- 1/8 – for one line of 5 x 8 dots with cursor
- 1/11 – for one line of 5 x 10 dots with cursor
- 1/16 – for one line of 5 x 8 dots with cursor
- Wide range of instruction functions display clear, cursor home, display on/off, cursor on/off, display character blink, cursor shift, display shift.
- Automatic reset circuit, which initializes the controller / driver ICs after power on.

3.9 PERSONAL COMPUTER (PC)

Interfacing the hard ware with the PC has the following advantages:

- Storing and retrieval of data becomes easier.
- Networking can be done and hence the entire system can be monitored online.
- Access can be user friendly.

Interfacing the hard ware with the PC is done using MAX232 (rs232)

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available. These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5μW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Features:

- Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)
- Low-Power Receive Mode in Shutdown (MAX223/MAX242)
- Meet All EIA/TIA-232E and V.28 Specifications
- Multiple Drivers and Receivers
- 3-State Driver and Receiver Outputs
- Open-Line Detection (MAX243)

CHAPTER 4: CONCLUSION

4.1 ADVANTAGES:

- Wireless transmission of data is done to enable usage in inaccessible areas, like industries.
- Data can be transmitted through FM to several computers simultaneously.
- A password lock is provided for protection purposes.
- The same system can be used for access control thereby ensuring security.
- Saves time and money by automatic tracking of baggage.
- This project is viable and cost effective.
- It is a simple foolproof system of tracking baggage with maximum reliability.

Other Applications:

The other applications of our project include,

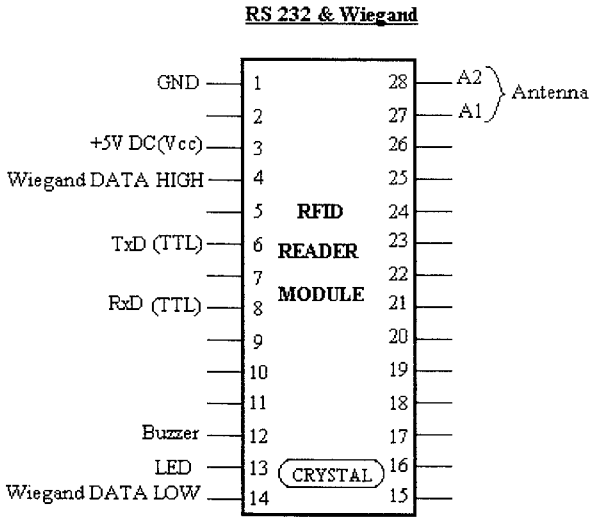
- Library Management
- Logistics
- Warehousing
- Toll Collection
- Animal Tracking
- Vehicle Identification

4.2 FUTURE WORK:

- To set a password lock for each individual baggage cleared at the airports.
- To create an applet for searching misplaced baggage.
- To improve versatility of the project.
- To make use of high frequency tags and readers to increase the sensitivity and range of operation.
- To transmit the data through the internet, so that it helps tracking globally.
- To implement a new feature called "anti-collision" where-in hundreds of tags can be tracked simultaneously.

APPENDIX A: HARDWARE

RFID 125 Reader Module Pin Diagram & Description



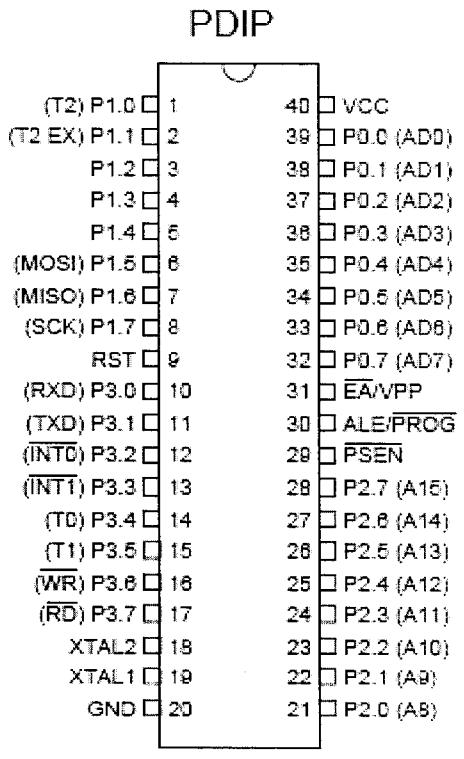
PIN NO.	*SIGNAL*	DESCRIPTION
Pin No : 6	TxD	Transmit data (TTL level) output from module to serial interface
Pin No : 4	Wiegand DATA HIGH (available in Wiegand)	It will give DATA HIGH signal.
Pin No : 8	RxD	Receive data (TTL level) input to the module from serial interface
Pin No : 12	Buzzer (active low)	Buzzer will buzz for 280 ms when tag is detected
Pin No : 13	LED (active low)	LED will glow for 280 ms when tag is detected
Pin No : 14	Wiegand DATA LOW (available in Wiegand)	It will give DATA LOW signal.
Pin No:27,28	Antenna Input	Loop Antenna should be connected.

Note:

1. Reader module has to be mounted on non-metallic surface, else it may affect the operation of reader.
2. Buzzer & LED are Active low signals.
3. For Buzzer & LED current limiting Resister has to be mounted. MAX current is 20mA. (470 or 510 ohms for LED and 240 or 270 Ohms for Buzzer)
4. LED's Anode and Buzzer's Positive marked pin to be connected to Vcc.
5. The antenna Inductance should be about 1 mH. The same needs tuning at Factory.
6. Wiegand out put format is also available in select readers.

MICROCONTROLLER 89S52

PIN 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 can 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. In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table. 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 use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 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 Pin	***Alternate Functions***
P 1.0	T2 (external count input to Timer/Counter 2), clock-out
P 1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P 1.5	MOSI (used for In-System Programming)

P 1.6	MISO (used for In-System Programming)
P 1.7	SCK (used for In-System Programming)

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 AT89S52, as shown in the following table. Port 3 also receives some control signals for Flash programming and verification.

Port Pin	Alternate Functions
P 3.0	RXD (serial input port)
P 3.1	TXD (serial output port)
P 3.2	INT0 (external interrupt 0)
P 3.3	INT1 (external interrupt 1)
P 3.4	T0 (timer 0 external input)
P 3.5	T1 (timer 1 external input)
P 3.6	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. This pin drives High for 96 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

ALE/PROG

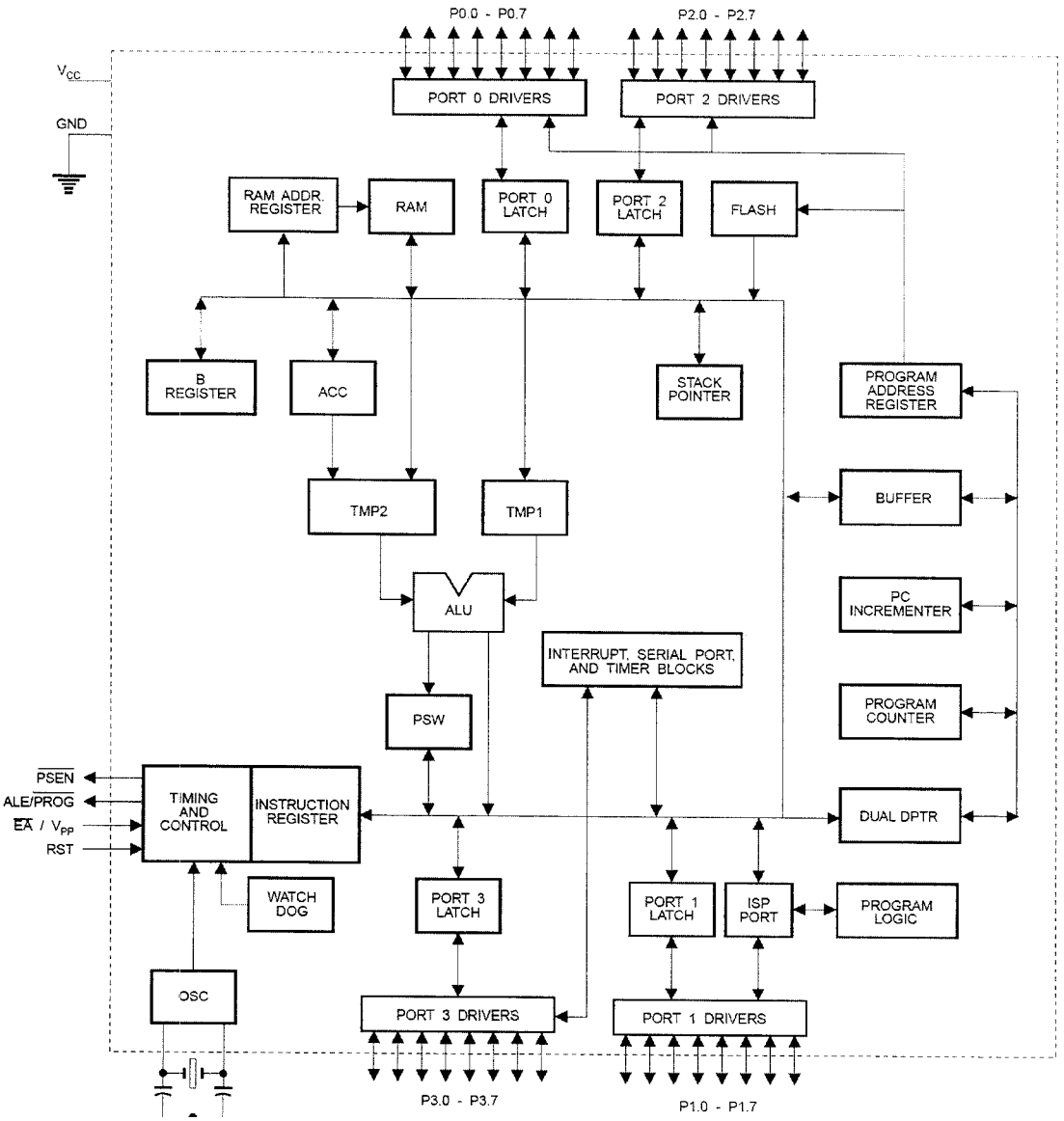
Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one

ALE pulse is skipped during each access to external data memory. If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. 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 micro-controller is in external execution mode.

PSEN

Program Store Enable (PSEN) is the read strobe to external program memory. When the AT89S52 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.



Block Diagram

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.

XTAL1

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

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. 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. 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 maxi-mum voltage high and low time specifications must be observed.

Idle Mode:

In idle mode, the CPU puts itself to sleep while all the on-chip 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 a hardware reset. Note that when idle mode 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 mode is terminated by a reset, the instruction following the one that invokes idle mode should not write to a port pin or to external memory.

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. Exit from Power-down mode can be initiated either by a hardware reset or by an enabled external interrupt.
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.

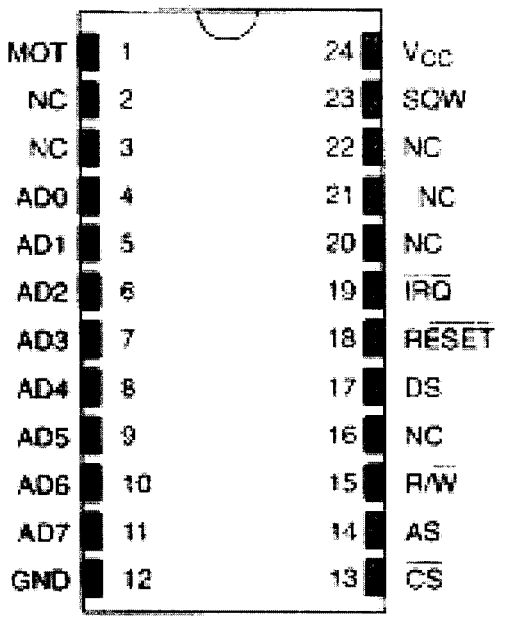
Status of external pins during Idle and Power Down Modes:

Mode	Program Memory	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
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

Program Memory Lock Bits:
The AT89S52 has three lock bits that can be left unprogrammed(U) or can be programmed (P) to obtain the additional features .When lock bit 1 is programmed, the logic level at the EA pins 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. The latched value of EA must agree with the current logic level at that pin in order for the device to function properly.

REAL TIME CLOCK – DS12887

PIN DIAGRAM



PIN DESCRIPTION:

AD0–AD7 – Multiplexed Address/Data Bus
N.C. – No Connection
MOT – Bus Type Selection
CS – Chip Select
AS – Address Strobe
R/W – Read/Write Input
DS – Data Strobe
RESET – Reset Input
IRQ – Interrupt Request Output
SQW – Square-Wave Output
VCC – +5V Supply
GND – Ground

ORDERING INFORMATION:

PART	PIN-PACKAGE	TOP MARK	TEMP RANGE
DS12887	24PDIP MODULE	DS12887	0°C TO +70°C

PIN DESCRIPTION:

GND, VCC –

DC power is provided to the device on these pins. VCC is the +5V input. When 5V are applied within normal limits, the device is fully accessible and data can be written and read. When VCC is below 4.25V typical, reads and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage. As VCC falls below 3V typical, the RAM and timekeeper are switched over to an internal lithium energy source. The timekeeping function maintains an accuracy of _1 minute per month at +25_C, regardless of the voltage input on the VCC pin.

SELECT BITS REGISTER A				t_{PI} PERIODIC INTERRUPT RATE	SQW OUTPUT FREQUENCY
RS3	RS2	RS1	RS0		
0	0	0	0	None	None
0	0	0	1	3.90625ms	256Hz
0	0	1	0	7.8125ms	128Hz
0	0	1	1	122.070µs	8.192kHz
0	1	0	0	244.141µs	4.096kHz
0	1	0	1	488.281µs	2.048kHz
0	1	1	0	976.5625µs	1.024kHz
0	1	1	1	1.953125ms	512Hz
1	0	0	0	3.90625ms	256Hz
1	0	0	1	7.8125ms	128Hz
1	0	1	0	15.625ms	64Hz
1	0	1	1	31.25ms	32Hz
1	1	0	0	62.5ms	16Hz
1	1	0	1	125ms	8Hz
1	1	1	0	250ms	4Hz
1	1	1	1	500ms	2Hz

PERIODIC INTERRUPT RATE AND SQUARE WAVE OUTPUT FREQUENCY:

MOT (Mode Select)

– The MOT pin offers the flexibility to choose between two bus types. When connected to VCC, Motorola bus timing is selected. When connected to GND or left disconnected, Intel bus timing is selected. The pin has an internal pull down resistance of approximately 20k_.

SQW (Square-Wave Output) –

The SQW pin can output a signal from one of 13 taps provided by the 15 internal divider stages of the RTC. Programming can change the frequency of the SQW pin. Register A, as shown in Table 1. The SQW signal can be turned on and off using the SQWE bit in Register B. The SQW signal is not available when VCC is less than 4.25V, typically.

AD0–AD7 (Multiplexed Bi-directional Address/Data Bus) –

Multiplexed buses save pins because address information and data information time-share the same signal paths. The addresses are present during the first portion of the bus cycle and the same pins and signal paths are used for data in the second portion of the cycle. Address/data multiplexing does not slow the access time of the DS12887 since the bus change from address to data occurs during the internal RAM access time. Addresses must be valid prior to the falling edge of AS/ ALE, at which time the DS12887 latches the address from AD0 to AD6. Valid write data must be present and held stable during the latter portion of the DS or WR pulses. In a read cycle the DS12887 outputs 8 bits of data during the latter portion of the DS or RD pulses. The read cycle is terminated and the bus returns to a high-impedance state as DS transitions low in the case of Motorola timing or as RD transitions high in the case of Intel timing.

AS (Address Strobe Input) –

A positive-going address-strobe pulse serves to de-multiplex the bus. The falling edge of AS/ALE causes the address to be latched within the DS12887. The next rising edge that occurs on the AS bus clears the address regardless of whether CS is asserted. Access commands should be sent in pairs.

DS (Data Strobe or Read Input) –

The DS/ RD pin has two modes of operation depending on the level of the MOT pin. When the MOT pin is connected to VCC, Motorola bus timing is selected. In this mode, DS is a positive pulse during the latter portion of the bus cycle and is called Data Strobe. During read cycles, DS signifies the time that the DS12887 is to drive the bi-directional bus. In write cycles the trailing edge of DS causes the DS12887 to latch the written data. When the MOT pin is connected to GND, Intel bus timing is selected. In this mode the DS pin is called Read (RD). RD identifies the time period when the DS12887 drives the bus with read data. The RD signal is the same definition as the output-enable (OE) signal on a typical memory.

R/W (Read/Write Input) –

The R/W pin also has two modes of operation. When the MOT pin is connected to VCC for Motorola timing, R/W is at a level that indicates whether the current cycle is a read or write. A read cycle is indicated with a high level on R/W while DS is high. A write cycle is indicated when R/W is low during DS. When the MOT pin is connected to GND for Intel timing, the R/W signal is an active-low signal called WR. In this mode, the R/W pin has the same meaning as the write-enable signal (WE) on generic RAMs.

CS (Chip-Select Input) –

The chip select signal must be asserted low for a bus cycle in the DS12887 to be accessed. CS must be kept in the active state during DS and AS for Motorola timing

and during RD and WR for Intel timing. Bus cycles that take place without asserting CS latch addresses but no access occur. When VCC is below 4.25V, the DS12887 internally inhibits access cycles by internally disabling the CS input. This action protects both the RTC data and RAM data during power outages.

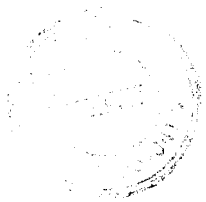
IRQ (Interrupt Request Output) –

The IRQ pin is an active-low output of the DS12887 that can be used as an interrupt input to a processor. The IRQ output remains low as long as the status bit causing the interrupt is present and the corresponding interrupt-enable bit is set. To clear the IRQ pin, the processor program normally reads the C register. The RESET pin also clears pending interrupts. When no interrupt conditions are present, the IRQ level is in the high-impedance state. Multiple interrupting devices can be connected to an IRQ bus. The IRQ bus is an open drain output and requires an external pull-up resistor.

RESET (Reset Input) –

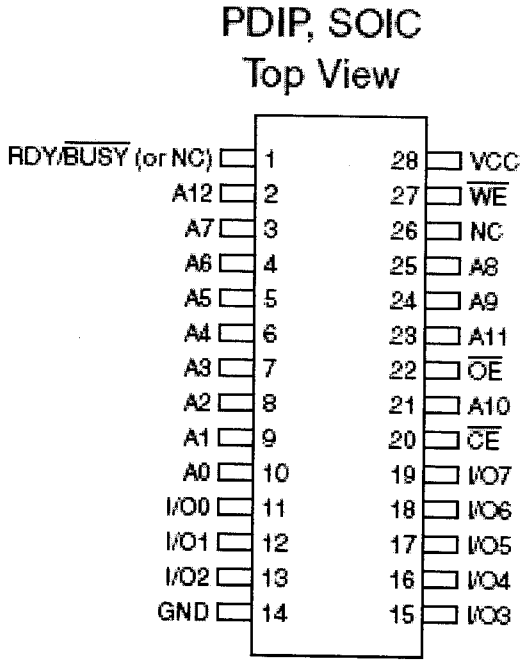
The RESET pin has no affect on the clock, calendar, or RAM. On power-up, the RESET pin can be held low for a time to allow the power supply to stabilize. The amount of time that RESET is held low is dependent on the application. However, if RESET is used on power-up, the time DS12887. RESET is low should exceed 200ms to ensure that the internal timer that controls the DS12887 on power up has timed out. When RESET is low and VCC is above 4.25V, the following occurs:

A) Periodic Interrupt Enable (PEI) bit is cleared to 0.
B) Alarm Interrupt Enable (AIE) bit is cleared to 0.
C) Update Ended Interrupt Flag (UF) bit is cleared to 0.
D) Interrupt Request Status Flag (IRQF) bit is cleared to 0.
E) Periodic Interrupt Flag (PF) bit is cleared to 0.
F) The device is not accessible until RESET is returned high.
G) Alarm Interrupt Flag (AF) bit is cleared to 0.
H) H. IRQ pin is in the high impedance state.
I) Square-Wave Output Enable (SQWE) bit is cleared to 0.
J) Update Ended Interrupt Enable (UIE) is cleared to 0.



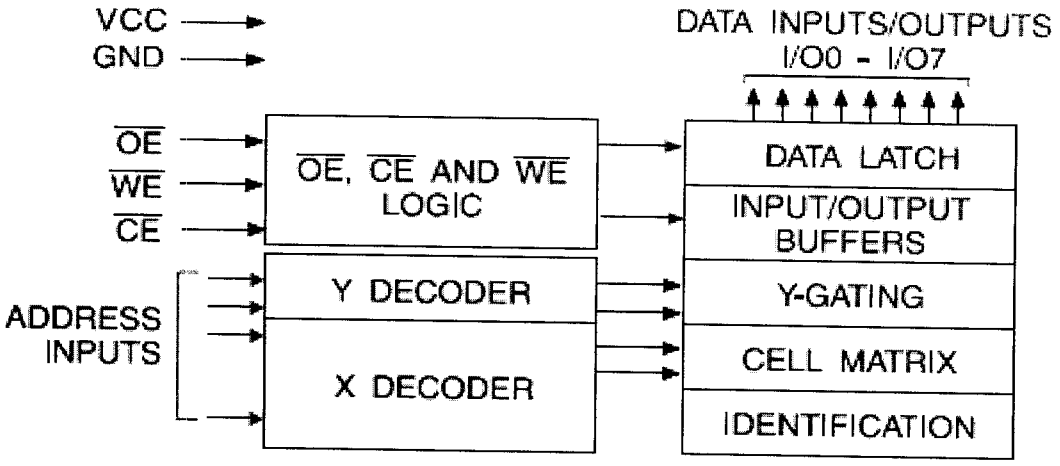
In a typical application RESET can be connected to VCC. This connection allows the DS12887 to go in and out of power fail without affecting any of the control registers.

ELECTRICALLY ERASABLE PROGRAMMABLE READ ONLY MEMORY (EEPROM)



Pin Configurations

- A0 - A12 Addresses
- CE Chip Enable
- OE Output Enable
- WE Write Enable
- I/O0 - I/O7 Data Inputs/Outputs
- RDY/BUSY Ready/Busy Output
- NC No Connect
- DC Don't Connect



Block Diagram

Absolute Maximum Ratings*

- Temperature under Bias ..... -55°C to +125°C
- Storage Temperature ..... -65°C to +150°C
- All Input Voltages (including NC Pins)

with Respect to Ground .....-0.6V to +6.25V

- All Output Voltages

with Respect to Ground .....-0.6V to VCC + 0.6V

- Voltage on OE and A9

with Respect to Ground .....-0.6V to +13.5V

Device Operation

READ:

The AT28C64 is accessed like a Static RAM. When CE and OE are low and WE is high, the data stored at the memory location determined by the address pins is asserted on the outputs. The outputs are put in a high impedance state whenever CE or OE is high. This dual line control gives designers increased flexibility in preventing bus contention.

BYTE WRITE:

Writing data into the AT28C64 is similar to writing into a Static RAM. A low pulse on the WE or CE input with OE high and CE or WE low (respectively) initiates a byte write. The address location is latched on the falling edge of WE (or CE); the new data is latched on the rising edge. Internally, the device performs a self-clear before write. Once a byte write has been started, it will automatically time itself to completion. Once a programming operation has been initiated and for the duration of tWC, a read operation will effectively be a polling operation.

FAST BYTE WRITE:

The AT28C64E offers a byte write time of 200 μs maximum. This feature allows the entire device to be rewritten in 1.6 seconds.

READY/BUSY:

Pin 1 is an open drain RDY/BUSY output that can be used to detect the end of a write cycle.

RDY/BUSY is actively pulled low during the write cycle and is released at the completion of the write. The open drain connection allows for OR-tying of several devices to the same RDY/BUSY line. The RDY/BUSY pin is not connected for the AT28C64X.

DATA POLLING:

The AT28C64 provides DATA Polling to signal the completion of a write cycle. During a write cycle, an attempted read of the data being written results in the complement of that data for I/O7 (the other outputs are indeterminate). When the write cycle is finished, true data appears on all outputs.

WRITE PROTECTION:

Inadvertent writes to the device are protected against in the following ways: (a) VCC sense – if VCC is below 3.8V (typical) the write function is inhibited; (b) VCC power on delay – once VCC has reached 3.8V the device will automatically time out 5 ms (typical) before allowing a byte write; and (c) write inhibit – holding any one of OE low, CE high or WE high inhibits byte write cycles.

Operating Modes

Mode	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	I/O
Read	V_{IL}	V_{IL}	V_{IH}	D_{OUT}
Write[2]	V_{IL}	V_{IH}	V_{IL}	D_{IN}
Standby/Write Inhibit	V_{IH}	X[1]	X	High Z
Write Inhibit	X	X	V_{IH}	
Write Inhibit	X	V_{IL}	X	
Output Disable	X	V_{IH}	X	High Z
Chip Erase	V_{IL}	V_H[3]	V_{IL}	High Z

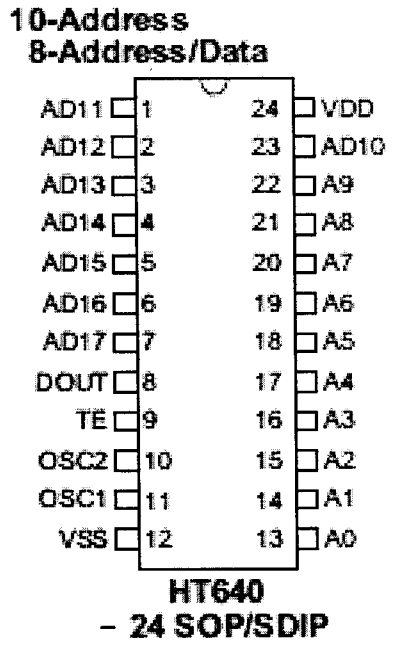
CHIP CLEAR:

The contents of the entire memory of the AT28C64 may be set to the high state by the CHIP CLEAR operation. By setting CE low and OE to 12 volts, the chip is cleared when a 10 msec low pulse is applied to WE.

DEVICE IDENTIFICATION:

An extra 32 bytes of EEPROM memory are available to the user for device identification. By raising A9 to 12 □0.5V and using address locations 1FE0H to 1FFFH the additional bytes may be written to or read from in the same manner as the regular memory array.

FM TRANSMITTER



Pin Description

Pin Name	I/O	Internal Connection	Description
A0~A11	I	TRANSMISSION GATE	Input pins for address A0~A11 setting They can be externally set to VDD, VSS, or left open.
AD10~AD17	I	TRANSMISSION GATE	Input pins for address/data (AD10~AD17) setting They can be externally set to VDD, VSS, or left open.
D12~D17	I	CMOS IN Pull-low	Input pins for data (D12~D17) setting and transmission enable (active high) They an be externally set to VDD or left open (see Note).
DOUT	O	CMOS OUT	Encoder data serial transmission output
LED	O	NMOS OUT	LED transmission enable indicator
TE	I	CMOS IN Pull-low	Transmission enable, active high (see Note).
OSC1	I	OSCILLATOR	Oscillator input pin
OSC2	O	OSCILLATOR	Oscillator output pin
VSS	I	—	Negative power supply (GND)
VDD	I	—	Positive power supply

Notes: D12~D17 are data input and transmission enable pins of the HT6187/HT6207/HT6247.
TE is the transmission enable pin of the HT600/HT640/HT680.

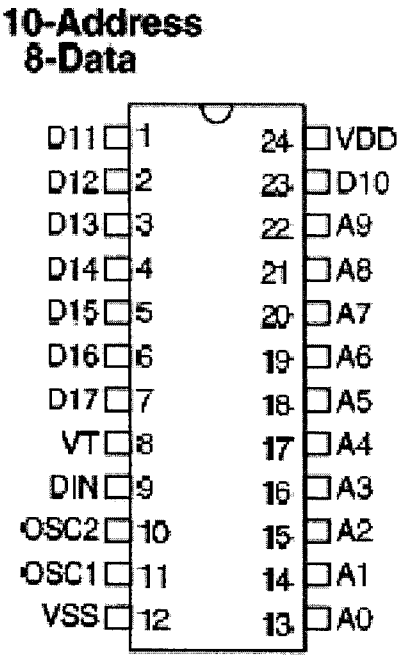
Electrical Characteristics

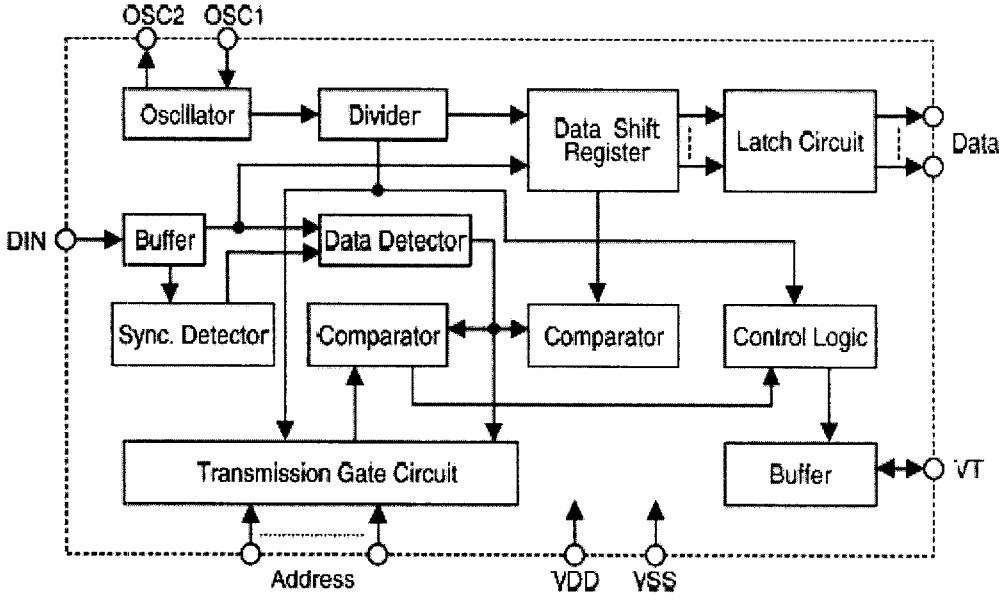
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	—	12	V
I_{STB}	Standby Current	3V	Oscillator stops	—	0.1	1	μA
		12V		—	2	4	μA
I_{DD}	Operating Current	5V	No load f_{OSC}=100kHz	—	250	500	μA
		12V		—	1200	2400	μA
I_{LED}	LED Sink Current	5V	V_{LED}=0.5V	1.5	3	—	mA
I_{DOUT}	Output Drive Current	5V	V_{OH}=0.9V_{DD} (Source)	-0.6	-1.2	—	mA
		5V	V_{OL}=0.1V_{DD} (Sink)	0.6	1.2	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IH}	"H" Input Voltage	—	—	0.8V_{DD}	—	VDD	V
V_{IL}	"L" Input Voltage	—	—	0	—	0.2V_{DD}	V
f_{OSC}	Oscillator Frequency	10V	R_{OSC}=330kΩ	—	100	—	kHz
R_{TE}	TE Pull-low Resistance	5V	V_{TE}=5V	—	1.5	3	MΩ
R_{DATA}	D12~D17 Pull-low Resistance	5V	V_{DATA}=5V	—	1.5	3	MΩ

FM RECEIVER





BLOCK DIAGRAM

Pin Description

Pin Name	I/O	Internal Connection	Description
A0~A17	I	TRANSMISSION GATE	Input pins for address A0~A17 setting They can be externally set to VDD. VSS. or left open.
D10~D17	O	CMOS OUT	Output data pins
DIN	I	CMOS IN	Serial data input pin
VT	O	CMOS OUT	Valid transmission, active high
OSC1	I	OSCILLATOR	Oscillator input pin
OSC2	O	OSCILLATOR	Oscillator output pin
VSS	I	—	Negative power supply (GND)
VDD	I	—	Positive power supply

LCD DISPLAY

FUNCTIONAL DESCRIPTION OF THE CONTROLLER IC

REGISTERS:

The controller IC has two 8 bit registers, an instruction register (IR) and a data register (DR). The IR stores the instruction codes and address information for display data RAM (DD RAM) and character generator RAM (CG RAM). The IR can be written, but not read by the MPU.

The DR temporally stores data to be written to /read from the DD RAM or CG RAM. The data written to DR by the MPU, is automatically written to the DD RAM or CG RAM as an internal operation.

When an address code is written to IR, the data is automatically transferred from the DD RAM or CG RAM to the DR. data transfer between the MPU is then completed when the MPU reads the DR. likewise, for the next MPU read of the DR, data in DD RAM or CG RAM at the address is sent to the DR automatically. Similarly, for the MPU write of the DR, the next DD RAM or CG RAM address is selected for the write operation.

The register selection table is as shown below:

RS	R/W	Operation
0	0	IR write as an internal operation
0	1	Read busy flag (DB7) and address counter (DB0 to DB6)
1	0	DR write as an internal operation (DR to DD RAM or CG RAM)
1	1	DR read as an internal operation (DD RAM or CG RAM to DR)

BUSY FLAG:

When the busy flag is1, the controller is in the internal operation mode, and the next instruction will not be accepted.

When RS = 0 and R/W = 1, the busy flag is output to DB7.

The next instruction must be written after ensuring that the busy flag is 0.

ADDRESS COUNTER:

The address counter allocates the address for the DD RAM and CG RAM read/write operation when the instruction code for DD RAM address or CG RAM address setting, is input to IR, the address code is transferred from IR to the address counter. After writing/reading the display data to/from the DD RAM or CG RAM, the address counter increments/decrements by one the address, as an internal operation. The data of the address counter is output to DB0 to DB6 while R/W = 1 and RS = 0.

DISPLAY DATA RAM (DD RAM)

The characters to be displayed are written into the display data RAM (DD RAM), in the form of 8 bit character codes present in the character font table. The extended capacity of the DD RAM is 80 x 8 bits i.e. 80 characters.

CHARATCER GENERATOR ROM (CG ROM)

The character generator ROM generates 5 x 8 dot 5 x 10 dot character patterns from 8 bit character codes. It generates 208, 5 x 8 dot character patterns and 32, 5 x 10 dot character patterns.

CHARACTER GENERATOR RAM (CG RAM)

In the character generator RAM, the user can rewrite character patterns by program. For 5 x 8 dots, eight character patterns can be written, and for 5 x 10 dots, four character patterns can be written.

INTERFACING THE MICROPROCESSOR / CONTROLLER:

The module, interfaced to the system, can be treated as RAM input/output, expanded or parallel I/O.

Since there is no conventional chip select signal, developing a strobe signal for the enable signal (E) and applying appropriate signals to the register select (RS) and read/write (R/W) signals are important.

The module is selected by gating a decoded module – address with the host – processor's read/write strobe. The resultant signal, applied to the LCDs enable (E) input, clocks in the data.

The 'E' signal must be a positive going digital strobe, which is active while data and control information are stable and true. The falling edge of the enable signal enables the data / instruction register of the controller. All module timings are referenced to specific edges of the 'E' signal. The 'E' signal is applied only when a specific module transaction is desired.

The read and write strobes of the host, which provides the 'E' signals, should not be linked to the module's R/W line. An address bit which sets up earlier in the host's machine cycle can be used as R/W.

When the host processor is so fast that the strobes are too narrow to serve as the 'E' pulse

a. Prolong these pulses by using the hosts 'Ready' input
b. Prolong the host by adding wait states
c. Decrease the Hosts Crystal frequency.

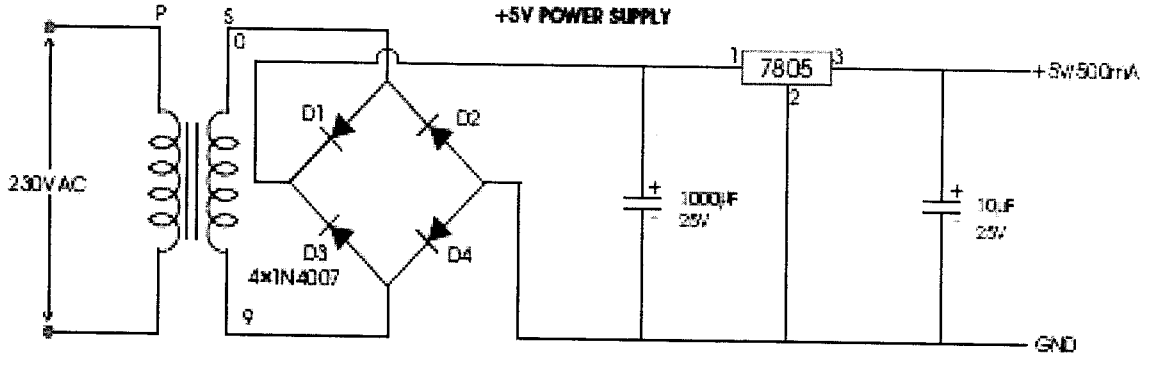
Inspite of doing the above mentioned, if the problem continues, latch both the data and control information and then activate the 'E' signal

When the controller is performing an internal operation he busy flag (BF) will set and will not accept any instruction. The user should check the busy flag or should provide a delay of approximately 2ms after each instruction. The module presents no difficulties while interfacing slower MPUs. The liquid crystal display module can be interfaced, either to 4-bit or 8-bit MPUs.

For 4-bit data interface, the bus lines DB4 to DB7 are used for data transfer, while DB0 to DB3 lines are disabled. The data transfer is complete when the 4-bit data has been transferred twice.

The busy flag must be checked after the 4-bit data has been transferred twice. Two more 4-bit operations then transfer the busy flag and address counter data. For 8-bit data interface, all eight-bus lines (DB0 to DB7) are used.

POWER SUPPLY



This unit consists of transformers, rectifiers, filters and regulators. The input AC voltage diode rectifier then provides a full wave rectified, typically 230Vrms is connected to a transformer to obtain the desired step-down AC voltage. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a DC voltage. This resulting DC voltage usually has some ripple or AC voltage variations. A regulator circuit can use this DC input to provide DC voltage that not only has less ripple voltage but also remains the same DC value even if the DC voltage varies, or when the load connected to the output DC voltage changes. The power supply unit comprises of two circuits, one providing a +5V supply and the other providing +12V and -12V supply. The circuit diagrams for the same are shown in the Figures.

APPENDIX B: SOFTWARE

ALGORITHM

Step 1: Switch on the power supply.

Step 2: Initialize LCD and serial port.

Step 3: Display time day and date.

Step 4: Check if the tag is sensed. If yes activate relay and sound alarm, display time day and date, else go to step 3.

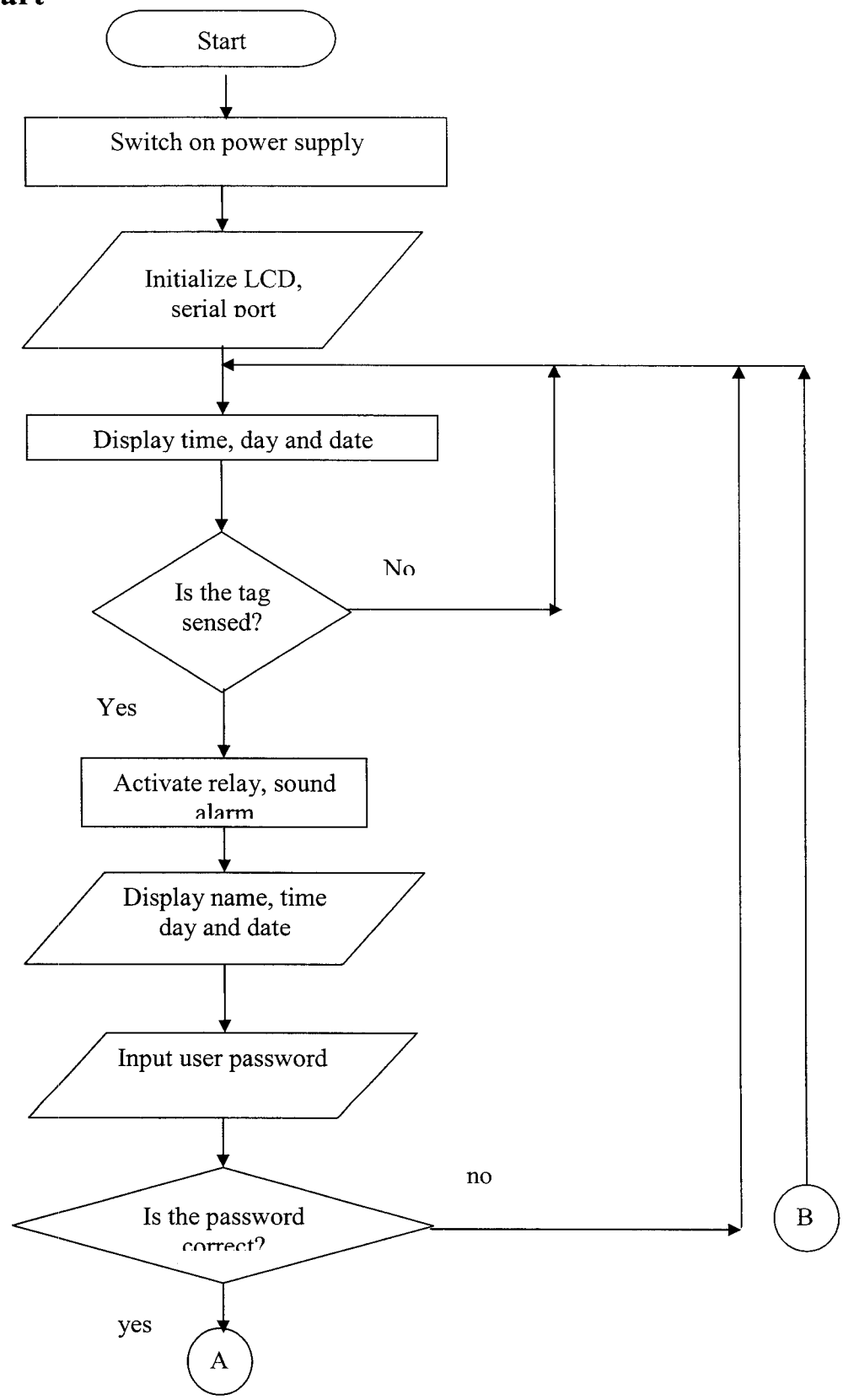
Step 5: Input user password.

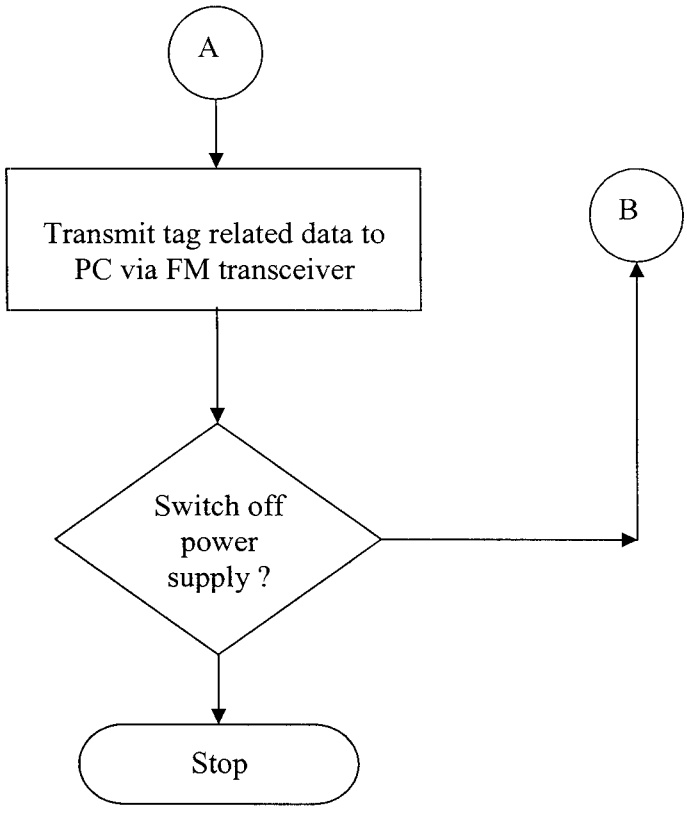
Step 6: Check if user password is correct. If yes transmit data to PC, else go to step 3.

Step 7: If supply is switched off go to step 8 else go to step 3.

Step 8: Stop.

Flowchart


Start
Switch on power supply
Initialize LCD, serial port
Display time, day and date
Is the tag sensed?
No
Yes
Activate relay, sound alarm
Display name, time day and date
Input user password
Is the password correct?
no
yes
A
B


A
Transmit tag related data to PC via FM transceiver
B
Switch off power supply ?
Stop

PROGRAM

```
#include <reg52.h>
#include<intrins.h>
void disp_month_ser_transm();
void disp_year_ser_transm();
void disp_day_ser_transm();
void display_ser_transm(unsigned char);
void ser_out_st_transm(unsigned char *,unsigned char);
void txm_transm();

void transm(unsigned char da);
void fun();
void recharge(unsigned int,unsigned int);
void htd(unsigned int);
void amount_check();
void chk_pos_amount();
void chk_inc_dec();
void scan_amount();
void read_amount(unsigned int);
void write_amount(unsigned int);

void welcome();
void cal(unsigned int);
void display();
void current();
void remain();
void enter_amount();
void scan_all();
void insufficient();
void enter();
void key_chk();
void thank();
void trans();
void incorrect();

void disp_month_ser();
void disp_year_ser();
```

```
void disp_day_ser();
void display_ser(unsigned char);

void txm();
void ser_out(unsigned char);
void ser_out_st(unsigned char *,unsigned char);
void time_display();
void ser_init();
void lcd_ini();
void datum(unsigned char);
void command(unsigned char);
void lcd_dis(unsigned char*,unsigned char);
void delay(unsigned int);
void all_disp();
void time_disp();
void disp_month();
void disp_day();
void disp_year();
void display1(unsigned char);
void transmit();
void tmr_ini();
void tx_support();
void scan();
void chk_pos();
void set_time();
void am_pm_disp();
sbit buzzer=P3^3;// buzzer is conected. but, it will be referred as relay,
throught the program
sbit relay=P3^4;
//sbit wdt=P1^5;// the output will be connected to the WDT(Watch Dog
Timer)
xdata unsigned char command1 _at_ 0xc100;// address to maintain the status
EN=1;RS=0
xdata unsigned char command2 _at_ 0x4100;// address to maintain both EN
& RS=0
xdata unsigned char datum1 _at_ 0xc000;// address to maintain EN=RS=1
xdata unsigned char datum2 _at_ 0x4000;// address to maintain EN=0, RS=1
// the memory locations starting with '8' belong to RTC
xdata unsigned char sec _at_ 0x8000;//seconds
xdata unsigned char min _at_ 0x8002;//minutes
xdata unsigned char hou _at_ 0x8004;//hours
```

```
xdata unsigned char dow _at_ 0x8006;//day of week(sunday=1)
xdata unsigned char dom _at_ 0x8007;// date of month
xdata unsigned char mon _at_ 0x8008;// month
xdata unsigned char yea _at_ 0x8009;//year
xdata unsigned char rega _at_ 0x800a;//Oscillator control Register
xdata unsigned char regb _at_ 0x800b;//
xdata unsigned char key _at_ 0xc400;//keys connected
//xdata unsigned char most _at_ 0x8050;
//xdata unsigned char least _at_ 0x8051;
idata unsigned char select_flag,alarm_flag,msb,lsb,alarm_set;
idata unsigned char hours,minutes,seconds,bits,pos,g,ent,summ;
unsigned char
year,date,month,day,flag2,ref,ref2,dd,cc,c,e,pass[5],detail_flag;
unsigned char
hou_on,hou_tn,min_on,min_tn,sec_on,sec_tn,var,date_tn,date_on,year_tn,ye
ar_on,either_flag;
idata unsigned int trans_count,gg;
unsigned char xdata *add;
unsigned char pw1[14];
unsigned char chk;
unsigned char inc,address_pass;
bit chk_bit,am;
idata unsigned int gg,amt,temp_amt,temp_amt1,temp_amt2;
unsigned char xdata *add;
unsigned char pass[5],enter_var,pos_ptr;
idata unsigned char tenthous,thous,hund,ten,one,hund_r;
idata unsigned int tenthous_r,thous_r;
unsigned char pass_word[5];
void main()
{
        ref=bits=buzzer=relay=0;// to maintain the initial values
//      wdt=0;// both the WDT & the backlight for LCD are connected in the
active low mode
        lcd_ini();// to initialise the LCD display
        tmr_ini();// to initialise the timer 0
   ser_init();
        rega=0x20;// both these lines are used to re-initialise the internal
oscillator of the RTC
        regb=0x04;
        summ=0;// to indicate that the buzzer is not ON*/
        command(0x01);
```

```
        while (1)
        {
        command(0x01);// to clear the display
inval:
         all_disp();
     if(chk_bit==1 && SBUF==13)
     {
      chk_bit=0;
      chk=0;
          ES=0;// to disable serial
      if(pw1[8]=='C' && pw1[9]=='5')
      {
       buzzer=1;
       welcome();
       buzzer=0;
            command(0x01);
            command(0x80);
            lcd_dis("   TITUS      ",16);
       time_display();
       delay(65000);
       delay(65000);
       inc=0;
       enter();
       key_chk();
       if(pass_word[0]==8 && pass_word[1]==6 && pass_word[2]==5 &&
pass_word[3]==2)       fun();
       else       incorrect();
      }
     else if(pw1[8]=='C' && pw1[9]=='4')
     {
/*     buzzer=1;
      welcome();
      buzzer=0;
           command(0x01);
           command(0x80);
           lcd_dis("    POORNI    ",16);
      time_display();
      delay(65000);
      delay(65000);

      inc=0;
```

```
        enter();
        key_chk();
        if(pass_word[0]==6 && pass_word[1]==5 && pass_word[2]==2 &&
 pass_word[3]==8)        fun();
        else        incorrect();*/
        relay=1;
        delay(65000);
        delay(65000);
        relay=0;
       }
      else if(pw1[8]=='C' && pw1[9]=='3')
      {
       buzzer=1;
       welcome();
       buzzer=0;
            command(0x01);
            command(0x80);
            lcd_dis("    VEENA     ",16);
       time_display();
       delay(65000);
       delay(65000);
       inc=0;
       enter();
       key_chk();
       if(pass_word[0]==2 && pass_word[1]==6 && pass_word[2]==5 &&
 pass_word[3]==8)        fun();
       else        incorrect();
      }
     else if(pw1[8]=='C' && pw1[9]=='7')
      {
       buzzer=1;
       welcome();
       buzzer=0;
            command(0x01);
            command(0x80);
            lcd_dis("    VINITH     ",16);
       time_display();
       delay(65000);
       delay(65000);
       inc=0;
       enter();
```

```
      key_chk();
      if(pass_word[0]==5 && pass_word[1]==2 && pass_word[2]==6 &&
pass_word[3]==8)       fun();
      else       incorrect();
      }
     else if(pw1[8]=='E' && pw1[9]=='E')
      {
      buzzer=1;
      welcome();
      buzzer=0;
           command(0x01);
           command(0x80);
           lcd_dis("   SIVAGURU     ",16);
      time_display();
      delay(65000);
      delay(65000);
      inc=0;
      enter();
      key_chk();
      if(pass_word[0]==5 && pass_word[1]==6 && pass_word[2]==2 &&
pass_word[3]==8)       fun();
      else       incorrect();
      }
           command(0x01);// to clear the display
           command(0x0c);
           ES=1;// to enable serial
     pass_word[0]=pass_word[1]=pass_word[2]=pass_word[3]=0x30;
     }
        e=key;
        if (e==0xbf)            set_time();
   goto inval;
       }
}
void key_chk()
{
       unsigned char abc;
  address_pass=0xc0;
chkhere:
       abc=key;// key' is the address where the keys are connected( see
declaration)// the value for the key pressed, is stored in a temporary register
get:
```

```
        if (flag2==1)// if true, it means that, a timeout has occured
        {
         goto ent_over;
        }
        e=key;
        if (e!=0xff) goto get;// to check whether the key has been released
        e=abc;
        delay(10000);
        if (e==0xfe)
    {
    pass_word[0]=pass_word[1]=pass_word[2]=pass_word[3]=0;
    inc=0;
    enter();
    address_pass=0xc0;
    }
        else if (e==0xfd)goto ent_over;//2 'ENTER' key
        else if (e==0xfb)
        {
    pass_word[inc]=2;
    command(address_pass);
//    datum(pass_word[inc]+0x30);
    datum('*');
    address_pass++;
    inc++;
        }
        else if (e==0xf7)
        {
    pass_word[inc]=5;
    command(address_pass);
//    datum(pass_word[inc]+0x30);
    datum('*');
    address_pass++;
    inc++;
   }
        else if (e==0xef)
        {
    pass_word[inc]=6;
    command(address_pass);
//    datum(pass_word[inc]+0x30);
    datum('*');
    address_pass++;
```

```
    inc++;
   }
       else if (e==0xdf)
       {
    pass_word[inc]=8;
    command(address_pass);
//   datum(pass_word[inc]+0x30);
    datum('*');
    address_pass++;
    inc++;
   }
  goto chkhere;
ent_over:;
}
void time_display()
{
       seconds=sec;//to get the seconds
       minutes=min;// to get the minutes
       hours=hou;//to get the hours
       am_pm_disp();// to display AM or PM
       command(0xcc);
       disp_day();// to display the 'DOW'
       datum(' ');
}
void set_time()// routine to get the new data for the realtime
{
       command(0x01);
       command(0x80);
       lcd_dis(" Real Time Mode ",16);
       delay(65535);
//     delay(65535);
//     delay(65535);
       command(0x01);
       all_disp();
enter:
       hou_tn=hours/10;// to load the temporary registers with the real time
data
       hou_on=hours%10;
       min_tn=minutes/10;
       min_on=minutes%10;
       sec_tn=seconds/10;
```

```
        sec_on=seconds%10;
        date_tn=date/10;
        date_on=date%10;
        year_tn=year/10;
        year_on=year%10;
        pos=0x80;
        command(pos);
        either_flag=1;
        ref=0;// these three lines are must, if u want to get out of the loop, if no
function is performed for about two minutes
        ref2=0;
        g=select_flag=detail_flag=0;
        flag2=5;
        scan();
        flag2=0;
        if (e==0xfe||pos==0) goto prev;
        else if (e==0xfd)// means that 'ENTER' key is pressed
        {
        hours=((hou_tn*10)+hou_on)|var;// this is to load the values, if
'ENTER' key is pressed
        minutes=(min_tn*10)+min_on;
        seconds=(sec_tn*10)+sec_on;
        date=(date_tn*10)+date_on;
        year=(year_tn*10)+year_on;
        if (date==0)goto ent;// because there is no date, termed "zero"
        if ((month==4||month==6||month==9||month==11)&&date>30)//
because these months have only 30 days
        {
ent:
        command(0x01);
        command(0x80);
        lcd_dis("Date/Month Wrong",16);
        delay(65535);
        delay(65535);
        command(0x01);
        command(0x80);
        display1(date);// to display the date
        command(0x85);
        disp_month();// to display the month
        command(0x8c);
        disp_year();
```

```
        am_pm_disp();
        command(0xcc);
        disp_day();
        goto enter;
        }
        else if (month==2)
        {
        if (date>29) goto ent;
        else if ((year%4!=0)&&date>28)goto ent;// this is to find whether it
was a leap year
        }
        rega=0xe0;// to avoid the the update provided by the internal oscillator,
while loading the new data to RTC
        regb=0x84;
        sec=seconds;
        min=minutes;
        hou=hours;
        mon=month;
        yea=year;
        dom=date;// Date of month
        dow=day;// Day of week
//      hou_alarm=min_alarm=sec_alarm=0;
        }
prev:
        rega=0x20;// to enable the oscillator
        regb=0x04;// re-initialise the 12Hr mode and to get the hex data mode
        lcd_ini();// to re-initialise the LCD display
        command(0x01);
        delay(49000);// this delay is must, inorder to give the RTC some time
to write the changed data into the internal registers
}
void all_disp()// the routine in which all the parameters will be displayed
{
        date=dom;// to get the particular date in a month
        month=mon;//to get the month
        year=yea;// to get the year
        day=dow;// to get the Day Of Week(DOW)
        command(0x80);
        display1(date);// to display the date
        command(0x85);
        disp_month();// to display the month
```

```
        command(0x8c);
        disp_year();// to display the year
        seconds=sec;//to get the seconds
        minutes=min;// to get the minutes
        hours=hou;//to get the hours
        am_pm_disp();// to display AM or PM
        command(0xcc);
        disp_day();// to display the 'DOW'
        datum(' ');
        command(0x8a);
        if (alarm_set==1) datum('*');// means that alarm has already been set
        else if (alarm_set==0) datum(' ');
}
void tx_support()// routine used to re-initialise the SCON register
{
        delay(40);
        SCON=0x58;// to enable both the receiver & transmitter
        delay(2000);
}
void scan()// used to detect the keypress. Regardless of whatever the program
may be. This is hard to understand without the hardware
{
        unsigned char abc;
        command(0x0f);
chkhere:
        abc=key;// key' is the address where the keys are connected( see
declaration)// the value for the key pressed, is stored in a temporary register
get:
        if (flag2==1)// if true, it means that, a timeout has occured
        {
        pos=flag2=0;
        goto ent_over;
        }
        e=key;
        if (e!=0xff) goto get;// to check whether the key has been released
        e=abc;
//      delay(10000);
        delay(1000);
        if (e==0xfe)goto ent_over;//1 'ESC' key. Can be used, when the user
makes some mistakes
       else if (e==0xfd)goto ent_over;//2 'ENTER' key
```

```
else if (e==0xfb)//3 'RIGHT' shift- to change the cursor position
{
 g=ref2=0;
 if (ent==1) goto chkhere;
 if (either_flag==1)
 {
  if (pos==0x8f) pos=0xbf;
  else if (pos==0xc9) pos=0xcc;
  pos++;
  if (pos==0x86) pos=0x8e;
  else if (pos==0x82) pos=0x85;
  if (pos==0xc2 || pos==0xc5 || pos==0xc8) pos++;
  else if (pos>0xcc) pos=0xcc;
 }
 else
 {
  if (pos==0xc3) pos=0xc2;
  pos++;
 }
 command(pos);
 goto chkhere;
}
else if (e==0xf7)//4 'LEFT' shift- to change the cursor position
{
g=ref2=0;
if (ent==1) goto chkhere;
if (either_flag==1)// means that the user wants to set time
{
 if (pos==0xc0) pos=0x90;
 else if (pos==0x80) pos=0x81;
 pos--;
 if (pos==0xcb) pos=0xc9;
 else if (pos==0xc2 || pos==0xc5 || pos==0xc8) pos--;
 else if (pos==0x8d) pos=0x85;
 else if (pos==0x84) pos=0x81;
}
else// may be used, when the user wants to enter password
{
 if (pos==0xc0) pos=0xc1;
 pos--;
}
```

```
        command(pos);
        goto chkhere;
        }
        else if (e==0xef)//5 DECREMENT
        {
        g=ref2=0;
        chk_pos();
        goto chkhere;
        }
        else if (e==0xdf)//6 INCREMENT
        {
        g=ref2=0;
        chk_pos();
        goto chkhere;
        }
/*      else if (e==0xbf)//7 used in the main routine, to set time / Alarm
        {

        }
        else if (e==0x7f)//8 'transmit' key(used only in the main routine)
        {

        }*/
        else goto chkhere;
        goto chkhere;
ent_over:;
}
void chk_pos()// it is not easy to understand this section without the hardware
by ur side
{
       if (pos==0x80)// the postions & the functions vary according to the
mode in which they are operated
        {
        if (e==0xdf)// means that, INCREMENT key has been pressed. This
condition may be used frequently
        {
        date_tn++;
        if (date_on>=2&&date_tn>2)date_tn=0;
        else if ((date_on==1||date_on==0)&&date_tn>3) date_tn=0;
        }
```

```
        else if (e==0xef)// means that, DECREMENT key has been pressed. It
may be used frequently
         {
          date_tn--;
          if (date_on>1&&date_tn==0xff) date_tn=2;
          else if (date_tn==0xff) date_tn=3;
         }
         datum(date_tn+0x30);
        }
        else if (pos==0x81)
        {
         if (e==0xdf)
         {
          if (date_tn==3&&date_on==0)
          {
           date_on=1;
           goto bef;
          }
          else if (date_tn==3&&date_on==1)
          {
           date_on=0;
           goto bef;
          }
          date_on++;
          if (date_tn==3&&date_on>1) date_on=1;
          else if (date_on>9) date_on=0;
         }
         else if (e==0xef)
         {
          date_on--;
          if (date_tn==3&&date_on==0xff) date_on=1;
          else if (date_on==0xff) date_on=9;
         }
bef:
         datum(date_on+0x30);
        }
        else if (pos==0x85)
        {
         if (e==0xdf)
         {
          month++;
```

```
 if (month>12) month=1;
 }
 else if (e==0xef)
 {
 month--;
 if (month==0||month==0xff) month=12;
 }
 disp_month();
 goto end;
}
else if (pos==0x8e)
{
 if (e==0xdf) year_tn++;
 else if (e==0xef) year_tn--;
 datum(year_tn+0x30);
}
else if (pos==0x8f)
{
 if (e==0xdf) year_on++;
 else if (e==0xef) year_on--;
 datum(year_on+0x30);
 goto end;
}
else if (pos==0xc0)
{
 if (either_flag==1)
 {
  if (hou_on>2) hou_tn=0;
  else if (hou_tn==1)hou_tn=0;
  else if (hou_tn==0)hou_tn=1;
  datum(hou_tn+0x30);
 }
}
else if (pos==0xc1)
{
 if (either_flag==1)// means that the user wants to set time
 {
  if (e==0xdf)
  {
   if (hou_tn==1 && hou_on==2 ) hou_on=0xff;
   hou_on+=1;
```

```
  if (hou_on>9) hou_on=0;
  }
  else if (e==0xef)
  {
  if (hou_on==0) hou_on=0x0a;
  hou_on-=1;
  if (hou_tn==1 && hou_on>2) hou_on=2;
  }
  datum(hou_on+0x30);
  goto end;
 }
}
else if (pos==0xc3)
{
 if (either_flag==1)
 {
 if (e==0xdf)
 {
  if (min_tn==5) min_tn=0xff;
  min_tn+=1;
 }
 else if (e==0xef)
 {
  if (min_tn==0) min_tn=6;
  min_tn-=1;
 }
 datum(min_tn+0x30);
 goto end;
 }
}
else if (pos==0xc4)
{
 if (e==0xdf)
 {
 min_on+=1;
 if (min_on>9) min_on=0;
 }
 else if (e==0xef)
 {
 if (min_on==0) min_on=0x0a;
 min_on-=1;
```

```
 }
 datum(min_on+0x30);
 goto end;
}
else if (pos==0xc6)
{
 if (e==0xdf)
 {
  if (sec_tn==5) sec_tn=0xff;
  sec_tn+=1;
 }
 else if (e==0xef)
 {
  if (sec_tn==0) sec_tn=6;
  sec_tn-=1;
 }
 datum(sec_tn+0x30);
 goto end;
}
else if (pos==0xc7)
{
if (e==0xdf)
{
sec_on+=1;
if (sec_on==10) sec_on=0;
}
else if (e==0xef)
{
if (sec_on==0) sec_on=0x0a;
sec_on-=1;
}
datum(sec_on+0x30);
goto end;
}
else if (pos==0xc9)// if AM or PM has to be set
{
if (var==0) var=0x80;
else if (var==0x80) var=0;
if (var==0) datum('A');
if (var==0x80) datum('P');
datum('M');
```

```
        goto end;
        }
        else if (pos==0xcc)
        {
        if (e==0xdf)
        {
        day++;
        if (day>7) day=1;
        }
        else if (e==0xef)
        {
        day--;
        if (day==0) day=7;
        }
        disp_day();
        }
end:
        command(pos);
}
void disp_month()// As the name indicates, it will be used only to display the
month
{
        if (month==1) lcd_dis("Jan",3);
        else if (month==2) lcd_dis("Feb",3);
        else if (month==3) lcd_dis("Mar",3);
        else if (month==4) lcd_dis("Apr",3);
        else if (month==5) lcd_dis("May",3);
        else if (month==6) lcd_dis("Jun",3);
        else if (month==7) lcd_dis("Jul",3);
        else if (month==8) lcd_dis("Aug",3);
        else if (month==9) lcd_dis("Sep",3);
        else if (month==0x0a) lcd_dis("Oct",3);
        else if (month==0x0b) lcd_dis("Nov",3);
        else if (month==0x0c) lcd_dis("Dec",3);
}
void disp_year()// used to display the year
{
        datum('2');
        datum('0');
        display1(year);
}
```

```
void disp_day()// used to display the day of week
{
        if (day==1)  lcd_dis("Sun",3);
        else if (day==2) lcd_dis("Mon",3);
        else if (day==3) lcd_dis("Tue",3);
        else if (day==4) lcd_dis("Wed",3);
        else if (day==5) lcd_dis("Thu",3);
        else if (day==6) lcd_dis("Fri",3);
        else if (day==7) lcd_dis("Sat",3);
}
void time_disp()// to display the time
{
        command(0xc0);
        display1(hours);// to display hours
        datum(':');
        display1(minutes);// to display minutes
        datum(':');
        display1(seconds);// to display seconds
}
void am_pm_disp()
{
        command(0xc8);// to provide the display address
        datum(' ');
        if (hours>=0x81)// to findout whether it is AM or PM( if the hours
value>80h, it indicates that it is PM)
        {
        var=0x80;// indicating PM
        hours=hours-0x80;// to get the exact time
        datum('P');// to display P
  am=1;
        }
        else
        {
        datum('A');// to display A
  am=0;
        var=0;// indicating AM
        }
        datum('M');
        time_disp();// to display time
}
```

```
void display1(unsigned char k)// General display routine to display the
numbers after hexadecimal to decimal
{
        dd=k;
        datum((dd/10)+0x30);
        datum((dd%10)+0x30);
}
void lcd_dis(unsigned char *dis,unsigned char rr) //general display routine.//
Refer to the program "ENERGY".
{
        for (dd=0;dd<rr;dd++)
        {
         datum(dis[dd]);
        }
}
void tmr_ini()
{
        TMOD=0x21;// to use, timer 1 in mode 2 & timer 0 in mode 1
        TH0=0x7c;// to get the desired time delay
        TL0=0xaf;
        EA=1;// to enable all the interrupts
        ET0=1;// to enable timer 0
        TR0=1;// to run the timer 0
}
void ser_int(void) interrupt 4 using 1
{
 if (RI)
  {
  chk_bit=1;
  pw1[chk++]=SBUF;
  RI=0;
  }
}
void timer0(void) interrupt 1 // refer to the program "new poultry"
{
        TR0=0;// to stop the timer
        c+=1;// to check for the desired time delay
        if (c==5)// means that the desired time delay has occured
        {
        c=0;// to clear the register
//      TR0=0;
```

```
//      if (wdt==1) wdt=0;// to give pulses for the watchdog timer
//      else if (wdt==0) wdt=1;
//      cc+=1;// the count value is used for backlight display control
        if (flag2==5)// means that the main loop is not under execution at
present
        {
        g+=1;
        if (g==99)// this one & the next loop are applied to keep the execution
track of the microcontroller in the main routine if no the user has left the loop
in between
        {
        ref2+=1;
        g=0;
        if (ref2==6)// means that the time out has occured & the loop should be
transfered to the main routine
        {
        ref2=g=0;
        flag2=1;
        }
        }
        }
        }
        TH0=0x5c;
        TL0=0xaf;
        TR0=1;
}
void lcd_ini()// to initialise the LCD display
{
        command(0x38);//TO ENABLE FUNCTION SET
        command(0x06);//ENTRY MODE SET
        command(0x0c);//DISPLAY ON/CURSER OFF
}
void command(unsigned char i)// to send the command for the LCD display
{
        command1=i;
        delay(50);
        command2=i;
        delay(50);
}
void datum(unsigned char j)// to send the data for the LCD display
{
```

```
        datum1=j;
        delay(50);
        datum2=j;
        delay(50);
}
void delay(unsigned int b)// general delay routine. The loop will continue, till
the value that was passed by the calling function becomes zero
{
        do b-=1;
        while (b!=0);
}
void ser_init()
{
        TMOD=0x21;
        TH1=0xfd;//e6 for 1200 baud rate,fd for 9600
        TR1=1;// to run the timer
        delay(255);
        SCON=0x58;// to enable both transmitting & receiving mode
        EA=1;// to enable all the interrupts
        ES=1;// to enable serial
}
void ser_out_st(unsigned char *st,unsigned char rr)
{
unsigned char w;
for (w=0;w<rr;w++)
{
 SBUF=st[w];
 tx_support();
}
}
void ser_out(unsigned char rr)
{
        SBUF=rr;
        tx_support();
}
void txm()
{
          display_ser(date);// to display the date
     ser_out(' ');
          disp_month_ser();// to display the month
     ser_out(' ');
```

```
        disp_year_ser();// to display the year
    ser_out(' ');
    display_ser(hours);
    ser_out(':');
    display_ser(minutes);
    ser_out(':');
    display_ser(seconds);
    ser_out(' ');
        if (am==1)   ser_out('P');
        else         ser_out('A');
    ser_out('M');
    ser_out(' ');
        disp_day_ser();// to display the 'DOW'
    ser_init();
}

void disp_month_ser()// As the name indicates, it will be used only to display
the month
{
     if (month==1) ser_out_st("Jan",3);
     else if (month==2) ser_out_st("Feb",3);
     else if (month==3) ser_out_st("Mar",3);
     else if (month==4) ser_out_st("Apr",3);
     else if (month==5) ser_out_st("May",3);
     else if (month==6) ser_out_st("Jun",3);
     else if (month==7) ser_out_st("Jul",3);
     else if (month==8) ser_out_st("Aug",3);
     else if (month==9) ser_out_st("Sep",3);
     else if (month==0x0a) ser_out_st("Oct",3);
     else if (month==0x0b) ser_out_st("Nov",3);
     else if (month==0x0c) ser_out_st("Dec",3);
}
void disp_year_ser()// used to display the year
{
     ser_out('2');
     ser_out('0');
     display_ser(year);
}
void disp_day_ser()// used to display the day of week
{
     if (day==1)  ser_out_st("Sun",3);
```

```
        else if (day==2) ser_out_st("Mon",3);
        else if (day==3) ser_out_st("Tue",3);
        else if (day==4) ser_out_st("Wed",3);
        else if (day==5) ser_out_st("Thu",3);
        else if (day==6) ser_out_st("Fri",3);
        else if (day==7) ser_out_st("Sat",3);
}
void display_ser(unsigned char k)// General display routine to display the
numbers after hexadecimal to decimal
{
        dd=k;
        ser_out((dd/10)+0x30);
        ser_out((dd%10)+0x30);
}
void enter()
{
  command(0x01);
  command(0x80);
  lcd_dis("ENTER  PASSWORD ",16);
  command(0xc0);
  lcd_dis("0000          ",16);
}
void welcome()
{
          command(0x01);
          command(0x80);
          lcd_dis("   WELCOME     ",16);
      delay(65000);
      delay(65000);
}
void thank()
{
          command(0x01);
          command(0x80);
          lcd_dis("   THANK YOU    ",16);
     delay(5000);
//      delay(65000);
}
void trans()
{
          command(0x01);
```

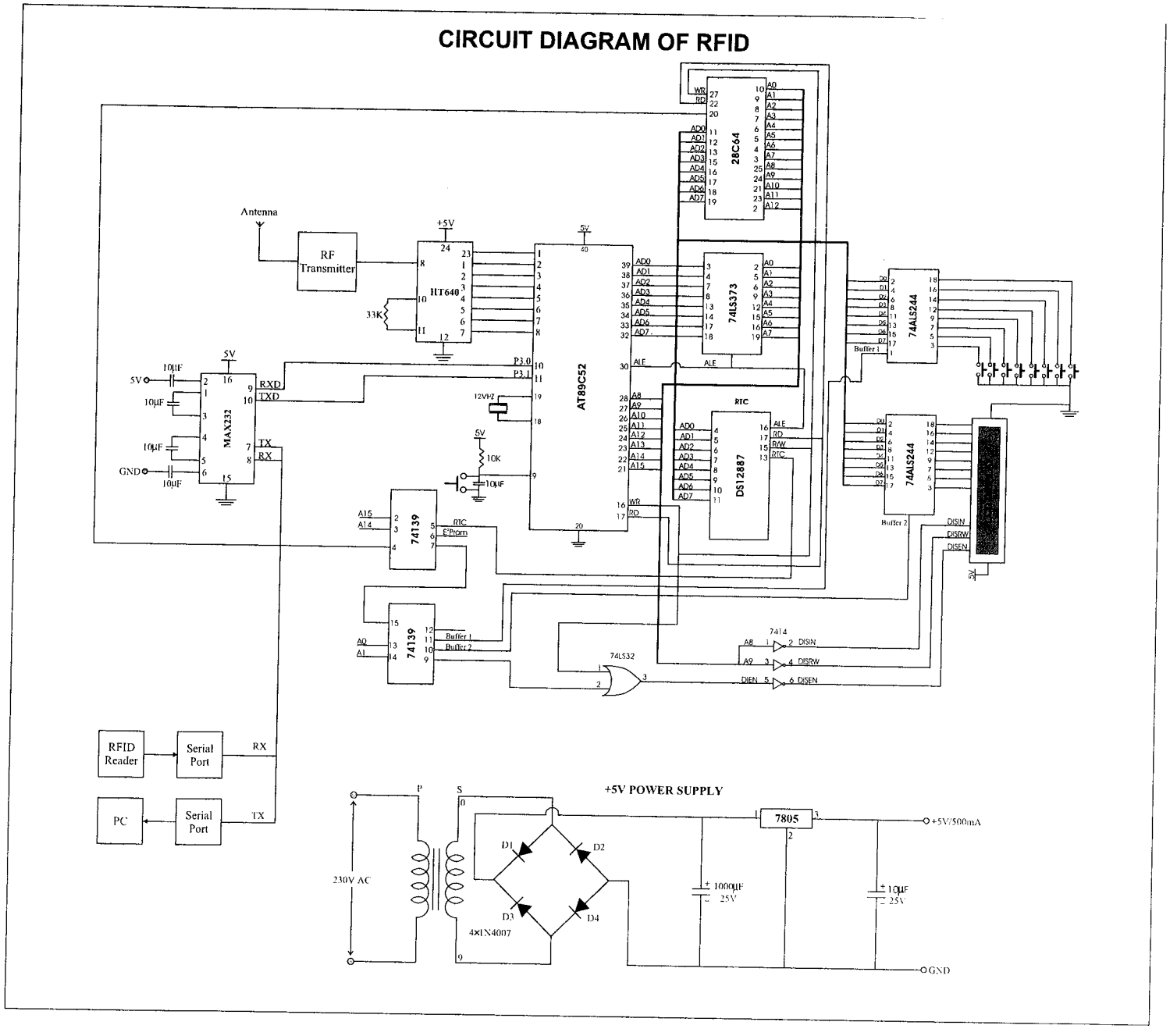
```
        command(0x80);
        lcd_dis("TRANSMITTING....",16);
}
void incorrect()
{
        command(0x01);
        command(0x80);
        lcd_dis("  INCORRECT  ",16);
        command(0xC0);
        lcd_dis("  PASSWORD  ",16);
    buzzer=1;
    delay(65000);
    delay(65000);
    buzzer=0;
}
void display()
{
     datum(thous+0x30);
     datum(hund+0x30);
     datum(ten+0x30);
     datum(one+0x30);
}
void htd(unsigned int hex_val)
{
tenthous=hex_val/0x2710;
tenthous_r=hex_val%0x2710;
thous=tenthous_r/0x3e8;
thous_r=tenthous_r%0x3e8;
hund=thous_r/0x64;
hund_r=thous_r%0x64;
ten=hund_r/0x0a;
one=hund_r%0x0a;
}
void fun()
{
    trans();
    transm(pw1[8]);
    transm(pw1[9]);
    txm_transm();
    thank();
```

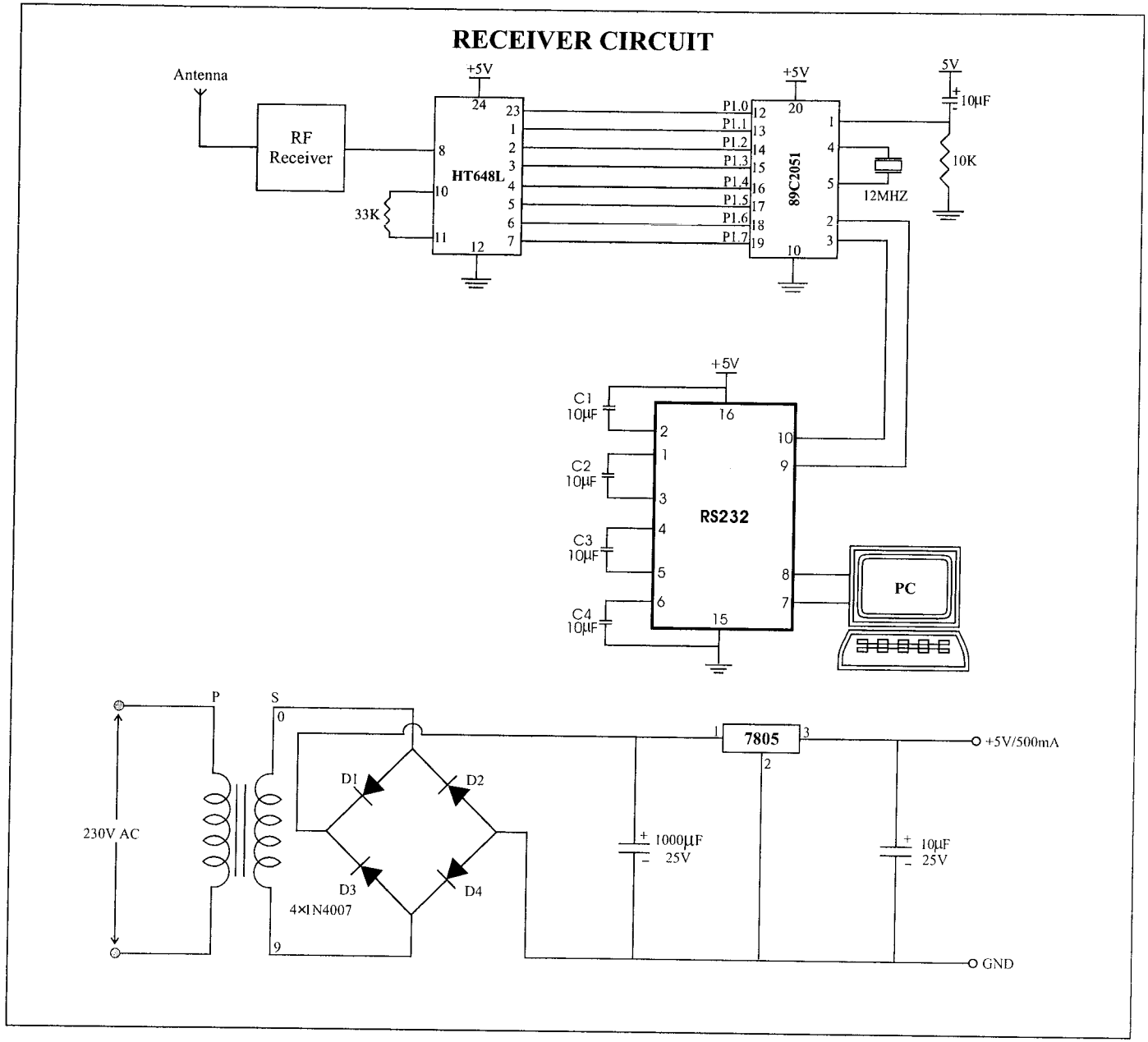
```
}
void transm(unsigned char da)
{
    P1=da;
    delay(20000);
    P1=0xff;
    delay(20000);
}
void txm_transm()
{
        display_ser_transm(date);// to display the date
     transm(' ');
        disp_month_ser_transm();// to display the month
     transm(' ');
        disp_year_ser_transm();// to display the year
     transm(' ');
     display_ser_transm(hours);
     transm(':');
     display_ser_transm(minutes);
     transm(':');
     display_ser_transm(seconds);
     transm(' ');
        if (am==1)   transm('P');
        else         transm('A');
     transm('M');
     transm(' ');
        disp_day_ser_transm();// to display the 'DOW'
     ser_init();
}

void disp_month_ser_transm()// As the name indicates, it will be used only to
display the month
{
      if (month==1) ser_out_st_transm("Jan",3);
      else if (month==2) ser_out_st_transm("Feb",3);
      else if (month==3) ser_out_st_transm("Mar",3);
      else if (month==4) ser_out_st_transm("Apr",3);
      else if (month==5) ser_out_st_transm("May",3);
      else if (month==6) ser_out_st_transm("Jun",3);
      else if (month==7) ser_out_st_transm("Jul",3);
      else if (month==8) ser_out_st_transm("Aug",3);
```

```
        else if (month==9) ser_out_st_transm("Sep",3);
        else if (month==0x0a) ser_out_st_transm("Oct",3);
        else if (month==0x0b) ser_out_st_transm("Nov",3);
        else if (month==0x0c) ser_out_st_transm("Dec",3);
}
void disp_year_ser_transm()// used to display the year
{
        transm('2');
        transm('0');
        display_ser_transm(year);
}
void disp_day_ser_transm()// used to display the day of week
{
        if (day==1)  ser_out_st_transm("Sun",3);
        else if (day==2) ser_out_st_transm("Mon",3);
        else if (day==3) ser_out_st_transm("Tue",3);
        else if (day==4) ser_out_st_transm("Wed",3);
        else if (day==5) ser_out_st_transm("Thu",3);
        else if (day==6) ser_out_st_transm("Fri",3);
        else if (day==7) ser_out_st_transm("Sat",3);
}
void display_ser_transm(unsigned char k)// General display routine to display
the numbers after hexadecimal to decimal
{
        dd=k;
        transm((dd/10)+0x30);
        transm((dd%10)+0x30);
}
void ser_out_st_transm(unsigned char *st,unsigned char rr)
{
 unsigned char w;
 for (w=0;w<rr;w++)
 {
  transm(st[w]);
 }
}
```

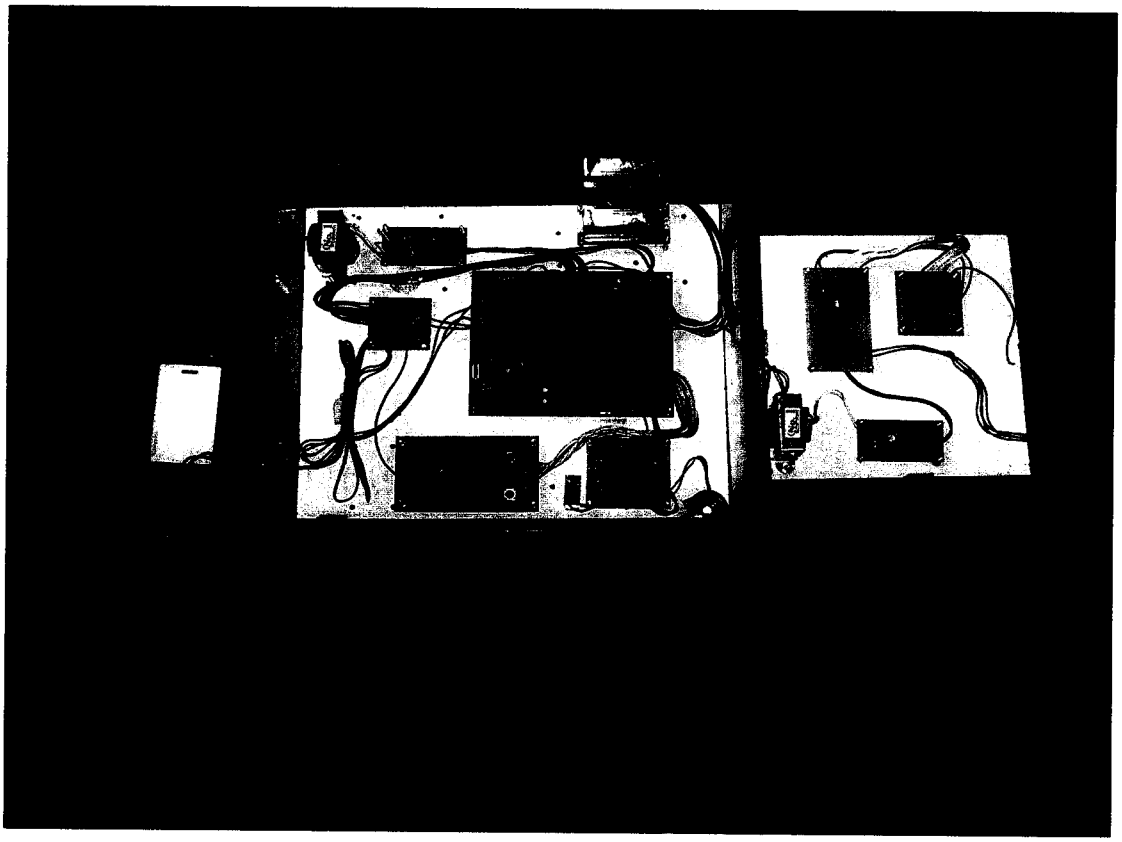
APPENDIX C: CIRCUIT DIAGRAM

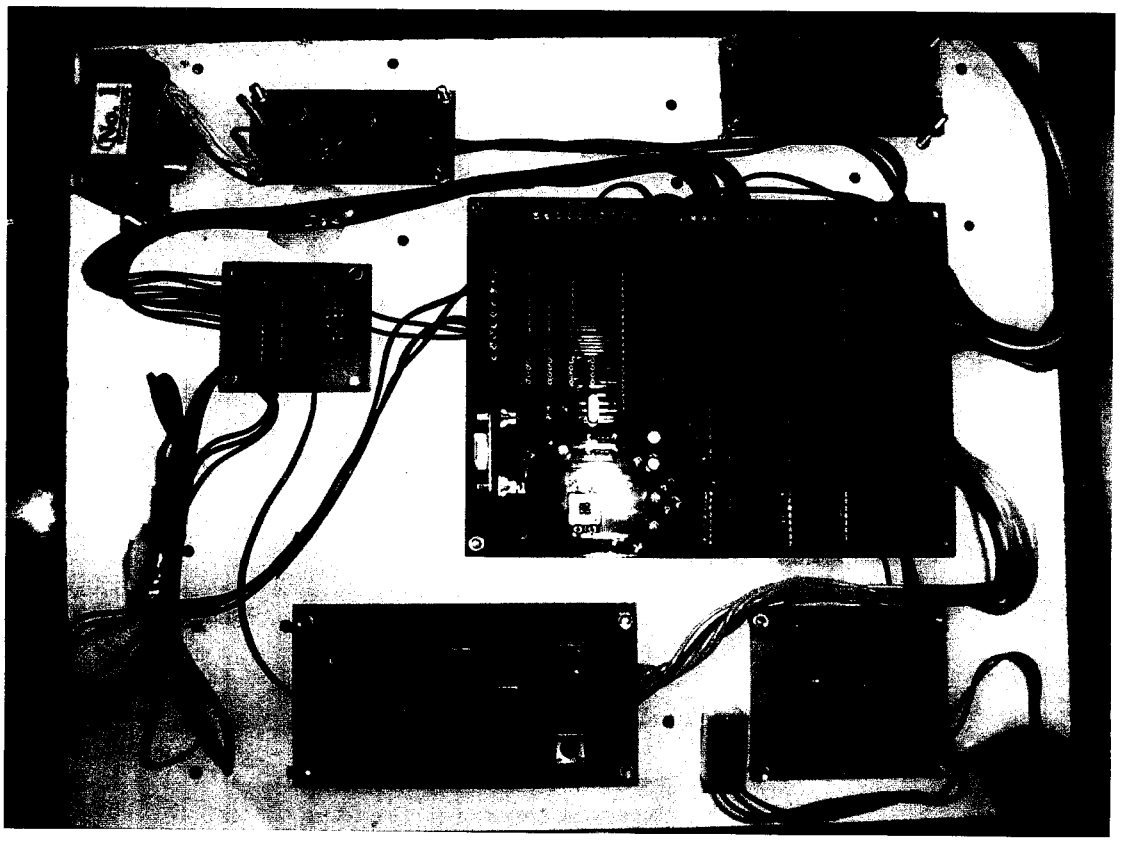
CIRCUIT DIAGRAM OF RFID




RECEIVER CIRCUIT
Antenna
RF Receiver
+5V
HT648L
33K
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
89C2051
12MHZ
10K
10µF
5V
C1 10µF
C2 10µF
C3 10µF
C4 10µF
RS232
PC
P
S
230V AC
D1
D2
D3
D4
4×1N4007
7805
+5V/500mA
1000µF 25V
10µF 25V
GND

PHOTOGRAPHS





REFERENCES

WEBSITES:

http://www.sparrel.com/

http://www.atmel.com/

http://www.rfid - handbook.com/

http://www.microchip.com/

http://www.philips.com/

http://www.rfidjournal.com/

http://www.aimglobal.org/