1.INTRODUCTION

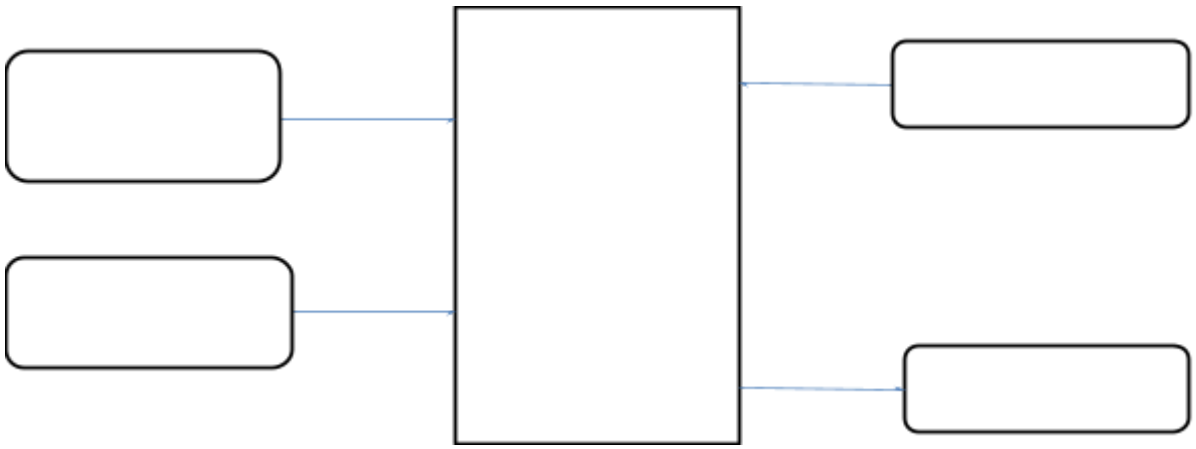
Two months before we all heard about the missing of a plane, but even after several days rescue operation doesn't became successful. Likewise there are so many cases around the globe about missing/crashed aeroplanes ,submarines and ships. The ultimate aim of this project is to design and develop a device which helps to retrieve the data from black box/captains deck to a safe place on the ground through air/water

2.BLOCK DIAGRAM AND EXPLANATION

This system consist of three main sections; a sensor module, MATLAB, Drone.

EMBEDDED SIDE [SENSOR MODULE]

The following block diagram shows the sensor module. It consist of a microcontroller (PIC16F877A), Accelerometer, Other optional sensors.



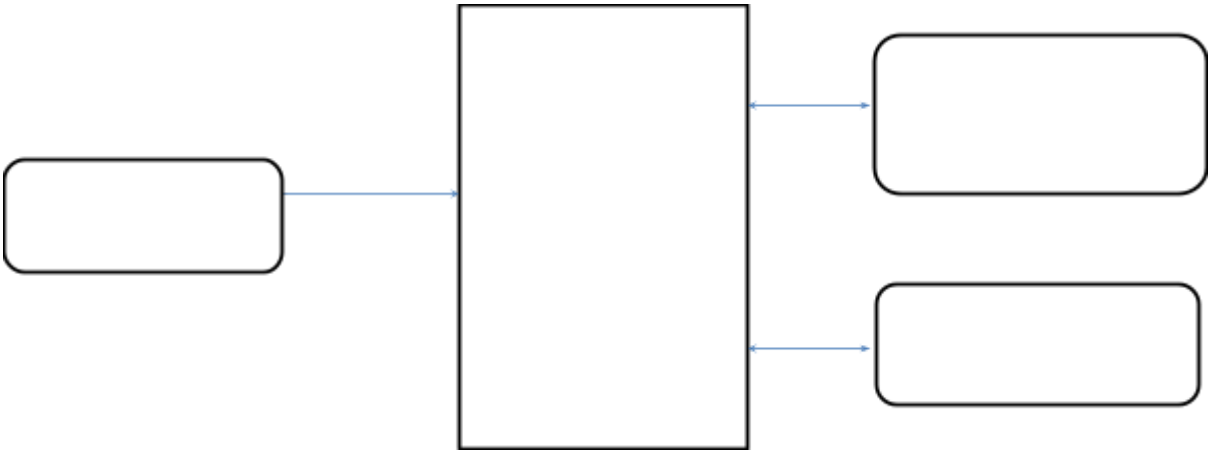
MICROCONTROLLER

Microcontroller is the most important part of the project. It is responsible for all the activities done inside the system except MATLAB side. Here the microcontroller used is microchip's PIC 16F877A.

ACCELEROMETER

An accelerometer is a device that measures proper acceleration. Proper acceleration is not the same as coordinate acceleration (rate of change of velocity).

MATLAB SIDE BLOCK DIAGRAM

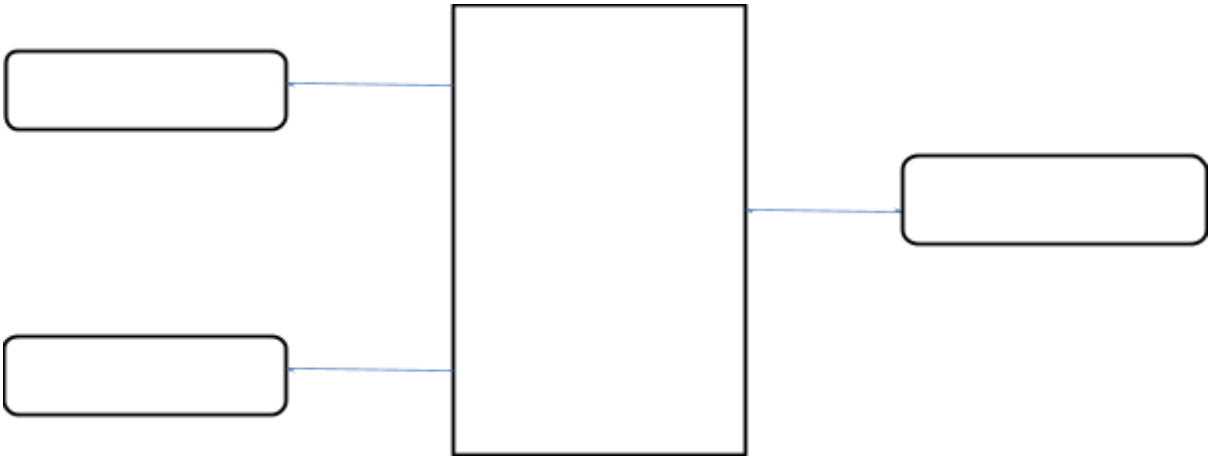


ZIGBEE

The XBee/ZBee (formerly known as Series 2 and Series 2 PRO) RF Modules were engineered to operate within the ZigBee protocol and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between remote devices. The modules operate within the 2.4 GHz frequency band.

Here it is used to send the data to the drone.

EMBEDDED SIDE [DRONE]



ZIGBEE

Here it is used to receive the data from MATLAB side.

DC MOTOR

To control the momment of the drone in different direction ,we use dc motor.

WORKING

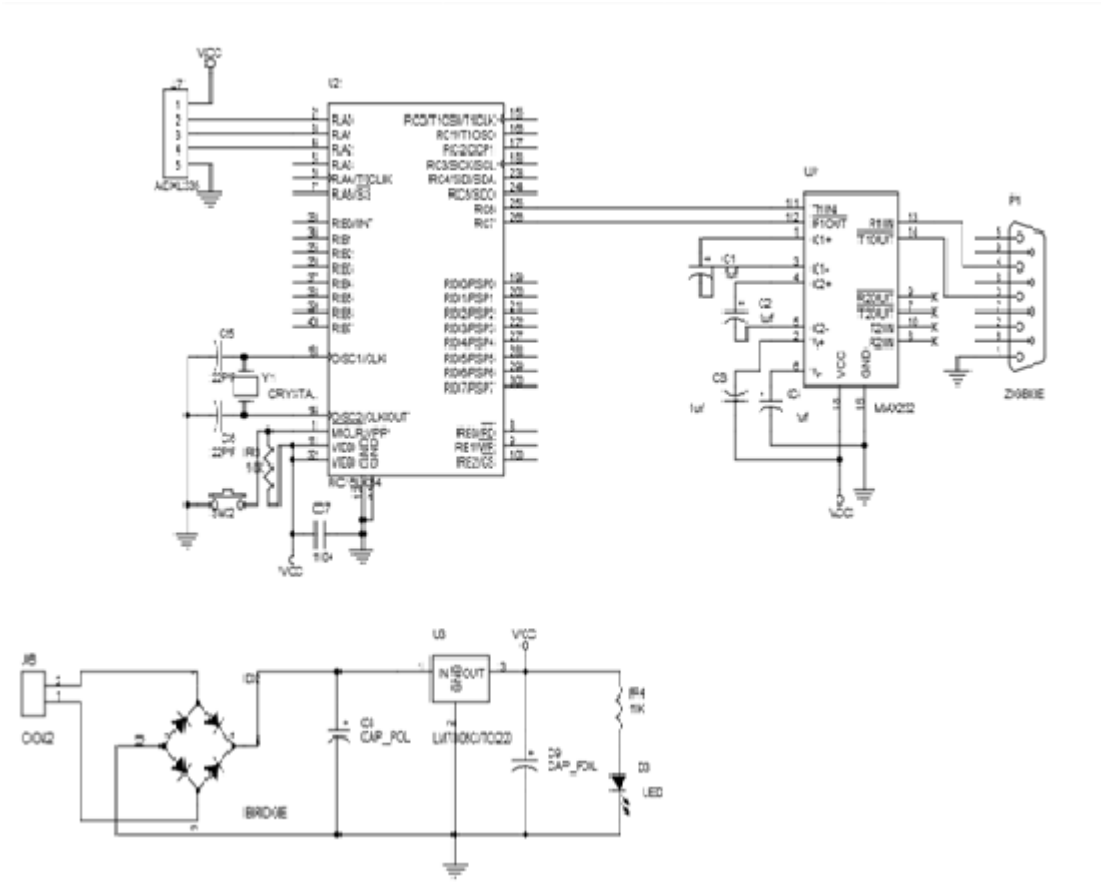
This is a system which can be use as an improoved version of Black box, as all we know Black box collect different data regarding a flight; So if anything bad happen, we will be able to know WHY and WHAT happend to the plane.

But the main problem is that, the black box will remain in the place even after the damage so it is not useful for a search and rescue operation. DARE solve this problem,it can be implemented inside any aeroplane. This device consist of two main block , Data acquisition and Drone/UAV Block; data acquisition section stores every data regarding the flight. When the parameters for a secure flight are compromised then the system detects it and flies back to a safe location with the data using a preloaded map.

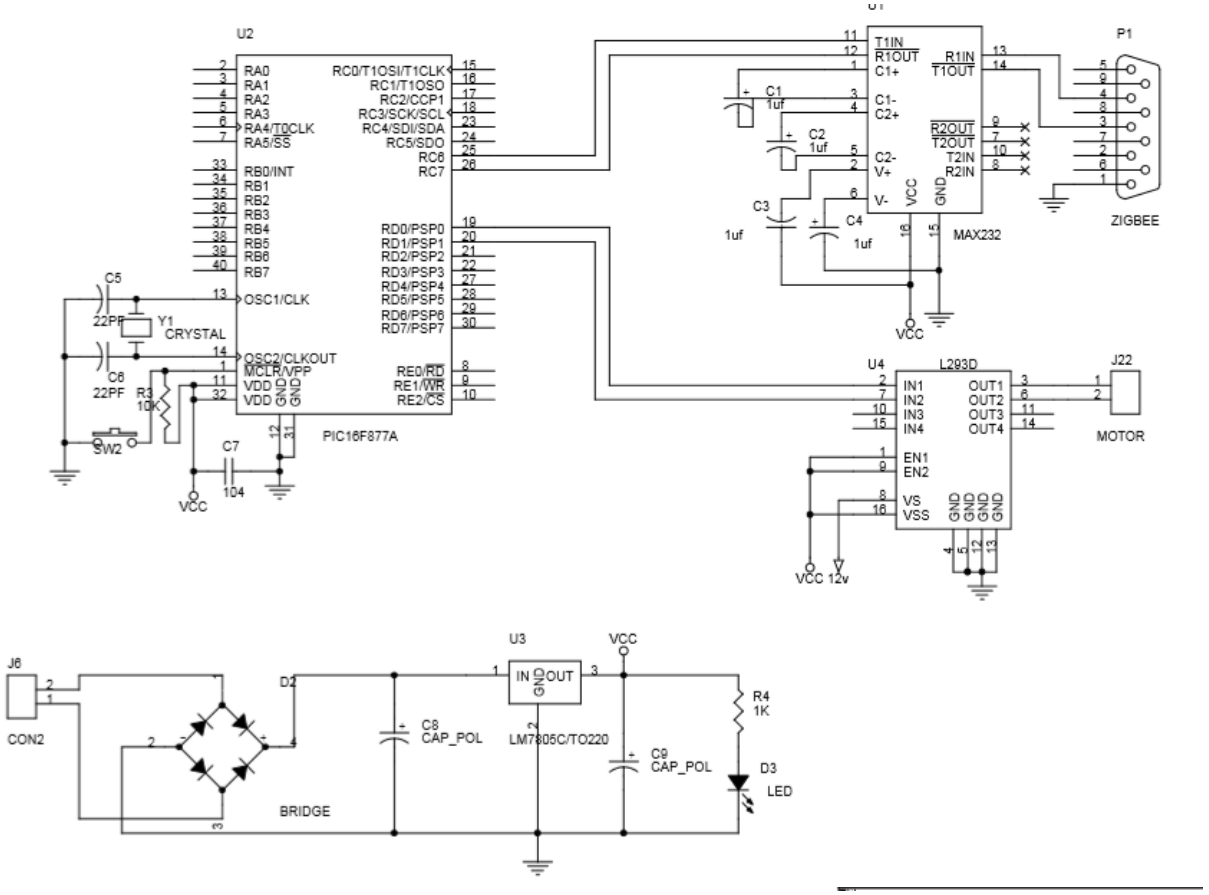
3.ARCHITECTURE

● CIRCUIT DIAGRAM

EMBEDDED SIDE



DRONE SIDE


U2
PIC16F877A
RA0
RA1
RA2
RA3
RA4/T0CLK
RA5/SS
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
OSC1/CLK
OSC2/CLKOUT
MCLR/VPP
VDD
VDD
RC0/T1OSI/T1CLK
RC1/T1OSO
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6
RC7
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
RE0/RD
RE1/WR
RE2/CS
C5
22PF
Y1
CRYSTAL
C6
22PF
R3
10K
SW2
C7
104
VCC
U1
MAX232
T1IN
R1OUT
C1+
C1-
C2+
C2-
V+
V-
R1IN
T1OUT
R2OUT
T2OUT
T2IN
R2IN
C1
1uf
C2
1uf
C3
1uf
C4
1uf
P1
ZIGBEE
U4
L293D
IN1
IN2
IN3
IN4
OUT1
OUT2
OUT3
OUT4
EN1
EN2
VS
VSS
J22
MOTOR
VCC 12v
J6
CON2
D2
BRIDGE
U3
LM7805C/TO220
IN
GND
OUT
C8
CAP_POL
C9
CAP_POL
R4
1K
D3
LED

MICROCONTROLLER

PIC micro controllers are low-cost computers-in-a-chip; they allow electronics designers and hobbyists add intelligence and functions that mimic big computers for almost any electronic product or project.

The programming of the system is done using a PIC micro controller 16F877. This powerful (200 nanosecond instruction execution) yet easy-to-program (only 35 single word instructions) CMOS FLASH-based 8-bit micro controller packs Microchip's powerful PIC® architecture into a 40-pin package and is upwards compatible with the PIC16C5X, PIC12CXXX and PIC16C7X devices. It is has five ports. I.e. port A, port B, port C, port D, port E. The PIC 16F877 has flash memory of 8K and Data memory of 368 bytes and Data EEPROM of 256 bytes.

WHY PIC IS USED?

Speed
High Performance RISC CPU
Instruction Set Simplicity
Integration Of Operation Features
Programmable Timer Options
Interrupt Control
EPROM/OTP/ROM Options
Inbuilt Modules
Low Power Consumption
Wide Operation Voltage Range :2.5to 6.0 Volt

Programmable Code Protection Mode

Power Saving Sleep Mode

PIC 16F877A- FEATURES High performance CPU

Only 35 instructions

All single cycle instruction except for program branches. Operating speed DC-20MHz,clock i/p DC-200ns instruction cycles

Up to 8Kx14 word of flash memory

Up to 368x8 bytes of data memory

Up to 256x8 bytes of EEPROM data memory

Interrupt compatibility

Power on reset

Power up timer and oscillator start up timer

Watch dog timer with its own chip RC oscillator for reliable operation

Programmable code protection power saving SLEEP mode

Low power, high speed CMOS FLASH/EEPROM technology

In circuit serial programming capability via two pins

Processor read write access to program memory

Single 5v in circuit serial programming capability

In circuit debugging via two pins

Wide operating voltage range

High sink or source current

Low power consumption

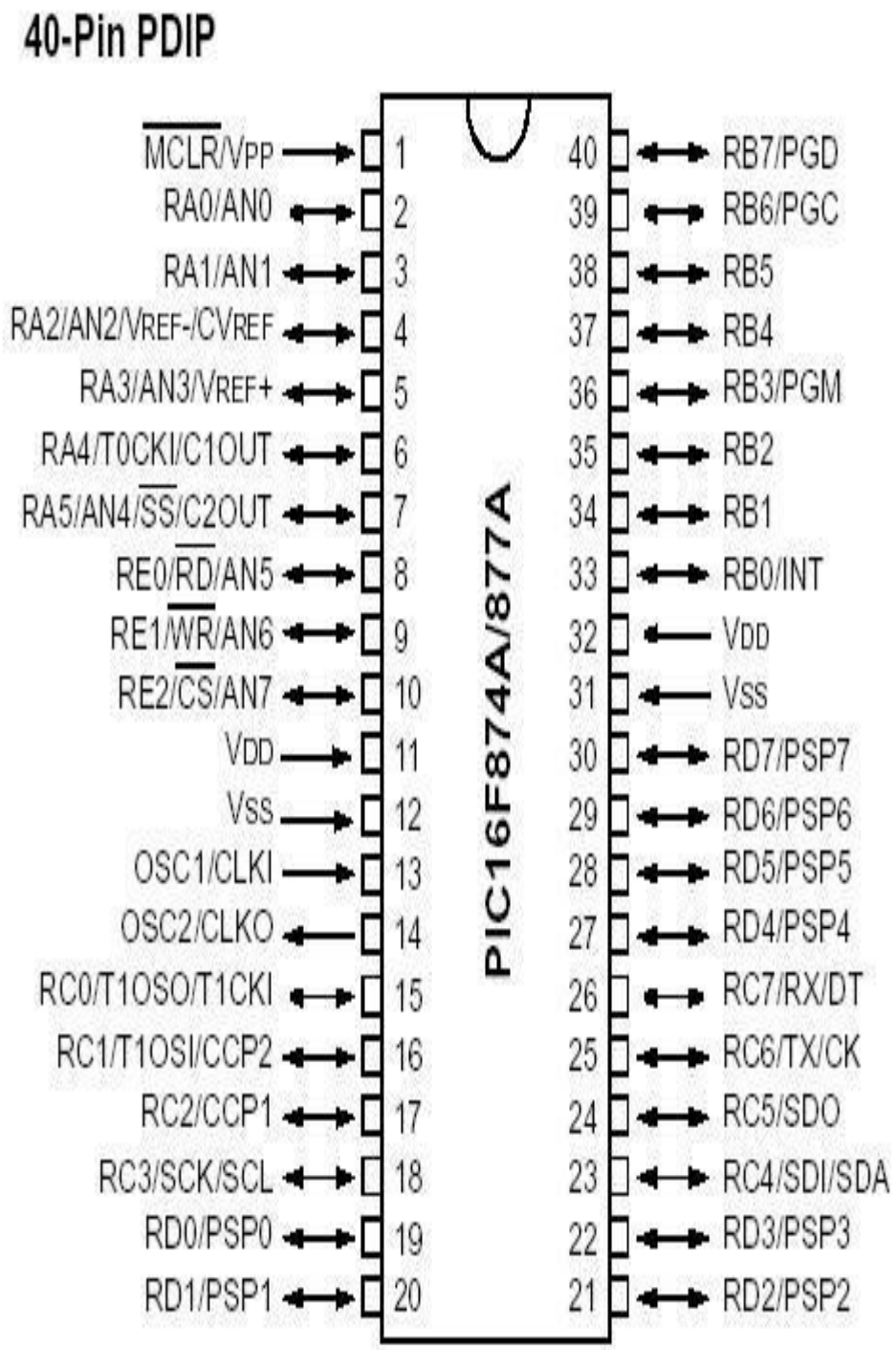


Fig:pin diagram of PIC 16F877A

PIC 16F877 PINOUT DESCRIPTION

Pin name	Dip pin #	Plccpin #	I/O type	Buffer type	description	qfp
Osc/clkin Osc/clkout	13 14	14 15	1 0	St/cmos -	Osc.crystal i/p. osc.crystal o/p	30 31
MCLR/Vpp	1	2	I/P	St	Master clear i/p or pgm o/p.	18
RA0/AN0 RA1/AN1 RA2/AN2/VREF- RA3/AN3/VREF+ RA4/TOCK1 RA5/SS/AN4	2 3 4 5 6 7	3 4 5 6 7 8	I/O I/O I/O I/O I/O I/O	TTL TTL TTL TTL SL TTL	Port A is bi directional i/p .	19 20 21 22 23 24
RB0/INT RB1 RB2 RB3/PGM	33 34 35 36	36 37 38 39	I/O I/O I/O I/O	TTL/ST1 TTL TTL TTL	Port B is a bi directional i/o port. Port b can	8 9 10 11

RB4	37	41	I/O	TTL	be s/w pgmed for pull up on	14
RB5	38	42	I/O	TTL		15
RB6/PGC	39	43	I/O	TTL/ST2		16
RB7/PGD	40	44	I/O	TTL/ST2		17

PROGRAM MEMORY

The PIC 16F87x devices have a 13-bit program counter, Capable of addressing an 8K x 14 program memory space. The PIC 16F877 has 8Kx 14 words of FLASH program memory. The RESET Vector is at 0000h and the interrupt vector is at 0004h.

DATA MEMORY

Data memory is partition in to multiple banks which contain the general purpose registers and special function registers. Bits RP1(status <6>) and RP0(status<5>) are the banks bits.

RP1	RP0	BANK
0	0	0
0	1	1
1	0	2

1	1	3

Each bank extends up to 7Fh (128bits). The lower location of each banks are reserved for the special function registers. About the special function registers are general purpose registers, implemented as the static RAM. All implemented banks contain special function registers .Some frequently used special function register from 1 bank may be mirrored in another bank for code reduction and quicker access.

General Purpose Register

The register file can be accessed either directly, or indirectly through the file select register (FSR).

SPECIAL PURPOSE REGISTER The Special purpose registers are registers used by the CPU and peripheral modules for controlling the desired operation of the devices. These devices are implemented as static RAM. Some examples of the SFR's are INDF, OPTION_REG, FSR, PCLATH e.t.c.

STATUS REGISTER The Status register contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory. The STATUS register can be the destination for any instruction, as with any other register.

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
			———	———			

IRP	RP1	RP0	TO	PD	Z	DC	C
Bit 7							Bit 0

BIT 7 IRP: Register Bank select bit

1=Bank 2, 3(100h-1FFh)

0=Bank 0, 1(00h-FFh)

Bit 6-5 RP1:RP0: Register bank selects bits

Bit 4 TO: time out bit

Bit 3 PD: power down bit

Bit 2 Z: Zero bit

Bit 1 DC: Digit carry

Bit 0 C: Carry

I/O PORTS ASSOCIATED WITH PIC 16F877 Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. PIC16F877 have FIVE ports (Port A, B,C, D, E).

PORT A & TRIS A Registers

Port A is a 6-bit wide, bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit will make the corresponding Port A pin as input. Clearing a TRISA bit will make the corresponding Port A pin as output.

Pin RA4 is multiplexed with the Timer 0 module clock input to become the RA4/T0CK1 pin. The RA4/T0CK1 pin is a Schmitt Trigger input and an open drain output. All other Port A pins have TTL input levels and full CMOS output drivers. Other Port A pins are multiplexed with analog input and analog Vref input. The operation of each pin is selected by cleaning/setting the control bits in the ADCON1 register.

TRISA register controls the direction of the RA pins, even when they are used as analog inputs. The user must ensure that the bits in the TRISA register are maintained set when using them as analog inputs.

PORT B AND THE TRISB Register

PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit will make the corresponding PORTB pin an input. Clearing a TRISB bit make the corresponding PORTB pin an output.Three pins of PORTB are multiplexed with the Low Voltage Programming function: RB3/PGN, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in the Special Features Section.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by cleaning bit RBPU (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pulls are disabled on a Power-on Reset.

Four of the PORTB pins, RB7:RB4 have interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur. The input pin (RB7:RB4) are compared with the old value latched on the last read

of PORTB. The “Mismatch” output of RB7:RB4 are OR’ed together to generate the RBPORT change interrupt with flag bit RBIF (INTCON<0>). This interrupt can wake the device from SLEEP. A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared

PORTC AND THE TRISTC Register

PORTC is an 8 bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting TRISC bit (=1) will make the corresponding PORTC pin an input. Clearing TRISC bit (=0) will make the corresponding PORTC pin an output. PORTC is multiplexed with several peripheral functions. PORTC pin have SCHMITT Trigger input buffers. When the I^2C module is enabled the PORTC<4:3> pin can be configure with normal I^2C levels.

PORT D AND THE TRIS D REGISTER

PORT D is an 8-bit port with Schmitt trigger input buffers. Each pin is individually configurable as an input or output.

.PORT E AND THE TRIS E REGISTER

PORT E has three pins which are individually configurable as input or output .The PORT E pins becomes I/O control inputs for the microprocessor port when bit PSPMODE(TRIS<4>) is set .PORT E pins are multiplexed with analog input . TRISE controls the direction of the RE pins.

LCD DISPLAY

Liquid crystal displays are generally more flexible than LED displays because they allow for a variety of text and/or graphics. LCDs require

less power LEDs making them suitable for lose power requirements.) LCDs are more readable in sunlight and an use backlights for night viewing. However, LCDs are more expensive than LED displays. LCDs come in a variety of sizes. Text displays are specified by their character size, the number of lines, and the number of characters per line.

• MATLAB

MATLABis a high-level language and interactive environment for numerical computation, visualization, and programming.Cleve Moler formulated MATLAB in the late 1970s. Using MATLAB, you can analyse data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spread sheets or traditional programming languages, such as C/C++ or Java. Different versions of MATLAB are available. The first version of MATLAB is MATLAB 6.5 (R13). This belongs to the 6.5 series. Preceding this the MATLAB 7 series was introduced and there comes a number of versions under this series.

Some of them are:-

MATLAB 7.4 (2007A), MATLAB 7.5 (2007B), MATLAB 7.6 (2008A), MATLAB 7.8 R2009a , MATLAB7.13 (2011B)

The MATLAB version MATLAB 7.8 R2009a 21 1.6.0_04under the series MATLAB 7 is used as the platform form the project.

General Features of MATLAB

High-level language for numerical computation, visualization, and application development

- Interactive environment for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
- Built-in graphics for visualizing data and tools for creating custom plots
- Development tools for improving code quality and maintainability and maximizing performance
- Tools for building applications with custom graphical interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft Excel

Different tool boxes available in MATLAB useful for the project are:-

1. Math, Statistics and Optimization

   Symbolic math toolbox

- Provides functions for solving and manipulating symbolic math expressions and performing arithmetic functions like differentiation, integration, simplification etc.

  Partial differential equation toolbox

- Contains tools for the study and solution of partial differential equations(PDE's)

  Curve fitting toolbox

- Provides an app and functions for fitting curves to perform data analysis,modeling and improves the quality of fits.

  Optimization toolbox

- Solve constrained and unconstrained continuous and discrete problems.

  Neural network toolbox

- Provides an app and functions for modeling complex nonlinear systems with supervise learning with feed forward radial basis.

  Statistics toolbox

- Provides statistical and machine learning algorithms for organizing, analyzing and modeling data.

2.Signal processing and communication

DSP system toolbox

- Provides algorithms for designing and simulating signal processing systems.

  Signal processing toolbox

- Provides industry- standard algorithms for analog and DSP.

Communication system toolbox

- Provides algorithms for designing,simulating and analyzing communication systems.

  Wavelet toolbox
- Develops functions and app by wavelet based algorithms for analysis, compression of signals and images.

3.Image processing and computer vision

Image processing toolbox

- Provides set of algorithms and functions for image enhancement, image segmentation, and visualization.

  computer vision system toolbox

- Provides algorithms for design and simulation of computer vision and video processing systems. Also includes object detection, motion detection and video processing.

  Image acquisition toolbox
- Enables to acquire images and video from cameras into MATLAB.

Applications of MATLAB

MATLAB can be used for a wide range of applications like:-

- ❖ Technical computing
- ❖ Embedded systems
- ❖ Control systems
- ❖ Digital signal processing
- ❖ Communication systems

- ❖ Image and video processing
- ❖ FPGA design

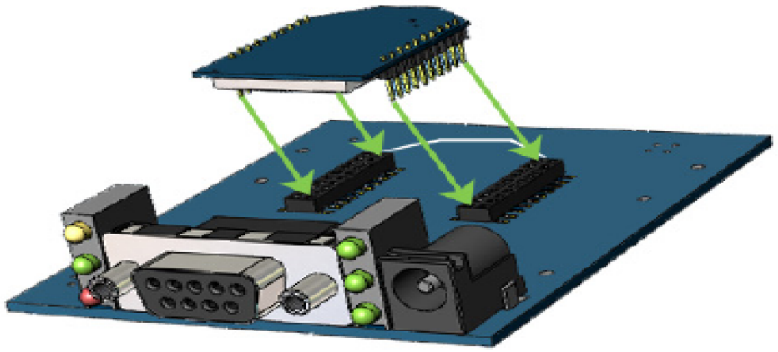
4.COMPONENTS USED

● ZIGBEE/XBEE MODULE

The XBee/ZBee (formerly known as Series 2 and Series 2 PRO) RF Modules were engineered to operate within the ZigBee protocol and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between remote devices. The modules operate within the 2.4 GHz frequency band.

Mounting Considerations:

The XBee modules were designed to mount into a receptacle (socket) and therefore do not require any soldering when mounting it to a board. The XBee PRO Development Kits contain RS-232 and USB interface boards which use two 20-pin receptacles to receive modules. Figure below XBee PRO 2.5 Module Mounting to an RS-232 Interface Board.



- **ACCELEROMETER**

An accelerometer measures proper acceleration, which is the acceleration it experiences relative to freefall and is the acceleration felt by people and objects. Put another way, at any point in space-time the equivalence principle guarantees the existence of a local inertial frame, and an accelerometer measures the acceleration relative to that frame. Such accelerations are popularly measured in terms of g-force.

An accelerometer at rest relative to the Earth's surface will indicate approximately 1 g upwards, because any point on the Earth's surface is accelerating upwards relative to the local inertial frame (the frame of a freely falling object near the surface). To obtain the acceleration due to motion with respect to the Earth, this "gravity offset" must be subtracted and corrections made for effects caused by the Earth's rotation relative to the inertial frame.


Z
X
Y
ADXL335
GND

L293D

Features

- 600 mA output current capability per channel
- 1.2A peak output current(non repetive) per channel
- Enable facility
- Over temperature protection
- Logical “0” input voltage upto 1.5 V
- Internal clamp diodes

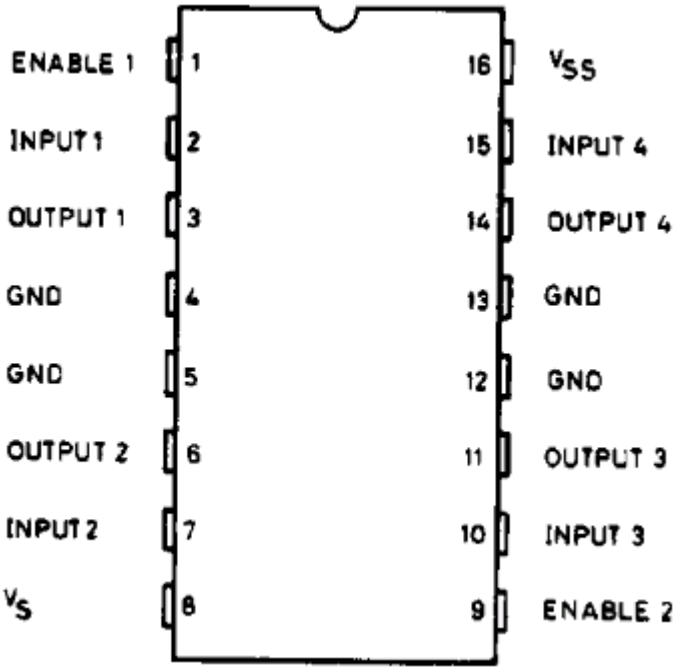
SO(12+4+4)

Powerdip (12+2+2)

ORDERING NUMBERS:

L293DD L293D

Pinout diagram of L293D



Description

The device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoids, DC and stepping motors)and switching power transistors.

To simplify use as two bridges each pair of channels is equipped with an enable input.A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

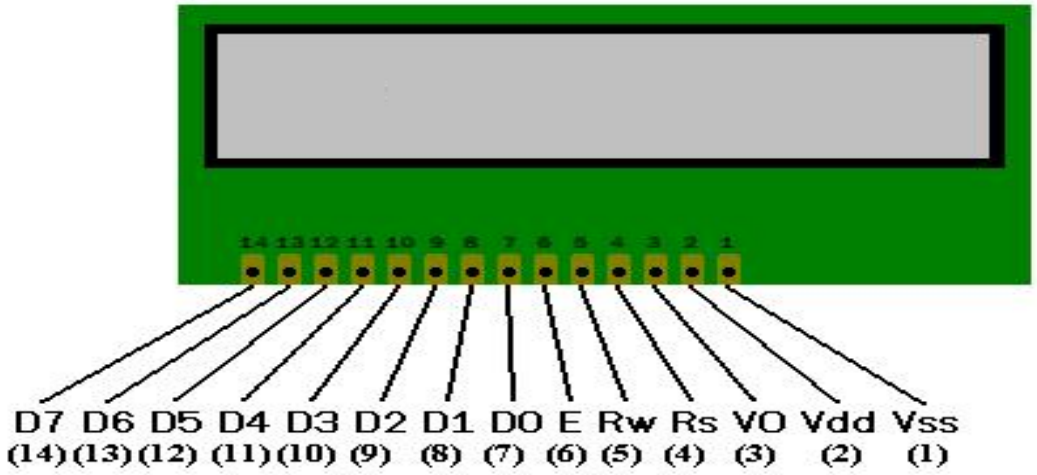
The L293D is assembled in a 16 lead plastic package which has 4 center pins connected together and used for heat sinking.

The L293D is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heat sinking.

● LCD MODULE

The LCD module, made by Crystallonics, is 16x2 line interactive displays. It needs a power supply of +5 V. The module has inbuilt controller chip, such as an HD44780, which acts as an interface between CPU and the row and column drivers. The controller takes care of generating characters, refreshing the display and so on. The module has a back light driven by a pair of pads separate from the interface pads .The LCD module works in two modes for communicating with the micro controller - 8 bit (byte) mode & 4 bit (nibble) mode. In the later case only the higher nibble i.e. pins DB4-DB7 is used for communication. For controlling the LCD module we have used only the port D.

PINOUT



PIN NO.	SYMBOL	FUNCTION
1	Vss	Ground
2	Vdd	+5v
3	Vo	Contrast Adjust
4	RS(H/L)	Register Select H=Data/L=Instruction
5	R/W (H/L)	Read/Write H=Read/L=Write
6	E	Enable
7	DB0	Data Pin 1
8	DB1	Data Pin 2
9	DB2	Data Pin 3
10	DB3	Data Pin4
11	DB4	Data Pin 5
12	DB5	Data Pin 6
13	DB6	Data Pin 7
14	DB7	Data Pin8
15	BL-	Back Light
16	BL+	Back Light

PINOUT DESCRIPTION

Contrast: A variable voltage applied to this pin controls the contrast. Use a potentiometer and adjust until you see the background.

Register Select: This pin selects whether you are sending the module a command or data.

Read/Write: This pin allows for bi-directional communications. For the discussions here, uni-directional communications will be used. Ground this pin.

Enable: This is the latch pin. A high-to-low transition causes the value on the data lines to be latched by the module.

DB0-DB7: Apply the data or commands to these pins.

- **POWER SUPPLY**

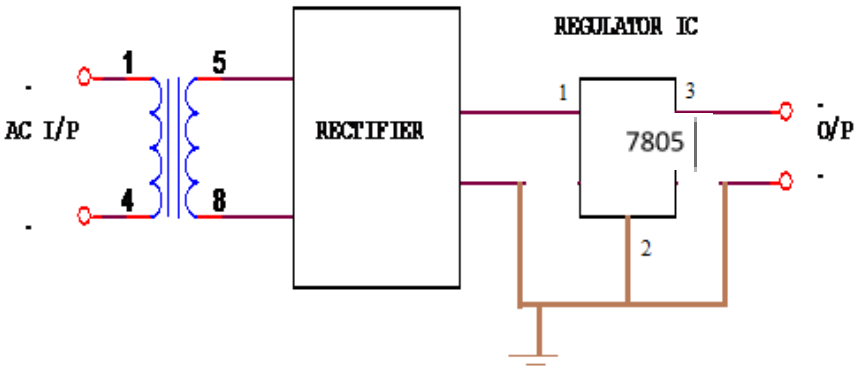
The system requires a regulated +5v supply for the semiconductors and a +12V unregulated supply for the relay. These can be delivered from the 230V domestic supply. Before applying this to the system we must step down this high voltage to an appropriate value. After that it should be rectified. To achieve +5 V DC we should regulate this.All this are run in the power supply circuitry.

Power supply is used to give sufficient power to the microcontroller. A step down transformer and a bridge rectifier is used here to convert AC to DC. A regulator IC is also used here to give constant supply.7805 IC is used for power supply and it is connected to the bridge rectifier.

A 12-0-12V step down transformer is connected to provide the necessary low voltage. The transformer also works as an Isolator between the hot and cold end. The hot end refers to the 230v supply, which is hazardous one, and the cold one refers to the low,safe voltage .Now the

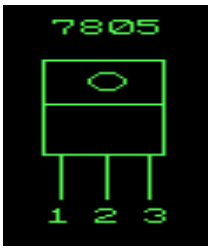
hot portion appears only at the primary of the transformer. The secondary of the transformer deliver 12v ac pulses along with a ground.

This ac supply goes to a center tap rectifier, which converts the ac into a unidirectional voltage. The ripples in the resulting supply is filtered and smoothed by a 2200micro FD/25V capacitor. The 0.1 microfarad capacitor bypasses any high frequency noises. The resulting supply has magnitude above 17V. This voltage is given to the regulated IC 7805.This IC provides a regulated 5V positive supply at its 3rd pin. This required input for this is more than 7.5V.



This circuit can give +5V output at about 150 mA current, but it can be increased to 1 A when good cooling is added to 7805 regulator chip. The circuit has over overload and terminal protection.

Pin out of the 7805 regulator IC.



1. Unregulated voltage in
2. Regulated voltage out
3. Ground

If we need other voltages than +5V, we can modify the circuit by replacing the 7805 chips with another regulator with different output voltage from regulator 78xx chip family. The last numbers in the the chip code tells the output voltage. Remember that the input voltage must be at least 3V greater than regulator output voltage ot otherwise the regulator does not work well.

- **RS-*232***

Electronic data communications between elements will generally fall into two broad categories: single-ended and differential. RS232 (single-ended) was introduced in 1962, and despite rumors for its early demise, has remained widely used through the industry. Independent channels are established for two-way (full-duplex) communications. The RS232 signals are represented by voltage levels with respect to a system common (power / logic ground). The "idle" state has the signal level negative with respect to common, and the "active" state has the signal level positive with respect to common.

The RS-232 interface presupposes a common ground between the DTE and DCE. This is a reasonable assumption when a short cable connects the DTE to the DCE, but with longer lines and connections between devices that may be on different electrical busses with different grounds, this may not be true.

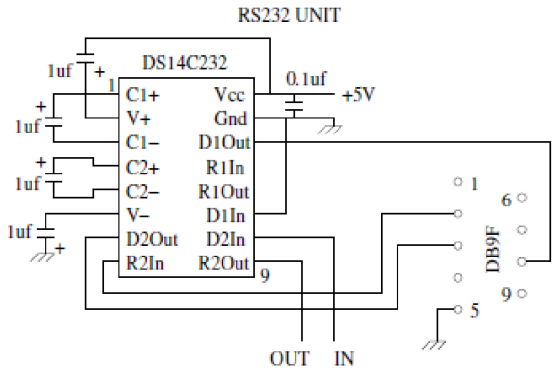


Fig Schematic diagram of RS232 interface circuit.

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 5µWVoltage levels

The RS-232 standard defines the voltage levels that correspond to logical and logical zero levels. Valid signals are plus or minus 3 to 15 volts. The range near zero volts is not a valid RS-232 level; logic one is defined as a negative voltage, the signal condition is called marking, and has the functional significance of OFF. Logic zero is positive, the signal condition is spacing, and has the function ON. The standard specifies a maximum open-circuit voltage of 25 volts; signal levels of ±5 V,±10 V,±12 V, and ±15 V are all commonly seen depending on the power supplies available within a device. RS-232 drivers and receivers must be able to withstand indefinite short circuit to ground or to any voltage level up to ±25 volts. The slew rate, or how fast the signal changes between levels, is also controlled.

Features

- Operate from Single +5V Power
- 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

Applications

- Portable Computers
- Low-Power Modems
- Interface Translation
- Battery-Powered RS-232 Systems

- **MAX 232**

It is a voltage level converter. It converts RS232 voltage levels to TTL .The serial port of PC uses RS232 voltage levels, and microcontroller uses TTL levels. To Match these voltage levels MAX232 IC is used. This IC includes a pair of transmitter and receiver. One advantage of using MAX232 is that, no negative voltage is required for its working. So need of dual voltage supply is eliminated.

The **MAX232** is an integrated circuit that converts signals from an RS- 232serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.

The drivers provide RS-232 voltage level outputs (approx. ± 7.5 V) from a single + 5 V supply via on-chip charge pumps and external

capacitors. This makes it useful for implementing RS-232 in devices that otherwise do not need any voltages outside the 0 V to + 5 V range, as power supply design does not need to be made more complicated just for driving the RS-232 in this case.

The receivers reduce RS-232 inputs (which may be as high as ± 25 V), to standard 5 V TTL levels. These receivers have a typical threshold of 1.3 V, and a typical hysteresis of 0.5 V.

The later MAX232A is backwards compatible with the original MAX232 but may operate at higher baud rates and can use smaller external capacitors – 0.1 μF in place of the 1.0 μF capacitors used with the original device. The newer MAX3232 is also backwards compatible, but operates at a broader voltage range, from 3 to 5.5V.

When a MAX232 IC receives a TTL level to convert, it changes a TTL Logic 0 to between +3 and +15V, and changes TTL Logic 1 to between -3 to -15V, and vice versa for converting from RS232 to TTL. This can be confusing when you realize that the RS232 Data Transmission voltages at a certain logic state are opposite from the RS232 Control Line voltages at the same logic state. To clarify the matter, see the table below.

RS232 Line Type & Logic Level	**RS232 Voltage**	**TTL Voltage to/from MAX232**
Data Transmission (Rx/Tx) Logic 0	+3V to +15V	0V

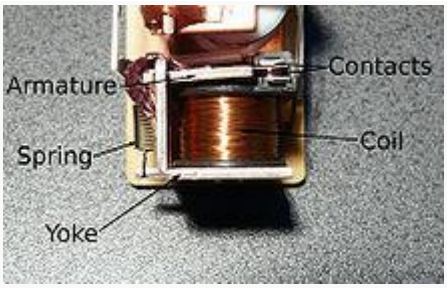
Data Transmission (Rx/Tx) Logic 1	-3V to -15V	5V
Control Signals (RTS/CTS/DTR/DSR) Logic 0	-3V to -15V	5V
Control-Signals (RTS/CTS/DTR/DSR) Logic 1	+3V to +15V	0V

- **RELAY**

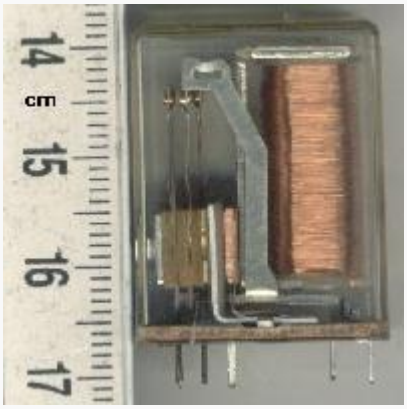
A relay is an electrically operated switch. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits, repeating the signal coming in from one circuit and re-transmitting it to another. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digital instruments still called "protective relays".

Basic design and operation



Simple electromechanical relay.



Small "cradle" relay often used in electronics. The "cradle" term refers to the shape of the relay's armature.

A simple electromagnetic relay consists of a coil of wire wrapped around a soft iron core, an iron yoke which provides a low reluctance path for magnetic flux, a movable iron armature, and one or more sets of contacts (there are two in the relay pictured). The armature is hinged to the yoke and mechanically linked to one or more sets of moving contacts. It is held in place by a spring so that when the relay is de-energized there is an air gap in the magnetic circuit. In this condition, one of the two sets of contacts in the relay pictured is closed, and the other set is open. Other relays may have more or fewer sets of contacts depending on their

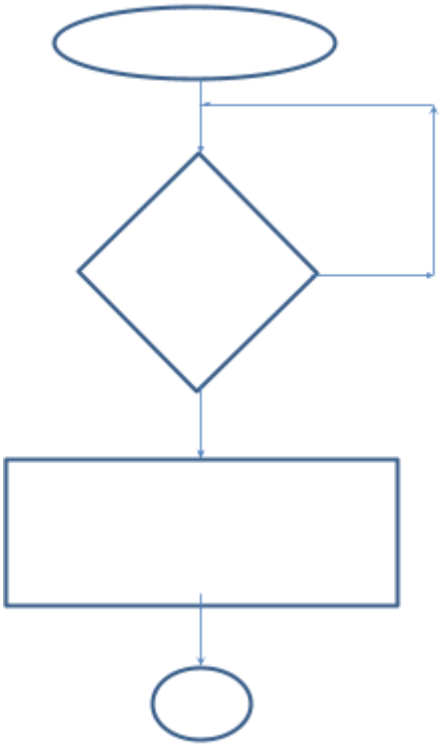
function. The relay in the picture also has a wire connecting the armature to the yoke. This ensures continuity of the circuit between the moving contacts on the armature, and the circuit track on the printed circuit board (PCB) via the yoke, which is soldered to the PCB.

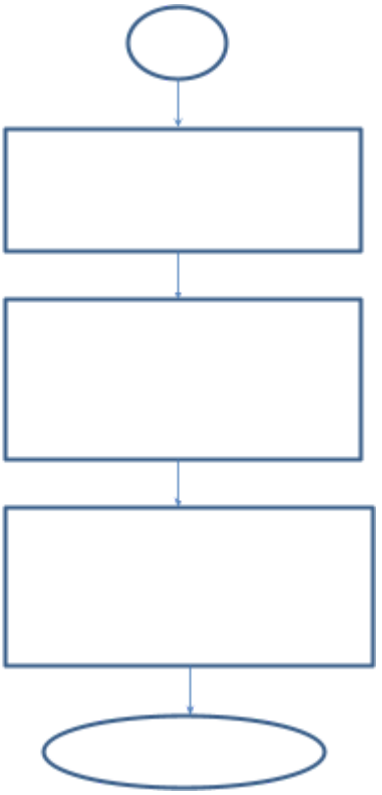
When an electric current is passed through the coil it generates a magnetic field that activates the armature and the consequent movement of the movable contact either makes or breaks (depending upon construction) a connection with a fixed contact. If the set of contacts was closed when the relay was de-energized, then the movement opens the contacts and breaks the connection, and vice versa if the contacts were open. When the current to the coil is switched off, the armature is returned by a force, approximately half as strong as the magnetic force, to its relaxed position. Usually this force is provided by a spring, but gravity is also used commonly in industrial motor starters. Most relays are manufactured to operate quickly. In a low-voltage application this reduces noise; in a high voltage or current application it reduces arcing.

When the coil is energized with direct current, a diode is often placed across the coil to dissipate the energy from the collapsing magnetic field at deactivation.Some automotive relays include a diode inside the relay case. Alternatively, a contact protection network consisting of a capacitor and resistor in series (snubber circuit) may absorb the surge. If the coil is designed to be energized with alternating current (AC), a small copper "shading ring" can be crimped to the end of the solenoid, creating a small out-of-phase current which increases the minimum pull on the armature during the AC cycle.[1]

5.SOFTWARE SECTION

FLOW CHART





C PROGRAM

```
#include<pic.h>

void delay(int x);
void delay1(void);
void delay2(int);
void command(char x);
void data(char x);
void datastr(const char *x);
void trans(char x);
char rec(void);
void strtrans(const char *x);
void main()
{
      int i,t,v,total=0;
      char d,x,y,z;
      char xa[4],ya[4],za[4];
      TRISA=0xFF;
      TRISE=0X0F;
      ADCON0=0x01;
      ADCON1=0x00;
      TRISB=0X00;
      TRISC=0x80;
      TRISD=0x00;
      OPTION=0X07;
      TXSTA=0X24;
      RCSTA=0X90;
      SPBRG=0X19;

      command(0x01);
      command(0x0E);
      command(0x80);
      command(0x38);

      command(0x01);
      datastr("   DARE DRONE   ");
      command(0xC0);
      datastr("FLIGHT COURSE...");
      delay(5);




      command(0x01);
      datastr("processing......");
      delay(2);

      command(0xC0);
      datastr("X");
      delay(2);
```

```
command(0xC6);
datastr("Y");
delay(2);

command(0xCC);
datastr("Z");
delay(2);

while(1)
{

/*Accelerometer  x*/
CHS2=0;
CHS1=0;
CHS0=0;
ADGO=1;
while(ADGO==1);
x=ADRESH;
xa[0]=x/100+0x30;
xa[1]=(x/10)%10+0x30;
xa[2]=(x%10)+0x30;
xa[3]='\0';
command(0XC1);
datastr(xa);
delay(10);

/*Accelerometer  y*/
CHS2=0;
CHS1=0;
CHS0=1;
ADGO=1;
while(ADGO==1);
y=ADRESH;
ya[0]=y/100+0x30;
ya[1]=(y/10)%10+0x30;
ya[2]=(y%10)+0x30;
ya[3]='\0';
command(0XC7);
datastr(ya);
delay(10);


/*Accelerometer  z*/
CHS2=0;
CHS1=1;
CHS0=0;
ADGO=1;
while(ADGO==1);
z=ADRESH;
za[0]=z/100+0x30;
za[1]=(z/10)%10+0x30;
za[2]=(z%10)+0x30;
za[3]='\0';
command(0XCD);
datastr(za);
```

```
      delay(20);


      /* checking */


      /*matlab tx */
      delay(2);

      trans('*');

      strtrans(xa);

      strtrans(ya);

      strtrans(za);
     trans('#');
     delay(250);

      }
}
void delay(int x)
{
int i,j;
for(j=0;j<x;j++)
for(i=0;i<=1000;i++);
}
void delay1(void)
{

TMR0=0X00;
while(TMR0IF==0);
TMR0IF=0;

}
void delay2(int k)
{
int i;
for(i=0;i<k*15;i++)
{
TMR0=0X00;
while(TMR0IF==0);
TMR0IF=0;
}
}
void command(char x)
{
      PORTD=x;
      RC0=0;

      RC3=1;
      delay(2);
      RC3=0;
}
void data(char x)
```

```
{
      PORTD=x;
      RC0=1;

      RC3=1;
      delay(2);
      RC3=0;
}
void datastr(const char *x)
{
      while(*x)
      {
      data(*x);
      *x++;
      }
}
void strtrans(const char *x)
{
            while(*x)
            {
            trans(*x);
            *x++;
            }
}
void trans(char x)
{
      TXREG=x;
      while(TXIF==0);
      TXIF=0;
}
char rec(void)
{
      while(RCIF==0);
      RCIF=0;
      if(OERR)
      {
      CREN=0;
      delay(1);
      CREN=1;
      }
      return(RCREG);
}
```

MATLAB PROGRAM

```
function varargout = readcheck(varargin)
% READCHECK MATLAB code for readcheck.fig
%      READCHECK, by itself, creates a new READCHECK or
raises the existing
%      singleton*.
%
%      H = READCHECK returns the handle to a new READCHECK or
the handle to
%      the existing singleton*.
%
%      READCHECK('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in READCHECK.M with the given
input arguments.
%
%      READCHECK('Property','Value',...) creates a new
READCHECK or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before readcheck_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to readcheck_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
readcheck

% Last Modified by GUIDE v2.5 08-Aug-2014 11:13:14

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @readcheck_OpeningFcn,
...
                   'gui_OutputFcn',  @readcheck_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before readcheck is made visible.
function readcheck_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to readcheck (see
VARARGIN)

% Choose default command line output for readcheck
clc;
instrreset;
handles.obj=serial('com3');
handles.obj1=serial('com1');
set(handles.obj1,'baudrate',9600);
set(handles.obj,'baudrate',9600);
set(handles.obj,'timeout',5);
set(handles.obj1,'timeout',5);
fopen(handles.obj);
fopen(handles.obj1);
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes readcheck wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = readcheck_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as
text
%        str2double(get(hObject,'String')) returns contents
of edit1 as a double


% --- Executes during object creation, after setting all
properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as
text
%        str2double(get(hObject,'String')) returns contents
of edit2 as a double


% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
```

```
% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as
text
%        str2double(get(hObject,'String')) returns contents
of edit3 as a double

% --- Executes during object creation, after setting all
properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)


% --- Executes on button press in pushbutton3.
```

```
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)


a=fscanf(handles.obj);
disp(a)

x=a(2:4)
y=a(5:7)
z=a(8:10)
x=str2num(x);
y=str2num(y);
z=str2num(z);
set(handles.edit1,'string',x)
 set(handles.edit2,'string',y)
 set(handles.edit3,'string',z)
if (x>80 && x<120 && y>80 && y<120 && z>80 && z<120 )

    disp('ok')
    fwrite(handles.obj1,'n');
    fwrite(handles.obj1,a);

else

    disp('nok')
fwrite(handles.obj1,'o');
 pause(2)
a1=get(handles.pushbutton5,'position');
sa=a1;
b=a1(1);
 set(handles.edit4,'visible','on')
 set(handles.text4,'visible','on')
 set(handles.pushbutton5,'visible','on')
for i=1:280
a1(1)=b+i*.3;
set(handles.pushbutton5,'position',a1)
pause(0.08)
end
set(handles.pushbutton5,'position',sa)
drawnow
fwrite(handles.obj1,'n');
fwrite(handles.obj1,a);
end
```

```
data=fscanf(handles.obj1);

set(handles.edit5,'string',data)

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of edit4 as
text
%        str2double(get(hObject,'String')) returns contents
of edit4 as a double

% --- Executes during object creation, after setting all
properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
%Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% Hints: get(hObject,'String') returns contents of edit5 as
text
%        str2double(get(hObject,'String')) returns contents
of edit5 as a double

% --- Executes during object creation, after setting all
properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

6. CONCLUSION

Our system has been worked very well for the data recovery of black box in the case of missing plane. In current scenario, a number of plane and ship missing happen. If we are able to implement our system, then the reason for missing the plane can be easily found, even before getting the black box/plane parts. Thus the time taken for rescue operation can be reduced.

FUTURE SCOPE

By using multi-base stations, an updated version of our system will reach the nearest station as soon as the ejection takes place, by sensing the nearest location from the trajectory.

REFERENCE

www.microchip.com
www.wikipedia.com
www.MATLAB.com
www.google.com
www.alldatasheet.com
www.electronicsforu.com

DATA SHEETS

MICROCHIP

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM),
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI	13	14	30	32		ST/CMOS(4)	Oscillator crystal or external clock input.
OSC1					I		Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS.
CLKI					I		External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO	14	15	31	33		—	Oscillator crystal or clock output.
OSC2					O		Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO					O		In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	18		ST	Master Clear (input) or programming voltage (output).
$\overline{MCLR}$					I		Master Clear (Reset) input. This pin is an active low Reset to the device.
VPP					P		Programming voltage input.
							PORTA is a bidirectional I/O port.
RA0/AN0	2	3	19	19		TTL	
RA0					I/O		Digital I/O.
AN0					I		Analog input 0.
RA1/AN1	3	4	20	20		TTL	
RA1					I/O		Digital I/O.
AN1					I		Analog input 1.
RA2/AN2/VREF-/CVREF	4	5	21	21		TTL	
RA2					I/O		Digital I/O.
AN2					I		Analog input 2.
VREF-					I		A/D reference voltage (Low) input.
CVREF					O		Comparator VREF output.
RA3/AN3/VREF+	5	6	22	22		TTL	
RA3					I/O		Digital I/O.
AN3					I		Analog input 3.
VREF+					I		A/D reference voltage (High) input.
RA4/T0CKI/C1OUT	6	7	23	23		ST	
RA4					I/O		Digital I/O – Open-drain when configured as output.
T0CKI					I		Timer0 external clock input.
C1OUT					O		Comparator 1 output.
RA5/AN4/$\overline{SS}$/C2OUT	7	8	24	24		TTL	
RA5					I/O		Digital I/O.
AN4					I		Analog input 4.
$\overline{SS}$					I		SPI slave select input.
C2OUT					O		Comparator 2 output.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	9		TTL/ST(1)	
RB0					I/O		Digital I/O.
INT					I		External interrupt.
RB1	34	37	9	10	I/O	TTL	Digital I/O.
RB2	35	38	10	11	I/O	TTL	Digital I/O.
RB3/PGM	36	39	11	12		TTL	
RB3					I/O		Digital I/O.
PGM					I		Low-voltage ICSP programming enable pin.
RB4	37	41	14	14	I/O	TTL	Digital I/O.
RB5	38	42	15	15	I/O	TTL	Digital I/O.
RB6/PGC	39	43	16	16		TTL/ST(2)	
RB6					I/O		Digital I/O.
PGC					I		In-circuit debugger and ICSP programming clock.
RB7/PGD	40	44	17	17		TTL/ST(2)	
RB7					I/O		Digital I/O.
PGD					I/O		In-circuit debugger and ICSP programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI	15	16	32	34		ST	
RC0					I/O		Digital I/O.
T1OSO					O		Timer1 oscillator output.
T1CKI					I		Timer1 external clock input.
RC1/T1OSI/CCP2	16	18	35	35		ST	
RC1					I/O		Digital I/O.
T1OSI					I		Timer1 oscillator input.
CCP2					I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	17	19	36	36		ST	
RC2					I/O		Digital I/O.
CCP1					I/O		Capture1 input, Compare1 output, PWM1 output.
RC3/SCK/SCL	18	20	37	37		ST	
RC3					I/O		Digital I/O.
SCK					I/O		Synchronous serial clock input/output for SPI mode.
SCL					I/O		Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	23	25	42	42		ST	
RC4					I/O		Digital I/O.
SDI					I		SPI data in.
SDA					I/O		I^2C data I/O.
RC5/SDO	24	26	43	43		ST	
RC5					I/O		Digital I/O.
SDO					O		SPI data out.
RC6/TX/CK	25	27	44	44		ST	
RC6					I/O		Digital I/O.
TX					O		USART asynchronous transmit.
CK					I/O		USART1 synchronous clock.
RC7/RX/DT	26	29	1	1		ST	
RC7					I/O		Digital I/O.
RX					I		USART asynchronous receive.
DT					I/O		USART synchronous data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTD is a bidirectional I/O port or Parallel Slave Port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	38		ST/TTL(3)	
RD0					I/O		Digital I/O.
PSP0					I/O		Parallel Slave Port data.
RD1/PSP1	20	22	39	39		ST/TTL(3)	
RD1					I/O		Digital I/O.
PSP1					I/O		Parallel Slave Port data.
RD2/PSP2	21	23	40	40		ST/TTL(3)	
RD2					I/O		Digital I/O.
PSP2					I/O		Parallel Slave Port data.
RD3/PSP3	22	24	41	41		ST/TTL(3)	
RD3					I/O		Digital I/O.
PSP3					I/O		Parallel Slave Port data.
RD4/PSP4	27	30	2	2		ST/TTL(3)	
RD4					I/O		Digital I/O.
PSP4					I/O		Parallel Slave Port data.
RD5/PSP5	28	31	3	3		ST/TTL(3)	
RD5					I/O		Digital I/O.
PSP5					I/O		Parallel Slave Port data.
RD6/PSP6	29	32	4	4		ST/TTL(3)	
RD6					I/O		Digital I/O.
PSP6					I/O		Parallel Slave Port data.
RD7/PSP7	30	33	5	5		ST/TTL(3)	
RD7					I/O		Digital I/O.
PSP7					I/O		Parallel Slave Port data.
							PORTE is a bidirectional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	25		ST/TTL(3)	
RE0					I/O		Digital I/O.
$\overline{RD}$					I		Read control for Parallel Slave Port.
AN5					I		Analog input 5.
RE1/$\overline{WR}$/AN6	9	10	26	26		ST/TTL(3)	
RE1					I/O		Digital I/O.
$\overline{WR}$					I		Write control for Parallel Slave Port.
AN6					I		Analog input 6.
RE2/$\overline{CS}$/AN7	10	11	27	27		ST/TTL(3)	
RE2					I/O		Digital I/O.
$\overline{CS}$					I		Chip select control for Parallel Slave Port.
AN7					I		Analog input 7.
VSS	12, 31	13, 34	6, 29	6, 30, 31	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	12, 35	7, 28	7, 8, 28, 29	P	—	Positive supply for logic and I/O pins.
NC	—	1, 17, 28, 40	12,13, 33, 34	13	—	—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

ABSOLUTE MAXIMUM RATINGS

Voltage Range on Any Pin Relative to Ground .....-0.5V to +7.0V
Operating Temperature Range (Noncondensing)
Commercial.....0°C to +70°C
Industrial .....-40°C to +85°C
Storage Temperature Range..... -55°C to +125°C
Soldering Temperature (DIP, leads).....+260°C for 10 seconds
Soldering Temperature (surface mount).....Refer to the JPC/JEDEC J-STD-020 Specification.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

(T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Logic 1 Input	V_{IH}		2.2		V_{CC} + 0.3	V
Logic 0 Input	V_{IL}		-0.3		+0.8	V
V_{BAT} Battery Voltage	V_{BAT}		2.0	3	3.5	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Leakage (SCL)	I_{LI}		-1		1	µA
I/O Leakage (SDA, SQW/OUT)	I_{LO}		-1		1	µA
Logic 0 Output (I_{OL} = 5mA)	V_{OL}				0.4	V
Active Supply Current (f_{SCL} = 100kHz)	I_{CCA}				1.5	mA
Standby Current	I_{CCS}	(Note 3)			200	µA
V_{BAT} Leakage Current	I_{BATLKG}			5	50	nA
Power-Fail Voltage (V_{BAT} = 3.0V)	V_{PF}		1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 0V, V_{BAT} = 3.0V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{BAT} Current (OSC ON); SQW/OUT OFF	I_{BAT1}			300	500	nA
V_{BAT} Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}			480	800	nA
V_{DAT} Data-Retention Current (Oscillator Off)	I_{BATDR}			10	100	nA

WARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High			2	1.4		V
Logic Pull-Up/Input Current	Normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate	All except MAX220, normal operation			200	116	kbits/sec
	MAX220			22	20	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA