ACKNOWLEDGEMENT

Every success stands as a testimony note only to the hardship but also to hearts behind it. Likewise, the present project has been undertaken and completed with direct and indirect help from many people and we would like to acknowledge the same.

First and foremost, we take immense pleasure in thanking the management and respected principal, Sumitha A, for providing us with the wider facilities.

We express our sincere thanks to Biju P K, Head of the department of electronics for giving us opportunity to present this project and for kindly suggestions.

Needless that teaching and non-teaching facility members had been a source of inspiration and kindly support in the conduct of our project session. We would also right to express my heartfelt thanks to our beloved parents for their blessing, our friends for their help and wishes for the successful completion of this project.

Above all we would like to thank the almighty God for the blessings that helped us to complete this venture smoothly.

ABSTRACT

Deafness or hearing-impairment is the loss of ability to hear normally whether permanent or unstable. It can be caused by environmental and genetic factors as well. According to WHO estimates, 360 million people worldwide are suffering from hearing loss and this could increase to 900 million by 2025. In a study conducted, it was found that hearing impairment students have a hard time adjusting with the other hearing classmates. They don’t feel belongingness and hence get admissions in special education institutions while lack of sign language interpreters in schools is one of the reasons for frustration observed in students. .

The system is based on sensor gloves which convert the sign language to speech. The glove is equipped with flex sensors and an accelerometer which senses the gesture. The controller identify the gesture and transmits a character corresponding to this gesture to an android device. And the respective voice for the character is speeches out through the android device.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Human beings have a natural ability to see, listen and interact with their external environment. Unfortunately, there are some people who are differently abled and do not have the ability to use their senses to the best extent possible. Deaf and dumb population is a result of the physical disability of hearing for deaf people and disability of speaking for dumb people. In the recent years, there has been a rapid increase in the number of hearing impaired and speech disabled victims due to birth defects, oral diseases and accidents. When a speech impaired person speaks to a normal person, the normal person finds it difficult to understand and asks the deaf and dumb person to show gestures for his/her needs. Those people have their own language to communicate with us; the only thing is that we need to understand their language.

Sign language is used by deaf and mute people and it is a communication skill that uses gestures instead of sound to convey meaning simultaneously combining hand shapes, orientation and movement of hands, arms or body and facial expressions to express fluidly a speaker’s thoughts. But most of the time normal people find it difficult to understand this sign language. This presents a major throwback for people in the deaf and dumb communities when they try to engage in interaction with others, especially in their educational, social and professional environments. Therefore, it is necessary to have an advance gesture recognition or sign language detection system to bridge this communication gap.

Smart glove has been designed to give voice to voiceless as this cause has been championed throughout history, as it’s safe to say that none of those efforts involved packing a bunch of sensors into a glove. The main objective is to help deaf and dumb people by removing communication barrier so they are not restricted in a small social circle and are also able to convey their feelings and emotions.

1.2 INTRODUCTION TO EMBEDDED SYSTEM

Embedded system is any electronic equipment with built in intelligence and dedicated software. All embedded systems are used either a microprocessor or a microcontroller. The application of these controllers makes user friendly cheaper solutions and enables to add features otherwise impossible to add features otherwise impossible to provide by other means.

Embedded devices can be defined as any devices with a microprocessor embedded in it that has a relatively focused functionality. The software for the embedded system is called firmware. The firmware is written in assembly language for time or resources critical Operation or using higher level languages like C or embedded C. These software will be simultaneously micro code simulations for the largest processor. Since they are supported to perform only specific task, these programs are stored in read only memories (ROMs)

Moreover they may need to know or minimal inputs from the user, hence the user interface like monitor, mouse and large key board etc. may be absent. Embedded systems are also known as Real time systems. Since they respond to an input or event and produce the result with guarantee time period. This time period can be a few micro seconds to days or months.

1.1.1 Embedded system development

The development of embedded system application, the hardware and software must go hand in hand. The software created by software engineers must be burned into or micro coded in to the hardware or the microcontroller produced by the VLSI engineers. The microcontrollers and the software micro coded in it together form the system for the particular application. The software program for the real time system is written either in assembly or high-level language such as C. the assembly language is used in the case of some critical application. Now day’ high-level languages replace most of the assembly language constructs.

1.1.2 Embedded system markets

Embedded technology is present in almost every electronic device we use today. There is embedded software inside the cellular phone, automobiles and their most at in air conditioners, industrial control equipment and scientific &medical equipment, defense uses communication satellites etc. embedded technology thus covers a broad range of products of which generalization is difficult.

The embedded intelligence can be found in 5 broad market. The first is consumer segment, which include home appliances and entertainment equipment. The second is automotive segment where a modern car has nearly 50 microcontrollers providing intelligence and Control, like keyless entry, antilock breaking airbags. The third is office automation which includes PC's, keyboards, copies and printers. The fourth market, Telecommunication, includes cellular phones, pagers and answering machines.

1.1.3 Characteristics of Embedded System

- Embedded systems are used to do some specific task rather than a general purpose computer for multi task.
- Embedded systems are not always separate devices. Most often they are built into device they control.
- It often runs with limited computer hardware resources.
- The software written for embedded system is called firmware & is stored in ROM or flash memory chips rather than a disk drive.

1.1.4 Advantages of Embedded System

- High performance: the integration of various ICs shortens the travelling route and time of data to be transmitted resulting in higher performance
- Low power consumption: the integration of various ICs eliminates buffers and other interface circuits. As the no of components is reduced, less power will be consumed.
- Slimmer and more compact: housed in a single separate package, the chip is smaller in size and there for occupies less space on the PCB. Hence products using embedded system are slimmer and more compact.
- Lower system costs: in the past several chips in separate packages were required to configure a system. now, just one system on –chip can replace all of these, dramatically reducing the packaging cost

1.3. MICROCONTROLLER VERSUS MICROPROCESSOR

The microprocessor is a clock driven semiconductor device consisting of electronic logic circuits. By microprocessor is meant the general-purpose microprocessors such as Intel’s x86 family. These microprocessors containing no RAM, no ROM, no I/O ports on the chip itself.

The microprocessor is capable of performing various computing functions and making decisions to change the sequence of program execution. The microprocessor is in many ways similar to CPU but includes all the logic circuitry including the control u nit, on one chip. The microprocessor is divided mainly in to three segments they are arithmetic logic unit (ALU), register array and control unit.

Arithmetic logic unit - This is the area of the microprocessor where various computing functions are performed on data. The ALU performs such operations as addition and subtractions, and such logic functions as AND, OR, exclusive OR.

Register Array- This area of microprocessors consists of various registers. These registers are primarily used to store data temporarily during the execution of a program and are accessible to the user through instruction.

Control Unit- The control unit provides the necessary timing and control signals to the all the operation in the microcomputer. It controls the flow of data between the microprocessor and memory and peripherals.

A microcontroller has a CPU in addition to a fixed amount of RAM, ROM, 110 ports and a timer all on a single chip. The fixed amount of on chip ROM, RAM, and number of I/O ports in microcontrollers make them ideal for many applications in which cost, and space are critical.

1.4 CRITERIA FOR CHOOSING A MICROCONTROLLER

1. The first and foremost criteria is that it must meet the task at hand efficiently and cost effectively. In analyzing the need for microcontroller-based project, first see whether an 8-bit or 32-bit microcontroller can best handle the computing the needs of the task most effectively. Other considerations are:
   - Packaging.
   - Power consumptions.
   - The amount of RAM and ROM on the chip.
   - The number of I/O pins and timer on the chip.
   - Cost per unit.
2. The second criteria in choosing a microcontroller are how easy it is to develop product around it.
3. The third criterion is its ready availability in needed quantities both now and in future.

1.5 ADVANTAGES OF PIC MICROCONTROLLER

Microchip’s PIC Microcontrollers are one of the most versatile tools you can use within an electronic because they offer flexible memory technologies and can support various hardware and software.

Superb Speed Capacity

The 8-bit, 16-bit and 32-bit microcontrollers have the speed capacity of up to an incredible 64 MIPS, which use an internal oscillator block. To put this into perspective, this is approximately sixteen times faster than the average AVR microcontrollers.

Simple Migration

In addition to offering excellent speed capabilities, PIC microcontrollers are simple to program. While setting up an electronic can sometimes be difficult, the microcontrollers can be easily integrated into a variety of electronic projects – shaving off some precious time to format a project. Due to the flexibility of the 8-bit, 16-bit or 32-bit PIC microcontrollers, it’s easy to scale a design up or down, with the lowest power available reaching 100 DMIPS. As your code requirements start to grow with a design, you can utilise the complete portfolio from 384b to 52kb of program memory. The PIC microcontrollers have been designed to adapt to an engineer’s needs, which is why the MCUs also feature upward architectures that can preserve investment in code development.

Reliability

In comparison to other microcontrollers, the PIC microcontroller is much more reliable, as it is less likely to malfunction when built into a device. It also offers a powerful performance thanks to the use of RISC architecture. A developer or engineer can also depend on the PIC microcontroller for easy integration of the interface, and they can also connect analogue devices without having to add additional circuitry, which is why many companies turn to the PIC microcontroller when creating a new prototype or device.

Integrated Peripherals

With a low total system cost and integrated peripherals, it’s hardly surprising that engineers have turned to 8-bit, 16-bit and 32-bit microcontrollers for the development of electronic projects. For example, the microcontrollers can provide a range of control features, such as a real-time clock, motor control and power supply, counters and capture/compare. There are also analog peripherals, such as A/D converters, comparators, D/A converters and op amps.

Improved Time to Market

As we mentioned earlier, the 8-bit, 16-bit and 32-bit PIC microcontrollers can each be easily migrated into an electronic project. What's more, they are a low-risk product that allows you to scale up or down on a project, depending on your needs. It will, therefore, be music to an engineer's ears that this low-risk development MCU can improve a device's time to market. The MCUs can also be supported by third-party hardware and software development tools, and engineers can also take advantage of the free C compilers that are currently available from Microchip.

Why 8-bit PIC MCUs?

If you are an engineer or developer looking for an easy migration path from 6 to 100 pins, without the need for next to no code, you should consider the 8-bit PIC MCU. The biggest benefit of the microcontroller is its vast range of peripherals, which allow you to increase the control within a system when required. Many people often turn to the 8-bit because a range of application functions can be combined onto the single PCU for a cost-effective design solution, as developers can utilize everything from power and motor control, system management, user interface and environmental sensing.

Why 16-bit MCUs?

If the 8-bit PIC microcontroller is stretching its capabilities within an electronic, you can scale-up a design to a 16-bit MCU, with the 16-bit PIC24 comprising of two subfamilies. For an affordable low-power performance that exceeds the 8-bit in memory, performance, and peripherals, consider the PIC24F. However, if you are looking for a greater performance and easy migration for a demanding application, the PIC24H/E may be the perfect solution, because it offers up to 70 MIPS performance, as well as additional memory, up to 150°C operation and extra peripherals.

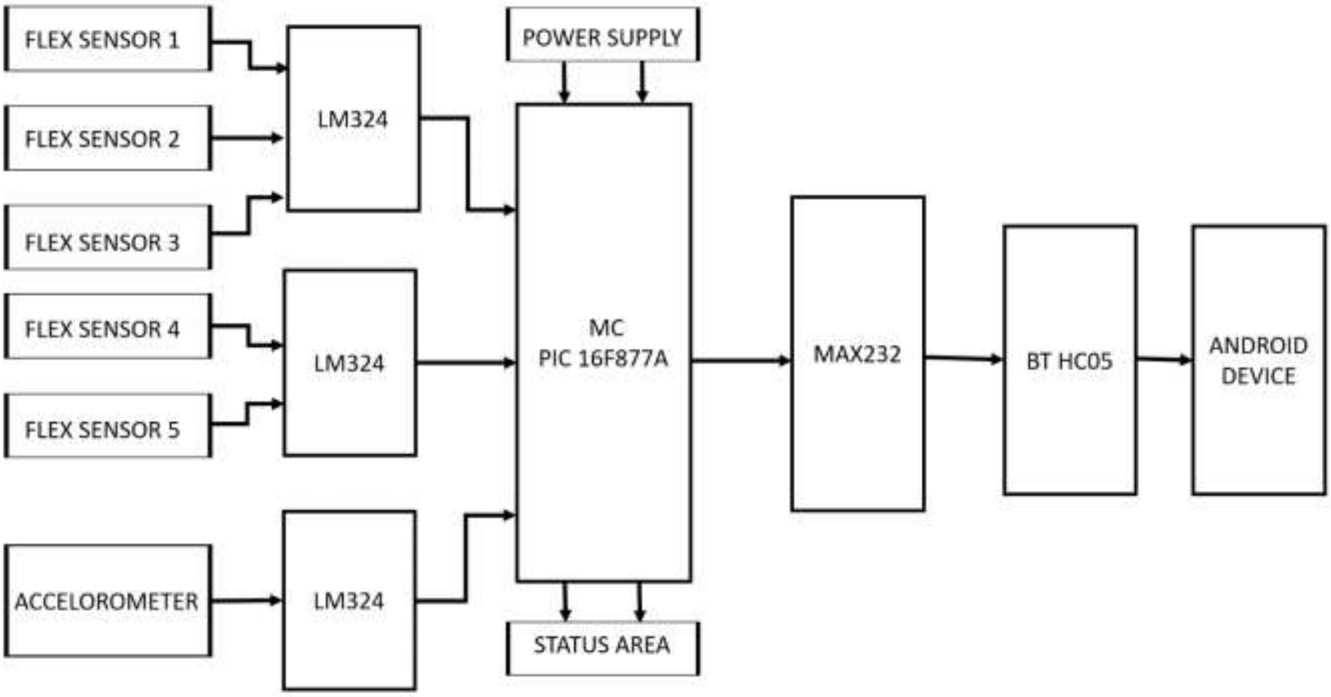
Why 32-bit MCUs?

If the 8-bit and 16-bit PIC MCUs do not fit your project needs, you might be best opting for the PIC32 family, which offers additional memory and performance, while maintaining the pin, peripheral and software capabilities you would find in the 16-bit MCU/DSC families from Microchip. The PIC32 family also has an operation capable of 330 DMIPS as well as data capabilities with a 512 KB RAM and 2048 KB Flash. Whatever a developer's design challenges, there is a PIC32 device to complement their requirements.

CHAPTER 2

BLOCK DIAGRAM

2.1 BLOCK DIAGRAM



2.2 BLOCK DIAGRAM DESCRIPTION

Five flex sensors are used in this system to detect sign language. Accelerometer placed on the back of the hand in order to determine the position of the hand. The output of the flex sensors and accelerometer are given to the LM324. LM324 compares the two voltages and the output is given to the microcontroller. Here PIC 16F887A microcontroller is used. The digital data is given to the microcontroller for further processing. Recognized gestures are matched with preferred data and if it matches given to android device through HC05 Bluetooth module. Then the android device produce audio output corresponding to the sign language.

1. MICROCONTROLLER
2. POWER SUPPLY
3. LM324
4. MAX 232
5. BT HC05
6. FLEX SENSOR
7. ACCELEROMETER
8. ANDROID DEVICE
9. STATUS AREA

2.2.1 MICROCONTROLLER

The microcontroller, which used here, is PIC16F877A .It consisting of 5 ports, ADC, CLK, &MCLR. These are inbuilt within 40 pins. The microcontroller accepts and gives the output in digital form. A microcontroller has a CPU in addition to a fixed amount of RAM, ROM I/O ports and a timer all on a single chip. The fixed amount of on chip ROM, RAM and number of I/O ports in microcontrollers make them ideal for many applications in which cost and space are critical.

Figure 2.2.1.1: PIC 16F877A Microcontroller

2.2.2 POWER SUPPLY

Here a 9v battery provides for the whole system. The microcontroller needs 5v.so it is converted through 7805 regulator. Two capacitors are also there, one for filtering and other for clearing the damages. The microcontroller and other devices get power supply from AC to DC adapter through 7805, 5V regulator. The adapter output will be 12V DC Non-regulated. The 7805 voltage regulators are used to convert 12V to 5V DC.

2.2.3 LM324

LM324 is a 14pin IC consisting of four independent operational amplifiers (op-amps) compensated in a single package. Op-amps are high gain electronic voltage amplifier with differential input and, usually, a single-ended output. The output voltage is many times higher than the voltage difference between input terminals of an op-amp.

2.2.4 MAX 232

It a IC used for TTL/CMOS to RS232 conversion. Most of Microcontrollers operates on TTL/CMOS logic that is it communicates through either 0V or +5V, but computers work with the help of RS232 which operates at logic level -24V or +24V. So, if we have to interface these microcontrollers with Computer we need to convert the TTL/CMOS logic to RS232 logic.

2.2.5 BT HC05

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Bluetooth Module is a Drop-in replacement for wired serial connections, transparent usage. You can use it simply for serial port replacement to establish connection between MCU and GPS, PC to your embedded project / Robot etc. The module can be configured for baud rates 1200 to 115200 bps. This is a Slave mode only Bluetooth Device. If you need a Master/Slave switchable device refer to this product: Bluetooth UART Module.

2.2.6 FLEX SENSOR

A flex sensor or bend sensor is a sensor that measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface. Since the resistance is directly proportional to the amount of bend it is used as goniometer, and often called flexible potentiometer.

2.2.7 ACCELEROMETER

The accelerometer is an electromechanical device that measures the force of acceleration caused by movement or by gravity or by vibration. These forces can be static like gravity force, dynamic senses movement or vibrations. Mathematically, acceleration is a measurement of the change in velocity or speed divided by time. The accelerometer can be used for both academic and consumer purposes. A dynamic accelerometer measures the gravitational pull. If we consider the consumer context, then uses can get a better understanding of the surrounding of an object. It captures each motion of the item, whether it is moving uphill, falling over, tilting, flying horizontally or aligning downward.

2.2.8 ANDROID DEVICE

It is an operating system based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005 Android was unveiled in 2007 along with the founding of the Open Handset Alliance - a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. The first publicly available Smartphone running Android, the HTC Dream, was released on October 22, 2008.

2.2.9 STATUS AREA

Used for LED Indication. It denotes the working Status of power supply, data transfer and program execution.

CHAPTER 3
HARDWARE SECTION

3.1 MICROCONTROLLER -PIC 16F877A

The PIC microcontroller PIC16F877A is one of the most renowned microcontrollers in the industry. This controller is very convenient to use, the coding or programming of this controller is also easier. One of the main advantages is that it can be write-erase as many times as possible because it uses FLASH memory technology. It has a total number of 40 pins and there are 33 pins for input and output. PIC16F877A is used in many pic microcontroller projects. PIC16F877A also have many application in digital electronics circuits.

PIC16F877A finds its applications in a huge number of devices. It is used in remote sensors, security and safety devices, home automation and in many industrial instruments. An EEPROM is also featured in it which makes it possible to store some of the information permanently like transmitter codes and receiver frequencies and some other related data. The cost of this controller is low and its handling is also easy. It’s flexible and can be used in areas where microcontrollers have never been used before as in coprocessor applications and timer functions etc.

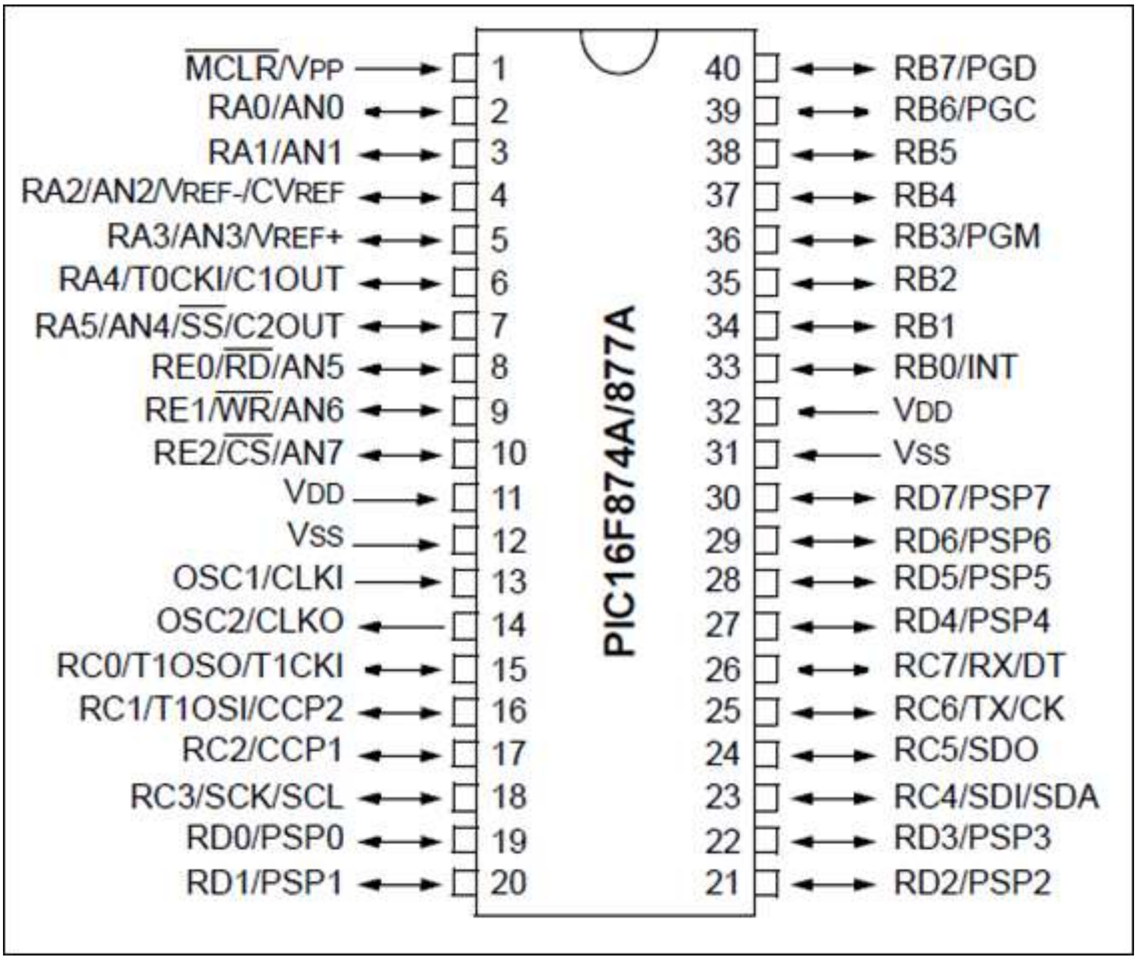


Figure 3.1.1: Pin out of PIC16F874A/877A

3.1.1 Pin Configuration and description of PIC16F877A Microcontroller

As it has been mentioned before, there are 40 pins of this microcontroller IC. It consists of two 8 bit and one 16-bit timer. Capture and compare modules, serial ports, parallel ports and five input/output ports are also present in it.

PIN 1: MCLR

The first pin is the master clear pin of this IC. It resets the microcontroller and is active low, meaning that it should constantly be given a voltage of 5V and if 0 V are given then the controller is reset. Resetting the controller will bring it back to the first line of the program that has been burned into the IC.

PIC16F877 reset

A push button and a resistor are connected to the pin. The pin is already being supplied by constant 5V. When we want to reset the IC, we just have to push the button which will bring the MCLR pin to 0 potential thereby resetting the controller.

PIN 2: RA0/AN0

PORTA consists of 6 pins, from pin 2 to pin 7, all of these are bidirectional input/output pins. Pin 2 is the first pin of this port. This pin can also be used as an analog pin AN0. It is built in analog to digital converter.

PIN 3: RA1/AN1

This can be the analog input 1

PIN 4: RA2/AN2/V ref -

It can also act as the analog input2. Or negative analog reference voltage can be given to it.

PIN 5: RA3/AN3/V ref +

It can act as the analog input 3. Or can act as the analog positive reference voltage.

PIN 6: RA0/T0CKI

To timer0 this pin can act as the clock input pin, the type of output is open drain.

PIN 7: RA5/SS/AN4

This can be the analog input 4. There is synchronous serial port in the controller also and this pin can be used as the slave select for that port.

PIN 8: RE0/RD/AN5

PORTE starts from pin 8 to pin 10 and this is also a bidirectional input output port. It can be the analog input 5 or for parallel slave port it can act as a ‘read control’ pin which will be active low.

PIN 9: RE1/WR/AN6

It can be the analog input 6. And for the parallel slave port it can act as the 'write control' which will be active low.

PIN 10: RE2/CS/A7

It can be the analog input 7, or for the parallel slave port it can act as the 'control select' which will also be active low just like read and write control pins.

PIN 11 and 32: VDD

These two pins are the positive supply for the input/output and logic pins. Both of them should be connected to 5V.

PIN 12 and 31: VSS

These pins are the ground reference for input/output and logic pins. They should be connected to 0 potential.

PIN 13: OSC1/CLKIN

This is the oscillator input or the external clock input pin.

PIN 14: OSC2/CLKOUT

This is the oscillator output pin. A crystal resonator is connected between pin 13 and 14 to provide external clock to the microcontroller. ¼ of the frequency of OSC1 is outputted by OSC2 in case of RC mode. This indicates the instruction cycle rate.

Crystal interfacing with PIC16F877A

PIN 15: RC0/T1OCO/T1CKI

PORTC consists of 8 pins. It is also a bidirectional input output port. Of them, pin 15 is the first. It can be the clock input of timer 1 or the oscillator output of timer 2.

PIN 16: RC1/T1OSI/CCP2

It can be the oscillator input of timer 1 or the capture 2 input/compare 2 output/ PWM 2 output.

PIN 17: RC2/CCP1

It can be the capture 1 input/ compare 1 output/ PWM 1 output.

PIN 18: RC3/SCK/SCL

It can be the output for SPI or I2C modes and can be the input/output for synchronous serial clock.

PIN 23: RC4/SDI/SDA

It can be the SPI data in pin. Or in I2C mode it can be data input/output pin.

PIN 24: RC5/SDO

It can be the data out of SPI in the SPI mode.

PIN 25: RC6/TX/CK

It can be the synchronous clock or USART Asynchronous transmit pin.

PIN 26: RC7/RX/DT

It can be the synchronous data pin or the USART receive pin.

PIN 19,20,21,22,27,28,29,30:

All of these pins belong to PORTD which is again a bidirectional input and output port. When the microprocessor bus is to be interfaced, it can act as the parallel slave port.

PIN 33-40: PORT B

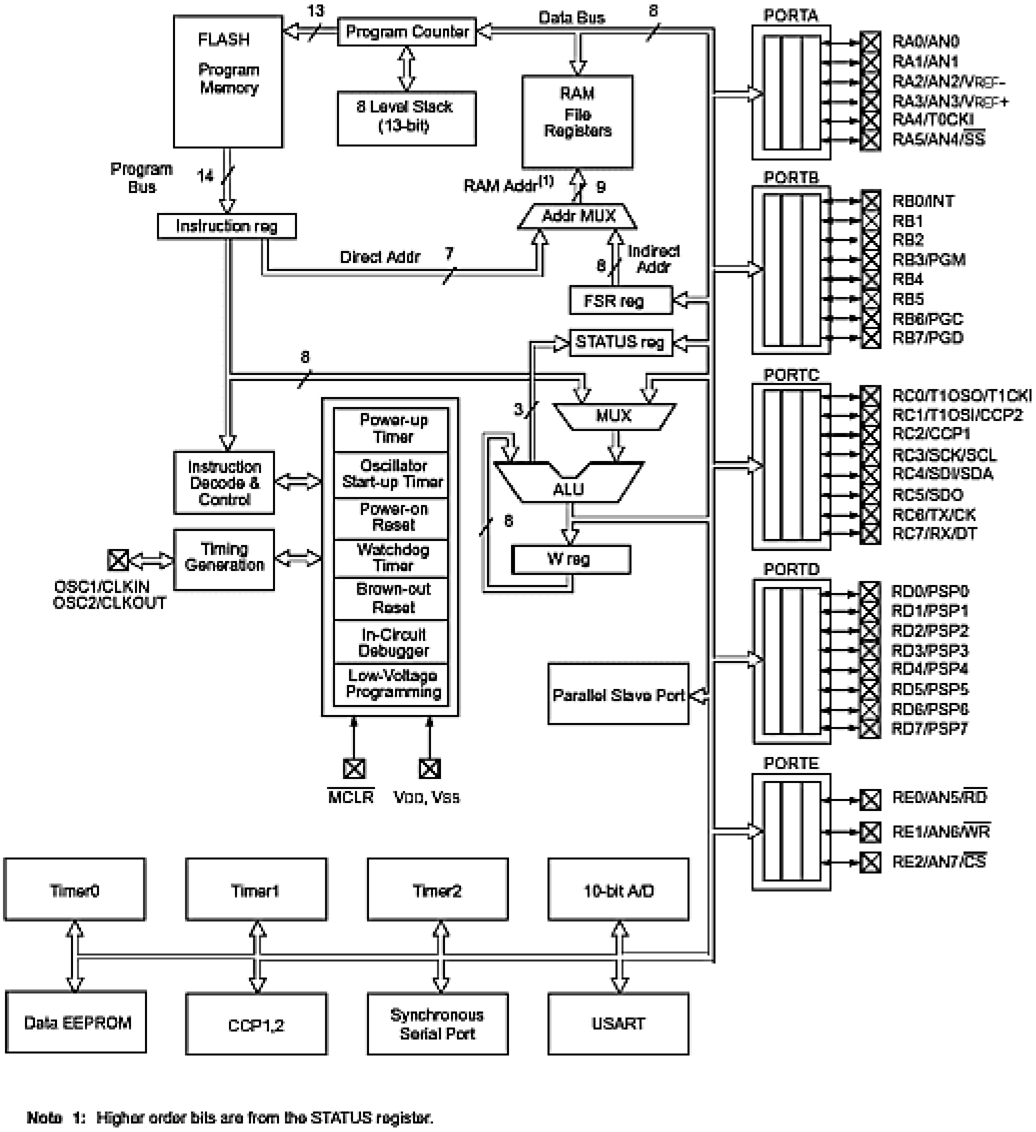
All these pins belong to PORTB. Out of which RB0 can be used as the external interrupt pin and RB6 and RB7 can be used as in-circuit debugger pins.

3.1.2 FEATURES OF PIC16F877A

- It is available in three packages known as PDIP, QFN, and TQFP. The first one comes with a 40-pin layout design while remaining two contains 44 pins on each layout.
- This PIC version, like other models in the PIC community, contains everything that is required to make an embedded system and drive automation.
- The PIC16F887 incorporates 256 bytes of EEPROM data memory, 368 bytes of RAM, and program memory of 8K
- Apart from self-programming capability, it also contains 2 Comparators, 10-bit Analog-to- Digital (A/D) converter with 14 channels, and capture, compare and PWM functions. IC16F877 is a 40-pin (for PDIP package) and 8-bit CMOS PIC Microcontroller that comes with Nano watt technology. Economical price and user-friendly architecture make this device easy to use and easy to configure.

3.1.3 Architecture of PIC 16F877A

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

Figure 3.1.3.1: Architecture of PIC 16F874A/77A

CENTRAL PROCESSOR UNIT (CPU)

We are not going to bore you with the operation of the CPU at this stage. However, we will just state that the CPU is manufactured with RISC technology as it is an important factor when deciding which microcontroller to use.

RISC stands for Reduced Instruction Set Computer, which gives the PIC16F877 two great advantages:

• The CPU only recognizes 35 simple instructions. Just to mention that in order to program other microcontrollers. In assembly language it is necessary to know more than 200 instructions by heart.

• The execution time is the same for almost all instructions, and lasts for 4 clock cycles. The oscillator frequency is stabilized by a quartz crystal. The execution time of jump and branch instructions is 2 clock cycles. It means that if the microcontroller's operating speed is 20MHz, the execution time of each instruction will be 200nS, i.e. the program will execute5 million instructions per second.

MEMORY

The PIC16F887 has three types of memory ROM, RAM and EEPROM. All of them will be separately discussed since each has specific functions, features and organization.

ROM MEMORY

ROM memory is used to permanently save the program being executed. This is why it is often called 'program memory'. The PIC16F887 has 8Kb of ROM (in total of 8192 locations). Since the ROM memory is made with FLASH technology, its contents can be changed by providing a special programming voltage (13V). However, it is not necessary to explain it in detail as being automatically performed by means of a special program on the PC and a simple electronic device called the programmer.

EEPROM MEMORY

Similar to program memory, the contents of EEPROM is permanently saved, even when the power goes off. However, unlike ROM, the contents of EEPROM can be changed during the operation of the microcontroller. This is why this memory (256 locations) is perfect for permanently saving some of the results created and used during the operation.

RAM MEMORY

This is the third and the most complex part of microcontroller memory. In this case, it consists of two parts: general-purpose registers and special-function registers (SFR). All these registers are divided in four memory banks to be explained later in the chapter. Even though both groups of registers are cleared when power goes off and even though they are manufactured in the same manner and act in a similar way, their functions do not have many things in common.

GENERAL-PURPOSE REGISTERS

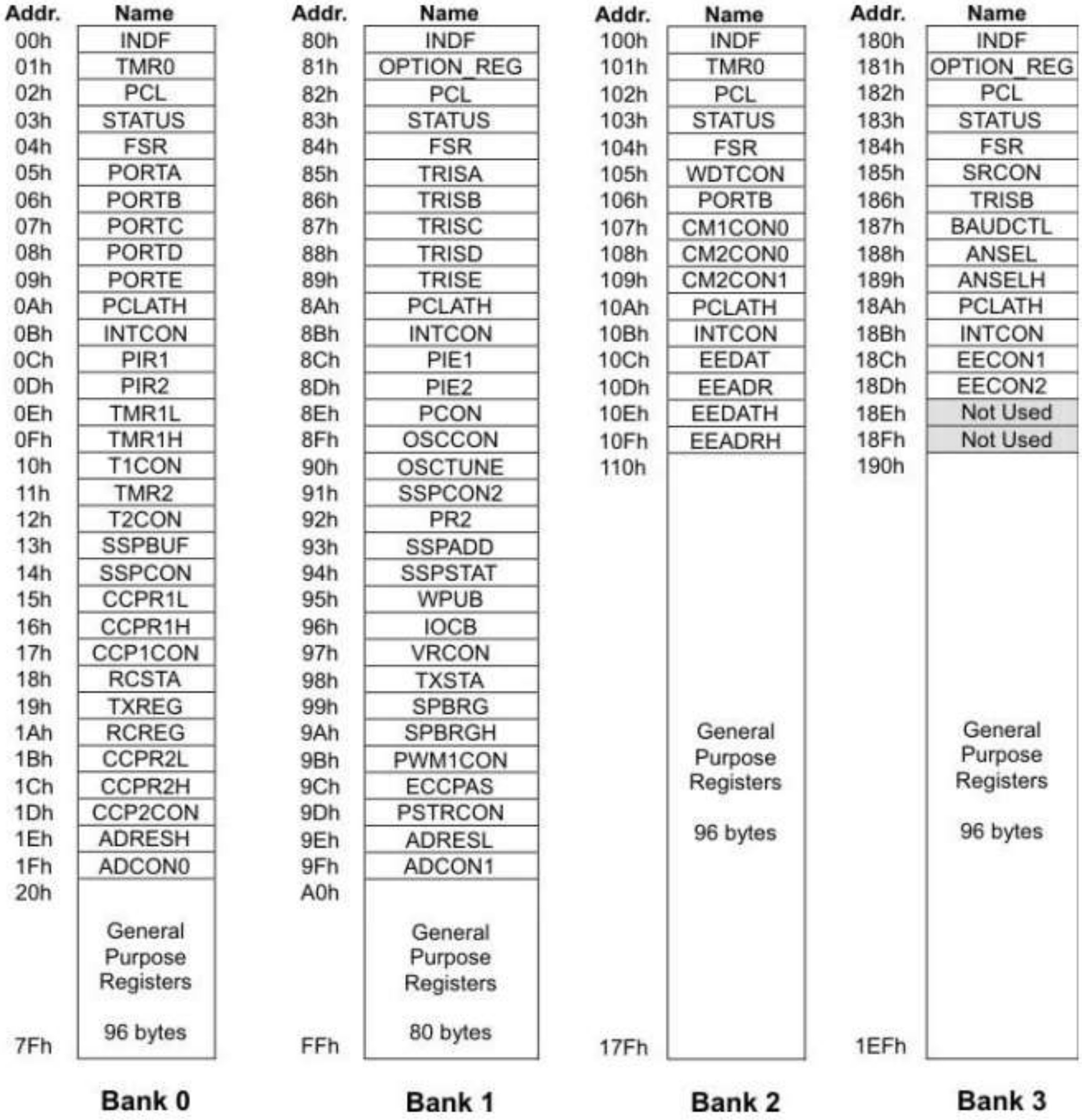
General-purpose registers are used for storing temporary data and results created during operation. For example, if the program performs counting (products on the assembly line), it is necessary to have a register which stands for what we in everyday life call 'sum'. Since the microcontroller is not creative at all, it is necessary to specify the address of some general-purpose register and assign it that function. A simple program to increment the value of this register by 1, after each product passes through a sensor, should be created. Now the microcontroller can execute the program as it knows what and where the sum to be incremented is.

SPECIAL FUNCTION REGISTERS (SFR)

Special-function registers are also RAM memory locations, but unlike general-purpose registers, their purpose is predetermined during manufacturing process and cannot be changed. Since their bits are connected to particular circuits on the chip (A/D converter, serial communication module, etc.), any change of their contents directly affects the operation of the microcontroller or some of its circuits. For example, the ADCON0 register controls the operation of A/D converter. By changing its bits it is determined which port pin is to be configured as converter input, the moment conversion is to start as well as the speed of conversion. Another feature of these memory locations is that they have their names (both registers and their bits), which considerably simplifies the process of writing a program. Since high-level programming languages can use the list of all registers with their exact addresses, it is enough to specify the name of a register in order to read or change its contents.

RAM MEMORY BANKS

The RAM memory is partitioned into four banks. Prior to accessing any register during program writing (in order to read or change its contents), it is necessary to select the bank which contains that register. Two bits of the STATUS register are used for bank selection to be discussed later. In order to simplify the operation, the most commonly used SFRs have the same address in all banks, which enables them to be easily accessed. Handling banks may be difficult only if you write a program in assembly language. When using higher programming languages such as C and compilers such as microC PRO for PIC, all you have to do is to specify the register name. On the basis of that, the compiler selects necessary bank and appropriate instructions used for bank selection will be built in the code during the process of compilation. You have been using only assembly language so far and this is the first time you use the C compiler.



Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name
00h	INDF	80h	INDF	100h	INDF	180h	INDF
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h	WDTCON	185h	SRCON
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h	CM1CON0	187h	BAUDCTL
08h	PORTD	88h	TRISD	108h	CM2CON0	188h	ANSEL
09h	PORTE	89h	TRISE	109h	CM2CON1	189h	ANSELH
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDAT	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEDATH	18Eh	Not Used
0Fh	TMR1H	8Fh	OSCCON	10Fh	EEADRH	18Fh	Not Used
10h	T1CON	90h	OSCTUNE	110h	General Purpose Registers 96 bytes	190h	General Purpose Registers 96 bytes
11h	TMR2	91h	SSPCON2				
12h	T2CON	92h	PR2				
13h	SSPBUF	93h	SSPADD				
14h	SSPCON	94h	SSPSTAT				
15h	CCPR1L	95h	WPUB				
16h	CCPR1H	96h	IOCB				
17h	CCP1CON	97h	VRCON				
18h	RCSTA	98h	TXSTA				
19h	TXREG	99h	SPBRG				
1Ah	RCREG	9Ah	SPBRGH				
1Bh	CCPR2L	9Bh	PWM1CON				
1Ch	CCPR2H	9Ch	ECCPAS				
1Dh	CCP2CON	9Dh	PSTRCON				
1Eh	ADRESH	9Eh	ADRESL				
1Fh	ADCON0	9Fh	ADCON1				
20h	General Purpose Registers 96 bytes	A0h	General Purpose Registers 80 bytes				
7Fh		FFh		17Fh		1EFh	
	Bank 0		Bank 1		Bank 2		Bank 3

Figure 3.1.3.2: Register Bank

STACK

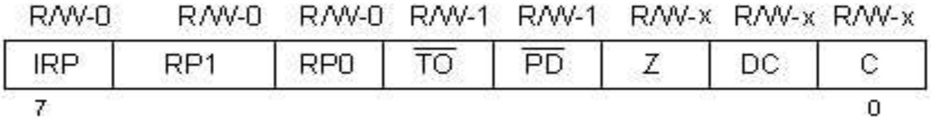
A part of RAM used as stack consists of eight 13-bit registers. Before the microcontroller starts to execute a subroutine (CALL instruction) or when an interrupt occurs, the address of the first next instruction to execute is pushed onto the stack, i.e. one of its registers. Thanks to that the microcontroller knows from where to continue regular program execution upon a subroutine or an interrupt execution. This address is cleared after returning to the program because there is no need to save it any longer, and one location of the stack becomes automatically available for further use. It is important to bear in mind that data is always circularly pushed onto the stack. It means that after the stack has been pushed eight times, the ninth push overwrites the value that was stored with the first push. The tenth push overwrites the second push and so on. Data overwritten in this way is not recoverable. In addition, the programmer cannot access these registers for write or read and there is no Status bit to indicate stack overflow or stack underflow conditions. For this reason, it is necessary to take special care of it during program writing.

INTERRUPT SYSTEM

The first thing the microcontroller does when an interrupt request arrives is to execute the current instruction and then stops the regular program execution. As a result, the current program memory address is automatically pushed onto the stack and the default address (predefined by the manufacturer) is written to the program counter. The location from where the program proceeds with execution is called an interrupt vector. For the PIC16F877 microcontroller, this address is 0004h. As seen in figure below, the location containing the interrupt vector is passed over during regular program execution.

STATUS Register

The STATUS register contains: the arithmetic status of data in the W register, the RESET status and the bank select bits for data memory.

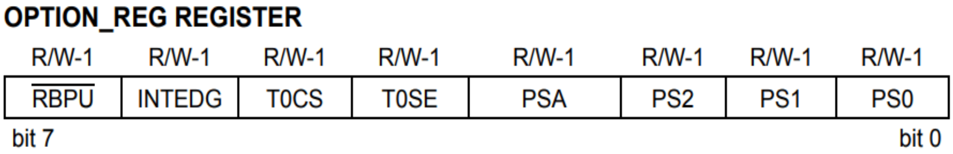


R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
7							0

Figure 3.1.3.3: Status Register

OPTION_REG Register

The OPTION_REG register contains various control bits to configure Timer0/WDT prescaler, timer TMR0, external interrupt and pull-ups on PORTB.



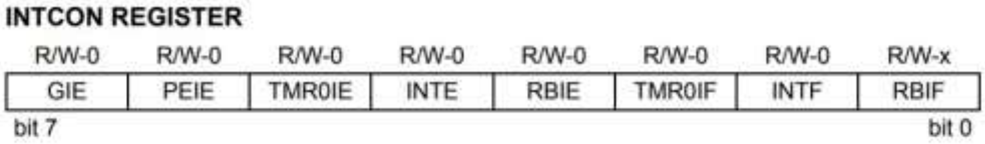
OPTION_REG REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Figure 3.1.3.5: Option Register

INTCON Register

The INTCON register contains various enable and flag bits for TMR0 register overflow, PORTB change and external INT pin interrupts.



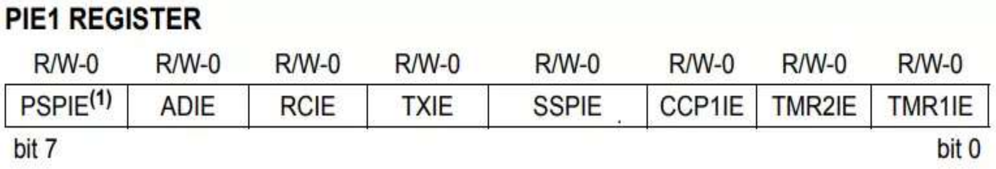
INTCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

Figure 3.1.3.6: INTCON Register

PIE1 Register

The PIE1 register contains peripheral interrupt enable bits.



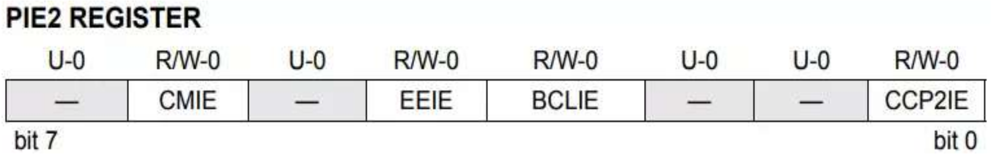
PIE1 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE$^{(1)}$	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Figure 3.1.3.7: PIE1 Register

PIE2 Register

The PIE2 Register also contains various interrupt enable bits.



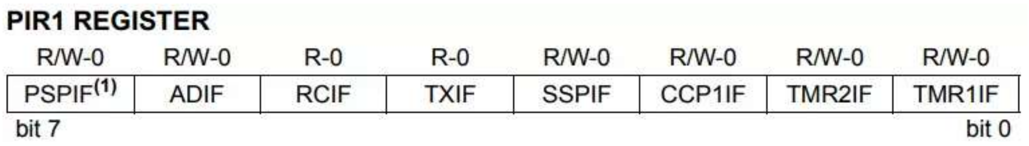
PIE2 REGISTER

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

Figure 3.1.3.8: PIE2 Register

PIR1 Register

The PIR1 register contains the interrupt flag bits.



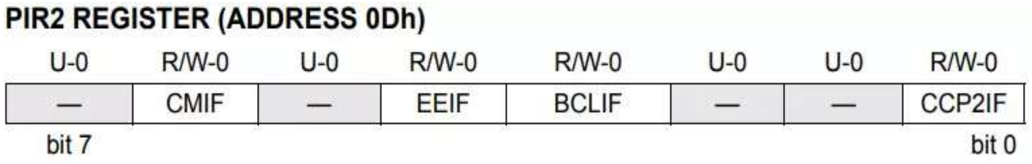
PIR1 REGISTER

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Figure 3.1.3.9:PIR1 Register

R2 Register

The PIR2 register contains the interrupt flag bits.



PIR2 REGISTER (ADDRESS 0Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF
bit 7							bit 0

Figure 3.1.3.10: PIR2 Register

PCON Register

The PCON register contains only two flag bits used to differentiate between Power-on reset, Brown-out reset, Watchdog Timer reset and external reset over the MCLR pin.

PCL AND PCLATH REGISTERS

The size of the program memory of the PIC16F887 is 8K and has 8192 locations for program storing. For this reason, the program counter must be 13-bits wide (213 = 8192). To enable access to any program memory location during operation, it is necessary to access its address through SFRs. Since all SFRs are 8-bits wide, this addressing register is ‘artificially’ created by dividing its 13 bits into two independent registers PCLATH and PCL.

INDIRECT ADDRESSING REGISTERS

In addition to direct addressing, which is logical and clear (it is sufficient to specify the address of a register to read its contents), this microcontroller is capable of performing indirect addressing by means of the INDF and FSR registers. It sometimes makes the process of writing a program easier. The whole procedure is enabled because the INDF register is not true one (physically does not exist), but only specifies the register the address of which is located in the FSR register. For this reason, write or read from the INDF register actually means write or read from the register the address of which is located in the FSR register. In other words, registers addresses are specified in the FSR register, and their content is stored in the INDF register.

INPUT/OUTPUT PORTS

In order to synchronize the operation of I/O ports with the internal 8-bit organization of the microcontroller, they are, similar to registers, grouped into five ports denoted by A, B, C, D and E. All of them have several features in common: For practical reasons, many I/O pins are multifunctional. If a pin performs any of these functions, it may not be used as a general-purpose input/output pin. Every port has its ‘satellite’, i.e. the corresponding TRIS register: TRISA, TRISB, TRISC etc. which determines the performance of port bits, but not their contents.

PORTA and TRISA Register

Port A is an 8-bit wide, bidirectional port. Bits of the TRISA and ANSEL registers control the Port A pins. All Port A pins act as digital inputs/outputs. Five of them can also be analog inputs (denoted by AN)

RA0 = AN0 (determined by the ANS0 bit of the ANSELregister)
RA1 = AN1 (determined by the ANS1 bit of the ANSELregister)
RA2 = AN2 (determined by the ANS2 bit of the ANSELregister)
RA3 = AN3 (determined by the ANS3 bit of the ANSELregister)
RA5 = AN4 (determined by the ANS4 bit of the ANSELregister)

ULPWU UNIT

The microcontroller is commonly used in devices which operate periodically and completely independently using a battery power supply. Minimum power consumption is one of the priorities here. Typical examples of such applications are: thermometers, fire detection sensors and the like. It is known that a reduction in clock frequency reduces the power consumption, thus one of the most convenient solutions to this problem is to slow down the clock, i.e. to use 32 KHz quartz crystal instead of 20MHz.

PORTB and TRISB register

Port B is an 8-bit wide, bidirectional port. Bits of the TRISB register determine the function of its pins.

RB0 = AN12 (determined by the ANS12 bit of the ANSELH register)
RB1 = AN10 (determined by the ANS10 bit of the ANSELH register)
RB2 = AN8 (determined by the ANS8 bit of the ANSELH register)
RB3 = AN9 (determined by the ANS9 bit of the ANSELH register)
RB4 = AN11 (determined by the ANS11 bit of the ANSELH register)
RB5 = AN13 (determined by the ANS13 bit of the ANSELH register)

PIN RB0/INT

The RB0/INT pin is the only ‘true’ external interrupt source. It can be configured to react to signal raising edge (zero-to-one transition) or signal falling edge (one-to-zero transition). The INTEDG bit of the OPTION_REG register selects the appropriate signal.

RB6 AND RB7 PINS

The PIC16F887 does not have any special pins for programming (the process of writing a program to ROM).Port pins, normally available as general-purpose I/O pins, are used for this purpose. To be more precise, it is about port B pins used for clock (RB6) and data transfer (RB7) during program loading. Besides, it is necessary to apply power supply voltage VDD (5V) as well as appropriate voltage VPP (12-14V) for FLASH Memory programming. During programming, VPP voltage is applied to the MCLR pin. You don’t have to think of all details concerning this process, nor which one of these voltages is applied first since the programmer’s electronics is in charge of that. What is very important here is that the program may be loaded to the microcontroller even after soldering it onto the target device. Normally, the loaded program can also be changed in the same way. This function is called ICSP In order to use it properly, it is necessary to plan ahead.

PORTC and TRISC register

Port C is an 8-bit wide, bidirectional port. Bits of the TRISC register determine the function of its pins. Similar to other ports, a logic one (1) in the TRISC register configures the appropriate portC pin as an input

PORTD and TRISD register

Port D is an 8-bit wide, bidirectional port. Bits of the TRISD register determine the function of its pins. A logic one (1) in the TRISD register configures the appropriate portD pin as an input.

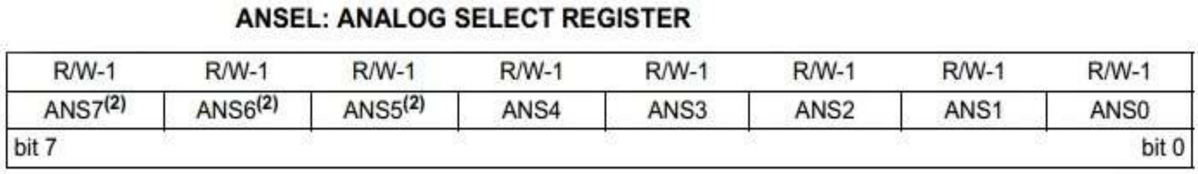
PORTE and TRISE register

Port E is a 4-bit wide, bidirectional port. The TRISE register’s bits determine the function of its pins. Similar to other ports, a logic one (1) in the TRISE register configures the appropriate PortE pin as an input. The exception is the RE3 pin which is always configured as an input.

RE0 = AN5 (determined by the ANS5 bit of the ANSELregister);
RE1 = AN6 (determined by the ANS6 bit of the ANSELregister); and
RE2 = AN7 (determined by the ANS7 bit of the ANSELregister)

ANSEL and ANSELH register

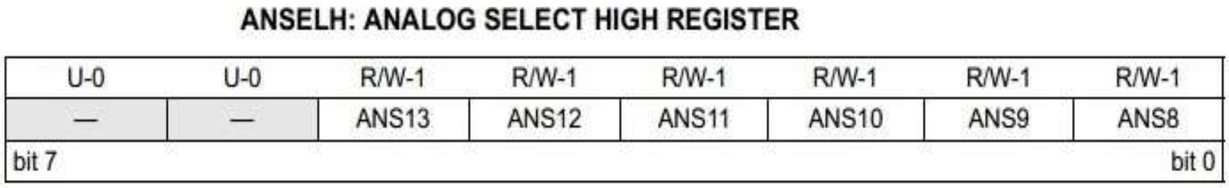
The ANSEL and ANSELH registers are used to configure the input mode of an I/O pin to analog or digital.



ANSEL: ANALOG SELECT REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANS7(2)	ANS6(2)	ANS5(2)	ANS4	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

Figure 3.1.3.11:ANSEL Register



ANSELH: ANALOG SELECT HIGH REGISTER

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
bit 7							bit 0

Figure 3.1.3.12: ANSELH Register

TIMER TMR0

The timer TMR0 has a wide range of application in practice. Very few programs don’t use it in some way. It is very convenient and easy to use for writing programs or subroutines for generating pulses of arbitrary duration, time measurement or counting external pulses (events) with almost no limitations.

TIMER TMR1

Timer TMR1 module is a 16-bit timer/counter, which means that it consists of two registers (TMR1L andTMR1H). It can count up 65.535 pulses in a single cycle, i.e. before the counting starts from zero similar to the timer TMR0, these registers can be read or written to at any moment. In case an overflow occurs, an interrupt is generated if enabled.

ADRESH and ADRESL Registers

The result obtained after converting an analog value into digital is a10-bit number that is to be stored in the ADRESH and ADRESL registers. There are two ways of handling it - left and right justification which simplifies its use to a great extent. The format of conversion result depends on the ADFM bit of the ADCON1 register. In the event that the A/D converter is not used, these registers may be used as general-purpose registers.

A/D ACQUISITION REQUIREMENTS

In order to enable the ADC to meet its specified accuracy, it is necessary to provide a certain time delay between selecting specific analog input and measurement itself. This time is called 'acquisition time' and mainly depends on the source impedance. There is an equation used to calculate this time accurately, which in the worst case amounts to approximately 20uS. So, if you want the conversion to be accurate, don’t forget this important detail

3.2 POWERSUPPLY

In most of electronic products or projects we need a power supply for converting mains AC voltage to a regulated DC voltage. Regulated power supply converts unregulated AC to a constant DC. A regulated power supply is used to ensure that the output remains constant even if the input changes. A regulated DC power supply is also known as a linear power supply, it is an embedded circuit and consists of various blocks. The regulated power supply will accept an AC input and give a constant DC output. The figure below shows the block diagram of a typical regulated DC power supply.

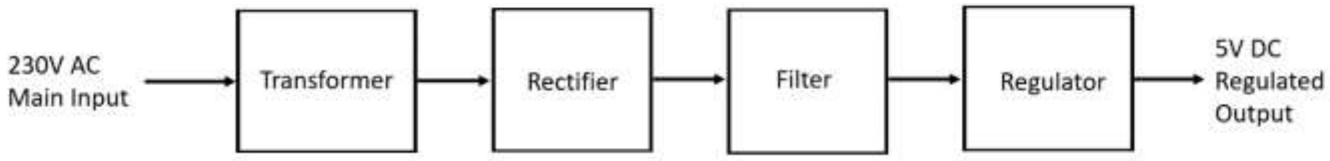


Figure 3.2.1: Block diagram of transformer

Blocks are described in more detail below:

Transformer - step-down high voltage AC mains to low voltage AC
Rectifier - converts to AC to DC but the DC output is varying
Filter - Smooth the DC from varying greatly to a small ripple
Regulator - eliminates ripple by setting DC output to a fixed voltage

TRANSFORMER

Transformer converts AC electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity is AC.

Step-up transformer increase voltage, step-down transformer reduces voltage. Most power supplies use a step down transformer to reduce the dangerously high mains voltage (230V in UK) to safer low voltage.

The input coil is called the primary and output coil is called the secondary. There is no electrical connection between the coils, instead they are linked by an alternating magnetic field created in the soft-iron core of the transformer. The two lines in the middle of the circuit symbol represent the core.

Transformers waste very little power out is (almost) equal to the power in. Note that as voltage is step down current is step up.

The ratio of the number of turns on each coil, called the turn’s ratio, determines the ratio of the voltage. A step-down transformer as a large number of turns on its primary (input) coil which is connected to the high voltage mains supply, and a small number of turns on its secondary (output) coil gives low output voltage

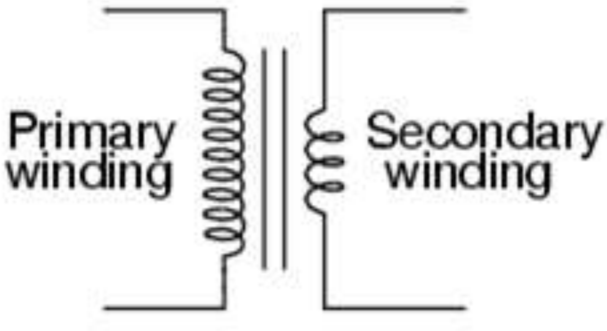


Figure 3.2.2: Step down Transformer

The low voltage AC output is suitable for lamps, heaters and special AC motors. It is not suitable for electronic circuits unless they include a rectifier and smoothing capacitor Transformers convert AC electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity in AC.

Step-up transformer increase voltage, step-down transformer reduce voltage. Most power supplies use a step down transformer to reduce the dangerously high mains voltage (230V) to a safer low voltage.

The input coil is called the primary and the output coil is called the secondary. There is no electrical connection between the two coils, instead they are linked by an alternative magnetic field created in the soft-iron core of the transformer. The two lines in the middle of the circuit symbol represent the core. Transformers waste very little power so the power out is (almost) equal to the power in. Note that as voltage is stepped down current is stepped up.

The ratio of the number of turns on each coil, called the turn’s ratio, determines the ratio of the voltages. A step down transformer has a large number of turns on its primary (input) coil which is connected to the high voltage mains supply, and a small number of turns on its secondary (output) coil to give a low output voltage.

Figure 3.2.3: 12V-0-12V Transformer

RECTIFIER

There are several ways of connecting diodes to make a rectifier to convert AC to DC. The bridge rectifier is the most important and it produces full-wave varying DC. A full-wave rectifier can also be made from just two diodes if a centre-tap transformer is used, but this is method is rarely used now that diodes are cheaper. A single diode can be used as a rectifier but it only uses the positive (+) parts of the AC wave to produce half wave varying DC.

BRIDGE RECTIFIER

Bridge rectifier consists of four diodes which are connected in the form a bridge. The diode is an uncontrolled rectifier which will conduct only forward bias and will not conduct during the reverse bias. If the diode anode voltage is greater than the cathode voltage then the diode is said to be in forward bias. During positive half cycle, diodes D2 and D4 will conduct and during negative half cycle diodes D1 and D3 will conduct. Thus, AC is converted into DC; here the obtained is not a pure DC as it consists of pulses. Hence, it is called as pulsating DC power. But voltage drop across the diodes is 1.4V; therefore, the peak voltage at the output of this rectifier circuit is 15V approx.

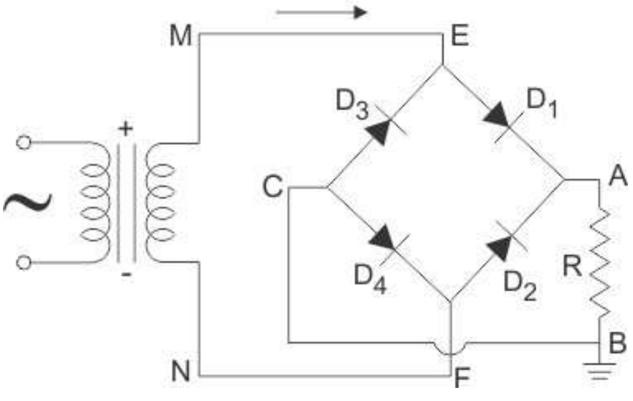


Figure 3.2.4: Circuit of Bridge Rectifier

FILTER

The rectified voltage from the rectifier is a pulsating DC voltage having very high ripple content. But this is not we want, we want a pure ripple free DC waveform. Hence a filter is used. Different types of filters are used such as capacitor filter, LC filter, Choke input filter, π type filter.

As the instantaneous voltage starts increasing the capacitor charges, it charges until the waveform reaches its peak value. When the instantaneous value starts reducing the capacitor starts discharging exponentially and slowly through the load (input of the regulator in this case). Hence, an almost constant DC value having very less ripple content is obtained.

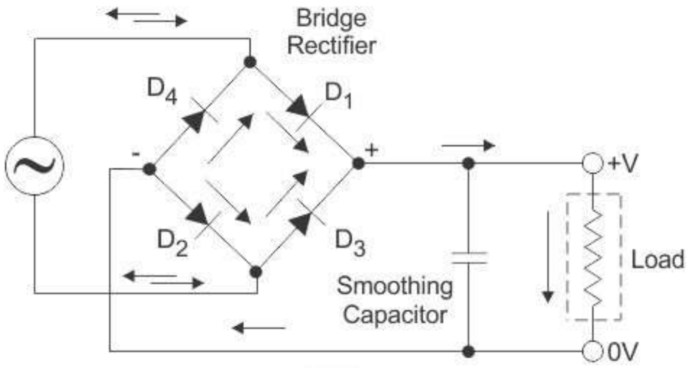


Figure 3.2.5: Smoothing Capacitor

REGULATOR

This is the last block in a regulated DC power supply. The output voltage or current will change or fluctuate when there is a change in the input from ac mains or due to change in load current at the output of the regulated power supply or due to other factors like temperature changes. This problem can be eliminated by using a regulator. A regulator will maintain the output constant even when changes at the input or any other changes occur. Transistor series regulator, Fixed and variable IC regulators or a zener diode operated in the zener region can be used depending on their applications. IC's like 78XX and 79XX (such as the IC 7805) are used to obtained fixed values of voltages at the output.

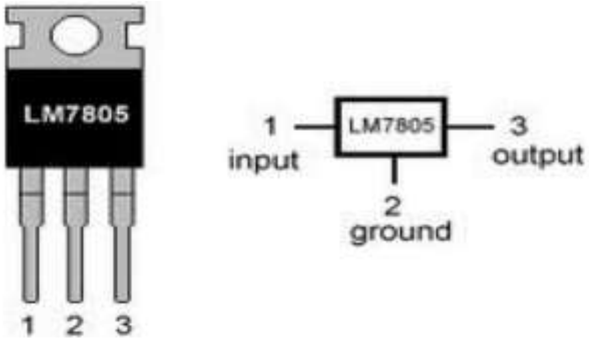


Figure 3.2.6: 7805 Voltage Regulator

Circuit Diagram

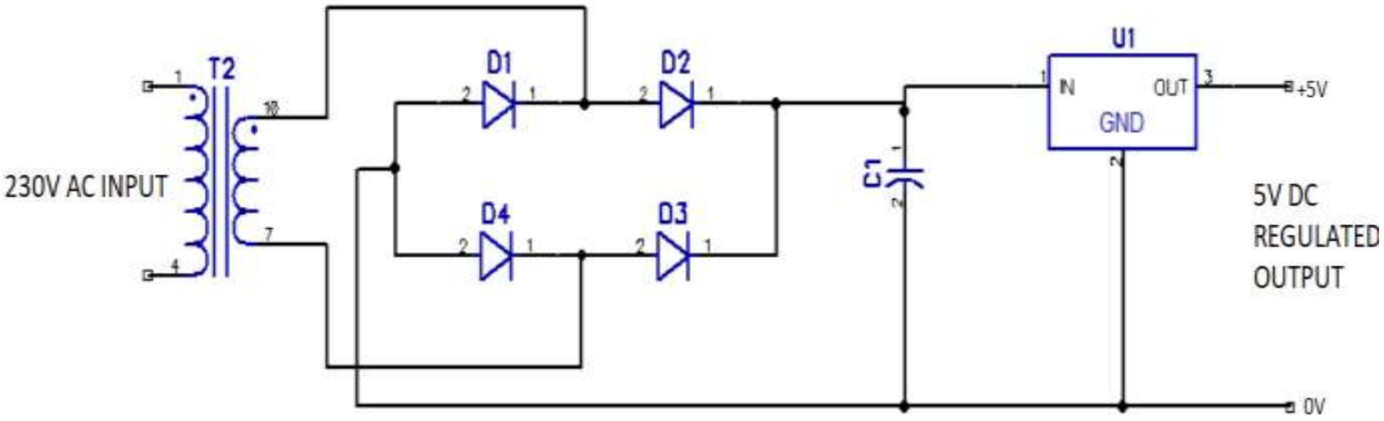
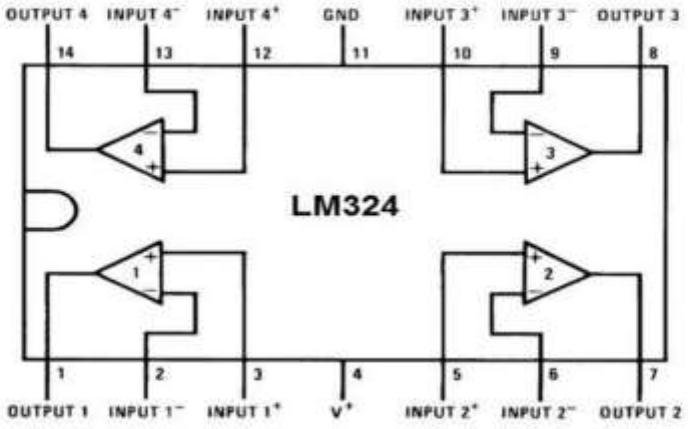


Figure 3.2.7: Power supply circuit

3.3 LM 324

LM324 is a 14pin IC consisting of four independent operational amplifiers (op-amps) compensated in a single package. Op-amps are high gain electronic voltage amplifier with differential input and, usually, a single-ended output. The output voltage is many times higher than the voltage difference between input terminals of an op-amp. These op-amps are operated by a single power supply LM324 and need for a dual supply is eliminated. They can be used as amplifiers, comparators, oscillators, rectifiers etc. The conventional op-amp applications can be more easily implemented with LM324.



3.3.1: Pin out of LM324

Pin Description of LM324

Pin No	Function	Name
1	Output of 1st comparator	Output 1
2	Inverting input of 1st comparator	Input 1-
3	Non-inverting input of 1st comparator	Input 1+
4	Supply voltage; 5V (up to 32V)	VCC
5	Non-inverting input of 2nd comparator	Input 2+
6	Inverting input of 2nd comparator	Input 2-
7	Output of 2nd comparator	Output 2
8	Output of 3rd comparator	Output 3
9	Inverting input of 3rd comparator	Input 3-
10	Non-inverting input of 3rd comparator	Input 3+
11	Ground (0V)	Ground
12	Non-inverting input of 4th comparator	Input 4+
13	Inverting input of 4th comparator	Input 4-
14	Output of 4th comparator	Output 4

Features and Specifications of LM324 IC

- Integrated with four Op-Amps in a single package
- Wide power supply Range
    1. Singe supply – 3V to 32V
    2. Dual supply – ±1.5V to ±16V
- Low Supply current – 700uA
- Single supply for four op-amp operation enables reliable operation
- Operating ambient temperature – 0˚C to 70˚C
- Soldering pin temperature – 260 ˚C (for 10 seconds – prescribed)

LM 324 IC Applications

The applications of IC LM324 include the following.

- Generally, this comparator is employed in the robot-like line following
- By using this IC, the conventional op-amp applications can be implemented very simply.
- This IC can be used as oscillators, rectifiers, amplifiers, comparators etc.

3.4 MAX 232

The MAX232 is an IC created in 1987 by maxim Integrated Products that converts signals from an RS232 serial 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.

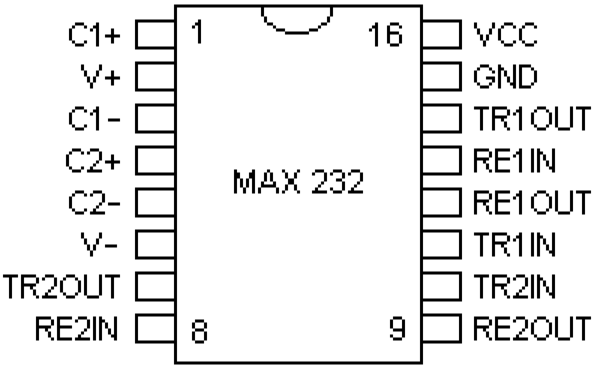


Figure 3.4.1: Pin out of MAX232

The drivers provide RS-232 voltage level outputs (approx. +7.5v) from a single +5v 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 voltage outside the 0v to +5v 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 +25v), to standard 5v TTL Levels. These receivers have a typical threshold of 1.3v, and a typical hysteresis of 0.5v. It is a line driver used to convert the TTL signal to RS-232 and vice versa. Our system needed for MAX232 IC for interfacing GSM modem, GPS and Zigbee with PIC microcontroller

Features of MAX232

- TTL/CMOS to RS232 Converter IC
- Can support two conversion at the same time (Dual drivers and Receivers)
- Easy to set-up and initialize
- Operating speed is 120kbit/s and operating current is 8mA
- ±30-V Input Levels
- Available in 16-pin PDIP, SO, SOIC packages

Pin Description of MAX232

Pin Number	Pin Name	Description
1	C1 +	Connects to the positive end of First capacitor
2	V+	Connects to one end of the capacitor & other end is grounded
3	C1 -	Connects to the negative end of First Capacitor
4	C2+	Connects to positive end of second capacitor
5	C2-	Connects to negative end of second capacitor
6	V-	Connects to one end of the capacitor & other end is grounded
7	T2 OUT	Transmission pin of second converter module for RS232 cable
8	R2 IN	Reception pin of second converter module for RS232 cable
9	R2 OUT	Reception pin of second converter module for Microcontroller(Rx)
10	T2 IN	Transmission pin of second converter module for Microcontroller (TX)
11	T1 IN	Transmission pin of first converter module for Microcontroller (TX)
12	R1 OUT	Reception pin of first converter module for Microcontroller(Rx)
13	R1 IN	Reception pin of first converter module for RS232 cable
14	T1 OUT	Transmission pin of first converter module for RS232 cable
15	Ground	Connects to the ground of the circuit
16	VCC	Connects to the supply voltage typically +5V

3.5 BLUETOOTH MODULE HC05

HC-05 is a compact Bluetooth Module (5V Serial TTL). The module has built-in Voltage regulator and 3V3 to 5V level converter that can be used to interface with 5V Micro-controllers. The module has only 5 pins (Standard 2.54mm berg strip) VCC, GND, TX, RX and RESET. The module is factory configured in Transparent Mode and hence there is no command required for normal operation.

The HC-05 is a Drop-in replacement for wired serial connections, transparent usage. We can use it simply for serial port replacement to establish connection between MCU and GPS, PC to our embedded project / Robot etc. Any serial stream from 9600 to 115200 bps can be passed seamlessly from your PC/PDA/MOBILE to your target board.

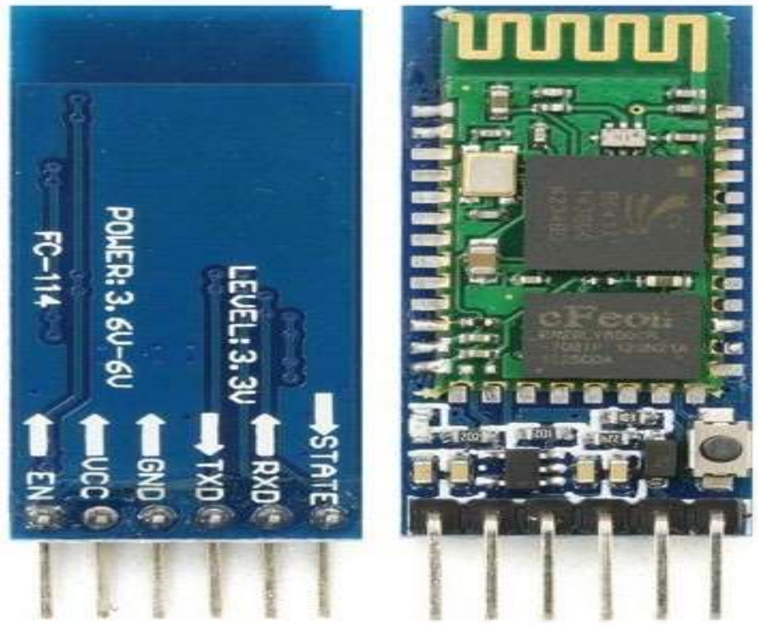


Figure 3.5.1: Bluetooth Module HC05

Pin out for the module are:

- GND Ground- Ground Level of Power supply,
- 5V Power Supply pin for Power Supply Input (5V)
- RXD Receive Pin for Data Reception
- TXD Transmit Pin for Data Transmission
- RST Reset- Reset Input (Internally Pulled-Up)

HC-05 Technical Specifications

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

3.6 FLEX SENSOR

A flex sensor or bend sensor is a sensor that measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface. Since the resistance is directly proportional to the amount of bend it is used as goniometer, and often called flexible potentiometer.

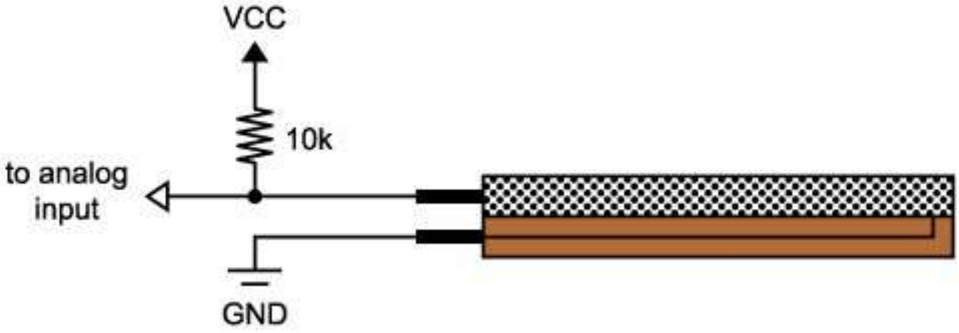


Figure 3.6.1: Flex Sensor

Flex sensors are usually available in two sizes. One is 2.2 inch and another is 4.5 inch. Although the sizes are different the basic function remains the same. They are also divided based on resistance. There are LOW resistance, MEDIUM resistance and HIGH resistance types. Choose the appropriate type depending on requirement. Here we are going to discuss 2.2inch Flex sensor that is FS-L-0055.A simple flex sensor 2.2" in length. As the sensor is flexed, the resistance across the sensor increases.

The resistance of the flex sensor changes when the metal pads are on the outside of the bend (text on inside of bend). Connector is 0.1" spaced and bread board friendly

Flex sensor Features and Specifications

- Operating voltage of FLEX SENSOR: 0-5V
- Can operate on LOW voltages
- Power rating: 0.5Watt (continuous), 1 Watt (peak)
- Operating temperature: -45ºC to +80ºC
- Flat Resistance: 25K Ω
- Resistance Tolerance: ±30%
- Bend Resistance Range: 45K to 125K Ohms (depending on bend)

How to Use Flex sensor

Flex sensor is basically a variable resistor whose terminal resistance increases when the sensor is bent. So this sensor resistance increases depends on surface linearity. So it is usually used to sense the changes in linearity.

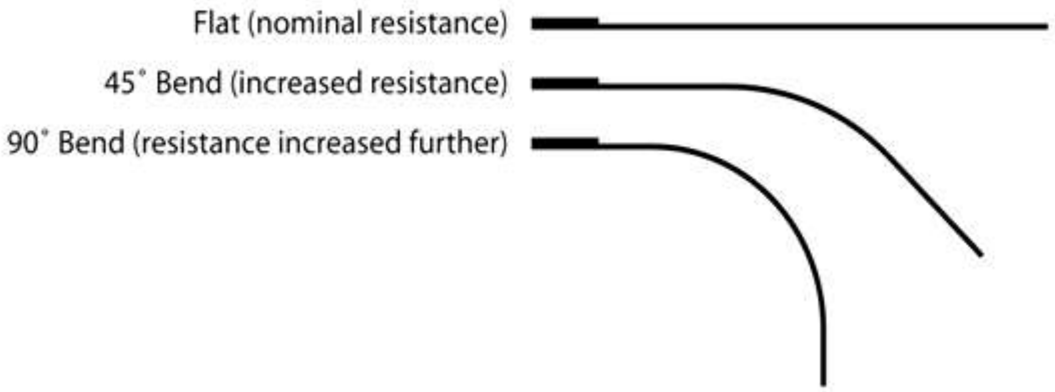

Flat (nominal resistance)
45˚ Bend (increased resistance)
90˚ Bend (resistance increased further)

Figure 3.6.2: Flex Sensor bending

When the surface of flex sensor is completely linear it will be having its nominal resistance. When it is bent 45º angle the flex sensor resistance increases to twice as before. And when the bent is 90º the resistance could go as high as four times the nominal resistance. So the resistance across the terminals rises linearly with bent angle. So in a sense the flex sensor converts flex angle to resistance parameter. For convenience convert this resistance parameter to voltage parameter, voltage divider circuit are used.

3.7 ACCELEROMETER

An accelerometer is a device that measures proper acceleration. Proper acceleration, being the acceleration (or rate of change of velocity) of a body in its own instantaneous rest frame, is not the same as coordinate acceleration, being the acceleration in a fixed coordinate system. By contrast, accelerometers in free fall (falling toward the center of the Earth at a rate of about 9.81 m/s2) will measure zero. An accelerometer is an electromechanical device that will measure acceleration force. It shows acceleration, only due to cause of gravity i.e. g force. It measures acceleration in g unit

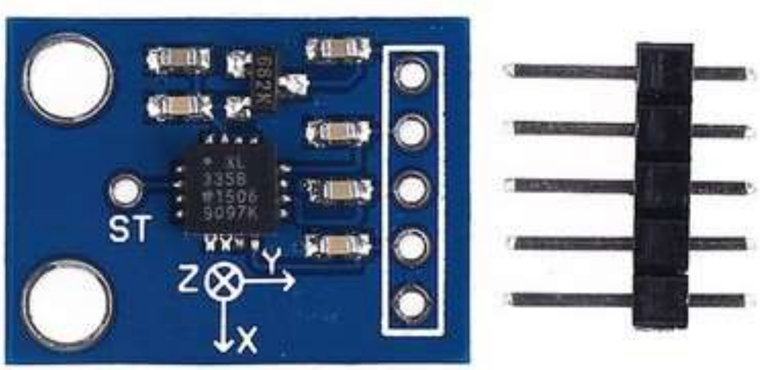


Figure 3.7.1: Accelerometer ADXL335

The ADXL335 gives complete 3-axis acceleration measurement.

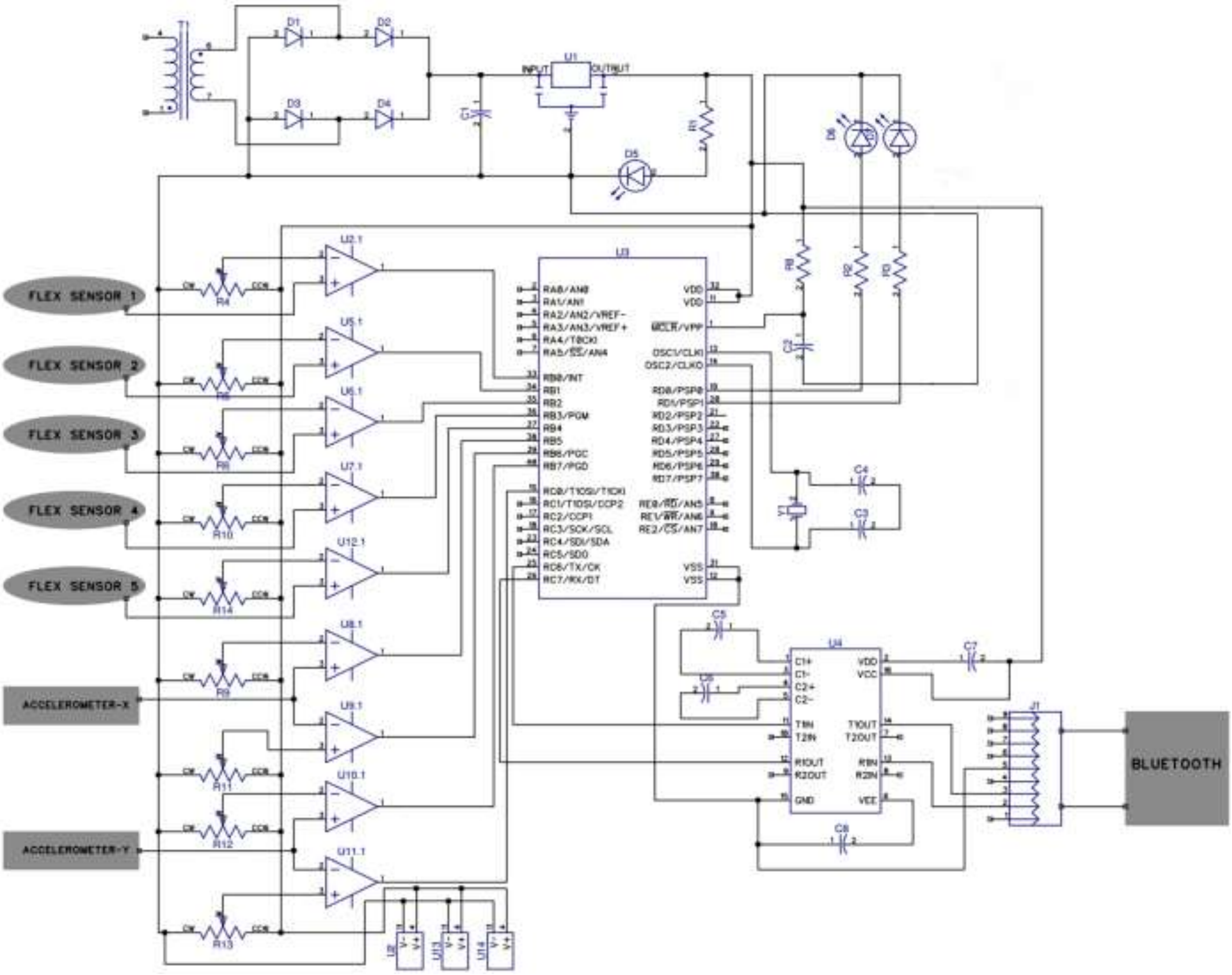
- This module measures acceleration within range ±3 g in the x, y and z axis.
- The output signals of this module are analog voltages that are proportional to the acceleration.
- It contains a poly silicon surface-micro machined sensor and signal conditioning circuitry.

Accelerometers have multiple applications in industry and science. Highly sensitive accelerometers are components of inertial navigation systems for aircraft and missiles. Accelerometers are used to detect and monitor vibration in rotating machinery. Accelerometers are used in tablet computers and digital cameras so that images on screens are always displayed upright. Accelerometers are used in drones for flight stabilization. Coordinated accelerometers can be used to measure differences in proper acceleration, particularly gravity, over their separation in space.

CHAPETER 4

CIRCUIT DIAGRAM

4.1 CIRCUIT DIAGRAM



4.2 CIRCUIT DESCRIPTION

The hardware section in the system consists of PIC16F877A microcontroller, Flex sensor, accelerometer, Bluetooth module, android device and power supply. . Each component in the circuit is operated by 12V and 5V DC supply which is derived from 230V AC supply by using a step-down transformer and a rectifier circuit. A bridge rectifier is used for this purpose. In order to regulate the rectifier 12V output to 5 V, we use a voltage regulator at the output of the bridge rectifier along with a number of capacitors in the output and input of voltage regulator. It provides better stabilization in voltage regulation and also removes ripples in the output voltage.

The regulated 5V is connected to microcontroller and VCC terminals of the other components used in the circuit. The ground of the power supply is connected to VSS of the controller and also respective GND terminal of the other elements. Flex sensors are connected to first 5 pins of PORT B. And accelerometer are connected to 15, 38, 39 and 40th pin of the microcontroller. The output is send to Android device through HC05. MAX232 is connect between PIC microcontroller and HC05 to convert TTL logic to RS232. Transmitter pin of the MAX232 is connect 25th pin. There are two LEDs are connected to 19th and 20th pins. They are Data LED and Program LEDs.

4.3 COMPONENTS

1. PIC16F877A MICROCONTROLLER
2. FLEX SENSOR
3. ACCELEROMETER – ADXL335
4. LM324
5. MAX 232
6. DB CONNECTOR
7. BLUETOOTH MODULE – HC05
8. 12V-0-12V TRANSFORMER
9. DIODES IN4007
10. IC 7805
11. RESISTORS 470Ω
12. VARIABLE RESISTORS 10KΩ
13. CAPACITOR 2200uF
14. CAPACITORS 10uF
15. CAPACITOR 100nF
16. CAPACITOR 33pF
17. CRYSTAL OSCILLATOR 20MHZ
18. LED

4.4 WORKING

Smart Glove is a system which which convert the sign languages to speech. Here each component in the circuit is operated by 12V and 5V DC supply which is derived from 230V AC supply by using a step-down transformer and a rectifier circuit. A bridge rectifier is used for this purpose. In order to regulate the rectifier 12V output to 5 V, we use a voltage regulator at the output of the bridge rectifier along with a number of capacitors in the output and input of voltage regulator. The 5V is obtained in the output in of voltage regulator, which is connected to PIC and VSS terminals other components. The first pin of the PIC is master clear pin. The master clear pin an optional external reset that is activated by pulling the pin low. Which is controlled by a configuration setting.

When power is ON, Microcontroller getting data from flex sensors and accelerometer. Tilting of the palm can be captured by the accelerometer where Flex sensors can measure the bend of the five fingers when making a sign. When the user performs a gesture/letter, signals are given to the LM324 comparator. LM324 compares the two voltages and the output is given to the microcontroller. The digital data is given to the microcontroller for further processing. The controller transmits a character corresponding to this gesture to an android device through HC05. The HC05 requires RS232 logic so MAX232 is used to convert TTL logic in to RS232. The respective voice for the character is speeches out in the android device.

CHAPTER 5

SOFTWARE SECTION

5.1 ANDROID

Android is an operating system based on Linux with a Java programming interface. It provides tools, e.g. a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance which is led by Google. Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. Dalvik uses special byte code. Therefore you cannot run standard Java byte code on Android. Android provides a tool "dx" which allows to convert Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) to simplify development Google provides the Android Development Tools (ADT) for Eclipse. The ADT performs automatically the conversion from class to dex files and creates the apk during deployment.

Figure 5.1.1: Android Logo

Important Android Components

An Android application consists out of the following parts:

- **Activity** - Represents the presentation layer of an Android application, e.g. a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application.
- **Views** - The User interface of Activities is built with widgets classes which inherent from "android.view.View". The layout of the views is managed by "android.view.ViewGroups".
- **Services** - perform background tasks without providing an UI. They can notify the user via the notification framework in Android.
- **Content Provider** - provides data to applications, via a content provider your application can share data with other applications. Android contains a SQLite DB which can serve as data provider
- **Intents** - are asynchronous messages which allow the application to request functionality from other services or activities. An application can call directly a service or activity (explicit intent)

or ask the Android system for registered services and applications for intent (implicit intents). For example the application could ask via intent for a contact application. Application registers them-selves to intent via an Intent Filter. Intents are a powerful concept as they allow to create a loosely coupled applications

- **Broadcast Receiver** - receives system messages and implicit intents, can be used to react to changed conditions in the system. An application can register as a broadcast receiver for certain events and can be started if such an event occurs.

Security and Permissions

Android defines certain permissions for certain tasks. For example if the application want to access the Internet it must define in its configuration file that it would like to use the related permission. During the installation of an Android application the user get a screen in which he needs to confirm the required permissions of the application.

Advantages of Android OS

- Android is the most widely used mobile OS in these days.
- Users have a vast choice to choose which apps they need. Apps are categorized into topics and every topic has large amount of apps to download.
- Also Support install third-party apps. These apps can be downloaded from different websites.
- There are a huge community of developers and users of android. So if you have any issue in the code or want to check authenticity of app then you can easily do this. You can check if the app is real by checking its reviews on Google play store.
- Android is adding new features in every release and removing old features that are not favorable to the users.
- Android is licensed under apache. Mobile companies change the code of android to make UI change a little bit. Developers have also access to the core code and can make changes to it.

5.2 MP LAB

MPLAB IDE is an integrated development environment that provides development engineers with the flexibility to develop and debug firmware for various Microchip devices**.** MPLAB IDE is a windows-based integrated development environment for the microchip technology incorporated PIC microcontroller and PIC digital signal controller families. In the MPLAB IDE we can:

- Create source code using the built-in editor.
- Assemble, compile and link source code using various language tools. An assembler, linker and librarian come with MPLAB IDE.
- Debug the executable logic by watching program flow with a simulator, such as MPLAB SIM, or in real time with an emulator, such as MPLAB IDE. Third party emulators that work with MPLAB IDE are also available.
- Make timing measurements.
- View variables in Watch windows.
- Program firmware into devices with programmers such as PICSTART Plus or PRO MATE II.
- Find quick answers to questions from the MPLAB IDE on-line.

Components of MPLAB IDE

The MPLAB IDE has both build-in components and plug-in modules to configure the system for variety of software and hardware tools.

- MPLAB IDE Built-in Components
- Additional Tools for MPLAB IDE
- Additional Optional Components for MPLAB IDE

MPLAB IDE Built-in Components

The build-in components consist of:

- **Project manager** : The project manager provides integration and communication between the IDE and the language tools
- **Editor**: The editor is a full-featured programmer’s text editor that also serves as a window into a debugger.
- **Assembler / Linker and Language Tools**: The assembler can be used stand-alone to assemble a single files, libraries and recompiled objects. The linker is responsible for positioning the compiled code into memory areas of the target microcontroller.

- **Debugger**: The microchip debugger allows breakpoints, single stepping, watch windows and all the features of a modern debugger for the MPLAB IDE. It works in conjunctions with the editor to reference information from the target being debugged back to the source code
- **Execution engines**: There are software simulators in MPLAB IDE for all PIC MCU and dsPIC DSC devices. These simulators use the PC to simulate the instructions and some peripheral functions of the PIC MCU devices. Optional in-circuit emulators and in-circuit debuggers are also available to test code and it runs in the applications hardware.

Getting start with MPLAB

The following texts are illustrate the procedures required to edit, assemble, simulate and program a PIC device with the MPLAB environment.

Step 1: Open MPLAB software and wait until it loads. Then you will see the workspace of the program.

Step 2: Now from Menu bar choose **Project >> Project Wizard**

Step 3: After the project Wizard will start.This will help you to create your project ,from the next menu you will have to choose the PIC ,you are going to use from the list,given there and after that you click on Next.

Step 4: Select the language tool

Step 5: Click finish to configure the project

Step 6: Go to file and click on new file

Figure 5.2.1: MPLAB IDE

5.3 DipTrace

DipTrace is EDA/CAD software for creating schematic diagrams and printed circuit boards. DipTrace has 4 modules: Schematic Capture Editor, PCB Layout Editor with built-in shape-based auto-router and 3D Preview & Export, Component Editor, and Pattern Editor.

DipTrace is an advanced PCB design software application that consists of 4 modules PCB Layout with efficient auto-router and auto-placer, schematic capture, component and pattern editors that allow you to design your own component libraries. DipTrace has a powerful automatic router, superior to many routers included in other PCB layout packages. It can route a single layer and multilayer circuit boards, and there is an option to auto route a single layer board with jumper wires, if required. DipTrace also provides you with external auto router support. Smart manual routing tools allow users to finalize the design and to get the results they want in a blink of an eye. There are number of verification features that allows you to control accuracy of your project. DipTrace modules allow you to exchange schematics, layouts and libraries with other EDA and CAD packages. Output formats are DXF, Gerber, Drill and G-code. Standard libraries contain more than 98,000 components.

Figure 5.3.1: DipTrace Software

Basic Features

- Multi-sheet and hierarchical schematics
- High-speed shape-based auto router
- Smart manual routing tools
- Differential pairs
- Wide import / export capabilities
- Advanced verifications with real-time DRC
- Real-time 3D PCB preview & STEP export

Schematic Capture

Advanced circuit design tool with support of multi-sheet and multi-level hierarchical schematics that delivers a number of features for visual and logical pin connections. Cross-module management ensures that principal circuits can be easily converted to PCB, back annotated, or imported/exported from/to other EDA, CAD formats and net-lists. DipTrace Schematic has ERC Verification and Spice export for external simulation

PCB Layout

Engineering tool for board design with smart manual routing, differential pairs, shape-based auto router, advanced verification, and wide import/export capabilities. Design requirements are defined by net classes, class-to-class rules, and detailed settings by object types for each class or layer. When routing with real-time DRC, the program reports errors on the fly before actually making them. DRC also checks length and phase tolerances for differential pairs. The board can be previewed in 3D and exported to STEP format for mechanical CAD modeling.

3D Preview and Export

This module includes real-time 3D preview & export feature. It shows the model of manufactured printed circuit board with all components installed. Rotate board in three axes, zoom in and out in real time, change colors of the board, copper areas, solder mask, silkscreen, and background. 3D preview works on all stages of the design. Board can be exported to STEP or VRML 2.0 formats for mechanical CAD modelling.

Component Editor

Manage component libraries and create single- or multi-part components by selecting a template and its dimensions, defining visual and electrical pin parameters, setting up a Spice model, and attaching pattern with a 3D model to finalize component creation. BSDL import, bulk pin naming, and pin manager tools for pins and buses. Importing libraries from different EDA formats. More than 130000 components in standard libraries.

Pattern Editor

Draw patterns with various types of shapes, pads, holes, and dimensions. Circle, Lines (headers, DIP), Square (QFP), Matrix (BGA), Rectangle (RQFP), and Zig-Zag standard templates. Creation of pattern is basically selecting a template, entering a couple of vital parameters, drawing the silkscreen, and launching automatic pad renumbering. Custom templates can be created for non-standard patterns. DXF import makes creating complex layouts easier.

5.4 EMBEDDED C

An embedded hardware device, depending on its size and capabilities, can have an operating system-such as embedded Linux- with limited or minimal functionality compared to a desktop version. For every small embedded devices, on OS might be entirely absent: it is not possible to write programs, compile, and run and debug the code in such small devices. In such a situation, it is necessary to use cross compilers (or assemblers), which compile programs written in a high-level language on a host system (typically a PC) and generate code.

For a target system (for example, an embedded device). If we write assembly programs and use an assembler running on a host to generate code for a target device, it is a cross assembler. So, we can write programs on our PC generate code for the embedded device and run it here. This solves the problem of creating executable code for embedded systems, but testing, debugging or tracing embedded programs are difficult.

Languages for programming embedded devices

C is the language of choice for most of the programming done for embedded systems. It might appear that the assembly language is intuitively the most obvious choice, since embedded programming is all about programming hardware devices such as microcontrollers. It is true that micro-controllers were initially programmed mostly in assembly language as with other embedded devices. It is not that difficult to write an assembly program since the assembly language produces the tightest code, making it possible to squeeze every possible byte of memory usage. However, the problem is that it becomes difficult to use for any reasonably-sized program, and even a slightly complicated device. The difficulties are in getting assembly programs to work correctly; and understanding, debugging, testing and, most importantly, maintaining them in the long run.

Also, high quality C compilers can often generate code that is comparable to the speed
of programs written in assembly. So, the benefits of using assembly for efficiency are negligible compared to the ease with which programmers can write C code. However, if performance is the key to make or break a device, then it is hard to beat assembly. For example, DSP (digital signal processing) devices are mostly programmed in assembly even today. Languages such as C++ have features that are often bulky, inefficient or inappropriate for use in resource constrained environments such as embedded devices. In particular, virtual functions and exception handling are two language features that are not efficient in terms of space and speed in embedded systems. Sometimes, C++ programming is used as 'Safe C", where only a small subset of C++ features is included. However, for convenience, most embedded projects pragmatically use C itself. Languages with 'managed runtime's, such as Java, are

mostly heavyweight. Running Java programs requires a Java Virtual Machine, which can take up a lot of resources.

Though Java is popular in high-end mobile phone because of the probability it provides and for browsing the Web, it is rarely suitable for use in small embedded devices. There are numerous special purposes or proprietary languages meant to be used in embedded systems such as B# and Dynamic C. Others, like Forth, are also well suited for the purpose. However, C is widely used and familiar to programmers worldwide, and its tools are easily available.

C in Embedded System

The following is a list of the most important differences between C programming for embedded systems and C programming for PCs.

Writing low-level code

In embedded programming, it is necessary to directly access the underlying hardware. For example, we might need to access a port, timer or a memory location. Similarly, we might need to do low-level programming activities like accessing the job queue, raise some signals, interrupts, etc. So, we need to write low level programs that directly access, modify or update hardware; this is an important characteristics of embedded C programs. C features such as pointers and bit-manipulation facilities enable us to program at the hardware level directly; so these features are used extensively in embedded programming.

Writing in-line assembly code

The C language provides a limited set of features, so it is not possible to use high-level C code to perform a specific function. For example, the device might have some instructions for which there is no direct equivalent in C code (for example, bit-wise rotation). In such cases, we can write assembly code embedded within C programs called ‘inline assembly’. The exact syntax depends on the Compiler. Using such techniques, it is possible to implement dynamic data structures such as linked lists and trees that still work under the limitations of embedded devices. Similarly, features such as recursion are not allowed in most of the embedded devices. One reason is that the recursion is inefficient in terms of space and time compared to iterative code. Fortunately, it is possible to write any recursive code as iterative code (through writing or understanding iterative versions of the code is not usually intuitive). To put it simply, costly language features (in terms of space and time) are either not available, or recommended in embedded programming. As programmers, we should find alternative ways of achieving the same functionality

.

Using limited C features or new language extensions

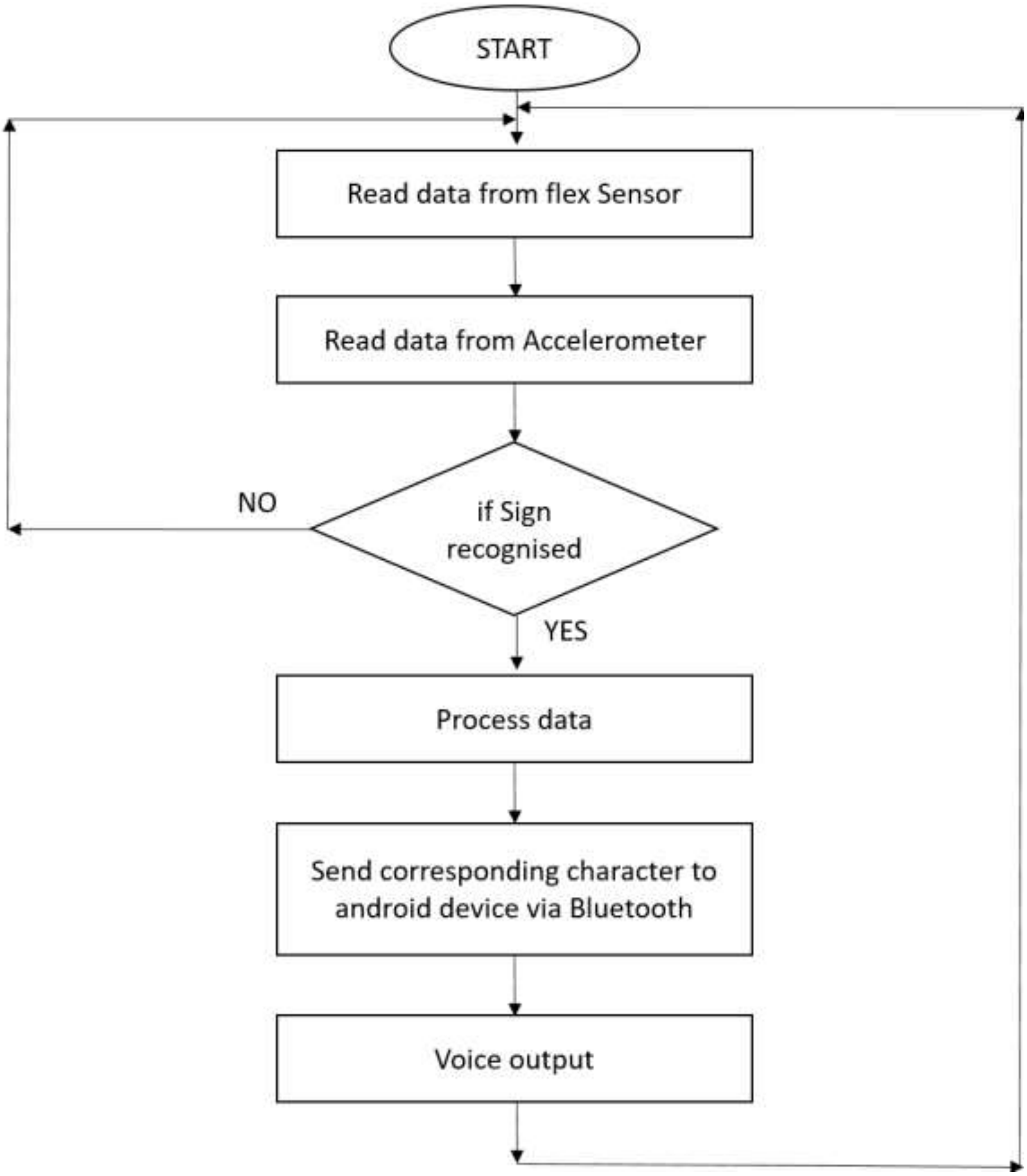
Some of the language features that we programming may need some extensions. For example, if the device does not support floating point numbers, then we cannot use floats or doubles in the programs. Problem occur the other way also: often, the underlying devices have hardware features that don't have direct support in the C programming language. For example, a device might support fixed-point numbers; however, C language doesn't have any support for this data type, so there is no direct way of programming

Embedded C compilers differ from regular C compilers in many ways; one important difference is the language features they support. Most of the embedded C compilers will not have full conformance to the ANSI C standard (because it is heavy-weight); rather, they will support the embedded c specification This specification is meant for use in embedded systems, and the most common extensions and features to enhance performance and convenience for accessing underlying hardware resources are provided in it. The advantage in using this specification is that a large number of compilers implement the specification, and thus it is easy to port embedded programs with it.

Characteristics of Embedded C

Like most imperative languages in the ALGOL tradition, C has facilities for structured programming and allows lexical variable scope and recursion, while a static type system prevents many unintended operations. In C, all executable code is contained within functions. Function parameters are always asked by value. Pass-by-reference is achieved in C by explicitly passing pointer values. Heterogeneous aggregate data types (struct) allow related data elements to be combined and manipulated as a unit. C program source text is free-format, using the semicolon as a statement terminator (not a delimiter).

5.5 FLOWCHART


START
Read data from flex Sensor
Read data from Accelerometer
if Sign recognised
NO
YES
Process data
Send corresponding character to android device via Bluetooth
Voice output

5.6 PROGRAM

```
#include<stdio.h>

#include<htc.h>

#include<pic.h>


#define _XTAL_FREQ   20000000                    //crystal frequecncy

#define BAUDRATE 9600                            //bps

#define stat_led RB7

#define data_led RB6

#define flex1 RB5

#define flex2 RB4

#define flex3 RB2

#define flex4 RB3

#define flex5 RB1

#define acc1 RC0

#define acc2 RC1

#define acc3 RC3

#define acc4 RC2


void delay(unsigned int k);

unsigned int count1,count2,count3;

unsigned char count4;

unsigned char count,rec_data;

unsigned int cnt;

bit flag1,flag2,onflag; //bit variable
```

```
//__CONFIG(0X1932);

__CONFIG(FOSC_HS&WDTE_OFF&PWRTE_ON&CP_OFF&BOREN_ON&LVP_OFF&CPD_OFF&DEBUG_OFF);                    //Used to burn in pic

void  delay(unsigned int k)

{

      for(int i=0;i<=k;i++)

      {

            for(int j=0;j<=1000;j++)

            {

            }

      }

}

void  InitTimer0(void)

{

        // Timer0 is 8bit timer, select T0CS and PSA to be zero

        OPTION_REG &= 0xC0;              // Make Prescalar 1:2

        T0IE = 1;                        // Enable Timer0 interrupt

        GIE = 1;                         // Enable global interrupts

}

void interrupt ISR(void)

{

        if(T0IF)  //If Timer0 Interrupt

        {

            count1++;

            if(count1>2000)

          {

               stat_led=~stat_led;

               count1=0;

          }

          T0IF = 0;   // Clear the interrupt
        }
}
void uart_init(void)
{
        TRISC6 = 1;                                   // TX Pin
        TRISC7 = 1;                                   // RX Pin
        SPBRG = ((_XTAL_FREQ/16)/BAUDRATE) - 1;
        BRGH  = 1;                                    // Fast baudrate
        SYNC  = 0;                                    // Asynchronous
        SPEN  = 1;                                   // Enable serial port pins
        CREN  = 1;                                   // Enable reception
        TXIE  = 0;                                   // Disable tx interrupts
        RCIE  = 1;                                   // Enable rx interrupts
        TX9   = 0;                                   // 8-bit transmission
        RX9   = 0;                                   // 8-bit reception
        TXEN  = 0;                                   // Reset transmitter
        TXEN  = 1;                                   // Enable the transmitter
}

 void main()
 {
        PORTA=0x00;
        TRISA=0x00;
        PORTB=0x00;
        TRISB=0x00;
        PORTC=0x00;
        TRISC=0x00;
        PORTD=0x00;
        TRISD=0x00;
        InitTimer0();

        uart_init();

        TRISC0=1;

        TRISC1=1;

        TRISC2=1;

        TRISC3=1;

        TRISB1=1;

        TRISB2=1;

        TRISB3=1;

        TRISB4=1;

        TRISB5=1;


        data_led=1;

        delay(100);

        data_led=0;

        onflag=1;


unsigned char rec_data=0;


while(1)

{
        if(acc1==1&&acc2==0&&acc3==0&&acc4==0&&flex1==0&&flex2==1&&flex3==1&&flex4==1&&flex5==1)

         {

                data_led=1;

                delay(100);

                TXREG='A';

                data_led=0;

                delay(100);

    }
```

```
        if(acc1==1&&acc2==0&&acc3==0&&acc4==0&&flex1==1&&flex2==0&&flex3==1&&flex4==1&&flex5==1)
         {
                data_led=1;
                delay(100);
                TXREG='B';
                data_led=0;
                delay(100);
    }
        if(acc1==1&&acc2==0&&acc3==0&&acc4==0&&flex1==1&&flex2==1&&flex3==0&&flex4==1&&flex5==1)
         {
                data_led=1;
                delay(100);
                TXREG='C';
                data_led=0;
                delay(100);
    }
        if(acc1==1&&acc2==0&&acc3==0&&acc4==0&&flex1==1&&flex2==1&&flex3==1&&flex4==0&&flex5==1)
         {
                data_led=1;
                delay(100);
                TXREG='D';
                data_led=0;
                delay(100);
     }
        if(acc1==1&&acc2==0&&acc3==0&&acc4==0&&flex1==1&&flex2==1&&flex3==1&&flex4==1&&flex5==0)
         {
                data_led=1;
                delay(100);

                TXREG='E';

                data_led=0;

                delay(100);

    }

        if(acc1==0&&acc2==1&&acc3==0&&acc4==0&&flex1==0&&flex2==1&&flex3==1&&flex
4==1&&flex5==1)

          {

                data_led=1;

                delay(100);

                TXREG='F';

                data_led=0;

                delay(100);

    }

        if(acc1==0&&acc2==1&&acc3==0&&acc4==0&&flex1==1&&flex2==0&&flex3==1&&flex
4==1&&flex5==1)

          {

                data_led=1;

                delay(100);

                TXREG='G';

                data_led=0;

                delay(100);

    }

        if(acc1==0&&acc2==1&&acc3==0&&acc4==0&&flex1==1&&flex2==1&&flex3==0&&flex
4==1&&flex5==1)

          {

                data_led=1;

                delay(100);

                TXREG='H';

                data_led=0;
```

```
                    delay(100);

    }

          if(acc1==0&&acc2==1&&acc3==0&&acc4==0&&flex1==1&&flex2==1&&flex3==1&&flex4==0&&flex5==1)

           {

                    data_led=1;

                    delay(100);

                    TXREG='I';

                    data_led=0;

                    delay(100);

     }

          if(acc1==0&&acc2==1&&acc3==0&&acc4==0&&flex1==1&&flex2==1&&flex3==1&&flex4==1&&flex5==0)

           {

                    data_led=1;

                    delay(100);

                    TXREG='J';

                    data_led=0;

                    delay(100);

    }

          if(acc1==0&&acc2==0&&acc3==1&&acc4==0&&flex1==0&&flex2==1&&flex3==1&&flex4==1&&flex5==1)

           {

                    data_led=1;

                    delay(100);

                    TXREG='K';

                    data_led=0;

                    delay(100);

    }

          if(acc1==0&&acc2==0&&acc3==1&&acc4==0&&flex1==1&&flex2==0&&flex3==1&&flex4==1&&flex5==1)
```

```
           {
                   data_led=1;
                   delay(100);
                   TXREG='L';
                   data_led=0;
                   delay(100);
    }
          if(acc1==0&&acc2==0&&acc3==1&&acc4==0&&flex1==1&&flex2==1&&flex3==0&&flex
4==1&&flex5==1)
           {
                   data_led=1;
                   delay(100);
                   TXREG='M';
                   data_led=0;
                   delay(100);
    }
          if(acc1==0&&acc2==0&&acc3==1&&acc4==0&&flex1==1&&flex2==1&&flex3==1&&flex
4==0&&flex5==1)
           {
                   data_led=1;
                   delay(100);
                   TXREG='N';
                   data_led=0;
                   delay(100);
     }
          if(acc1==0&&acc2==0&&acc3==1&&acc4==0&&flex1==1&&flex2==1&&flex3==1&&flex
4==1&&flex5==0)
           {
                   data_led=1;
                   delay(100);
```

```
                TXREG='O';

                data_led=0;

                delay(100);

   }

        if(acc1==0&&acc2==0&&acc3==0&&acc4==1&&flex1==0&&flex2==1&&flex3==1&&flex4==1&&flex5==1)

         {

                data_led=1;

                delay(100);

                TXREG='P';

                data_led=0;

                delay(100);

   }

        if(acc1==0&&acc2==0&&acc3==0&&acc4==1&&flex1==1&&flex2==0&&flex3==1&&flex4==1&&flex5==1)

         {

                data_led=1;

                delay(100);

                TXREG='Q';

                data_led=0;

                delay(100);

   }

        if(acc1==0&&acc2==0&&acc3==0&&acc4==1&&flex1==1&&flex2==1&&flex3==0&&flex4==1&&flex5==1)

         {

                data_led=1;

                delay(100);

                TXREG='R';

                data_led=0;

                delay(100);
```

```
    }
        if(acc1==0&&acc2==0&&acc3==0&&acc4==1&&flex1==1&&flex2==1&&flex3==1&&flex
4==0&&flex5==1)
          {
                data_led=1;
                delay(100);
                TXREG='S';
                data_led=0;
                delay(100);
     }
        if(acc1==0&&acc2==0&&acc3==0&&acc4==1&&flex1==1&&flex2==1&&flex3==1&&flex
4==1&&flex5==0)
          {
                data_led=1;
                delay(100);
                TXREG='T';
                data_led=0;
                delay(100);
    }
}
}
```

CHAPTER 6

PCB DESIGNING AND FABRICATION

6.1 BASIC STEPS IN PCB MANUFACTURING

Forming a printed circuit board is essential and the most prominent step in the formation of one electronic device. For a device to work properly the components we planned to use should be well placed in a PCB. In this section we are explaining about the formation of our PCB. The design of a PCB can be considered as the last step in electronic circuit designing.

In the electronic circuit performance and reliability depends on the productivity of PCB. Assembling and servicing ability also depends on the design. A proper PCB ensures that various components are interconnected as per the circuit diagram. Once they have been placed on the PCB in their proper positions and subsequently soldered PCB design and fabrication techniques have undergone so much of development that it has become a subject in itself. Double sided PCBs, multiplayer PCBs with plated through holes (PTH), flexible PCBs, etc. are only some of the developments. Manufacturing of PCB involves the following steps.

1 Print and etch
2 Print, plate and etch

The single sided PCBs are usually using the print and etch method and the double-sided plates through hole boards are made by print, plate and etch method. The production of multiplayer boards uses both the technique

Penalization

The schematic or the artwork of this circuit applied by the customer is transformed to working positive or negative films. The circuit is repeated conveniently to accommodate economically as many circuits as possible in a panel, which can be operated in every sequent steps in the PCBs process. This is called penalization.

Drilling

This is the state of the art operation. Very small holes are drilled with a high speed CMC drilling machine.

Plating

The heart of the PCB manufacturing process lies in the electrolytic plating process. The holes drilled are treated both mechanically and chemically before depositing the copper by the electrolytic copper plating process

Etching

Once a multiplayer board is drilled and electro less copper is deposited the image available in the form of a firm is transferred on to the outside by photo printing process. The boards are then with copper and tin. This is called etching.

Solder mask

Since PCB design may call for very close spacing conductors, a solder mask has to be applied on both sides of the circuit to avoid bridging of conductors. This ink is applied by screening. This is dried, exposed to UV, developed in a mild alkaline solution and finally treated by both UV and thermal energy.

Hot air leveling

After the above-mentioned process, the circuit pads are soldered using hot air leveling process while removing the board from solder path, hot air blown on both the sides of the board through air knives in the machine leaving the board soldered and leveled.

Procedure:

The first step in making electronic equipment is PCB manufacturing PCBs of some particular circuits are readily available in the market, yet it may not have of out desired size and shape. We can make the printed circuit board of our desire in our laboratory. Number of methods are available for making PCB. The simplest method is drawing the method on a copper clad board with enchant resistant ink or paint. This method is suitable where there is no need for precision and quantity required in only one or two pieces. But with the use of ICs and crowding of the components on the board. This method become unsuitable as one have a very steady hand in drawing the lines of required thickness with paint and brush. Another method is to make silk screen stencil by a photographic process, paint the pattern on a copper clad board, etch and drill the holes.

The fabrication of PCB includes the following steps:

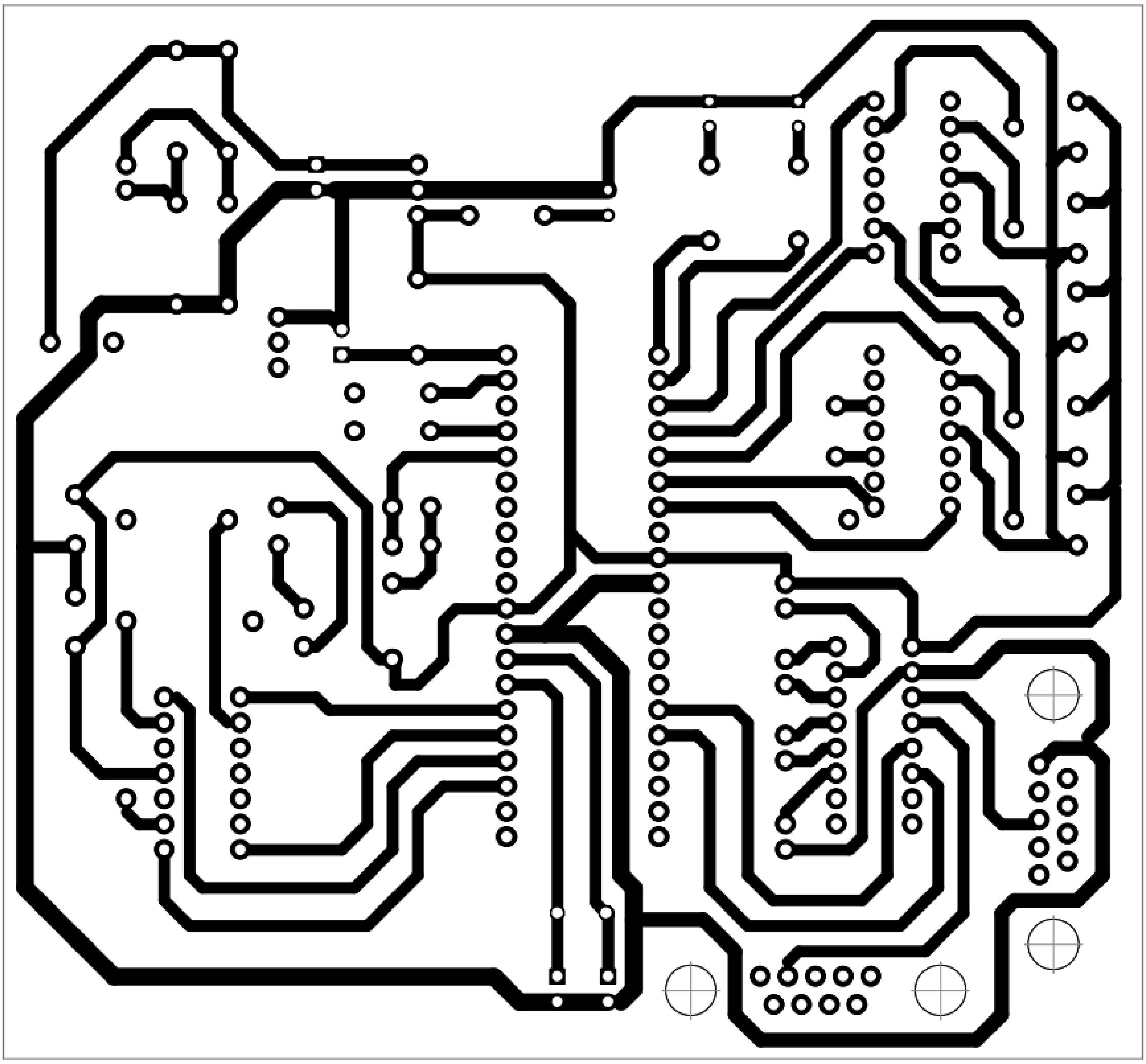
i. Preparing the PCB pattern.

ii. Transferring the pattern in the PCB.

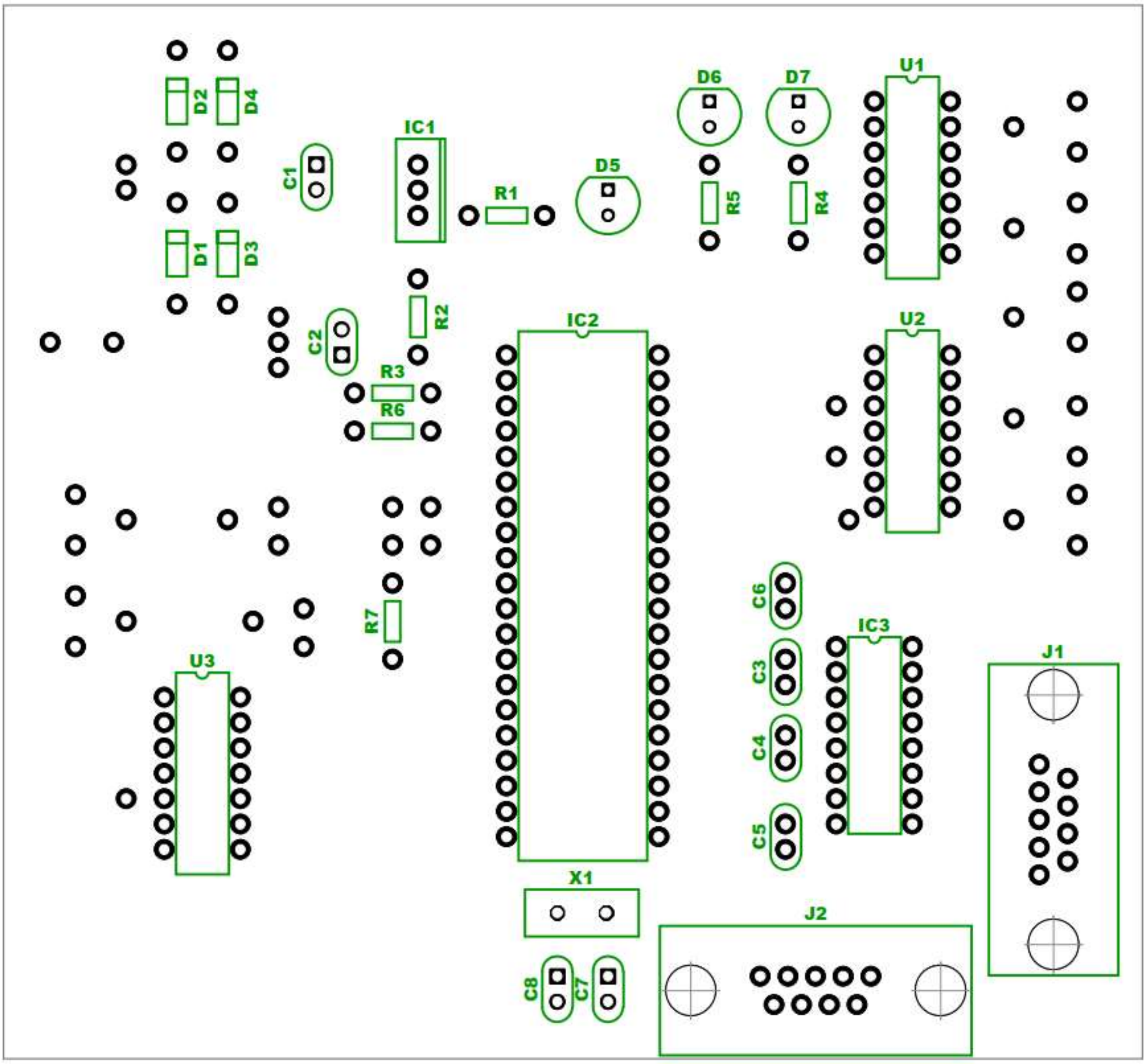
iii. Developing the PCB.

iv. Finishing touches

PCB LAYOUT



COMPONENT LAYOUT



ADVANTAGES

- It is anyone can operate easily
- Real time translation possible
- This project is useful for differently abled, speech impaired and paralyzed patients who cannot speak properly.
- This project can be applied in hospitals, those who are dumb and deaf can communicate with doctors.
- It will help in the military field for making the communication possible in between two soldiers who are not facing each other but closer.
- Gesture can be used to control interactions for entertainment purposes such as gaming to make the game player's experience more interactive.

DISADVANTAGES

- Range up to 10 meters only.
- This is not applicable for paralyzed people
- Facial expressions are not considered

APPLICATIONS

- This project is useful for differently abled, speech impaired and paralyzed patients who cannot speak properly.
- This project can be applied in hospitals, those who are dumb and deaf can communicate with doctors.
- It will help in the military field for making the communication possible in between two soldiers who are not facing each other but closer.
- It can be used in applications like remote handling, smooth traffic control

FUTURE SCOPE

- This project is useful for differently abled, speech impaired and paralyzed patients who cannot speak properly.
- For more reliable and low complexity of the circuit microcontroller can be replaced by other Advanced Microcontrollers.
- In this project many types of other applications can be added with using the different type of sensors like Heartbeat sensors for heartbeat monitoring and Temperature sensor for body temperature monitoring.
- This project is did not focus on facial expressions although it is well known that facial expressions convey important part of sign languages.
- In future it can be used in applications like remote handling, smooth traffic control and for rehabilitation of patients affected by trauma.
- Using the IOT technology we can connect this for live updates of the patient or person like locations, status body conditions etc.

CONCLUTION

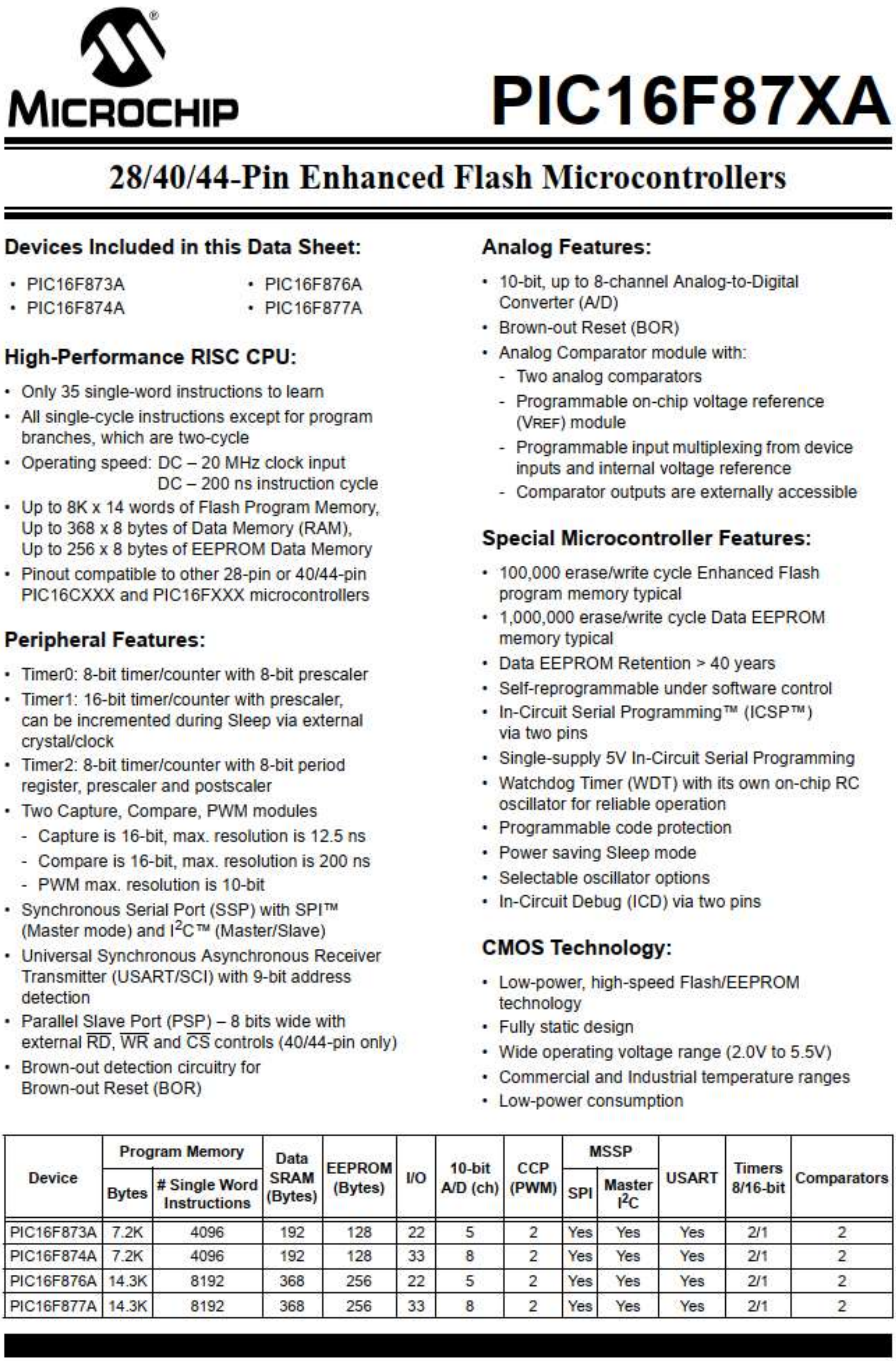
The talking glove: Low cost gesture recognition system was successfully designed and developed according to the requirements and specifications .The system used PIC microcontroller as the heart of the project. We introduced this project to lower the communication gap between the deaf and dumb community and the normal world. The mute people can use the glove to perform sign language and it will be converted into speech so that normal people can easily understand.

As we all know communication is the only medium by which we can share out thoughts or convey the message but for a person with disability faces difficulty in communication. This system will ease the communication problems of disabled persons. There is a lot of future scope for this project in many fields like military, hospitals etc. This project can also be developed or modified according to the rising needs and demands.

REFERENCES

1. The Microcontroller and Embedded system (Kenneth J Azelea, Dhanajay V Grade)
2. Solanki Krunal, “Microcontroller Based Sign Language Glove” International Journal for Scientific Research & Development (USRD), Vol. 1, Issue 4, 2013, pp 831-833.
3. https://en.m.wikipedia.org/wiki/Flex_sensor
4. https://en.m.wikipedia.org/wiki/Android_(operating_system)
5. http://www.alldatasheets.com
6. https://components101.com
7. INTERNET

DATASHEETS



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^{2}C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{\text{RD}}$, $\overline{\text{WR}}$ and $\overline{\text{CS}}$ 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

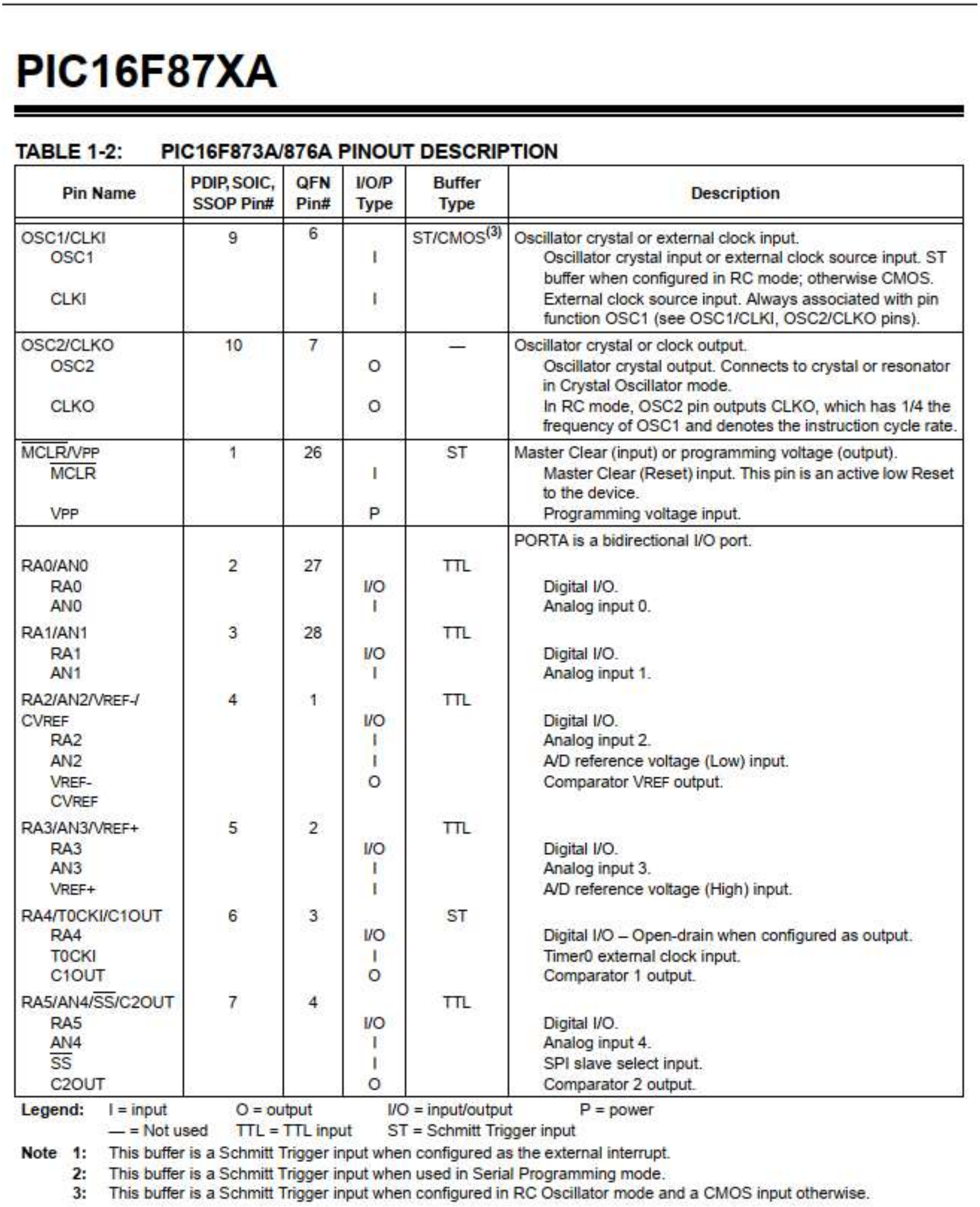
Special Microcontroller Features:

- 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^{2}C			
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



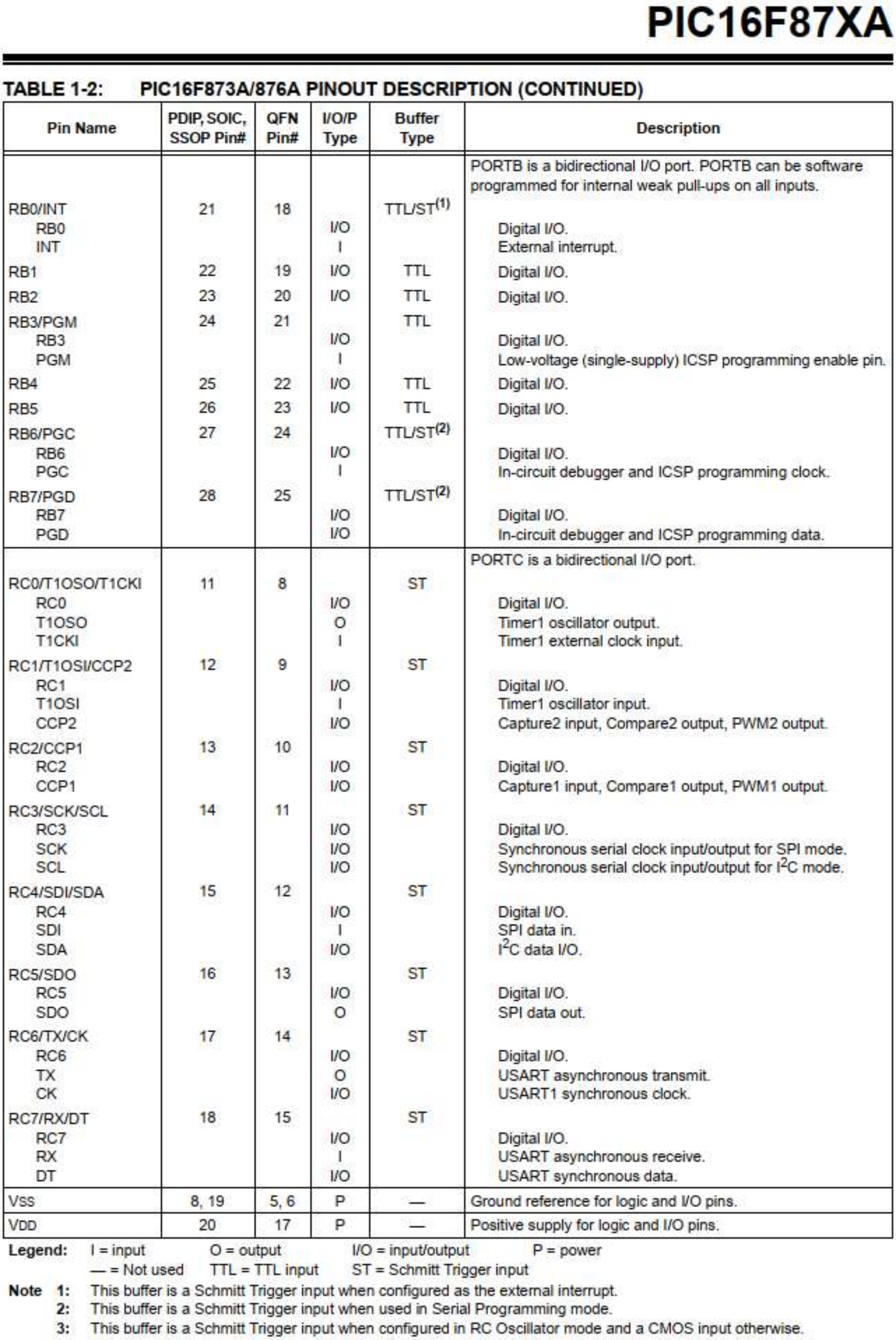
PIC16F87XA

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI	9	6		ST/CMOS$^{(3)}$	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	10	7		—	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.
$\overline{\text{MCLR}}$/VPP	1	26		ST	Master Clear (input) or programming voltage (output).
$\overline{\text{MCLR}}$			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	27		TTL	
RA0			I/O		Digital I/O.
AN0			I		Analog input 0.
RA1/AN1	3	28		TTL	
RA1			I/O		Digital I/O.
AN1			I		Analog input 1.
RA2/AN2/VREF-/ CVREF	4	1		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	2		TTL	
RA3			I/O		Digital I/O.
AN3			I		Analog input 3.
VREF+			I		A/D reference voltage (High) input.
RA4/T0CKI/C1OUT	6	3		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/$\overline{\text{SS}}$/C2OUT	7	4		TTL	
RA5			I/O		Digital I/O.
AN4			I		Analog input 4.
$\overline{\text{SS}}$			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.



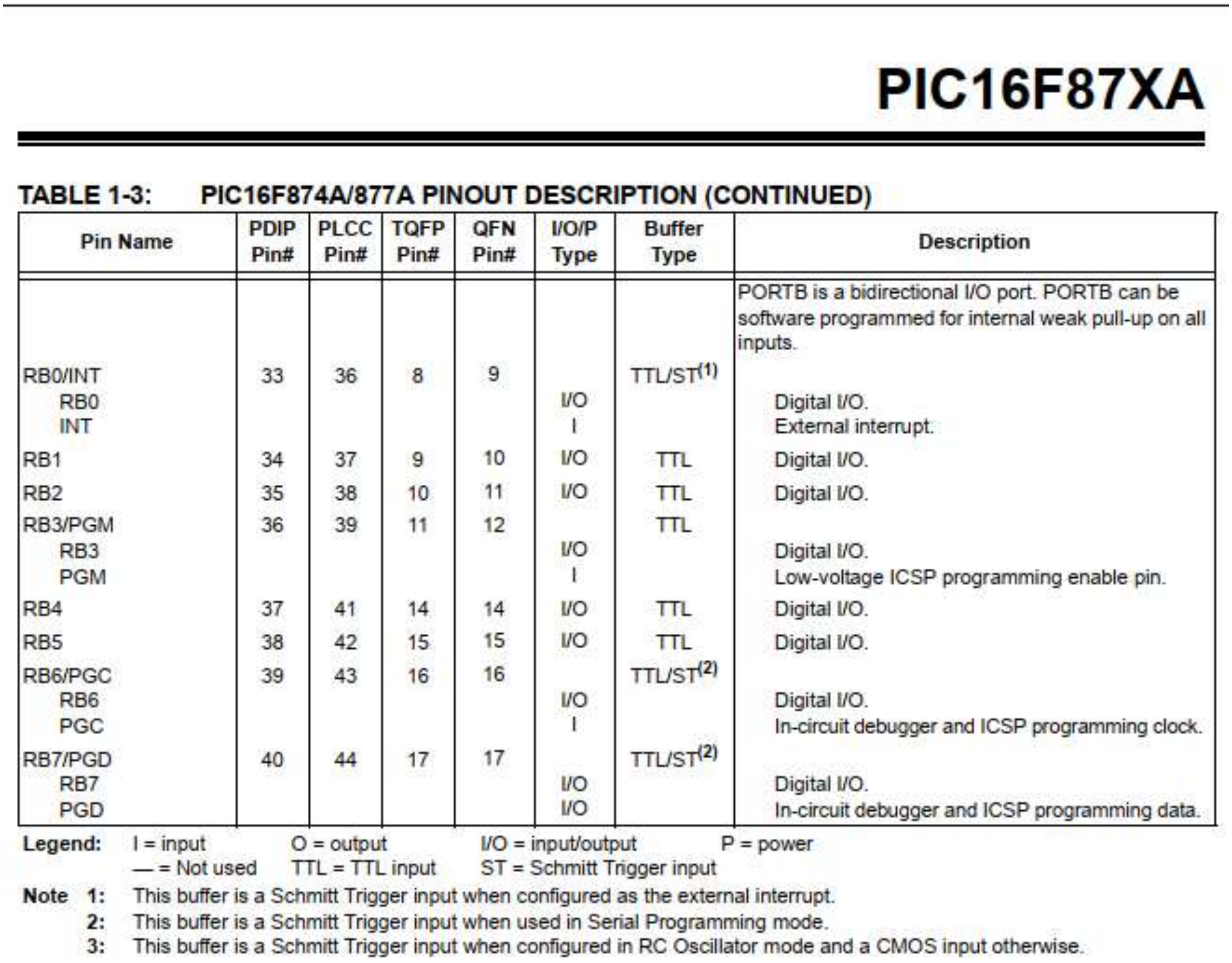
PIC16F87XA

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP, SOIC, SSOP 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-ups on all inputs.
RB0/INT	21	18		TTL/ST[1]	
RB0			I/O		Digital I/O.
INT			I		External interrupt.
RB1	22	19	I/O	TTL	Digital I/O.
RB2	23	20	I/O	TTL	Digital I/O.
RB3/PGM	24	21		TTL	
RB3			I/O		Digital I/O.
PGM			I		Low-voltage (single-supply) ICSP programming enable pin.
RB4	25	22	I/O	TTL	Digital I/O.
RB5	26	23	I/O	TTL	Digital I/O.
RB6/PGC	27	24		TTL/ST[2]	
RB6			I/O		Digital I/O.
PGC			I		In-circuit debugger and ICSP programming clock.
RB7/PGD	28	25		TTL/ST[2]	
RB7			I/O		Digital I/O.
PGD			I/O		In-circuit debugger and ICSP programming data.
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI	11	8		ST	
RC0			I/O		Digital I/O.
T1OSO			O		Timer1 oscillator output.
T1CKI			I		Timer1 external clock input.
RC1/T1OSI/CCP2	12	9		ST	
RC1			I/O		Digital I/O.
T1OSI			I		Timer1 oscillator input.
CCP2			I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	13	10		ST	
RC2			I/O		Digital I/O.
CCP1			I/O		Capture1 input, Compare1 output, PWM1 output.
RC3/SCK/SCL	14	11		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^{2}C mode.
RC4/SDI/SDA	15	12		ST	
RC4			I/O		Digital I/O.
SDI			I		SPI data in.
SDA			I/O		I^{2}C data I/O.
RC5/SDO	16	13		ST	
RC5			I/O		Digital I/O.
SDO			O		SPI data out.
RC6/TX/CK	17	14		ST	
RC6			I/O		Digital I/O.
TX			O		USART asynchronous transmit.
CK			I/O		USART1 synchronous clock.
RC7/RX/DT	18	15		ST	
RC7			I/O		Digital I/O.
RX			I		USART asynchronous receive.
DT			I/O		USART synchronous data.
VSS	8, 19	5, 6	P	—	Ground reference for logic and I/O pins.
VDD	20	17	P	—	Positive supply for logic and I/O pins.

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.



PIC16F87XA

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

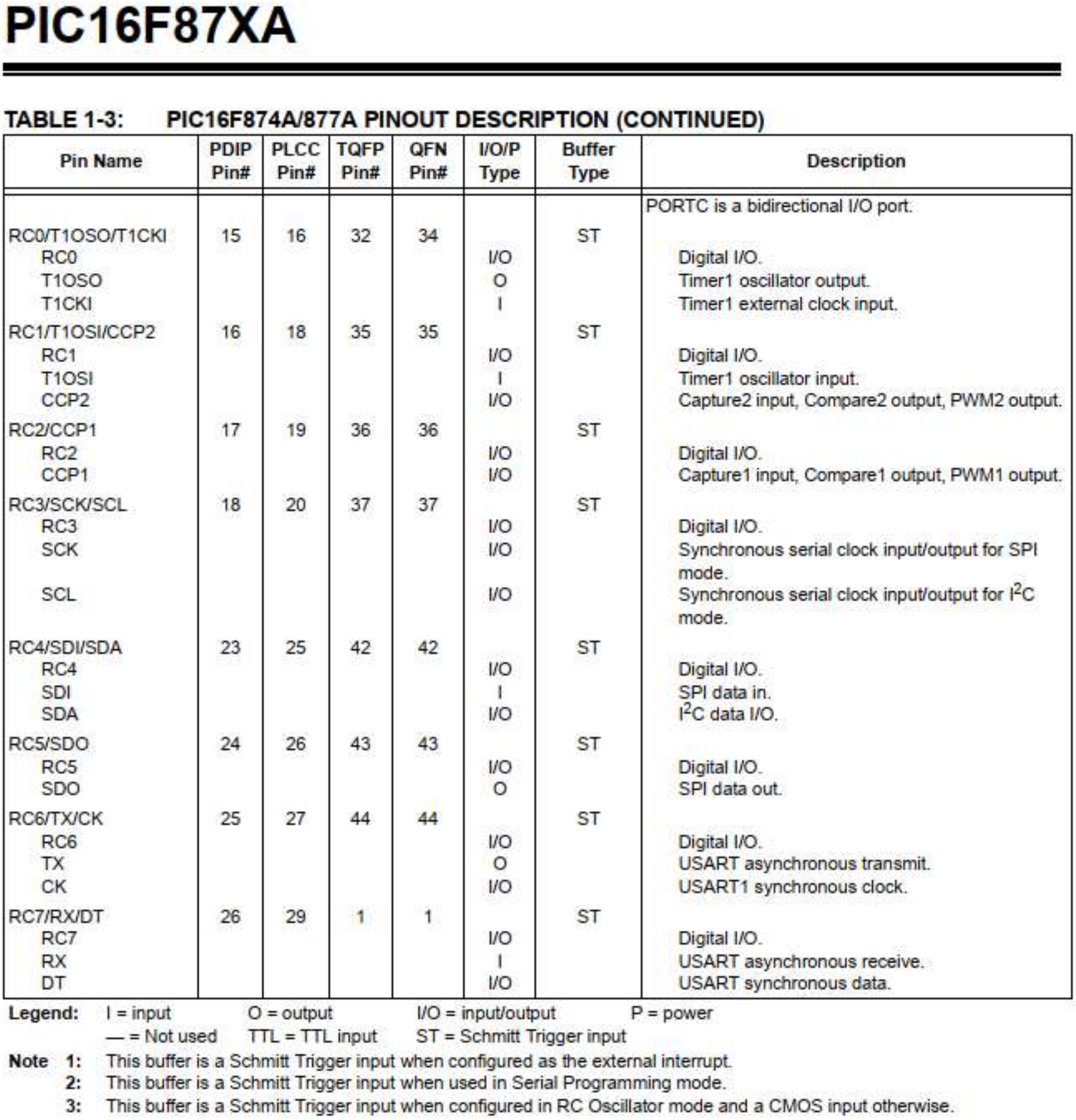
Pin Name	PDIP Pin#	PLCC Pin#	TQFP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
							PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	9		TTL/ST(1)	
RB0					I/O		Digital I/O.
INT					I		External interrupt.
RB1	34	37	9	10	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.



PIC16F87XA

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^{2}C mode.
RC4/SDI/SDA	23	25	42	42		ST	
RC4					I/O		Digital I/O.
SDI					I		SPI data in.
SDA					I/O		I^{2}C 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.

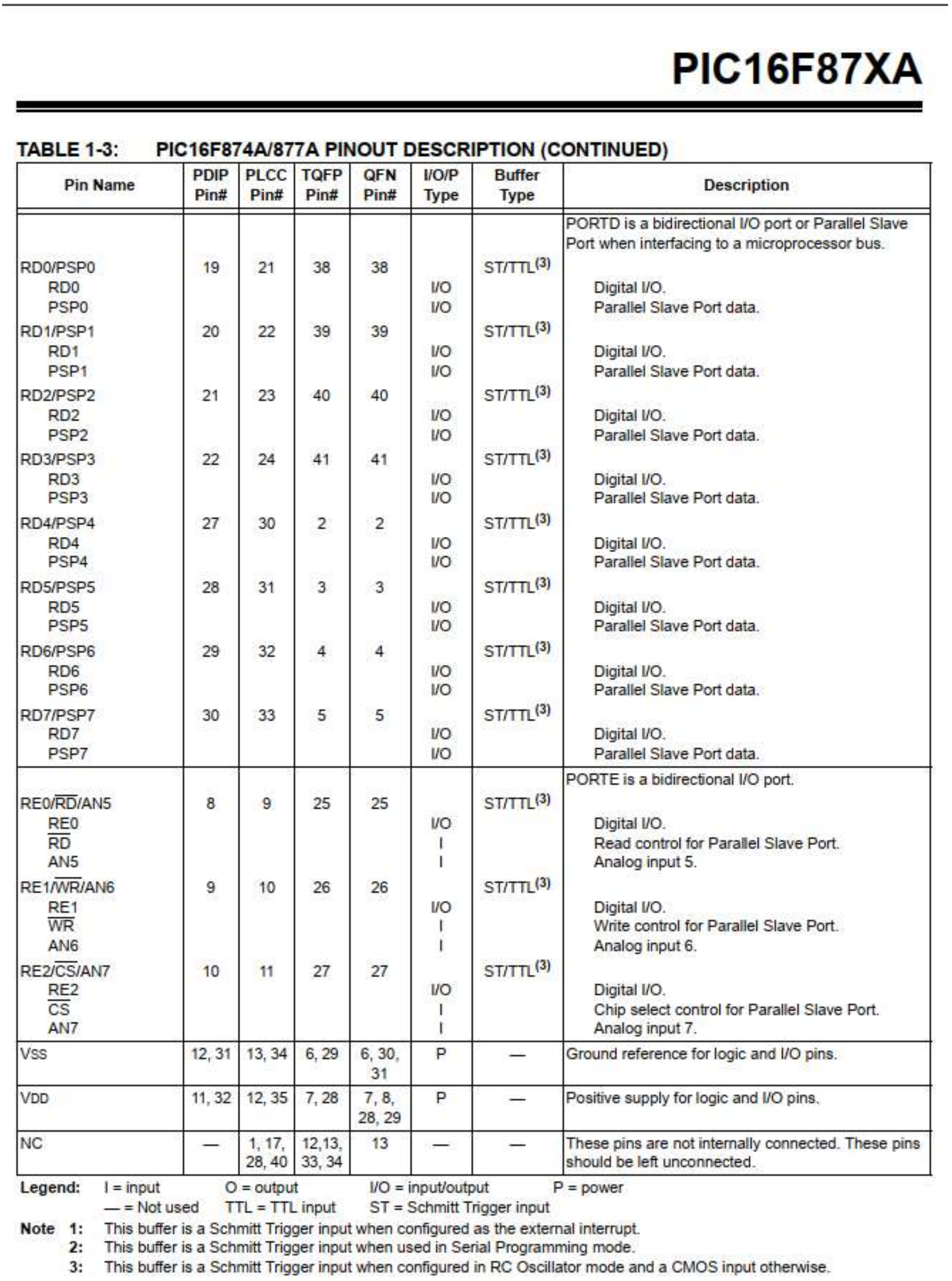


TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

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/$\overline{\text{RD}}$/AN5	8	9	25	25		ST/TTL(3)	
RE0					I/O		Digital I/O.
$\overline{\text{RD}}$					I		Read control for Parallel Slave Port.
AN5					I		Analog input 5.
RE1/$\overline{\text{WR}}$/AN6	9	10	26	26		ST/TTL(3)	
RE1					I/O		Digital I/O.
$\overline{\text{WR}}$					I		Write control for Parallel Slave Port.
AN6					I		Analog input 6.
RE2/$\overline{\text{CS}}$/AN7	10	11	27	27		ST/TTL(3)	
RE2					I/O		Digital I/O.
$\overline{\text{CS}}$					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.

PIC16F87XA

2.0 MEMORY ORGANIZATION

There are three memory blocks in each of the PIC16F87XA devices. The program memory and data memory have separate buses so that concurrent access can occur and is detailed in this section. The EEPROM data memory block is detailed in **Section 3.0 "Data EEPROM and Flash Program Memory"**.

Additional information on device memory may be found in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

2.1 Program Memory Organization

The PIC16F87XA devices have a 13-bit program counter capable of addressing an 8K word x 14 bit program memory space. The PIC16F876A/877A devices have 8K words x 14 bits of Flash program memory, while PIC16F873A/874A devices have 4K words x 14 bits. Accessing a location above the physically implemented address will cause a wraparound.

The Reset vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PIC16F876A/877A PROGRAM MEMORY MAP AND STACK

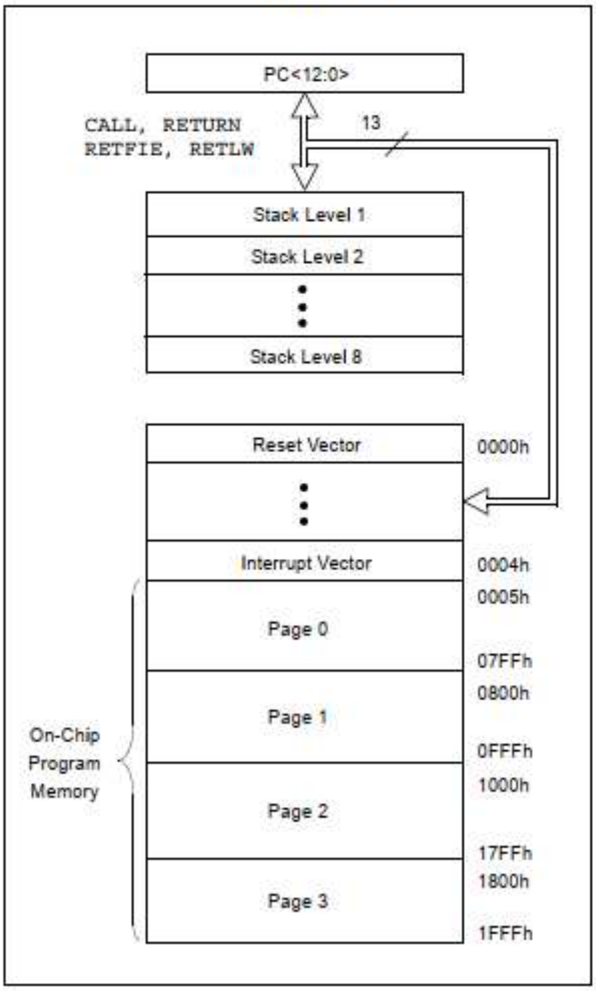
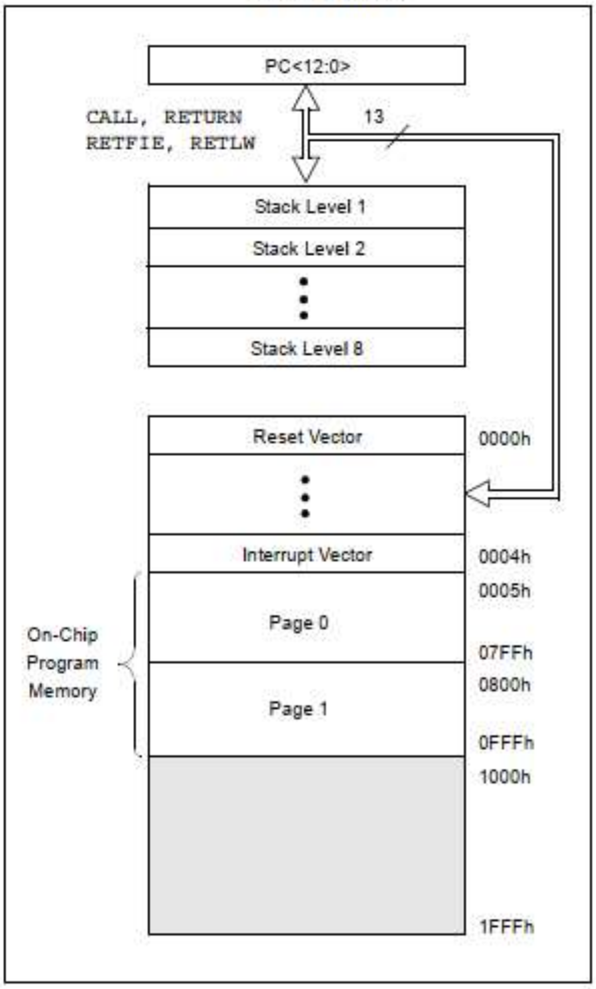
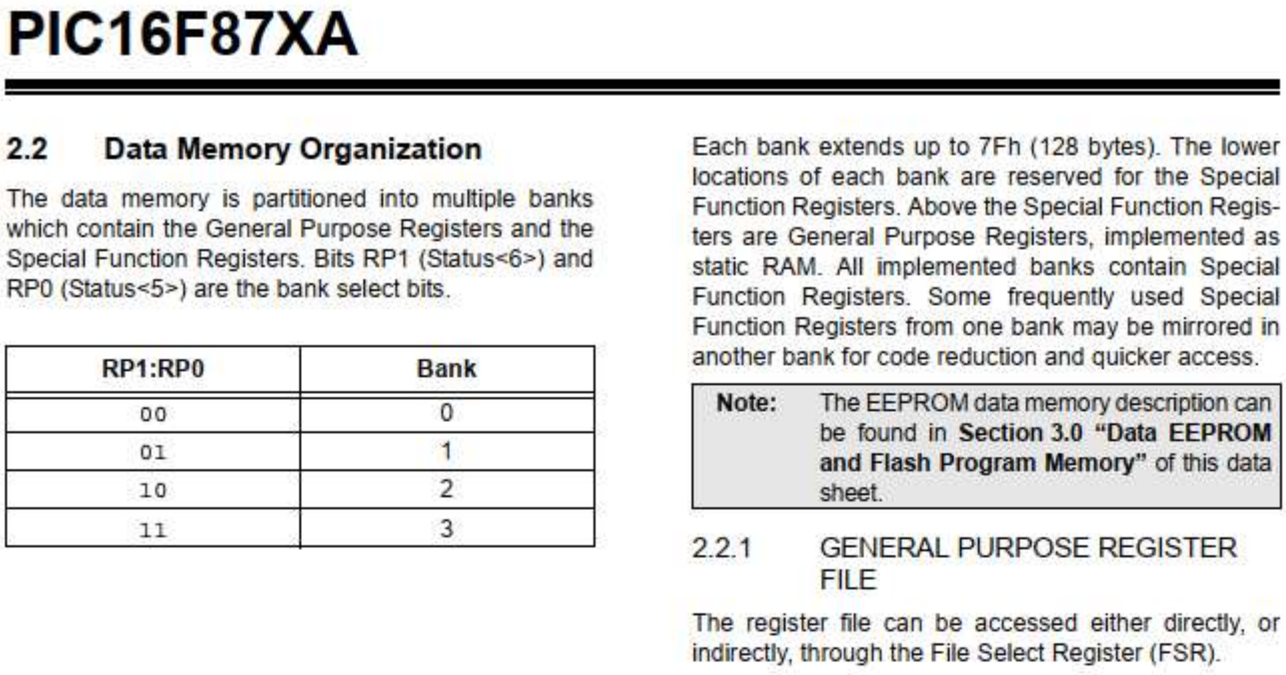


FIGURE 2-2: PIC16F873A/874A PROGRAM MEMORY MAP AND STACK





PIC16F87XA

2.2 Data Memory Organization

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

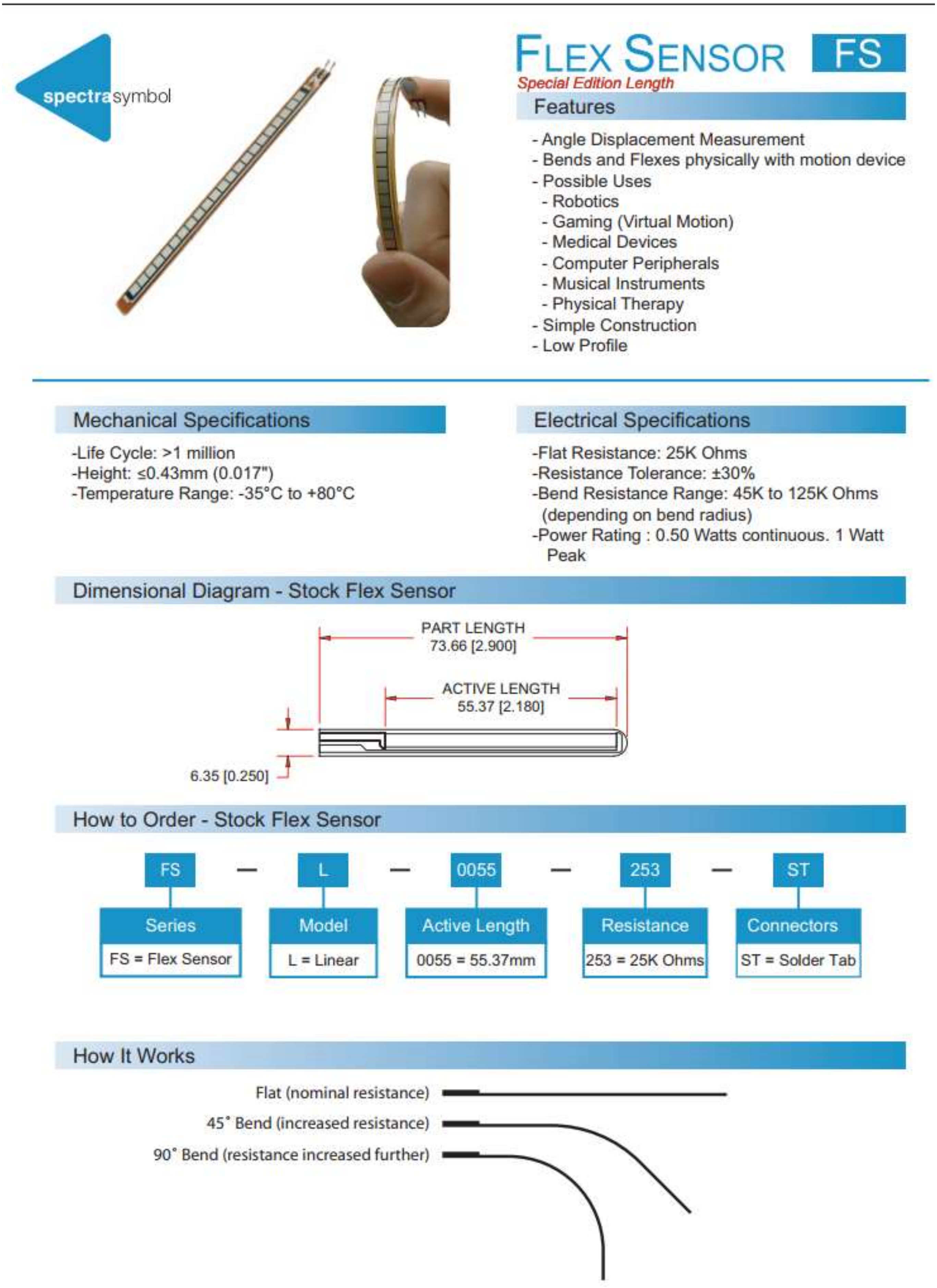
RP1:RP0	Bank
00	0
01	1
10	2
11	3

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

Note: The EEPROM data memory description can be found in **Section 3.0 "Data EEPROM and Flash Program Memory"** of this data sheet.

2.2.1 GENERAL PURPOSE REGISTER FILE

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



spectrasymbol

FLEX SENSOR FS

Special Edition Length

Features

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Possible Uses
  - Robotics
  - Gaming (Virtual Motion)
  - Medical Devices
  - Computer Peripherals
  - Musical Instruments
  - Physical Therapy
- Simple Construction
- Low Profile

Mechanical Specifications

-Life Cycle: >1 million
-Height: ≤0.43mm (0.017")
-Temperature Range: -35°C to +80°C

Electrical Specifications

-Flat Resistance: 25K Ohms
-Resistance Tolerance: ±30%
-Bend Resistance Range: 45K to 125K Ohms (depending on bend radius)
-Power Rating : 0.50 Watts continuous. 1 Watt Peak

Dimensional Diagram - Stock Flex Sensor

How to Order - Stock Flex Sensor

FS — L — 0055 — 253 — ST

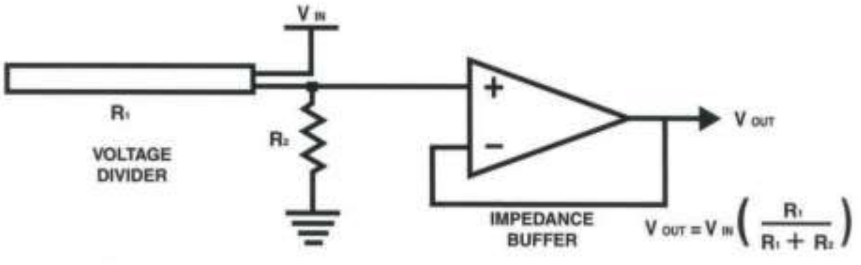
Series	Model	Active Length	Resistance	Connectors
FS = Flex Sensor	L = Linear	0055 = 55.37mm	253 = 25K Ohms	ST = Solder Tab

How It Works

Flat (nominal resistance)
45˚ Bend (increased resistance)
90˚ Bend (resistance increased further)

Schematics

BASIC FLEX SENSOR CIRCUIT:

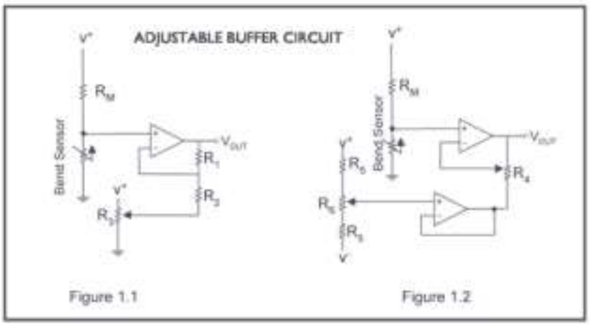


Following are notes from the ITP Flex Sensor Workshop

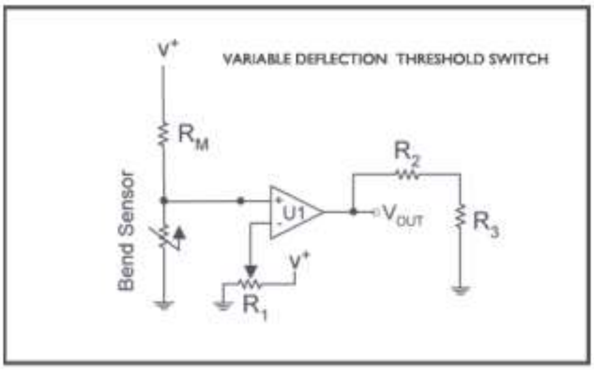
"The impedance buffer in the [Basic Flex Sensor Circuit] (above) is a single sided operational amplifier, used with these sensors because the low bias current of the op amp reduces errer due to source impedance of the flex sensor as voltage divider. Suggested op amps are the LM358 or LM324."

"You can also test your flex sensor using the simplest circut, and skip the op amp."

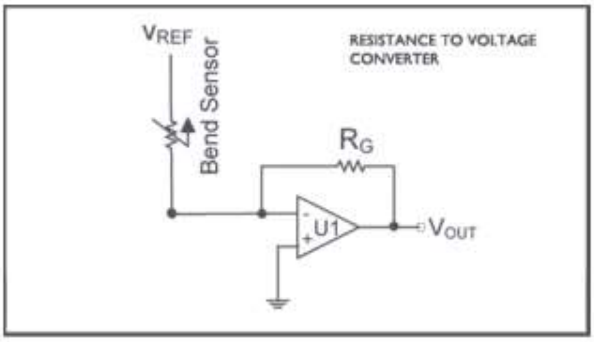
"Adjustable Buffer - a potentiometer can be added to the circuit to adjust the sensitivity range."

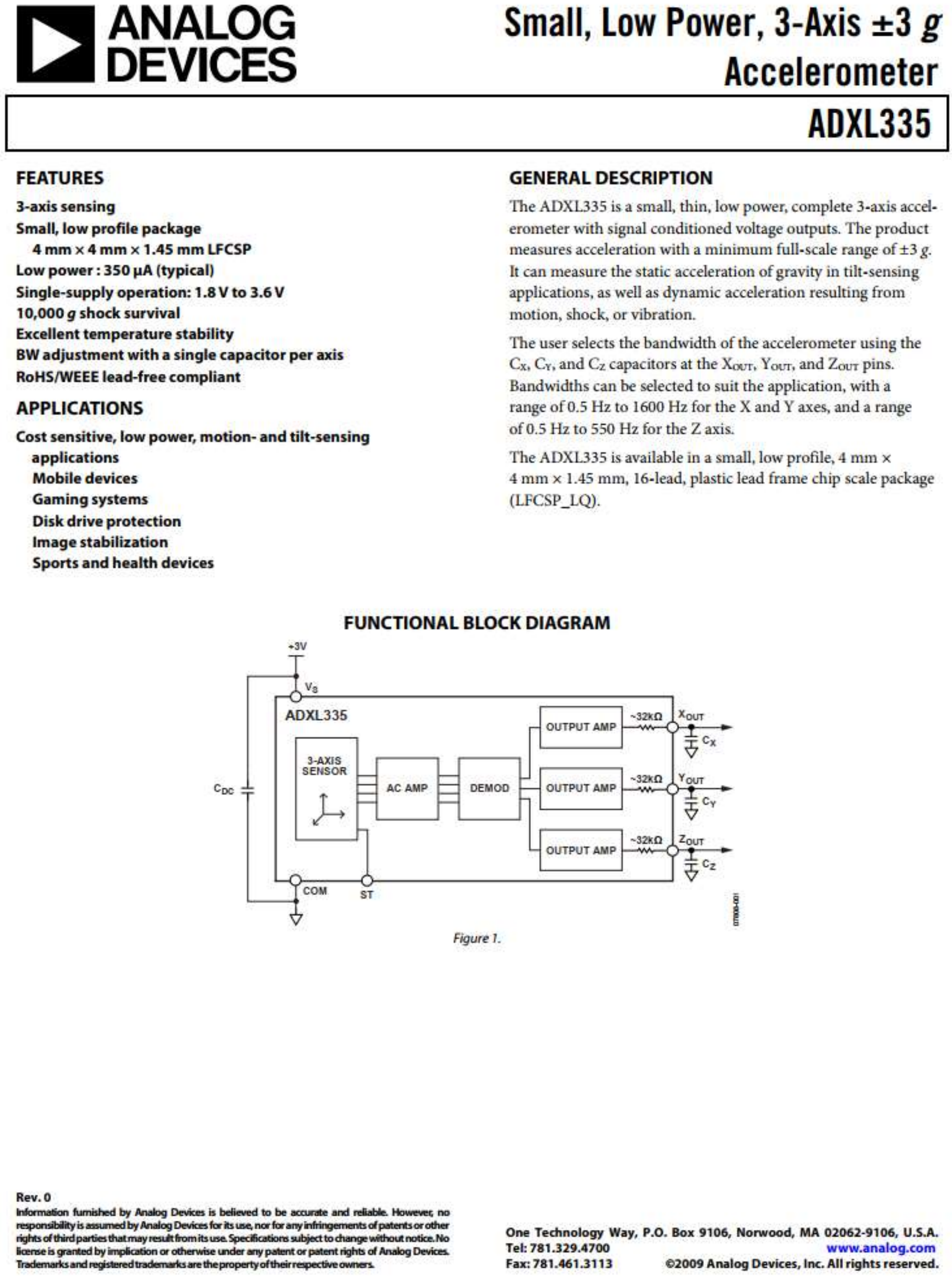


"Variable Deflection Threshold Switch - an op amp is used and outputs either high or low depending on the voltage of the inverting input. In this way you can use the flex sensor as a switch without going through a microcontroller."



"Resistance to Voltage Converter - use the sensor as the input of a resistance to voltage converter using a dual sided supply op-amp. A negative reference voltage will give a positive output. Should be used in situations when you want output at a low degree of bending."





ANALOG DEVICES

Small, Low Power, 3-Axis ±3 *g* Accelerometer

ADXL335

FEATURES

3-axis sensing
Small, low profile package
4 mm × 4 mm × 1.45 mm LFCSP
Low power : 350 μA (typical)
Single-supply operation: 1.8 V to 3.6 V
10,000 *g* shock survival
Excellent temperature stability
BW adjustment with a single capacitor per axis
RoHS/WEEE lead-free compliant

APPLICATIONS

Cost sensitive, low power, motion- and tilt-sensing applications
Mobile devices
Gaming systems
Disk drive protection
Image stabilization
Sports and health devices

GENERAL DESCRIPTION

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3 *g*. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the C_X, C_Y, and C_Z capacitors at the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL335 is available in a small, low profile, 4 mm × 4 mm × 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSP_LQ).

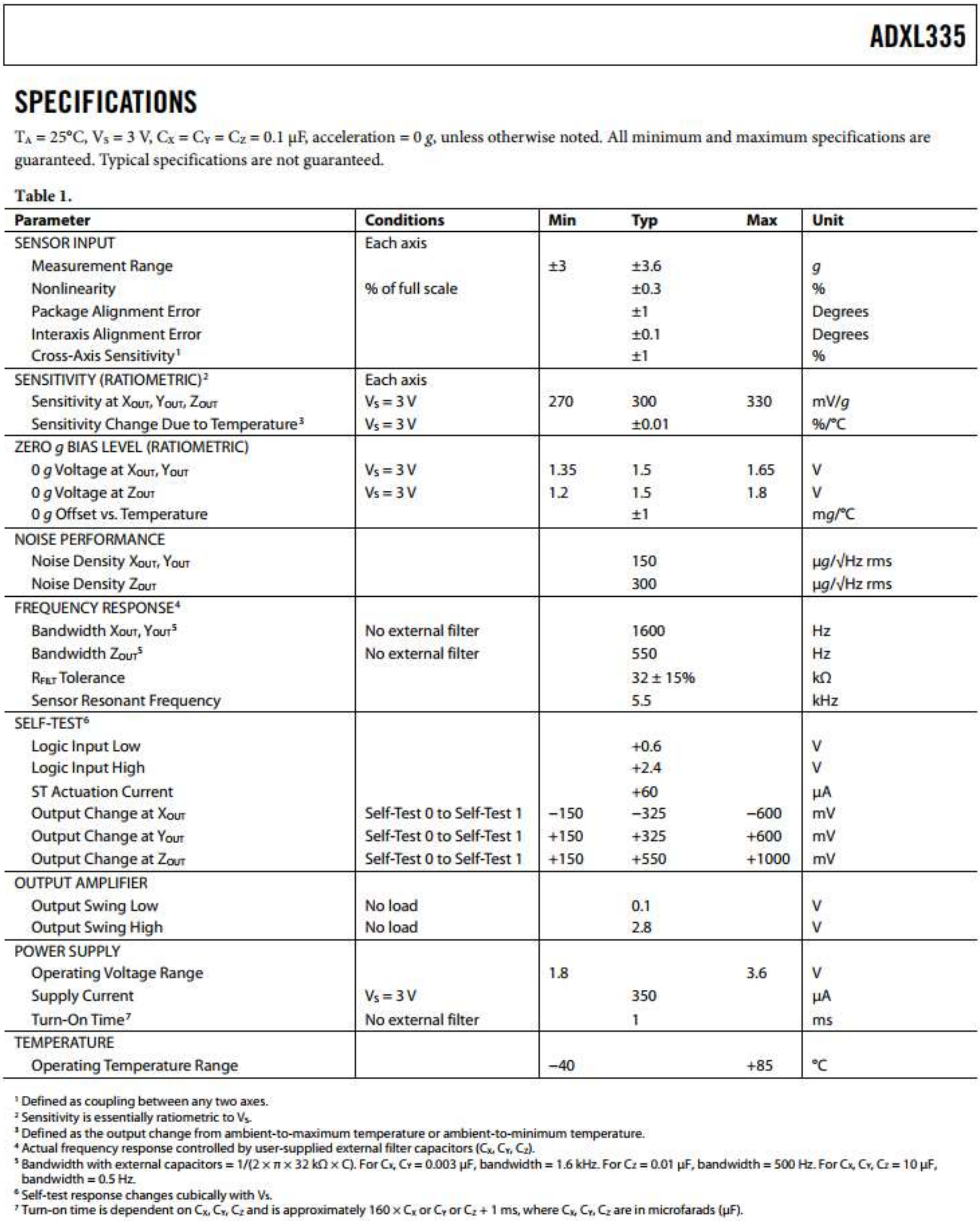
FUNCTIONAL BLOCK DIAGRAM

Figure 1.

Rev. 0
Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Specifications subject to change without notice. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices. Trademarks and registered trademarks are the property of their respective owners.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

©2009 Analog Devices, Inc. All rights reserved.



ADXL335

SPECIFICATIONS

$T_A = 25°C$, $V_S = 3$ V, $C_X = C_Y = C_Z = 0.1$ µF, acceleration = 0 *g*, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[1]			±1		%
SENSITIVITY (RATIOMETRIC)[2]	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	$V_S = 3$ V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature[3]	$V_S = 3$ V		±0.01		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)					
0 *g* Voltage at X_{OUT}, Y_{OUT}	$V_S = 3$ V	1.35	1.5	1.65	V
0 *g* Voltage at Z_{OUT}	$V_S = 3$ V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		m*g*/°C
NOISE PERFORMANCE					
Noise Density X_{OUT}, Y_{OUT}			150		µ*g*/√Hz rms
Noise Density Z_{OUT}			300		µ*g*/√Hz rms
FREQUENCY RESPONSE[4]					
Bandwidth X_{OUT}, Y_{OUT}[5]	No external filter		1600		Hz
Bandwidth Z_{OUT}[5]	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF-TEST[6]					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X_{OUT}	Self-Test 0 to Self-Test 1	−150	−325	−600	mV
Output Change at Y_{OUT}	Self-Test 0 to Self-Test 1	+150	+325	+600	mV
Output Change at Z_{OUT}	Self-Test 0 to Self-Test 1	+150	+550	+1000	mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	$V_S = 3$ V		350		µA
Turn-On Time[7]	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C

[1] Defined as coupling between any two axes.

[2] Sensitivity is essentially ratiometric to V_S.

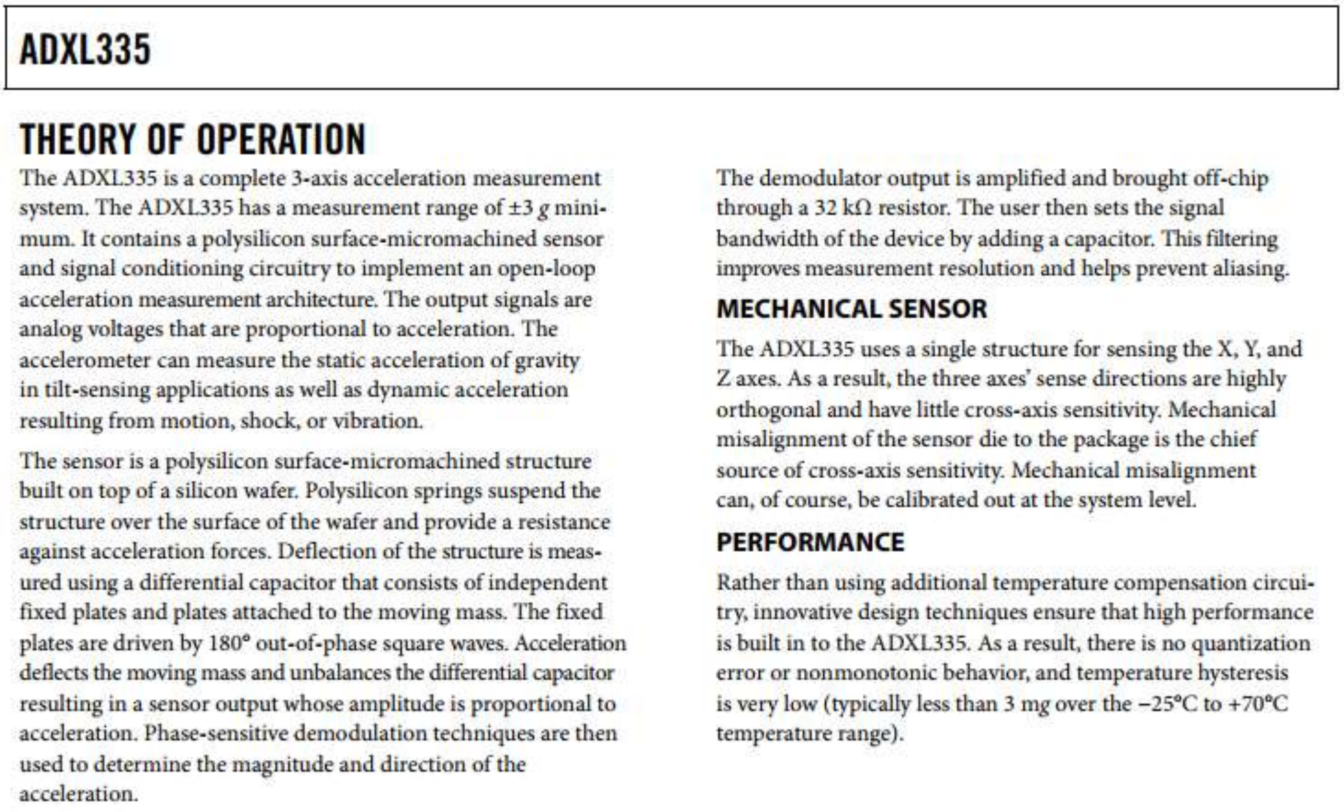
[3] Defined as the output change from ambient-to-maximum temperature or ambient-to-minimum temperature.

[4] Actual frequency response controlled by user-supplied external filter capacitors (C_X, C_Y, C_Z).

[5] Bandwidth with external capacitors = 1/(2 × π × 32 kΩ × C). For C_X, C_Y = 0.003 µF, bandwidth = 1.6 kHz. For C_Z = 0.01 µF, bandwidth = 500 Hz. For C_X, C_Y, C_Z = 10 µF, bandwidth = 0.5 Hz.

[6] Self-test response changes cubically with V_S.

[7] Turn-on time is dependent on C_X, C_Y, C_Z and is approximately 160 × C_X or C_Y or C_Z + 1 ms, where C_X, C_Y, C_Z are in microfarads (µF).



ADXL335

THEORY OF OPERATION

The ADXL335 is a complete 3-axis acceleration measurement system. The ADXL335 has a measurement range of ±3 *g* minimum. It contains a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement an open-loop acceleration measurement architecture. The output signals are analog voltages that are proportional to acceleration. The accelerometer can measure the static acceleration of gravity in tilt-sensing applications as well as dynamic acceleration resulting from motion, shock, or vibration.

The sensor is a polysilicon surface-micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and plates attached to the moving mass. The fixed plates are driven by 180° out-of-phase square waves. Acceleration deflects the moving mass and unbalances the differential capacitor resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation techniques are then used to determine the magnitude and direction of the acceleration.

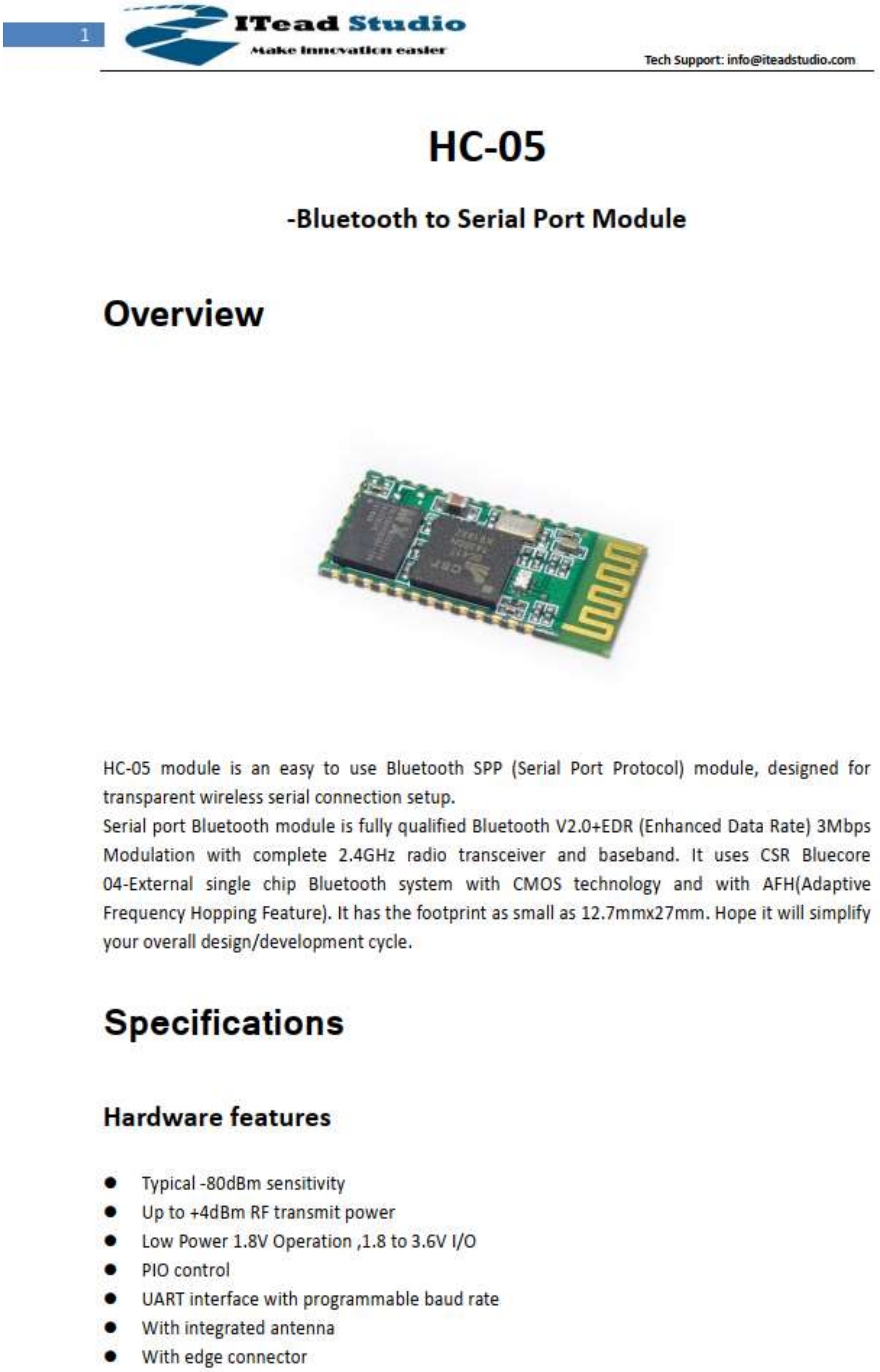
The demodulator output is amplified and brought off-chip through a 32 kΩ resistor. The user then sets the signal bandwidth of the device by adding a capacitor. This filtering improves measurement resolution and helps prevent aliasing.

MECHANICAL SENSOR

The ADXL335 uses a single structure for sensing the X, Y, and Z axes. As a result, the three axes' sense directions are highly orthogonal and have little cross-axis sensitivity. Mechanical misalignment of the sensor die to the package is the chief source of cross-axis sensitivity. Mechanical misalignment can, of course, be calibrated out at the system level.

PERFORMANCE

Rather than using additional temperature compensation circuitry, innovative design techniques ensure that high performance is built in to the ADXL335. As a result, there is no quantization error or nonmonotonic behavior, and temperature hysteresis is very low (typically less than 3 m*g* over the −25°C to +70°C temperature range).



HC-05

-Bluetooth to Serial Port Module

Overview

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.

Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

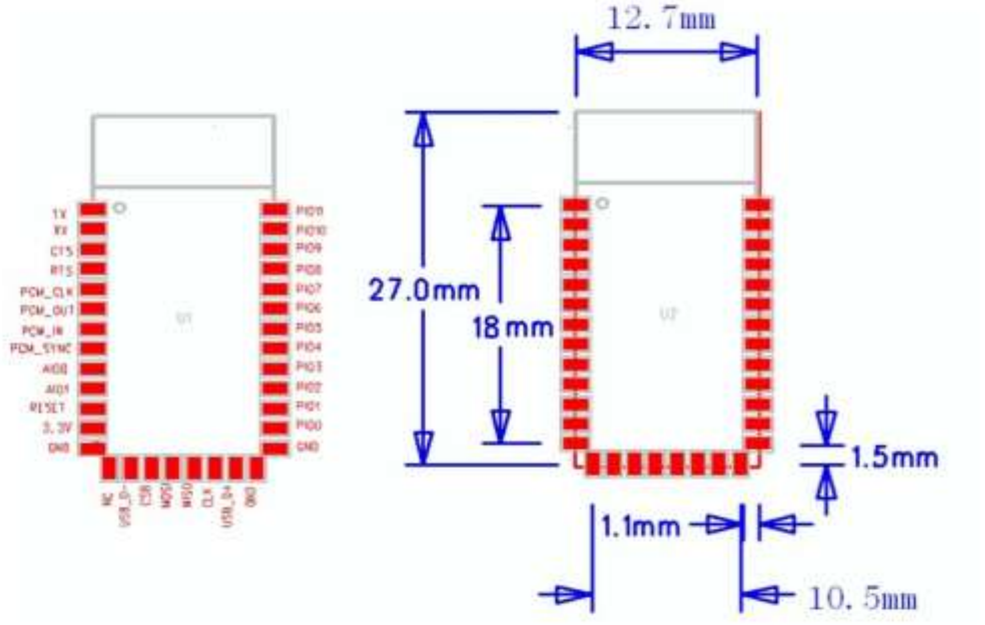
Hardware features

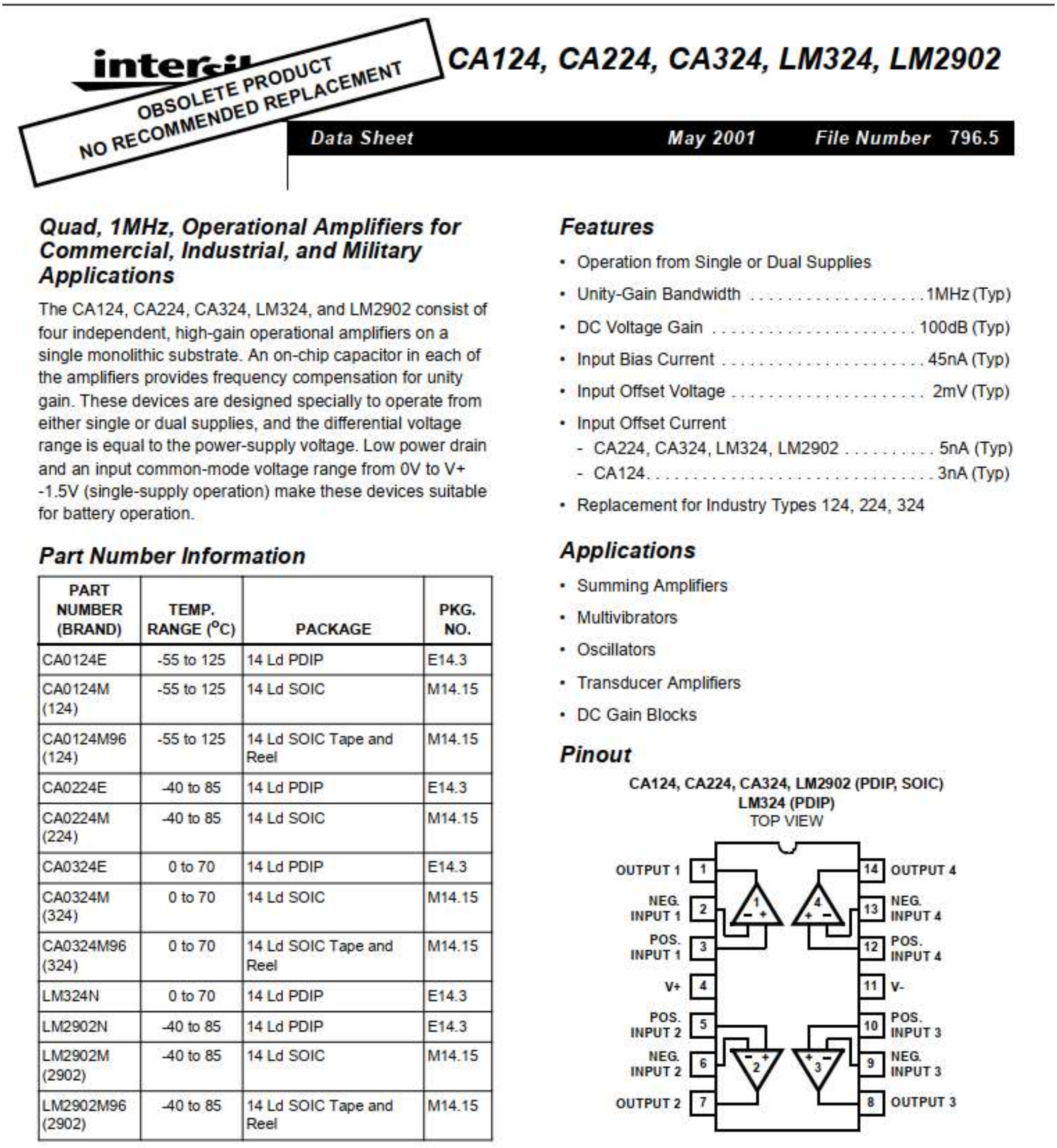
- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1,Parity:No parity, Data control: has. Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Hardware





intersil

OBSOLETE PRODUCT
NO RECOMMENDED REPLACEMENT

CA124, CA224, CA324, LM324, LM2902

Data Sheet May 2001 File Number 796.5

Quad, 1MHz, Operational Amplifiers for Commercial, Industrial, and Military Applications

The CA124, CA224, CA324, LM324, and LM2902 consist of four independent, high-gain operational amplifiers on a single monolithic substrate. An on-chip capacitor in each of the amplifiers provides frequency compensation for unity gain. These devices are designed specially to operate from either single or dual supplies, and the differential voltage range is equal to the power-supply voltage. Low power drain and an input common-mode voltage range from 0V to V+ -1.5V (single-supply operation) make these devices suitable for battery operation.

Part Number Information

PART NUMBER (BRAND)	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
CA0124E	-55 to 125	14 Ld PDIP	E14.3
CA0124M (124)	-55 to 125	14 Ld SOIC	M14.15
CA0124M96 (124)	-55 to 125	14 Ld SOIC Tape and Reel	M14.15
CA0224E	-40 to 85	14 Ld PDIP	E14.3
CA0224M (224)	-40 to 85	14 Ld SOIC	M14.15
CA0324E	0 to 70	14 Ld PDIP	E14.3
CA0324M (324)	0 to 70	14 Ld SOIC	M14.15
CA0324M96 (324)	0 to 70	14 Ld SOIC Tape and Reel	M14.15
LM324N	0 to 70	14 Ld PDIP	E14.3
LM2902N	-40 to 85	14 Ld PDIP	E14.3
LM2902M (2902)	-40 to 85	14 Ld SOIC	M14.15
LM2902M96 (2902)	-40 to 85	14 Ld SOIC Tape and Reel	M14.15

Features

- Operation from Single or Dual Supplies
- Unity-Gain Bandwidth . . . . 1MHz (Typ)
- DC Voltage Gain . . . . 100dB (Typ)
- Input Bias Current . . . . 45nA (Typ)
- Input Offset Voltage . . . . 2mV (Typ)
- Input Offset Current
  - CA224, CA324, LM324, LM2902 . . . . 5nA (Typ)
  - CA124. . . . . 3nA (Typ)
- Replacement for Industry Types 124, 224, 324

Applications

- Summing Amplifiers
- Multivibrators
- Oscillators
- Transducer Amplifiers
- DC Gain Blocks

Pinout

CA124, CA224, CA324, LM2902 (PDIP, SOIC)
LM324 (PDIP)
TOP VIEW

CA124, CA224, CA324, LM324, LM2902

Typical Performance Curves

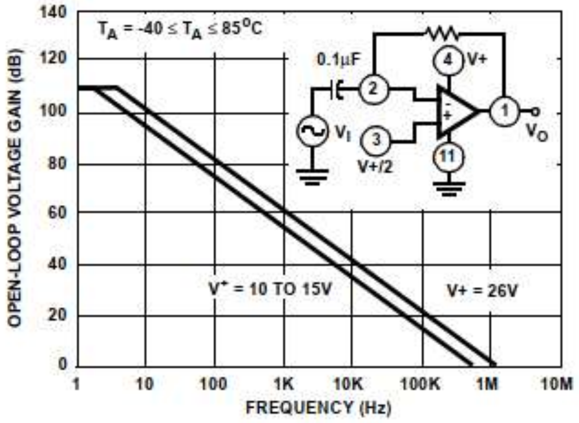


FIGURE 1. OPEN LOOP FREQUENCY RESPONSE

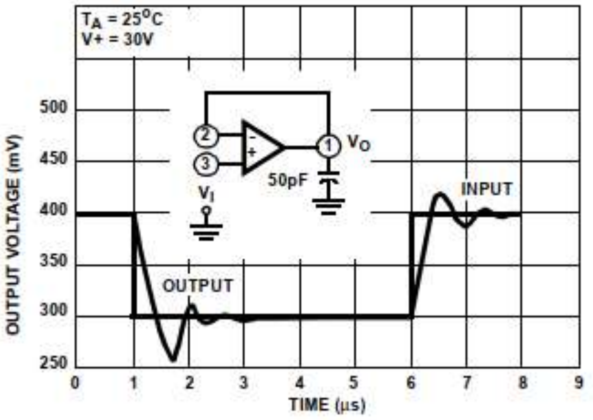


FIGURE 2. VOLTAGE FOLLOWER PULSE RESPONSE (SMALL SIGNAL)

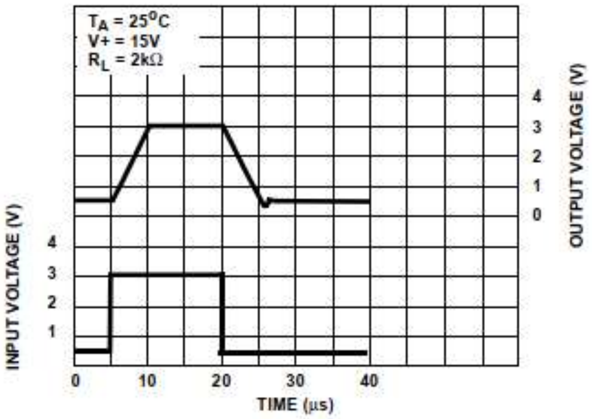


FIGURE 3. VOLTAGE FOLLOWER PULSE RESPONSE (LARGE SIGNAL)

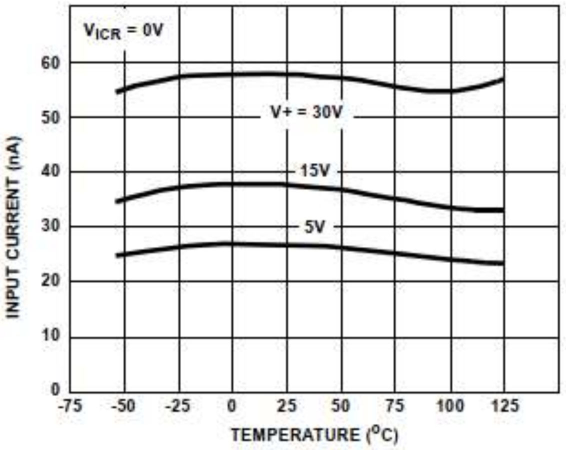


FIGURE 4. INPUT CURRENT vs AMBIENT TEMPERATURE

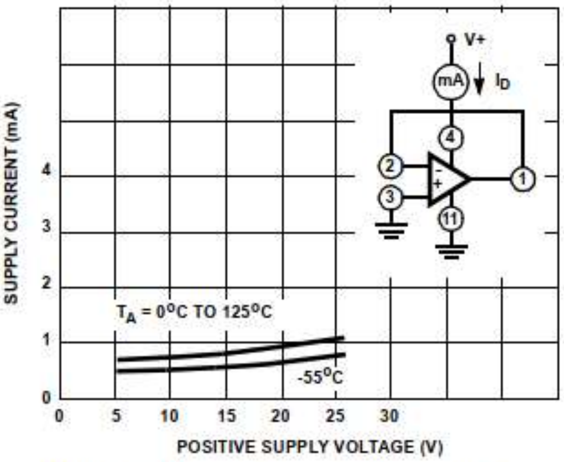


FIGURE 5. SUPPLY CURRENT vs SUPPLY VOLTAGE

CA124, CA224, CA324, LM324, LM2902

Typical Performance Curves (Continued)

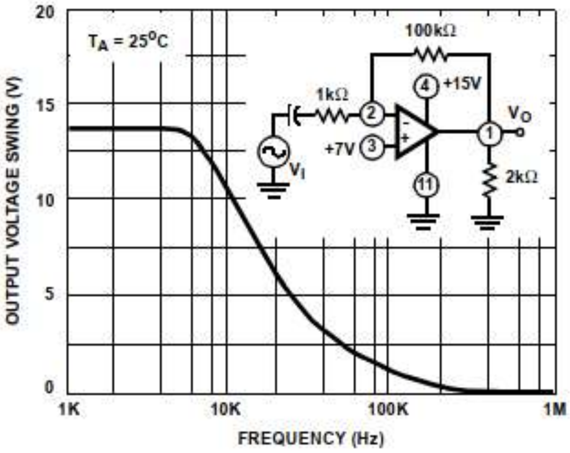


FIGURE 6. LARGE SIGNAL FREQUENCY RESPONSE

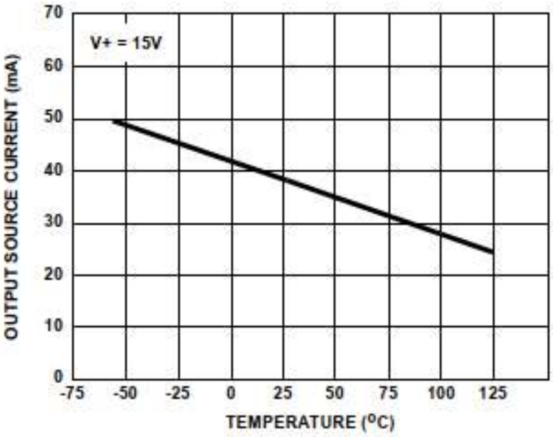


FIGURE 7. OUTPUT CURRENT vs AMBIENT TEMPERATURE

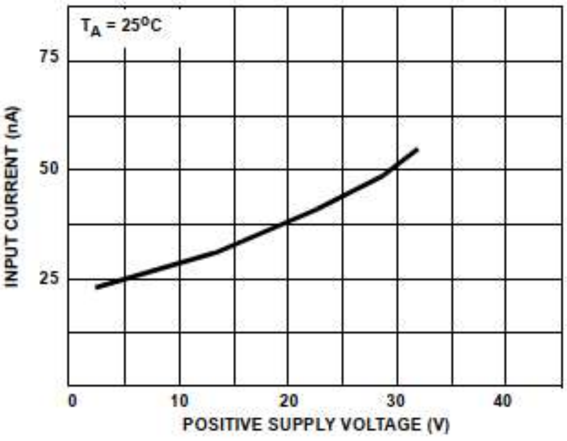


FIGURE 8. INPUT CURRENT vs SUPPLY VOLTAGE

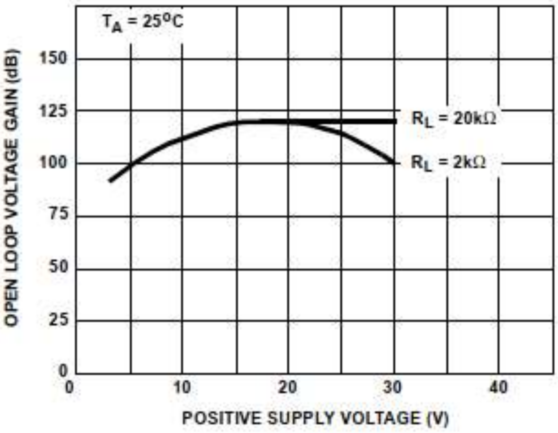
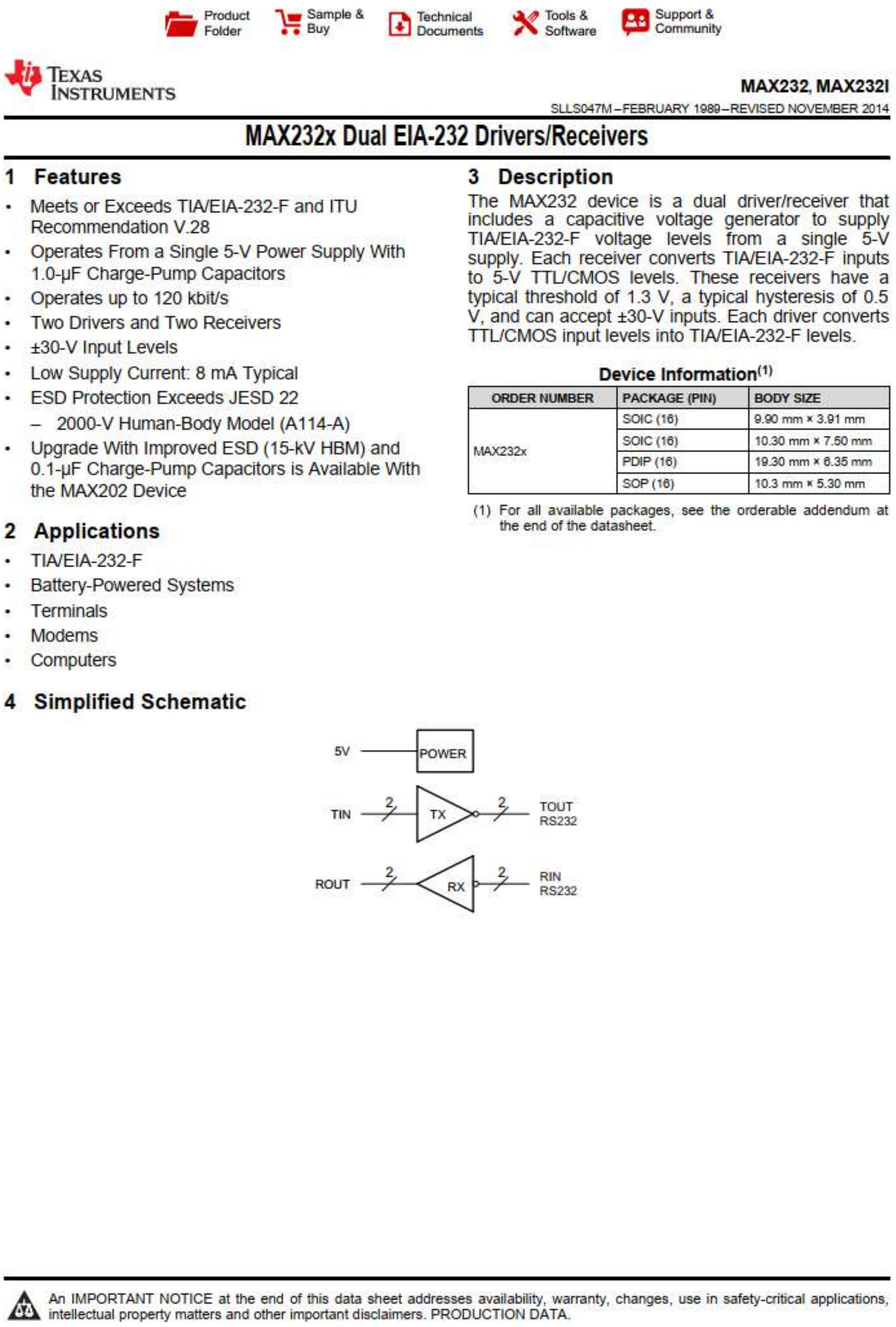


FIGURE 9. VOLTAGE GAIN vs SUPPLY VOLTAGE



Product Folder   Sample & Buy   Technical Documents   Tools & Software   Support & Community

TEXAS INSTRUMENTS

MAX232, MAX232I

SLLS047M – FEBRUARY 1989 – REVISED NOVEMBER 2014

MAX232x Dual EIA-232 Drivers/Receivers

1 Features

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0-µF Charge-Pump Capacitors
- Operates up to 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current: 8 mA Typical
- ESD Protection Exceeds JESD 22
  - – 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1-µF Charge-Pump Capacitors is Available With the MAX202 Device

2 Applications

- TIA/EIA-232-F
- Battery-Powered Systems
- Terminals
- Modems
- Computers

3 Description

The MAX232 device is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels.

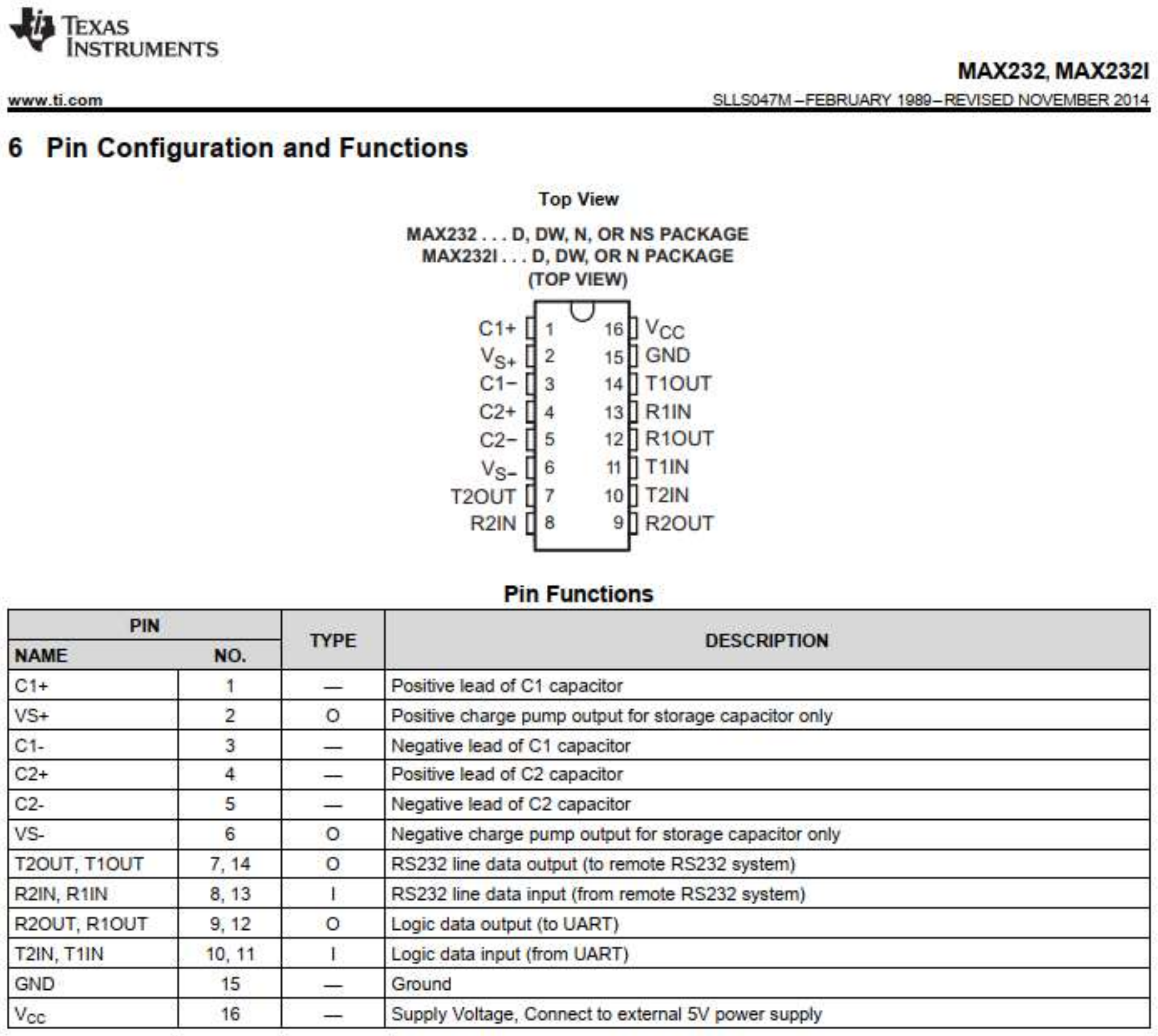
Device Information(1)

ORDER NUMBER	PACKAGE (PIN)	BODY SIZE
MAX232x	SOIC (16)	9.90 mm × 3.91 mm
	SOIC (16)	10.30 mm × 7.50 mm
	PDIP (16)	19.30 mm × 6.35 mm
	SOP (16)	10.3 mm × 5.30 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

4 Simplified Schematic

An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.



TEXAS INSTRUMENTS

MAX232, MAX232I

www.ti.com

SLLS047M – FEBRUARY 1989 – REVISED NOVEMBER 2014

6 Pin Configuration and Functions

Top View

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)

C1+ 1
V_{S+} 2
C1− 3
C2+ 4
C2− 5
V_{S-} 6
T2OUT 7
R2IN 8
16 V_{CC}
15 GND
14 T1OUT
13 R1IN
12 R1OUT
11 T1IN
10 T2IN
9 R2OUT

Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
C1+	1	—	Positive lead of C1 capacitor
VS+	2	O	Positive charge pump output for storage capacitor only
C1-	3	—	Negative lead of C1 capacitor
C2+	4	—	Positive lead of C2 capacitor
C2-	5	—	Negative lead of C2 capacitor
VS-	6	O	Negative charge pump output for storage capacitor only
T2OUT, T1OUT	7, 14	O	RS232 line data output (to remote RS232 system)
R2IN, R1IN	8, 13	I	RS232 line data input (from remote RS232 system)
R2OUT, R1OUT	9, 12	O	Logic data output (to UART)
T2IN, T1IN	10, 11	I	Logic data input (from UART)
GND	15	—	Ground
V_{CC}	16	—	Supply Voltage, Connect to external 5V power supply