DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

REAL TIME ACQUISITION AND WIRELESS TRANSMISSION OF ECG SIGNAL

by
Mehmet ZEYBEK

February, 2007
İZMİR

REAL TIME ACQUISITION AND WIRELESS TRANSMISSION OF ECG SIGNAL

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
in Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical & Electronics Engineering, Electrical & Electronics Engineering Program

by

Mehmet ZEYBEK

February, 2007

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**REAL TIME ACQUISITION AND WIRELESS TRANSMISSION OF ECG SIGNAL**"completed by **MEHMET ZEYBEK** under supervision of **ASST. PROF. DR. DAMLA KUNTALP** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. DAMLA KUNTALP

_____

Supervisor

…….. ……..

_____ _____

(Jury Member) (Jury Member)

_____

Prof.Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

First and foremost, I would like to express my respects and gratitude to my supervisor Asst. Prof. Dr. Damla KUNTALP. Many thanks for all her guidance, supports, suggestions, and patience on initiating, improving, and completing this thesis.

I would like to thank to Asst. Prof. Dr. Mehmet KUNTALP for his support and guidance throughtout this study.

Thanks to Scientific Research Projects department of Dokuz Eylül University for its monetary support; my valuable friend Murat TAVLI, İbrahim UĞURLU from UDEA Co., Haldun ÖZDEMİR from Medispro Co., Res. Asst. Evren DEĞİRMENCİ from METU, and my colleagues from Türk Telekom for their supports and guidance.

I would like thank to my family, my mother Ferda, my father Mustafa, and my sister Aslı. They've always supported, helped, and encouraged me.

Thanks to my fiancee Mine, for her love, patience, and endless supports.

Finally, I thank all my friends, whose names I couldn't mention here, for their good wishes and supports.

REAL TIME ACQUISITION AND WIRELESS TRANSMISSION OF ECG SIGNAL

ABSTRACT

Wireless communication technologies have been used in many areas of our lives recently. In addition to its many applications in communication sector, it has been started to use in biomedical sector, nowadays. There are many applications called as telemetry in this area.

In this study; we aimed to find new solutions for some difficulties in the biomedical applications by using wireless systems. With this aim, a mobile ECG acquisition circuit was designed. ECG signals obtained from the circuit were transmitted to the main computer by using RF technology. Then, transmitted ECG signals were drawn graphically on the main computer's screen.

Realized system consists of four main parts such as; ECG acquisition circuit, RF transmitter circuit including ADC, RF receiver circuit, and computer software.

The system has a structure which can be developed easily. So, the system will be able to be converted into a complex system which can be used for several medical signals and for several patients simultaneously by some improvements which will be done in the future on both for the hardware and the software.

Keywords : ECG, wireless, RF, telemetry, biomedical.

EKG SİNYALİNİN GERÇEK ZAMANLI OLARAK ALINMASI VE KABLOSUZ OLARAK İLETİLMESİ

ÖZ

Kablosuz erişim teknolojileri, son yıllarda hayatımızın birçok alanında yerini almıştır. İletişim alanındaki uygulamalarından sonra medikal alanda da sıkça kullanılmaya başlanan ve en genel anlamda telemetri olarak adlandırılan kablosuz erişim teknikleri ile bu alanda birçok uygulama geliştirilmiştir.

Bu çalışmada, biyomedikal uygulamalardaki bir takım zorluklara kablosuz uygulamalar ile yeni çözümler üretmek amaçlanmıştır. Bu amaç doğrultusunda, taşınabilir bir EKG ölçüm cihazı yapılmış ve bu cihaz ile elde edilen EKG işaretleri kablosuz (RF) olarak merkezi bir bilgisayara aktarılıp sinyallerin bu bilgisayarda görüntülenmesi sağlanmıştır.

Gerçekleştirilen sistem; EKG işareti elde etme devresi, analog-sayısal çevirici birimini de içeren radyo frekans verici devresi, radyo frekans alıcı devresi ve bilgisayar yazılımı olmak üzere dört ana kısımdan oluşmaktadır.

Oluşturulan sistem kolaylıkla geliştirilebilir bir yapıya sahiptir. Bu sayede ileride yapılacak donanımsal ve yazılmsal iyileştirmeler ile sistem birçok medikal işaret ve aynı anda birden fazla hasta için kullanılabilir bir hale dönüştürülebilecektir.

Anahtar sözcükler: EKG, kablosuz, RF (radyo frekans), telemetri, biyomedikal.

CONTENTS

Page

M.Sc THESIS EXAMINATION RESULT FORM ii
ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZ v

CHAPTER ONE - INTRODUCTION 1

CHAPTER TWO - THE HEART & ECG 3

2.1. The Heart 3
2.1.1 Anatomy of the Heart 3
2.1.2 Cardiac Histology 5
2.1.3 Metabolism in the Heart 7
2.1.4 The Action Potential of the Cardiac Cell 7

2.2 Electrocardiograph (ECG) 9
2.2.1 Discovery of the ECG 9
2.2.2 Excercise ECG 11
2.2.3 A Typical ECG Signal Acquisition 11
2.2.4 Requirements for an Electrocardiograph 12
2.2.5 Lead Configuration 13
2.2.6 Right-Leg Driver 14
2.2.7 Electrodes 14
2.2.7.1 Metal Plate Electrode 14
2.2.7.2 Mobile Type Electrode 15
2.2.7.3 Flexible Electrode 15
2.2.7.4 Suction Electrode 15

2.2.7.5 Disposable Electrode 15
2.2.7.6 Dried Electrode 15

CHAPTER THREE - SYSTEM DESIGN 17

3.1 ECG Acquisition Circuit 17
3.1.1 The Cable 17
3.1.2 Instrumentation Amplifier (AD624) 17
3.1.2.1 Setting the Gain 18
3.1.2.2 Noise Performance 19
3.1.2.3 Input Common Mode Range 19
3.1.2.4 Low Voltage Operation 20
3.1.3 AD712 20
3.1.3.1 Noise Performance 21
3.1.3.2 Low Voltage Operation 21
3.1.4 Shield Driver Circuit 21
3.1.5 Filter Circuits 22
3.1.5.1 Low Pass Filter 22
3.1.5.2 High Pass Filter 23
3.1.6 Base Line Positioning Circuit 24
3.1.7 Second Amplifier Circuit 25
3.1.8 Circuit Schema and PCB 26

3.2 ADC and Transmitter Circuits 28
3.2.1 16F877A Microcontroller 29
3.2.1.1 Overview of the File Registers 31
3.2.1.2 Special Function Registers 31
3.2.1.3 Overview of the ADC 31
3.2.1.4 Overview of the Hardware USART 34
3.2.2 UDEA UTR-C12 U RF Transceiver 34
3.2.3 Circuit Schema and PCB 38
3.2.4 Data Transmission Protocol 39

3.3 RF Receiver and RS232 Circuits 40
3.3.1 RS232 Serial Interface 41
3.3.1.1 MAX232 43
3.3.2 Receiver Unit Circuit Schema and PCB 44
3.3.3 Receiving RF Data and Serial Communication with PC 46

3.4 ECG Drawing Software 47
3.4.1 About the Software 47
3.4.2. Serial Port Reader 47
3.4.3 Main Program 48
3.4.3.1 Configuring the Communication Parameters 49
3.4.3.2 New Patient Registering 50
3.4.3.3 Capturing ECG Data 51
3.4.3.4 Saving ECG Data 53
3.4.3.5 Drawing the Old Records 53
3.4.3.6 Searching the Old Records 54

CHAPTER FOUR - RESULTS 56

4.1 ECG Acquisition Circuit 56

4.2 RF Transmitter & Receiver Circuits 59

4.3 ECG Drawing Software 62

4.4 Test of the System 63

CHAPTER FIVE - DISCUSSION 65

5.1 Future Works 65

REFERENCES 67

APPENDICES 70

APPENDIX A AD624 Datasheet 70
APPENDIX B AD712 Datasheet..... 70
APPENDIX C 16F877A Datasheet 70
APPENDIX D UTR-C12 U Datasheet..... 70
APPENDIX E Source Codes of the PICs..... 70
APPENDIX F Source Code of the ECG Drawing Software..... 70

CHAPTER ONE

INTRODUCTION

The term **telemedicine** is the delivery of medicine at a distance. Telemedicine is not a new concept. It has existed for several years in the form of the telephone and fax machines. In recent years, with the improvements made in access techniques, technology, and communications systems, telemedicine has expanded. It has become a feasible alternative for smaller and rural medical facilities to provide routine and specialized services.

Recent breakthroughs in communication technologies have stimulated the development and demonstration projects in telemedicine. There are several products and projects (Finch, 1999) within mobile ECG recording using Internet solutions (Hernandez, Mora, Villegas & Carrault, 2001), Bluetooth technology (Andreasson, Ekström, Fard, Castano & Johnson, 2002), cellular phones (Istepanian, Woodward, Gorilas & Balos, 1998) (Schiller, 2003), WAP-based implementations (Hung & Zhang, 2003), and wireless local area networks, WLAN (Rollins, Killingsworth, Walcott, Justice & Ideker, 2000).

This thesis proposes a system to acquire and transmit ECG signal using wireless transmission medium. The system consists of four main parts; the ECG signal acquisition circuit, the ADC & transmitter unit, receiver unit, and the computer software. Three electrodes are placed on the patient skin. One electrode is used as a positive lead, one as a negative lead, and the third as a virtual ground or reference. The positive and negative signals are fed into an instrumentation amplifier (AD624) which has high CMMR, then a series of filters and amplifiers. The virtual ground provided by the third electrode is connected to the right leg driver.

Once the ECG signal is acquired properly then the signal is sent to an ADC included in the microcontroller (16F877A). After the sampling of the ECG signal, converted data is transferred to RF Module (UTR-C12 U). As soon as the signal is received by the receiver unit, it is sent to serial port of the PC according to RS-232

protocol by the microcontroller. Received data from the serial port is drawn graphically by the software running on the PC. Block diagram of the system is shown in Figure 1.1.

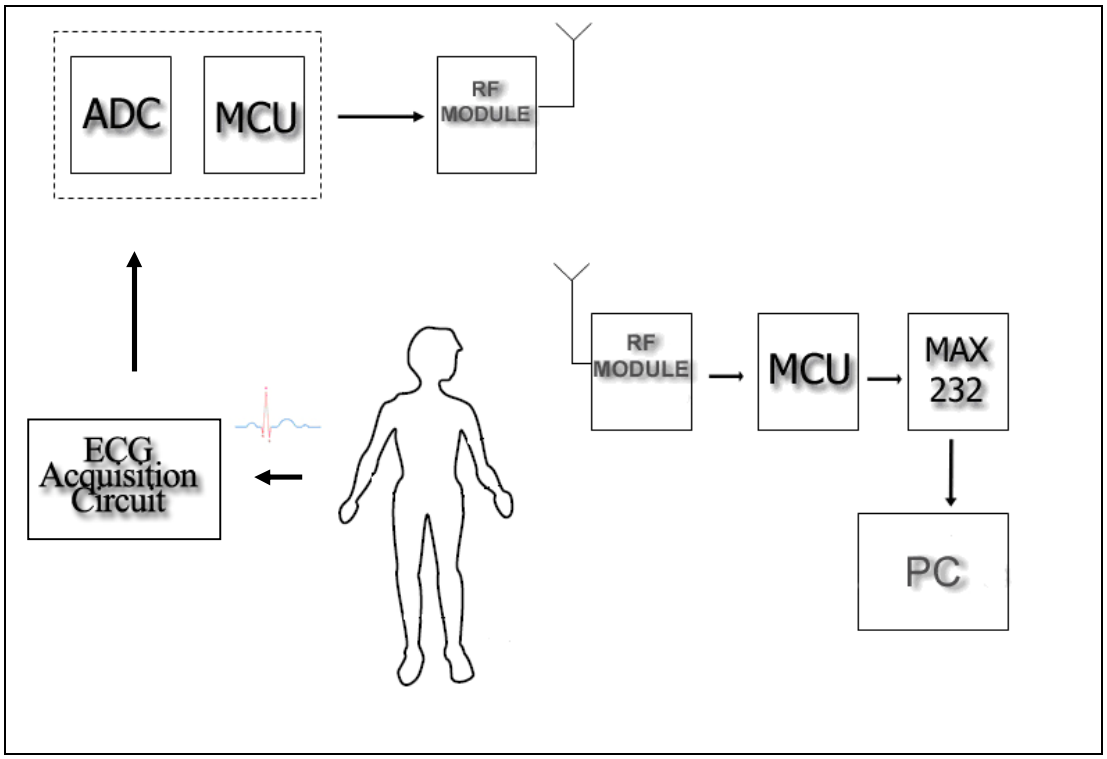


Figure 1.1 Block diagram of the system.

The above descriptions are intended to provide a general understanding about how this wireless ECG system was designed. In the pages that follow, more detailed information will be provided.

CHAPTER TWO

THE HEART & ECG

2.1. The Heart

2.1.1 Anatomy of the Heart

The heart has four chambers, two atrias and two ventricles, through which the blood is pumped (see Figure 2.1). Valves of the heart control the blood flow direction.

Walls of the heart include a fibrous "skeleton" for attachment of the valves, the cardiac muscle, and a specialized conduction system.

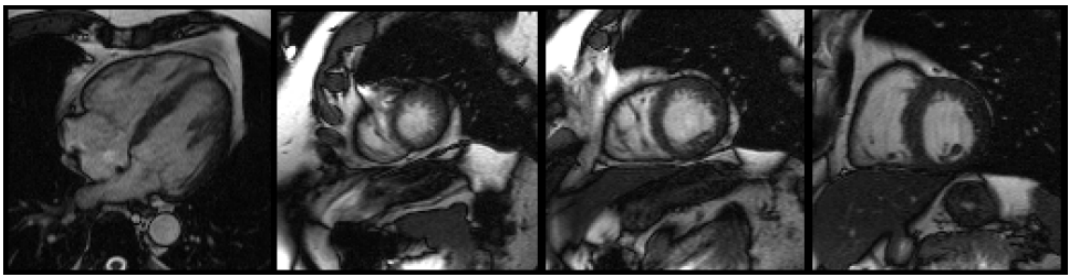


Figure 2.1 Four cross sections of the heart from left to right (Marieb, 1998).

A cardiac cycle starts with contraction of the atria, with subsequent blood flow from the atria to the ventricles. As the ventricles are contracting during the systolic phase, blood is pumped from the left ventricle to the systemic circulation, and from the right ventricle to the lungs. During the diastolic phase, the atria and also to some extent the ventricles are again filled with blood.

Because of the pressure generation of the ventricles, atrial walls are thinner than the ventricular walls. Normally, the wall thickness of the right ventricle is about 0.5 cm and on the left ventricle about 1.5 cm. However, the volume of fully expanded ventricles is approximately the same.

Purkinje observed the ventricular conduction system of the heart in 1845. He used a heart of a sheep for this observation. Then, Tawara presented his works about conduction system of the heart in the end of the last century. Tawara supposed that, the so far mysterious Purkinje network was in fact a continuation of the His bundle, and that the conduction of the electrical signal occurred along this network at a faster rate than through the rest of the myocardium. Later investigations confirmed that, the differences between species which are shown by Tawara (Tranum-Jensen, Wilde, Vermeulen, & Janse, 1991; Massing & James, 1976).

The anatomy of the human cardiac conduction system is complex. On the other hand, this complexity in the conduction network exists mainly to provide necessary redundancy to this critical system.

The ventricular conduction system starts from the atrioventricular node (Tawara node) as the His bundle. The His bundle passes through the atrioventricular ring into the membranous interventricular septum. Usually, the bundle then travels for about 15 mm on the left side of the septum (Massing & James, 1976), and gives off the left and right bundle branches.

Although the structure of the conduction system in the left ventricle is generally described as bifascicular (Rosenbaum, 1970), this has no basis in the actual anatomy. The term “hemiblock" refers to functional anomalies of the left conduction system, but it is also electro physiologically a very crude classification. The fascicles in the branch are highly interconnected. The anterior rim of the left bundle travels towards the left anterior papillary muscle, while the posterior rim is oriented toward the left posterior papillary muscle.

The right bundle branch usually begins from the distal part of the His bundle (Titus, 1973). It courses subendocardially and intramyocardially towards the right anterior papillary muscle. At the papillary muscle, it divides into fascicles that continue as the right Purkinje network leading the activation to the posterior papillary muscle and the right ventricular free wall.

The Purkinje fibres travel on the endocardial surface of the trabecular muscles in the human heart. These fibres are clearly visible in ungulates, and these animals have frequently been used in the studies of the conduction system. On the other hand, no connection has been shown to exist between the endocardial Purkinje fibres and the midmyocardial (M) cells in the human heart, although these cells resemble each other in their electrophysiological properties (Yan, Shimizu, & Antzelevitch, 1998). In contrast, the Purkinje fibres run also intramurally in the pig heart (Tranum-Jensen, Wilde, Vermeulen, & Janse, 1991).

2.1.2 Cardiac Histology

Muscle fibres consist of large number of myofibrils. Myofibrils are comprised of myofilaments (actin and myosin). The organization of these filaments are determined the muscle's characteristic. The contraction of a muscle happens as the two types of myofilaments slide past each other. The energy for the contraction is provided by adenosine triphosphate.

The cardiac muscle fibres consist of several muscle cells. These cells are aligned end to end and connected by so called intercalated discs that convey the electrical activation between cells. The electrical connection between two cells takes place through the gap junctions present in the intercalated discs. Some cells in the fibre may also join with two other cells creating a branching of the fibre (see Figure 2.2). The muscle cells of the heart are usually approximately 100 μm long, and 10-40 μm in diameter. The shape and size of the these cardiac muscle cells depends on their location in the heart. For instance, the cells which are on the epicardium or on the endocardium are typically smaller in diameter than intramural cells.

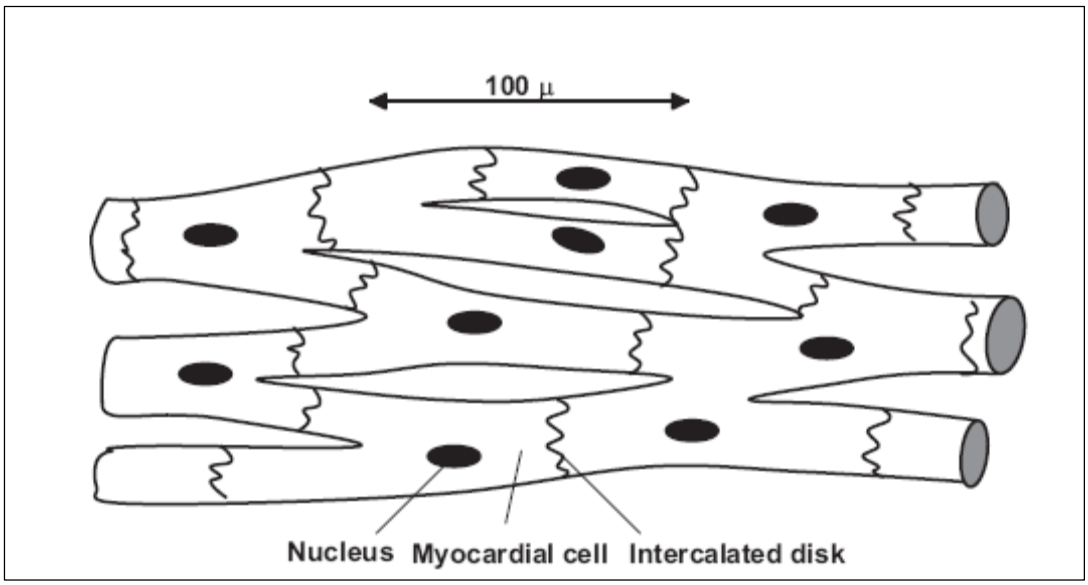


Figure 2.2 Organization of the cardiac muscle. Long, cross-striated muscle fibres make up the cardiac muscle. Between cells, the electrical activation is conveyed by the intercalated discs. Branching is also typical of the cardiac muscle cells (Ross, 1989).

The conduction system of the heart consists of specialized cardiac muscle cells. They transfer the electrical signal at a rate approximately four times faster (1–2 m/s) than the cardiac muscle cells (0.3–0.8 m/s) (Dam & Janse, 1989; Jack, Noble, & Tsien, 1983). The cells of the conduction system are responsible for the coordination of the atria & ventricles' contraction.

The cells which are in the conduction system are smaller than myocardial cells. There are less transverse gap junctions between the cells (Truex, 1974; Martinez-Palomo, Alanis, & Benitez, 1970), which leads to higher conduction velocity in these cells. From the Purkinje network, the activation is transferred to the myocardium by transitional cells in Purkinje-myocardial junctions (PMJs) (Tranum-Jensen, Wilde, Vermeulen, & Janse, 1991). These cells amplify the current so that the low-impedance load of the myocardium does not hamper propagation in the conduction network (Mendez, Mueller, & Urguiaga, 1970). This property may facilitate re-entry through the Purkinje system, as reported in some studies (Berenfeld & Jalife, 1998).

2.1.3 Metabolism in the Heart

Cardiac muscle cells need energy primarily for two purposes: to carry out mechanical work (contraction and relaxation) and to maintain excitability with the aid of ion pumps. Like in any muscle, the cardiac muscle cells receive their energy from the breaking of the ATP molecules. These molecules are constructed again in the cell if sufficient oxygen is present. The oxygenation of the cardiac muscle cells is provided by coronary arteries and the cardiac microcirculation.

The coronary arterial circulation consists of arteries on the epicardium and in the cardiac muscle. Three major coronary arteries supply the heart: left anterior descending artery (LAD), left circumflex artery (LCX), and right coronary artery (RCA). Although all parts of the heart are supplied mainly by one coronary artery, in a normal heart there are several collateral arteries taking care of blood supply, e.g., in case of a narrowing in one coronary artery in coronary artery disease (CAD). However, these collateral arteries supply blood to subendocardial areas less than the coronary arteries (Ross, 1989).

Decrease in blood supply to the cardiac muscle results in a development of ischemia or ultimately myocardial infarction, decreased availability of nutrition and buildup of metabolites. This syndrome is a result of, e.g., ischemic heart disease. The lack of blood supply causes "chest pain", i.e. angina pectoris, or in a more severe case a "heart attack", i.e. myocardial infarction (Katz, 2000).

2.1.4 The Action Potential of the Cardiac Cell

The cardiac muscle cell's resting membrane potential, which is defined against the potential of the extra cellular medium, is between -80 mV and -95 mV. Because of activation from contiguous cells or leakage currents, if this potential is raised above -60 mV, an action potential is generated. First, the sodium channels which is voltage sensitive open letting sodium ions flow into the cell and depolarizing the membrane to +20 mV. This rapid transition is followed by a plateau phase at 0 mV in some cells, when permeabilities for all ions are close to zero. The opening of the slower

potassium channels starts the repolarization phase of the membrane, and the membrane potential returns to its resting value, possibly through a modest undershoot. Small changes in the resting membrane potential are overcome by the increased potassium current due to deflection from the equilibrium potential of this ion.

After the fast upstroke of the membrane potential, the conductance of sodium decreases rapidly as the channels enter an inactive state due to depolarization of the membrane. The depolarization activates the influx of calcium ions, further leading to a massive release of calcium from the sarcoplasmic reticulum. The increase of intracellular calcium concentration starts the contraction of the muscle cell by acting on the troponin C in the myofilaments. At the same time, a transient external current of potassium ions starts to repolarize the membrane. After the closing of calcium channels, the external flux of potassium ions finishes the repolarization of the membrane, and the membrane potential returns close to the equilibrium potential of potassium. The ion pumps and many other smaller ionic currents also take part in the action potential of the cardiac muscle cell (Carmeliet, 1993).

Because of the different forms of cells and different arrangements for action potential conduction between cells, different activation velocities appear in different tissue types. The typical conduction velocity in a long nerve ending is 10–100 m/s, whereas the velocity in a cardiac muscle cell is only about 0.3–0.8 m/s in longitudinal direction (Dam & Janse, 1989; Jack, Noble, & Tsien, 1983). The propagation speed depends also strongly on cardiac muscle fibre orientation in the transverse direction; the propagation speed is less than half of the speed in longitudinal direction. In the cells belonging to the conduction system, the propagation velocity can be up to 2 m/s.

2.2 Electrocardiograph (ECG)

2.2.1 Discovery of the ECG

The idea of the ECG has been around since 1887, yet it remains one of the most important indicators of proper heart functioning today. The ECG is a non-invasive technique, meaning that this signal can be measured without entering the body at all. Electrodes are placed on the patients' skin to detect the bioelectric potentials given off by the heart that reach the skin's surface (Cromwell, Leslie, Weibell, & Pfeiffer (1980).

The cyclic changes in the heart cell's polarization are what produce the standard ECG wave, shown below in Figure 2.3. The letters which are the names given to different wave sections. The flat section that comes before the P-wave is called the baseline. The table shown in Table 2.1 highlights ECG sections and what they indicate.

Table 2.1 Electrocardiogram basic characteristics

Section of Electrocardiogram	**Source**
P-wave	Atrial Excitation
QRS-Complex	Atriarepolarization+Ventricle depolarization
T-Wave	Ventricle depolarization
P-Q Interval	Excitation timing delay

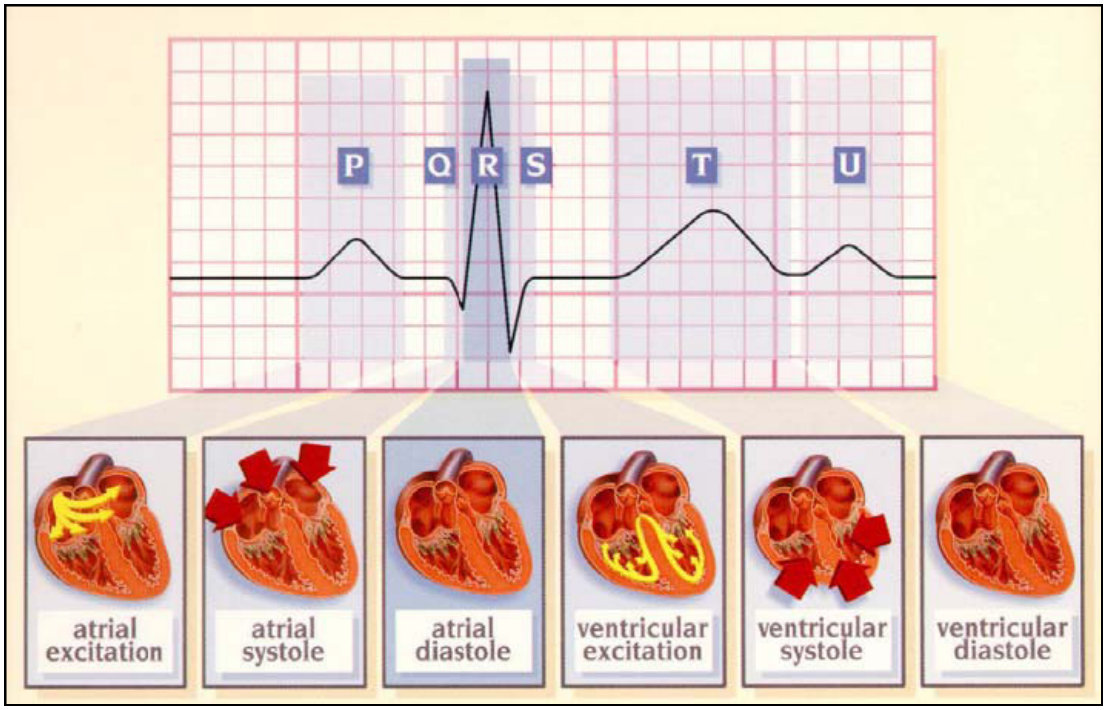


Figure 2.3 An electrocardiogram waveform.

There are many methodologies for hooking-up a subject for an ECG study. As many as 12 electrodes could be hooked up to a patient to offer the examiner a wide selection of waveforms, each highlighting various heart characteristics better than others. However, three electrodes used in our system can still record a typical ECG waveform. One electrode serves as the right leg driver and the others serve to measure the heart's electrical activity. We have chosen disposable skin electrodes for our system. It is important that the ground electrode (for right leg driver) be away from the heart, away from a lot of muscle activity. The waveform noise level and clarity is very sensitive to the placement of these electrodes and they need to be adjusted, to find the optimal location.

The ECG wave serves as an excellent tool for heart analysis. The ECG wave has been studied over the years to yield nominal values of both individual wave amplitudes and time intervals between waves. A summary of these values is shown in Table 2.2.

Table 2.2 Nominal ECG parameters

ECG Wave	*Amplitude*	*ECG Interval*	*Duration*
P	0.25 mV	P-R	0.12 - 0.20 sec
R	1.60 mV	Q-T	0.35 – 0.44 sec
Q	25% of R	S-T	0.05 – 0.15 sec
T	0.1-0.5 mV	Q-R-S	0.09 sec

By comparing a given ECG wave's parameters against these nominal values, insight can be found into potential problems. Our system is able to take an ECG waveform from the patient and transmit it on 433 MHz to a receiving unit where the ECG signal can be viewed on the PC software.

2.2.2 Exercise ECG

Electrocardiograms often are obtained during exercise. These are important diagnostic tests. As exercise intensity increases, the heart must beat faster and work harder to deliver more blood to active muscles. If the heart is diseased, an indication may show up on the electrocardiogram as the heart increases its rate of work. Exercise ECGs are also invaluable tools for research in exercise physiology because they provide a convenient method for tracking cardiac changes during acute and chronic exercise (Costill, 1999).

2.2.3 A Typical ECG Signal Acquisition

Figure 2.4 shows a block diagram of a typical single channel electrocardiograph. In that chain it is apparent that all filtering is done in the analog domain, while the microcontroller is used principally for communication and other downstream purposes.

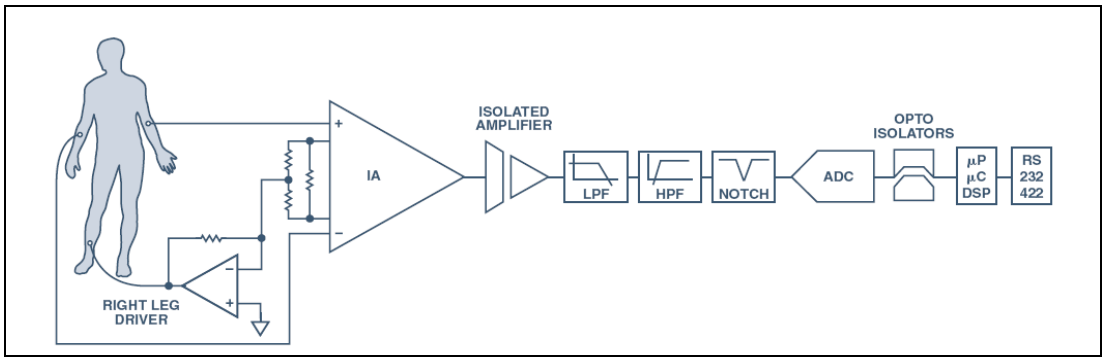


Figure 2.4 Block diagram of single channel ECG.

2.2.4 Requirements for an Electrocardiograph

The front end of an ECG must be able to deal with extremely weak signals ranging from 0.5 mV to 5.0 mV, combined with a DC component of up to ∓300 mV resulting from the electrode plus a common-mode component of up to 1.5 V, resulting from the potential between the electrodes and ground. The useful bandwidth of an ECG signal, depending on the application, can range from 0.5 Hz to 50 Hz (for a monitoring application in intensive care units), up to 1 kHz for late-potential measurements (pacemaker detection). A standard clinical ECG application has a bandwidth of 0.05 Hz to 100 Hz. Also we use this range too.

ECG signals may be corrupted by various kinds of noise. The main sources of noise are:

- Power-line interference: 50–60 Hz pickup and harmonics from the power mains
- Electrode contact noise: variable contact between the electrode and the skin, causing baseline drift
- Motion artifacts: shifts in the baseline caused by changes in the electrode-skin impedance

- Muscle contraction: electromyogram-type signals (EMG) are generated and mixed with the ECG signals

- Respiration, causing drift in the baseline

- Electromagnetic interference from other electronic devices, with the electrode wires serving as antennas, and

- Noise coupled from other electronic devices, usually at high frequencies.

For meaningful and accurate detection, steps have to be taken to filter out or discard all these noise sources (Analog Devices, 2003).

2.2.5 Lead Configuration

To perform an electrocardiograph, it is important that more than one lead (also known as a channel) be recorded in order to accurately describe the heart's electrical activity. For a two or three channel ECG, only the leads in the frontal plane are required. Typical electrode placement is shown in Figure 2.5.

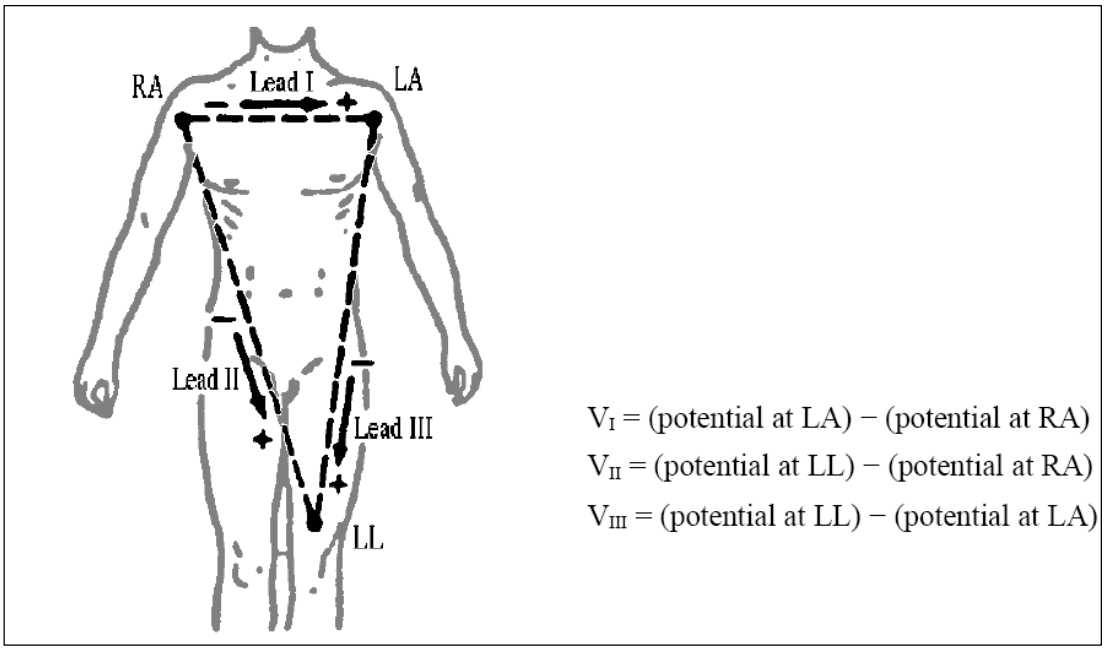


Figure 2.5 Typical electrode placements (Townsend, 2001).

Very often an electrode is also placed on the right leg (RL) and grounded or connected to a Driven- Right-Leg circuit.

2.2.6 Right-Leg Driver

In modern ECG recording systems, the patient is often not grounded. Instead of this, the right leg electrode is connected to the output of an auxiliary opamp. The common mode voltage on the body is sensed by two averaging resistors, inverted and fed back to the right leg. This circuit actually drives a very small amount of current (less than 1 μA) into the right leg to equal the displacement currents flowing in the body. The body therefore becomes a summing junction in a feedback loop and the negative feedback from this circuit drives the common mode voltage to a low value.

The circuit also helps to increase the patient's safety. If an abnormally high voltage should appear between the patient and ground due to electrical leakage or other means, the auxiliary opamp in the right leg circuit saturates. This effectively ungrounds the patient since the amplifier can no longer drive the right leg. The resistance between the patient and ground is usually several MΩ and is therefore large enough to protect the patient. With a 5 MΩ resistor, for examples, and a supply voltage of 9 V, the amplifier will saturate at a current of approximately 2 μA.

2.2.7 Electrodes

One of the most important factors to obtain a correct ECG signal is used the electrode suitable and high in quality. There are various ECG electrodes.

2.2.7.1 Metal Plate Electrode

It is shaped disk or rectangular. It is used alloy of Ni-Ag (Ag-AgCl). It is applied gel to between skin and electrode. It is attached with special rubber or strap. It is used for ECG, EMG, and EEG. It has big surface and small impedance.

2.2.7.2 Mobile Type Electrode

The electrode is in a guard and also there is electrolyte gel in this. The electrode is mounted with plaster. Even the electrode moves, the electrolyte gel moves with the electrode. In this way noise becomes smaller. Ag-AgCl is used in this type.

2.2.7.3 Flexible Electrode

One surface of this electrode is a flexible plaster. This type electrodes used for children can conduct X-ray. The thickness of Ag-AgCl is 2μm.

2.2.7.4 Suction Electrode

This type of electrodes for ECG has cylindrical tube to contact with skin and vacuum pomp. It has small surface and big impedance.

2.2.7.5 Disposable Electrode

This type of electrodes for ECG uses Ag-AgCl electrodes. It has a gel layer. It is throwed after used. Because of easy of use, ARBO disposable electrode was used in our project.

2.2.7.6 Dried Electrode

Don't need any gel or solution. SiO_2 is used on the base of the electrode.

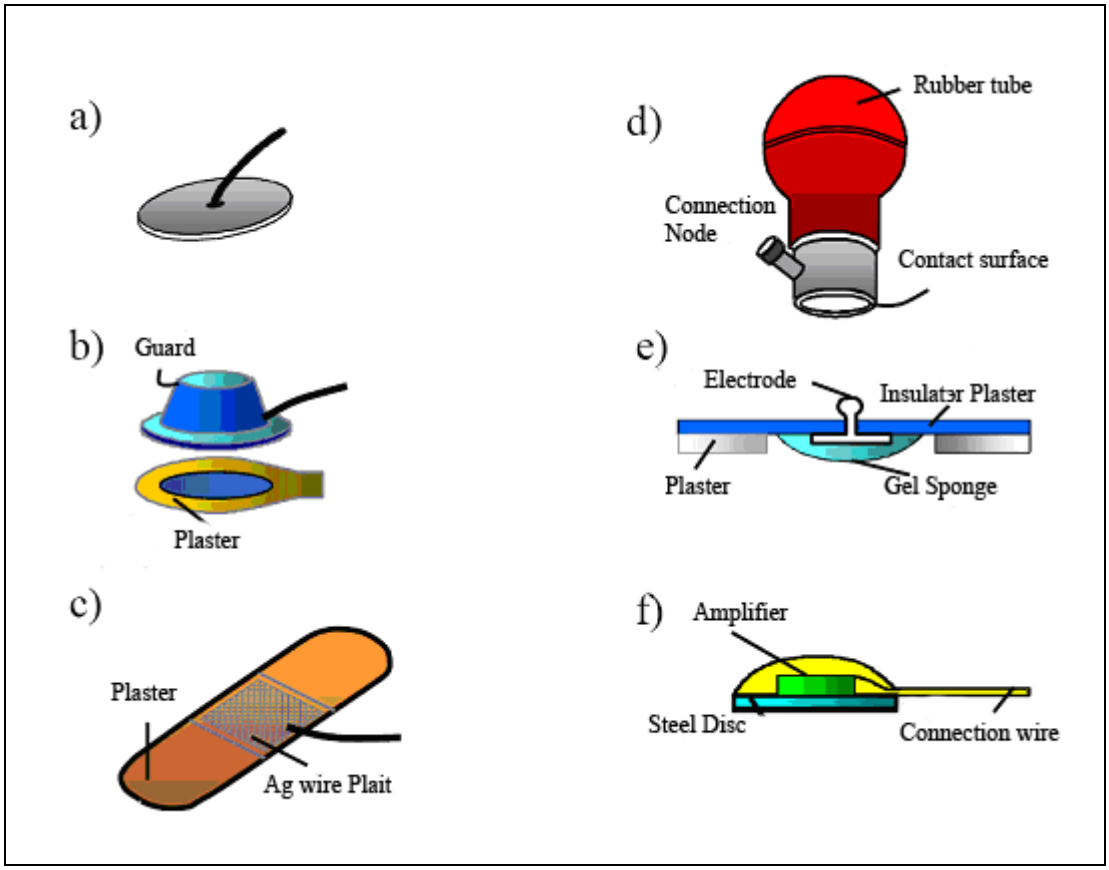


Figure 2.6 Types of surface electrodes.

CHAPTER THREE

SYSTEM DESIGN

3.1 ECG Acquisition Circuit

A circuit with two electrodes and with right leg driver has been designed to obtain an ECG signal.

3.1.1 The Cable

One of the most important factors is the cables by which after taking electrical signals on body surface are carried to amplifier. For this, a cable mechanism with two parts was designed.

Three RG174 shielded cable ends of which crocodile mounted to was cut 110 cm in length. On the electrode side, the shields of these cables were left free. But on the other side were connected together. The cables coming here separately after this point connected to quad shielded cable 0.22 mm in diameter and 150 cm in length and then mounted to female connector. Also a male connector was added to the box which includes ECG acquisition and RF module circuits. The aim of using these male and female connectors is to make easier connection between the cable and the box.

3.1.2 Instrumentation Amplifier (AD624)

V_{pp} of the ECG signal from the patient is generally about 1 mV. This is clearly a very low level signal, which is unusable at its current amplitude. We added two gain stages to boost this V_{pp} up to fully utilize our rail levels (set to –9V, 9V by our power supply). Because of the initial low level of the signal, noise was a serious issue. White noise alone was enough to distort any input signal that we received from the patient. Examining the input signals directly with a scope, the signal would look indistinguishable from the scope readout with nothing connected up to its probes.

Because of this problem we used a differential amplifier, allowing us to effectively ignore the common signal (main, RFI, and EMI noises) between the positive and negative leads from the patient. We used an AD624 low-power instrumentation amplifier from Analog Devices. The gain for this stage was set to 100, and the output was single ended.

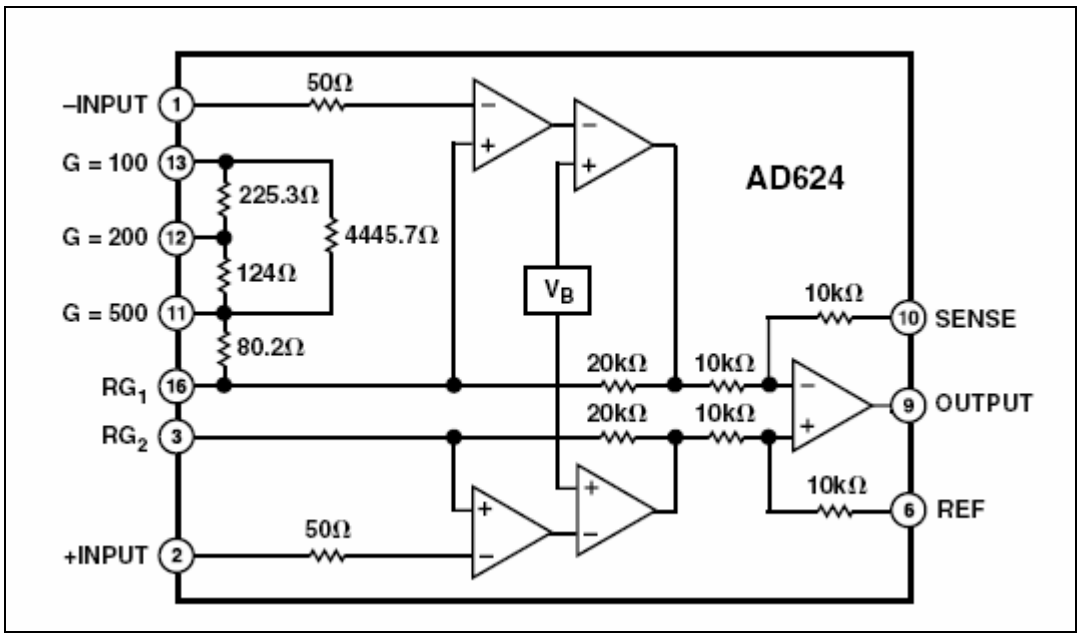


Figure 3.1 Block diagram of the AD624.

3.1.2.1 Setting the Gain

The AD624 includes high accuracy pretrimmed internal gain resistors. These allow for single connection programming of gains of 1, 100, 200, and 500. Additionally, a variety of gains including a pretrimmed gain of 1000 can be achieved through series and parallel combinations of the internal resistors. If desired value of gain is not attainable using the internal resistors, a single resistor can be used to achieve any gain between 1 and 10,000. This resistor connected between pins 3 and 16 programs the gain according to the formula;

$$R_G = \frac{40k\Omega}{G-1}$$

The 40kΩ term comes from the sum of the two internal feedback resistors of pre-amplifiers of the AD624 chip (shown in Figure 3.1). These on-chip metal film resistors are laser trimmed to accurate absolute values.

In our design, we used two 200 Ω resistors to set gain of AD624 is 100.

$$G = \frac{40k\Omega}{2 \cdot (200\Omega)} + 1$$

$$G \cong 100$$

3.1.2.2 Noise Performance

The AD624 is designed to provide noise performance near the theoretical noise floor. There are two sources of noise in an instrumentation amplifier, the input noise, predominantly generated by the differential input stage, and the output noise, generated by the output amplifier.

The low frequency noise is approximately 0.3 μV_{pp} measured from 0.1 to 10Hz (G $\geq$ 100).

3.1.2.3 Input Common Mode Range

The linear input voltage range of the input circuitry of the AD624 is from approximately 1.4 V below the positive supply voltage to 1.7 V above the negative supply. As a differential input voltage causes the output voltage increase, however, the linear input range will be limited by the output voltage swing of input amplifiers. So, the linear common mode input range is related to the output voltage of the complete amplifier. This behavior also depends on supply voltage.

Input overload can produce an output voltage that appears normal. For example, if an input overload condition drives both input amplifiers to their positive output swing limit, the difference voltage measured by the output amplifier will be near

zero. The output of the AD624 will be near 0 V even though both inputs are overloaded.

3.1.2.4 Low Voltage Operation

The AD624 can be operated on power supplies as low as ±6 V. Performance remains excellent with power supplies ranging from ±6 V to ±18 V.

In the designed our ECG circuit ±9 V DC feeding voltage was obtained by using two 9V batteries. These values were used to prevent the saturation of operational amplifiers that are input and output of ECG signals.

The datasheet of the AD624 is given in Appendix A.

3.1.3 AD712

AD712 is a high speed, precision monolithic dual op-amp which offers high performance. It has very low offset voltage and offset voltage drift. Also, its CMMR is 88 dB. AD712 was preferred to use for filtering and also amplifying because of these performance benefits. Owing to it has two operational amplifiers, AD712 takes little place.

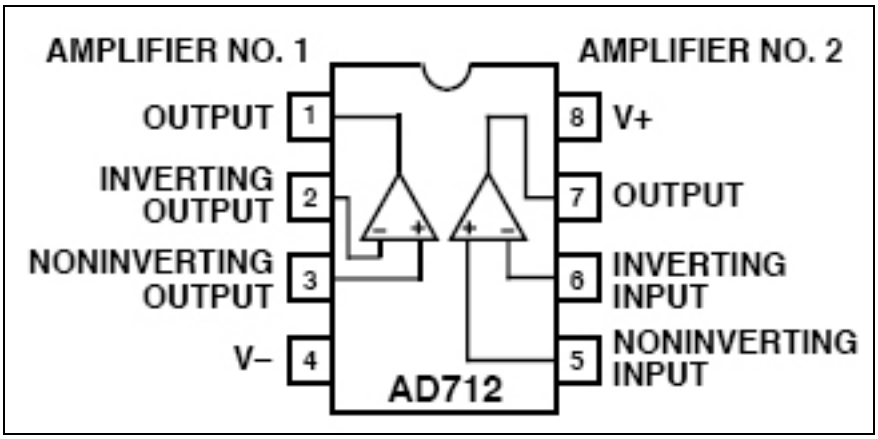


Figure 3.2 Block diagram of AD712.

3.1.3.1 Noise Performance

The AD712 is designed to provide good noise performance. The low frequency noise is approximately 4 μV_{pp} measured from 0.1 to 10Hz. This value was appropriate for our design.

3.1.3.2 Low Voltage Operation

The AD6712 can be operated on power supplies as low as ±4.5 V. Performance remains excellent with power supplies ranging from ±4.5 V to ±18 V. Due to the fact that we use two 9 V batteries in our design, AD712 is suitable for our design.

The datasheet of the AD712 is given in Appendix B.

3.1.4 Shield Driver Circuit

Noise sources (i.e. power line, GSM, etc.) affect on our circuit. Especially, power line noise was the biggest problem to acquire ECG signal. Therefore, in addition to right-leg driver, a shield driver circuit (see Figure 3.2) was designed for better noise cancellation. For this, common mode voltage (V_G) which was taken mid-point of the two 200 Ω resistors (Rg) was transferred to positive input of the first op-amp block of the AD712. Output of the first block is equal to V_G. Then, this common mode voltage was applied to shields of the cables. Thus, a feedback was provided by this common mode voltage. So, common mode rejection performance of the circuit was increased with this improvement. Second op-amp block of the AD712 was used for the right-leg driver circuit. Right-leg driver and shield driver circuits are shown in Figure 3.3.

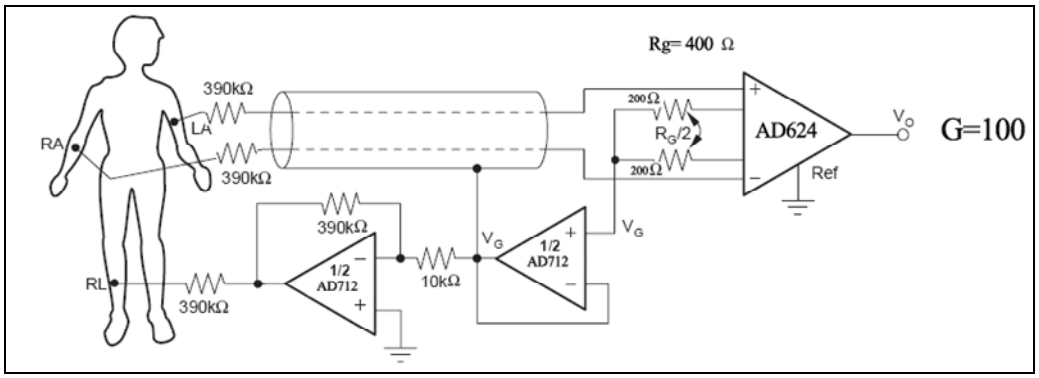


Figure 3.3 ECG amplifier with right-leg drive.

3.1.5 Filter Circuits

3.1.5.1 Low Pass Filter

In our research on the origins of the ECG signal and some of its characteristics, we found that almost all of the ECG signal's frequency content was below 100 Hz. For this reason, we decided to use low pass filter to attenuate frequencies higher than this value. We designed two 2^{nd} order active low pass filters whose cut-off frequencies are about 30 Hz. Because of the cut-off frequency of the filters, we didn't have to use any notch filter to reject 50 Hz power line noise. Also the other noises such as RFI and EMI are rejected by these filters.

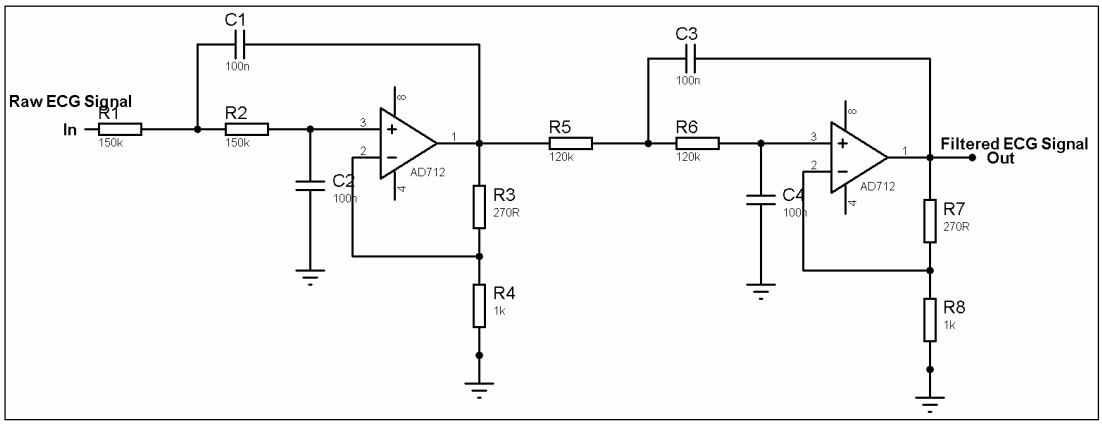


Figure 3.4 Low pass filter.

$$f_{cut}\ (3\ dB) = \frac{1}{2 \cdot \pi \cdot \sqrt{R_1 \cdot C_1 \cdot R_2 \cdot C_2}}$$

$$f_{cut}\ (3\ dB) = \frac{1}{2 \cdot \pi \cdot \sqrt{150k \cdot 33n \cdot 150k \cdot 33n}}$$

$$f_{cut}\ (3\ dB) \cong 30Hz$$

3.1.5.2 High Pass Filter

Before amplifying the signal further, we decided to use a simple RC high pass filter to block any DC bias that the signal was riding on. The DC level of the signal tended to fluctuate and caused difficulty in subsequent circuit stages. The time constant for this filter was set to pass very low frequencies (on the order of 0.5 Hz) without attenuation. This is nearly the lowest frequency contribution of the ECG frequency spectrum. Thanks to *Base Line Filtering Switch*, the ECG signal can be selected whether high pass filtered or not.

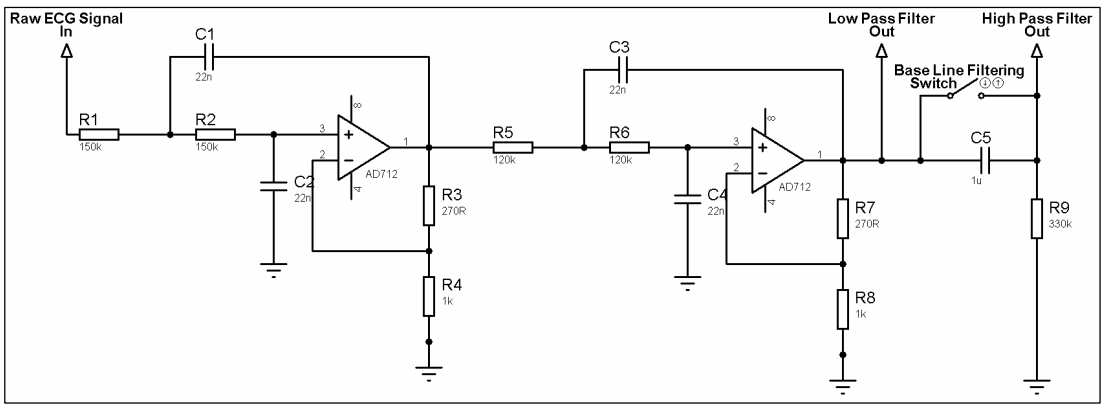


Figure 3.5 Low pass and high pass filters.

$$f_{cut}\ (3\ dB) = \frac{1}{2 \times \pi \times R1 \times C1}$$

$$f_{cut}\ (3\ dB) = \frac{1}{2 \times \pi \times 330k\Omega \times 1\mu F}$$

f_{cut} (3 dB) $\cong 0.5Hz$

3.1.6 Base Line Positioning Circuit

There is a 300 mV DC component derived from biologic noise (motion of muscle) and transferred to difference amplifier by electrodes. The amplitude of this DC component increases by motion of muscles and this causes ECG signal to oscillate between negative and positive regions.

At the ADC stage, we used the internal ADC of the 16F877A. This ADC converts voltages between 0-5 V to 8 bit digital data. Because of specifications of the microcontroller, input of the ADC must be positive value. As a result of this, when ECG signal in negative region, negative value is transferred to input of the ADC. Because of negative input value, ADC can't be able to convert to digital data correctly. So, transferred data to the receiver unit will be wrong. For all of these reasons, we designed a base line positioning circuit to keep the signal in the positive region. With this circuit, base line of the ECG signal can be shifted any desired position between -9 V and +9 V. This circuit is shown in Figure 3.6.

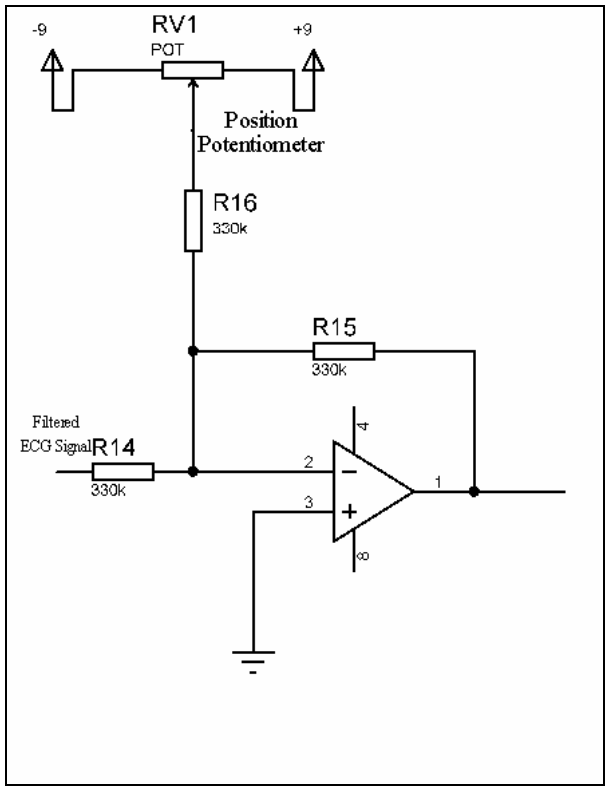


Figure 3.6 Position circuit.

3.1.7 Second Amplifier Circuit

ECG signal is amplified 100 times by AD624 in the first section of our circuit (see Figure 3.6). Therefore ECG signal which has 1 mV_{pp} amplitude, output of the AD624 is about 100 mV. But this value is very small for the ADC to convert. Because, each bits of ADC is equal 20 mV ($\frac{5000mV}{2^8}$).

It is necessary that, ECG signal is between 500 mV and 1000 mV for accurate sampling. So as to provide this, ECG signal is amplified again with an adjustable value between 5 and 10 times after filtering.

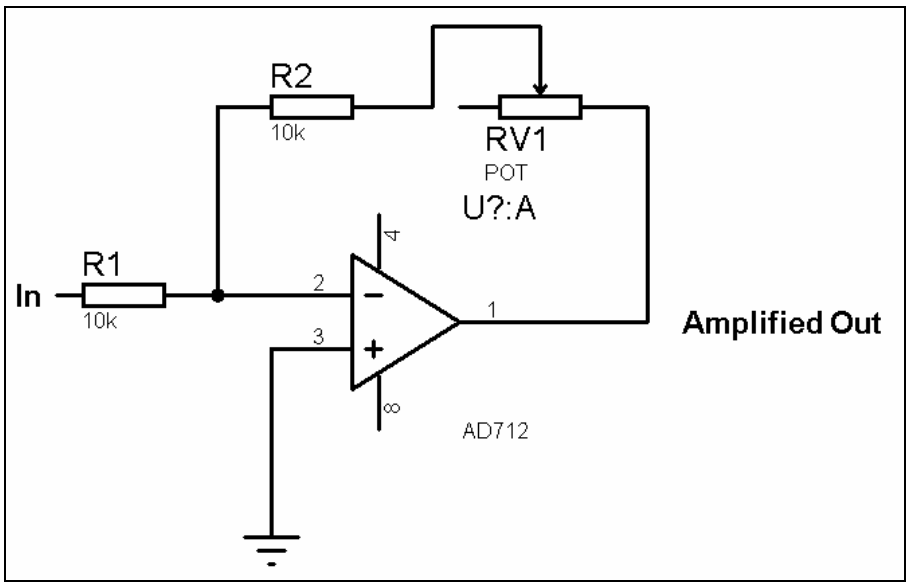


Figure 3.7 Second gain circuit.

$$V_{out} = -\left(\frac{R_2 + RV_1}{R_1}\right) \text{x } V_{in}$$

Where RV1 = 100 kΩ;

Voltage Gain (G) ≅ 1 - 11

3.1.8 Circuit Schema and PCB

Complete circuit resumed the last form after all of the components discussed had been designed and shown in the Figure 3.8. Circuit Schema and PCB was drawn by PROTEUS software belonging to Labcenter Electronics Ltd.

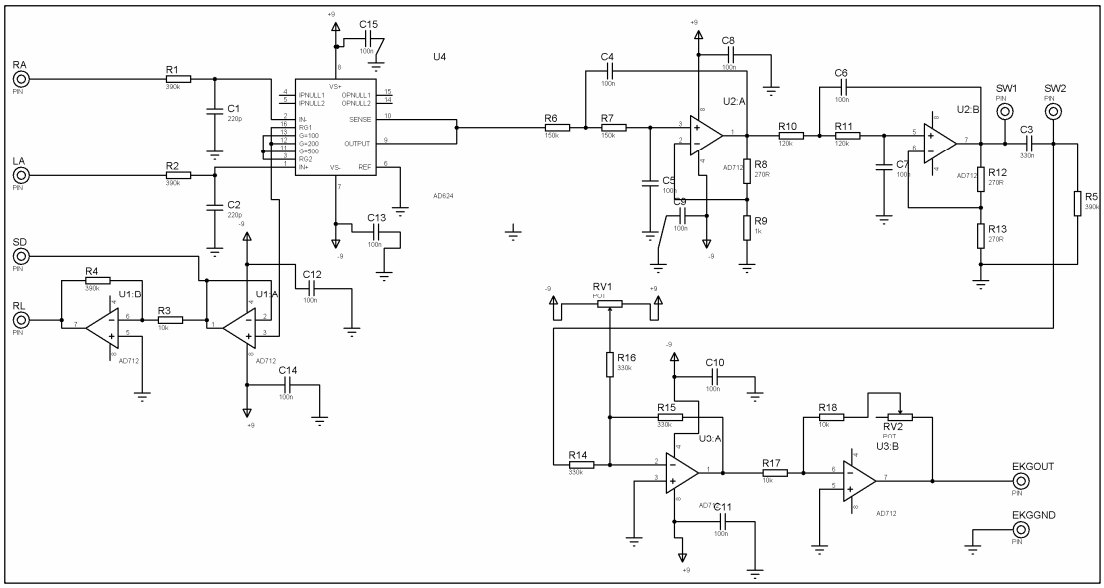


Figure 3.8 ECG acquisition circuit schema.

PCB and the material arrangement of the circuit are shown below.

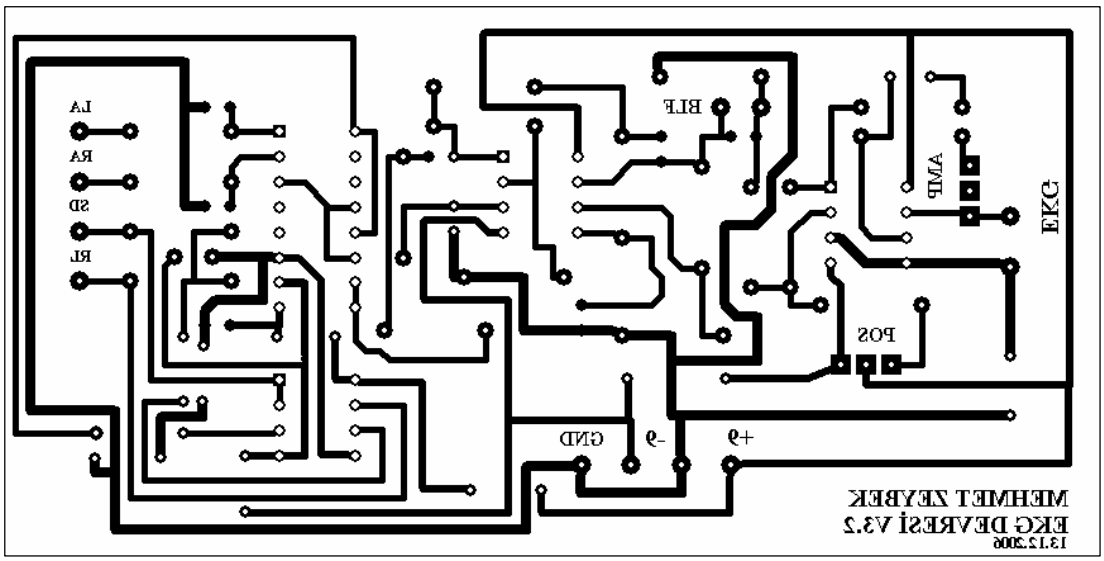


Figure 3.9 PCB of the ECG acquisition circuit.

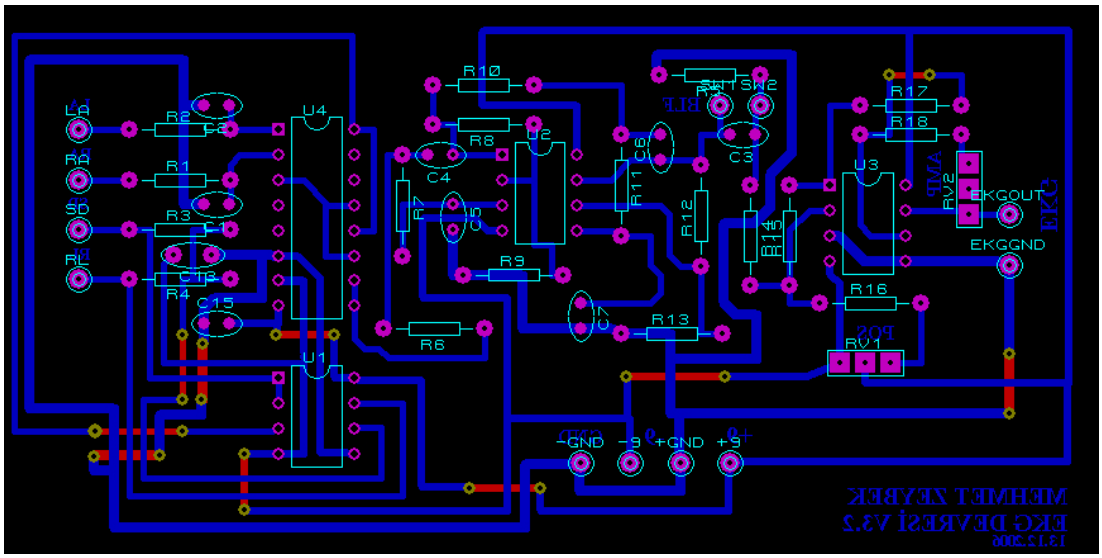


Figure 3.10 PCB and component placement of the circuit.

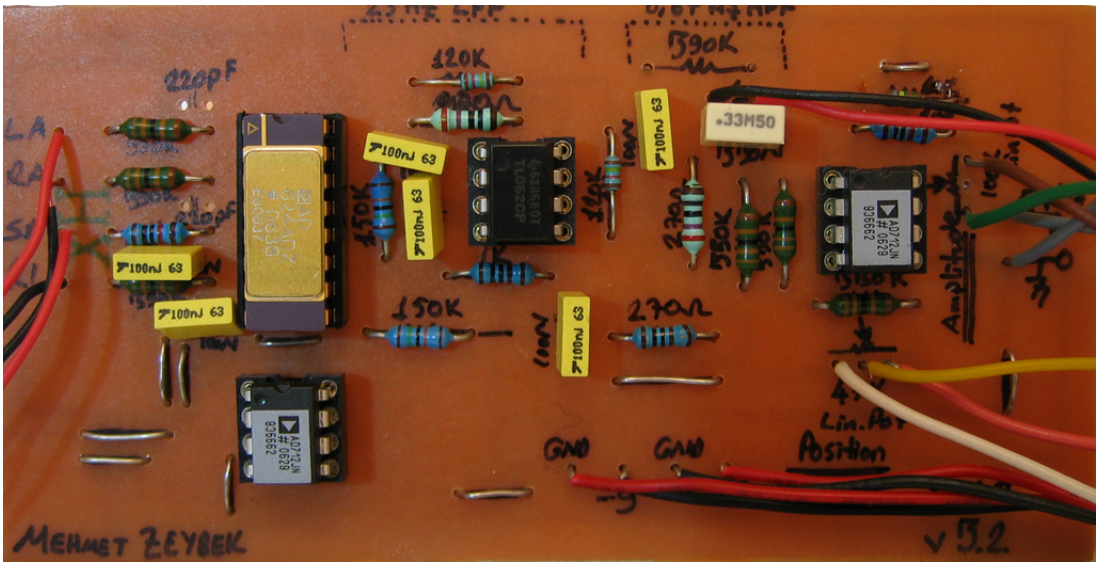


Figure 3.11 Top view of the circuit.

3.2 ADC and Transmitter Circuit

ADC must be used to convert the analog ECG signal which is obtained from the designed ECG acquisition circuit to the digital form. 16F877A microcontroller which includes in microprocessor, ADC, memory was used in our design. The specifications of 16F877A are shown below.

3.2.1 16F877A Microcontroller

PIC series microcontrollers are developed by Microchip. The meaning of PIC is, Peripheral Interface Controller, input output processor. Firstly in 1994, it was developed because of need for rapid and cheap solution to reduce load at input and output of big processor with 16- bits and 32-bits.

PIC16F877A, after PIC16F84, which is the most popular PIC microcontroller, is conspicuous for presenting new and developed possibilities to users. In PIC16F877A of which memory of programme is flash ROM, as in PIC16F84 loaded programme can be reloaded after deleting electrically.

In addition to 8192x14 words of FLASH program memory, 256 data memory bytes, and 368 bytes of user RAM, PIC16F877A also features an integrated 8 channels 10-bit analogue to digital converter. Peripherals include two 8-bit timers, one 16-bit timer, a Watchdog timer, Brown-Out-Reset (BOR), In-Circuit-Serial Programming™, RS-485 type UART for multi-drop data acquisition applications, and I2C™ or SPI™ communications capability for peripheral expansion. Precision timing interfaces are accommodated through two CCP modules and two PWM modules (Microchip, 2003).

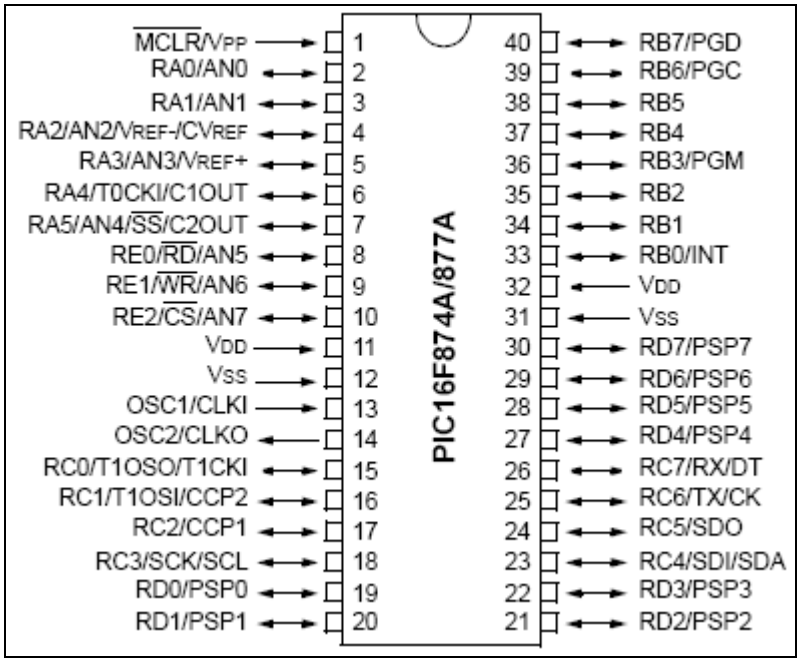


Figure 3.12 Pins of 16F877A.

Table 3.1 Key features of 16F877A

Key Features	**PIC16F877A**
Operating Frequency	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	8K
Data Memory (bytes)	368
EEPROM Data Memory (bytes)	256
Interrupts	15
I/O Ports	Ports A, B, C, D, E
Timers	3
Capture/Compare/PWM modules	2
Serial Communications	MSSP, USART
Parallel Communications	PSP
10-bit Analog-to-Digital Module	8 input channels
Analog Comparators	2
Instruction Set	35 Instructions
Packages	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

3.2.1.1 Overview of the File Registers

The data memory is partitioned into multiple banks which contain the general purpose registers and the special function registers. Bits RP1 and RP0 are the bank select bits, and they are found in the STATUS register.

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 the general purpose registers, implemented as static RAM. All implemented banks contain special function registers. Some “high use” special function registers from one bank may be mirrored in another bank for code reduction and quicker access. Also notice that there are 16 general purpose global registers, these registers can be accessed from any bank.

3.2.1.2 Special Function Registers

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM.

3.2.1.3 Overview of the ADC

At first, it appears that the PIC16F877A has 8 built-in ADCs. Figure 3.13 shows a simplified block diagram of the analogue to digital converter module, clearly there is only one 10-bit ADC which can be connected to only one of eight input pins at any one time.

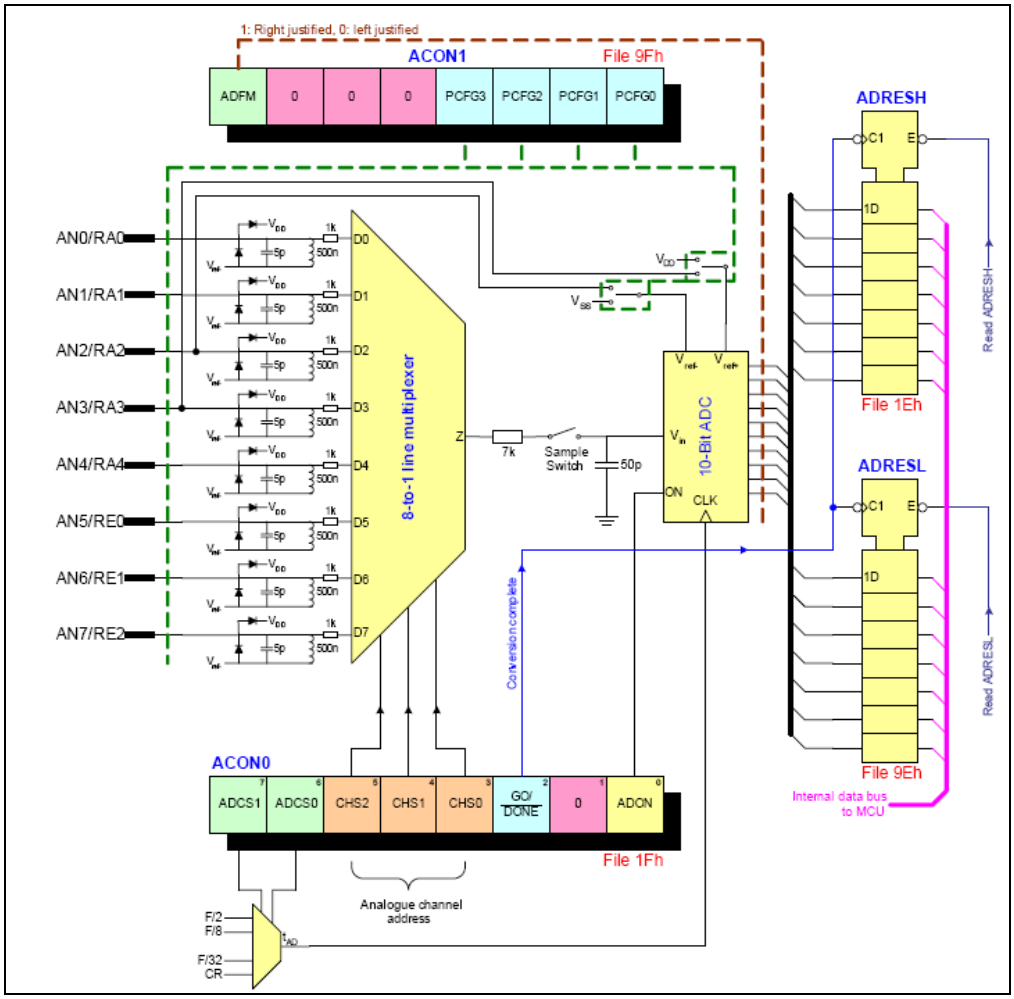


Figure 3.13 Simplified block diagram of the 16F877A ADC module.

The successive approximation technique uses a network of precision capacitors configured to allow consecutive halving of a fixed voltage VREF to be switched in to an analogue comparator, which acts as the balance scale.

Generally the network of capacitors is valued in powers of two to subdivide the analogue reference voltage (e.g. 1, 2, 4, 8, 16, etc…). This sampling acquisition process takes a finite time due to the charging time constant and is specified in the data sheet as 19.72 µS.

Obtained ECG data is transferred to pin RA0 of 16F877A. Internal ADC was configured by codes which are shown below.

DEFINE ADC_BITS 8
DEFINE ADC_SAMPLEUS 100

Thus, ECG signal is converted to digital form with minimum distortion by taking 10,000 samples per second.

Internal RC oscillator was used as clock source for ADC with the line,

DEFINE ADC_CLOCK3

written in the code.

As a result of 8-bit analog to digital conversion output signal can take a value between 0 with 255. If analog input is 0 V, output will be binary 00000000 (0). If analog input is 5 V, output will be binary 11111111 (255).

Resolution = 5000mV/256 = 19.53 mV

According to this, nearly each 20 mV will be represented with a bit.

Because of the output of the obtained ECG signal is changing between 1.5V-3.5V out put of ADC will change between

$$\frac{1500mV}{20mV} = 75 = b'01001011'$$

and

$$\frac{3500mV}{20mV} = 175 = b'10101111'$$

3.2.1.4 Overview of the Hardware USART

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. The USART can be configured for asynchronous operation (UART) for communication with a PC or synchronous operation for communicating with peripheral devices such as DAC or DAC integrated circuit.

Note bit SPEN (RCSTA, 7) and bits TRISC, 7 & 6 have to be set in order to configure pin PC6/TX/CK and RC7/RX/DT for USART operation. PicBasic Pro which we used will automatically configure these bits.

By the code written to PIC with the lines below,

DEFINE HSER_RCSTA 90H
DEFINE HSER_TXSTA 20H
DEFINE HSER_BAUD 2400

after hardware USART activates, it will communicate at 2,400 baud.

The datasheet of the 16F877A is given in Appendix C.

3.2.2 UDEA UTR-C12 U RF Transceiver

Analog ECG signal converted 8-bit digital is transmitted to RF Transceiver at 2,400 baud by using USART by 16F877A. While choosing RF transceiver, it is decided to use native production UDEA UTR-C12 U RF by taking into consideration the criteria like easy assurance and after-sale support.

Some features of UTR-C12 series which has two models, Manchester and UART, are listed below.

- It can be produce 434 MHz or 868 MHz UHF band. It is compatible with EN 300 220,

- Narrow band FSK modulation and high frequency stability,

- It is ideal for battery-powered application by low current consumption.

As seen from the data sheet, RF module must be fed with 3 V DC. Microcontroller used in circuit is fed with 5V DC. For this, it was obtained a 5 V DC voltage by adding 78C05, regulator IC, to system fed with 9V battery. After feeding 16F877A with this voltage, it was provided feeding of RF module by obtaining 3 V DC by using LM317T adjustable regulator IC. The schemas related to this circuit are shown in Figure 3.14.

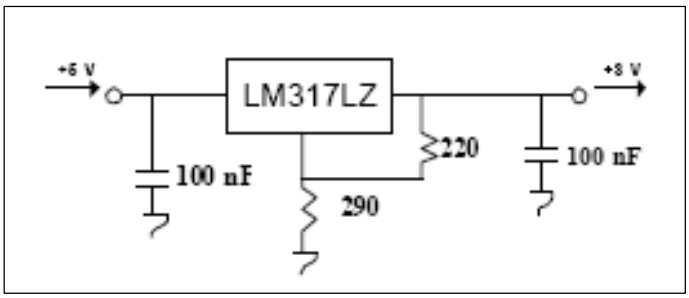


Figure 3.14 Power supply circuit for RF module.

There is a digital input-output (DIO) pin for sending and receiving data in the UART mode RF transceiver modules. The data which is wanted to send is transferred to DIO pin when the RF module is in the transmitter mode. Furthermore, DIO pin can be used as output pin for received data when the RF module is in the receiver mode.

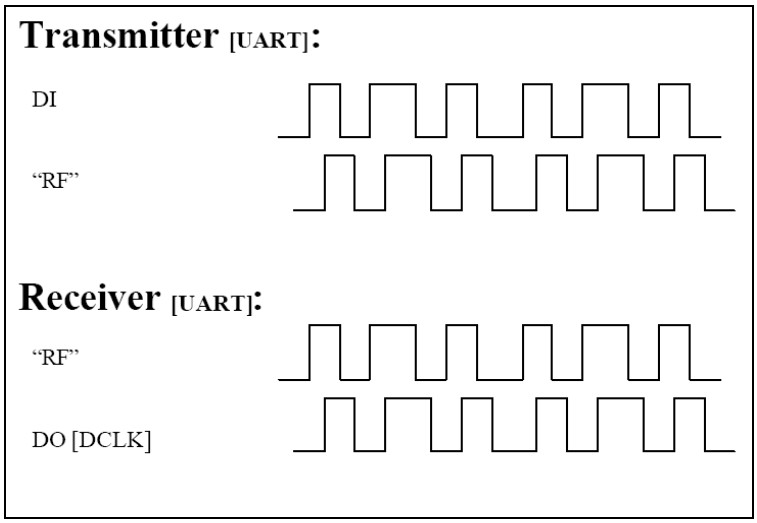


Figure 3.15 Transparent asynchronous UART mode.

RF module was used both at the transmitter and the receiver unit. It is mentioned that it is necessary to give a wake up signal for starting to operate at the transmitter mode as given in the datasheet.

For this reason a "wake up signal" is given to switch the module to transmitter mode before starting to send data. Data is transmitted after this wake up signal. Module is woken up by the command of;

SEROUT2 PORTD.0,396,[REP$AA\5, REP$00\5, REP$FF\5]

and the data is made ready to send. Firstly the signal of hexAA is sent 5 times from the transmitter port at 2,400 baud, then the signal of hex00 is sent 5 times, and module is made ready by sending finally the signal of hexFF 5 times. After this signal module is ready and is waiting to transmit data.

It sends ECG signal which is taken from channel numbered 0 of ADC to RF module at 2,400 baud via portc5, which is hardware USART port, as serial.

At this time another problem comes out. There is logic level difference between microcontroller and the RF module. Logic one level is 5V DC for microcontroller

and 3V DC for the RF module. This problem was solved through a voltage divider circuit as given Figure 3.16.

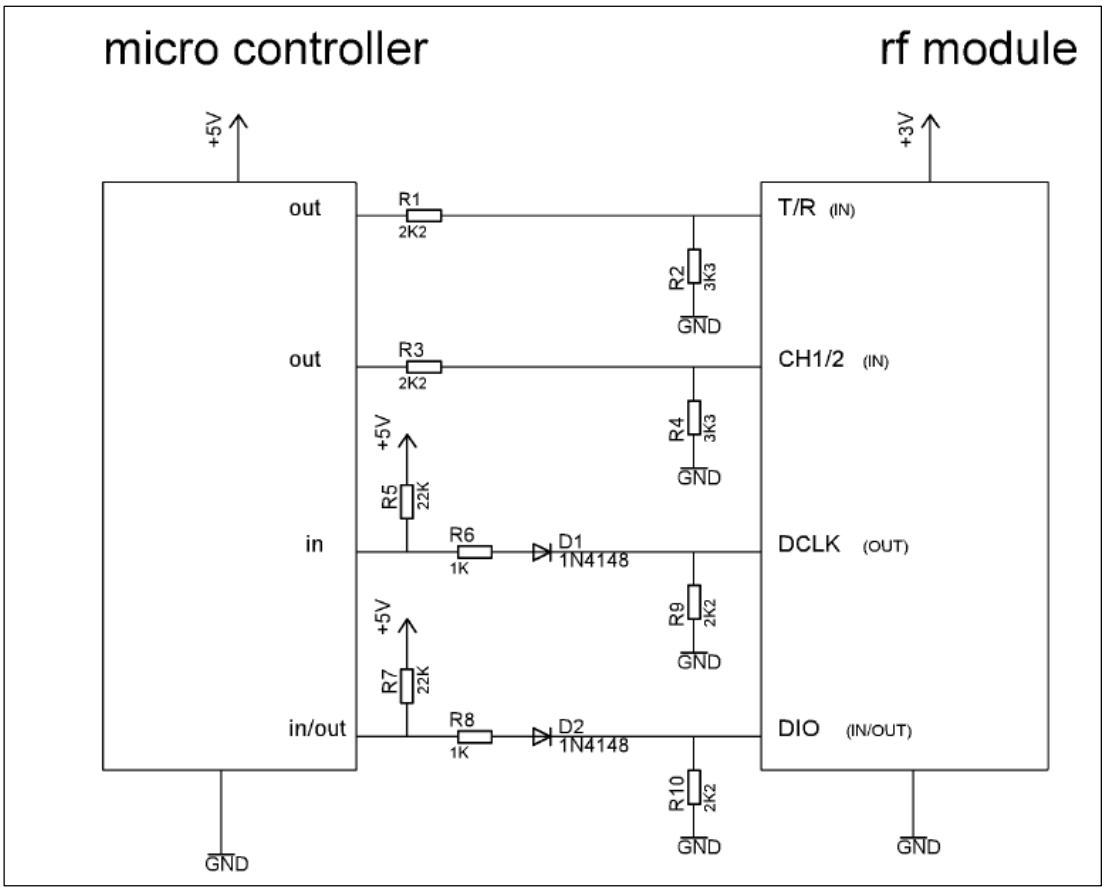


Figure 3.16 Connection between microcontroller and RF module with voltage regulation.

The microcontroller has both CMOS and TTL input pins. Minimum logic one level is 2V DC for the TTL pins and 4V DC for the CMOS pins. Because of this, received signal from the RF module is transferred to the microcontroller's RB4 pin which has TTL characteristic. Although received logic one signal from the RF module has low voltage, it is perceived as logic one by microcontroller.

The datasheet of the UTR-C12 U is given in Appendix D.

3.2.3 Circuit Schema and PCB

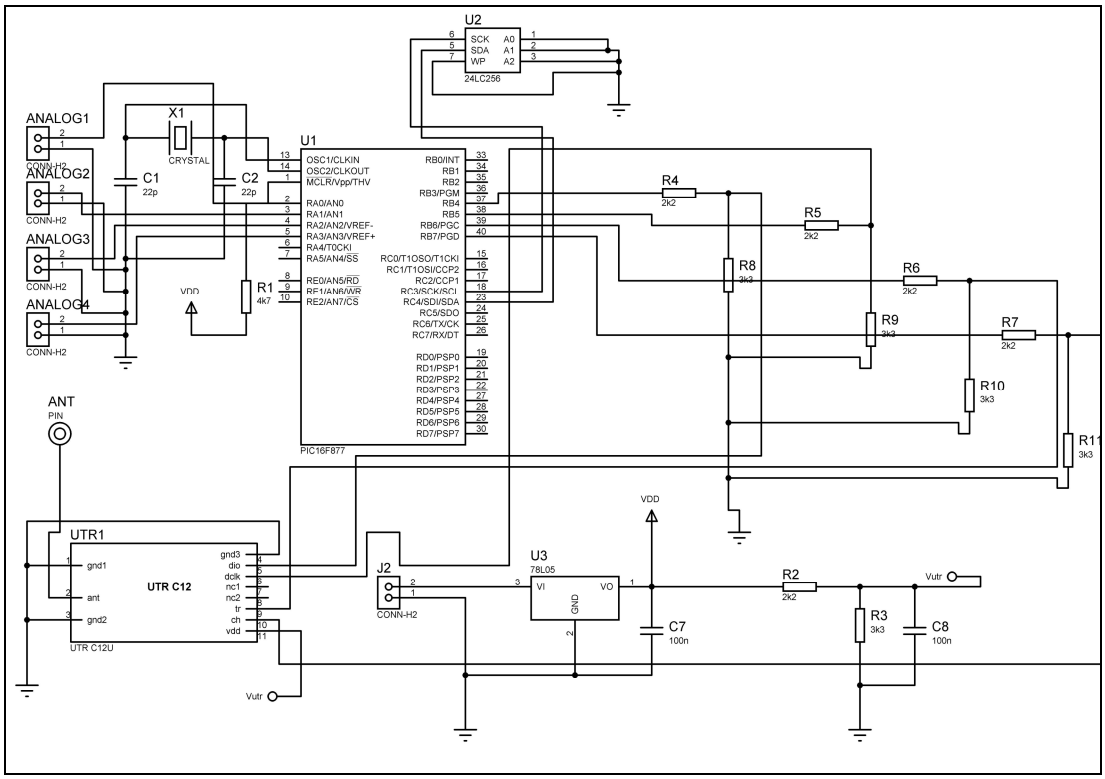


Figure 3.17 ADC and RF transmitter circuit schema.

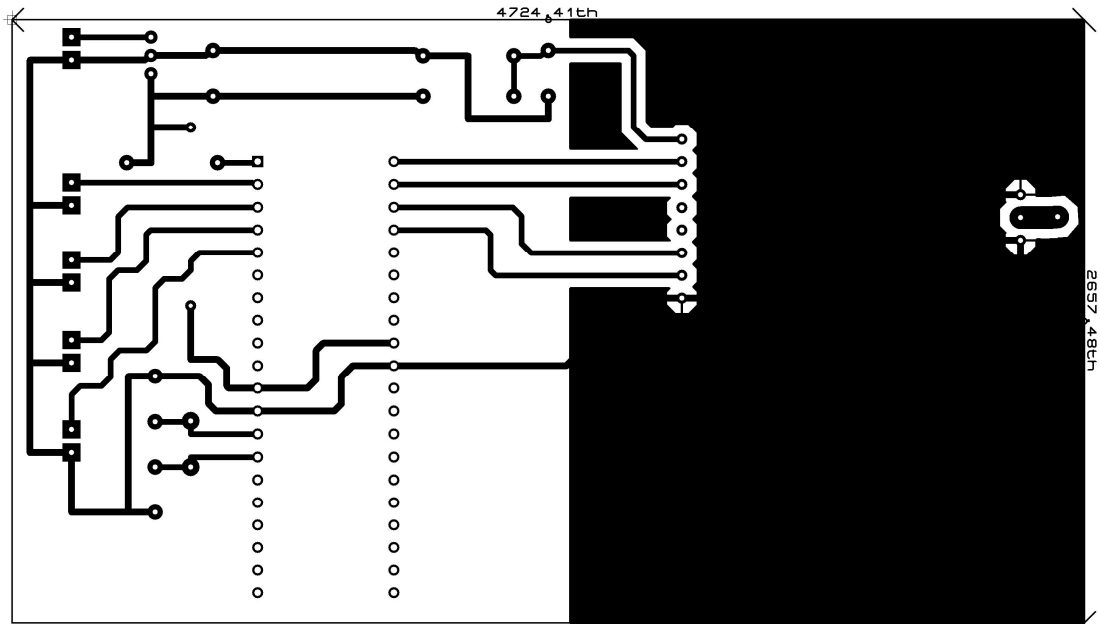


Figure 3.18 PCB of the ADC and RF transmitter circuit.

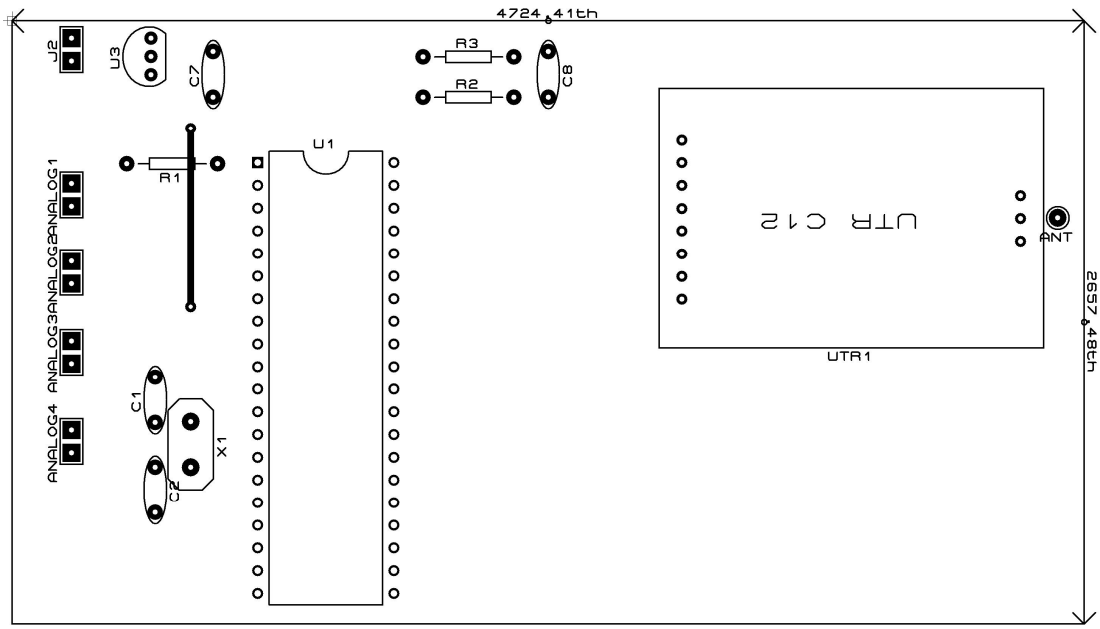


Figure 3.19 Component placement of the ADC and transmitter circuit.

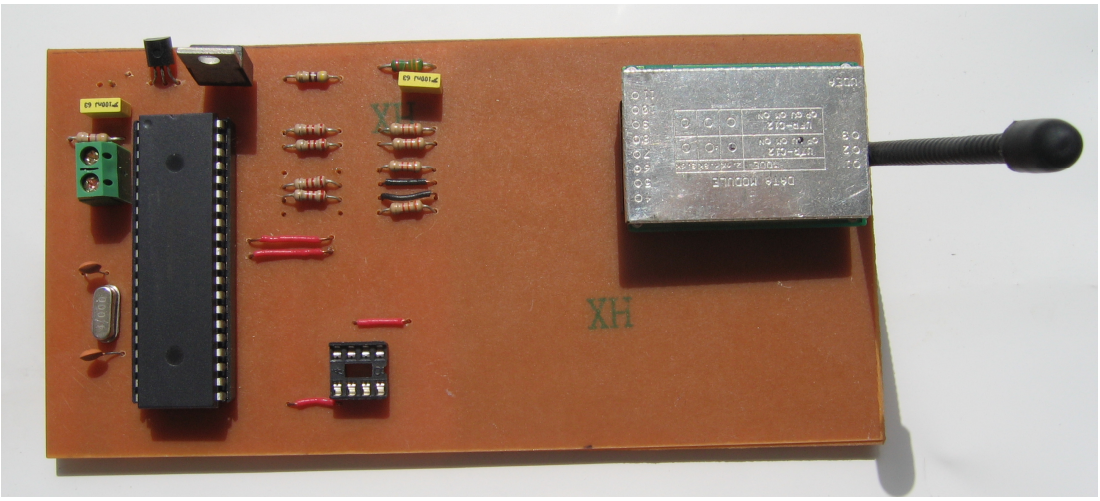


Figure 3.20 Top view of the transmitter unit.

3.2.4 Data Transmission Protocol

After converting analog ECG data to digital, it is coded before transmission by the RF module. Coding is used to prevent any disturbances from other devices using the same ISM band.

Data converted to digital, is transmitted in series each of which is 8-bit to the RF module by the microcontroller. During this transmission, a character series chosen for each patient by microcontroller accompanies this data.

```
ADCIN 0, AL
        BAK:  IF ADCON0.2=1 THEN BAK
SEROUT2 PORTD.0, 396, [("M"), AL]
```

Signal converted to digital by ADC from channel 0, after registered to variable "**AL"**, is sent serially from port RD0 to RF module with character "**M**" at 2,400 baud. The RF module transmits this obtained data serially to the receiver unit which is connected to the computer.

Character "**M**" sent with data, is controlled within the receiving data by the code that will be written to microcontroller. If there is "**M**" within the receiving data series, then it is accepted as "significant data". So, the data which is sent by the other devices in the same environment won't cause chaos and mistake.

The number of transmitter connected to the patients can be increased by this protocol. By courtesy of this, receiver will be able to determine that which transmitter sends which data by using different "character series" for each patient. This control mechanism in receiver code is explained as detailed in Chapter 3.3.3.1.

3.3 RF Receiver and RS232 Circuits

ECG data taken from patient after converted to digital form by ADC is transmitted at 433 MHz by RF module. At the receiver side also UTR-C12 U RF transceiver module and 16F877A microcontroller were used. The receiver side which is like that transmitter side as concept is designed to communicate with PC via serial port by RS232 protocol. For this, again hardware USART interface of 16F877A and MAX232 buffer IC were used.

3.3.1 RS232 Serial Interface

RS232 is simple, universal, well understood and supported, but it has some serious shortcomings as a data interface. Its origins predate modern computers and it contains many features that are not relevant to the modern user. It can control very old primitive modems and has many control signals to do this in hardware, but often it is used without these old control and status lines. Its major feature is that it does not require the transmission of a clock, the reception of a 'start bit' is enough to cause the receiver to time all its actions from this one edge. This is called asynchronous transmission.

RS232 allows a 5% difference in transmitted timings and receiver chip timings. This is important if using a PIC as the datasheet specifies the error of the baud rate generator at certain baud rates (the higher the baud rate, the higher the % error), as long as this error is less than 5% the RS232 standard is capable of coping.

Electronic data communications between elements will generally fall into two broad categories; single-ended and differential. RS232 (single-ended) was introduced in 1962, and despite rumours for its early demise, has remained widely used.

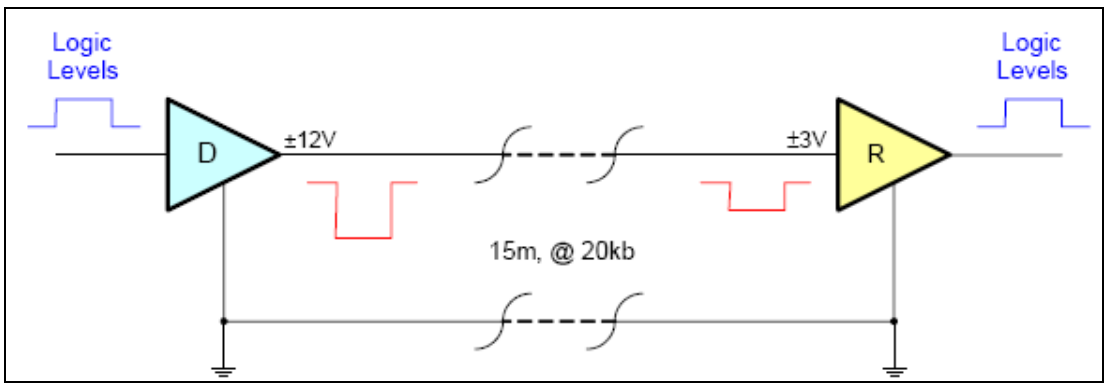


Figure 3.21 Illustration of RS232, one driver and one receiver.

Both RS232 and RS423 are unbalanced (or single-ended) standards, where the receiver measures the potential between signal line and ground reference. Even though the transmitter and receiver grounds are usually connected though the transmission line return, the impedance over a long distance may support a

significant difference in the two ground potentials, which will degrade noise immunity. Furthermore, any noise induced from the outside will affect signal lines differently from the ground return due to their dissimilar electrical characteristics – hence the name unbalanced (Katzen, 2000).

RS232 data is bi-polar, e.g. a +3 to +15 volt indicates an SPACE (ON) while a -3 to -15 volt indicates an MARK (OFF). Modern computer equipment ignores the negative level and accepts a zero voltage level as the MARK (OFF) state. This means circuits powered by 5 VDC are capable of driving RS232 circuits directly; however, the overall range that the RS232 signal may be transmitted / received is dramatically reduced.

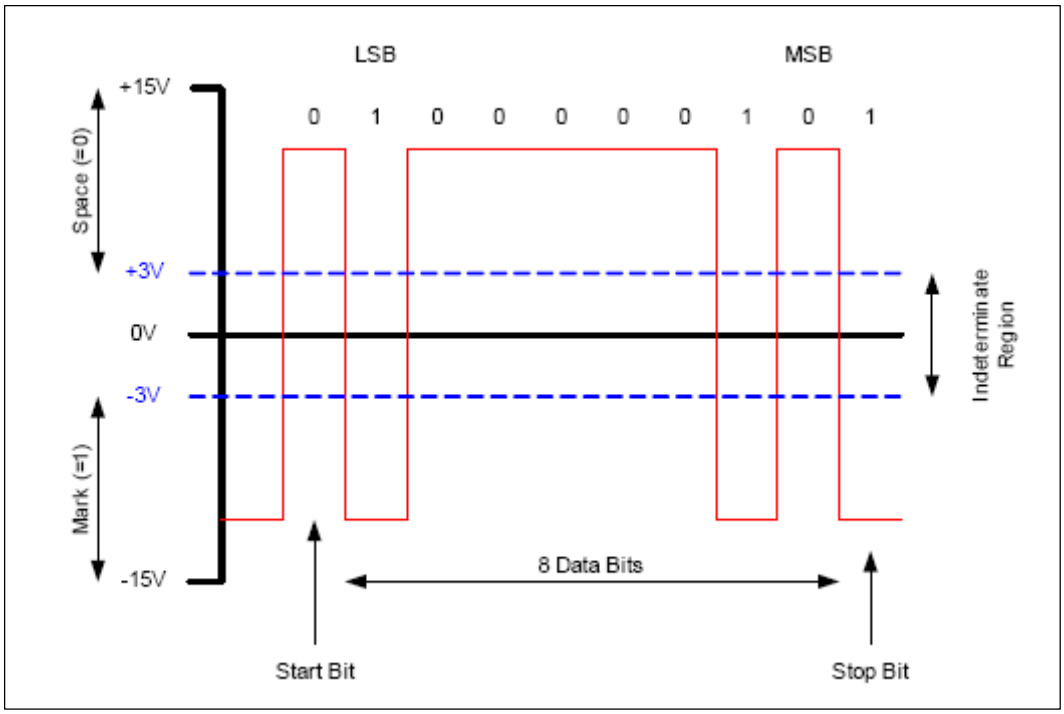


Figure 3.22 Illustration of how data is transmitted over RS232.

The output signal level usually swings between +15 V and -15 V. The 'dead area' between +3 V and -3 V is designed to give some noise immunity. This dead area can vary for various RS232 like definitions, for example the definition for V.10 has a noise margin from +0.3 V to -0.3 V. Many receivers designed for RS232 are sensitive to differentials of 1 V or less.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Receive Data	7	Request to Send
3	Transmit Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Figure 3.23 9-Pin RS232 D- connector, and pin description.

Typical line drivers / receivers chips for RS232 are the Maxim MAX232 or MAX233 chips the original specification states that RS232 should drive 15.24 meters (50 feet), but modern line driver/receivers can manage much better than this.

3.3.1.1 MAX232

The MAX232 (from family of line drivers / receivers) is intended for all EIA/TIA-232E and V.28/ V.24 communications interfaces, particularly applications where ±12 V is not available. MAX232 is especially useful in battery-powered systems, since its low power shutdown mode reduces power dissipation to less than 5µW.

3.3.2 Receiver Unit Circuit Schema and PCB

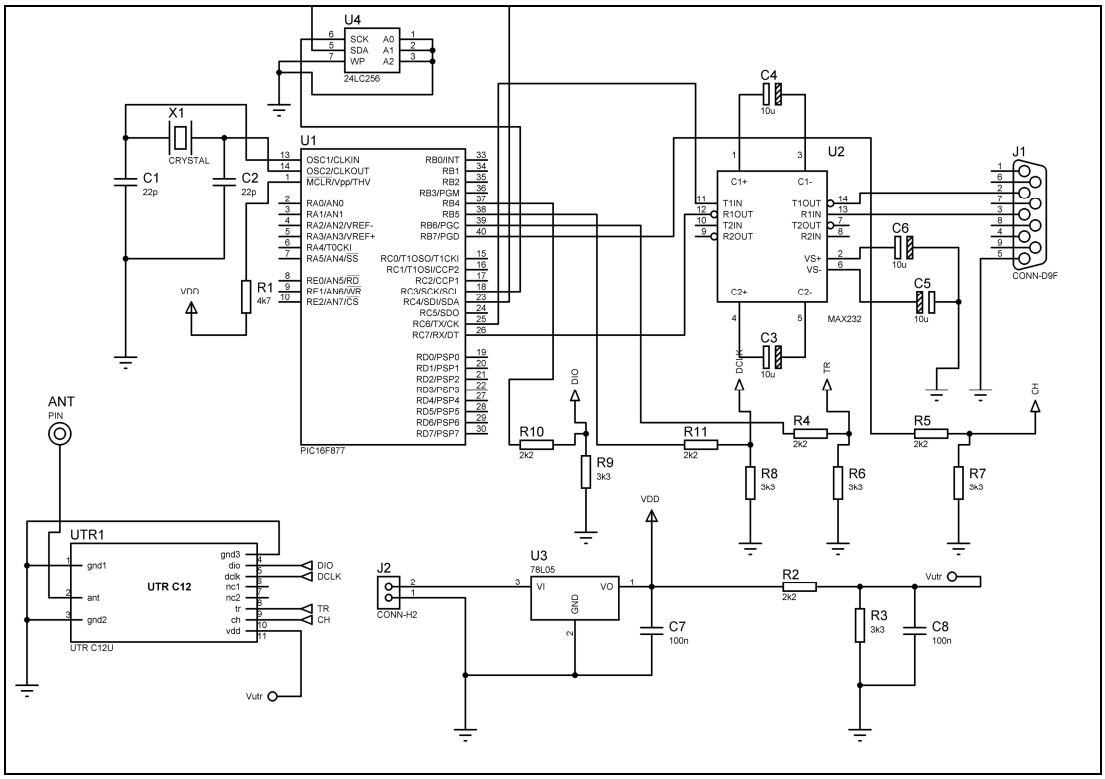


Figure 3.24 RF receiver circuit schema.

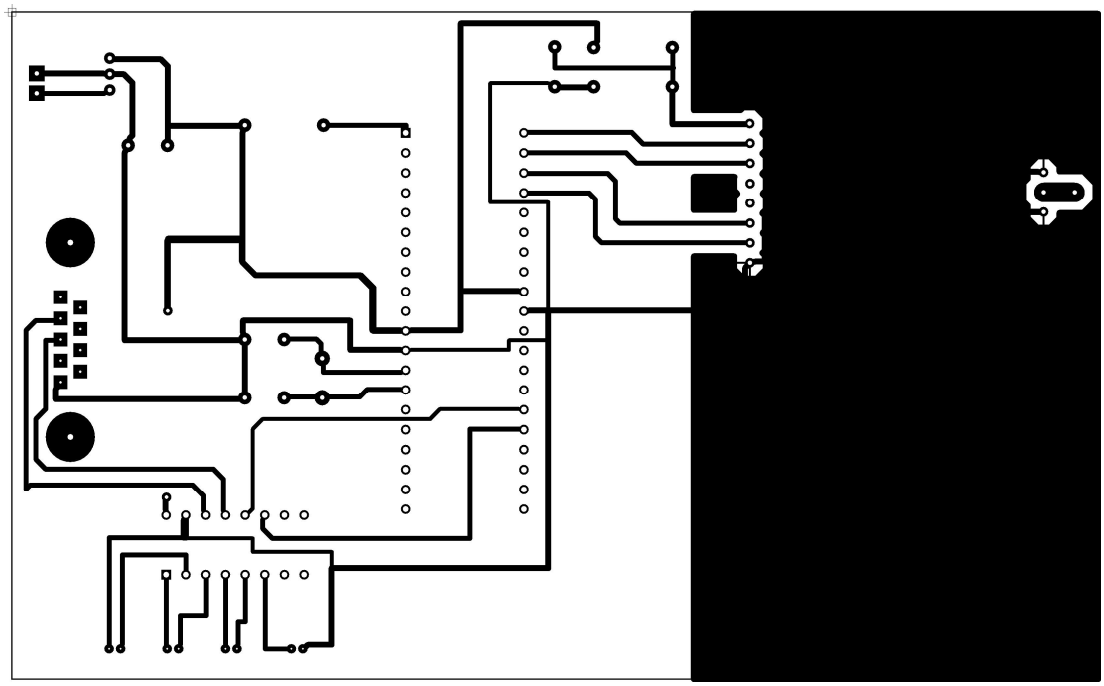


Figure 3.25 PCB of the receiver unit.

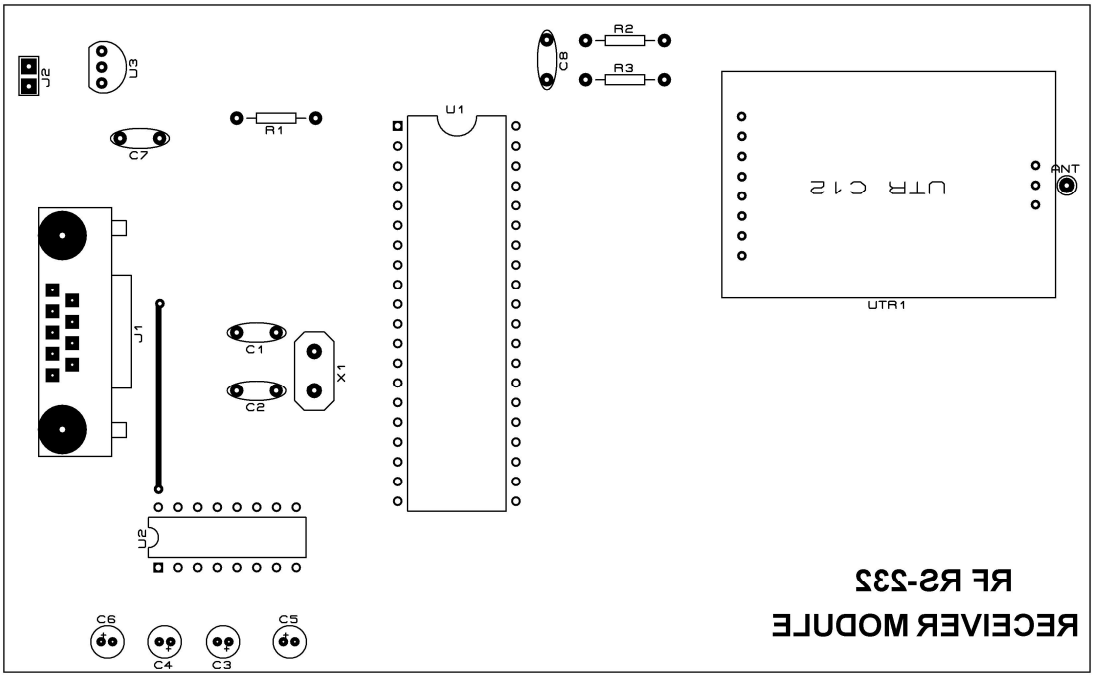


Figure 3.26 Component placement of the receiver circuit.

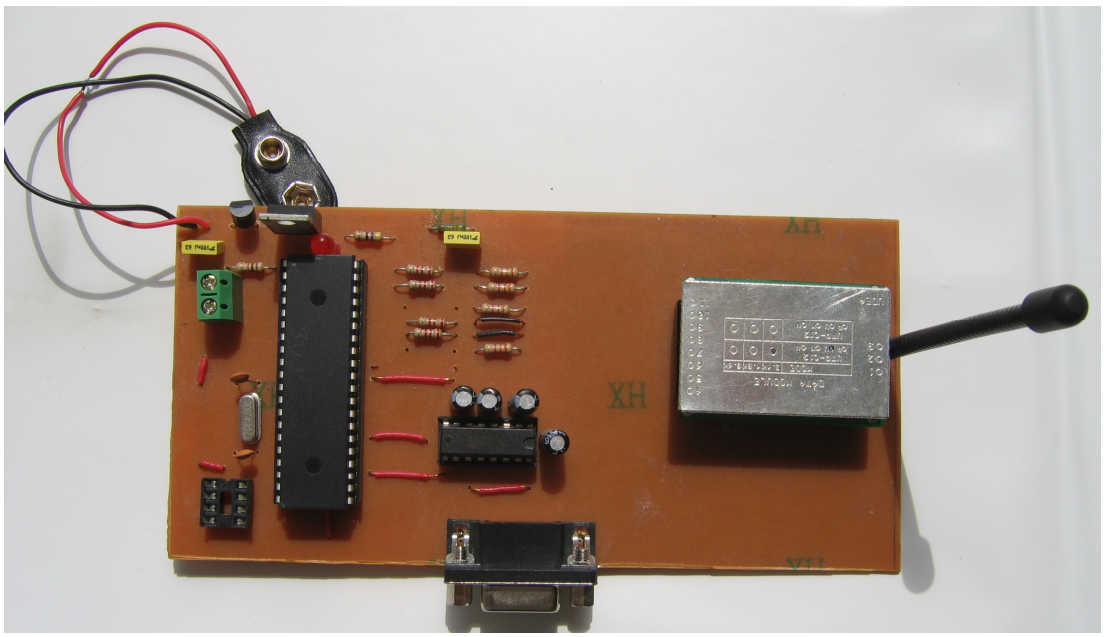


Figure 3.27 Top view of the receiver circuit.

3.3.3 Receiving RF Data and Serial Communication with PC

Data received serially at 433 MHz by UTR-C12 U RF transceiver is processed by giving to end of 16F877A by microcontroller. And then it is given to serial port of PC via MAX232 IC by hardware USART.

3.3.3.1 RF Data Receive Protocol

Data coming from RF module is transmitted to RB4 pin of the microcontroller. Data taken from RB4 while controlling is recorded to register to control for protocol explained in Chapter 3.2.4.

SERIN2 PORTB.4,396,[WAIT("M"),AL1]

Character "**M**" is searched in the data coming from port RB4 by the line in the code run on microcontroller above. If character is found, coming data after this is registered variable as called "**AL1"**. If character "**M**" isn't found, it is waited until it is found.

Because of "**M**" and "**m**" are perceived as different characters, it is necessary to be careful at this point.

So, both the effect of other 433 MHz ISM band devices is eliminated and ECG data of more than one patient can be taken by the same receiver and transferred to PC.

The codes of the microcontrollers are given in the Appendix E.

3.4 ECG Drawing Software

3.4.1 About the Software

ECG signal obtained from the patient is transmitted wireless and then it is transferred to serial port of the PC according to RS232 protocol by the receiver unit.

It is necessary that data received on serial port are collected by a software run on the PC. For this reason a software named "**EKG Data Profiler"** has been developed by using Microsoft Visual Studio 2005 on the .Net platform.

There are two parts of the software. First one is called "port reader". It "listens" PC's serial port to read coming data from the receiver unit. Second one is the main part of the program. It draws ECG data come from serial port graphically, and does other things such as patient recording, ECG data saving etc.

3.4.2. Serial Port Reader

Serial port reader written with standard C++ collects the data come from serial port according to RS232 protocol, and sends to main program. Its name is "commservice".

While the code was writing, data transmit parameters of the receiver unit were kept in the code. For this reason, the communication parameters of the software are 2,400 baud, 8-bit, non parity, and 1 stop bit (8N1). We can change and determine these parameters by using "options" menu in the main program. Determined parameters are saved in "config" file automatically by the software. When the serial port reader program starts running, it is reading datas on the serial port according to these parameters which are in the "config" file.

```
fwrite(lpBuf,1, *dwRead, stdout);
fflush(stdout);
```

with the codes shown above, serial port reader program transfers the data received form the serial port to main program by using “standard output” command (stdout).

It is necessary that port reader program has to give a command from the main program to read serial port. Port reader reads the port as long as the main program send to it value of “**1**”.

There isn’t any handshaking algorithm for the communication between the receiver unit and PC. It is just controlled that whether serial port was opened or not, whether there is a data on the serial port buffer or not etc. In this respect, that the truth of the datas are checked neither by the module nor by the PC.

The software just reads the data for now. However, if some improvements are appended to the code, the software can be sent data to receiver unit too.

3.4.3 Main Program

Microsoft Visual Studio 2005 has been used to ECG signal as a graphic on the PC. This way, the software looks more visual, and also code has been easy to develop. MSDN library of the Microsoft has been playing the most important role in the code used in this study.

Main screen of the software is shown in Figure 3.28.

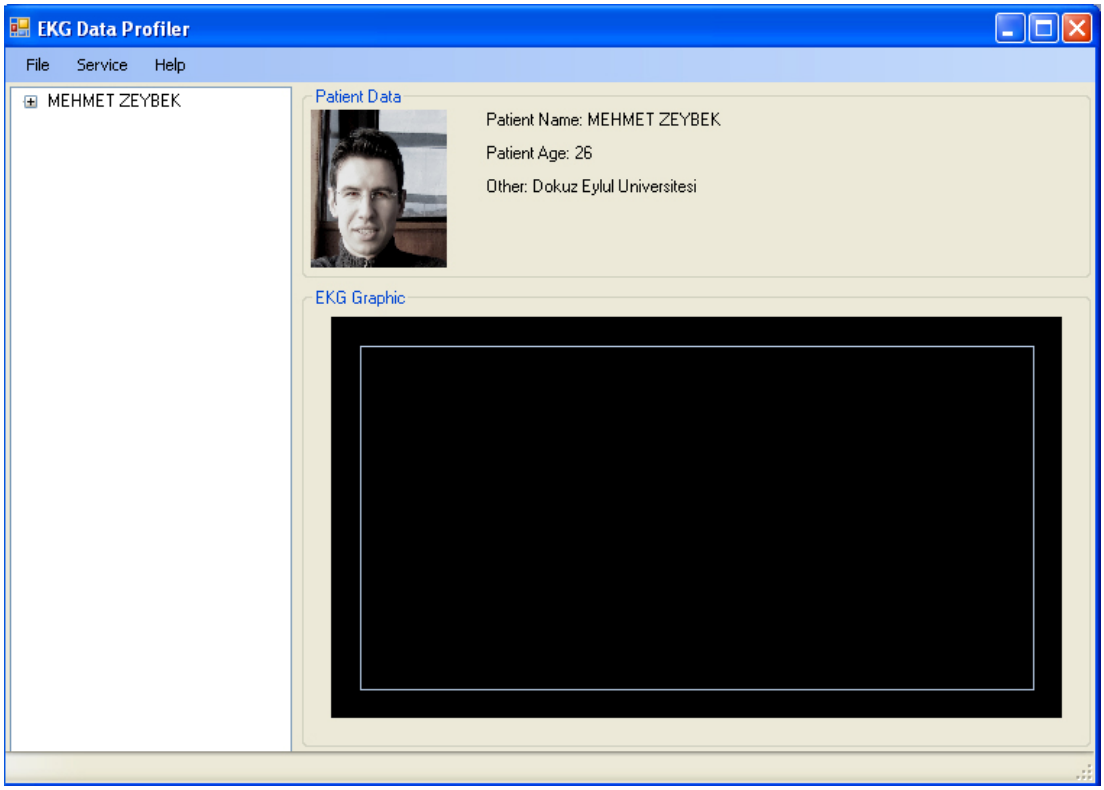


Figure 3.28 Main screen of the software.

Main program is separated into 3 parts;

- Part for patient informations (top right side)
- Part for patient data lists (left side)
- Part for real-time & past time ECG signal (bottom right side)

3.4.3.1 Configuring the Communication Parameters

Before reading the data from the serial port, both two devices have to speak the same language. For this, some configurations must be done.

First, some parameters of RS232 protocol are determined such as com port number, baud rate, parity, and stop bit. For this reason, “Service Options” section of the “Options” sub-menu which is under the “Service” main menu is used.

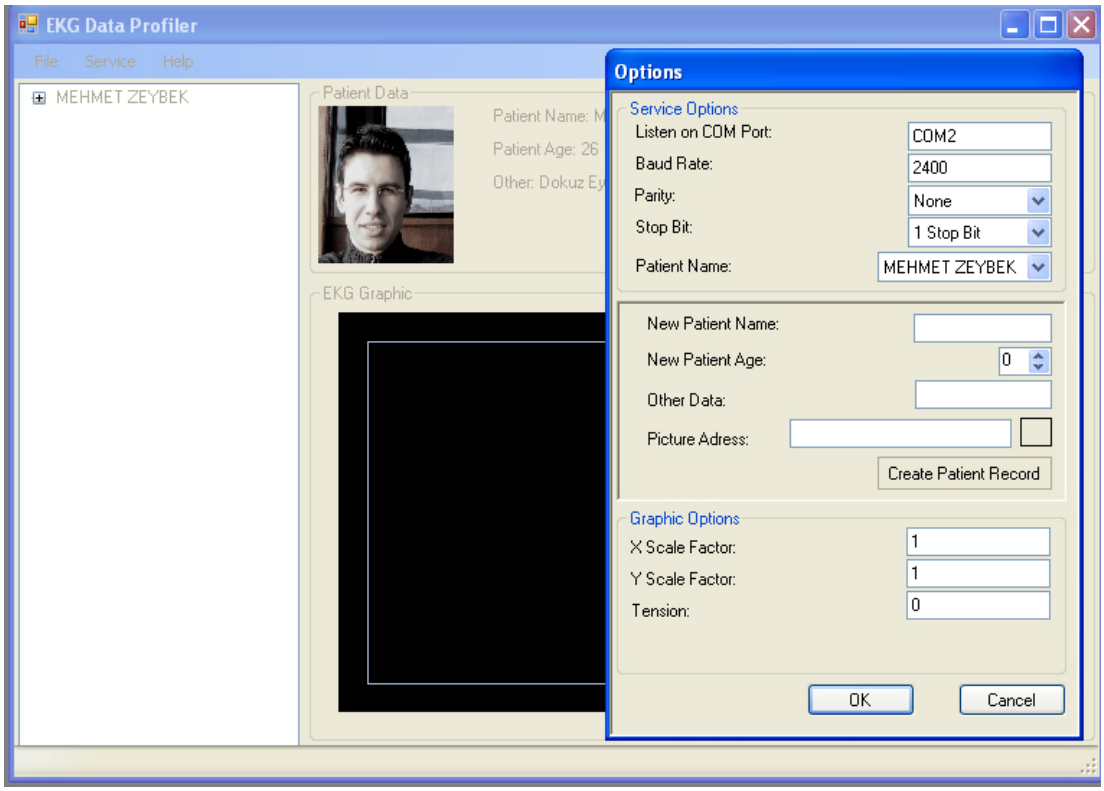


Figure 3.29 Communication parameters menu screen.

Determined parameters chosen in this sub-menu are saved in a file which is called "config".

3.4.3.2 New Patient Registering

For new patient registering "Create Patient Record" section of the "Options" sub-menu which is under the "Service" main menu is used.

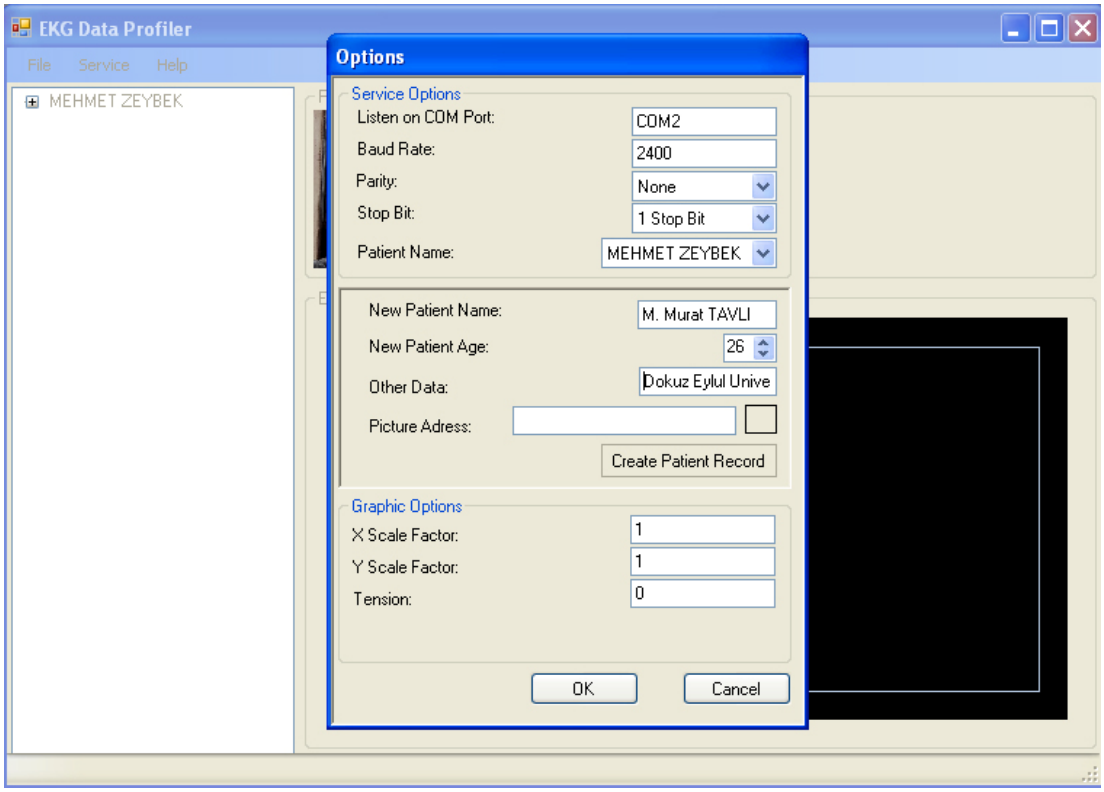


Figure 3.30 Patient registering menu screen.

We can describe some information of the patients such as name, age, etc. in here.

A "patient_name.xml" file is created for every registered patient and whole patient informations are kept in this file.

3.4.3.3 Capturing ECG Data

We are ready to capture ECG data after the serial port parameters configured and related patient registered. To capture ECG data, "Start Service" command button which is under the "Service" menu is clicked. By this command, value of "**1**" is sent to port reader program (commservice) mentioned before. As soon as this command is received, "commservice" starts reading the serial port in accordance with the parameters in the "config" file.

For drawing the ECG data received from serial port in main screen of the software one more procedure is necessary. By using "Switch To Live Data" command, which

is under the “Service” menu again, ECG data obtained from the patient starts real-time scrolling right-bottom side on the main screen. A screen image of this is shown in Figure 3.31.

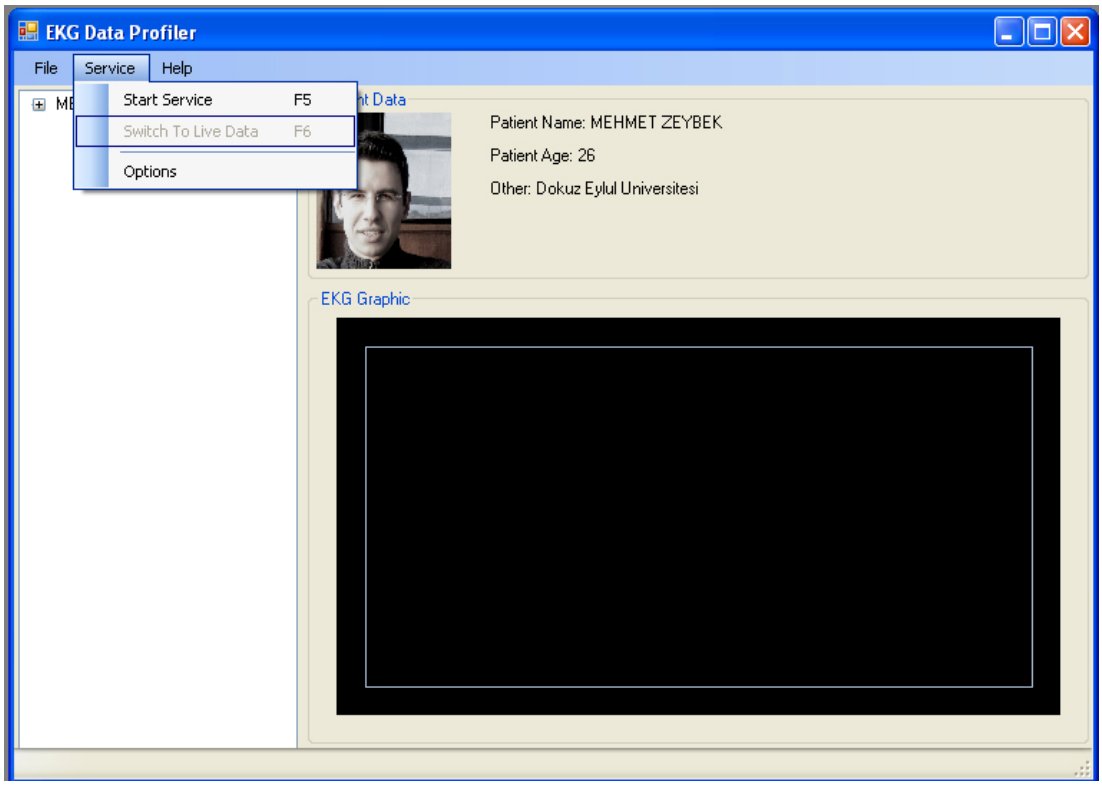


Figure 3.31 ECG capturing menu screen.

While the old parts of the graphic are going to left, new datas are being rendered in the right side of the graphic screen. If it is desired, old data can be shown by using left and right arrow buttons of keyboard.

Data received during this procedure are saved in a file. It is recorded as “patient_name.ECG”. Name, information, and ECG data of the patients are stored in this file with date information.

3.4.3.4 Saving ECG Data

After drawing the patient's ECG, it is possible to save ECG data by using "Export ECG Data" sub-menu under the "File" main menu. This is shown in Figure 3.32.

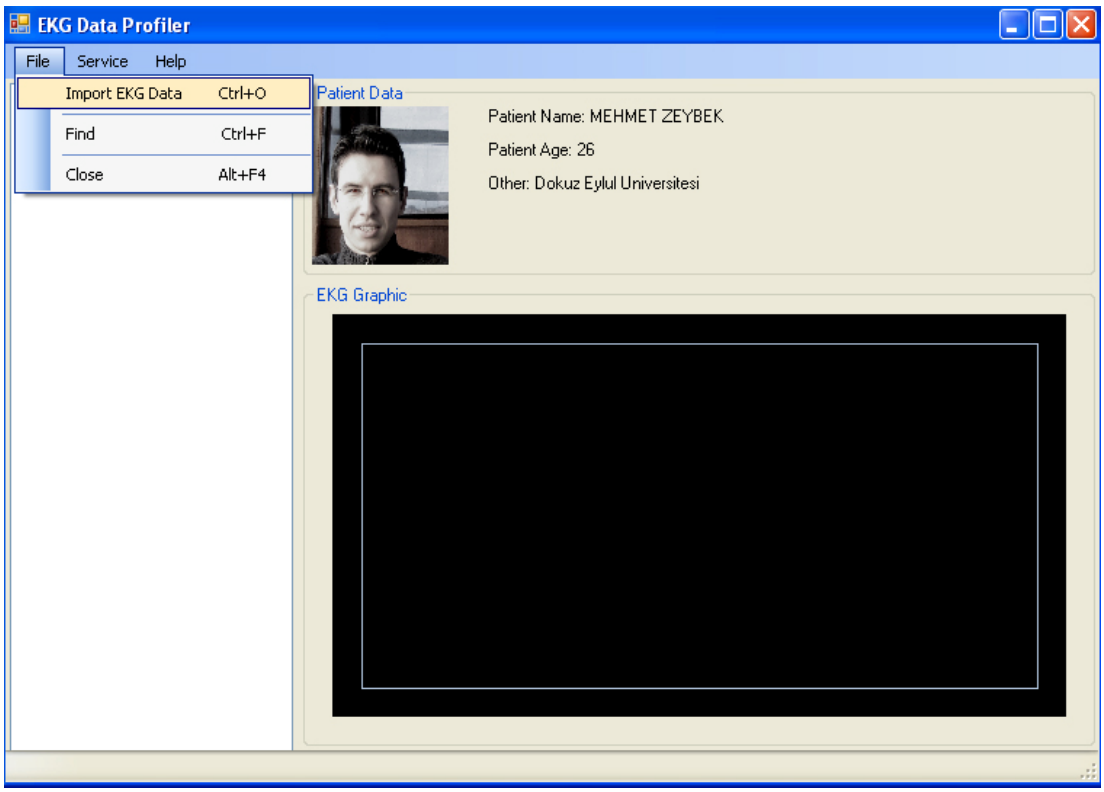


Figure 3.32 ECG data saving menu screen.

3.4.3.5 Drawing the Old Records

A list box takes up a position on the left side of the software screen. There are patients' names in this list box. By double clicking over any patient name previous ECG recordings of the relevant patient can be seen in that list box. In Figure 3.33 an image of this procedure can be shown.

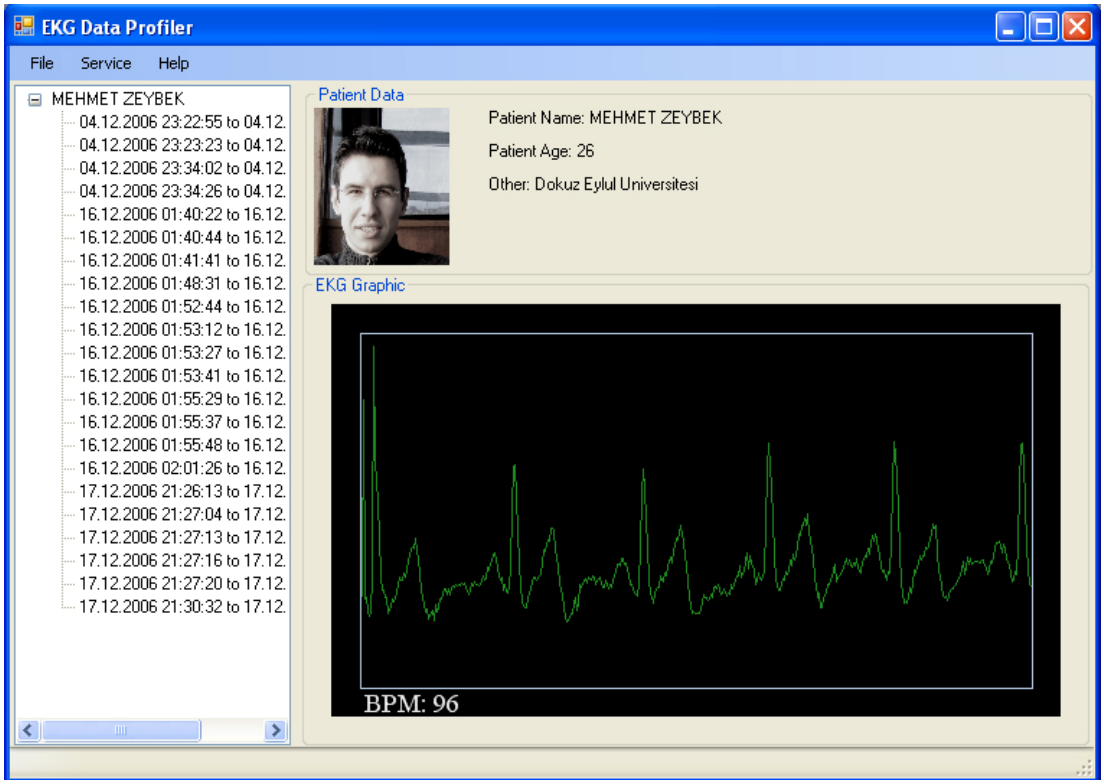


Figure 3.33 List box of the old ECG records screen.

3.4.3.6 Searching the Old Records

We mentioned before that patients' records are stored in a "patient_name.ECG". To search for any patient's record we can use "Find" sub-menu which is under the "File" main menu.

When the “Find” menu opens, we see screen image which is shown below in Figure 3.34.

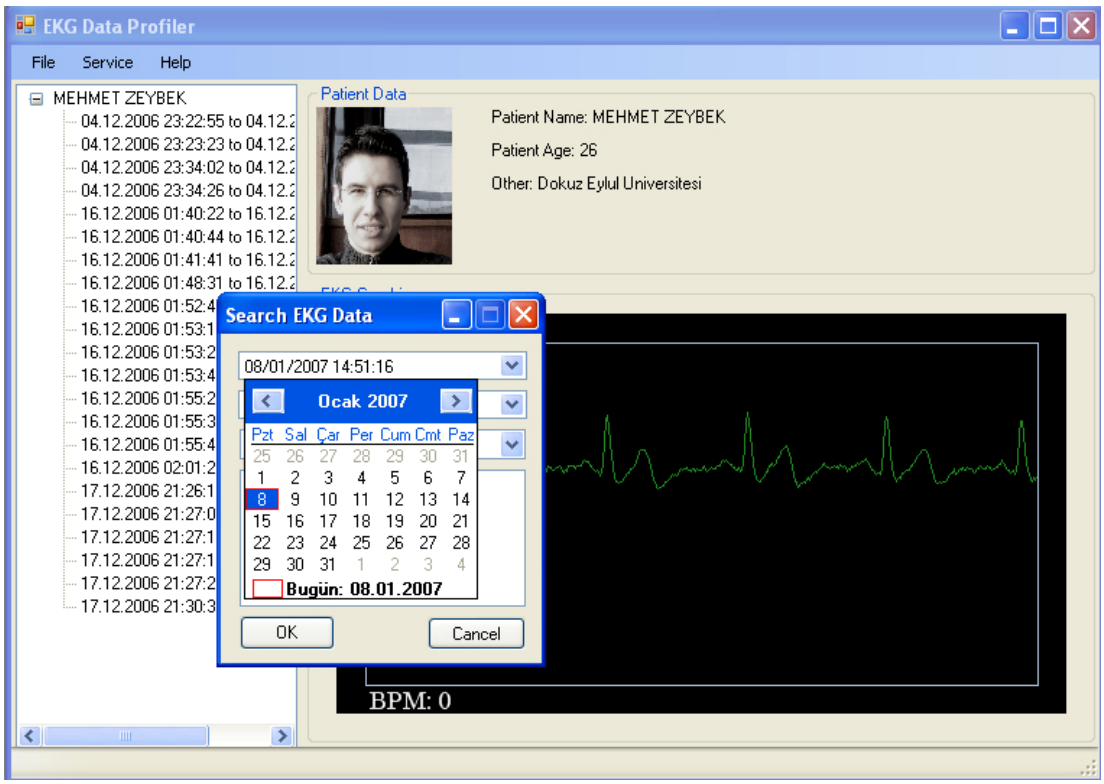


Figure 3.34 Find menu screen.

Here, we can search in the records with patients’ names. Searching finds and displays us the records of chosen patient between determined date ranges. By choosing a record wished, that record renders on the screen.

The codes of the ECG Drawing Software are given in the Appendix F.

CHAPTER FOUR

RESULTS

4.1 ECG Acquisition Circuit

Firstly, we tested the ECG Acquisition circuit which has 2 electrodes and right leg driver in a laboratory.

The measure of the electrical potential of the heart from the skin appeared as the biggest problem. Both 50 Hz noise and the other electromagnetic noises from the devices which were in the laboratory caused big distortion on the ECG signal which was carried by the shielded cable from the electrodes. You can see the distortioned signal on the inputs of the AD624 in Figure 4.1.

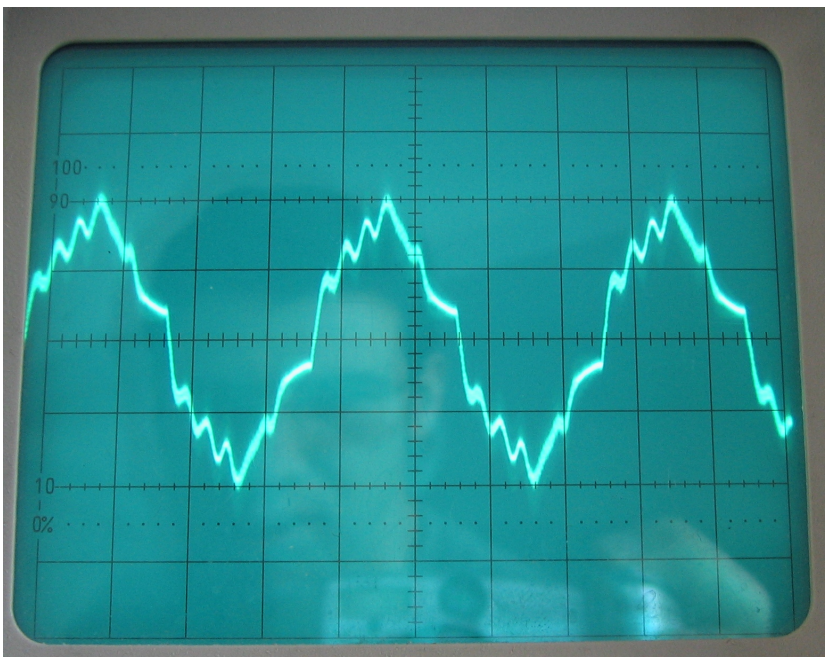


Figure 4.1 50 Hz noise on the inputs of the AD624.

Low amplitude ECG signal obtained from the electrodes is only 100 times amplified by the instrumentation amplifier. The reason of that is the voltage level of the surrounding noise signals is much more than the ECG signal's. If we amplify 500 times, pre-opamp of the instrumentation amplifier can saturate.

Because of the fact that our ECG signal voltage on the output of the AD624 was approximately 100 mV (peak-to-peak), output signal has still some noise (distortion) especially the 50 Hz noise. You can see output signal of the AD624 in Figure 4.2.

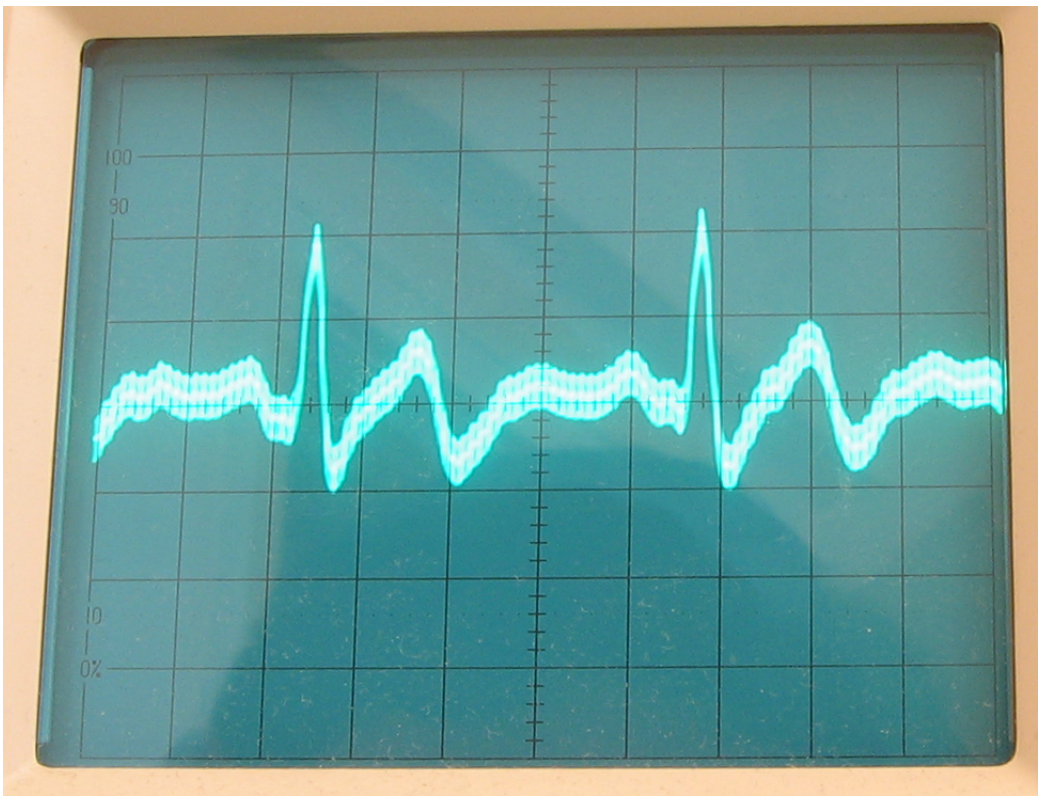


Figure 4.2 Output signal of the AD624.

Then, output signal of the AD624 was passed from low pass filter and high pass filter respectively. In this way, output signal of the filters has only frequency component between 0.5 Hz and 30 Hz. All the other frequency components are removed by the LPF and HPF. In addition to this, our signal was smoother and has less noise than the previous one. On the other hand, amplitude of the filtered signal was smaller than the previous one.



Figure 4.3 High and low pass filtered signal.

After the last amplifier and position control blocks, our ECG signal amplitude was about 1.5 V (peak-to-peak). In this section, amplitude of the signal amplified by between 1 and 10 times. Thanks to position control circuit, we could adjust the base line of the ECG signal. The acquired ECG signal looks like in Figure 4.4.

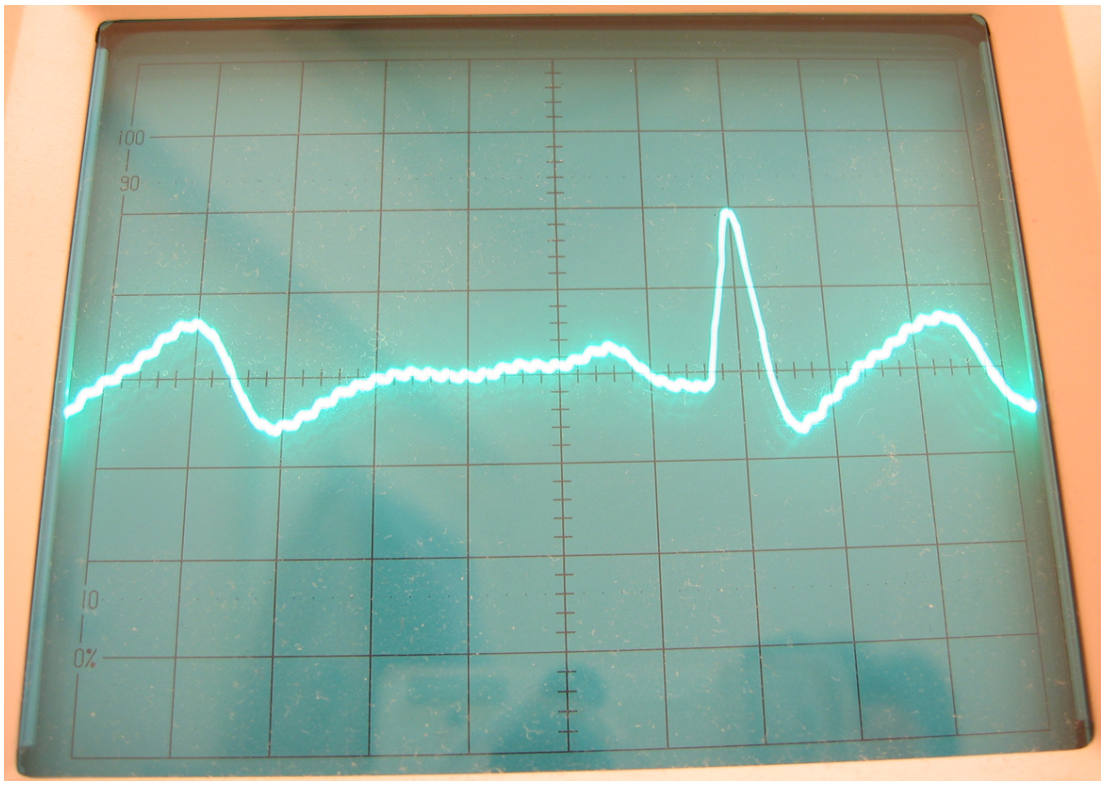


Figure 4.4 ECG signal with 2.5 V voltage reference.

We assume that, the achieved signal is acceptable to complete the project and we went on to test of other parts of the system.

4.2 RF Transmitter & Receiver Circuits

The 433 MHz RF Transceiver circuits projected for wireless ECG signal communication was ready after its PCBs were formed and mounted.

To test the circuits, firstly a fixed signal was applied to the DIO pin of the RF module. Then we checked the signal spectrum of the RF module via Spectrum Analyzer. As we expect, RF module ran on 433 MHz perfectly. Measurement of the RF module signal spectrum is shown below.

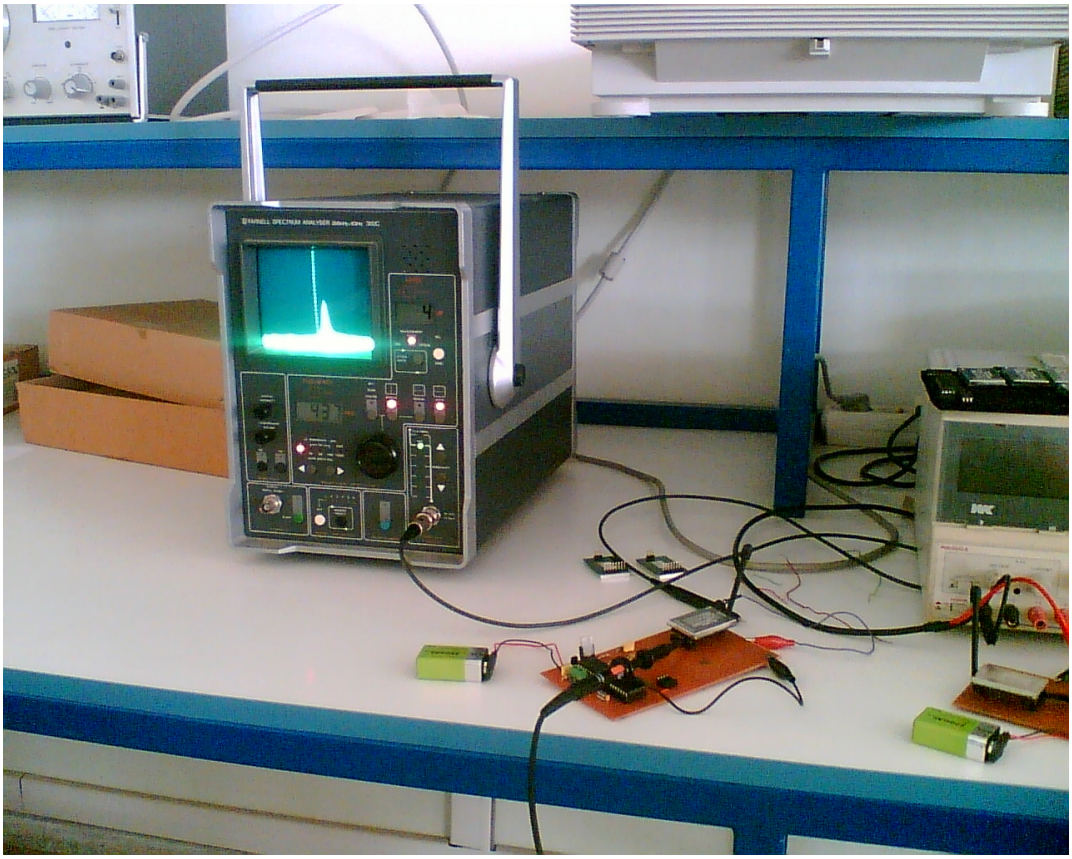


Figure 4.5 Signal spectrum of the RF module.

Carrying frequency of the RF modules which was measured approximately 433.92 MHz. This value is the same with indicated value in its datasheet.

Later, transmitter and receiver units were tested together. A square wave string was supplied to DIO pin the transmitter unit by 16F877A microcontroller. The signal on the input of the transmitter unit was 2.8 V (peak-to-peak), while on the output of the receiver unit was 2.4 V (peak-to-peak). Also there was a little phase shifting between transmitted and received signals.

Because of the fact that DIO pin of the RF module was connected to TTL input of the microcontroller, the maximum amplitude of the received signal was 2.4 V DC was not a problem. Input signal of the transmitter unit and output signal of the receiver unit is shown in Figure 4.6.

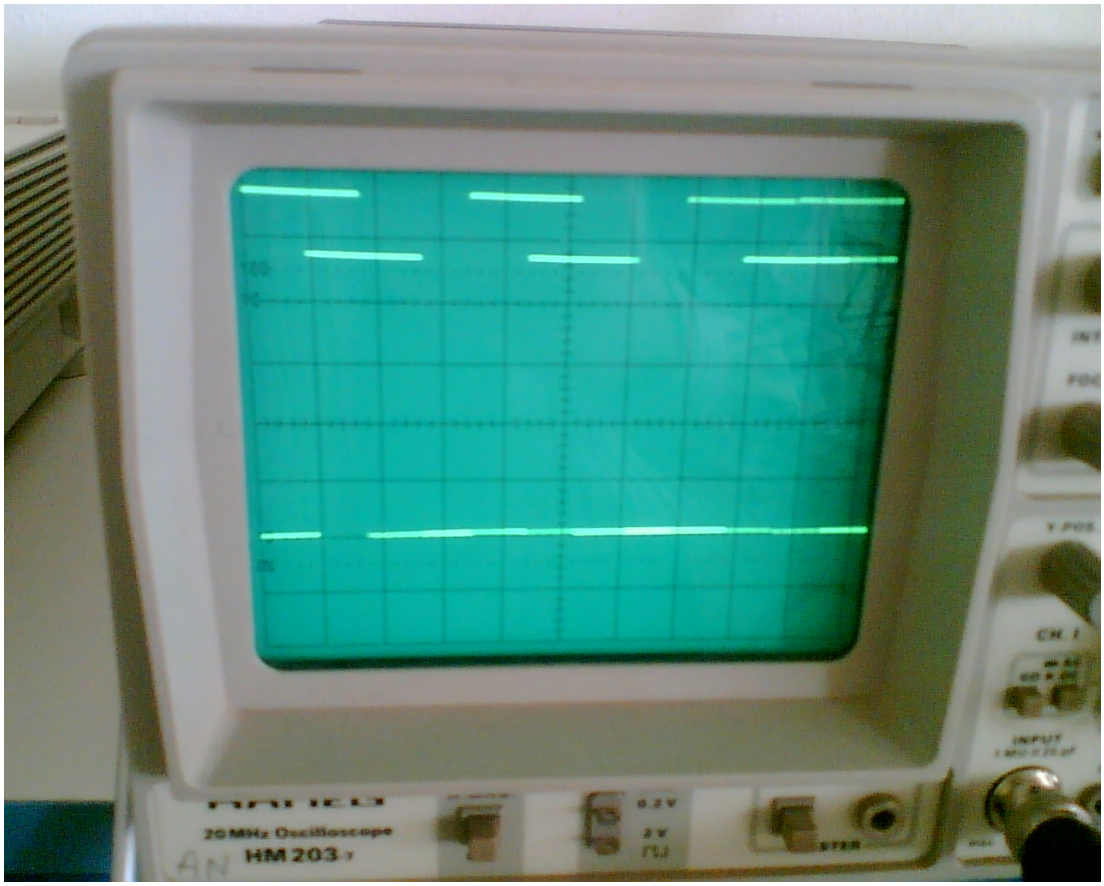


Figure 4.6 Input and output signals of the RFmodule.

A distance test was applied to the units too. These tests were made for both indoor and outdoor (max. 100m). That electromagnetic waves were communicated correctly were provided by whip antennas connected to two devices. Tests results showed that; transmitted signals reached to the other device both indoor and outdoor. Moreover, these tests were applied for both 300 baud and 2,400 baud. It was found out that the data communication was running correctly for transmitting and receiving.

Another test applied for the internal ADC of the microcontroller. For this reason, a battery was connected to the ADC via a potentiometer. While the input voltage of the ADC was changing, a sampling was done. Test results were good for this parts of the system too. Afterwards, transmitter and receiver unit were tested for higher sampling frequency via the signal generator.

Applied tests showed that, transmitter unit could sample the ECG signal for every 100 micro second and transmit them. In consideration of 200 Hz is enough for sampling of the ECG signal, this value is too much than we need.

4.3 ECG Drawing Software

Some tests for the PC software developing to show ECG signals graphically were done. By the codes running on the microcontroller of the transmitter unit side;

```
SEROUT2 PORTD.0,396,[REP$AA\5, REP$00\5, REP$FF\5]
    FOR i=0 TO 255
      SEROUT2 PORTD.0,396,[("M"),i]
    NEXT i
```

a signal that goes from 0 to 255 one by one was sent successively to the receiver unit. This signal looks like a right triangle. We expected that, this signal was received by the receiver unit and transferred to the PC exactly the same. That the signal was received by the receiver unit and transferred to the PC exactly the same was expected. It was necessary that the transmitted signal was drawn by the software which was developed.

After all tests, all the things which are expected on the beginning of the tests were achieved completely. As seen in the Figure 4.7, the signal which was transmitted was received by the receiver unit perfectly. Then, it was transferred to PC and drawn graphically on the software screen.

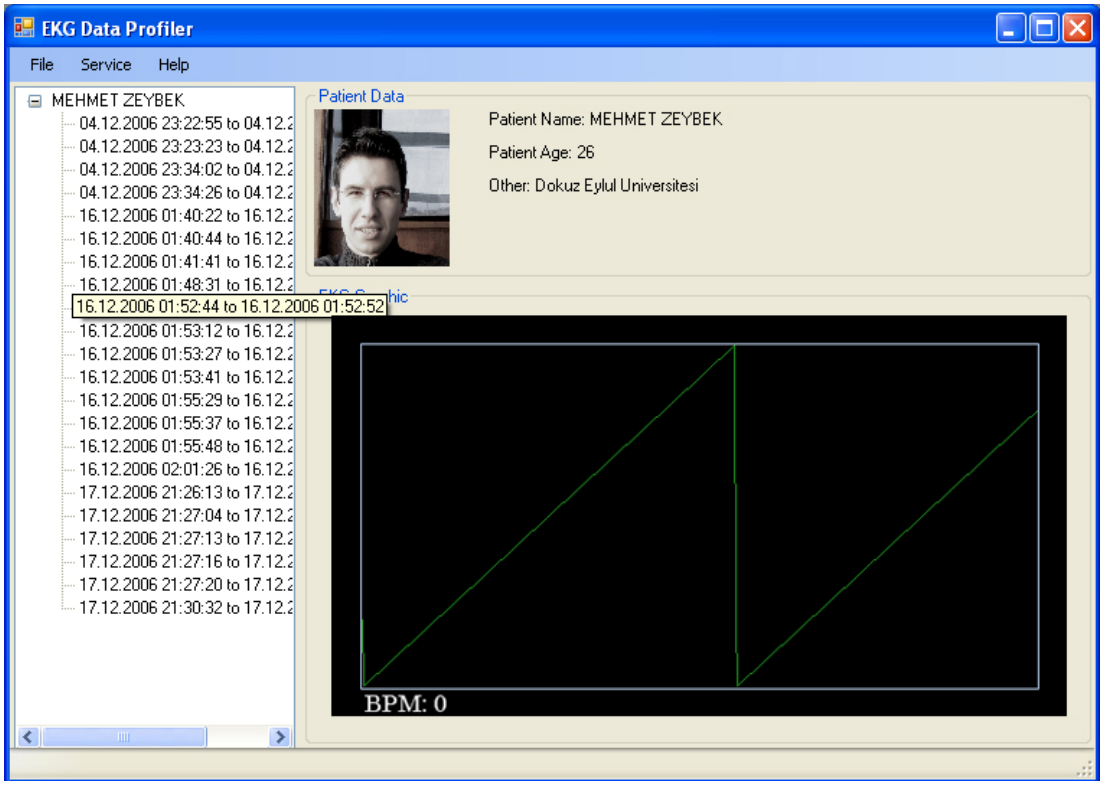


Figure 4.7 Test signal on the main screen of the PC software.

While the software running, some faults were occurred. For instance when the new patient had been recorded, a new .xml file was created by the software. However patients name was not showed until restart the software.

4.4 Test of the System

After all parts of the system connected with each other, last test was applied. Acquired signal from ECG circuit was fed into the input of the ADC. Then, a sample was taken per 1 ms and transmitted via RF module. The signal received from the receiver unit transferred to the PC via RS232 serial port.

There is a little difference between two ECG signals one from oscilloscope screen and the other from PC screen. Nevertheless, two signals resemble each other in general.

Figure 4.8 shows the real time ECG signal on the PC software screen.

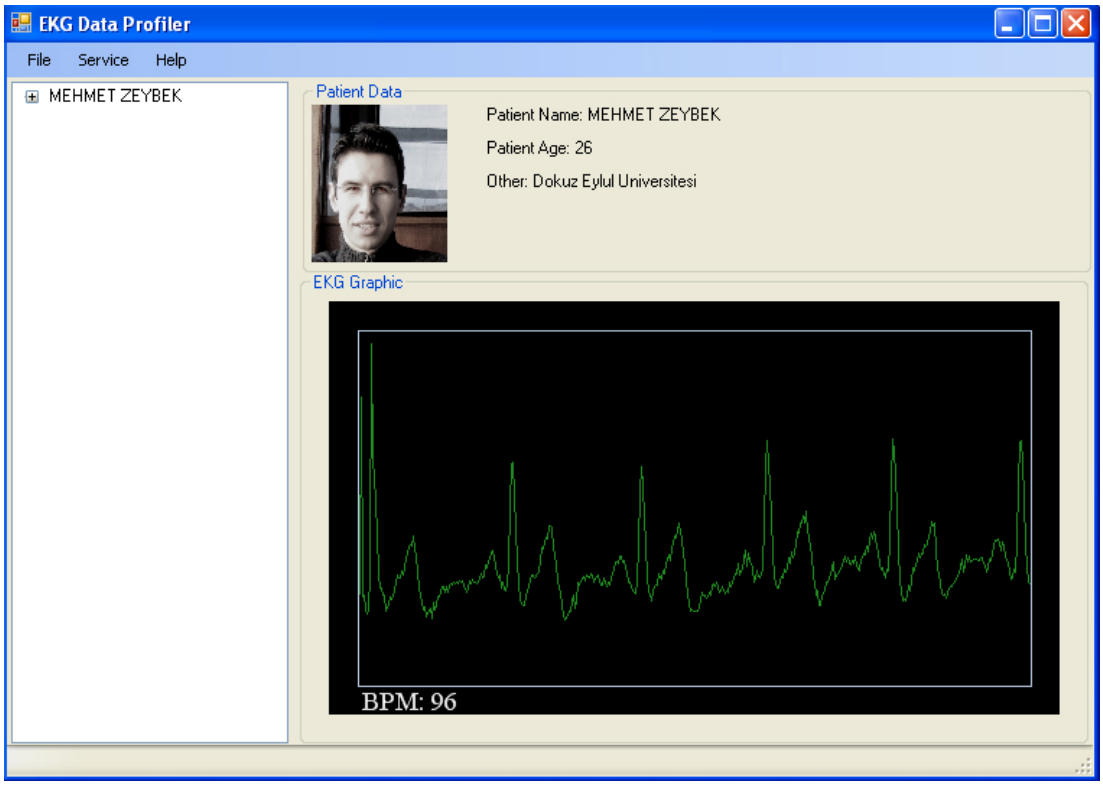


Figure 4.8 ECG signal on the PC software.

CHAPTER FIVE

DISCUSSION

In this study, mentioned before, we aimed to find some new solutions for wireless biomedical applications. By this aim, an ECG acquisition and wireless transmission system was designed.

All system parts are designed, realized, and tested for acceptable performance individually. After the all parts of the system connected to each other, whole system was tested. Then ECG signal was transmitted by RF module to PC, and also ECG signal was drawn graphically by the software run on the PC.

As can be shown in Figure 4.8, ECG graphic on the software screen is quite good. Nevertheless, some improvements can be applied to the system.

5.1 Future Works

It is possible to improve this wireless ECG system. Firstly, if a high speed RF module is used, system may have higher sampling rate. Also, if higher resolution value (e.g. 12-bit) is chosen, drawn signal will be more similar to real ECG signal. Furthermore, 12 channels ECG system can be possible, if higher speed microcontroller is used. System may be stored more patient records with appending a database to PC software.

When the code of the microcontrollers is improved, several patients' ECG signals can be transmitted simultaneously by the system.

Our system can not be used only for ECG signal, but also can be used for other biomedical signals. If some other acquisition circuits such EMG, blood pressure, sphygmus etc. are designed and appended to our design, new system which will be occurred can be used as a bedside or central monitor in the hospitals or medical centers.

Also internet access can be integrated in our system easily. For this, an advanced database must be created for the patients' records. Because of the fact that MS SQL is compatible with .Net platform which we used, this program should be used to create this database. In addition to this, if a dynamic web page is composed for remote access for the authorized people (doctors, etc.), our system can be used as a "*remote patient monitoring system*". All of the data from the patients are can be observed via internet, after the things mentioned above are realized.

In case of these future works are realized, our study can be a new aspect for our medical sector.

REFERENCES

Andreasson J., Ekström M., Fard A., Castano J. G., & Johnson T. (2002). Remote System for Patient Monitoring Using Bluetooth. presented at Sensors.

Berenfeld O., & Jalife J. (1998). Purkinje-muscle reentry as a mechanism of polymorphic ventricular arrhythmias in a 3-dimensional model of the ventricles. *Circ Res*, (82), 1063–1077.

Carmeliet E. (1993). Mechanisms and control of repolarization. *Eur Heart J*, (98), 3–13.

Combos E. & Hartman E. (2003). *ECG Front-End Design is Simplified with MicroConverter*. USA: Analog Devices.

Costill D. L. (1999). *Physiology of Sport and Exercise* (2nd ed.). Champaign: Human Kinetics.

Cromwell, Weibell L. & F., & Pfeiffer E. A. (1980). *Biomedical Instrumentation and Measurements.* NJ: Prentice-Hall, Inc.

Dam R.,& Janse M., (1989). *Activation of the heart. Comprehensive Electrocardiology*. New York: Pergamon Press.

Finch C. (1999). Mobile Computing in Healthcare. *Health Menagement Technology, vol. 20,* 63-64.

Hernandez A.I., Mora F., Villegas G., & Carrault G. (2001). Real-Time ECG Transmission Via Internet for Nonclinical Applications. *IEEE Trans. Inf. Technol. Biomed*, *vol. 5*, 253-57.

Hung K.H., & Zhang Y.-T. (2003). Implementation of a WAP-Based Telemedicine System for Patient Monitoring. *IEEE Trans. Inf. Technol. Biomed.*, *vol. 7*, 101-107.

Istepanian R. H., Woodward B., Gorilas E., & Balos P.A. (1998). Desing of mobile telemedicine systems using GSM and IS-54 cellular telephone standards. *J Telemed Telecare*, *vol. 4*, 80-82.

Jack J., Noble D., & Tsien R. (1983). *Electric Current Flow in Excitable Cells*. Oxford: Clarendon Press,

Katz A. (2000). *Physiology of the Heart* (3rd ed.). USA: Lippincott Williams & Wilkins.

Katzen S. (2000). *The Quintessential PIC Microcontroller*.NY: Springer.

Marieb E. N. (1998). *Human Anatomy and Physiology Fourth Edition.* California: Benjamin/Cummings Publishing Company.

Martinez-Palomo A., Alanis J., & Benitez D. (1970). Transitional cardiac cells of the conductive system of the dog heart. *J Cell Biol*, (47), 1–17.

Massing G. & James T. (1976). Anatomical configuration of the His bundle and bundle branches in the human heart. *Circulation*, (53), 609–621.

Mendez C., Mueller W., & Urguiaga X. (1970). Propagation of impulses across the Purkinje fibre-muscle junctions in the dog heart. *Circ Res*, (26), 135–150.

Microchip (2003). *16F87XA Datasheet*. www.microchip.com

Rollins D. L., Killingsworth C. R., Walcott G.P., Justice R.K., & Ideker R.E. (2000). A Telemetry System for the Study of Spontaneous Cardiac Arrhytmias. *IEEE Trans. on Biomed. Eng*., *vol. 47*, 887-892.

Rosenbaum B. (1970). The hemiblocks: Diagnostic criteria and clinical significance. *Modern Concepts of the Cardiovascular Disease*, (39), 141–146.

Ross M. (1989). Muscular tissue. *Histology - A Text and Atlas*, 203–241.

SCHILLER (2003). *ECG Holter Recorder MT-120.* 09.08.2006, http://www.schiller.ch/products.html.

Titus J. (1973). Normal anatomy of the human cardiac conduction system. *Mayo Clinic Proc*, (48), 24–30.

Townsend N. (2001). *ECG Instrumentations*. Retrieved 14.10.2006 from http://www.robots.ox.ac.uk/~neil/teaching/lectures/med_elec/notes2.pdf

Tranum-Jensen J., Wilde A., Vermeulen J., & Janse M. (1991). Morphology of electrophysiologically identified junctions between Purkinje fibres and ventricular muscle in rabbit and pig hearts. *Circ Res*, (69), 429–437.

Truex R. (1974). Myocardial cell diameters in primate hearts. *Am J Anat*, (135), 269–280.

Webster J.G. (1998). *Medical Instrumentation. Application and Design* (3rd ed.). Wiley.

Yan G.-X., Shimizu W., & Antzelevitch C. (1998). Characteristics and distribution of M cells in arterially perfused canine left ventricular wedge preparations. *Circulation*, (98), 1921–1927.

APPENDICES

APPENDIX A	AD624 Datasheet
APPENDIX B	AD712 Datasheet
APPENDIX C	16F877A Datasheet
APPENDIX D	UTR-C12 U Datasheet
APPENDIX E	Source Codes of the PICs
APPENDIX F	Source Code of the ECG Drawing Software

All appendices are in CD.

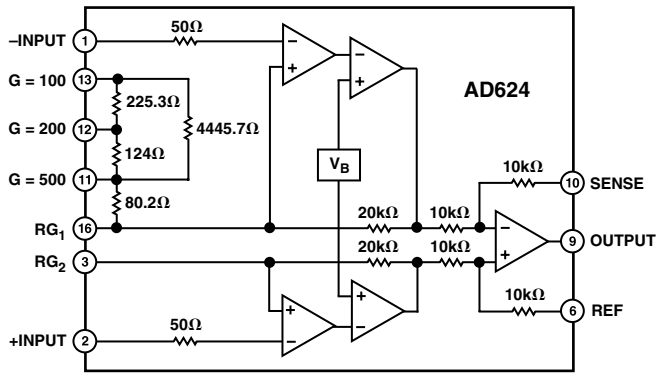
Precision Instrumentation Amplifier

AD624

FEATURES

Low Noise: 0.2 μV p-p 0.1 Hz to 10 Hz
Low Gain TC: 5 ppm max (G = 1)
Low Nonlinearity: 0.001% max (G = 1 to 200)
High CMRR: 130 dB min (G = 500 to 1000)
Low Input Offset Voltage: 25 μV, max
Low Input Offset Voltage Drift: 0.25 μV/°C max
Gain Bandwidth Product: 25 MHz
Pin Programmable Gains of 1, 100, 200, 500, 1000
No External Components Required
Internally Compensated

FUNCTIONAL BLOCK DIAGRAM



PRODUCT DESCRIPTION

The AD624 is a high precision, low noise, instrumentation amplifier designed primarily for use with low level transducers, including load cells, strain gauges and pressure transducers. An outstanding combination of low noise, high gain accuracy, low gain temperature coefficient and high linearity make the AD624 ideal for use in high resolution data acquisition systems.

The AD624C has an input offset voltage drift of less than 0.25 μV/°C, output offset voltage drift of less than 10 μV/°C, CMRR above 80 dB at unity gain (130 dB at G = 500) and a maximum nonlinearity of 0.001% at G = 1. In addition to these outstanding dc specifications, the AD624 exhibits superior ac performance as well. A 25 MHz gain bandwidth product, 5 V/μs slew rate and 15 μs settling time permit the use of the AD624 in high speed data acquisition applications.

The AD624 does not need any external components for pretrimmed gains of 1, 100, 200, 500 and 1000. Additional gains such as 250 and 333 can be programmed within one percent accuracy with external jumpers. A single external resistor can also be used to set the 624's gain to any value in the range of 1 to 10,000.

PRODUCT HIGHLIGHTS

1. The AD624 offers outstanding noise performance. Input noise is typically less than 4 nV/$\sqrt{\text{Hz}}$ at 1 kHz.
2. The AD624 is a functionally complete instrumentation amplifier. Pin programmable gains of 1, 100, 200, 500 and 1000 are provided on the chip. Other gains are achieved through the use of a single external resistor.
3. The offset voltage, offset voltage drift, gain accuracy and gain temperature coefficients are guaranteed for all pretrimmed gains.
4. The AD624 provides totally independent input and output offset nulling terminals for high precision applications. This minimizes the effect of offset voltage in gain ranging applications.
5. A sense terminal is provided to enable the user to minimize the errors induced through long leads. A reference terminal is also provided to permit level shifting at the output.

REV. C

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 which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 **www.analog.com**
Fax: 781/326-8703 **© Analog Devices, Inc., 1999**

AD624—SPECIFICATIONS (@ V_S = ±15 V, R_L = 2 kΩ and T_A = +25°C, unless otherwise noted)

Model	AD624A			AD624B			AD624C			AD624S			Units
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
GAIN													
Gain Equation (External Resistor Gain Programming)		$\left[\frac{40,000}{R_G}+1\right]\pm 20\%$			$\left[\frac{40,000}{R_G}+1\right]\pm 20\%$			$\left[\frac{40,000}{R_G}+1\right]\pm 20\%$			$\left[\frac{40,000}{R_G}+1\right]\pm 20\%$		
Gain Range (Pin Programmable)	1 to 1000			1 to 1000			1 to 1000			1 to 1000			
Gain Error													
G = 1			**±0.05**			**±0.03**			**±0.02**			**±0.05**	%
G = 100			**±0.25**			**±0.15**			**±0.1**			**±0.25**	%
G = 200, 500			**±0.5**			**±0.35**			**±0.25**			**±0.5**	%
Nonlinearity													
G = 1			±0.005			±0.003			±0.001			±0.005	%
G = 100, 200			±0.005			±0.003			±0.001			±0.005	%
G = 500			±0.005			±0.005			±0.005			±0.005	%
Gain vs. Temperature													
G = 1			5			5			5			5	ppm/°C
G = 100, 200			10			10			10			10	ppm/°C
G = 500			25			15			15			15	ppm/°C
VOLTAGE OFFSET (May be Nulled)													
Input Offset Voltage			**200**			**75**			**25**			**75**	µV
vs. Temperature			**2**			**0.5**			**0.25**			**2.0**	µV/°C
Output Offset Voltage			**5**			**3**			**2**			**3**	mV
vs. Temperature			**50**			**25**			**10**			**50**	µV/°C
Offset Referred to the Input vs. Supply (V_{OUT})													
G = 1	**70**			**75**			**80**			**75**			dB
G = 100, 200	**95**			**105**			**110**			**105**			dB
G = 500	**100**			**110**			**115**			**110**			dB
INPUT CURRENT													
Input Bias Current			**±50**			**±25**			**±15**			**±50**	nA
vs. Temperature		±50			±50			±50			±50		pA/°C
Input Offset Current			**±35**			**±15**			**±10**			**±35**	nA
vs. Temperature		±20			±20			±20			±20		pA/°C
INPUT													
Input Impedance													
Differential Resistance		10^9			10^9			10^9			10^9		Ω
Differential Capacitance		10			10			10			10		pF
Common-Mode Resistance		10^9			10^9			10^9			10^9		Ω
Common-Mode Capacitance		10			10			10			10		pF
Input Voltage Range[1]													
Max Differ. Input Linear (V_{DL})	±10			±10			±10			±10			V
Max Common-Mode Linear (V_{CM})		$12\ V-\left(\frac{G}{2}\times V_D\right)$			$12\ V-\left(\frac{G}{2}\times V_D\right)$			$12\ V-\left(\frac{G}{2}\times V_D\right)$			$12\ V-\left(\frac{G}{2}\times V_D\right)$		V
Common-Mode Rejection dc to 60 Hz with 1 kΩ Source Imbalance													
G = 1	**70**			**75**			**80**			**70**			dB
G = 100, 200	**100**			**105**			**110**			**100**			dB
G = 500	**110**			**120**			**130**			**110**			dB
OUTPUT RATING													
V , R_L = 2 kΩ		±10			±10			±10			±10		V
DYNAMIC RESPONSE													
Small Signal –3 dB													
G = 1		1			1			1			1		MHz
G = 100		150			150			150			150		kHz
G = 200		100			100			100			100		kHz
G = 500		50			50			50			50		kHz
G = 1000		25			25			25			25		kHz
Slew Rate		5.0			5.0			5.0			5.0		V/µs
Settling Time to 0.01%, 20 V Step													
G = 1 to 200		15			15			15			15		µs
G = 500		35			35			35			35		µs
G = 1000		75			75			75			75		µs
NOISE													
Voltage Noise, 1 kHz													
R.T.I.		4			4			4			4		nV/√Hz
R.T.O.		75			75			75			75		nV/√Hz
R.T.I., 0.1 Hz to 10 Hz													
G = 1		10			10			10			10		µV p-p
G = 100		0.3			0.3			0.3			0.3		µV p-p
G = 200, 500, 1000		0.2			0.2			0.2			0.2		µV p-p
Current Noise													
0.1 Hz to 10 Hz		60			60			60			60		pA p-p
SENSE INPUT													
R_{IN}	8	10	12	8	10	12	8	10	12	8	10	12	kΩ
I_{IN}		30			30			30			30		µA
Voltage Range	±10			±10			±10			±10			V
Gain to Output		1			1			1			1		%

Model	AD624A			AD624B			AD624C			AD624S			Units
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
REFERENCE INPUT													
R_{IN}	16	20	24	16	20	24	16	20	24	16	20	24	kΩ
I_{IN}		30			30			30			30		μA
Voltage Range	±10			±10			±10			±10			V
Gain to Output		1			1			1			1		%
TEMPERATURE RANGE													
Specified Performance	–25		+85	–25		+85	–25		+85	–55		+125	°C
Storage	–65		+150	–65		+150	–65		+150	–65		+150	°C
POWER SUPPLY													
Power Supply Range	**±6**	**±15**	**±18**	**±6**	**±15**	**±18**	**±6**	**±15**	**±18**	**±6**	**±15**	**±18**	V
Quiescent Current		3.5	5		3.5	5		3.5	5		3.5	5	mA

NOTES

[1]V_{DL} is the maximum differential input voltage at G = 1 for specified nonlinearity, V_{DL} at other gains = 10 V/G. V_D = actual differential input voltage.
Example: G = 10, V_D = 0.50. V_{CM} = 12 V – (10/2 × 0.50 V) = 9.5 V.

Specifications subject to change without notice.

Specifications shown in **boldface** are tested on all production unit at final electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in boldface are tested on all production units.

ABSOLUTE MAXIMUM RATINGS*

Supply Voltage . . . . . ±18 V
Internal Power Dissipation . . . . . 420 mW
Input Voltage . . . . . $\pm V_S$
Differential Input Voltage . . . . . $\pm V_S$
Output Short Circuit Duration . . . . . Indefinite
Storage Temperature Range . . . . . –65°C to +150°C
Operating Temperature Range
AD624A/B/C . . . . . –25°C to +85°C
AD624S . . . . . –55°C to +125°C
Lead Temperature (Soldering, 60 secs) . . . . . +300°C

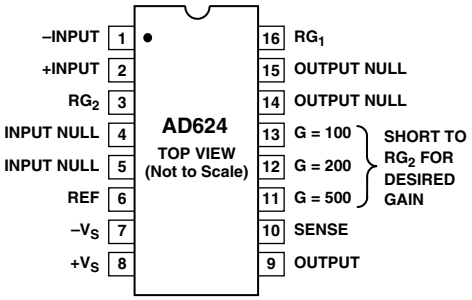
*Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ORDERING GUIDE

Model	Temperature Range	Package Description	Package Option
AD624AD	–25°C to +85°C	16-Lead Ceramic DIP	D-16
AD624BD	–25°C to +85°C	16-Lead Ceramic DIP	D-16
AD624CD	–25°C to +85°C	16-Lead Ceramic DIP	D-16
AD624SD	–55°C to +125°C	16-Lead Ceramic DIP	D-16
AD624SD/883B*	–55°C to +125°C	16-Lead Ceramic DIP	D-16
AD624AChips	–25°C to +85°C	Die	
AD624SChips	–25°C to +85°C	Die	

*See Analog Devices' military data sheet for 883B specifications.

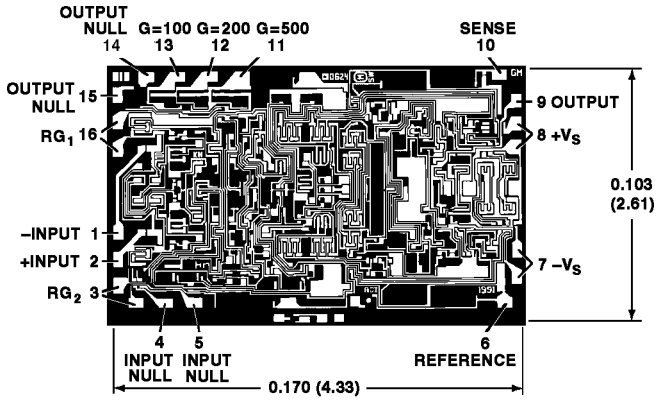
CONNECTION DIAGRAM



FOR GAINS OF 1000 SHORT RG_1 TO PIN 12 AND PINS 11 AND 13 TO RG_2

METALIZATION PHOTOGRAPH

Contact factory for latest dimensions
Dimensions shown in inches and (mm).



PAD NUMBERS CORRESPOND TO PIN NUMBERS FOR THE D-16 16-LEAD CERAMIC PACKAGE

AD624–Typical Characteristics

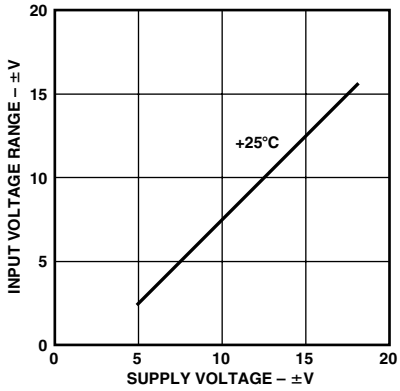


Figure 1. Input Voltage Range vs. Supply Voltage, G = 1

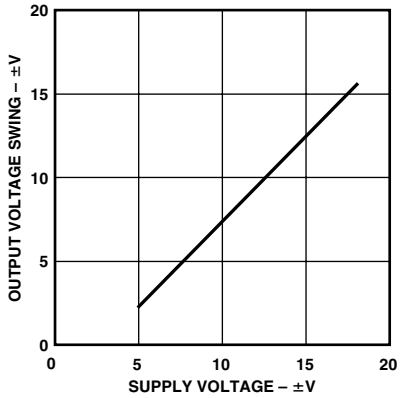


Figure 2. Output Voltage Swing vs. Supply Voltage

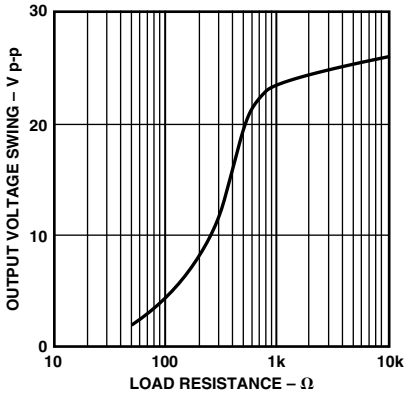


Figure 3. Output Voltage Swing vs. Load Resistance

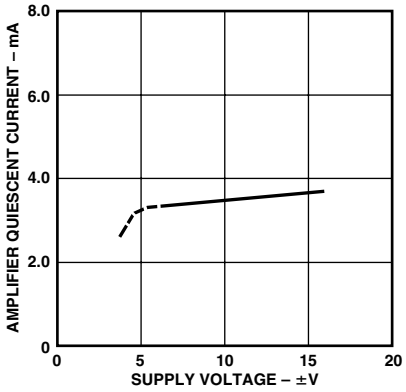


Figure 4. Quiescent Current vs. Supply Voltage

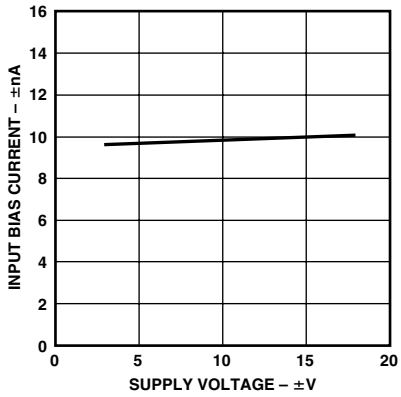


Figure 5. Input Bias Current vs. Supply Voltage

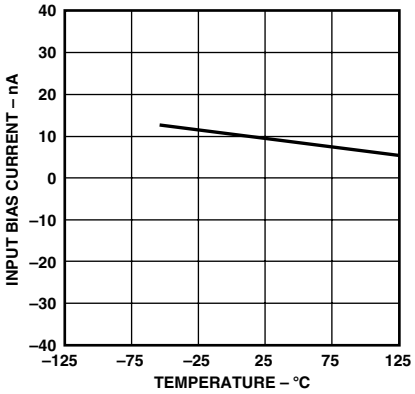


Figure 6. Input Bias Current vs. Temperature

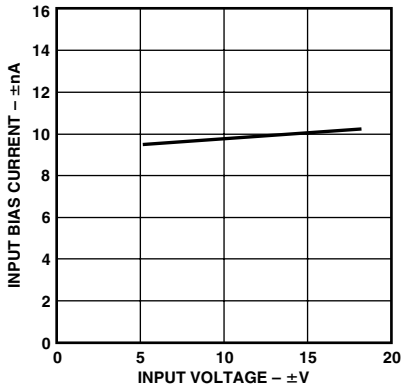


Figure 7. Input Bias Current vs. CMV

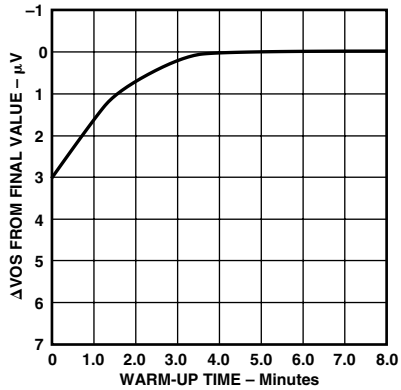


Figure 8. Offset Voltage, RTI, Turn On Drift

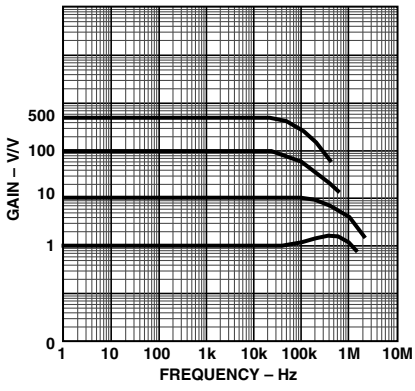


Figure 9. Gain vs. Frequency

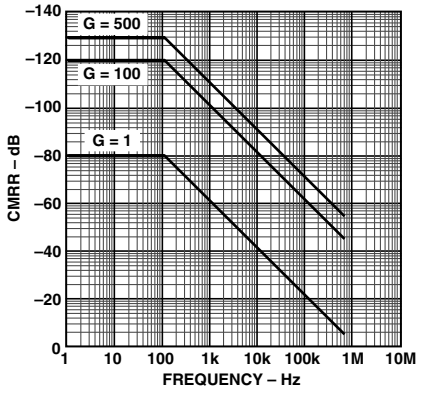


Figure 10. CMRR vs. Frequency RTI, Zero to 1k Source Imbalance

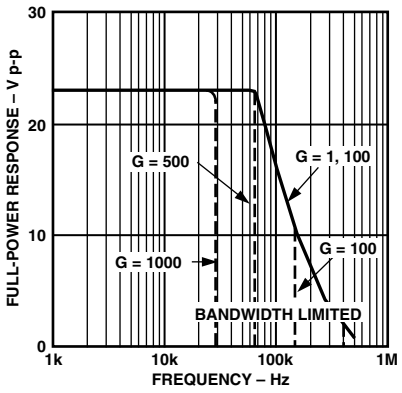


Figure 11. Large Signal Frequency Response

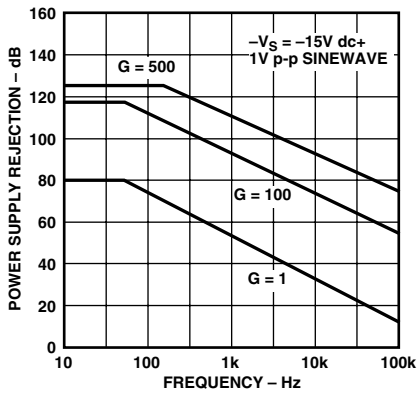


Figure 12. Positive PSRR vs. Frequency

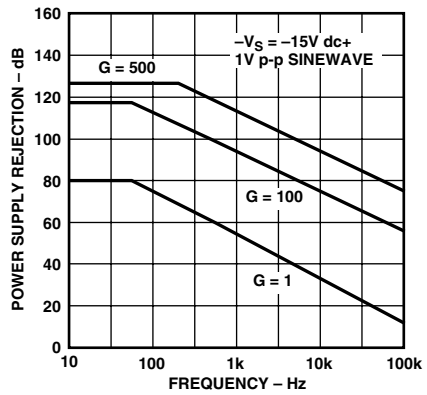


Figure 13. Negative PSRR vs. Frequency

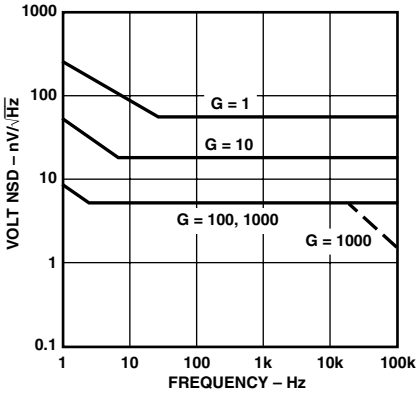


Figure 14. RTI Noise Spectral Density vs. Gain

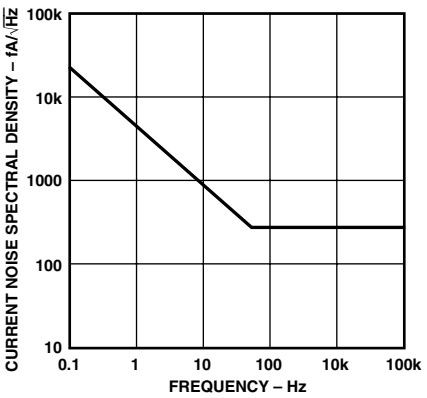


Figure 15. Input Current Noise

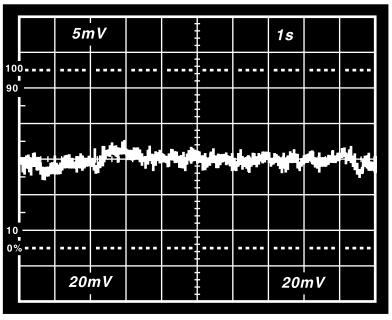


Figure 16. Low Frequency Voltage Noise, G = 1 (System Gain = 1000)

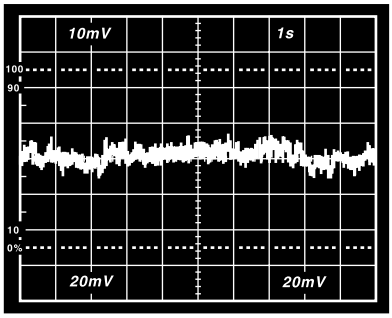


Figure 17. Low Frequency Voltage Noise, G = 1000 (System Gain = 100,000)

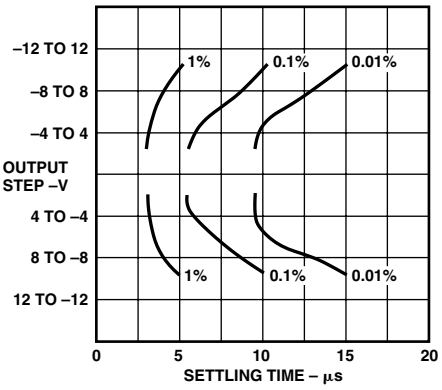


Figure 18. Settling Time, Gain = 1

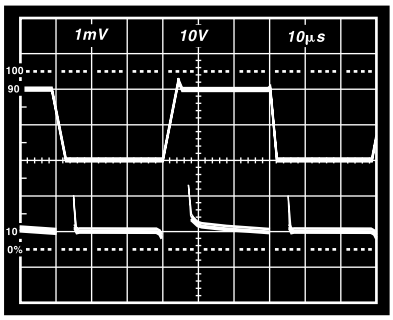


Figure 19. Large Signal Pulse Response and Settling Time, G = 1

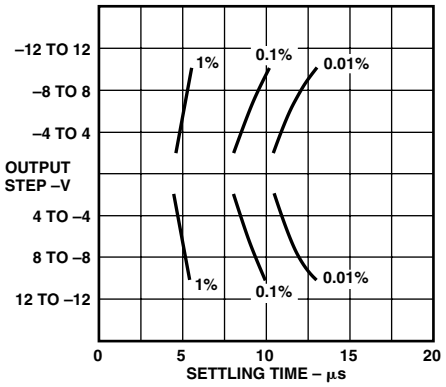


Figure 20. Settling Time Gain = 100

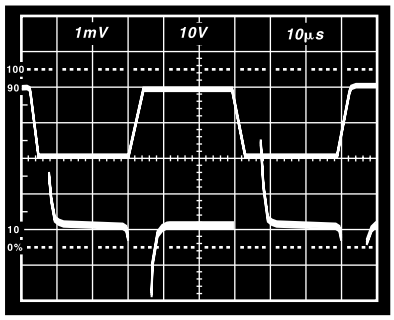


Figure 21. Large Signal Pulse Response and Settling Time, G = 100

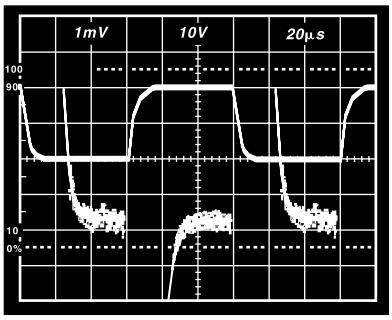


Figure 22. Range Signal Pulse Response and Settling Time, G = 500

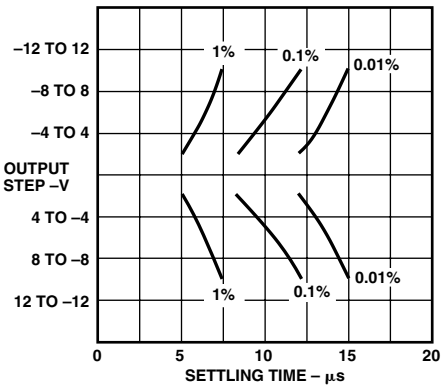


Figure 23. Settling Time Gain = 1000

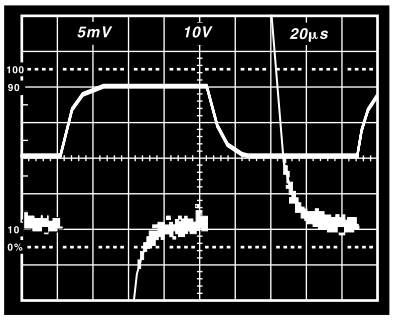


Figure 24. Large Signal Pulse Response and Settling Time, G = 1000

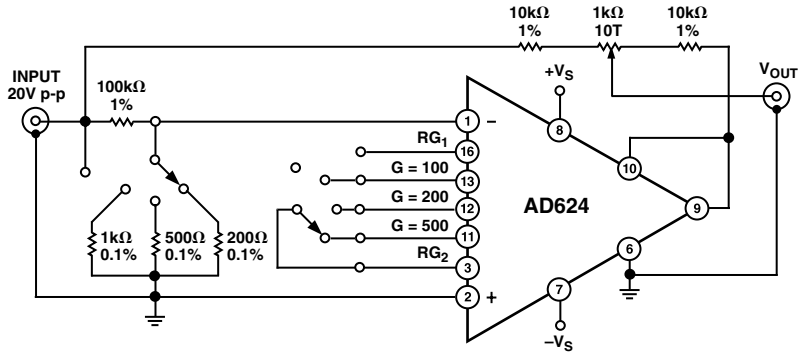


Figure 25. Settling Time Test Circuit

THEORY OF OPERATION

The AD624 is a monolithic instrumentation amplifier based on a modification of the classic three-op-amp instrumentation amplifier. Monolithic construction and laser-wafer-trimming allow the tight matching and tracking of circuit components and the high level of performance that this circuit architecture is capable of.

A preamp section (Q1–Q4) develops the programmed gain by the use of feedback concepts. Feedback from the outputs of A1 and A2 forces the collector currents of Q1–Q4 to be constant thereby impressing the input voltage across R_G.

The gain is set by choosing the value of R_G from the equation, Gain = $\frac{40\ k}{R_G}$ + 1. The value of R_G also sets the transconductance of the input preamp stage increasing it asymptotically to the transconductance of the input transistors as R_G is reduced for larger gains. This has three important advantages. First, this approach allows the circuit to achieve a very high open loop gain of 3×10^8 at a programmed gain of 1000 thus reducing gain related errors to a negligible 3 ppm. Second, the gain bandwidth product which is determined by C3 or C4 and the input transconductance, reaches 25 MHz. Third, the input voltage noise reduces to a value determined by the collector current of the input transistors for an RTI noise of 4 nV/$\sqrt{\text{Hz}}$ at G ≥ 500.

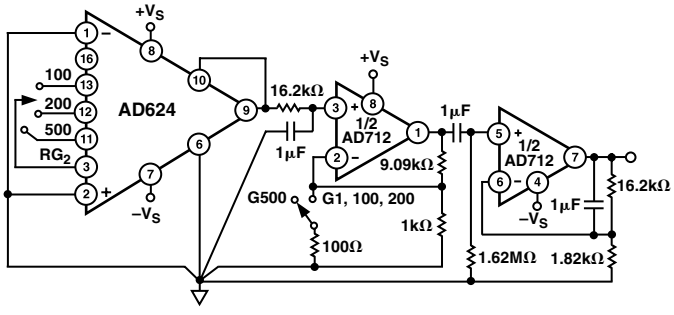


Figure 26. Noise Test Circuit

INPUT CONSIDERATIONS

Under input overload conditions the user will see R_G + 100 Ω and two diode drops (~1.2 V) between the plus and minus inputs, in either direction. If safe overload current under all conditions is assumed to be 10 mA, the maximum overload voltage is ~ ±2.5 V. While the AD624 can withstand this continuously, momentary overloads of ±10 V will not harm the device. On the other hand the inputs should never exceed the supply voltage.

The AD524 should be considered in applications that require protection from severe input overload. If this is not possible, external protection resistors can be put in series with the inputs of the AD624 to augment the internal (50 Ω) protection resistors. This will most seriously degrade the noise performance. For this reason the value of these resistors should be chosen to be as low as possible and still provide 10 mA of current limiting under maximum continuous overload conditions. In selecting the value of these resistors, the internal gain setting resistor and the 1.2 volt drop need to be considered. For example, to protect the device from a continuous differential overload of 20 V at a gain of 100, 1.9 kΩ of resistance is required. The internal gain resistor is 404 Ω; the internal protect resistor is 100 Ω. There is a 1.2 V drop across D1 or D2 and the base-emitter junction of either Q1 and Q3 or Q2 and Q4 as shown in Figure 27, 1400 Ω of external resistance would be required (700 Ω in series with each input). The RTI noise in this case would be

$$\sqrt{4\,KTR_{ext} + (4\,nV/\sqrt{Hz})^2} = 6.2\,nV/\sqrt{Hz}$$

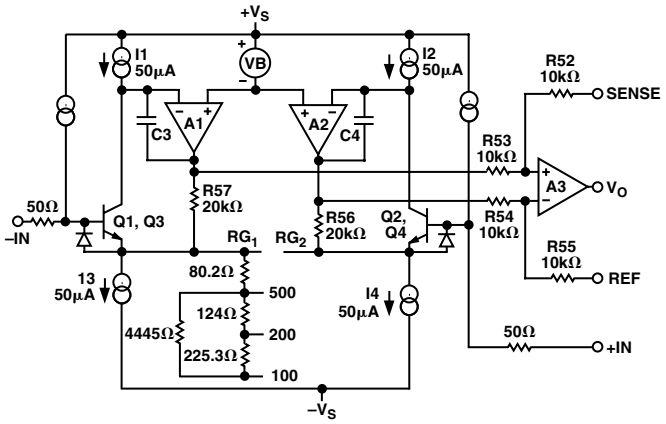


Figure 27. Simplified Circuit of Amplifier; Gain Is Defined as (R56 + R57)/(R_G) + 1. For a Gain of 1, R_G Is an Open Circuit.

INPUT OFFSET AND OUTPUT OFFSET

Voltage offset specifications are often considered a figure of merit for instrumentation amplifiers. While initial offset may be adjusted to zero, shifts in offset voltage due to temperature variations will cause errors. Intelligent systems can often correct for this factor with an autozero cycle, but there are many small-signal high-gain applications that don't have this capability.

Voltage offset and offset drift each have two components; input and output. Input offset is that component of offset that is

directly proportional to gain i.e., input offset as measured at the output at G = 100 is 100 times greater than at G = 1. Output offset is independent of gain. At low gains, output offset drift is dominant, while at high gains input offset drift dominates. Therefore, the output offset voltage drift is normally specified as drift at G = 1 (where input effects are insignificant), while input offset voltage drift is given by drift specification at a high gain (where output offset effects are negligible). All input-related numbers are referred to the input (RTI) which is to say that the effect on the output is "G" times larger. Voltage offset vs. power supply is also specified at one or more gain settings and is also RTI.

By separating these errors, one can evaluate the total error independent of the gain setting used. In a given gain configuration both errors can be combined to give a total error referred to the input (R.T.I.) or output (R.T.O.) by the following formula:

Total Error R.T.I. = input error + (output error/gain)

Total Error R.T.O. = (Gain × input error) + output error

As an illustration, a typical AD624 might have a +250 μV output offset and a –50 μV input offset. In a unity gain configuration, the *total* output offset would be 200 μV or the sum of the two. At a gain of 100, the output offset would be –4.75 mV or: +250 μV + 100 (–50 μV) = –4.75 mV.

The AD624 provides for both input and output offset adjustment. This optimizes nulling in very high precision applications and minimizes offset voltage effects in switched gain applications. In such applications the input offset is adjusted first at the highest programmed gain, then the output offset is adjusted at G = 1.

GAIN

The AD624 includes high accuracy pretrimmed internal gain resistors. These allow for single connection programming of gains of 1, 100, 200 and 500. Additionally, a variety of gains including a pretrimmed gain of 1000 can be achieved through series and parallel combinations of the internal resistors. Table I shows the available gains and the appropriate pin connections and gain temperature coefficients.

The gain values achieved via the combination of internal resistors are extremely useful. The temperature coefficient of the gain is dependent primarily on the mismatch of the temperature coefficients of the various internal resistors. Tracking of these resistors is extremely tight resulting in the low gain TCs shown in Table I.

If the desired value of gain is not attainable using the internal resistors, a single external resistor can be used to achieve any gain between 1 and 10,000. This resistor connected between

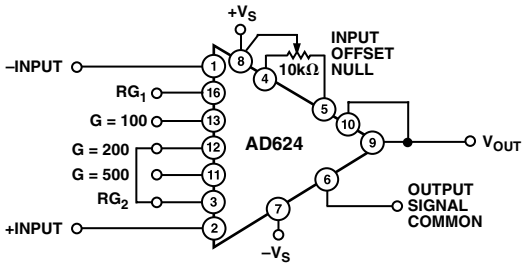


Figure 28. Operating Connections for G = 200

Table I.

Gain (Nominal)	Temperature Coefficient (Nominal)	Pin 3 to Pin	Connect Pins
1	–0 ppm/°C	–	–
100	–1.5 ppm/°C	13	–
125	–5 ppm/°C	13	11 to 16
137	–5.5 ppm/°C	13	11 to 12
186.5	–6.5 ppm/°C	13	11 to 12 to 16
200	–3.5 ppm/°C	12	–
250	–5.5 ppm/°C	12	11 to 13
333	–15 ppm/°C	12	11 to 16
375	–0.5 ppm/°C	12	13 to 16
500	–10 ppm/°C	11	–
624	–5 ppm/°C	11	13 to 16
688	–1.5 ppm/°C	11	11 to 12; 13 to 16
831	+4 ppm/°C	11	16 to 12
1000	0 ppm/°C	11	16 to 12; 13 to 11

Pins 3 and 16 programs the gain according to the formula

$$R_G = \frac{40k}{G-1}$$

(see Figure 29). For best results R_G should be a precision resistor with a low temperature coefficient. An external R_G affects both gain accuracy and gain drift due to the mismatch between it and the internal thin-film resistors R56 and R57. Gain accuracy is determined by the tolerance of the external R_G and the absolute accuracy of the internal resistors (±20%). Gain drift is determined by the mismatch of the temperature coefficient of R_G and the temperature coefficient of the internal resistors (–15 ppm/°C typ), and the temperature coefficient of the internal interconnections.

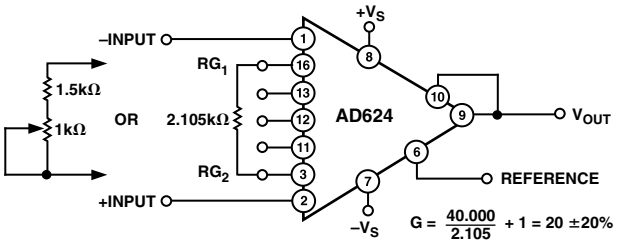


Figure 29. Operating Connections for G = 20

The AD624 may also be configured to provide gain in the output stage. Figure 30 shows an H pad attenuator connected to the reference and sense lines of the AD624. The values of R1, R2 and R3 should be selected to be as low as possible to minimize the gain variation and reduction of CMRR. Varying R2 will precisely set the gain without affecting CMRR. CMRR is determined by the match of R1 and R3.

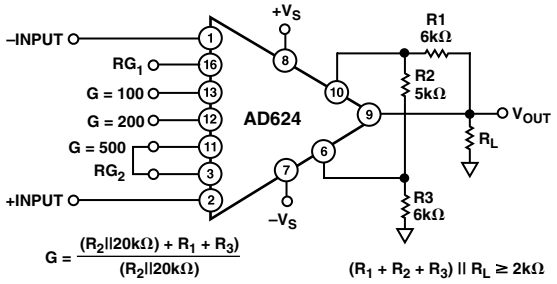


Figure 30. Gain of 2500

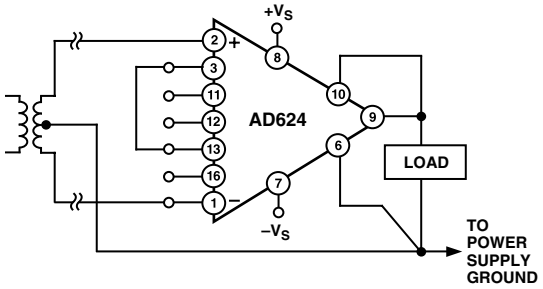
NOISE

The AD624 is designed to provide noise performance near the theoretical noise floor. This is an extremely important design criteria as the front end noise of an instrumentation amplifier is the ultimate limitation on the resolution of the data acquisition system it is being used in. There are two sources of noise in an instrument amplifier, the input noise, predominantly generated by the differential input stage, and the output noise, generated by the output amplifier. Both of these components are present at the input (and output) of the instrumentation amplifier. At the input, the input noise will appear unaltered; the output noise will be attenuated by the closed loop gain (at the output, the output noise will be unaltered; the input noise will be amplified by the closed loop gain). Those two noise sources must be root sum squared to determine the total noise level expected at the input (or output).

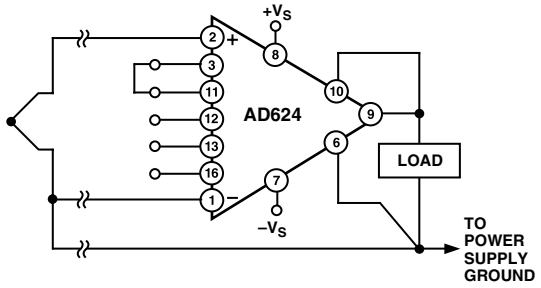
The low frequency (0.1 Hz to 10 Hz) voltage noise due to the output stage is 10 µV p-p, the contribution of the input stage is 0.2 µV p-p. At a gain of 10, the RTI voltage noise would be 1 µV p-p, $\sqrt{\left(\frac{10}{G}\right)^2 + (0.2)^2}$. The RTO voltage noise would be 10.2 µV p-p, $\sqrt{10^2 + \left(0.2(G)\right)^2}$. These calculations hold for applications using either internal or external gain resistors.

INPUT BIAS CURRENTS

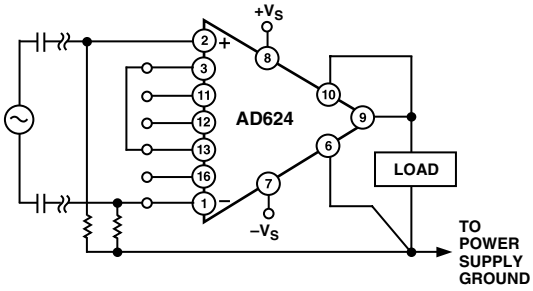
Input bias currents are those currents necessary to bias the input transistors of a dc amplifier. Bias currents are an additional source of input error and must be considered in a total error budget. The bias currents when multiplied by the source resistance imbalance appear as an additional offset voltage. (What is of concern in calculating bias current errors is the change in bias current with respect to signal voltage and temperature.) Input offset current is the difference between the two input bias currents. The effect of offset current is an input offset voltage whose magnitude is the offset current times the source resistance.



a. Transformer Coupled



b. Thermocouple



c. AC-Coupled

Figure 31. Indirect Ground Returns for Bias Currents

Although instrumentation amplifiers have differential inputs, there must be a return path for the bias currents. If this is not provided, those currents will charge stray capacitances, causing the output to drift uncontrollably or to saturate. Therefore, when amplifying "floating" input sources such as transformers and thermocouples, as well as ac-coupled sources, there must still be a dc path from each input to ground, (see Figure 31).

COMMON-MODE REJECTION

Common-mode rejection is a measure of the change in output voltage when both inputs are changed by equal amounts. These specifications are usually given for a full-range input voltage change and a specified source imbalance. "Common-Mode Rejection Ratio" (CMRR) is a ratio expression while "Common-Mode Rejection" (CMR) is the logarithm of that ratio. For example, a CMRR of 10,000 corresponds to a CMR of 80 dB.

In an instrumentation amplifier, ac common-mode rejection is only as good as the differential phase shift. Degradation of ac common-mode rejection is caused by unequal drops across differing track resistances and a differential phase shift due to varied stray capacitances or cable capacitances. In many applications shielded cables are used to minimize noise. This technique can create common-mode rejection errors unless the shield is properly driven. Figures 32 and 33 shows active data guards which are configured to improve ac common-mode rejection by "bootstrapping" the capacitances of the input cabling, thus minimizing differential phase shift.

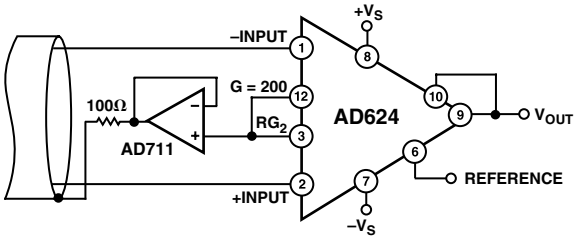


Figure 32. Shield Driver, G ≥ 100

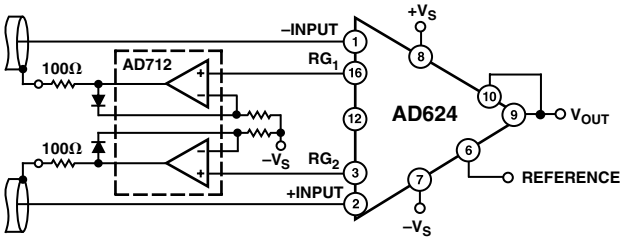


Figure 33. Differential Shield Driver

GROUNDING

Many data-acquisition components have two or more ground pins which are not connected together within the device. These grounds must be tied together at one point, usually at the system power supply ground. Ideally, a single solid ground would be desirable. However, since current flows through the ground wires and etch stripes of the circuit cards, and since these paths have resistance and inductance, hundreds of millivolts can be generated between the system ground point and the data acquisition components. Separate ground returns should be provided to minimize the current flow in the path from the most sensitive points to the system ground point. In this way supply currents and logic-gate return currents are not summed into the same return path as analog signals where they would cause measurement errors (see Figure 34).

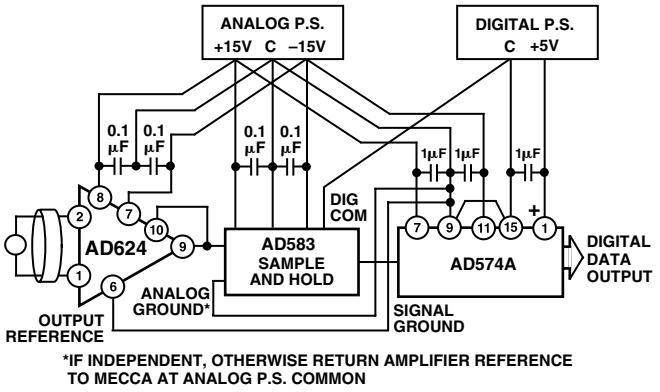


Figure 34. Basic Grounding Practice

Since the output voltage is developed with respect to the potential on the reference terminal an instrumentation amplifier can solve many grounding problems.

SENSE TERMINAL

The sense terminal is the feedback point for the instrument amplifier's output amplifier. Normally it is connected to the instrument amplifier output. If heavy load currents are to be drawn through long leads, voltage drops due to current flowing through lead resistance can cause errors. The sense terminal can be wired to the instrument amplifier at the load thus putting the IxR drops "inside the loop" and virtually eliminating this error source.

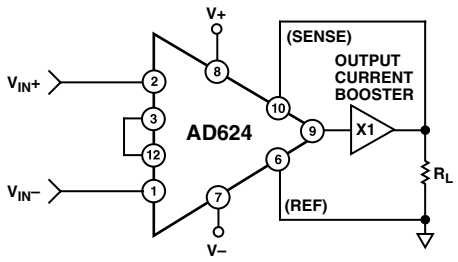


Figure 35. AD624 Instrumentation Amplifier with Output Current Booster

Typically, IC instrumentation amplifiers are rated for a full ±10 volt output swing into 2 kΩ. In some applications, however, the need exists to drive more current into heavier loads. Figure 35 shows how a current booster may be connected "inside the loop" of an instrumentation amplifier to provide the required current without significantly degrading overall performance. The effects of nonlinearities, offset and gain inaccuracies of the buffer are reduced by the loop gain of the IA output amplifier. Offset drift of the buffer is similarly reduced.

REFERENCE TERMINAL

The reference terminal may be used to offset the output by up to ±10 V. This is useful when the load is "floating" or does not share a ground with the rest of the system. It also provides a direct means of injecting a precise offset. It must be remembered that the total output swing is ±10 volts, from ground, to be shared between signal and reference offset.

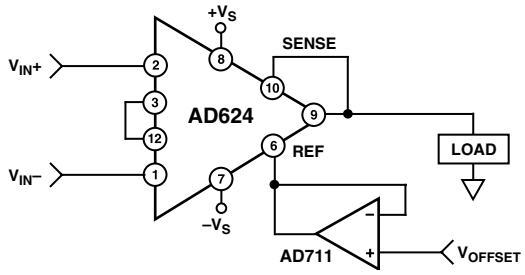


Figure 36. Use of Reference Terminal to Provide Output Offset

When the IA is of the three-amplifier configuration it is necessary that nearly zero impedance be presented to the reference terminal. Any significant resistance, including those caused by PC layouts or other connection techniques, which appears between the reference pin and ground will increase the gain of the noninverting signal path, thereby upsetting the common-mode rejection of the IA. Inadvertent thermocouple connections created in the sense and reference lines should also be avoided as they will directly affect the output offset voltage and output offset voltage drift.

In the AD624 a reference source resistance will unbalance the CMR trim by the ratio of 10 kΩ/R_{REF}. For example, if the reference source impedance is 1 Ω, CMR will be reduced to 80 dB (10 kΩ/1 Ω = 80 dB). An operational amplifier may be used to provide that low impedance reference point as shown in Figure 36. The input offset voltage characteristics of that amplifier will add directly to the output offset voltage performance of the instrumentation amplifier.

An instrumentation amplifier can be turned into a voltage-to-current converter by taking advantage of the sense and reference terminals as shown in Figure 37.

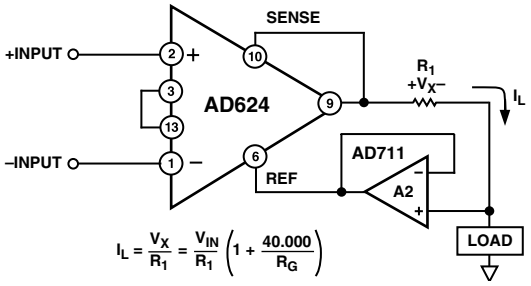
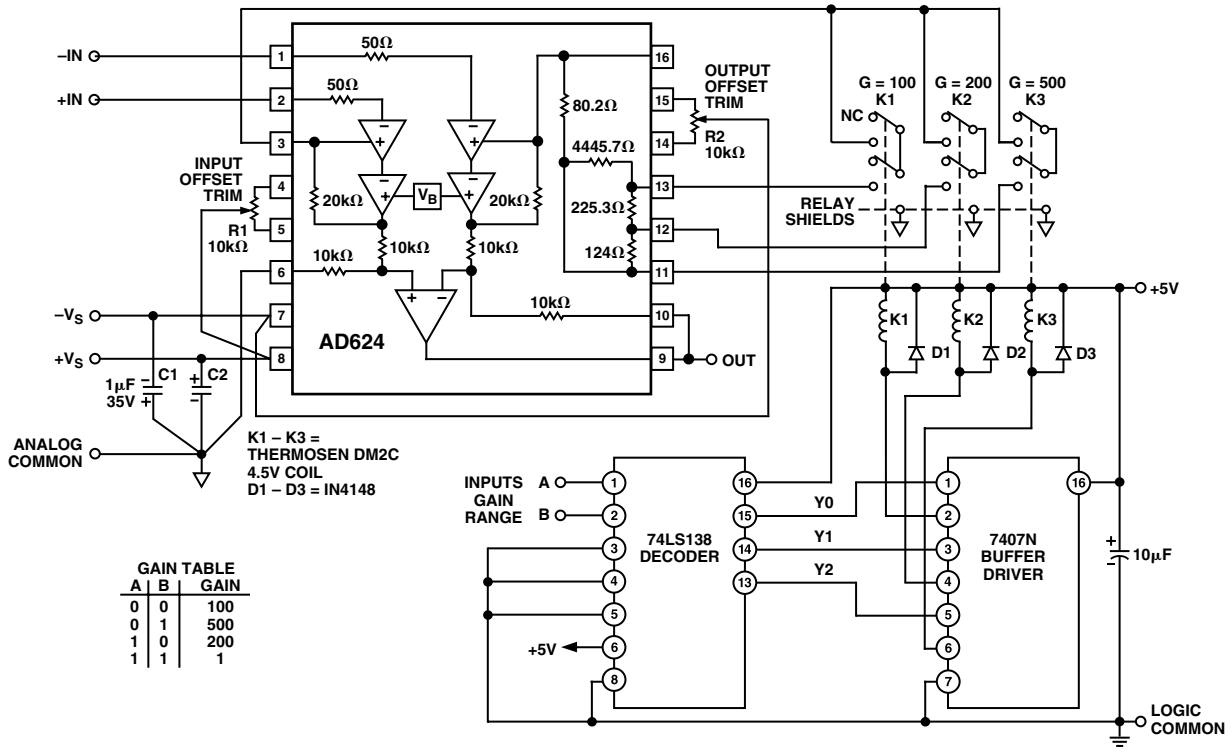


Figure 37. Voltage-to-Current Converter



GAIN TABLE

A	B	GAIN
0	0	100
0	1	500
1	0	200
1	1	1

Figure 38. Gain Programmable Amplifier

By establishing a reference at the "low" side of a current setting resistor, an output current may be defined as a function of input voltage, gain and the value of that resistor. Since only a small current is demanded at the input of the buffer amplifier A2, the forced current I_L will largely flow through the load. Offset and drift specifications of A2 must be added to the output offset and drift specifications of the IA.

PROGRAMMABLE GAIN

Figure 38 shows the AD624 being used as a software programmable gain amplifier. Gain switching can be accomplished with mechanical switches such as DIP switches or reed relays. It should be noted that the "on" resistance of the switch in series with the internal gain resistor becomes part of the gain equation and will have an effect on gain accuracy.

A significant advantage in using the internal gain resistors in a programmable gain configuration is the minimization of thermocouple signals which are often present in multiplexed data acquisition systems.

If the full performance of the AD624 is to be achieved, the user must be extremely careful in designing and laying out his circuit to minimize the remaining thermocouple signals.

The AD624 can also be connected for gain in the output stage. Figure 39 shows an AD547 used as an active attenuator in the output amplifier's feedback loop. The active attenuation presents a very low impedance to the feedback resistors therefore minimizing the common-mode rejection ratio degradation.

Another method for developing the switching scheme is to use a DAC. The AD7528 dual DAC which acts essentially as a pair of switched resistive attenuators having high analog linearity and symmetrical bipolar transmission is ideal in this application. The multiplying DAC's advantage is that it can handle inputs of either polarity or zero without affecting the programmed gain. The circuit shown uses an AD7528 to set the gain (DAC A) and to perform a fine adjustment (DAC B).

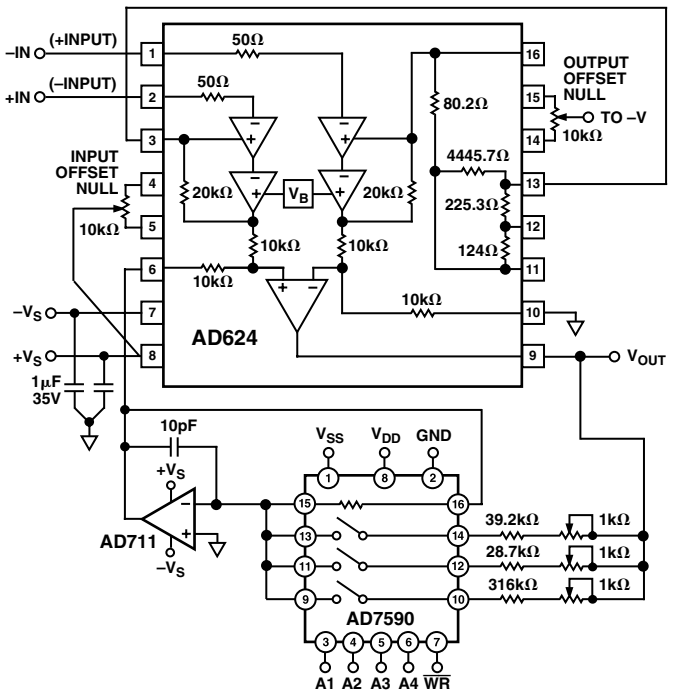


Figure 39. Programmable Output Gain

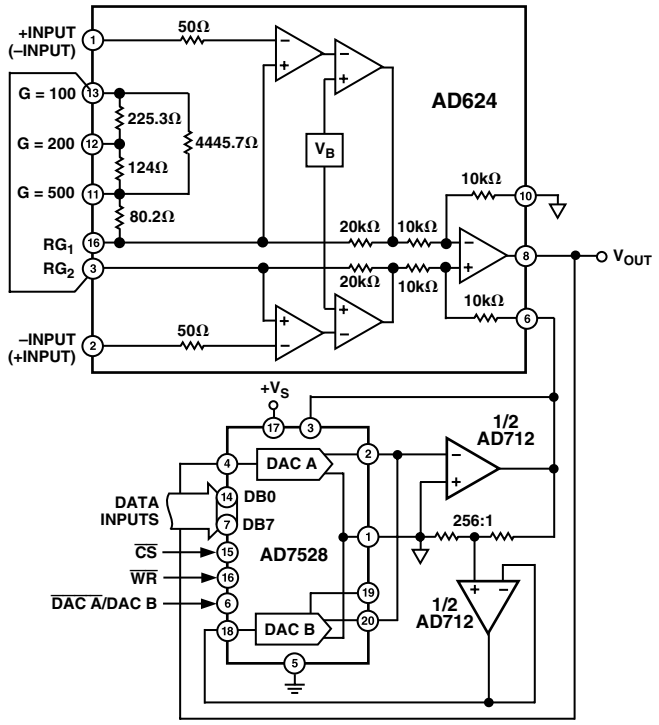


Figure 40. Programmable Output Gain Using a DAC

AUTOZERO CIRCUITS

In many applications it is necessary to provide very accurate data in high gain configurations. At room temperature the offset effects can be nulled by the use of offset trimpots. Over the operating temperature range, however, offset nulling becomes a problem. The circuit of Figure 41 shows a CMOS DAC operating in the bipolar mode and connected to the reference terminal to provide software controllable offset adjustments.

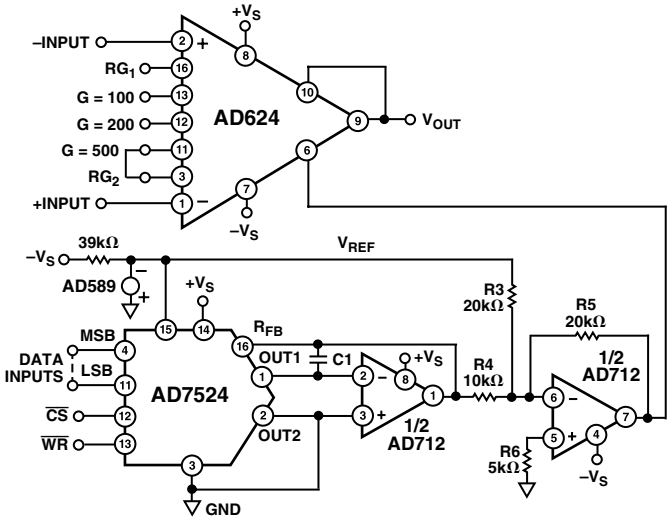


Figure 41. Software Controllable Offset

In many applications complex software algorithms for autozero applications are not available. For these applications Figure 42 provides a hardware solution.

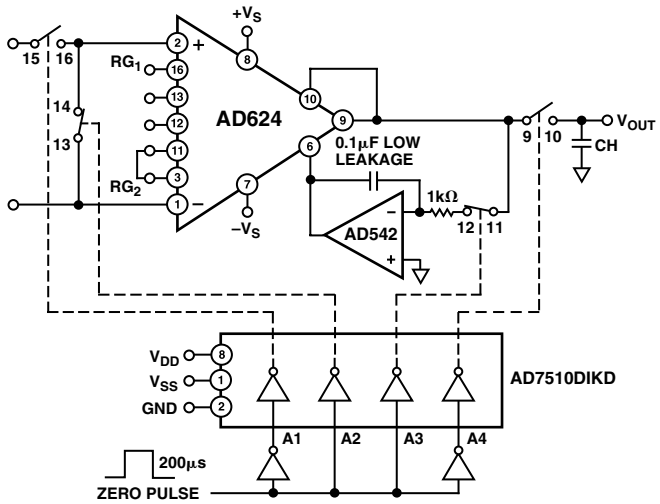


Figure 42. Autozero Circuit

The microprocessor controlled data acquisition system shown in Figure 43 includes includes both autozero and autogain capability. By dedicating two of the differential inputs, one to ground and one to the A/D reference, the proper program calibration cycles can eliminate both initial accuracy errors and accuracy errors over temperature. The autozero cycle, in this application, converts a number that appears to be ground and then writes that same number (8 bit) to the AD624 which eliminates the zero error since its output has an inverted scale. The autogain cycle converts the A/D reference and compares it with full scale. A multiplicative correction factor is then computed and applied to subsequent readings.

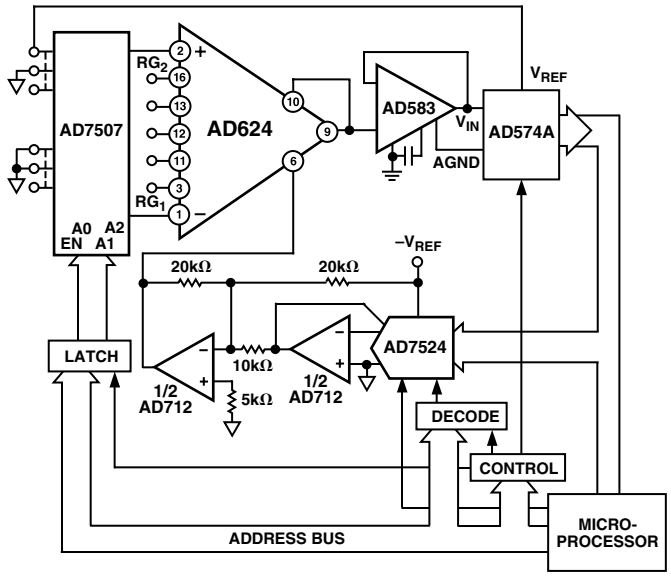


Figure 43. Microprocessor Controlled Data Acquisition System

WEIGH SCALE

Figure 44 shows an example of how an AD624 can be used to condition the differential output voltage from a load cell. The 10% reference voltage adjustment range is required to accommodate the 10% transducer sensitivity tolerance. The high linearity and low noise of the AD624 make it ideal for use in applications of this type particularly where it is desirable to measure small changes in weight as opposed to the absolute value. The addition of an autogain/autotare cycle will enable the system to remove offsets, gain errors, and drifts making possible true 14-bit performance.

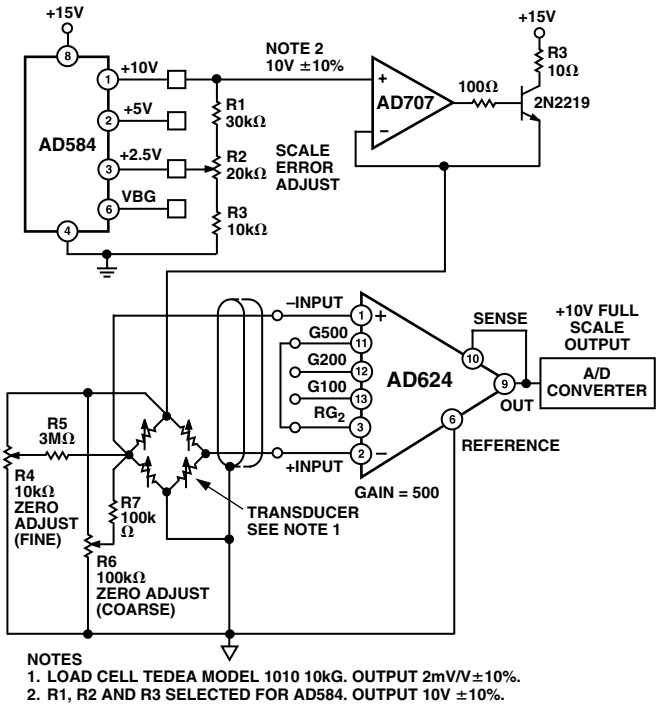


Figure 44. AD624 Weigh Scale Application

AC BRIDGE

Bridge circuits which use dc excitation are often plagued by errors caused by thermocouple effects, l/f noise, dc drifts in the electronics, and line noise pickup. One way to get around these problems is to excite the bridge with an ac waveform, amplify the bridge output with an ac amplifier, and synchronously demodulate the resulting signal. The ac phase and amplitude information from the bridge is recovered as a dc signal at the output of the synchronous demodulator. The low frequency system noise, dc drifts, and demodulator noise all get mixed to the carrier frequency and can be removed by means of a low-pass filter. Dynamic response of the bridge must be traded off against the amount of attenuation required to adequately suppress these residual carrier components in the selection of the filter.

Figure 45 is an example of an ac bridge system with the AD630 used as a synchronous demodulator. The oscilloscope photograph shows the results of a 0.05% bridge imbalance caused by the 1 Meg resistor in parallel with one leg of the bridge. The top trace represents the bridge excitation, the upper middle trace is the amplified bridge output, the lower-middle trace is the output of the synchronous demodulator and the bottom trace is the filtered dc system output.

This system can easily resolve a 0.5 ppm change in bridge impedance. Such a change will produce a 6.3 mV change in the low-pass filtered dc output, well above the RTO drifts and noise.

The AC-CMRR of the AD624 decreases with the frequency of the input signal. This is due mainly to the package-pin capacitance associated with the AD624's internal gain resistors. If AC-CMRR is not sufficient for a given application, it can be trimmed by using a variable capacitor connected to the amplifier's RG_2 pin as shown in Figure 45.

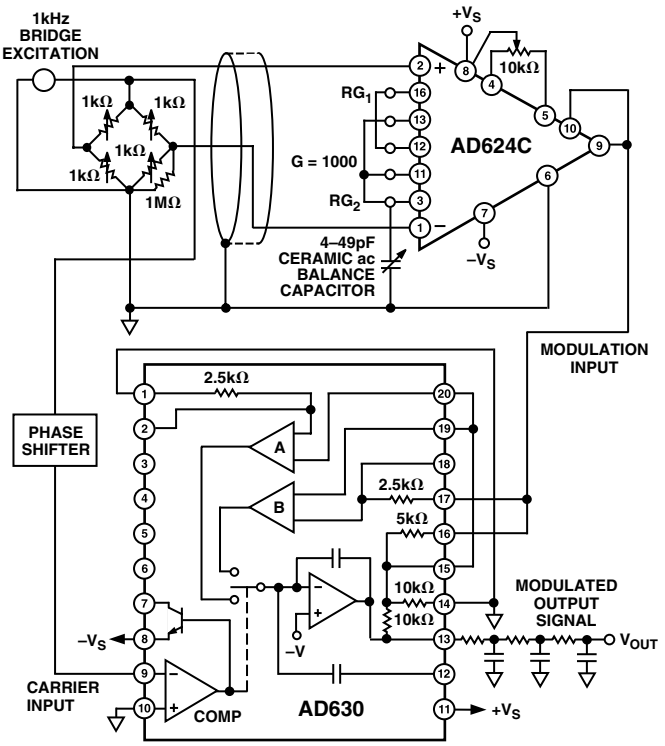


Figure 45. AC Bridge

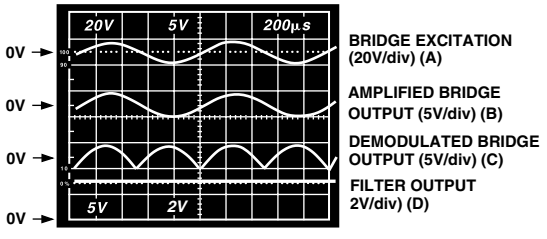


Figure 46. AC Bridge Waveforms

ERROR BUDGET ANALYSIS

To illustrate how instrumentation amplifier specifications are applied, we will now examine a typical case where an AD624 is required to amplify the output of an unbalanced transducer. Figure 47 shows a differential transducer, unbalanced by ≈5 Ω, supplying a 0 to 20 mV signal to an AD624C. The output of the IA feeds a 14-bit A to D converter with a 0 to 2 volt input voltage range. The operating temperature range is –25°C to +85°C. Therefore, the largest change in temperature ΔT within the operating range is from ambient to +85°C (85°C – 25°C = 60°C.)

In many applications, differential linearity and resolution are of prime importance. This would be so in cases where the absolute value of a variable is less important than changes in value. In these applications, only the irreducible errors (20 ppm = 0.002%) are significant. Furthermore, if a system has an intelligent processor monitoring the A to D output, the addition of an autogain/autozero cycle will remove all reducible errors and may eliminate the requirement for initial calibration. This will also reduce errors to 0.002%.

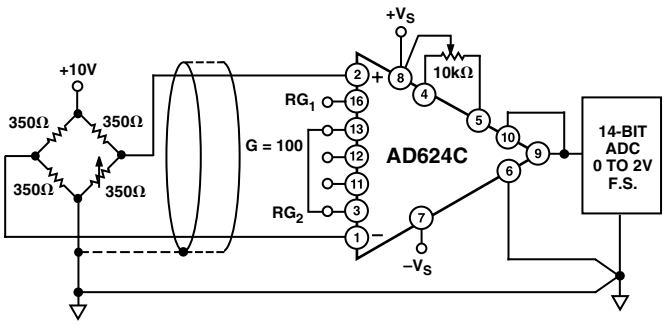


Figure 47. Typical Bridge Application

Table II. Error Budget Analysis of AD624CD in Bridge Application

Error Source	AD624C Specifications	Calculation	Effect on Absolute Accuracy at T_A = +25°C	Effect on Absolute Accuracy at T_A = +85°C	Effect on Resolution
Gain Error	±0.1%	±0.1% = 1000 ppm	1000 ppm	1000 ppm	–
Gain Instability	10 ppm	(10 ppm/°C) (60°C) = 600 ppm	_	600 ppm	–
Gain Nonlinearity	±0.001%	±0.001% = 10 ppm	–	–	10 ppm
Input Offset Voltage	±25 μV, RTI	±25 μV/20 mV = ±1250 ppm	1250 ppm	1250 ppm	–
Input Offset Voltage Drift	±0.25 μV/°C	(±0.25 μV/°C) (60°C)= 15 μV 15 μV/20 mV = 750 ppm	–	750 ppm	–
Output Offset Voltage[1]	±2.0 mV	±2.0 mV/20 mV = 1000 ppm	1000 ppm	1000 ppm	–
Output Offset Voltage Drift[1]	±10 μV/°C	(±10 μV/°C) (60°C) = 600 μV 600 μV/20 mV = 300 ppm	–	300 ppm	–
Bias Current–Source Imbalance Error	±15 nA	(±15 nA)(5 Ω) = 0.075 μV 0.075 μV/20mV = 3.75 ppm	3.75 ppm	3.75 ppm	–
Offset Current–Source Imbalance Error	±10 nA	(±10 nA)(5 Ω) = 0.050 μV 0.050 μV/20 mV = 2.5 ppm	2.5 ppm	2.5 ppm	–
Offset Current–Source Resistance Error	±10 nA	(10 nA) (175 Ω) = 1.75 μV 1.75 μV/20 mV = 87.5 ppm	87.5 ppm	87.5 ppm	–
Offset Current–Source Resistance–Drift	±100 pA/°C	(100 pA/°C) (175 Ω) (60°C) = 1 μV 1 μV/20 mV = 50 ppm	–	50 ppm	–
Common-Mode Rejection 5 V dc	115 dB	115 dB = 1.8 ppm × 5 V = 9 μV 9 μV/20 mV = 444 ppm	450 ppm	450 ppm	–
Noise, RTI (0.1 Hz–10 Hz)	0.22 μV p-p	0.22 μV p-p/20 mV = 10 ppm	_	–	10 ppm
		Total Error	3793.75 ppm	5493.75 ppm	20 ppm

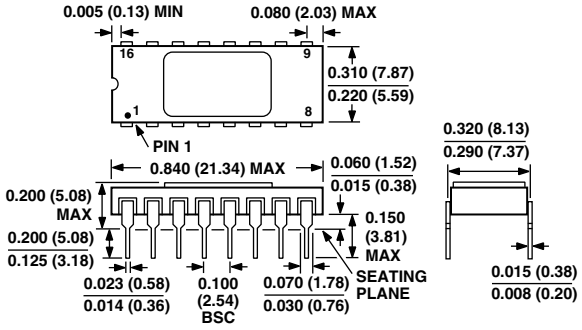
NOTE
[1]Output offset voltage and output offset voltage drift are given as RTI figures.

For a comprehensive study of instrumentation amplifier design and applications, refer to the *Instrumentation Amplifier Application Guide*, available free from Analog Devices.

OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

Side-Brazed Solder Lid Ceramic DIP
(D-16)



C805d–0–7/99

PRINTED IN U.S.A.

Dual Precision, Low Cost, High Speed BiFET Op Amp

AD712

FEATURES

- **Enhanced replacement for LF412 and TL082**
- **AC performance**
  - **Settles to ±0.01% in 1.0 µs**
  - **16 V/µs minimum slew rate (AD712J)**
  - **3 MHz minimum unity-gain bandwidth (AD712J)**
- **DC performance**
  - **200 V/mV minimum open-loop gain (AD712K)**
  - **Surface mount available in tape and reel in accordance with the EIA-481A standard**
  - **MIL-STD-883B parts available**
  - **Single version available: AD711**
  - **Quad version: AD713**
  - **Available in PDIP, SOIC_N, and CERDIP packages**

CONNECTION DIAGRAM

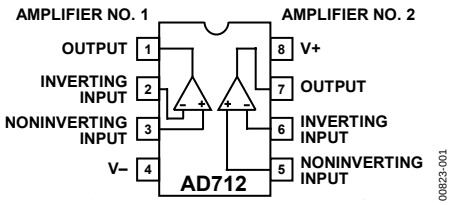


Figure 1. 8-Lead PDIP (N-Suffix), SOIC_N (R-Suffix), and CERDIP (Q-Suffix)

GENERAL DESCRIPTION

The AD712 is a high speed, precision, monolithic operational amplifier offering high performance at very modest prices. Its very low offset voltage and offset voltage drift are the results of advanced laser wafer trimming technology. These performance benefits allow the user to easily upgrade existing designs that use older precision BiFETs and, in many cases, bipolar op amps.

The superior ac and dc performance of this op amp makes it suitable for active filter applications. With a slew rate of 16 V/µs and a settling time of 1 µs to ±0.01%, the AD712 is ideal as a buffer for 12-bit digital-to-analog and analog-to-digital converters and as a high speed integrator. The settling time is unmatched by any similar IC amplifier.

The combination of excellent noise performance and low input current also make the AD712 useful for photo diode preamps. Common-mode rejection of 88 dB and open-loop gain of 400 V/mV ensure 12-bit performance even in high speed unity-gain buffer circuits.

The AD712 is pinned out in a standard op amp configuration and is available in seven performance grades. The AD712J and AD712K are rated over the commercial temperature range of 0°C to 70°C. The AD712A is rated over the industrial temperature range of −40°C to +85°C. The AD712S is rated over the military temperature range of −55°C to +125°C and is available processed to MIL-STD-883B, Rev. C.

Extended reliability PLUS screening is available, specified over the commercial and industrial temperature ranges. PLUS screening includes 168-hour burn-in, in addition to other environmental and physical tests.

The AD712 is available in 8-lead PDIP, SOIC_N, and CERDIP packages.

PRODUCT HIGHLIGHTS

1. The AD712 offers excellent overall performance at very competitive prices.
2. The Analog Devices, Inc. advanced processing technology and 100% testing guarantee a low input offset voltage (3 mV maximum, J grade). Input offset voltage is specified in the warmed-up condition.
3. Together with precision dc performance, the AD712 offers excellent dynamic response. It settles to ±0.01% in 1 µs and has a minimum slew rate of 16 V/µs. Thus, this device is ideal for applications such as DAC and ADC buffers that require a combination of superior ac and dc performance.

Rev. G
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 **©2006 Analog Devices, Inc. All rights reserved.**

TABLE OF CONTENTS

Features 1
Connection Diagram 1
General Description 1
Product Highlights 1
Revision History 2
Specifications 3
Absolute Maximum Ratings 5
ESD Caution 5
Typical Performance Characteristics 6
Settling Time 11
Optimizing Settling Time 11
Op Amp Settling Time—A Mathematical Model 12
Applications Information 14
Guarding 14
Digital-to-Analog Converter Applications 14
Noise Characteristics 15
Driving the Analog Input of an Analog-to-Digital Converter 15
Driving a Large Capacitive Load 16
Filters 17
Active Filter Applications 17
Second-Order Low-Pass Filter 17
9-Pole Chebychev Filter 18
Outline Dimensions 19
Ordering Guide 20

REVISION HISTORY

8/06—Rev. F to Rev. G

Edits to Figure 1 1
Change to 9-Pole Chebychev Filter Section 18

6/06—Rev. E to Rev. F

Updated Format ..... Universal
Deleted B, C, and T Models ..... Universal
Changes to General Description 1
Changes to Product Highlights 1
Changes to Specifications Section 3
Changes to Figure 43 15

7/02—Rev. D to Rev. E

Edits to Features 1

9/01—Rev. C to Rev. D

Edits to Features 1
Edits to General Description 1
Edits to Connection Diagram 1
Edits to Ordering Guide 3
Deleted Metalization Photograph 3
Edits to Absolute Maximum Ratings 3
Edits to Figure 7 9
Edits to Outline Dimensions 15

SPECIFICATIONS

V_S = ±15 V @ T_A = 25°C, unless otherwise noted. Specifications in **boldface** are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in **boldface** are tested on all production units.

Table 1.

	AD712J/A/S			AD712K			
Parameter	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
INPUT OFFSET VOLTAGE[1]							
Initial Offset		0.3	**3/1/1**		0.2	**1.0**	mV
T_{MIN} to T_{MAX}			4/2/**2**			**2.0**	mV
vs. Temp		7	20/20/**20**		7	**10**	µV/°C
vs. Supply	**76**	95		**80**	100		dB
T_{MIN} to T_{MAX}	76/76/**76**			**80**			dB
Long-Term Offset Stability		15			15		µV/month
INPUT BIAS CURRENT[2]							
V_{CM} = 0 V		25	**75**		20	**75**	pA
V_{CM} = 0 V @ T_{MAX}		0.6/1.6/26	1.7/4.8/77		0.5	1.7	nA
V_{CM} = ±10 V			**100**			**100**	pA
INPUT OFFSET CURRENT							
V_{CM} = 0 V		10	**25**		5	**25**	pA
V_{CM} = 0 V @ T_{MAX}		0.3/0.7/11	0.6/1.6/26		0.1	0.6	nA
MATCHING CHARACTERISTICS							
Input Offset Voltage			**3/1/1**			**1.0**	mV
T_{MIN} to T_{MAX}			4/2/**2**			**2.0**	mV
Input Offset Voltage Drift			20/20/**20**			**10**	µV/°C
Input Bias Current			**25**			**25**	pA
Crosstalk							
@ f = 1 kHz		120			120		dB
@ f = 100 kHz		90			90		dB
FREQUENCY RESPONSE							
Small Signal Bandwidth	3.0	4.0		3.4	4.0		MHz
Full Power Response		200			200		kHz
Slew Rate	**16**	20		**18**	20		V/µs
Settling Time to 0.01%		1.0	1.2		1.0	1.2	µs
Total Harmonic Distortion		0.0003			0.0003		%
INPUT IMPEDANCE							
Differential		3×10^{12}||5.5			3×10^{12}||5.5		Ω||pF
Common Mode		3×10^{12}||5.5			3×10^{12}||5.5		Ω||pF
INPUT VOLTAGE RANGE							
Differential[3]		±20			±20		V
Common-Mode Voltage[4]		+14.5, −11.5			+14.5, −11.5		V
T_{MIN} to T_{MAX}	**$-V_S$ + 4**		**$+V_S$ − 2**	**$-V_S$ + 4**		**$+V_S$ − 2**	V
Common-Mode Rejection Ratio							
V_{CM} = ±10 V	**76**	88		**80**	88		dB
T_{MIN} to T_{MAX}	76/76/**76**	84		**80**	84		dB
V_{CM} = ±11 V	**70**	84		**76**	84		dB
T_{MIN} to T_{MAX}	70/70/**70**	80		**74**	80		dB
INPUT VOLTAGE NOISE		2			2		µV p-p
		45			45		nV/√Hz
		22			22		nV/√Hz
		18			18		nV/√Hz
		16			16		nV/√Hz

Parameter	AD712J/A/S Min	AD712J/A/S Typ	AD712J/A/S Max	AD712K Min	AD712K Typ	AD712K Max	Unit
INPUT CURRENT NOISE		0.01			0.01		pA/√Hz
OPEN-LOOP GAIN	**150**	400		**200**	400		V/mV
	100/100/**100**			**100**			V/mV
OUTPUT CHARACTERISTICS							
Voltage	**+13, −12.5**	+13.9, −13.3		**+13, −12.5**	+13.9, −13.3		V
	±12/±12/**±12**	+13.8, −13.1		**±12**	+13.8, −13.1		V
Current		+25			+25		mA
POWER SUPPLY							
Rated Performance		±15			±15		V
Operating Range	**±4.5**		**±18**	**±4.5**		**±18**	V
Quiescent Current		+5.0	**+6.8**		+5.0	**+6.0**	mA

[1] Input offset voltage specifications are guaranteed after 5 minutes of operation at T_A = 25°C.
[2] Bias current specifications are guaranteed maximum at either input after 5 minutes of operation at T_A = 25°C. For higher temperatures, the current doubles every 10°C.
[3] Defined as voltage between inputs, such that neither exceeds ±10 V from ground.
[4] Typically exceeding −14.1 V negative common-mode voltage on either input results in an output phase reversal.

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Supply Voltage	±18 V
Internal Power Dissipation[1]	
Input Voltage[2]	±18 V
Output Short-Circuit Duration	Indefinite
Differential Input Voltage	$+V_S$ and $-V_S$
Storage Temperature Range	
Q-Suffix	−65°C to +150°C
N-Suffix and R-Suffix	−65°C to +125°C
Operating Temperature Range	
AD712J/K	0°C to 70°C
AD712A	−40°C to +85°C
AD712S	−55°C to +125°C
Lead Temperature Range (Soldering 60 sec)	300°C

[1] Thermal characteristics:
8-lead PDIP package: θ_{JA} = 165°C/W
8-lead CERDIP package: θ_{JC} = 22°C/W; θ_{JA} = 110°C/W
8-lead SOIC package: θ_{JA} = 100°C/W

[2] For supply voltages less than ±18 V, the absolute maximum voltage is equal to the supply voltage.

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although this product features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.



TYPICAL PERFORMANCE CHARACTERISTICS

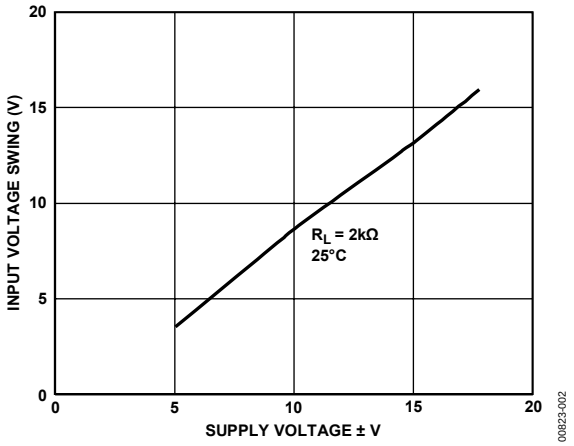


Figure 2. Input Voltage Swing vs. Supply Voltage

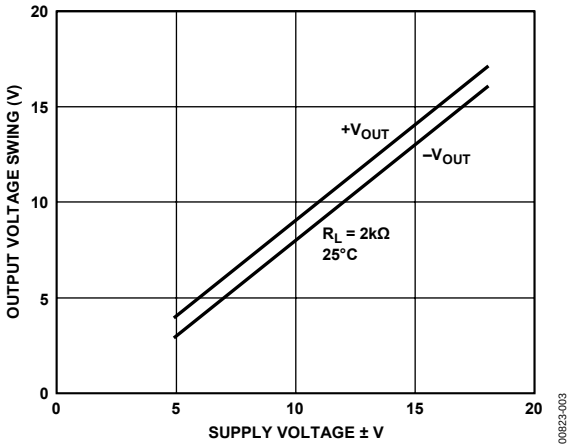


Figure 3. Output Voltage Swing vs. Supply Voltage

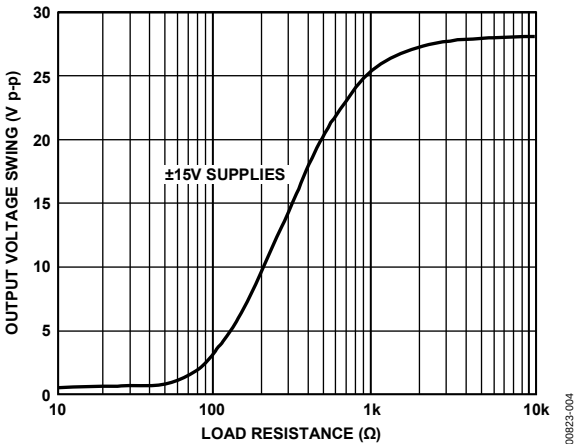


Figure 4. Output Voltage Swing vs. Load Resistance

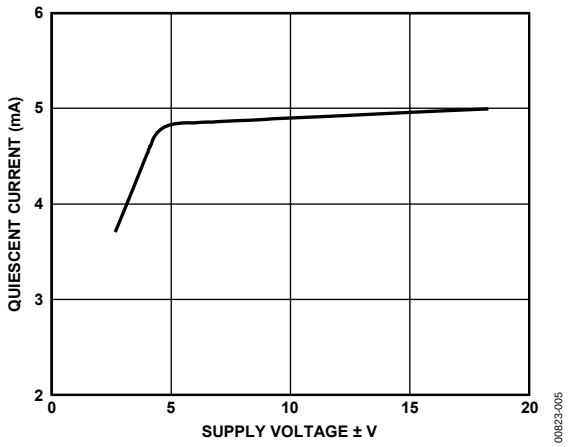


Figure 5. Quiescent Current vs. Supply Voltage

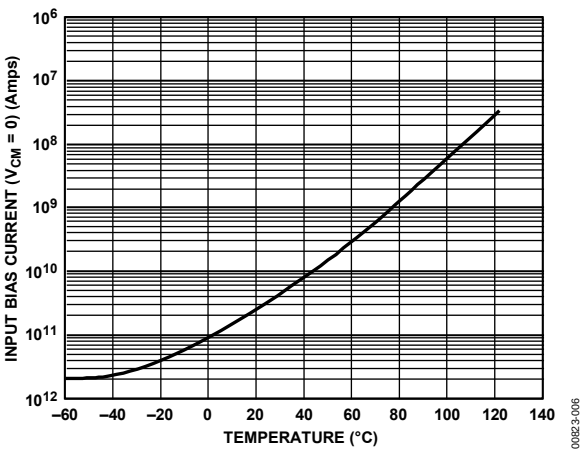


Figure 6. Input Bias Current vs. Temperature

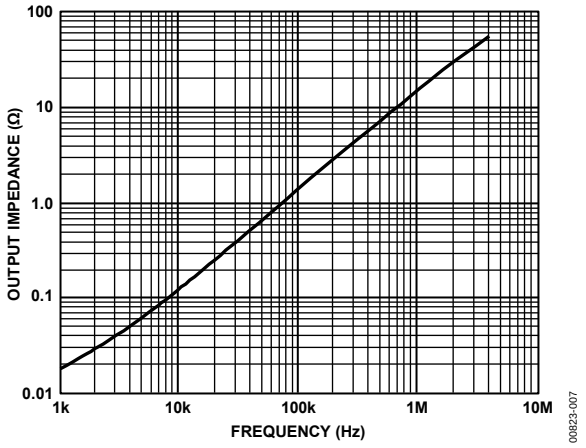


Figure 7. Output Impedance vs. Frequency

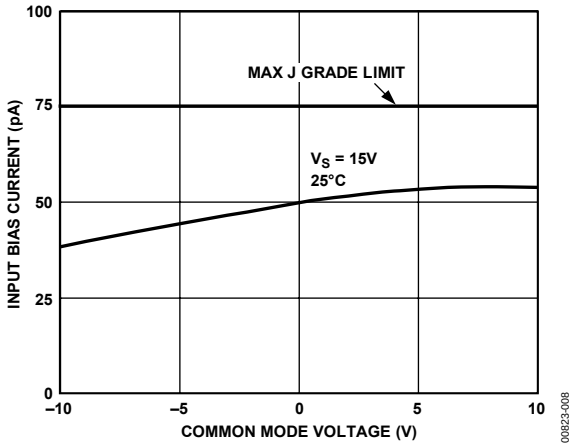


Figure 8. Input Bias Current vs. Common-Mode Voltage

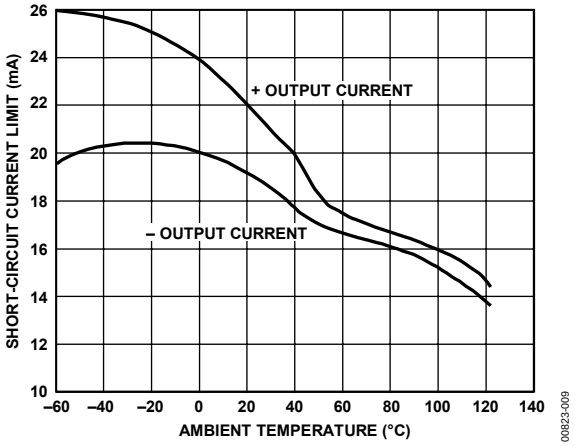


Figure 9. Short-Circuit Current Limit vs. Temperature

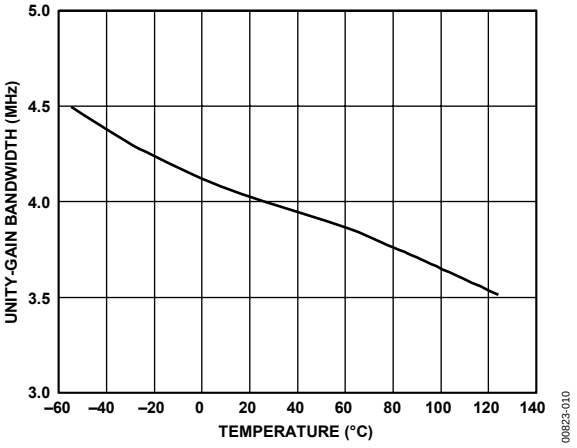


Figure 10. Unity-Gain Bandwidth vs. Temperature

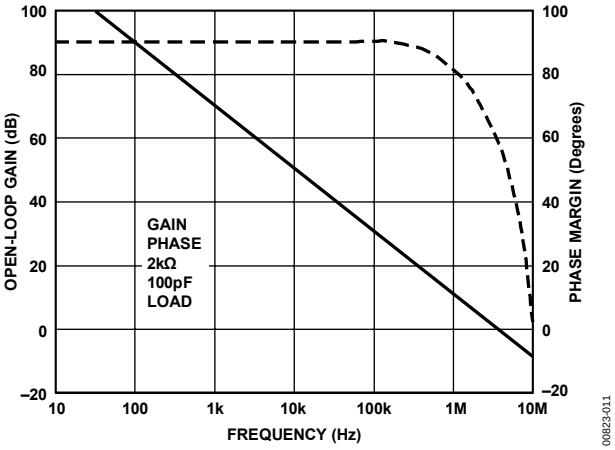


Figure 11. Open-Loop Gain and Phase Margin vs. Frequency

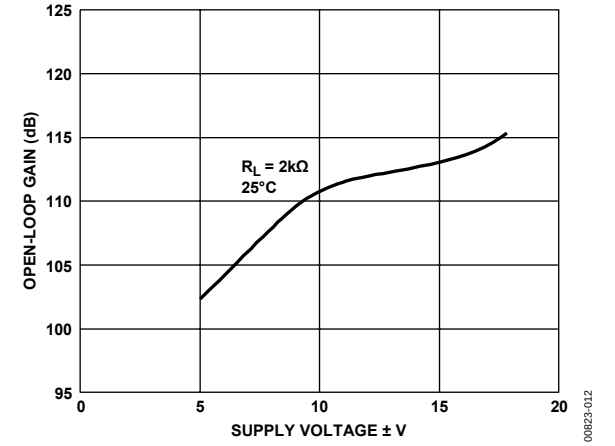


Figure 12. Open-Loop Gain vs. Supply Voltage

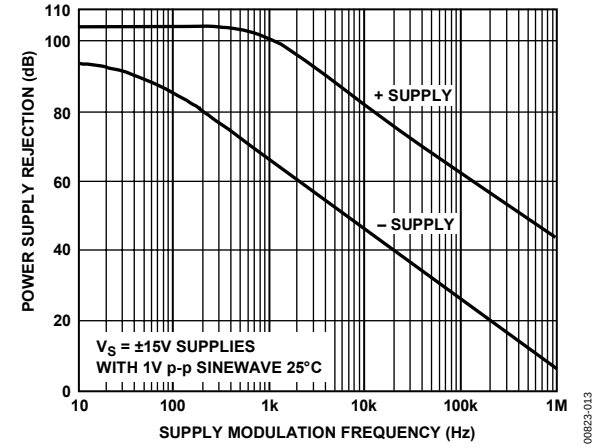


Figure 13. Power Supply Rejection vs. Frequency

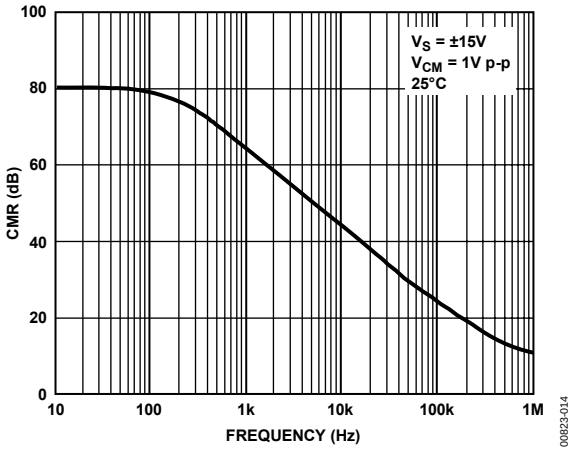


Figure 14. Common-Mode Rejection vs. Frequency

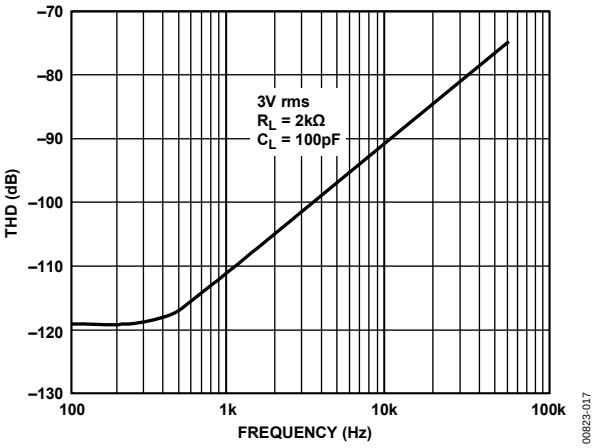


Figure 17. Total Harmonic Distortion vs. Frequency

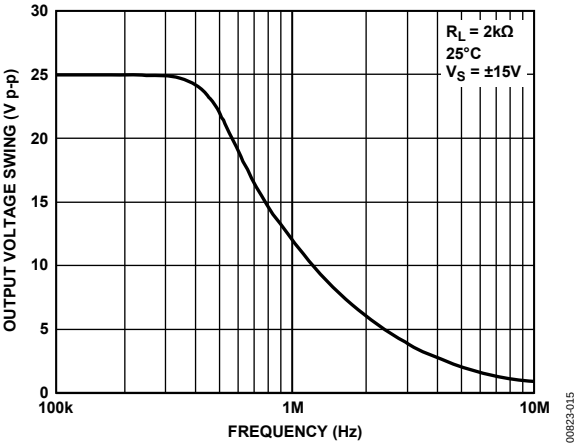


Figure 15. Large Signal Frequency Response

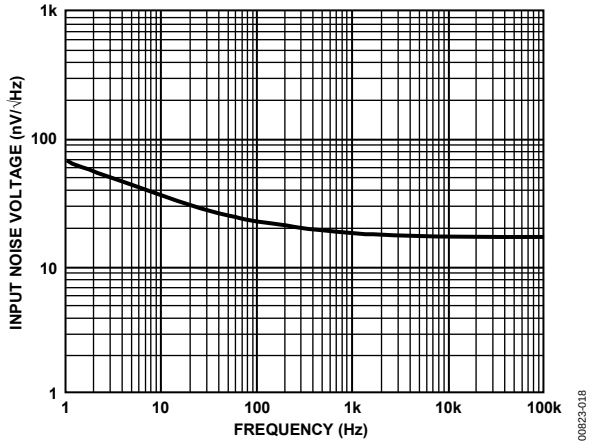


Figure 18. Input Noise Voltage Spectral Density

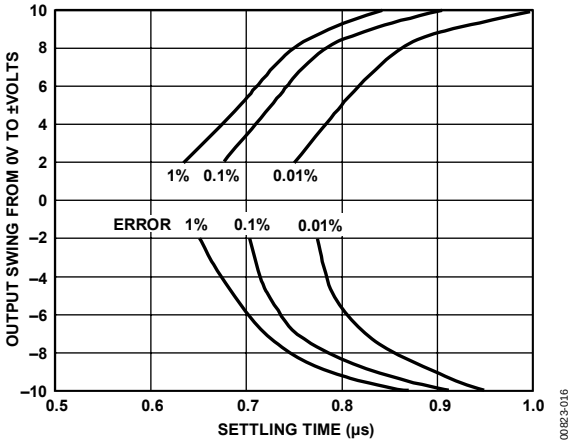


Figure 16. Output Swing and Error vs. Settling Time

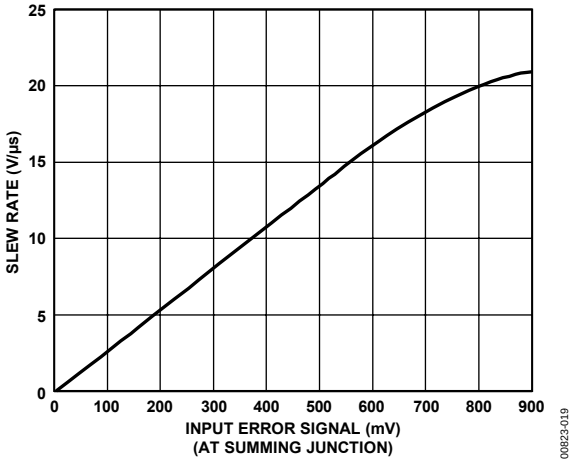


Figure 19. Slew Rate vs. Input Error Signal

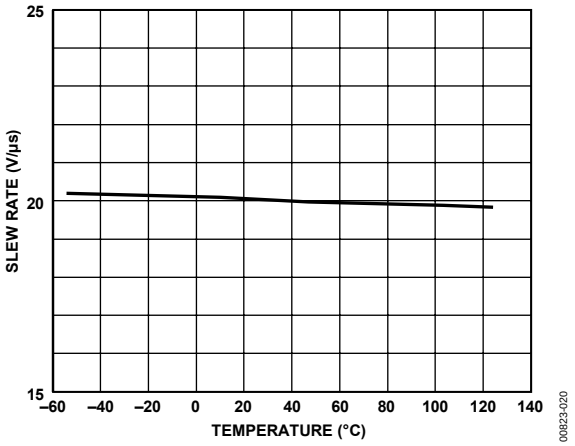


Figure 20. Slew Rate vs. Temperature

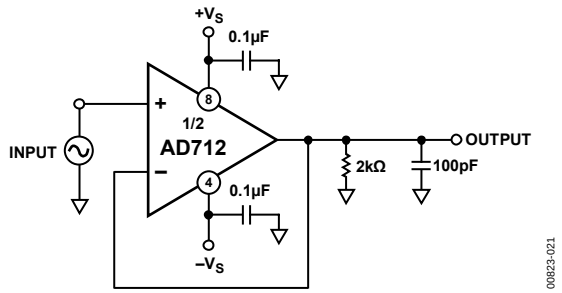


Figure 21. THD Test Circuit

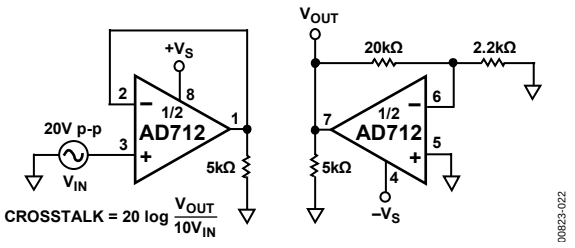


Figure 22. Crosstalk Test Circuit

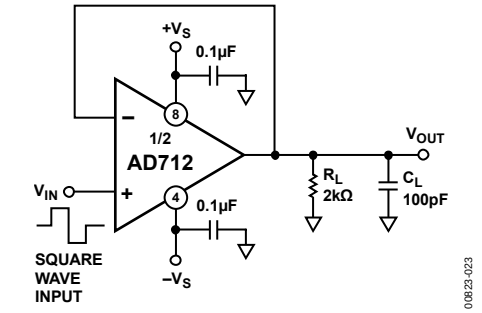


Figure 23. Unity-Gain Follower

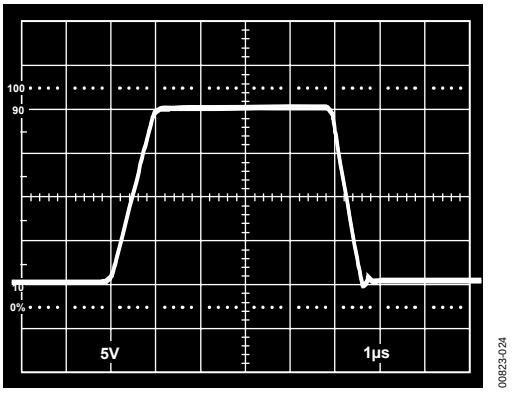


Figure 24. Unity-Gain Follower Pulse Response (Large Signal)

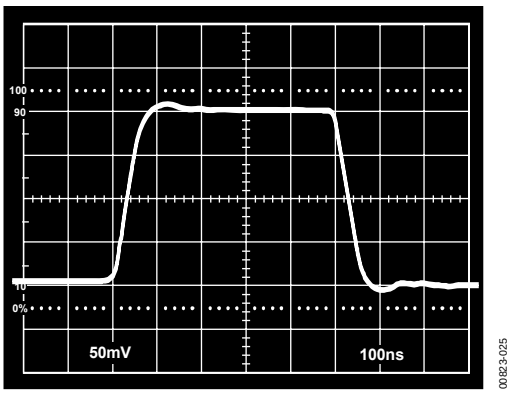


Figure 25. Unity-Gain Follower Pulse Response (Small Signal)

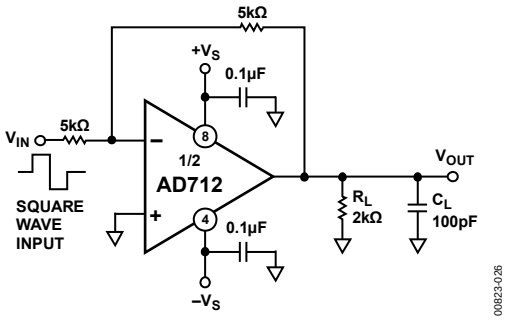


Figure 26. Unity-Gain Inverter

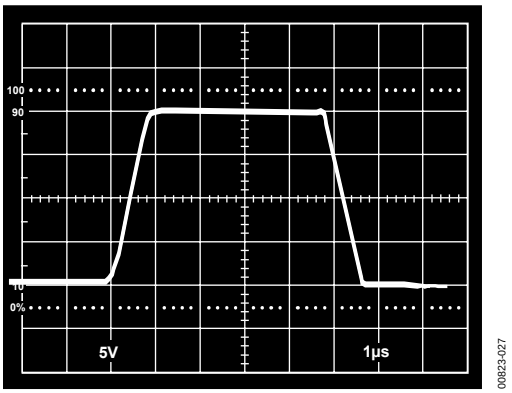


Figure 27. Unity-Gain Inverter Pulse Response (Large Signal)

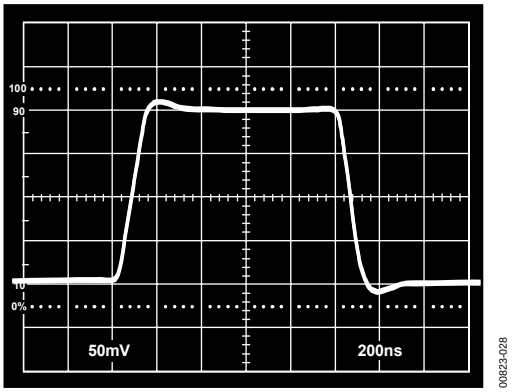


Figure 28. Unity-Gain Inverter Pulse Response (Small Signal)

SETTLING TIME

OPTIMIZING SETTLING TIME

Most bipolar high speed digital-to-analog converters (DACs) have current outputs; therefore, for most applications, an external op amp is required for a current-to-voltage conversion. The settling time of the converter/op amp combination depends on the settling time of the DAC and output amplifier. A good approximation is

$$t_S\,Total = \sqrt{(t_S\,DAC)^2 + (t_S\,AMP)^2}$$

The settling time of an op amp DAC buffer varies with the noise gain of the circuit, the DAC output capacitance, and the amount of external compensation capacitance across the DAC output scaling resistor.

Settling time for a bipolar DAC is typically 100 ns to 500 ns. Previously, conventional op amps have required much longer settling times than have typical state-of-the-art DACs; therefore, the amplifier settling time has been the major limitation to a high speed, voltage output, digital-to-analog function. The introduction of the AD71x family of op amps with their 1 µs (to ±0.01% of final value) settling time permits the full high speed capabilities of most modern DACs to be realized.

In addition to a significant improvement in settling time, the low offset voltage, low offset voltage drift, and high open-loop gain of the AD71x family assure 12-bit accuracy over the full operating temperature range.

The excellent high speed performance of the AD712 is shown in the oscilloscope photos in Figure 30 and Figure 31. Measurements were taken using a low input capacitance amplifier connected directly to the summing junction of the AD712 and both figures show a worst-case situation: full-scale input transition. The 4 kΩ [10 kΩ||8 kΩ = 4.4 kΩ] output impedance of the DAC, together with a 10 kΩ feedback resistor, produce an op amp noise gain of 3.25. The current output from the DAC produces a 10 V step at the op amp output (0 to −10 V shown in Figure 30, and −10 V to 0 V shown in Figure 31).

Therefore, with an ideal op amp, settling to ±1/2 LSB (±0.01%) requires that 375 µV or less appears at the summing junction. This means that the error between the input and output (that voltage which appears at the AD712 summing junction) must be less than 375 µV. As shown in Figure 30, the total settling time for the AD712/AD565A combination is 1.2 microseconds.

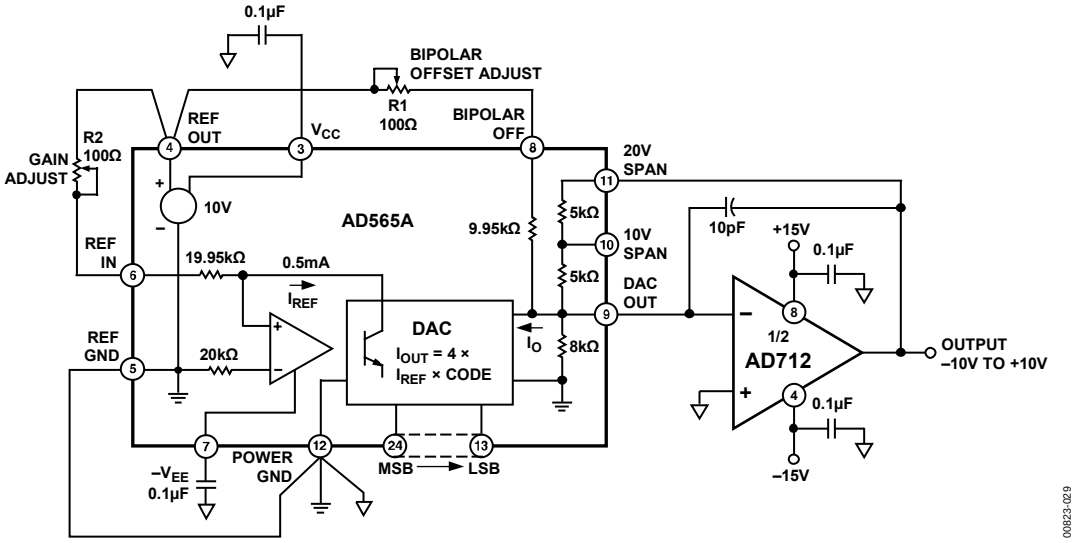


Figure 29. ±10 V Voltage Output Bipolar DAC

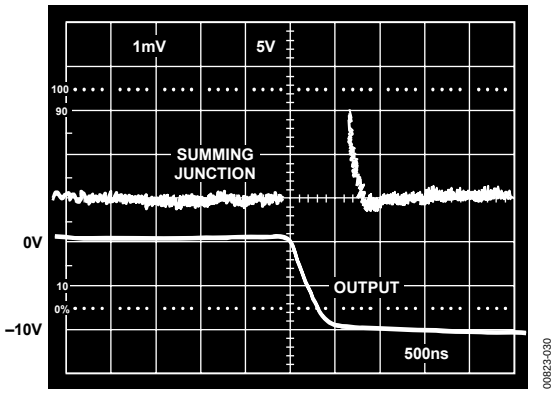


Figure 30. Settling Characteristics for AD712 with AD565A, Full-Scale Negative Transition

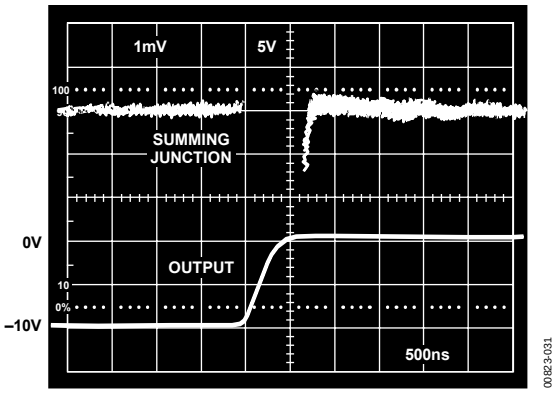


Figure 31. Settling Characteristics for AD712 with AD565A, Full-Scale Positive Transition

OP AMP SETTLING TIME—A MATHEMATICAL MODEL

The design of the AD712 gives careful attention to optimizing individual circuit components; in addition, a careful trade-off was made: the gain bandwidth product (4 MHz) and slew rate (20 V/μs) were chosen to be high enough to provide very fast settling time but not too high to cause a significant reduction in phase margin (and therefore, stability). Thus designed, the AD712 settles to ±0.01%, with a 10 V output step, in under 1 μs, while retaining the ability to drive a 250 pF load capacitance when operating as a unity-gain follower.

If an op amp is modeled as an ideal integrator with a unity-gain crossover frequency of $\omega_O/2\pi$, then Equation 1 accurately describes the small signal behavior of the circuit of Figure 32, consisting of an op amp connected as an I-to-V converter at the output of a bipolar or CMOS DAC. This equation would completely describe the output of the system if not for the finite slew rate and other nonlinear effects of the op amp.

$$\frac{V_O}{I_{IN}} = \frac{-R}{\frac{R(C_X)}{\omega_O}s^2 + \left(\frac{G_N}{\omega_O} + RC_f\right)s + 1} \quad (1)$$

Where

$\frac{\omega_O}{2\pi}$ = unity-gain frequency of the op amp.

G_N = noise gain of circuit $\left(1+\frac{R}{R_O}\right)$.

This equation can then be solved for C_f

$$C_X = \frac{2 - G_N}{R\omega_O} + 2\sqrt{\frac{RC_X\omega_O + (1 - G_N)}{R\omega_O}} \quad (2)$$

In these equations, Capacitance C_X is the total capacitance appearing at the inverting terminal of the op amp. When modeling a DAC buffer application, the Norton equivalent circuit shown in Figure 32 can be used directly; Capacitance C_X is the total capacitance of the output of the DAC plus the input capacitance of the op amp (because the two are in parallel).

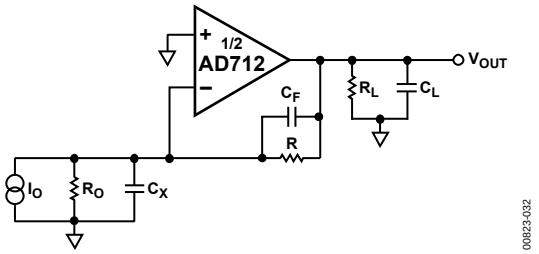


Figure 32. Simplified Model of the AD712 Used as a Current-Out DAC Buffer

When R_O and I_O are replaced with their Thevenin V_{IN} and R_{IN} equivalents, the general-purpose inverting amplifier shown in Figure 33 is created. Note that when using this general model, Capacitance C_X is either the input capacitance of the op amp, if a simple inverting op amp is being simulated, or the combined capacitance of the DAC output and the op amp input if the DAC buffer is being modeled.

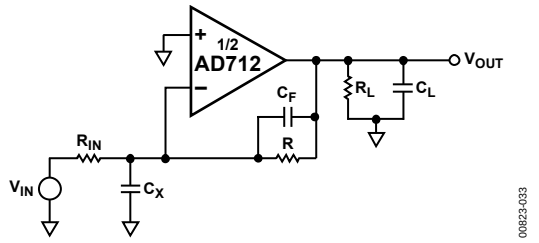


Figure 33. Simplified Model of the AD712 Used as an Inverter

In either case, Capacitance C_X causes the system to go from a one-pole to a two-pole response; this additional pole increases settling time by introducing peaking or ringing in the op amp output. Because the value of C_X can be estimated with reasonable accuracy, Equation 2 can be used to choose a small capacitor (C_F) to cancel the input pole and optimize amplifier response. Figure 34 is a graphical solution of Equation 2 for the AD712 with R = 4 kΩ.

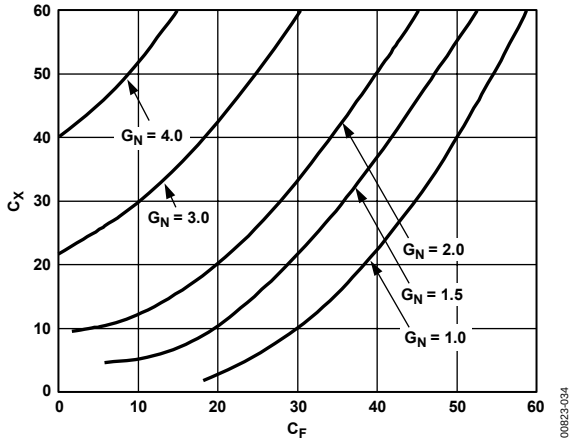


Figure 34. Value of Capacitor C_F vs. Value of C_X

The photos of Figure 35 and Figure 36 show the dynamic response of the AD712 in the settling test circuit of Figure 37.

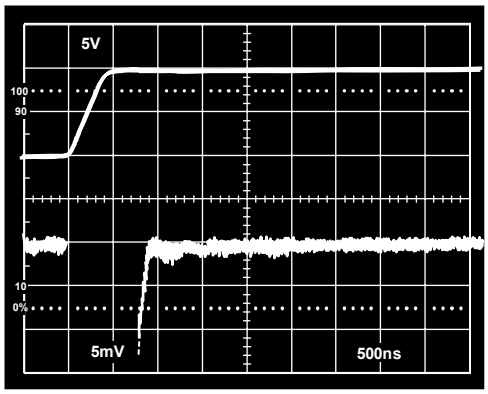


Figure 35. Settling Characteristics 0 V to +10 V Step
Upper Trace: Output of AD712 Under Test (5 V/Div)
Lower Trace: Amplified Error Voltage (0.01%/Div)

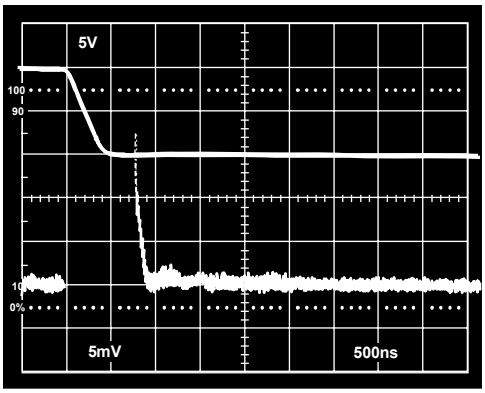


Figure 36. Settling Characteristics 0 V to −10 V Step
Upper Trace: Output of AD712 Under Test (5 V/Div)
Lower Trace: Amplified Error Voltage (0.01%/Div)

The input of the settling time fixture is driven by a flat top pulse generator. The error signal output from the false summing node of A1 is clamped, amplified by A2, and then clamped again. The error signal is thus clamped twice: once to prevent overloading Amplifier A2 and then a second time to avoid overloading the oscilloscope preamp. The Tektronix oscilloscope preamp type 7A26 was carefully chosen because it does not overload with these input levels. Amplifier A2 needs to be a very high speed FET-input op amp; it provides a gain of 10, amplifying the error signal output of A1.

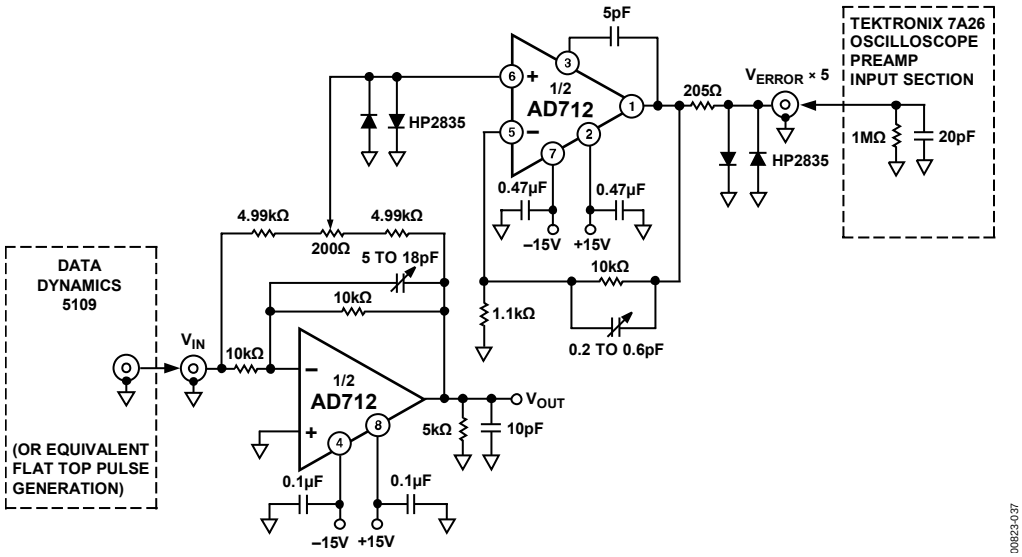


Figure 37. Settling Time Test Circuit

APPLICATIONS INFORMATION

GUARDING

The low input bias current (15 pA) and low noise characteristics of the AD712 BiFET op amp make it suitable for electrometer applications such as photo diode preamplifiers and picoampere current-to-voltage converters. The use of a guarding technique, such as that shown in Figure 38, in printed circuit board layout and construction is critical to minimize leakage currents. The guard ring is connected to a low impedance potential at the same level as the inputs. High impedance signal lines should not be extended for any unnecessary length on the printed circuit board.

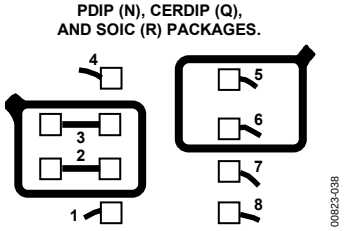


Figure 38. Board Layout for Guarding Inputs

DIGITAL-TO-ANALOG CONVERTER APPLICATIONS

The AD712 is an excellent output amplifier for CMOS DACs. It can be used to perform both 2-quadrant and 4-quadrant operations. The output impedance of a DAC using an inverted R-2R ladder approaches R for codes containing many 1s, and 3R for codes containing a single 1. For codes containing all 0s, the output impedance is infinite.

For example, the output resistance of the AD7545 modulates between 11 kΩ and 33 kΩ. Therefore, with an 11 kΩ DAC internal feedback resistance, the noise gain varies from 2 to 4/3. This changing noise gain modulates the effect of the input offset voltage of the amplifier, resulting in nonlinear DAC amplifier performance.

The AD712K with guaranteed 700 μV offset voltage minimizes this effect to achieve 12-bit performance.

Figure 39 and Figure 40 show the AD712 and AD7545 (12-bit CMOS DAC) configured for unipolar binary (2-quadrant multiplication) or bipolar (4-quadrant multiplication) operation. Capacitor C1 provides phase compensation to reduce overshoot and ringing.

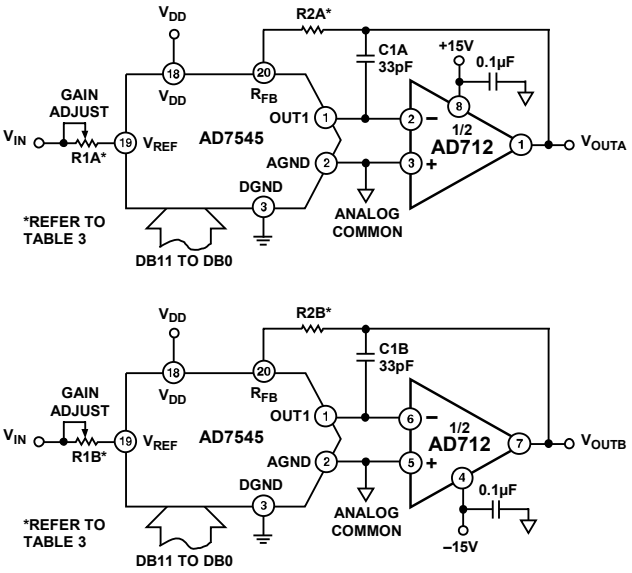


Figure 39. Unipolar Binary Operation

R1 and R2 calibrate the zero offset and gain error of the DAC. Specific values for these resistors depend upon the grade of AD7545 and are listed in Table 3.

Table 3. Recommended Trim Resistor Values vs. Grades of the AD7545 for V_{DD} = 5 V

Trim Resistor	JN/AQ	KN/BQ	LN	GLN
R1	500 Ω	200 Ω	100 Ω	20 Ω
R2	150 Ω	68 Ω	33 Ω	6.8 Ω

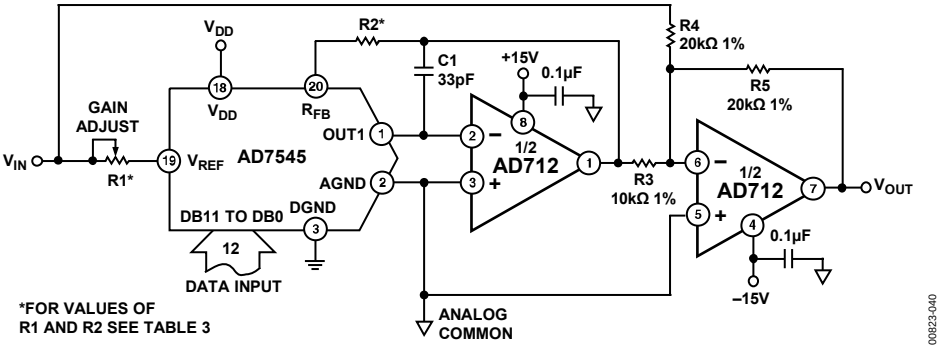


Figure 40. Bipolar Operation

Figure 41 and Figure 42 show the settling time characteristics of the AD712 when used as a DAC output buffer for the AD7545.

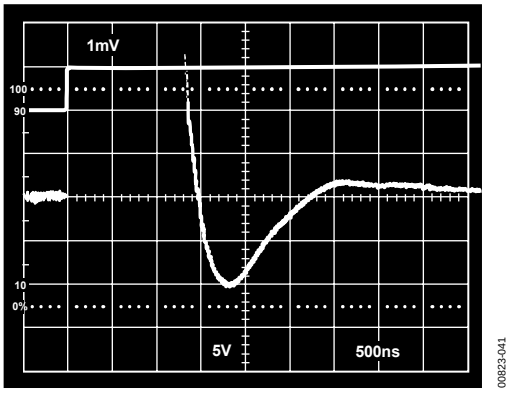


Figure 41. Positive Settling Characteristics for AD712 with AD7545

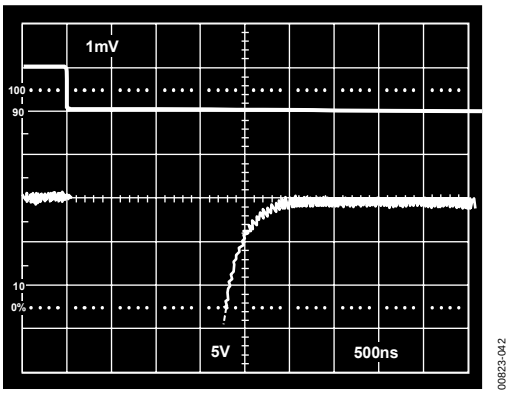


Figure 42. Negative Settling Characteristics for AD712 with AD7545

NOISE CHARACTERISTICS

The random nature of noise, particularly in the flicker noise region, makes it difficult to specify in practical terms. At the same time, designers of precision instrumentation require certain guaranteed maximum noise levels to realize the full accuracy of their equipment. All grades of the AD712 are sample tested on an AQL basis to a limit of 6 µV p-p, 0.1 Hz to 10 Hz.

DRIVING THE ANALOG INPUT OF AN ANALOG-TO-DIGITAL CONVERTER

An op amp driving the analog input of an ADC, such as that shown in Figure 43, must be capable of maintaining a constant output voltage under dynamically changing load conditions. In successive approximation converters, the input current is compared to a series of switched trial currents. The comparison point is diode clamped, but can deviate several hundred millivolts resulting in high frequency modulation of analog-to-digital input current. The output impedance of a feedback amplifier is made artificially low by the loop gain. At high frequencies, where the loop gain is low, the amplifier output impedance can approach its open-loop value. Most IC amplifiers exhibit a minimum open-loop output impedance of 25 Ω due to current-limiting resistors.

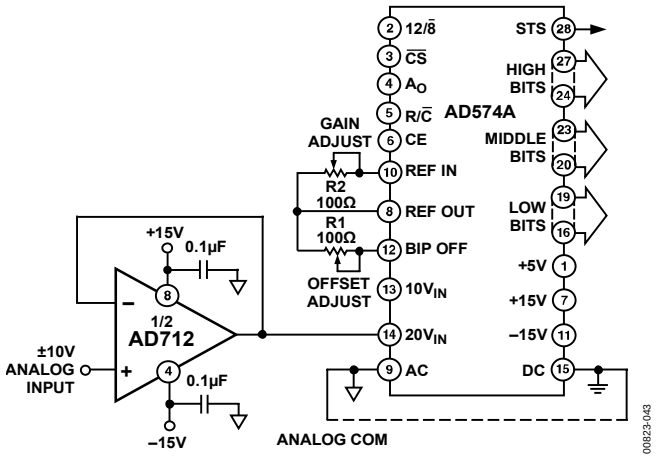


Figure 43. AD712 as An ADC Unity-Gain Buffer

A few hundred microamps reflected from the change in converter loading can introduce errors in instantaneous input voltage. If the analog-to-digital conversion speed is not excessive and the bandwidth of the amplifier is sufficient, the amplifier output returns to the nominal value before the converter makes its comparison. However, many amplifiers have relatively narrow bandwidth yielding slow recovery from output transients. The AD712 is ideally suited to drive high speed analog-to-digital converters because it offers both wide bandwidth and high open-loop gain.

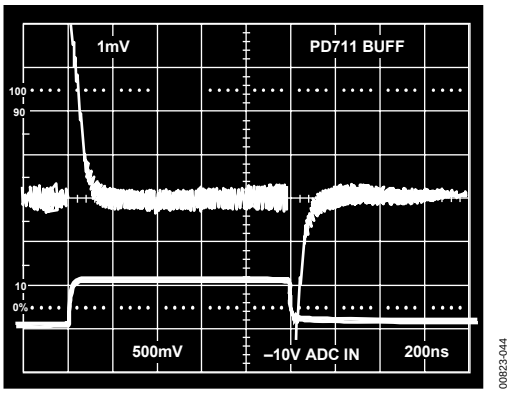


Figure 44. ADC Input Unity Gain Buffer Recovery Times, –10 V ADC IN

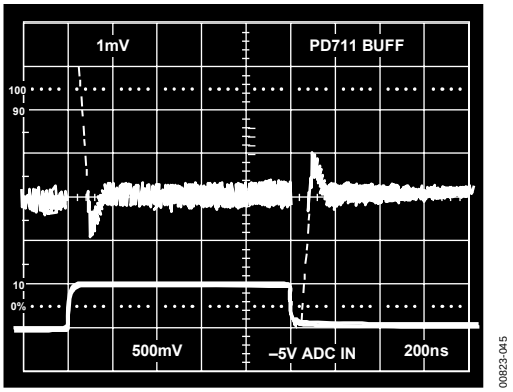


Figure 45. ADC Input Unity Gain Buffer Recovery Times, –5 V ADC IN

DRIVING A LARGE CAPACITIVE LOAD

The circuit in Figure 46 uses a 100 Ω isolation resistor that enables the amplifier to drive capacitive loads exceeding 1500 pF; the resistor effectively isolates the high frequency feedback from the load and stabilizes the circuit. Low frequency feedback is returned to the amplifier summing junction via the low-pass filter formed by the 100 Ω series resistor and the Load Capacitance C_L. Figure 47 shows a typical transient response for this connection.

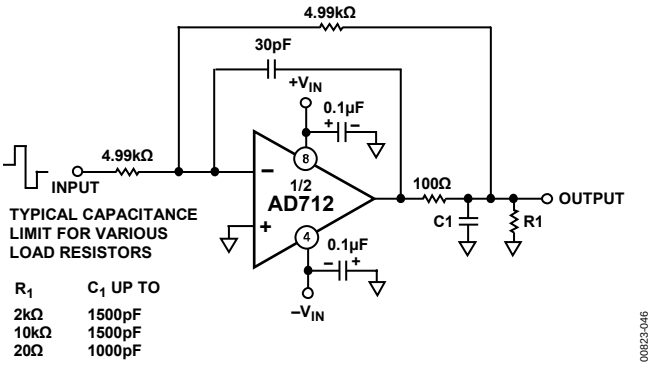


Figure 46. Circuit for Driving a Large Capacitive Load

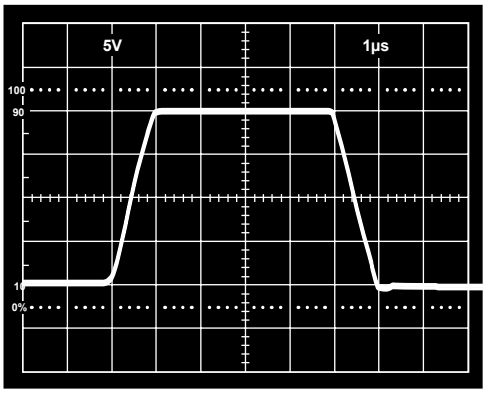


Figure 47. Transient Response R_L = 2 kΩ, C_L = 500 pF

FILTERS

ACTIVE FILTER APPLICATIONS

In active filter applications using op amps, the dc accuracy of the amplifier is critical to optimal filter performance. The amplifier offset voltage and bias current contribute to output error. Offset voltage is passed by the filter and can be amplified to produce excessive output offset. For low frequency applications requiring large value input resistors, bias currents flowing through these resistors also generate an offset voltage.

In addition, at higher frequencies, the op amp dynamics must be carefully considered. Here, slew rate, bandwidth, and open-loop gain play a major role in op amp selection. The slew rate must be fast as well as symmetrical to minimize distortion. The amplifier bandwidth in conjunction with the filter gain dictates the frequency response of the filter.

The use of a high performance amplifier such as the AD712 minimizes both dc and ac errors in all active filter applications.

SECOND-ORDER LOW-PASS FILTER

Figure 48 depicts the AD712 configured as a second-order, Butterworth low-pass filter. With the values as shown, the corner frequency is 20 kHz; however, the wide bandwidth of the AD712 permits a corner frequency as high as several hundred kilohertz. Equations for component selection are as follows:

R1 = R2 = A user selected value (10 kΩ to 100 kΩ, typical)

$$C1 \text{ (in farads)} = \frac{1.414}{(2\pi)(f_{cutoff})(R1)}$$

$$C2 = \frac{0.707}{(2\pi)(f_{cutoff})(R1)}$$

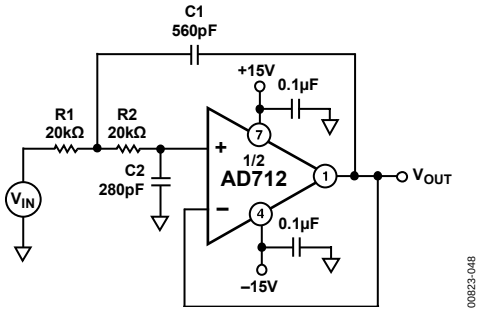


Figure 48. Second-Order Low-Pass Filter

An important property of filters is their out-of-band rejection. The simple 20 kHz low-pass filter shown in Figure 48, can be used to condition a signal contaminated with clock pulses or sampling glitches that have considerable energy content at high frequencies.

The low output impedance and high bandwidth of the AD712 minimize high frequency feedthrough as shown in Figure 49. The upper trace is that of another low cost BiFET op amp showing 17 dB more feedthrough at 5 MHz.

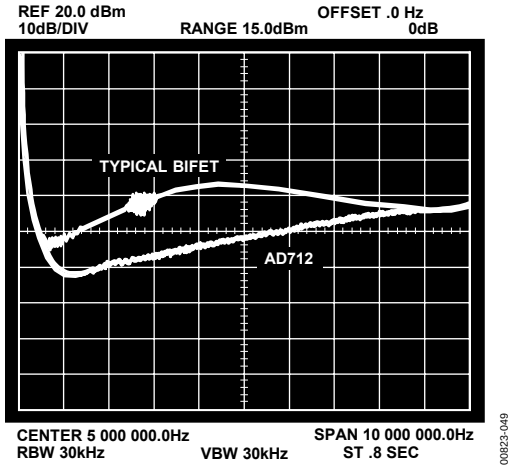


Figure 49. High Frequency Feedthrough

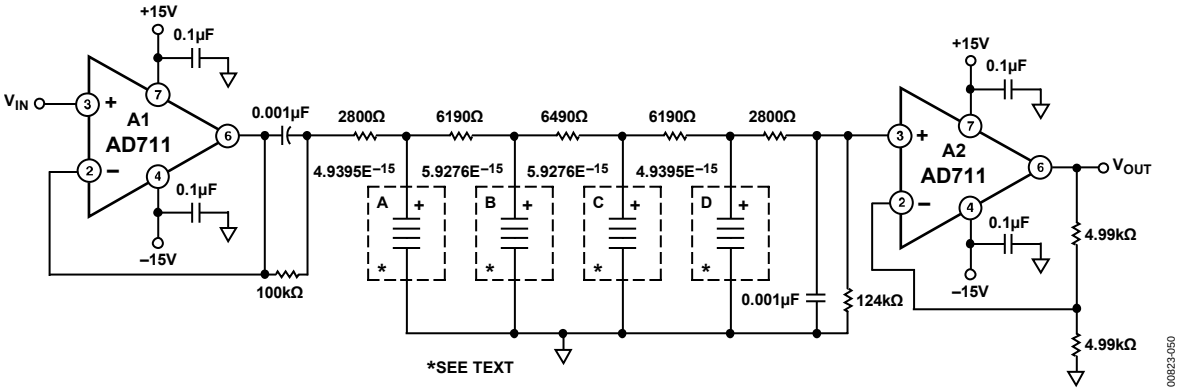


Figure 50. 9-Pole Chebychev Filter

9-POLE CHEBYCHEV FILTER

Figure 50 and Figure 51 show the AD712 and its dual counterpart, the AD711, as a 9-pole Chebychev filter using active frequency dependent negative resistors (FDNRs). With a cutoff frequency of 50 kHz and better than 90 dB rejection, it can be used as an antialiasing filter for a 12-bit data acquisition system with 100 kHz throughput.

As shown in Figure 50, the filter is comprised of four FDNRs (A, B, C, D) having values of 4.9395×10^{-15} and 5.9276×10^{-15} farad-seconds. Each FDNR active network provides a two-pole response for a total of eight poles. The ninth pole consists of a 0.001 µF capacitor and a 124 kΩ resistor at Pin 3 of Amplifier A2. Figure 51 depicts the circuits for each FDNR with the proper selection of R. To achieve optimal performance, the 0.001 µF capacitors must be selected for 1% or better matching and all resistors should have 1% or better tolerance.

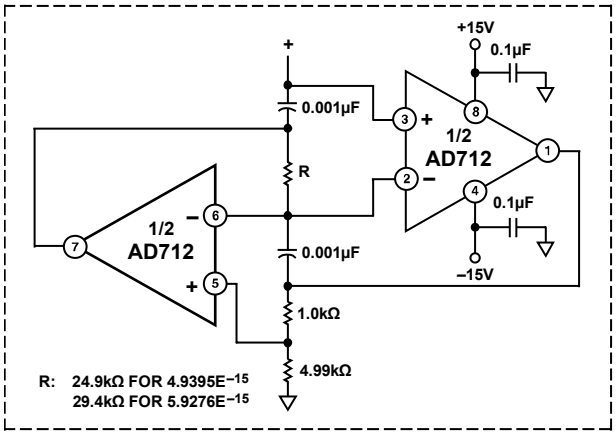


Figure 51. FDNR for 9-Pole Chebychev Filter

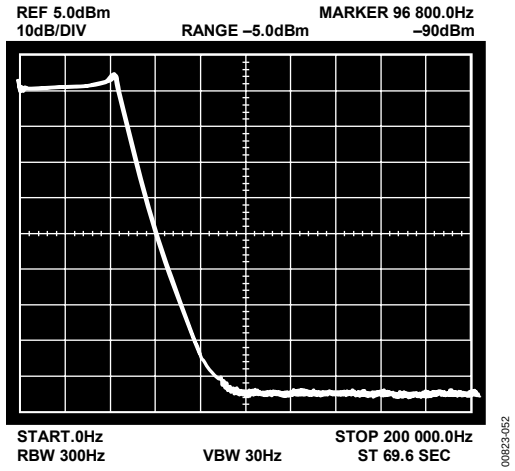
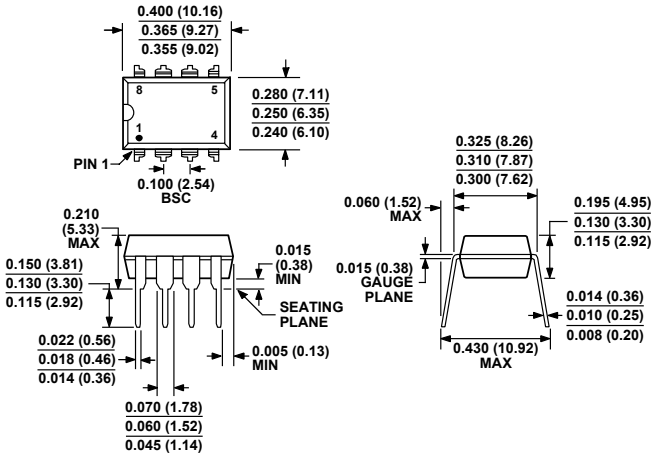


Figure 52. High Frequency Response for 9-Pole Chebychev Filter

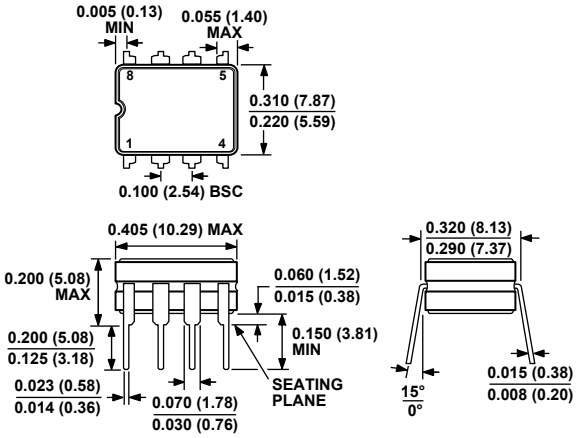
OUTLINE DIMENSIONS



COMPLIANT TO JEDEC STANDARDS MS-001-BA

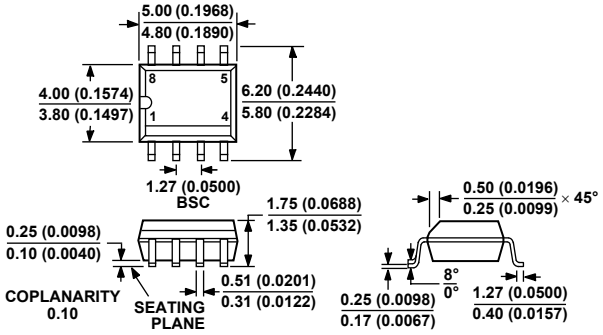
CONTROLLING DIMENSIONS ARE IN INCHES; MILLIMETER DIMENSIONS (IN PARENTHESES) ARE ROUNDED-OFF INCH EQUIVALENTS FOR REFERENCE ONLY AND ARE NOT APPROPRIATE FOR USE IN DESIGN. CORNER LEADS MAY BE CONFIGURED AS WHOLE OR HALF LEADS.

Figure 53. 8-Lead Plastic Dual In-Line Package [PDIP]
(N-8)
Dimensions shown in inches and (millimeters)



CONTROLLING DIMENSIONS ARE IN INCHES; MILLIMETER DIMENSIONS (IN PARENTHESES) ARE ROUNDED-OFF INCH EQUIVALENTS FOR REFERENCE ONLY AND ARE NOT APPROPRIATE FOR USE IN DESIGN.

Figure 54. 8-Lead Ceramic Dual In-Line Package [CERDIP]
(Q-8)
Dimensions shown in inches and (millimeters)



COMPLIANT TO JEDEC STANDARDS MS-012-AA

CONTROLLING DIMENSIONS ARE IN MILLIMETERS; INCH DIMENSIONS (IN PARENTHESES) ARE ROUNDED-OFF MILLIMETER EQUIVALENTS FOR REFERENCE ONLY AND ARE NOT APPROPRIATE FOR USE IN DESIGN.

Figure 55. 8-Lead Standard Small Outline Package [SOIC_N]
Narrow Body
(R-8)
Dimensions shown in millimeters and (inches)

ORDERING GUIDE

Model	Temperature Range	Package Description	Package Option
AD712AQ	−40°C to +85°C	8-Lead CERDIP	Q-8
AD712JN	0°C to 70°C	8-Lead PDIP	N-8
AD712JNZ[1]	0°C to 70°C	8-Lead PDIP	N-8
AD712JR	0°C to 70°C	8-Lead SOIC_N	R-8
AD712JR-REEL	0°C to 70°C	8-Lead SOIC_N	R-8
AD712JR-REEL7	0°C to 70°C	8-Lead SOIC_N	R-8
AD712JRZ[1]	0°C to 70°C	8-Lead SOIC_N	R-8
AD712JRZ-REEL[1]	0°C to 70°C	8-Lead SOIC_N	R-8
AD712JRZ-REEL7[1]	0°C to 70°C	8-Lead SOIC_N	R-8
AD712KN	0°C to 70°C	8-Lead PDIP	N-8
AD712KNZ[1]	0°C to 70°C	8-Lead PDIP	N-8
AD712KR	0°C to 70°C	8-Lead SOIC_N	R-8
AD712KR-REEL	0°C to 70°C	8-Lead SOIC_N	R-8
AD712KR-REEL7	0°C to 70°C	8-Lead SOIC_N	R-8
AD712KRZ[1]	0°C to 70°C	8-Lead SOIC_N	R-8
AD712KRZ-REEL[1]	0°C to 70°C	8-Lead SOIC_N	R-8
AD712KRZ-REEL7[1]	0°C to 70°C	8-Lead SOIC_N	R-8
AD712SQ/883B	−55°C to +125°C	8-Lead CERDIP	Q-8

[1] Z = Pb-free part.

©2006 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners.
C00823-0-8/06(G)



PIC16F87XA
Data Sheet

28/40/44-Pin Enhanced Flash Microcontrollers

© 2003 Microchip Technology Inc.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, MPLAB, PIC, PICmicro, PICSTART, PRO MATE and PowerSmart are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

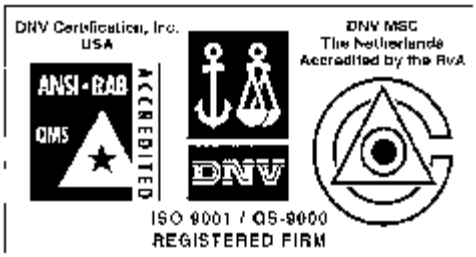
Application Maestro, dsPICDEM, dsPICDEM.net, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICkit, PICDEM, PICDEM.net, PowerCal, PowerInfo, PowerMate, PowerTool, rfLAB, rfPIC, Select Mode, SmartSensor, SmartShunt, SmartTel and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2003, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

© 2003 Microchip Technology Inc.

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

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

Peripheral Features:

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

Analog Features:

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

Special Microcontroller Features:

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

CMOS Technology:

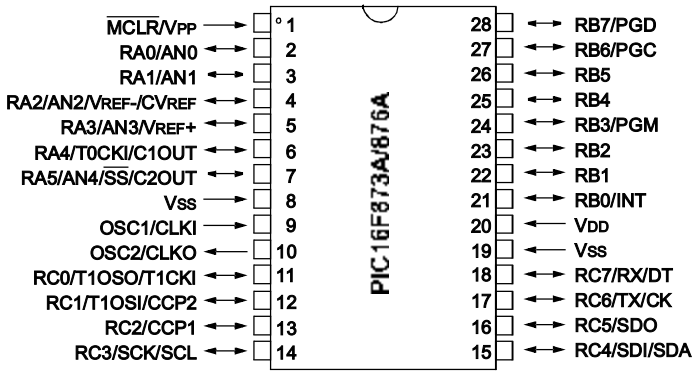
- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

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

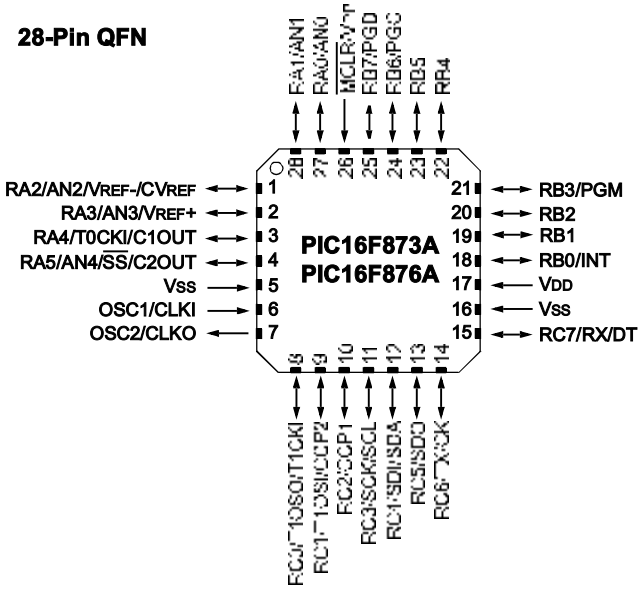
© 2003 Microchip Technology Inc.

Pin Diagrams

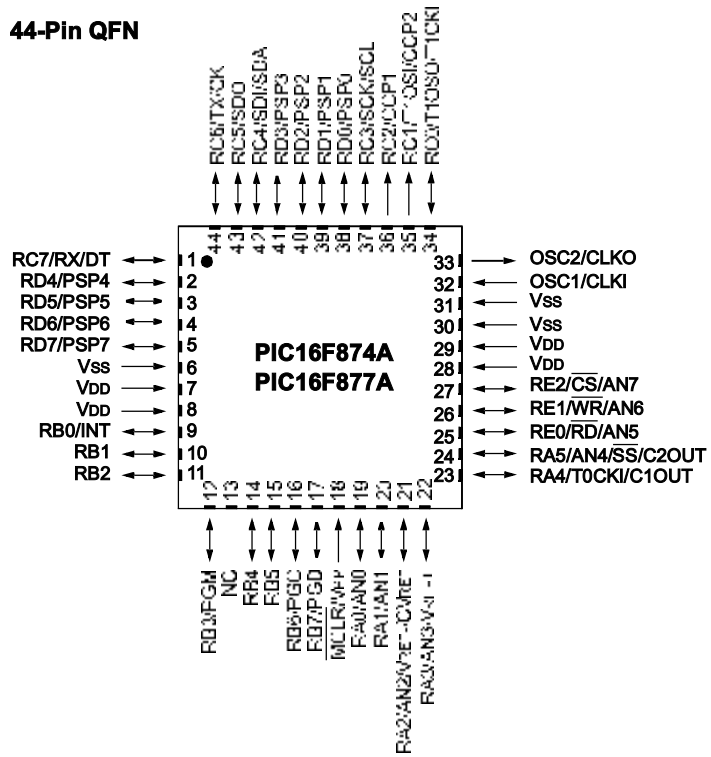
28-Pin PDIP, SOIC, SSOP



28-Pin QFN



44-Pin QFN



© 2003 Microchip Technology Inc.

Pin Diagrams (Continued)

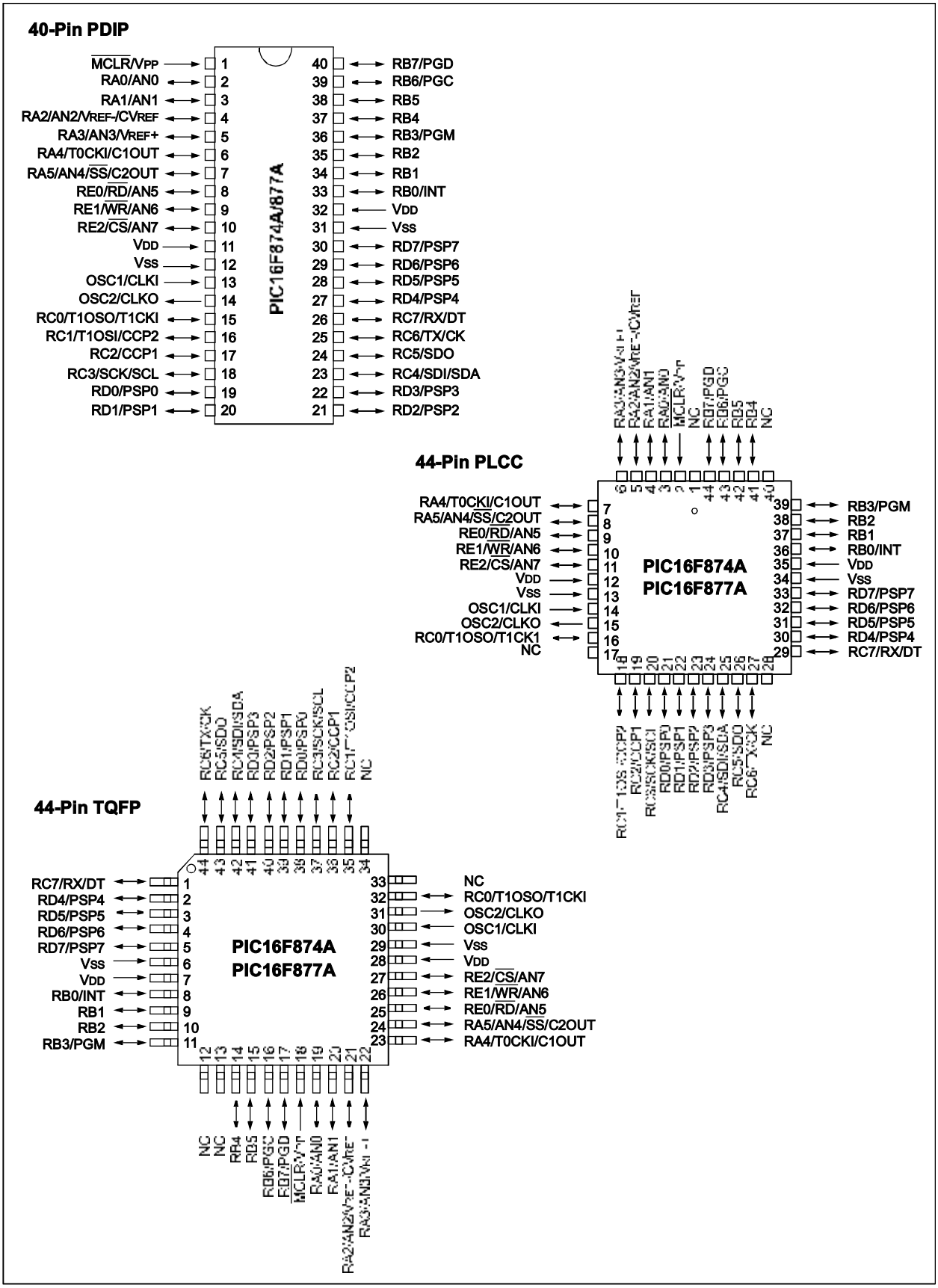


Table of Contents

1.0 Device Overview 5
2.0 Memory Organization 15
3.0 Data EEPROM and Flash Program Memory 33
4.0 I/O Ports 41
5.0 Timer0 Module 53
6.0 Timer1 Module 57
7.0 Timer2 Module 61
8.0 Capture/Compare/PWM Modules 63
9.0 Master Synchronous Serial Port (MSSP) Module 71
10.0 Addressable Universal Synchronous Asynchronous Receiver Transmitter (USART) 111
11.0 Analog-to-Digital Converter (A/D) Module 127
12.0 Comparator Module 135
13.0 Comparator Voltage Reference Module 141
14.0 Special Features of the CPU 143
15.0 Instruction Set Summary 159
16.0 Development Support 167
17.0 Electrical Characteristics 173
18.0 DC and AC Characteristics Graphs and Tables 197
19.0 Packaging Information 209
Appendix A: Revision History 219
Appendix B: Device Differences 219
Appendix C: Conversion Considerations 220
Index 221
On-Line Support 229
Systems Information and Upgrade Hot Line 229
Reader Response 230
PIC16F87XA Product Identification System 231

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our Web site at **www.microchip.com/cn** to receive the most current information on all of our products.

© 2003 Microchip Technology Inc.

1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

PIC16F873A/876A devices are available only in 28-pin packages, while PIC16F874A/877A devices are available in 40-pin and 44-pin packages. All devices in the PIC16F87XA family share common architecture with the following differences:

- The PIC16F873A and PIC16F874A have one-half of the total on-chip memory of the PIC16F876A and PIC16F877A
- The 28-pin devices have three I/O ports, while the 40/44-pin devices have five
- The 28-pin devices have fourteen interrupts, while the 40/44-pin devices have fifteen
- The 28-pin devices have five A/D input channels, while the 40/44-pin devices have eight
- The Parallel Slave Port is implemented only on the 40/44-pin devices

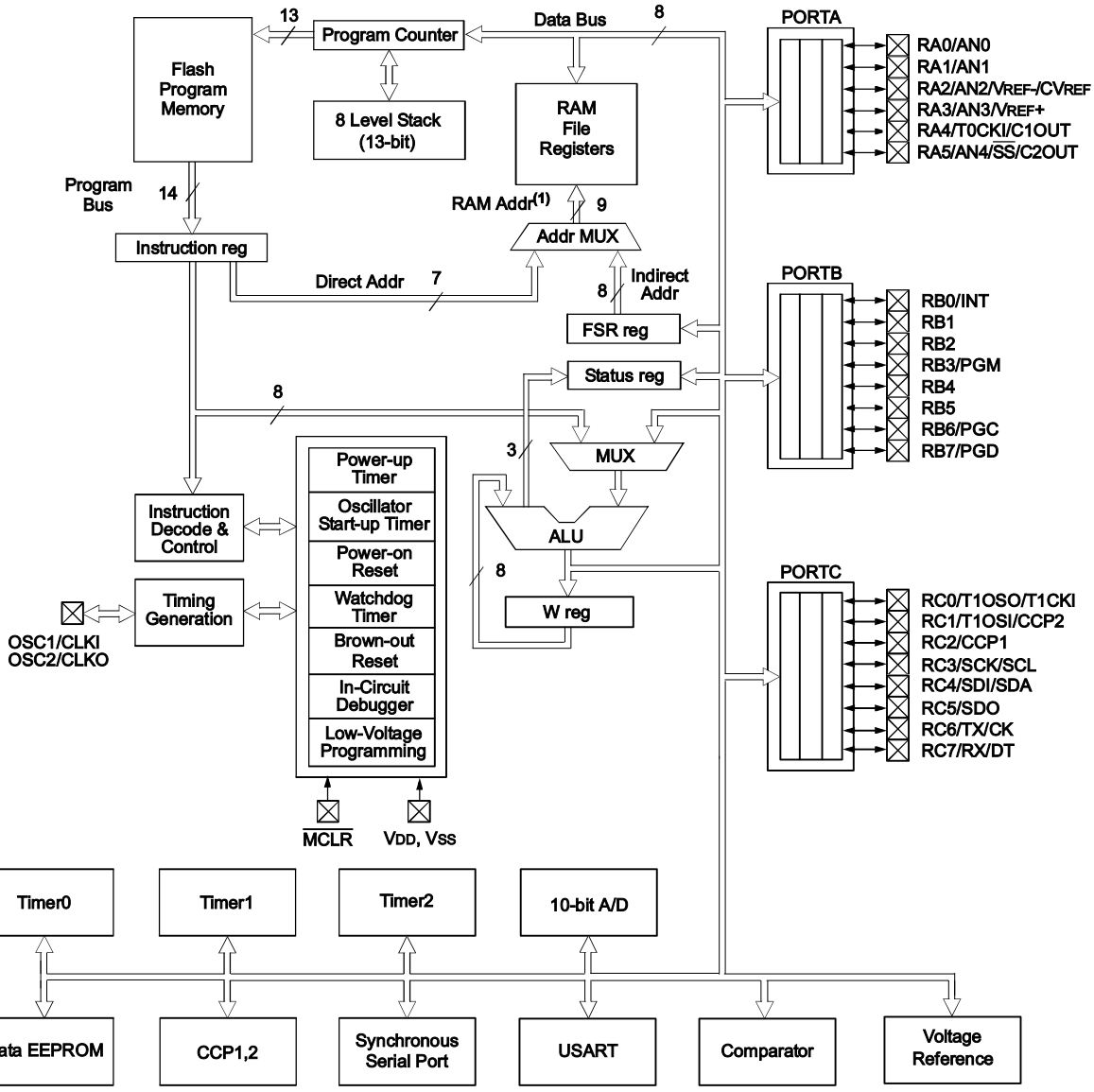
The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

Additional information may be found in the PICmicro® Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The Reference Manual should be considered a complementary document to this data sheet and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

TABLE 1-1: PIC16F87XA DEVICE FEATURES

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

FIGURE 1-1: PIC16F873A/876A BLOCK DIAGRAM

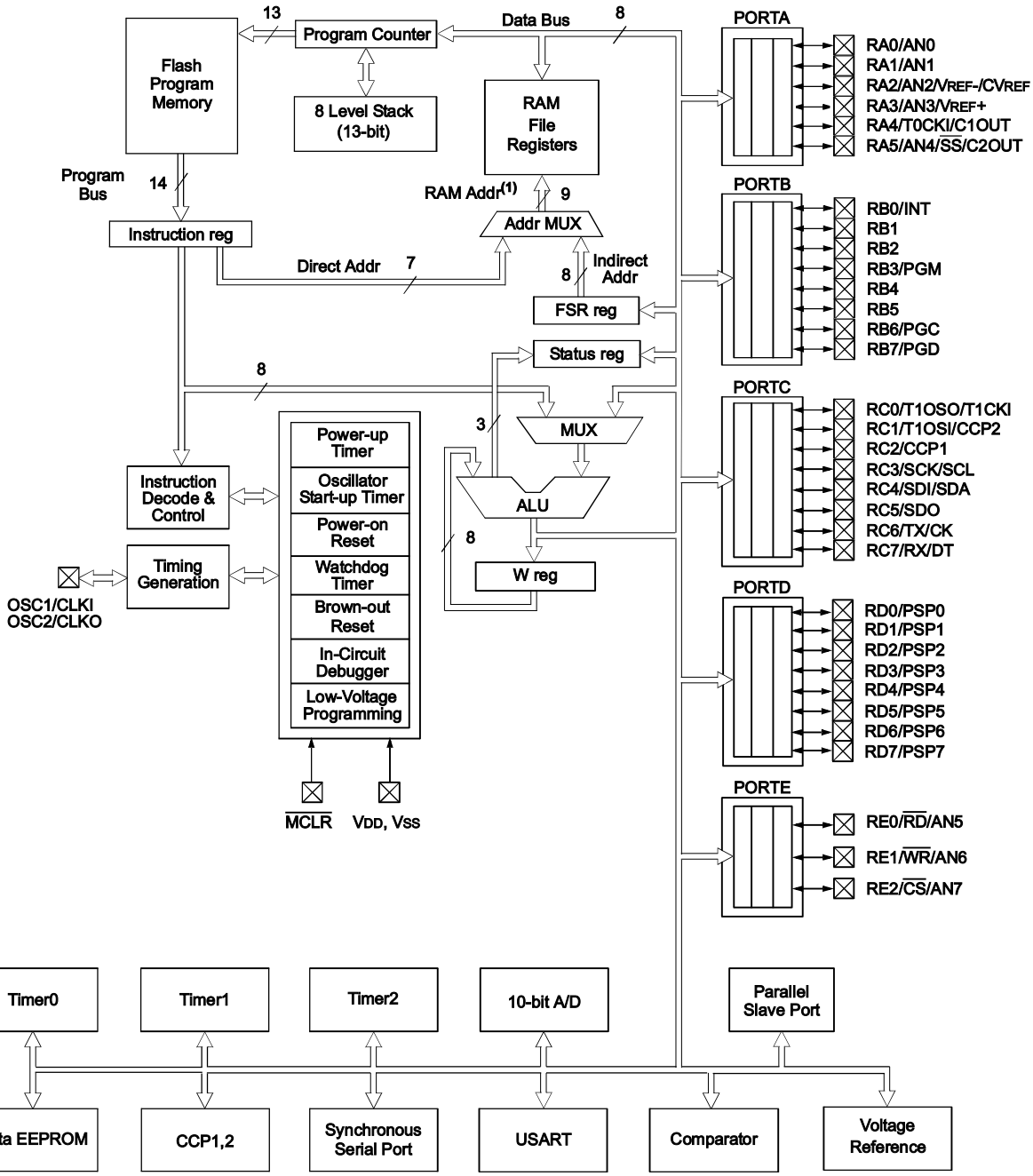


Device	Program Flash	Data Memory	Data EEPROM
PIC16F873A	4K words	192 Bytes	128 Bytes
PIC16F876A	8K words	368 Bytes	256 Bytes

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

© 2003 Microchip Technology Inc.

FIGURE 1-2: PIC16F874A/877A BLOCK DIAGRAM



Device	Program Flash	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

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

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.
$\overline{MCLR}$/VPP	1	26		ST	Master Clear (input) or programming voltage (output).
$\overline{MCLR}$			I		Master Clear (Reset) input. This pin is an active low Reset to the device.
VPP			P		Programming voltage input.
					PORTA is a bidirectional I/O port.
RA0/AN0	2	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-/	4	1		TTL	
CVREF			I/O		Digital I/O.
RA2			I		Analog input 2.
AN2			I		A/D reference voltage (Low) input.
VREF-			O		Comparator VREF output.
CVREF					
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/$\overline{SS}$/C2OUT	7	4		TTL	
RA5			I/O		Digital I/O.
AN4			I		Analog input 4.
$\overline{SS}$			I		SPI slave select input.
C2OUT			O		Comparator 2 output.

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

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

© 2003 Microchip Technology Inc.

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^2C mode.
RC4/SDI/SDA	15	12		ST	
RC4			I/O		Digital I/O.
SDI			I		SPI data in.
SDA			I/O		I^2C 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.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION

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

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

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

© 2003 Microchip Technology Inc.

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

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

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

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

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

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

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

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

© 2003 Microchip Technology Inc.

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

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

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

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

NOTES:

© 2003 Microchip Technology Inc.

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

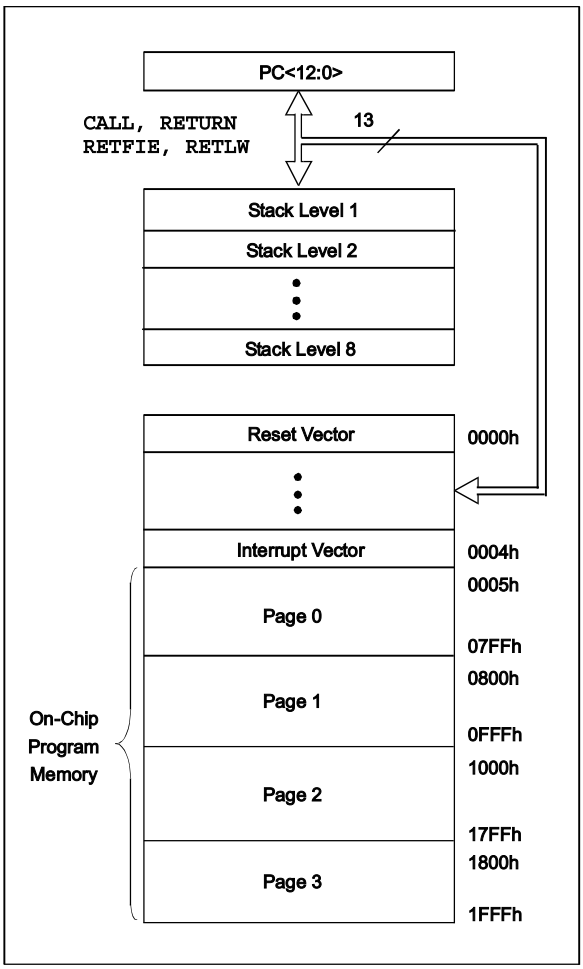
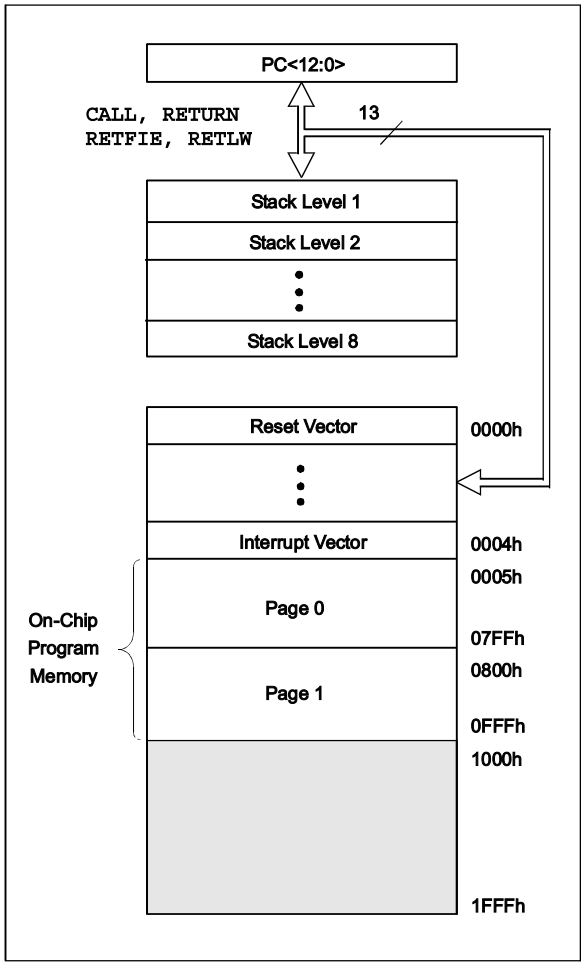


FIGURE 2-2: PIC16F873A/874A PROGRAM MEMORY MAP AND STACK



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.

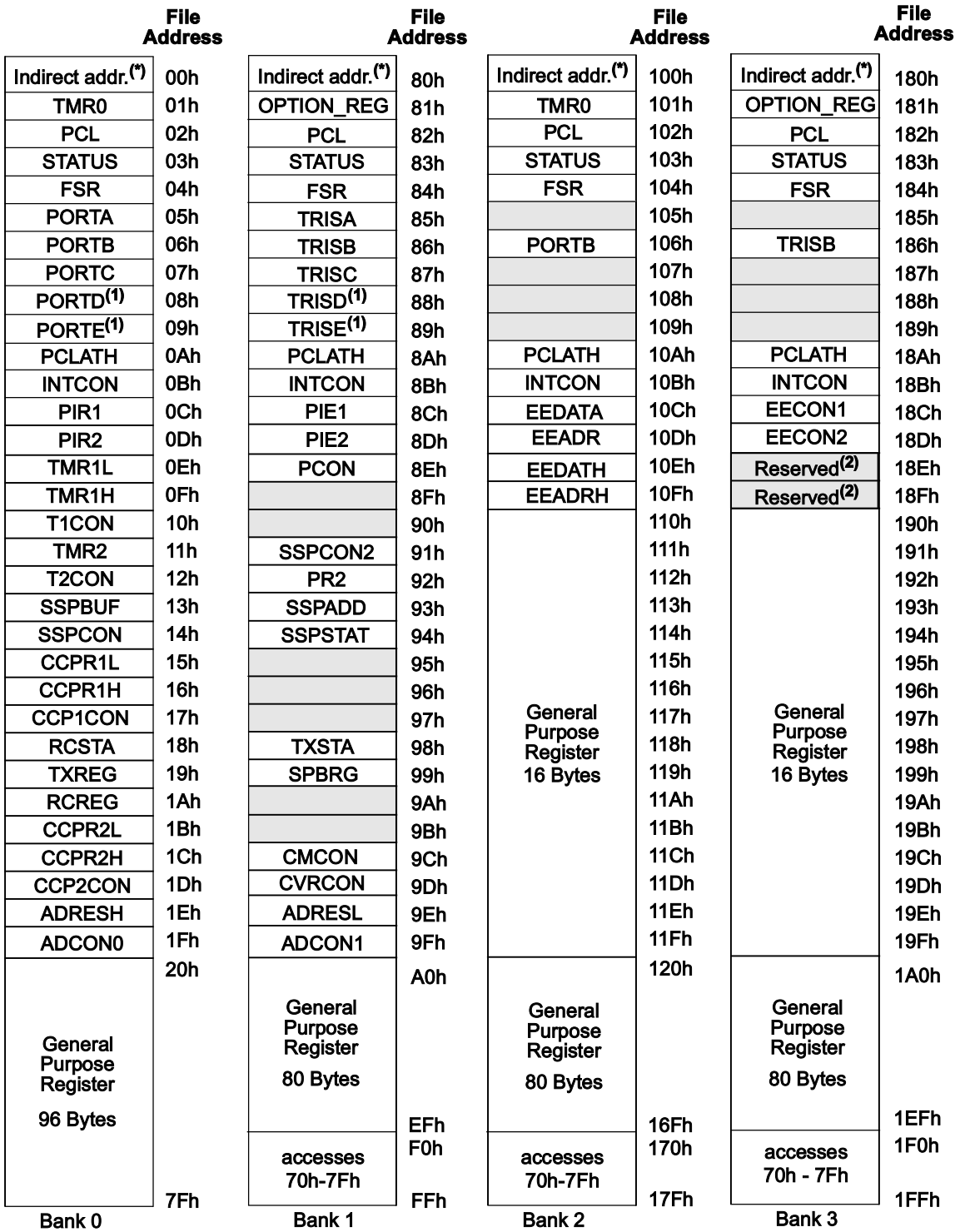
Note:	The EEPROM data memory description can be found in **Section 3.0 "Data EEPROM and Flash Program Memory"** of this data sheet.

2.2.1 GENERAL PURPOSE REGISTER FILE

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

© 2003 Microchip Technology Inc.

FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP



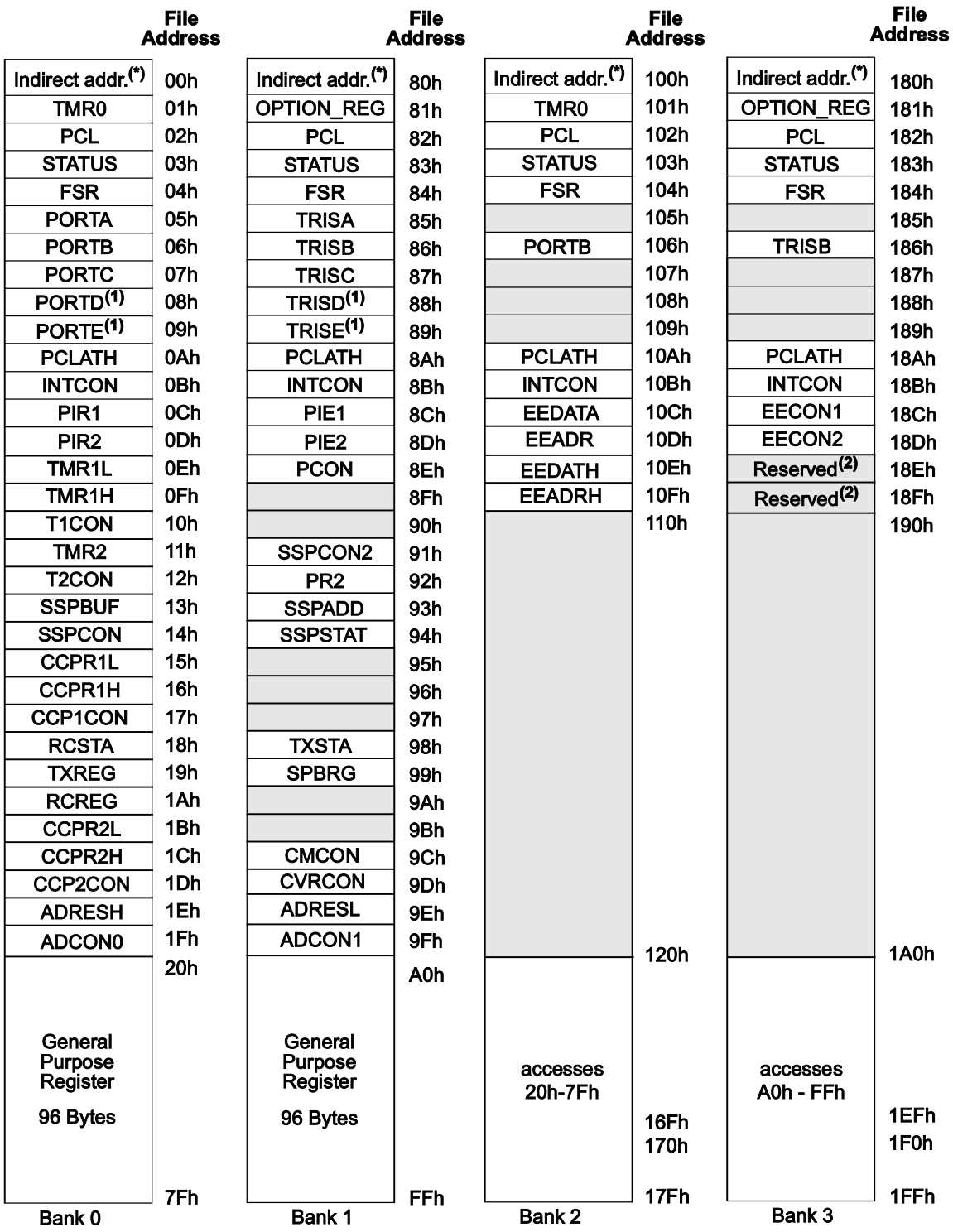
Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.

2: These registers are reserved; maintain these registers clear.

FIGURE 2-4: PIC16F873A/874A REGISTER FILE MAP



Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873A.
2: These registers are reserved; maintain these registers clear.

© 2003 Microchip Technology Inc.

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
01h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	43, 150
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	47, 150
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	48, 150
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	49, 150
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
0Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	26, 150
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	28, 150
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	57, 150
11h	TMR2	Timer2 Module Register								0000 0000	62, 150
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	61, 150
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	79, 150
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	82, 82, 150
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	63, 150
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	63, 150
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	64, 150
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	112, 150
19h	TXREG	USART Transmit Data Register								0000 0000	118, 150
1Ah	RCREG	USART Receive Data Register								0000 0000	118, 150
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	63, 150
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	63, 150
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	64, 150
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	133, 150
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	127, 150

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
81h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
83h(3)	STATUS	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	0001 1xxx	22, 150
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	43, 150
86h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
87h	TRISC	PORTC Data Direction Register								1111 1111	47, 150
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	48, 151
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	50, 151
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
8Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	25, 151
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	27, 151
8Eh	PCON	—	—	—	—	—	—	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	---- --qq	29, 151
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	83, 151
92h	PR2	Timer2 Period Register								1111 1111	62, 151
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	79, 151
94h	SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	79, 151
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	111, 151
99h	SPBRG	Baud Rate Generator Register								0000 0000	113, 151
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	135, 151
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	141, 151
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	133, 151
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	128, 151

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

© 2003 Microchip Technology Inc.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
101h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	30, 150
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
10Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	39, 151
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	39, 151
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						--xx xxxx	39, 151
10Fh	EEADRH	—	—	—	—(5)	EEPROM Address Register High Byte				---- xxxx	39, 151
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
18Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	34, 151
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	39, 151
18Eh	—	Reserved; maintain clear								0000 0000	—
18Fh	—	Reserved; maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

2.2.2.1 Status Register

The Status register contains the arithmetic status of the ALU, the Reset status and the bank select bits for data memory.

The Status register can be the destination for any instruction, as with any other register. If the Status register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable, therefore, the result of an instruction with the Status register as destination may be different than intended.

For example, `CLRF STATUS`, will clear the upper three bits and set the Z bit. This leaves the Status register as `000u u1uu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the Status register because these instructions do not affect the Z, C or DC bits from the Status register. For other instructions not affecting any status bits, see **Section 15.0 "Instruction Set Summary"**.

Note:	The C and DC bits operate as a $\overline{borrow}$ and digit $\overline{borrow}$ bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

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

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
`1` = Bank 2, 3 (100h-1FFh)
`0` = Bank 0, 1 (00h-FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
`11` = Bank 3 (180h-1FFh)
`10` = Bank 2 (100h-17Fh)
`01` = Bank 1 (80h-FFh)
`00` = Bank 0 (00h-7Fh)
Each bank is 128 bytes.

bit 4 **$\overline{TO}$**: Time-out bit
`1` = After power-up, `CLRWDT` instruction or `SLEEP` instruction
`0` = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
`1` = After power-up or by the `CLRWDT` instruction
`0` = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
`1` = The result of an arithmetic or logic operation is zero
`0` = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
(for $\overline{borrow}$, the polarity is reversed)
`1` = A carry-out from the 4th low order bit of the result occurred
`0` = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
`1` = A carry-out from the Most Significant bit of the result occurred
`0` = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

2.2.2.2 OPTION_REG Register

The OPTION_REG Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the external INT interrupt, TMR0 and the weak pull-ups on PORTB.

Note: To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

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

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device

2.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB port change and external RB0/INT pin interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

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

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

2.2.2.4 PIE1 Register

The PIE1 register contains the individual enable bits for the peripheral interrupts.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

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

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note 1: PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.5 PIR1 Register

The PIR1 register contains the individual flag bits for the peripheral interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

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

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit(1)
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred
Note 1: PSPIF is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag bit
1 = The SSP interrupt condition has occurred and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI – A transmission/reception has taken place.
- I^2C Slave – A transmission/reception has taken place.
- I^2C Master
  - A transmission/reception has taken place.
  - The initiated Start condition was completed by the SSP module.
  - The initiated Stop condition was completed by the SSP module.
  - The initiated Restart condition was completed by the SSP module.
  - The initiated Acknowledge condition was completed by the SSP module.
  - A Start condition occurred while the SSP module was Idle (multi-master system).
  - A Stop condition occurred while the SSP module was Idle (multi-master system).

0 = No SSP interrupt condition has occurred

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

2.2.2.6 PIE2 Register

The PIE2 register contains the individual enable bits for the CCP2 peripheral interrupt, the SSP bus collision interrupt, EEPROM write operation interrupt and the comparator interrupt.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-6: PIE2 REGISTER (ADDRESS 8Dh)

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

bit 7 **Unimplemented:** Read as '0'

bit 6 **CMIE**: Comparator Interrupt Enable bit
1 = Enables the comparator interrupt
0 = Disable the comparator interrupt

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIE**: EEPROM Write Operation Interrupt Enable bit
1 = Enable EEPROM write interrupt
0 = Disable EEPROM write interrupt

bit 3 **BCLIE**: Bus Collision Interrupt Enable bit
1 = Enable bus collision interrupt
0 = Disable bus collision interrupt

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.7 PIR2 Register

The PIR2 register contains the flag bits for the CCP2 interrupt, the SSP bus collision interrupt, EEPROM write operation interrupt and the comparator interrupt.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-7: 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

bit 7 **Unimplemented:** Read as '0'

bit 6 **CMIF**: Comparator Interrupt Flag bit
1 = The comparator input has changed (must be cleared in software)
0 = The comparator input has not changed

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIF**: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **BCLIF**: Bus Collision Interrupt Flag bit
1 = A bus collision has occurred in the SSP when configured for I^2C Master mode
0 = No bus collision has occurred

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IF**: CCP2 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

2.2.2.8 PCON Register

The Power Control (PCON) register contains flag bits to allow differentiation between a Power-on Reset (POR), a Brown-out Reset (BOR), a Watchdog Reset (WDT) and an external MCLR Reset.

Note:	BOR is unknown on Power-on Reset. It must be set by the user and checked on subsequent Resets to see if BOR is clear, indicating a brown-out has occurred. The BOR status bit is a "don't care" and is not predictable if the brown-out circuit is disabled (by clearing the BODEN bit in the configuration word).

REGISTER 2-8: PCON REGISTER (ADDRESS 8Eh)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-1
—	—	—	—	—	—	POR	BOR
bit 7							bit 0

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **POR**: Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

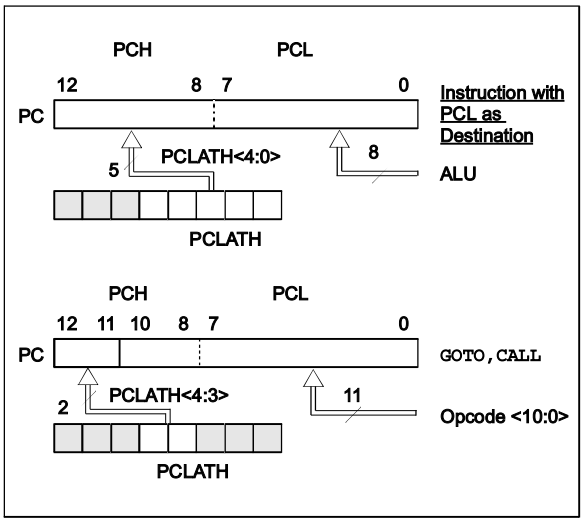
bit 0 **BOR**: Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.3 PCL and PCLATH

The Program Counter (PC) is 13 bits wide. The low byte comes from the PCL register which is a readable and writable register. The upper bits (PC<12:8>) are not readable, but are indirectly writable through the PCLATH register. On any Reset, the upper bits of the PC will be cleared. Figure 2-5 shows the two situations for the loading of the PC. The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in the figure shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

FIGURE 2-5: LOADING OF PC IN DIFFERENT SITUATIONS



2.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to the application note, *AN556, "Implementing a Table Read"* (DS00556).

2.3.2 STACK

The PIC16F87XA family has an 8-level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed, or an interrupt causes a branch. The stack is POP'ed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

Note 1:	There are no status bits to indicate stack overflow or stack underflow conditions.
2:	There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

2.4 Program Memory Paging

All PIC16F87XA devices are capable of addressing a continuous 8K word block of program memory. The CALL and GOTO instructions provide only 11 bits of address to allow branching within any 2K program memory page. When doing a CALL or GOTO instruction, the upper 2 bits of the address are provided by PCLATH<4:3>. When doing a CALL or GOTO instruction, the user must ensure that the page select bits are programmed so that the desired program memory page is addressed. If a return from a CALL instruction (or interrupt) is executed, the entire 13-bit PC is popped off the stack. Therefore, manipulation of the PCLATH<4:3> bits is not required for the RETURN instructions (which POPs the address from the stack).

Note:	The contents of the PCLATH register are unchanged after a RETURN or RETFIE instruction is executed. The user must rewrite the contents of the PCLATH register for any subsequent subroutine calls or GOTO instructions.

Example 2-1 shows the calling of a subroutine in page 1 of the program memory. This example assumes that PCLATH is saved and restored by the Interrupt Service Routine (if interrupts are used).

EXAMPLE 2-1: CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0

```
        ORG 0x500
        BCF PCLATH,4
        BSF PCLATH,3    ;Select page 1
                        ;(800h-FFFh)
        CALL SUB1_P1    ;Call subroutine in
        :               ;page 1 (800h-FFFh)
        :
        ORG 0x900       ;page 1 (800h-FFFh)
SUB1_P1
        :               ;called subroutine
                        ;page 1 (800h-FFFh)
        :
        RETURN          ;return to
                        ;Call subroutine
                        ;in page 0
                        ;(000h-7FFh)
```


© 2003 Microchip Technology Inc.

2.5 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

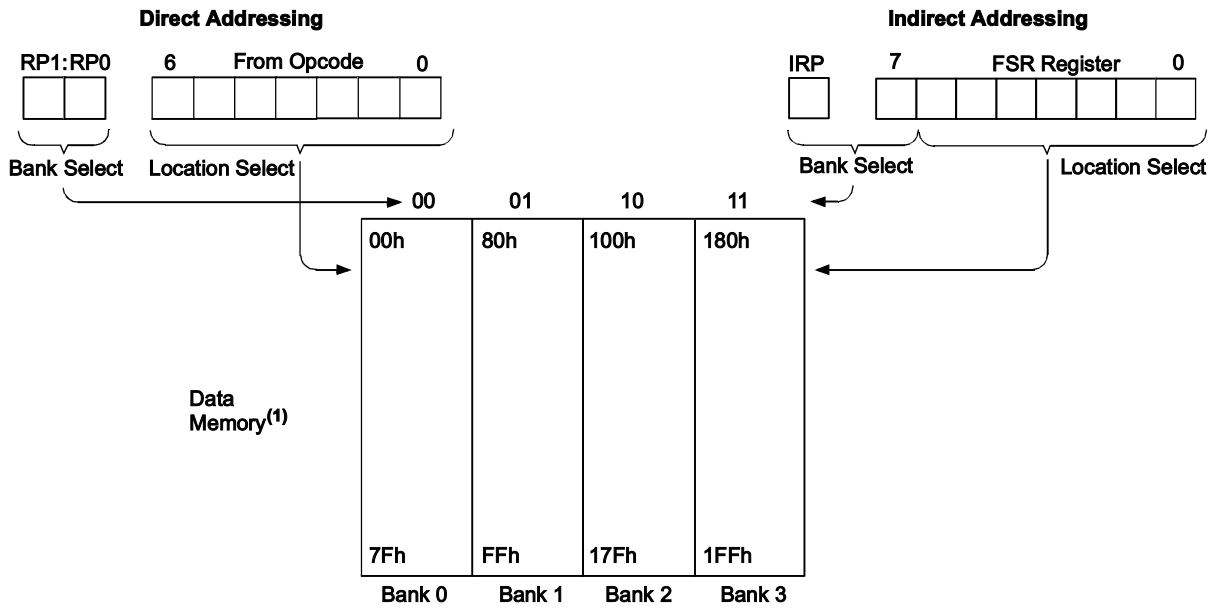
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = 0) will read 00h. Writing to the INDF register indirectly results in a no operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (Status<7>) as shown in Figure 2-6.

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: INDIRECT ADDRESSING

```
        MOVLW  0x20    ;initialize pointer
        MOVWF  FSR     ;to RAM
NEXT    CLRF   INDF    ;clear INDF register
        INCF   FSR,F   ;inc pointer
        BTFSS  FSR,4   ;all done?
        GOTO   NEXT    ;no clear next
CONTINUE
        :              ;yes continue
```

FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Note 1: For register file map detail, see Figure 2-3.

NOTES:

© 2003 Microchip Technology Inc.

3.0 DATA EEPROM AND FLASH PROGRAM MEMORY

The data EEPROM and Flash program memory is readable and writable during normal operation (over the full VDD range). This memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers. There are six SFRs used to read and write this memory:

- EECON1
- EECON2
- EEDATA
- EEDATH
- EEADR
- EEADRH

When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and EEADR holds the address of the EEPROM location being accessed. These devices have 128 or 256 bytes of data EEPROM (depending on the device), with an address range from 00h to FFh. On devices with 128 bytes, addresses from 80h to FFh are unimplemented and will wraparound to the beginning of data EEPROM memory. When writing to unimplemented locations, the on-chip charge pump will be turned off.

When interfacing the program memory block, the EEDATA and EEDATH registers form a two-byte word that holds the 14-bit data for read/write and the EEADR and EEADRH registers form a two-byte word that holds the 13-bit address of the program memory location being accessed. These devices have 4 or 8K words of program Flash, with an address range from 0000h to 0FFFh for the PIC16F873A/874A and 0000h to 1FFFh for the PIC16F876A/877A. Addresses above the range of the respective device will wraparound to the beginning of program memory.

The EEPROM data memory allows single-byte read and write. The Flash program memory allows single-word reads and four-word block writes. Program memory write operations automatically perform an erase-before-write on blocks of four words. A byte write in data EEPROM memory automatically erases the location and writes the new data (erase-before-write).

The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device for byte or word operations.

When the device is code-protected, the CPU may continue to read and write the data EEPROM memory. Depending on the settings of the write-protect bits, the device may or may not be able to write certain blocks of the program memory; however, reads of the program memory are allowed. When code-protected, the device programmer can no longer access data or program memory; this does NOT inhibit internal reads or writes.

3.1 EEADR and EEADRH

The EEADRH:EEADR register pair can address up to a maximum of 256 bytes of data EEPROM or up to a maximum of 8K words of program EEPROM. When selecting a data address value, only the LSByte of the address is written to the EEADR register. When selecting a program address value, the MSByte of the address is written to the EEADRH register and the LSByte is written to the EEADR register.

If the device contains less memory than the full address reach of the address register pair, the Most Significant bits of the registers are not implemented. For example, if the device has 128 bytes of data EEPROM, the Most Significant bit of EEADR is not implemented on access to data EEPROM.

3.2 EECON1 and EECON2 Registers

EECON1 is the control register for memory accesses.

Control bit, EEPGD, determines if the access will be a program or data memory access. When clear, as it is when reset, any subsequent operations will operate on the data memory. When set, any subsequent operations will operate on the program memory.

Control bits, RD and WR, initiate read and write or erase, respectively. These bits cannot be cleared, only set, in software. They are cleared in hardware at completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental, premature termination of a write operation.

The WREN bit, when set, will allow a write or erase operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write (or erase) operation is interrupted by a $\overline{MCLR}$ or a WDT Time-out Reset during normal operation. In these situations, following Reset, the user can check the WRERR bit and rewrite the location. The data and address will be unchanged in the EEDATA and EEADR registers.

Interrupt flag bit, EEIF in the PIR2 register, is set when the write is complete. It must be cleared in software.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the EEPROM write sequence.

Note:	The self-programming mechanism for Flash program memory has been changed. On previous PIC16F87X devices, Flash programming was done in single-word erase/write cycles. The newer PIC18F87XA devices use a four-word erase/write cycle. See **Section 3.6 "Writing to Flash Program Memory"** for more information.

REGISTER 3-1: EECON1 REGISTER (ADDRESS 18Ch)

R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	—	—	—	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD**: Program/Data EEPROM Select bit
1 = Accesses program memory
0 = Accesses data memory
Reads '0' after a POR; this bit cannot be changed while a write operation is in progress.

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **WRERR:** EEPROM Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{\text{MCLR}}$ or any WDT Reset during normal operation)
0 = The write operation completed

bit 2 **WREN:** EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a write cycle. The bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM read; RD is cleared in hardware. The RD bit can only be set (not cleared) in software.
0 = Does not initiate an EEPROM read

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

3.3 Reading Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADR register, clear the EEPGD control bit (EECON1<7>) and then set control bit RD (EECON1<0>). The data is available in the very next cycle in the EEDATA register; therefore, it can be read in the next instruction (see Example 3-1). EEDATA will hold this value until another read or until it is written to by the user (during a write operation).

The steps to reading the EEPROM data memory are:

1. Write the address to EEADR. Make sure that the address is not larger than the memory size of the device.
2. Clear the EEPGD bit to point to EEPROM data memory.
3. Set the RD bit to start the read operation.
4. Read the data from the EEDATA register.

EXAMPLE 3-1: DATA EEPROM READ

```
BSF     STATUS,RP1      ;
BCF     STATUS,RP0      ; Bank 2
MOVF    DATA_EE_ADDR,W  ; Data Memory
MOVWF   EEADR           ; Address to read
BSF     STATUS,RP0      ; Bank 3
BCF     EECON1,EEPGD    ; Point to Data
                        ; memory
BSF     EECON1,RD       ; EE Read
BCF     STATUS,RP0      ; Bank 2
MOVF    EEDATA,W        ; W = EEDATA
```

3.4 Writing to Data EEPROM Memory

To write an EEPROM data location, the user must first write the address to the EEADR register and the data to the EEDATA register. Then the user must follow a specific write sequence to initiate the write for each byte.

The write will not initiate if the write sequence is not exactly followed (write 55h to EECON2, write AAh to EECON2, then set WR bit) for each byte. We strongly recommend that interrupts be disabled during this code segment (see Example 3-2).

Additionally, the WREN bit in EECON1 must be set to enable write. This mechanism prevents accidental writes to data EEPROM due to errant (unexpected) code execution (i.e., lost programs). The user should keep the WREN bit clear at all times, except when updating EEPROM. The WREN bit is not cleared by hardware

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. The WR bit will be inhibited from being set unless the WREN bit is set. At the completion of the write cycle, the WR bit is cleared in hardware and the EE Write Complete Interrupt Flag bit (EEIF) is set. The user can either enable this interrupt or poll this bit. EEIF must be cleared by software.

The steps to write to EEPROM data memory are:

1. If step 10 is not implemented, check the WR bit to see if a write is in progress.
2. Write the address to EEADR. Make sure that the address is not larger than the memory size of the device.
3. Write the 8-bit data value to be programmed in the EEDATA register.
4. Clear the EEPGD bit to point to EEPROM data memory.
5. Set the WREN bit to enable program operations.
6. Disable interrupts (if enabled).
7. Execute the special five instruction sequence:
   - Write 55h to EECON2 in two steps (first to W, then to EECON2)
   - Write AAh to EECON2 in two steps (first to W, then to EECON2)
   - Set the WR bit
8. Enable interrupts (if using interrupts).
9. Clear the WREN bit to disable program operations.
10. At the completion of the write cycle, the WR bit is cleared and the EEIF interrupt flag bit is set. (EEIF must be cleared by firmware.) If step 1 is not implemented, then firmware should check for EEIF to be set, or WR to clear, to indicate the end of the program cycle.

EXAMPLE 3-2: DATA EEPROM WRITE

```
BSF     STATUS,RP1      ;
BSF     STATUS,RP0
BTFSC   EECON1,WR       ;Wait for write
GOTO    $-1             ;to complete
BCF     STATUS,  RP0    ;Bank 2
MOVF    DATA_EE_ADDR,W  ;Data Memory
MOVWF   EEADR           ;Address to write
MOVF    DATA_EE_DATA,W  ;Data Memory Value
MOVWF   EEDATA          ;to write
BSF     STATUS,RP0      ;Bank 3
BCF     EECON1,EEPGD    ;Point to DATA
                        ;memory
BSF     EECON1,WREN     ;Enable writes

BCF     INTCON,GIE      ;Disable INTs.
MOVLW   55h             ;
MOVWF   EECON2          ;Write 55h
MOVLW   AAh             ;
MOVWF   EECON2          ;Write AAh
BSF     EECON1,WR       ;Set WR bit to
                        ;begin write
BSF     INTCON,GIE      ;Enable INTs.
BCF     EECON1,WREN     ;Disable writes
```

Required Sequence

3.5 Reading Flash Program Memory

To read a program memory location, the user must write two bytes of the address to the EEADR and EEADRH registers, set the EEPGD control bit (EECON1<7>) and then set control bit RD (EECON1<0>). Once the read control bit is set, the program memory Flash controller will use the next two instruction cycles to read the data. This causes these two instructions immediately following the "`BSF EECON1,RD`" instruction to be ignored. The data is available in the very next cycle in the EEDATA and EEDATH registers; therefore, it can be read as two bytes in the following instructions. EEDATA and EEDATH registers will hold this value until another read or until it is written to by the user (during a write operation).

EXAMPLE 3-3: FLASH PROGRAM READ

```
        BSF     STATUS, RP1       ;
        BCF     STATUS, RP0       ; Bank 2
        MOVLW   MS_PROG_EE_ADDR   ;
        MOVWF   EEADRH            ; MS Byte of Program Address to read
        MOVLW   LS_PROG_EE_ADDR   ;
        MOVWF   EEADR             ; LS Byte of Program Address to read
        BSF     STATUS, RP0       ; Bank 3
        BSF     EECON1, EEPGD     ; Point to PROGRAM memory
        BSF     EECON1, RD        ; EE Read
;
Required Sequence
        NOP
        NOP                       ; Any instructions here are ignored as program
                                  ; memory is read in second cycle after BSF EECON1,RD
;
        BCF     STATUS, RP0       ; Bank 2
        MOVF    EEDATA, W         ; W = LS Byte of Program EEDATA
        MOVWF   DATAL             ;
        MOVF    EEDATH, W         ; W = MS Byte of Program EEDATA
        MOVWF   DATAH             ;
```


© 2003 Microchip Technology Inc.

3.6 Writing to Flash Program Memory

Flash program memory may only be written to if the destination address is in a segment of memory that is not write-protected, as defined in bits WRT1:WRT0 of the device configuration word (Register 14-1). Flash program memory must be written in four-word blocks. A block consists of four words with sequential addresses, with a lower boundary defined by an address, where EEADR<1:0> = 00. At the same time, all block writes to program memory are done as erase and write operations. The write operation is edge-aligned and cannot occur across boundaries.

To write program data, it must first be loaded into the buffer registers (see Figure 3-1). This is accomplished by first writing the destination address to EEADR and EEADRH and then writing the data to EEDATA and EEDATH. After the address and data have been set up, then the following sequence of events must be executed:

1. Set the EEPGD control bit (EECON1<7>).
2. Write 55h, then AAh, to EECON2 (Flash programming sequence).
3. Set the WR control bit (EECON1<1>).

All four buffer register locations **MUST** be written to with correct data. If only one, two or three words are being written to in the block of four words, then a read from the program memory location(s) not being written to must be performed. This takes the data from the program location(s) not being written and loads it into the EEDATA and EEDATH registers. Then the sequence of events to transfer data to the buffer registers must be executed.

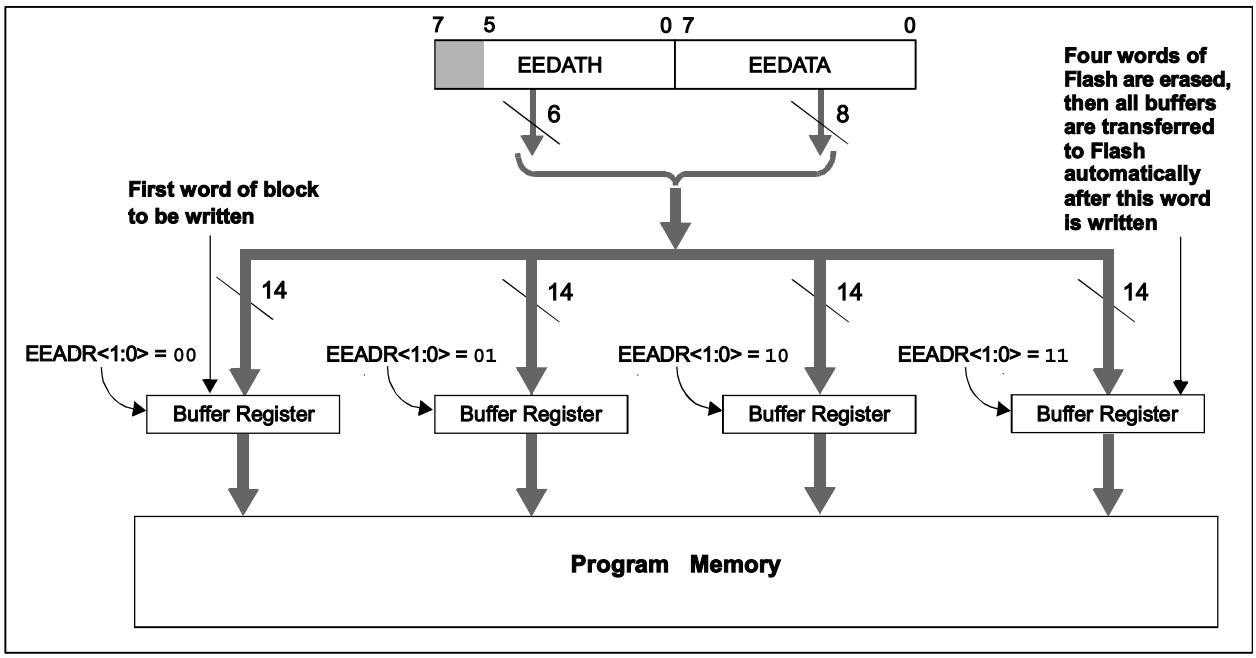
To transfer data from the buffer registers to the program memory, the EEADR and EEADRH must point to the last location in the four-word block (EEADR<1:0> = 11). Then the following sequence of events must be executed:

1. Set the EEPGD control bit (EECON1<7>).
2. Write 55h, then AAh, to EECON2 (Flash programming sequence).
3. Set control bit WR (EECON1<1>) to begin the write operation.

The user must follow the same specific sequence to initiate the write for each word in the program block, writing each program word in sequence (00,01,10,11). When the write is performed on the last word (EEADR<1:0> = 11), the block of four words are automatically erased and the contents of the buffer registers are written into the program memory.

After the "BSF EECON1,WR" instruction, the processor requires two cycles to set up the erase/write operation. The user must place two NOP instructions after the WR bit is set. Since data is being written to buffer registers, the writing of the first three words of the block appears to occur immediately. The processor will halt internal operations for the typical 4 ms, only during the cycle in which the erase takes place (i.e., the last word of the four-word block). This is not Sleep mode as the clocks and peripherals will continue to run. After the write cycle, the processor will resume operation with the third instruction after the EECON1 write instruction. If the sequence is performed to any other location, the action is ignored.

FIGURE 3-1: BLOCK WRITES TO FLASH PROGRAM MEMORY



An example of the complete four-word write sequence is shown in Example 3-4. The initial address is loaded into the EEADRH:EEADR register pair; the four words of data are loaded using indirect addressing.

EXAMPLE 3-4: WRITING TO FLASH PROGRAM MEMORY

```
; This write routine assumes the following:
;
; 1. A valid starting address (the least significant bits = '00')is loaded in ADDRH:ADDRL
; 2. The 8 bytes of data are loaded, starting at the address in DATADDR
; 3. ADDRH, ADDRL and DATADDR are all located in shared data memory 0x70 - 0x7f
;
        BSF     STATUS,RP1          ;
        BCF     STATUS,RP0          ; Bank 2
        MOVF    ADDRH,W             ; Load initial address
        MOVWF   EEADRH              ;
        MOVF    ADDRL,W             ;
        MOVWF   EEADR               ;
        MOVF    DATAADDR,W          ; Load initial data address
        MOVWF   FSR                 ;
LOOP    MOVF    INDF,W              ; Load first data byte into lower
        MOVWF   EEDATA              ;
        INCF    FSR,F               ; Next byte
        MOVF    INDF,W              ; Load second data byte into upper
        MOVWF   EEDATH              ;
        INCF    FSR,F               ;
        BSF     STATUS,RP0          ; Bank 3
        BSF     EECON1,EEPGD        ; Point to program memory
        BSF     EECON1,WREN         ; Enable writes
        BCF     INTCON,GIE          ; Disable interrupts (if using)
        MOVLW   55h                 ; Start of required write sequence:
        MOVWF   EECON2              ; Write 55h
        MOVLW   AAh                 ;
        MOVWF   EECON2              ; Write AAh
        BSF     EECON1,WR           ; Set WR bit to begin write
        NOP                         ; Any instructions here are ignored as processor
                                    ; halts to begin write sequence
        NOP                         ; processor will stop here and wait for write complete
                                    ; after write processor continues with 3rd instruction
        BCF     EECON1,WREN         ; Disable writes
        BSF     INTCON,GIE          ; Enable interrupts (if using)
        BCF     STATUS,RP0          ; Bank 2
        INCF    EEADR,F             ; Increment address
        MOVF    EEADR,W             ; Check if lower two bits of address are '00'
        ANDLW   0x03                ; Indicates when four words have been programmed
        XORLW   0x03                ;
        BTFSC   STATUS,Z            ; Exit if more than four words,
        GOTO    LOOP                ; Continue if less than four words
```

Required Sequence

© 2003 Microchip Technology Inc.

3.7 Protection Against Spurious Write

There are conditions when the device should not write to the data EEPROM or Flash program memory. To protect against spurious writes, various mechanisms have been built-in. On power-up, WREN is cleared. Also, the Power-up Timer (72 ms duration) prevents an EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

3.8 Operation During Code-Protect

When the data EEPROM is code-protected, the microcontroller can read and write to the EEPROM normally. However, all external access to the EEPROM is disabled. External write access to the program memory is also disabled.

When program memory is code-protected, the microcontroller can read and write to program memory normally, as well as execute instructions. Writes by the device may be selectively inhibited to regions of the memory depending on the setting of bits WR1:WR0 of the configuration word (see **Section 14.1 "Configuration Bits"** for additional information). External access to the memory is also disabled.

TABLE 3-1: REGISTERS/BITS ASSOCIATED WITH DATA EEPROM AND FLASH PROGRAM MEMORIES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other Resets
10Ch	EEDATA	EEPROM/Flash Data Register Low Byte								xxxx xxxx	uuuu uuuu
10Dh	EEADR	EEPROM/Flash Address Register Low Byte								xxxx xxxx	uuuu uuuu
10Eh	EEDATH	—	—	EEPROM/Flash Data Register High Byte						xxxx xxxx	---0 q000
10Fh	EEADRH	—	—	—	EEPROM/Flash Address Register High Byte					xxxx xxxx	---- ----
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	---0 q000
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	---- ----
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	-0-0 0--0
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	-0-0 0--0

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends upon condition.
Shaded cells are not used by data EEPROM or Flash program memory.

NOTES:

© 2003 Microchip Technology Inc.

4.0 I/O PORTS

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

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual (DS33023).

4.1 PORTA and the TRISA Register

PORTA is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, the value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open-drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

Other PORTA pins are multiplexed with analog inputs and the analog VREF input for both the A/D converters and the comparators. The operation of each pin is selected by clearing/setting the appropriate control bits in the ADCON1 and/or CMCON registers.

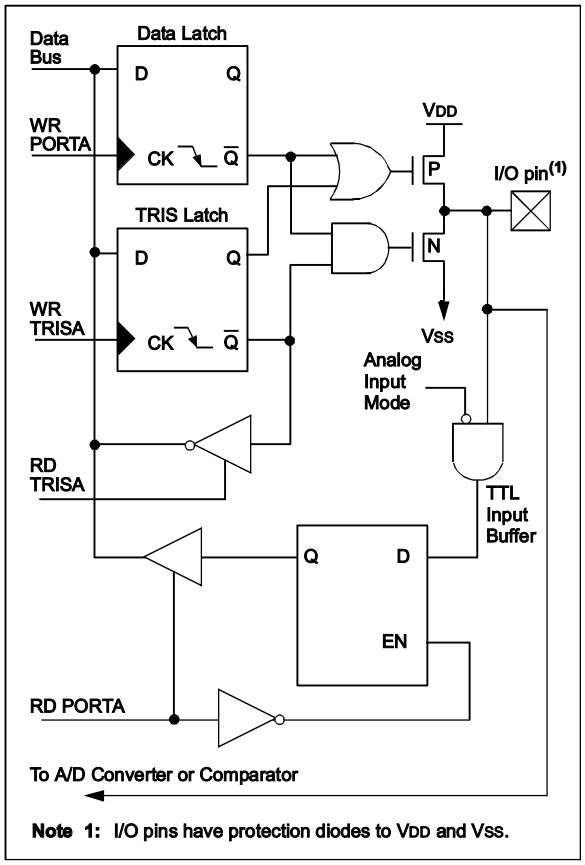
Note:	On a Power-on Reset, these pins are configured as analog inputs and read as '0'. The comparators are in the off (digital) state.

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

EXAMPLE 4-1: INITIALIZING PORTA

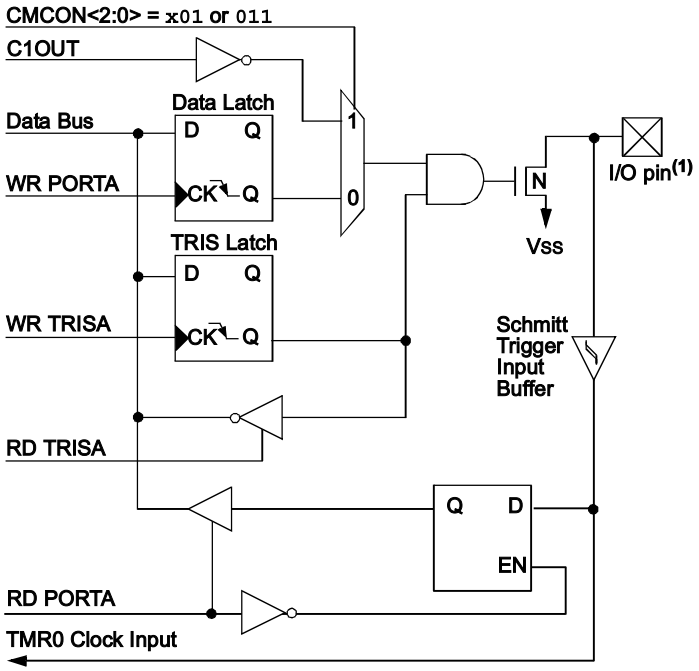
```
BCF     STATUS, RP0   ;
BCF     STATUS, RP1   ; Bank0
CLRF    PORTA         ; Initialize PORTA by
                      ; clearing output
                      ; data latches
BSF     STATUS, RP0   ; Select Bank 1
MOVLW   0x06          ; Configure all pins
MOVWF   ADCON1        ; as digital inputs
MOVLW   0xCF          ; Value used to
                      ; initialize data
                      ; direction
MOVWF   TRISA         ; Set RA<3:0> as inputs
                      ; RA<5:4> as outputs
                      ; TRISA<7:6>are always
                      ; read as '0'.
```

FIGURE 4-1: BLOCK DIAGRAM OF RA3:RA0 PINS



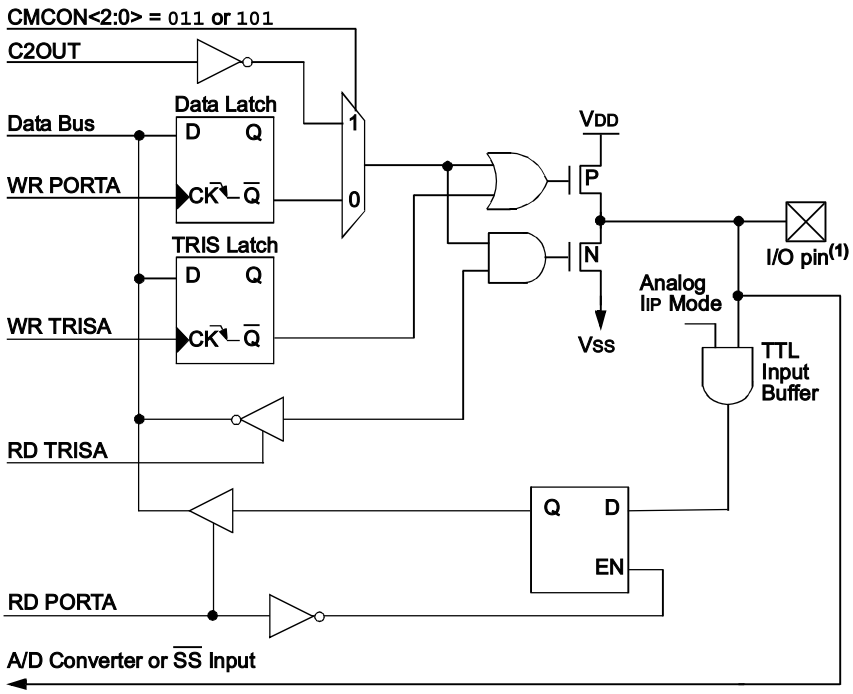
Note 1: I/O pins have protection diodes to VDD and VSS.

FIGURE 4-2: BLOCK DIAGRAM OF RA4/T0CKI PIN



Note 1: I/O pin has protection diodes to Vss only.

FIGURE 4-3: BLOCK DIAGRAM OF RA5 PIN



Note 1: I/O pin has protection diodes to VDD and Vss.

© 2003 Microchip Technology Inc.

TABLE 4-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	TTL	Input/output or analog input.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-/CVREF	bit 2	TTL	Input/output or analog input or VREF- or CVREF.
RA3/AN3/VREF+	bit 3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI/C1OUT	bit 4	ST	Input/output or external clock input for Timer0 or comparator output. Output is open-drain type.
RA5/AN4/$\overline{SS}$/C2OUT	bit 5	TTL	Input/output or analog input or slave select input for synchronous serial port or comparator output.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 4-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

Note: When using the SSP module in SPI Slave mode and $\overline{SS}$ enabled, the A/D converter must be set to one of the following modes, where PCFG3:PCFG0 = 0100, 0101, 011x, 1101, 1110, 1111.

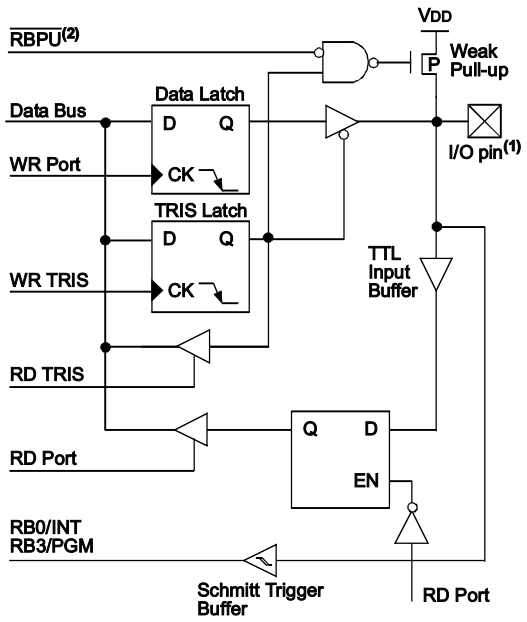
4.2 PORTB and the TRISB Register

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Three pins of PORTB are multiplexed with the In-Circuit Debugger and Low-Voltage Programming function: RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in **Section 14.0 "Special Features of the CPU"**.

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

FIGURE 4-4: BLOCK DIAGRAM OF RB3:RB0 PINS



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

Four of the PORTB pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB port change interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.

b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

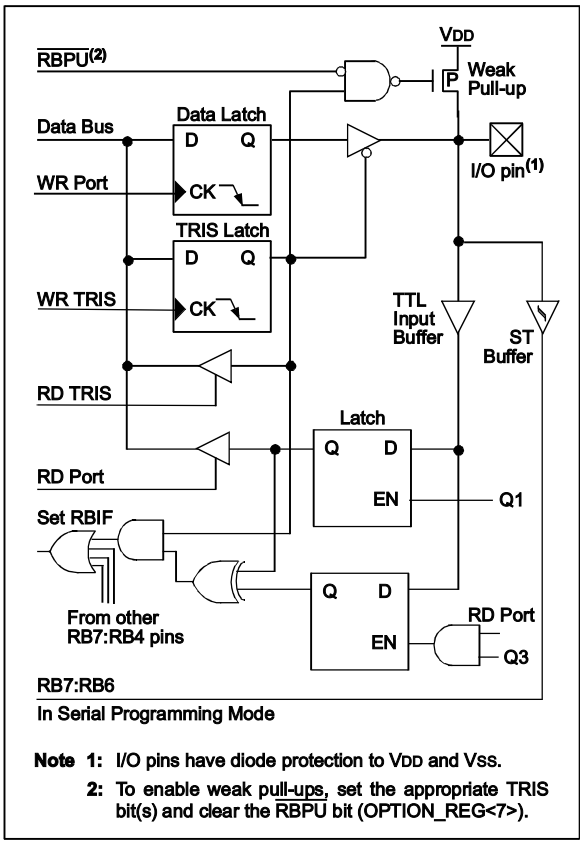
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

This interrupt-on-mismatch feature, together with software configurable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression. Refer to the application note, *AN552, "Implementing Wake-up on Key Stroke"* (DS00552).

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

RB0/INT is discussed in detail in **Section 14.11.1 "INT Interrupt"**.

FIGURE 4-5: BLOCK DIAGRAM OF RB7:RB4 PINS



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

© 2003 Microchip Technology Inc.

TABLE 4-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT	bit 0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit 1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit 2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM(3)	bit 3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit 4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit 5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/PGC	bit 6	TTL/ST(2)	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit 7	TTL/ST(2)	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: 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 or in-circuit debugger.

3: Low-Voltage ICSP Programming (LVP) is enabled by default which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

TABLE 4-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

4.3 PORTC and the TRISC Register

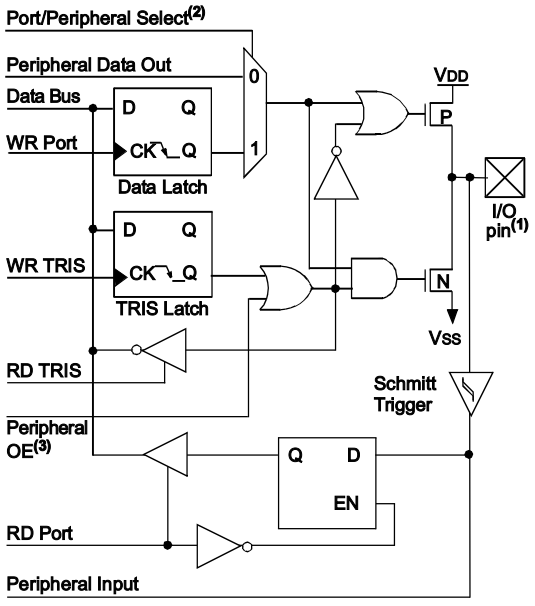
PORTC is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

PORTC is multiplexed with several peripheral functions (Table 4-5). PORTC pins have Schmitt Trigger input buffers.

When the I^2C module is enabled, the PORTC<4:3> pins can be configured with normal I^2C levels, or with SMBus levels, by using the CKE bit (SSPSTAT<6>).

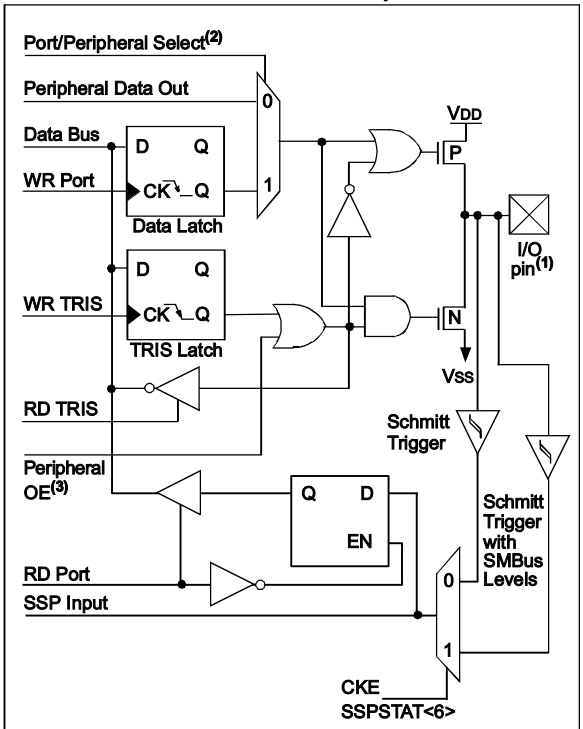
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (BSF, BCF, XORWF) with TRISC as the destination, should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

FIGURE 4-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<2:0>, RC<7:5>



Note 1: I/O pins have diode protection to VDD and VSS.
2: Port/Peripheral Select signal selects between port data and peripheral output.
3: Peripheral OE (Output Enable) is only activated if Peripheral Select is active.

FIGURE 4-7: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<4:3>



Note 1: I/O pins have diode protection to VDD and VSS.
2: Port/Peripheral Select signal selects between port data and peripheral output.
3: Peripheral OE (Output Enable) is only activated if Peripheral Select is active.

© 2003 Microchip Technology Inc.

TABLE 4-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit 0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit 1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit 2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit 3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit 4	ST	RC4 can also be the SPI data in (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit 5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit 6	ST	Input/output port pin or USART asynchronous transmit or synchronous clock.
RC7/RX/DT	bit 7	ST	Input/output port pin or USART asynchronous receive or synchronous data.

Legend: ST = Schmitt Trigger input

TABLE 4-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

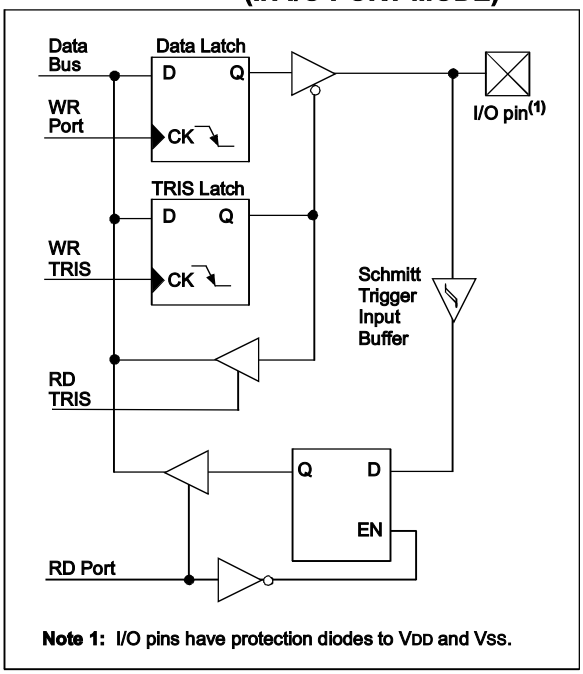
4.4 PORTD and TRISD Registers

Note: PORTD and TRISD are not implemented on the 28-pin devices.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

PORTD can be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

FIGURE 4-8: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 4-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit 0	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 0.
RD1/PSP1	bit 1	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 2.
RD3/PSP3	bit 3	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 3.
RD4/PSP4	bit 4	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 4.
RD5/PSP5	bit 5	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 5.
RD6/PSP6	bit 6	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 6.
RD7/PSP7	bit 7	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 4-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

© 2003 Microchip Technology Inc.

4.5 PORTE and TRISE Register

Note: PORTE and TRISE are not implemented on the 28-pin devices.

PORTE has three pins (RE0/$\overline{RD}$/AN5, RE1/$\overline{WR}$/AN6 and RE2/$\overline{CS}$/AN7) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make certain that the TRISE<2:0> bits are set and that the pins are configured as digital inputs. Also, ensure that ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

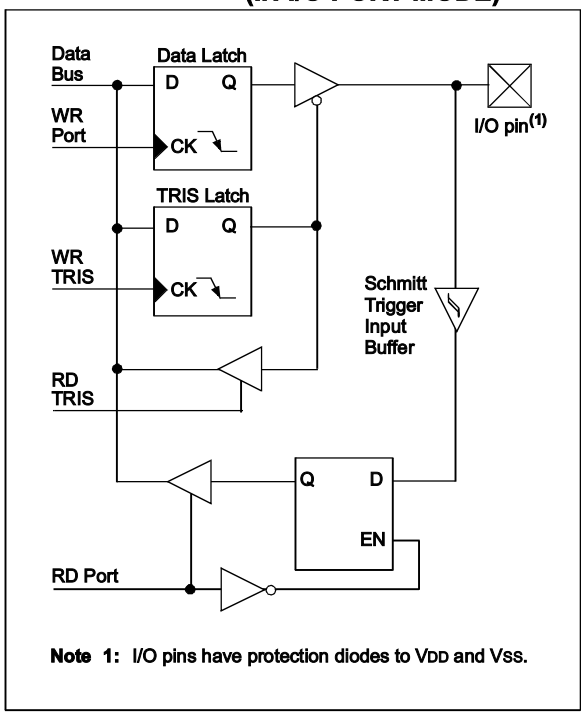
Register 4-1 shows the TRISE register which also controls the Parallel Slave Port operation.

PORTE pins are multiplexed with analog inputs. When selected for analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Note: On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

FIGURE 4-9: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 4-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit 0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected).
RE1/$\overline{WR}$/AN6	bit 1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected).
RE2/$\overline{CS}$/AN7	bit 2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 4-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

REGISTER 4-1: TRISE REGISTER (ADDRESS 89h)

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	Bit 2	Bit 1	Bit 0
bit 7							bit 0

Parallel Slave Port Status/Control Bits:

bit 7 **IBF:** Input Buffer Full Status bit
1 = A word has been received and is waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF**: Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV**: Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE**: Parallel Slave Port Mode Select bit
1 = PORTD functions in Parallel Slave Port mode
0 = PORTD functions in general purpose I/O mode

bit 3 **Unimplemented**: Read as '0'

PORTE Data Direction Bits:

bit 2 **Bit 2**: Direction Control bit for pin RE2/$\overline{CS}$/AN7
1 = Input
0 = Output

bit 1 **Bit 1**: Direction Control bit for pin RE1/$\overline{WR}$/AN6
1 = Input
0 = Output

bit 0 **Bit 0**: Direction Control bit for pin RE0/$\overline{RD}$/AN5
1 = Input
0 = Output

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

4.6 Parallel Slave Port

The Parallel Slave Port (PSP) is not implemented on the PIC16F873A or PIC16F876A.

PORTD operates as an 8-bit wide Parallel Slave Port, or microprocessor port, when control bit PSPMODE (TRISE<4>) is set. In Slave mode, it is asynchronously readable and writable by the external world through $\overline{RD}$ control input pin, RE0/$\overline{RD}$/AN5, and $\overline{WR}$ control input pin, RE1/$\overline{WR}$/AN6.

The PSP can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting bit PSPMODE enables port pin RE0/$\overline{RD}$/AN5 to be the $\overline{RD}$ input, RE1/$\overline{WR}$/AN6 to be the $\overline{WR}$ input and RE2/$\overline{CS}$/AN7 to be the $\overline{CS}$ (Chip Select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (set). The A/D port configuration bits, PCFG3:PCFG0 (ADCON1<3:0>), must be set to configure pins RE2:RE0 as digital I/O.

There are actually two 8-bit latches: one for data output and one for data input. The user writes 8-bit data to the PORTD data latch and reads data from the port pin latch (note that they have the same address). In this mode, the TRISD register is ignored since the external device is controlling the direction of data flow.

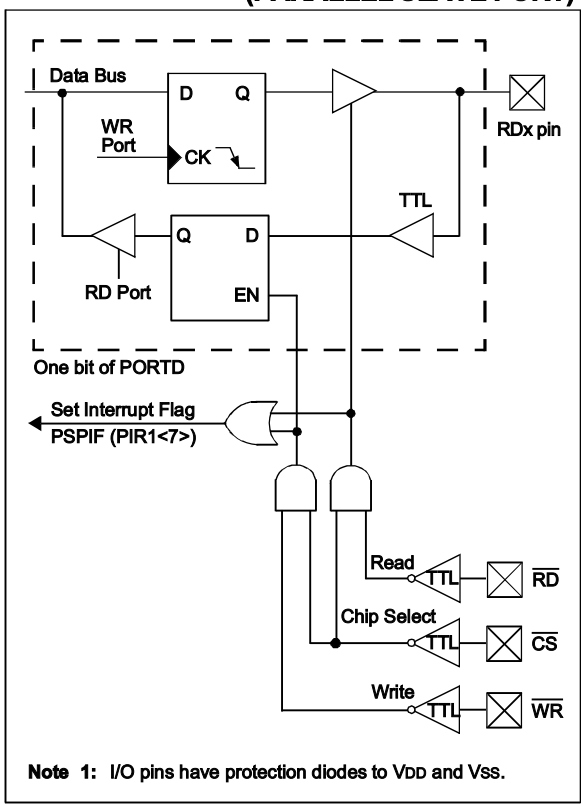
A write to the PSP occurs when both the $\overline{CS}$ and $\overline{WR}$ lines are first detected low. When either the $\overline{CS}$ or $\overline{WR}$ lines become high (level triggered), the Input Buffer Full (IBF) status flag bit (TRISE<7>) is set on the Q4 clock cycle, following the next Q2 cycle, to signal the write is complete (Figure 4-11). The interrupt flag bit, PSPIF (PIR1<7>), is also set on the same Q4 clock cycle. IBF can only be cleared by reading the PORTD input latch. The Input Buffer Overflow (IBOV) status flag bit (TRISE<5>) is set if a second write to the PSP is attempted when the previous byte has not been read out of the buffer.

A read from the PSP occurs when both the $\overline{CS}$ and $\overline{RD}$ lines are first detected low. The Output Buffer Full (OBF) status flag bit (TRISE<6>) is cleared immediately (Figure 4-12), indicating that the PORTD latch is waiting to be read by the external bus. When either the $\overline{CS}$ or $\overline{RD}$ pin becomes high (level triggered), the interrupt flag bit PSPIF is set on the Q4 clock cycle, following the next Q2 cycle, indicating that the read is complete. OBF remains low until data is written to PORTD by the user firmware.

When not in PSP mode, the IBF and OBF bits are held clear. However, if flag bit IBOV was previously set, it must be cleared in firmware.

An interrupt is generated and latched into flag bit PSPIF when a read or write operation is completed. PSPIF must be cleared by the user in firmware and the interrupt can be disabled by clearing the interrupt enable bit PSPIE (PIE1<7>).

FIGURE 4-10: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)



Note 1: I/O pins have protection diodes to VDD and VSS.

FIGURE 4-11: PARALLEL SLAVE PORT WRITE WAVEFORMS

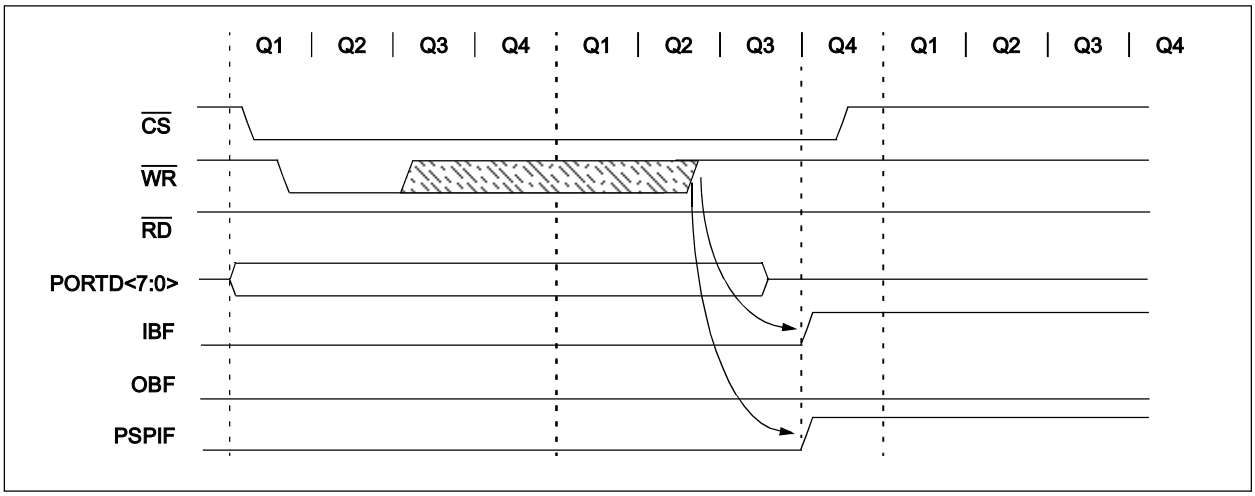


FIGURE 4-12: PARALLEL SLAVE PORT READ WAVEFORMS

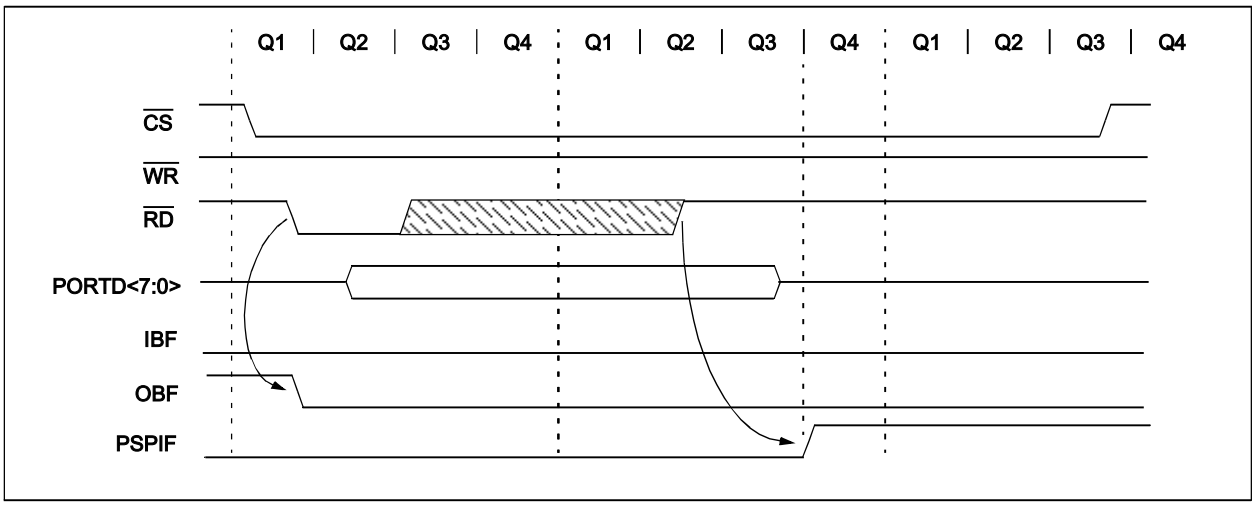


TABLE 4-11: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
08h	PORTD	Port Data Latch when written; Port pins when read								xxxx xxxx	uuuu uuuu
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Parallel Slave Port.

Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873A/876A; always maintain these bits clear.

© 2003 Microchip Technology Inc.

5.0 TIMER0 MODULE

The Timer0 module timer/counter has the following features:

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

Figure 5-1 is a block diagram of the Timer0 module and the prescaler shared with the WDT.

Additional information on the Timer0 module is available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

Timer mode is selected by clearing bit T0CS (OPTION_REG<5>). In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

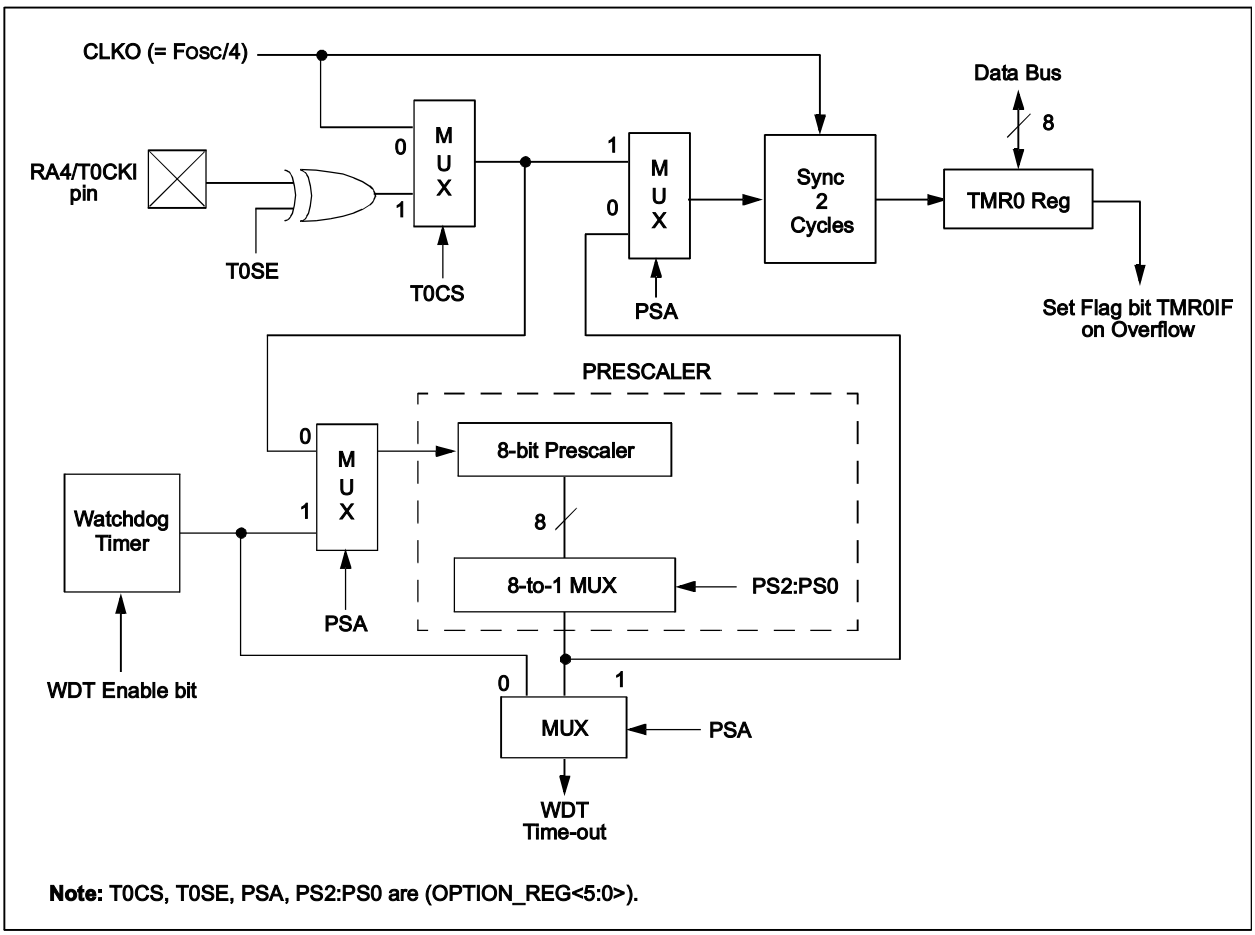
Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In Counter mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (OPTION_REG<4>). Clearing bit T0SE selects the rising edge. Restrictions on the external clock input are discussed in detail in **Section 5.2 "Using Timer0 with an External Clock"**.

The prescaler is mutually exclusively shared between the Timer0 module and the Watchdog Timer. The prescaler is not readable or writable. **Section 5.3 "Prescaler"** details the operation of the prescaler.

5.1 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit TMR0IF (INTCON<2>). The interrupt can be masked by clearing bit TMR0IE (INTCON<5>). Bit TMR0IF must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from Sleep since the timer is shut-off during Sleep.

FIGURE 5-1: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

5.2 Using Timer0 with an External Clock

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks. Therefore, it is necessary for T0CKI to be high for at least 2 TOSC (and a small RC delay of 20 ns) and low for at least 2 TOSC (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

5.3 Prescaler

There is only one prescaler available which is mutually exclusively shared between the Timer0 module and the Watchdog Timer. A prescaler assignment for the Timer0 module means that there is no prescaler for the Watchdog Timer and vice versa. This prescaler is not readable or writable (see Figure 5-1).

The PSA and PS2:PS0 bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF 1`, `MOVWF 1`, `BSF 1,x`....etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

Note:	Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count, but will not change the prescaler assignment.

REGISTER 5-1: 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

bit 7 **$\overline{\text{RBPU}}$**

bit 6 **INTEDG**

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note:	To avoid an unintended device Reset, the instruction sequence shown in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

© 2003 Microchip Technology Inc.

TABLE 5-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
01h,101h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

NOTES:

© 2003 Microchip Technology Inc.

6.0 TIMER1 MODULE

The Timer1 module is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L) which are readable and writable. The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit, TMR1IE (PIE1<0>).

Timer1 can operate in one of two modes:

- As a Timer
- As a Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

In Timer mode, Timer1 increments every instruction cycle. In Counter mode, it increments on every rising edge of the external clock input.

Timer1 can be enabled/disabled by setting/clearing control bit, TMR1ON (T1CON<0>).

Timer1 also has an internal "Reset input". This Reset can be generated by either of the two CCP modules (**Section 8.0 "Capture/Compare/PWM Modules"**). Register 6-1 shows the Timer1 Control register.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI/CCP2 and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored and these pins read as '0'.

Additional information on timer modules is available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

REGISTER 6-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 prescale value
10 = 1:4 prescale value
01 = 1:2 prescale value
00 = 1:1 prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)

bit 2 **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON**: Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

6.1 Timer1 Operation in Timer Mode

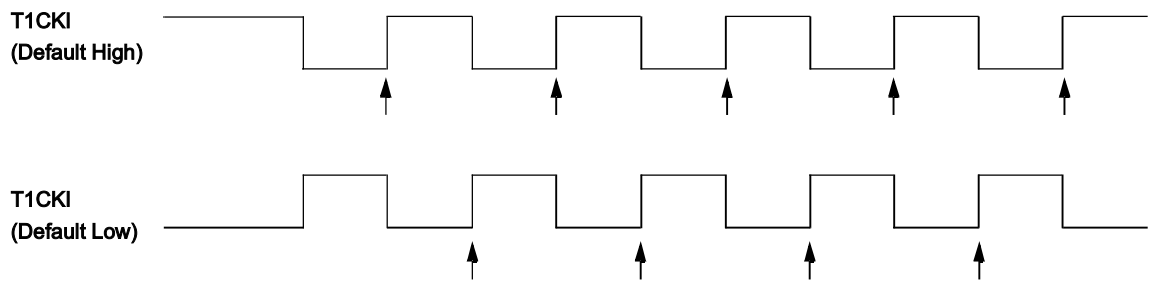
Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is FOSC/4. The synchronize control bit, $\overline{\text{T1SYNC}}$ (T1CON<2>), has no effect since the internal clock is always in sync.

6.2 Timer1 Counter Operation

Timer1 may operate in either a Synchronous, or an Asynchronous mode, depending on the setting of the TMR1CS bit.

When Timer1 is being incremented via an external source, increments occur on a rising edge. After Timer1 is enabled in Counter mode, the module must first have a falling edge before the counter begins to increment.

FIGURE 6-1: TIMER1 INCREMENTING EDGE



Note: Arrows indicate counter increments.

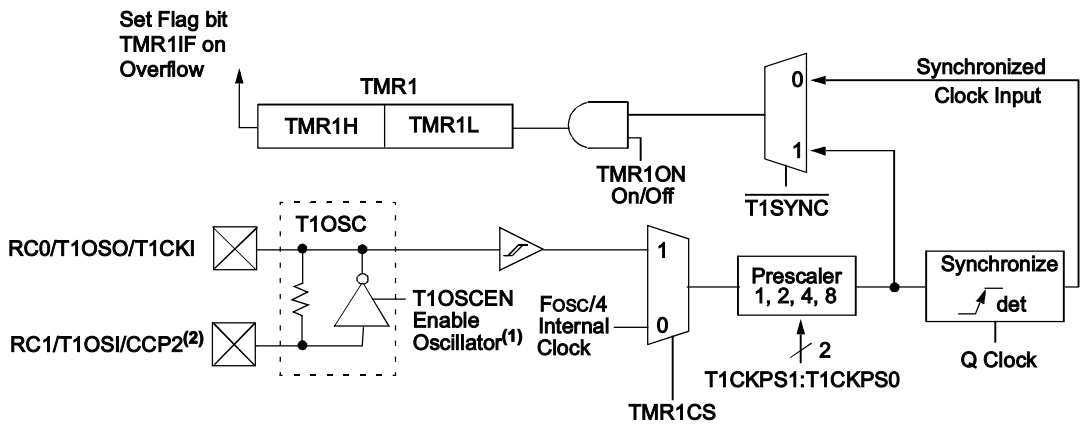
6.3 Timer1 Operation in Synchronized Counter Mode

Counter mode is selected by setting bit TMR1CS. In this mode, the timer increments on every rising edge of clock input on pin RC1/T1OSI/CCP2 when bit T1OSCEN is set, or on pin RC0/T1OSO/T1CKI when bit T1OSCEN is cleared.

If $\overline{\text{T1SYNC}}$ is cleared, then the external clock input is synchronized with internal phase clocks. The synchronization is done after the prescaler stage. The prescaler stage is an asynchronous ripple counter.

In this configuration, during Sleep mode, Timer1 will not increment even if the external clock is present since the synchronization circuit is shut-off. The prescaler, however, will continue to increment.

FIGURE 6-2: TIMER1 BLOCK DIAGRAM



Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

© 2003 Microchip Technology Inc.

6.4 Timer1 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{T1SYNC}}$ (T1CON<2>) is set, the external clock input is not synchronized. The timer continues to increment asynchronous to the internal phase clocks. The timer will continue to run during Sleep and can generate an interrupt-on-overflow which will wake-up the processor. However, special precautions in software are needed to read/write the timer.

In Asynchronous Counter mode, Timer1 cannot be used as a time base for capture or compare operations.

6.4.1 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L while the timer is running from an external asynchronous clock will ensure a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers while the register is incrementing. This may produce an unpredictable value in the timer register.

Reading the 16-bit value requires some care. Examples 12-2 and 12-3 in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023) show how to read and write Timer1 when it is running in Asynchronous mode.

6.5 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit, T1OSCEN (T1CON<3>). The oscillator is a low-power oscillator, rated up to 200 kHz. It will continue to run during Sleep. It is primarily intended for use with a 32 kHz crystal. Table 6-1 shows the capacitor selection for the Timer1 oscillator.

The Timer1 oscillator is identical to the LP oscillator. The user must provide a software time delay to ensure proper oscillator start-up.

TABLE 6-1: CAPACITOR SELECTION FOR THE TIMER1 OSCILLATOR

Osc Type	Freq.	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF
These values are for design guidance only.			
Crystals Tested:			
32.768 kHz	Epson C-001R32.768K-A		± 20 PPM
100 kHz	Epson C-2 100.00 KC-P		± 20 PPM
200 kHz	STD XTL 200.000 kHz		± 20 PPM

Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

6.6 Resetting Timer1 Using a CCP Trigger Output

If the CCP1 or CCP2 module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer1.

Note:	The special event triggers from the CCP1 and CCP2 modules will not set interrupt flag bit, TMR1IF (PIR1<0>).

Timer1 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1 or CCP2, the write will take precedence.

In this mode of operation, the CCPRxH:CCPRxL register pair effectively becomes the period register for Timer1.

6.7 Resetting of Timer1 Register Pair (TMR1H, TMR1L)

TMR1H and TMR1L registers are not reset to 00h on a POR, or any other Reset, except by the CCP1 and CCP2 special event triggers.

T1CON register is reset to 00h on a Power-on Reset, or a Brown-out Reset, which shuts off the timer and leaves a 1:1 prescale. In all other Resets, the register is unaffected.

6.8 Timer1 Prescaler

The prescaler counter is cleared on writes to the TMR1H or TMR1L registers.

TABLE 6-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

© 2003 Microchip Technology Inc.

7.0 TIMER2 MODULE

Timer2 is an 8-bit timer with a prescaler and a postscaler. It can be used as the PWM time base for the PWM mode of the CCP module(s). The TMR2 register is readable and writable and is cleared on any device Reset.

The input clock (Fosc/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits T2CKPS1:T2CKPS0 (T2CON<1:0>).

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon Reset.

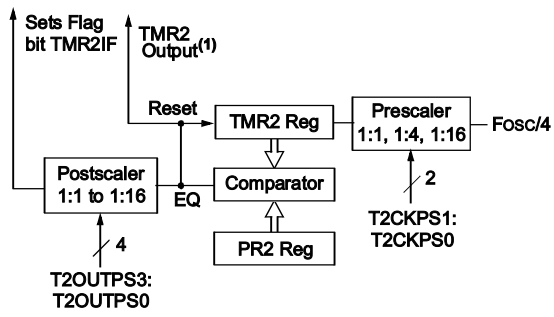
The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit, TMR2IF (PIR1<1>)).

Timer2 can be shut-off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

Register 7-1 shows the Timer2 Control register.

Additional information on timer modules is available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

FIGURE 7-1: TIMER2 BLOCK DIAGRAM



Note 1: TMR2 register output can be software selected by the SSP module as a baud clock.

REGISTER 7-1: T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 postscale
0001 = 1:2 postscale
0010 = 1:3 postscale
•
•
•
1111 = 1:16 postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.1 Timer2 Prescaler and Postscaler

The prescaler and postscaler counters are cleared when any of the following occurs:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (POR, $\overline{MCLR}$ Reset, WDT Reset or BOR)

TMR2 is not cleared when T2CON is written.

7.2 Output of TMR2

The output of TMR2 (before the postscaler) is fed to the SSP module, which optionally uses it to generate the shift clock.

TABLE 7-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
11h	TMR2	Timer2 Module's Register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

© 2003 Microchip Technology Inc.

8.0 CAPTURE/COMPARE/PWM MODULES

Each Capture/Compare/PWM (CCP) module contains a 16-bit register which can operate as a:

- 16-bit Capture register
- 16-bit Compare register
- PWM Master/Slave Duty Cycle register

Both the CCP1 and CCP2 modules are identical in operation, with the exception being the operation of the special event trigger. Table 8-1 and Table 8-2 show the resources and interactions of the CCP module(s). In the following sections, the operation of a CCP module is described with respect to CCP1. CCP2 operates the same as CCP1 except where noted.

CCP1 Module:

Capture/Compare/PWM Register 1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. The special event trigger is generated by a compare match and will reset Timer1.

CCP2 Module:

Capture/Compare/PWM Register 2 (CCPR2) is comprised of two 8-bit registers: CCPR2L (low byte) and CCPR2H (high byte). The CCP2CON register controls the operation of CCP2. The special event trigger is generated by a compare match and will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

Additional information on CCP modules is available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023) and in application note *AN594, "Using the CCP Module(s)"* (DS00594).

TABLE 8-1: CCP MODE – TIMER RESOURCES REQUIRED

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

TABLE 8-2: INTERACTION OF TWO CCP MODULES

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time base
Capture	Compare	The compare should be configured for the special event trigger which clears TMR1
Compare	Compare	The compare(s) should be configured for the special event trigger which clears TMR1
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt)
PWM	Capture	None
PWM	Compare	None

REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY**: PWM Least Significant bits

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

8.1 Capture Mode

In Capture mode, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 register when an event occurs on pin RC2/CCP1. An event is defined as one of the following:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

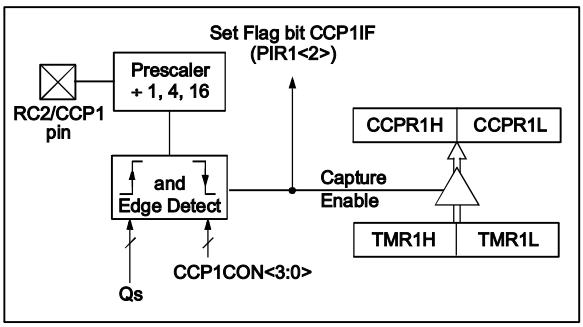
The type of event is configured by control bits, CCP1M3:CCP1M0 (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCP1IF (PIR1<2>), is set. The interrupt flag must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value is overwritten by the new value.

8.1.1 CCP PIN CONFIGURATION

In Capture mode, the RC2/CCP1 pin should be configured as an input by setting the TRISC<2> bit.

Note:	If the RC2/CCP1 pin is configured as an output, a write to the port can cause a Capture condition.

FIGURE 8-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



8.1.2 TIMER1 MODE SELECTION

Timer1 must be running in Timer mode, or Synchronized Counter mode, for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

8.1.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCP1IE (PIE1<2>) clear to avoid false interrupts and should clear the flag bit, CCP1IF, following any such change in operating mode.

8.1.4 CCP PRESCALER

There are four prescaler settings, specified by bits CCP1M3:CCP1M0. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared, therefore, the first capture may be from a non-zero prescaler. Example 8-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the "false" interrupt.

EXAMPLE 8-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF    CCP1CON      ; Turn CCP module off
MOVLW   NEW_CAPT_PS  ; Load the W reg with
                     ; the new prescaler
                     ; move value and CCP ON
MOVWF   CCP1CON      ; Load CCP1CON with this
                     ; value
```

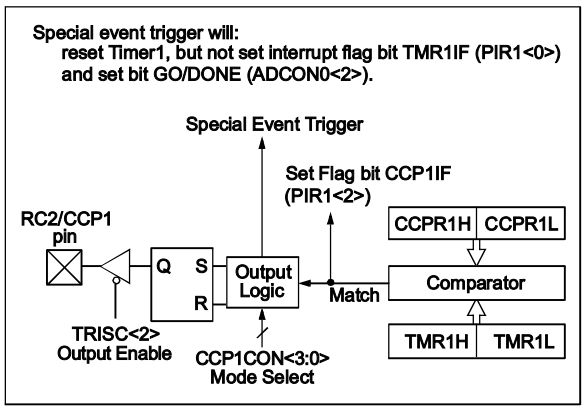
8.2 Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against the TMR1 register pair value. When a match occurs, the RC2/CCP1 pin is:

- Driven high
- Driven low
- Remains unchanged

The action on the pin is based on the value of control bits, CCP1M3:CCP1M0 (CCP1CON<3:0>). At the same time, interrupt flag bit CCP1IF is set.

FIGURE 8-2: COMPARE MODE OPERATION BLOCK DIAGRAM



8.2.1 CCP PIN CONFIGURATION

The user must configure the RC2/CCP1 pin as an output by clearing the TRISC<2> bit.

Note:	Clearing the CCP1CON register will force the RC2/CCP1 compare output latch to the default low level. This is not the PORTC I/O data latch.

8.2.2 TIMER1 MODE SELECTION

Timer1 must be running in Timer mode, or Synchronized Counter mode, if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

8.2.3 SOFTWARE INTERRUPT MODE

When Generate Software Interrupt mode is chosen, the CCP1 pin is not affected. The CCPIF bit is set, causing a CCP interrupt (if enabled).

8.2.4 SPECIAL EVENT TRIGGER

In this mode, an internal hardware trigger is generated which may be used to initiate an action.

The special event trigger output of CCP1 resets the TMR1 register pair. This allows the CCPR1 register to effectively be a 16-bit programmable period register for Timer1.

The special event trigger output of CCP2 resets the TMR1 register pair and starts an A/D conversion (if the A/D module is enabled).

Note:	The special event trigger from the CCP1 and CCP2 modules will not set interrupt flag bit TMR1IF (PIR1<0>).

© 2003 Microchip Technology Inc.

8.3 PWM Mode (PWM)

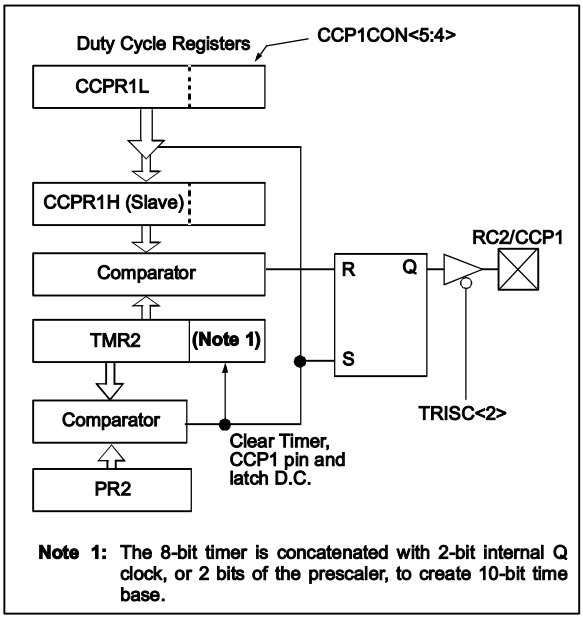
In Pulse Width Modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Note:	Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 8-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see **Section 8.3.3 "Setup for PWM Operation"**.

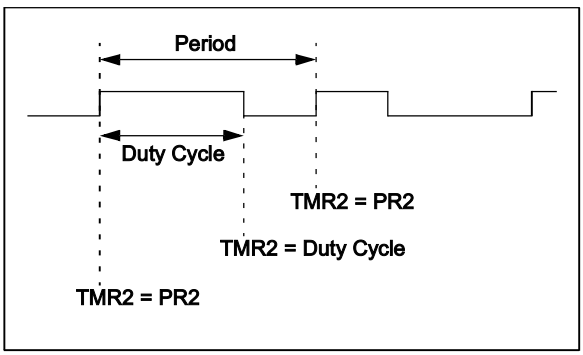
FIGURE 8-3: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: The 8-bit timer is concatenated with 2-bit internal Q clock, or 2 bits of the prescaler, to create 10-bit time base.

A PWM output (Figure 8-4) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 8-4: PWM OUTPUT



8.3.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note:	The Timer2 postscaler (see **Section 7.1 "Timer2 Prescaler and Postscaler"**) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

8.3.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitch-free PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the following formula.

EQUATION 8-1:

$$\text{Resolution} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

8.3.3 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the TRISC<2> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

TABLE 8-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 20 MHz

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12kHz	156.3 kHz	208.3 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFFh	0xFFh	0xFFh	0x3Fh	0x1Fh	0x17h
Maximum Resolution (bits)	10	10	10	8	7	5.5

TABLE 8-4: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE AND TIMER1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by Capture and Timer1.

Note 1: The PSP is not implemented on 28-pin devices; always maintain these bits clear.

© 2003 Microchip Technology Inc.

TABLE 8-5: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
11h	TMR2	Timer2 Module's Register								0000 0000	0000 0000
92h	PR2	Timer2 Module's Period Register								1111 1111	1111 1111
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PWM and Timer2.

Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

NOTES:

© 2003 Microchip Technology Inc.

9.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

9.1 Master SSP (MSSP) Module Overview

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

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)
  - Full Master mode
  - Slave mode (with general address call)

The I^2C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

9.2 Control Registers

The MSSP module has three associated registers. These include a status register (SSPSTAT) and two control registers (SSPCON and SSPCON2). The use of these registers and their individual configuration bits differ significantly, depending on whether the MSSP module is operated in SPI or I^2C mode.

Additional details are provided under the individual sections.

9.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

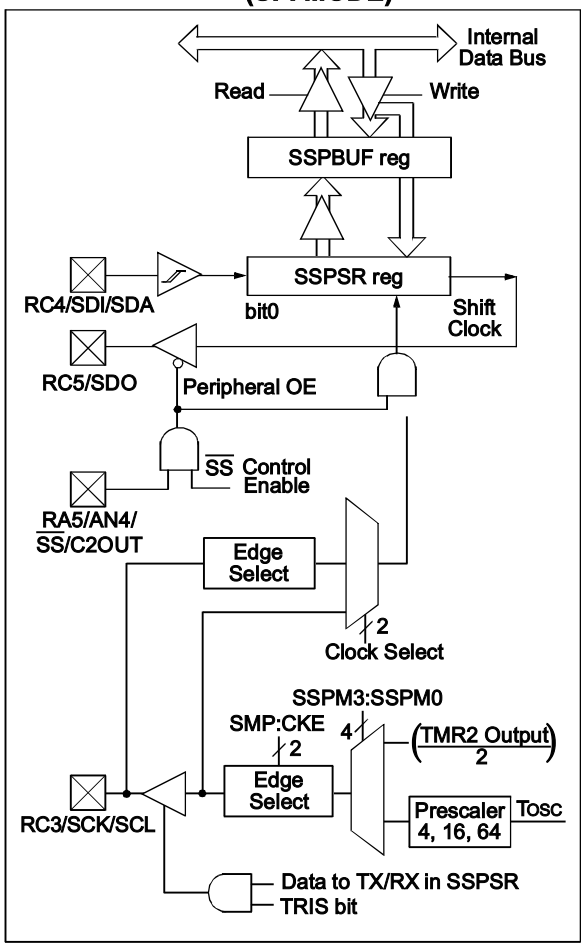
- Serial Data Out (SDO) – RC5/SDO
- Serial Data In (SDI) – RC4/SDI/SDA
- Serial Clock (SCK) – RC3/SCK/SCL

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SS}$) – RA5/AN4/$\overline{SS}$/C2OUT

Figure 9-1 shows the block diagram of the MSSP module when operating in SPI mode.

FIGURE 9-1: MSSP BLOCK DIAGRAM (SPI MODE)



Note:	When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON<3:0> = 0100), the state of the $\overline{SS}$ pin can affect the state read back from the TRISC<5> bit. The Peripheral OE signal from the SSP module in PORTC controls the state that is read back from the TRISC<5> bit (see **Section 4.3 "PORTC and the TRISC Register"** for information on PORTC). If Read-Modify-Write instructions, such as BSF, are performed on the TRISC register while the $\overline{SS}$ pin is high, this will cause the TRISC<5> bit to be set, thus disabling the SDO output.

9.3.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register (SSPCON)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON register is readable and writable. The lower six bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

REGISTER 9-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE) (ADDRESS 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Select bit
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state
Note: Polarity of clock state is set by the CKP bit (SSPCON1<4>).

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit
Used in I^2C mode only.

bit 4 **P:** Stop bit
Used in I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** Start bit
Used in I^2C mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ bit information
Used in I^2C mode only.

bit 1 **UA:** Update Address bit
Used in I^2C mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

REGISTER 9-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE) (ADDRESS 14h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)

1 = The SSPBUF register is written while it is still transmitting the previous word. (Must be cleared in software.)

0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit

SPI Slave mode:

1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow. (Must be cleared in software.)

0 = No overflow

Note: In Master mode, the overflow bit is not set, since each new reception (and transmission) is initiated by writing to the SSPBUF register.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables serial port and configures SCK, SDO, SDI, and $\overline{SS}$ as serial port pins

0 = Disables serial port and configures these pins as I/O port pins

Note: When enabled, these pins must be properly configured as input or output.

bit 4 **CKP:** Clock Polarity Select bit

1 = Idle state for clock is a high level

0 = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

0101 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ can be used as I/O pin.

0100 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control enabled.

0011 = SPI Master mode, clock = TMR2 output/2

0010 = SPI Master mode, clock = FOSC/64

0001 = SPI Master mode, clock = FOSC/16

0000 = SPI Master mode, clock = FOSC/4

Note: Bit combinations not specifically listed here are either reserved or implemented in I^2C mode only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

9.3.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the eight bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full detect bit, BF (SSPSTAT<0>), and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the write collision detect bit, WCOL (SSPCON<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. Buffer Full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 9-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP Status register (SSPSTAT) indicates the various status conditions.

EXAMPLE 9-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
LOOP  BTFSS  SSPSTAT, BF    ;Has data been received(transmit complete)?
      BRA    LOOP           ;No
      MOVF   SSPBUF, W      ;WREG reg = contents of SSPBUF
      MOVWF  RXDATA         ;Save in user RAM, if data is meaningful
      MOVF   TXDATA, W      ;W reg = contents of TXDATA
      MOVWF  SSPBUF         ;New data to xmit
```


© 2003 Microchip Technology Inc.

9.3.3 ENABLING SPI I/O

To enable the serial port, SSP Enable bit, SSPEN (SSPCON<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPCON registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed. That is:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> bit cleared
- SCK (Master mode) must have TRISC<3> bit cleared
- SCK (Slave mode) must have TRISC<3> bit set
- $\overline{SS}$ must have TRISC<4> bit set

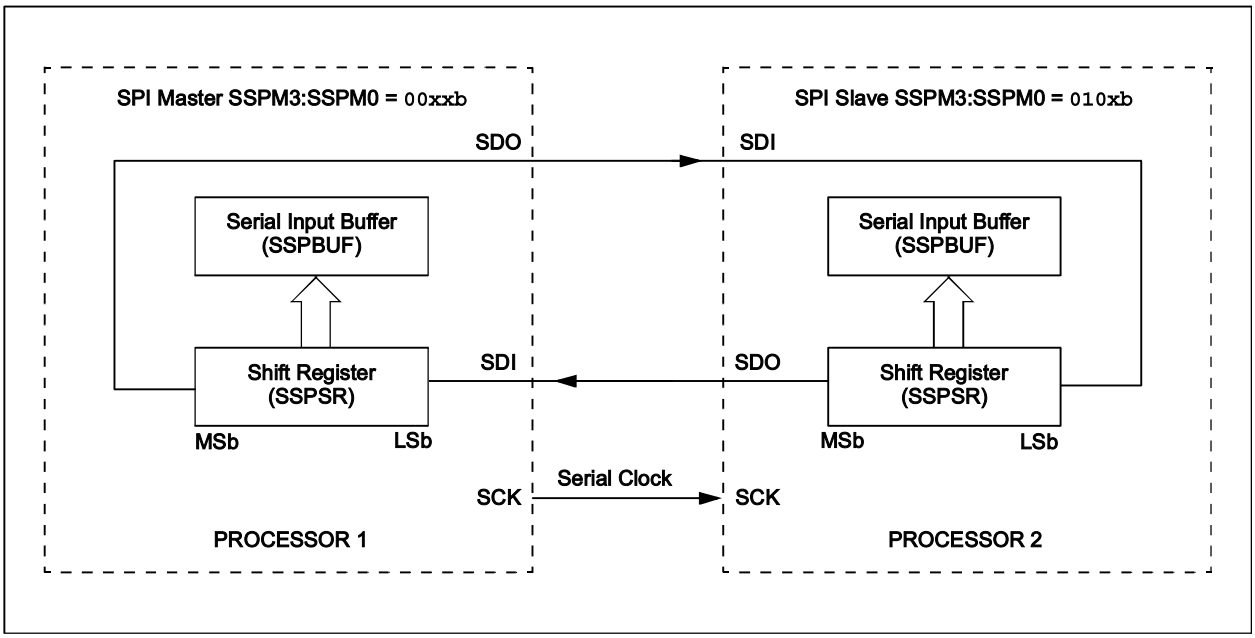
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

9.3.4 TYPICAL CONNECTION

Figure 9-2 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data
- Master sends dummy data – Slave sends data

FIGURE 9-2: SPI MASTER/SLAVE CONNECTION



9.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 9-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

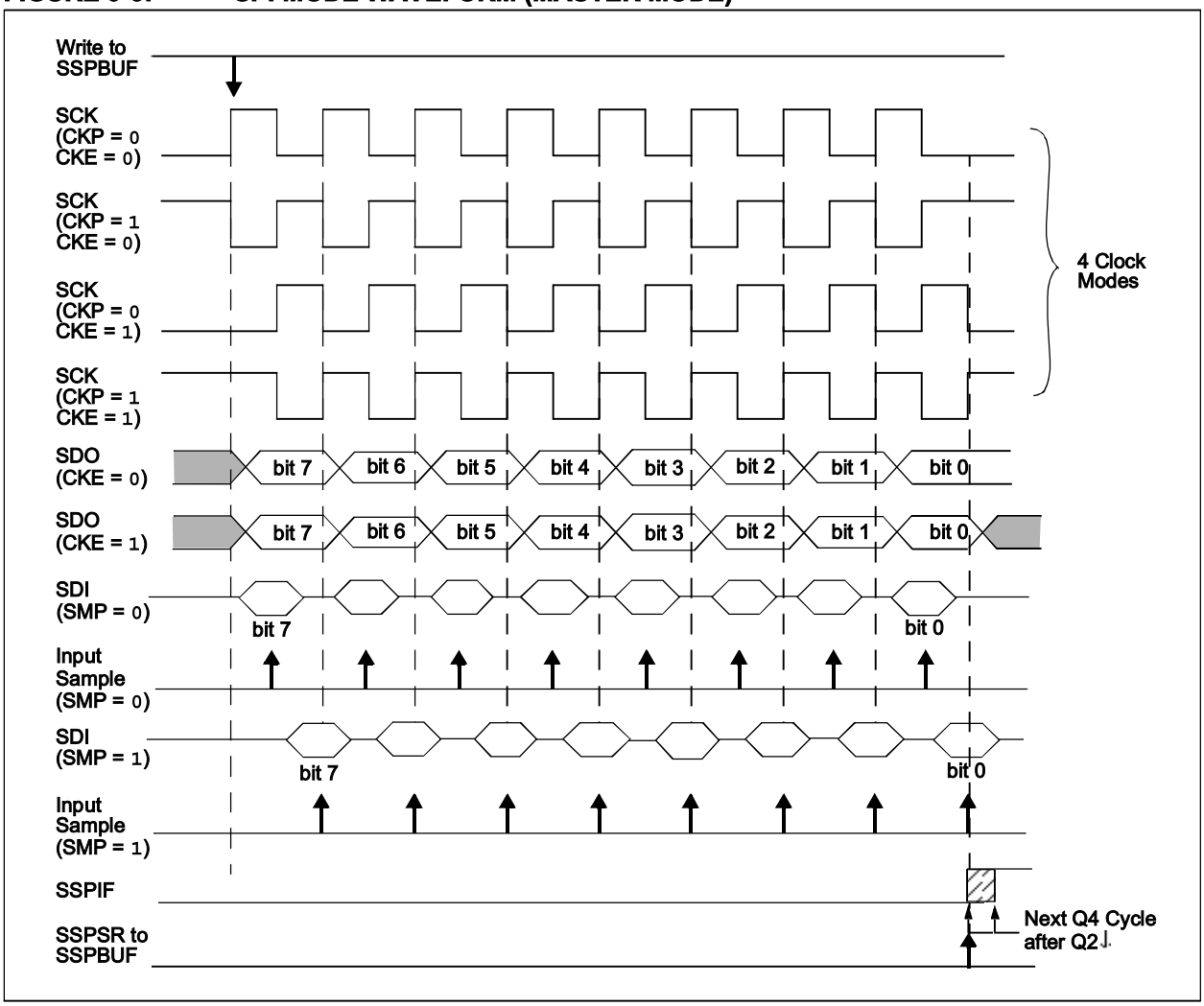
The clock polarity is selected by appropriately programming the CKP bit (SSPCON<4>). This then, would give waveforms for SPI communication as shown in Figure 9-3, Figure 9-5 and Figure 9-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 9-3 shows the waveforms for Master mode. When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

FIGURE 9-3: SPI MODE WAVEFORM (MASTER MODE)



© 2003 Microchip Technology Inc.

9.3.6 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device will wake-up from Sleep.

9.3.7 SLAVE SELECT SYNCHRONIZATION

The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON<3:0> = 04h). The pin must not be driven low for the $\overline{SS}$ pin to function as an input. The data latch must be high. When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven. When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable, depending on the application.

Note 1:	When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON<3:0> = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to VDD.
2:	If the SPI is used in Slave Mode with CKE set, then the $\overline{SS}$ pin control must be enabled.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the $\overline{SS}$ pin to a high level or clearing the SSPEN bit.

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.

FIGURE 9-4: SLAVE SYNCHRONIZATION WAVEFORM

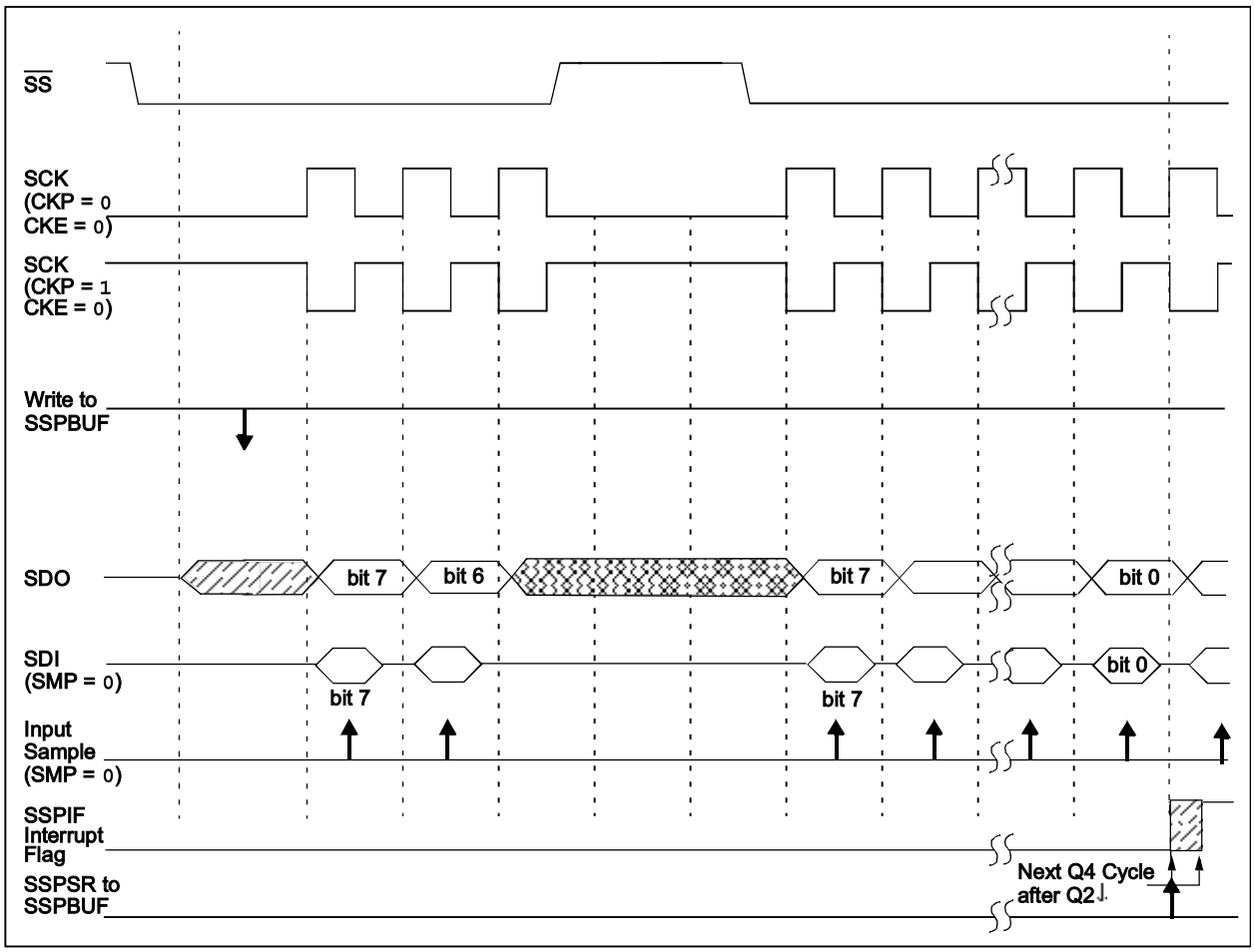


FIGURE 9-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

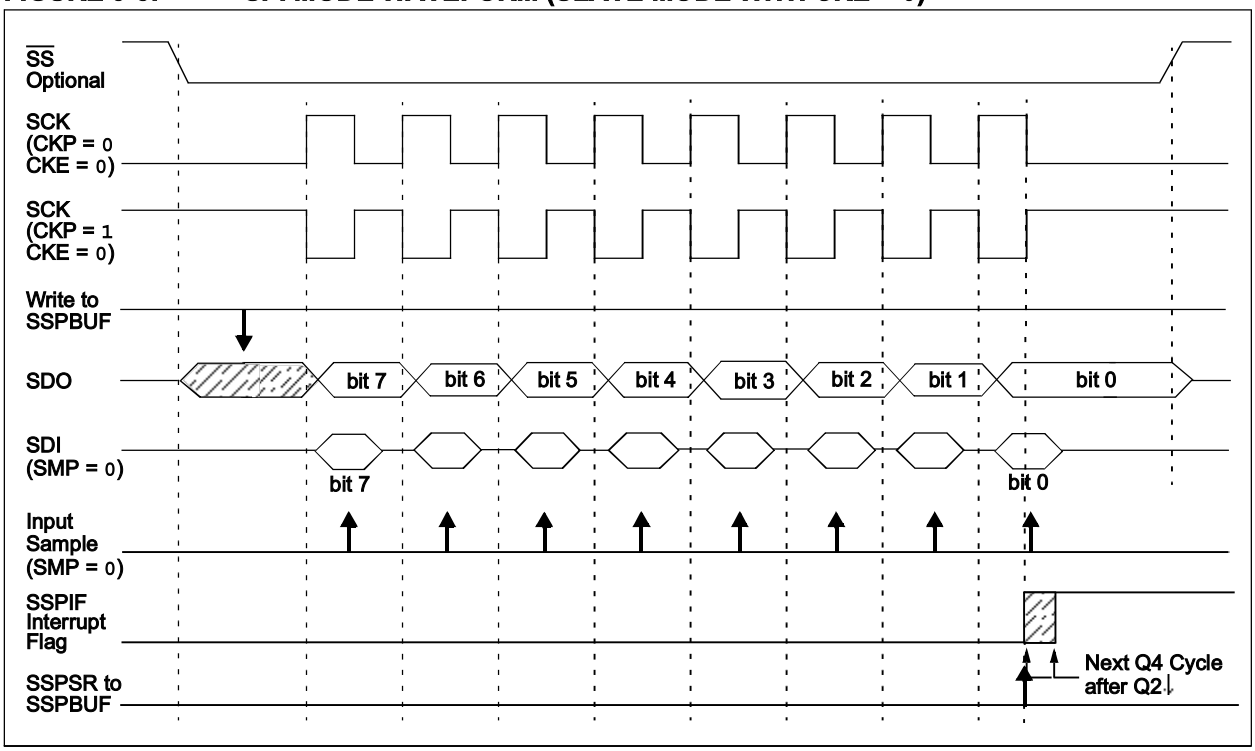
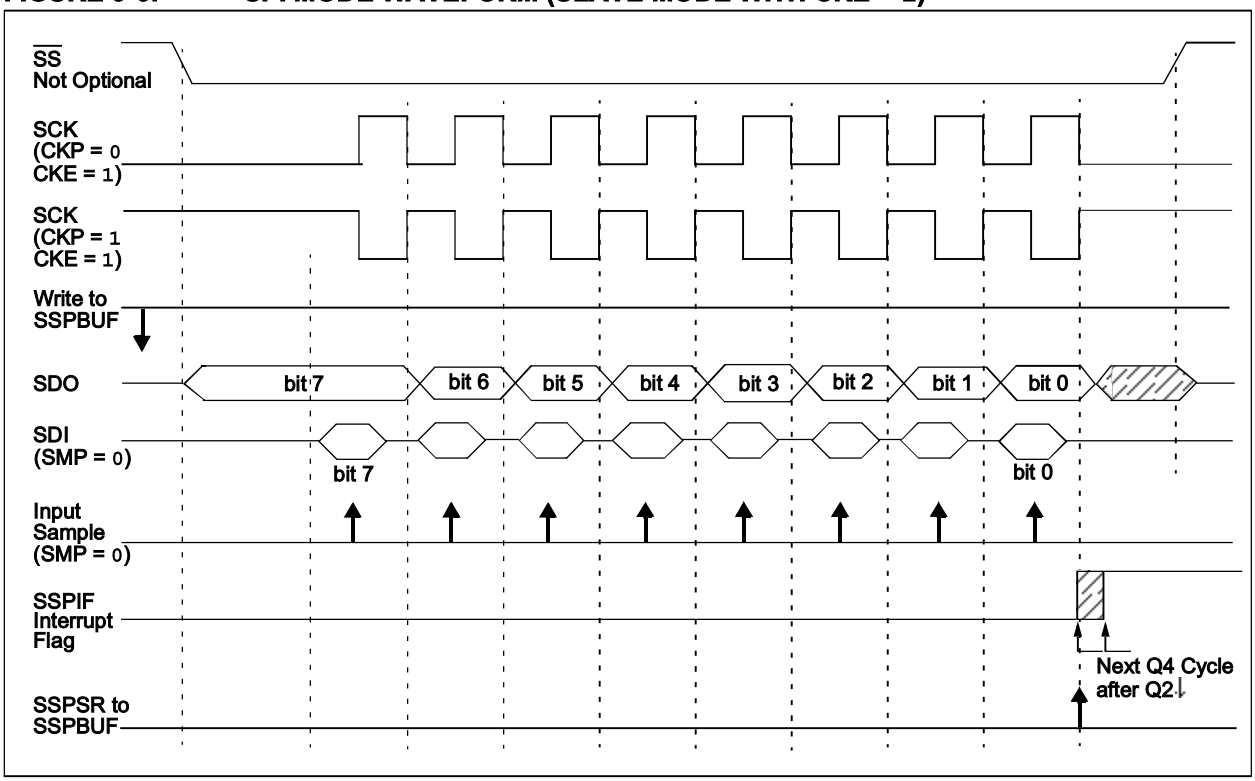


FIGURE 9-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



© 2003 Microchip Technology Inc.

9.3.8 SLEEP OPERATION

In Master mode, all module clocks are halted and the transmission/reception will remain in that state until the device wakes from Sleep. After the device returns to normal mode, the module will continue to transmit/ receive data.

In Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device from Sleep.

9.3.9 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

9.3.10 BUS MODE COMPATIBILITY

Table 9-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

TABLE 9-1: SPI BUS MODES

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

There is also a SMP bit which controls when the data is sampled.

TABLE 9-2: REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
TRISA	—	PORTA Data Direction Register							--11 1111	--11 1111
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'.
Shaded cells are not used by the MSSP in SPI mode.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on 28-pin devices; always maintain these bits clear.

9.4 I²C Mode

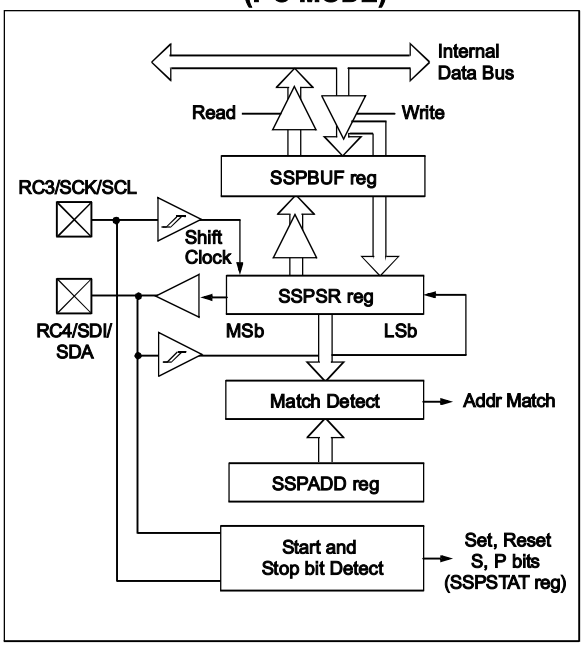
The MSSP module in I²C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit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

Two pins are used for data transfer:

- Serial clock (SCL) – RC3/SCK/SCL
- Serial data (SDA) – RC4/SDI/SDA

The user must configure these pins as inputs or outputs through the TRISC<4:3> bits.

FIGURE 9-7: MSSP BLOCK DIAGRAM (I²C MODE)



9.4.1 REGISTERS

The MSSP module has six registers for I²C operation. These are:

- MSSP Control Register (SSPCON)
- MSSP Control Register 2 (SSPCON2)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible
- MSSP Address Register (SSPADD)

SSPCON, SSPCON2 and SSPSTAT are the control and status registers in I²C mode operation. The SSPCON and SSPCON2 registers are readable and writable. The lower six bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

SSPADD register holds the slave device address when the SSP is configured in I²C Slave mode. When the SSP is configured in Master mode, the lower seven bits of SSPADD act as the baud rate generator reload value.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

© 2003 Microchip Technology Inc.

REGISTER 9-3: SSPSTAT: MSSP STATUS REGISTER (I^2C MODE) (ADDRESS 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Slew Rate Control bit

In Master or Slave mode:

1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)

0 = Slew rate control enabled for high-speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit

In Master or Slave mode:

1 = Enable SMBus specific inputs

0 = Disable SMBus specific inputs

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit

In Master mode:

Reserved.

In Slave mode:

1 = Indicates that the last byte received or transmitted was data

0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit

1 = Indicates that a Stop bit has been detected last

0 = Stop bit was not detected last

Note: This bit is cleared on Reset and when SSPEN is cleared.

bit 3 **S:** Start bit

1 = Indicates that a Start bit has been detected last

0 = Start bit was not detected last

Note: This bit is cleared on Reset and when SSPEN is cleared.

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ bit information (I^2C mode only)

In Slave mode:

1 = Read

0 = Write

Note: This bit holds the R/$\overline{W}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{ACK}$ bit.

In Master mode:

1 = Transmit is in progress

0 = Transmit is not in progress

Note: ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Idle mode.

bit 1 **UA:** Update Address (10-bit Slave mode only)

1 = Indicates that the user needs to update the address in the SSPADD register

0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit

In Transmit mode:

1 = Receive complete, SSPBUF is full

0 = Receive not complete, SSPBUF is empty

In Receive mode:

1 = Data Transmit in progress (does not include the $\overline{ACK}$ and Stop bits), SSPBUF is full

0 = Data Transmit complete (does not include the $\overline{ACK}$ and Stop bits), SSPBUF is empty

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 9-4: SSPCON1: MSSP CONTROL REGISTER 1 (I^2C MODE) (ADDRESS 14h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:

1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started. (Must be cleared in software.)

0 = No collision

In Slave Transmit mode:

1 = The SSPBUF register is written while it is still transmitting the previous word. (Must be cleared in software.)

0 = No collision

In Receive mode (Master or Slave modes):

This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte. (Must be cleared in software.)

0 = No overflow

In Transmit mode:

This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins

0 = Disables the serial port and configures these pins as I/O port pins

Note: When enabled, the SDA and SCL pins must be properly configured as input or output.

bit 4 **CKP:** SCK Release Control bit

In Slave mode:

1 = Release clock

0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In Master mode:

Unused in this mode.

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

1111 = I^2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

1110 = I^2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled

1011 = I^2C Firmware Controlled Master mode (Slave Idle)

1000 = I^2C Master mode, clock = FOSC/(4 * (SSPADD + 1))

0111 = I^2C Slave mode, 10-bit address

0110 = I^2C Slave mode, 7-bit address

Note: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

REGISTER 9-5: SSPCON2: MSSP CONTROL REGISTER 2 (I^2C MODE) (ADDRESS 91h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

bit 7 **GCEN:** General Call Enable bit (Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)
1 = Not Acknowledge
0 = Acknowledge

Note: Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence Idle

bit 3 **RCEN:** Receive Enable bit (Master mode only)
1 = Enables Receive mode for I^2C
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit (Master mode only)
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enabled bit (Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enabled/Stretch Enabled bit
In Master mode:
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is enabled for slave transmit only (PIC16F87X compatibility)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I^2C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

9.4.2 OPERATION

The MSSP module functions are enabled by setting MSSP Enable bit, SSPEN (SSPCON<5>).

The SSPCON register allows control of the I^2C operation. Four mode selection bits (SSPCON<3:0>) allow one of the following I^2C modes to be selected:

- I^2C Master mode, clock = OSC/4 (SSPADD + 1)
- I^2C Slave mode (7-bit address)
- I^2C Slave mode (10-bit address)
- I^2C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I^2C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I^2C Firmware Controlled Master mode, slave is Idle

Selection of any I^2C mode, with the SSPEN bit set, forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRISC bits. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

9.4.3 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

The I^2C Slave mode hardware will always generate an interrupt on an address match. Through the mode select bits, the user can also choose to interrupt on Start and Stop bits

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge ($\overline{ACK}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this $\overline{ACK}$ pulse:

- The buffer full bit, BF (SSPSTAT<0>), was set before the transfer was received.
- The overflow bit, SSPOV (SSPCON<6>), was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF (PIR1<3>) is set. The BF bit is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I^2C specification, as well as the requirement of the MSSP module, are shown in timing parameter #100 and parameter #101.

9.4.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8 bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match, and the BF and SSPOV bits are clear, the following events occur:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The Buffer Full bit, BF, is set.
3. An $\overline{ACK}$ pulse is generated.
4. MSSP Interrupt Flag bit, SSPIF (PIR1<3>), is set (interrupt is generated if enabled) on the falling edge of the ninth SCL pulse.

In 10-bit Address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/$\overline{W}$ (SSPSTAT<2>) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '`11110 A9 A8 0`', where '`A9`' and '`A8`' are the two MSbs of the address. The sequence of events for 10-bit address is as follows, with steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of address (bits SSPIF, BF and bit UA (SSPSTAT<1>) are set).
2. Update the SSPADD register with second (low) byte of address (clears bit UA and releases the SCL line).
3. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
4. Receive second (low) byte of address (bits SSPIF, BF and UA are set).
5. Update the SSPADD register with the first (high) byte of address. If match releases SCL line, this will clear bit UA.
6. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of address (bits SSPIF and BF are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.

© 2003 Microchip Technology Inc.

9.4.3.2 Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low ($\overline{ACK}$).

When the address byte overflow condition exists, then the No Acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set or bit SSPOV (SSPCON<6>) is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR1<3>) must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPCON<0> = 1), RC3/SCK/SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting bit CKP (SSPCON<4>). See **Section 9.4.4 "Clock Stretching"** for more detail.

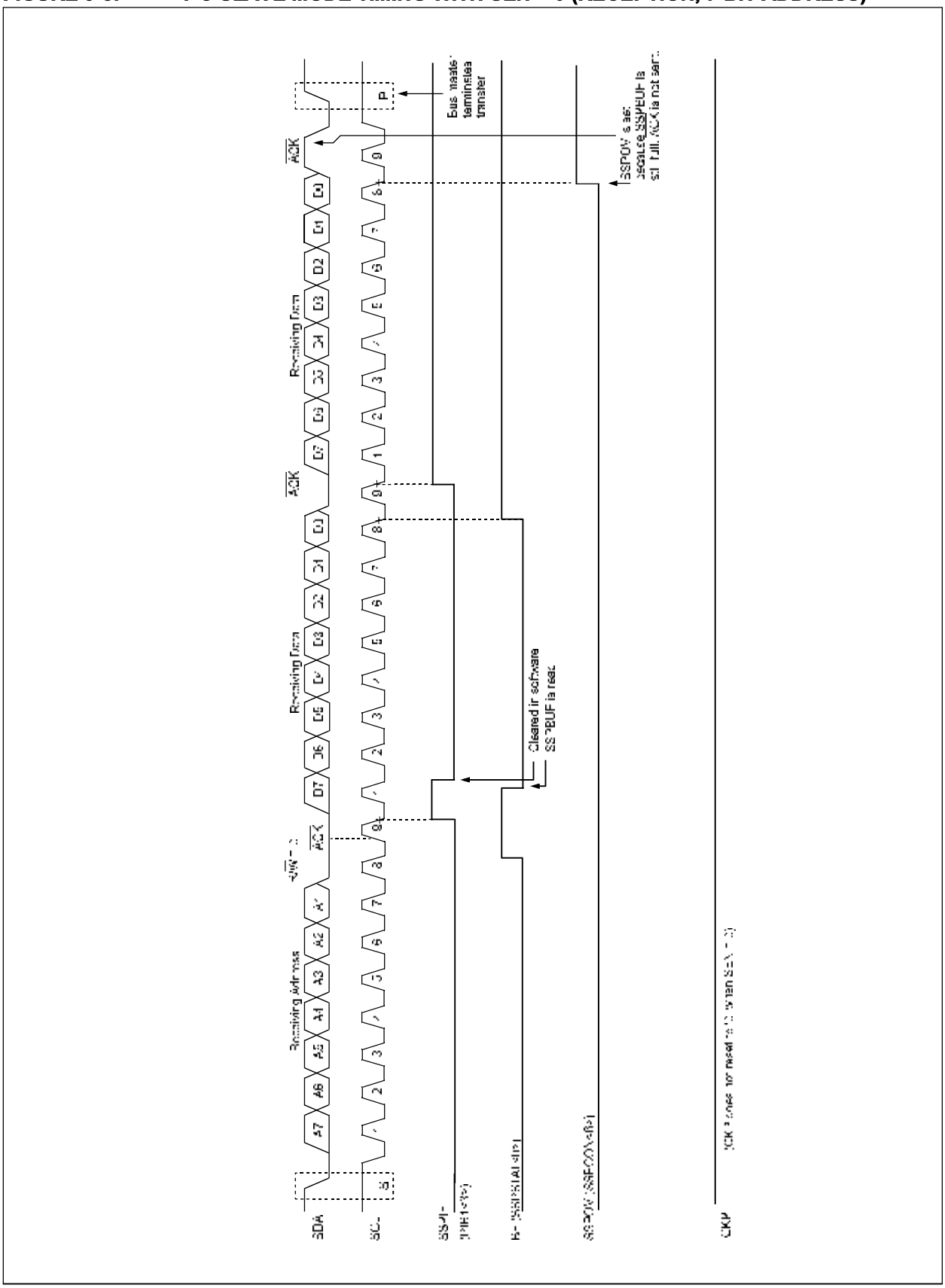
9.4.3.3 Transmission

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit and pin RC3/SCK/SCL is held low regardless of SEN (see **Section 9.4.4 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then pin RC3/SCK/SCL should be enabled by setting bit CKP (SSPCON<4>). The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (Figure 9-9).

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{ACK}$), then the data transfer is complete. In this case, when the $\overline{ACK}$ is latched by the slave, the slave logic is reset (resets SSPSTAT register) and the slave monitors for another occurrence of the Start bit. If the SDA line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPBUF register. Again, pin RC3/SCK/SCL must be enabled by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

FIGURE 9-8: I^2C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)



© 2003 Microchip Technology Inc.

FIGURE 9-9: I^2C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)

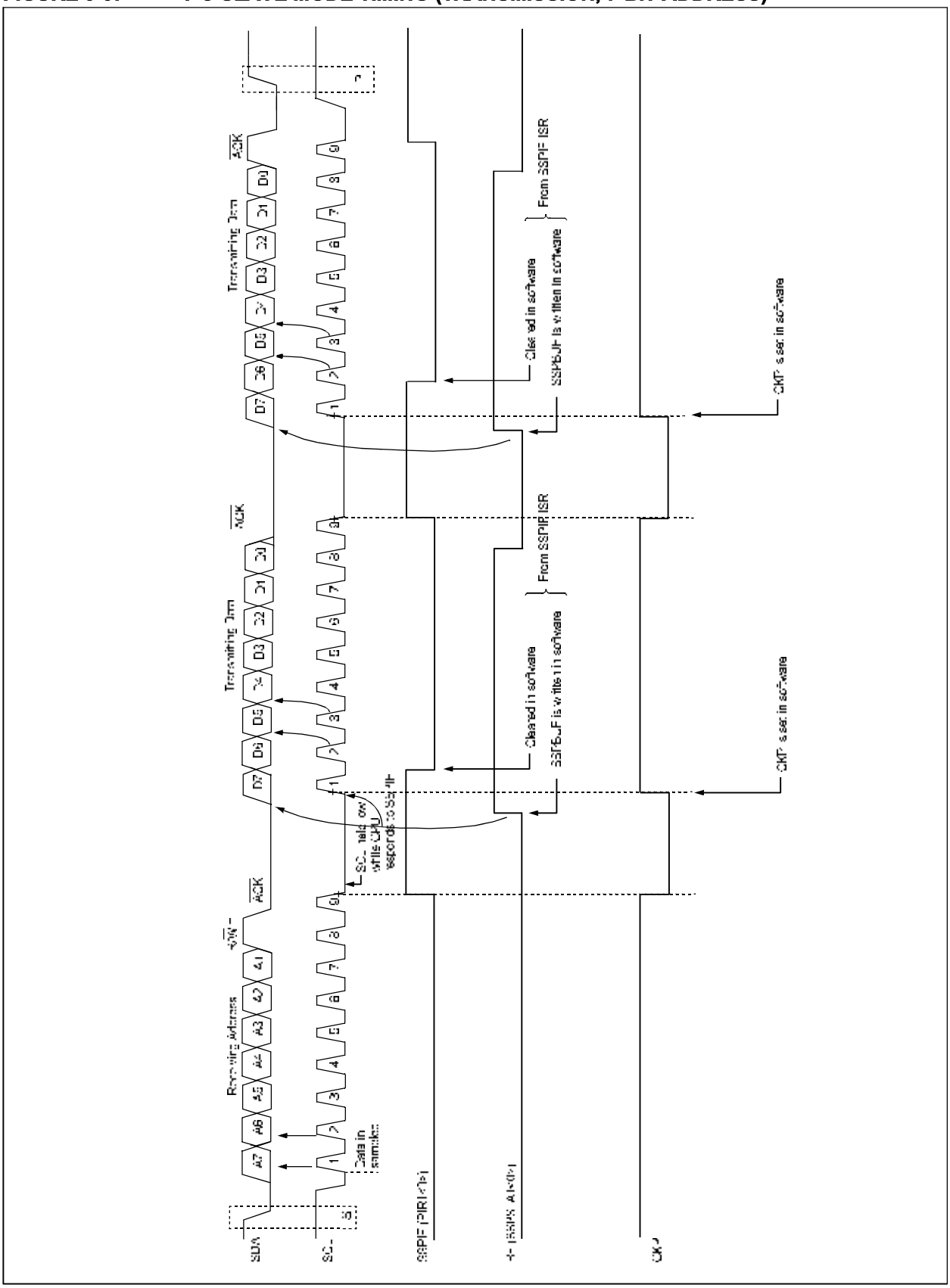
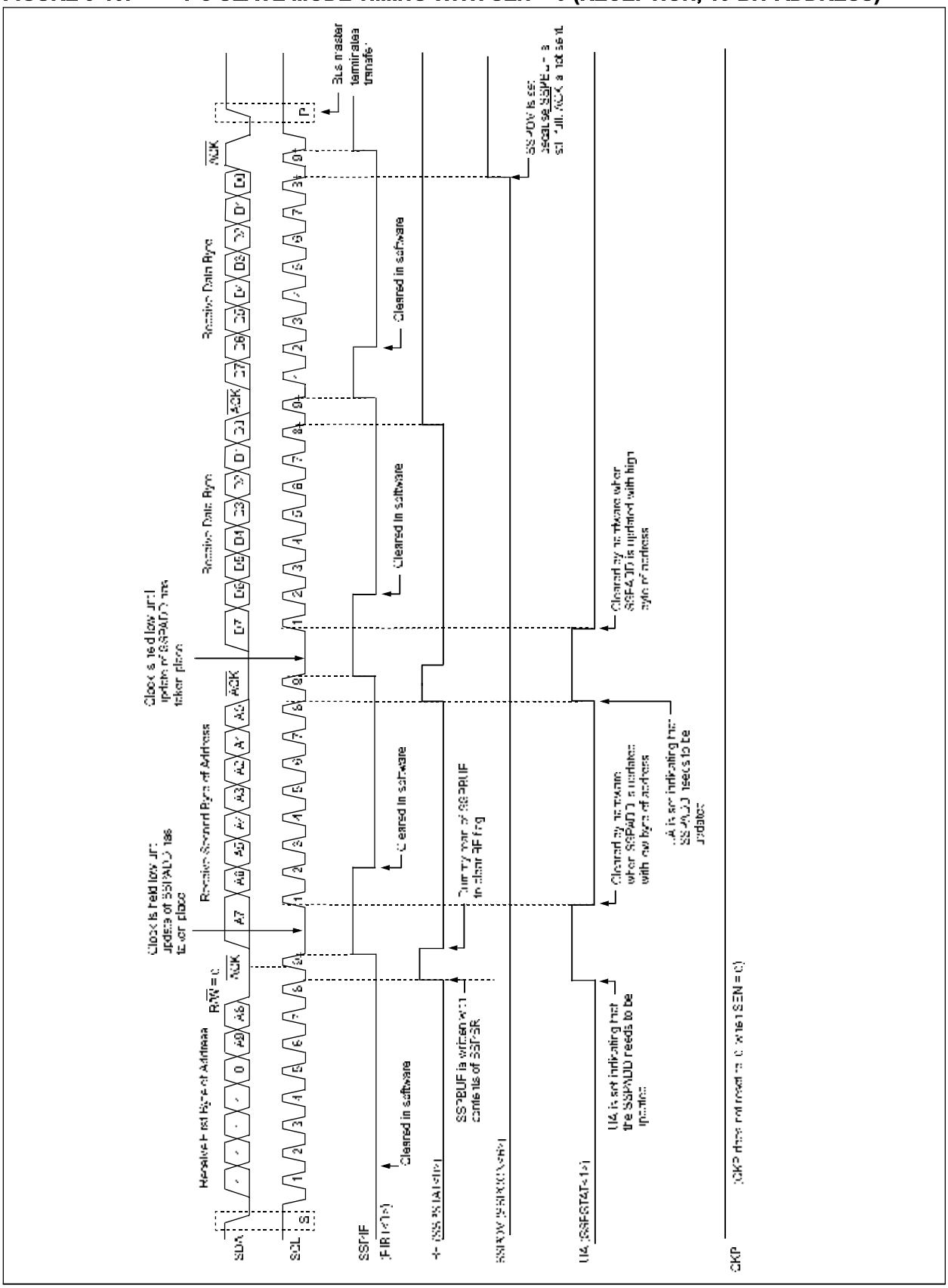
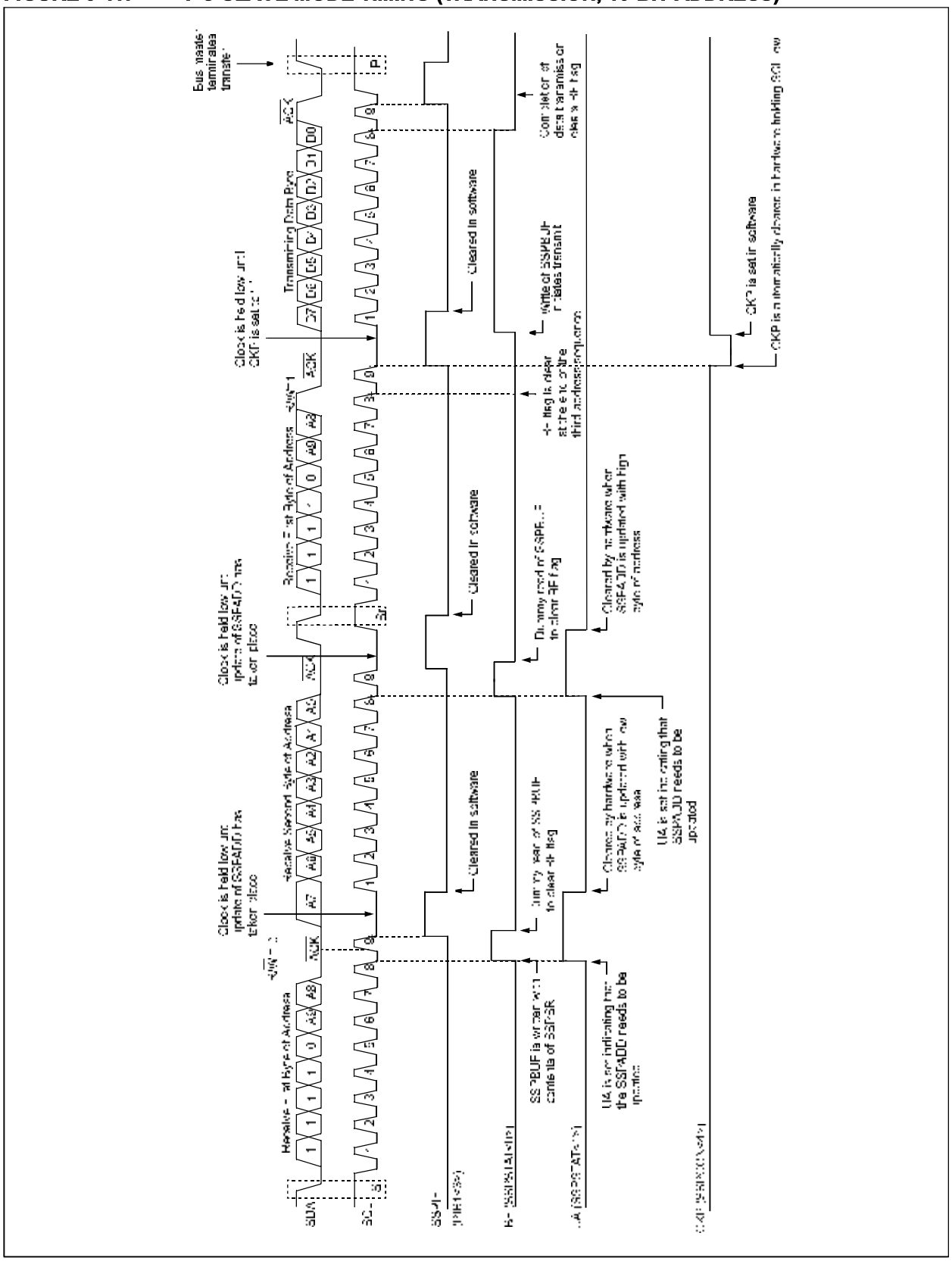


FIGURE 9-10: I^2C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)



© 2003 Microchip Technology Inc.

FIGURE 9-11: I^2C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)



9.4.4 CLOCK STRETCHING

Both 7 and 10-bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

9.4.4.1 Clock Stretching for 7-bit Slave Receive Mode (SEN = 1)

In 7-bit Slave Receive mode, on the falling edge of the ninth clock at the end of the ACK sequence, if the BF bit is set, the CKP bit in the SSPCON register is automatically cleared, forcing the SCL output to be held low. The CKP bit being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 9-13).

Note 1: If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

9.4.4.2 Clock Stretching for 10-bit Slave Receive Mode (SEN = 1)

In 10-bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address, with the R/W bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

Note: If the user polls the UA bit and clears it by updating the SSPADD register before the falling edge of the ninth clock occurs and if the user hasn't cleared the BF bit by reading the SSPBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching, on the basis of the state of the BF bit, only occurs during a data sequence, not an address sequence.

9.4.4.3 Clock Stretching for 7-bit Slave Transmit Mode

7-bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock, if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another transmit sequence (see Figure 9-9).

Note 1: If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit.

9.4.4.4 Clock Stretching for 10-bit Slave Transmit Mode

In 10-bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-bit Slave Receive mode. The first two addresses are followed by a third address sequence, which contains the high order bits of the 10-bit address and the R/W bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-bit Slave Transmit mode (see Figure 9-11).

© 2003 Microchip Technology Inc.

9.4.4.5 Clock Synchronization and the CKP Bit

When the CKP bit is cleared, the SCL output is forced to '0'; however, setting the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I^2C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I^2C bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 9-12).

FIGURE 9-12: CLOCK SYNCHRONIZATION TIMING

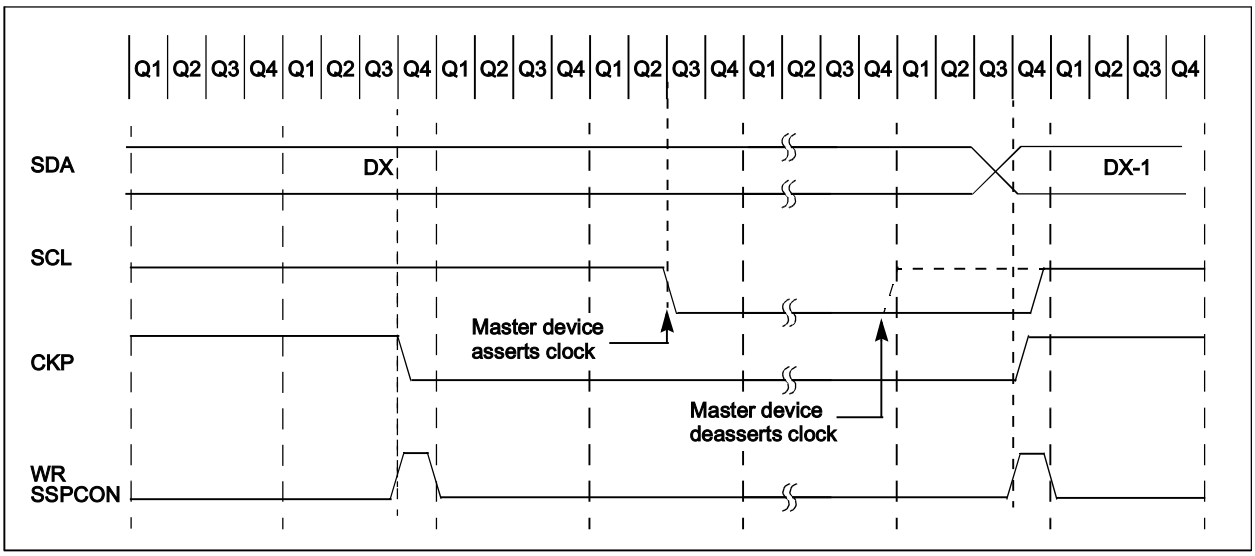
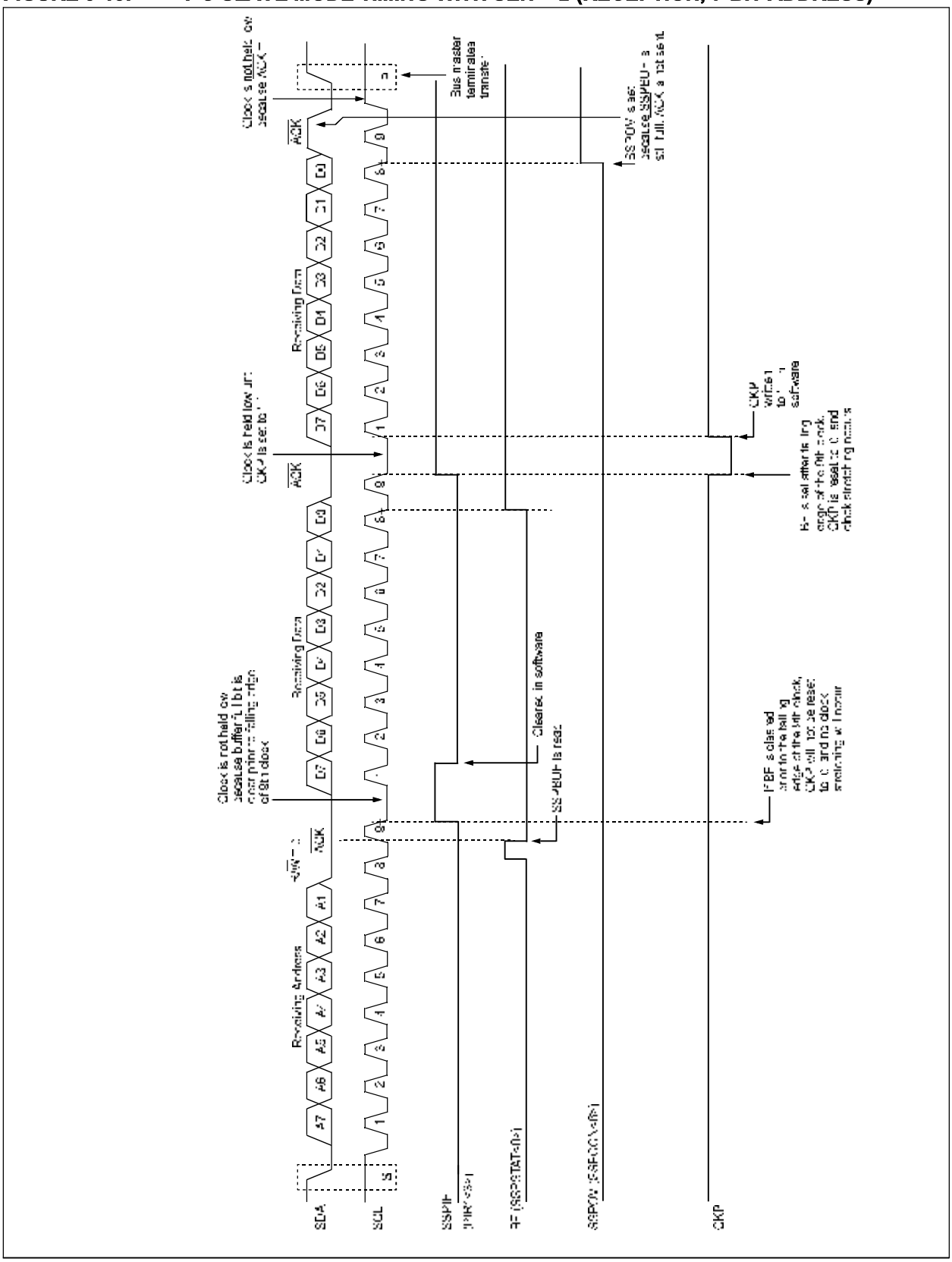
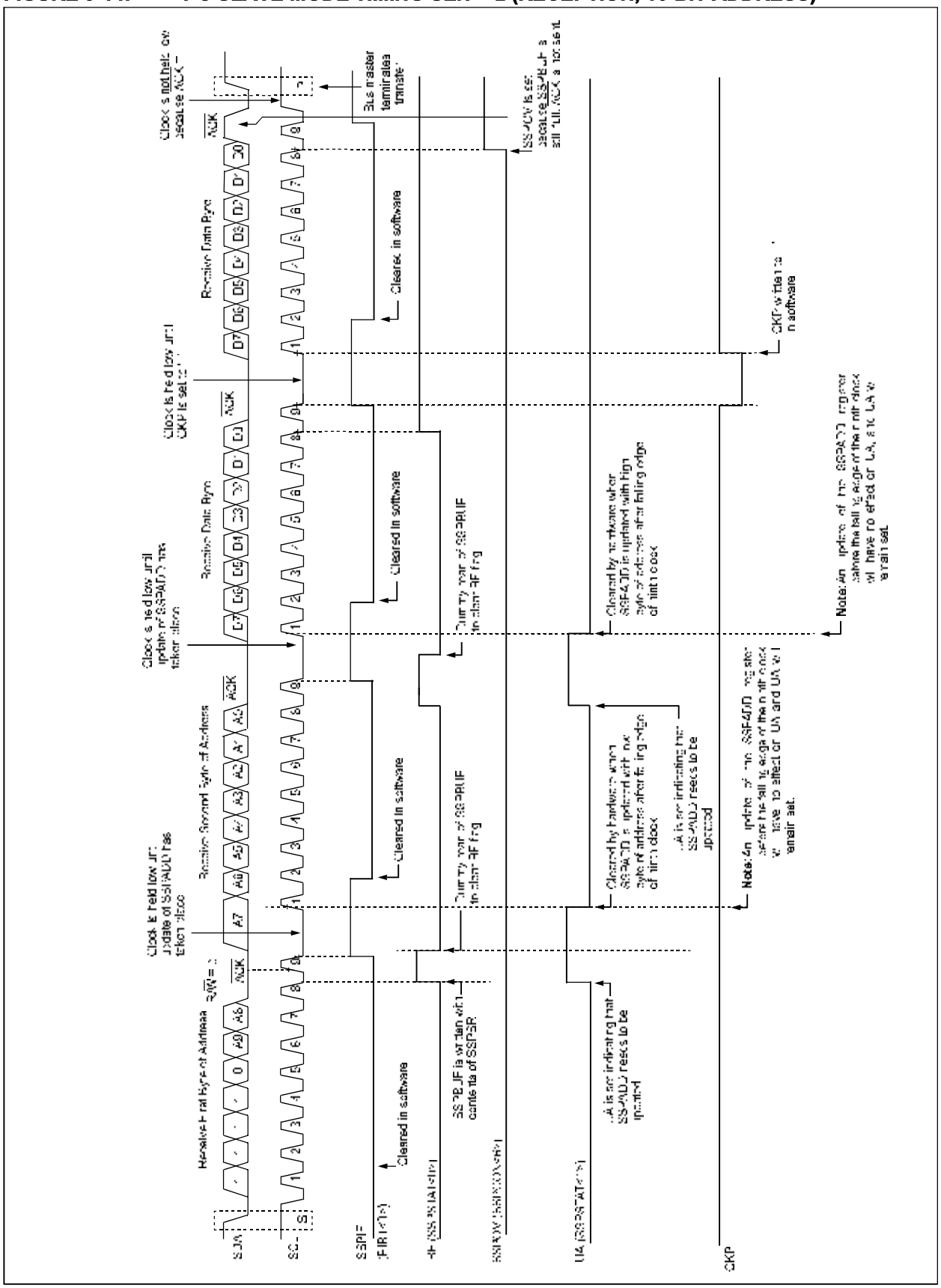


FIGURE 9-13: I^2C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)



© 2003 Microchip Technology Inc.

FIGURE 9-14: I²C SLAVE MODE TIMING SEN = 1 (RECEPTION, 10-BIT ADDRESS)



9.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I^2C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an Acknowledge.

The general call address is one of eight addresses reserved for specific purposes by the I^2C protocol. It consists of all '0's with R/$\overline{W}$ = 0.

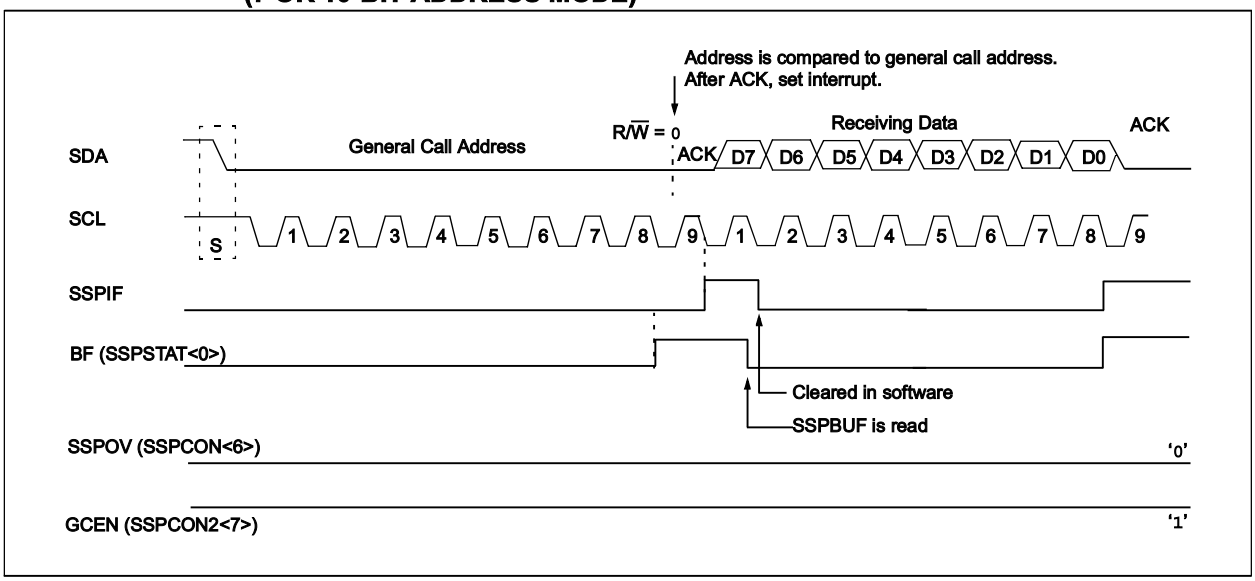
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> set). Following a Start bit detect, 8 bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag bit is set (eighth bit) and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the SSPIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF. The value can be used to determine if the address was device specific or a general call address.

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match and the UA bit is set (SSPSTAT<1>). If the general call address is sampled when the GCEN bit is set, while the slave is configured in 10-bit Address mode, then the second half of the address is not necessary, the UA bit will not be set and the slave will begin receiving data after the Acknowledge (Figure 9-15).

FIGURE 9-15: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESS MODE)



© 2003 Microchip Technology Inc.

9.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit is set or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I^2C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

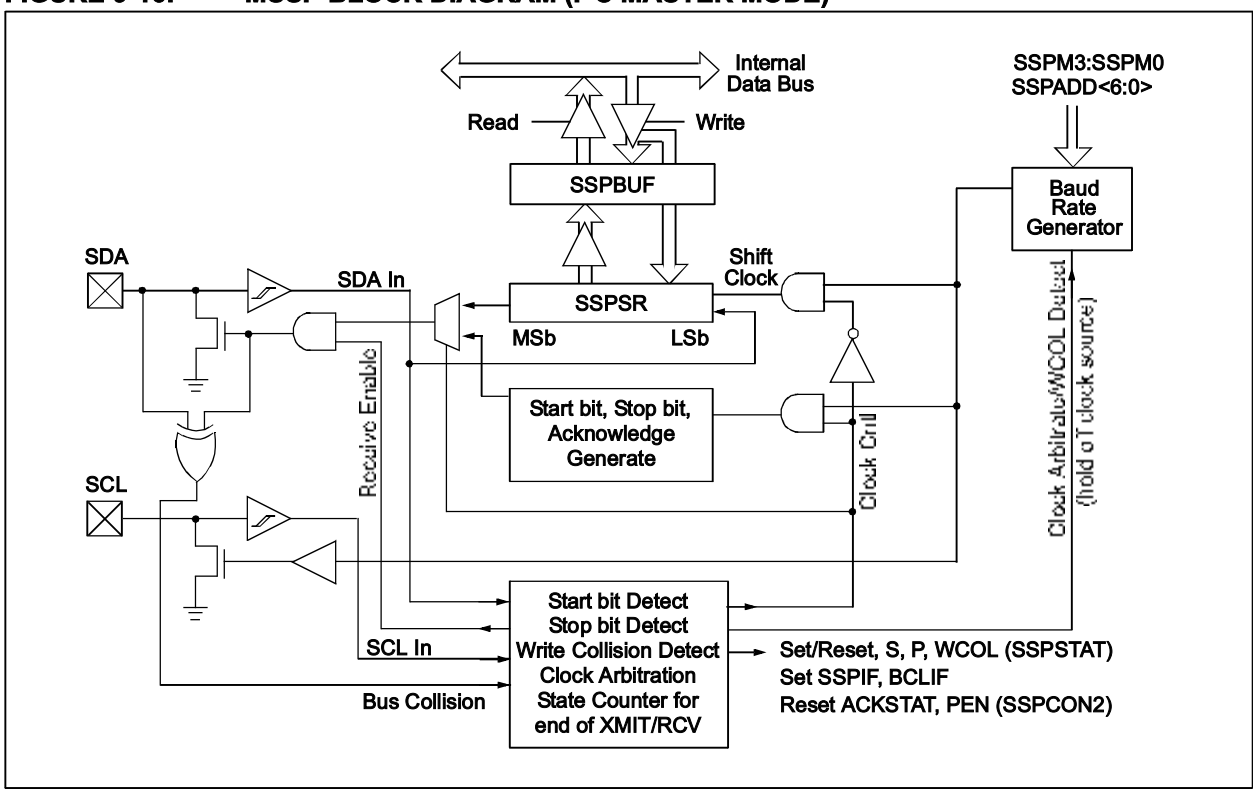
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register, initiating transmission of data/address.
4. Configure the I^2C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

Note: The MSSP module, when configured in I^2C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

The following events will cause SSP Interrupt Flag bit, SSPIF, to be set (SSP interrupt if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmit
- Repeated Start

FIGURE 9-16: MSSP BLOCK DIAGRAM (I^2C MASTER MODE)



9.4.6.1 I²C Master Mode Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

The baud rate generator used for the SPI mode operation is used to set the SCL clock frequency for either 100 kHz, 400 kHz or 1 MHz I²C operation. See **Section 9.4.7 "Baud Rate Generator"** for more detail.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the Start Enable bit, SEN (SSPCON2<0>).
2. SSPIF is set. The MSSP module will wait the required Start time before any other operation takes place.
3. The user loads the SSPBUF with the slave address to transmit.
4. Address is shifted out the SDA pin until all 8 bits are transmitted.
5. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
11. The user generates a Stop condition by setting the Stop Enable bit, PEN (SSPCON2<2>).
12. Interrupt is generated once the Stop condition is complete.

© 2003 Microchip Technology Inc.

9.4.7 BAUD RATE GENERATOR

In I^2C Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPADD register (Figure 9-17). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY) on the Q2 and Q4 clocks. In I^2C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

Table 9-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

FIGURE 9-17: BAUD RATE GENERATOR BLOCK DIAGRAM

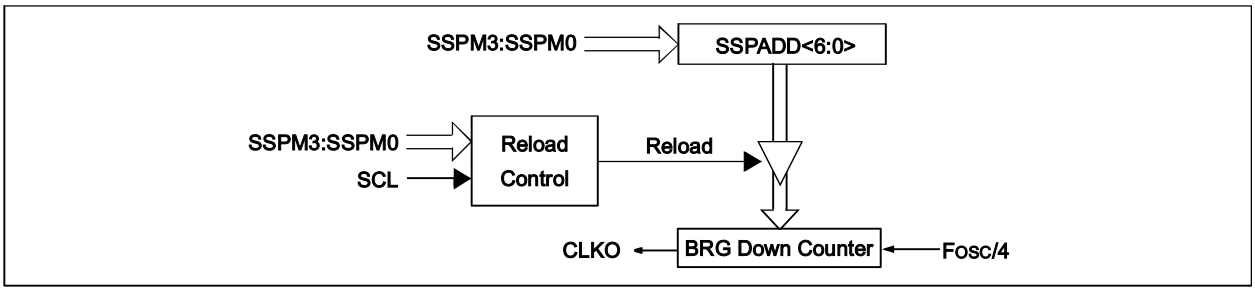


TABLE 9-3: I^2C CLOCK RATE W/BRG

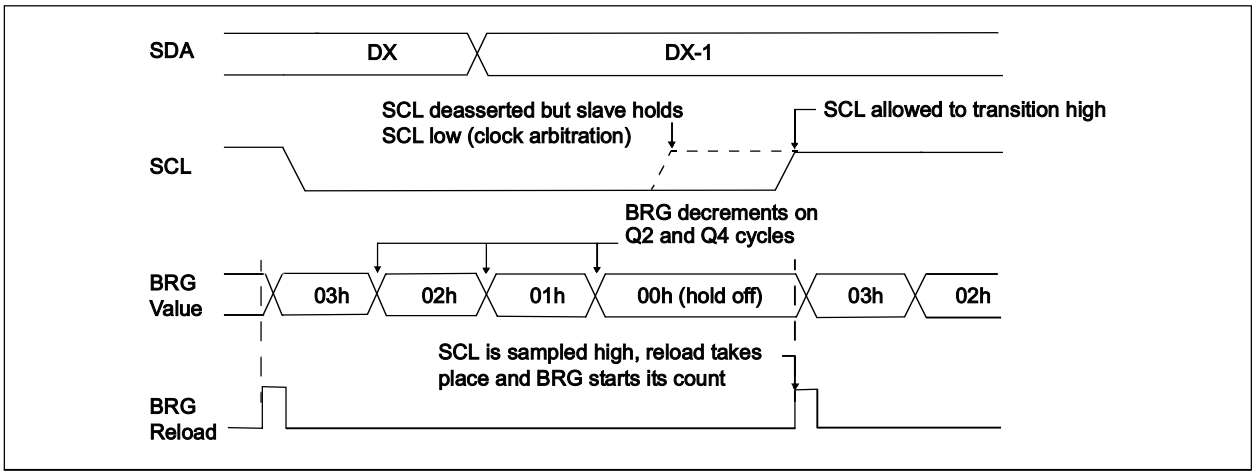
FCY	FCY*2	BRG Value	FSCL (2 Rollovers of BRG)
10 MHz	20 MHz	19h	400 kHz(1)
10 MHz	20 MHz	20h	312.5 kHz
10 MHz	20 MHz	3Fh	100 kHz
4 MHz	8 MHz	0Ah	400 kHz(1)
4 MHz	8 MHz	0Dh	308 kHz
4 MHz	8 MHz	28h	100 kHz
1 MHz	2 MHz	03h	333 kHz(1)
1 MHz	2 MHz	0Ah	100 kHz
1 MHz	2 MHz	00h	1 MHz(1)

Note 1: The I^2C interface does not conform to the 400 kHz I^2C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

9.4.7.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count, in the event that the clock is held low by an external device (Figure 9-17).

FIGURE 9-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



© 2003 Microchip Technology Inc.

9.4.8 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start condition enable bit, SEN (SSPCON2<0>). If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low, while SCL is high, is the Start condition and causes the S bit (SSPSTAT<3>) to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit (SSPCON2<0>) will be automatically cleared by hardware, the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

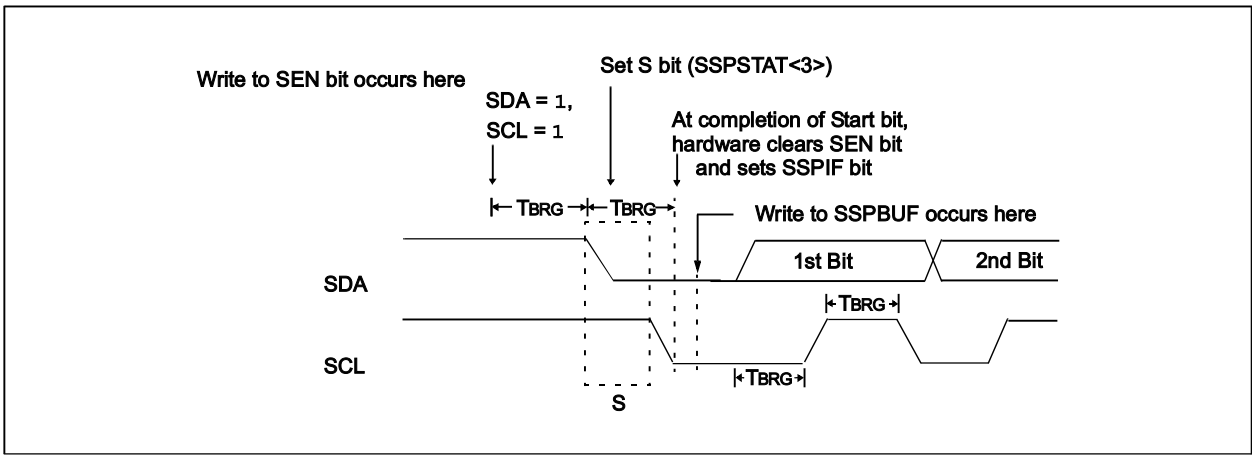
Note:	If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag (BCLIF) is set, the Start condition is aborted and the I²C module is reset into its Idle state.

9.4.8.1 WCOL Status Flag

If the user writes the SSPBUF when a Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

Note:	Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the Start condition is complete.

FIGURE 9-19: FIRST START BIT TIMING



9.4.9 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I²C logic module is in the Idle state. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded with the contents of SSPADD<5:0> and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA = 0) for one TBRG while SCL is high. Following this, the RSEN bit (SSPCON2<1>) will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit (SSPSTAT<3>) will be set. The SSPIF bit will not be set until the Baud Rate Generator has timed out.

Note 1: If RSEN is programmed while any other event is in progress, it will not take effect.

2: A bus collision during the Repeated Start condition occurs if:

- SDA is sampled low when SCL goes from low to high.
- SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data '1'.

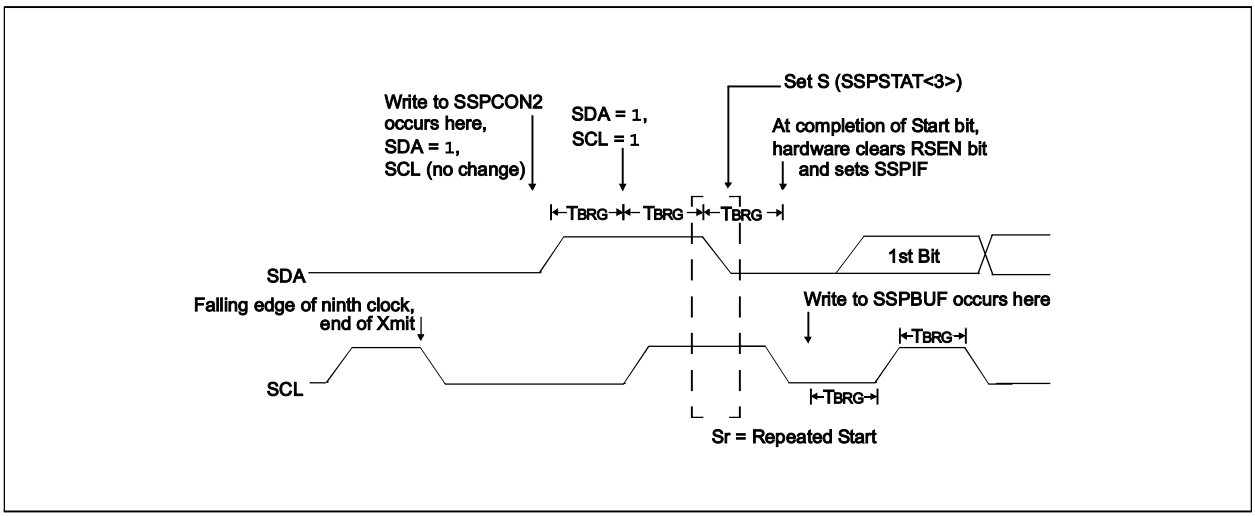
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

9.4.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Repeated Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

Note: Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

FIGURE 9-20: REPEAT START CONDITION WAVEFORM



© 2003 Microchip Technology Inc.

9.4.10 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full flag bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time specification, parameter #106). SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification, parameter #107). When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time, if an address match occurred or if data was received properly. The status of $\overline{ACK}$ is written into the ACKDT bit on the falling edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock, the SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 9-21).

After the write to the SSPBUF, each bit of address will be shifted out on the falling edge of SCL, until all seven address bits and the $\overline{R/W}$ bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the ACK bit is loaded into the ACKSTAT status bit (SSPCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

9.4.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPSTAT<0>) is set when the CPU writes to SSPBUF and is cleared when all eight bits are shifted out.

9.4.10.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

WCOL must be cleared in software.

9.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does Not Acknowledge ($\overline{ACK}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call) or when the slave has properly received its data.

9.4.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPCON2<3>).

Note:	The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high to low/low to high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state, awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>).

9.4.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

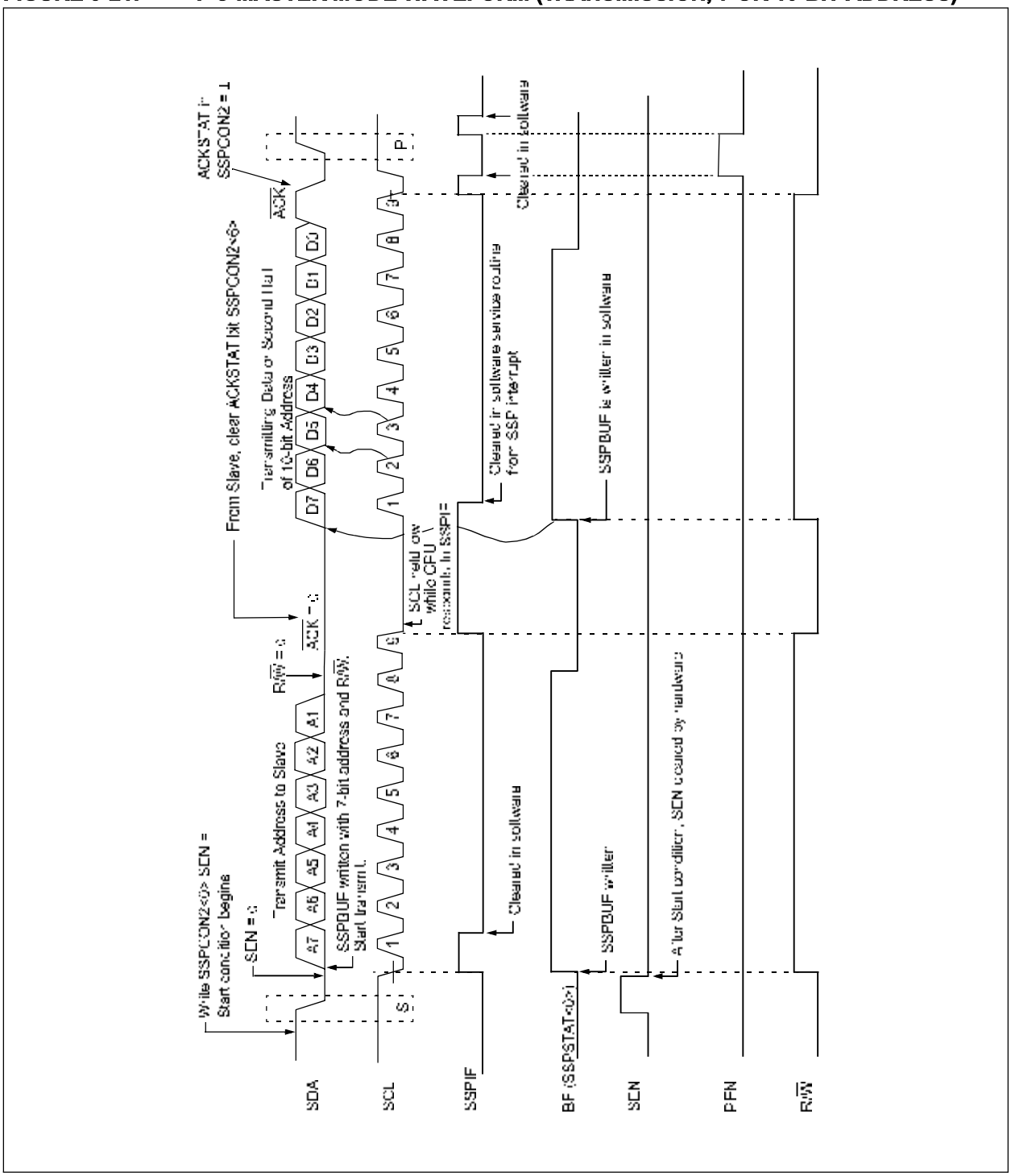
9.4.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

9.4.11.3 WCOL Status Flag

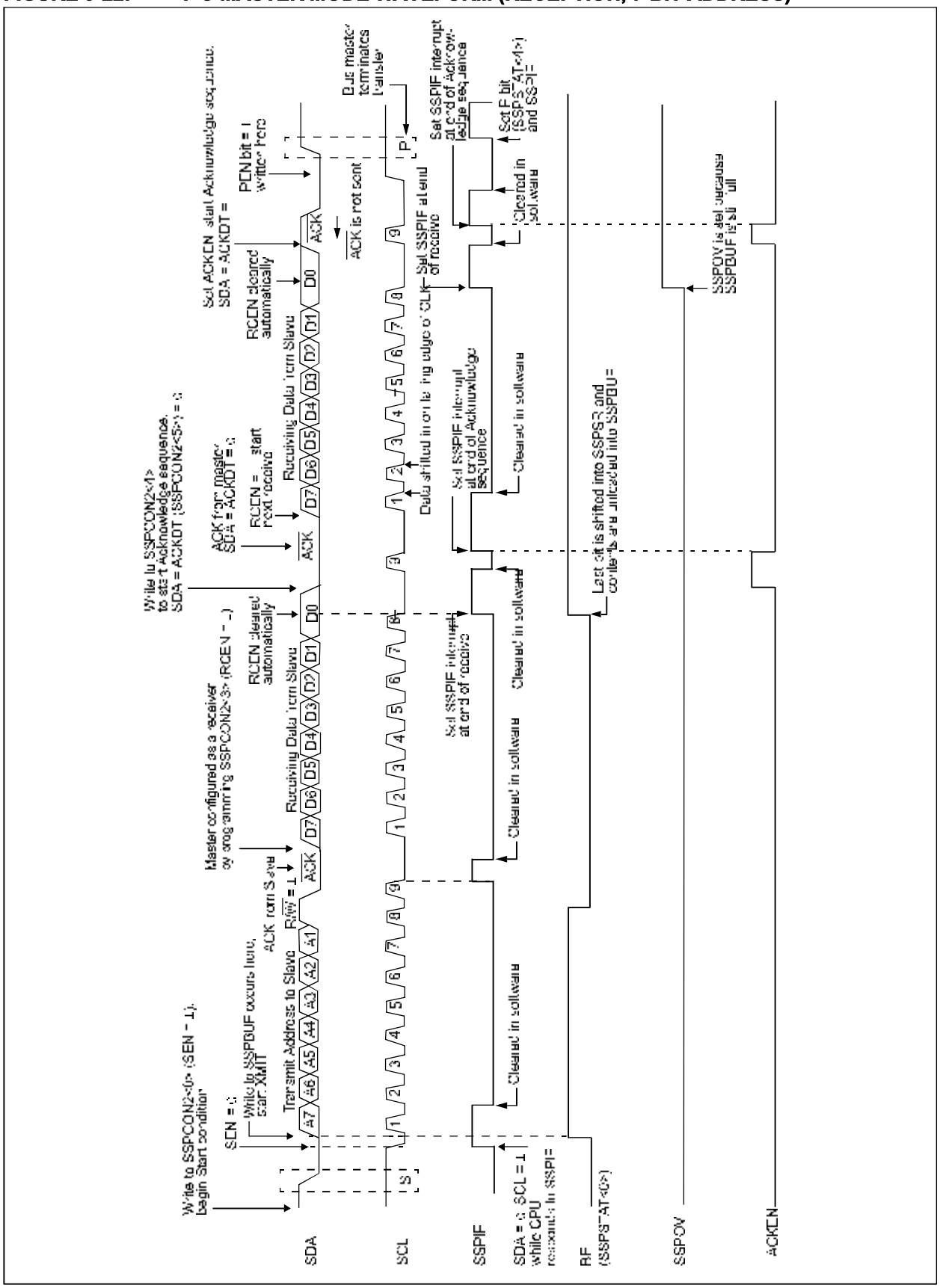
If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

FIGURE 9-21: I^2C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)



© 2003 Microchip Technology Inc.

FIGURE 9-22: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



9.4.12 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>). When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the baud rate generator is turned off and the MSSP module then goes into Idle mode (Figure 9-23).

9.4.12.1 WCOL Status Flag

If the user writes the SSPBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

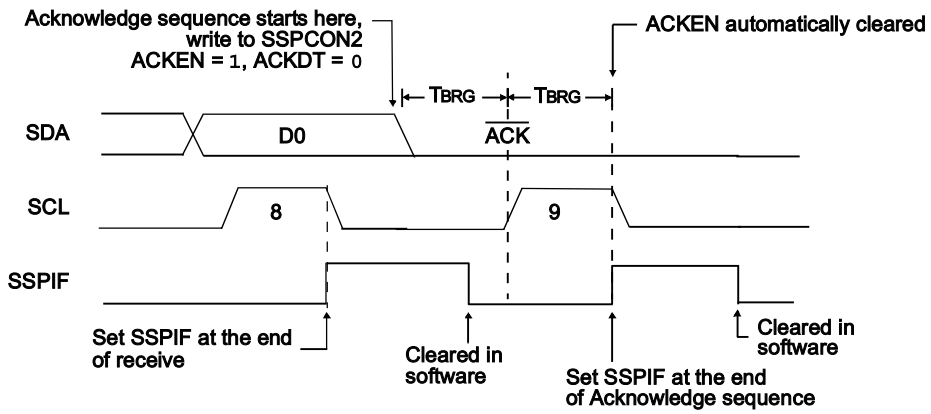
9.4.13 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN (SSPCON2<2>). At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to 0. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 9-24).

9.4.13.1 WCOL Status Flag

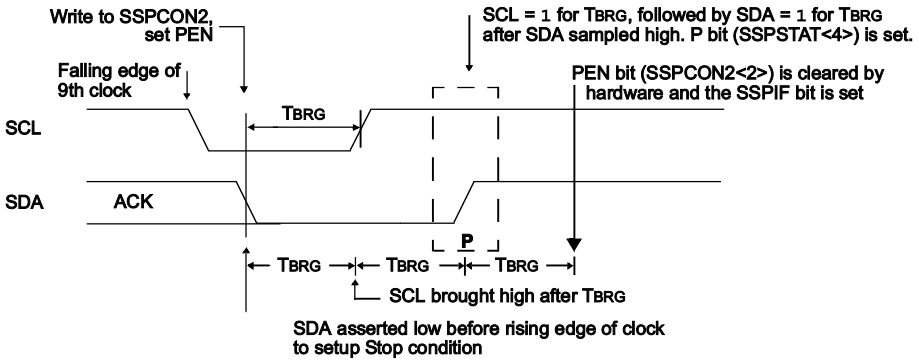
If the user writes the SSPBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

FIGURE 9-23: ACKNOWLEDGE SEQUENCE WAVEFORM



Note: TBRG = one Baud Rate Generator period.

FIGURE 9-24: STOP CONDITION RECEIVE OR TRANSMIT MODE



Note: TBRG = one Baud Rate Generator period.

© 2003 Microchip Technology Inc.

9.4.14 SLEEP OPERATION

While in Sleep mode, the I^2C module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

9.4.15 EFFECT OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

9.4.16 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit (SSPSTAT<4>) is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDA line must be monitored for arbitration to see if the signal level is at the expected output level. This check is performed in hardware with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

9.4.17 MULTI -MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = 0, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF, and reset the I^2C port to its Idle state (Figure 9-25).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I^2C bus is free, the user can resume communication by asserting a Start condition.

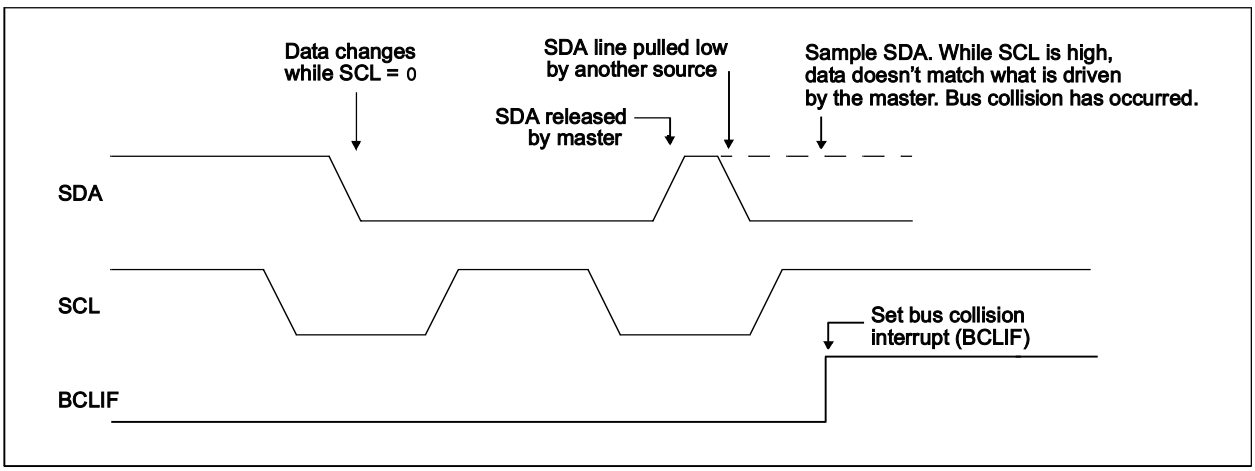
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I^2C bus is free, the user can resume communication by asserting a Start condition.

The Master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I^2C bus can be taken when the P bit is set in the SSPSTAT register or the bus is Idle and the S and P bits are cleared.

FIGURE 9-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



9.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 9-26).

b) SCL is sampled low before SDA is asserted low (Figure 9-27).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

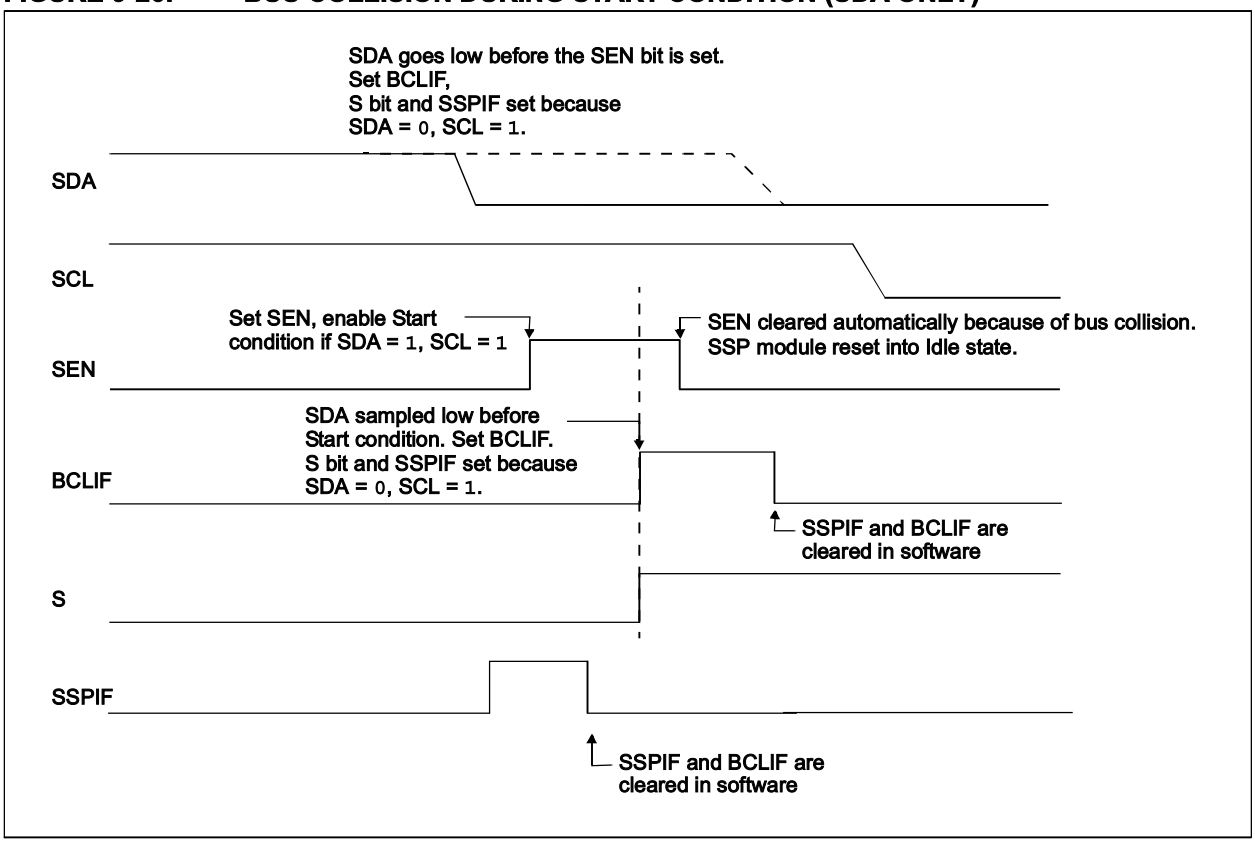
- the Start condition is aborted,
- the BCLIF flag is set and
- the MSSP module is reset to its Idle state (Figure 9-26).

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded from SSPADD<6:0> and counts down to 0. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 9-28). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to 0 and during this time, if the SCL pin is sampled as '0', a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

Note:	The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

FIGURE 9-26: BUS COLLISION DURING START CONDITION (SDA ONLY)



© 2003 Microchip Technology Inc.

FIGURE 9-27: BUS COLLISION DURING START CONDITION (SCL = 0)

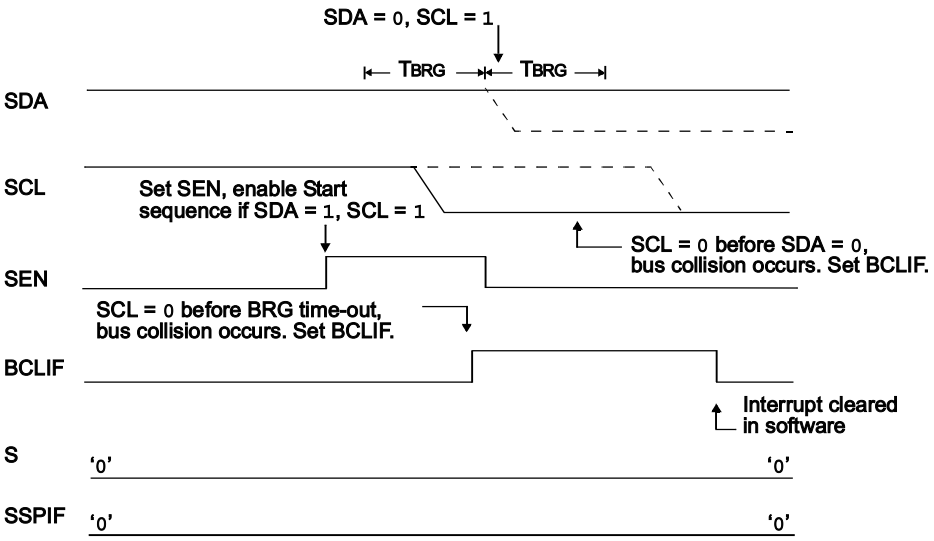
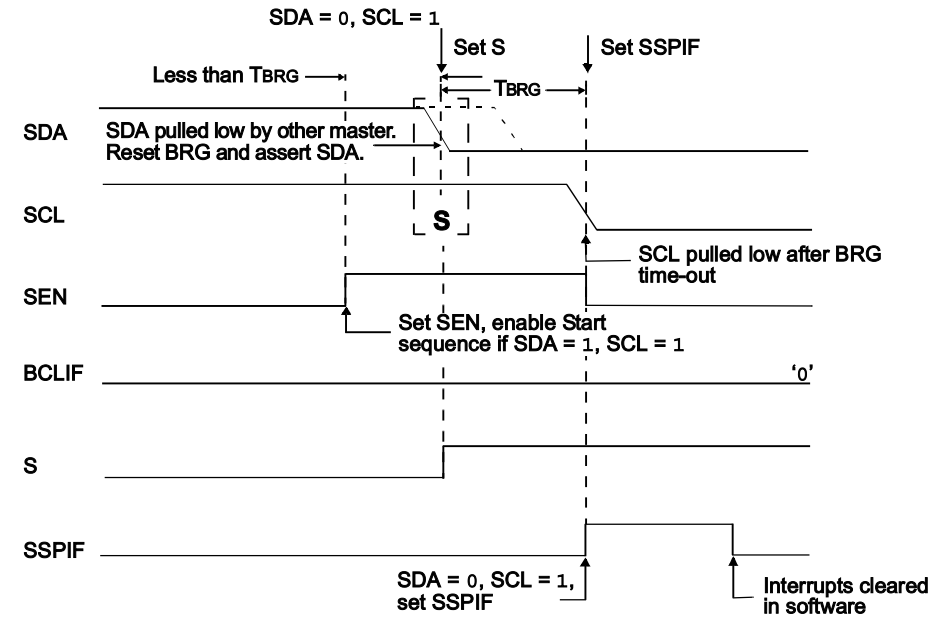


FIGURE 9-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



9.4.17.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from low level to high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to 0. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', see Figure 9-29). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high to low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high to low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition (Figure 9-30).

If at the end of the BRG time-out, both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete.

FIGURE 9-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

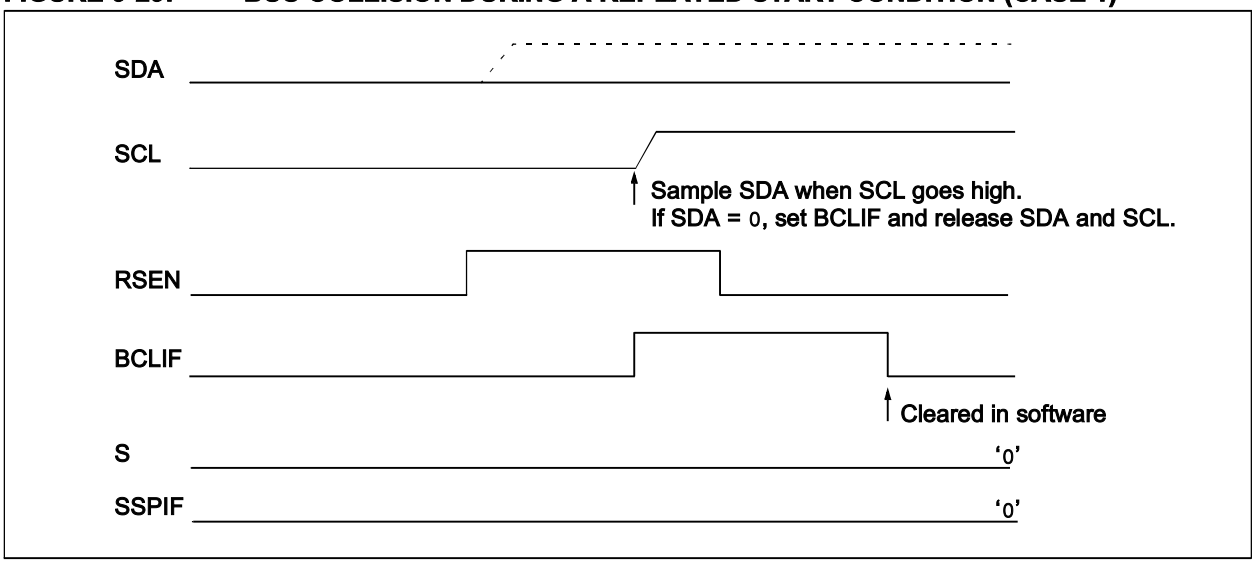
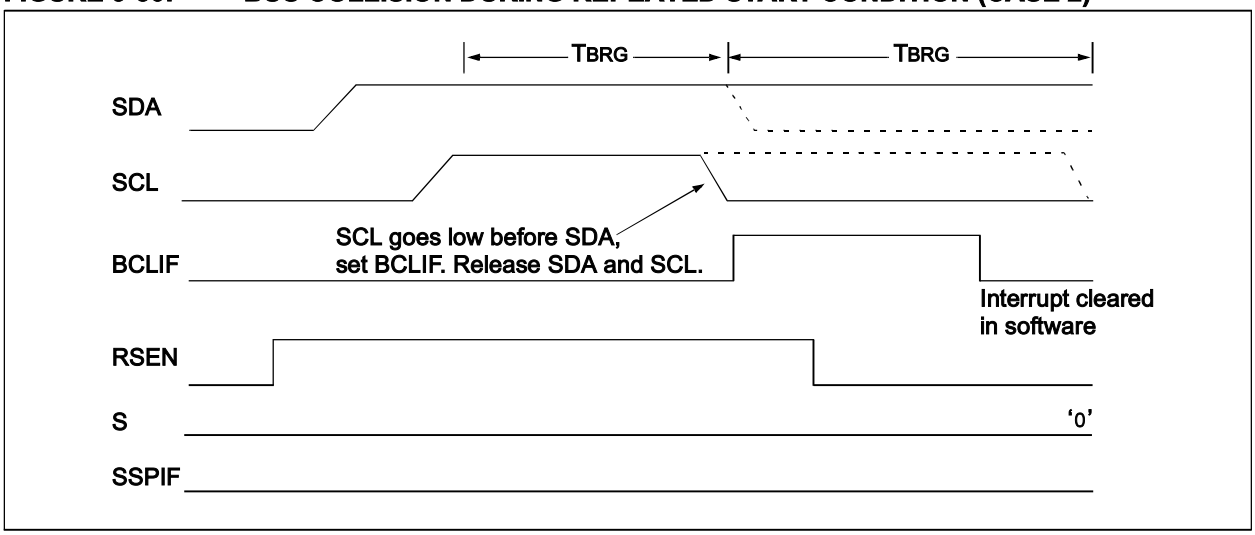


FIGURE 9-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



© 2003 Microchip Technology Inc.

9.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD<6:0> and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 9-31). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 9-32).

FIGURE 9-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)

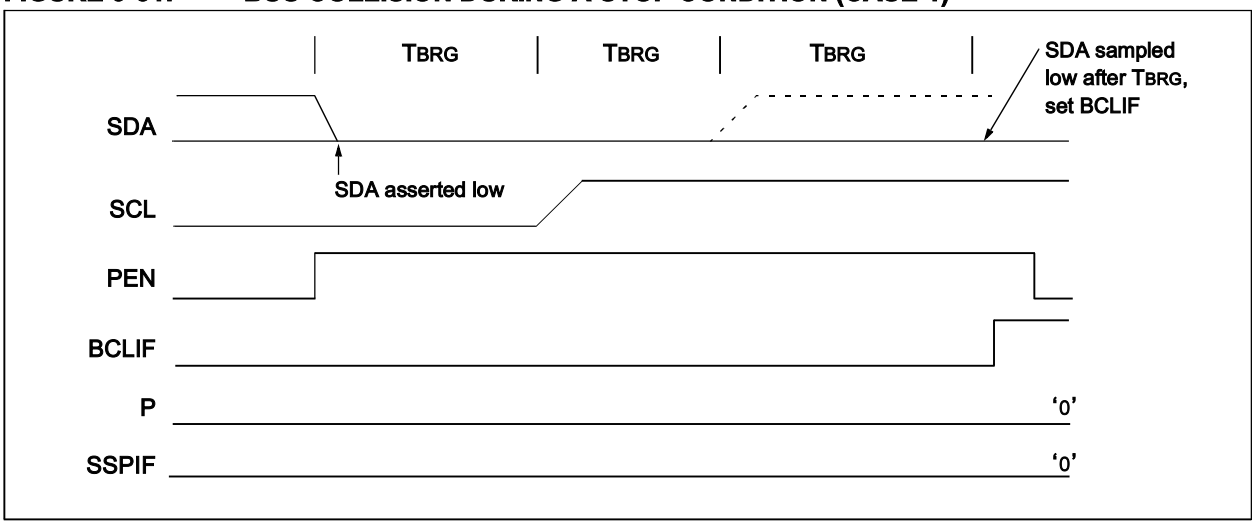
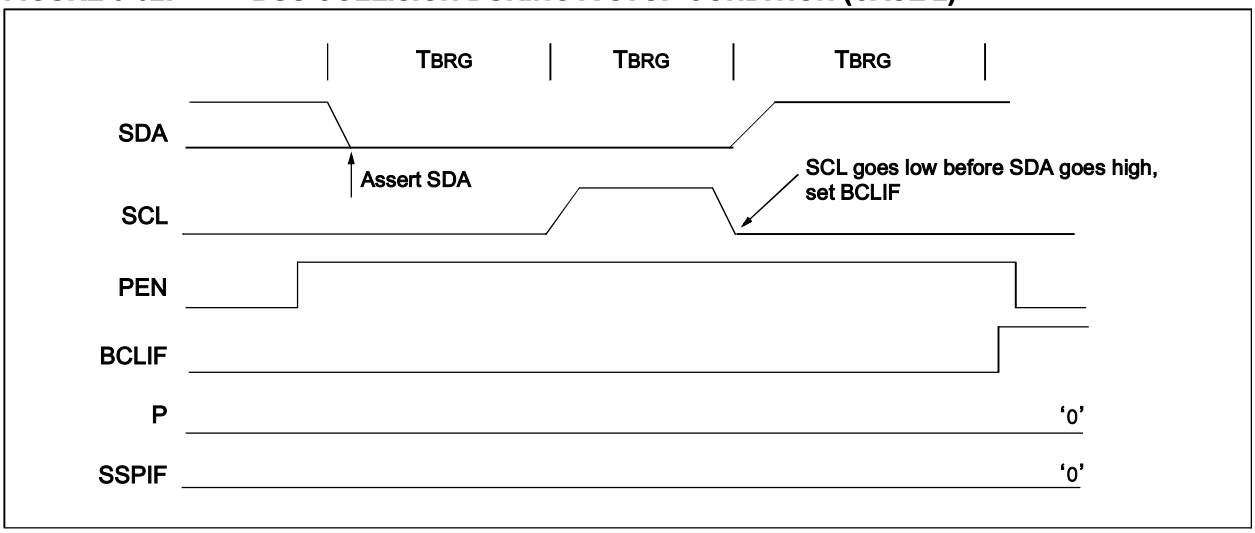


FIGURE 9-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)



NOTES:

© 2003 Microchip Technology Inc.

10.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers, or it can be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The USART can be configured in the following modes:

- Asynchronous (full-duplex)
- Synchronous – Master (half-duplex)
- Synchronous – Slave (half-duplex)

Bit SPEN (RCSTA<7>) and bits TRISC<7:6> have to be set in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter.

The USART module also has a multi-processor communication capability using 9-bit address detection.

REGISTER 10-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
Note: SREN/CREN overrides TXEN in Sync mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be Parity bit

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 10-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data (can be parity bit but must be calculated by user firmware)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

© 2003 Microchip Technology Inc.

10.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 10-1 shows the formula for computation of the baud rate for different USART modes which only apply in Master mode (internal clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRG register can be calculated using the formula in Table 10-1. From this, the error in baud rate can be determined.

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the FOSC/(16 (X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

10.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 10-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = FOSC/(64 (X + 1))	Baud Rate = FOSC/(16 (X + 1))
1	(Synchronous) Baud Rate = FOSC/(4 (X + 1))	N/A

Legend: X = value in SPBRG (0 to 255)

TABLE 10-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

TABLE 10-3: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	-	255	0.977	-	255	0.610	-	255
LOW	312.500	-	0	250.000	-	0	156.250	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207	0.3	0	191
1.2	1.202	0.17	51	1.2	0	47
2.4	2.404	0.17	25	2.4	0	23
9.6	8.929	6.99	6	9.6	0	5
19.2	20.833	8.51	2	19.2	0	2
28.8	31.250	8.51	1	28.8	0	1
33.6	-	-	-	-	-	-
57.6	62.500	8.51	0	57.6	0	0
HIGH	0.244	-	255	0.225	-	255
LOW	62.500	-	0	57.6	-	0

TABLE 10-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000		0	625.000	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-
1.2	1.202	0.17	207	1.2	0	191
2.4	2.404	0.17	103	2.4	0	95
9.6	9.615	0.16	25	9.6	0	23
19.2	19.231	0.16	12	19.2	0	11
28.8	27.798	3.55	8	28.8	0	7
33.6	35.714	6.29	6	32.9	2.04	6
57.6	62.500	8.51	3	57.6	0	3
HIGH	0.977	-	255	0.9	-	255
LOW	250.000	-	0	230.4	-	0

© 2003 Microchip Technology Inc.

10.2 USART Asynchronous Mode

In this mode, the USART uses standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is 8 bits. An on-chip, dedicated, 8-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The transmitter and receiver are functionally independent but use the same data format and baud rate. The baud rate generator produces a clock, either x16 or x64 of the bit shift rate, depending on bit BRGH (TXSTA<2>). Parity is not supported by the hardware but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during Sleep.

Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

10.2.1 USART ASYNCHRONOUS TRANSMITTER

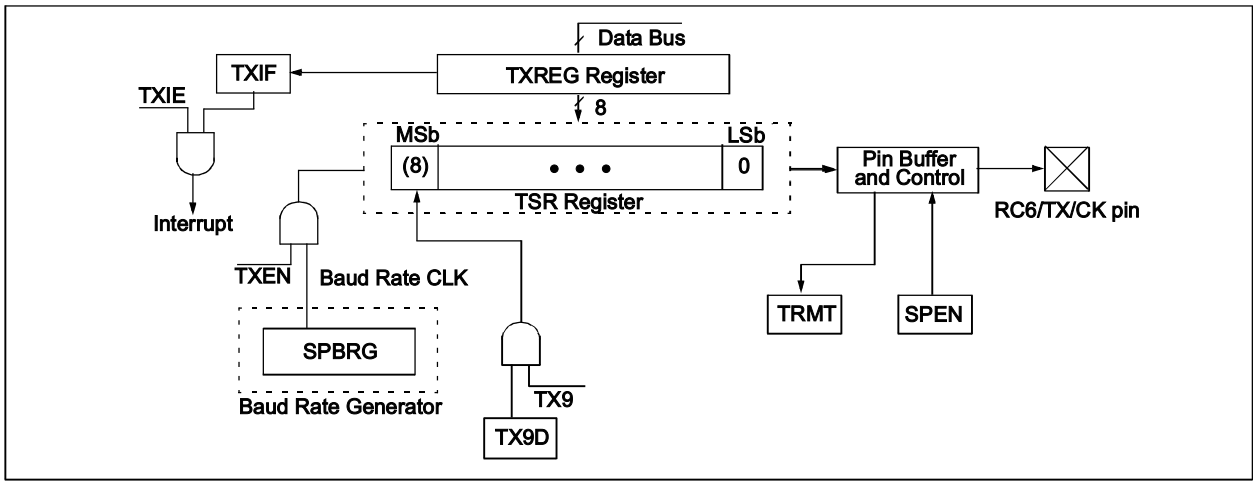
The USART transmitter block diagram is shown in Figure 10-1. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The shift register obtains its data from the Read/Write Transmit Buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit, TXIF (PIR1<4>), is set. This interrupt can be enabled/disabled by setting/clearing enable bit, TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. Status bit TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty.

Note 1:	The TSR register is not mapped in data memory so it is not available to the user.
2:	Flag bit TXIF is set when enable bit TXEN is set. TXIF is cleared by loading TXREG.

Transmission is enabled by setting enable bit, TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data and the Baud Rate Generator (BRG) has produced a shift clock (Figure 10-2). The transmission can also be started by first loading the TXREG register and then setting enable bit TXEN. Normally, when transmission is first started, the TSR register is empty. At that point, transfer to the TXREG register will result in an immediate transfer to TSR, resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 10-3). Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. As a result, the RC6/TX/CK pin will revert to high-impedance.

In order to select 9-bit transmission, transmit bit TX9 (TXSTA<6>) should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG register can result in an immediate transfer of the data to the TSR register (if the TSR is empty). In such a case, an incorrect ninth data bit may be loaded in the TSR register.

FIGURE 10-1: USART TRANSMIT BLOCK DIAGRAM



When setting up an Asynchronous Transmission, follow these steps:

1. Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH (**Section 10.1 "USART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, then set transmit bit TX9.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).
8. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

FIGURE 10-2: ASYNCHRONOUS MASTER TRANSMISSION

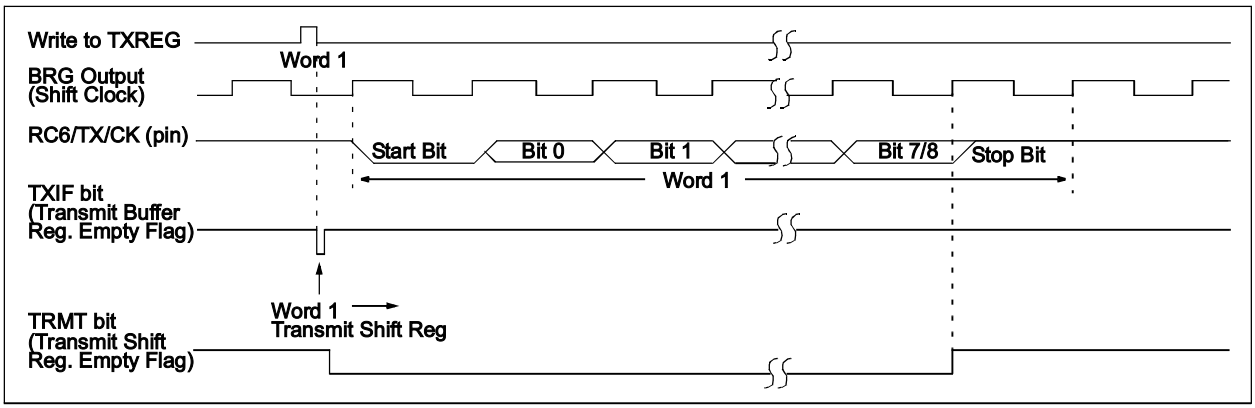
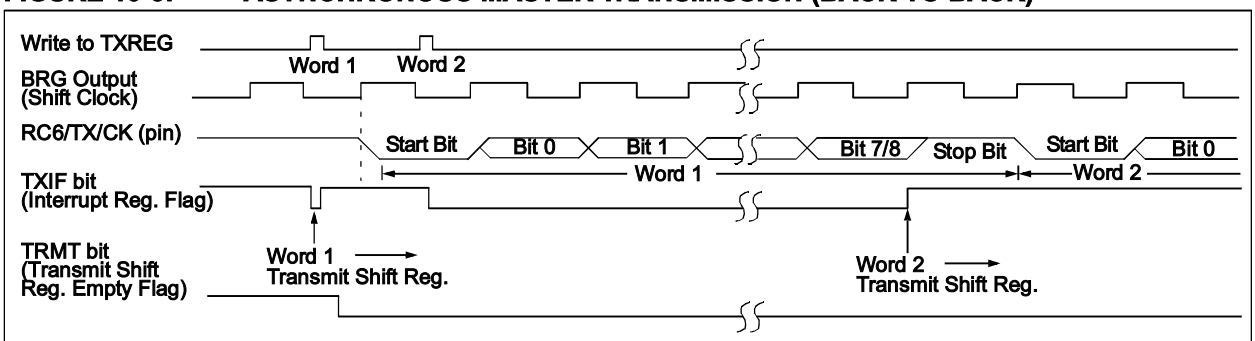


FIGURE 10-3: ASYNCHRONOUS MASTER TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 10-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

© 2003 Microchip Technology Inc.

10.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 10-4. The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter, operating at x16 times the baud rate; whereas the main receive serial shifter operates at the bit rate or at FOSC.

Once Asynchronous mode is selected, reception is enabled by setting bit CREN (RCSTA<4>).

The heart of the receiver is the Receive (Serial) Shift Register (RSR). After sampling the Stop bit, the received data in the RSR is transferred to the RCREG register (if it is empty). If the transfer is complete, flag bit, RCIF (PIR1<5>), is set. The actual interrupt can be enabled/disabled by setting/clearing enable bit, RCIE (PIE1<5>). Flag bit RCIF is a read-only bit which is cleared by the hardware. It is cleared when the RCREG register has been read and is empty. The RCREG is a double-buffered register (i.e., it is a two-deep FIFO). It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting to the RSR register. On the detection of the Stop bit of the third byte, if the RCREG register is still full, the Overrun Error bit, OERR (RCSTA<1>), will be set. The word in the RSR will be lost. The RCREG register can be read twice to retrieve the two bytes in the FIFO. Overrun bit OERR has to be cleared in software. This is done by resetting the receive logic (CREN is cleared and then set). If bit OERR is set, transfers from the RSR register to the RCREG register are inhibited and no further data will be received. It is, therefore, essential to clear error bit OERR if it is set. Framing error bit, FERR (RCSTA<2>), is set if a Stop bit is detected as clear. Bit FERR and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG will load bits RX9D and FERR with new values, therefore, it is essential for the user to read the RCSTA register before reading the RCREG register in order not to lose the old FERR and RX9D information.

FIGURE 10-4: USART RECEIVE BLOCK DIAGRAM

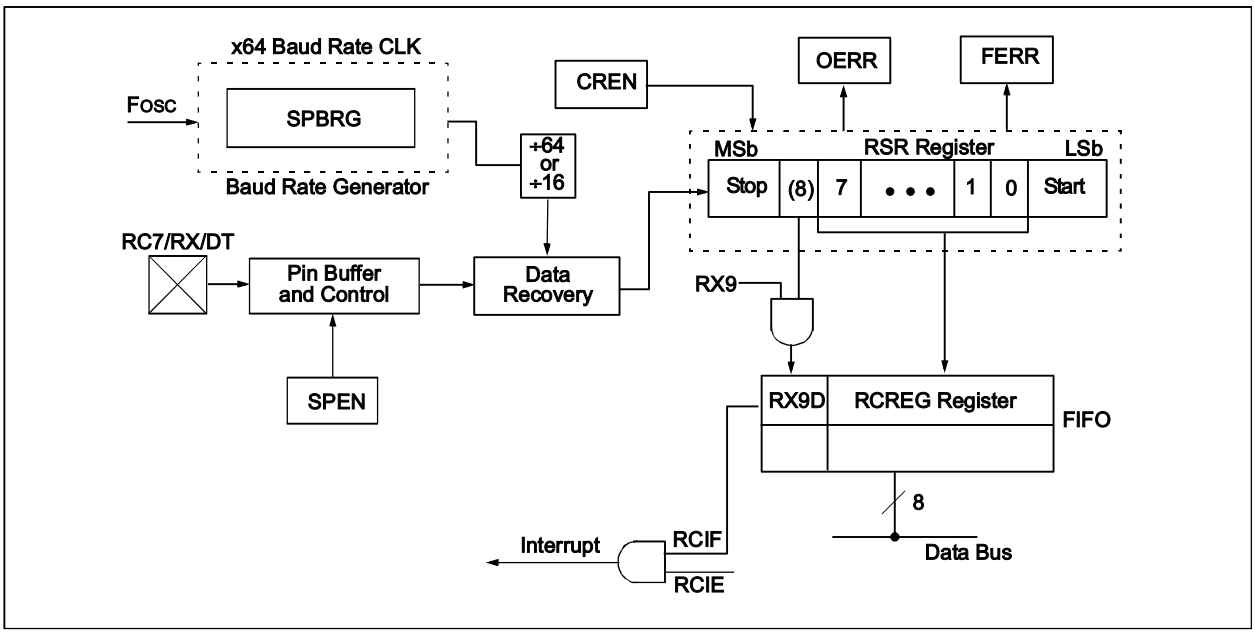
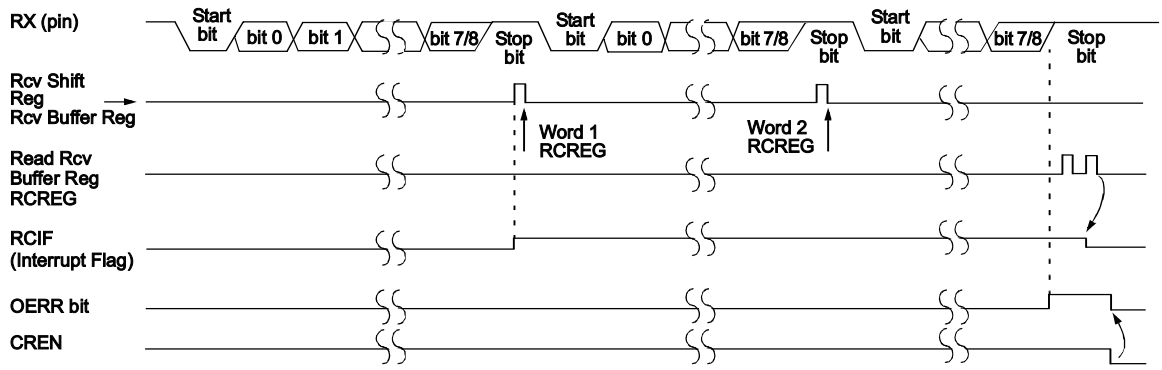


FIGURE 10-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (Receive Buffer) is read after the third word, causing the OERR (Overrun Error) bit to be set.

When setting up an Asynchronous Reception, follow these steps:

1. Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH (**Section 10.1 "USART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit RCIE.
4. If 9-bit reception is desired, then set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE is set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.
10. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

TABLE 10-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

© 2003 Microchip Technology Inc.

10.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

When setting up an Asynchronous Reception with address detect enabled:

- Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH.
- Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
- If interrupts are desired, then set enable bit RCIE.
- Set bit RX9 to enable 9-bit reception.
- Set ADDEN to enable address detect.
- Enable the reception by setting enable bit CREN.
- Flag bit RCIF will be set when reception is complete, and an interrupt will be generated if enable bit RCIE was set.
- Read the RCSTA register to get the ninth bit and determine if any error occurred during reception.
- Read the 8-bit received data by reading the RCREG register to determine if the device is being addressed.
- If any error occurred, clear the error by clearing enable bit CREN.
- If the device has been addressed, clear the ADDEN bit to allow data bytes and address bytes to be read into the receive buffer and interrupt the CPU.

FIGURE 10-6: USART RECEIVE BLOCK DIAGRAM

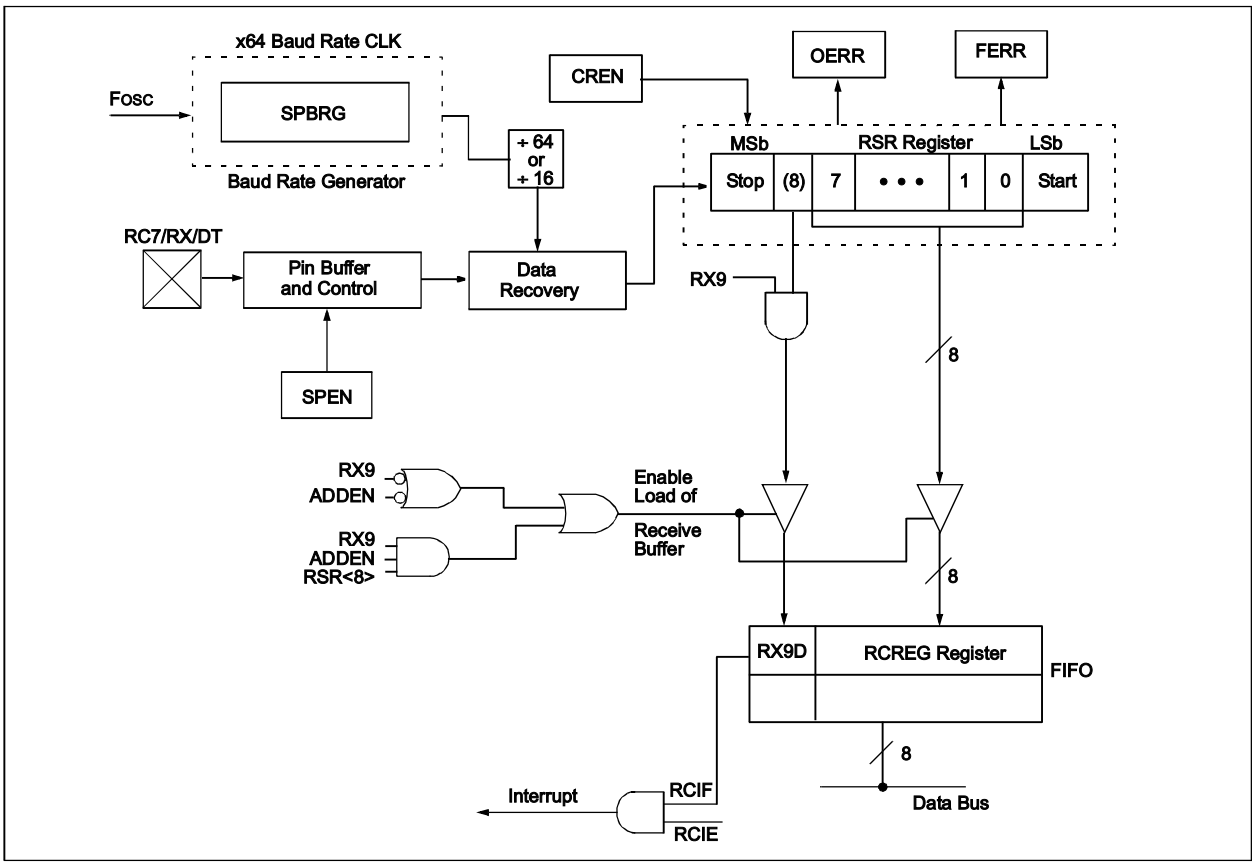
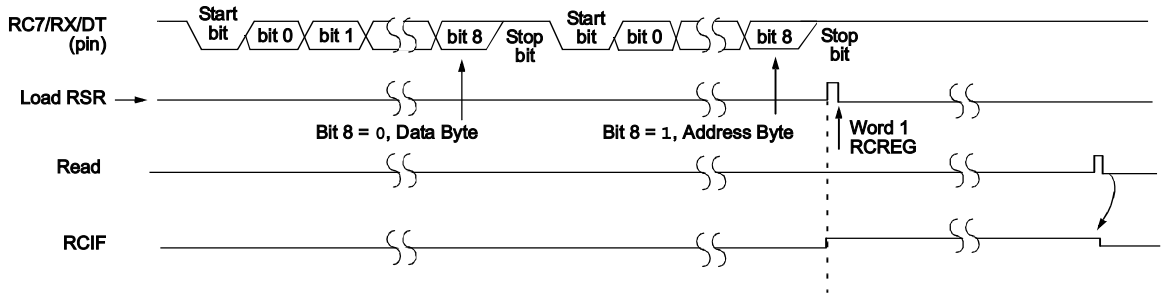
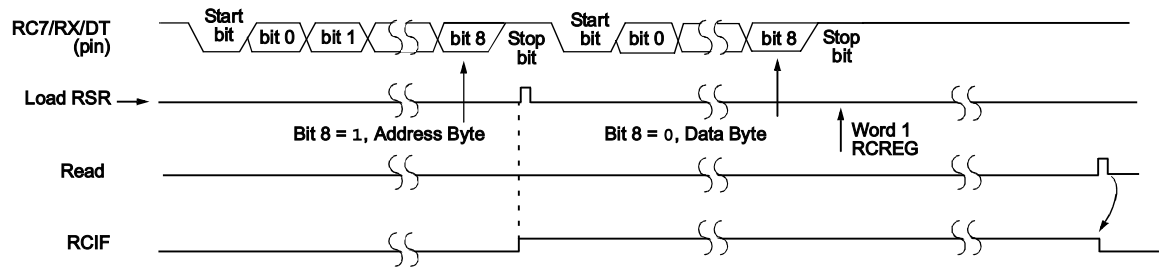


FIGURE 10-7: ASYNCHRONOUS RECEPTION WITH ADDRESS DETECT



Note: This timing diagram shows a data byte followed by an address byte. The data byte is not read into the RCREG (Receive Buffer) because ADDEN = 1.

FIGURE 10-8: ASYNCHRONOUS RECEPTION WITH ADDRESS BYTE FIRST



Note: This timing diagram shows a data byte followed by an address byte. The data byte is not read into the RCREG (Receive Buffer) because ADDEN was not updated and still = 0.

TABLE 10-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

© 2003 Microchip Technology Inc.

10.3 USART Synchronous Master Mode

In Synchronous Master mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit, SYNC (TXSTA<4>). In addition, enable bit, SPEN (RCSTA<7>), is set in order to configure the RC6/TX/CK and RC7/RX/DT I/O pins to CK (clock) and DT (data) lines, respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting bit, CSRC (TXSTA<7>).

10.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 10-6. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The shift register obtains its data from the Read/Write Transmit Buffer register, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCYCLE), the TXREG is empty and interrupt bit, TXIF (PIR1<4>), is set. The interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

Transmission is enabled by setting enable bit, TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the CK line. Data out is stable around the falling edge of the synchronous clock (Figure 10-9). The transmission can also be started by first loading the TXREG register and then setting bit TXEN (Figure 10-10). This is advantageous when slow baud rates are selected since the BRG is kept in Reset when bits TXEN, CREN and SREN are clear. Setting enable bit TXEN will start the BRG, creating a shift clock immediately. Normally, when transmission is first started, the TSR register is empty so a transfer to the TXREG register will result in an immediate transfer to TSR, resulting in an empty TXREG. Back-to-back transfers are possible.

Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. The DT and CK pins will revert to high-impedance. If either bit CREN or bit SREN is set during a transmission, the transmission is aborted and the DT pin reverts to a high-impedance state (for a reception). The CK pin will remain an output if bit CSRC is set (internal clock). The transmitter logic, however, is not reset, although it is disconnected from the pins. In order to reset the transmitter, the user has to clear bit TXEN. If bit SREN is set (to interrupt an on-going transmission and receive a single word), then after the single word is received, bit SREN will be cleared and the serial port will revert back to transmitting since bit TXEN is still set. The DT line will immediately switch from High-Impedance Receive mode to transmit and start driving. To avoid this, bit TXEN should be cleared.

In order to select 9-bit transmission, the TX9 (TXSTA<6>) bit should be set and the ninth bit should be written to bit TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG can result in an immediate transfer of the data to the TSR register (if the TSR is empty). If the TSR was empty and the TXREG was written before writing the “new” TX9D, the “present” value of bit TX9D is loaded.

Steps to follow when setting up a Synchronous Master Transmission:

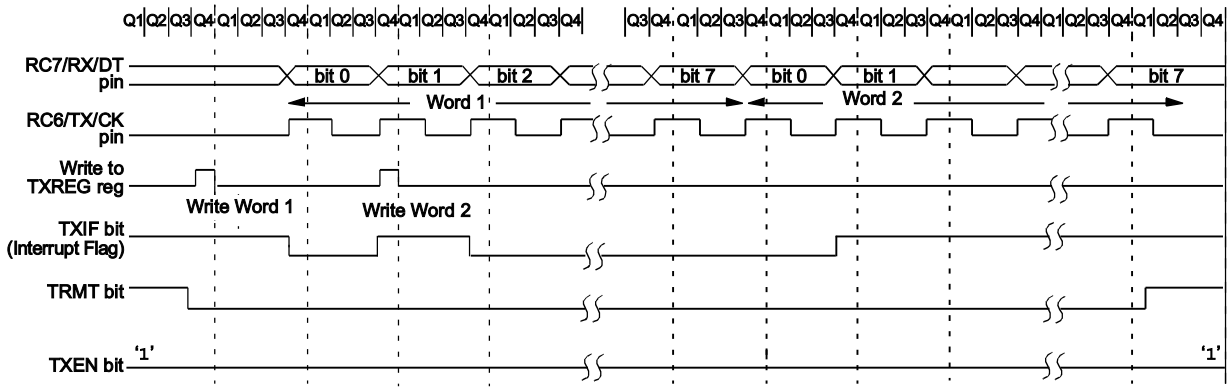
1. Initialize the SPBRG register for the appropriate baud rate (**Section 10.1 “USART Baud Rate Generator (BRG)”**).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.
8. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

TABLE 10-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

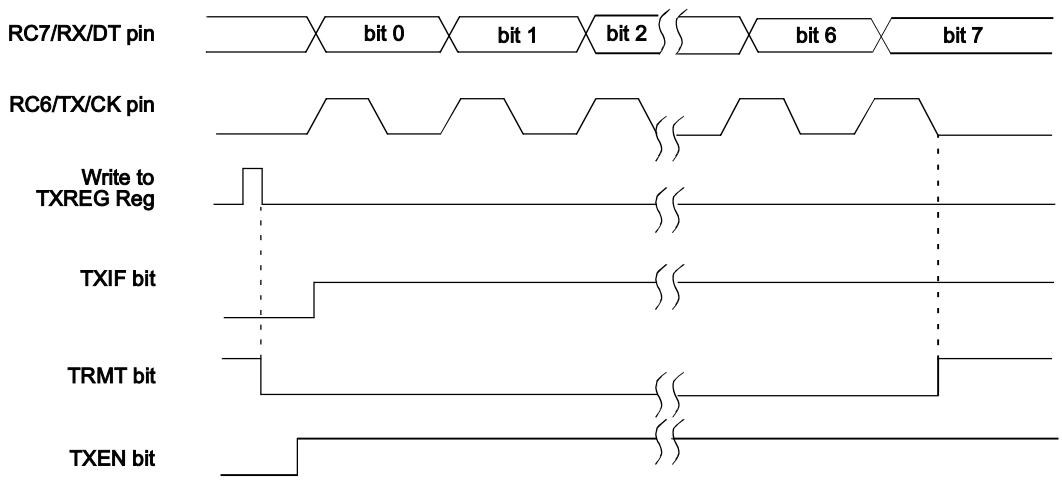
Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.
Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

FIGURE 10-9: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode; SPBRG = 0. Continuous transmission of two 8-bit words.

FIGURE 10-10: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)



© 2003 Microchip Technology Inc.

10.3.2 USART SYNCHRONOUS MASTER RECEPTION

Once Synchronous mode is selected, reception is enabled by setting either enable bit, SREN (RCSTA<5>), or enable bit, CREN (RCSTA<4>). Data is sampled on the RC7/RX/DT pin on the falling edge of the clock. If enable bit SREN is set, then only a single word is received. If enable bit CREN is set, the reception is continuous until CREN is cleared. If both bits are set, CREN takes precedence. After clocking the last bit, the received data in the Receive Shift Register (RSR) is transferred to the RCREG register (if it is empty). When the transfer is complete, interrupt flag bit, RCIF (PIR1<5>), is set. The actual interrupt can be enabled/disabled by setting/clearing enable bit, RCIE (PIE1<5>). Flag bit RCIF is a read-only bit which is reset by the hardware. In this case, it is reset when the RCREG register has been read and is empty. The RCREG is a double-buffered register (i.e., it is a two-deep FIFO). It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting into the RSR register. On the clocking of the last bit of the third byte, if the RCREG register is still full, then Overrun Error bit, OERR (RCSTA<1>), is set. The word in the RSR will be lost. The RCREG register can be read twice to retrieve the two bytes in the FIFO. Bit OERR has to be cleared in software (by clearing bit CREN). If bit OERR is set, transfers from the RSR to the RCREG are inhibited so it is essential to clear bit OERR if it is set. The ninth receive bit is buffered the same way as the receive data. Reading the RCREG register will load bit RX9D with a new value, therefore, it is essential for the user to read the RCSTA register before reading RCREG in order not to lose the old RX9D information.

When setting up a Synchronous Master Reception:

1. Initialize the SPBRG register for the appropriate baud rate (**Section 10.1 "USART Baud Rate Generator (BRG)"**).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, then set enable bit RCIE.
5. If 9-bit reception is desired, then set bit RX9.
6. If a single reception is required, set bit SREN. For continuous reception, set bit CREN.
7. Interrupt flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
8. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If any error occurred, clear the error by clearing bit CREN.
11. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

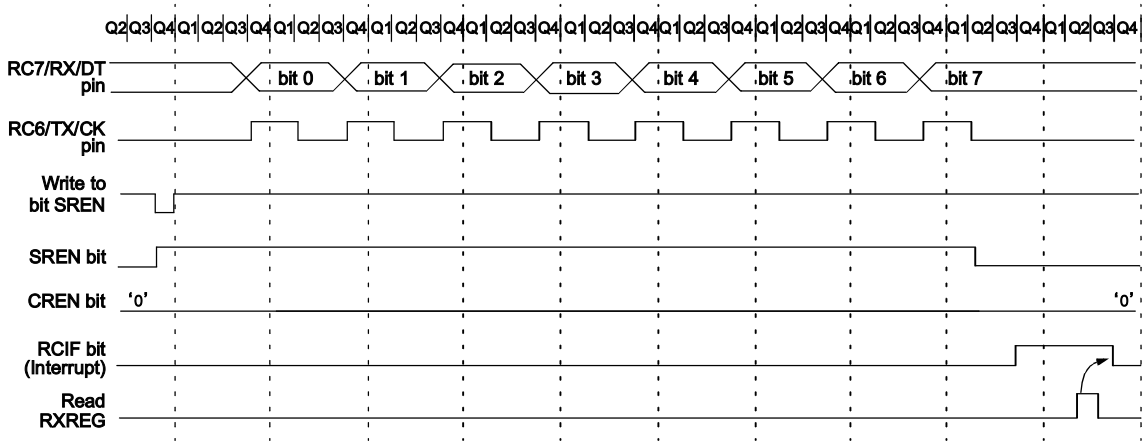
TABLE 10-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

FIGURE 10-11: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



Note: Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRG = 0.

10.4 USART Synchronous Slave Mode

Synchronous Slave mode differs from the Master mode in the fact that the shift clock is supplied externally at the RC6/TX/CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in Sleep mode. Slave mode is entered by clearing bit, CSRC (TXSTA<7>).

10.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes is identical, except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in TXREG register.
c) Flag bit TXIF will not be set.
d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit TXIF will now be set.
e) If enable bit TXIE is set, the interrupt will wake the chip from Sleep and if the global interrupt is enabled, the program will branch to the interrupt vector (0004h).

When setting up a Synchronous Slave Transmission, follow these steps:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. Clear bits CREN and SREN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, then set bit TX9.
5. Enable the transmission by setting enable bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.
8. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

© 2003 Microchip Technology Inc.

TABLE 10-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.
Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices; always maintain these bits clear.

10.4.2 USART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of the Sleep mode. Bit SREN is a "don't care" in Slave mode.

If receive is enabled by setting bit CREN prior to the SLEEP instruction, then a word may be received during Sleep. On completely receiving the word, the RSR register will transfer the data to the RCREG register and if enable bit RCIE bit is set, the interrupt generated will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector (0004h).

When setting up a Synchronous Slave Reception, follow these steps:

1. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. If interrupts are desired, set enable bit RCIE.
3. If 9-bit reception is desired, set bit RX9.
4. To enable reception, set enable bit CREN.
5. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
6. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit CREN.
9. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

TABLE 10-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous slave reception.
Note 1: Bits PSPIE and PSPIF are reserved on 28-pin devices, always maintain these bits clear.

NOTES:

© 2003 Microchip Technology Inc.

11.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has five inputs for the 28-pin devices and eight for the 40/44-pin devices.

The conversion of an analog input signal results in a corresponding 10-bit digital number. The A/D module has high and low-voltage reference input that is software selectable to some combination of VDD, VSS, RA2 or RA3.

The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D clock must be derived from the A/D's internal RC oscillator.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)

The ADCON0 register, shown in Register 11-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 11-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be the voltage reference) or as digital I/O.

Additional information on using the A/D module can be found in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

REGISTER 11-1: ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	**00**	FOSC/2
0	**01**	FOSC/8
0	**10**	FOSC/32
0	**11**	FRC (clock derived from the internal A/D RC oscillator)
1	**00**	FOSC/4
1	**01**	FOSC/16
1	**10**	FOSC/64
1	**11**	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits

000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

Note: The PIC16F873A/876A devices only implement A/D channels 0 through 4; the unimplemented selections are reserved. Do not select any unimplemented channels with these devices.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit

1 = A/D converter module is powered up
0 = A/D converter module is shut-off and consumes no operating current

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 11-2: ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in shaded area and in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	FOSC/2
0	01	FOSC/8
0	10	FOSC/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	FOSC/4
1	01	FOSC/16
1	10	FOSC/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: On any device Reset, the port pins that are multiplexed with analog functions (ANx) are forced to be an analog input.

© 2003 Microchip Technology Inc.

The ADRESH:ADRESL registers contain the 10-bit result of the A/D conversion. When the A/D conversion is complete, the result is loaded into this A/D Result register pair, the GO/$\overline{\text{DONE}}$ bit (ADCON0<2>) is cleared and the A/D interrupt flag bit ADIF is set. The block diagram of the A/D module is shown in Figure 11-1.

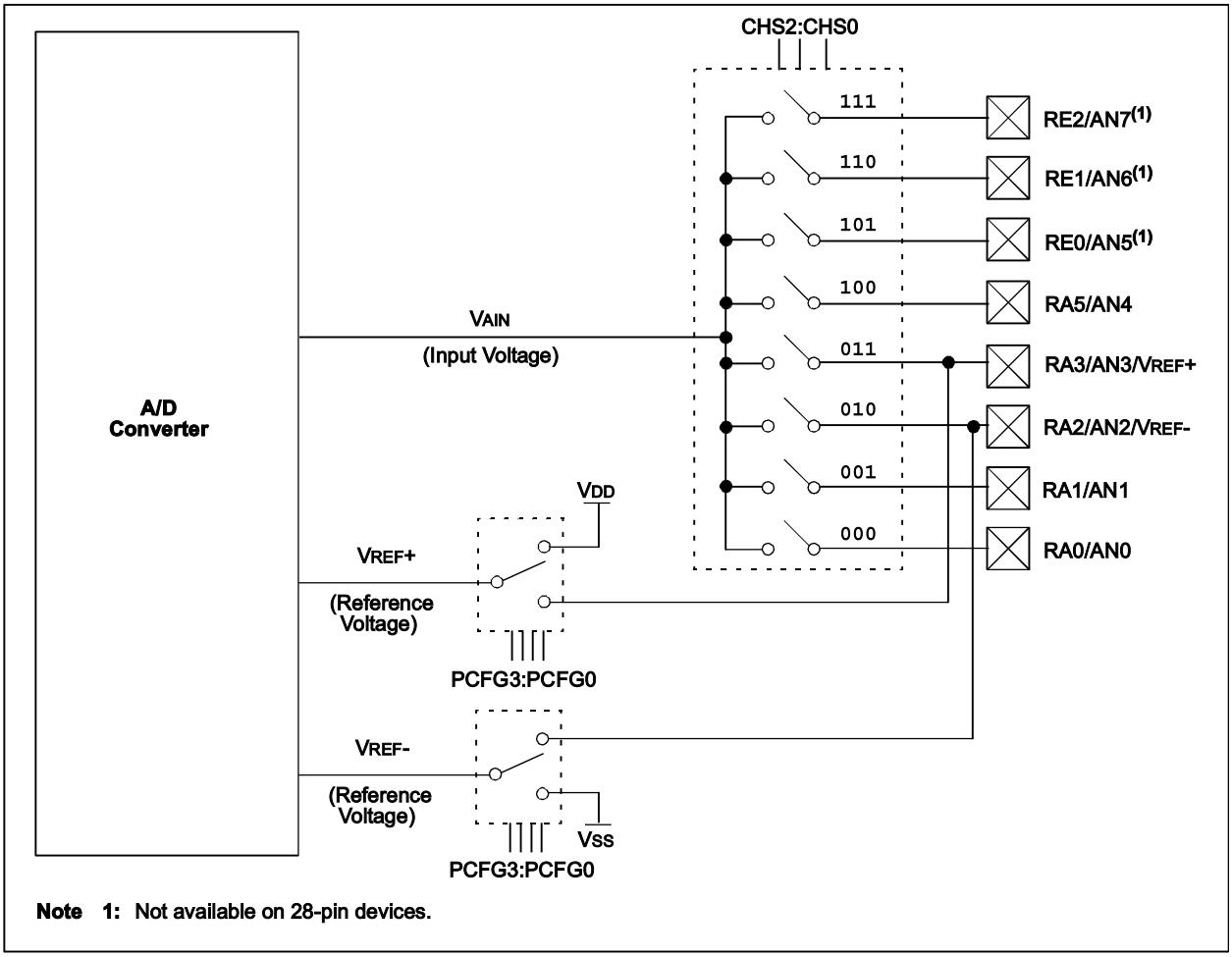
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as inputs.

To determine sample time, see **Section 11.1 "A/D Acquisition Requirements"**. After this acquisition time has elapsed, the A/D conversion can be started.

To do an A/D Conversion, follow these steps:

1. Configure the A/D module:
   - Configure analog pins/voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set PEIE bit
   - Set GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared (interrupts disabled); OR
   - Waiting for the A/D interrupt
6. Read A/D Result register pair (ADRESH:ADRESL), clear bit ADIF if required.
7. For the next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as TAD.

FIGURE 11-1: A/D BLOCK DIAGRAM



Note 1: Not available on 28-pin devices.

11.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 11-2. The source impedance (RS) and the internal sampling switch impedance (RSS) directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD); see Figure 11-2. **The maximum recommended impedance for analog sources is 2.5 kΩ.** As the impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (changed), this acquisition must be done before the conversion can be started.

To calculate the minimum acquisition time, Equation 11-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

To calculate the minimum acquisition time, TACQ, see the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

EQUATION 11-1: ACQUISITION TIME

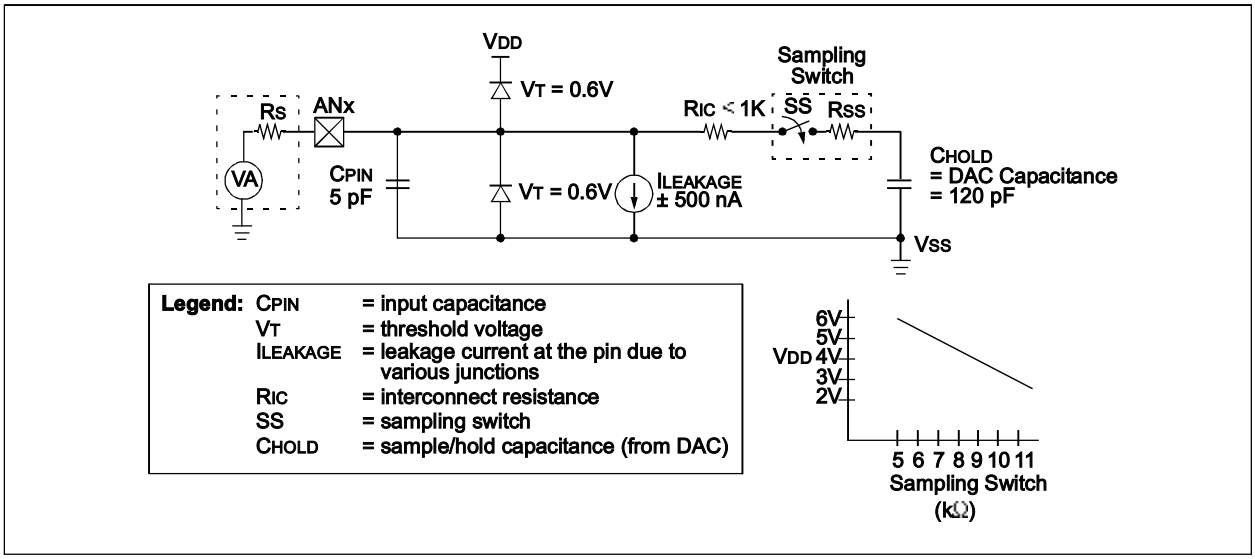
$$
\begin{aligned}
T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\
&= T_{AMP} + T_C + T_{COFF} \\
&= 2\,\mu s + T_C + [(\text{Temperature} - 25°C)(0.05\,\mu s/°C)] \\
T_C &= C_{HOLD}(R_{IC} + R_{SS} + R_S)\ln(1/2047) \\
&= -120\text{ pF}(1\text{ k}\Omega + 7\text{ k}\Omega + 10\text{ k}\Omega)\ln(0.0004885) \\
&= 16.47\,\mu s \\
T_{ACQ} &= 2\,\mu s + 16.47\,\mu s + [(50°C - 25°C)(0.05\,\mu s/°C) \\
&= 19.72\,\mu s
\end{aligned}
$$

Note 1: The reference voltage (VREF) has no effect on the equation since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 2.5 kΩ. This is required to meet the pin leakage specification.

FIGURE 11-2: ANALOG INPUT MODEL



© 2003 Microchip Technology Inc.

11.2 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires a minimum 12 TAD per 10-bit conversion. The source of the A/D conversion clock is software selected. The seven possible options for TAD are:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal A/D module RC oscillator (2-6 µs)

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 1.6 µs.

Table 11-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

11.3 Configuring Analog Port Pins

The ADCON1 and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS2:CHS0 bits and the TRIS bits.

Note 1: When reading the port register, any pin configured as an analog input channel will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.

2: Analog levels on any pin that is defined as a digital input (including the AN7:AN0 pins) may cause the input buffer to consume current that is out of the device specifications.

TABLE 11-1: TAD vs. MAXIMUM DEVICE OPERATING FREQUENCIES (STANDARD DEVICES (F))

AD Clock Source (TAD)		Maximum Device Frequency
Operation	ADCS2:ADCS1:ADCS0	
2 TOSC	000	1.25 MHz
4 TOSC	100	2.5 MHz
8 TOSC	001	5 MHz
16 TOSC	101	10 MHz
32 TOSC	010	20 MHz
64 TOSC	110	20 MHz
RC(1, 2, 3)	x11	**(Note 1)**

Note 1: The RC source has a typical TAD time of 4 µs but can vary between 2-6 µs.

2: When the device frequencies are greater than 1 MHz, the RC A/D conversion clock source is only recommended for Sleep operation.

3: For extended voltage devices (LF), please refer to **Section 17.0 "Electrical Characteristics"**.

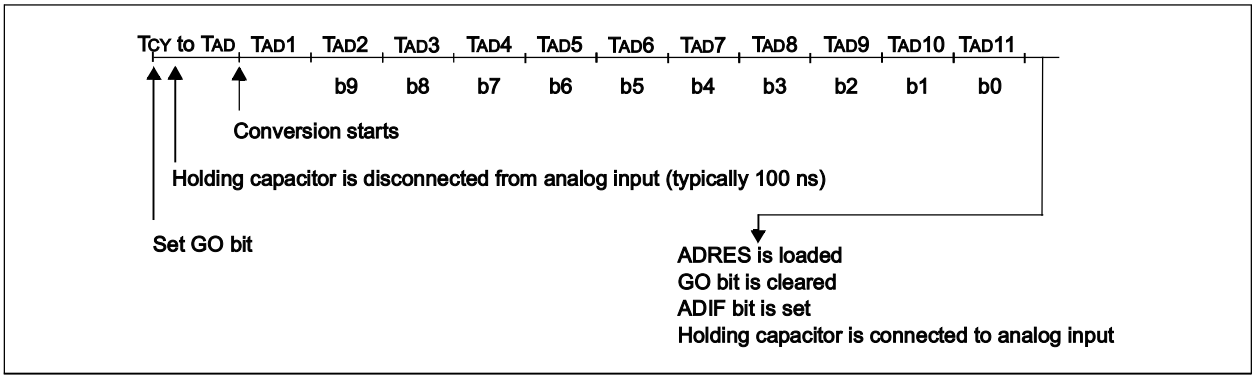
11.4 A/D Conversions

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. That is, the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers). After the A/D conversion is aborted, the next acquisition on the selected channel is automatically started. The GO/$\overline{\text{DONE}}$ bit can then be set to start the conversion.

In Figure 11-3, after the GO bit is set, the first time segment has a minimum of TCY and a maximum of TAD.

Note:	The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

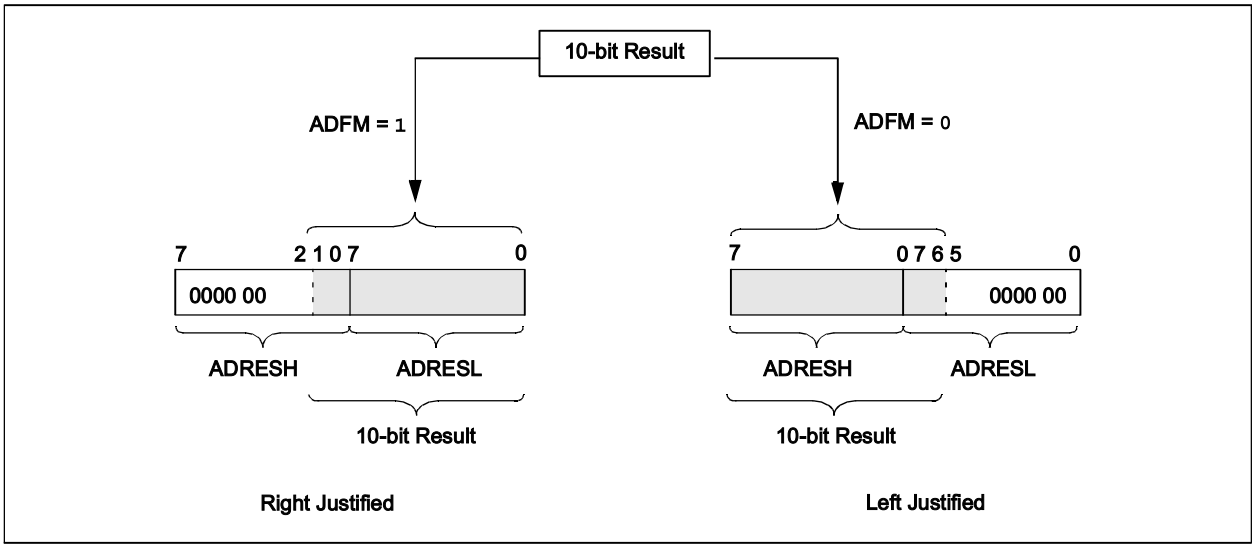
FIGURE 11-3: A/D CONVERSION TAD CYCLES



11.4.1 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16 bits wide. The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D Format Select bit (ADFM) controls this justification. Figure 11-4 shows the operation of the A/D result justification. The extra bits are loaded with '0's. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

FIGURE 11-4: A/D RESULT JUSTIFICATION



© 2003 Microchip Technology Inc.

11.5 A/D Operation During Sleep

The A/D module can operate during Sleep mode. This requires that the A/D clock source be set to RC (ADCS1:ADCS0 = 11). When the RC clock source is selected, the A/D module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed which eliminates all digital switching noise from the conversion. When the conversion is completed, the GO/DONE bit will be cleared and the result loaded into the ADRES register. If the A/D interrupt is enabled, the device will wake-up from Sleep. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

When the A/D clock source is another clock option (not RC), a SLEEP instruction will cause the present conversion to be aborted and the A/D module to be turned off, though the ADON bit will remain set.

Turning off the A/D places the A/D module in its lowest current consumption state.

Note: For the A/D module to operate in Sleep, the A/D clock source must be set to RC (ADCS1:ADCS0 = 11). To allow the conversion to occur during Sleep, ensure the SLEEP instruction immediately follows the instruction that sets the GO/DONE bit.

11.6 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion is aborted. All A/D input pins are configured as analog inputs.

The value that is in the ADRESH:ADRESL registers is not modified for a Power-on Reset. The ADRESH:ADRESL registers will contain unknown data after a Power-on Reset.

TABLE 11-2: REGISTERS/BITS ASSOCIATED WITH A/D

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on MCLR, WDT
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON	0000 00-0	0000 00-0
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	--0u 0000
89h(1)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
09h(1)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

Note 1: These registers are not available on 28-pin devices.

NOTES:

© 2003 Microchip Technology Inc.

12.0 COMPARATOR MODULE

The comparator module contains two analog comparators. The inputs to the comparators are multiplexed with I/O port pins RA0 through RA3, while the outputs are multiplexed to pins RA4 and RA5. The on-chip voltage reference (**Section 13.0 "Comparator Voltage Reference Module"**) can also be an input to the comparators.

The CMCON register (Register 12-1) controls the comparator input and output multiplexers. A block diagram of the various comparator configurations is shown in Figure 12-1.

REGISTER 12-1: CMCON REGISTER

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
bit 7							bit 0

bit 7 **C2OUT**: Comparator 2 Output bit
When C2INV = 0:
1 = C2 VIN+ > C2 VIN-
0 = C2 VIN+ < C2 VIN-
When C2INV = 1:
1 = C2 VIN+ < C2 VIN-
0 = C2 VIN+ > C2 VIN-

bit 6 **C1OUT**: Comparator 1 Output bit
When C1INV = 0:
1 = C1 VIN+ > C1 VIN-
0 = C1 VIN+ < C1 VIN-
When C1INV = 1:
1 = C1 VIN+ < C1 VIN-
0 = C1 VIN+ > C1 VIN-

bit 5 **C2INV**: Comparator 2 Output Inversion bit
1 = C2 output inverted
0 = C2 output not inverted

bit 4 **C1INV**: Comparator 1 Output Inversion bit
1 = C1 output inverted
0 = C1 output not inverted

bit 3 **CIS**: Comparator Input Switch bit
When CM2:CM0 = 110:
1 = C1 VIN- connects to RA3/AN3
C2 VIN- connects to RA2/AN2
0 = C1 VIN- connects to RA0/AN0
C2 VIN- connects to RA1/AN1

bit 2 **CM2:CM0**: Comparator Mode bits
Figure 12-1 shows the Comparator modes and CM2:CM0 bit settings.

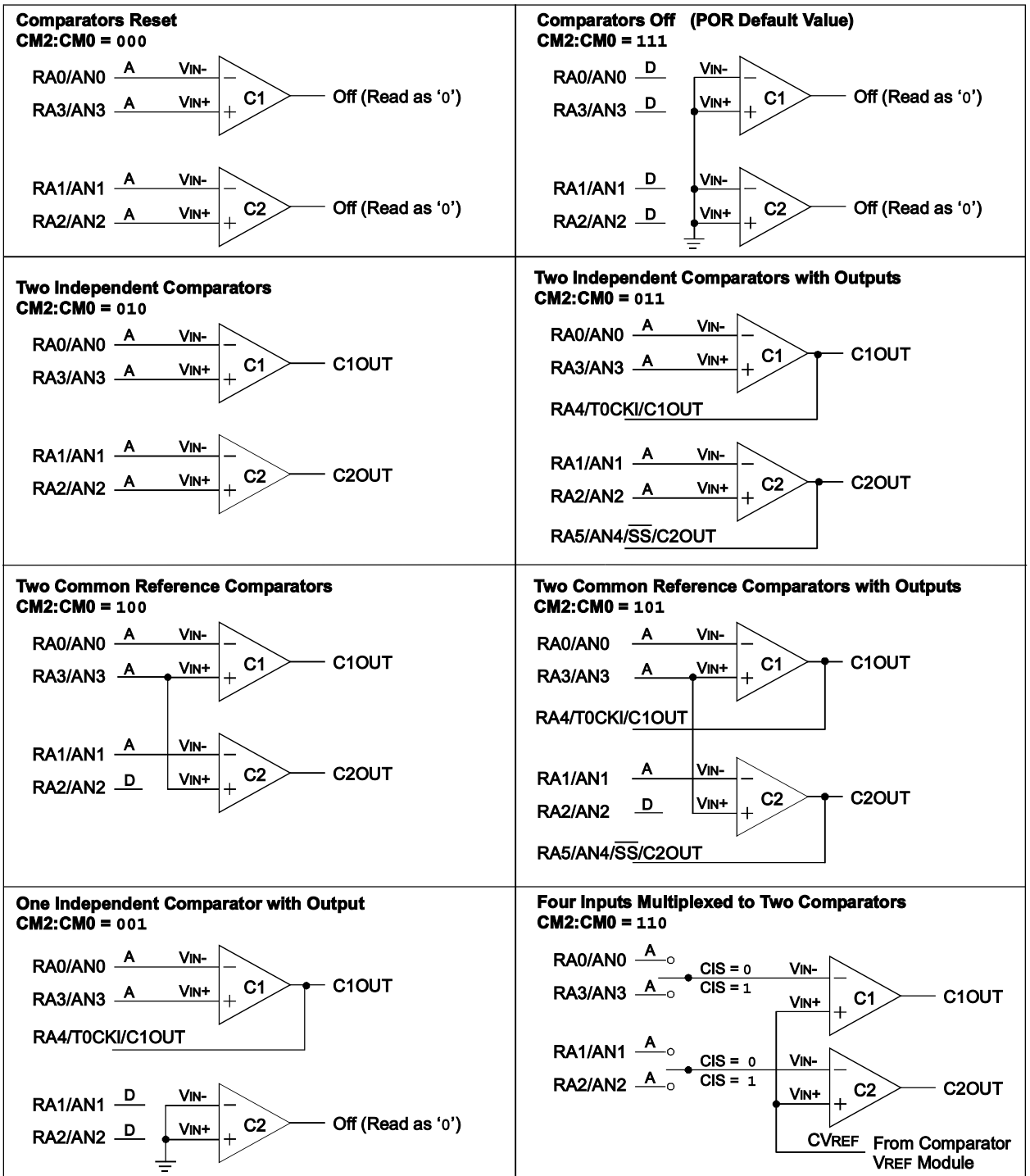
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

12.1 Comparator Configuration

There are eight modes of operation for the comparators. The CMCON register is used to select these modes. Figure 12-1 shows the eight possible modes. The TRISA register controls the data direction of the comparator pins for each mode. If the Comparator mode is changed, the comparator output level may not be valid for the specified mode change delay shown in **Section 17.0 "Electrical Characteristics"**.

Note:	Comparator interrupts should be disabled during a Comparator mode change. Otherwise, a false interrupt may occur.

FIGURE 12-1: COMPARATOR I/O OPERATING MODES



A = Analog Input, port reads zeros always. D = Digital Input. CIS (CMCON<3>) is the Comparator Input Switch.

© 2003 Microchip Technology Inc.

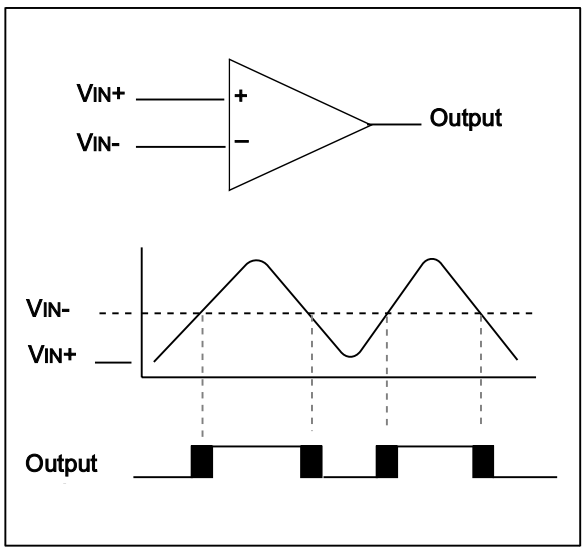
12.2 Comparator Operation

A single comparator is shown in Figure 12-2 along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 12-2 represent the uncertainty due to input offsets and response time.

12.3 Comparator Reference

An external or internal reference signal may be used depending on the comparator operating mode. The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is adjusted accordingly (Figure 12-2).

FIGURE 12-2: SINGLE COMPARATOR



12.3.1 EXTERNAL REFERENCE SIGNAL

When external voltage references are used, the comparator module can be configured to have the comparators operate from the same or different reference sources. However, threshold detector applications may require the same reference. The reference signal must be between VSS and VDD and can be applied to either pin of the comparator(s).

12.3.2 INTERNAL REFERENCE SIGNAL

The comparator module also allows the selection of an internally generated voltage reference for the comparators. **Section 13.0 "Comparator Voltage Reference Module"** contains a detailed description of the Comparator Voltage Reference module that provides this signal. The internal reference signal is used when comparators are in mode, CM<2:0> = 110 (Figure 12-1). In this mode, the internal voltage reference is applied to the VIN+ pin of both comparators.

12.4 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. If the internal reference is changed, the maximum delay of the internal voltage reference must be considered when using the comparator outputs. Otherwise, the maximum delay of the comparators should be used (**Section 17.0 "Electrical Characteristics"**).

12.5 Comparator Outputs

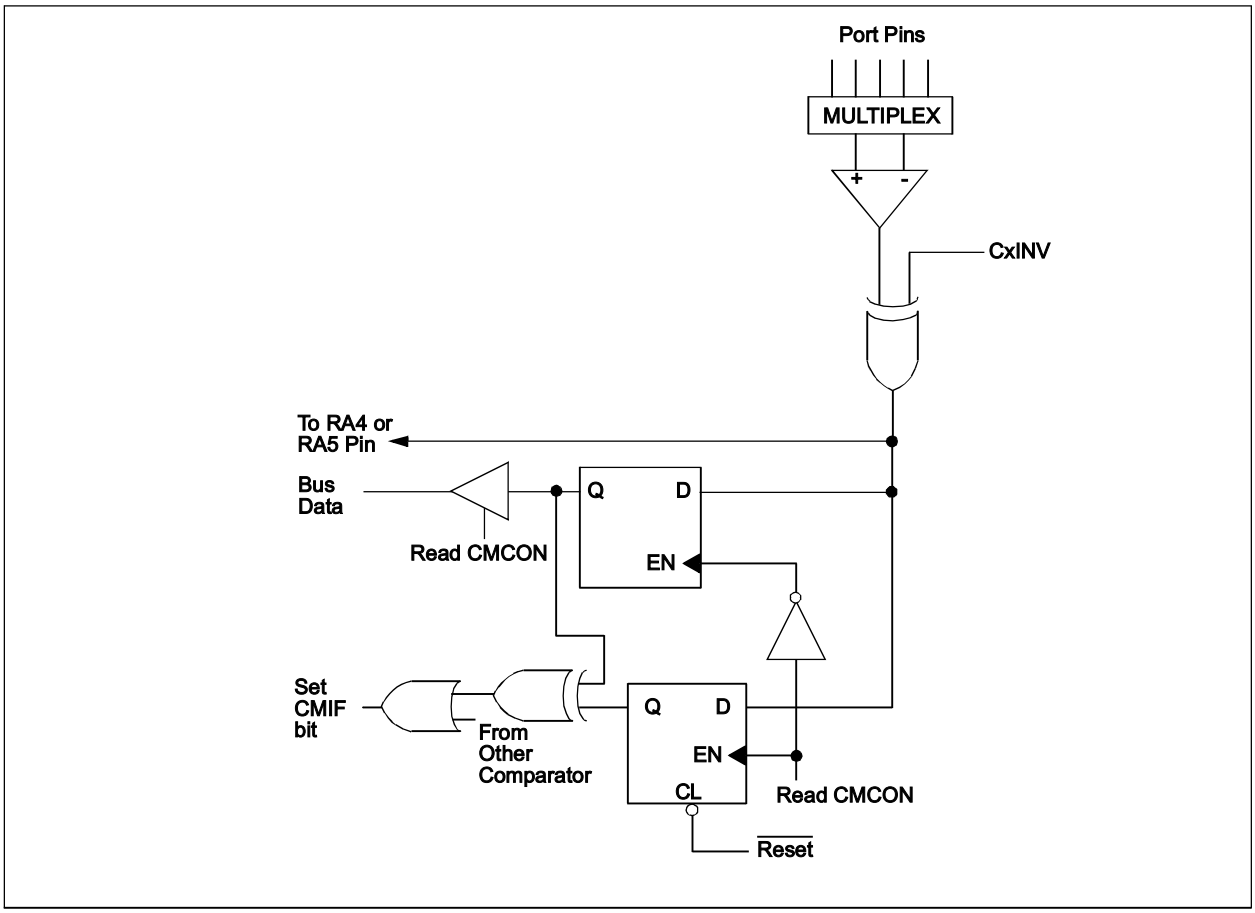
The comparator outputs are read through the CMCON register. These bits are read-only. The comparator outputs may also be directly output to the RA4 and RA5 I/O pins. When enabled, multiplexors in the output path of the RA4 and RA5 pins will switch and the output of each pin will be the unsynchronized output of the comparator. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications. Figure 12-3 shows the comparator output block diagram.

The TRISA bits will still function as an output enable/disable for the RA4 and RA5 pins while in this mode.

The polarity of the comparator outputs can be changed using the C2INV and C1INV bits (CMCON<4:5>).

Note 1:	When reading the Port register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert an analog input according to the Schmitt Trigger input specification.
2:	Analog levels on any pin defined as a digital input may cause the input buffer to consume more current than is specified.
3:	RA4 is an open collector I/O pin. When used as an output, a pull-up resistor is required.

FIGURE 12-3: COMPARATOR OUTPUT BLOCK DIAGRAM



12.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that occurred. The CMIF bit (PIR registers) is the Comparator Interrupt Flag. The CMIF bit must be reset by clearing it ('0'). Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

The CMIE bit (PIE registers) and the PEIE bit (INTCON register) must be set to enable the interrupt. In addition, the GIE bit must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

Note:	If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF (PIR registers) interrupt flag may not get set.

The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of CMCON will end the mismatch condition.

b) Clear flag bit CMIF.

A mismatch condition will continue to set flag bit CMIF. Reading CMCON will end the mismatch condition and allow flag bit CMIF to be cleared.

© 2003 Microchip Technology Inc.

12.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake-up the device from Sleep mode when enabled. While the comparator is powered up, higher Sleep currents than shown in the power-down current specification will occur. Each operational comparator will consume additional current as shown in the comparator specifications. To minimize power consumption while in Sleep mode, turn off the comparators, CM<2:0> = 111, before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

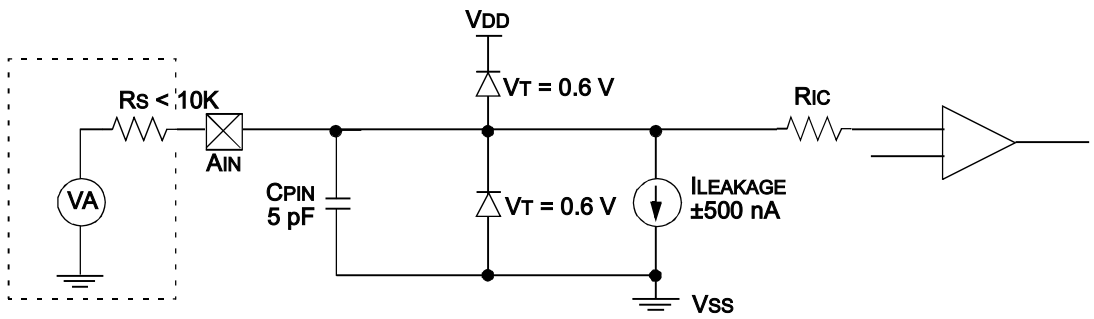
12.8 Effects of a Reset

A device Reset forces the CMCON register to its Reset state, causing the comparator module to be in the Comparator Off mode, CM<2:0> = 111. This ensures compatibility to the PIC16F87X devices.

12.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 12-4. Since the analog pins are connected to a digital output, they have reverse biased diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up condition may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 12-4: ANALOG INPUT MODEL



Legend:

CPIN	=	Input Capacitance
VT	=	Threshold Voltage
ILEAKAGE	=	Leakage Current at the pin due to various junctions
RIC	=	Interconnect Resistance
RS	=	Source Impedance
VA	=	Analog Voltage

TABLE 12-1: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INTIE	RBIE	TMR0IF	INTIF	RBIF	0000 000x	0000 000u
0Dh	PIR2	—	CMIF	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-- 0000	-0-- 0000
8Dh	PIE2	—	CMIE	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-- 0000	-0-- 0000
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

© 2003 Microchip Technology Inc.

13.0 COMPARATOR VOLTAGE REFERENCE MODULE

The Comparator Voltage Reference Generator is a 16-tap resistor ladder network that provides a fixed voltage reference when the comparators are in mode '110'. A programmable register controls the function of the reference generator. Register 13-1 lists the bit functions of the CVRCON register.

As shown in Figure 13-1, the resistor ladder is segmented to provide two ranges of CVREF values and has a power-down function to conserve power when the reference is not being used. The comparator reference supply voltage (also referred to as CVRSRC) comes directly from VDD. It should be noted, however, that the voltage at the top of the ladder is CVRSRC – VSAT, where VSAT is the saturation voltage of the power switch transistor. This reference will only be as accurate as the values of CVRSRC and VSAT.

The output of the reference generator may be connected to the RA2/AN2/VREF-/CVREF pin. This can be used as a simple D/A function by the user if a very high-impedance load is used. The primary purpose of this function is to provide a test path for testing the reference generator function.

REGISTER 13-1: CVRCON CONTROL REGISTER (ADDRESS 9Dh)

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0
bit 7							bit 0

bit 7 **CVREN**: Comparator Voltage Reference Enable bit
1 = CVREF circuit powered on
0 = CVREF circuit powered down

bit 6 **CVROE:** Comparator VREF Output Enable bit
1 = CVREF voltage level is output on RA2/AN2/VREF-/CVREF pin
0 = CVREF voltage level is disconnected from RA2/AN2/VREF-/CVREF pin

bit 5 **CVRR**: Comparator VREF Range Selection bit
1 = 0 to 0.75 CVRSRC, with CVRSRC/24 step size
0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size

bit 4 **Unimplemented:** Read as '0'

bit 3-0 **CVR3:CVR0:** Comparator VREF Value Selection bits 0 ≤ VR3:VR0 ≤ 15
When CVRR = 1:
CVREF = (VR<3:0>/ 24) • (CVRSRC)
When CVRR = 0:
CVREF = 1/4 • (CVRSRC) + (VR3:VR0/ 32) • (CVRSRC)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 13-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM

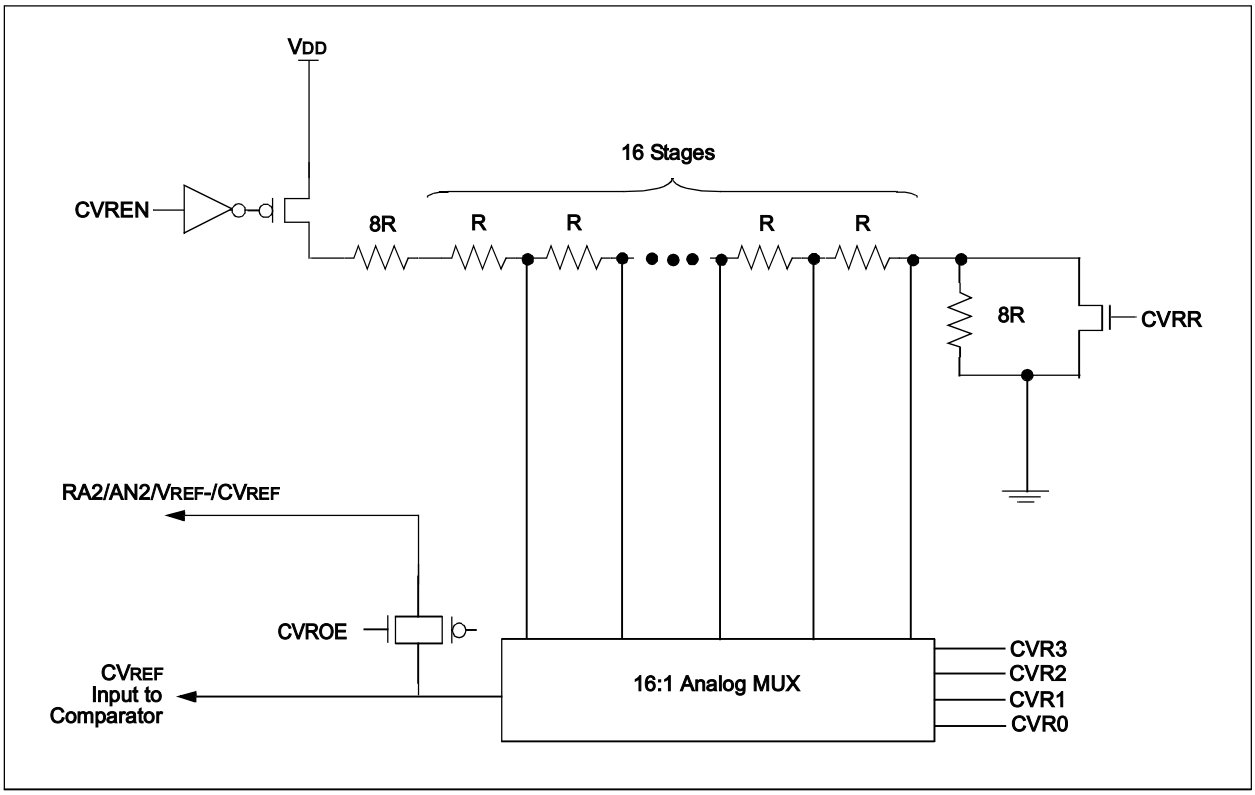


TABLE 13-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'.
Shaded cells are not used with the comparator voltage reference.

© 2003 Microchip Technology Inc.

14.0 SPECIAL FEATURES OF THE CPU

All PIC16F87XA devices have a host of features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection. These are:

- Oscillator Selection
- Reset
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Sleep
- Code Protection
- ID Locations
- In-Circuit Serial Programming
- Low-Voltage In-Circuit Serial Programming
- In-Circuit Debugger

PIC16F87XA devices have a Watchdog Timer which can be shut-off only through configuration bits. It runs off its own RC oscillator for added reliability.

There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay of 72 ms (nominal) on power-up only. It is designed to keep the part in Reset while the power supply stabilizes. With these two timers on-chip, most applications need no external Reset circuitry.

Sleep mode is designed to offer a very low current power-down mode. The user can wake-up from Sleep through external Reset, Watchdog Timer wake-up or through an interrupt.

Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost while the LP crystal option saves power. A set of configuration bits is used to select various options.

Additional information on special features is available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

14.1 Configuration Bits

The configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1') to select various device configurations. The erased or unprogrammed value of the Configuration Word register is 3FFFh. These bits are mapped in program memory location 2007h.

It is important to note that address 2007h is beyond the user program memory space which can be accessed only during programming.

REGISTER 14-1: CONFIGURATION WORD (ADDRESS 2007h)(1)

R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
CP	—	DEBUG	WRT1	WRT0	CPD	LVP	BOREN	—	—	$\overline{\text{PWRTEN}}$	WDTEN	Fosc1	Fosc0
bit 13													bit0

bit 13 **CP:** Flash Program Memory Code Protection bit
1 = Code protection off
0 = All program memory code-protected

bit 12 **Unimplemented:** Read as '1'

bit 11 **DEBUG:** In-Circuit Debugger Mode bit
1 = In-Circuit Debugger disabled, RB6 and RB7 are general purpose I/O pins
0 = In-Circuit Debugger enabled, RB6 and RB7 are dedicated to the debugger

bit 10-9 **WRT1:WRT0** Flash Program Memory Write Enable bits
For PIC16F876A/877A:
11 = Write protection off; all program memory may be written to by EECON control
10 = 0000h to 00FFh write-protected; 0100h to 1FFFh may be written to by EECON control
01 = 0000h to 07FFh write-protected; 0800h to 1FFFh may be written to by EECON control
00 = 0000h to 0FFFh write-protected; 1000h to 1FFFh may be written to by EECON control
For PIC16F873A/874A:
11 = Write protection off; all program memory may be written to by EECON control
10 = 0000h to 00FFh write-protected; 0100h to 0FFFh may be written to by EECON control
01 = 0000h to 03FFh write-protected; 0400h to 0FFFh may be written to by EECON control
00 = 0000h to 07FFh write-protected; 0800h to 0FFFh may be written to by EECON control

bit 8 **CPD:** Data EEPROM Memory Code Protection bit
1 = Data EEPROM code protection off
0 = Data EEPROM code-protected

bit 7 **LVP**: Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit
1 = RB3/PGM pin has PGM function; low-voltage programming enabled
0 = RB3 is digital I/O, HV on $\overline{\text{MCLR}}$ must be used for programming

bit 6 **BOREN**: Brown-out Reset Enable bit
1 = BOR enabled
0 = BOR disabled

bit 5-4 **Unimplemented:** Read as '1'

bit 3 **$\overline{\text{PWRTEN}}$**: Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 2 **WDTEN**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **Fosc1:Fosc0**: Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

Note 1: The erased (unprogrammed) value of the Configuration Word is 3FFFh.

© 2003 Microchip Technology Inc.

14.2 Oscillator Configurations

14.2.1 OSCILLATOR TYPES

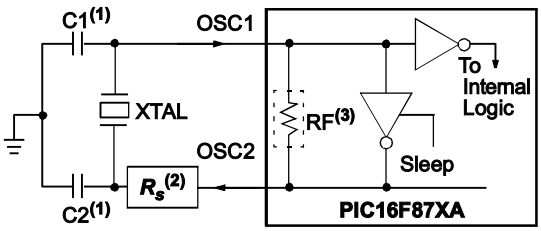
The PIC16F87XA can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1 and FOSC0) to select one of these four modes:

- LP Low-Power Crystal
- XT Crystal/Resonator
- HS High-Speed Crystal/Resonator
- RC Resistor/Capacitor

14.2.2 CRYSTAL OSCILLATOR/CERAMIC RESONATORS

In XT, LP or HS modes, a crystal or ceramic resonator is connected to the OSC1/CLKI and OSC2/CLKO pins to establish oscillation (Figure 14-1). The PIC16F87XA oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturer's specifications. When in XT, LP or HS modes, the device can have an external clock source to drive the OSC1/CLKI pin (Figure 14-2).

FIGURE 14-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP OSC CONFIGURATION)



Note 1: See Table 14-1 and Table 14-2 for recommended values of C1 and C2.

2: A series resistor (R_S) may be required for AT strip cut crystals.

3: RF varies with the crystal chosen.

FIGURE 14-2: EXTERNAL CLOCK INPUT OPERATION (HS, XT OR LP OSC CONFIGURATION)

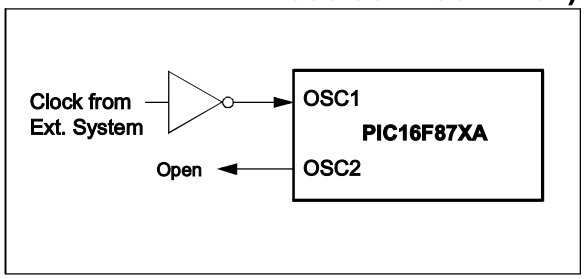


TABLE 14-1: CERAMIC RESONATORS

Ranges Tested:			
Mode	**Freq.**	**OSC1**	**OSC2**
XT	455 kHz 2.0 MHz 4.0 MHz	68-100 pF 15-68 pF 15-68 pF	68-100 pF 15-68 pF 15-68 pF
HS	8.0 MHz 16.0 MHz	10-68 pF 10-22 pF	10-68 pF 10-22 pF
These values are for design guidance only. See notes following Table 14-2.			
Resonators Used:			
2.0 MHz	Murata Erie CSA2.00MG	± 0.5%	
4.0 MHz	Murata Erie CSA4.00MG	± 0.5%	
8.0 MHz	Murata Erie CSA8.00MT	± 0.5%	
16.0 MHz	Murata Erie CSA16.00MX	± 0.5%	
All resonators used did not have built-in capacitors.			

TABLE 14-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF
These values are for design guidance only. See notes following this table.			
Crystals Used			
32 kHz	Epson C-001R32.768K-A		± 20 PPM
200 kHz	STD XTL 200.000KHz		± 20 PPM
1 MHz	ECS ECS-10-13-1		± 50 PPM
4 MHz	ECS ECS-40-20-1		± 50 PPM
8 MHz	EPSON CA-301 8.000M-C		± 30 PPM
20 MHz	EPSON CA-301 20.000M-C		± 30 PPM

Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

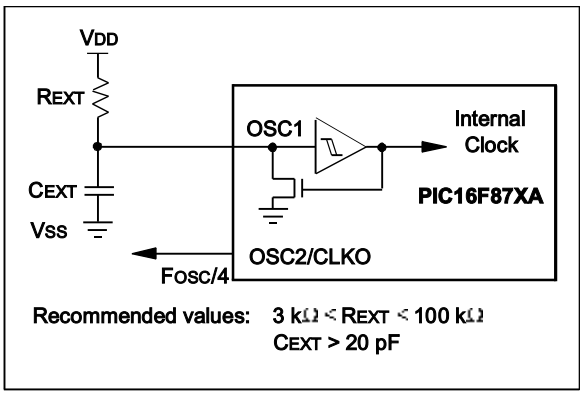
3: R_S may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specification.

4: When migrating from other PICmicro® devices, oscillator performance should be verified.

14.2.3 RC OSCILLATOR

For timing insensitive applications, the "RC" device option offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 14-3 shows how the R/C combination is connected to the PIC16F87XA.

FIGURE 14-3: RC OSCILLATOR MODE



© 2003 Microchip Technology Inc.

14.3 Reset

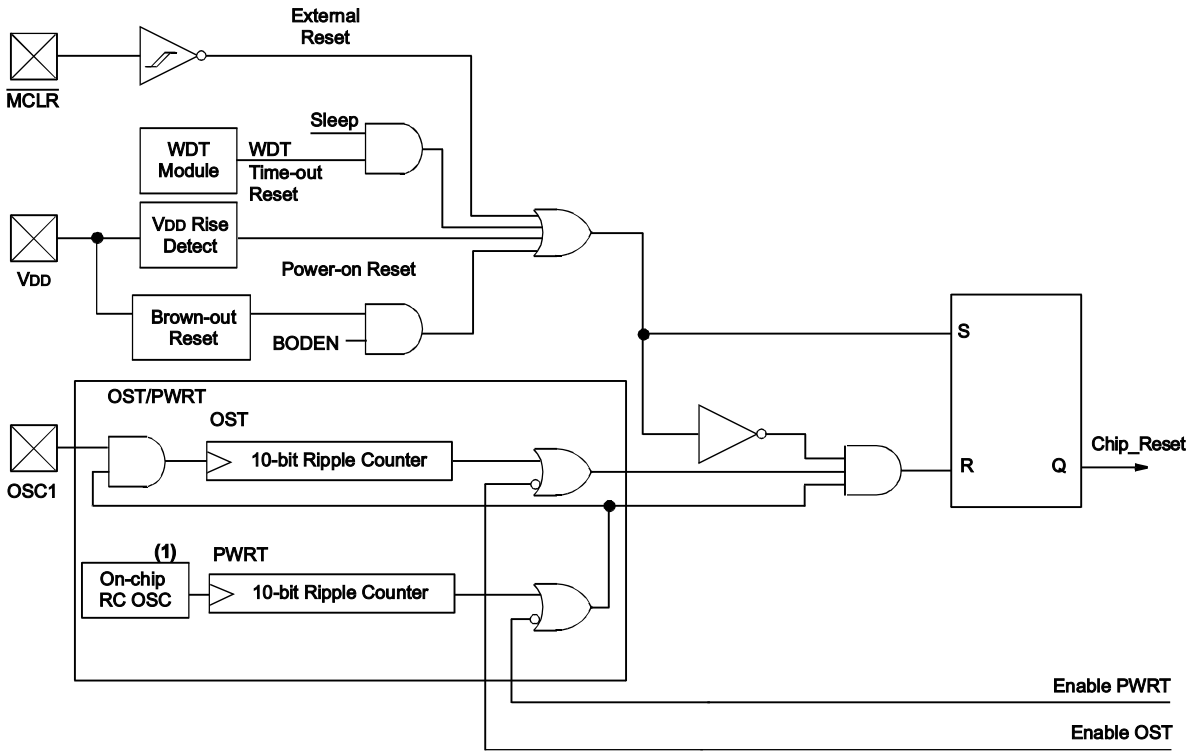
The PIC16F87XA differentiates between various kinds of Reset:

- Power-on Reset (POR)
- $\overline{\text{MCLR}}$ Reset during normal operation
- $\overline{\text{MCLR}}$ Reset during Sleep
- WDT Reset (during normal operation)
- WDT Wake-up (during Sleep)
- Brown-out Reset (BOR)

Some registers are not affected in any Reset condition. Their status is unknown on POR and unchanged in any other Reset. Most other registers are reset to a "Reset state" on Power-on Reset (POR), on the $\overline{\text{MCLR}}$ and WDT Reset, on $\overline{\text{MCLR}}$ Reset during Sleep and Brown-out Reset (BOR). They are not affected by a WDT wake-up which is viewed as the resumption of normal operation. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are set or cleared differently in different Reset situations as indicated in Table 14-4. These bits are used in software to determine the nature of the Reset. See Table 14-6 for a full description of Reset states of all registers.

A simplified block diagram of the on-chip Reset circuit is shown in Figure 14-4.

FIGURE 14-4: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: This is a separate oscillator from the RC oscillator of the CLKI pin.

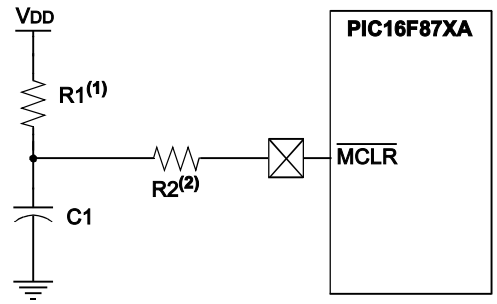
14.4 $\overline{\text{MCLR}}$

PIC16F87XA devices have a noise filter in the $\overline{\text{MCLR}}$ Reset path. The filter will detect and ignore small pulses.

It should be noted that a WDT Reset does not drive $\overline{\text{MCLR}}$ pin low.

The behavior of the ESD protection on the $\overline{\text{MCLR}}$ pin differs from previous devices of this family. Voltages applied to the pin that exceed its specification can result in both Resets and current consumption outside of device specification during the Reset event. For this reason, Microchip recommends that the $\overline{\text{MCLR}}$ pin no longer be tied directly to VDD. The use of an RCR network, as shown in Figure 14-5, is suggested.

FIGURE 14-5: RECOMMENDED $\overline{\text{MCLR}}$ CIRCUIT



Note 1: R1 < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

2: R2 > than 1K will limit any current flowing into $\overline{\text{MCLR}}$ from the external capacitor C, in the event of $\overline{\text{MCLR}}$/VPP breakdown due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

14.5 Power-on Reset (POR)

A Power-on Reset pulse is generated on-chip when VDD rise is detected (in the range of 1.2V-1.7V). To take advantage of the POR, tie the $\overline{\text{MCLR}}$ pin to VDD through an RC network, as described in **Section 14.4 "MCLR"**. A maximum rise time for VDD is specified. See **Section 17.0 "Electrical Characteristics"** for details.

When the device starts normal operation (exits the Reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in Reset until the operating conditions are met. Brown-out Reset may be used to meet the start-up conditions. For additional information, refer to application note, *AN607,* "*Power-up Trouble Shooting*" (DS00607).

14.6 Power-up Timer (PWRT)

The Power-up Timer provides a fixed 72 ms nominal time-out on power-up only from the POR. The Power-up Timer operates on an internal RC oscillator. The chip is kept in Reset as long as the PWRT is active. The PWRT's time delay allows VDD to rise to an acceptable level. A configuration bit is provided to enable or disable the PWRT.

The power-up time delay will vary from chip to chip due to VDD, temperature and process variation. See **Section 17.0 "Electrical Characteristics"** for details (TPWRT, parameter #33).

14.7 Oscillator Start-up Timer (OST)

The Oscillator Start-up Timer (OST) provides a delay of 1024 oscillator cycles (from OSC1 input) after the PWRT delay is over (if PWRT is enabled). This helps to ensure that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP and HS modes and only on Power-on Reset or wake-up from Sleep.

14.8 Brown-out Reset (BOR)

The configuration bit, BODEN, can enable or disable the Brown-out Reset circuit. If VDD falls below VBOR (parameter D005, about 4V) for longer than TBOR (parameter #35, about 100 µS), the brown-out situation will reset the device. If VDD falls below VBOR for less than TBOR, a Reset may not occur.

Once the brown-out occurs, the device will remain in Brown-out Reset until VDD rises above VBOR. The Power-up Timer then keeps the device in Reset for TPWRT (parameter #33, about 72 mS). If VDD should fall below VBOR during TPWRT, the Brown-out Reset process will restart when VDD rises above VBOR with the Power-up Timer Reset. The Power-up Timer is always enabled when the Brown-out Reset circuit is enabled, regardless of the state of the PWRT configuration bit.

14.9 Time-out Sequence

On power-up, the time-out sequence is as follows: the PWRT delay starts (if enabled) when a POR Reset occurs. Then, OST starts counting 1024 oscillator cycles when PWRT ends (LP, XT, HS). When the OST ends, the device comes out of Reset.

If $\overline{\text{MCLR}}$ is kept low long enough, the time-outs will expire. Bringing $\overline{\text{MCLR}}$ high will begin execution immediately. This is useful for testing purposes or to synchronize more than one PIC16F87XA device operating in parallel.

Table 14-5 shows the Reset conditions for the Status, PCON and PC registers, while Table 14-6 shows the Reset conditions for all the registers.

© 2003 Microchip Technology Inc.

14.10 Power Control/Status Register (PCON)

The Power Control/Status Register, PCON, has up to two bits depending upon the device.

Bit 0 is the Brown-out Reset Status bit, $\overline{BOR}$. The $\overline{BOR}$ bit is unknown on a Power-on Reset. It must then be set by the user and checked on subsequent Resets to see if it has been cleared, indicating that a BOR has occurred.

When the Brown-out Reset is disabled, the state of the $\overline{BOR}$ bit is unpredictable and is, therefore, not valid at any time.

Bit 1 is the Power-on Reset Status bit, $\overline{POR}$. It is cleared on a Power-on Reset and unaffected otherwise. The user must set this bit following a Power-on Reset.

TABLE 14-3: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up		Brown-out	Wake-up from Sleep
	$\overline{PWRTE}$ = 0	$\overline{PWRTE}$ = 1		
XT, HS, LP	72 ms + 1024 Tosc	1024 Tosc	72 ms + 1024 Tosc	1024 Tosc
RC	72 ms	—	72 ms	—

TABLE 14-4: STATUS BITS AND THEIR SIGNIFICANCE

$\overline{POR}$	$\overline{BOR}$	$\overline{TO}$	$\overline{PD}$	Condition
0	x	1	1	Power-on Reset
0	x	0	x	Illegal, $\overline{TO}$ is set on $\overline{POR}$
0	x	x	0	Illegal, $\overline{PD}$ is set on $\overline{POR}$
1	0	1	1	Brown-out Reset
1	1	0	1	WDT Reset
1	1	0	0	WDT Wake-up
1	1	u	u	$\overline{MCLR}$ Reset during normal operation
1	1	1	0	$\overline{MCLR}$ Reset during Sleep or Interrupt Wake-up from Sleep

Legend: x = don't care, u = unchanged

TABLE 14-5: RESET CONDITIONS FOR SPECIAL REGISTERS

Condition	Program Counter	Status Register	PCON Register
Power-on Reset	000h	0001 1xxx	---- --0x
$\overline{MCLR}$ Reset during normal operation	000h	000u uuuu	---- --uu
$\overline{MCLR}$ Reset during Sleep	000h	0001 0uuu	---- --uu
WDT Reset	000h	0000 1uuu	---- --uu
WDT Wake-up	PC + 1	uuu0 0uuu	---- --uu
Brown-out Reset	000h	0001 1uuu	---- --u0
Interrupt Wake-up from Sleep	PC + 1(1)	uuu1 0uuu	---- --uu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0'

Note 1: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

TABLE 14-6: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Devices				Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset	Wake-up via WDT or Interrupt
W	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	73A	74A	76A	77A	N/A	N/A	N/A
TMR0	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	73A	74A	76A	77A	0000 0000	0000 0000	PC + 1(2)
STATUS	73A	74A	76A	77A	0001 1xxx	000q quuu(3)	uuuq quuu(3)
FSR	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	73A	74A	76A	77A	--0x 0000	--0u 0000	--uu uuuu
PORTB	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTD	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	73A	74A	76A	77A	---- -xxx	---- -uuu	---- -uuu
PCLATH	73A	74A	76A	77A	---0 0000	---0 0000	---u uuuu
INTCON	73A	74A	76A	77A	0000 000x	0000 000u	uuuu uuuu(1)
PIR1	73A	74A	76A	77A	r000 0000	r000 0000	ruuu uuuu(1)
	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu(1)
PIR2	73A	74A	76A	77A	-0-0 0--0	-0-0 0--0	-u-u u--u(1)
TMR1L	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1H	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	73A	74A	76A	77A	--00 0000	--uu uuuu	--uu uuuu
TMR2	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
T2CON	73A	74A	76A	77A	-000 0000	-000 0000	-uuu uuuu
SSPBUF	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPCON	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
CCPR1L	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1H	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	73A	74A	76A	77A	--00 0000	--00 0000	--uu uuuu
RCSTA	73A	74A	76A	77A	0000 000x	0000 000x	uuuu uuuu
TXREG	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
RCREG	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
CCPR2L	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2H	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
ADRESH	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	73A	74A	76A	77A	0000 00-0	0000 00-0	uuuu uu-u
OPTION_REG	73A	74A	76A	77A	1111 1111	1111 1111	uuuu uuuu
TRISA	73A	74A	76A	77A	--11 1111	--11 1111	--uu uuuu
TRISB	73A	74A	76A	77A	1111 1111	1111 1111	uuuu uuuu
TRISC	73A	74A	76A	77A	1111 1111	1111 1111	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition, r = reserved, maintain clear. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in INTCON, PIR1 and/or PIR2 will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

3: See Table 14-5 for Reset value for specific condition.

© 2003 Microchip Technology Inc.

TABLE 14-6: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Devices				Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets, WDT Reset	Wake-up via WDT or Interrupt
TRISD	73A	74A	76A	77A	1111 1111	1111 1111	uuuu uuuu
TRISE	73A	74A	76A	77A	0000 -111	0000 -111	uuuu -uuu
PIE1	73A	74A	76A	77A	r000 0000	r000 0000	ruuu uuuu
	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
PIE2	73A	74A	76A	77A	-0-0 0--0	-0-0 0--0	-u-u u--u
PCON	73A	74A	76A	77A	---- --qq	---- --uu	---- --uu
SSPCON2	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
PR2	73A	74A	76A	77A	1111 1111	1111 1111	1111 1111
SSPADD	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	73A	74A	76A	77A	--00 0000	--00 0000	--uu uuuu
TXSTA	73A	74A	76A	77A	0000 -010	0000 -010	uuuu -uuu
SPBRG	73A	74A	76A	77A	0000 0000	0000 0000	uuuu uuuu
CMCON	73A	974	76A	77A	0000 0111	0000 0111	uuuu uuuu
CVRCON	73A	74A	76A	77A	000- 0000	000- 0000	uuu- uuuu
ADRESL	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON1	73A	74A	76A	77A	00-- 0000	00-- 0000	uu-- uuuu
EEDATA	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEADR	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATH	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEADRH	73A	74A	76A	77A	xxxx xxxx	uuuu uuuu	uuuu uuuu
EECON1	73A	74A	76A	77A	x--- x000	u--- u000	u--- uuuu
EECON2	73A	74A	76A	77A	---- ----	---- ----	---- ----

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition, r = reserved, maintain clear. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in INTCON, PIR1 and/or PIR2 will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

3: See Table 14-5 for Reset value for specific condition.

FIGURE 14-6: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ TIED TO V_{DD} VIA RC NETWORK)

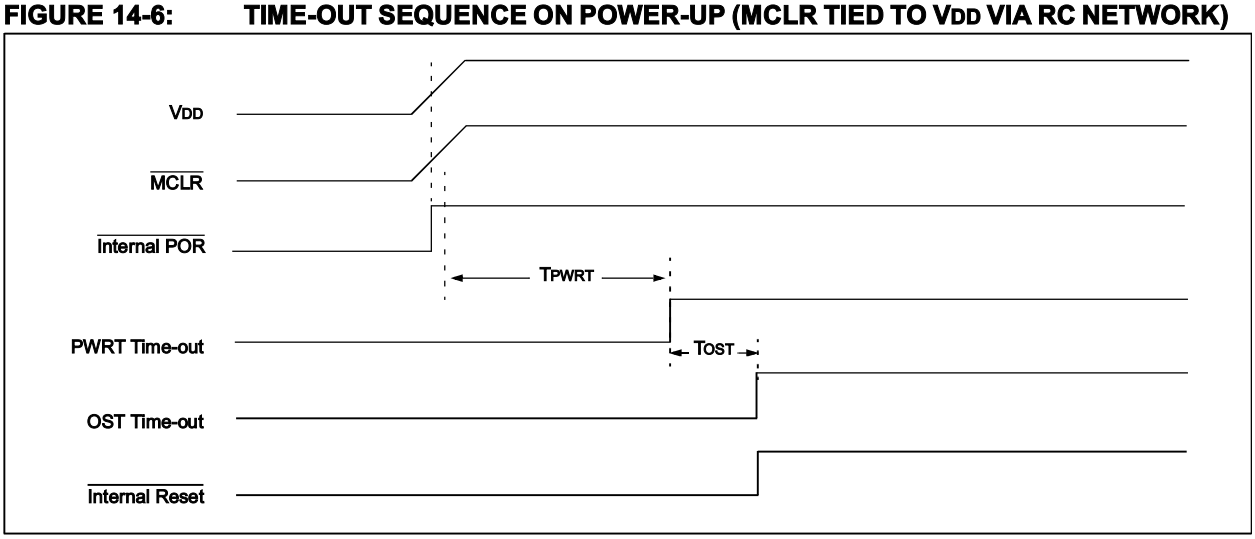


FIGURE 14-7: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO VDD): CASE 1

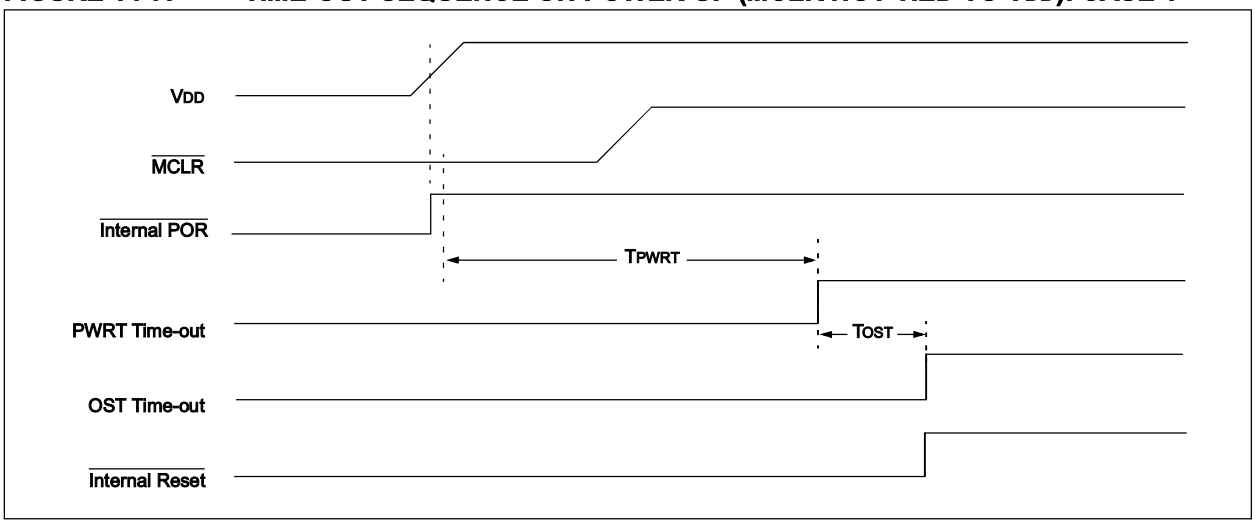


FIGURE 14-8: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO VDD): CASE 2

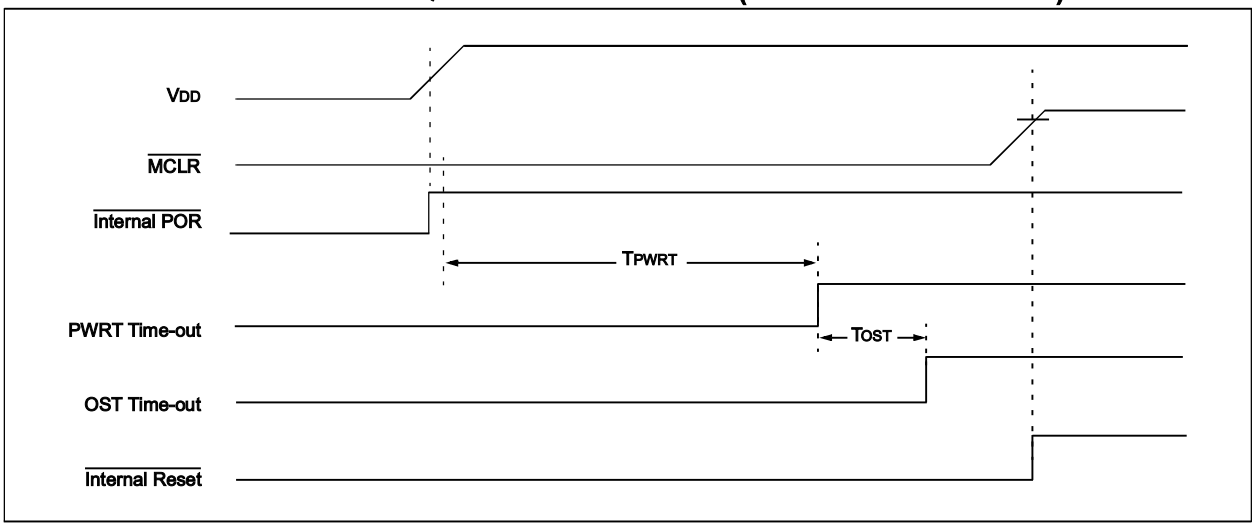
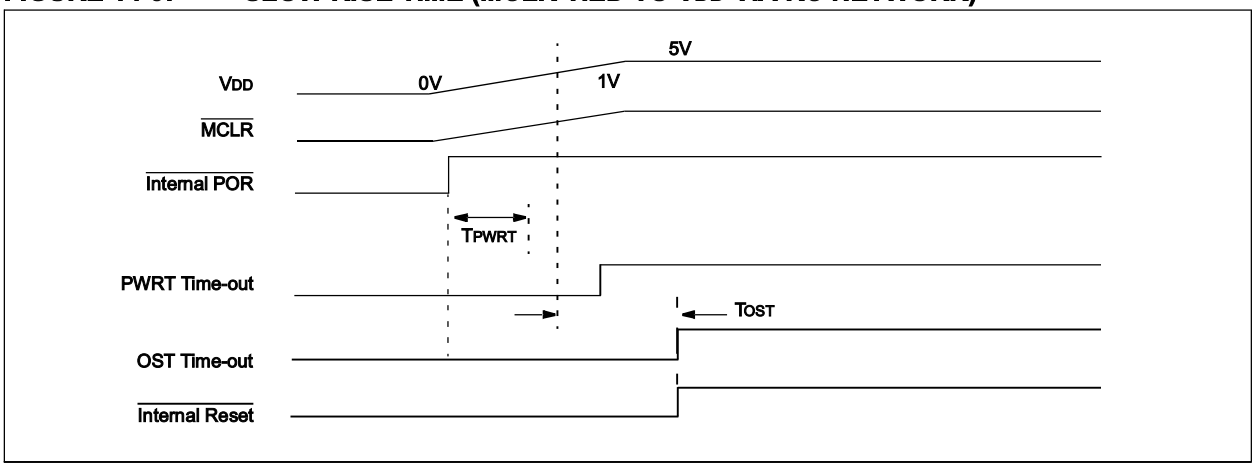


FIGURE 14-9: SLOW RISE TIME ($\overline{MCLR}$ TIED TO VDD VIA RC NETWORK)



© 2003 Microchip Technology Inc.

14.11 Interrupts

The PIC16F87XA family has up to 15 sources of interrupt. The Interrupt Control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits.

Note:	Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

A global interrupt enable bit, GIE (INTCON<7>), enables (if set) all unmasked interrupts or disables (if cleared) all interrupts. When bit GIE is enabled and an interrupt's flag bit and mask bit are set, the interrupt will vector immediately. Individual interrupts can be disabled through their corresponding enable bits in various registers. Individual interrupt bits are set regardless of the status of the GIE bit. The GIE bit is cleared on Reset.

The "return from interrupt" instruction, `RETFIE`, exits the interrupt routine, as well as sets the GIE bit, which re-enables interrupts.

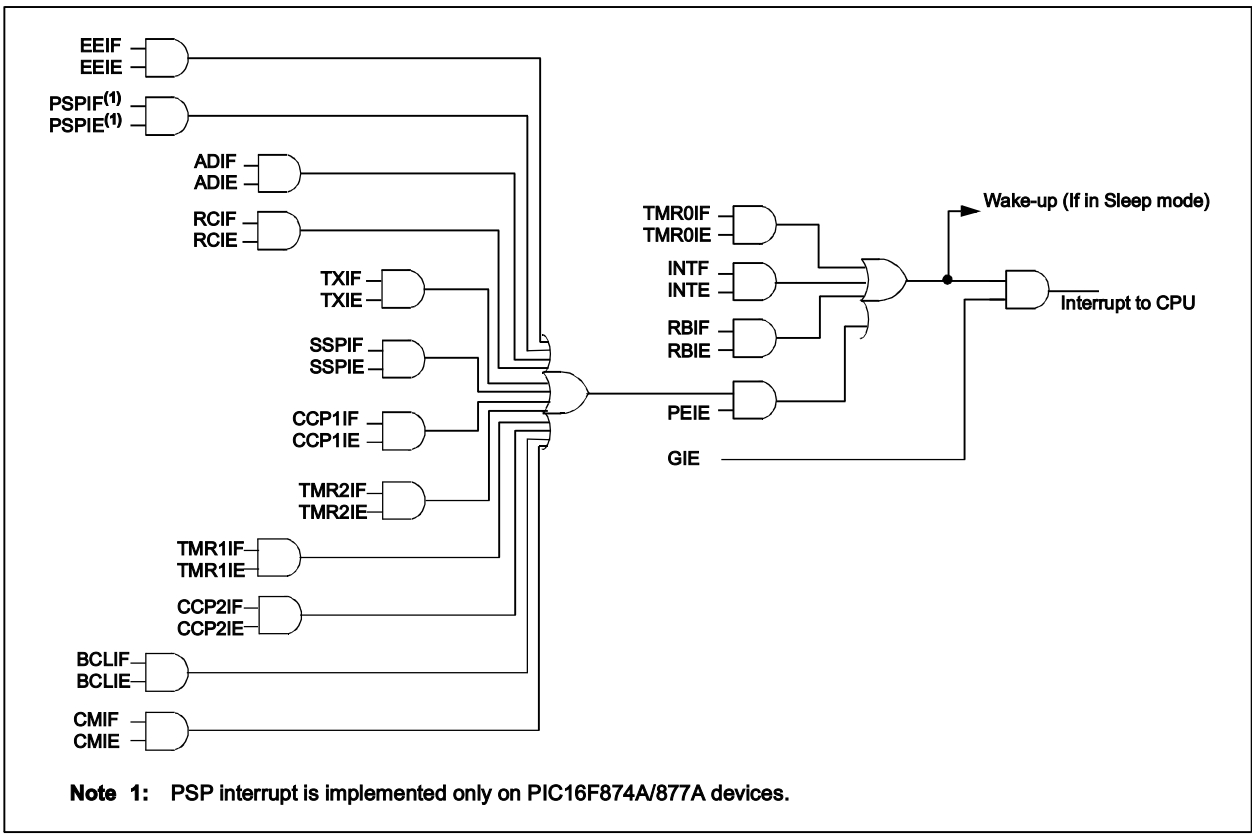
The RB0/INT pin interrupt, the RB port change interrupt and the TMR0 overflow interrupt flags are contained in the INTCON register.

The peripheral interrupt flags are contained in the Special Function Registers, PIR1 and PIR2. The corresponding interrupt enable bits are contained in Special Function Registers, PIE1 and PIE2, and the peripheral interrupt enable bit is contained in Special Function Register, INTCON.

When an interrupt is responded to, the GIE bit is cleared to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with 0004h. Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends when the interrupt event occurs. The latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set regardless of the status of their corresponding mask bit, PEIE bit or GIE bit.

FIGURE 14-10: INTERRUPT LOGIC



Note 1: PSP interrupt is implemented only on PIC16F874A/877A devices.

14.11.1 INT INTERRUPT

External interrupt on the RB0/INT pin is edge triggered, either rising if bit INTEDG (OPTION_REG<6>) is set or falling if the INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, flag bit, INTF (INTCON<1>), is set. This interrupt can be disabled by clearing enable bit, INTE (INTCON<4>). Flag bit INTF must be cleared in software in the Interrupt Service Routine before re-enabling this interrupt. The INT interrupt can wake-up the processor from Sleep if bit INTE was set prior to going into Sleep. The status of global interrupt enable bit, GIE, decides whether or not the processor branches to the interrupt vector following wake-up. See **Section 14.14 "Power-down Mode (Sleep)"** for details on Sleep mode.

14.11.2 TMR0 INTERRUPT

An overflow (FFh → 00h) in the TMR0 register will set flag bit, TMR0IF (INTCON<2>). The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). See **Section 5.0 "Timer0 Module"**.

14.11.3 PORTB INTCON CHANGE

An input change on PORTB<7:4> sets flag bit, RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<4>). See **Section 4.2 "PORTB and the TRISB Register"**.

14.12 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt (i.e., W register and Status register). This will have to be implemented in software.

For the PIC16F873A/874A devices, the register W_TEMP must be defined in both Banks 0 and 1 and must be defined at the same offset from the bank base address (i.e., If W_TEMP is defined at 0x20 in Bank 0, it must also be defined at 0xA0 in Bank 1). The registers, PCLATH_TEMP and STATUS_TEMP, are only defined in Bank 0.

Since the upper 16 bytes of each bank are common in the PIC16F876A/877A devices, temporary holding registers, W_TEMP, STATUS_TEMP and PCLATH_TEMP, should be placed in here. These 16 locations don't require banking and therefore, make it easier for context save and restore. The same code shown in Example 14-1 can be used.

EXAMPLE 14-1: SAVING STATUS, W AND PCLATH REGISTERS IN RAM

```
    MOVWF    W_TEMP           ;Copy W to TEMP register
    SWAPF    STATUS,W         ;Swap status to be saved into W
    CLRF     STATUS           ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
    MOVWF    STATUS_TEMP      ;Save status to bank zero STATUS_TEMP register
    MOVF     PCLATH, W        ;Only required if using pages 1, 2 and/or 3
    MOVWF    PCLATH_TEMP      ;Save PCLATH into W
    CLRF     PCLATH           ;Page zero, regardless of current page
    :
    :(ISR)                    ;(Insert user code here)
    :
    MOVF     PCLATH_TEMP, W   ;Restore PCLATH
    MOVWF    PCLATH           ;Move W into PCLATH
    SWAPF    STATUS_TEMP,W    ;Swap STATUS_TEMP register into W
                              ;(sets bank to original state)
    MOVWF    STATUS           ;Move W into STATUS register
    SWAPF    W_TEMP,F         ;Swap W_TEMP
    SWAPF    W_TEMP,W         ;Swap W_TEMP into W
```


© 2003 Microchip Technology Inc.

14.13 Watchdog Timer (WDT)

The Watchdog Timer is a free running, on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKI pin. That means that the WDT will run even if the clock on the OSC1/CLKI and OSC2/CLKO pins of the device has been stopped, for example, by execution of a SLEEP instruction.

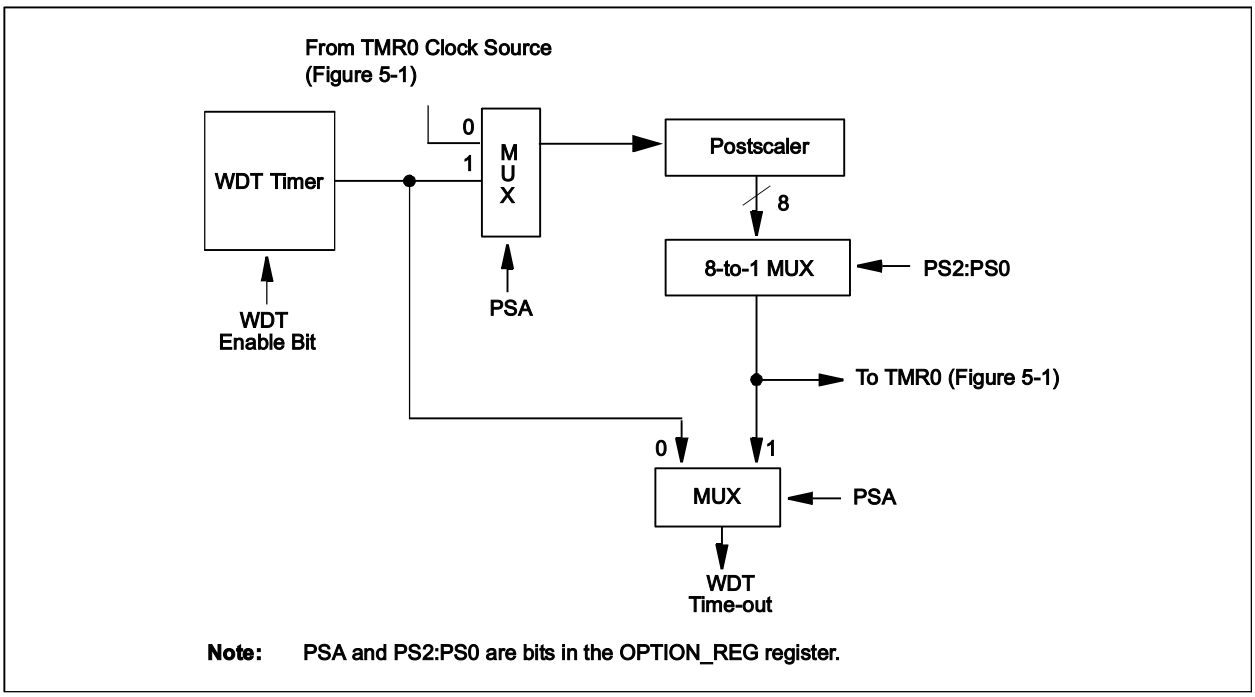
During normal operation, a WDT time-out generates a device Reset (Watchdog Timer Reset). If the device is in Sleep mode, a WDT time-out causes the device to wake-up and continue with normal operation (Watchdog Timer Wake-up). The $\overline{TO}$ bit in the Status register will be cleared upon a Watchdog Timer time-out.

The WDT can be permanently disabled by clearing configuration bit, WDTE (**Section 14.1 "Configuration Bits"**).

WDT time-out period values may be found in **Section 17.0 "Electrical Characteristics"** under parameter #31. Values for the WDT prescaler (actually a postscaler but shared with the Timer0 prescaler) may be assigned using the OPTION_REG register.

Note 1:	The CLRWDT and SLEEP instructions clear the WDT and the postscaler, if assigned to the WDT and prevent it from timing out and generating a device Reset condition.
2:	When a CLRWDT instruction is executed and the prescaler is assigned to the WDT, the prescaler count will be cleared but the prescaler assignment is not changed.

FIGURE 14-11: WATCHDOG TIMER BLOCK DIAGRAM



Note: PSA and PS2:PS0 are bits in the OPTION_REG register.

TABLE 14-7: SUMMARY OF WATCHDOG TIMER REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
2007h	Config. bits	(1)	BODEN(1)	CP1	CP0	$\overline{PWRTE}$(1)	WDTE	FOSC1	FOSC0
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Legend: Shaded cells are not used by the Watchdog Timer.

Note 1: See Register 14-1 for operation of these bits.

14.14 Power-down Mode (Sleep)

Power-down mode is entered by executing a SLEEP instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the $\overline{PD}$ bit (Status<3>) is cleared, the $\overline{TO}$ (Status<4>) bit is set and the oscillator driver is turned off. The I/O ports maintain the status they had before the SLEEP instruction was executed (driving high, low or high-impedance).

For lowest current consumption in this mode, place all I/O pins at either VDD or VSS, ensure no external circuitry is drawing current from the I/O pin, power-down the A/D and disable external clocks. Pull all I/O pins that are high-impedance inputs, high or low externally, to avoid switching currents caused by floating inputs. The T0CKI input should also be at VDD or VSS for lowest current consumption. The contribution from on-chip pull-ups on PORTB should also be considered.

The $\overline{MCLR}$ pin must be at a logic high level (VIHMC).

14.14.1 WAKE-UP FROM SLEEP

The device can wake-up from Sleep through one of the following events:

1. External Reset input on $\overline{MCLR}$ pin.
2. Watchdog Timer wake-up (if WDT was enabled).
3. Interrupt from INT pin, RB port change or peripheral interrupt.

External $\overline{MCLR}$ Reset will cause a device Reset. All other events are considered a continuation of program execution and cause a "wake-up". The $\overline{TO}$ and $\overline{PD}$ bits in the Status register can be used to determine the cause of device Reset. The $\overline{PD}$ bit, which is set on power-up, is cleared when Sleep is invoked. The $\overline{TO}$ bit is cleared if a WDT time-out occurred and caused wake-up.

The following peripheral interrupts can wake the device from Sleep:

1. PSP read or write (PIC16F874/877 only).
2. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
3. CCP Capture mode interrupt.
4. Special event trigger (Timer1 in Asynchronous mode using an external clock).
5. SSP (Start/Stop) bit detect interrupt.
6. SSP transmit or receive in Slave mode (SPI/I^2C).
7. USART RX or TX (Synchronous Slave mode).
8. A/D conversion (when A/D clock source is RC).
9. EEPROM write operation completion.
10. Comparator output changes state.

Other peripherals cannot generate interrupts since during Sleep, no on-chip clocks are present.

When the SLEEP instruction is being executed, the next instruction (PC + 1) is prefetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is set (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt address (0004h). In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

14.14.2 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

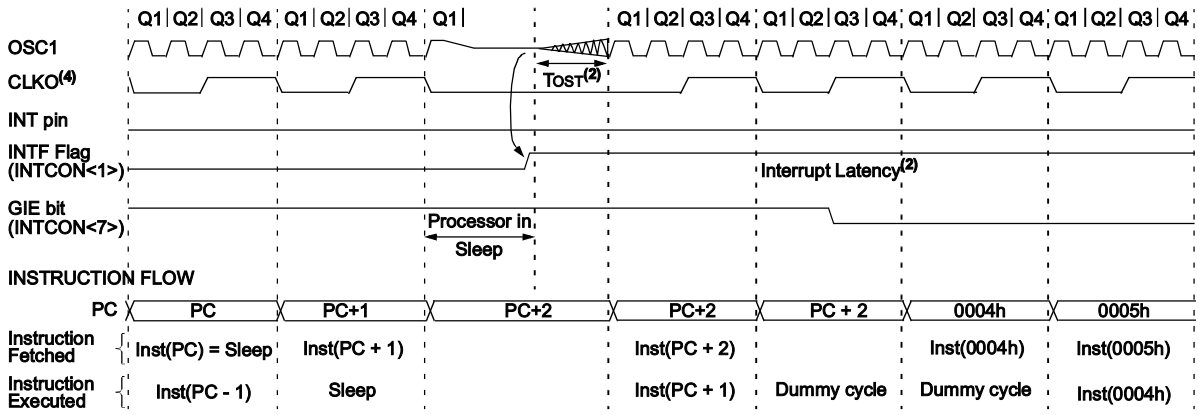
- If the interrupt occurs **before** the execution of a SLEEP instruction, the SLEEP instruction will complete as a NOP. Therefore, the WDT and WDT postscaler will not be cleared, the $\overline{TO}$ bit will not be set and $\overline{PD}$ bits will not be cleared.
- If the interrupt occurs **during or after** the execution of a SLEEP instruction, the device will immediately wake-up from Sleep. The SLEEP instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the $\overline{TO}$ bit will be set and the $\overline{PD}$ bit will be cleared.

Even if the flag bits were checked before executing a SLEEP instruction, it may be possible for flag bits to become set before the SLEEP instruction completes. To determine whether a SLEEP instruction executed, test the $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the SLEEP instruction was executed as a NOP.

To ensure that the WDT is cleared, a CLRWDT instruction should be executed before a SLEEP instruction.

© 2003 Microchip Technology Inc.

FIGURE 14-12: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: XT, HS or LP Oscillator mode assumed.
2: TOST = 1024 TOSC (drawing not to scale). This delay will not be there for RC Oscillator mode.
3: GIE = 1 assumed. In this case, after wake- up, the processor jumps to the interrupt routine. If GIE = 0, execution will continue in-line.
4: CLKO is not available in these oscillator modes but shown here for timing reference.

14.15 In-Circuit Debugger

When the DEBUG bit in the configuration word is programmed to a '0', the in-circuit debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® ICD. When the microcontroller has this feature enabled, some of the resources are not available for general use. Table 14-8 shows which features are consumed by the background debugger.

TABLE 14-8: DEBUGGER RESOURCES

I/O pins	RB6, RB7
Stack	1 level
Program Memory	Address 0000h must be NOP
	Last 100h words
Data Memory	0x070 (0x0F0, 0x170, 0x1F0) 0x1EB-0x1EF

To use the in-circuit debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to $\overline{MCLR}$/VPP, VDD, GND, RB7 and RB6. This will interface to the in-circuit debugger module available from Microchip or one of the third party development tool companies.

14.16 Program Verification/Code Protection

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

14.17 ID Locations

Four memory locations (2000h-2003h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are not accessible during normal execution but are readable and writable during program/verify. It is recommended that only the 4 Least Significant bits of the ID location are used.

14.18 In-Circuit Serial Programming

PIC16F87XA microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

When using ICSP, the part must be supplied at 4.5V to 5.5V if a bulk erase will be executed. This includes reprogramming of the code-protect, both from an on state to an off state. For all other cases of ICSP, the part may be programmed at the normal operating voltages. This means calibration values, unique user IDs or user code can be reprogrammed or added.

For complete details of serial programming, please refer to the *"PIC16F87XA Flash Memory Programming Specification"* (DS39589).

14.19 Low-Voltage (Single-Supply) ICSP Programming

The LVP bit of the configuration word enables low-voltage ICSP programming. This mode allows the microcontroller to be programmed via ICSP using a VDD source in the operating voltage range. This only means that VPP does not have to be brought to VIHH but can instead be left at the normal operating voltage. In this mode, the RB3/PGM pin is dedicated to the programming function and ceases to be a general purpose I/O pin. During programming, VDD is applied to the MCLR pin. To enter Programming mode, VDD must be applied to the RB3/PGM provided the LVP bit is set. The LVP bit defaults to on ('1') from the factory.

Note 1: The High-Voltage Programming mode is always available, regardless of the state of the LVP bit, by applying VIHH to the MCLR pin.

2: While in Low-Voltage ICSP mode, the RB3 pin can no longer be used as a general purpose I/O pin.

3: When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device.

4: RB3 should not be allowed to float if LVP is enabled. An external pull-down device should be used to default the device to normal operating mode. If RB3 floats high, the PIC16F87XA device will enter Programming mode.

5: LVP mode is enabled by default on all devices shipped from Microchip. It can be disabled by clearing the LVP bit in the CONFIG register.

6: Disabling LVP will provide maximum compatibility to other PIC16CXXX devices.

If Low-Voltage Programming mode is not used, the LVP bit can be programmed to a '0' and RB3/PGM becomes a digital I/O pin. However, the LVP bit may only be programmed when programming is entered with VIHH on MCLR. The LVP bit can only be charged when using high voltage on MCLR.

It should be noted, that once the LVP bit is programmed to '0', only the High-Voltage Programming mode is available and only High-Voltage Programming mode can be used to program the device.

When using low-voltage ICSP, the part must be supplied at 4.5V to 5.5V if a bulk erase will be executed. This includes reprogramming of the code-protect bits from an on state to an off state. For all other cases of low-voltage ICSP, the part may be programmed at the normal operating voltage. This means calibration values, unique user IDs or user code can be reprogrammed or added.

© 2003 Microchip Technology Inc.

15.0 INSTRUCTION SET SUMMARY

The PIC16 instruction set is highly orthogonal and is comprised of three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

Each PIC16 instruction is a 14-bit word divided into an **opcode** which specifies the instruction type and one or more **operands** which further specify the operation of the instruction. The formats for each of the categories is presented in Figure 15-1, while the various opcode fields are summarized in Table 15-1.

Table 15-2 lists the instructions recognized by the MPASM™ Assembler. A complete description of each instruction is also available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the bit affected by the operation, while 'f' represents the address of the file in which the bit is located.

For **literal and control** operations, 'k' represents an eight or eleven-bit constant or literal value

One instruction cycle consists of four oscillator periods; for an oscillator frequency of 4 MHz, this gives a normal instruction execution time of 1 µs. All instructions are executed within a single instruction cycle, unless a conditional test is true, or the program counter is changed as a result of an instruction. When this occurs, the execution takes two instruction cycles with the second cycle executed as a NOP.

Note:	To maintain upward compatibility with future PIC16F87XA products, do not use the OPTION and TRIS instructions.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

15.1 READ-MODIFY-WRITE OPERATIONS

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

For example, a "CLRF PORTB" instruction will read PORTB, clear all the data bits, then write the result back to PORTB. This example would have the unintended result that the condition that sets the RBIF flag would be cleared.

TABLE 15-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

FIGURE 15-1: GENERAL FORMAT FOR INSTRUCTIONS

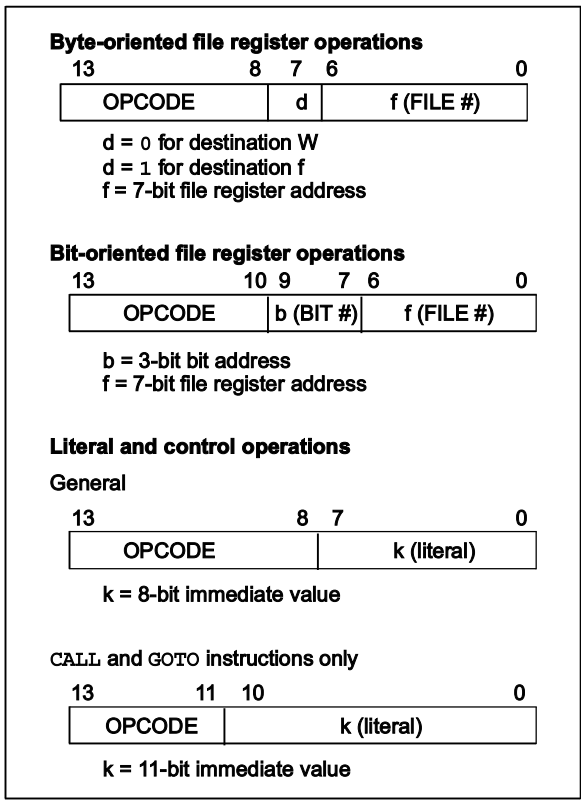


TABLE 15-2: PIC16F87XA INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add Literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND Literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to Address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR Literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move Literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from Interrupt	2	00	0000	0000	1001		
RETLW	k	Return with Literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from Literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR Literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

© 2003 Microchip Technology Inc.

15.2 Instruction Descriptions

ADDLW	Add Literal and W
Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight-bit literal 'k' and the result is placed in the W register.

ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

ANDLW	AND Literal with W
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.

ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2 TCY instruction.

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b' in register 'f' is '0', the next instruction is discarded and a NOP is executed instead, making this a 2 TCY instruction.

CALL	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	($\bar{f}$) → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

© 2003 Microchip Technology Inc.

DECFSZ **Decrement f, Skip if 0**

Syntax: [*label*] DECFSZ f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) - 1 → (destination);
skip if result = 0

Status Affected: None

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.
If the result is '1', the next instruction is executed. If the result is '0', then a NOP is executed instead, making it a 2 TCY instruction.

GOTO **Unconditional Branch**

Syntax: [*label*] GOTO k

Operands: 0 ≤ k ≤ 2047

Operation: k → PC<10:0>
PCLATH<4:3> → PC<12:11>

Status Affected: None

Description: GOTO is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction.

INCF **Increment f**

Syntax: [*label*] INCF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) + 1 → (destination)

Status Affected: Z

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

INCFSZ **Increment f, Skip if 0**

Syntax: [*label*] INCFSZ f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) + 1 → (destination),
skip if result = 0

Status Affected: None

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.
If the result is '1', the next instruction is executed. If the result is '0', a NOP is executed instead, making it a 2 TCY instruction.

IORLW **Inclusive OR Literal with W**

Syntax: [*label*] IORLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) .OR. k → (W)

Status Affected: Z

Description: The contents of the W register are OR'ed with the eight-bit literal 'k'. The result is placed in the W register.

IORWF **Inclusive OR W with f**

Syntax: [*label*] IORWF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (W) .OR. (f) → (destination)

Status Affected: Z

Description: Inclusive OR the W register with register 'f'. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

RLF **Rotate Left f through Carry**

Syntax: [*label*] RLF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is stored back in register 'f'.



RETURN **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

RRF **Rotate Right f through Carry**

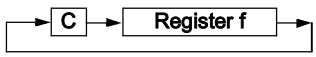
Syntax: [*label*] RRF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.



SLEEP

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Description: The power-down status bit, $\overline{PD}$, is cleared. Time-out status bit, $\overline{TO}$, is set. Watchdog Timer and its prescaler are cleared.
The processor is put into Sleep mode with the oscillator stopped.

SUBLW **Subtract W from Literal**

Syntax: [*label*] SUBLW k

Operands: 0 ≤ k ≤ 255

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Description: The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register.

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) - (W) → (destination)

Status Affected: C, DC, Z

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

© 2003 Microchip Technology Inc.

SWAPF	Swap Nibbles in f
Syntax:	[*label*] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed in register 'f'.

XORLW	Exclusive OR Literal with W
Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR'ed with the eight-bit literal 'k'. The result is placed in the W register.

XORWF	Exclusive OR W with f
Syntax:	[*label*] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

NOTES:

© 2003 Microchip Technology Inc.

16.0 DEVELOPMENT SUPPORT

The PICmicro® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB C30 C Compiler
  - MPLAB ASM30 Assembler/Linker/Library
- Simulators
  - MPLAB SIM Software Simulator
  - MPLAB dsPIC30 Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB ICE 4000 In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM.net™ Demonstration Board
  - PICDEM 2 Plus Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 4 Demonstration Board
  - PICDEM 17 Demonstration Board
  - PICDEM 18R Demonstration Board
  - PICDEM LIN Demonstration Board
  - PICDEM USB Demonstration Board
- Evaluation Kits
  - KEELOQ®
  - PICDEM MSC
  - microID®
  - CAN
  - PowerSmart®
  - Analog

16.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16-bit microcontroller market. The MPLAB IDE is a Windows® based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor with color coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High level source code debugging
- Mouse over variable inspection
- Extensive on-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or C)
- One touch assemble (or compile) and download to PICmicro emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files (assembly or C)
  - absolute listing file (mixed assembly and C)
  - machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost effective simulators, through low cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increasing flexibility and power.

16.2 MPASM Assembler

The MPASM assembler is a full-featured, universal macro assembler for all PICmicro MCUs.

The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects
- User defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

16.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI C compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

16.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB object librarian manages the creation and modification of library files of pre-compiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

16.5 MPLAB C30 C Compiler

The MPLAB C30 C compiler is a full-featured, ANSI compliant, optimizing compiler that translates standard ANSI C programs into dsPIC30F assembly language source. The compiler also supports many command-line options and language extensions to take full advantage of the dsPIC30F device hardware capabilities, and afford fine control of the compiler code generator.

MPLAB C30 is distributed with a complete ANSI C standard library. All library functions have been validated and conform to the ANSI C library standard. The library includes functions for string manipulation, dynamic memory allocation, data conversion, time-keeping, and math functions (trigonometric, exponential and hyperbolic). The compiler provides symbolic information for high level source debugging with the MPLAB IDE.

16.6 MPLAB ASM30 Assembler, Linker, and Librarian

MPLAB ASM30 assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 compiler uses the assembler to produce it's object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

16.7 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC hosted environment by simulating the PICmicro series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any pin. The execution can be performed in Single-Step, Execute Until Break, or Trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and MPLAB C18 C Compilers, as well as the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent, economical software development tool.

16.8 MPLAB SIM30 Software Simulator

The MPLAB SIM30 software simulator allows code development in a PC hosted environment by simulating the dsPIC30F series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any of the pins.

The MPLAB SIM30 simulator fully supports symbolic debugging using the MPLAB C30 C Compiler and MPLAB ASM30 assembler. The simulator runs in either a Command Line mode for automated tasks, or from MPLAB IDE. This high speed simulator is designed to debug, analyze and optimize time intensive DSP routines.

© 2003 Microchip Technology Inc.

16.9 MPLAB ICE 2000 High Performance Universal In-Circuit Emulator

The MPLAB ICE 2000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PICmicro microcontrollers. Software control of the MPLAB ICE 2000 in-circuit emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PICmicro microcontrollers.

The MPLAB ICE 2000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

16.10 MPLAB ICE 4000 High Performance Universal In-Circuit Emulator

The MPLAB ICE 4000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for high-end PICmicro microcontrollers. Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICD 4000 is a premium emulator system, providing the features of MPLAB ICE 2000, but with increased emulation memory and high speed performance for dsPIC30F and PIC18XXXX devices. Its advanced emulator features include complex triggering and timing, up to 2 Mb of emulation memory, and the ability to view variables in real-time.

The MPLAB ICE 4000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

16.11 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PICmicro MCUs and can be used to develop for these and other PICmicro microcontrollers. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip's In-Circuit Serial Programming™ (ICSP™) protocol, offers cost effective in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single-stepping and watching variables, CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real-time. MPLAB ICD 2 also serves as a development programmer for selected PICmicro devices.

16.12 PRO MATE II Universal Device Programmer

The PRO MATE II is a universal, CE compliant device programmer with programmable voltage verification at V_{DDMIN} and V_{DDMAX} for maximum reliability. It features an LCD display for instructions and error messages and a modular detachable socket assembly to support various package types. In Stand-Alone mode, the PRO MATE II device programmer can read, verify, and program PICmicro devices without a PC connection. It can also set code protection in this mode.

16.13 PICSTART Plus Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low-cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus development programmer supports most PICmicro devices up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

16.14 PICDEM 1 PICmicro Demonstration Board

The PICDEM 1 demonstration board demonstrates the capabilities of the PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The sample microcontrollers provided with the PICDEM 1 demonstration board can be programmed with a PRO MATE II device programmer, or a PICSTART Plus development programmer. The PICDEM 1 demonstration board can be connected to the MPLAB ICE in-circuit emulator for testing. A prototype area extends the circuitry for additional application components. Features include an RS-232 interface, a potentiometer for simulated analog input, push button switches and eight LEDs.

16.15 PICDEM.net Internet/Ethernet Demonstration Board

The PICDEM.net demonstration board is an Internet/ Ethernet demonstration board using the PIC18F452 microcontroller and TCP/IP firmware. The board supports any 40-pin DIP device that conforms to the standard pinout used by the PIC16F877 or PIC18C452. This kit features a user friendly TCP/IP stack, web server with HTML, a 24L256 Serial EEPROM for Xmodem download to web pages into Serial EEPROM, ICSP/MPLAB ICD 2 interface connector, an Ethernet interface, RS-232 interface, and a 16 x 2 LCD display. Also included is the book and CD-ROM *"TCP/IP Lean, Web Servers for Embedded Systems,"* by Jeremy Bentham.

16.16 PICDEM 2 Plus Demonstration Board

The PICDEM 2 Plus demonstration board supports many 18-, 28-, and 40-pin microcontrollers, including PIC16F87X and PIC18FXX2 devices. All the necessary hardware and software is included to run the demonstration programs. The sample microcontrollers provided with the PICDEM 2 demonstration board can be programmed with a PRO MATE II device programmer, PICSTART Plus development programmer, or MPLAB ICD 2 with a Universal Programmer Adapter. The MPLAB ICD 2 and MPLAB ICE in-circuit emulators may also be used with the PICDEM 2 demonstration board to test firmware. A prototype area extends the circuitry for additional application components. Some of the features include an RS-232 interface, a 2 x 16 LCD display, a piezo speaker, an on-board temperature sensor, four LEDs, and sample PIC18F452 and PIC16F877 Flash microcontrollers.

16.17 PICDEM 3 PIC16C92X Demonstration Board

The PICDEM 3 demonstration board supports the PIC16C923 and PIC16C924 in the PLCC package. All the necessary hardware and software is included to run the demonstration programs.

16.18 PICDEM 4 8/14/18-Pin Demonstration Board

The PICDEM 4 can be used to demonstrate the capabilities of the 8, 14, and 18-pin PIC16XXXX and PIC18XXXX MCUs, including the PIC16F818/819, PIC16F87/88, PIC16F62XA and the PIC18F1320 family of microcontrollers. PICDEM 4 is intended to showcase the many features of these low pin count parts, including LIN and Motor Control using ECCP. Special provisions are made for low power operation with the supercapacitor circuit, and jumpers allow on-board hardware to be disabled to eliminate current draw in this mode. Included on the demo board are provisions for Crystal, RC or Canned Oscillator modes, a five volt regulator for use with a nine volt wall adapter or battery, DB-9 RS-232 interface, ICD connector for programming via ICSP and development with MPLAB ICD 2, 2x16 liquid crystal display, PCB footprints for H-Bridge motor driver, LIN transceiver and EEPROM. Also included are: header for expansion, eight LEDs, four potentiometers, three push buttons and a prototyping area. Included with the kit is a PIC16F627A and a PIC18F1320. Tutorial firmware is included along with the User's Guide.

16.19 PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756A, PIC17C762 and PIC17C766. A programmed sample is included. The PRO MATE II device programmer, or the PICSTART Plus development programmer, can be used to reprogram the device for user tailored application development. The PICDEM 17 demonstration board supports program download and execution from external on-board Flash memory. A generous prototype area is available for user hardware expansion.

© 2003 Microchip Technology Inc.

16.20 PICDEM 18R PIC18C601/801 Demonstration Board

The PICDEM 18R demonstration board serves to assist development of the PIC18C601/801 family of Microchip microcontrollers. It provides hardware implementation of both 8-bit Multiplexed/Demultiplexed and 16-bit Memory modes. The board includes 2 Mb external Flash memory and 128 Kb SRAM memory, as well as serial EEPROM, allowing access to the wide range of memory types supported by the PIC18C601/801.

16.21 PICDEM LIN PIC16C43X Demonstration Board

The powerful LIN hardware and software kit includes a series of boards and three PICmicro microcontrollers. The small footprint PIC16C432 and PIC16C433 are used as slaves in the LIN communication and feature on-board LIN transceivers. A PIC16F874 Flash microcontroller serves as the master. All three microcontrollers are programmed with firmware to provide LIN bus communication.

16.22 PICkit™ 1 Flash Starter Kit

A complete “development system in a box”, the PICkit Flash Starter Kit includes a convenient multi-section board for programming, evaluation and development of 8/14-pin Flash PIC® microcontrollers. Powered via USB, the board operates under a simple Windows GUI. The PICkit 1 Starter Kit includes the user's guide (on CD ROM), PICkit 1 tutorial software and code for various applications. Also included are MPLAB® IDE (Integrated Development Environment) software, software and hardware “Tips 'n Tricks for 8-pin Flash PIC® Microcontrollers” Handbook and a USB Interface Cable. Supports all current 8/14-pin Flash PIC microcontrollers, as well as many future planned devices.

16.23 PICDEM USB PIC16C7X5 Demonstration Board

The PICDEM USB Demonstration Board shows off the capabilities of the PIC16C745 and PIC16C765 USB microcontrollers. This board provides the basis for future USB products.

16.24 Evaluation and Programming Tools

In addition to the PICDEM series of circuits, Microchip has a line of evaluation kits and demonstration software for these products.

- KEELOQ evaluation and programming tools for Microchip’s HCS Secure Data Products
- CAN developers kit for automotive network applications
- Analog design boards and filter design software
- PowerSmart battery charging evaluation/ calibration kits
- IrDA® development kit
- microID development and rfLab™ development software
- SEEVAL® designer kit for memory evaluation and endurance calculations
- PICDEM MSC demo boards for Switching mode power supply, high power IR driver, delta sigma ADC, and flow rate sensor

Check the Microchip web page and the latest Product Line Card for the complete list of demonstration and evaluation kits.

NOTES:

2003 Microchip Technology Inc.

17.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias ..... -55 to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to V_{SS} (except V_{DD}, $\overline{MCLR}$. and RA4) ..... -0.3V to (V_{DD} + 0.3V)
Voltage on V_{DD} with respect to V_{SS} ..... -0.3 to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} **(Note 2)** ..... 0 to +14V
Voltage on RA4 with respect to Vss ..... 0 to +8.5V
Total power dissipation **(Note 1)** 1.0W
Maximum current out of V_{SS} pin ..... 300 mA
Maximum current into V_{DD} pin ..... 250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ± 20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD}) ..... ± 20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by PORTA, PORTB and PORTE (combined) **(Note 3)** ..... 200 mA
Maximum current sourced by PORTA, PORTB and PORTE (combined) **(Note 3)** ..... 200 mA
Maximum current sunk by PORTC and PORTD (combined) **(Note 3)** ..... 200 mA
Maximum current sourced by PORTC and PORTD (combined) **(Note 3)** ..... 200 mA

Note 1: Power dissipation is calculated as follows: Pdis = V_{DD} x {I_{DD} - $\sum I_{OH}$} + $\sum$ {(V_{DD} - V_{OH}) x I_{OH}} + $\sum$(V_{OL} x I_{OL})

2: Voltage spikes below V_{SS} at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$ pin rather than pulling this pin directly to V_{SS}.

3: PORTD and PORTE are not implemented on PIC16F873A/876A devices.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

FIGURE 17-1: PIC16F87XA VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL, EXTENDED)

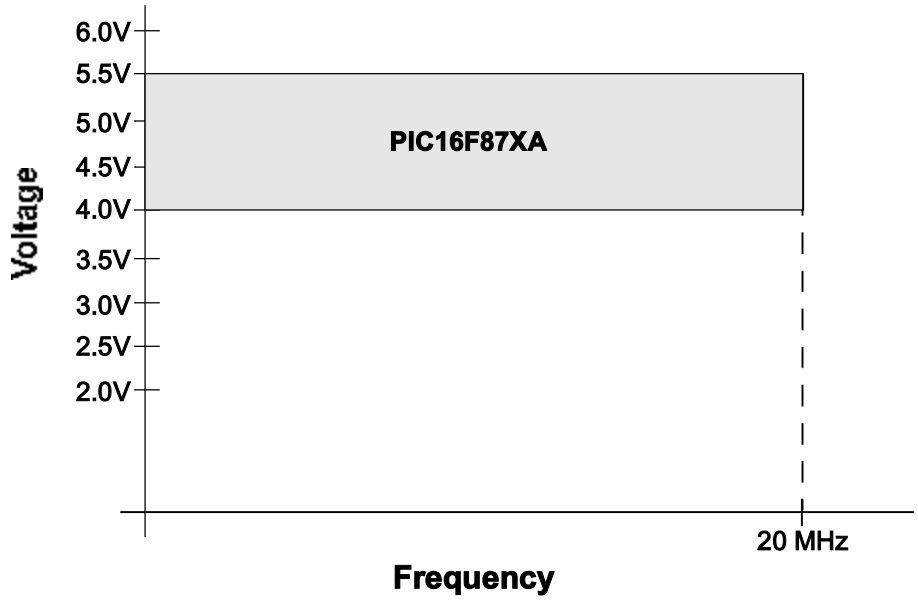
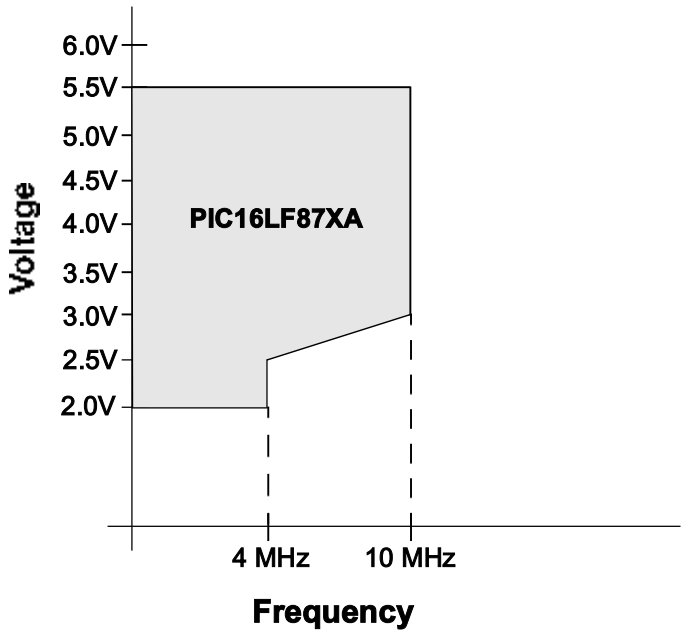


FIGURE 17-2: PIC16LF87XA VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



FMAX = (6.0 MHz/V) (VDDAPPMIN – 2.0V) + 4 MHz

Note 1: VDDAPPMIN is the minimum voltage of the PICmicro® device in the application.

2: FMAX has a maximum frequency of 10 MHz.

© 2003 Microchip Technology Inc.

17.1 DC Characteristics: PIC16F873A/874A/876A/877A (Industrial, Extended) PIC16LF873A/874A/876A/877A (Industrial)

PIC16LF873A/874A/876A/877A (Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC16F873A/874A/876A/877A (Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Symbol**	**Characteristic/ Device**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	VDD	**Supply Voltage**					
D001		16LF87XA	2.0	—	5.5	V	All configurations (DC to 10 MHz)
D001		16F87XA	4.0	—	5.5	V	All configurations
D001A			VBOR		5.5	V	BOR enabled, FMAX = 14 MHz(7)
D002	VDR	**RAM Data Retention Voltage(1)**	—	1.5	—	V	
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	VSS	—	V	See **Section 14.5 "Power-on Reset (POR)"** for details
D004	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See **Section 14.5 "Power-on Reset (POR)"** for details
D005	VBOR	**Brown-out Reset Voltage**	3.65	4.0	4.35	V	BODEN bit in configuration word enabled

Legend: Rows with standard voltage device data only are shaded for improved readability.

† Data in "Typ" column is at 5V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading, switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

3: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

5: Timer1 oscillator (when enabled) adds approximately 20 μA to the specification. This value is from characterization and is for design guidance only. This is not tested.

6: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

7: When BOR is enabled, the device will operate correctly until the VBOR voltage trip point is reached.

17.1 DC Characteristics: PIC16F873A/874A/876A/877A (Industrial, Extended) PIC16LF873A/874A/876A/877A (Industrial) (Continued)

PIC16LF873A/874A/876A/877A (Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC16F873A/874A/876A/877A (Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Symbol**	**Characteristic/ Device**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	IDD	**Supply Current(2,5)**					
D010		16LF87XA	—	0.6	2.0	mA	XT, RC osc configurations, FOSC = 4 MHz, VDD = 3.0V
D010		16F87XA	—	1.6	4	mA	XT, RC osc configurations, FOSC = 4 MHz, VDD = 5.5V
D010A		16LF87XA	—	20	35	µA	LP osc configuration, FOSC = 32 kHz, VDD = 3.0V, WDT disabled
D013		16F87XA	—	7	15	mA	HS osc configuration, FOSC = 20 MHz, VDD = 5.5V
D015	ΔIBOR	**Brown-out Reset Current(6)**	—	85	200	µA	BOR enabled, VDD = 5.0V

Legend: Rows with standard voltage device data only are shaded for improved readability.

† Data in "Typ" column is at 5V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading, switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

5: Timer1 oscillator (when enabled) adds approximately 20 µA to the specification. This value is from characterization and is for design guidance only. This is not tested.

6: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

7: When BOR is enabled, the device will operate correctly until the VBOR voltage trip point is reached.

© 2003 Microchip Technology Inc.

17.1 DC Characteristics: PIC16F873A/874A/876A/877A (Industrial, Extended) PIC16LF873A/874A/876A/877A (Industrial) (Continued)

PIC16LF873A/874A/876A/877A (Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC16F873A/874A/876A/877A (Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Symbol**	**Characteristic/ Device**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	IPD	**Power-down Current(3,5)**					
D020		16LF87XA	—	7.5	30	µA	VDD = 3.0V, WDT enabled, -40°C to +85°C
D020		16F87XA	—	10.5	42	µA	VDD = 4.0V, WDT enabled, -40°C to +85°C
					60	µA	VDD = 4.0V, WDT enabled, -40°C to +125°C (extended)
D021		16LF87XA	—	0.9	5	µA	VDD = 3.0V, WDT disabled, 0°C to +70°C
D021		16F87XA	—	1.5	16	µA	VDD = 4.0V, WDT disabled, -40°C to +85°C
					20	µA	VDD = 4.0V, WDT disabled, -40°C to +125°C (extended)
D021A		16LF87XA		0.9	5	µA	VDD = 3.0V, WDT disabled, -40°C to +85°C
D021A		16F87XA		1.5	19	µA	VDD = 4.0V, WDT disabled, -40°C to +85°C
D023	ΔIBOR	**Brown-out Reset Current(6)**	—	85	200	µA	BOR enabled, VDD = 5.0V

Legend: Rows with standard voltage device data only are shaded for improved readability.

† Data in "Typ" column is at 5V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading, switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

5: Timer1 oscillator (when enabled) adds approximately 20 µA to the specification. This value is from characterization and is for design guidance only. This is not tested.

6: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

7: When BOR is enabled, the device will operate correctly until the VBOR voltage trip point is reached.

17.2 DC Characteristics: PIC16F873A/874A/876A/877A (Industrial, Extended) PIC16LF873A/874A/876A/877A (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC specification (Section 17.1)				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**					
		I/O ports:					
D030		with TTL buffer	VSS	—	0.15 VDD	V	For entire VDD range
D030A			VSS	—	0.8V	V	4.5V ≤ VDD ≤ 5.5V
D031		with Schmitt Trigger buffer	VSS	—	0.2 VDD	V	
D032		$\overline{MCLR}$, OSC1 (in RC mode)	VSS	—	0.2 VDD	V	
D033		OSC1 (in XT and LP modes)	VSS	—	0.3V	V	**(Note 1)**
		OSC1 (in HS mode)	VSS	—	0.3 VDD	V	
		Ports RC3 and RC4:		—			
D034		with Schmitt Trigger buffer	VSS	—	0.3 VDD	V	For entire VDD range
D034A		with SMBus	-0.5	—	0.6	V	For VDD = 4.5 to 5.5V
	VIH	**Input High Voltage**					
		I/O ports:		—			
D040		with TTL buffer	2.0	—	VDD	V	4.5V ≤ VDD ≤ 5.5V
D040A			0.25 VDD + 0.8V	—	VDD	V	For entire VDD range
D041		with Schmitt Trigger buffer	0.8 VDD	—	VDD	V	For entire VDD range
D042		$\overline{MCLR}$	0.8 VDD	—	VDD	V	
D042A		OSC1 (in XT and LP modes)	1.6V	—	VDD	V	**(Note 1)**
		OSC1 (in HS mode)	0.7 VDD	—	VDD	V	
D043		OSC1 (in RC mode)	0.9 VDD	—	VDD	V	
		Ports RC3 and RC4:					
D044		with Schmitt Trigger buffer	0.7 VDD	—	VDD	V	For entire VDD range
D044A		with SMBus	1.4	—	5.5	V	For VDD = 4.5 to 5.5V
D070	IPURB	**PORTB Weak Pull-up Current**	50	250	400	µA	VDD = 5V, VPIN = VSS, -40°C TO +85°C
	IIL	**Input Leakage Current(2, 3)**					
D060		I/O ports	—	—	±1	µA	VSS ≤ VPIN ≤ VDD, pin at high-impedance
D061		$\overline{MCLR}$, RA4/T0CKI	—	—	±5	µA	VSS ≤ VPIN ≤ VDD
D063		OSC1	—	—	±5	µA	VSS ≤ VPIN ≤ VDD, XT, HS and LP osc configuration

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC16F87XA be driven with external clock in RC mode.

2: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

© 2003 Microchip Technology Inc.

17.2 DC Characteristics: PIC16F873A/874A/876A/877A (Industrial, Extended) PIC16LF873A/874A/876A/877A (Industrial) (Continued)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC specification (Section 17.1)				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	VOL	**Output Low Voltage**					
D080		I/O ports	—	—	0.6	V	IOL = 8.5 mA, VDD = 4.5V, -40°C to +85°C
D083		OSC2/CLKO (RC osc config)	—	—	0.6	V	IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C
	VOH	**Output High Voltage**					
D090		I/O ports(3)	VDD – 0.7	—	—	V	IOH = -3.0 mA, VDD = 4.5V, -40°C to +85°C
D092		OSC2/CLKO (RC osc config)	VDD – 0.7	—	—	V	IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C
D150*	VOD	**Open-Drain High Voltage**	—	—	8.5	V	RA4 pin
		Capacitive Loading Specs on Output Pins					
D100	COSC2	OSC2 pin	—	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101	CIO	All I/O pins and OSC2 (RC mode)	—	—	50	pF	
D102	CB	SCL, SDA (I^2C mode)	—	—	400	pF	
		Data EEPROM Memory					
D120	ED	Endurance	100K	1M	—	E/W	-40°C to +85°C
D121	VDRW	VDD for read/write	VMIN	—	5.5	V	Using EECON to read/write, VMIN = min. operating voltage
D122	TDEW	Erase/write cycle time	—	4	8	ms	
		Program Flash Memory					
D130	EP	Endurance	10K	100K	—	E/W	-40°C to +85°C
D131	VPR	VDD for read	VMIN	—	5.5	V	VMIN = min. operating voltage
D132A		VDD for erase/write	VMIN	—	5.5	V	Using EECON to read/write, VMIN = min. operating voltage
D133	TPEW	Erase/Write cycle time	—	4	8	ms	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC16F87XA be driven with external clock in RC mode.

2: The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

TABLE 17-1: COMPARATOR SPECIFICATIONS

Operating Conditions:	3.0V < VDD < 5.5V, -40°C < TA < +85°C (unless otherwise stated) 4.0V < VDD < 5.5V, -40°C < TA < +125°C (unless otherwise stated)						
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D300	VIOFF	Input Offset Voltage	—	± 5.0	± 10	mV	
D301	VICM	Input Common Mode Voltage*	0	-	VDD – 1.5	V	
D302	CMRR	Common Mode Rejection Ratio*	55	-	—	dB	
300 300A	TRESP	Response Time*(1)	—	150	400 600	ns ns	PIC16F87XA PIC16LF87XA
301	TMC2OV	Comparator Mode Change to Output Valid*	—	—	10	µs	

* These parameters are characterized but not tested.

Note 1: Response time measured with one comparator input at (VDD – 1.5)/2 while the other input transitions from VSS to VDD.

TABLE 17-2: VOLTAGE REFERENCE SPECIFICATIONS

Operating Conditions:	3.0V < VDD < 5.5V, -40°C < TA < +85°C (unless otherwise stated) 4.0V < VDD < 5.5V, -40°C < TA < +125°C (unless otherwise stated)						
Spec No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D310	VRES	Resolution	VDD/24	—	VDD/32	LSb	
D311	VRAA	Absolute Accuracy	— —	— —	1/2 1/2	LSb LSb	Low Range (VRR = 1) High Range (VRR = 0)
D312	VRUR	Unit Resistor Value (R)*	—	2k	—	Ω	
310	TSET	Settling Time*(1)	—	—	10	µs	

* These parameters are characterized but not tested.

Note 1: Settling time measured while VRR = 1 and VR<3:0> transitions from '0000' to '1111'.

© 2003 Microchip Technology Inc.

17.3 Timing Parameter Symbology

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I^2C specifications only)
4. Ts (I^2C specifications only)

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

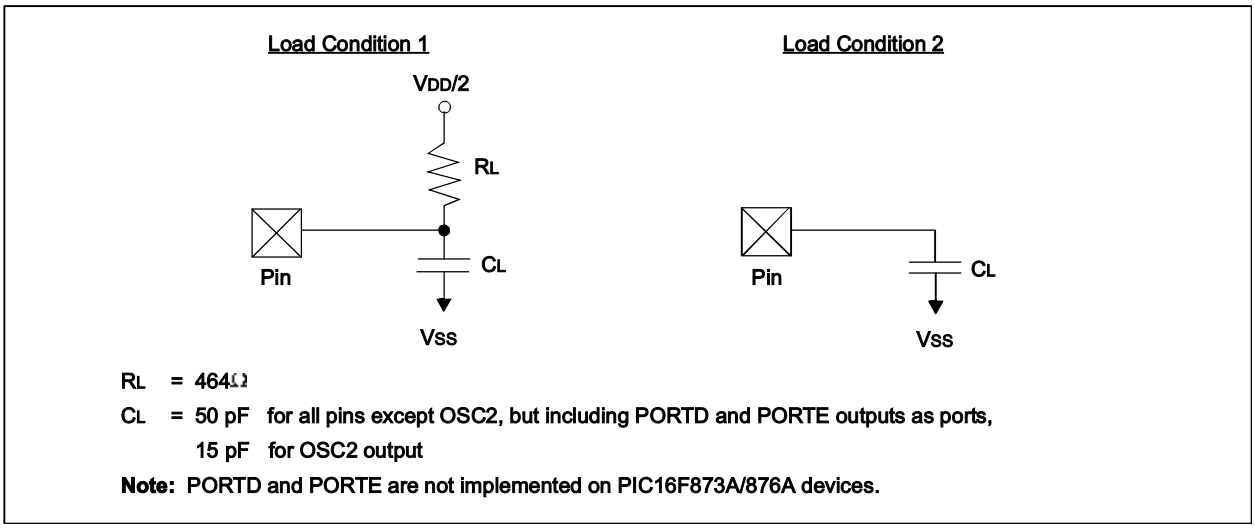
Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (High-impedance)	V	Valid
L	Low	Z	High-impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low

TCC:ST (I^2C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	Data input hold	STO	Stop condition
STA	Start condition		

FIGURE 17-3: LOAD CONDITIONS



RL = 464Ω

CL = 50 pF for all pins except OSC2, but including PORTD and PORTE outputs as ports,
15 pF for OSC2 output

Note: PORTD and PORTE are not implemented on PIC16F873A/876A devices.

FIGURE 17-4: EXTERNAL CLOCK TIMING

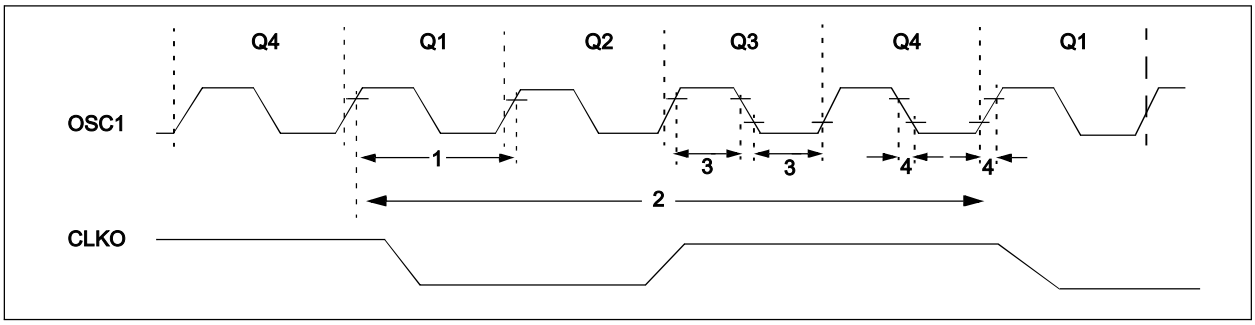


TABLE 17-3: EXTERNAL CLOCK TIMING REQUIREMENTS

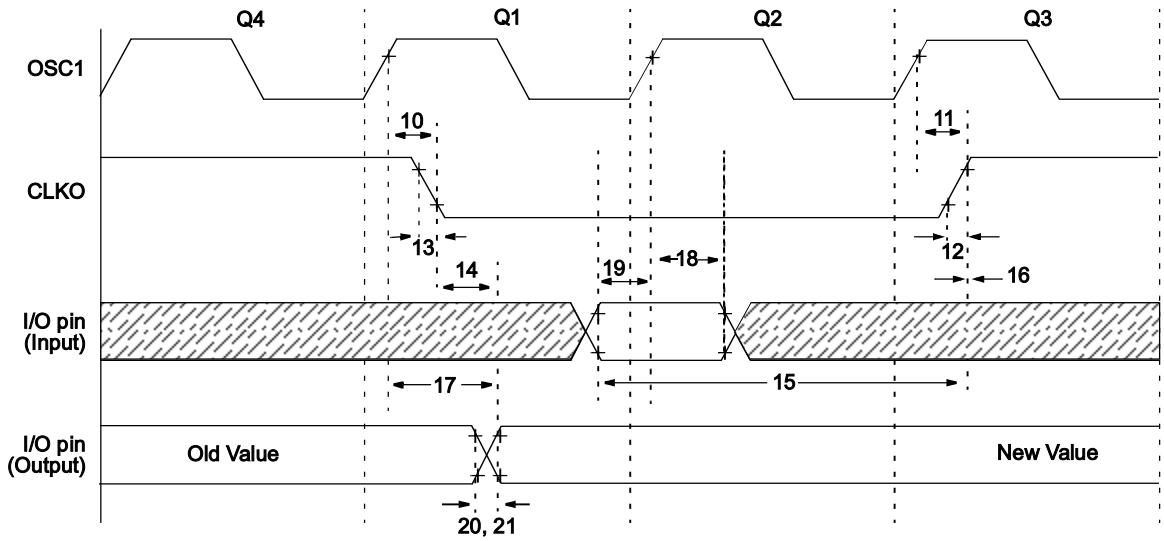
Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
	FOSC	External CLKI Frequency (Note 1)	DC	—	1	MHz	XT and RC Osc mode
			DC	—	20	MHz	HS Osc mode
			DC	—	32	kHz	LP Osc mode
		Oscillator Frequency (Note 1)	DC	—	4	MHz	RC Osc mode
			0.1	—	4	MHz	XT Osc mode
			4	—	20	MHz	HS Osc mode
			5	—	200	kHz	LP Osc mode
1	TOSC	External CLKI Period (Note 1)	1000	—	—	ns	XT and RC Osc mode
			50	—	—	ns	HS Osc mode
			5	—	—	µs	LP Osc mode
		Oscillator Period (Note 1)	250	—	—	ns	RC Osc mode
			250	—	1	µs	XT Osc mode
			100	—	250	ns	HS Osc mode
			50	—	250	ns	HS Osc mode
			31.25	—	—	µs	LP Osc mode
2	TCY	Instruction Cycle Time (Note 1)	200	TCY	DC	ns	TCY = 4/FOSC
3	TOSL, TOSH	External Clock in (OSC1) High or Low Time	100	—	—	ns	XT oscillator
			2.5	—	—	µs	LP oscillator
			15	—	—	ns	HS oscillator
4	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	—	25	ns	XT oscillator
			—	—	50	ns	LP oscillator
			—	—	15	ns	HS oscillator

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type, under standard operating conditions, with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the "max." cycle time limit is "DC" (no clock) for all devices.

© 2003 Microchip Technology Inc.

FIGURE 17-5: CLKO AND I/O TIMING



Note: Refer to Figure 17-3 for load conditions.

TABLE 17-4: CLKO AND I/O TIMING REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
10*	TosH2ckL	OSC1 ↑ to CLKO ↓		—	75	200	ns	**(Note 1)**
11*	TosH2ckH	OSC1 ↑ to CLKO ↑		—	75	200	ns	**(Note 1)**
12*	TckR	CLKO Rise Time		—	35	100	ns	**(Note 1)**
13*	TckF	CLKO Fall Time		—	35	100	ns	**(Note 1)**
14*	TckL2ioV	CLKO ↓ to Port Out Valid		—	—	0.5 Tcy + 20	ns	**(Note 1)**
15*	TioV2ckH	Port In Valid before CLKO ↑		Tosc + 200	—	—	ns	**(Note 1)**
16*	TckH2ioI	Port In Hold after CLKO ↑		0	—	—	ns	**(Note 1)**
17*	TosH2ioV	OSC1 ↑ (Q1 cycle) to Port Out Valid		—	100	255	ns	
18*	TosH2ioI	OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time)	Standard (F)	100	—	—	ns	
			Extended (LF)	200	—	—	ns	
19*	TioV2osH	Port Input Valid to OSC1 ↑ (I/O in setup time)		0	—	—	ns	
20*	TioR	Port Output Rise Time	Standard (F)	—	10	40	ns	
			Extended (LF)	—	—	145	ns	
21*	TioF	Port Output Fall Time	Standard (F)	—	10	40	ns	
			Extended (LF)	—	—	145	ns	
22††*	Tinp	INT pin High or Low Time		Tcy	—	—	ns	
23††*	Trbp	RB7:RB4 Change INT High or Low Time		Tcy	—	—	ns	

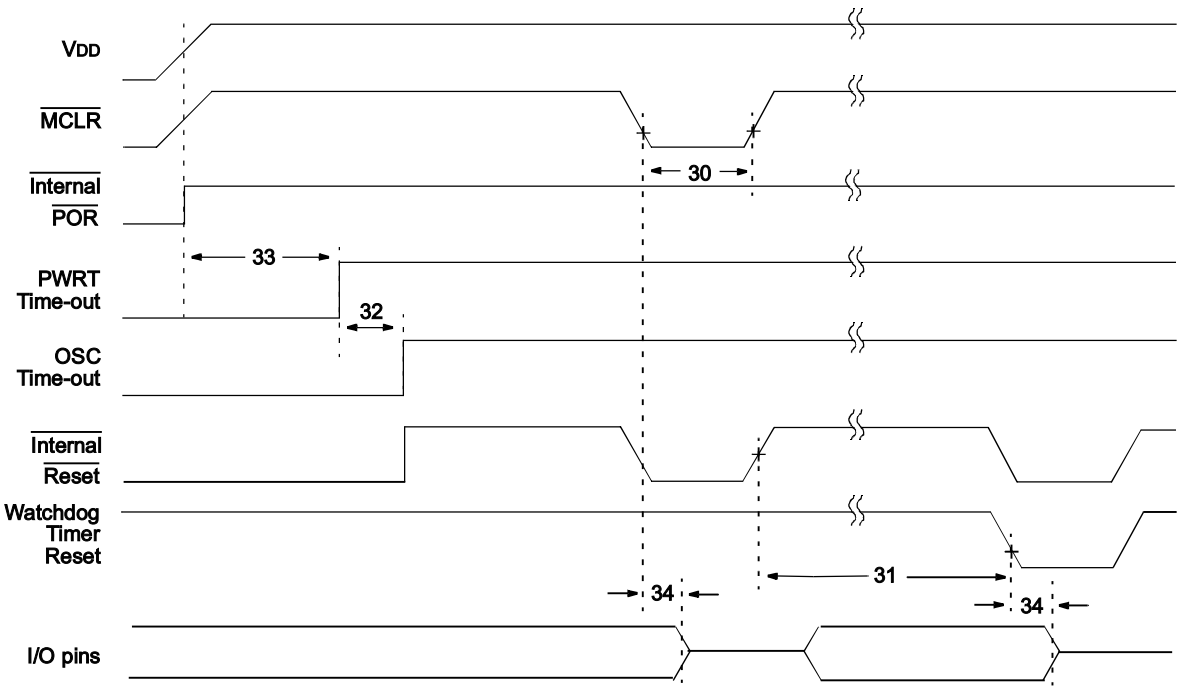
* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

†† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC mode where CLKO output is 4 x Tosc.

FIGURE 17-6: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note: Refer to Figure 17-3 for load conditions.

FIGURE 17-7: BROWN-OUT RESET TIMING

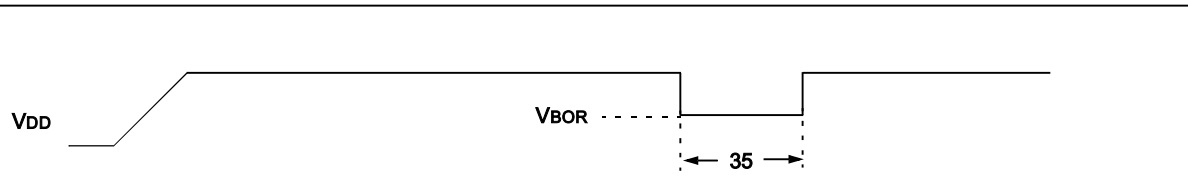


TABLE 17-5: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

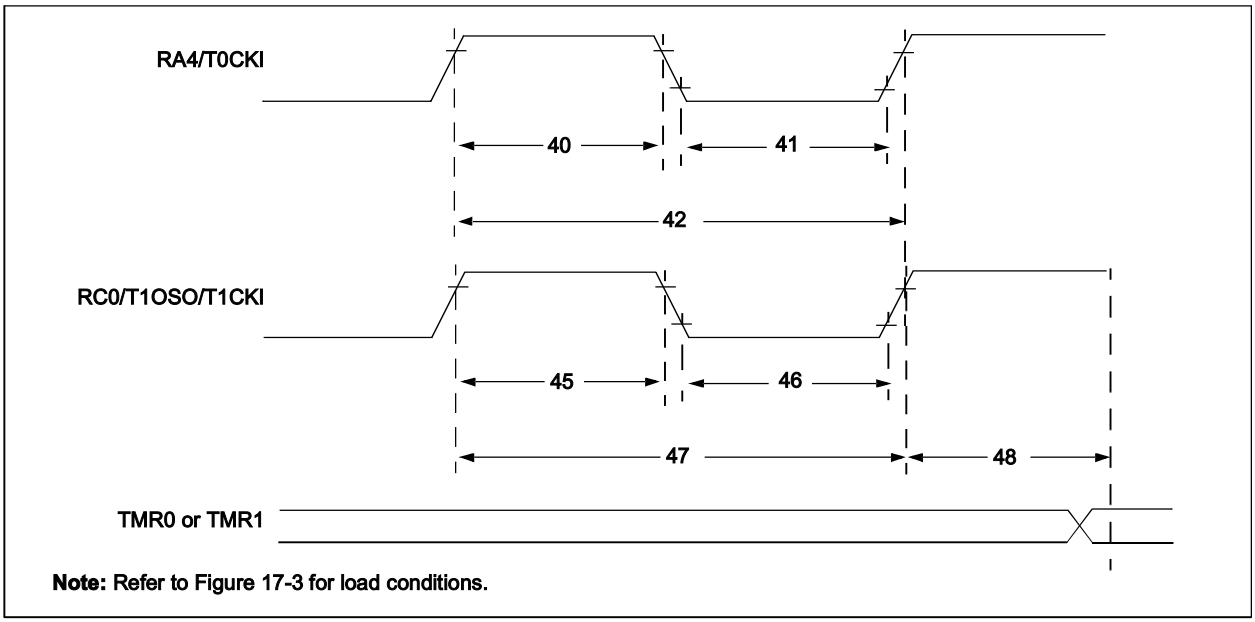
Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
30	TMCL	MCLR Pulse Width (low)	2	—	—	µs	VDD = 5V, -40°C to +85°C
31*	TWDT	Watchdog Timer Time-out Period (no prescaler)	7	18	33	ms	VDD = 5V, -40°C to +85°C
32	TOST	Oscillation Start-up Timer Period	—	1024 TOSC	—	—	TOSC = OSC1 period
33*	TPWRT	Power-up Timer Period	28	72	132	ms	VDD = 5V, -40°C to +85°C
34	TIOZ	I/O High-Impedance from MCLR Low or Watchdog Timer Reset	—	—	2.1	µs	
35	TBOR	Brown-out Reset Pulse Width	100	—	—	µs	VDD ≤ VBOR (D005)

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

© 2003 Microchip Technology Inc.

FIGURE 17-8: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



Note: Refer to Figure 17-3 for load conditions.

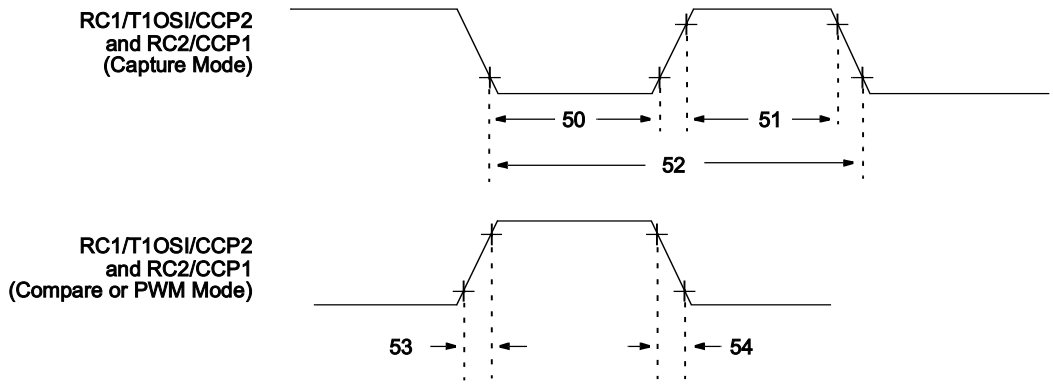
TABLE 17-6: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param No.	Symbol	Characteristic			Min	Typ†	Max	Units	Conditions
40*	TT0H	T0CKI High Pulse Width		No Prescaler	0.5 TCY + 20	—	—	ns	Must also meet parameter 42
				With Prescaler	10	—	—	ns	
41*	TT0L	T0CKI Low Pulse Width		No Prescaler	0.5 TCY + 20	—	—	ns	Must also meet parameter 42
				With Prescaler	10	—	—	ns	
42*	TT0P	T0CKI Period		No Prescaler	TCY + 40	—	—	ns	
				With Prescaler	Greater of: 20 or TCY + 40 / N	—	—	ns	N = prescale value (2, 4,..., 256)
45*	TT1H	T1CKI High Time	Synchronous, Prescaler = 1		0.5 TCY + 20	—	—	ns	Must also meet parameter 47
			Synchronous, Prescaler = 2, 4, 8	Standard(F)	15	—	—	ns	
				Extended(LF)	25	—	—	ns	
			Asynchronous	Standard(F)	30	—	—	ns	
				Extended(LF)	50	—	—	ns	
46*	TT1L	T1CKI Low Time	Synchronous, Prescaler = 1		0.5 TCY + 20	—	—	ns	Must also meet parameter 47
			Synchronous, Prescaler = 2, 4, 8	Standard(F)	15	—	—	ns	
				Extended(LF)	25	—	—	ns	
			Asynchronous	Standard(F)	30	—	—	ns	
				Extended(LF)	50	—	—	ns	
47*	TT1P	T1CKI Input Period	Synchronous	Standard(F)	Greater of: 30 or TCY + 40 / N	—	—	ns	N = prescale value (1, 2, 4, 8)
				Extended(LF)	Greater of: 50 or TCY + 40 / N				N = prescale value (1, 2, 4, 8)
			Asynchronous	Standard(F)	60	—	—	ns	
				Extended(LF)	100	—	—	ns	
	FT1	Timer1 Oscillator Input Frequency Range (oscillator enabled by setting bit T1OSCEN)			DC	—	200	kHz	
48	TCKEZTMR1	Delay from External Clock Edge to Timer Increment			2 TOSC	—	7 TOSC	—	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 17-9: CAPTURE/COMPARE/PWM TIMINGS (CCP1 AND CCP2)



Note: Refer to Figure 17-3 for load conditions.

TABLE 17-7: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP1 AND CCP2)

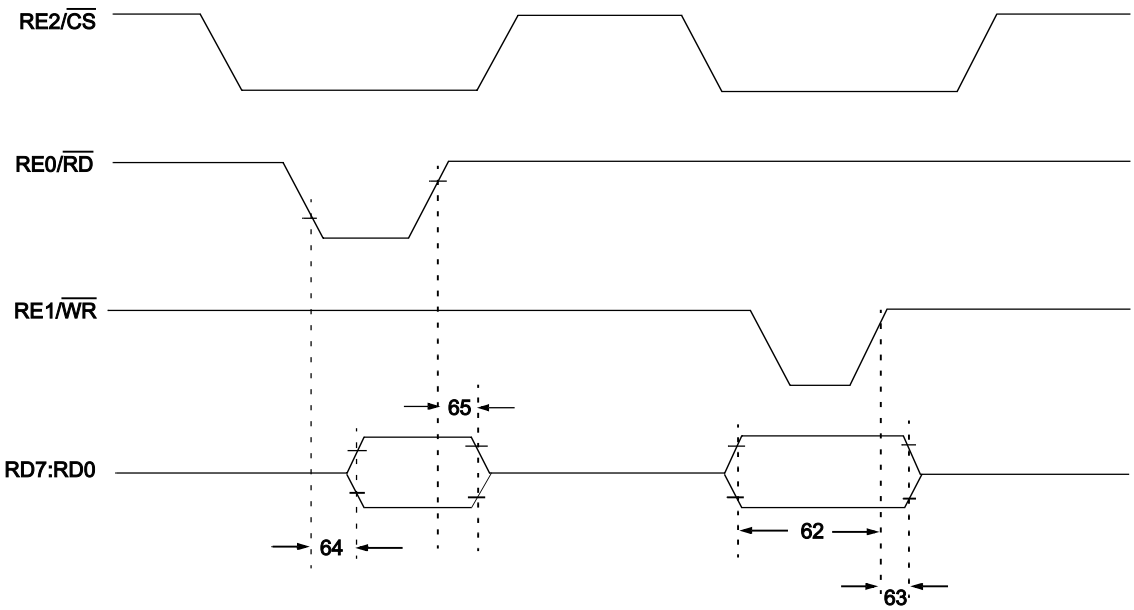
Param No.	Symbol	Characteristic			Min	Typ†	Max	Units	Conditions
50*	TccL	CCP1 and CCP2 Input Low Time	No Prescaler		0.5 TCY + 20	—	—	ns	
			With Prescaler	Standard(F)	10	—	—	ns	
				Extended(LF)	20	—	—	ns	
51*	TccH	CCP1 and CCP2 Input High Time	No Prescaler		0.5 TCY + 20	—	—	ns	
			With Prescaler	Standard(F)	10	—	—	ns	
				Extended(LF)	20	—	—	ns	
52*	TccP	CCP1 and CCP2 Input Period			$\frac{3\ T_{CY} + 40}{N}$	—	—	ns	N = prescale value (1, 4 or 16)
53*	TccR	CCP1 and CCP2 Output Rise Time		Standard(F)	—	10	25	ns	
				Extended(LF)	—	25	50	ns	
54*	TccF	CCP1 and CCP2 Output Fall Time		Standard(F)	—	10	25	ns	
				Extended(LF)	—	25	45	ns	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

© 2003 Microchip Technology Inc.

FIGURE 17-10: PARALLEL SLAVE PORT TIMING (PIC16F874A/877A ONLY)



Note: Refer to Figure 17-3 for load conditions.

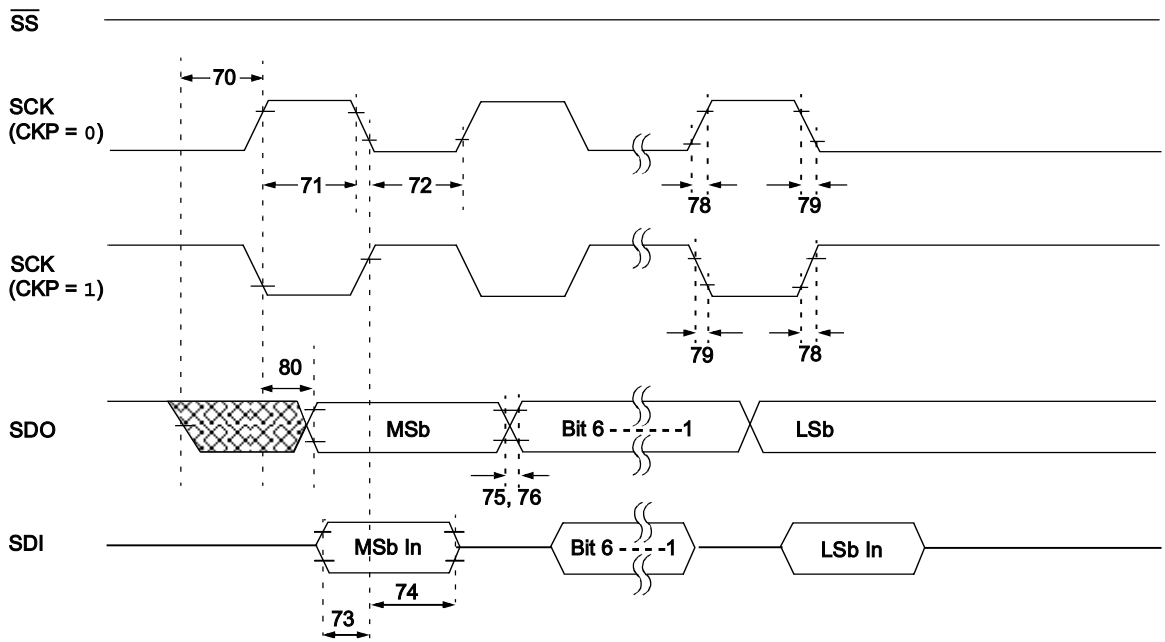
TABLE 17-8: PARALLEL SLAVE PORT REQUIREMENTS (PIC16F874A/877A ONLY)

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
62	TDTV2WRH	Data In Valid before $\overline{WR}$ ↑ or $\overline{CS}$ ↑ (setup time)		20	—	—	ns	
63*	TWRH2DTI	$\overline{WR}$ ↑ or $\overline{CS}$ ↑ to Data–in Invalid (hold time)	Standard(F)	20	—	—	ns	
			Extended(LF)	35	—	—	ns	
64	TRDL2DTV	$\overline{RD}$ ↓ and $\overline{CS}$ ↓ to Data–out Valid		—	—	80	ns	
65	TRDH2DTI	$\overline{RD}$ ↑ or $\overline{CS}$ ↓ to Data–out Invalid		10	—	30	ns	

* These parameters are characterized but not tested.

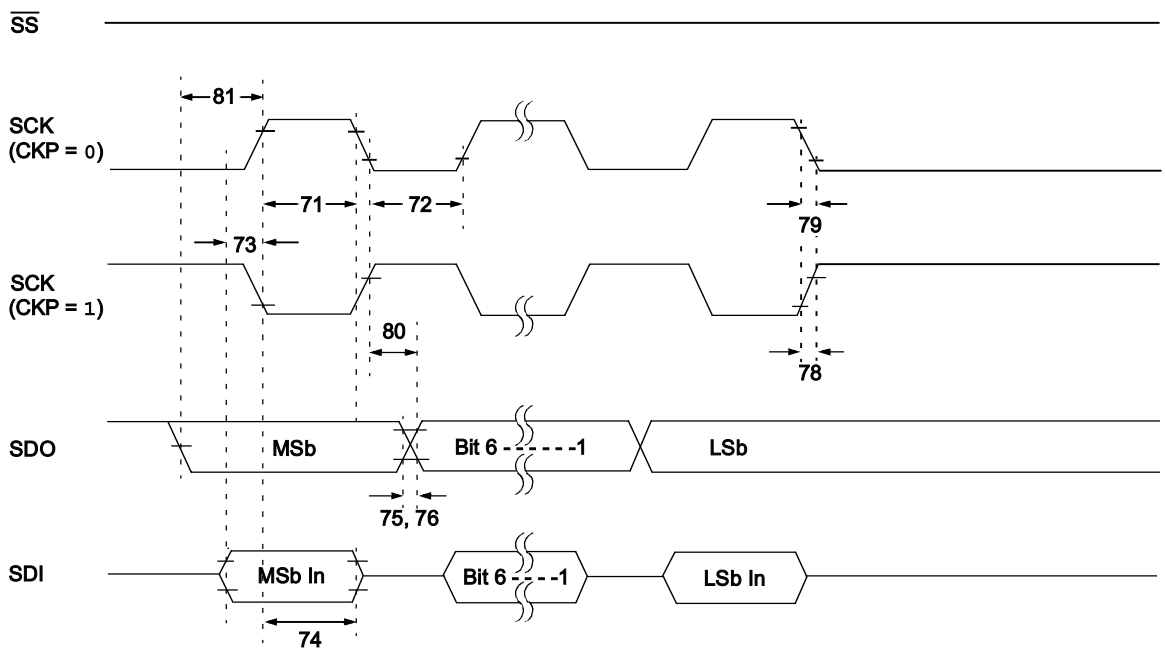
† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 17-11: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)



Note: Refer to Figure 17-3 for load conditions.

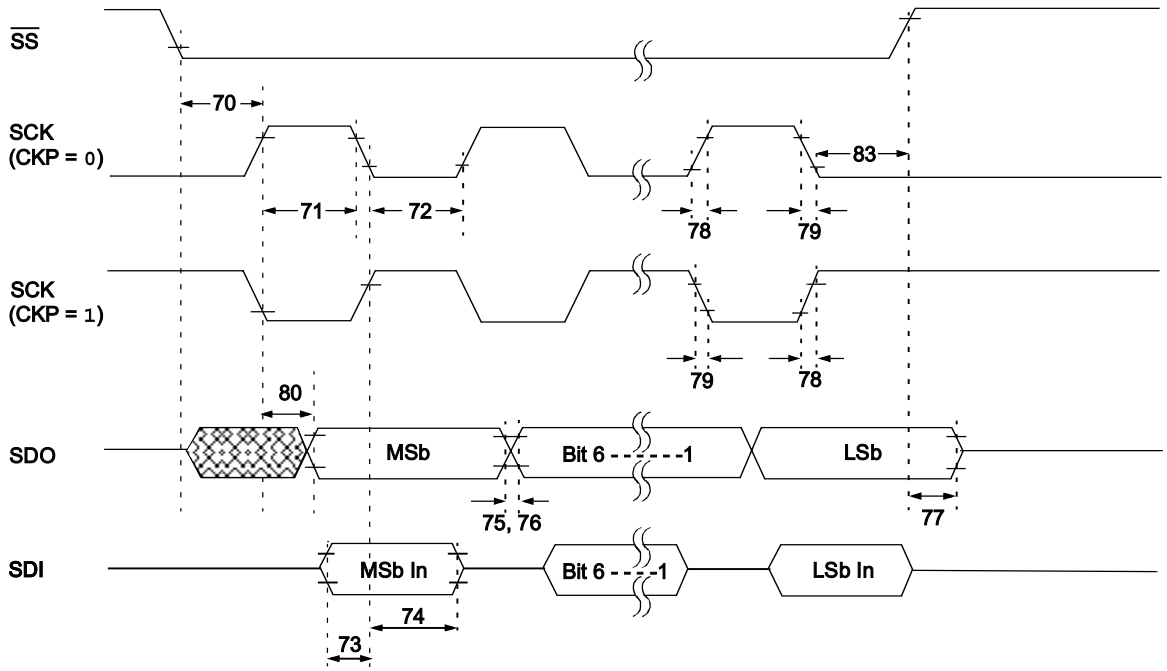
FIGURE 17-12: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)



Note: Refer to Figure 17-3 for load conditions.

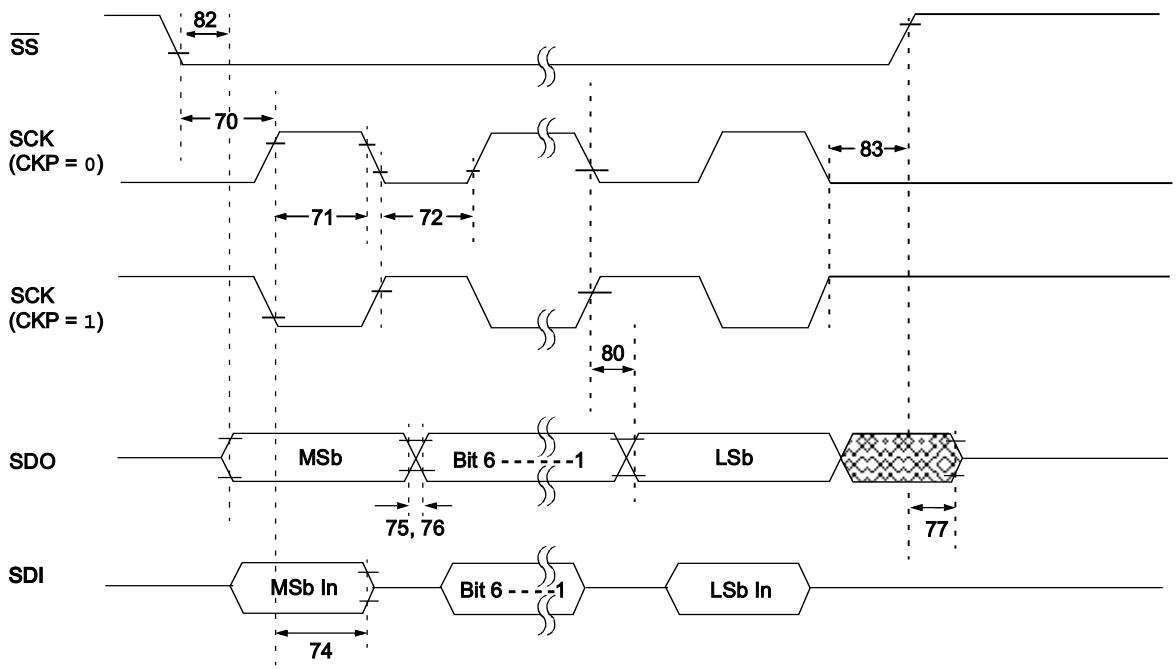
© 2003 Microchip Technology Inc.

FIGURE 17-13: SPI SLAVE MODE TIMING (CKE = 0)



Note: Refer to Figure 17-3 for load conditions.

FIGURE 17-14: SPI SLAVE MODE TIMING (CKE = 1)



Note: Refer to Figure 17-3 for load conditions.

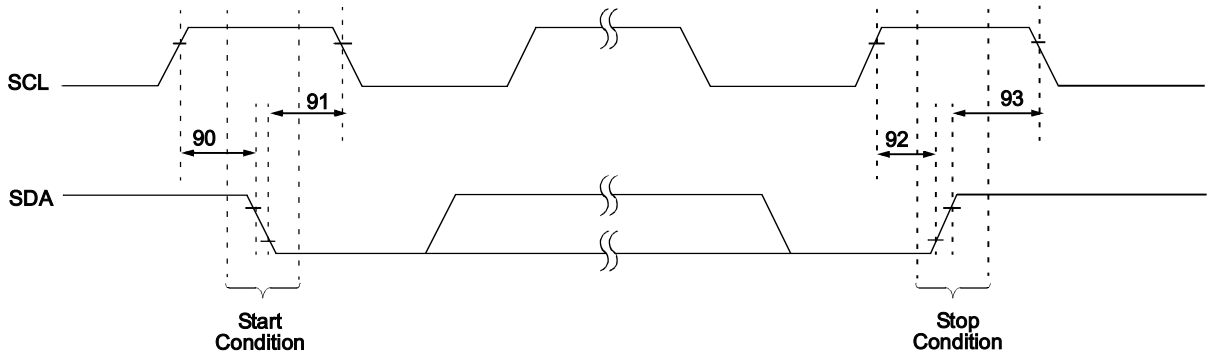
TABLE 17-9: SPI MODE REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
70*	TssL2scH, TssL2scL	$\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input		Tcy	—	—	ns	
71*	TscH	SCK Input High Time (Slave mode)		Tcy + 20	—	—	ns	
72*	TscL	SCK Input Low Time (Slave mode)		Tcy + 20	—	—	ns	
73*	TdiV2scH, TdiV2scL	Setup Time of SDI Data Input to SCK Edge		100	—	—	ns	
74*	TscH2diL, TscL2diL	Hold Time of SDI Data Input to SCK Edge		100	—	—	ns	
75*	TdoR	SDO Data Output Rise Time	Standard(F) Extended(LF)	— —	10 25	25 50	ns ns	
76*	TdoF	SDO Data Output Fall Time		—	10	25	ns	
77*	TssH2doZ	$\overline{SS}$ ↑ to SDO Output High-Impedance		10	—	50	ns	
78*	TscR	SCK Output Rise Time (Master mode)	Standard(F) Extended(LF)	— —	10 25	25 50	ns ns	
79*	TscF	SCK Output Fall Time (Master mode)		—	10	25	ns	
80*	TscH2doV, TscL2doV	SDO Data Output Valid after SCK Edge	Standard(F) Extended(LF)	— —	— —	50 145	ns	
81*	TdoV2scH, TdoV2scL	SDO Data Output Setup to SCK Edge		Tcy	—	—	ns	
82*	TssL2doV	SDO Data Output Valid after $\overline{SS}$ ↓ Edge		—	—	50	ns	
83*	TscH2ssH, TscL2ssH	$\overline{SS}$ ↑ after SCK Edge		1.5 Tcy + 40	—	—	ns	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 17-15: I^2C BUS START/STOP BITS TIMING



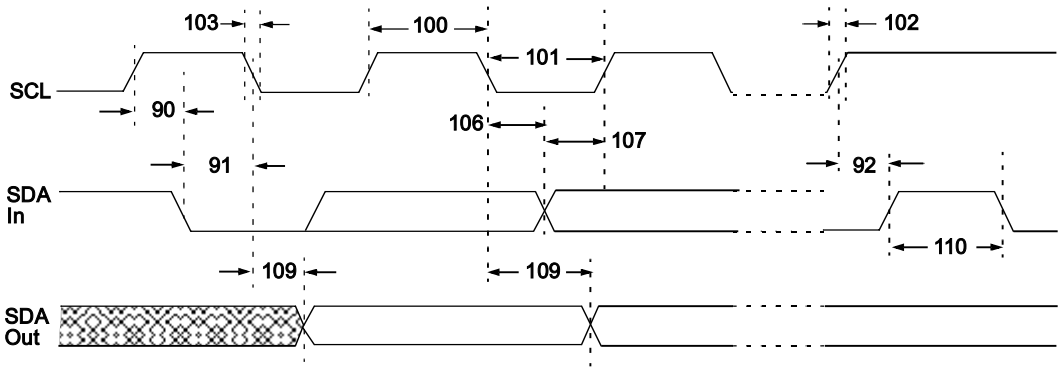
Note: Refer to Figure 17-3 for load conditions.

© 2003 Microchip Technology Inc.

TABLE 17-10: I^2C BUS START/STOP BITS REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
90	TSU:STA	Start condition	100 kHz mode	4700	—	—	ns	Only relevant for Repeated Start condition
		Setup time	400 kHz mode	600	—	—		
91	THD:STA	Start condition	100 kHz mode	4000	—	—	ns	After this period, the first clock pulse is generated
		Hold time	400 kHz mode	600	—	—		
92	TSU:STO	Stop condition	100 kHz mode	4700	—	—	ns	
		Setup time	400 kHz mode	600	—	—		
93	THD:STO	Stop condition	100 kHz mode	4000	—	—	ns	
		Hold time	400 kHz mode	600	—	—		

FIGURE 17-16: I^2C BUS DATA TIMING



Note: Refer to Figure 17-3 for load conditions.

TABLE 17-11: I^2C BUS DATA REQUIREMENTS

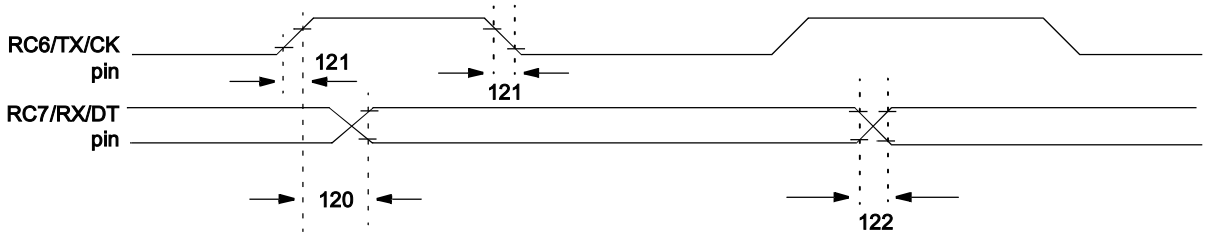
Param No.	Sym	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	4.0	—	µs	
			400 kHz mode	0.6	—	µs	
			SSP Module	0.5 TCY	—		
101	TLOW	Clock Low Time	100 kHz mode	4.7	—	µs	
			400 kHz mode	1.3	—	µs	
			SSP Module	0.5 TCY	—		
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	Cb is specified to be from 10 to 400 pF
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4.7	—	µs	Only relevant for Repeated Start condition
			400 kHz mode	0.6	—	µs	
91	THD:STA	Start Condition Hold Time	100 kHz mode	4.0	—	µs	After this period, the first clock pulse is generated
			400 kHz mode	0.6	—	µs	
106	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	µs	
107	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4.7	—	µs	
			400 kHz mode	0.6	—	µs	
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns	**(Note 1)**
			400 kHz mode	—	—	ns	
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	µs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	µs	
	CB	Bus Capacitive Loading		—	400	pF	

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

2: A fast mode (400 kHz) I^2C bus device can be used in a standard mode (100 kHz) I^2C bus system, but the requirement that, TSU:DAT ≥ 250 ns, must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line, TR MAX. + TSU:DAT = 1000 + 250 = 1250 ns (according to the standard mode I^2C bus specification), before the SCL line is released.

© 2003 Microchip Technology Inc.

FIGURE 17-17: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING



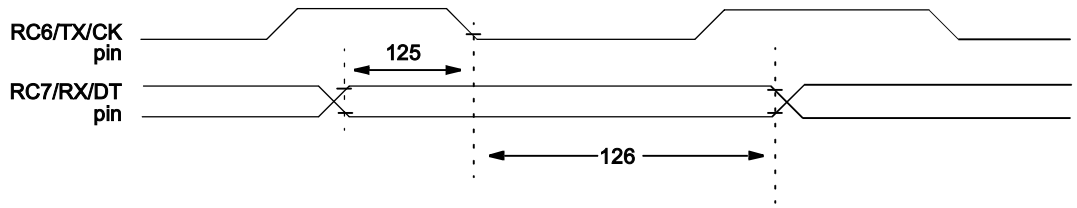
Note: Refer to Figure 17-3 for load conditions.

TABLE 17-12: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
120	TCKH2DTV	SYNC XMIT (MASTER & SLAVE) Clock High to Data Out Valid	Standard(F)	—	—	80	ns	
			Extended(LF)	—	—	100	ns	
121	TCKRF	Clock Out Rise Time and Fall Time (Master mode)	Standard(F)	—	—	45	ns	
			Extended(LF)	—	—	50	ns	
122	TDTRF	Data Out Rise Time and Fall Time	Standard(F)	—	—	45	ns	
			Extended(LF)	—	—	50	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 17-18: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 17-3 for load conditions.

TABLE 17-13: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
125	TDTV2CKL	SYNC RCV (MASTER & SLAVE) Data Setup before CK ↓ (DT setup time)	15	—	—	ns	
126	TCKL2DTL	Data Hold after CK ↓ (DT hold time)	15	—	—	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

TABLE 17-14: A/D CONVERTER CHARACTERISTICS:PIC16F873A/874A/876A/877A (INDUSTRIAL)
PIC16LF873A/874A/876A/877A (INDUSTRIAL)

Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
A01	NR	Resolution		—	—	10-bits	bit	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A03	EIL	Integral Linearity Error		—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A04	EDL	Differential Linearity Error		—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A06	EOFF	Offset Error		—	—	< ± 2	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A07	EGN	Gain Error		—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A10	—	Monotonicity		—	guaranteed(3)	—	—	VSS ≤ VAIN ≤ VREF
A20	VREF	Reference Voltage (VREF+ – VREF-)		2.0	—	VDD + 0.3	V	
A21	VREF+	Reference Voltage High		AVDD – 2.5V		AVDD + 0.3V	V	
A22	VREF-	Reference Voltage Low		AVSS – 0.3V		VREF+ – 2.0V	V	
A25	VAIN	Analog Input Voltage		VSS – 0.3V	—	VREF + 0.3V	V	
A30	ZAIN	Recommended Impedance of Analog Voltage Source		—	—	2.5	kΩ	**(Note 4)**
A40	IAD	A/D Conversion Current (VDD)	PIC16F87XA	—	220	—	μA	Average current consumption when A/D is on **(Note 1)**
			PIC16LF87XA	—	90	—	μA	
A50	IREF	VREF Input Current **(Note 2)**		—	—	5	μA	During VAIN acquisition. Based on differential of VHOLD to VAIN to charge CHOLD, see **Section 11.1 "A/D Acquisition Requirements"**.
				—	—	150	μA	During A/D conversion cycle

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: When A/D is off, it will not consume any current other than minor leakage current. The power-down current spec includes any such leakage from the A/D module.

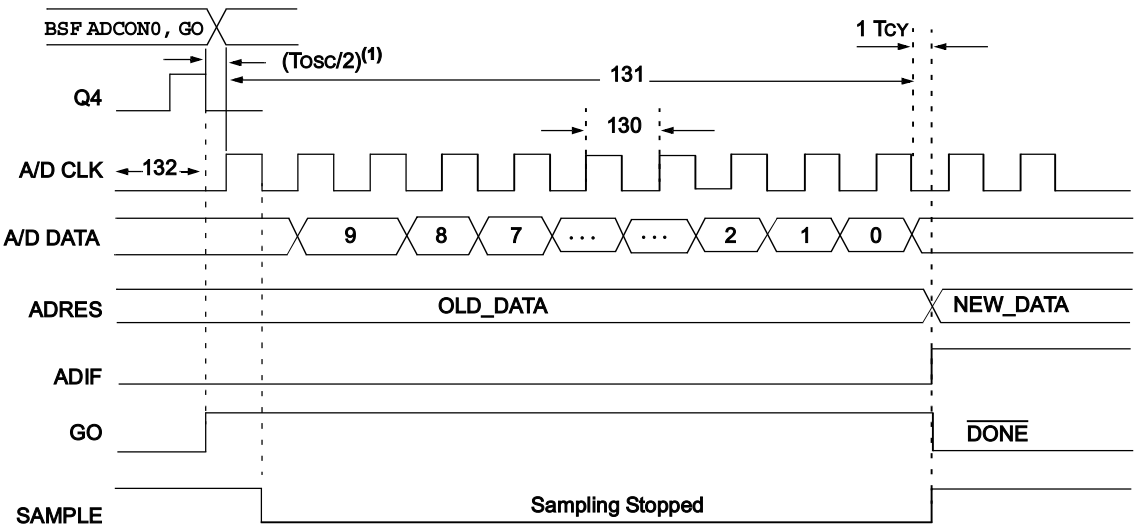
2: VREF current is from RA3 pin or VDD pin, whichever is selected as reference input.

3: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

4: Maximum allowed impedance for analog voltage source is 10 kΩ This requires higher acquisition time.

© 2003 Microchip Technology Inc.

FIGURE 17-19: A/D CONVERSION TIMING



Note: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

TABLE 17-15: A/D CONVERSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
130	TAD	A/D Clock Period	PIC16F87XA	1.6	—	—	µs	TOSC based, VREF ≥ 3.0V
			PIC16LF87XA	3.0	—	—	µs	TOSC based, VREF ≥ 2.0V
			PIC16F87XA	2.0	4.0	6.0	µs	A/D RC mode
			PIC16LF87XA	3.0	6.0	9.0	µs	A/D RC mode
131	TCNV	Conversion Time (not including S/H time) **(Note 1)**			—	12	TAD	
132	TACQ	Acquisition Time		**(Note 2)**	40	—	µs	
				10*	—	—	µs	The minimum time is the amplifier settling time. This may be used if the "new" input voltage has not changed by more than 1 LSb (i.e., 20.0 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).
134	TGO	Q4 to A/D Clock Start		—	TOSC/2 §	—	—	If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

Note 1: ADRES register may be read on the following TCY cycle.

2: See **Section 11.1 "A/D Acquisition Requirements"** for minimum conditions.

NOTES:

© 2003 Microchip Technology Inc.

18.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

Note: The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "Maximum" or "minimum" represents (mean + 3σ) or (mean – 3σ) respectively, where σ is a standard deviation, over the whole temperature range.

FIGURE 18-1: TYPICAL IDD vs. FOSC OVER VDD (HS MODE)

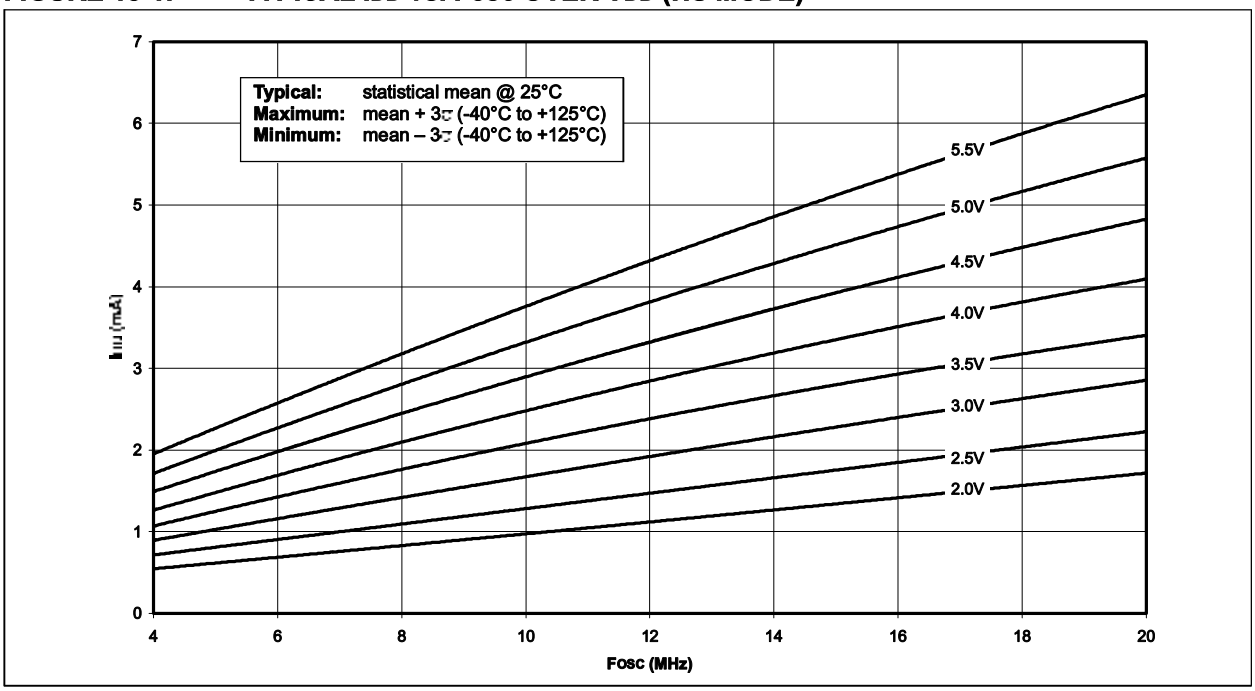


FIGURE 18-2: MAXIMUM IDD vs. FOSC OVER VDD (HS MODE)

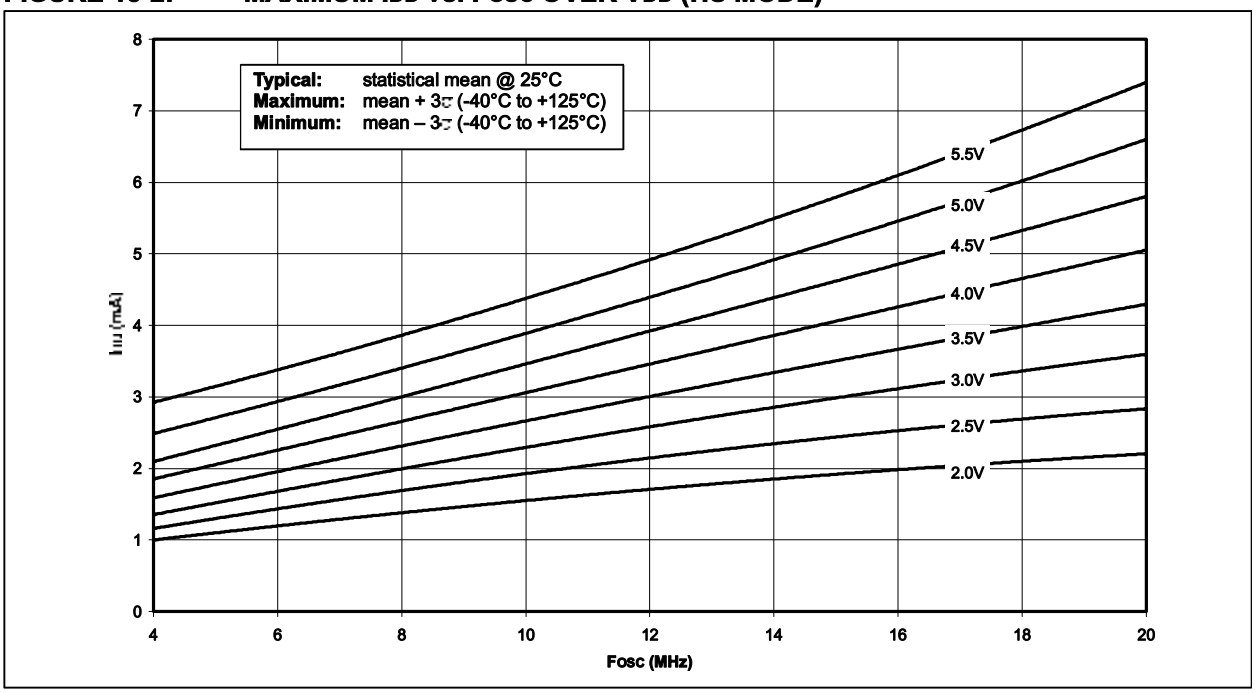


FIGURE 18-3: TYPICAL IDD vs. FOSC OVER VDD (XT MODE)

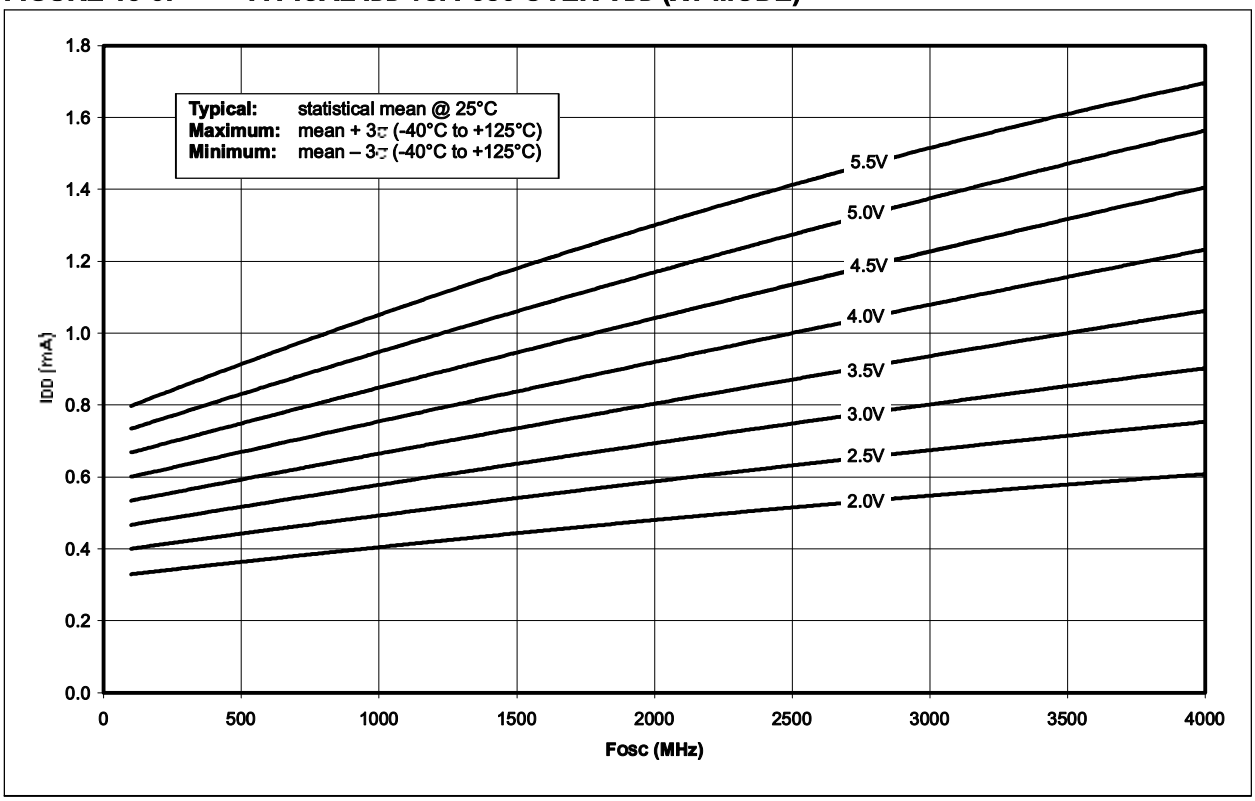
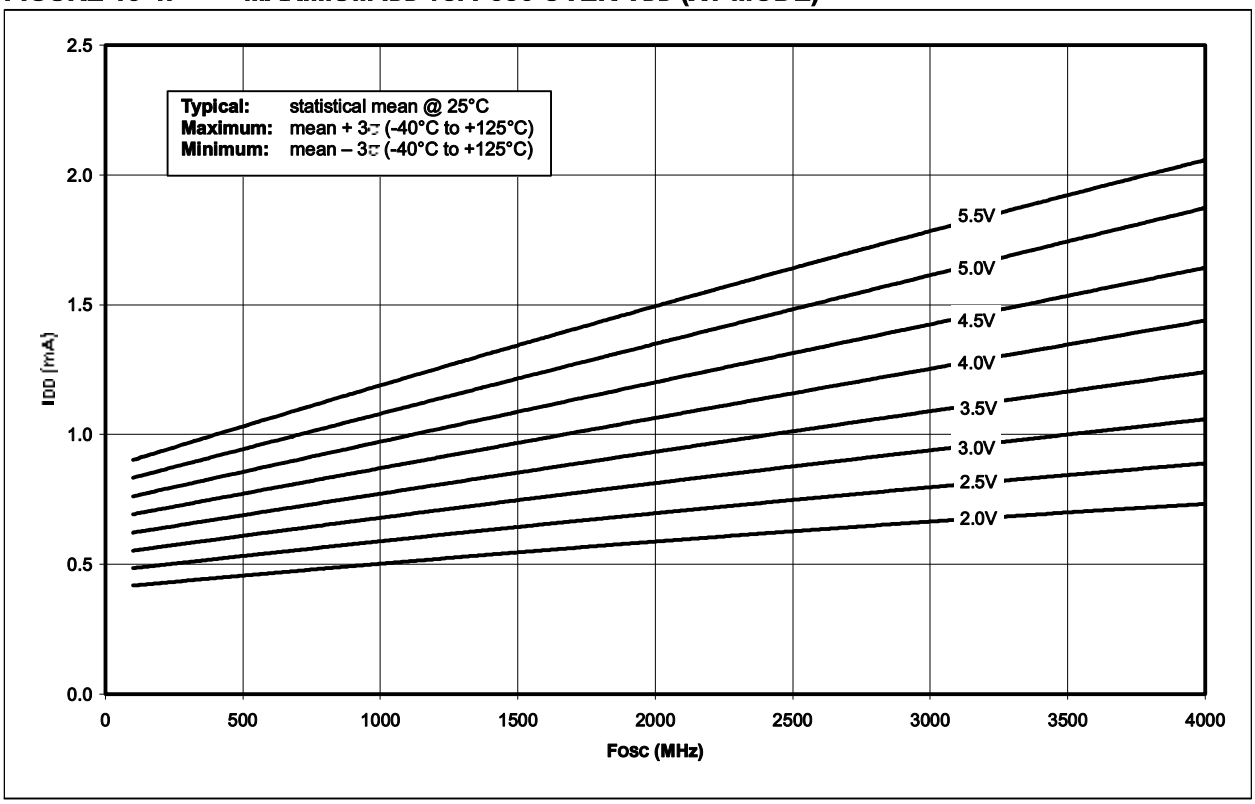


FIGURE 18-4: MAXIMUM IDD vs. FOSC OVER VDD (XT MODE)



© 2003 Microchip Technology Inc.

FIGURE 18-5: TYPICAL IDD vs. FOSC OVER VDD (LP MODE)

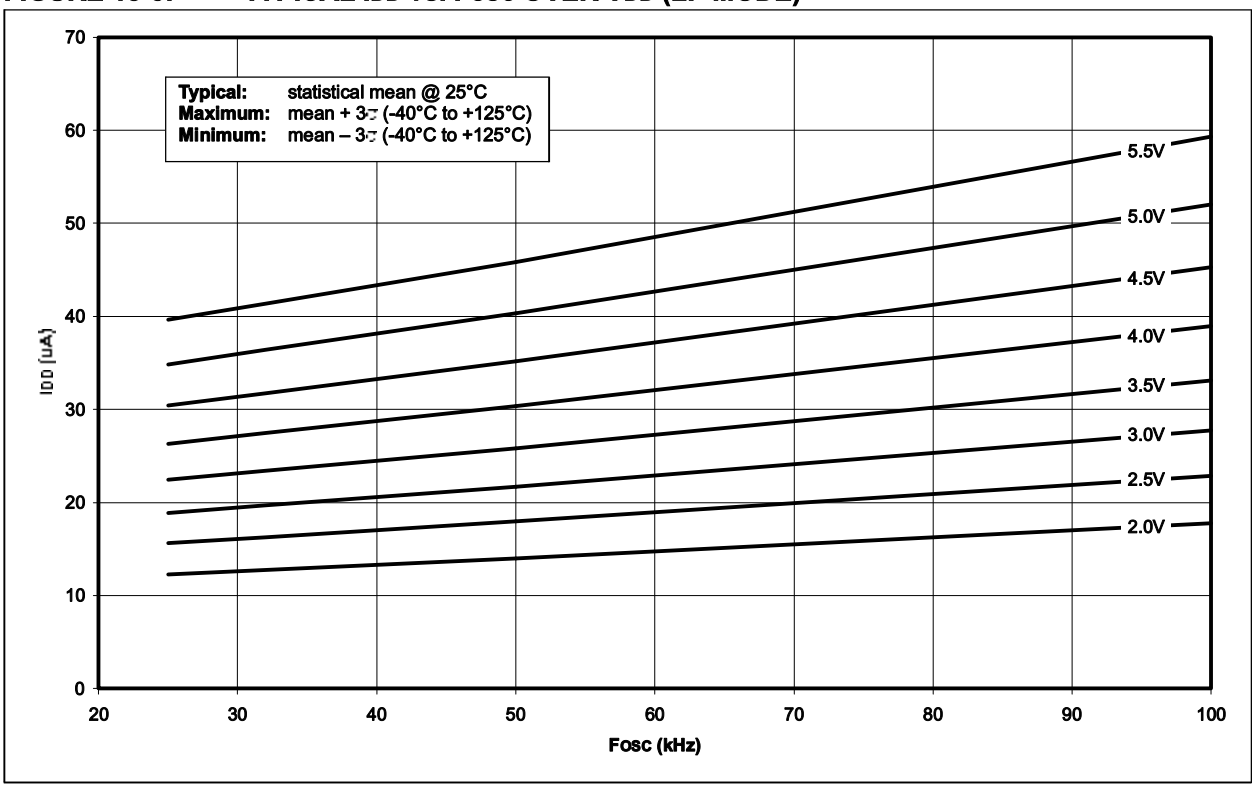


FIGURE 18-6: MAXIMUM IDD vs. FOSC OVER VDD (LP MODE)

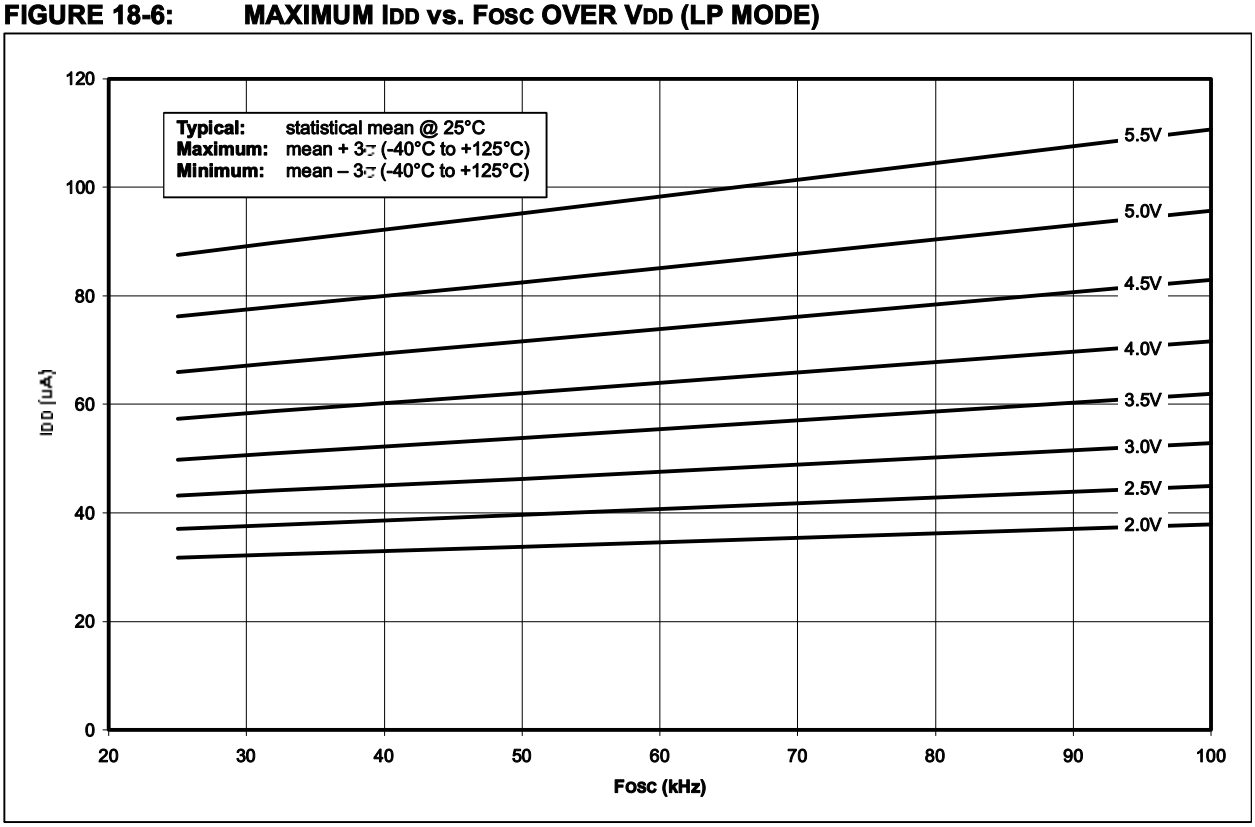


FIGURE 18-7: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 20 pF, +25°C)

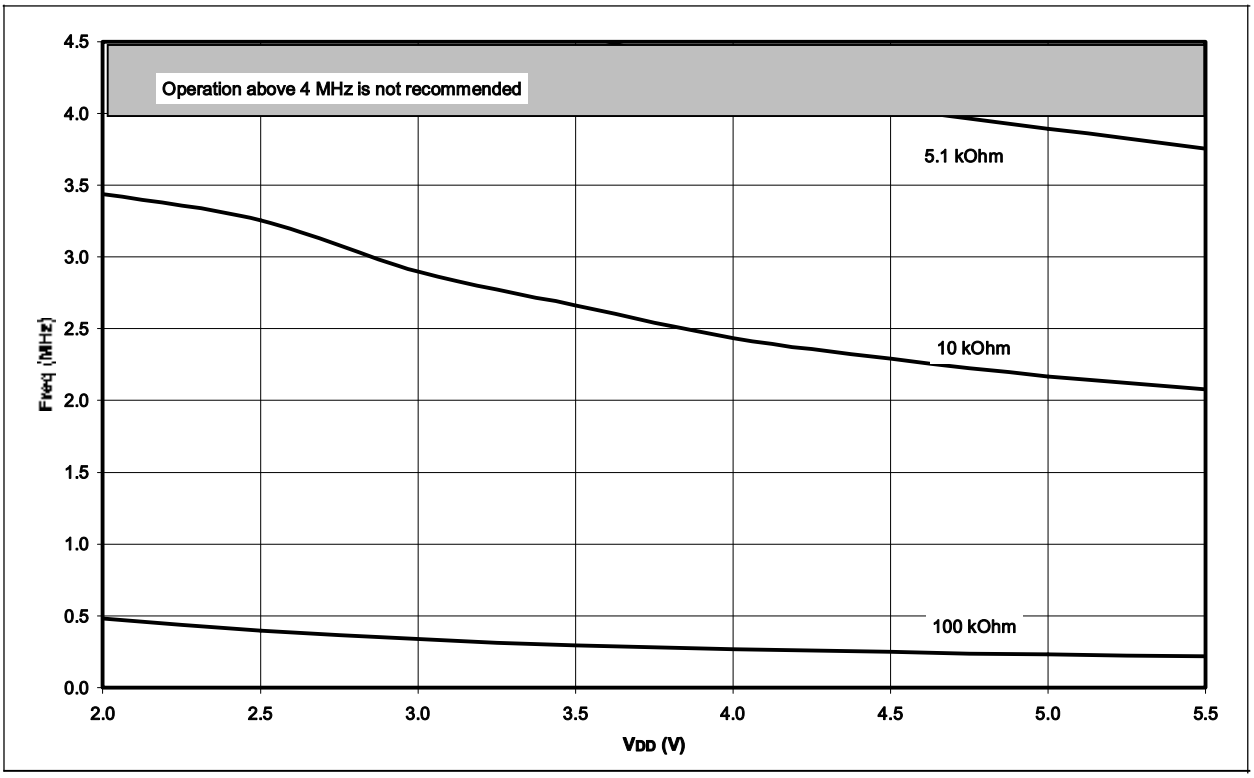
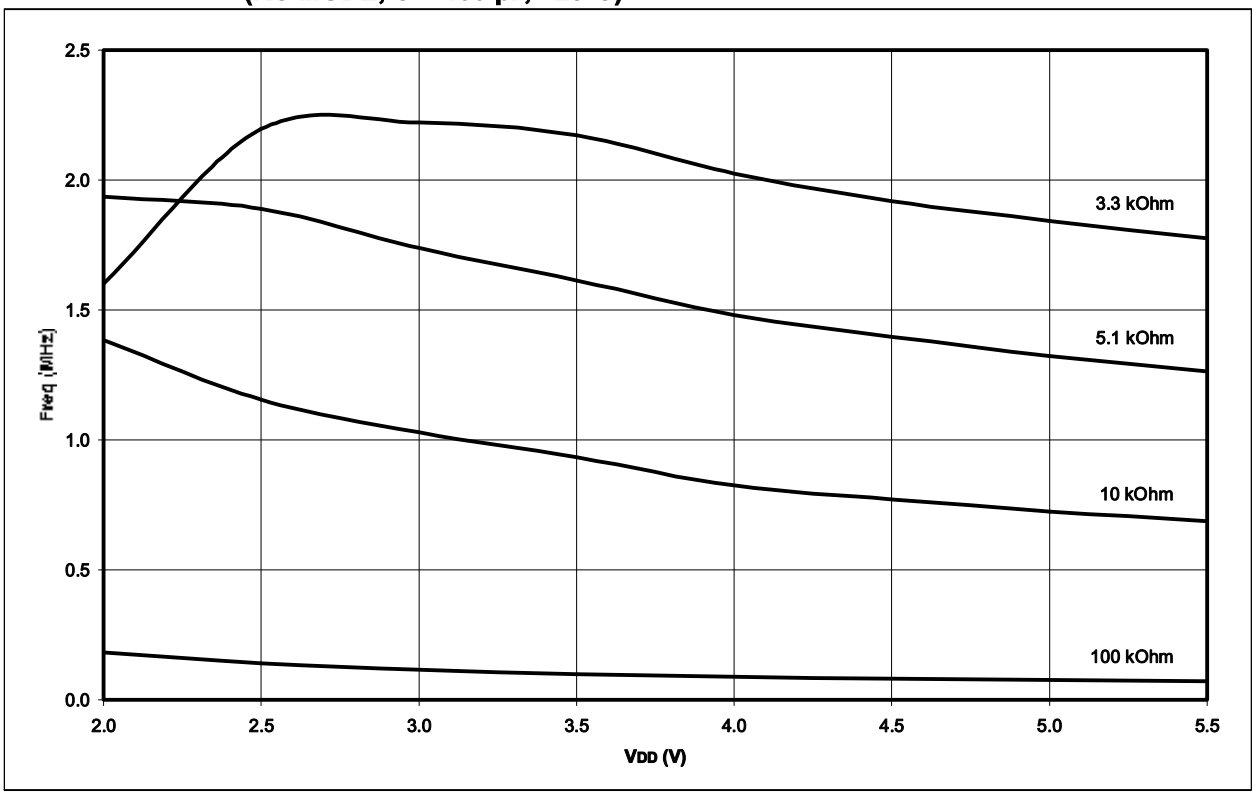


FIGURE 18-8: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 100 pF, +25°C)



© 2003 Microchip Technology Inc.

FIGURE 18-9: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 300 pF, +25°C)

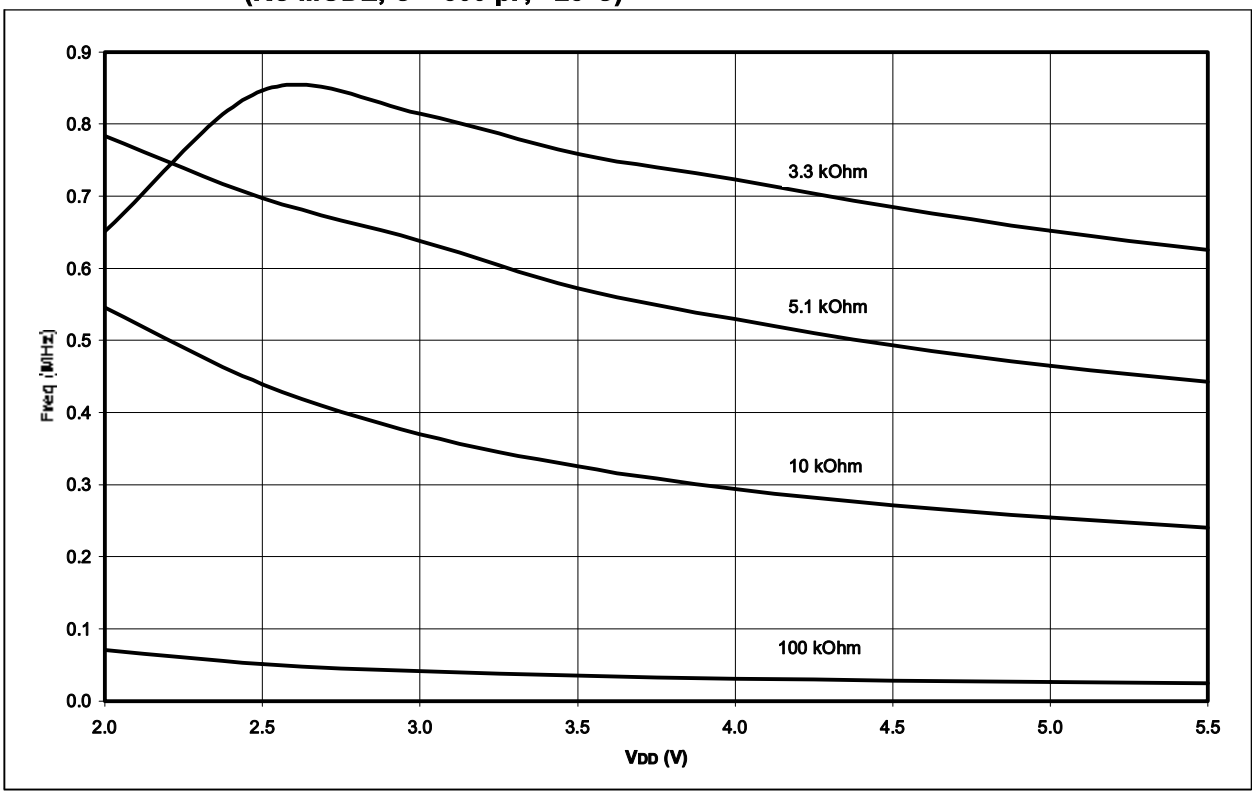


FIGURE 18-10: IPD vs. VDD, -40°C TO +125°C (SLEEP MODE, ALL PERIPHERALS DISABLED)

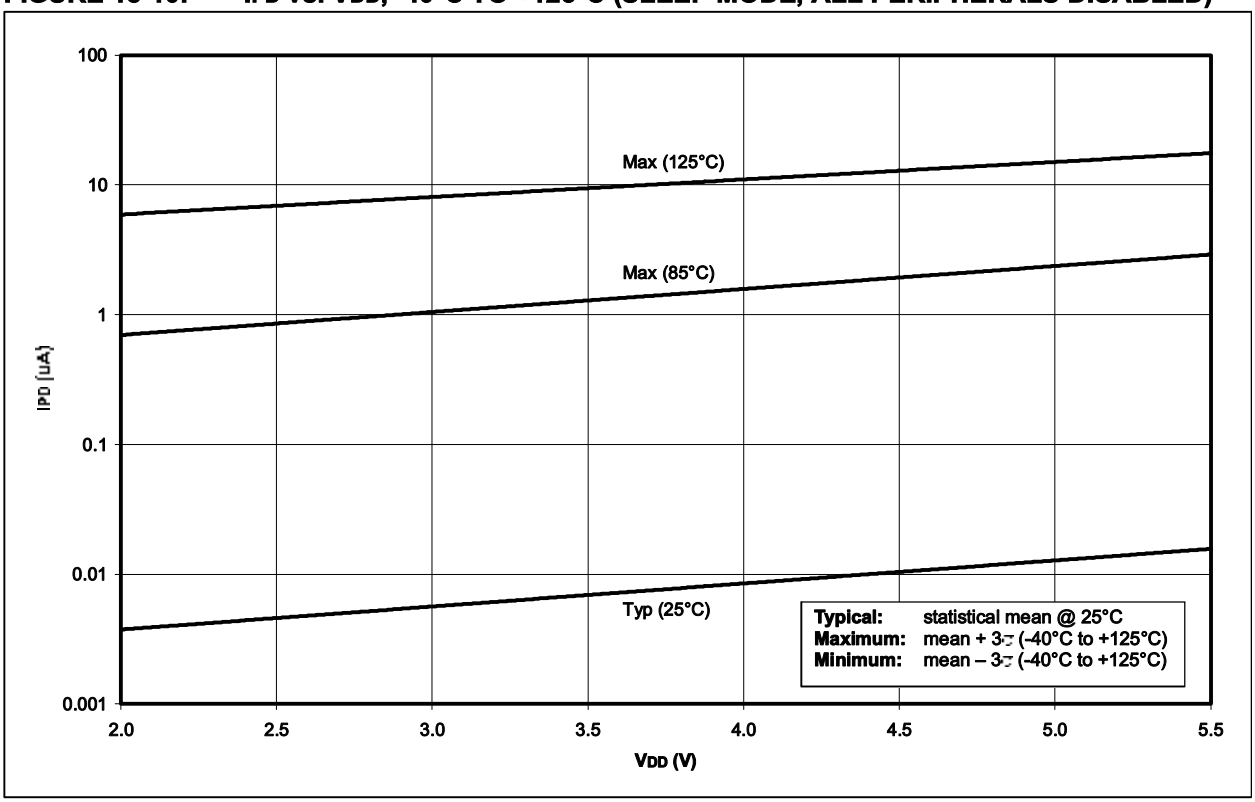


FIGURE 18-11: TYPICAL AND MAXIMUM ΔITMR1 vs. VDD OVER TEMPERATURE (-10°C TO +70°C, TIMER1 WITH OSCILLATOR, XTAL = 32 kHz, C1 AND C2 = 47 pF)

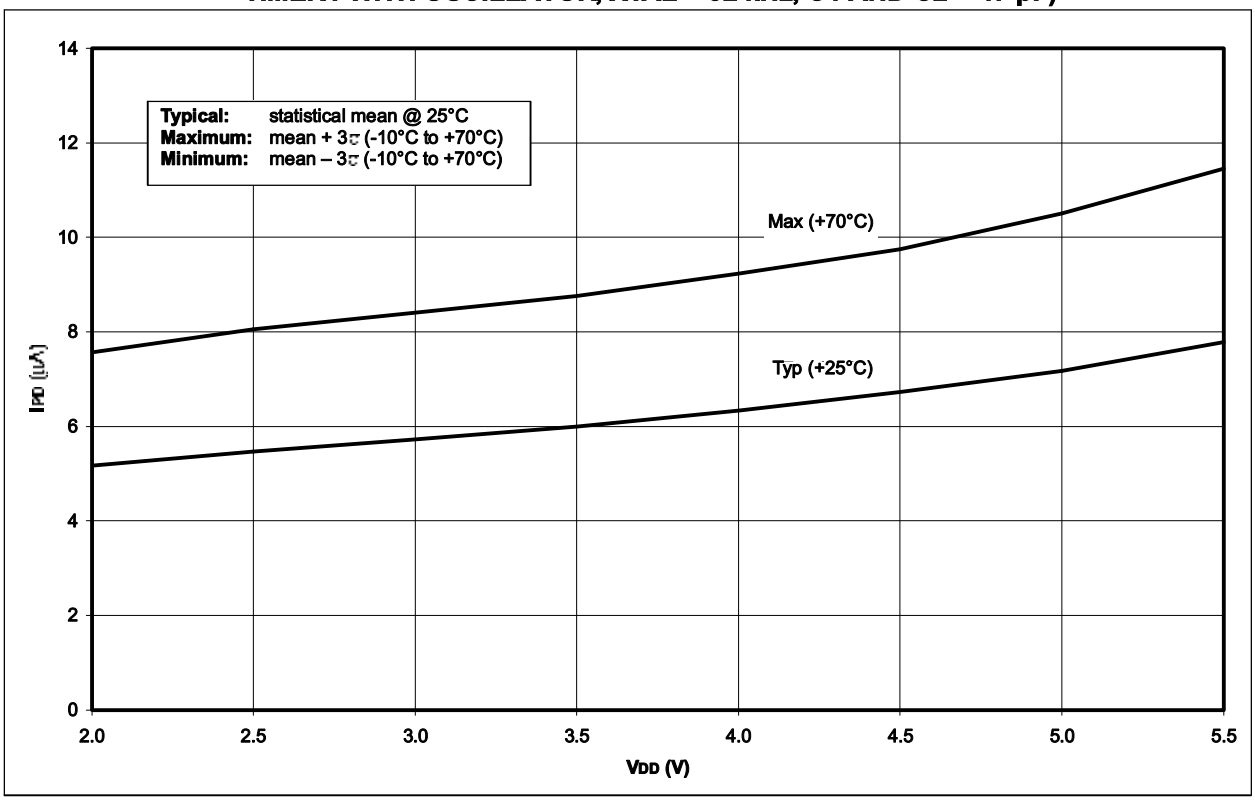
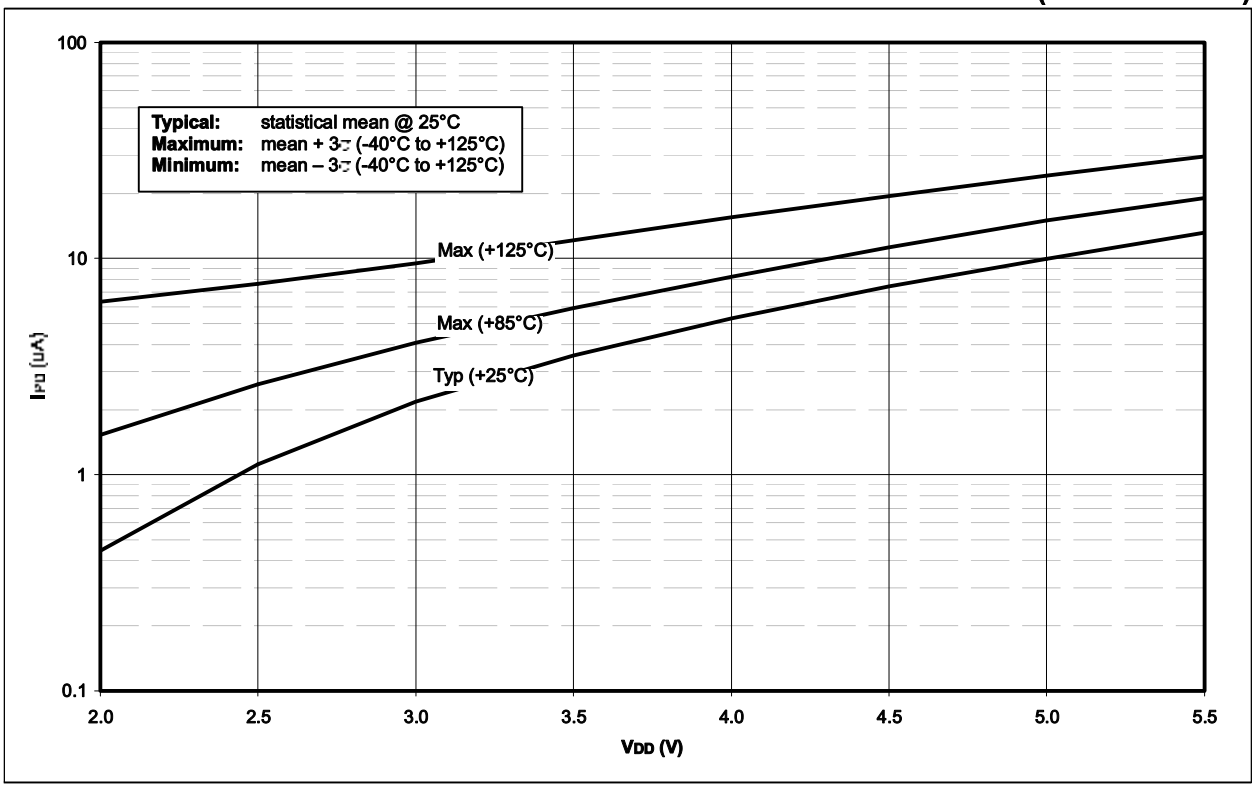


FIGURE 18-12: TYPICAL AND MAXIMUM ΔIWDT vs. VDD OVER TEMPERATURE (WDT ENABLED)



© 2003 Microchip Technology Inc.

FIGURE 18-13: ΔIBOR vs. VDD OVER TEMPERATURE

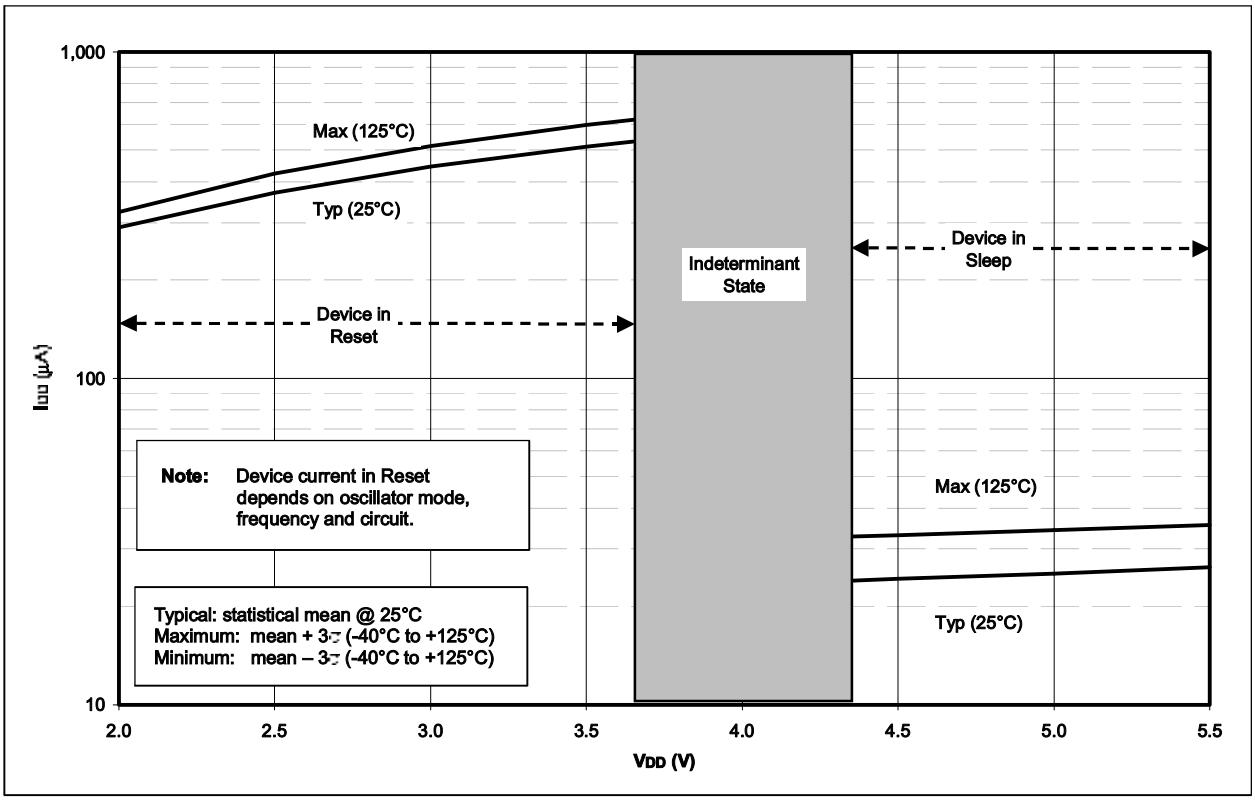


FIGURE 18-14: TYPICAL, MINIMUM AND MAXIMUM WDT PERIOD vs. VDD (-40°C TO +125°C)

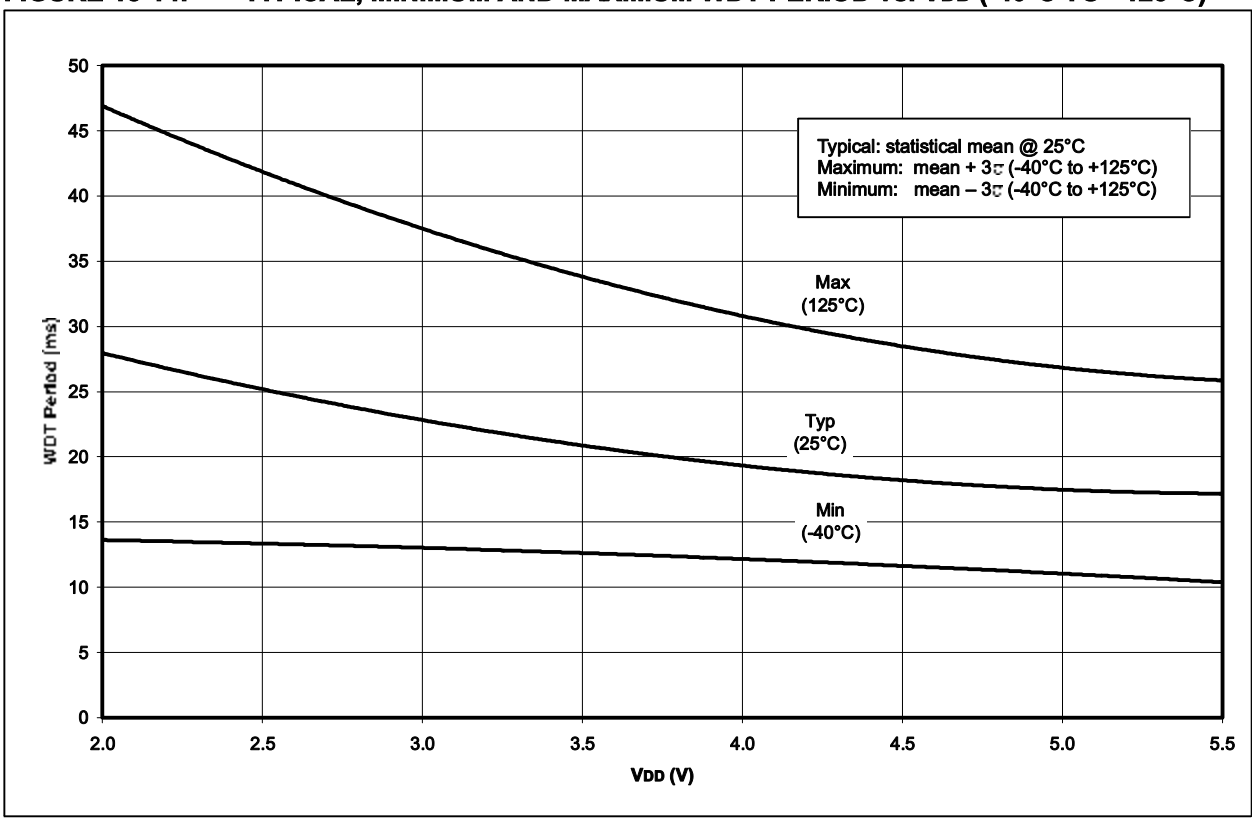


FIGURE 18-15: AVERAGE WDT PERIOD vs. VDD OVER TEMPERATURE (-40°C TO +125°C)

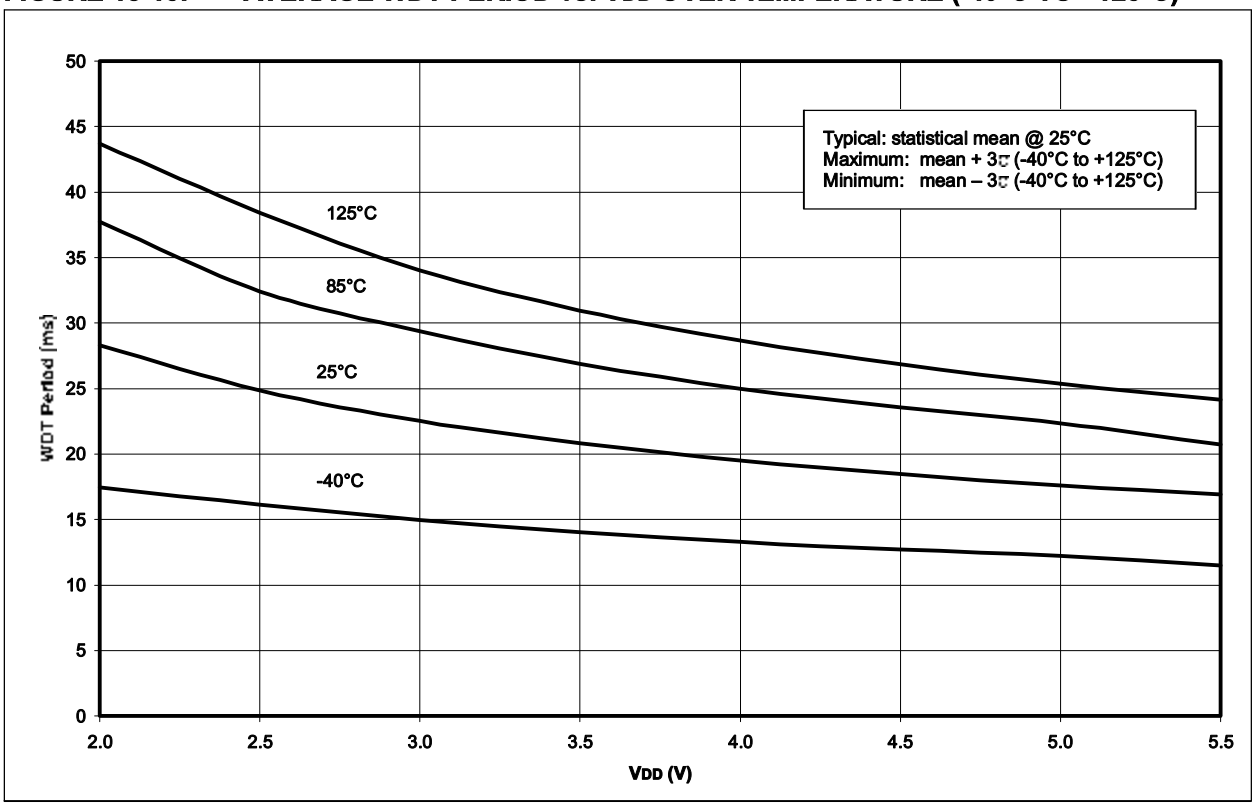
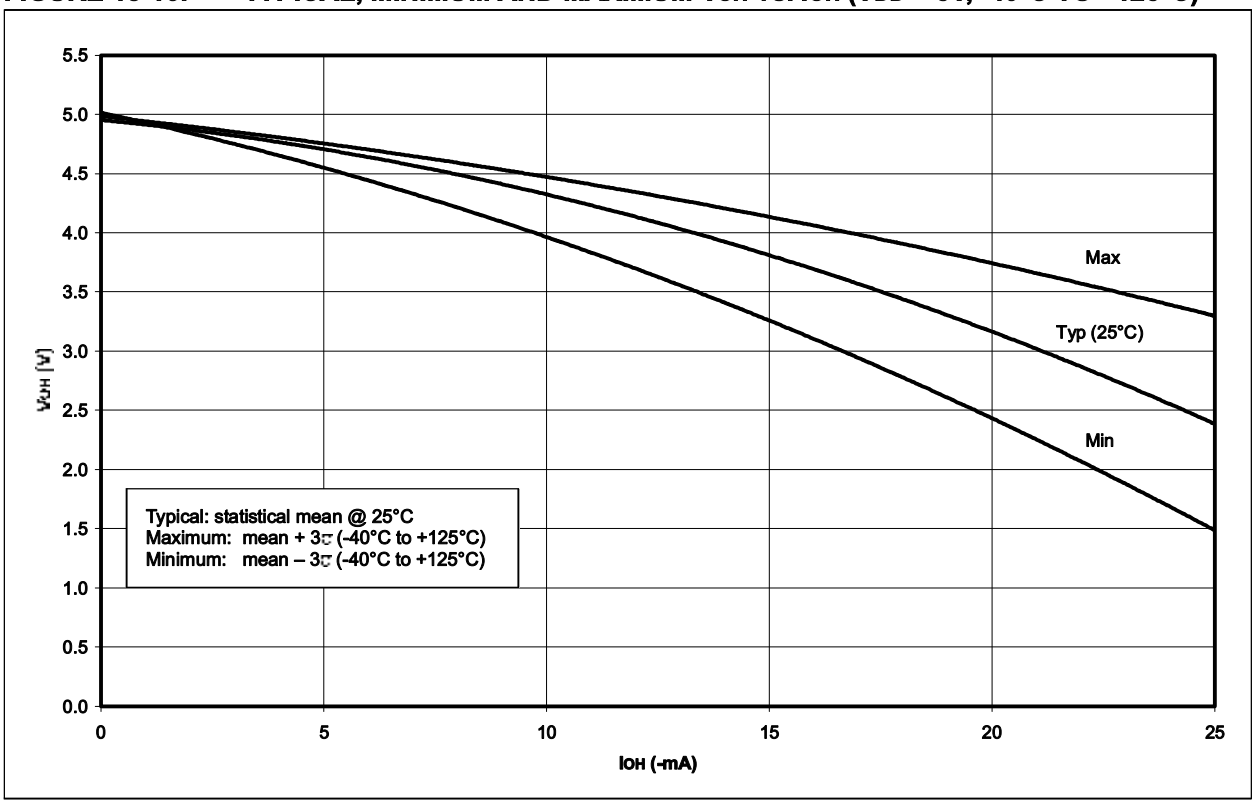


FIGURE 18-16: TYPICAL, MINIMUM AND MAXIMUM VOH vs. IOH (VDD = 5V, -40°C TO +125°C)



© 2003 Microchip Technology Inc.

FIGURE 18-17: TYPICAL, MINIMUM AND MAXIMUM VOH vs. IOH (VDD = 3V, -40°C TO +125°C)

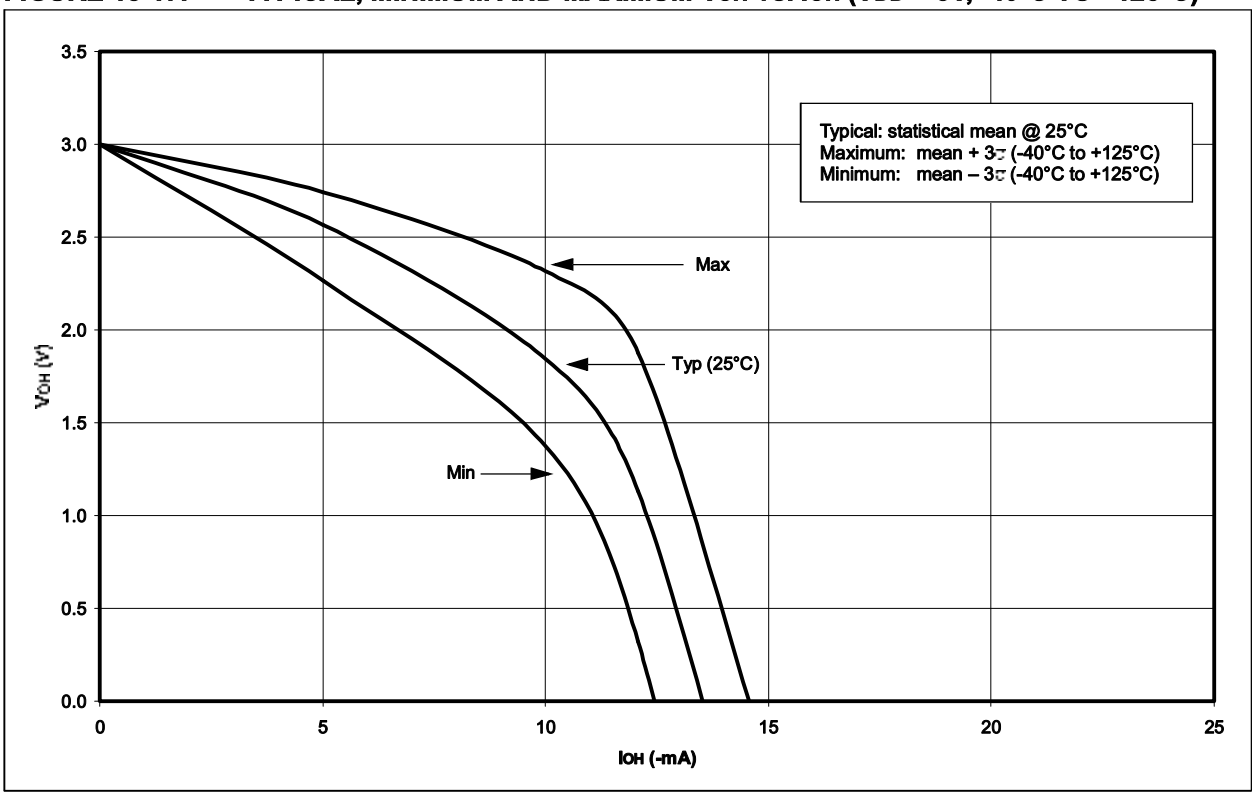


FIGURE 18-18: TYPICAL, MINIMUM AND MAXIMUM VOL vs. IOL (VDD = 5V, -40°C TO +125°C)

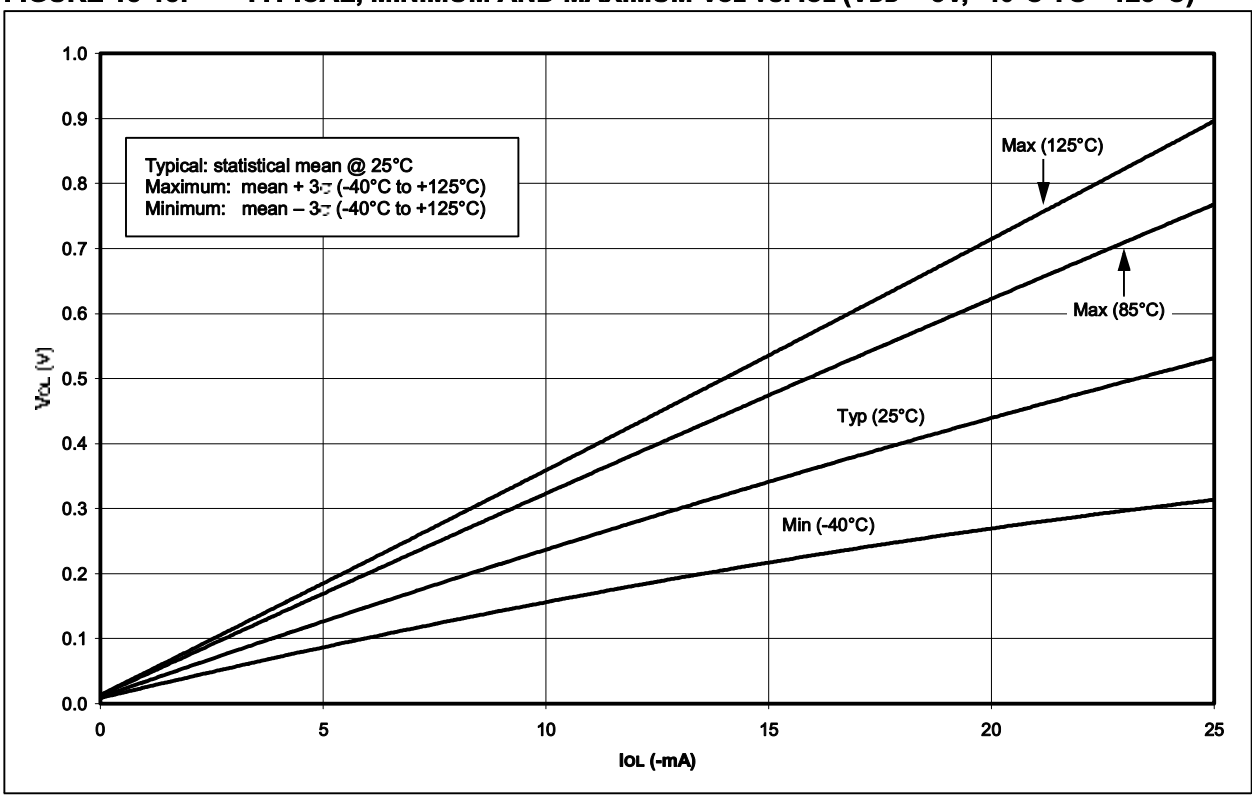


FIGURE 18-19: TYPICAL, MINIMUM AND MAXIMUM VOL vs. IOL (VDD = 3V, -40°C TO +125°C)

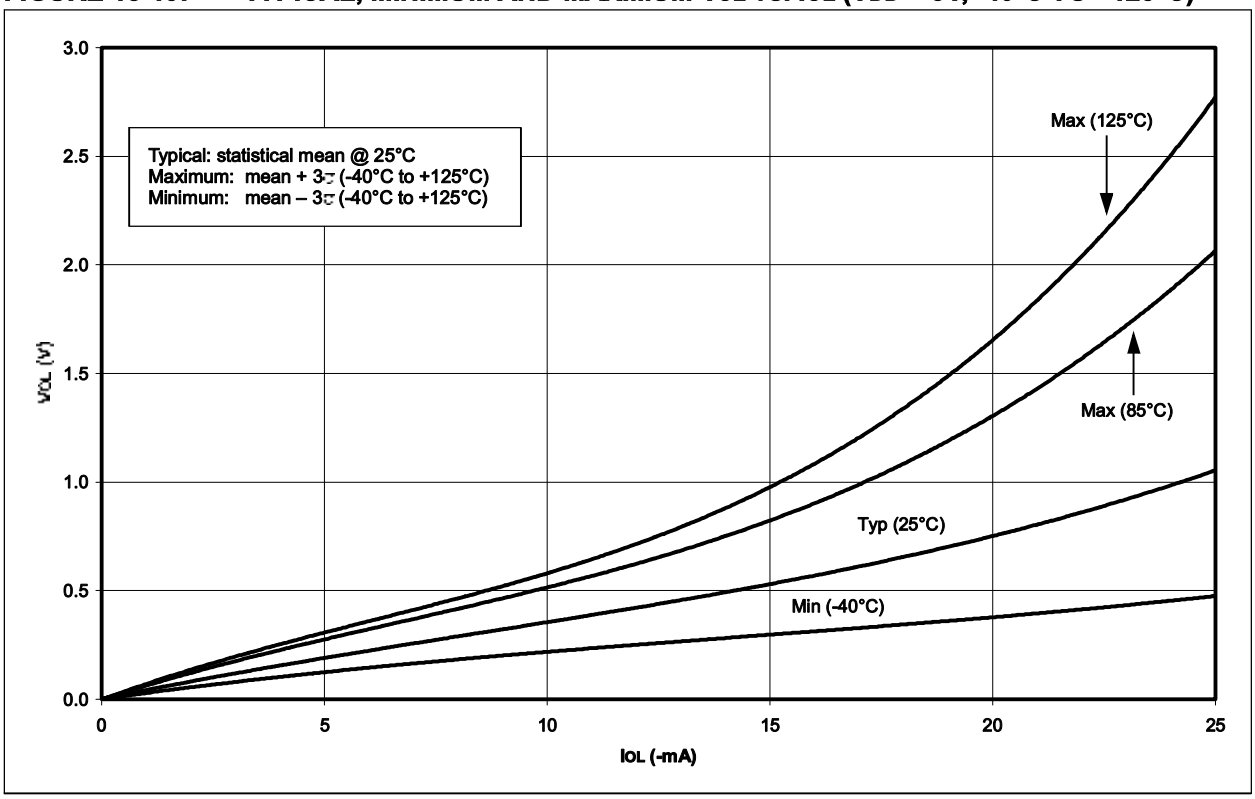
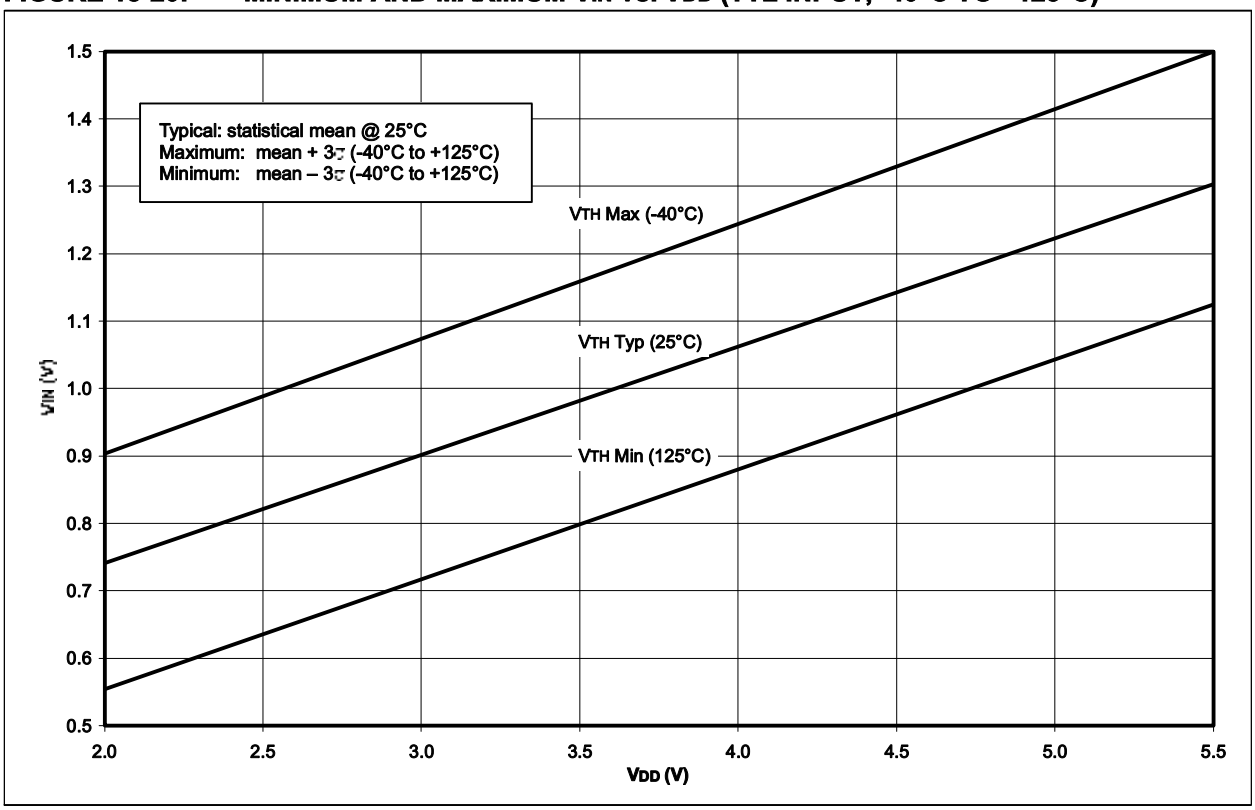


FIGURE 18-20: MINIMUM AND MAXIMUM VIN vs. VDD (TTL INPUT, -40°C TO +125°C)



© 2003 Microchip Technology Inc.

FIGURE 18-21: MINIMUM AND MAXIMUM VIN vs. VDD (ST INPUT, -40°C TO +125°C)

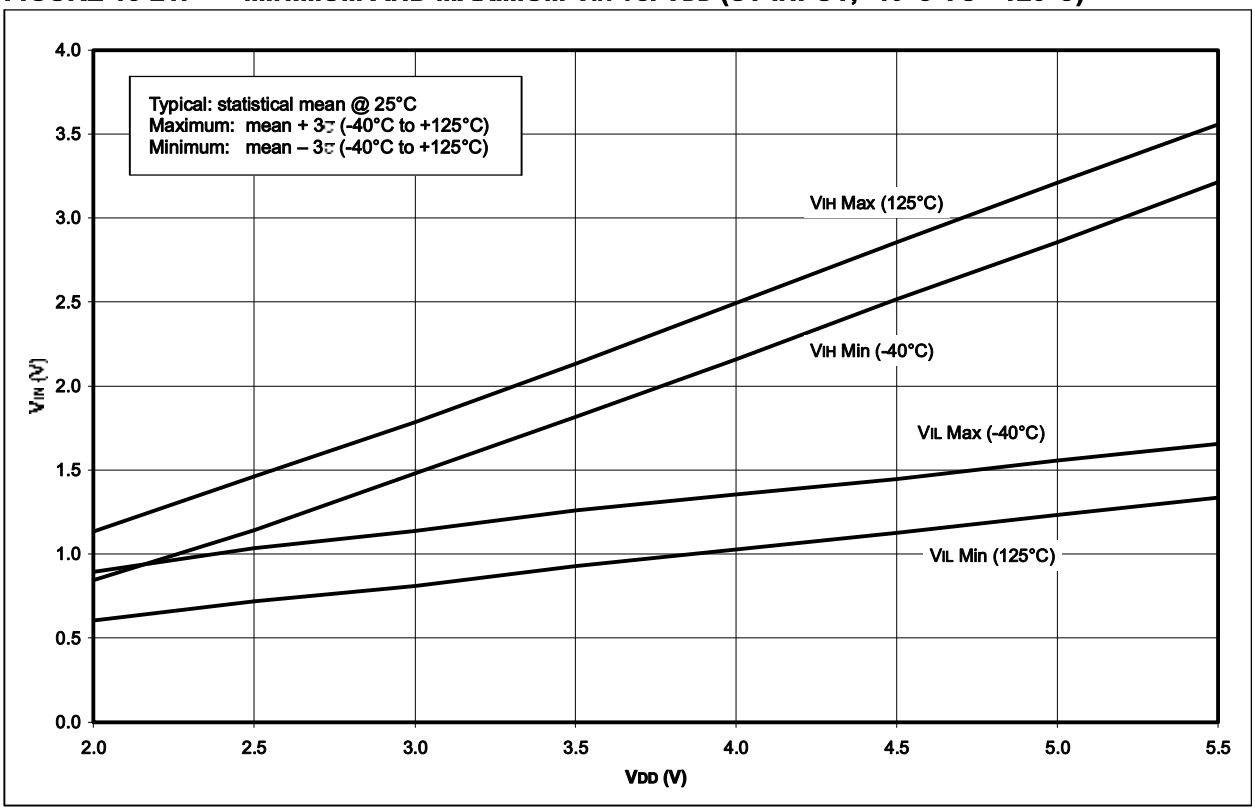


FIGURE 18-22: MINIMUM AND MAXIMUM VIN vs. VDD (I^2C INPUT, -40°C TO +125°C)

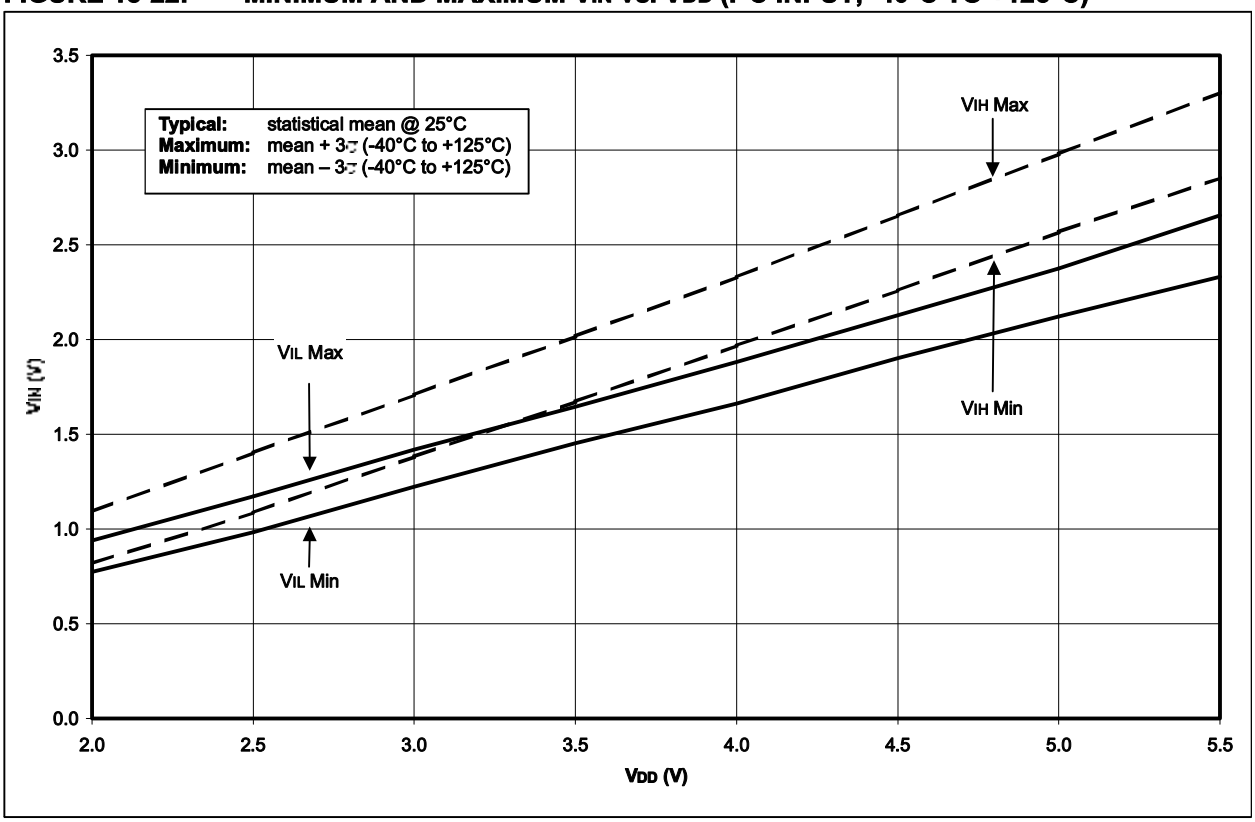


FIGURE 18-23: A/D NONLINEARITY vs. VREFH (VDD = VREFH, -40°C TO +125°C)

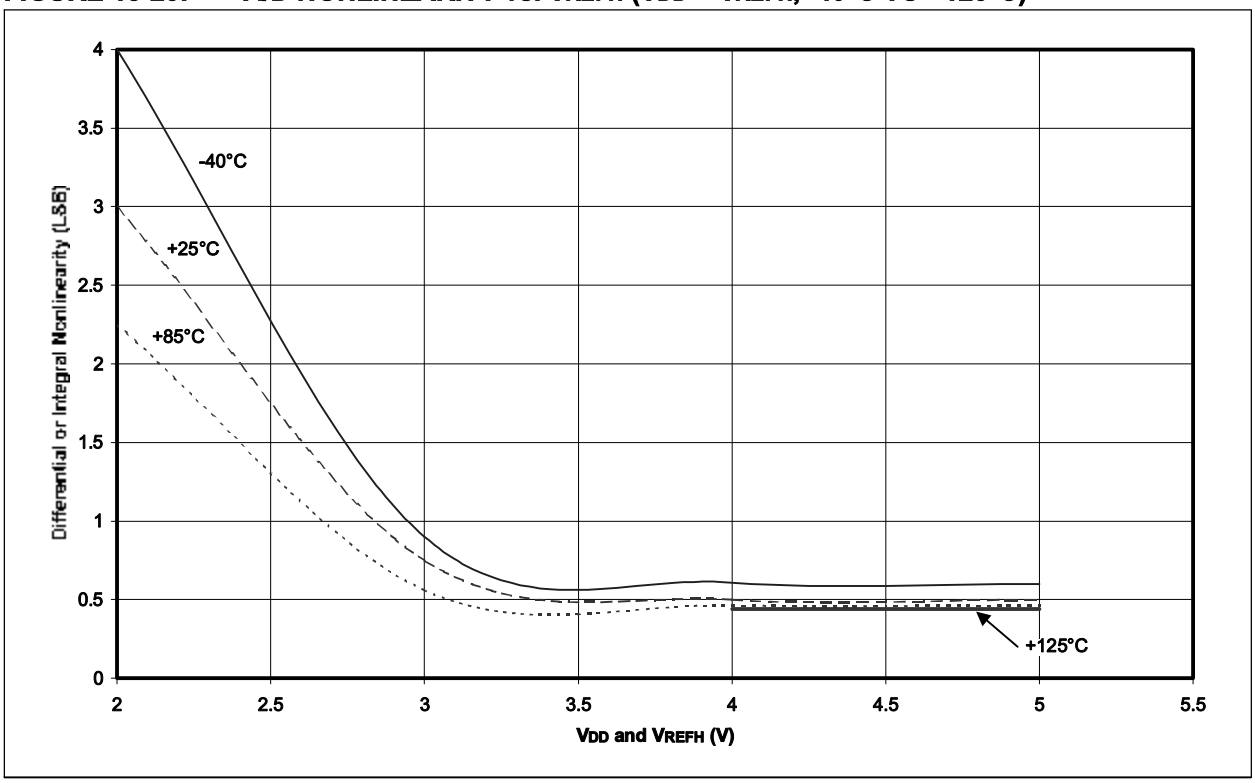
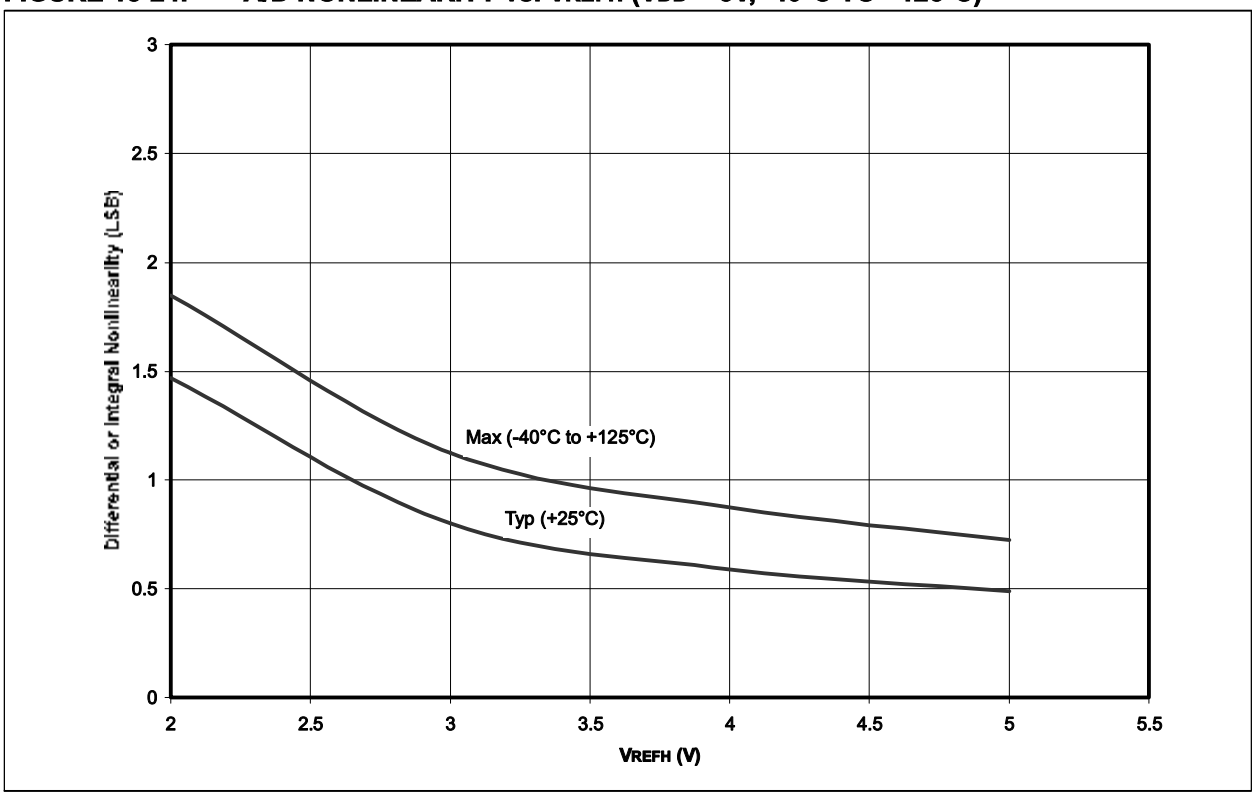


FIGURE 18-24: A/D NONLINEARITY vs. VREFH (VDD = 5V, -40°C TO +125°C)

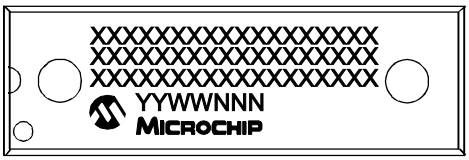


© 2003 Microchip Technology Inc.

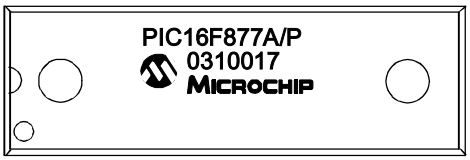
19.0 PACKAGING INFORMATION

19.1 Package Marking Information

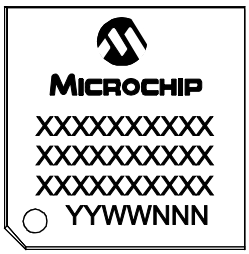
40-Lead PDIP



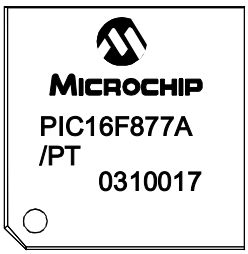
Example



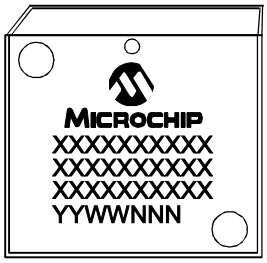
44-Lead TQFP



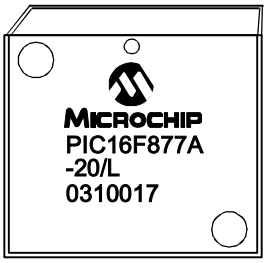
Example



44-Lead PLCC



Example



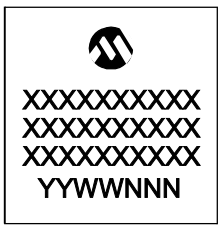
Legend:	XX...X	Customer specific information*
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

* Standard PICmicro device marking consists of Microchip part number, year code, week code, and traceability code. For PICmicro device marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

Package Marking Information (Cont'd)

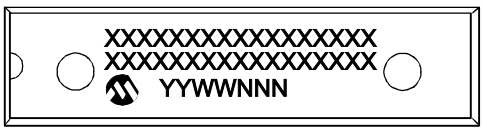
44-Lead QFN



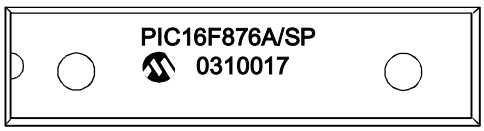
Example



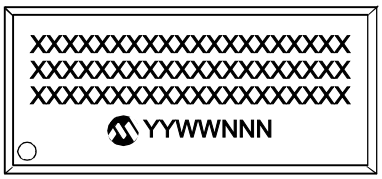
28-Lead PDIP (Skinny DIP)



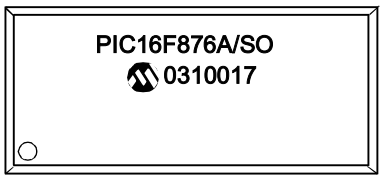
Example



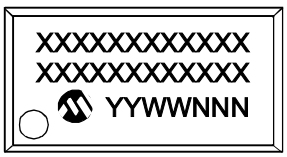
28-Lead SOIC



Example



28-Lead SSOP



Example



28-Lead QFN

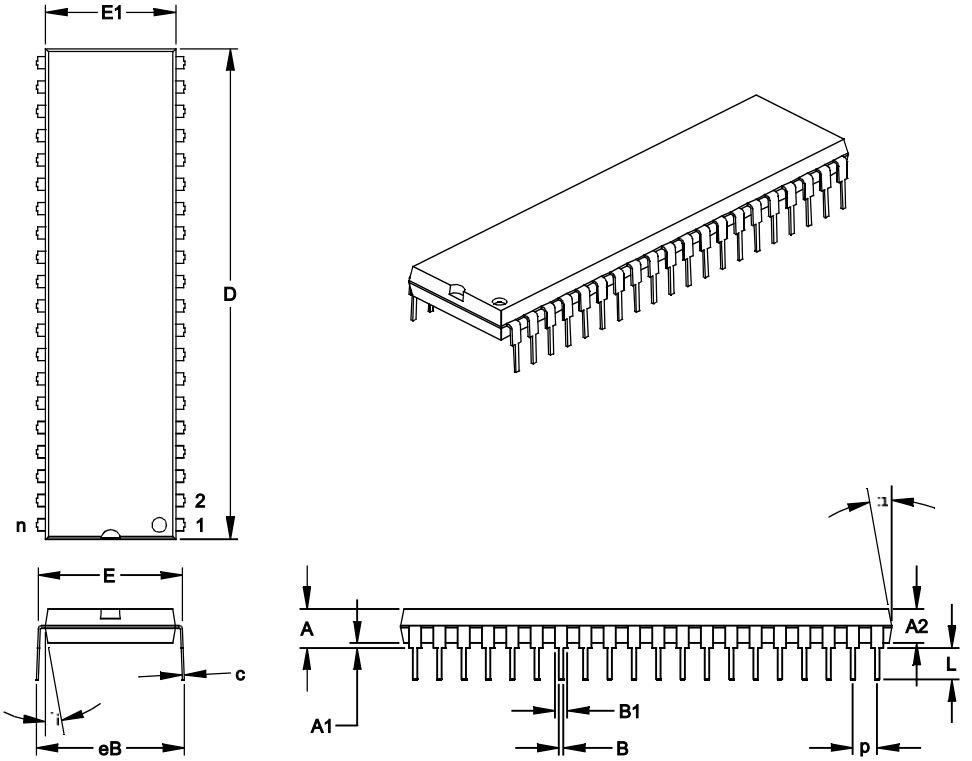


Example



© 2003 Microchip Technology Inc.

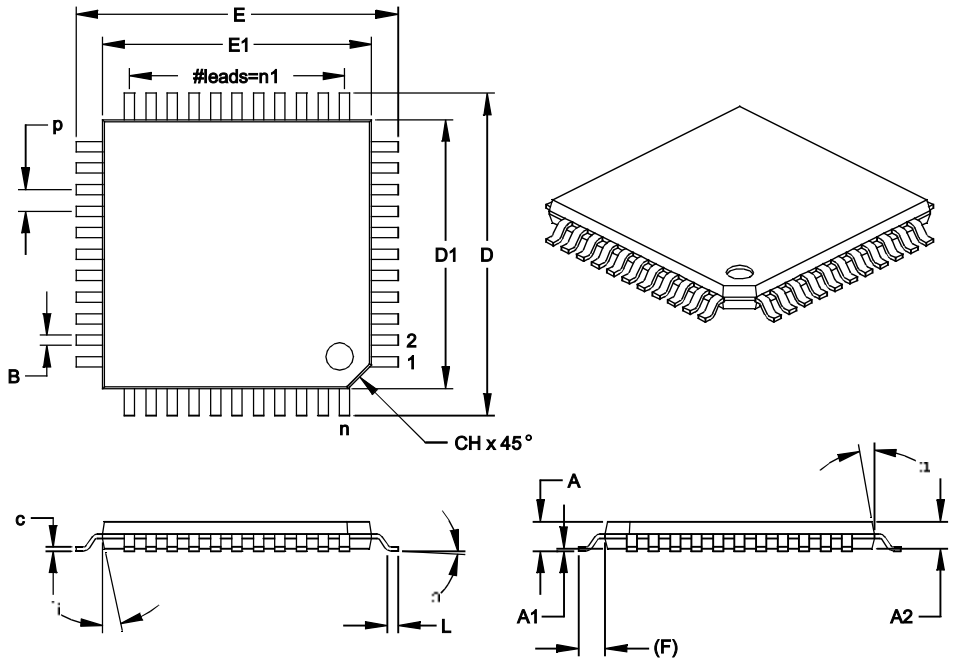
40-Lead Plastic Dual In-line (P) – 600 mil (PDIP)



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		40			40	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.160	.175	.190	4.06	4.45	4.83
Molded Package Thickness	A2	.140	.150	.160	3.56	3.81	4.06
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.595	.600	.625	15.11	15.24	15.88
Molded Package Width	E1	.530	.545	.560	13.46	13.84	14.22
Overall Length	D	2.045	2.058	2.065	51.94	52.26	52.45
Tip to Seating Plane	L	.120	.130	.135	3.05	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.030	.050	.070	0.76	1.27	1.78
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56
Overall Row Spacing §	eB	.620	.650	.680	15.75	16.51	17.27
Mold Draft Angle Top	:1	5	10	15	5	10	15
Mold Draft Angle Bottom	¨i	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-011
Drawing No. C04-016

44-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)



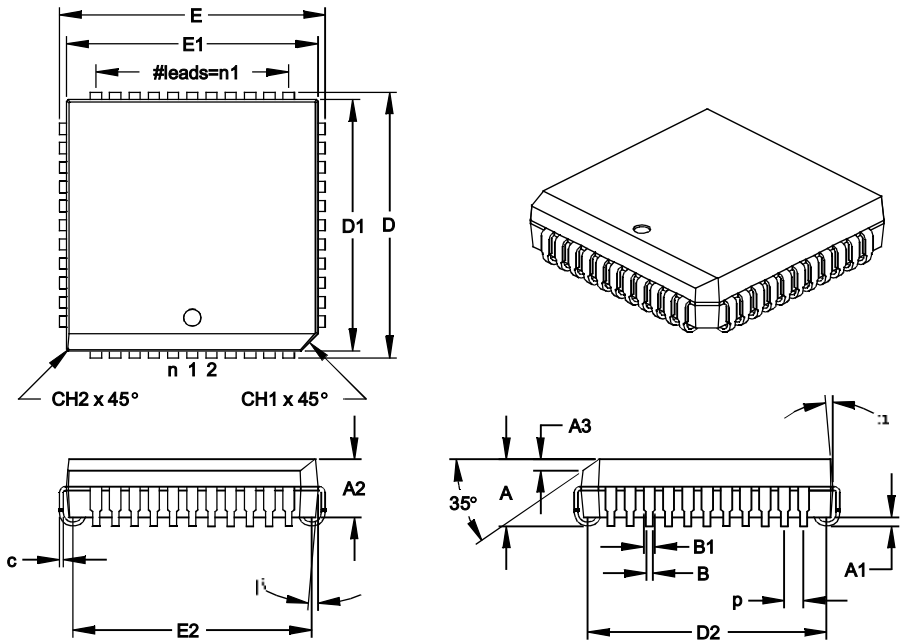
Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.031			0.80	
Pins per Side	n1		11			11	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.004	.006	0.05	0.10	0.15
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039		1.00		
Foot Angle	φ	0	3.5	7	0	3.5	7
Overall Width	E	.463	.472	.482	11.75	12.00	12.25
Overall Length	D	.463	.472	.482	11.75	12.00	12.25
Molded Package Width	E1	.390	.394	.398	9.90	10.00	10.10
Molded Package Length	D1	.390	.394	.398	9.90	10.00	10.10
Lead Thickness	c	.004	.006	.008	0.09	0.15	0.20
Lead Width	B	.012	.015	.017	0.30	0.38	0.44
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-076

© 2003 Microchip Technology Inc.

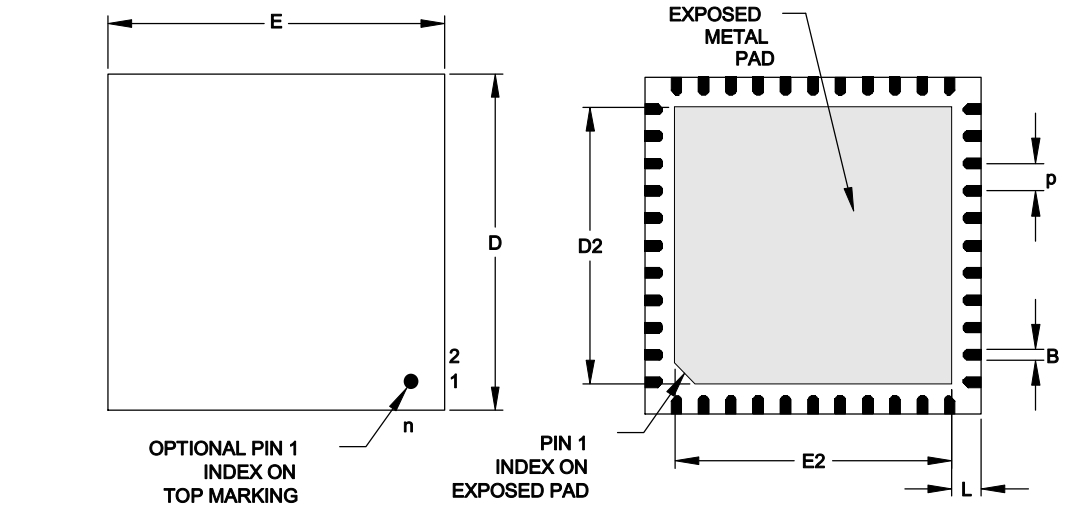
44-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)



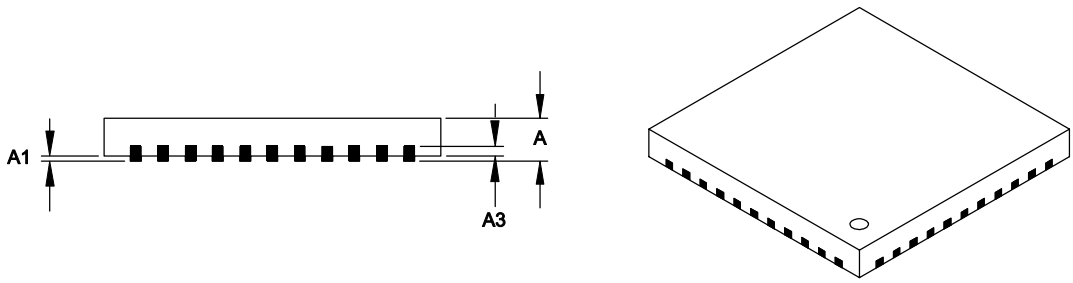
Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.050			1.27	
Pins per Side	n1		11			11	
Overall Height	A	.165	.173	.180	4.19	4.39	4.57
Molded Package Thickness	A2	.145	.153	.160	3.68	3.87	4.06
Standoff §	A1	.020	.028	.035	0.51	0.71	0.89
Side 1 Chamfer Height	A3	.024	.029	.034	0.61	0.74	0.86
Corner Chamfer 1	CH1	.040	.045	.050	1.02	1.14	1.27
Corner Chamfer (others)	CH2	.000	.005	.010	0.00	0.13	0.25
Overall Width	E	.685	.690	.695	17.40	17.53	17.65
Overall Length	D	.685	.690	.695	17.40	17.53	17.65
Molded Package Width	E1	.650	.653	.656	16.51	16.59	16.66
Molded Package Length	D1	.650	.653	.656	16.51	16.59	16.66
Footprint Width	E2	.590	.620	.630	14.99	15.75	16.00
Footprint Length	D2	.590	.620	.630	14.99	15.75	16.00
Lead Thickness	c	.008	.011	.013	0.20	0.27	0.33
Upper Lead Width	B1	.026	.029	.032	0.66	0.74	0.81
Lower Lead Width	B	.013	.020	.021	0.33	0.51	0.53
Mold Draft Angle Top	α1	0	5	10	0	5	10
Mold Draft Angle Bottom	β1	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-048

44-Lead Plastic Quad Flat No Lead Package (ML) 8x8 mm Body (QFN)



TOP VIEW BOTTOM VIEW



Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.026 BSC			0.65 BSC	
Overall Height	A	.031	.035	.039	0.80	0.90	1.00
Standoff	A1	.000	.001	.002	0	0.02	0.05
Base Thickness	A3		.010 REF			0.25 REF	
Overall Width	E		.315 BSC			8.00 BSC	
Exposed Pad Width	E2	.262	.268	.274	6.65	6.80	6.95
Overall Length	D		.315 BSC			8.00 BSC	
Exposed Pad Length	D2	.262	.268	.274	6.65	6.80	6.95
Lead Width	B	.012	.013	.013	0.30	0.33	0.35
Lead Length	L	.014	.016	.018	0.35	0.40	0.45

*Controlling Parameter

Notes:

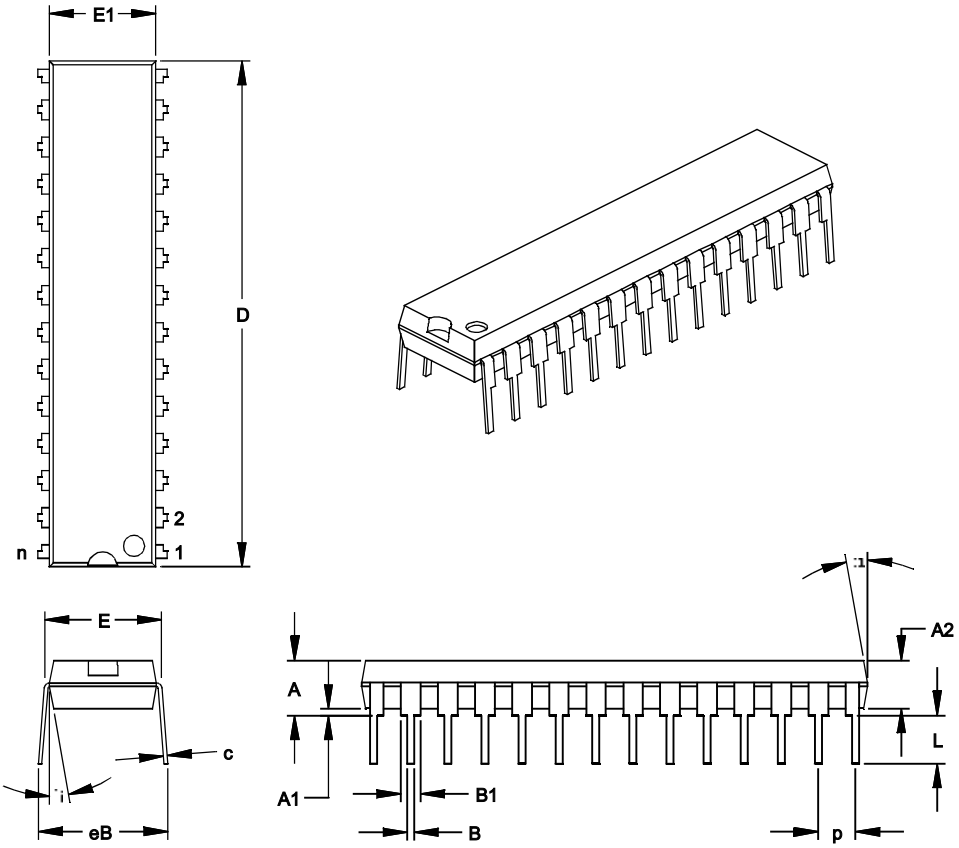
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

JEDEC equivalent: M0-220

Drawing No. C04-103

© 2003 Microchip Technology Inc.

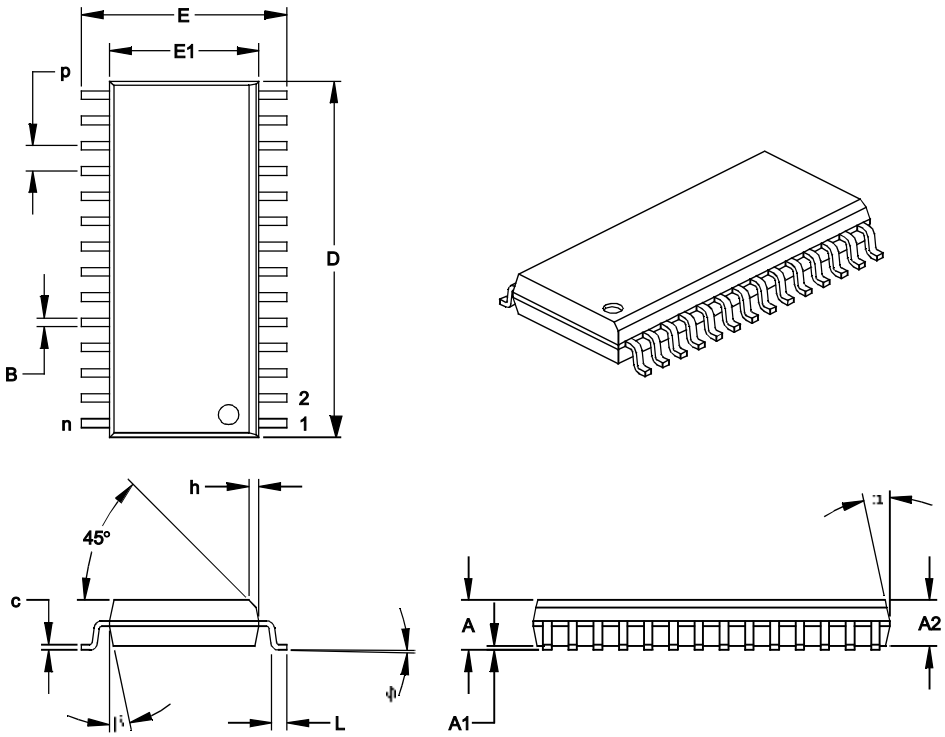
28-Lead Skinny Plastic Dual In-line (SP) – 300 mil (PDIP)



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.140	.150	.160	3.56	3.81	4.06
Molded Package Thickness	A2	.125	.130	.135	3.18	3.30	3.43
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.300	.310	.325	7.62	7.87	8.26
Molded Package Width	E1	.275	.285	.295	6.99	7.24	7.49
Overall Length	D	1.345	1.365	1.385	34.16	34.67	35.18
Tip to Seating Plane	L	.125	.130	.135	3.18	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.040	.053	.065	1.02	1.33	1.65
Lower Lead Width	B	.016	.019	.022	0.41	0.48	0.56
Overall Row Spacing §	eB	.320	.350	.430	8.13	8.89	10.92
Mold Draft Angle Top		5	10	15	5	10	15
Mold Draft Angle Bottom		5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimension D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-095
Drawing No. C04-070

28-Lead Plastic Small Outline (SO) – Wide, 300 mil (SOIC)



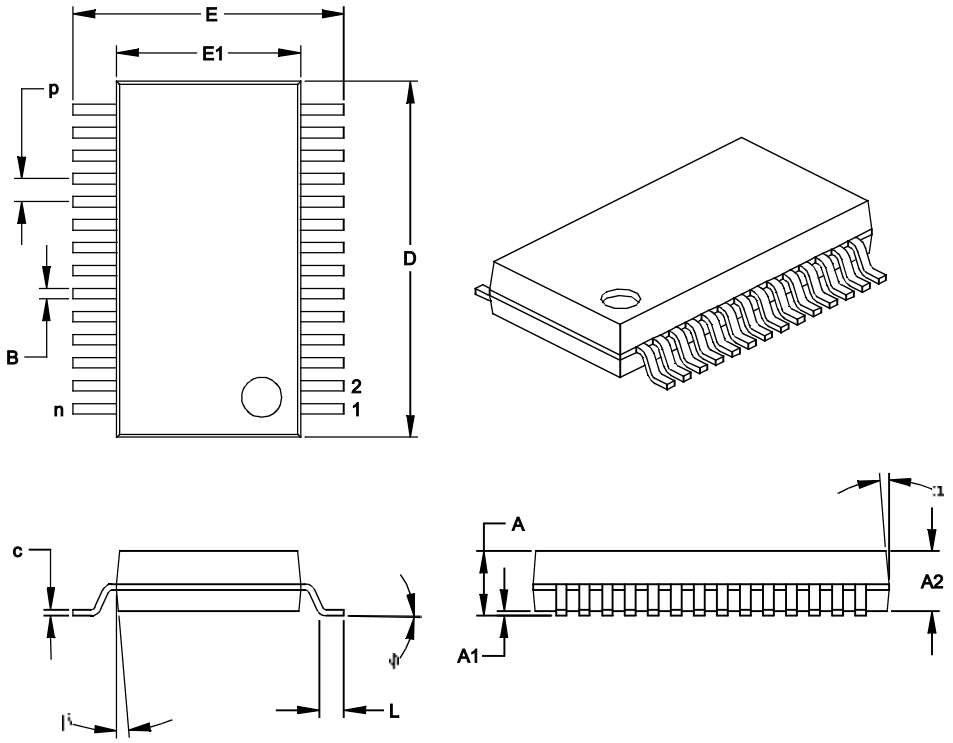
Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	10.01	10.34	10.67
Molded Package Width	E1	.288	.295	.299	7.32	7.49	7.59
Overall Length	D	.695	.704	.712	17.65	17.87	18.08
Chamfer Distance	h	.010	.020	.029	0.25	0.50	0.74
Foot Length	L	.016	.033	.050	0.41	0.84	1.27
Foot Angle Top	φ	0	4	8	0	4	8
Lead Thickness	c	.009	.011	.013	0.23	0.28	0.33
Lead Width	B	.014	.017	.020	0.36	0.42	0.51
Mold Draft Angle Top	α	0	12	15	0	12	15
Mold Draft Angle Bottom	β	0	12	15	0	12	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-013
Drawing No. C04-052

© 2003 Microchip Technology Inc.

28-Lead Plastic Shrink Small Outline (SS) – 209 mil, 5.30 mm (SSOP)

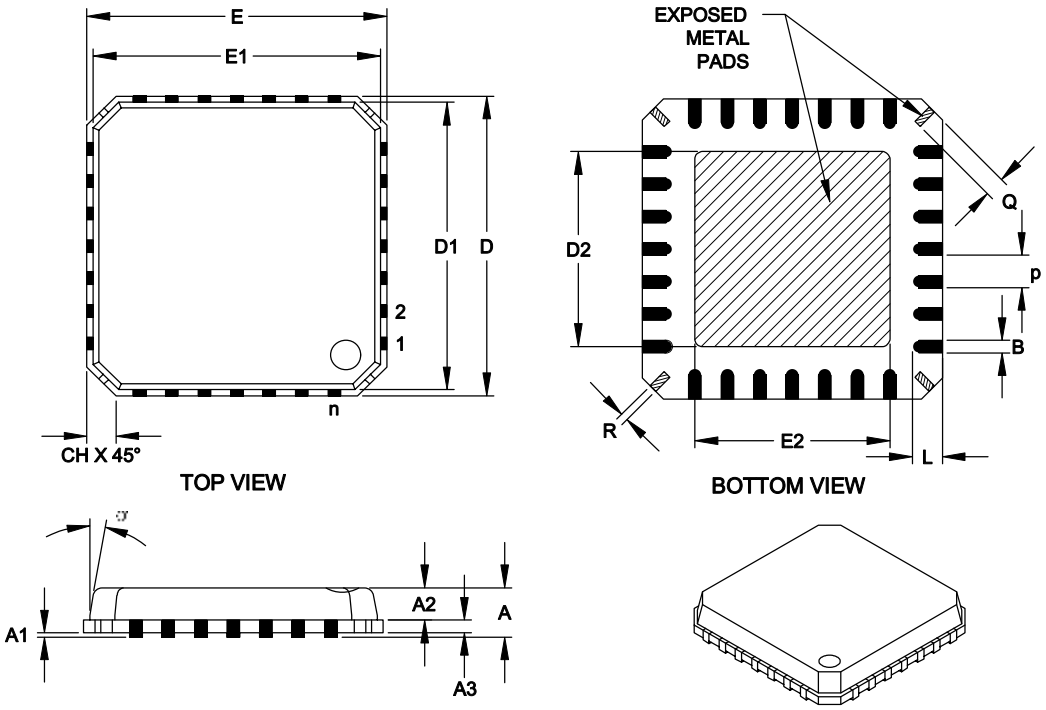


Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.026			0.65	
Overall Height	A	.068	.073	.078	1.73	1.85	1.98
Molded Package Thickness	A2	.064	.068	.072	1.63	1.73	1.83
Standoff §	A1	.002	.006	.010	0.05	0.15	0.25
Overall Width	E	.299	.309	.319	7.59	7.85	8.10
Molded Package Width	E1	.201	.207	.212	5.11	5.25	5.38
Overall Length	D	.396	.402	.407	10.06	10.20	10.34
Foot Length	L	.022	.030	.037	0.56	0.75	0.94
Lead Thickness	c	.004	.007	.010	0.10	0.18	0.25
Foot Angle	φ	0	4	8	0.00	101.60	203.20
Lead Width	B	.010	.013	.015	0.25	0.32	0.38
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-150
Drawing No. C04-073

28-Lead Plastic Quad Flat No Lead Package (ML) 6x6 mm Body, Punch Singulated (QFN)



Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.026 BSC			0.65 BSC	
Overall Height	A		.033	.039		0.85	1.00
Molded Package Thickness	A2		.026	.031		0.65	0.80
Standoff	A1	.000	.0004	.002	0.00	0.01	0.05
Base Thickness	A3		.008 REF			0.20 REF	
Overall Width	E		.236 BSC			6.00 BSC	
Molded Package Width	E1		.226 BSC			5.75 BSC	
Exposed Pad Width	E2	.140	.146	.152	3.55	3.70	3.85
Overall Length	D		.236 BSC			6.00 BSC	
Molded Package Length	D1		.226 BSC			5.75 BSC	
Exposed Pad Length	D2	.140	.146	.152	3.55	3.70	3.85
Lead Width	B	.009	.011	.014	0.23	0.28	0.35
Lead Length	L	.020	.024	.030	0.50	0.60	0.75
Tie Bar Width	R	.005	.007	.010	0.13	0.17	0.23
Tie Bar Length	Q	.012	.016	.026	0.30	0.40	0.65
Chamfer	CH	.009	.017	.024	0.24	0.42	0.60
Mold Draft Angle Top				12°			12°

*Controlling Parameter

Notes:

Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

JEDEC equivalent: mMO-220

Drawing No. C04-114

© 2003 Microchip Technology Inc.

APPENDIX A: REVISION HISTORY

Revision A (November 2001)

Original data sheet for PIC16F87XA devices. The devices presented are enhanced versions of the PIC16F87X microcontrollers discussed in the "*PIC16F87X Data Sheet*" (DS30292).

Revision B (October 2003)

This revision includes the DC and AC Characteristics Graphs and Tables. The Electrical Specifications in **Section 17.0 "Electrical Characteristics"** have been updated and there have been minor corrections to the data sheet text.

APPENDIX B: DEVICE DIFFERENCES

The differences between the devices in this data sheet are listed in Table B-1.

TABLE B-1: DIFFERENCES BETWEEN DEVICES IN THE PIC16F87XA FAMILY

	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Slave Port	No	Yes	No	Yes
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

APPENDIX C: CONVERSION CONSIDERATIONS

Considerations for converting from previous versions of devices to the ones listed in this data sheet are listed in Table C-1.

TABLE C-1: CONVERSION CONSIDERATIONS

Characteristic	PIC16C7X	PIC16F87X	PIC16F87XA
Pins	28/40	28/40	28/40
Timers	3	3	3
Interrupts	11 or 12	13 or 14	14 or 15
Communication	PSP, USART, SSP (SPI, I^2C Slave)	PSP, USART, SSP (SPI, I^2C Master/Slave)	PSP, USART, SSP (SPI, I^2C Master/Slave)
Frequency	20 MHz	20 MHz	20 MHz
Voltage	2.5V-5.5V	2.2V-5.5V	2.0V-5.5V
A/D	8-bit, 4 conversion clock selects	10-bit, 4 conversion clock selects	10-bit, 7 conversion clock selects
CCP	2	2	2
Comparator	—	—	2
Comparator Voltage Reference	—	—	Yes
Program Memory	4K, 8K EPROM	4K, 8K Flash (Erase/Write on single-word)	4K, 8K Flash (Erase/Write on four-word blocks)
RAM	192, 368 bytes	192, 368 bytes	192, 368 bytes
EEPROM Data	None	128, 256 bytes	128, 256 bytes
Code Protection	On/Off	Segmented, starting at end of program memory	On/Off
Program Memory Write Protection	—	On/Off	Segmented, starting at beginning of program memory
Other	—	In-Circuit Debugger, Low-Voltage Programming	In-Circuit Debugger, Low-Voltage Programming

© 2003 Microchip Technology Inc.

INDEX

A

A/D 127
Acquisition Requirements 130
ADCON0 Register 127
ADCON1 Register 127
ADIF Bit 129
ADRESH Register 127
ADRESL Register 127
Analog Port Pins 49, 51
Associated Registers and Bits 133
Calculating Acquisition Time 130
Configuring Analog Port Pins 131
Configuring the Interrupt 129
Configuring the Module 129
Conversion Clock 131
Conversions 132
Converter Characteristics 194
Effects of a Reset 133
GO/DONE Bit 129
Internal Sampling Switch (Rss) Impedance 130
Operation During Sleep 133
Result Registers 132
Source Impedance 130
A/D Conversion Requirements 195
Absolute Maximum Ratings 173
ACKSTAT 101
ADCON0 Register 19
ADCON1 Register 20
Addressable Universal Synchronous Asynchronous Receiver Transmitter. *See* USART.
ADRESH Register 19
ADRESL Register 20
Analog-to-Digital Converter. *See* A/D.
Application Notes
AN552 (Implementing Wake-up on Key Stroke) 44
AN556 (Implementing a Table Read) 30
Assembler
MPASM Assembler 167
Asynchronous Reception
Associated Registers 118, 120
Asynchronous Transmission
Associated Registers 116

B

Banking, Data Memory 16, 22
Baud Rate Generator 97
Associated Registers 113
BCLIF 28
BF 101
Block Diagrams
A/D 129
Analog Input Model 130, 139
Baud Rate Generator 97
Capture Mode Operation 65
Comparator I/O Operating Modes 136
Comparator Output 138
Comparator Voltage Reference 142
Compare Mode Operation 66
Crystal/Ceramic Resonator Operation (HS, XT or LP Osc Configuration) 145
External Clock Input Operation (HS, XT or LP Osc Configuration) 145
Interrupt Logic 153
MSSP (I^2C Mode) 80
MSSP (SPI Mode) 71
On-Chip Reset Circuit 147
PIC16F873A/PIC16F876A Architecture 6
PIC16F874A/PIC16F877A Architecture 7
PORTC
Peripheral Output Override (RC2:0, RC7:5) Pins 46
Peripheral Output Override (RC4:3) Pins 46
PORTD (in I/O Port Mode) 48
PORTD and PORTE (Parallel Slave Port) 51
PORTE (In I/O Port Mode) 49
RA3:RA0 Pins 41
RA4/T0CKI Pin 42
RA5 Pin 42
RB3:RB0 Pins 44
RB7:RB4 Pins 44
RC Oscillator Mode 146
Recommended MCLR Circuit 148
Simplified PWM Mode 67
Timer0/WDT Prescaler 53
Timer1 58
Timer2 61
USART Receive 117, 119
USART Transmit 115
Watchdog Timer 155
BOR. *See* Brown-out Reset.
BRG. *See* Baud Rate Generator.
BRGH Bit 113
Brown-out Reset (BOR) 143, 147, 148, 149, 150
BOR Status (BOR Bit) 29
Bus Collision During a Repeated Start Condition 108
Bus Collision During a Start Condition 106
Bus Collision During a Stop Condition 109
Bus Collision Interrupt Flag bit, BCLIF 28

C

C Compilers
MPLAB C17 168
MPLAB C18 168
MPLAB C30 168
Capture/Compare/PWM (CCP) 63
Associated Registers
Capture, Compare and Timer1 68
PWM and Timer2 69
Capture Mode 65
CCP1IF 65
Prescaler 65
CCP Timer Resources 63
Compare
Special Event Trigger Output of CCP1 66
Special Event Trigger Output of CCP2 66
Compare Mode 66
Software Interrupt Mode 66
Special Event Trigger 66
Interaction of Two CCP Modules (table) 63
PWM Mode 67
Duty Cycle 67
Example Frequencies/Resolutions (table) 68
PWM Period 67
Special Event Trigger and A/D Conversions 66

Capture/Compare/PWM Requirements
(CCP1 and CCP2) 186
CCP. *See* Capture/Compare/PWM.
CCP1CON Register 19
CCP2CON Register 19
CCPR1H Register 19, 63
CCPR1L Register 19, 63
CCPR2H Register 19, 63
CCPR2L Register 19, 63
CCPxM0 Bit 64
CCPxM1 Bit 64
CCPxM2 Bit 64
CCPxM3 Bit 64
CCPxX Bit 64
CCPxY Bit 64
CLKO and I/O Timing Requirements 183
CMCON Register 20
Code Examples
Call of a Subroutine in Page 1 from Page 0 30
Indirect Addressing 31
Initializing PORTA 41
Loading the SSPBUF (SSPSR) Register 74
Reading Data EEPROM 35
Reading Flash Program Memory 36
Saving Status, W and PCLATH Registers
in RAM 154
Writing to Data EEPROM 35
Writing to Flash Program Memory 38
Code Protection 143, 157
Comparator Module 135
Analog Input Connection
Considerations 139
Associated Registers 140
Configuration 136
Effects of a Reset 139
Interrupts 138
Operation 137
Operation During Sleep 139
Outputs 137
Reference 137
Response Time 137
Comparator Specifications 180
Comparator Voltage Reference 141
Associated Registers 142
Computed GOTO 30
Configuration Bits 143
Configuration Word 144
Conversion Considerations 220
CVRCON Register 20

D

Data EEPROM and Flash Program Memory
EEADR Register 33
EEADRH Register 33
EECON1 Register 33
EECON2 Register 33
EEDATA Register 33
EEDATH Register 33
Data EEPROM Memory
Associated Registers 39
EEADR Register 33
EEADRH Register 33
EECON1 Register 33
EECON2 Register 33
Operation During Code-Protect 39
Protection Against Spurious Writes 39
Reading 35
Write Complete Flag Bit (EEIF) 33
Writing 35
Data Memory 16
Bank Select (RP1:RP0 Bits) 16, 22
General Purpose Registers 16
Register File Map 17, 18
Special Function Registers 19
DC and AC Characteristics Graphs and Tables 197
DC Characteristics 175–179
Demonstration Boards
PICDEM 1 170
PICDEM 17 170
PICDEM 18R PIC18C601/801 171
PICDEM 2 Plus 170
PICDEM 3 PIC16C92X 170
PICDEM 4 170
PICDEM LIN PIC16C43X 171
PICDEM USB PIC16C7X5 171
PICDEM.net Internet/Ethernet 170
Development Support 167
Device Differences 219
Device Overview 5
Direct Addressing 31

E

EEADR Register 21, 33
EEADRH Register 21, 33
EECON1 Register 21, 33
EECON2 Register 21, 33
EEDATA Register 21
EEDATH Register 21
Electrical Characteristics 173
Errata 4
Evaluation and Programming Tools 171
External Clock Timing Requirements 182
External Interrupt Input (RB0/INT). *See* Interrupt Sources.
External Reference Signal 137

F

Firmware Instructions 159
Flash Program Memory
Associated Registers 39
EECON1 Register 33
EECON2 Register 33
Reading 36
Writing 37
FSR Register 19, 20, 31

G

General Call Address Support 94

© 2003 Microchip Technology Inc.

I

I/O Ports 41
I2C Bus Data Requirements 192
I²C Bus Start/Stop Bits Requirements 191
I²C Mode
  Registers 80
I²C Mode 80
  ACK Pulse 84, 85
  Acknowledge Sequence Timing 104
  Baud Rate Generator 97
  Bus Collision
    Repeated Start Condition 108
    Start Condition 106
    Stop Condition 109
  Clock Arbitration 98
  Effect of a Reset 105
  General Call Address Support 94
  Master Mode 95
    Operation 96
    Repeated Start Timing 100
  Master Mode Reception 101
  Master Mode Start Condition 99
  Master Mode Transmission 101
  Multi-Master Communication, Bus Collision
    and Arbitration 105
  Multi-Master Mode 105
  Read/Write Bit Information (R/W Bit) 84, 85
  Serial Clock (RC3/SCK/SCL) 85
  Slave Mode 84
    Addressing 84
    Reception 85
    Transmission 85
  Sleep Operation 105
  Stop Condition Timing 104
ID Locations 143, 157
In-Circuit Debugger 143, 157
  Resources 157
In-Circuit Serial Programming (ICSP) 143, 158
INDF Register 19, 20, 31
Indirect Addressing 31
  FSR Register 16
Instruction Format 159
Instruction Set 159
  ADDLW 161
  ADDWF 161
  ANDLW 161
  ANDWF 161
  BCF 161
  BSF 161
  BTFSC 161
  BTFSS 161
  CALL 162
  CLRF 162
  CLRW 162
  CLRWDT 162
  COMF 162
  DECF 162
  DECFSZ 163
  GOTO 163
  INCF 163
  INCFSZ 163
  IORLW 163
  IORWF 163
  RETURN 164
  RLF 164
  RRF 164
  SLEEP 164
  SUBLW 164
  SUBWF 164
  SWAPF 165
  XORLW 165
  XORWF 165
  Summary Table 160
INT Interrupt (RB0/INT). *See* Interrupt Sources.
INTCON Register 24
  GIE Bit 24
  INTE Bit 24
  INTF Bit 24
  PEIE Bit 24
  RBIE Bit 24
  RBIF Bit 24, 44
  TMR0IE Bit 24
  TMR0IF Bit 24
Inter-Integrated Circuit. *See* I²C.
Internal Reference Signal 137
Internal Sampling Switch (Rss) Impedance 130
Interrupt Sources 143, 153
  Interrupt-on-Change (RB7:RB4) 44
  RB0/INT Pin, External 9, 11, 154
  TMR0 Overflow 154
  USART Receive/Transmit Complete 111
Interrupts
  Bus Collision Interrupt 28
  Synchronous Serial Port Interrupt 26
Interrupts, Context Saving During 154
Interrupts, Enable Bits
  Global Interrupt Enable (GIE Bit) 24, 153
  Interrupt-on-Change (RB7:RB4)
    Enable (RBIE Bit) 24, 154
  Peripheral Interrupt Enable (PEIE Bit) 24
  RB0/INT Enable (INTE Bit) 24
  TMR0 Overflow Enable (TMR0IE Bit) 24
Interrupts, Flag Bits
  Interrupt-on-Change (RB7:RB4) Flag
    (RBIF Bit) 24, 44, 154
  RB0/INT Flag (INTF Bit) 24
  TMR0 Overflow Flag (TMR0IF Bit) 24, 154

L

Loading of PC 30
Low-Voltage ICSP Programming 158
Low-Voltage In-Circuit Serial Programming 143

M

Master Clear (MCLR) 8
  MCLR Reset, Normal Operation 147, 149, 150
  MCLR Reset, Sleep 147, 149, 150
Master Synchronous Serial Port (MSSP). *See* MSSP.
MCLR 148
MCLR/VPP 10
Memory Organization 15
  Data EEPROM Memory 33
  Data Memory 16
  Flash Program Memory 33
  Program Memory 15
MPLAB ASM30 Assembler, Linker, Librarian 168
MPLAB ICD 2 In-Circuit Debugger 169
MPLAB ICE 2000 High-Performance Universal
  In-Circuit Emulator 169

MPLAB ICE 4000 High-Performance Universal
In-Circuit Emulator 169
MPLAB Integrated Development
Environment Software 167
MPLINK Object Linker/MPLIB Object Librarian 168
MSSP 71
I²C Mode. *See* I²C.
SPI Mode 71
SPI Mode. *See* SPI.
MSSP Module
Clock Stretching 90
Clock Synchronization and the CKP Bit 91
Control Registers (General) 71
Operation 84
Overview 71
SPI Master Mode 76
SPI Slave Mode 77
SSPBUF 76
SSPSR 76
Multi-Master Mode 105

O

Opcode Field Descriptions 159
OPTION_REG Register 23
INTEDG Bit 23
PS2:PS0 Bits 23
PSA Bit 23
RBPU Bit 23
T0CS Bit 23
T0SE Bit 23
OSC1/CLKI Pin 8, 10
OSC2/CLKO Pin 8, 10
Oscillator Configuration
HS 145, 149
LP 145, 149
RC 145, 146, 149
XT 145, 149
Oscillator Selection 143
Oscillator Start-up Timer (OST) 143, 148
Oscillator, WDT 155
Oscillators
Capacitor Selection 146
Ceramic Resonator Selection 145
Crystal and Ceramic Resonators 145
RC 146

P

Package Information
Marking 209
Packaging Information 209
Paging, Program Memory 30
Parallel Slave Port (PSP) 13, 48, 51
Associated Registers 52
RE0/RD/AN5 Pin 49, 51
RE1/WR/AN6 Pin 49, 51
RE2/CS/AN7 Pin 49, 51
Select (PSPMODE Bit) 48, 49, 50, 51
Parallel Slave Port Requirements
(PIC16F874A/ 877A Only) 187
PCL Register 19, 20, 30
PCLATH Register 19, 20, 30
PCON Register 20, 29, 149
BOR Bit 29
POR Bit 29
PIC16F87XA Product Identification System 231
PICkit 1 Flash Starter Kit 171
PICSTART Plus Development Programmer 169
PIE1 Register 20, 25
PIE2 Register 20, 27
Pinout Descriptions
PIC16F873A/PIC16F876A 8
PIR1 Register 19, 26
PIR2 Register 19, 28
POP 30
POR. *See* Power-on Reset.
PORTA 8, 10
Associated Registers 43
Functions 43
PORTA Register 19, 41
TRISA Register 41
PORTB 9, 11
Associated Registers 45
Functions 45
PORTB Register 19, 44
Pull-up Enable (RBPU Bit) 23
RB0/INT Edge Select (INTEDG Bit) 23
RB0/INT Pin, External 9, 11, 154
RB7:RB4 Interrupt-on-Change 154
RB7:RB4 Interrupt-on-Change Enable
(RBIE Bit) 24, 154
RB7:RB4 Interrupt-on-Change Flag
(RBIF Bit) 24, 44, 154
TRISB Register 21, 44
PORTB Register 21
PORTC 9, 12
Associated Registers 47
Functions 47
PORTC Register 19, 46
RC3/SCK/SCL Pin 85
RC6/TX/CK Pin 112
RC7/RX/DT Pin 112, 113
TRISC Register 46, 111
PORTD 13, 51
Associated Registers 48
Functions 48
Parallel Slave Port (PSP) Function 48
PORTD Register 19, 48
TRISD Register 48
PORTE 13
Analog Port Pins 49, 51
Associated Registers 50
Functions 49
Input Buffer Full Status (IBF Bit) 50
Input Buffer Overflow (IBOV Bit) 50
Output Buffer Full Status (OBF Bit) 50
PORTE Register 19, 49
PSP Mode Select (PSPMODE Bit) 48, 49, 50, 51
RE0/RD/AN5 Pin 49, 51
RE1/WR/AN6 Pin 49, 51
RE2/CS/AN7 Pin 49, 51
TRISE Register 49
Postscaler, WDT
Assignment (PSA Bit) 23
Rate Select (PS2:PS0 Bits) 23
Power-down Mode. *See* Sleep.
Power-on Reset (POR) 143, 147, 148, 149, 150
POR Status (POR Bit) 29
Power Control (PCON) Register 149
Power-down (PD Bit) 22, 147
Power-up Timer (PWRT) 143
Time-out (TO Bit) 22, 147

© 2003 Microchip Technology Inc.

Power-up Timer (PWRT) 148
PR2 Register 20, 61
Prescaler, Timer0
Assignment (PSA Bit) 23
Rate Select (PS2:PS0 Bits) 23
PRO MATE II Universal Device Programmer 169
Program Counter
Reset Conditions 149
Program Memory 15
Interrupt Vector 15
Paging 30
Program Memory Map and Stack
(PIC16F873A/874A) 15
Program Memory Map and Stack
(PIC16F876A/877A) 15
Reset Vector 15
Program Verification 157
Programming Pin (VPP) 8
Programming, Device Instructions 159
PSP. *See* Parallel Slave Port.
Pulse Width Modulation. *See* Capture/Compare/PWM, PWM Mode.
PUSH 30

R

RA0/AN0 Pin 8, 10
RA1/AN1 Pin 8, 10
RA2/AN2/VREF-/CVREF Pin 8, 10
RA3/AN3/VREF+ Pin 8, 10
RA4/T0CKI/C1OUT Pin 8, 10
RA5/AN4/SS/C2OUT Pin 8, 10
RAM. *See* Data Memory.
RB0/INT Pin 9, 11
RB1 Pin 9, 11
RB2 Pin 9, 11
RB3/PGM Pin 9, 11
RB4 Pin 9, 11
RB5 Pin 9, 11
RB6/PGC Pin 9, 11
RB7/PGD Pin 9, 11
RC0/T1OSO/T1CKI Pin 9, 12
RC1/T1OSI/CCP2 Pin 9, 12
RC2/CCP1 Pin 9, 12
RC3/SCK/SCL Pin 9, 12
RC4/SDI/SDA Pin 9, 12
RC5/SDO Pin 9, 12
RC6/TX/CK Pin 9, 12
RC7/RX/DT Pin 9, 12
RCREG Register 19
RCSTA Register 19
ADDEN Bit 112
CREN Bit 112
FERR Bit 112
OERR Bit 112
RX9 Bit 112
RX9D Bit 112
SPEN Bit 111, 112
SREN Bit 112
RD0/PSP0 Pin 13
RD1/PSP1 Pin 13
RD2/PSP2 Pin 13
RD3/PSP3 Pin 13
RD4/PSP4 Pin 13
RD5/PSP5 Pin 13
RD6/PSP6 Pin 13
RD7/PSP7 Pin 13
RE0/RD/AN5 Pin 13
RE1/WR/AN6 Pin 13
RE2/CS/AN7 Pin 13
Read-Modify-Write Operations 159
Register File 16
Register File Map (PIC16F873A/874A) 18
Register File Map (PIC16F876A/877A) 17
Registers
ADCON0 (A/D Control 0) 127
ADCON1 (A/D Control 1) 128
CCP1CON/CCP2CON (CCP Control 1 and CCP Control 2) 64
CMCON (Comparator Control) 135
CVRCON (Comparator Voltage Reference Control) 141
EECON1 (EEPROM Control 1) 34
FSR 31
INTCON 24
OPTION_REG 23, 54
PCON (Power Control) 29
PIE1 (Peripheral Interrupt Enable 1) 25
PIE2 (Peripheral Interrupt Enable 2) 27
PIR1 (Peripheral Interrupt Request 1) 26
PIR2 (Peripheral Interrupt Request 2) 28
RCSTA (Receive Status and Control) 112
Special Function, Summary 19
SSPCON (MSSP Control 1, I2C Mode) 82
SSPCON (MSSP Control 1, SPI Mode) 73
SSPCON2 (MSSP Control 2, I2C Mode) 83
SSPSTAT (MSSP Status, I2C Mode) 81
SSPSTAT (MSSP Status, SPI Mode) 72
Status 22
T1CON (Timer1 Control) 57
T2CON (Timer2 Control) 61
TRISE Register 50
TXSTA (Transmit Status and Control) 111
Reset 143, 147
Brown-out Reset (BOR). *See* Brown-out Reset (BOR).
MCLR Reset. *See* MCLR.
Power-on Reset (POR). *See* Power-on Reset (POR).
Reset Conditions for PCON Register 149
Reset Conditions for Program Counter 149
Reset Conditions for Status Register 149
WDT Reset. *See* Watchdog Timer (WDT).
Reset, Watchdog Timer, Oscillator Start-up Timer, Power-up Timer and Brown-out Reset Requirements 184
Revision History 219

S

SCI. *See* USART.
SCK 71
SDI 71
SDO 71
Serial Clock, SCK 71
Serial Communication Interface. *See* USART.
Serial Data In, SDI 71
Serial Data Out, SDO 71
Serial Peripheral Interface. *See* SPI.
Slave Select Synchronization 77
Slave Select, SS 71
Sleep 143, 147, 156
Software Simulator (MPLAB SIM) 168
Software Simulator (MPLAB SIM30) 168
SPBRG Register 20
Special Features of the CPU 143

Special Function Registers 19
Special Function Registers (SFRs) 19
Speed, Operating 1
SPI Mode 71, 77
Associated Registers 79
Bus Mode Compatibility 79
Effects of a Reset 79
Enabling SPI I/O 75
Master Mode 76
Master/Slave Connection 75
Serial Clock 71
Serial Data In 71
Serial Data Out 71
Slave Select 71
Slave Select Synchronization 77
Sleep Operation 79
SPI Clock 76
Typical Connection 75
SPI Mode Requirements 190
SS 71
SSP
SPI Master/Slave Connection 75
SSPADD Register 20
SSPBUF Register 19
SSPCON Register 19
SSPCON2 Register 20
SSPIF 26
SSPOV 101
SSPSTAT Register 20
R/W Bit 84, 85
Stack 30
Overflows 30
Underflow 30
Status Register
C Bit 22
DC Bit 22
IRP Bit 22
PD Bit 22, 147
RP1:RP0 Bits 22
TO Bit 22, 147
Z Bit 22
Synchronous Master Reception
Associated Registers 123
Synchronous Master Transmission
Associated Registers 122
Synchronous Serial Port Interrupt 26
Synchronous Slave Reception
Associated Registers 125
Synchronous Slave Transmission
Associated Registers 125

T

T1CKPS0 Bit 57
T1CKPS1 Bit 57
T1CON Register 19
T1OSCEN Bit 57
T1SYNC Bit 57
T2CKPS0 Bit 61
T2CKPS1 Bit 61
T2CON Register 19
TAD 131
Time-out Sequence 148
Timer0 53
Associated Registers 55
Clock Source Edge Select (T0SE Bit) 23
Clock Source Select (T0CS Bit) 23
External Clock 54
Interrupt 53
Overflow Enable (TMR0IE Bit) 24
Overflow Flag (TMR0IF Bit) 24, 154
Overflow Interrupt 154
Prescaler 54
T0CKI 54
Timer0 and Timer1 External Clock Requirements 185
Timer1 57
Associated Registers 60
Asynchronous Counter Mode 59
Reading and Writing to 59
Counter Operation 58
Operation in Timer Mode 58
Oscillator 59
Capacitor Selection 59
Prescaler 60
Resetting of Timer1 Registers 60
Resetting Timer1 Using a CCP Trigger Output 59
Synchronized Counter Mode 58
TMR1H 59
TMR1L 59
Timer2 61
Associated Registers 62
Output 62
Postscaler 61
Prescaler 61
Prescaler and Postscaler 62
Timing Diagrams
A/D Conversion 195
Acknowledge Sequence 104
Asynchronous Master Transmission 116
Asynchronous Master Transmission
(Back to Back) 116
Asynchronous Reception 118
Asynchronous Reception with
Address Byte First 120
Asynchronous Reception with
Address Detect 120
Baud Rate Generator with Clock Arbitration 98
BRG Reset Due to SDA Arbitration During
Start Condition 107
Brown-out Reset 184
Bus Collision During a Repeated
Start Condition (Case 1) 108
Bus Collision During Repeated
Start Condition (Case 2) 108
Bus Collision During Start Condition
(SCL = 0) 107
Bus Collision During Start Condition
(SDA Only) 106
Bus Collision During Stop Condition
(Case 1) 109
Bus Collision During Stop Condition
(Case 2) 109
Bus Collision for Transmit and Acknowledge 105
Capture/Compare/PWM (CCP1 and CCP2) 186
CLKO and I/O 183
Clock Synchronization 91
External Clock 182
First Start Bit 99

© 2003 Microchip Technology Inc.

I²C Bus Data 191
I²C Bus Start/Stop Bits 190
I²C Master Mode (Reception, 7-bit Address) 103
I²C Master Mode (Transmission, 7 or 10-bit Address) 102
I²C Slave Mode (Transmission, 10-bit Address) 89
I²C Slave Mode (Transmission, 7-bit Address) 87
I²C Slave Mode with SEN = 1 (Reception, 10-bit Address) 93
I²C Slave Mode with SEN = 0 (Reception, 10-bit Address) 88
I²C Slave Mode with SEN = 0 (Reception, 7-bit Address) 86
I²C Slave Mode with SEN = 1 (Reception, 7-bit Address) 92
Parallel Slave Port (PIC16F874A/877A Only) 187
Parallel Slave Port (PSP) Read 52
Parallel Slave Port (PSP) Write 52
Repeat Start Condition 100
Reset, Watchdog Timer, Start-up Timer and Power-up Timer 184
Slave Mode General Call Address Sequence (7 or 10-bit Address Mode) 94
Slave Synchronization 77
Slow Rise Time (MCLR Tied to VDD via RC Network) 152
SPI Master Mode (CKE = 0, SMP = 0) 188
SPI Master Mode (CKE = 1, SMP = 1) 188
SPI Mode (Master Mode) 76
SPI Mode (Slave Mode with CKE = 0) 78
SPI Mode (Slave Mode with CKE = 1) 78
SPI Slave Mode (CKE = 0) 189
SPI Slave Mode (CKE = 1) 189
Stop Condition Receive or Transmit Mode 104
Synchronous Reception (Master Mode, SREN) 124
Synchronous Transmission 122
Synchronous Transmission (Through TXEN) 122
Time-out Sequence on Power-up (MCLR Not Tied to VDD)
Case 1 152
Case 2 152
Time-out Sequence on Power-up (MCLR Tied to VDD via RC Network) 151
Timer0 and Timer1 External Clock 185
USART Synchronous Receive (Master/Slave) 193
USART Synchronous Transmission (Master/Slave) 193
Wake-up from Sleep via Interrupt 157
Timing Parameter Symbology 181
TMR0 Register 19
TMR1CS Bit 57
TMR1H Register 19
TMR1L Register 19
TMR1ON Bit 57
TMR2 Register 19
TMR2ON Bit 61
TMRO Register 21
TOUTPS0 Bit 61
TOUTPS1 Bit 61
TOUTPS2 Bit 61
TOUTPS3 Bit 61
TRISA Register 20
TRISB Register 20
TRISC Register 20
TRISD Register 20
TRISE Register 20
IBF Bit 50
IBOV Bit 50
OBF Bit 50
PSPMODE Bit 48, 49, 50, 51
TXREG Register 19
TXSTA Register 20
BRGH Bit 111
CSRC Bit 111
SYNC Bit 111
TRMT Bit 111
TX9 Bit 111
TX9D Bit 111
TXEN Bit 111

U

USART 111
Address Detect Enable (ADDEN Bit) 112
Asynchronous Mode 115
Asynchronous Receive (9-bit Mode) 119
Asynchronous Receive with Address Detect. *See* Asynchronous Receive (9-bit Mode).
Asynchronous Receiver 117
Asynchronous Reception 118
Asynchronous Transmitter 115
Baud Rate Generator (BRG) 113
Baud Rate Formula 113
Baud Rates, Asynchronous Mode (BRGH = 0) 114
Baud Rates, Asynchronous Mode (BRGH = 1) 114
High Baud Rate Select (BRGH Bit) 111
Sampling 113
Clock Source Select (CSRC Bit) 111
Continuous Receive Enable (CREN Bit) 112
Framing Error (FERR Bit) 112
Mode Select (SYNC Bit) 111
Overrun Error (OERR Bit) 112
Receive Data, 9th Bit (RX9D Bit) 112
Receive Enable, 9-bit (RX9 Bit) 112
Serial Port Enable (SPEN Bit) 111, 112
Single Receive Enable (SREN Bit) 112
Synchronous Master Mode 121
Synchronous Master Reception 123
Synchronous Master Transmission 121
Synchronous Slave Mode 124
Synchronous Slave Reception 125
Synchronous Slave Transmit 124
Transmit Data, 9th Bit (TX9D) 111
Transmit Enable (TXEN Bit) 111
Transmit Enable, 9-bit (TX9 Bit) 111
Transmit Shift Register Status (TRMT Bit) 111
USART Synchronous Receive Requirements 193

V

VDD Pin 9, 13
Voltage Reference Specifications 180
Vss Pin 9, 13

W

Wake-up from Sleep 143, 156
Interrupts 149, 150
MCLR Reset 150
WDT Reset 150
Wake-up Using Interrupts 156
Watchdog Timer
Register Summary 155
Watchdog Timer (WDT) 143, 155
Enable (WDTE Bit) 155
Postscaler. *See* Postscaler, WDT.
Programming Considerations 155
RC Oscillator 155
Time-out Period 155
WDT Reset, Normal Operation 147, 149, 150
WDT Reset, Sleep 147, 149, 150
WCOL 99, 101, 104
WCOL Status Flag 99
WWW, On-Line Support 4

© 2003 Microchip Technology Inc.

ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape® or Microsoft® Internet Explorer. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Web Site

The Microchip web site is available at the following URL:

www.microchip.com

The file transfer site is available by using an FTP service to connect to:

ftp://ftp.microchip.com

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

SYSTEMS INFORMATION AND UPGRADE HOT LINE

The Systems Information and Upgrade Line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive the most current upgrade kits. The Hot Line Numbers are:

1-800-755-2345 for U.S. and most of Canada, and

1-480-792-7302 for the rest of the world.

042003

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager Total Pages Sent _____

RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (_____) _____ - _____ FAX: (_____) _____ - _____

Application (optional):

Would you like a reply? ___Y ___N

Device: PIC16F87XA Literature Number: DS39582B

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this document easy to follow? If not, why?

4. What additions to the document do you think would enhance the structure and subject?

5. What deletions from the document could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?

© 2003 Microchip Technology Inc.

PIC16F87XA PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	X	/XX	XXX
Device	Temperature Range	Package	Pattern

Device	PIC16F87XA(1), PIC16F87XAT(2); VDD range 4.0V to 5.5V PIC16LF87XA(1), PIC16LF87XAT(2); VDD range 2.0V to 5.5V	
Temperature Range	I	= -40°C to +85°C (Industrial)
Package	ML	= QFN (Metal Lead Frame)
	PT	= TQFP (Thin Quad Flatpack)
	SO	= SOIC
	SP	= Skinny Plastic DIP
	P	= PDIP
	L	= PLCC
	S	= SSOP

Examples:

a) PIC16F873A-I/P 301 = Industrial temp., PDIP package, normal VDD limits, QTP pattern #301.

b) PIC16LF876A-I/SO = Industrial temp., SOIC package, Extended VDD limits.

c) PIC16F877A-I/P = Industrial temp., PDIP package, 10 MHz, normal VDD limits.

Note 1: F = CMOS Flash
LF = Low-Power CMOS Flash

2: T = in tape and reel - SOIC, PLCC, TQFP packages only

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Atlanta
3780 Mansell Road, Suite 130
Alpharetta, GA 30022
Tel: 770-640-0034
Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848
Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, IN 46902
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888
Fax: 949-263-1338

Phoenix
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966
Fax: 480-792-4338

San Jose
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950
Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100
Fax: 86-10-85282104

China - Chengdu
Rm. 2401-2402, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200
Fax: 86-28-86766599

China - Fuzhou
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506
Fax: 86-591-7503521

China - Hong Kong SAR
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700
Fax: 86-21-6275-5060

China - Shenzhen
Rm. 1812, 18/F, Building A, United Plaza
No. 5022 Binhe Road, Futian District
Shenzhen 518033, China
Tel: 86-755-82901380
Fax: 86-755-8295-1393

China - Shunde
Room 401, Hongjian Building
No. 2 Fengxiangnan Road, Ronggui Town
Shunde City, Guangdong 528303, China
Tel: 86-765-8395507 Fax: 86-765-8395571

China - Qingdao
Rm. B505A, Fullhope Plaza,
No. 12 Hong Kong Central Rd.
Qingdao 266071, China
Tel: 86-532-5027355 Fax: 86-532-5027205

India
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5932 or
82-2-558-5934

Singapore
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Kaohsiung Branch
30F - 1 No. 8
Min Chuan 2nd Road
Kaohsiung 806, Taiwan
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan
Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria
Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45-4420-9895 Fax: 45-4420-9910

France
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany
Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy
Via Quasimodo, 12
20025 Legnano (MI)
Milan, Italy
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands
P. A. De Biesbosch 14
NL-5152 SC Drunen, Netherlands
Tel: 31-416-690399
Fax: 31-416-690340

United Kingdom
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-118-921-5869
Fax: 44-118-921-5820

07/28/03

© 2003 Microchip Technology Inc.

UTR-C12
UHF DATA TRANSCEIVER



ÜRÜN KILAVUZU
Version 1.4 (ARALIK 2004)

IVEDIK ORGANIZE SAN. BÖL. 21. CADDE 609. SOKAK NO:15
06370 OSTIM / ANKARA
TEL NO :(312) 395 68 75 - 76 FAKS NO:(312) 395 68 77
http:// www.udea.com.tr

GENEL ÖZELLIKLER

Özellikler :

* 434 MHz veya 868 MHZ UHF bandinda üretilebilir. EN 300 220 uyumlu.
* Dar band FSK modülasyon ve yüksek frekans kararliligi.
* Düsük akim sarfiyati ile pilli uygulamalar için idealdir.

Uygulama Alanlari :

* Uzaktan kontrol sistemleri
* Ev otomasyon
* Telemetry sistemleri
* Güvenlik amaçli alarm sistemleri
* Pager sistemleri
* Oyuncak

Genel Açiklamalar :

UTR-C12 UHF FSK data modül, Kisa Mesafe Erisimli Telsiz Cihazlarinin Temel Standartlari ile Kurma ve Kullanma Esaslari Hakkinda Yönetmelik (TGM-STK-001) 'in 433-434MHz. ISM bandi ile ilgili bölümünü kapsiyacak sekilde tasarlanmistir.

UTR-C12 Data Modülü müsteri taleplerine göre 2 farkli model olarak üretilmektedir. UTR-C12 modellerimiz sunlardir :

MODEL	TYPE	DATA FORMAT
UTR-C12 U	Transceiver Modül	UART
UTR-C12 M	Transceiver Modül	Manchester

UTR-C12 PCB montajina uygun tasarlanmistir. Basit bir kablo kullanilarak anten baglantisi yapilabilir.

PIN ÖZELLIKLERI

Pin No	Pin-Ismi	Input/Output	Açiklama	
1,3,4	GND		Kontrol kartinizin toprak hatina baglayiniz.	
2	ANT		Anten baglanti noktasi.	
5	DIO	I/O	Data Input/Output	
6	DCLK	-	Clock Output	UTR-C12M
	NC		Kullanilmiyor	UTR-C12U
7	NC	-	Kullanilmiyor	
8	NC	-	Kullanilmiyor	
9	T/R	I	Verici/Alici Seçim Pini	UTR-C12M/UTR-C12U
10	CH ½	I	Kanal Seçim pini	
11	+3V		+3VDC besleme terminali	Regüle edilmis voltaj kaynagi kullanilmalidir.

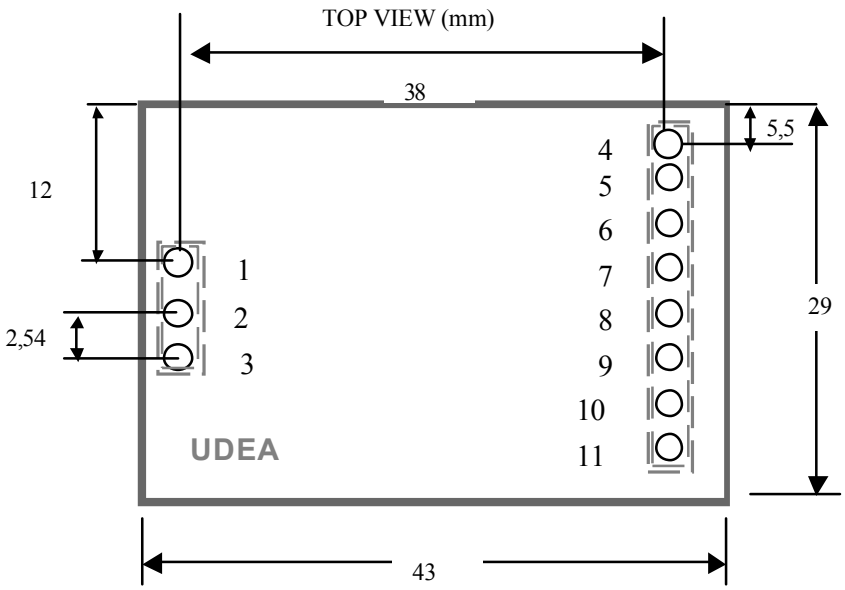


Figure 1. Ebat

TEKNIK ÖZELLIKLER

	Min.	Typ.	Max	Unit	Not
Besleme Voltaji	2,7	3	3,3	Vdc	Regüle edilmis voltaj kaynagi kullanin. ±100 mV
Akim Sarfiyati TX mod		23.4		mA	
Akim Sarfiyati RX mod		16.9		mA	
Logic “0” DI volt	0		0.3Vcc	Vdc	
Logic “1” DI volt	0.7Vcc		Vcc	Vdc	
Logic “0” DO volt	0		0.4	Vdc	
Logic “1” DO volt	2.5		Vcc	Vdc	
Çalisma Sicakligi	-20		+55	°C	ETSI 300 220
Depolama Sicakligi	-50		+150	°C	
Ebat	42 X 28 X 8 mm				

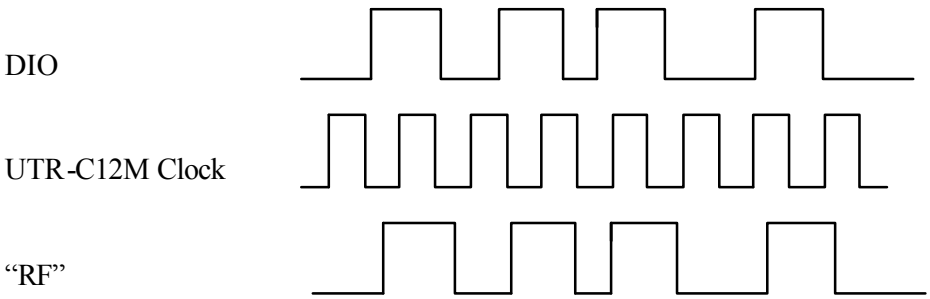
TEKNIK ÖZELLIKLER

	Min.	Typ.	Max	Unit	Not
RX Modu					
Alici duyarliligi		-115	-120	dBm	2.4 kBaud
Bandwidth		7.5		kHz	2.4 kBaud
System Noise Bandwidth	9.6		307.2	kHz	
LO kaçak			-57	dBm	
Saturation (maximum input lavel)		5		dB	
RX on time	3		17	Baud	
TX Modu					
Data Rate	0.45		153.6	Kbps	Üretim araligi
Çikis Gücü		9	10	dBm	
FM deviation	1	-	10	kHz	
TX on Time		1		ms	

DATA FORMAT

UTR-C12M / UTR-C12U				Frekans
T/R	CH ½	FONKSIYON		
1	1	RX off	TX on	FREKANS-1
1	0	RX off	TX on	FREKANS-2
0	1	RX on	TX off	FREKANS-1
0	0	RX on	TX off	FREKANS-2

Transmitter [Synchronous Manchester]:



Receiver [Synchronous Manchester]:

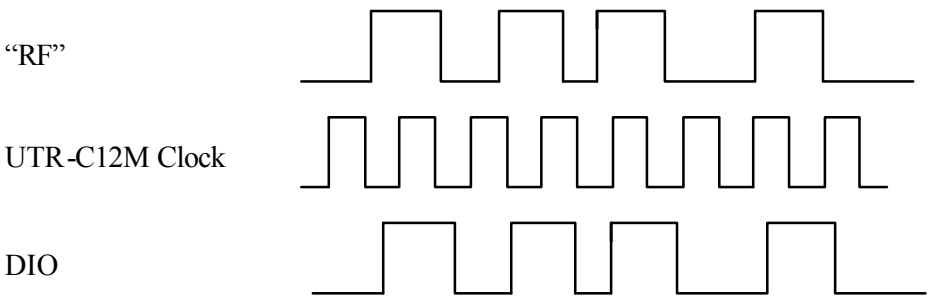
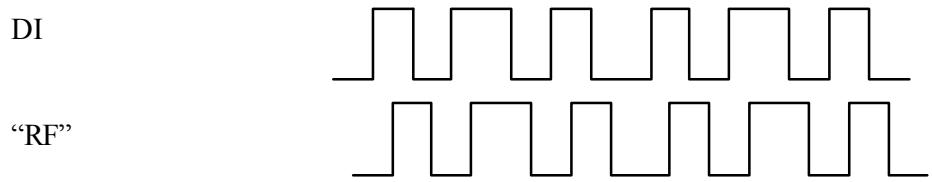


Figure 2. Synchronous Manchester encoded mode

Transmitter [UART]:



Receiver [UART]:

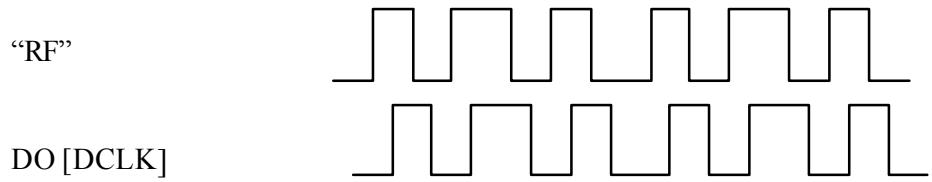


Figure 3. Transparent Asynchronous UART mode

KULLANIM TALIMATLARI

UTR-C12 modülü kullanicinin sistemine entegre olabilecek sekilde tasarlanmistir. Özel uygulamalar için kullanilabilecek nihayi bir ürün degildir. Elektronik sistemin içinde kullanilabilecek özel bir komponenet olarak ele alinmalidir. Kullanicinin temel elektronik bilgisine sahip olmasi gereklidir. RF teknoloji hakinda bilgi sahibi olunmasi kullanim açisindan oldukça faydalidir. RF ile ilgili zor kisimlarin önemli bir bölümü modül içinde çözülmüs durumdadir. Asagida bunlara ek bilgileri bulabilirsiniz.

Besleme Voltaji :
UTR-C12 içerisinde bir voltaj regülatorü bulunmamaktadir. Tasarim pil kullanimi düsünülerek yapilmistir. Bu nedenle besleme voltajinda belirtilen degerlere dikkat edilmelidir.

Modül belirtilen degerlerin altinda bir besleme yapildiginda kararsiz çalisacaktir. Besleme voltaji +3 VDC ve tapraklama GND baglantisi belirtilen degerlerin üzerinde veya ters olursa, modülde kalici tahribatlara yolaçilabilir. Düsük akim sarfiyatlarinin saglanabilmesi için modül içerisine ters polarizasyondan koruyacak bir devre konulmamistir.

En yüksek degerdeki besleme voltaji verilebilecek durumdaysa, besleme hatti üzerine ters polarizasyondan korunmak için basit bir diyot eklenebilir.

Besleme voltajinda çalisma sürecinde ±100 mV degisimlerin üzerindeki degisimler modulun kararsiz çalismasina neden olur. Besleme devresinde regülator IC kullanilmasi önerilir.

Connecting to microcontroller:
Microcontroller modül konfigrasyonu için 2 out pini kullanir. Iki yönlü bir pin RF olarak gönderilmek istenen Data'nin module verilmesi ve RF olarak gelen sinyalin modülden alinmasi için kullanilir. Modülün DCLK pini microcontollerin input pinine baglanmali. **Synchronous Manchester**, transmit mod'ta UTR-C12 DCLK pininden data clock çikisi verir, modülün DIO pini data input konumundadir. Microcontroller RF olarak göndermek istedigi datayi DCLK'un yükselen kenarinda verir. Receive mod'ta UTR-C12 sencronizasyonu gerçeklestirerek DCLK pininden clock and DIO pininden data çikisi verir.

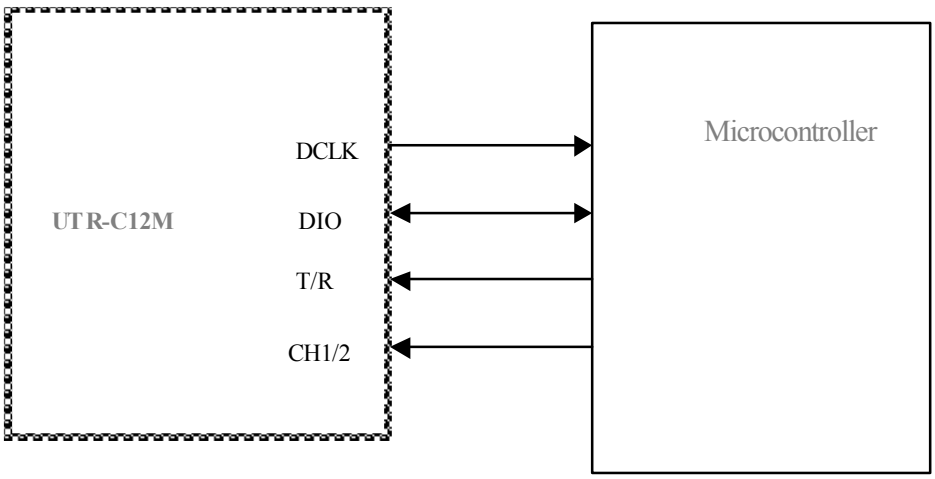


Figure 4. Microcontroller baglantisi M tipi modul

Microcontroller modül konfigrasyonu için 2 out pini kullanir. Bir output pini RF olarak gönderilmek istenen Data'nin module(DIN) verilmesi, bir input pini de RF olarak gelen sinyalin modülden(DOUT) alinmasi için kullanilir. **UART**, transmit mod'ta DIO pini data input pinidir, data RF modülasyona sencronizasyon ve encode edilmeden gönderilir. Receive mod'ta demodule edilen sinyal DIO pininden output olarak verilir.

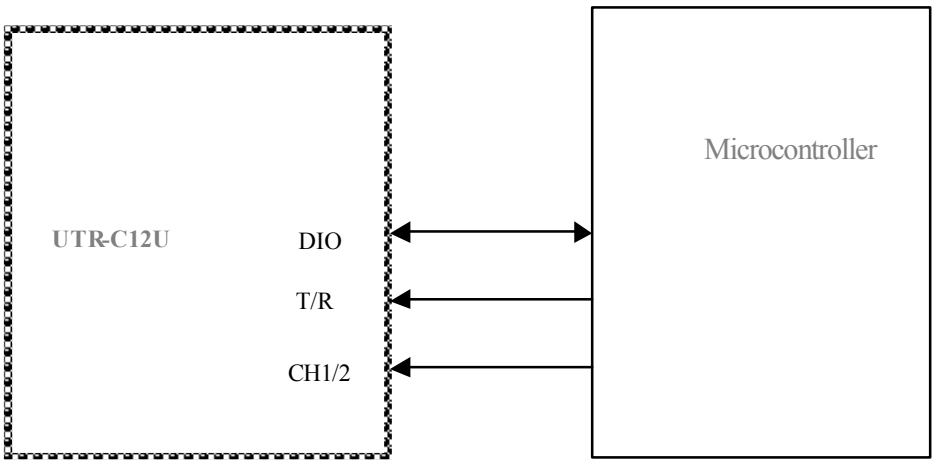


Figure 5. Microcontroller baglantisi U tipi modul

Data Format :
UART modüllerinde, data almak ve data göndermek için DIO pini bulunur. DIO pinine, Modül Transmitter moduna alinarak RF olarak gönderilmek istenen data verilir. DIO pini ayni zamanda, modül Receiver modunda iken RF ten alinan sinyallerin demodüle edilerek verildigi çikistir.

Synchronous Manchester modül'de, data almak ve göndermek üzere DI O pini bulunur. DIO pinine, modül Transmitter moduna alinarak RF olarak gönderilmek istenen data modül'den okunacak DCLK esliginde verilir. DIO pini ayni zamanda , modül Receiver modunda iken RF ten alinan sinyallerin demodüle edilerek DCLK esliginde verildigi çikistir.

Modüller UART (Transparent Asynchronous) veya Synchronous Manchester data iletisimine uygun olarak üretilirler. Satin alma islemi sirasinda kullanmak istediginiz iletisim modunu özellikle belirtmeniz gerekmektedir.

Kontrol PIN :
T/R pini modülün transmitter veya receiver moduna sokulmasi için kullanilir.
CH ½ pini kanal seçmek için kullanilir.

Anten:
Verimli data transferi ve alimi için gerekli en önemli iki nokta iyi bir anten ve dogru RF topraklama seçilmesidir. Anten olmadan datanin uzun mesafelere gönderilmesi mümkün degildir.

Modül basit bir anten baglanti pinine sahiptir. Uygun bir UHF anten dogrudan bu pine baglanabilir. UTR-C12 modülüne baglanabilecek en basit anten 17.3cm uzunlugundaki bir kablonun anten girisine lehimlenmesidir. Anteni, modülden uzak bir yere baglamaniz gerekiyorsa 50 Ohm Coax anten kablosu kullanmaniz gerekmektedir. Anten kablosunun topraklamasi, modulun anten girisine yakin bir yerden yapilmalidir.

Mümkün olmasina ragmen, alici duyarliliginin düsecegi için modülü ve anteni PCB üzerinden giden bir yol ile baglamanizi tavsiye etmiyoruz.

Asagidaki kurallar size yardimci olacaktir:

* Anten 50 ohm empedancli olmalidir.
* Lambda/4 whip anten boyu 433MHz. için yaklasik 17.3 cm dir.
* Anteni modül dik gelecek sekilde yukariya veya asagiya dogru monte ediniz.
* Anteni metal bir hazne içine koymayiniz.
* Insan vücudu metal objeler gibi etkiler gösterebilirler. Tasinabilir alici veya vericiler vücuttan uzak bir sekilde elde tutulmalidir.
* En iyi iletisim mesafesi her iki taraftaki antenlerin birbirini görmesi ile elde edilebilir. Herhangi bir obje veya metal bir engel iletisim mesafesini düsürecektir.
* Sinyal göndermeleri, gönderilen sinyallerin metal yüzeylerden, binalardan vb. gelen yansimalardan etkilenirler. Bu yanlis data alimlarina yolaçabilir.

```
'*************************************************************************
'*  Name    : ADC & TRANSMITTER.BAS
*
'*  Author  : MEHMET ZEYBEK
*
'*  Notice  : Copyright (c) 2006  MEHMET ZEYBEK
*
'*          : All Rights Reserved
*
'*  Date    : 09.08.2006
*
'*  Version : 6.4
'*  Notes   :
*
'*          :
'*************************************************************************
INCLUDE "MODEDEFS.BAS"      'Includes all serial comm. parameters
'-------------------------------------------------------------------------*
DEFINE      ADC_BITS    8      'Bits of A/D conversion
DEFINE      ADC_CLOCK   3      'Clock source (3=rc)
DEFINE      ADC_SAMPLEUS 100   'Sampling time in micro seconds

TRISA=%11111111             'All ports of A are input
TRISB=%00000000             'All ports of B are output
TRISC=%00000000             'All ports of C are output
TRISD=%00000010             'DIO pin is output, DCLK pin is input
TRISE=%00000000             'All ports of E are output
'-------------------------------------------------------------------------*
ECG_DATA      VAR     BYTE
'-------------------------------------------------------------------------*
PORTA=0                     'Value of Port A is 0
PORTB=0                     'Value of Port B is 0
PORTC=0                     'Value of Port C is 0
PORTD=0                     'Value of Port D is 0
PORTE=0                     'Value of Port E is 0
ECG_DATA=0                  'Value of ECG_DATA is 0
'-------------------------------------------------------------------------*
PAUSE 100                   'Pause 100 mili seconds
PORTD.2=1                   'Module is set transmitter mode
PORTD.3=0                   'Module is set 433 MHz
START:
    SEROUT2 PORTD.0,396,[REP$AA\5, REP$00\5, REP$FF\5]
      'Wake-Up signal is transmitted
ECG_CAPTURE:
    ADCIN 0,ECG_DATA
      'ECG signal is received by RA0
ADC_COMPLETE:
    IF ADCON0.2=1 THEN ADC_COMPLETE
      'When the consersion is completed, ECG data will be sent
    SEROUT2 PORTD.0,396,[("M"),ECG_DATA]
      'ECG data is sent with "M" character as 2400 baud
   GOTO ECG_CAPTURE
      'Capture ECG signal again
END
'*************************************************************************
```

```
'*************************************************************************
'*  Name    : RECEIVER.BAS
*
'*  Author  : MEHMET ZEYBEK
*
'*  Notice  : Copyright (c) 2006  MEHMET ZEYBEK
*
'*          : All Rights Reserved
*
'*  Date    : 09.08.2006
*
'*  Version : 4.6
*
'*  Notes   :
*
'*          :
*
'*************************************************************************
INCLUDE "MODEDEFS.BAS"      'Includes all serial comm. parameters
'-------------------------------------------------------------------------*
DEFINE HSER_RCSTA 90H       'Hardware USART
DEFINE HSER_TXSTA 20H       'parameters
DEFINE HSER_BAUD 2400       'are set
'-------------------------------------------------------------------------*
ECG_DATA    VAR BYTE
'-------------------------------------------------------------------------*
PORTA=0                     'Value of Port A is 0
PORTB=0                     'Value of Port B is 0
PORTC=0                     'Value of Port C is 0
PORTD=0                     'Value of Port D is 0
PORTE=0                     'Value of Port E is 0
ECG_DATA=0                  'Value of ECG_DATA is 0
'-------------------------------------------------------------------------*
PAUSE 100                                   'Pause 100 mili seconds
PORTD.2=0                                   'Module is set receiver mode
PORTD.3=0                                   'Module is set 433 MHz
RECEIVE:
    SERIN2 PORTB.4,396,[WAIT("M"),ECG_DATA]
      'If character "M" is received, ECG signal will be received in ECG_DATA
    HSEROUT [ECG_DATA]                      'Received ECG signal sent to PC
GOTO RECEIVE                                'Receive ECG signal again
END
'*************************************************************************
```

```
// ekgwin32r.cpp : Defines the entry point for the application.
//

#include "stdafx.h"
#include "ekgwin32r.h"
#include <time.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#include <string.h>
#include <tchar.h>

#define INSTANCES 1
#define READ_TIMEOUT 1000 // milliseconds--timeout is zero, so reply comes at that
    millisecond immediately
#define TRYOUT 10

int tryout = 0;
typedef struct tm tm;
tm * tarih(void);

tm * tarih(void)
{
    time_t *saat;
    saat = (time_t *) malloc(sizeof(time_t));
    time(saat);                         //seconds since 1970
    tm *saat1;
    saat1 = ( tm *) malloc(sizeof( tm));
    saat1 = localtime(saat);            //exchanging the seconds to some format
    return saat1;
}
}
int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPTSTR    lpCmdLine,
                     int       nCmdShow)
{

MSG msg;
char isimuz;
DWORD  sayi1;
HANDLE control;
int sayi;
FILE *logFile;
FILE  *ayar;
FILE *yazma;
int k,a,b;
int j=0;
int m=0;
int l=0;
char *alconfig;
char *portName;
char *alisim;

if(!nCmdShow) return -1;

logFile = fopen("log.txt", "w");
ayar = fopen("config", "r");

        alconfig = (char *)  malloc (11);
        portName = (char *) malloc(5);
        a=fgetc(ayar);
        alisim = (char*)malloc(a+1+4);
        for(b=0 ; b<a ; b++)
        {*(alisim+b)= fgetc(ayar);}
        alisim[a] = '.';
        alisim[a+1] = 'e';
        alisim[a+2] = 'k';
        alisim[a+3] = 'g';
        *(alisim+a + 4)='\0';
        fgets(alconfig,11,ayar);
        fgets(portName,5,ayar);
```

```
        fprintf(logFile,"%s\n",alisim);
        yazma = fopen(alisim,"r");
        if(yazma!=NULL)
        {
            fprintf(logFile,"dosya mevcut: %s\n",alisim);
            fclose(yazma);
            control=CreateFile(
                alisim,
                GENERIC_WRITE | GENERIC_READ,
                0,
                NULL,
                OPEN_EXISTING,
                FILE_ATTRIBUTE_NORMAL | FILE_FLAG_WRITE_THROUGH,
                NULL);
            fprintf(logFile,"%d\n",GetLastError());
          ReadFile(control, &isimuz, sizeof(char),&sayi1, NULL);
          fprintf(logFile,"isimuz:%d\n",(int)isimuz);
          SetFilePointer(control, ( int)isimuz,NULL,FILE_CURRENT);
                while(1)
            {
                if(ReadFile(control, &sayi, sizeof(int),&sayi1, NULL)==0)
                    break;
                else
                {
                    if(sayi1==0)
                    {
                        fprintf(logFile,"ddd: %d\n",sayi1);
                        break;
                    }
                    else if(sayi != 0)
                    {
                        fprintf(logFile,"dosya mevcut: %d\n",sayi);
                        SetFilePointer(control, sayi+12,NULL,FILE_CURRENT);

                    }
                    else
                    {

                        SetFilePointer(control, -4 ,NULL,FILE_CURRENT);
                        SetEndOfFile(control);
                        break;
                    }
                }
            }
        CloseHandle(control);
        }
        else
        {
            fprintf(logFile,"dosya yaratiliyor: %s\n",alisim);
            yazma = fopen(alisim,"w");
            fprintf(yazma,"%c",(char) a);
            for(b=0;b<a;b++)
            {
                fprintf(yazma,"%c",*(alisim+b));
            }
            fclose(yazma);
        }


        fprintf(logFile,"%s\n",alconfig);
        fprintf(logFile,"%s\n",portName);
        fclose(ayar);



    tm * tarih1;
    tarih1 = tarih();
    int p=0;
    char g=0;
    fprintf(logFile,"tarih yaziliyor...\n");
    yazma = fopen(alisim, "ab");
    fwrite(&p, sizeof(int), 1, yazma);
    fprintf(yazma, "%c%c%c%c%c%c%c%c%c%c%c%c",tarih1->tm_year, tarih1->tm_mon, tarih1->
```

```
    tm_mday, tarih1->tm_hour, tarih1->tm_min , tarih1->tm_sec,g,g,g,g,g,g);
    fclose(yazma);
    fprintf(logFile,"tarih yazildi.\n");




//OPENING THE PORT


    HANDLE hComm;
    hComm = CreateFile( portName,
                    GENERIC_READ,
                    0,
                    0,
                    OPEN_EXISTING,
                    FILE_FLAG_OVERLAPPED,
                    0);
if (hComm == INVALID_HANDLE_VALUE)
{
    fprintf(logFile,"port acilamadi.\n");
    return 0;
}
else
{
    fprintf(logFile,"port acildi.\n");
// error opening port; abort

}

//SERIAL PORT SETTINGS

DCB dcb;

    FillMemory(&dcb, sizeof(dcb), 0);
    dcb.DCBlength = sizeof(dcb);
    if (!BuildCommDCB(alconfig, &dcb))
   {
      // Couldn't build the DCB. Usually a problem with the communications specification
    string.
      fprintf(logFile,"ayarlar yuklenemedi, config dosyası sorunu ya da DCB  sorunu olabilir
    .\n");
      //return FALSE;
   }
    else if (dcb.BaudRate == CBR_2400)
   {// DCB is ready for use.
    fprintf(logFile,"ayarlar yazildi.\n");
   }
   else
   {
       fprintf(logFile,"ayarlar yuklenemedi, config dosyasi sorunu ya da DCB  sorunu
    olabilir.(baud rate yanlis)\n");
   }
     //set the dcb and control if the dcb set correctly
   if(!SetCommState(hComm, &dcb))
   {
        fprintf(logFile,"ayarlar yuklenemedi.\n");
        return 0;
    }
    else
        fprintf(logFile,"ayarlar yuklendi.\n");




//READING THE SERIAL DATA

unsigned char *lpBuf;
lpBuf = ( unsigned char *) malloc(1000);
OVERLAPPED *osReader = new OVERLAPPED;
```

```
FillMemory(osReader,sizeof(OVERLAPPED), 0);
DWORD READ_BUF_SIZE = 1000;
DWORD * dwRead = new DWORD(0);
BOOL fWaitingOnRead = FALSE;


// Create the overlapped event. Must be closed before exiting to avoid a handle leak.
osReader->hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

fprintf(logFile,"olay baslatiliyor.\n");


if (osReader->hEvent == NULL)
  fprintf(logFile,"olay baslamasinda sorun cikti.");

// Error creating overlapped event; abort.

 //Configuring the commtimeouts-- necessary for healty reading - http://www.idi.ntnu.no/~
    petrovic/ir/lnp_test.c
COMMTIMEOUTS timeouts;

GetCommTimeouts(hComm, &timeouts);
timeouts.ReadIntervalTimeout = 50;
timeouts.ReadTotalTimeoutConstant  = 0; //1000;
timeouts.ReadTotalTimeoutMultiplier = 0;//1 + 8*1000/2400;
timeouts.WriteTotalTimeoutConstant =0;
timeouts.WriteTotalTimeoutMultiplier = 0;
SetCommTimeouts(hComm, &timeouts);

//SETCOMMMASK FUNCTION - enables the second parameter for the given handle
/*DWORD dwCommEvent;
 char chRead;
if(!SetCommMask(hComm, EV_RXCHAR))
{
for(; ;)
    {
        if(WaitCommEvent(hComm, &dwCommEvent,NULL))
        {
            if (ReadFile(hComm, &chRead,100,dwRead,NULL))
                printf("bir byte okundu");
            else
            {
            printf("hata var");
            break;
            }

        }

        else
        {
            printf("WaitCommEventte Hata var.\n");
            break;
        }
    }
}
else
{
    printf("set basarili\n");
}
*/

char oku;
oku = '1';
while( oku=='1')
{



if (!fWaitingOnRead)
    {
    // Issue read operation.
   if (!ReadFile(hComm, lpBuf, READ_BUF_SIZE, dwRead, osReader))
   {
```

```
        if (GetLastError() != ERROR_IO_PENDING)
        {
              fprintf(logFile,"kominikasyonda hata var\n");
              break;
        }
         // read not delayed?
         // Error in communications; report it.
        else
        {
           fWaitingOnRead = TRUE;
           fprintf(logFile,"veri bekliyor.\n");
        }
     }
    else
    {
        //read completed immediately
     yazma = fopen(alisim, "ab");
     fprintf(logFile,"okuma basarili. Okunan verisayisi: %d\n", *dwRead);
     fwrite(lpBuf,1, *dwRead, stdout);
     fflush(stdout);
     for(m=0;m<(int)*dwRead;m++)
     {
         fprintf( yazma, "%c",  *(lpBuf+m) );
         *(lpBuf+m)= 0;
         j++;
     }
     scanf("%c",&oku);
     fprintf(logFile,"dosya acildi: %s\n",alisim);
     fclose(yazma);
     fprintf(logFile,"dosyaya yazildi ve dosya kapatildi.yazilan : %u\n",*dwRead);

     continue;
    }
}
//CONTROL THE SERIAL DATA AND WRITE IT IMMEDIATELY TO THE HARDDISK
DWORD dwRes;
int c = 0;
if (fWaitingOnRead)
{
   dwRes = WaitForSingleObject(osReader->hEvent, READ_TIMEOUT);

   switch(dwRes)
   {
      // Read completed.
      case WAIT_OBJECT_0:
          if (!GetOverlappedResult(hComm, osReader, dwRead, FALSE))
              fprintf(logFile,"islem tamamlanamadi.\n");

              // Error in communications;
          else
          {  // Read completed successfully.
            fprintf(logFile,"islem tamamlandi.okunan: %d\n",*dwRead);
            if((yazma = fopen(alisim, "ab")) != NULL)
            {
                fprintf(logFile,"dosya acildi: %s\n",alisim);
                fwrite(lpBuf,1, *dwRead, stdout);
                fflush(stdout);
                for(m=0; m<*dwRead; m++)
                {
                    fprintf( yazma,"%c",  *(lpBuf+m) );
                    *(lpBuf+m)= 0;
                    j++;
                }
                    scanf("%c",&oku);
                    fclose(yazma);
                    fprintf(logFile,"dosyaya yazildi ve dosya kapatildi.yazilan : %u\n",*
     dwRead);

            }
          }
          //  Reset flag so that another opertion can be issued.
                fWaitingOnRead = FALSE;
                break;
```

```
      case WAIT_TIMEOUT:
                GetOverlappedResult(hComm, osReader, dwRead, FALSE);
                fprintf(logFile,"buffer tamamlanamadan durdu, dosyaya yaziliyor...\n");
                fprintf(logFile,"okunan veri sayisi:%u\n", *dwRead);
                fWaitingOnRead = FALSE;

         if((yazma = fopen(alisim, "ab")) != NULL)
         {
            if( *dwRead == 0)
        {

                fprintf(logFile,"veri alinamadi.\n");

        }
         else
        {       fwrite(lpBuf,1, *dwRead, stdout);
                fflush(stdout);
                for(m=0;m<*dwRead;m++)
                {
                    fprintf( yazma, "%c",  *(lpBuf+m) );
                    *(lpBuf+m)= 0;
                    j++;

                }

                scanf("%c",&oku);
                fprintf(logFile,"dosyaya yazildi ve dosya kapatildi.\n");
                fclose(yazma);
        }
            }

         break;
            // Operation isn't complete yet. fWaitingOnRead flag isn't
            // changed since I'll loop back around, and I don't want
            // to issue another read until the first one finishes.
            // This is a good time to do some background work.


      default:

            fprintf(logFile,"%d WaitForSingleObject hatasi, overlapped e bak..",GetLastError↙
    ());

            // Error in the WaitForSingleObject; abort.
            // This indicates a problem with the OVERLAPPED structure's
            // event handle.
          break;

   }

}
}

tarih1 = tarih();
yazma = fopen(alisim, "r+b");
char name_size;
//fscanf(yazma, "%c", &name_size);
fseek(yazma,  -j -6 , SEEK_END);
fprintf(yazma, "%c%c%c%c%c%c", tarih1->tm_year , tarih1->tm_mon, tarih1->tm_mday , tarih1-> ↙
    tm_hour, tarih1->tm_min , tarih1->tm_sec);
fseek(yazma, -16, SEEK_CUR);
fwrite(&j, sizeof(int), 1, yazma);
fclose(yazma);

fclose(logFile);
CloseHandle(osReader->hEvent);

return 0;
}
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.Xml.Serialization;
using System.IO;


namespace EKGP
{
    using EKGData = EKGP.EKGDataFileProcessor.PatientData.EKGData;
    using PatientData = EKGP.EKGDataFileProcessor.PatientData;


    public partial class mForm : Form
    {
        public class PipeInstance
        {
            public IntPtr pipeHandle;
            public byte[] inputBuffer;
            public Int32 inputBufferSize;
            public byte[] outputBuffer;
            public Int32 outputBufferSize;
            Win32Com.OVERLAPPED overlapped;
        }

        public class FormOptions
        {
            public float xScaleFactor = 1.0F;
            public float yScaleFactor = 1.0F;
            public float properTension = 0F;
            public string comPort = "COM1";
            public string baudRate = "2400";
            public string parity = "n";
            public string nBits = "8";
            public string stopBit = "1";
            public string patientName = "";
        }

        [DllImport("user32.dll")]
        unsafe
        internal static extern Boolean PostMessage(void* h, uint msgs, UIntPtr wParam,
    IntPtr lParam);

        internal const UInt32 WM_CLOSE = 0x0010;

        #region fieldlist
        private ArrayList pointList = null;
        private ArrayList pLList = null;
        public List<PatientData> pData = new List<PatientData>();
        private Graphics graphicsHandler;
        private BufferedGraphicsContext bufferedContext;
        private BufferedGraphics bufferedGraphics;
        public FormOptions formOptions = new FormOptions();
        private float properXFactor = 1.0F;
        private float properYFactor = 1.0F;
        private bool provideLiveData = false;
        public bool serviceStarted = false;
        private Process serviceProcessInfo = null;
        public List<string> patientList;
        public BinaryWriter bWriter = null;
        public Stream fStream = null;
        public StreamWriter wStream = null;
        public StreamReader rStream = null;
        public string loadedPatient;
```

```
        public int xTranslation = 0;
        public int yTranslation = 0;
        public int xOffset = 0;
        private const int WM_QUERYENDSESSION = 0x11;
        private bool exitApplication = false;
        private int uThreshold = 150;
        private float dThreshold;
        private int wl1Threshold = 5;
        private int wl2Threshold =130;
        private int bpmCount = 0;
        private int bpm = 0;
        #endregion

        public mForm()
        {
            InitializeComponent();
            graphicsHandler = rpfgroupBox.CreateGraphics();
            bufferedContext = new BufferedGraphicsContext();
            bufferedGraphics = bufferedContext.Allocate(
                rpfgroupBox.CreateGraphics(), new Rectangle(rpfgroupBox.ClientRectangle.Left
    + 20, rpfgroupBox.ClientRectangle.Top + 20,
                rpfgroupBox.Width - 40, rpfgroupBox.Height - 40));
            properXFactor = 1.0F;
            properYFactor = (float)(rpfgroupBox.Height - 82) / 256.0F;

        }

        public static List<string> GetPatientList()
        {
            DirectoryInfo dInfo = new DirectoryInfo(Application.StartupPath);
            FileInfo[] filesInCurDir = dInfo.GetFiles("*.ekg");
            BinaryReader bReader;
            Stream fStream;
            List<string> patientList = new List<string>();
            foreach (FileInfo cFile in filesInCurDir)
            {
                fStream = cFile.OpenRead();
                bReader = new BinaryReader(fStream);
                patientList.Add(bReader.ReadString());
                bReader.Close();
            }
            return patientList;
        }

        protected override void WndProc(ref Message m)
        {
            if (m.Msg == WM_QUERYENDSESSION)
            {
                exitApplication = true;
            }
            base.WndProc(ref m);
        }

        private void LoadEKGGraphic(ArrayList pointsList, int offset, int numOfSegments)
        {
            Pen myPen = new Pen(Brushes.Black);
            Point[] thePointFList = new Point[pointsList.Count];
            for (int i = 0; i < pointsList.Count; i++)
                thePointFList[i] = new Point(((Point)pointsList[i]).X, ((Point)pointsList
    [i]).Y);

            Matrix tMatrix = new Matrix();
            tMatrix.Scale(properXFactor, properYFactor);

            tMatrix.TransformPoints(thePointFList);
            Matrix tMatrix2 = new Matrix();
            tMatrix2.Translate(41F, 41F);
            tMatrix2.TransformPoints(thePointFList);
            Matrix tMatrix3 = new Matrix();
            if (provideLiveData && rpfgroupBox.Width - 82 > thePointFList.Length)
            {
                tMatrix3.Translate(rpfgroupBox.Width - thePointFList.Length - 82, 0);
                tMatrix3.TransformPoints(thePointFList);
```

```
            }

            Matrix tMatrix4 = new Matrix();
            tMatrix4.Translate(xTranslation, 0);
            tMatrix4.TransformPoints(thePointFList);

            bufferedGraphics.Graphics.DrawCurve(myPen, thePointFList, offset, Math.Min
    (thePointFList.Length - 1, rpfgroupBox.Width - 82), formOptions.properTension);
        }

        private void LoadEKGGraphicFromArray(Point[] pointsList, int offset, int
    numOfSegments)
        {
            Pen myPen = new Pen(Brushes.White);
            Matrix tMatrix = new Matrix();
            tMatrix.Scale(properXFactor, properYFactor);

            tMatrix.TransformPoints(pointsList);
            Matrix tMatrix2 = new Matrix();
            tMatrix2.Translate(41F, 41F);
            tMatrix2.TransformPoints(pointsList);
            Matrix tMatrix3 = new Matrix();
            if (provideLiveData && rpfgroupBox.Width - 82 > pointsList.Length)
            {
                tMatrix3.Translate(rpfgroupBox.Width - pointsList.Length, 0);
                tMatrix3.TransformPoints(pointsList);
            }

            bufferedGraphics.Graphics.DrawCurve(myPen, pointsList, offset, Math.Min
    (pointsList.Length - 1, rpfgroupBox.Width - 82), formOptions.properTension);
        }


        private void LoadPoints(byte[] yOffsets)
        {
            pointList = new ArrayList();
            int xOffset = 0;

            foreach (byte yOffset in yOffsets)
            {
                pointList.Add(new Point(xOffset, 255 - yOffset));
                xOffset++;
            }
        }

        private void mForm_Resize(object sender, EventArgs e)
        {
            lptreeView.Size = new System.Drawing.Size(
                msplitContainer.Panel1.ClientSize.Width - 3,
                msplitContainer.Panel1.ClientSize.Height);
            pDataGroupBox.Size = new System.Drawing.Size(
                msplitContainer.Panel2.ClientSize.Width - 6,
                Math.Max(msplitContainer.Panel2.Height / 4 - 4, 132));

            rpfgroupBox.Location = new Point(3, pDataGroupBox.Height + 10);

            rpfgroupBox.Size = new System.Drawing.Size(
                msplitContainer.Panel2.ClientSize.Width - 6 ,
                msplitContainer.Panel2.ClientSize.Height - pDataGroupBox.Height - 13 );

            bufferedContext.MaximumBuffer = rpfgroupBox.Size ;
            bufferedGraphics = bufferedContext.Allocate(
                rpfgroupBox.CreateGraphics(), new Rectangle(rpfgroupBox.ClientRectangle.Left
     + 20, rpfgroupBox.ClientRectangle.Top + 20,
                rpfgroupBox.Width - 40, rpfgroupBox.Height - 40));

           Rectangle newR = new Rectangle(rpfgroupBox.ClientRectangle.Left, rpfgroupBox.
    ClientRectangle.Top,
                rpfgroupBox.ClientRectangle.Width, rpfgroupBox.ClientRectangle.Height);
                newR.Inflate(-20, -20);
            bufferedGraphics.Graphics.FillRectangle(Brushes.MidnightBlue, newR);
            newR.Inflate(-20, -20);
            bufferedGraphics.Graphics.DrawRectangle(Pens.LightSteelBlue, newR);
```

```
            if (pointList != null)
            {
                LoadEKGGraphic(pointList, Math.Abs(xTranslation), Math.Min(pointList.Count - 1, rpfgroupBox.Width - 82));
                //drawSRectangle();
                drawTRectangle();
                drawSRectangle();
                //drawFRectangle();
            }

            properYFactor = (float)(rpfgroupBox.Height - 82) / 255.0F;

            rpfgroupBox.Invalidate();

        }

        private void msplitContainer_SplitterMoved(object sender, SplitterEventArgs e)
        {
            mForm_Resize(null, null);
        }

        private void mForm_Load(object sender, EventArgs e)
        {
            this.SForm = new SearchForm();
            if (File.Exists(Application.StartupPath + @"\" + "options.xml"))
            {
                XmlSerializer xmlSerializer = new XmlSerializer(typeof(FormOptions));
                TextReader txtReader = new StreamReader(Application.StartupPath + @"\" + "options.xml");
                formOptions = (FormOptions)xmlSerializer.Deserialize(txtReader);
            }
            else
            {
                if (GetPatientList().Count == 0)
                {
                    MessageBox.Show("You must create at least one patient record in order to procede.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    this.Enabled = false;
                    oForm optionsForm = new oForm();
                    optionsForm.groupBox1.Enabled = false;
                    optionsForm.groupBox2.Enabled = false;
                    optionsForm.Show();
                }
            }
            Process[] commservices = Process.GetProcessesByName("ekgwin32rm");
            if (commservices.Length != 0)
            {
                startServiceToolStripMenuItem.Text = "Stop Service";
                switchToLiveDataToolStripMenuItem.Enabled = true;
                serviceStarted = true;
                //wStream = commservices[0].StandardInput;
                serviceProcessInfo = commservices[0];
                //commservices[0].BeginOutputReadLine();
            }

            string[] fList = Directory.GetFiles(Application.StartupPath, "*.ekg");
            foreach (string pFile in fList)
            {
                try
                {
                    pData.Add(EKGDataFileProcessor.ProcessEKGFile(pFile));
                }
                catch(IOException exp)
                {
                    MessageBox.Show("The program can't load the patient data file: " +
                        fList + "\n" + exp.Message);
                }
            }

            TreeNode RootNode;
```

```
            foreach (EKGDataFileProcessor.PatientData pDElement in pData)
            {
                RootNode = new TreeNode(pDElement.patientName);
                foreach (EKGDataFileProcessor.PatientData.EKGData ekgData in pDElement.
    ekgDataList)
                    RootNode.Nodes.Add(ekgData.dataStartDate.ToString() + " to " + ekgData.
    dataEndDate.ToString());
                lptreeView.Nodes.Add(RootNode);
            }

            SForm.Hide();

            drawMRectangle();
            drawSRectangle();
            rpfgroupBox.Refresh();

        }

        private void ımportEKGDataToolStripMenuItem_Click(object sender, EventArgs e)
        {
            openEKGFileDialog.FileName = Directory.GetCurrentDirectory();
            openEKGFileDialog.ShowDialog();
        }

        private void openEKGFileDialog_FileOk(object sender, CancelEventArgs e)
        {
            pData.Clear();
            foreach(PatientData pList in pData)
            {
                if (pList.patientName == openEKGFileDialog.FileNames[0])
                    return;
            }

            pData.Add(EKGDataFileProcessor.ProcessEKGFile(openEKGFileDialog.FileNames[0]));
            TreeNode newNode;
            newNode = new TreeNode(pData[0].patientName);
            int index = lptreeView.Nodes.Add(newNode);

            foreach (EKGData ekgData in pData[0].ekgDataList)
            {
                newNode = new TreeNode(ekgData.dataStartDate.ToString() + " to " + ekgData.
    dataEndDate.ToString());
                lptreeView.Nodes[index].Nodes.Add(newNode);
            }

            LoadPatientRecord(openEKGFileDialog.FileName.Substring(0, openEKGFileDialog.
    FileName.LastIndexOf("."))
                + @"\" + pData[0].patientName + ".xml");
            pDataGroupBox.Refresh();
            rpfgroupBox.Refresh();
            lptreeView.Refresh();
            closeToolStripMenuItem.Enabled = true;


        }
        private void rpfgroupBox_Paint(object sender, PaintEventArgs e)
        {
            bufferedGraphics.Render(e.Graphics);

        }
        private void drawMRectangle()
        {
            Rectangle newR = new Rectangle(rpfgroupBox.ClientRectangle.Left, rpfgroupBox.
    ClientRectangle.Top,
                rpfgroupBox.ClientRectangle.Width, rpfgroupBox.ClientRectangle.Height);
            newR.Inflate(-20, -20);
            bufferedGraphics.Graphics.FillRectangle(Brushes.MidnightBlue, newR);
            newR.Inflate(-20, -20);
            bufferedGraphics.Graphics.DrawRectangle(Pens.LightSteelBlue, newR);

        }
        private void drawSRectangle()
        {
```

```
            Rectangle newR = new Rectangle(rpfgroupBox.ClientRectangle.Left, rpfgroupBox.
    ClientRectangle.Top,
                rpfgroupBox.ClientRectangle.Width, rpfgroupBox.ClientRectangle.Height);

            newR.Inflate(-40, -40);
            bufferedGraphics.Graphics.DrawRectangle(Pens.LightSteelBlue, newR);
        }
        private void drawTRectangle()
        {
            Rectangle newR = new Rectangle(rpfgroupBox.ClientRectangle.Left, rpfgroupBox.
    ClientRectangle.Top,
                40, rpfgroupBox.ClientRectangle.Bottom);

            bufferedGraphics.Graphics.FillRectangle(Pens.MidnightBlue.Brush, newR);
        }
        private void drawFRectangle()
        {
            Rectangle newR = new Rectangle(rpfgroupBox.ClientRectangle.Right - 40,
    rpfgroupBox.ClientRectangle.Top,
                rpfgroupBox.ClientRectangle.Right, rpfgroupBox.Bottom);
            bufferedGraphics.Graphics.FillRectangle(Pens.MidnightBlue.Brush, newR);
        }

        private List<Point> getPointsOnLine(int lOffset, ArrayList pointList)
        {
            List<Point> pOnLine = new List<Point>();
            Point thePoint;
            Point nPoint;

            for(int i = lOffset; i < Math.Min(lOffset + 500, pointList.Count); i++)
            {
                thePoint = ((Point)pointList[i]);

                if (thePoint.Y == uThreshold)
                {
                    pOnLine.Add(new Point(thePoint.X, thePoint.Y));
                    continue;
                }
                if(i == pointList.Count - 1)
                    continue;
                nPoint = ((Point)pointList[i + 1]);
                if (nPoint.Y == uThreshold)
                    continue;
                if(Math.Sign(uThreshold - thePoint.Y) != Math.Sign(uThreshold - ((Point)
    pointList[i+1]).Y))
                {
                    if (Math.Abs(uThreshold - thePoint.Y) < Math.Abs(uThreshold - ((Point)
    pointList[i + 1]).Y))
                        pOnLine.Add(new Point(thePoint.X, thePoint.Y));
                    else
                        pOnLine.Add(new Point(((Point)pointList[i + 1]).X, ((Point)pointList
    [i + 1]).Y));
                    //i++;
                }
            }
            return pOnLine;
        }

        private int getIntervals(List<Point> pList)
        {
            Point thePoint;
            Point nPoint;
            int iCount = 0;
            int interval;

            for(int i = 0; i < pList.Count; i++)
            {
                if (i == pList.Count - 1)
                    break;
                thePoint = ((Point)pList[i]);
                nPoint = ((Point)pList[i + 1]);

                interval = Math.Abs(nPoint.X - thePoint.X);
```

```
            if (interval > wl1Threshold && interval < wl2Threshold)
                iCount++;
        }
        return iCount;
    }
    private void updateBPM()
    {
        bufferedGraphics.Graphics.DrawString("BPM: " + bpm.ToString(), new Font
(FontFamily.GenericMonospace, 14, FontStyle.Bold), Brushes.WhiteSmoke,
            rpfgroupBox.ClientRectangle.Right - 120, rpfgroupBox.ClientRectangle.Bottom
- 40);
    }
    private void mtimer_Tick(object sender, EventArgs e)
    {
        if (pLList == null || pLList.Count <= 19)
        {
            //drawMRectangle();
            //drawSRectangle();
            //rpfgroupBox.Update();
            return;
        }
        if (bpmCount == 120)
        {
            bpm = getIntervals(getPointsOnLine(Math.Max(0, xOffset - 6000), pLList)) *
10;
            bpmCount -= 20;
        }

        int i = pLList.Count - 15000;
        ArrayList newPList = new ArrayList();
        Point thePoint;
        xOffset += 5;
        if (pLList.Count > 15000)
            pLList.RemoveRange(0, i);
        else i = 0;
        xOffset -= i;

        Point[] points = new Point[xOffset - ((xOffset >= rpfgroupBox.Width - 82) ?
xOffset - rpfgroupBox.Width + 82 : 0)];
        int m = 0;
        for(int k = 0; k < pLList.Count; k++)
        {
            thePoint = (Point)(pLList[k]);
            newPList.Add(new Point(thePoint.X - i, thePoint.Y));
        }

        for (int k = (xOffset >= rpfgroupBox.Width - 82) ? xOffset - rpfgroupBox.Width +
 82 : 0;
            k < xOffset; k++)
        {
            if (m == pLList.Count)
                break;
            thePoint = (Point)(pLList[k]);
            points[m].X = m;
            points[m].Y = thePoint.Y;
            m++;
        }

        pLList = newPList;
        bpmCount++;

        if (provideLiveData)
        {
            drawMRectangle();
            if (bpm != 0)
                updateBPM();
            LoadEKGGraphic(new ArrayList(points), 0, 0);
            drawSRectangle();

            bufferedGraphics.Render();
            rpfgroupBox.Update();
        }
```

```
        }

        private void closeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (pData == null || pData.Count == 0)
            {
                SForm.sOKButton.Enabled = false;
                return;
            }
            SForm.comboBox1.Items.Clear();
            foreach (EKGDataFileProcessor.PatientData pListData in pData)
                SForm.comboBox1.Items.Add(pListData.patientName);
            SForm.comboBox1.SelectedIndex = 0;
            SForm.Show();
        }

        private void closeToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            exitApplication = true;
            if (serviceStarted)
            {
                startServiceToolStripMenuItem_Click(null, null);
                serviceStarted = false;
            }
            Application.Exit();
        }

        public byte[] ConvertStringToByteArray(string stringToConvert)
        {
            return UnicodeEncoding.Default.GetBytes(stringToConvert);//serviceProcessInfo.
    StandardOutput.CurrentEncoding.GetBytes(stringToConvert);
        }


        private void startServiceToolStripMenuItem_Click(object sender, EventArgs e)
        {
            int lasterror;
            if(serviceStarted)
            {
                startServiceToolStripMenuItem.Enabled = false;
                if(wStream != null) wStream.Write('0');
                serviceProcessInfo.WaitForExit(4000);
                if (!serviceProcessInfo.HasExited)
                {
                    unsafe {
                        if(!PostMessage(serviceProcessInfo. MainWindowHandle.ToPointer(),
    WM_CLOSE, UIntPtr.Zero, IntPtr.Zero))
                        lasterror = Marshal.GetLastWin32Error();
                    }
                    serviceProcessInfo.WaitForExit(3000);
                }
                if (!serviceProcessInfo.HasExited)
                    serviceProcessInfo.Kill();
                serviceStarted = false;
                startServiceToolStripMenuItem.Text = "Start Service";
                switchToLiveDataToolStripMenuItem.Enabled = false;
                switchToLiveDataToolStripMenuItem.Text = "Switch To Live Data";
                mtimer.Enabled = false;
                provideLiveData = false;
                drawMRectangle();
                drawSRectangle();
                bufferedGraphics.Render();
                rpfgroupBox.Update();
                startServiceToolStripMenuItem.Enabled = true;

                return;
            }

            Stream cfStream = new FileStream("config", FileMode.Open, FileAccess.Read);
            BinaryReader bRead = new BinaryReader(cfStream);
            LoadPatientRecord(bRead.ReadString() + ".xml");
            bRead.Close();
```

```
            startServiceToolStripMenuItem.Enabled = false;
            Process process = new Process();
            process.StartInfo.FileName = "commservice.exe";

            process.StartInfo.UseShellExecute = false;
            process.StartInfo.RedirectStandardInput = true;
            process.StartInfo.RedirectStandardOutput = true;
            process.EnableRaisingEvents = true;
            process.SynchronizingObject = this;
            process.Exited += new EventHandler(process_Exited);

            process.StartInfo.CreateNoWindow = false;
            process.OutputDataReceived += new DataReceivedEventHandler
    (process_OutputDataReceived);

            try
            {
                process.Start();
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
                this.serviceProcessInfo = null;
                serviceStarted = false;
                return;
            }
            finally
            {
                startServiceToolStripMenuItem.Enabled = true;
            }
            mtimer.Enabled = true;
            startServiceToolStripMenuItem.Text = "Stop Service";
            switchToLiveDataToolStripMenuItem.Enabled = true;
            this.serviceProcessInfo = process;
            wStream = process.StandardInput;
            wStream.AutoFlush = true;

            process.BeginOutputReadLine();
            wStream.Write('1');
            serviceStarted = true;


        }

        void process_Exited(object sender, EventArgs e)
        {
            if (serviceProcessInfo.ExitCode != 0)
                MessageBox.Show("Service process has exited unexpectedly with error code: "
    +
                    serviceProcessInfo.ExitCode, "Error", MessageBoxButtons.OK,
    MessageBoxIcon.Error);


            //serviceProcessInfo.Close();
            //serviceProcessInfo = null;
            return;

        }

        void  process_OutputDataReceived(object sender, DataReceivedEventArgs outLine)
        {
            if(!String.IsNullOrEmpty(outLine.Data))
            {
                byte[] dataArray = ConvertStringToByteArray(outLine.Data);
                if (pLList == null) pLList = new ArrayList();
                foreach(byte bPoint in dataArray)
                    pLList.Add(new Point(pLList.Count, 255 - bPoint));
            }
            wStream.Write('1');
        }

        private void switchToLiveDataToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            provideLiveData = provideLiveData ? false : true;

            switchToLiveDataToolStripMenuItem.Text = (provideLiveData ? "Stop Live Data" :
    "Switch To Live Data");
            if (provideLiveData)
            {
                //mtimer.Enabled = true;
                xTranslation = 0;
            }
            //else
                //mtimer.Enabled = false;
            pointList = null;
            drawMRectangle();
            drawSRectangle();
            bufferedGraphics.Render();
            rpfgroupBox.Update();
        }

        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            float xScaleFactor = formOptions.xScaleFactor;
            float yScaleFactor = formOptions.yScaleFactor;

            int lPoint = contextMenuStrip1.RectangleToScreen(new Rectangle(0, 0, 10, 10)).
    Left - lptreeView.RectangleToScreen(new Rectangle(0, 0, 10, 10)).Left;
            int tPoint = contextMenuStrip1.RectangleToScreen(new Rectangle(0, 0, 10, 10)).
    Top - lptreeView.RectangleToScreen(new Rectangle(0, 0, 10, 10)).Top; ;
            TreeNode exportedNode = lptreeView.GetNodeAt(lPoint, tPoint);

            if (exportedNode.Parent == null) return;
            contextMenuStrip1.Items[0].Enabled = false;
            int fIndex = exportedNode.Parent.Index;
            EKGData ekgDataToConvert = EKGDataFileProcessor.GetDataBlock(
                pData[exportedNode.Parent.Index], exportedNode.Index);

            int bitmapWidth = (int)(xScaleFactor * (ekgDataToConvert.ekgData.Length + 40));
            int bitmapHeight = (int)(yScaleFactor * (256 + 40));
            Bitmap exportedImage = new Bitmap(bitmapWidth, bitmapHeight, System.Drawing.
    Imaging.PixelFormat.Format24bppRgb);
            Graphics handleToBitmap = Graphics.FromImage(exportedImage);

            int indexer = 0;
            PointF[] pointsList = new PointF[ekgDataToConvert.ekgData.Length];
            foreach (byte b in ekgDataToConvert.ekgData)
                pointsList[indexer] = (new PointF(indexer++, 255 - b));
            Matrix tMatrix = new Matrix();
            tMatrix.Scale(xScaleFactor, yScaleFactor);
            tMatrix.Translate(20, 20);
            tMatrix.TransformPoints(pointsList);

            handleToBitmap.FillRectangle(Brushes.Black, new Rectangle(0, 0, bitmapWidth,
    bitmapHeight));
            handleToBitmap.DrawCurve(Pens.ForestGreen, pointsList, formOptions.
    properTension);
            handleToBitmap.Save();
            exportedImage.Save(Application.StartupPath + @"\" + pData[fIndex].patientName/*
    + "data on " + pData.ekgDataList[exportedNode.Index].dataStartDate */+ ".bmp");
        }

        private void backgroundWorker1_RunWorkerCompleted(object sender,
    RunWorkerCompletedEventArgs e)
        {
            contextMenuStrip1.Items[0].Enabled = true;
        }

        private void lptreeView_MouseClick(object sender, MouseEventArgs e)
        {
            TreeNode selectedNode = null;
            selectedNode = lptreeView.GetNodeAt(e.X, e.Y);
            if (e.Button == MouseButtons.Right)
                contextMenuStrip1.Show(lptreeView, e.X, e.Y);
        }
```

```
        private void exportEKGDataToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            float xScaleFactor = formOptions.xScaleFactor;
            float yScaleFactor = formOptions.yScaleFactor;

            int lPoint = contextMenuStrip1.RectangleToScreen(new Rectangle(0, 0, 10, 10)).
    Left - lptreeView.RectangleToScreen(new Rectangle(0, 0, 10, 10)).Left;
            int tPoint = contextMenuStrip1.RectangleToScreen(new Rectangle(0, 0, 10, 10)).
    Top - lptreeView.RectangleToScreen(new Rectangle(0, 0, 10, 10)).Top; ;
            TreeNode exportedNode = lptreeView.GetNodeAt(lPoint, tPoint);

            if (exportedNode.Parent == null) return;
            contextMenuStrip1.Items[0].Enabled = false;
            int fIndex = exportedNode.Parent.Index;
            EKGData ekgDataToConvert = EKGDataFileProcessor.GetDataBlock(
                pData[exportedNode.Parent.Index], exportedNode.Index);

            int bitmapWidth = (int)(xScaleFactor * (ekgDataToConvert.ekgData.Length + 40));
            int bitmapHeight = (int)(yScaleFactor * (256 + 40));
            Bitmap exportedImage = new Bitmap(bitmapWidth, bitmapHeight, System.Drawing.
    Imaging.PixelFormat.Format24bppRgb);
            Graphics handleToBitmap = Graphics.FromImage(exportedImage);

            int indexer = 0;
            PointF[] pointsList = new PointF[ekgDataToConvert.ekgData.Length];
            foreach (byte b in ekgDataToConvert.ekgData)
                pointsList[indexer] = (new PointF(indexer++, 255 - b));
            Matrix tMatrix = new Matrix();
            tMatrix.Scale(xScaleFactor, yScaleFactor);
            tMatrix.Translate(20, 20);
            tMatrix.TransformPoints(pointsList);

            handleToBitmap.FillRectangle(Brushes.Black, new Rectangle(0, 0, bitmapWidth,
    bitmapHeight));
            handleToBitmap.DrawCurve(Pens.ForestGreen, pointsList, formOptions.
    properTension);
            handleToBitmap.Save();
            exportedImage.Save(Application.StartupPath + @"\" + pData[fIndex].patientName/*
    + "data on " + pData.ekgDataList[exportedNode.Index].dataStartDate */+ ".bmp");
            //backgroundWorker1.RunWorkerAsync();
        }

        private void optionsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Enabled = false;
            oForm optionsForm = new oForm();
            optionsForm.Show();
        }

        private void LoadPatientRecord(string pName)
        {
            oForm.PatientRecord pRecord;
            if (File.Exists(pName))
            {
                XmlSerializer xmlSerializer = new XmlSerializer(typeof(oForm.PatientRecord))
    ;
                TextReader txtReader = new StreamReader(pName);
                pRecord = (oForm.PatientRecord) xmlSerializer.Deserialize(txtReader);

                pNameL.Text = "Patient Name: " + pRecord.patientName;
                pAgeL.Text = "Patient Age: " + pRecord.patientAge;
                pOtherL.Text = "Other: " + pRecord.Other;

                if (File.Exists(pRecord.pictureAddress))
                {
                    pictureBox1.Image = Image.FromFile(pRecord.pictureAddress);
                }
            }
            else
            {
                MessageBox.Show("The program can't load the patient record!", "Warning",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                //pNameL.Text = "Patient Name: " + pData[0].patientName;
```

```
        }
    }


    private void lptreeView_NodeMouseDoubleClick(object sender,
TreeNodeMouseClickEventArgs e)
    {
        if (e.Node.Parent == null)
        {
            if (loadedPatient == e.Node.Text) return;
            LoadPatientRecord(Application.StartupPath + @"\" + e.Node.Text + ".xml");
            loadedPatient = e.Node.Text;
            return;
        }
        else xTranslation = 0;
        if (provideLiveData)
        {
            provideLiveData = false;
            switchToLiveDataToolStripMenuItem.Text = "Switch To Live Data";
        }
        LoadPatientRecord(Application.StartupPath + @"\" + e.Node.Parent.Text + ".xml");
        loadedPatient = e.Node.Text;

        LoadPoints((EKGDataFileProcessor.GetDataBlock(pData[e.Node.Parent.Index], e.Node
.Index)).ekgData);

        drawMRectangle();
        drawSRectangle();
        LoadEKGGraphic(pointList, 0 , 0);
        bufferedGraphics.Render();
        rpfgroupBox.Update();
    }

    private void exportEKGDataToolStripMenuItem1_DropDownItemClicked(object sender,
ToolStripItemClickedEventArgs e)
    {
    }

    private void rpfgroupBox_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)
    {
        if (provideLiveData) return;
        if (pointList == null) return;
        if (e.KeyCode != Keys.Left && e.KeyCode != Keys.Right)
            return;
        drawMRectangle();
        if (e.KeyCode == Keys.Left)
        {
            xTranslation += 4;
            if (xTranslation >= 0) xTranslation = 0;
            LoadEKGGraphic(pointList, Math.Abs(xTranslation), 0);
        }
        if (e.KeyCode == Keys.Right)
        {
            xTranslation -= 4;
            if (((-1 * xTranslation) + rpfgroupBox.Width - 82) >= pointList.Count)
                xTranslation += 4;
            LoadEKGGraphic(pointList, Math.Abs(xTranslation), 0);
        }
        drawTRectangle();
        drawFRectangle();
        drawSRectangle();
        bufferedGraphics.Render();
        rpfgroupBox.Update();
    }

    private void rpfgroupBox_Leave(object sender, EventArgs e)
    {
        rpfgroupBox.Focus();
    }

    private void rpfgroupBox_MouseHover(object sender, EventArgs e)
    {
        rpfgroupBox.Focus();
```

```
        }

        private void notifyIcon1_DoubleClick(object sender, EventArgs e)
        {    notifyIcon1.Visible = false;
             this.Visible = true;
        }

        private void mForm_VisibleChanged(object sender, EventArgs e)
        {
            //base.OnVisibleChanged(e);
            notifyIcon1.Visible = Visible ? false : true;

        }

        private void mForm_FormClosing(object sender, FormClosingEventArgs e)
        {
            switch (exitApplication)
            {
                case true:
                    e.Cancel = false;
                    notifyIcon1.Visible = false;
                    break;
                case false:
                    e.Cancel = true;
                    Visible = false;
                    break;
            }
            //base.OnClosing(e);
        }

    }

    public class EKGDataFileProcessor
    {
        public class PatientData
        {
            public class EKGData
            {
                public DateTime dataStartDate;
                public DateTime dataEndDate;
                public byte[] ekgData;
                public int dSize = 0;

                internal EKGData(byte[] dataStartDate, byte[] dataEndDate, byte[] ekg)
                {
                      this.dataStartDate = new DateTime((int)dataStartDate[0] + 1900, (int) ↙
    dataStartDate[1] + 1,
                            (int)dataStartDate[2], (int)dataStartDate[3],
                            (int)dataStartDate[4], (int)dataStartDate[5]);

                      this.dataEndDate = new DateTime((int)dataEndDate[0] + 1900, (int)      ↙
    dataEndDate[1] + 1,
                            (int)dataEndDate[2], (int)dataEndDate[3],
                            (int)dataEndDate[4], (int)dataEndDate[5]);
                      ekgData = ekg;
                }
            }
            public string patientName;
            public List<EKGData> ekgDataList;
            public string pFileAddress;
            public List<long> blockList = new List<long>();

        }

        private EKGDataFileProcessor() {}

        public static PatientData ProcessEKGFile(string fileName)
        {
            Stream fstream = new FileStream(fileName, FileMode.Open, FileAccess.Read,        ↙
    FileShare.ReadWrite);
            BinaryReader freader = new BinaryReader(fstream);

            PatientData patientData = new PatientData();
```

```
            uint binaryDataSize;

            patientData.ekgDataList = new List<EKGData>();

            patientData.patientName = freader.ReadString();

            patientData.pFileAddress = fileName;

            while(freader.BaseStream.Position < freader.BaseStream.Length)
            {
                patientData.blockList.Add(freader.BaseStream.Position);
                binaryDataSize = freader.ReadUInt32();
                if (binaryDataSize == 0) break;
                patientData.ekgDataList.Add(new EKGData(
                    freader.ReadBytes(6),
                    freader.ReadBytes(6),
                    null));

                freader.BaseStream.Seek(binaryDataSize, SeekOrigin.Current);
                //freader.ReadBytes((int)binaryDataSize)));

                //break;
            }
            freader.Close();
            fstream.Close();

            return patientData;
        }
        public static EKGData GetDataBlock(PatientData pData, int index)
        {
            if(pData == null)
                throw new NullReferenceException("Null PatientData reference");
            if(index >= pData.blockList.Count)
                throw new IndexOutOfRangeException("Data block index out of range.");

            Stream fStream = new FileStream(pData.pFileAddress, FileMode.Open, FileAccess.
    Read, FileShare.ReadWrite);
            BinaryReader bReader = new BinaryReader(fStream);

            fStream.Seek(pData.blockList[index], SeekOrigin.Begin);

            uint binaryDataSize = bReader.ReadUInt32();

            EKGData ekgData = new EKGData(bReader.ReadBytes(6),
                bReader.ReadBytes(6),
                bReader.ReadBytes((int)binaryDataSize));

            return ekgData;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Xml.Serialization;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace EKGP
{
    public partial class oForm : Form
    {
        public class PatientRecord
        {
            public string patientName;
            public int patientAge;
            public string Other;
            public string pictureAddress;
        }

        public oForm()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            mForm ownerForm = (mForm)Application.OpenForms["mForm"];
            float tmpValue1, tmpValue2, tmpValue3, tmpValue4;
            uint tmpValue5;
            bool returnValue;
            bool serviceSettingsChanged = false;
            bool graphicSettingsChanged = false;
            try
            {
                returnValue = float.TryParse(textBox7.Text, out tmpValue1);
                if (returnValue)
                {
                    graphicSettingsChanged = true;
                }
                else throw new Exception("Invalid value for X Scale Factor!");

                returnValue = float.TryParse(textBox8.Text, out tmpValue2);
                if (returnValue)
                {
                    graphicSettingsChanged = true;
                }
                else throw new Exception("Invalid value for Y Scale Factor!");

                returnValue = float.TryParse(textBox9.Text, out tmpValue3);
                if (returnValue)
                {
                    graphicSettingsChanged = true;
                }
                else throw new Exception("Invalid value for Tension");

                Regex rEx = new Regex("COM[1-9]");
                Match rMatch = rEx.Match(textBox1.Text, 0, 4);
                if (!rMatch.Success || textBox1.Text.Length != 4)
                    throw new Exception("Invalid value for COM Port!");

                returnValue = uint.TryParse(textBox2.Text, out tmpValue5);
                if (!(returnValue && tmpValue5 < 10000))
                    throw new Exception("Invalid value for baudrate!");

                if (comboBox1.SelectedIndex == -1)
                    throw new Exception("Please choose a patient name!");
            }
            catch (Exception exc)
            {
```

```
                MessageBox.Show(exc.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
    Error);
                return;
            }
            ownerForm.formOptions.xScaleFactor = tmpValue1;
            ownerForm.formOptions.yScaleFactor = tmpValue2;
            ownerForm.formOptions.properTension = tmpValue3;
            ownerForm.formOptions.comPort = textBox1.Text;
            ownerForm.formOptions.baudRate = textBox2.Text;
            ownerForm.formOptions.nBits = "8";
            ownerForm.formOptions.patientName = (string)(comboBox1.Items[comboBox1.
    SelectedIndex]);

            switch (comboBox2.SelectedIndex)
            {
                case 0:
                    ownerForm.formOptions.parity = "n";
                    break;
            }

            switch (comboBox3.SelectedIndex)
            {
                case 0:
                    ownerForm.formOptions.stopBit = "1";
                    break;
            }


            XmlSerializer xmlSerializer = new XmlSerializer(typeof(mForm.FormOptions));

            try
            {
                TextWriter txtWriter = new StreamWriter(Application.StartupPath + @"\" +
    "options.xml");

                xmlSerializer.Serialize(txtWriter, ownerForm.formOptions);

                txtWriter.Close();
                Stream fStream = new FileStream(Application.StartupPath + @"\" + "config",
    FileMode.Create, FileAccess.Write);
                BinaryWriter cWriter = new BinaryWriter(fStream);
                cWriter.Write((byte)ownerForm.formOptions.patientName.Length);
                cWriter.Close();

                TextWriter config = new StreamWriter(Application.StartupPath + @"\" +
    "config", true);

                config.Write(ownerForm.formOptions.patientName);
                config.Write(ownerForm.formOptions.baudRate + "," +
                    ownerForm.formOptions.parity + "," +
                    ownerForm.formOptions.nBits + "," +
                    ownerForm.formOptions.stopBit +
                    ownerForm.formOptions.comPort);
                config.Close();
                ownerForm.Enabled = true;
                this.Dispose();
            }
            catch (IOException exc)
            {
                MessageBox.Show(exc.Message + "\n" + "Source of error: " + exc.Source);
            }
        }

        private void oForm_Load(object sender, EventArgs e)
        {
            mForm ownerForm = (mForm)Application.OpenForms["mForm"];
            mForm.FormOptions options = ownerForm.formOptions;

            this.textBox1.Text = options.comPort;
            this.textBox2.Text = options.baudRate;
            if (options.parity == "n")
                this.comboBox2.SelectedIndex = 0;
            if (options.stopBit == "1")
```

```
            this.comboBox3.SelectedIndex = 0;

        this.textBox7.Text = options.xScaleFactor.ToString();
        this.textBox8.Text = options.yScaleFactor.ToString();
        this.textBox9.Text = options.properTension.ToString();

        List<string> patientList = mForm.GetPatientList();
        comboBox1.Items.Clear();
        foreach (string pName in patientList)
            comboBox1.Items.Add(pName);
        if(patientList.Count != 0) comboBox1.SelectedIndex = 0;

    }

    private void button2_Click(object sender, EventArgs e)
    {
        mForm ownerForm = (mForm)Application.OpenForms["mForm"];
        ownerForm.Enabled = true;
        this.Hide();
        this.Dispose();

    }

    private void button3_Click(object sender, EventArgs e)
    {
        foreach (string patient in mForm.GetPatientList())
        {
            if (patient == this.textBox5.Text)
            {
                MessageBox.Show("Record of " + patient + " already exists.\nPlease choose another name.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
        }

        PatientRecord pRecord = new PatientRecord();
        pRecord.patientName = this.textBox5.Text;
        pRecord.patientAge = (int)this.numericUpDown1.Value;
        pRecord.Other = this.textBox6.Text;
        pRecord.pictureAddress = this.textBox3.Text;

        XmlSerializer xmlSerializer = new XmlSerializer(typeof(PatientRecord));

        TextWriter tWriter = new StreamWriter(Application.StartupPath + @"\" + pRecord.patientName + ".xml");

        xmlSerializer.Serialize(tWriter, pRecord);

        comboBox1.Items.Add(this.textBox5.Text);
        comboBox1.SelectedItem = this.textBox5.Text;
        if (!this.groupBox1.Enabled) this.groupBox1.Enabled = true;
        if (!this.groupBox2.Enabled) this.groupBox2.Enabled = true;

        Stream fStream = new FileStream(Application.StartupPath + @"\" + pRecord.patientName + ".ekg", FileMode.CreateNew, FileAccess.Write);
        BinaryWriter bWriter = new BinaryWriter(fStream);

        bWriter.Write(pRecord.patientName);
        bWriter.Close();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        this.Enabled = false;
        openFileDialog1.ShowDialog();
    }

    private void openFileDialog1_FileOk(object sender, CancelEventArgs e)
    {
        textBox3.Text = openFileDialog1.FileName;
        this.Enabled = true;
    }
}
```

```
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Text;
using System.Windows.Forms;

namespace EKGP
{
    public partial class SearchForm : Form
    {
        private List<EKGDataFileProcessor.PatientData> PData;
        private bool searched = false;
        private List<int> indexList = new List<int>();

        public List<EKGDataFileProcessor.PatientData> pData
        {
            get
            {
                return PData;
            }
            set
            {
                PData = value;
                foreach (EKGDataFileProcessor.PatientData dElement in pData)
                {
                    this.listBox1.Items.Add(dElement.patientName);
                }
            }
        }
        public SearchForm()
        {
            InitializeComponent();
            mForm myOwner = (mForm)Application.OpenForms["mForm"];

            this.dateTimePicker1.Format = DateTimePickerFormat.Custom;
            this.dateTimePicker1.CustomFormat = @"dd/MM/yyyy HH:mm:ss";
            this.dateTimePicker2.Format = DateTimePickerFormat.Custom;
            this.dateTimePicker2.CustomFormat = @"dd/MM/yyyy HH:mm:ss";

        }
        private int date_intersection(DateTime start1, DateTime end1,
            DateTime start2, DateTime end2, out TimeSpan tSpan)
        {
            if(start1 >= start2 && end1 <= end2)
            {
                tSpan = (TimeSpan)(end1 - start1);
                return 0;
            }
            if(start1 >= start2)
            {
                tSpan = (TimeSpan)(end2 - start1);
                return 1;
            }
            if(end1 <= end2)
            {
                tSpan = (TimeSpan)(end1 - start2);
                return 2;
            }
            if(end1 > end2)
            {
                tSpan = (TimeSpan)(end2 - start2);
                return 3;
            }
            tSpan = TimeSpan.Zero;
            return -1;
        }



        private void sOKButton_Click(object sender, EventArgs e)
```

```
        {
            mForm myOwner = (mForm)Application.OpenForms["mForm"];
            DateTime StartDate = dateTimePicker1.Value;
            DateTime EndDate = dateTimePicker2.Value;
            sOKButton.Enabled = false;
            int i = 0;
            int m = 0;
            if (searched)
            {
                TimeSpan tSpan;
                int rValue;
                List<Point[]> pList = new List<Point[]>();
                Point[] cpList;
                List<DateTime> dsList = new List<DateTime>();
                List<DateTime> deList = new List<DateTime>();
                i = 0;
                foreach (int index in listBox1.SelectedIndices)
                {
                    EKGDataFileProcessor.PatientData.EKGData pBlock =
                        EKGDataFileProcessor.GetDataBlock(myOwner.pData[comboBox1.
    SelectedIndex], indexList[index]);
                    rValue = date_intersection(pBlock.dataStartDate,  pBlock.dataEndDate,
                        StartDate, EndDate, out tSpan);

                    switch(rValue)
                    {
                        case 0:
                            dsList.Add(pBlock.dataStartDate);
                            deList.Add(pBlock.dataStartDate);
                            cpList = new Point[pBlock.ekgData.Length];
                            m = 0;
                            foreach(byte pByte in pBlock.ekgData)
                            {
                                cpList[m].X = i;
                                cpList[m].Y = pByte;
                                i++;
                                m++;
                            }
                            pList.Add(cpList);
                            break;
                        case 1:
                            dsList.Add(pBlock.dataStartDate);
                            deList.Add(EndDate);
                            cpList = new Point[(tSpan.Seconds) * 100];
                            for(int k = 0; k < cpList.Length; k++)
                            {
                                cpList[k].X = i;
                                cpList[k].Y = pBlock.ekgData[k];
                                i++;
                            }
                            pList.Add(cpList);
                            break;
                        case 2:
                            dsList.Add(StartDate);
                            deList.Add( pBlock.dataEndDate);
                            cpList = new Point[(tSpan.Seconds) * 100];
                            m = 0;
                            for (int k = pBlock.ekgData.Length - tSpan.Seconds * 100; k <
    pBlock.ekgData.Length; k++)
                            {
                                cpList[m].X = i;
                                cpList[m].Y = pBlock.ekgData[k];
                                i++;
                                m++;
                            }
                            pList.Add(cpList);
                            break;
                        case 3:
                            dsList.Add(StartDate);
                            deList.Add(EndDate);
                            cpList = new Point[tSpan.Seconds * 100];
                            m = 0;
                            for (int k = ((TimeSpan)(StartDate - pBlock.dataStartDate)).
```

```
Seconds * 100;
                                k < ((TimeSpan)(EndDate - pBlock.dataStartDate)).Seconds *
100; k++)
                            {
                                cpList[m].X = i;
                                cpList[m].Y = pBlock.ekgData[k];
                                i++;
                                m++;
                            }
                            pList.Add(cpList);
                            break;
                        }
                        i += 40;
                    }
                    float xScaleFactor = myOwner.formOptions.xScaleFactor;
                    float yScaleFactor = myOwner.formOptions.yScaleFactor;

                    int bitmapWidth = (int)(xScaleFactor * (i + 40));
                    int bitmapHeight = (int)(yScaleFactor * (256 + 40));

                    Bitmap exportedImage = new Bitmap(bitmapWidth, bitmapHeight, System.Drawing.
Imaging.PixelFormat.Format24bppRgb);
                    Graphics handleToBitmap = Graphics.FromImage(exportedImage);
                    handleToBitmap.FillRectangle(Brushes.Black, new Rectangle(0, 0, bitmapWidth,
 bitmapHeight));

                    foreach(Point[] epList in pList)
                    {
                        Matrix tMatrix = new Matrix();
                        tMatrix.Scale(xScaleFactor, yScaleFactor);
                        tMatrix.Translate(20, 20);
                        tMatrix.TransformPoints(epList);

                        handleToBitmap.DrawCurve(Pens.ForestGreen, epList, myOwner.formOptions.
properTension);
                    }

                    handleToBitmap.Save();
                    exportedImage.Save(Application.StartupPath + @"\" + myOwner.pData[comboBox1.
SelectedIndex].patientName/* + "data on " + pData.ekgDataList[exportedNode.Index].
dataStartDate */+ ".bmp");
                    searched = false;
                    pList = null;
                    listBox1.Items.Clear();
                    Hide();
                    sOKButton.Enabled = true;
                    return;
                }

                if(myOwner.pData == null)
                {
                    MessageBox.Show("You must load a valid patient data file first!", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    Hide();
                    return;
                }
                if (((TimeSpan)(EndDate - StartDate)).Seconds <= 3)
                    MessageBox.Show("Time span of search interval must be greater than 3 seconds
.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                List<DateTime> dateList = new List<DateTime>();

                 i = 0;
                foreach (EKGDataFileProcessor.PatientData.EKGData ekgData in myOwner.pData
[comboBox1.SelectedIndex].ekgDataList)
                {
                    if (!(ekgData.dataStartDate > EndDate || ekgData.dataEndDate < StartDate))
                    {
                        listBox1.Items.Add(StartDate + " to " + EndDate);
                        indexList.Add(i);
                        i++;
                    }

                }
```

```
            searched = true;
            this.sOKButton.Text = "Export";
            sOKButton.Enabled = true;

            return;
        }


        private void cancelButton_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            Hide();
        }
    }
}
```