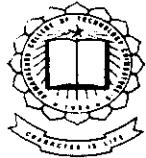


# EMBEDDED CONTROL RF APPLICATIONS



Estd-1984

2003 - 2004

## PROJECT REPORT

P-1151

### Submitted by

C.N.MANIARASU -2KEEE19  
S.RANJIT KUMAR -2KEEE36  
JAISON JEEVAN -2KEEE62  
R.VIVEK -2KEEE60

### Guided by

MRS. N.KALAIARASI  
(Sr.Lecturer)

*In partial fulfillment of the requirements for the  
award of the degree of bachelor of engineering in electrical and  
electronics engineering branch of bharathiar university, coimbatore*



Estd-1984

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

**KUMARAGURU COLLEGE OF TECHNOLOGY**

COIMBATORE - 641 006



ISO 9001:2000  
Certified

2003-2004



**KUMARAGURU COLLEGE OF TECHNOLOGY**  
COIMBATORE - 641 006  
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING



ISO 9001:2000  
Certified

Estd-1984

## **CERTIFICATE**

*This is to certify that the project report entitled*

### **EMBEDDED CONTROL RF APPLICATIONS**

*Is the bonafide work done by*

**C.N.MANIARASU-2KEEE19**

**S.RANJITKUMAR-2KEEE36**

**JAISON JEEVAN-2KEEE62**

**R.VIVEK -2KEEE60**

*In partial fulfillment of the requirements for the Award of the degree of  
Bachelor of Engineering In Electrical and Electronics Engineering  
Branch of Bharathiar university, Coimbatore*

**Guide**

**Date:** 10.3.04

**Head of the Department**

Certified that the candidate \_\_\_\_\_ with  
University Register No. \_\_\_\_\_ was examined  
in Project work Viva-Voce Examination on \_\_\_\_\_

**Internal Examiner**

**External Examiner**

**SALZER ELECTRONICS LTD.**

Samichettipalayam, Coimbatore - 641 047, India.

**salzer**

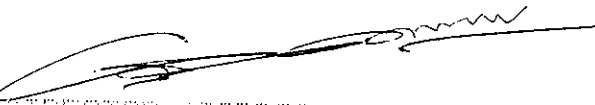
March 6, 2004

CERTIFICATE

This is to certify that **Mr.C.N.MANIARASU (2KEEE19)**, Final year **B.E(EEE)** student of **KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**, has successfully completed his Project work entitled "**EMBEDDED CONTROL RF APPLICATIONS**", as a part of his course; in our company from **NOVEMBER,2003 to MARCH,2004**.

He has evinced keen interest in absorbing the nature, concept and functions of our Organisation and his conduct and behaviour were **good** during the period.

**TL. SALZER ELECTRONICS LIMITED,**

  
**DIRECTOR (CORPORATE AFFAIRS) &  
COMPANY SECRETARY.**



Phone : 2692531 Fax : + + 91 422 2692170 Grams : SALZER  
E-mail : salzersales@satyam.net.in  
E-mail : salzer@vsnl.com

# SALZER ELECTRONICS LTD.

Samichettipalayam, Coimbatore - 641 047, India.

# salzer

March 6, 2004

## CERTIFICATE

This is to certify that **Mr. S. RANJIT KUMAR (2KEEE36)**, Final year **B.E(EEE)** student of **KUNARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**; has successfully completed his Project work entitled "**EMBEDDED CONTROL RF APPLICATIONS**", as a part of his course, in our company from **NOVEMBER, 2003 to MARCH, 2004**.

He has evinced keen interest in absorbing the nature, concept and functions of our Organisation and his conduct and character were **good** during the period.

FOR SALZER ELECTRONICS LIMITED,



DIRECTOR (CORPORATE AFFAIRS) &  
COMPANY SECRETARY.



Phone : 2692531 Fax : + + 91 422 2692170 Grams : SALZER  
E-mail : salzersales@satyam.net.in  
E-mail : salzer@vsnl.com

**SALZER ELECTRONICS LTD.**

Samichettipalayam, Coimbatore - 641 047, India.

**salzer**

March 6, 2004

CERTIFICATE

This is to certify that **Mr. JAISON JEEVAN (2KEEE62)**, Final year **B.E.(EEE)** student of **KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**, has successfully completed his Project work entitled "**EMBEDDED CONTROL RF APPLICATIONS**", as a part of his course, in our company from **NOVEMBER, 2003 to MARCH, 2004**.

He has evinced keen interest in absorbing the nature, concept and functions of our Organisation and his conduct and character were **good** during the period.

For SALZER ELECTRONICS LIMITED,



DIRECTOR (CORPORATE AFFAIRS) &  
COMPANY SECRETARY.



Phone : 2692531 Fax : + + 91 422 2692170 Grams : SALZER  
E-mail : salzersales@satyam.net.in  
E-mail : salzer@vsnl.com

**SALZER ELECTRONICS LTD.**

Samichettipalayam, Coimbatore - 641 047, India.

**salzer**

March 6, 2004

CERTIFICATE

This is to certify that **Mr. R. VIVEK (2KEEE60)**, Final year **B.E.(EEE)** student of **KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**, has successfully completed his Project work entitled "**EMBEDDED CONTROL RF APPLICATIONS**", as a part of his course, in our company from **NOVEMBER, 2003 to MARCH, 2004**.

He has evinced keen interest in absorbing the nature, concept and functions of our Organisation and his conduct and character were **good** during the period.

For **SALZER ELECTRONICS LIMITED**,



**DIRECTOR (CORPORATE AFFAIRS) & COMPANY SECRETARY.**



Phone : 2692531 Fax : + + 91 422 2692170 Grams : SALZER  
E-mail : salzersales@satyam.net.in  
E-mail : salzer@vsnl.com

## **ACKNOWLEDGEMENT**

We are greatly indebted to our revered Principal **Dr.K.K.Padmanabhan B.Sc.(Engg),M.Tech,Ph.D,MUSTEM,FTE**,who has been the motivating force behind all our deeds.

We earnestly express our Head of the Department **Dr.T.M.Kameswaran B.E,M.Sc,(Engg),Ph.D,MISTE,Sr.M.I.E.E.E,FIE**,for the encouragement and guidance he has given us to do this project.

With deep sense of gratitude we thank our project guide Sr.Lecturer **Mrs.N.Kalaiarasi, M.E,MISTE,AMIE,MSSI**, whose valuable guidance,constant encouragement and suggestions in carrying out this project.

We thank **Mr.RameshBabu, Asst.Manager (R&D), SALZER ELECTRONICS**, for providing us with technical assistance and guiding us through this project.

We also thank our staff members and friends for helping us with relevant details and especially their constant encouragement.

## **SYNOPSIS**

In the fast moving world, the usage of wireless communication applications, is one of the indeed part of the communication system. So this type of communication only done in a effective manner with the help of RF technology, which is used in our project. Embedded system is one of the emerging fields in the area of communication system. So we incorporate the embedded system in this RF technology in order to increase the stability and reliability of the system.

The main objective of our project is to do wireless applications in Cordless speaker, Message Transfer and ON-OFF control.

The Cordless speaker is designed as a portable type. The effective range of the speaker is 100 meters, apart from this, the system is provided with security system of simple code lock.

The wireless message transfer system is designed to transfer the alphabets, digits and some special symbols. The incorporation of embedded system makes the system to transfer the message at a higher speed.

The ON-OFF control of equipment is designed such that it can be controlled from any remote places. This type of system will be highly useful in places where to make efficient remote control.



# CONTENTS

	Page No
<b>1.INTRODUCTION</b>	
1.1 Introduction To RF	2
1.2 Introduction To Embedded System	3
1.3 Peripheral Interface Controller	
1.3.1 PIC 16F84A Block Diagram	4
1.3.2 Memory Organization	5
1.3.3 I/O Ports	10
1.3.4 Timers	13
<b>2.MODULATION</b>	
2.1 Modulation Techniques	26
2.2 Amplitude Modulation	27
2.3 Frequency Modulation	27
<b>3.RECEPTION AND DEMODULATION</b>	
3.1 Super Heterodyne Reception	29
3.2 Demodulation	30
<b>4.HARDWARE</b>	
4.1 FM Transmitter	32
4.2 Digital Code Lock	33
4.3 FM Receiver	35
4.4 Transmitter	36
4.5 Receiver	37
<b>5.SOFTWARE</b>	
5.1 Flow Chart	39
5.2 Coding	41
<b>6.CONCLUSION</b>	45

APPENDIX

BIBILOGRAPHY

## **INTRODUCTION TO RF**

In our day to day life , wireless communication plays a vital role. RF is a radio frequency signal. Radio frequency spectrum is 100 Mhz to 30 Ghz. In this propagation is used as LOS (Line Of Sight).The wave length of Rf is the range of 10-1 meters.

In Radio frequency communication the signal transfer rate is very high. The noise in the RF signal is very less, also it has no interference with any other signal. The design of high frequency circuits is little bit complicated process. The main application fields used in the RF are short distance communication, radar, aeronautical navigation system and also in GPS. For the transmission of RF signal it uses RF transmitter and receiver.

### **TRANSMITTER**

It is a electronic device that transmit information by means of radio waves. The signal intelligence is translated in terms of high frequency waves. The process of intelligence translation into high frequency is termed as modulation. All transmitting system must there for , have a section for generation of high frequency carrier wave, a section for converting information into electrical pulses and amplifying them to the required level , a section for modulating the carrier with signal intelligence, amplification stage for increasing the level of the modulated wave to the desired power and antenna system for transmitting these signals into the free space.

### **RECEIVER**

It is a electronic device that pick up the desired signal from the numerous signal propagating at that time through the atmosphere, amplifies the desired signal to the required level,recovers from it the original modulating signal and eventually display it in the desired manner. Before the signal is fed to loud speaker the signal is power amplified.

## INTRODUCTION TO EMBEDDED SYSTEM

An Embedded Product uses a microchip (or) micro controller to do a task. In an Embedded system there is only one application software that is typically burned into ROM. In this Project PIC-16f84A Microcontroller, operates with 8 bit data.

The embedded system comprises variety of hardware and software components, which perform specific function in the host system. It is designed around a microcontroller unit, which integrates on-chip program memory, on-chip oscillator, interrupt, port.

PIC executes most of its instruction in  $0.2 \mu\text{s}$ , at its maximum clock rate of 20MHz. It consists of 35 instruction sets, therefore it is easy for programming. It runs efficiently at one instruction per clock cycle, at high oscillator frequency of 20MHz efficiently. Versatile timer provides input, control output and also internal timing for program execution. Timer resets the PIC when there is a malfunction in the circuit. PIC provides up to 12 independent interrupt sources and also can control when the CPU deals with each source. The peripherals make use of standard 16 pin shift register to add any number of I/O pins.

# BLOCK DIAGRAM OF PIC-16F84A

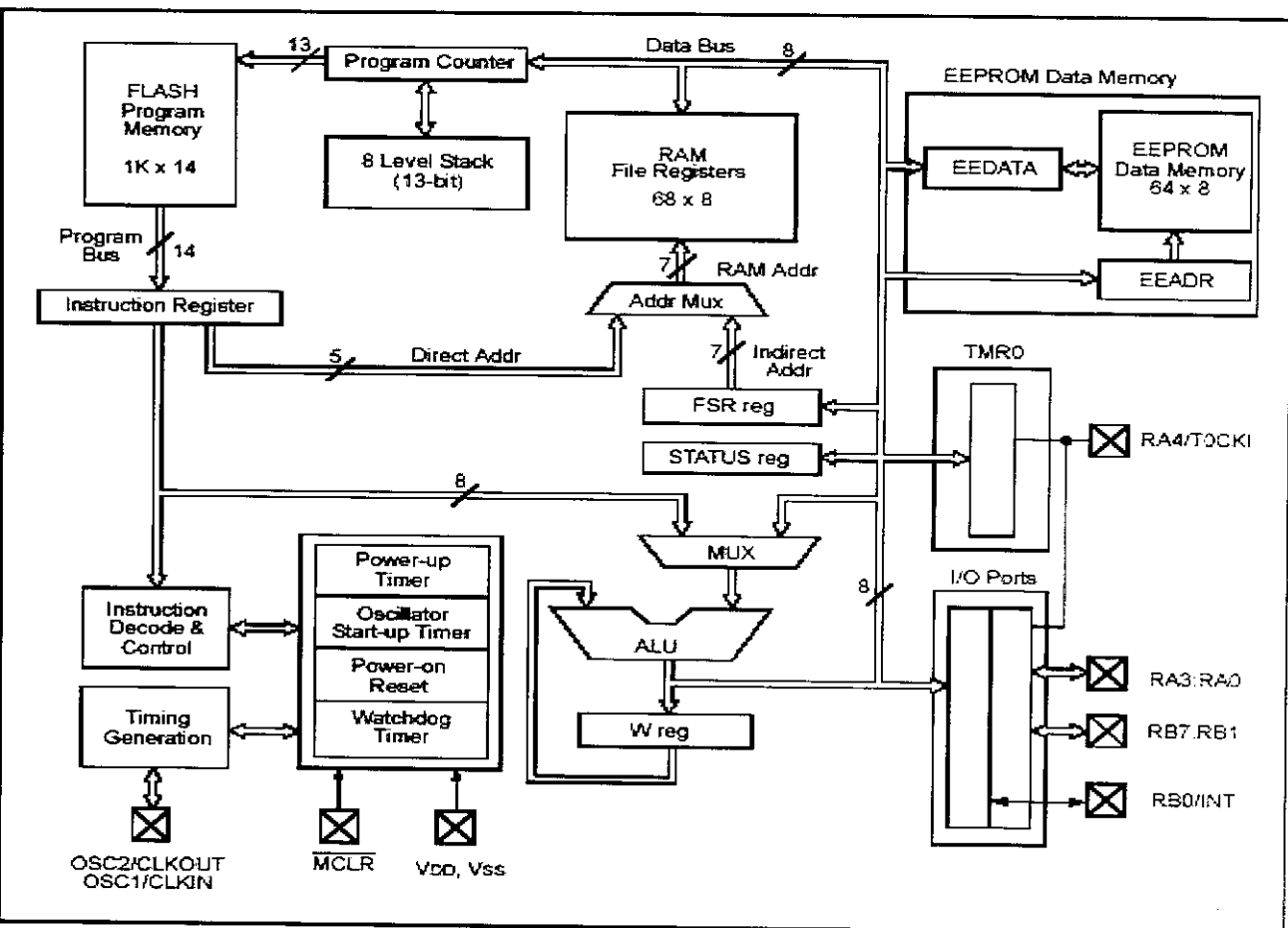


Fig.1

The general block diagram of PIC-16F84A is shown in fig1. PIC –16F84A belongs to the Mid family of the PIC micro microcontroller devices. This works with 8-bit data. The program code is written through hi-tech C language and stored in flash memory. Memory organization of PIC consists of two blocks namely Data memory and program memory. Program memory contains 1K words, which translates to 1024 instructions. The data memory (RAM) contains 68 Bytes. PIC uses special function registers, which are stored in static RAM. Each register is of 8-bit wide. The programming in PIC is easy, since it only uses 35 instruction sets.

## MEMORY ORGANIZATION

The Memory organization of PIC-16f84A consists of two memory's.

They are program memory and data memory. The data memory consists of special function registers and general purpose RAM. The data memory also consists of Data EEPROM memory, which has 64 bytes of address.

The address range is from 0H-3FH.

## PROGRAM MEMORY ORGANIZATION

The PIC16FXCX has a 13- bit program counter capable of addressing an 8K x 14 program memory space. The PIC 16F84A, the first 1K x 14 (000h – 3FFh) are physically implemented reset vector is at 0000h Location.

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A

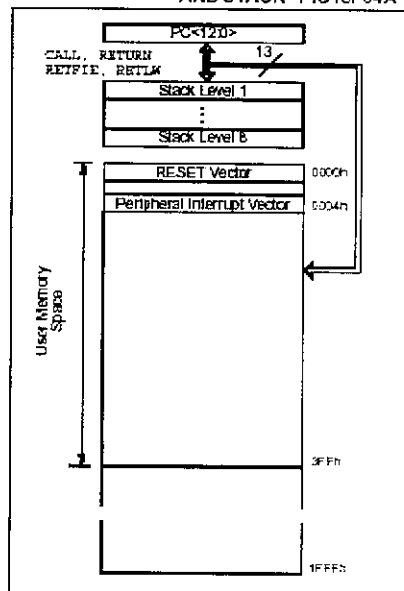


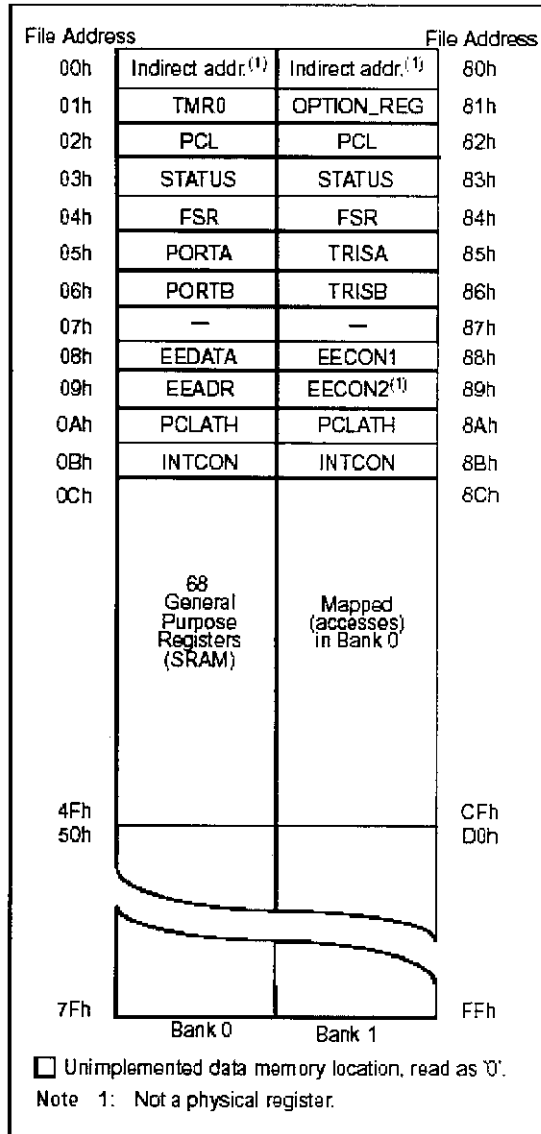
Fig.2.1

## **DATA MEMORY ORGANIZATION**

The data memory is partitioned into two areas. The first is the Special Function Register (SFR) area, while the second the General Purpose Registers (GPR) area. The SFR control the operation of the device. The GPR area is blanked to allow greater than 116 bytes of general purpose RAM.

The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register. Instructions MOVWF and MOVF can move values from the W register to any location in the register file ("F") and vice – versa. The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (SFSR). Indirect addressing uses the present value of the RP0 bit for access into the blanked which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS <5>). Setting the RP0 bit selects Bank 1. Each Bank extends upto 7Fh (128 bytes). The first twelve locations of each Bank are reserved for the Special Function Registers. The remaining are General Purpose Registers implemented as static RAM.

**FIGURE 2-2: REGISTER FILE MAP - PIC16F84A**



## REGISTERS

### SPECIAL FUNCTION REGISTERS

The Special Function Registers are used by the CPU and Peripheral to control the device operation. These registers are static RAM. The special function registers can be classified into two sets, core and peripheral.

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

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	TO	PD	Z	DC	C
					bit 0		
bit 7							

bit 7-6 Unimplemented: Maintain as '0'

bit 5 RP0: Register Bank Select bits (used for direct addressing)

01 = Bank 1 (80h - FFh)

00 = Bank 0 (00h - 7Fh)

bit 4  $\overline{TO}$ : Time-out bit

1 = After power-up,  $\overline{CLRWDT}$  instruction, or  $\overline{SLEEP}$  instruction

0 = A WDT time-out occurred

bit 3  $\overline{PD}$ : Power-down bit

1 = After power-up or by the  $\overline{CLRWDT}$  instruction

0 = By execution of the  $\overline{SLEEP}$  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/borrow bit ( $\overline{ADDWF}$ ,  $\overline{ADDLW}$ ,  $\overline{SUBLW}$ ,  $\overline{SUBWF}$  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/borrow bit ( $\overline{ADDWF}$ ,  $\overline{ADDLW}$ ,  $\overline{SUBLW}$ ,  $\overline{SUBWF}$  instructions) (for  $\overline{borrow}$ , the polarity is reversed)

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: A subtraction is executed by adding the two's complement of the second operand. For rotate ( $\overline{RRF}$ ,  $\overline{RLF}$ ) 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



The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory. As with any register, The STATUS register is the destination for an instruction that affects the Z, DC or C bits, and then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the TO and 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 leave the STATUS register as 000u u1uu (where u = unchanged). Only the BCF, BSF, SWAPF and MOVWF instructions should be used to alter the STATUS register because these instructions do not affect any status bit.

## **PCL AND PCLATH**

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 13 bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<12:8> bits and is not directly readable or writable. All updates to the PCH register go through the PCLATH register.

## **STACK**

The stack allows a combination of up to 8 program calls and interrupts to occur. The stack contains the return address from this branch in program execution. Midrange devices have 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

CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, TETLW or a RETFIE instruction execution. PCLATCH is not modified when the stack is PUSHed or POPed. 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).

## **INPUT AND OUTPUT PORTS**

General purpose i/o pins can be considered the simplest of peripherals. They allow PIC micro controller to monitor and control other devices. Some pins are multiplexed with an alternate functions. These functions Depend on which peripheral features are on the device, thus multiplexing adds flexibility to the system.

### **PORT A AND TRISA REGISTERS**

PORT A is a 5-bit wide bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (=1) will make the corresponding PORT A pin an output, i.e., put the contents of the output driver in a hi-impedance mode. Clearing a TRISA bit (=0) will make the corresponding PORT A pin an output, i.e., put the contents of the output latch on the selected pin. Reading the PORT A register reads the status of the pins whereas writing to it will write to the port latch. All write operations are ready-modify-write operations. Therefore a write to a port implies that the port pins are read, this 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/T0CK1 pin. The RA4/T0CK1 pin

is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

## **INITIALIZING PORT A**

```
BCF STATUS, RP0;
```

```
CLRF PORT ; Initialize PORT A by
```

```
; clearing output
```

```
; data latches
```

```
BSF STATUS, RP0 ; Select Bank 1
```

```
MOVLW 0 x 0 F ; Value used to
```

```
; initialize data
```

```
; direction
```

```
MOVWF TRISA ; Set RA<3:0> as inputs
```

```
; RA4 as output
```

```
; TRISA <7:5> are always read as '0'.
```

## **PORT B AND TRISB REGISTERS**

PORT B is an 8-bit wide bi-directional 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 hi-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.

## INITIALIZING PORT B

```
BCF STATUS, RP0;
CLRF PORTB; Initialize PORT B by
; Clearing output
; Data latches
BSF STATUS, RP0; Select Bank 1
MOVLW 0 x CF; Value used to
; Initialize data
; Direction
MOVWF TRISB; Set RB <3:0> as inputs
; RB<5:4> as outputs
; RB <7:6> as inputs
```

Each of the PORT B pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit RBPU (OPTION <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.

Four of PORTB's pins, RB7:RB4, have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB) 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.

## **TIMER 0**

The Timer 0 module timer / counter has the following features:

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

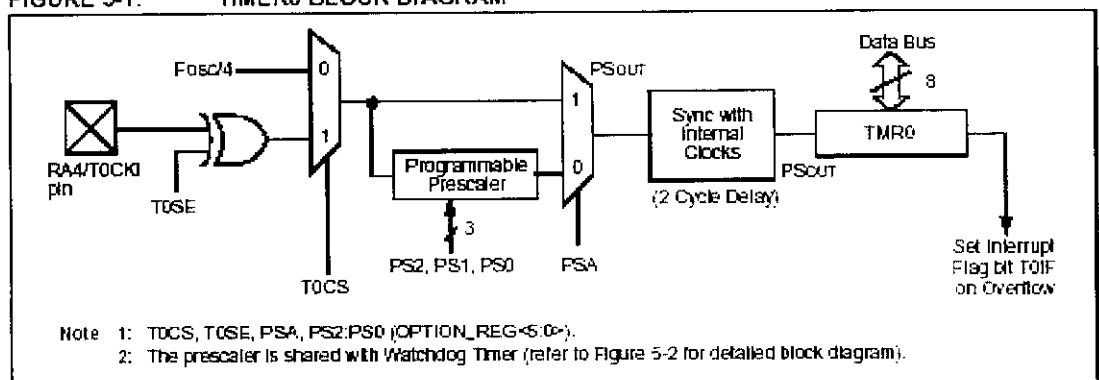
## **TIMER 0 OPERATION**

Timer 0 can operate as a timer or as a counter. 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 (OPTIONREG<5>).

If The Timer 0 is operated in counter mode, in case of this counter mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CK1. The incrementing edge is determined by the Timer 0 Source Edge Select bit T0SE (OPTION\_REG<4>). Clearing bit T0SE selects the rising edge. Restrictions on the external clock input are discussed below. When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer 0 after synchronization.

FIGURE 5-1: TIMER0 BLOCK DIAGRAM



## INSTRUCTION SET SUMMARY

### TYPES OF INSTRUCTION

Each PIC16CXXX 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.

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 number of bit affected by the operation, while 'f' represents the number 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. The opcode field descriptions and the general format for instructions is shown on appendix A.

## **I/O PROGRAMMING CONSIDERATIONS**

### **BI-DIRECTIONAL I/O PORTS**

Any instruction which writes, operates internally as a read followed by a write operation. The BCF and BSF instructions, for example, read the register into the CPU, execute the bit operation and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit 5 of PORT B will cause all eight bits of PORT B to be read into the CPU. Then the BSF operation takes place on bit 5 and PORTB is written to the output latches. If another bit of PORT B is used as a bi-directional I/O pin (i.e., bit0) and it is defined as an input at this time, the input signal present on the pin itself particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit 0 is switched into output mode later on, the content of the data latch is unknown. Reading the port register, reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read-modify-write instructions (i.e., BCF, BSF, etc.,) on a port, the value of the port pins is read, the desired operation is done to this value, and this value is then written to the port latch. A pin actively outputting a Low or High should not be driven from external devices at the same time in order to change the level on this pin ("wired-or", "wired-and"). The resulting high output current may damage the chip.



## **SUCCESSIVE OPERATIONS ON I/O PORTS**

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 5-5). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such that the pin voltage stabilizes (load dependent) before the next instruction which causes that file to be read into the CPU is executed. Otherwise, the previous stats of that pin may be read into the CPU rather than the new state. When in doubt, it is better to separate these instructions with a NOP or another instruction not accessing this I/O port. Example 5-1 shows the effect of two sequential read-modify-write instructions (e.g., BCF, BSF, etc.) on an I/O port.

## **SPECIAL FEATURES OF THE CPU**

What sets a micro controller apart from other processors are special circuits to deal with the needs of real time applications. The PIC16F8X has a host of such features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operation mode and offer code protection.

These features are:

- OSCILLATOR selection.
- Reset – Power-on Reset (POR)
- Interrupts.
- Watchdog Timer (WDT)
- SLEEP ,CODE PROTECTION etc..

The PIC16F8X has 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. This design keeps the device in reset while the power supply stabilizes. With these two timers on-chip, most applications need no external reset circuitry. SLEEP through external reset, Watchdog Timer time-out or through an interrupt. Several oscillator options are provided 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 are used to select the various options.

## **CONFIGURATION BITS**

The configuration bits can be programmed (read as '0') or let unprogrammed (read as '1') to select various device configuration. These bits are mapped in program memory location 2007h. Address 2007h is beyond the user program memory space and it belongs to the special test / configuration memory space (2000h – 3FFFh). This space can only be accessed during programming. To find out how to program the PIC 16C84, refer to PIC16C84 EEPROM Memory Programming Specification (DS30189).

## OSCILLATOR CONFIGURATIONS

### OSCILLATOR TYPES

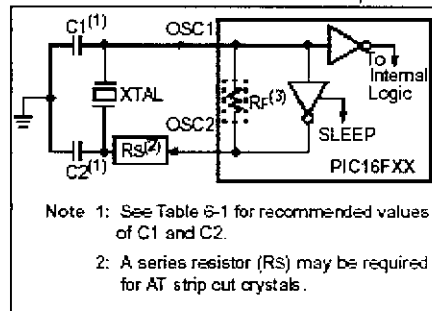
The PIC16F84A can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1 and FOSC0) to select one of these four modes:

- LP Low Power Crystal
- XT Crystal / Resonator
- HS High Speed Crystal / Resonator
- RC Resistor / Capacitor

#### (i) CRYSTAL OSCILLATOR / CERAMIC RESONATORS

In XT, LP or HS modes a crystal or ceramic resonator is connected to the OSC1 / CLKIN and OSC2 / CLKOUT pins to establish oscillation.

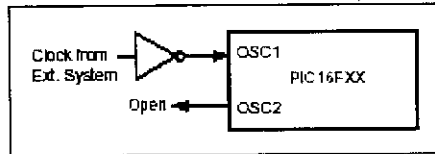
FIGURE 6-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP OSC CONFIGURATION)



The PIC16F84A oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal

manufacturers specifications. When in XT, LP or HR modes, the device can have an external clock source to drive the OSC1/CLKIN pin.

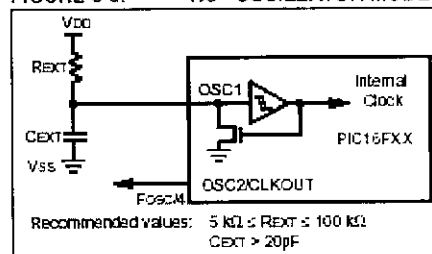
FIGURE 6-2: EXTERNAL CLOCK INPUT OPERATION (HS, XT OR LP OSC CONFIGURATION)



### (ii) 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 ( $R_{ext}$ ) values, capacitor ( $C_{ext}$ ) 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 also affects the oscillation frequency, especially for low  $C_{ext}$  values. The user needs to take into account variation due to tolerance of the external R and C components.

FIGURE 6-3: RC OSCILLATOR MODE



## **RESET**

The PIC 16F84A differentiates between various kinds of reset, Generally Reset logic is used to place the device into a known state.the source of the reset can be determined by using the device status bits.The reset logic is designed with features that reduce system cost and increase system reliability.

The various types of resets are as follows:

- Power-on Reset (POR)
- MCLR reset during normal operation
- MCLR reset during SLEEP
- WDT Reset (during normal operation)
- WDT Wake-up (during SLEEP)

### **POWER ON RESET (POR)**

A POWER – on Reset pulse is generated on-chip when VDD rise is detected in the range of 1.2 V – 1.7 V). To take advantage of the POR, just tie the MCLR pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create Power-on Reset. A minimum rise time for VDD must be met for this to operate properly. See Electrical Specifications for details. When the device starts normal operation (exists the reset condition), device operating parametes (voltage, frequency, temperature,..) must be meet to ensure operation. If these conditions are not met, the device must be held in reset until the operating conditions are met. For additional information, refer to

Application Note AN 607, “Power-up Trouble Shootin”. The POR circuit does not produce an internal reset when VDD declines.

### **POWER-UP TIMER (PWRT)**

The Power-up Timer (PWRT)A provides a fixed 72 ms nominal timeout (TPWRT) from POR. The power-up Timer operates on an internal RC oscillator. The chip is kept in rest as long as the PWRT is active. The PWRT delay allows the VDD to rise to an acceptable level. A configuration bit, PWRTE, can enable / disable the PWRT. The power-up time delay TPWRT will vary from chip to chip due to VDD, temperature, and process variation.

### **OSCILLATOR START-UP TIMER (OST)**

The oscillator Start-up Timer (OST) provides a 1024 oscillator cycle delay (from OSC1 input) after the PWRT delay ends. This ensures the crystal oscillator or resonator has started and stabilized. The OST time-out (TOST) is invoked only for XT, LP and HS modes and only on Power-on Reset or wake-up from SLEEP. When VDD rises very slowly, it is possible that the TPWRT time-out and TOST time-out will expire before VDD has reached its final value. In this case, an external power-on reset circuit may be necessary.

### **DATA EEPROM MEMORY**

The EEPROM data memory is readable and writable during normal operation (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 four SFRs used to read and write this memory. These Registers are:

- EECON1
- EECON2 (Not a physically implemented register)
- EEDATA
- EEADR

EEDATA holds the 8-bit data for read/write, and EEADR holds the address of the EEPROM location being accessed. PIC 16F84A devices have 64 bytes of data. EPROM with an address range from 0h to 3Fh. The EEPROM data memory allows byte read and write. A byte write automatically erases the location and writes the new data (erase before write). The EEPROM data memory is rated for high erase / write cycles. The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature as well as from chip to chip. Please refer to AC specifications for exact limits. When the device is code protected, the CPU may continue to read and write the data EEPROM memory. The device programmer can no longer access this memory. EECON register is shown on appendix B.

## **READING THE EEPROM DATA MEMORY**

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

## WRITING TO THE EEPROM DATA 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 sequence to initiate the write for each byte.

The write will not initiate if the above 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. 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 pull this bit. EEIF must be cleared by software.



## **MODULATION TECHNIQUES**

Before going into the various details of modulation techniques, let us acquainted with the concept of modulation. Modulation is the process where in some parameters of high frequency signal, termed as carrier is varied in accordance with the signal to be transmitted. The signal to be transmitted is known as the modulating signal. Various modulation methods have been developed for transmission of signals with minimum possible distortion.

The various modulation techniques that are available for signal transmission are given below:

1. Amplitude modulation
2. Frequency modulation
3. Phase modulation
4. Pulse modulation

First three methods come under analog modulation where as in digital modulation a pulse train is used as a carrier.

### **ANALOG MODULATION**

Analog Modulation may be divided into three parts:

1. Amplitude modulation
2. Phase modulation
3. Frequency Modulation

## **AMPLITUDE MODULATION**

The process of Amplitude Modulation consists of varying the peak amplitude of the modulating signal in proportion to the instantaneous amplitude of the modulating signal. The signal to noise ratio (S/N) is comparatively less in Amplitude Modulation. This means that the data transmitted using amplitude modulation is more prone to noise. Also the power required for AM is more when compared with FM. So we should select a modulation scheme which consumes less power and is less prone to noise.

## **PHASE MODULATION**

Phase shift keying (PSK) is also called Phase modulation. The signal differs by phase shift instead of frequency or amplitude. Typically, a signal's phase shift is measured relative to the previous signal. In such case, the term differential phase shift keying (DPSK) is often used. Here  $n$  bits can be assigned a signal having one of the  $2^n$  phase shifts, giving a technique in which the bit rate is  $n$  times the baud rate.

## **FREQUENCY MODULATION**

In frequency modulation, the frequency of the carrier signal is varied in accordance with the instantaneous amplitude of the modulating signal, without any variations in the amplitude of the carrier wave.

## **INTRODUCTION TO RECEPTION**

The process of recovering the original modulating signal from a modulated wave is known as demodulation or detection. The type of demodulation technique employed depends on the technique used. This project uses FM transmission for transmission of data.

To receive the FM signal the most popularly used technique is super heterodyne reception. The forthcoming section deals with principle of super reception.

## **SUPER HETERODYNE RECEPTION**

It is the process of operation on modulated radio waves to obtain similarly modulated waves of a different frequency. In general this process uses the locally generated wave, which determines the change of frequency. The word super stands for SUPER-sonic HET-aerodyne which means that the heterodyning takes place with the resulting output frequency greater than audio. Thus the principle of reception involves one or more changes of frequency before the modulating signal is extracted from the modulated wave. The above principle is used in the reception of FM waves.

Super heterodyne reception method has the following advantage over other methods:

1. Improved selectivity.
2. Improved receiver stability.
3. Higher gain per stage because the intermediate frequency (IF) amplifier are operated at a lower frequency.
4. Uniform bandwidth because of fixed intermediate frequency.

## DEMODULATION

The process of extracting modulating signal from a frequency modulated carrier is known as frequency modulation or detection. The electronic circuits that perform the demodulation process are called FM detectors. The FM detectors perform the detection in two steps:

- 1.It converts the frequency modulated signal into its corresponding amplitude signal by using frequency dependant circuits ie, circuits whose output voltages depends on input frequency from which original modulating signal is detected. Such circuits are called frequency discriminators.

- 2.The original modulating signal is recovered from this AM signal (converted from FM to AM in previous step)by using a linear diode envelope detector. This threshold can be improved by pre-emphasis and de-emphasis circuits.

## FM TRANSMITTER

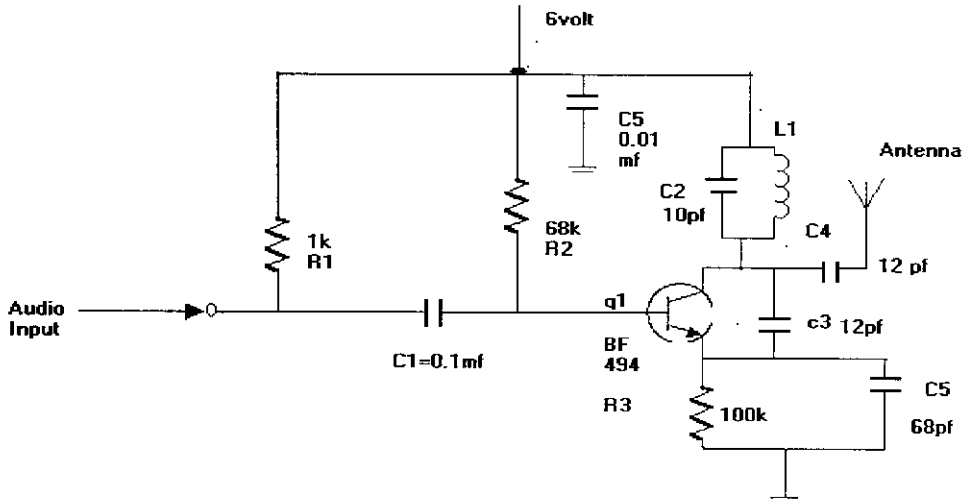


Fig 4.1

A simple but practical FM transmitter circuit is given in the Fig 4.1. This circuit can be operated 9v supply. The signals are passed from the encoder circuit to the base of the transistor Q1 through the capacitor c1.Q1 is a NPN transistor and its collector is given positive supply through the resistance R3 while its base is given the forward supply through the resistance R2 ,the voltage signals are given to base of the transistor Q1 through the capacitor c1.

A capacitor c3 has also been connected between the collector and the emitter of transistor q1.This capacitor triggers the oscillations . As soon as the Audio signal is received at the base then the transistor starts to oscillate and it generates FM frequency, which is given to the antenna through the capacitor .

## SIMPLE CODE LOCK SYSTEM

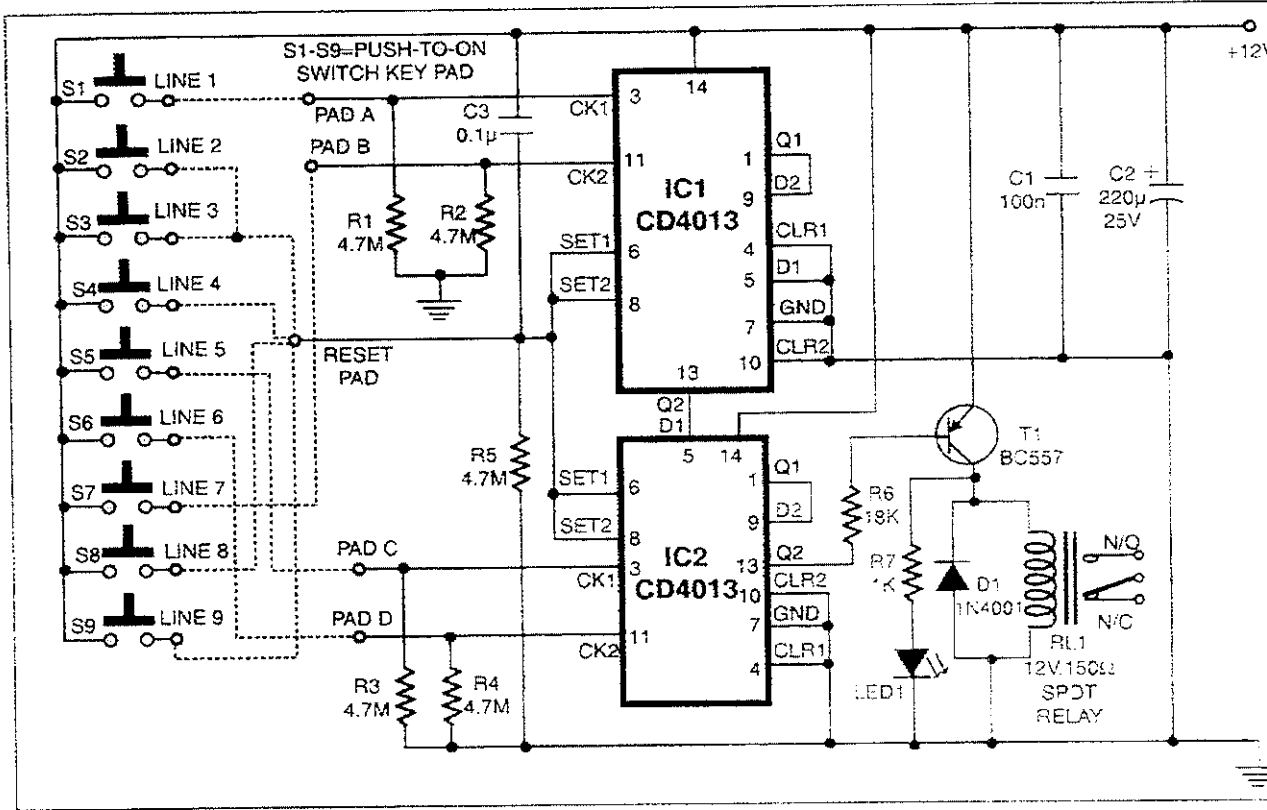


Fig 4.2

This is an electronic combination of lock, which responds only to the right sequence of four digits that are keyed in remotely. If a wrong key is pressed, it resets the lock. The lock code can be set by connecting the line wires to the pads A,B,C,D in the fig 4.2. The circuit built around two CD4013 dual d-flip flop Ics. The clock pins of the four flipflops are connected to A,B,C, and D pads. The correct code sequence for energisation of relay RL1 is realized by clocking points A,B,C, and D in that order. The five remaining switches are connected to reset pad which resets all the flipflops.

## Working Of the circuit

The circuit working is based upon two CD4013 dual D flipflop Ics. Each CD 4013 IC has a pair of D-FF with independent data, set, reset and clock inputs and Q and Q bar outputs. Here, four digit code is generated using these four D-FF available in Ics. In this it successively shift the low level pulse from first to last FF when right keys are pressed in correct sequence and activate the relay as final output if code entered is correct. For this, the first D-FF of IC1 has its data input D1 grounded and output pin Q of each FF connected to input pin of next FF. Also, as the logic levels present at D input of each FF is transferred to the Q output only during positive-going transition of the clock pulse, the correct sequence pads A, B, C, and D are connected to clock pins of four FFs.

The four clock pins are normally held low through resistors R1 to R4 separately. But when switches connected to respective pads are pressed, a positive going clock pulse is generated to shift low level data input of one FF to next FF. Output of last FF drives PNP transistor T1 BC557 to activate relay RL1 after correct code is entered. Diode d1 avoids chattering of relay while LED 1 provides proper indication. Resistance R6 and R7 act as current limiting resistors. Dc power supply is filtered using capacitors C1 and C2 and to avoid indeterminate state during switching ON operation power on reset capacitor c3 is connected at reset pad. Clear input of each FF (pin R) is grounded to avoid any false triggering. As set pin is also active high input, all four set inputs are normally held low through resistor R7 and also connected to a common reset pad. For making task of guessing the correct code difficult any number of switches can be combined to this reset pad. Pressing any wrong switch resets all FFs eliminating any easy way to crack the code.

# FM RECIEVER

C

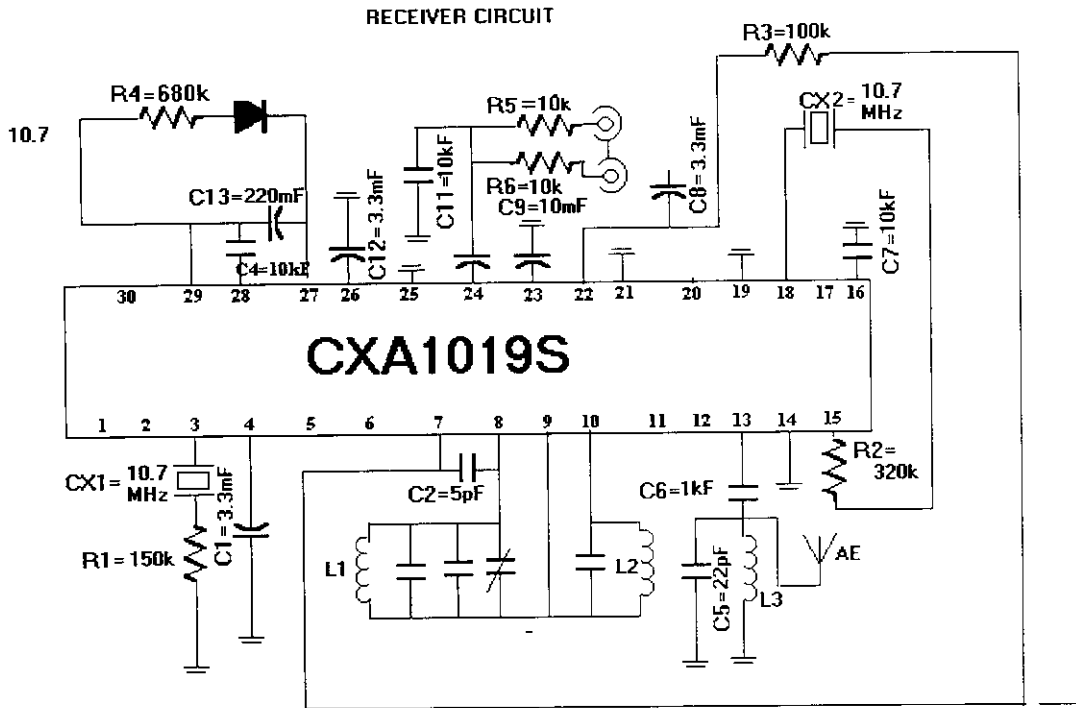


Fig 4.3

IC CXA1019s is a thirty pin DIL IC. Facility to connect a tuning indicator has been provided in this IC apart from the various sections built within the IC. A 10.7 Mhz ceramic filter has been used in this circuit in place of the IFTS. An audio output section is also built within the IC apart from all the necessary sections. The audio output section of this IC has not been used in the give circuit. The signal received at the IC pin number 24 is given to the pin 1 of the volume control and the pin2 of the volume controllers connected to the audio output sections. This section is not shown in the given circuit.

The IC pin no:27 is the positive supply pin which is given 9 volt supply through a resistance R4. This supply is filtered by the capacitor c1. IC pin no:7 and \* are connected to the oscillator section and a coil L1 is connected at the pin no:8. The capacitor c3 and a button trimmer B1 are connected parallel to this coil L1. The desired frequency is selected by this trimmer RF coil L2 is connected at the IC pin no 10 and a capacitor c7 and a button trimmer B2 are connected in parallel to this coil.



# TRANSMITTER

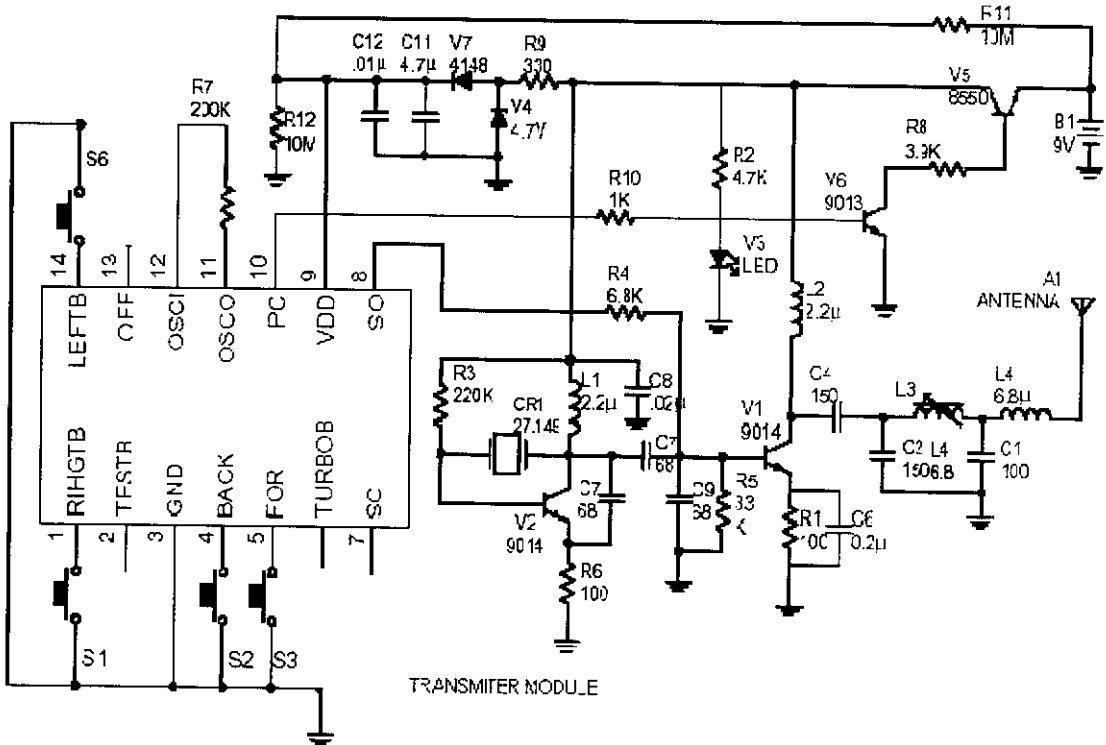


Fig 4.4

This transmitter is used for both ON-OFF control and message transfer. The transmitter requires 9v supply. In the transmitter part, each signal is encoded by encoder IC TX2. The signal is converted by Intermediate frequency by modulator. Derived RF signal is obtained by mixing the signal with mixer.

The filter circuit provides some frequency to pass while rejecting the much lower frequency. Then the signal is fed to antenna. In ON -Off control, it controls four applications. Here four switches are provided in the transmitter part. When a key is pressed a corresponding frequency is generated and signal is encoded amplified and transmitted via through antenna.

In message transfer, these keys one is to shift the cursor position, other two keys carry alphabets. In this transfer each alphabet has some time sequence LCD display is provided to know what message is typed.

# RECEIVER

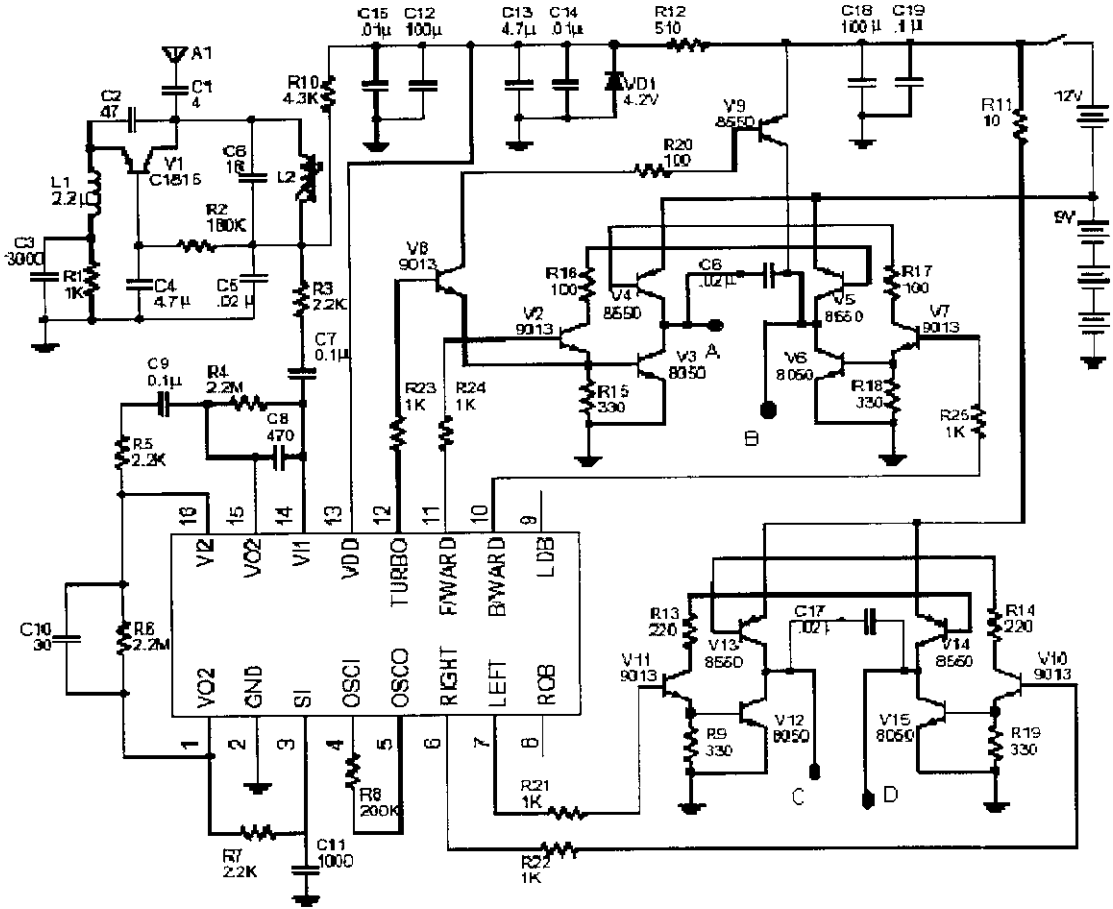


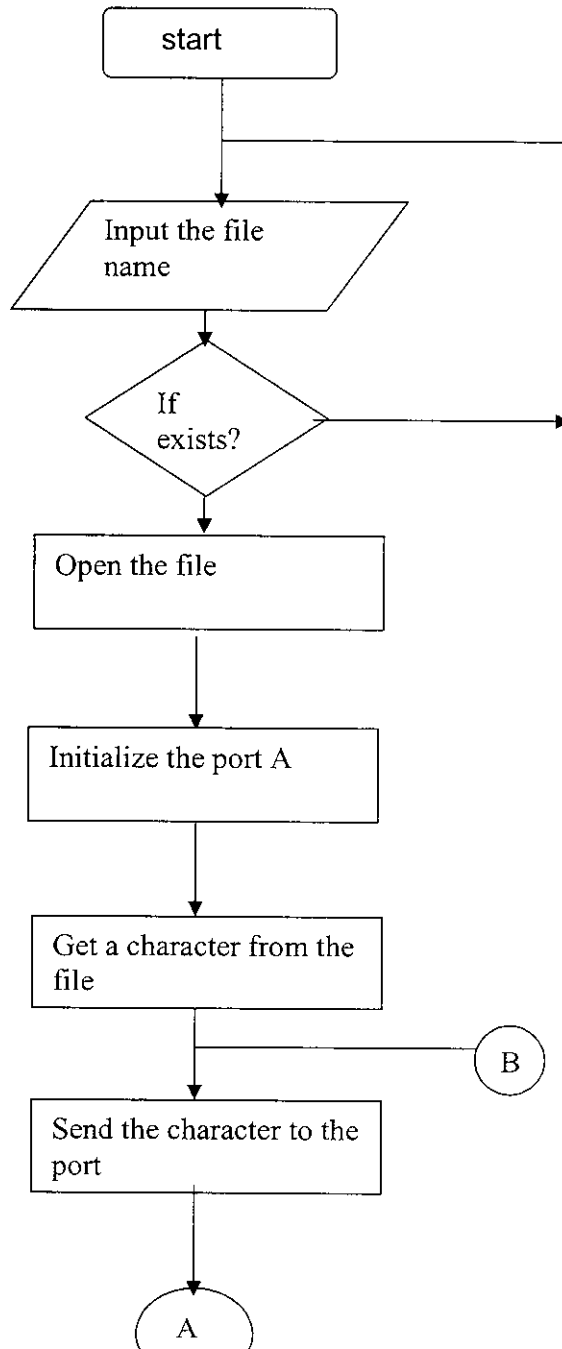
Fig 4.5

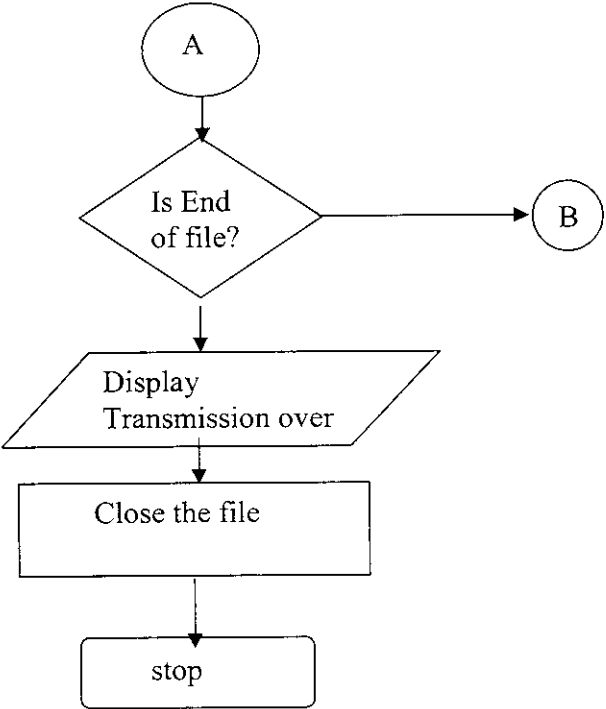
This receiver is common for both on-off and message transfer. Antenna receives only the corresponding bandwidth. The receiver requires 12V supply. The RFC choke avoids the dc biased signals. Filter circuit provides the selected signals by filtering out received signal and undesired frequency passing the signal with the desired frequency band. Decoder RX2 is provided to detect the the signal.

In on-off part four relays are provided, when a switch is pressed in transmitter, the corresponding relay is energized and its application made on or off.

In message part, the display is through LCD. Each alphabet carrier time sequenced which is detected by decoder and corresponding alphabet is displayed. The receiver part is connected with microcontroller unit.

# FLOW CHART





## CODING FOR ON-OFF CONTROL SYSTEM

RA0-RA4-----INPUTS

RB4-RB7-----OUTPUTS

```
*/void delay2(char);
#include "pic.h"
#define RLY1 RB4
#define RLY2 RB5
#define RLY3 RB6
#define RLY4 RB7
#define IN1 RA0
#define IN2 RA1
#define IN3 RA2
#define IN4 RA3
void main()
{
TRISA=0X0F;
TRISB=0X00;
while(1)
{
    if(RA0==1)
    {
        if(RLY1==1)
            RLY1=0;
        else
            RLY1=1;
        delay2(1);
    }
    if(RA1==1)
    {
        if(RLY2==1)
```

```
                RLY2=0;
            else
                RLY2=1;
            delay2(1);
        }
        if(RA2==1)
        {
            if(RLY3==1)
                RLY3=0;
            else
                RLY3=1;
            delay2(1);
        }
        if(RA3==1)
        {
            if(RLY4==1)
                RLY4=0;
            else
                RLY4=1;
            delay2(1);
        }
    }
} //END OF WHILE
} //END OF MAIN
```

```
void delay2(char rec)
{
    int d1,d2,d3;
    for(d3=0;d3<rec;d3++)
        for(d1=0;d1<255;d1++)
            for(d2=0;d2<325;d2++);
}
```

## CODING FOR MESSAGE TRANSFER SYSTEM

```
#include "pic.h"
#include "lcd1.c"
void delay2(char);
const char text[3][17]={"Loaded$","Distance:$","Amount:$"};
const char
k1look[]={',','0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I','J','K','L','M','N'};
const char
k2look[]={',','O','P','Q','R','S','T','U','V','W','X','Y','Z','!','!','!','!','!','!','@','#','$','%','^','&', '*', '-', '+'};
char ii,add;
char k1,k2,k3;
void main()
{
ADCON1=0X07;
TRISA=0x0f;
TRISB=0X00;
TRISC=0X00;
init();
add=0x80;
while(1)
{
if(RA2==1)
{
delay2(1);
add++;
if(add>0x8f)
add=0xc0;
if(add>0xcf)
add=0x80;
k1=k2=k3=0;
}
}
while(RA2);
```

```

}
if(RA0==1)
{
    delay2(1);
    k1++;
    if(k1>24)
        k1=1;
    address(add);
    disp(k1look[k1]);
    while(RA0);
}
if(RA3==1)
{
    delay2(1);
    k2++;
    if(k2>24)
        k2=1;
    address(add);
    disp(k2look[k2]);
while(RA3);

}
}
}
void delay2(char rec)
{
int e1,e2,e3;
for(e3=0;e3<rec;e3++)
for(e1=0;e1<155;e1++)
for(e2=0;e2<325;e2++);
}

```



## **CONCLUSION**

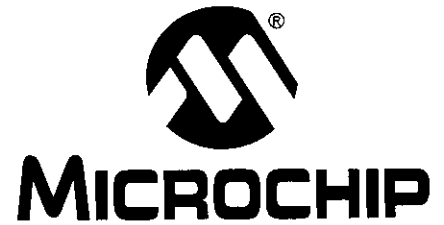
The remote operation using wireless communication to ON/OFF, message transfer and cordless speaker has been designed and implemented successfully. The complete system is more reliable than the existing manual methods in the industrial process. The integration of hardware and software provide efficient operation of the system. Moreover the developed system is quite simple and user friendly and economical. The system has been working satisfactorily with the designed values and the system will fulfill the industrial requirement.

## **FUTURE ENHANCEMENT**

“Change is the only changeless phenomenon in the world over”

We held this saying right from the inception of this project. New technologies emerge and subside as time goes on, but our project stands against winds of time. Our project can be enhanced in the following ways.

- ❖ In the cordless speaker, transmission range can be extended up to a range of 3Km.
- ❖ In Message transfer, it is possible to fix the modified type of Keypad.
- ❖ In On-Off control, it is possible to control the equipment from a remote place by extending the range of transmission.



**PIC16F84A**  
**Data Sheet**

18-pin Enhanced FLASH/EEPROM  
8-bit Microcontroller

---

**Note the following details of the code protection feature on PICmicro® MCUs.**

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products 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 PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be 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 product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

---

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, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, KEELoq, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

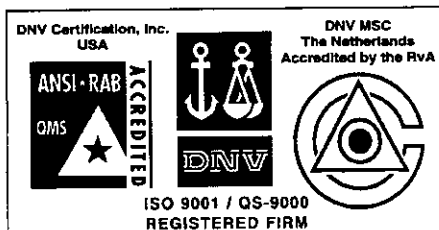
Total Endurance, ICSP, In-Circuit Serial Programming, Filter-Lab, MXDEV, microID, FlexROM, fuzzyLAB, MPASM, MPLINK, MPLIB, PICC, PICDEM, PICDEM.net, ICEPIC, Migratable Memory, FanSense, ECONOMONITOR, Select Mode and microPort are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term 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.

© 2001, 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. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELoq® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.*



# PIC16F84A

## 18-pin Enhanced FLASH/EEPROM 8-Bit Microcontroller

### High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input  
DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

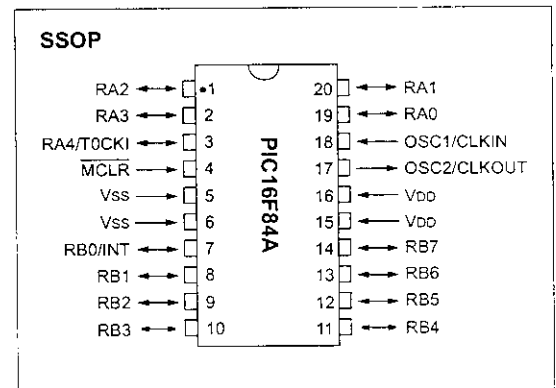
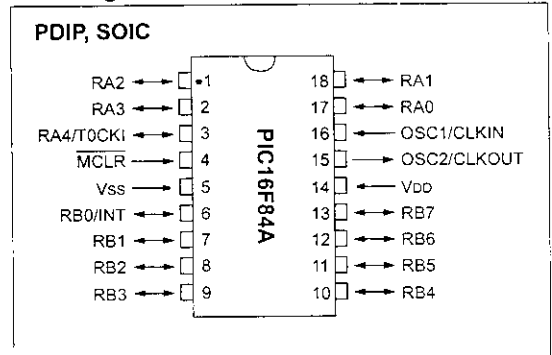
### Peripheral Features:

- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

### Special Microcontroller Features:

- 10,000 erase/write cycles Enhanced FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options

### Pin Diagrams



### CMOS Enhanced FLASH/EEPROM Technology:

- Low power, high speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

# PIC16F84A

## Table of Contents

1.0 Device Overview .....	3
2.0 Memory Organization .....	5
3.0 Data EEPROM Memory .....	13
4.0 I/O Ports .....	15
5.0 Timer0 Module .....	19
6.0 Special Features of the CPU .....	21
7.0 Instruction Set Summary .....	35
8.0 Development Support .....	43
9.0 Electrical Characteristics .....	49
10.0 DC/AC Characteristic Graphs .....	61
11.0 Packaging Information .....	71
Appendix A: Revision History .....	75
Appendix B: Conversion Considerations .....	76
Appendix C: Migration from Baseline to Mid-Range Devices .....	78
Index .....	79
On-Line Support .....	83
Reader Response .....	84
PIC16F84A Product Identification System .....	85

## 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](mailto: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](http://www.microchip.com/cn) to receive the most current information on all of our products.

# PIC16F84A

## 1.0 DEVICE OVERVIEW

This document contains device specific information for the operation of the PIC16F84A device. Additional information may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023), which may be downloaded from the Microchip website. 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.

The PIC16F84A belongs to the mid-range family of the PICmicro® microcontroller devices. A block diagram of the device is shown in Figure 1-1.

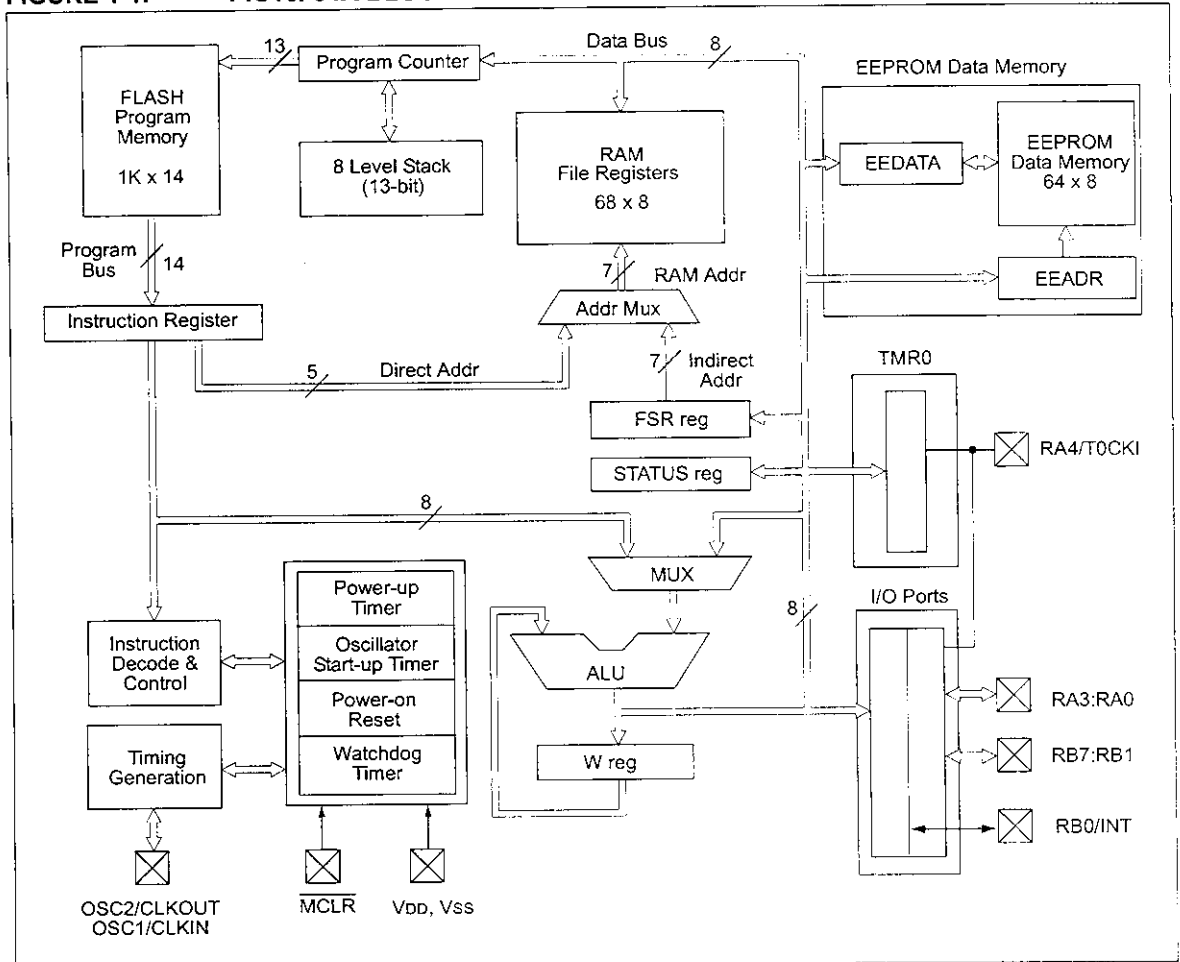
The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.

There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupt
- Timer0 clock input

Table 1-1 details the pinout of the device with descriptions and details for each pin.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM



# PIC16F84A

TABLE 1-1: PIC16F84A PINOUT DESCRIPTION

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS <sup>(3)</sup>	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
MCLR	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
RA0	17	17	19	I/O	TTL	<p>PORTA is a bi-directional I/O port.</p> <p>Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.</p>
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CK1	3	3	3	I/O	ST	
RB0/INT	6	6	7	I/O	TTL/ST <sup>(1)</sup>	<p>PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.</p> <p>RB0/INT can also be selected as an external interrupt pin.</p> <p>Interrupt-on-change pin.</p> <p>Interrupt-on-change pin.</p> <p>Interrupt-on-change pin.</p> <p>Serial programming clock.</p> <p>Interrupt-on-change pin.</p> <p>Serial programming data.</p>
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	
RB5	11	11	12	I/O	TTL	
RB6	12	12	13	I/O	TTL/ST <sup>(2)</sup>	
RB7	13	13	14	I/O	TTL/ST <sup>(2)</sup>	
Vss	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	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.  
**Note 2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.  
**Note 3:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

# PIC16F84A

## 2.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh. More details on the EEPROM memory can be found in Section 3.0.

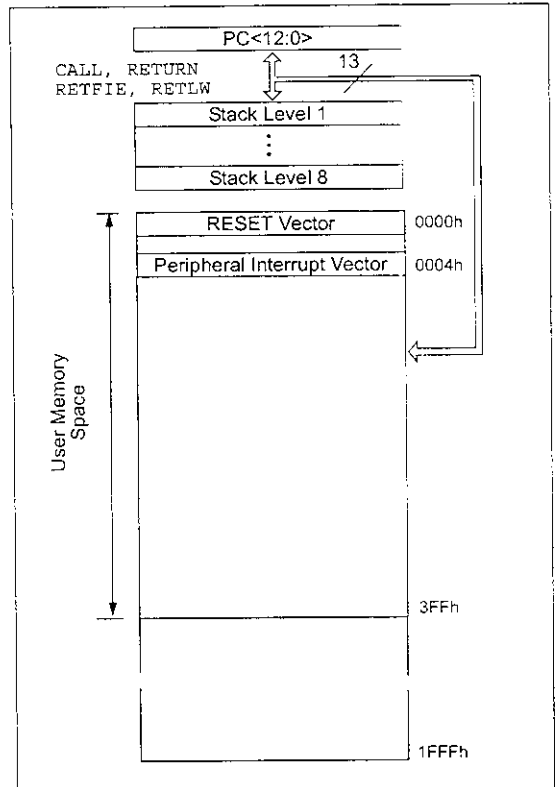
Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

### 2.1 Program Memory Organization

The PIC16FXX has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16F84A, the first 1K x 14 (0000h-03FFh) are physically implemented (Figure 2-1). Accessing a location above the physically implemented address will cause a wraparound. For example, for locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, and 1C20h, the instruction will be the same.

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

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A





# PIC16F84A

## 2.2 Data Memory Organization

The data memory is partitioned into two areas. The first is the Special Function Registers (SFR) area, while the second is the General Purpose Registers (GPR) area. The SFRs control the operation of the device.

Portions of data memory are banked. This is for both the SFR area and the GPR area. The GPR area is banked to allow greater than 116 bytes of general purpose RAM. The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register. Figure 2-2 shows the data memory map organization.

Instructions MOVWF and MOVF can move values from the W register to any location in the register file ("F"), and vice-versa.

The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (FSR) (Section 2.5). Indirect addressing uses the present value of the RP0 bit for access into the banked areas of data memory.

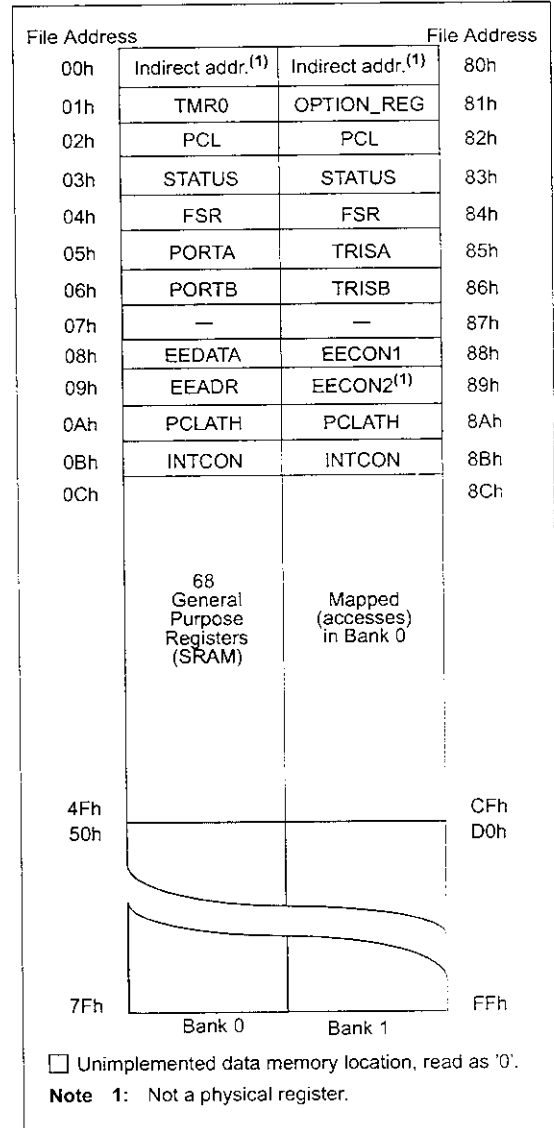
Data memory is partitioned into two banks which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS<5>). Setting the RP0 bit selects Bank 1. Each Bank extends up to 7Fh (128 bytes). The first twelve locations of each Bank are reserved for the Special Function Registers. The remainder are General Purpose Registers, implemented as static RAM.

### 2.2.1 GENERAL PURPOSE REGISTER FILE

Each General Purpose Register (GPR) is 8-bits wide and is accessed either directly or indirectly through the FSR (Section 2.5).

The GPR addresses in Bank 1 are mapped to addresses in Bank 0. As an example, addressing location 0Ch or 8Ch will access the same GPR.

FIGURE 2-2: REGISTER FILE MAP - PIC16F84A



# PIC16F84A

## 2.3 Special Function Registers

The Special Function Registers (Figure 2-2 and Table 2-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

**TABLE 2-1: SPECIAL FUNCTION REGISTER FILE SUMMARY**

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
<b>Bank 0</b>											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- --	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS <sup>(2)</sup>	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA <sup>(4)</sup>	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	--x xxxx	16
06h	PORTB <sup>(5)</sup>	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC <sup>(1)</sup>			---	0000	11	
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	10
<b>Bank 1</b>											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- --	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS <sup>(2)</sup>	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register			---	1111	16	
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 xxx0	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- --	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC <sup>(1)</sup>			---	0000	11	
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

**Note 1:** The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.

**2:** The  $\overline{TO}$  and  $\overline{PD}$  status bits in the STATUS register are not affected by a MCLR Reset.

**3:** Other (non power-up) RESETS include: external RESET through MCLR and the Watchdog Timer Reset.

**4:** On any device RESET, these pins are configured as inputs.

**5:** This is the value that will be in the port output latch.

# PIC16F84A

## 2.3.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. 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 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).

Only the `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions should be used to alter the STATUS register (Table 7-2), because these instructions do not affect any status bit.

**Note 1:** The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F84A and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

**2:** The C and DC bits operate as a borrow and digit borrow out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

**3:** When 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. The specified bit(s) will be updated according to device logic

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

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 7-6 **Unimplemented:** Maintain as '0'

bit 5 **RP0:** Register Bank Select bits (used for direct addressing)

01 = Bank 1 (80h - FFh)

00 = Bank 0 (00h - 7Fh)

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/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for 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/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for borrow, the polarity is reversed)

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:** 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

## 7.0 INSTRUCTION SET SUMMARY

Each PIC16CXX 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 PIC16CXX instruction set summary in Table 7-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 7-1 shows the opcode field descriptions.

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 number of 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.

**TABLE 7-1: 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

The instruction set is highly orthogonal and is grouped into three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

All instructions are executed within one single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles with the second cycle executed as a NOP. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1  $\mu$ s. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2  $\mu$ s.

Table 7-2 lists the instructions recognized by the MPASM™ Assembler.

Figure 7-1 shows the general formats that the instructions can have.

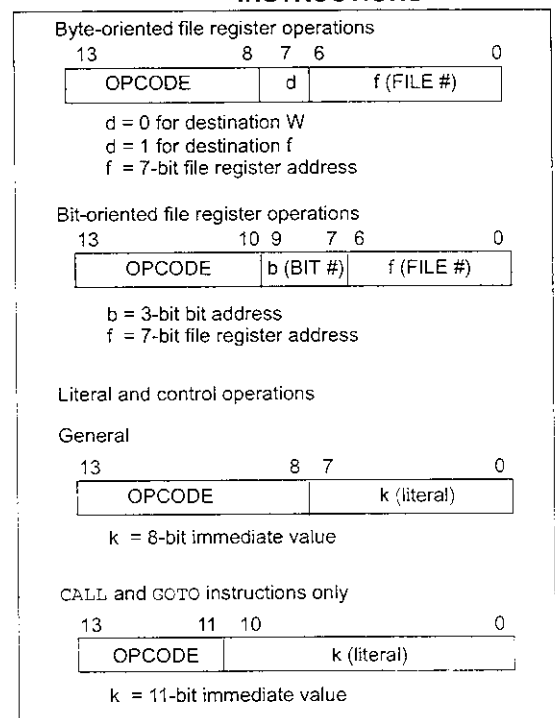
**Note:** To maintain upward compatibility with future PIC16CXX products, **do not use the OPTION and TRIS instructions.**

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

**FIGURE 7-1: GENERAL FORMAT FOR INSTRUCTIONS**



A description of each instruction is available in the PICmicro™ Mid-Range Reference Manual (DS33023).

# PIC16F84A

TABLE 7-2: PIC16CXXX INSTRUCTION SET

Mnemonic, Operands	Description	Cycles	14-Bit Opcode		Status Affected	Notes
			MSb	LSb		
<b>BYTE-ORIENTED FILE REGISTER OPERATIONS</b>						
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	1fff 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	Z 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	Z 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	1fff 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	Z 1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110	dfff ffff	Z 1,2
<b>BIT-ORIENTED FILE REGISTER OPERATIONS</b>						
BCF	f, b	Bit Clear f	1	01 00bb	bfff ffff	Z 1,2
BSF	f, b	Bit Set f	1	01 01bb	bfff ffff	Z 1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb	bfff ffff	Z 3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb	bfff ffff	Z 3
<b>LITERAL AND CONTROL OPERATIONS</b>						
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	$\overline{TO,PD}$
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	$\overline{TO,PD}$
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'.
- Note 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.
- Note 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).

## 7.1 Instruction Descriptions

<b>ADDLW</b>	<b>Add Literal and W</b>
Syntax:	<i>[label]</i> ADDLW k
Operands:	$0 \leq k \leq 255$
Operation:	$(W) + k \rightarrow (W)$
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.

<b>BCF</b>	<b>Bit Clear f</b>
Syntax:	<i>[label]</i> BCF f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b \leq 7$
Operation:	$0 \rightarrow (f<b>)$
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

<b>ADDWF</b>	<b>Add W and f</b>
Syntax:	<i>[label]</i> ADDWF f,d
Operands:	$0 \leq f \leq 127$ $d \in \{0,1\}$
Operation:	$(W) + (f) \rightarrow (\text{destination})$
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'.

<b>BSF</b>	<b>Bit Set f</b>
Syntax:	<i>[label]</i> BSF f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b \leq 7$
Operation:	$1 \rightarrow (f<b>)$
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

<b>ANDLW</b>	<b>AND Literal with W</b>
Syntax:	<i>[label]</i> ANDLW k
Operands:	$0 \leq k \leq 255$
Operation:	$(W) .\text{AND.} (k) \rightarrow (W)$
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.

<b>BTFSS</b>	<b>Bit Test f, Skip if Set</b>
Syntax:	<i>[label]</i> BTFSS f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b < 7$
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 2TCY instruction.

<b>ANDWF</b>	<b>AND W with f</b>
Syntax:	<i>[label]</i> ANDWF f,d
Operands:	$0 \leq f \leq 127$ $d \in \{0,1\}$
Operation:	$(W) .\text{AND.} (f) \rightarrow (\text{destination})$
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'.

# PIC16F84A

## **BTFSC** Bit Test, Skip if Clear

**Syntax:** `[label] BTFSC f,b`  
**Operands:**  $0 \leq f \leq 127$   
 $0 \leq b \leq 7$   
**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 2TCY instruction.

## **CALL** Call Subroutine

**Syntax:** `[label] CALL k`  
**Operands:**  $0 \leq k \leq 2047$   
**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:**  $0 \leq f \leq 127$   
**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:**  $0 \leq f \leq 127$   
 $d \in [0,1]$   
**Operation:** ( $\hat{f}$ ) → (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:**  $0 \leq f \leq 127$   
 $d \in [0,1]$   
**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'.

# PIC16F84A

---

## DECFSZ      Decrement f, Skip if 0

**Syntax:**      [*label*] DECFSZ f,d

**Operands:**     $0 \leq f \leq 127$   
                   $d \in [0,1]$

**Operation:**     $(f) - 1 \rightarrow (\text{destination});$   
                  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 2TCY instruction.

---

## INCFSZ      Increment f, Skip if 0

**Syntax:**      [*label*] INCFSZ f,d

**Operands:**     $0 \leq f \leq 127$   
                   $d \in [0,1]$

**Operation:**     $(f) + 1 \rightarrow (\text{destination});$   
                  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 2TCY instruction.

---

## GOTO        Unconditional Branch

**Syntax:**      [*label*] GOTO k

**Operands:**     $0 \leq k \leq 2047$

**Operation:**     $k \rightarrow \text{PC}<10:0>$   
                   $\text{PCLATH}<4:3> \rightarrow \text{PC}<12:11>$

**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.

---

## IORLW      Inclusive OR Literal with W

**Syntax:**      [*label*] IORLW k

**Operands:**     $0 \leq k \leq 255$

**Operation:**     $(W) .\text{OR. } k \rightarrow (W)$

**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.

---

## INCF        Increment f

**Syntax:**      [*label*] INCF f,d

**Operands:**     $0 \leq f \leq 127$   
                   $d \in [0,1]$

**Operation:**     $(f) + 1 \rightarrow (\text{destination})$

**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'.

---

## IORWF      Inclusive OR W with f

**Syntax:**      [*label*] IORWF f,d

**Operands:**     $0 \leq f \leq 127$   
                   $d \in [0,1]$

**Operation:**     $(W) .\text{OR. } (f) \rightarrow (\text{destination})$

**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'.



# PIC16F84A

---

**MOVF**            **Move f**

---

Syntax:            [ *label* ] MOVF f,d

Operands:         $0 \leq f \leq 127$   
                     $d \in [0,1]$

Operation:        (f) → (destination)

Status Affected: Z

Description:      The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.

**MOVLW**          **Move Literal to W**

---

Syntax:            [ *label* ] MOVLW k

Operands:         $0 \leq k \leq 255$

Operation:         $k \rightarrow (W)$

Status Affected: None

Description:      The eight-bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

**MOVWF**          **Move W to f**

---

Syntax:            [ *label* ] MOVWF f

Operands:         $0 \leq f \leq 127$

Operation:        (W) → (f)

Status Affected: None

Description:      Move data from W register to register 'f'.

**NOP**             **No Operation**

---

Syntax:            [ *label* ] NOP

Operands:        None

Operation:        No operation

Status Affected: None

Description:      No operation.

**RETFIE**          **Return from Interrupt**

---

Syntax:            [ *label* ] RETFIE

Operands:        None

Operation:        TOS → PC,  
                    1 → GIE

Status Affected: None

**RETLW**          **Return with Literal in W**

---

Syntax:            [ *label* ] RETLW k

Operands:         $0 \leq k \leq 255$

Operation:         $k \rightarrow (W)$ ;  
                    TOS → PC

Status Affected: None

Description:      The W register is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

**RETURN**          **Return from Subroutine**

---

Syntax:            [ *label* ] RETURN

Operands:        None

Operation:        TOS → PC

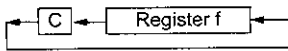
Status Affected: None

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.

# PIC16F84A

## RLF Rotate Left f through Carry

**Syntax:** [label] RLF f,d  
**Operands:**  $0 \leq f \leq 127$   
 $d \in [0,1]$   
**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'.



## SUBLW Subtract W from Literal

**Syntax:** [label] SUBLW k  
**Operands:**  $0 \leq k \leq 255$   
**Operation:**  $k - (W) \rightarrow (W)$   
**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.

## RRF Rotate Right f through Carry

**Syntax:** [label] RRF f,d  
**Operands:**  $0 \leq f \leq 127$   
 $d \in [0,1]$   
**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'.



## SUBWF Subtract W from f

**Syntax:** [label] SUBWF f,d  
**Operands:**  $0 \leq f \leq 127$   
 $d \in [0,1]$   
**Operation:**  $(f) - (W) \rightarrow (\text{destination})$   
**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'.

## SLEEP

**Syntax:** [label] SLEEP  
**Operands:** None  
**Operation:**  $00h \rightarrow \text{WDT}$ ,  
 $0 \rightarrow \text{WDT prescaler}$ ,  
 $1 \rightarrow \overline{\text{TO}}$ ,  
 $0 \rightarrow \overline{\text{PD}}$   
**Status Affected:**  $\overline{\text{TO}}$ ,  $\overline{\text{PD}}$   
**Description:** The power-down status bit,  $\overline{\text{PD}}$  is cleared. Time-out status bit,  $\overline{\text{TO}}$  is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.

## SWAPF Swap Nibbles in f

**Syntax:** [label] SWAPF f,d  
**Operands:**  $0 \leq f \leq 127$   
 $d \in [0,1]$   
**Operation:**  $(f<3:0>) \rightarrow (\text{destination}<7:4>)$ ,  
 $(f<7:4>) \rightarrow (\text{destination}<3:0>)$   
**Status Affected:** None  
**Description:** The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in W register. If 'd' is 1, the result is placed in register 'f'.

# PIC16F84A

---

<b>XORLW</b>	<b>Exclusive OR Literal with W</b>
Syntax:	<code>[label] XORLW k</code>
Operands:	$0 \leq k \leq 255$
Operation:	$(W) \text{ .XOR. } k \rightarrow (W)$
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.

<b>XORWF</b>	<b>Exclusive OR W with f</b>
Syntax:	<code>[label] XORWF f,d</code>
Operands:	$0 \leq f \leq 127$ $d \in \{0,1\}$
Operation:	$(W) \text{ .XOR. } (f) \rightarrow (\text{destination})$
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'.



**GENERAL DESCRIPTION**

The SM-T2 / SM-R2 is a pair of CMOS chip, it was designed for remote controlled car applications.

The SM-T2 / SM-R2 has five control keys for controlling the motions ( i.e. forward, backward rightward, leftward and the turbo function ) of the remote controller car.

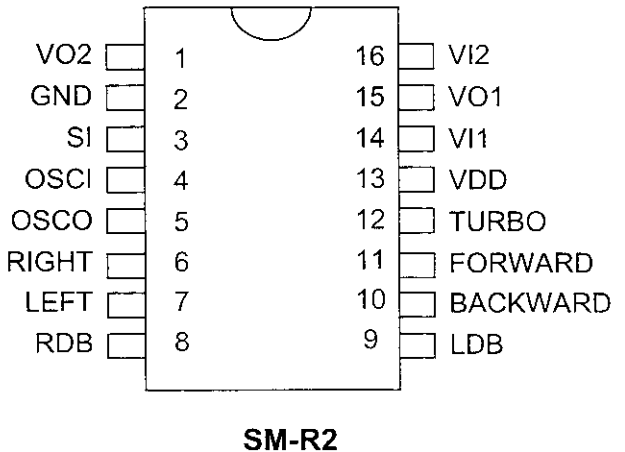
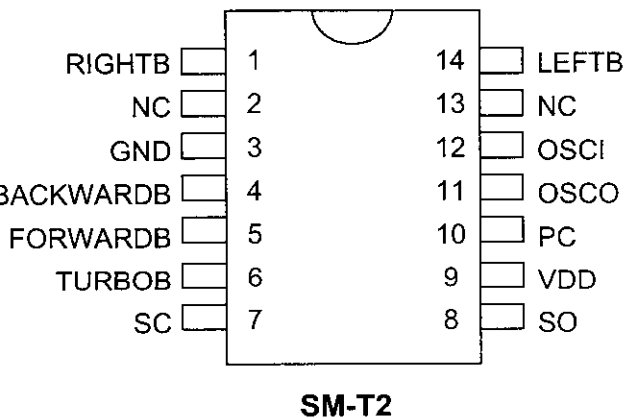
**FEATURES**

- \* Power supply : 2.4V - 5.0V
- \* Low stand-by current
- \* Five functions control
- \* Auto-power-off function for SM-T2
- \* Few external components are needed

**APPLICATION**

\* Remote control car etc.

**PIN ASSIGNMENTS (TOP VIEW)**





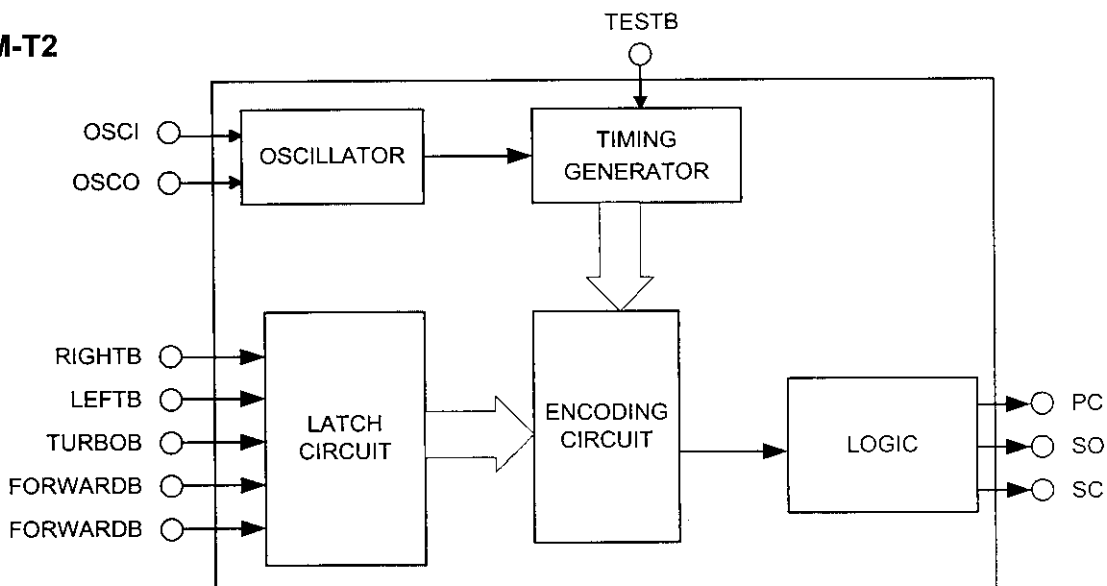
SAMHOP Microelectronics Corp.

# SM-T2/SM-R2

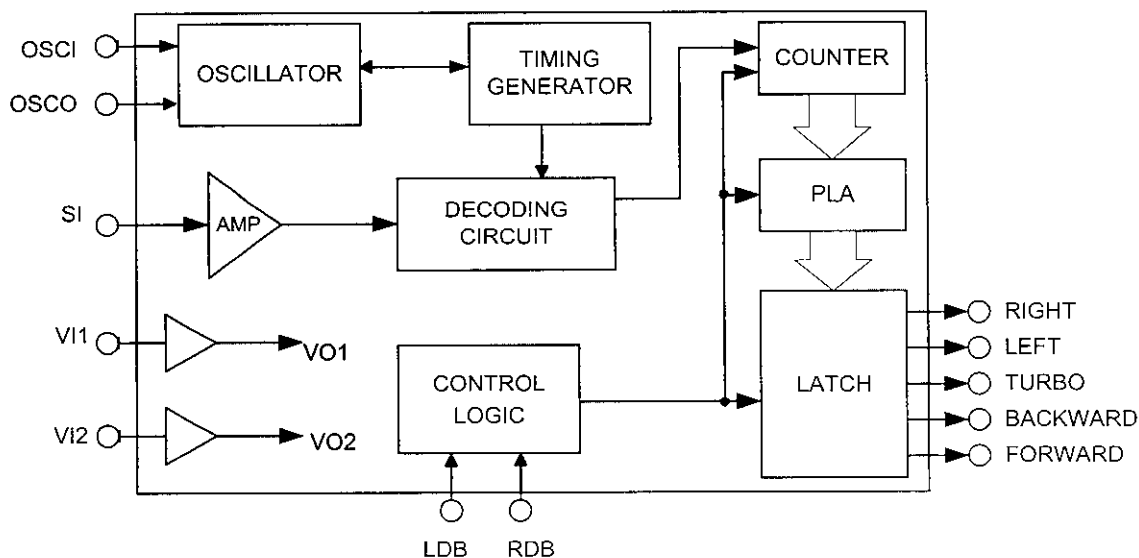
REMOTE CONTROLLER  
WITH FIVE FUNCTION

## BLOCK DIAGRAM

### SM-T2



### SM-R2





SAMHOP Microelectronics Corp.

# SM-T2/SM-R2

REMOTE CONTROLLER  
WITH FIVE FUNCTION

## PIN DESCRIPTION

### SM-T2

Pin No.	Pin Name	Description
1	RIGHTB	The rightward function will be selected, if this pin is connected to GND
2, 13	NC	
3	GND	Negative power supply
4	BACKWARDB	The backward function will be selected, if this pin is connected to GND
5	FORWARDB	The forward function will be selected, if this pin is connected to GND
6	TURBOB	The turbo function will be selected, if this pin is connected to GND
7	SC	Output pin of the encoding signal with carrier frequency
8	SO	
9	VDD	Positive power supply
10	PC	Power control output pin
11	OSCO	Oscillator output pin
12	OSCI	Oscillator input pin
14	LEFTB	The leftward function will be selected, if this pin is connected to GND

### SM-R2

Pin No.	Pin Name	Description
1	VO2	Inverter 2 output pin for power amplify
2	GND	Negative power supply
3	SI	Input pin of the encoding signal
4	OSCI	Oscillator input pin
5	OSCO	Oscillator output pin
6	RIGHT	Rightward output pin
7	LEFT	Leftward output pin
8	RDB	
9	LDB	Leftward function disable, if this pin is connected to GND
10	BACKWARD	Backward output pin
11	FORWARD	Forward output pin
12	TURBO	TURBO output pin
13	VDD	Positive power supply
14	VI1	Inverter 1 input pin for power amplify
15	VO1	Inverter 1 output pin for power amplify

## **BIBLIOGRAPHY**

- ◆ Carl J.Weisman, "Essential Guide To RF and Wireless"

LPE Publications,1997 edition.

- ◆ PeatMan, "Design With PIC Microcontroller",

LPE Publications,2000 edition

- ◆ Anok Singh," Principles of Communication Engineering" ,

S. Chand Publications 1999 edition.

- ◆ [www.Electronicsforu.com](http://www.Electronicsforu.com)

- ◆ [www.Discovercircuits.com](http://www.Discovercircuits.com)

- ◆ [www.Electronicscircuits.com](http://www.Electronicscircuits.com)