
PICDEM-1 USER'S GUIDE

Information contained in this publication regarding device applications and the like is intended by way of suggestion only. 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. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip.

The Microchip logo and name are registered trademarks of Microchip Technology Incorporated. PICmicro, PICMASTER, PICSTART, and PRO MATE are a registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. MPLAB is a trademark of Microchip Technology Inc.

© 1999 Microchip Technology Incorporated.

IBM is a registered trademark of International Business Machines Corporation.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

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

PICDEM-1 User's Guide



Table of Contents

Chapter 1. Introduction

1.1 Welcome 1

1.2 PICDEM-1 Demonstration Board 2

1.3 Sample Devices 3

1.4 Sample Programs 3

1.5 PICDEM-1 User's Guide 3

1.6 Reference Documents 4

Chapter 2. Getting Started

2.1 PICDEM-1 as a Stand-alone Board –
Preprogrammed Device 5

2.2 PICDEM-1 as a Stand-alone Board –
Sample Programs 6

2.3 PICDEM-1 Used with the PICMASTER Emulator – Sample Programs
8

2.4 PICDEM-1 Used with MPLAB-ICE – Sample Programs 10

Chapter 3. Tutorials

3.1 Main Routine 12

3.2 Source Code – PIC16C54 13

Chapter 4. A/D Demo – PIC16C71

4.1 Main Routine 16

4.2 InitializeAD Routine 17

4.3 SetupDelay Routine 17

4.4 Detailed Code Description 18

4.5 Source Code – PIC16C71 20

Chapter 5. EEPROM Demo – PIC16F84

5.1 Main Routine 24

5.2 WriteAll Routine 25

5.3 CheckAll Routine 26

PICDEM-1 User's Guide

5.4	BlinkLEDs Routine	27
5.5	Detailed Code Description	27
5.6	Source Code – PIC16F84	32

Chapter 6. USART Demo – PIC17C42

6.1	Main Routine	36
6.2	Service_pref and Service_rcv Routines	37
6.3	Init_ports and Init_serial Routines	38
6.4	Xmit Routine	39
6.5	Detailed Code Description	39
6.6	Source Code – PIC17C42	42

Appendix A. Hardware Detail

A.1	Processor Sockets	45
A.2	Display	45
A.3	Power Supply	45
A.4	RS-232 Serial Port	45
A.5	Switches	46
A.6	Oscillator Options	46
A.7	Analog Input	46
A.8	Sample Devices	46
A.9	Board Layout and Schematics	47

Index

Index	49
-------------	----

Worldwide Sales and Service

Worldwide Sales and Service	52
-----------------------------------	----

PICDEM-1 USER'S GUIDE

Chapter 1. Introduction

1.1 Welcome

Thank you for purchasing the PICDEM-1 demonstration board from Microchip Technology Incorporated. The PICDEM-1 is a simple board which demonstrates the capabilities of several Microchip microcontrollers, specifically from the PIC16C5X, PIC16C7XX, PIC16X8X, and PIC17C4X families.

The PICDEM-1 can be used stand-alone with a programmed part, or with an emulator system, such as PICMASTER[®]. Sample programs are provided to demonstrate the unique features of the supported devices.

The PICDEM-1 Kit comes with the following:

1. PICDEM-1 Demonstration Board
2. Sample device
3. Sample programs (3.5-inch disk)
4. PICDEM-1 Demonstration Board User's Guide (this document)

If you are missing any part of the kit, please contact your nearest Microchip sales office, listed in the back of this publication, for help.



Figure 1.1: PICDEM-1 Kit

PICDEM-1 User's Guide

1.2 PICDEM-1 Demonstration Board

The PICDEM-1 demonstration board has the following hardware features:

1. 18-, 28- and 40-pin DIP sockets (Although 3 sockets are provided, only one device may be used at a time.)
2. On-board +5V regulator for direct input from 9V, 100 mA AC/DC wall adapter or hooks for a +5V, 100 mA regulated DC supply.
3. RS-232C socket and associated hardware for direct connection to RS-232C interface.
4. 5K pot for devices with analog inputs.
5. Three push button switches for external stimulus and RESET.
6. Eight red LEDs connected to PORTB for displaying 8-bit binary values.
7. Socket for "canned" crystal oscillator.
8. Unpopulated holes provided for crystal connection.
9. Jumper to disconnect on-board RC oscillator (approximately 2 MHz).
10. Prototype area for user hardware.

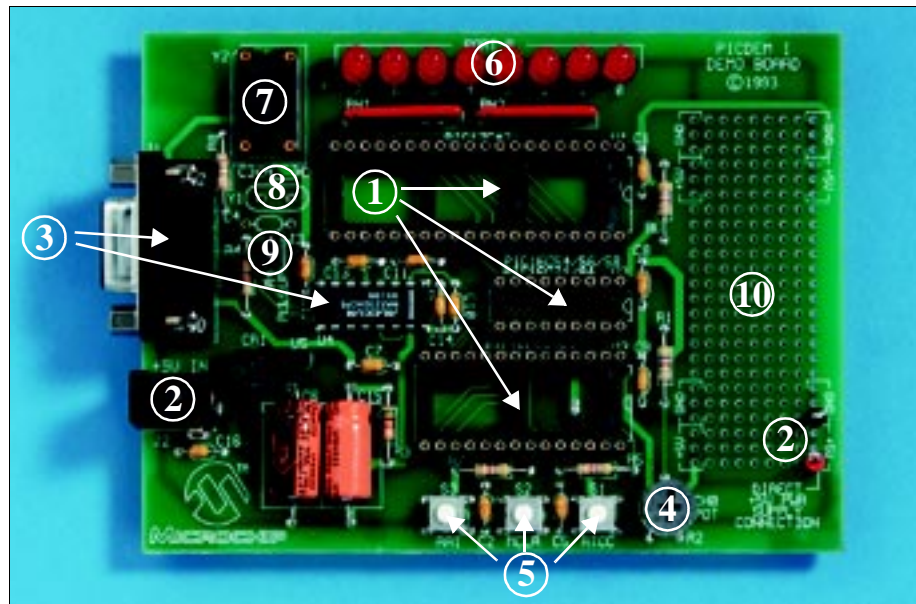


Figure 1.2: PICDEM-1 Hardware

1.3 Sample Devices

Several UV erasable devices are included. The device types may change. The supplied devices are typically one of the following:

- PIC16C54/55/56/57/58 or equivalent
- PIC16C71 or equivalent
- PIC16F83/84 or equivalent
- PIC17C42/43/44 or equivalent

1.4 Sample Programs

The PICDEM-1 Kit includes a 3.5" disk with sample demonstration programs on them. These programs may be used with the included sample device or with an emulator system. The programs are:

- `tut.asm` – Tutorial
- `demo71.asm` – PIC16C71 Demo
- `demo84.asm` – PIC16F84 Demo
- `demo42.asm` – PIC17C42 Demo

1.5 PICDEM-1 User's Guide

This document describes the PICDEM-1 demonstration board, tutorial and demonstration software, to give the user a brief overview of the PICmicro[®] supported, as well as the PICMASTER. Detailed information on individual microcontrollers may be found in the device's respective data sheet. Detailed information on the PICMASTER emulation system may be found in the *PICMASTER User's Guide* (DS51037).

Chapter 1: Introduction – This chapter introduces the PICDEM-1 and provides a brief description of the hardware.

Chapter 2: Getting Started – This chapter goes through a basic step-by-step process for getting your PICDEM-1 up and running as a stand-alone board or on the PICMASTER emulator system.

Chapter 3: Tutorials – This chapter provides a detailed description of the tutorial programs.

Chapter 4: A/D Demo – PIC16C71 – This chapter provides a detailed description of the demonstration program for the A/D module on the PIC16C71.

Chapter 5: EEPROM Demo – PIC16F84 – This chapter provides a detailed description of the demonstration program for the data EEPROM on the PIC16F84.

Chapter 6: USART Demo – PIC17C42 – This chapter provides a detailed description of the demonstration program for the serial (RS-232) communications capabilities of the PIC17C42.

Appendix A: Hardware Detail: This appendix describes in detail the hardware of the PICDEM-1 board.

PICDEM-1 User's Guide

1.6 Reference Documents

Reference Documents may be obtained by contacting your nearest Microchip sales office (listed in the back of this document) or by downloading via the Microchip website (www.microchip.com).

- *Microchip Technical Library CD-ROM* (DS00161) or individual data sheets
- *MPLAB™ IDE, Simulator and Editor User's Guide* (DS51025)
- *MPASM User's Guide with MPLINK and MPLIB* (DS33014)
- *MPLAB PRO MATE User's Guide* (DS30082)
- *PICSTART Plus User's Guide* (DS51028)
- *MPLAB PICMASTER Emulator User's Guide* (DS30421)
- *PICMASTER Probe Specification* (DS51024)
- *MPLAB-ICE User's Guide* (DS51159)
- *Microchip Third Party Guide* (DS00104)
- *PICmicro Midrange Reference Manual* (DS33023)

PICDEM-1 USER'S GUIDE

Chapter 2. Getting Started

The PICDEM-1 may be used as a stand-alone board or with an emulator. The emulator discussed in this chapter is the PICMASTER. However, other emulators may be used. For a list of PICmicro-compatible emulators, please refer to the *Microchip Third Party Handbook* or the *Development Tool Ordering Guide*.

2.1 PICDEM-1 as a Stand-alone Board – Preprogrammed Device

The PICDEM-1 may be demonstrated immediately by following the steps listed below:

- Make sure the preprogrammed sample device is in the appropriate socket on the PICDEM-1 board.
- Make sure there is a jumper on J3 (to enable the on-board RC oscillator).
- Apply power to the PICDEM-1 (Figure 2.1). For information on acceptable power sources, see Appendix A.
- Press pushbutton S3 repeatedly to see the LEDs count up from 00h to FFh. Press pushbutton S2 to reset.



Figure 2.1: PICDEM-1 Stand-Alone

PICDEM-1 User's Guide

2.2 PICDEM-1 as a Stand-alone Board – Sample Programs

To demonstrate PICDEM-1 operation with one of the sample programs, the sample device will have to be erased and reprogrammed. Once the device has been reprogrammed:

- Make sure the sample device is in the appropriate socket on the PICDEM-1 board.
- Make sure there is a jumper on J3 (to enable the on-board RC oscillator).
- Apply power to the PICDEM-1 (Figure 2.1). For information on acceptable power sources, see Appendix A.
- Consult the appropriate chapter in this document for information on the execution of each demo.

2.2.1 Erasing the Sample Device

To erase an EPROM device:

- Remove any labels covering the device window. If you do not have a windowed device (Figure 2.2), you cannot reprogram it. A windowed version of all EPROM devices may be ordered by requesting the JW package.
- Place the device in an Ultraviolet (UV) EPROM Eraser. The amount of time required to completely erase a UV erasable device depends on: the wavelength of the light, its intensity, distance from UV source, and the process technology of the device (how small are the memory cells).
- Verify that the device is blank (e.g., perform a blank check) before attempting to program it.

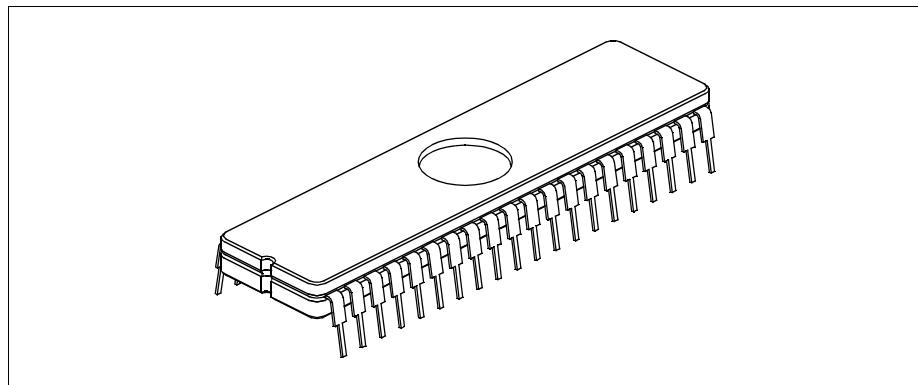


Figure 2.2: Windowed Device

Chapter 2. Getting Started

To erase an EEPROM/Flash device:

- Enable your programmer in MPLAB. If you change programmers, you will have to restart MPLAB.
- Place the device in the programmer.
- Select *Erase Program Memory* or the equivalent erase command from the Programmer menu.

Note: You do not have to erase an EEPROM/Flash device before reprogramming it.

2.2.2 Reprogramming the Sample Device

To reprogram the erased sample device, the following will be necessary:

1. Sample programs installed on the hard drive

The PICDEM-1 package includes a 3.5-inch disk which contains sample programs for all the processor types supported. Instructions on how to install the programs can be found in the readme file also on the disk.

2. An assembler, such as MPASM available with MPLAB

Sample programs may be used to program the sample device once they have been assembled. Microchip Technology's MPLAB Integrated Development Environment (IDE) includes an assembler (MPASM). However, other assemblers may be used. For a list of PICmicro-compatible assemblers, please refer to the *Microchip Third Party Handbook*.

3. A device programmer, such as PRO MATE[®] II or PICSTART[®] Plus

Once the sample program is in hex file format, a programmer may be used to program a blank device. Microchip Technology's PRO MATE II or PICSTART Plus programmers may be used. Both are compatible with MPLAB. However, other programmers may be used. For a list of PICmicro-compatible programmers, please refer to the *Microchip Third Party Handbook*.

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

Note: Microchip does not recommend code protecting windowed devices.

PICDEM-1 User's Guide

2.3 PICDEM-1 Used with the PICMASTER Emulator – Sample Programs

To use the PICDEM-1 with the PICMASTER emulator, the following will be necessary:

1. PICDEM-1 demonstration board.
2. PICMASTER emulator pod.
3. PICMASTER device-specific probe.
4. Logic probes with power supply pins or external power supply (optional).
5. PC with MPLAB IDE software and MPLAB User's Guide.
6. Ribbon cable to connect pod with probe.
7. Pin connector to connect probe with PICDEM-1.

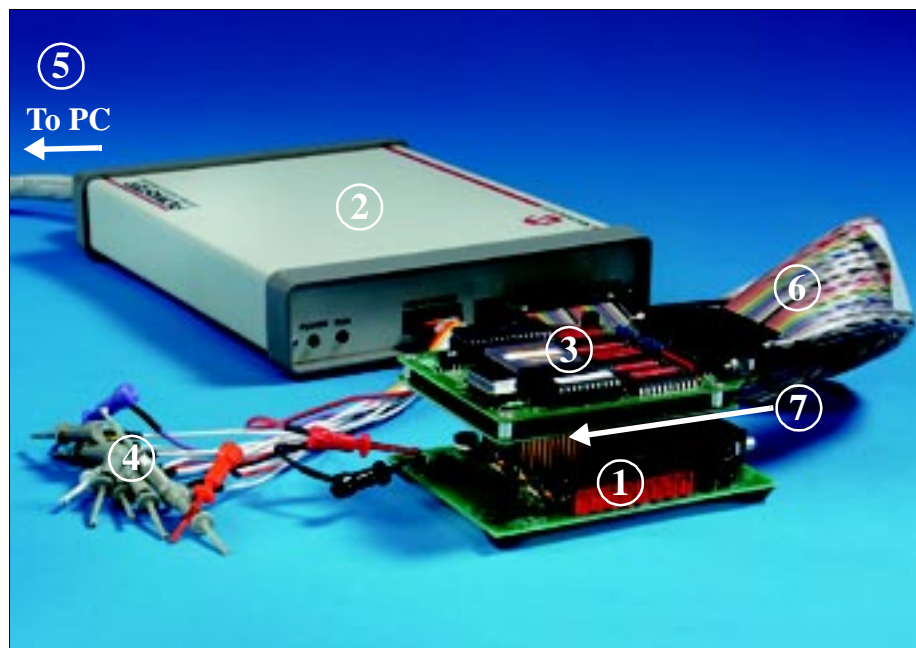


Figure 2.3: PICDEM-1 with the PICMASTER Emulator

Chapter 2. Getting Started

This chapter goes through a basic step-by-step process for getting your PICDEM-1 up and running using the PICMASTER emulator system.

1. Assemble the PICMASTER emulator
 - Follow the instructions in your PICMASTER emulator package and connect the emulator pod to an IBM[®] compatible PC. **Make certain that the power switches for both the PC and the PICMASTER are in the OFF position before connecting.**
 - Connect the appropriate probe to the PICMASTER pod using the ribbon cable provided (6 of Figure 2.3). For information on which probe to choose, refer to the *PICMASTER Probe Specification*.
 - Connect the logic probes to the emulator pod.
2. Prepare the PICDEM-1
 - Remove any preprogrammed microcontrollers installed in the PICDEM-1.
 - Make sure there is a jumper on J3 (to enable the on-board RC oscillator).
3. Prepare the probe – AC1650011 Example
 - Determine the power source and set J5 accordingly; +5 VSYS comes from the PICMASTER system (internal) or +5 VEXT comes from an external power source.
 - Determine whether or not to enable master clear \overline{MCLR} by connecting/disconnecting J6. It is recommended that J6 be connected (\overline{MCLR} enabled) for most applications.
 - Select the oscillator option to be used. Table 2.1 lists the modifications necessary for each option.

TABLE: 2.4 OSCILLATOR SELECTION

Oscillator selection on PICDEM-1	Modification on PICDEM-1	Modification on Probe
RC	J3 installed, Y2 empty, Y1 empty	Select J4 = EXTCLK, Select SW1 = 11 (RC)
Crystal	J3 removed, Y2 empty, crystal in Y1, caps C3 and C4 installed	Select J4 = EXTCLK, Select SW1 = 10 (HS), 01 (XT) or 00 (LP)
Canned Crystal	J3 removed, oscillator in Y2, Y1, C3, C4 empty	Select J4 = EXTCLK, Select SW1 = 10 (HS), 01 (XT) or 00 (LP)
PICMASTER's Oscillator	J3 removed, Y2, Y1 empty	Select J4 = INTCLK

PICDEM-1 User's Guide

4. Connect the PICDEM-1 to the emulator.
 - Connect the PICMASTER header probe to the appropriate socket on the PICDEM-1 using a connector (7 of Figure 2.3). Example: If you are emulating the PIC16F84, connect the 18-pin header on the probe card to the 18-pin socket on PICDEM-1 (U2).
5. Power supply considerations.
 - If the PICMASTER power probes are to be used to power the PICDEM-1, connect the Red probe clip to the hook marked +5V on PICDEM-1 and the Black probe clip to the hook marked GND on PICDEM-1.

Note: If an external wall mount power supply is being used, **do not** connect it to the PICDEM-1 yet.

6. Power-up sequence
The following power up sequence must be followed **exactly**:
 - Power up the PC.
 - Start Microsoft® Windows®.
 - Power up the PICMASTER.
 - If an external wall mount power supply is being used for the PICDEM-1, connect it now.
7. Installing the sample programs
The PICDEM-1 package includes a 3.5-inch disk which contains sample programs for all the processor types supported. Instructions on how to install the programs can be found in the readme file also on the disk.
8. Starting MPLAB
 - Start MPLAB by double clicking on the MPLAB icon (Windows 3.1 or later) or selecting it from the Start menu (Windows 95/98).
 - Refer to the *MPLAB User's Guide* for how to configure the application to work with the emulator and how to run the demonstration programs.

2.4 PICDEM-1 Used with MPLAB-ICE – Sample Programs

Microchip's latest emulator, MPLAB-ICE, may be used with the PICDEM-1 board by using PICMASTER probes. Refer to the *MPLAB-ICE User's Guide* (DS51159) for information on how to use PICMASTER probes with MPLAB-ICE. Then follow the setup directions from the previous section.

PICDEM-1 USER'S GUIDE

Chapter 3. Tutorials

The `tut.asm` tutorial program is pre-programmed into the sample device. This program is listed on the included 3.5-inch program disk for user reference. For example, if the sample device has been reprogrammed with another sample program, the tutorial may be reassembled and reprogrammed into the device.

The tutorial program functions as follows. Pressing the switch S3 causes the LEDs to binary count up to FFh (255 decimal). Past FFh (all LEDs on), the count rolls over to zero (all LEDs off). The program may be reset by pressing the switch S2.

For detailed information on the PICDEM-1 hardware, please refer to Appendix A.

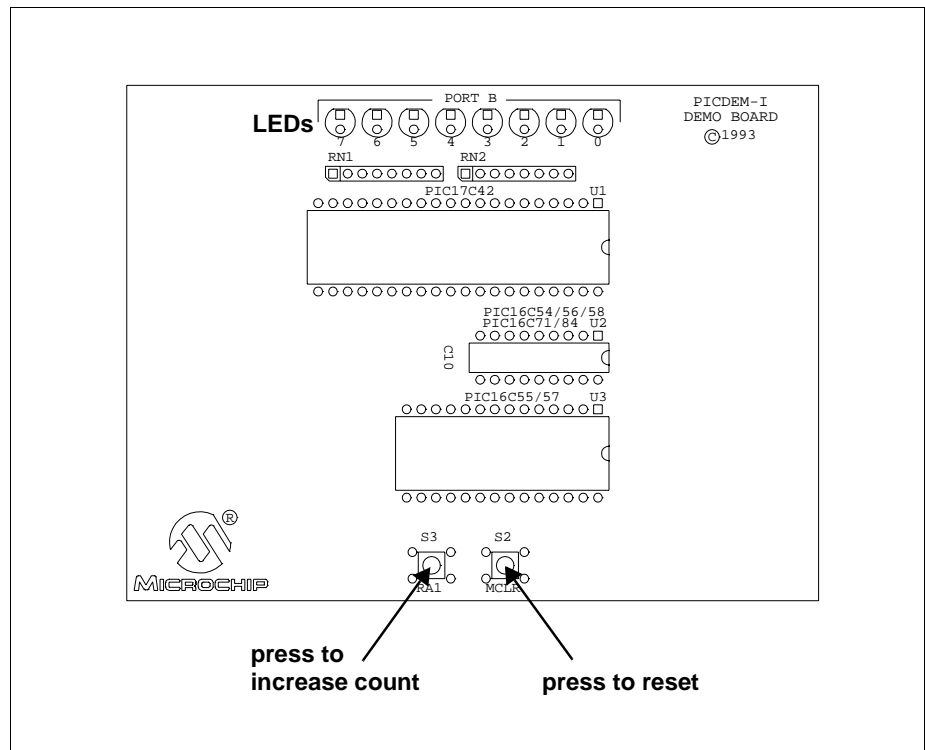


Figure 3.1: Tutorial

PICDEM-1 User's Guide

3.1 Main Routine

The tutorial program is extremely simple. It begins by configuring PORTB as all outputs to drive the eight LEDs and initializing a counter. Then it waits in an infinite loop until S3 is pressed. This increases the counter by one, and this number is displayed in binary on the LEDs. Once the count reaches FFh, the next S3 press will roll the count back to 00h.

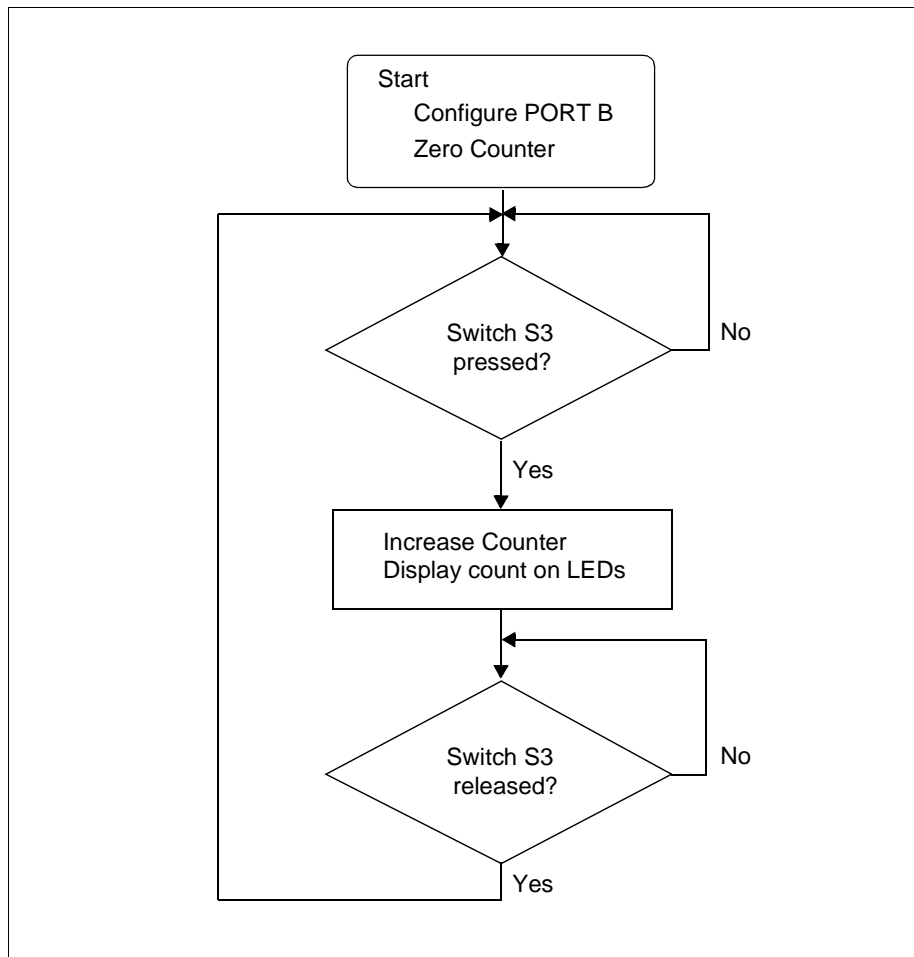


Figure 3.2: Main Routine

3.2 Source Code – PIC16C54

The tutorial program listed below is for the PIC16C54 device. It can be modified for all other supported devices. See files titled `tut???.asm`, where ?? represents the last two numbers of a device.

```
list p=16c54
;
; This program runs on the PICDEM-1 demo board.
; In the Demo board, Port B is connected to 8 LEDs.
; RA1 is connected to a switch (S3). This program increments
; the file register count each time S3 is pressed.
; The value of count is displayed on the LEDs connected
; to Port B.
; Net result is that Leds should increment in a binary
; manner every time S3 is pressed.
;
#include <P16C5X.INC>
;
COUNT equ 0x10
;
org 00h
Start
movlw 0
movwf PORTB ; config port b as output
tris PORTB
clrf COUNT ; clr count
Loop
btfss PORTA,1 ; see if RA1 pressed
goto IncCount ; yes then inc count
Endloop
goto Loop ; else check again
IncCount
incf COUNT,F ;inc count
movf COUNT,W ;
movwf PORTB ; display on port b
Debounce
btfss PORTA,1 ; wait for key release
goto Debounce ; not release then wait
Enddebounce
goto Loop ; else check key press again
;
;
org 0x01FF ; 16C54 reset vector.
goto Start
;
end
```

PICDEM-1 User's Guide

NOTES:

PICDEM-1 USER'S GUIDE

Chapter 4. A/D Demo – PIC16C71

The `demo71.asm` program is a simple implementation of the PIC16C71's analog-to-digital (A/D) converter. The program reads A/D channel 0 and displays the results on the LEDs connected to PORTB. If the potentiometer is turned all the way clockwise, all of the LEDs are off. If the potentiometer is turned all the way counter-clockwise, all of the LEDs are turned on.

PIC16C5X and PIC16X8X devices do not have an on-board A/D converter. However, an external A/D circuit may be constructed in the prototype area and the board modified to accommodate it. For more information, please refer to *AN513 – Analog to Digital Conversion Using a PIC16C54* (DS00513).

For more information on A/D module operation, please refer to the *PICmicro Midrange Microcontroller Family Reference Manual* for an operational description and a list of related application notes. For detailed information on the PICDEM-1 hardware, please refer to Appendix A.

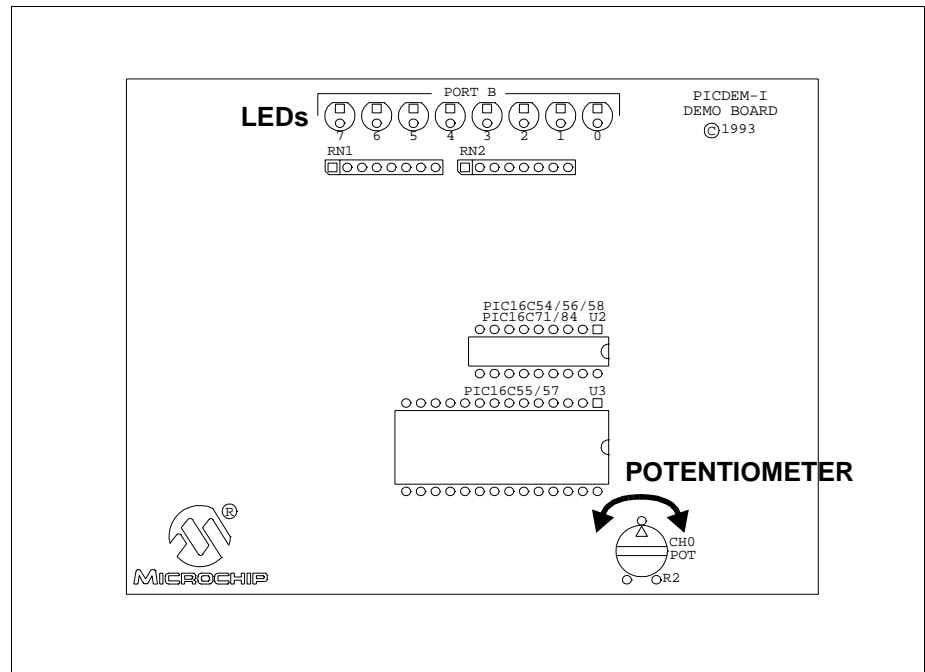


Figure 4.1: A/D Demo

PICDEM-1 User's Guide

4.1 Main Routine

The main routine of `demo71.asm` handles the initialization of the PIC16C71 and then uses the A/D interrupt flag to signal a completed A/D conversion and update the value on the LEDs. The first few lines set PORTB as outputs (for the LEDs). `InitializeAD`, the Analog-to-Digital converter initialization routine, is then called. After the return from `InitializeAD`, the value in the A/D output register is retrieved and displayed on the LEDs. The routine then waits for `TAD` (A/D conversion clock) and then starts an A/D conversion of a value specified by the potentiometer. When the A/D conversion is complete (ADIF bit set), the program loops back to display the current conversion value on the LEDs and to get another A/D value for conversion.

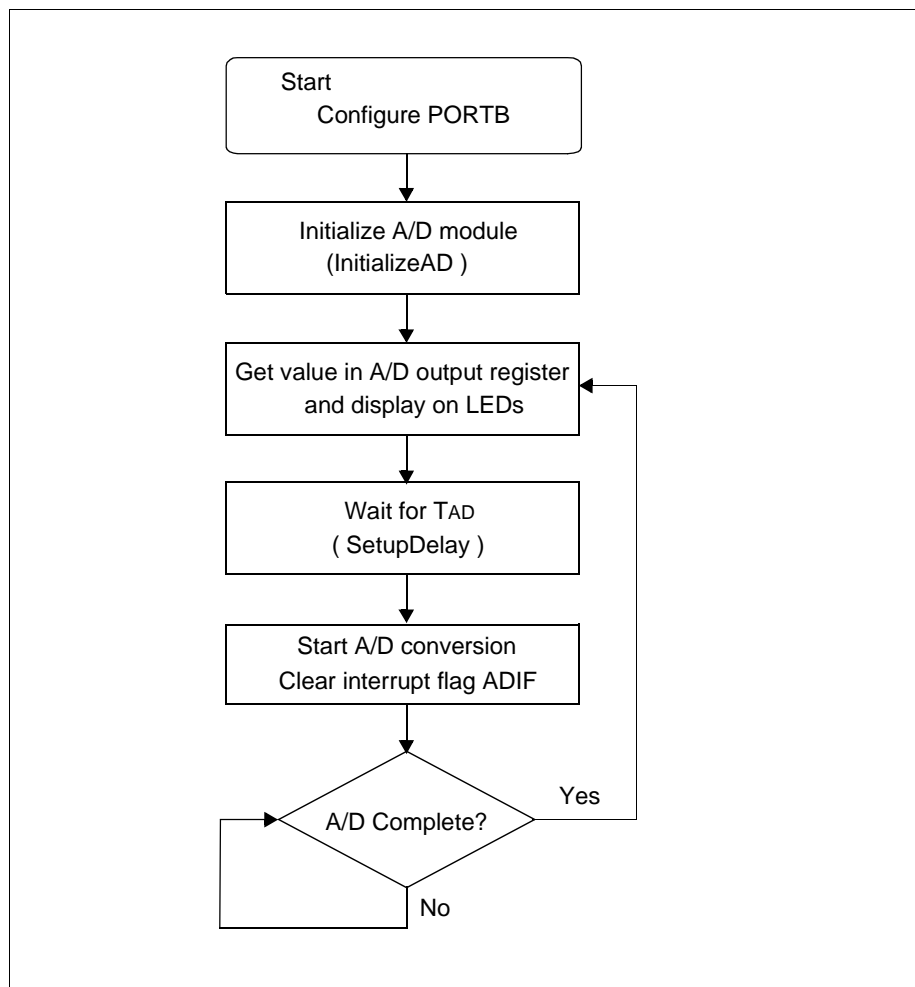


Figure 4.2: Main Routine

Chapter 4. A/D Demo – PIC16C71

4.2 InitializeAD Routine

The `InitializeAD` routine sets up the A/D converter for use with the PICDEM-1 board. The routine sets Ch0 through Ch3 as analog inputs, sets the A/D to use the internal RC oscillator, selects channel 0, and clears the A/D result register.

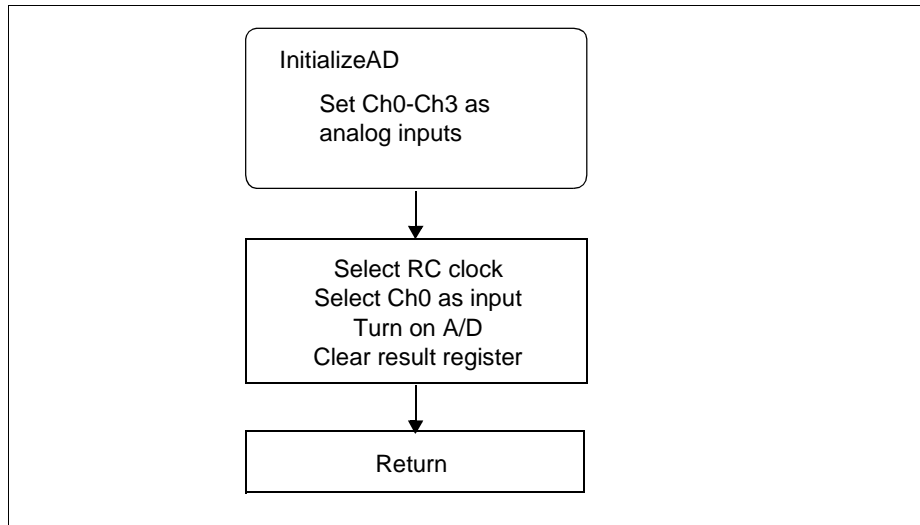


Figure 4.3: Initialize A/D

4.3 SetupDelay Routine

This subroutine delays for greater than one A/D converter clock period (2 to 6 μ s when using the internal RC clock as this program does) to allow for sampling time.

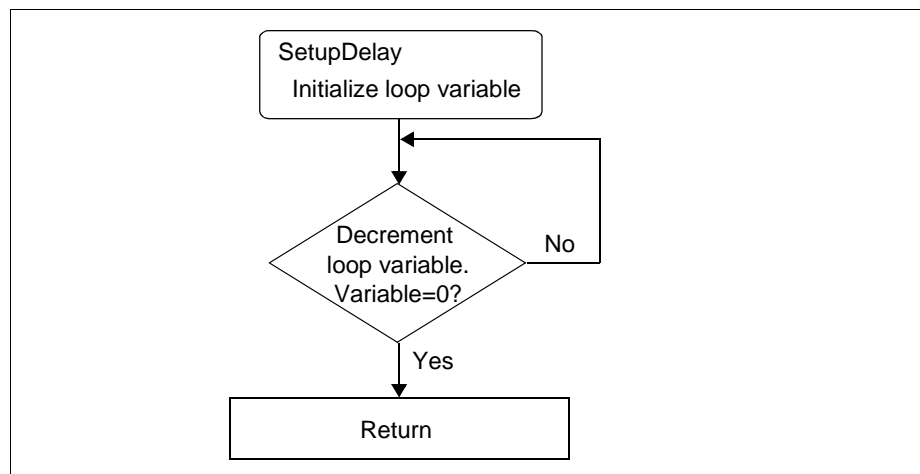


Figure 4.4: Setup Delay

PICDEM-1 User's Guide

4.4 Detailed Code Description

The program starts out by defining three constants: TEMP, a general purpose file register; ADIF, the bit location (1) of the A/D interrupt flag in ADCON0; and ADGO, the A/D GO/DONE bit in ADCON0. The next sections of code define the reset and interrupt vectors.

```
TEMP    equ    10h
ADIF    equ    1           ;A/D interrupt flag
ADGO    equ    2           ;A/D enable bit
;
    org    0x00
    goto   Start          ;reset vector
;
    org    0x04
    goto   Service_int    ;interrupt vector
```

The actual program begins at Start, where PORTB is defined as all outputs by writing 0's to the PORTB tristate register TRISB. Note that 0's were written to the PORTB register prior to setting the port to outputs so that the port would initialize with all outputs in the low state.

```
Start
    movlw  B'00000000'    ;set port b as
    movwf  PORTB          ;all outputs
    bsf    STATUS, RP0
    CLRF   TRISB          ;
    bcf    STATUS, RP0
;
    call   InitializeAD
```

After initializing PORTB, InitializeAD is called. The page register is set for page 1, RA0 through RA3 are selected to be analog inputs, and VREF is selected to be VDD (if all four pins are selected to be analog inputs, VREF is set to VDD by default) by writing to ADCON1. The page register is then reset for page 0, the internal RC oscillator is selected for the conversion clock, and ADON, the A/D on/off bit is set so that the A/D converter module is operating. The last action this routine takes before returning is to clear the A/D result register.

```
InitializeAD
    bsf    STATUS,5       ;select pg1
    movlw  B'00000000'    ;select ch0-ch3...
    movwf  ADCON1         ;as analog inputs
    bcf    STATUS,5       ;select pg0
    movlw  B'11000001'    ;select:RC,ch0..
    movwf  ADCON0         ;turn on A/D.
    clrf   ADRES         ;clr result reg.
    return
```

Chapter 4. A/D Demo – PIC16C71

The Update routine reads the A/D result register, writes the value it got to the LEDs connected to PORTB, calls `SetupDelay`, clears the interrupt flag and enables the next conversion.

Update

```
movf    ADRES,W           ;get a/d value
movwf   PORTB             ;output to port b
call    SetupDelay        ;setup time >= 20uS.
bcf     ADCON0,ADIF       ;clear int flag
bsf     ADCON0,ADGO       ;start new conversion
```

The `SetupDelay` routine provides a software delay so that the sample and hold circuit has enough time to settle before the A/D conversion is started. The SD loop takes 2 μ s when TEMP is not equal to zero, so a value of nine loaded into TEMP would provide a delay of 18 μ s. With the 2 μ s execution time of the two moves before the SD loop and the call and return execution times, a delay of greater than 20 μ s is provided for settling.

SetupDelay

```
movlw   .3
movwf   TEMP
```

SD

```
decfsz  TEMP, F
goto    SD
return
```

The Loop routine tests the A/D interrupt flag, and loops until it indicates that the conversion has finished (`ADIF = 0`). When the conversion has finished, it jumps back to Loop so that the A/D value can be read and output to the LEDs.

Loop

```
btfsc   ADCON0,ADIF      ;a/d done?
goto    Update           ;yes then update new value.
goto    Loop             ;no then keep checking
```

PICDEM-1 User's Guide

4.5 Source Code – PIC16C71

```
LIST P=16C71
;
; TITLE "Single channel A/D (SAD)"
; This program is a simple implementation of the PIC16C71's
; A/D. One channel is used (CH0).
; The A/D is configured as follows:
;     Vref = +5V internal.
;     A/D Osc. = internal RC
;     A/D Channel = CH0
; Hardware for this program is the PICDEM1 board. The program
; converts the potentiometer value on CH0 and displays it as
; a 8 bit binary value on PORTB.
;
;
;     #include <P16C71.INC>
;
TEMP     equ     10h
ADIF     equ     1           ;A/D interrupt flag
ADGO     equ     2           ;A/D enable bit
;
;     org     0x00
;     goto    Start         ;reset vector
;
;     org     0x04
;     goto    Service_int   ;interrupt vector
;
;
;     org     0x10
Start
    movlw   B'00000000'     ;set port b as
    movwf   PORTB          ;all outputs
    bsf     STATUS, RP0
    CLRF    TRISB          ;
    bcf     STATUS, RP0
;
;     call    InitializeAD
Update
    movf    ADRES,W        ;get a/d value
    movwf   PORTB         ;output to port b
    call    SetupDelay     ;setup time >= 20uS.
    bcf     ADCON0,ADIF    ;clear int flag
    bsf     ADCON0,ADGO    ;start new conversion
Loop
    btfsc   ADCON0,ADIF    ;a/d done?
    goto    Update        ;yes then update new value.
    goto    Loop          ;no then keep checking
;
;
```


Chapter 4. A/D Demo – PIC16C71

```
;No interrupts are enabled, so if the program ever reaches here,  
;it will be returned with the global interrupts disabled.
```

```
Service_int  
    return                ;do not enable global.
```

```
;  
;  
;
```

```
;Initializes and sets up the A/D hardware.  
;Select ch0 to ch3 as analog inputs, fosc/2 and read ch0.
```

```
;
```

```
InitializeAD  
    bsf     STATUS,5        ;select pg1  
    movlw   B'00000000'     ;select ch0-ch3...  
    movwf   ADCON1          ;as analog inputs  
    bcf     STATUS,5        ;select pg0  
    movlw   B'11000001'     ;select:RC,ch0..  
    movwf   ADCON0          ;turn on A/D.  
    clrf   ADRES            ;clr result reg.  
    return
```

```
;
```

```
;This routine is a software delay of 20uS for the a/d setup.  
;At 4Mhz clock, the loop takes 2uS, so initialize TEMP with  
;a value of 9 to give 18uS, plus the move etc. should result in  
;a total time of > 20uS.
```

```
SetupDelay  
    movlw   .3  
    movwf   TEMP  
SD  
    decfsz  TEMP, F  
    goto   SD  
    return
```

```
END
```

PICDEM-1 User's Guide

NOTES:

PICDEM-1 USER'S GUIDE

Chapter 5. EEPROM Demo – PIC16F84

The `demo84.asm` program demonstrates the use of the on-board data EEPROM of the PIC16X8X devices. The program writes 00h through FFh to each address of the data EEPROM, and then reads back the value. The LEDs indicate the data presently being written. If an error occurs, all of the LEDs flash.

For detailed information on the PICDEM-1 hardware, please refer to Appendix A.

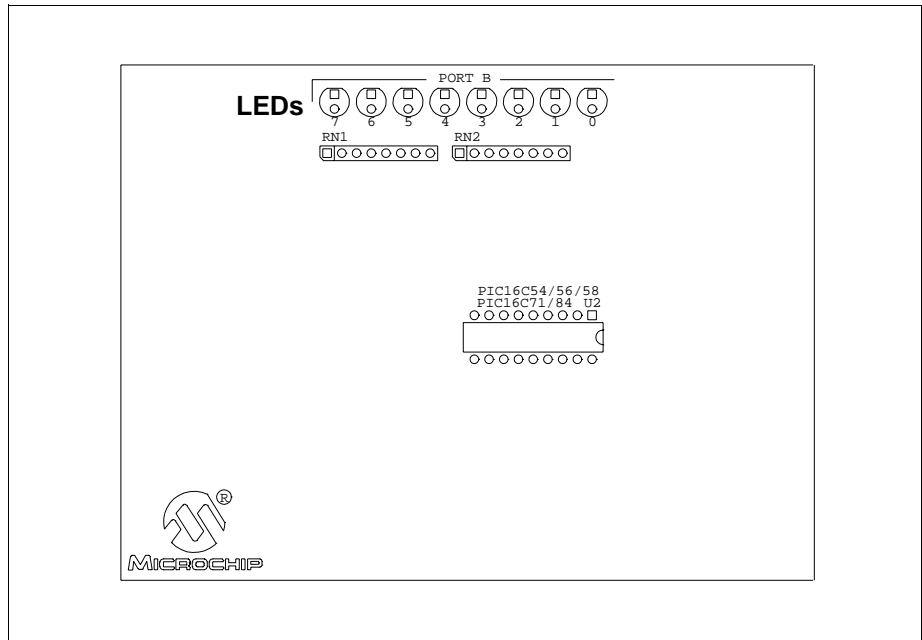


Figure 5.1: EEPROM Demo

PICDEM-1 User's Guide

5.1 Main Routine

The program begins by clearing all interrupt flags and configuring PORTB as outputs. Then the value of PORTB is used, plus one, as the value written to all EEPROM locations (`WriteAll`). These values are then checked (`CheckAll`). If no errors are found, the value displayed on PORTB is increased by one and the program loops back to use this value, plus one, to write to all EEPROM locations again. Once the value written has reached FFh, adding one will cause it to roll over to 00h.

If an error is encountered during the check, the program branches to `BlinkLEDs`, which flashes the LEDs.

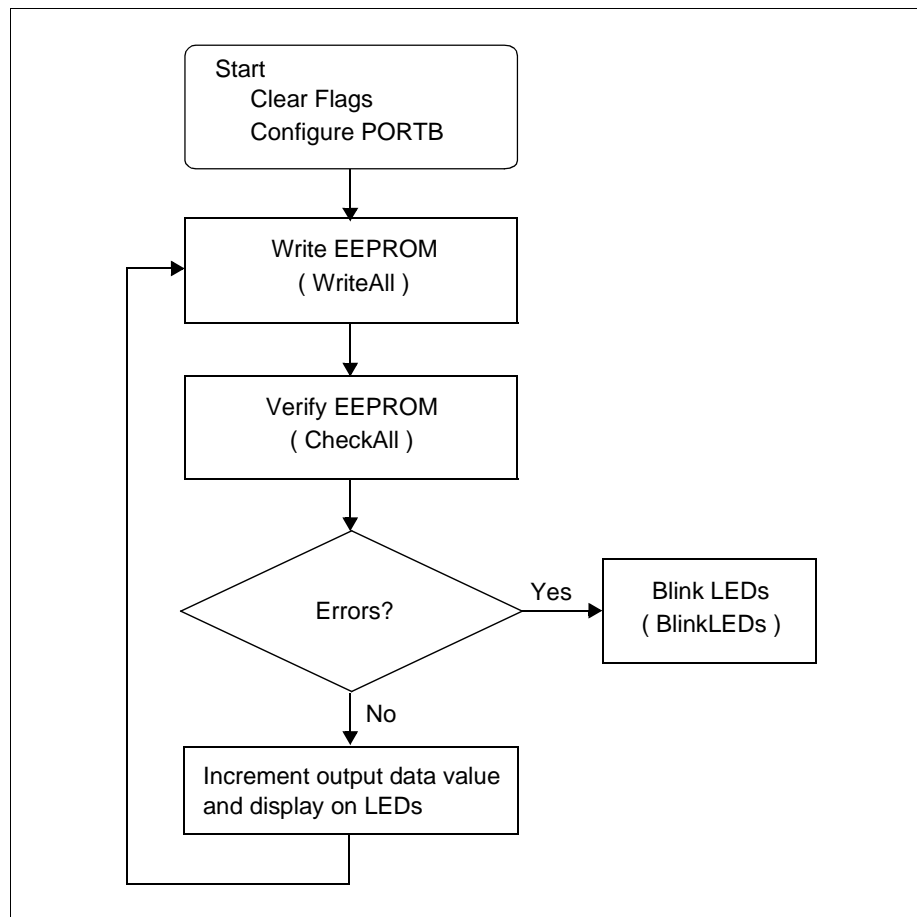


Figure 5.2: Main Routine

Chapter 5. EEPROM Demo – PIC16F84

5.2 WriteAll Routine

This routine writes the value on the LEDs, plus one, to all EEPROM locations.

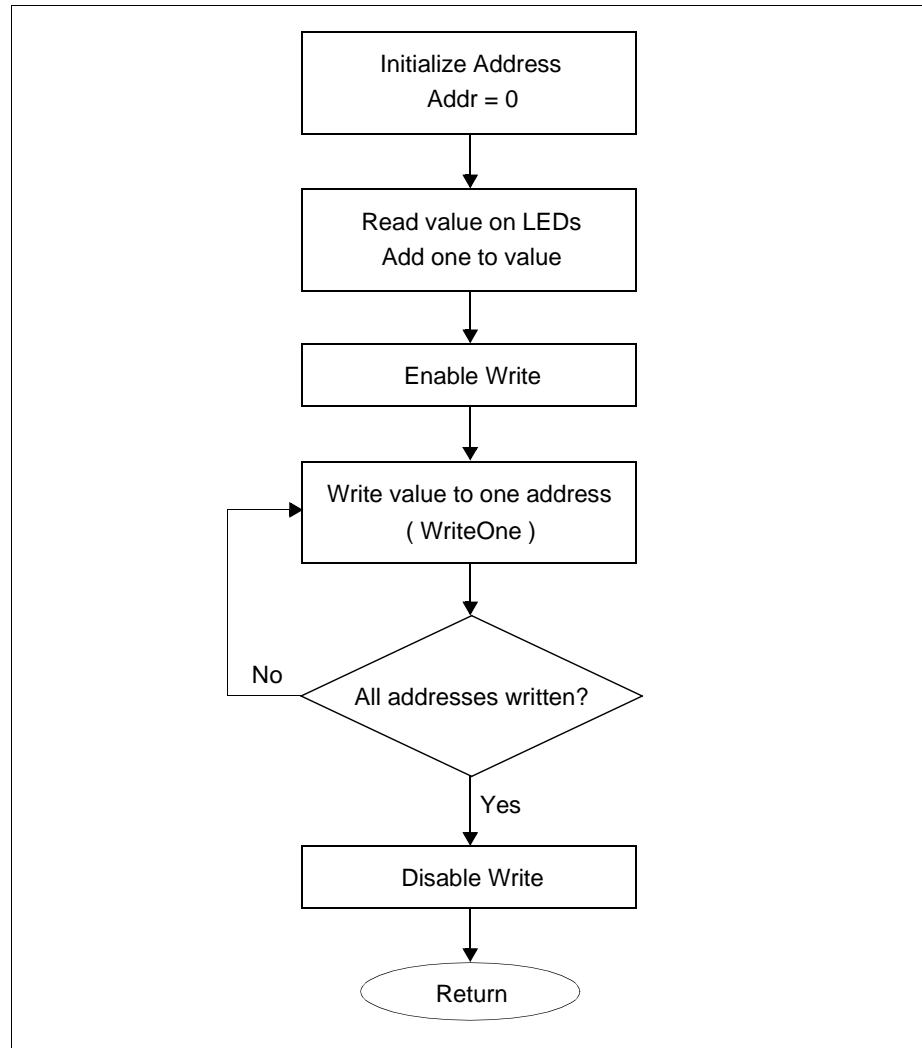


Figure 5.3: Write All

5.3 CheckAll Routine

This routine checks all of the EEPROM locations written by `WriteAll`. If an error is encountered, it branches to the `BlinkLEDs` routine.

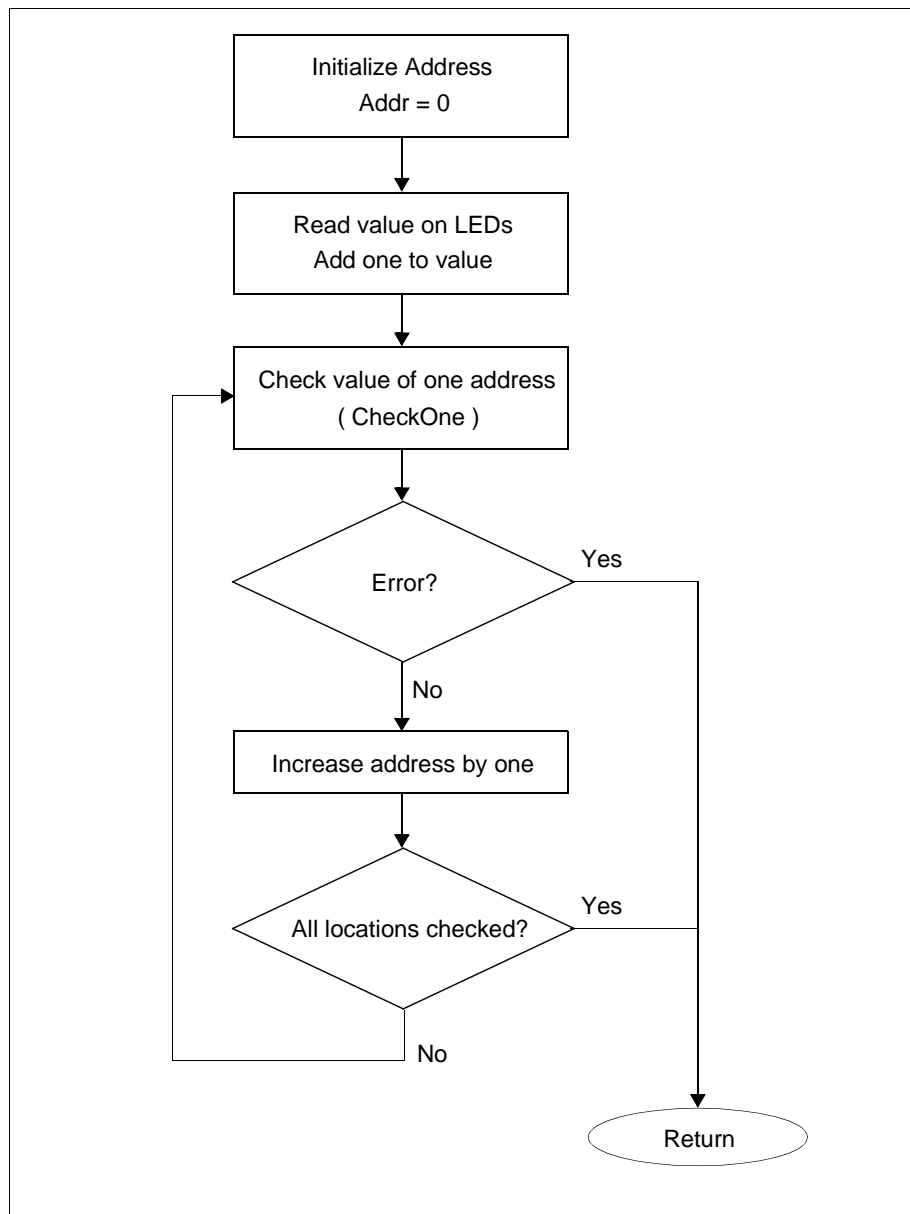


Figure 5.4: Check All

Chapter 5. EEPROM Demo – PIC16F84

5.4 BlinkLEDs Routine

This routine flashes the LEDs if an error is encountered while checking the values written to the EEPROM.

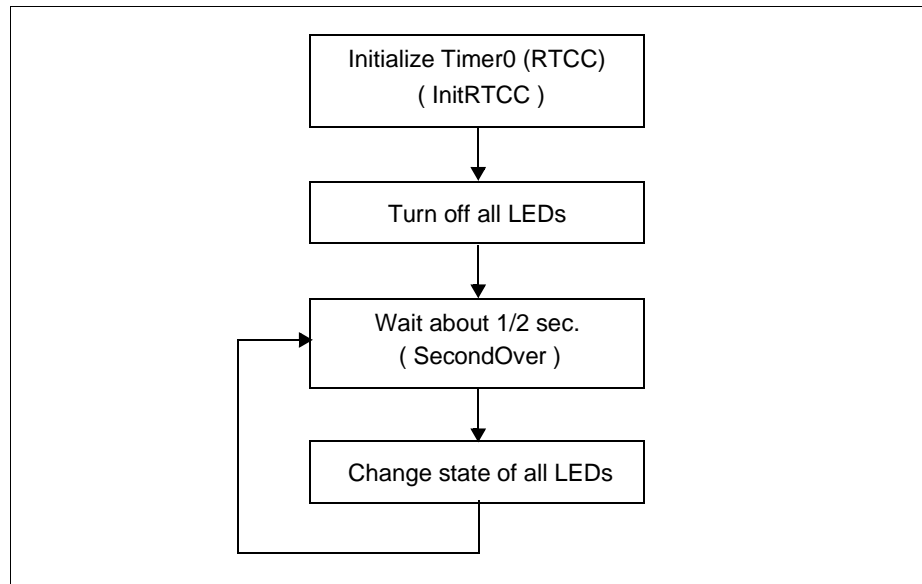


Figure 5.5: BlinkLEDs

5.5 Detailed Code Description

The program starts out by defining three constants; `TIME`, a counter used in the flashing of the LEDs, `GPFLAG`, a general purpose flag register, and `EEPERROR`, a bit used to pass the error status of the EEPROM check. The next sections of code define the reset and interrupt vectors.

```
TIME      equ      0x10
GPFLAG    equ      0x20          ;Define flag register
EEPERROR  equ      0           ;GPFLAG,0
;
    org      0
    goto    Start
;
    org      4
    goto    ServiceRtcc
```

PICDEM-1 User's Guide

The next section of code performs the initialization of GPFLAG and PORTB.

Start

```
clrf    GPFLAG           ;clr all flags
movlw   B'00000000'     ;make port b outputs
movwf   PORTB
bsf     STATUS, RP0
clrf    TRISB           ;      /
bcf     STATUS, RP0
```

After the initialization is completed the program goes into a loop where all locations of the data EEPROM are written to and then read from for verification. If all locations verify correctly, the value displayed on the LEDs is incremented, if a mismatch between what was written and what is read is found, the LEDs are blinked at a one Hertz rate.

Next

```
call    WriteAll        ;write to all locations
call    CheckAll        ;verify all locations
btfsc   GPFLAG,EEPERROR ;no error then skip
goto    BlinkLeds      ;else blink leds.
incf    PORTB, F       ;inc value in port b
goto    Next
```

WriteAll writes the value of what is displayed on the LEDs plus one to each location of the EEPROM. Notice that the write enable bit, WREN, in the EEPROM control register, EECON1 is set for the entire duration and does not have to be re-enabled for each byte.

WriteAll

```
clrf    EEADR           ;start at addr = 0
incf    PORTB,W         ;read current value+1
movwf   EEDATA         ;ld. data reg.
bsf     STATUS,RP0     ;select pg 1
bsf     EECON1,WREN    ;enable write operation
bcf     STATUS,RP0     ;select pg 0
```

WAl

```
call    WriteOne       ;write to a location
incf    EEADR, F       ;inc address
btfss   EEADR,6        ;all 64 done?
goto    WAl           ;no then do next
bsf     STATUS,RP0     ;pg 1
bcf     EECON1,WREN    ;disable write
bcf     STATUS,RP0     ;pg 0
return
```


Chapter 5. EEPROM Demo – PIC16F84

`WriteOne` is where the actual writing of the EEPROM takes place. EEPROM writes will not initiate if the proper sequence (write 55h to `EECON2`, write AAh to `EECON2`, then set the WR bit) is not followed with exact timing. In order to ensure this, interrupts have not been enabled. After the writing of the data, and initiating the write sequence, the loop, `W01`, waits for the WR bit to go low indicating that the write has completed.

Note: The timing in `WriteOne` is critical. Do not single step or do a software trace through this routine.

```
WriteOne
    bsf    STATUS,RP0        ;page 1
    movlw  0x55              ;do write seq.
    movwf  EECON2            ;
    movlw  0xaa              ;
    movwf  EECON2            ;
    bsf    EECON1,WR        ;initiate write

W01
    btfsc  EECON1,WR        ;write complete?
    goto   W01              ;no then keep checking
    bcf    STATUS,RP0        ;pg 0
    return
```

Refer to PIC16F8X Data Sheet for EEPROM programming requirements.

`CheckAll` reads each location of the EEPROM and verifies that it matches the value displayed on the LEDs plus one. If the value does not match, bit `EEPERROR` of file register `GPFLAG` will be set and the routine will return without verifying the other locations.

```
CheckAll
    clrf   EEADR             ;start at addr = 0
    incf   PORTB,W          ;ld. data to inspect

CA1
    call   CheckOne         ;check location
    btfsc  GPFLAG,EEPERROR  ;any error?
    return ;yes then quit
    incf   EEADR, F         ;inc address
    btfss  EEADR,6          ;all 64 checked?
    goto   CA1              ;no then do next
    return
```

`CheckOne` does the actual reading of the EEPROM byte and verifying that it is correct. Although it is not necessary to check any flag bits during EEPROM reads, they can be done immediately after setting the address in `EEADR` and setting the read control bit (`RD EECON1<0>`), the RD bit will be cleared by the hardware when the data has been placed in the `EEDATA` register. If there is a verification error, the `EEPERROR` bit is set in the `GPFLAG` register.

PICDEM-1 User's Guide

```
CheckOne
    bsf    STATUS,RP0      ;pg 1
    bsf    EECON1,RD      ;do a read
CO1
    btfsc  EECON1,RD      ;rd done?
    goto  CO1             ;no then loop
    bcf    STATUS,RP0     ;pg 0
    xorwf  EEDATA, F      ;compare?
    btfss  STATUS,Z       ;same then skip
    bsf    GPFLAG,EEPERROR ;set error flag
    return
```

BlinkLeds sets up the RTCC to interrupt upon time-out, turns off the LEDs, and when one second has expired, toggles the LEDs.

```
BlinkLeds
    call   InitRtcc
    clrf  PORTB           ;turn off leds
BL1
    call  SecondOver     ;wait for 1/2 second
    comf  PORTB, F       ;toggle leds
    goto  BL1
```

InitRtcc assigns the prescaler to the RTCC and sets it for divide by 256. It then enables interrupts and initializes the TIME register to eight.

```
InitRtcc
    bsf    STATUS, RP0
    movlw  B'10000111'   ;rtcc inc = tcylc/256
    movwf  OPTION_REG    ;
    bcf    STATUS, RP0
    clrf  TMR0           ;start time
    movlw  B'10100000'   ;enable interrupts
    movwf  INTCON        ;
    movlw  .8            ;initialize time
    movwf  TIME
    return
```

ServiceRtcc is the interrupt service routine and will be called each time the RTCC times out. It first verifies that the interrupt was caused by the RTCC and if it was it decrements TIME. Whether it was an RTCC interrupt or not, all interrupts are cleared and the RTCC interrupt is enabled before returning.

```
ServiceRtcc
    btfsc  INTCON,T0IF   ;rtcc interrupt?
    decf  TIME, F        ;yes then dec time
    clrf  INTCON         ;clr all interrupts
    bsf   INTCON,T0IE    ;enable RTIE
    retfie                ;not zero then return
```

Chapter 5. EEPROM Demo – PIC16F84

SecondOver waits for the TIME register to reach zero, restores a count of eight in it, and returns.

```
SecondOver
    movf    TIME,W           ;check if time = 0
    btfss  STATUS,Z         ;      /
    goto   SecondOver      ;no then loop
    movlw  .8               ;load for 1/2 second
    movwf  TIME
    return
```

PICDEM-1 User's Guide

5.6 Source Code – PIC16F84

```
list p=16f84
;
; This program demonstrates the PIC16F84.
; To demo the EEPROM capability of the PIC16F84, then
; a value is incrementally written and verified in all
; 64 locations of the EEPROM. If an error occurs at any location,
; then all leds on portb will blink at a 1 sec rate.
;
#include <P16F84.INC>
;
TIME      equ      0x10
GPFLAG    equ      0x20          ;Define flag register
EEPERROR  equ      0            ;GPFLAG,0
;
org       0
goto     Start
;
org       4
goto     ServiceRtcc
;
org       10
Start
    clrf   GPFLAG          ;clr all flags
    movlw B'00000000'     ;make port b outputs
    movwf PORTB
    bsf   STATUS, RP0
    clrf TRISB             ;      /
    bcf  STATUS, RP0
Next
    call  WriteAll         ;write to all locations
    call  CheckAll        ;verify all locations
    btfsc GPFLAG,EEPERROR ;no error then skip
    goto  BlinkLeds      ;else blink leds.
    incf  PORTB, F       ;inc value in port b
    goto  Next
;
WriteAll
    clrf  EEADR           ;start at addr = 0
    incf  PORTB,W        ;read current value+1
    movwf EEData         ;ld. data reg.
    bsf   STATUS,RP0     ;select pg 1
    bsf   EECON1,WREN    ;enable write operation
    bcf   STATUS,RP0     ;select pg 0
```

Chapter 5. EEPROM Demo – PIC16F84

```
WAl
    call    WriteOne      ;write to a location
    incf   EEADR, F      ;inc address
    btfss  EEADR,6       ;all 64 done?
    goto   WAl          ;no then do next
    bsf    STATUS,RP0    ;pg 1
    bcf    EECON1,WREN   ;disable write
    bcf    STATUS,RP0    ;pg 0
    return

;
WriteOne
    bsf    STATUS,RP0    ;page 1
    movlw  0x55          ;do write seq.
    movwf  EECON2        ;      /
    movlw  0xaa          ;      /
    movwf  EECON2        ;      /
    bsf    EECON1,WR     ;initiate write
W01
    btfsc  EECON1,WR     ;write complete?
    goto   W01          ;no then keep checking
    bcf    STATUS,RP0    ;pg 0
    return

;
CheckAll
    clrf   EEADR        ;start at addr = 0
    incf   PORTB,W      ;ld. data to inspect
CA1
    call   CheckOne     ;check location
    btfsc  GPFLAG,EEPERROR ;any error?
    return ;yes then quit
    incf   EEADR, F     ;inc address
    btfss  EEADR,6     ;all 64 checked?
    goto   CA1         ;no then do next
    return

;
CheckOne
    bsf    STATUS,RP0    ;pg 1
    bsf    EECON1,RD     ;do a read
CO1
    btfsc  EECON1,RD     ;rd done?
    goto   CO1         ;no then loop
    bcf    STATUS,RP0    ;pg 0
    xorwf  EEDATA, F     ;compare?
    btfss  STATUS,Z     ;same then skip
    bsf    GPFLAG,EEPERROR ;set error flag
    return

;
```

PICDEM-1 User's Guide

```
BlinkLeds
    call    InitRtcc
    clrf   PORTB           ;turn off leds
BL1
    call   SecondOver      ;wait for 1/2 second
    comf   PORTB, F        ;toggle leds
    goto  BL1
;
InitRtcc
    bsf    STATUS, RP0
    movlw  B'10000111'     ;rtcc inc = tcylc/256
    movwf  OPTION_REG      ;      /
    bcf    STATUS, RP0
    clrf   TMR0            ;start time
    movlw  B'10100000'     ;enable interrupts
    movwf  INTCON          ;      /
    movlw  .8              ;initialize time
    movwf  TIME
    return
;
ServiceRtcc
    btfsc  INTCON,T0IF     ;rtcc interrupt?
    decf   TIME, F        ;yes then dec time
    clrf   INTCON         ;clr all interrupts
    bsf    INTCON,T0IE     ;enable RTIE
    retfie ;not zero then return
;
SecondOver
    movf   TIME,W         ;check if time = 0
    btfss  STATUS,Z       ;      /
    goto   SecondOver     ;no then loop
    movlw  .8             ;load for 1/2 second
    movwf  TIME
    return

end
```

PICDEM-1 USER'S GUIDE

Chapter 6. USART Demo – PIC17C42

The `demo42.asm` program demonstrates the USART peripheral of the PIC17C42. The program waits for S1 to be pressed, increments a counter, and transmits it through the serial port. The RX and TX lines of the serial port must be shorted at the DB9 connector (pins 2 and 3). The value received from the serial port is displayed on the LEDs.

Other devices do not have a serial port implemented in hardware. However, if the user wishes to use a serial port, one may be implemented in software. For more information, please refer to *AN555 – Software Implementation of Asynchronous Serial I/O* (DS00555).

For more information on USART module operation, please refer to the *PICmicro Midrange Microcontroller Family Reference Manual* for an operational description and a list of related application notes. For detailed information on the PICDEM-1 hardware, please refer to Appendix A.

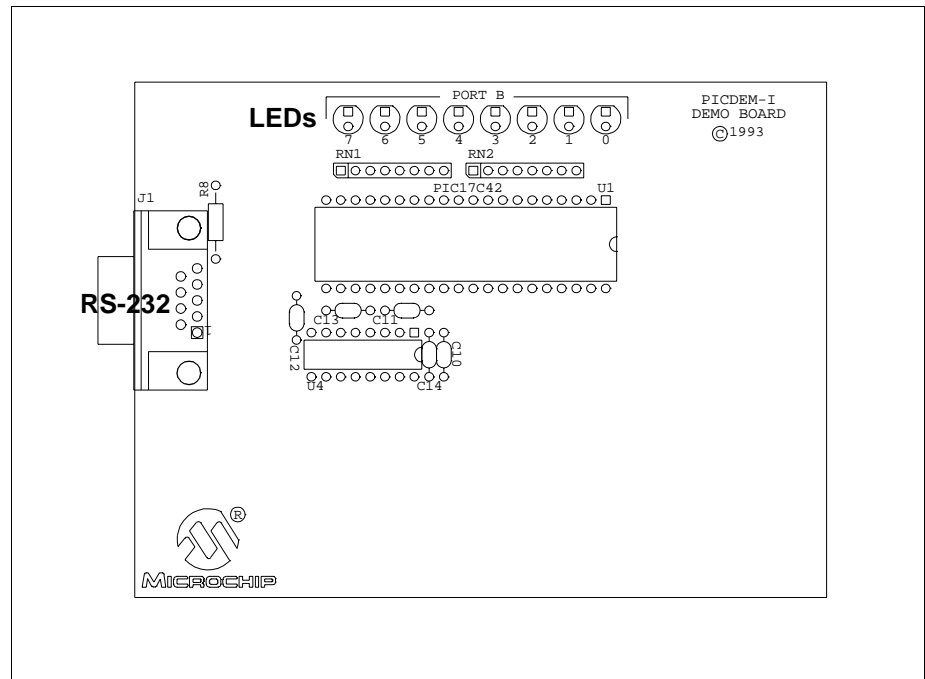


Figure 6.1: USART Demo

PICDEM-1 User's Guide

6.1 Main Routine

The program `demo42.asm` begins by setting up the interrupts. The main routine actually commences at `Start`. It first clears data memory (RAM), and then calls two subroutines to initialize the ports. After initialization, the program waits for S1 to be pressed, then goes to the `Xmit` routine. On returning from transmit, the program loops to wait for another key press.

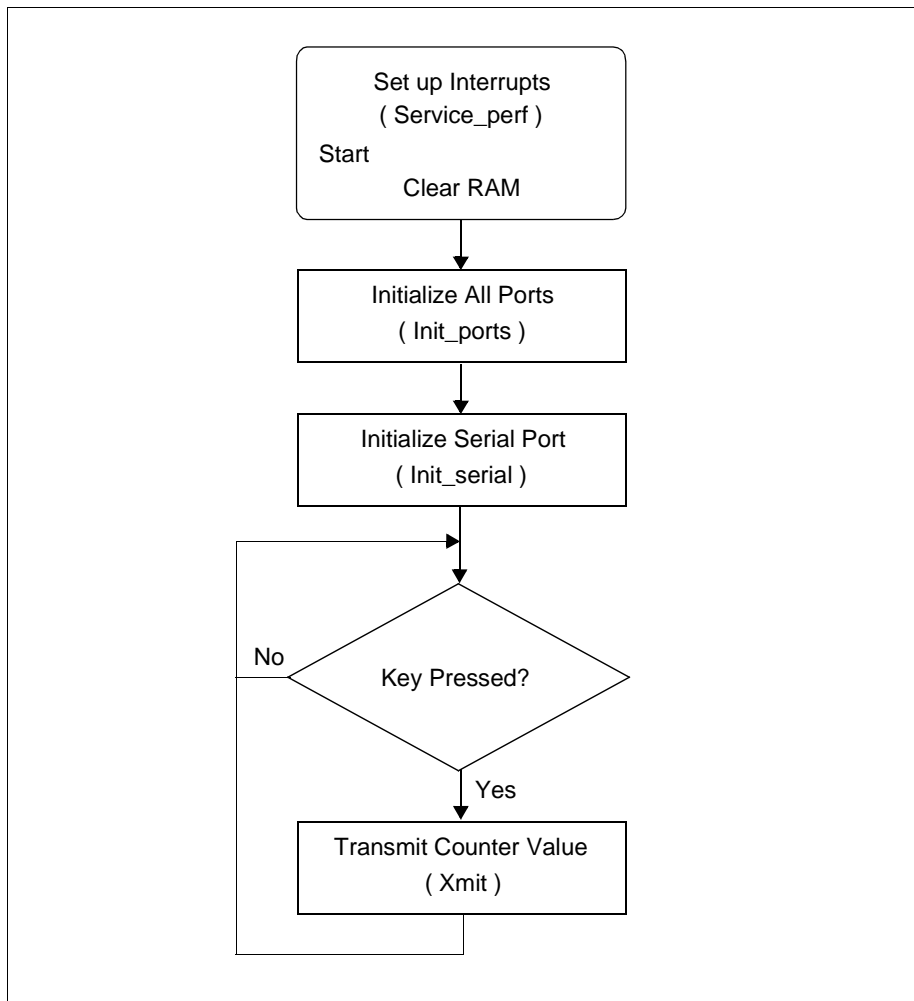


Figure 6.2: Main Routine

6.2 Service_pref and Service_rcv Routines

The routines `Service_pref` and `Service_rcv` are the peripheral interrupt service routines. When an interrupt occurs, the program goes to `Service_pref`. Interrupts are disabled, and the receive buffer is checked. If it is not full, interrupts are re-enabled and routine returns to the main program. If the buffer is full, then `Service_rcv` is called. This routine outputs the buffer data to the LEDs, clears the interrupts and then re-enables the receive interrupt.

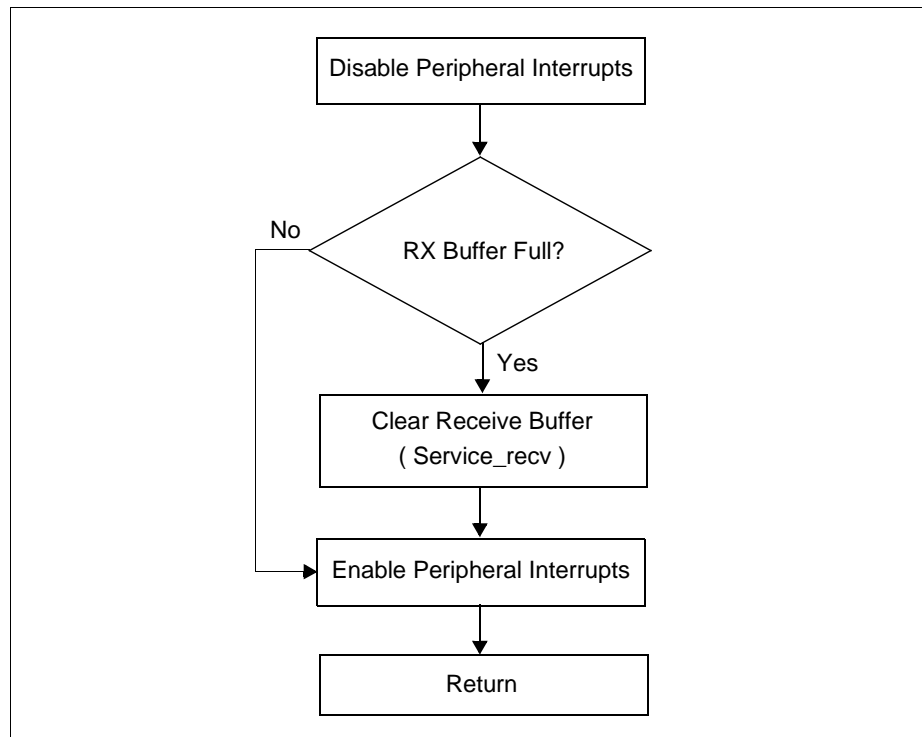


Figure 6.3: Service_pref Routine

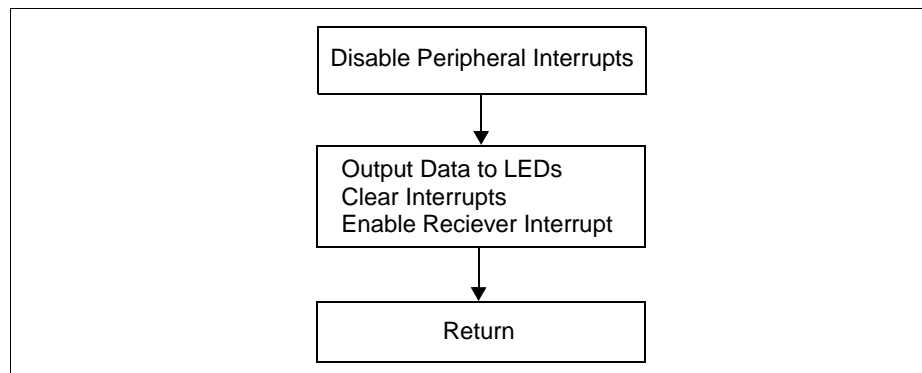


Figure 6.4: Service_rcv Routine

6.3 Init_ports and Init_serial Routines

The `Init_ports` routine simply initializes PORTB.

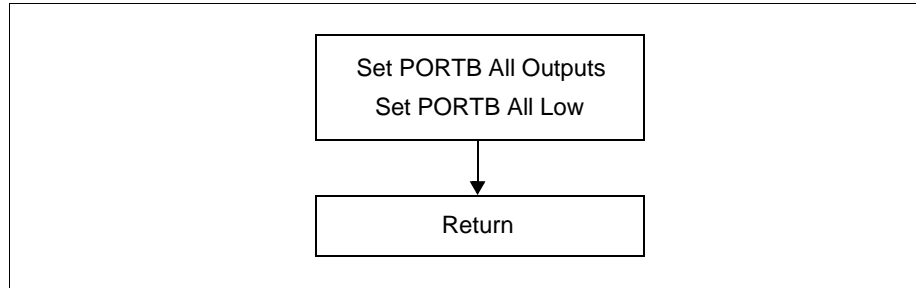


Figure 6.5: Initialize Ports Routine

`Init_serial` initializes the USART. The baud rate is set to 9600, reception and transmit are enabled, and corresponding interrupts are also enabled.

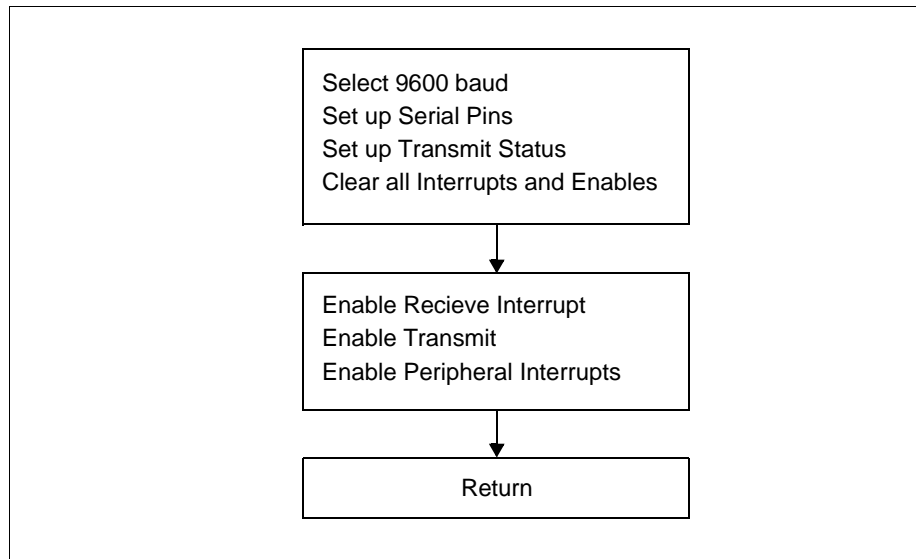


Figure 6.6: Initialize Serial Port (USART)

Chapter 6. USART Demo – PIC17C42

6.4 Xmit Routine

The `Xmit` routine increases the value of count by one and then transmits this value. Then the routine waits for S1 to be released (debounce) before returning to the main routine.

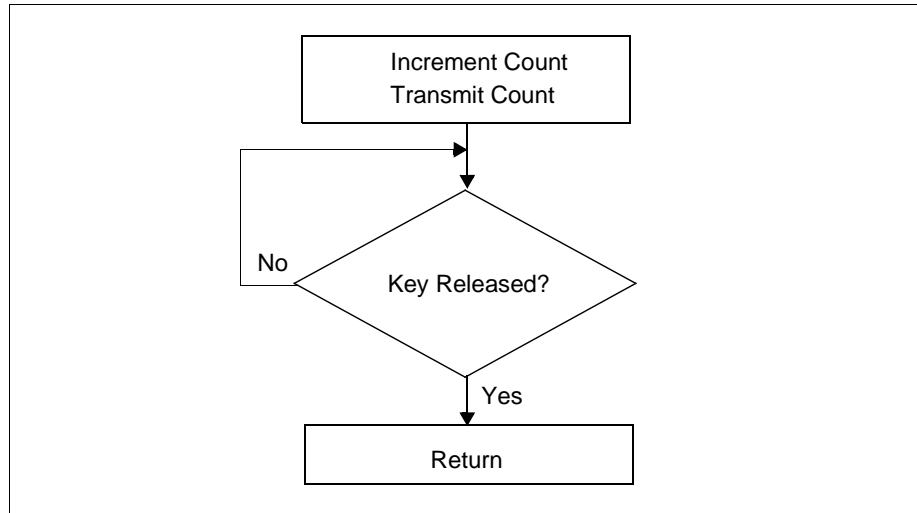


Figure 6.7: Transmit Routine

6.5 Detailed Code Description

The program begins by defining the reset, RTCC interrupt, and peripheral interrupt vectors. If for some reason the RTCC interrupt was enabled and the program reached `Rtcc_int`, the program would return with the interrupts enabled.

```
org    0
goto   Start
;
org    0x0010      ; vector for rtcc interrupt
Rtcc_int
retfie           ; return with ints enabled
;
org    0x0020      ; vector for peripheral interrupt
Perf_int
goto   Service_perf ; service the interrupts
;
org    0x0030
```

The main program routine clears RAM from 20h to 0FFh, calls routines to initialize the ports and the serial I/O, and loops while waiting for a keypress. When a keypress is detected, the main program calls the `Xmit` routine.

PICDEM-1 User's Guide

```
Start                                ; start main program
    movlw    0x20                    ; clear ram space
    movfp   WREG,FSR0                ; use indirect addressing
Start1
    clrf    INDF0, F                 ; clear ram
    incfsz  FSR0, F                  ; from 0x20 to 0xff
    goto   Start1                    ;      /
    call   Init_ports                ; initialize all ports
    call   Init_serial               ; initialize serial port
Loop
    movpf   PORTA,TEMP               ; get porta
    btfss  TEMP,1                    ; check for key press
    call   Xmit                       ; send next value
    goto   Loop                       ; check again
```

The routine `Init_ports` sets all pins of PORTB to be outputs and sets them to the low state (LEDs off).

```
Init_ports
    movlb   0                        ; select bank 0
    clrf   DDRB, F                    ; port b all ouputs
    clrf   PORTB, F                   ; and all low.
    return
```

`Init_serial` initializes the USART. The baud rate is set to 9600 for both transmit and receive, the receiver is enabled in an asynchronous continuous reception mode with 8-bit words. The transmitter is enabled in an asynchronous, 8-bit mode.

```
Init_serial
    movlb   0                        ; select bank 0
    movlw   .25                       ; select 9600 baud
    movfp   WREG,SPBRG                ;      /
    movlw   0x90                       ; set up serial pins
    movfp   WREG,RCSTA                ;      /
    clrf   TXSTA, F                   ; setup transmit status
    movlb   1                          ; select bank 1
    clrf   PIR, F                      ; clear all interrupts
    clrf   PIE, F                      ; clear all enables
    bsf    _rcie                       ; enable receive interrupt
    movlb   0                          ; select bank 0
    bsf    _txen                       ; enable transmit
    bsf    _peie                       ; enable peripheral ints
    retfie
```

Chapter 6. USART Demo – PIC17C42

The `Service_perf` routine services any interrupts received. The routine first disables all peripheral interrupts, then checks the receive buffer for data. If the receive buffer flag is set, the data is displayed on the LEDs by the `Service_rcv` routine. In either case the peripheral interrupts are re-enabled and the routine exited.

```
Service_perf
    bcf    _peie          ; disable perf int
    movlb  1              ; bank 1
    btfsc  _rbfl         ; RX buffer full?
    call   Service_rcv   ; yes then service
Exit_perf
    bsf    _peie          ; enable peripheral int
    retfie
;
; This routine services the serially received info.

Service_rcv
    movlb  0
    movfp  RCREG,WREG    ; load received value
    movwf  PORTB
End_rcv
    clrf   INTSTA, F     ; clear all int
    bsf    _rcie         ; enable receive
    return
```

The `Xmit` routine increments file register `COUNT` and outputs it to the serial port. It then waits for the key to be released before returning.

```
Xmit
    incf   COUNT, F
    movlb  0
    movfp  COUNT, TXREG
Debounce
    movpf  PORTA, TEMP   ;
    btfss  TEMP, 1      ; Wait for key release
    goto   Debounce
    return
```

PICDEM-1 User's Guide

6.6 Source Code – PIC17C42

```
list p=17c42

;      TITLE   Demo board software.
; PortB of the PIC17C42 is connected to the 8 leds and
; a switch (S1) is connected to the RTCC input. The software waits for
; S1 to be pressed at which point a counter is incremented
; and its value is transmitted through the serial port. The RX
; and TX lines (pins 2 and 3) must be shorted at the DB9 Connector so that
; the transmitted value is sent right back to the receive buffer.
; The receive interrupt is generated and the value in the
; receive buffer is displayed as a 8 bit value on port B.
; The net result is that every time S1 is pressed the LEDs will
; increment in a binary count.
;
COUNT          equ      0x20
TEMP            equ      0x19      ; Temporary storage location
;
;      #INCLUDE  <p17C42.inc>
;
;      #INCLUDE  <demo42.inc>
;
;      org      0
;      goto     Start
;
;      org      0x0010      ; vector for rtcc interrupt
Rtcc_int
;      retfie      ; return with ints enabled
;
;      org      0x0020      ; vector for peripheral interrupt
Perf_int
;      goto     Service_perf ; service the interrupts
;
;      org      0x0030
;
Start            ; start main program
;      movlw    0x20      ; clear ram space
;      movfp   WREG,FSR0  ; use indirect addressing
Start1
;      clrf    INDF0, F    ; clear ram
;      incfsz  FSR0, F    ; from 0x20 to 0xff
;      goto   Start1      ; /
;      call   Init_ports  ; initialize all ports
;      call   Init_serial ; initialize serial port
Loop
;      movpf   PORTA,TEMP ; get porta
```

Chapter 6. USART Demo – PIC17C42

```
        btfss    TEMP,1          ; check for key press
        call     Xmit           ; send next value
        goto    Loop           ; check again
;
;
Init_ports
        movlb   0              ; select bank 0
        clrfs   DDRB, F        ; port b all ouptuts
        clrfs   PORTB, F       ; and all low.
        return
;
; This routine initailizes the serial port. Baud rate, receive interrupt
; software flags are initailized.
;
Init_serial
        movlb   0              ; select bank 0
        movlw   .25            ; select 9600 baud
        movfp   WREG,SPBRG     ;      /
        movlw   0x90           ; set up serial pins
        movfp   WREG,RCSTA     ;      /
        clrfs   TXSTA, F       ; setup transmit status
        movlb   1              ; select bank 1
        clrfs   PIR, F         ; clear all interrupts
        clrfs   PIE, F         ; clear all enables
        bsf     _rcie          ; enable receive interrupt
        movlb   0              ; select bank 0
        bsf     _txen          ; enable transmit
        bsf     _peie          ; enable peripheral ints
        retfie
;
; This routine handles all the peripheral interrupts. In this example
; only the receive interrupt is handled.
; check for receive interrupts only
;
Service_perf
        bcf     _peie          ; disable perf int
        movlb   1              ; bank 1
        btfs   _rbfl          ; RX buffer full?
        call    Service_rcv    ; yes then service
Exit_perf
        bsf     _peie          ; enable peripheral int
        retfie
;
; This routine services the serially received info.

Service_rcv
        movlb   0
        movfp   RCREG,WREG     ; load received value
        movwf   PORTB
```

PICDEM-1 User's Guide

```
End_rcv
    clrf    INTSTA, F        ; clear all int
    bsf    _rcie            ; enable receive
    return

;
;
Xmit
    incf   COUNT, F
    movlb 0
    movfp COUNT, TXREG

Debounce
    movpf  PORTA, TEMP      ;
    btfss TEMP, 1          ; Wait for key release
    goto  Debounce
    return

;
    END
```


Appendix A. Hardware Detail

The PICDEM-1 hardware is extremely simple and is intended to illustrate the ease of use of several PICmicro parts. The PICDEM-1 features the following hardware elements:

A.1 Processor Sockets

Although three sockets are provided, only one device may be used at a time.

- 18-pin socket
- 28-pin socket
- 40-pin socket

A.2 Display

Eight red LEDs are connected to PORTB of each processor type. The PORTB pins are set high to light the LEDs.

A.3 Power Supply

There are two ways to supply power to PICDEM-1:

- A 9V, 100 mA unregulated AC or DC supply can be plugged into J2.
- A +5V, 100 mA regulated DC supply can be connected to the hooks provided.

PICMASTER users have a regulated +5V power supply available in the logic probe connector and can easily connect to the hooks on PICDEM-1 (Red probe to +5V and Black probe to GND).

Note: The exact power up sequence listed in Getting Started (Section 2) of this guide must be followed.

A.4 RS-232 Serial Port

An RS-232 level shifting IC has been provided with all necessary hardware to support connection of an RS-232 host through the DB9 connector. The port is configured as DCE, and can be connected to a PC using a straight through cable.

PIC17C4X devices have their RX and TX pins tied to the RX and TX lines of the Max232A. The other devices do not have a serial port implemented in hardware, but an RS-232 interface may be implemented in software using pins RA2 and RA3.

PICDEM-1 User's Guide

A.5 Switches

Three switches provide the following functions:

- S1 – Real-Time Clock/Counter (RTCC), or Timer0, input
- S2 – $\overline{\text{MCLR}}$ to hard reset the processor
- S3 – Active low switch connected to RA1

Switches S1 and S2 have debounce capacitors whereas S3 does not, allowing the user to investigate key debounce techniques.

When pressed, the switches are grounded. When idle, they are pulled high (+5V).

A.6 Oscillator Options

- RC oscillator (2 MHz approximately) supplied. This oscillator may be disabled by removing jumper J3.
- Pads provided for user furnished crystal and two capacitors.
- Socket provided for clock oscillator.

A.7 Analog Input

A 5K ohm potentiometer is connected through a series 470 ohm resistor to AN0.

The pot can be adjusted from VDD to GND to provide an analog input to the parts with an A/D module. Parts without A/D will read a digital high or low depending upon the position of the potentiometer.

A.8 Sample Devices

A sample part programmed with a simple program is included.

Table A.1 lists the I/O features and port connections for each processor type.

Table: A.1 PORT CONNECTIONS

Processor	LEDs	RS-232 TX/RX ⁽¹⁾	S1 RTCC	S2	S3	CH0 POT R2 ⁽²⁾
PIC16C5X, PIC16C7XX, PIC16X8X	PORTB	RA2/RA3	T0CKI	MCLR	RA1	RA0
PIC17C4X	PORTB	RA2/RA3	T0CKI	$\overline{\text{MCLR}}$	NC ⁽³⁾	NC ⁽³⁾

Note 1: USART module available only on PIC17C4X devices.

2: A/D module available only on PIC16C7XX devices.

3: NC: No connection

Appendix A. Hardware Detail

A.9 Board Layout and Schematics

The following figures show the parts layout (silkscreen) and schematics for the PICDEM-1 board.

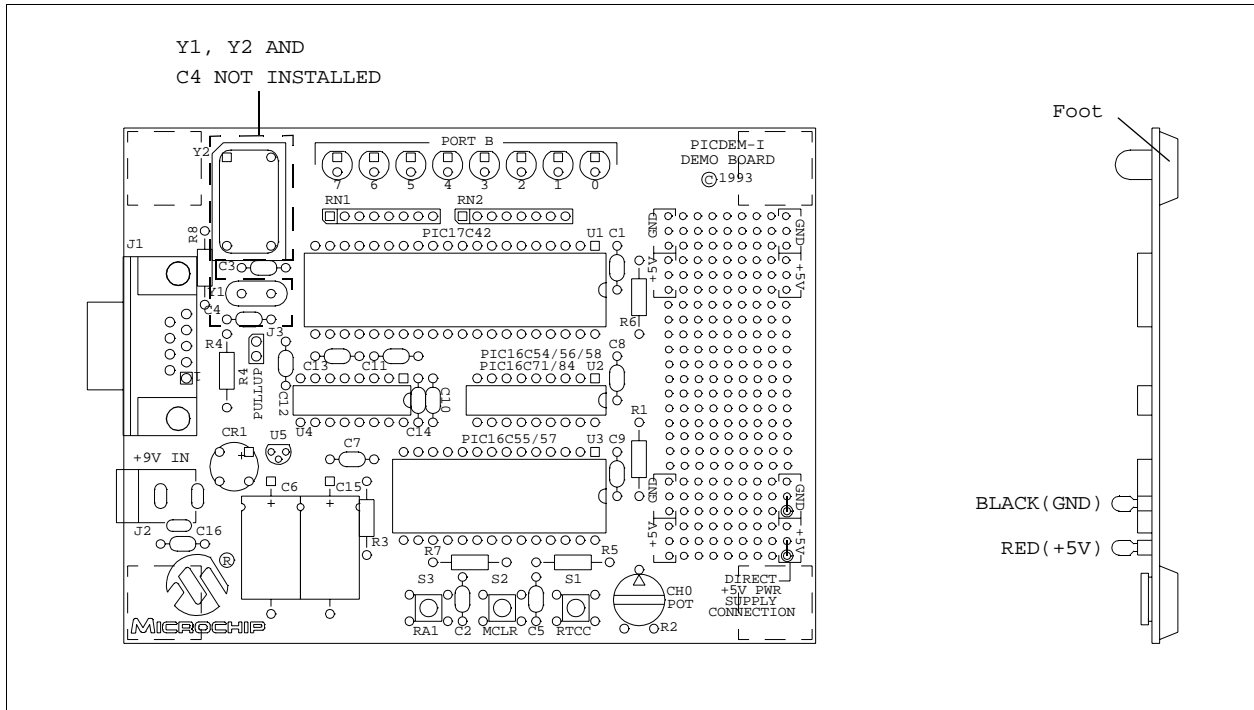


Figure A.1: PICDEM-1 Parts Layout

PICDEM-1 User's Guide

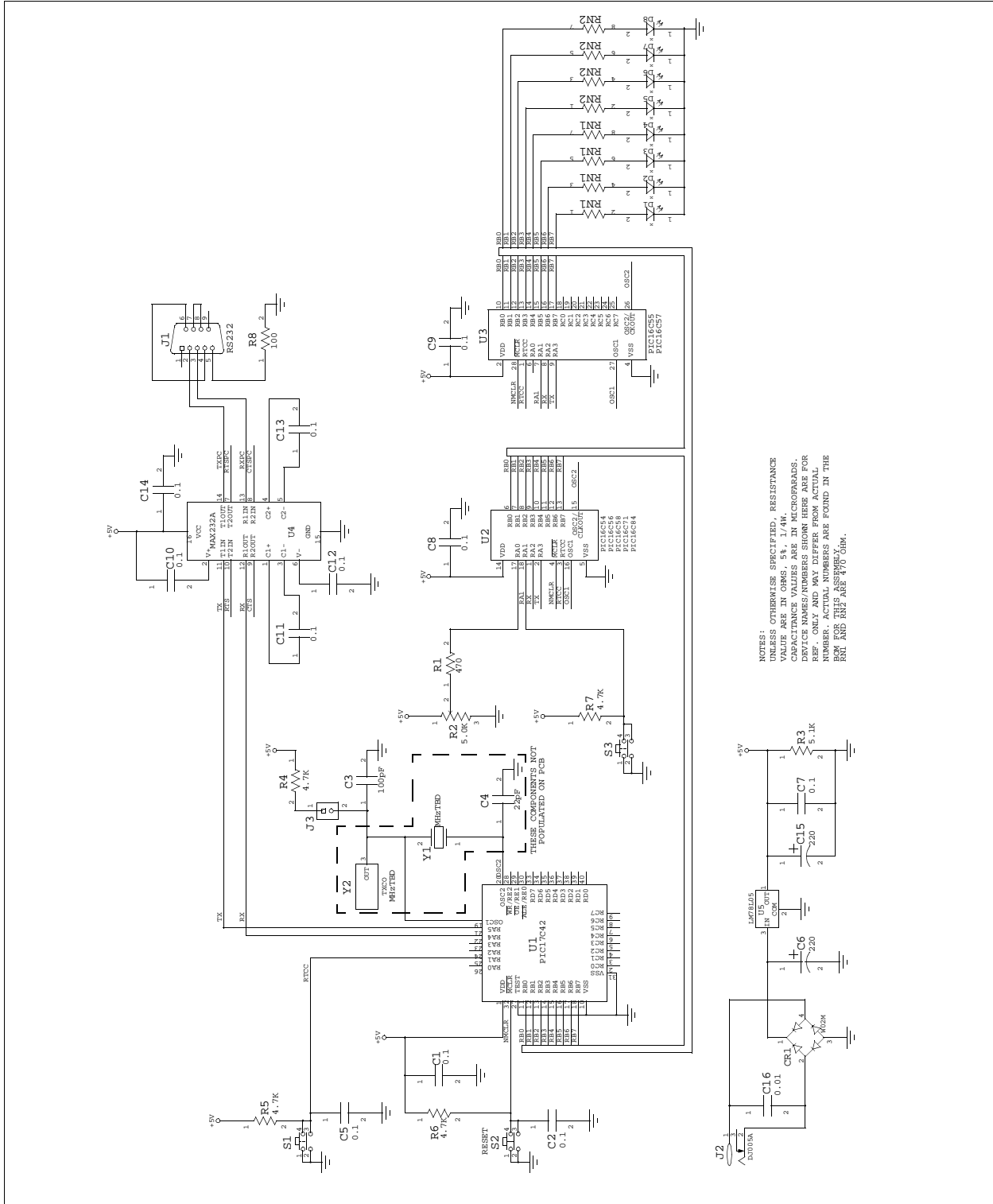
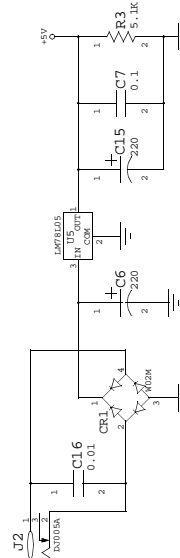


Figure A.2 PICDEM-1 Schematic

NOTES:
 UNLESS OTHERWISE SPECIFIED, RESISTANCE
 VALUE ARE IN OHMS, 5%, 1/4W.
 CAPACITANCE VALUES ARE IN MICROFARADS.
 DEVICE NAMES/NUMBERS SHOWN HERE ARE FOR
 REF. ONLY AND MAY DIFFER FROM ACTUAL
 COMPONENTS. PART NUMBERS ARE FOUND IN THE
 BOM FOR THIS ASSEMBLY.
 R11 AND R12 ARE 470 OHM.





Index

A

A/D Demo 15
A/D Input 2, 15, 46

B

Board 1, 2, 5, 45
 Parts Layout 47
 Power Supply 5, 45
 Schematic 48
 Silkscreen 47

D

Data EEPROM 23
Demonstration Board. *See* Board
Demonstration Programs. *See* Sample Programs

E

EEPROM Demo 23

H

Hardware 45

L

LEDs 2, 11, 15, 23, 45

M

MPASM 4, 7
MPLAB 4, 7, 8
MPLAB-ICE 4, 10

P

PIC16C71, A/D Demo 15
PIC16F84, EEPROM Demo 23
PIC17C42, USART Demo 35
PICDEM-1 Board. *See* Board
PICDEM-1 Kit 1
PICMASTER 4, 8
PICSTART® Plus 4, 7
PRO MATE® II 4, 7
Pushbuttons. *See* Switches

R

RC Oscillator 46
Reference Documents 4
RS-232 2, 35, 45

S

Sample Devices 1, 3, 6, 46
Sample Programs 1, 3, 6
Schematic 48
SCI 35
Source Code
 demo42.asm 42
 demo71.asm 20
 demo84.asm 32
 tut.asm 13
Switches 2, 5, 11, 46

T

Tutorials 11

U

USART Demo 35

W

Windowed Device 6

PICDMEM-1 User's Guide

NOTES:

NOTES:

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

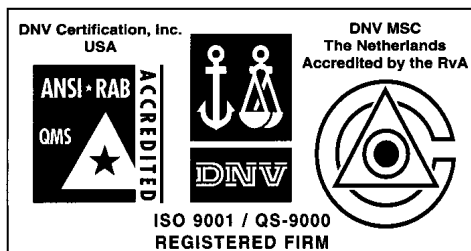
dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

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

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

© 2002, 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.



MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: <http://www.microchip.com>

Rocky Mountain

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta

500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia

Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing

Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu

Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou

Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai

Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen

Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong

Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India

Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan

Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark

Microchip Technology Nordic ApS
Regus Business Centre
Lautrup høj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

03/01/02