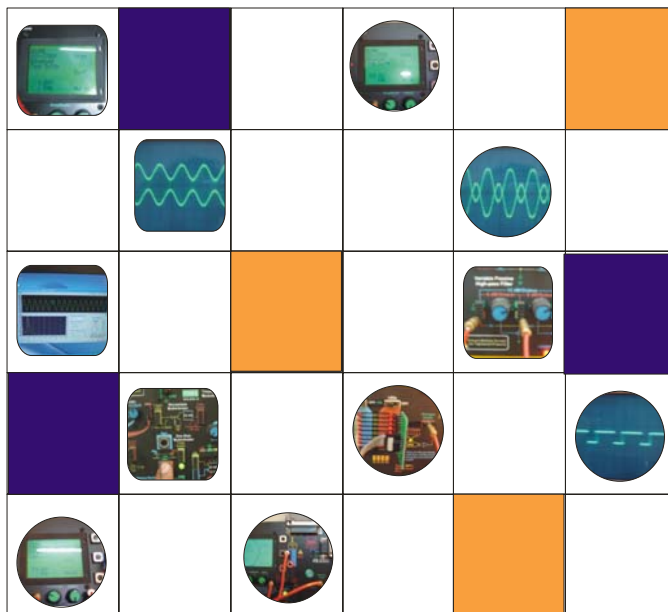


RIMSResearch Instrumentation
& Measurement Systems**DEV-2763**

μ -Controller Based Trainer

EXPERIMENTS

PART NO. 2763-00-321



**COMPREHENSIVE AND ILLUSTRATED
EASY EXPERIMENTS STARTUP
LAB MANUAL**

THANK YOU FOR CHOOSING RIMS EDUCATION PRODUCTS AND SERVICES

Once you have made it through this guide, you will have a firm grip on your lab experiments and operations of the RIMS product you are using. How to get your training equipment operational, basic maintenance and setting up desired experiments will just be a breeze. Everything you need for a quick and easy start is presented here—useful hints and tips makes it simple to conduct your lab and hands-on training sessions. We are happy that you have joined our vast community of over 30 thousand valued users, which grow as we bring you the latest technology at most competitive prices. We value your business and hope that you will enjoy being an important member of the RIMS Education Community.

Customer Support Team



EU, USA and Canada

Weston Villa, 37 Wolsey Road, Esher, Surrey
United Kingdom KT10 8NT
www.rims-tech.co.uk

Middle East & Asia Pacific

632-B Chakala Scheme-III Rawalpindi
Pakistan 46000
www.rimsedu.com

© RIMS 1999-2007. All Rights Reserved. No part of this manual is to be copied, modified or sold in any form without prior permission of RIMS EDUCATION for any further queries please visit our website at <http://www.rims-tech.co.uk>

WARRANTY

The media on which you receive RIMS Technologies software/hardware are warranted for defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. RIMS Technologies will, at its option, repair or replace software/hardware media that do not execute programming instructions if RIMS Technologies receives notice of such defects during the warranty period. RIMS Technologies does not warrant that the operation of the software/hardware shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. RIMS Technologies will pay the shipping costs of returning to the owner parts which are covered by warranty.

RIMS Technologies believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, RIMS Technologies reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult RIMS Technologies if errors are suspected.

In no event shall RIMS Technologies be liable for any damages arising out of or related to this document or the information contained in it. EXCEPT AS SPECIFIED HEREIN, RIMS TECHNOLOGIES MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. RIMS TECHNOLOGIES WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of RIMS Technologies will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against RIMS Technologies must be brought within one year after the cause of action accrues. RIMS Technologies shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the RIMS Technologies installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, act of God, fire, flood, accident, actions of third parties, or other events outside reasonable control.

COPYRIGHT

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of RIMS Technologies.

TRADEMARKS

RIMS™, ThinPoint™, Power to Sense and Control™, INSPTE™, LiveLabs™, RIMS Technologies™, BOX™, RIMS-Scope™, StateView™, rims-tech.co.uk™, RIMS-DAQ™, RIMS Students Zone™, and RIMS-Passport™ are trademarks of RIMS Technologies.

Product and company names mentioned herein are trademarks or trade names of rims technologies.

PATENTS

For patents covering RIMS Technologies products, refer to the RIMS Website www.rims-tech.co.uk.

WARNING REGARDING USE OF RIMS TECHNOLOGIES PRODUCTS

(1) RIMS Technologies products are not designed with components and testing for a level of reliability suitable for use in or in connection with surgical implants or as critical components in any life support systems whose failure to perform can reasonably be expected to cause significant injury to a human and also for industrial or specify critical application.

(2) In any application, including the above, reliability of operation of the software/hardware products can be impaired by adverse factors, including but not limited to fluctuations in electrical power supply, computer hardware malfunctions, computer operating system software/hardware fitness, fitness of compilers and development software/hardware used to develop an application, installation errors, software and hardware compatibility problems, malfunctions or failures of electronic monitoring or control devices, transient failures of electronic systems (hardware and/or software), unanticipated uses or misuses, or errors on the part of the user or applications designer (adverse factors such as these are hereafter collectively termed "system failures"). Any application where a system failure would create a risk of harm to property or persons (including the risk of bodily injury and death) should not be reliant solely upon one form of electronic system due to the risk of system failure. To avoid damage, injury, or death, the user or application designer must take reasonably prudent steps to protect against system failures, including but not limited to back-up or shut down mechanisms. Because each end-user system is customized and differs from rims technologies' testing platforms and because a user or application designer may use rims technologies products in combination with other products in a manner not evaluated or contemplated by rims technologies, the user or application designer is ultimately responsible for verifying and validating the suitability of rims technologies products whenever rims technologies products are incorporated in a system or application, including, without limitation, the appropriate design, process and safety level of such system or application.

(3) All efforts have been done to ensure the correctness of the information or media provided explicitly or implicitly for each training system. However RIMS technologies do not take any responsibility for the losses or otherwise any issues arising from the mistake in the media provided. RIMS would strive to ensure that the mistakes are corrected and communicated to all its customers.

Welcome to RIMS μ -Controller Based Trainer

- Around The 89C5X μ -Microcontroller
- Writing On The LCD
- Bit And Bit Addressing The Port
- Serial Communication With The PC
- Generating A Square Wave Using A Microcontroller
- Measuring The Waveform Of A Frequency
- INTERFACING EXTERNAL HARDWARE TO THE MICROCONTROLLER

Product Title: EXPERIMENTS

Document Code: DEV2763-00-321

Revision 2.0.0 dated January 2007

© RIMS 1999-2007. All Rights Reserved. No part of this manual is to be copied, modified or sold in any form without prior permission of RIMS EDUCATION for any further queries please visit our website at <http://www.rims-tech.co.uk>

STEP 1**AROUND THE 89C5X
MICROCONTROLLER**

The 89C5x is a 40 pin IC. This section contains details of the pins significant for operations. Quite broadly, the 89C5x has four 8 bit ports: port 0, 1, 2 and 3. Port 0 is a dual purpose port on pins 32-39. It is used as a general purpose I/O port. For larger designs with external memory it becomes a multiplexed address and data bus. Port 1 is a dedicated I/O port on pins 1-8. Port 2 is dual purpose port serving both as a general purpose I/O port and as the high byte of the address bus for designs with external code memory or more than 256 bytes of external data memory. The port 2 is connected to the LCD. Port 3 is a dual purpose port on pins 10-17. These pins are multifunctional with each having an alternate purpose related to special features of the 89C5x. These ports are either byte addressable or bit addressable, i.e. the port can be accessed as a byte as shown below:

```
P0 = 0xff; // set all the pins of the port 0 as high.  
P1 = 0x0f; // sets the MSB of the Port 1 as high  
P2 = 0xf0; //sets the LSB of the Port 2 as high
```

Or, a specific pin can be addressed individually as shown:
sbit pin = P0^1; // Declaration of Pin 2 of Port 0 with variable name 'pin' as sbit.

```
.  
. .  
pin = 1; // Set 'pin' as high  
pin = 0; // Set 'pin' as low
```

Similarly the 89C5x has RAM spaces which are bit addressable or byte addressable. It has special function registers for working with timers, serial port and interrupts, which are both byte addressable and bit addressable. The use of these registers would become clear in the following labs. The language we would be using to program the microcontroller is C51.

STEP 2 WRITING ON THE LCD

Objective

To familiarize the student with microcontroller programming, uploading and running.

Apparatus

The Microcontroller based trainer, the Microcontroller programmer, the 89C5x microcontroller, a computer with KEIL IDE installed.

Procedure

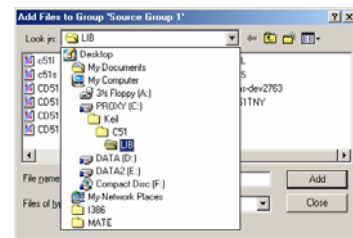
Open a new project in KEIL IDE and name it Lab_1. After selecting the appropriate chip and initializing the settings, right click on the source folder and click on Add files to project. Go to the lib folder as shown in figure. Select the following library file by first selecting file of type as .lib files
Add this file to your project. Now create a new source code file enter the following code in it, save it with the name Lab_1_code.c and add to your Source Group

```
#include <lcd4bitx51.h> // To include library functions for LCD
```

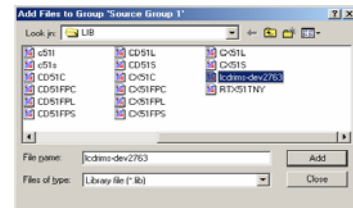
```
void main()
{
  int i,j;
```

```
  while(1) // Effectively an embedded program never ends
  {
    InitLCD_rimsDEV2763(); // Initialize the LCD Display
    for(i=0;i<1000;i++) // Cause a delay of 1 seconds
      approximately
        for(j=0;j<100;j++);
    PrintLCD("Hello World"); // Display on the LCD Display
    Hello world
    for(i=0;i<1000;i++)
      for(j=0;j<100;j++);
  } }
```

Compile the program and then upload it into the microcontroller. After loading the program in the microcontroller place it in the Ziff socket of the Microcontroller based trainer and close the knob. You



Adding Files



Assigning the Library for LCD

should see the text “Hello World” written on the LCD screen. Now modify the above program as under:

```
#include <lcd4bitx51.h>
void main()
{
int i,j,k;

while(1)
{
    InitLCD_rimsDEV2763();
    for(i=0;i<200;i++)
        for(j=0;j<100;j++);
    LocateLCD(1,k); // Place the cursor at the location indicated
    PrintLCD("Hello World");
    for(i=0;i<200;i++)
        for(j=0;j<100;j++);

k++; // Increment k so the text scrolls on the display
    if(k > 16) // The LCD can display up to 16 characters
        k = 1;
}
}
```

Once loaded this program would display a message “Hello World” on the LCD scrolling from left to right. The program is self explanatory for an experienced C/C+ programmer. The InitLCD_rims DEV2763 () automatically initializes the LCD and clears the display of any previous text. The ‘for’ loop next inserts a short delay. Locate LCD (1, k) brings the current cursor to position 1, k. Where 1 is the row number and k is the column number (the LCD has 2 rows and 16 columns). There for loop again inserts a short delay. k++ increments the column number and the next if statement checks to see if the column number has exceeded 16 if so it reinitializes it back to 1. The same program runs over and over again because of the while loop.

Lab Discussion

In this lab, you learned how to program the microcontroller to display text the LCD. You can consider the LCD of the analogous to your monitor; the only difference is that the LCD only has 2 rows and 16 columns as compared to the monitor which has 60 rows and 80 columns for text. Secondly, the LCD can only display text messages.

STEP 3**BIT AND BYTE ADDRESSING THE PORTS****Objective**

To introduce the concepts of bit and byte address ability of the ports.

Apparatus

Same as in lab 1, circuit patching wire, stripper

Procedure

Cut 8 pieces of 6" long wire and strip off the ends of the wire 1.5cms approximately. Place one end of each wire in the connectors of the microcontroller from 32 to 39. Connect the other end of the wires to the LEDs on the microcontroller based trainer. Open a new project in KEIL IDE and name it Lab_2. After initialization enters the following program, save it as Lab_2_code.c and add it to your source group. Compile it and then load it.

```
#include <reg51.h> // Necessary for including 89C5x functions

void main()
{
  int i;
  P0 = 0x00; // Initialize the port 0

  while(1)
  {
    P0 += 1; // Increment the value of port 0 by 1
    for(i=0;i<1000;i++); // a short delay
    if(P0 == 0xff) //Check to see if Port 0 is equal to 0xff
      P0 = 0x00;
  }
}
```

This program addresses the port 0 and increments its value by using byte addressing. The port is initialized to 0x00 and subsequently incremented by 1 till it reaches 0xff where it is again reinitialized to 0x00. The pattern can also be seen on the LED panel on the Microcontroller based trainer, when it is switched on with the IC in the ZIF Socket. Various LEDs will also come on as the port value is incremented from 00000000 (all LEDs Off) to 11111111 (all LEDs On).

In the same way, port 1, port 2 and port 3 can be byte addressed as P1, P2 and P3 respectively.

Remove all the circuit patching wires, except for the wire in pin number 39 and 38. Connect the other end of the wires to any two logic switches. Open KEIL IDE once again open a new project as Lab_2_1. Do the settings as you did in lab 1. and also add the lcdrim-dev2763.lib file as explained in lab 1. Now enter the following program save, it as Lab_2_1_code.c and add it to your source group. Compile it and load it.

```
#include <lcd4bitx51.h> // Library file needed to include LCD functions
sbit pin1 = P0^0; // Declare pin 1 of port 0 as sbit pin1
sbit pin2 = P0^1; // Declare pin 2 of port 0 as sbit pin2
void main() {
int i,j;
P0 = 0x00;
while(1)
{
InitLCD_rimsDEV2763();
if(pin1) // Check to see if pin1 is High
{
LocateLCD(1,1);
PrintLCD("pin1 is high");}
else
{
LocateLCD(1,1);
PrintLCD("pin1 is low");}
if(pin2)
{
LocateLCD(2,1);
PrintLCD("pin2 is high");}
else
{
LocateLCD(2,1);
PrintLCD("pin2 is low");}
for(i=0;i<500;i++)
for(j=0;j<100;j++)
}
}
```

This is another example of addressing the pins but this time it is bit addressing. This program checks to see if pin 1 and pin 2 of port 0 named as P0^0 and P0^1 respectively are high or low and displays the result on the LCD. You can toggle the switches and see what happens. In the same way ports 1, 2 and 3 can be bit addressed by using P1^x, P2^x, P3^x respectively.

Lab Discussion

In this lab, you did bit and byte addressing of the ports. In addition you also verified the concepts of input and output on the ports. In this way ports can be used to drive an LED on or check to see if a certain input is being applied on the port pins.

STEP 4**SERIAL COMMUNICATION WITH THE PC****Objective**

To introduce the concepts of serial communication for the 89C5x

Apparatus

Serial cable, PC with serial COM port and Hyper Terminal Software installed, RIMS Micro-controller trainer

Procedure

The 89C5x has a serial port that has four modes of operations. We will use the mode, which is referred to as 8-bit auto reload mode with variable baud rate. The mode of the serial port is set with the Special function register, SCON. The baud rate of the serial port is set by timer 1 and is set by special function register, TMOD. Connect one end of the serial cable to the COM port on the PC and the other end to the serial connector of the microcontroller based trainer. Open a new project in KEIL IDE and name it Lab_3. After initializing the settings, open a new text file; enter the code given below and save it as Lab_3_code.c

```
#include <reg51.h>
#include <studio.h> // Included for printf() function
void main()
{
    int i,j;
    SCON = 0x52; // Initialize the serial port in 8-bit auto reload mode
    TMOD = 0x20; // Initialize the timer
    TH1 = -24; // Reload value of the timer for baud rate 1200 bps
    TR1 = 1; // Start the timer
    TI=1; // Essential settings
    RI=0; // Essential settings

    while(1)
    {
        printf("RIMS DEV 2763"); // Sends RIMS DEV2763 on the serial port

        for(i=0;i<700;i++)
            for(j=0;j<100;j++); // Inserts a short delay
    }
}
```

In the PC open the Hyper Terminal software. Set the baud rate to 1200 bps. The text "RIMS DEV2763" should start appearing on the screen.

Lab Discussion

This lab shows how to establish serial communication over a serial cable from a microcontroller to a PC. The same link can be established between one microcontroller and another with one sending and the other receiving. The serial port provided with the microcontroller is a full duplex communication link with variable baud rate. The baud rate is selected by the statement `TH1 = -24`. This value establishes a baud rate of 1200 bps. A value of `-12` gives a baud rate of 2400 bps, `-6` gives 4800 bps and `-3` gives 9600 bps. In this lab the `printf ()` function was used for transmitting data from the microcontroller to the PC. In C/C++ `printf()` sends the string passed to the function to the standard input/output which, in case of the microcontroller, is the serial port. You might be thinking that you can use the `scanf ()` function for receiving data from the serial port. Yes, you can. However, both `printf ()` and `scanf ()` impose very heavy burden on the program size. A better method for transmitting and receiving data is given in the next lab.

STEP 5**GENERATING A SQUARE WAVE
USING A MICROCONTROLLER****Objective**

To teach the student how to generate square waves of different frequencies using the built-in timers of the 89C5X series microcontrollers

Apparatus

Microcontroller Trainer, MultiMeter.

Procedure

In this lab you will generate a square wave on pin 1 of port 0 and read its frequency on a multimeter. Open KEIL IDE and enter the following program in a new project and file:

```
#include <reg52.h>
sbit i0=P1^0;          //defines the single bit variable
sbit FreqOut=P0^0;    //defines the single bit variable
void main (void) {
    P1=0xff; //initialize Port 1 for input
    P0=0x00; //initialize Port 0 for output
    while(1){
        /* 10 KHz square wave generation */
        while (!i0){          // loop if Port 1 pin0 is low
            TMOD=0x02;      // set the timer0 to 8 bit auto reload MODE
            TH0=-50;        // loads the register TH0 for 50uSec delay
            TR0=1;          // starts the timer
            while(!TF0);    // on timer overflow TF0=1
            TF0=0;          // resets the timer flag
            FreqOut=!FreqOut; // complements the Port0 pin1          /*
        }
        /* 1 KHz square wave generation */
        while (i0){          // if Port 1 pin0 changes to high
            TMOD=0x01;      // set the timer0 to 16 bit timer MODE
            TH0=0xFE;        // loads the register TH0 with FE
            TL0=0x0C;        // loads the register TL0 with 0C
            TR0=1;          // starts the timer
            while(!TF0);    // checks the timer overflow flag bit
            TR0=0;          // stops the timer
            TF0=0;          // resets the timer flag
            FreqOut=!FreqOut; // complements the Port0 pin1
        }
    }
}

void Delay1(void)        //function for delay of 20000uSec
int l,j,k;
    for (i=0;i<1;i++){
        for (j=0;j<100;j++){
            for (k=0;k<100;k++){
                }
            }
        }
    }
```

The comments effectively explain what the program does in this case.

Lab Discussion

As mentioned earlier, the 89C5x microcontroller has two timers (89C52 has 3). These timers are 16 bit and can be used to time various events, perform event counts, or cause delays etc. The timers can also be used in 13 bit and 8 bit modes as specified by the special function register TMOD of the microcontroller. `TMOD = 0x02;` sets the microcontroller's timer 0 in 16 bit mode. Then the THx and TLx (TH0 for timer 0 and TH1 for timer 1) bytes set the timer reload value. The timers initially start from the value specified by THx and TLx and go to ffff. After ffff the timer overflows and the timer flag TFX is set (TF0 for timer 0 and TF1 for timer 1). In this lab you generated a square wave using the 8-bit auto reload mode and the 16-bit mode.

STEP 6**MEASURING THE FREQUENCY OF A WAVEFORM****Objective**

To measure the frequency of a waveform, using a microcontroller and displaying it on the LCD.

Apparatus

Microcontroller based trainer, Circuit patching wire, MultiMeter.

Procedure

From the previous Lab you learned that the microcontroller's timer can be used to generate delays, time events etc. Another very useful application of the microcontroller's timers is event counting. An event, quite broadly, is a transition of state. The technique employed to measure the frequency is to, use one of the timers as an event counter. The other timer simply provides the baud rate for the serial port operation. The measured frequency is calibrated, displayed on the LCD and sent across the serial port. A GUI made in Lab Windows/CVI running on the PC end, which captures the value of the frequency on the serial port and displays it in various formats. Open a new project in the KEIL IDE, and name it as Lab_6. Add the settings for using the LCD. Open a new text file, enter the following code and save it as Lab_6_code.c

```
#include <stdio.h>
#include <lcd4bitx51.h>

sbit frequencyPin=P3^2; // The pin to which the frequency is applied
sbit low_freq=P0^0; // If this pin is high it indicates low frequency
sbit high_freq=P0^1; // If this pin is high it indicates high frequency
unsigned char k,l;
float count;

unsigned char buffer[20];
void main()
{

    TMOD = 0x25; /* Initialize both timers timer 0 as event counter
timer 1 for baud rate generation */
```

```

SCON = 0x52; // Select serial port mode
TH1=-3; // Serial port baud rate 9600 bps
TR1=1; // Start timer 1
TI=1;

P0=0;
InitLCD_rimsDEV2763(); // Initialize LCD

LocateLCD(2,1);
PrintLCD("Demo RIMS");

while(1)
{
    TH0=0; // Load timer zero high byte with zero
    TL0=0; // Load timer zero low byte with zero
    TR1=0; // Ensure timer zero is off
    TR0=1; // Start timer zero for event counting
    for(k=0;k<2;k++) // Insert a short delay
        for(l=0;l<200;l++);
    TR0=0;
    /* Stop timer zero, The number of events counted are stored in TH0 and
    TL0 */
    count =(((float)TH0*256.0+ (float)TL0)*0.9505971)* 300.0 /1000.0;
    /* The above statement is a calibration statement for frequency*/
    if (count<1.0)
    // Settings to measure high and low frequency
        {low_freq=1;high_freq=0;}
    else if (count>100.0)
        {low_freq=0;high_freq=1;}
    else
        {low_freq=0;high_freq=0;}
    sprintf(buffer,"f = %7.4fKHz",count);
    /* Make a string buffer for LCD and serial port*/
    LocateLCD(1,1);
    PrintLCD(buffer);

    TR1=1;
    TI=1;
    for(k=0;k<15;k++)
    // Block for transmitting the buffer to port
    {
        while(!TI);
        TI=0;
        SBUF = buffer[k];
    }
}

```

After saving, compiling and loading this program, cut off 3 5" wires and strip off the ends. Connect the pin number 39, 38 of the microcontroller to two LEDs. Connect one end of another piece of wire to clock generator connector and the other end to pin 12 of the

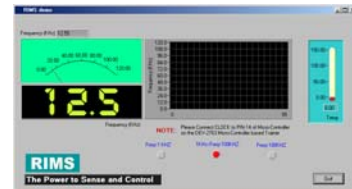
microcontroller. Connect the serial port connector of the trainer to the serial port of PC using the given cable. Go to the folder KEIL C51 EXAMPLES FREQUENCY CVI. Click on the application VMS.

The following application should appear on the screen,

Now insert the programmed microcontroller into the trainer. Switch it on, and press the Reset button. Adjust the frequency from the Frequency knob. You should see a changing frequency on the GUI. Also you would see the value of the generated frequency on the LCD.

Lab Discussion

In this lab we discussed the concepts of measuring frequency, and using a timer as an event counter.



Frequency Meter in Windows Environment

STEP 7**INTERFACING EXTERNAL HARDWARE TO THE MICROCONTROLLER****Objective**

To teach the student how to interface an external device, specifically an analog to digital converter to the microcontroller

Apparatus

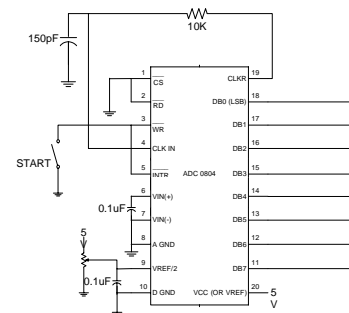
Microcontroller based trainer, Programmer, ADC0804 A-D converter.

Procedure

To be able to interface any kind of hardware, you will have to clearly understand its datasheet, as most of the required information is given in it. For this case as well, do a thorough study of the datasheet of the ADC0804, understand as much as you can as every thing cannot be told in this manual. The A-D converter converts, analog signals to 8-bit digital format. Before interfacing an external IC you have to make sure that it is of TTL type, so that it matches the microcontroller inputs and outputs. If it is of CMOS type then you would have to use a buffer, (a TTL to CMOS converter) the CD4050. The ADC-0804 can drive TTL loads, so you don't need a buffer. Patch a circuit as shown below and connect the connectors from DB1 to DB8 to the connectors of port 0 of the microcontroller. Take a circuit patching wire and connect the Vin(+) 6pin to **Waveform** connector on the Trainer.

Open a new project in KEIL IDE name it Lab_7. Add the settings to run the LCD. Enter the following code,

```
#include <lcd4bitx51.h>
#include <stdio.h> // Required for the function sprintf
void main()
{
    unsigned char buffer[17], readValue=0,i;
    float voltage;
    P0 = 0xff; // Initialize port 0 to read external values
    P0 = 0x00;
```



```
InitLCD_rimsDEV2763();
PrintLCD("Value read :");
while(1) {
for(i=0;i<100;i++) // Read 100 values and average them
    readValue += P0;

readValue = readValue / 100;

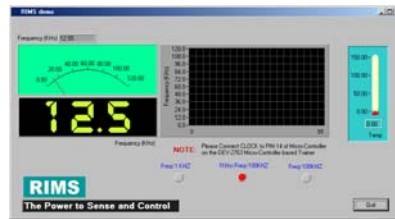
// Convert unsigned char value to 5 Volt floating voltage value
voltage =( readValue*5) / 256;

sprintf(buffer,"%3.2f Volts\r0");
LocateLCD(2,1);
PrintLCD(buffer);
}
}
```

In the above example, one thing to keep in mind is that the A to D converter is running in self-clocking and free running mode. It automatically converts any analog signal applied to its input into 8-bit digital code at the output. So you read 100 values and took their average, and then converted it back into the voltage format and display it on the LCD.

Lab Discussion

This lab served as an introduction to interfacing external hardware to the microcontroller. As discussed earlier, before interfacing any external device to the microcontroller, the datasheet of the particular device should be first consulted. Besides the A/D, a host of devices can be interfaced to the microcontroller.



RIMS

EDUCATION

EU, USA and Canada

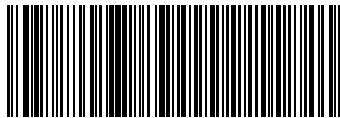
Weston Villa, 37 Wolsey Road, Esher, Surrey
United Kingdom KT10 8NT

www.rims-tech.co.uk

Middle East & Asia Pacific

632-B Chakala Scheme-III Rawalpindi
Pakistan 46000

www.rimsedu.com



DEV2763-00-321

Product Title: Experiment Work Book

Document Code: DEV2763-00-321

Revision 2.0.1 dated January 2007

© RIMS 1999-2007. All Rights Reserved. No part of this manual is to be copied, modified or sold in any form without prior permission of RIMS EDUCATION

For any further queries please visit our website at

<http://www.rims-tech.co.uk>