ARM HOW-TO GUIDE

Interfacing I2C-EEPROM with LPC2148
Contents at a Glance

ARM7 LPC2148 Primer Board ................................................. 3
I2C (Inter Integrated Circuit) .................................................. 3
EEPROM .................................................................................. 4
Interfacing I2C - EEPROM ..................................................... 4
Interfacing I2C – EEPROM with LPC2148 ......................... 6
Pin Assignment with LPC2148 ............................................... 6
Circuit Diagram to Interface I2C–EEPROM with LPC2148 ..7
Source Code ............................................................................ 7
C Program to I2C – EEPROM Display using LPC2148 ....... 8
Testing the I2C – EEPROM with LPC2148 ......................... 16
General Information ............................................................. 18

Join the Technical Community Today!
http://www.pantechsolutions.net
ARM7 LPC2148 Primer Board

The ARM7 LPC2148 Primer board is specifically designed to help students to master the required skills in the area of embedded systems. The kit is designed in such a way that all the possible features of the microcontroller will be easily used by the students. The kit supports in-system programming (ISP) which is done through a serial port.

NXP’s ARM7 (LPC2148), ARM Primer Kit is proposed to smooth the progress of developing and debugging of various designs encompassing of High speed 32-bit Microcontrollers.

I2C (Inter Integrated Circuit)

The I2C (Inter-IC) bus is a bi-directional two-wire serial bus that provides a communication link between integrated circuits (ICs). I2C is a synchronous protocol that allows a master device to initiate communication with a slave device. Data is exchanged between these devices.
**EEPROM**

EEPROM (electrically erasable programmable read-only memory) is user-modifiable read-only memory (ROM) that can be erased and reprogrammed (written to) repeatedly through the application of higher than normal electrical voltage. It is a type of non-volatile memory used in computers and other electronic devices to store small amounts of data that must be saved when power is removed, e.g., calibration tables or device configuration.

**Interfacing I2C - EEPROM**

Fig. 1 shows how to interface the EEPROM with microcontroller through I2C. I2C is a Master-Slave protocol. I2C has a clock pulse along with the data. Normally, the master device controls the clock line, SCL. This line dictates the timing of all transfers on the I2C bus. No data will be transferred unless the clock is manipulated. All slaves are controlled by the same clock, SCL.
I2c bus supports many devices, each device is recognized by a unique address—whether it’s a microcontroller, LCD Driver, memory or keyboard interface and can operate as transmitter or receiver based on the functioning of the device. The controller designed controls the EEPROM device through I2C protocol. The I2C Controller here acts as a master device and controls EEPROM which acts as a slave. The read-write operations are accomplished by sending a set of control signals including the address and/or data bits. The control signals must be accompanied with proper clock signals.

Fig. 1 Interfacing I2C - EEPROM to Microcontroller
Interfacing I2C – EEPROM with LPC2148

We now want to Read, write and Erase EEPROM by using I2C in LPC2148 Primer Board. Wiring up an I2C based EEPROM to the I2C port is relatively simple. The basic operation of the I2C based EEPROM's is to send a command, such as WRITE, followed by an address and the data. In WRITE operation, the EEPROM to store the data.

In **LPC2148 Primer Kit**, 2 nos. of EEPROM lines are controlled by I2C Enabled drivers. I2C Lines serial clock **SCL (P0.2)**, serial data **SDA (P0.3)** connected to the I2C based serial EEPROM IC. The EEPROM read & write operations are done in LPC2148 Primer Kit by using these SDA & SCL I2C lines.

**Pin Assignment with LPC2148**

<table>
<thead>
<tr>
<th>AT 24xx</th>
<th>I2C EEPROM</th>
<th>LPC2148 Lines</th>
<th>Serial EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCL</td>
<td>SCL</td>
<td>SCL1 - (P0.11)</td>
<td></td>
</tr>
<tr>
<td>SDA</td>
<td>SDA</td>
<td>SDA1 - (P0.14)</td>
<td></td>
</tr>
</tbody>
</table>
Circuit Diagram to Interface I2C–EEPROM with LPC2148

Source Code

The Interfacing I2C – EEPROM with LPC2148 program is very simple and straightforward that read, write and erase operations in EEPROM by using I2C & the value is displayed in serial port. A delay is occurring in every single data read or write in EEPROM. The delay depends on compiler how it optimizes the loops as soon as you make changes in the options the delay changes.
C Program to I2C – EEPROM Display using LPC2148

Title : Program to read, write & erase of I2C - EEPROM

```
#include <LPC214x.h>
#include <stdio.h>
#include <string.h>
#include "UART.h"

#define SW3   1<<24
#define SW4   1<<25
#define SW5   1<<26
#define SW6   1<<27
#define SW7   1<<28

#define MAX   12
#define AA    2
#define SI    3
#define STO   4
#define STA   5
#define I2EN  6

void I2C_ISR(void)__irq;
void Wait (unsigned int);
void I2C_Init (void);
int I2C_Start (unsigned int Slave_Addr);
int I2C_Write (unsigned char *Buff, unsigned int Count);

char ReLoad[MAX] = {0x00,0x00,'A','R','M','7','P','R','I',
                     'M','E','R'};
char Buff[MAX] = {0x00,0x00,'A','R','M','7','P','R','I',
                  'M','E','R'};
unsigned char Rec[MAX] = {"NO-DATA!"};
unsigned char index = 0;
unsigned char flag = 0, ii, Ready=0, Erase = 0;
```
void I2C_ISR(void)__irq
{
    if (I2C1CONSET & 0x08)
    {
        switch (I2C1STAT)
        {
        case (0x08):
            if (flag == 'W')
            {
                I2C1CONCLR = 1 << STO;
                I2C1CONCLR = 1 << STA
                I2C1DAT = 0xA0;
                I2C1CONCLR = 1 << SI;
            }
            else if (flag == 'R')
            {
                I2C1DAT = 0xA0;
                I2C_Start (0xA1);
                I2C1CONCLR = 1 << SI;
            }
            index = 0;
            break;
        case (0x10):
            I2C1CONCLR = 1 << STA;
            if (flag == 'W')
            {
                I2C1DAT = 0xA0;
            }
            else if (flag == 'R')
            {
                I2C1CONCLR = 1 << STA;
                I2C1CONCLR = 1 << STO;
                I2C1DAT = 0xA1;
                I2C1CONCLR = 1 << SI;
                index = 0;
            }
            break;
        case (0x18):
            I2C1CONCLR = 0x20;
            I2C1CONCLR = 0x38;
            break;
        }

    }
}

Join the Technical Community Today!
http://www.pantechsolutions.net
I2C1DAT = Buff[index];
index++;
break;
case (0x20):
    I2C1CONCLR = 0x20;
    I2C1CONCLR = 0x38;
    I2C1DAT = Buff[index];
    index++;
    break;
case (0x28):
    if (index < MAX)
    {
        if (flag == 'W')
        {
            I2C1DAT = Buff[index];
            I2C1CONCLR = 0x20;
            I2C1CONCLR = 0x38;
        }
        else if ((index > 1) && flag == 'R')
        {
            I2C1CONCLR = 0x18;
            I2C1CONSET = 1 << STA;
        }
        else
        {
            I2C1DAT = Buff[index];
            I2C1CONCLR = 0x20;
            I2C1CONCLR = 0x38;
        }
        index++;
    }
    else
    {
        index = 0;
        flag = 'R';
        I2C1CONSET = 1 << STO;
        if (Erase == 1)
        {
            UART0_PutS("\n\r Memory Erase successful..! \n");
            Erase = 0;
        }
else {
    UART0_PutS ("\n\r Data Successfully Written on Memory!\n");
}
I2C1CONCLR = 1 << STA;
I2C1CONCLR = 1 << SI;
}
break;
case (0x30):  
    I2C1CONCLR = 0x20;
    if (index < MAX) {
        if (flag == 'W') {
            I2C1DAT = Buff[index];
        }
        index++;
    } else {
        index = 0;
        flag = 0;
        I2C1CONSET = 0x10;
        I2C1CONCLR = 1 << SI;
    }
    break;
case (0x38):  
    I2C1CONSET = 0x20;
    break;
case (0x40):  
    I2C1CONSET = 1 << AA;
    I2C1CONCLR = 1 << STA;
    I2C1CONCLR = 1 << STO;
    I2C1CONCLR = 1 << SI;
    break;
case (0x48):  
    I2C1CONSET = 1 << STA;
    break;
case (0x50):

    I2C1CONSET  =  1 << AA;
if (index < MAX)
{
    Rec [index]  =  I2C1DAT;
    index++;
}
else
{
    I2C1CONSET  =  1 << STO;
    I2C1CONCLR  =  1 << SI;
    index = 0;
    Ready  =  'T';
}
break;
case (0x58):
    Rec [index]  =  I2C1DAT;
    I2C1CONCLR  =  1 << STA;
    I2C1CONCLR  =  1 << STO;
break;
}
I2C1CONCLR  =  1 << SI;
VICVectAddr  =  0x00;
}

int main()
{
    unsigned int i;
    VPBDIV  =  0x02;
    PINSEL0  =  0x30C00005;  //P0.11-SCLK1 & P0.14-SDA1
    PINSEL2  &=  0xFFFFFFFF3;
    IO1DIR  =  0x00 << SW3;
    IO1DIR  |=  0xFF << 16;
    UART0_Init (9600);
    VICIntSelect = 0<<19;
    VICVectCntl0 = 0x020 | 19 ;
    VICVectAddr0 = (unsigned long)I2C_ISR;
    VICIntEnable = 1<<19;
}
I2C_Init();
UART0_PutS ("********** ARM Primer LPC-2148 I2C EEPROM Demo **********\n\n\r"); UART0_PutS (">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n\n\r"); UART0_PutS ("[~] Turn SW 20 ON to Write default data to EEPROM! \n\r"); UART0_PutS ("[~] Turn SW 21 ON to Read and Display data from EEPROM! \n\r"); UART0_PutS ("[~] Turn SW 22 ON to Erase data from EEPROM \n\r"); IO1CLR = 0xFF << 16; while (1) {
    if(((IOPIN1 & SW3) == 0)
    {
        ii = 0;
        IOCLR1 = 0xFF << 16;
        IOSET1 = 1 << 16;
        while (ii < MAX)
        {
            Buff [ii] = ReLoad [ii];
            ii++;
        }
        Wait (5000);
        flag = 'W';
        I2C_Start (0x70);
        for (i=0;i<30;i++) Wait(10000);
        I2C1CONCLR = 1 << SI;
        while (!((IOPIN1 & SW3)));
        Wait (5000);Wait (5000);
    } else if ((IOPIN1 & SW4) == 0
    {
        IOCLR1 = 0xFF << 16;
        IOSET1 = 1 << 17;
        flag = 'R';
        I2C_Start (0x70);
        for (i=0;i<30;i++) Wait(10000);
I2C1CONCLR = 1 << SI;
while (Ready == 'F');

if (Ready == 'T')
{
    ii=0;
    UART0_PutS ("\n\r The Read Data are: \t");
    while (ii<MAX)
    {
        //U0THR = '
';
        Wait (1000);
        U0THR = Rec[ii];
        Wait (1000);
        ii++;
    }
    UART0_PutC ('\n');
    Wait (1000);
    Ready = 'F';
}
while (!(IOPIN1 & SW4));
Wait (5000);Wait (5000);Wait (5000);Wait (5000);
if ((IOPIN1 & SW5) == 0)
{
    IOCLR1 = 0xFF << 16;
    IOSET1 = 1 << 18;
    ii = 2;
    Erase = 1;
    while (ii < MAX)
    {
        Buff[ii] = 0xFF;
        ii++;
    }
    flag = 'W';
    I2C_Start (0x70);
    for (i=0;i<30;i++) Wait(10000);
    I2C1CONCLR = 1 << SI;
    while (!(IOPIN1 & SW5));
    Wait (5000);Wait (5000);Wait (5000);Wait (5000);
}

Join the Technical Community Today!
http://www.pantechsolutions.net
void Delay(void) {
    unsigned int i, j;
    for (i=0; i<150; i++)
        for (j=0; j<900; j++);}

void Wait (unsigned int Delay) {
    while (Delay--);
}

void I2C_Init (void) {
    I2C1SCLH = 150; //50%duty, I2CFreq->100KHz, PCLK=30MHz
    I2C1SCLL = 150;
    I2C1CONSET = 1 << I2EN;   //Enable I2C 1
}

int I2C_Start (unsigned int Slave_Addr) {
    I2C1CONCLR = 1 << STO;
    I2C1CONSET = 1 << AA;
    I2C1CONSET = 1 << STA;
    return 0;
}

int I2C_Write (unsigned char *Buff, unsigned int Count) {
    while (Count--)
    {
        I2C1DAT = *Buff++;
    }
    return 0;
}
To compile the above C code you need the KEIL software. They must be properly set up and a project with correct settings must be created in order to compile the code. To compile the above code, the C file must be added to the project.

In Keil, you want to develop or debug the project without any hardware setup. You must compile the code for generating HEX file. In debugging Mode, you want to check the port output without LPC2148 Primer Board.

The Flash Magic software is used to download the hex file into your microcontroller IC LPC2148 through UART0.

**Testing the I2C – EEPROM with LPC2148**

Give +3.3V power supply to LPC2148 Primer Board; the EEPROM device is connected with the LPC2148 Primer Board. First check the entire EEPROM device fixed properly. A serial cable is connected between the microcontroller and PC.

Join the Technical Community Today!
http://www.pantechsolutions.net
In PC, open the Hyper Terminal for displaying the values from EEPROM through I2C.

- If you want to write any data into EEPROM then you just pressed the switch, sw20. Now, you can write any values into the EEPROM permanently.

- If you want to read data from EEPROM then you just pressed the switch, sw21. Now you can read any data from EEPROM. If any data is written by you in EEPROM particular address then you just read that data from that particular address by using that switch.

- If you want to erase any data from EEPROM then you just pressed the switch, sw22. Now you can erase any data from EEPROM as you wish.

- All the above operations are performed in EEPROM with EEPROM address. When the EEPROM address is correct, then only you can write, read, and erase data’s correctly in EEPROM.

Join the Technical Community Today!
http://www.pantechsolutions.net
If any data is not coming in Hyper Terminal, then you just check the serial cable is working or not. Otherwise you just check the code with debugging mode in Keil. If you want to see more details about debugging just see the videos in below link.

➢ How to Create & Debug a Project in Keil.

General Information

• For proper working use the components of exact values as shown in Circuit file. Wherever possible use new components.

• Solder everything in a clean way. A major problem arises due to improper soldering, solder jumps and loose joints. Use the exact value crystal shown in schematic.

• More instructions are available in following articles,

➢ User Manual of LPC2148 Primer Board.
➢ Tutorial of how to create & Debug a project in Keil.
Did you enjoy the read?

Pantech solutions creates information packed technical documents like this one every month. And our website is a rich and trusted resource used by a vibrant online community of more than 1,00,000 members from organization of all shapes and sizes.
What do we sell?

Our products range from Various Microcontroller development boards, DSP Boards, FPGA/CPLD boards, Communication Kits, Power electronics, Basic electronics, Robotics, Sensors, Electronic components and much more. Our goal is to make finding the parts and information you need easier and affordable so you can create awesome projects and training from Basic to Cutting edge technology.

Join the Technical Community Today!
http://www.pantechsolutions.net