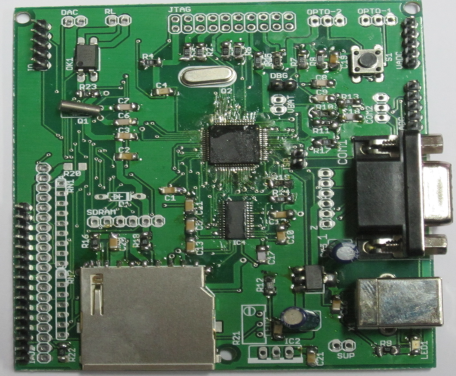


**Emtron Technologies   
Flat No-101, B3 Wing, 1st Floor, Divyam Hights, Gilbert Hill, Shreenath Nagar, Andheri –West, Mumbai-58   
+91-8080181911 E-mail:** [**emtron.tech@gmail.com**](mailto:emtron.tech@gmail.com)**, www.emtrontech.in**

ARM Processors

Development board

Lab Manual



**Introduction:**

The LPC2141/2/4/6/8 microcontrollers are based on a 32/16 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and a unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30 % with minimal performance penalty.

Due to their tiny size and low power consumption, LPC2141/2/4/6/8 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. A blend of serial communications interfaces ranging from a USB 2.0 Full Speed device, multiple UARTS, SPI, SSP to I2Cs and on-chip SRAM of 8 kB up to 40 kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for industrial control and medical systems.

**LPC2148 Features:**

**•** 16/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.

**•** 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory. 128 bit wide interface/accelerator enables high speed 60 MHz operation.

**•** In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software.

Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms.

**•** EmbeddedICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip RealMonitor software and high speed tracing of instruction execution.

**•** USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM. In addition, the LPC2146/8 provide 8 kB of on-chip RAM accessible to USB by DMA.

**•** One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14 analog inputs, with conversion times as low as 2.44 μs per channel.

**•** Single 10-bit D/A converter provides variable analog output.

**•** Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.

**•** Low power real-time clock with independent power and dedicated 32 kHz clock input.

**•** Multiple serial interfaces including two UARTs (16C550), two Fast I2C-bus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.

**•** Vectored interrupt controller with configurable priorities and vector addresses.

**•** Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.

**•** Up to nine edge or level sensitive external interrupt pins available. 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 μs.

**•** On-chip integrated oscillator operates with an external crystal in range from 1 MHz to 30 MHz and with an external oscillator up to 50 MHz.

**•** Power saving modes include Idle and Power-down.

**•** Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.

**•** Processor wake-up from Power-down mode via external interrupt, USB, Brown-Out Detect (BOD) or Real-Time Clock (RTC).

**•** Single power supply chip with Power-On Reset (POR) and BOD circuits:

**–** CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O pads.

**Board Features:**

This ARM board is an evaluation board using LPC2148 ARM7TMDI based microcontroller. The LPC2148 microcontroller has 512KB of internal flash and 32+8K RAM. Following are the salient features of the board.

Power supply: DC 9V power with LED link.

On-board linear regulators generate +3.3V/500mA and +5v/500mA from power supply.

USB connector (as alternate power source).

RS232 connectors (2)

JTAG connector

SD/MMC connector

ZigBee connector

USB B-type connector with Link-LED

2 line X 16 character LCD with back light control

64x128 Graphics LCD connector

Configurable for manual and automatic program download (ISP) via serial port.

Controllable LED on Port P0.31

Optical isolated port on Port 1.24

**Connectors:**

I/O1: 2 line X 16 character LCD

I/O2: 64x128 Graphics LCD connector

VBT: 3.3V battery connector

JTAG: JTAG connector

DBG: JTAG debug enable

RL: Optical isolated port

ISP: In System Programming

OPTO-1: 3.3V, Port P0.16 and GND

OPTO-2: 3.3V, Port P1.25 and GND

VADC: AD0.1, AD0.2, AD0.3, AD0.4 and GND

VADC: AD1.2, AD1.3, AD1.4, AD1.5 and GND

DAC: AOUT

X1: USB

SUP: 9V supply

COM1: UART1

COM2: UART2

Z: Zigbee/Bluetooth

SDRAM: SPI Port

**System Requirements**

Windows XP

Serial port

USB port

**Connecting the hardware:**

After unpacking the board connect a DC supply of 9V/800mA to the DC jack to power the board. The board can also be powered through USB. To test all the features on the board you would need the following accessories:

* USB cable
* DB-9 straight Full and Half modem serial cable

Once you have all these accessories connected to the board you can run through a simple test to verify that all the peripherals are working fine. Please refer to the ‘Hardware Configuration’ section for testing all the peripherals.

**Programming the board:**

This board can be programmed through 20-pin JTAG or through serial port using ‘Flash Magic’. ‘Flash Magic’ is a freeware windows utility used download the hex file format onto the board. Flash Magic can be downloaded from here **http://www.flashmagictool.com/**. If your PC does not have a serial port; use a USB to serial converter to download the hex file using the Flash Magic utility. For programming with JTAG your system should have a USB port and the supporting IDE which can communicate to the processor core over JTAG interface. We have successfully tested board with uLink JTAG IDE. A LINUX utility to download the hex file can be found here **http://www.pjrc.com/arm/lpc2k\_pgm/.**

**Programming board Through ISP:**

The board can be programmed through ISP in two modes:

* Auto Mode
* Manual Mode

Auto Mode:

To program in Auto mode you need a full serial cable. Short the jumper pins 1 & 2 of ISP and connect the full serial cable to COM1. When board is powered ON black boxes will be displayed on LCD. Open Flash Magic tool, select the appropriate COM port, set the Baud rate to less than or equal to 38400 bps, select device as LPC2148, interface as 'None (ISP)' and oscillator frequency as 12MHz. Specify the path of your HEX file and click START. The status is shown at the bottom on the Flash Magic window. In the 'Step 4 - Options' check 'Verify after programming' and 'Fill unused flash' options. Checking the 'Set Code Read Prot' option will not allow you will program with JTAG. So keep it unchecked unless required.

Note:

1. In Auto mode under the 'Options' tab select 'Advanced options'. In this under 'Hardware Config' tab make sure the options 'Use DTR and RTS to control RTS and P0.14' and 'Keep RTS asserted while COM port open' are checked. The values of T1 and T2 are set to 100ms and 200ms by default.

2. After programming the board in Auto mode you should disconnect the serial cable and remove the Jumper and reset the controller. This is a known issue.

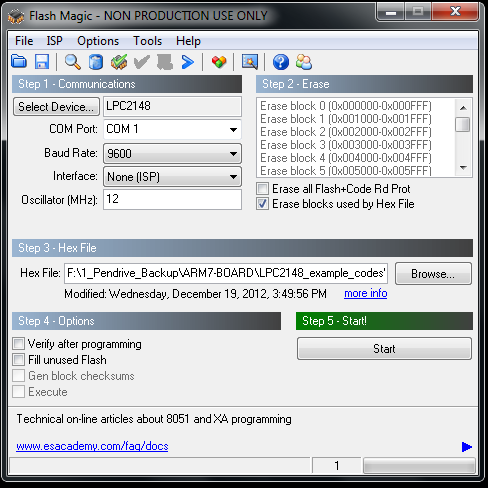
2. Manual Mode:

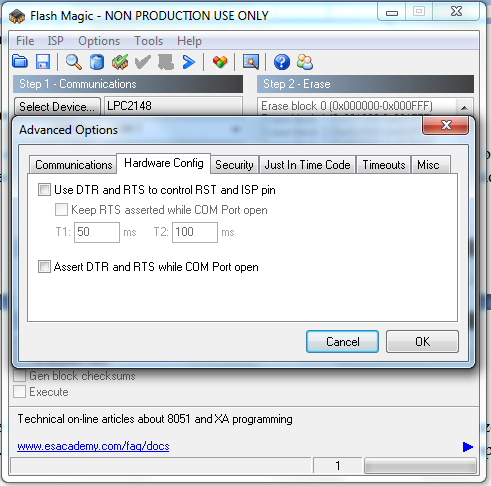
To program in Manual mode you need a half serial cable (which just has TX, RX and GND wire connected). Close the jumper ISP and connect the half serial cable to COM1and power the board. To make the board enter programming mode:

* Hold down (ISP) and (RESET), then release (RESET) first and finally (ISP).
* The controller enters the boot loader mode if during RESET the ISP pin is low

Flash Magic Settings:

1. Select Device: LPC2148
2. COM Port: COM1
3. Baud Rate: 9600
4. Interface: (None) ISP
5. Oscillator: 12 Mhz
6. Erase Blocks used by Hex file: yes
7. Advanced Options: Use DTR and RTS to control RST and ISP pin: NO





**COM1 & COM2:**

Open the hyper terminal on your PC and write USRT programs to test com port.

To test the UART you can use either a full modem or half modem cable.

**JTAG connector:**

To enable debugging on the board connect jumper to DBG and connect the JTAG to debug port. We have successfully tested the board with JTAG interface using uLink JTAG. To test this feature you need to have the necessary software support on your PC.

**User Interface Switch:**

The three pin connector OPTO-1 (P0.16), OPTO-2 (P1.25) is connected to one of the external interrupt lines of LPC. To test this interface simply press the switch and you should hear the beep sound on the buzzer. This confirms that both the interrupt line and the buzzer module are working fine. Please ensure that you have connected the buzzer jumper appropriately.

**Buzzer:**

Connect external buzzer to two pin connector (DAC), when the board is turned on or RESET you will hear a beep after few seconds. This is how the user can confirm the status of the Buzzer.

**SD/MMC connector:**

Insert a SD card in the SD card holder; the status of the SD card will be displayed on LCD upon power cycle or reset of the board. If the SD card is inserted properly “SD card – OK” is displayed on LCD else it displays “SD card- Not OK”. During manufacturing the board is tested with Kingston’s 1GB SD card.

*Note: The SD/MMC card being tested should be formatted with FAT file system (Not FAT32 or NTFS format).*

**LCD display:**

To enable the LCD connect all 16 pins of I/O1 to LCD. A default message “Emtron Technologies” will be displayed and later status of SD/MMC and I2C is displayed. An optional 20-pin I/O2 port is provided to interface 64x128 graphics LCD.

**ZigBee/Bluetooth:**

For wireless transmission between two ARM kit, the ZigBee protocol can used. ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power wireless M2M networks. The ZigBee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands including 2.4 GHz, 900 MHz and 868 MHz. This port is available on Z connector of the board.

**RTC :**

A SPI connector SDCARD is provided for external RTC. Internal ARM RTC is connected with 32.767KHz crystal.

**ADC :**

The ADC port is provided on VADC, IADC connectors. To test the ADC interface connect external potentiometer and rotate the POT, as the POT position varies the output number on LCD varies.

**Overview:**

For the working with board there are certain tools that need to be installed. The tools required to work with bard are:

• Flash Magic.

The flash magic tool can be downloaded from the following link:

**http://www.flashmagictool.com/**

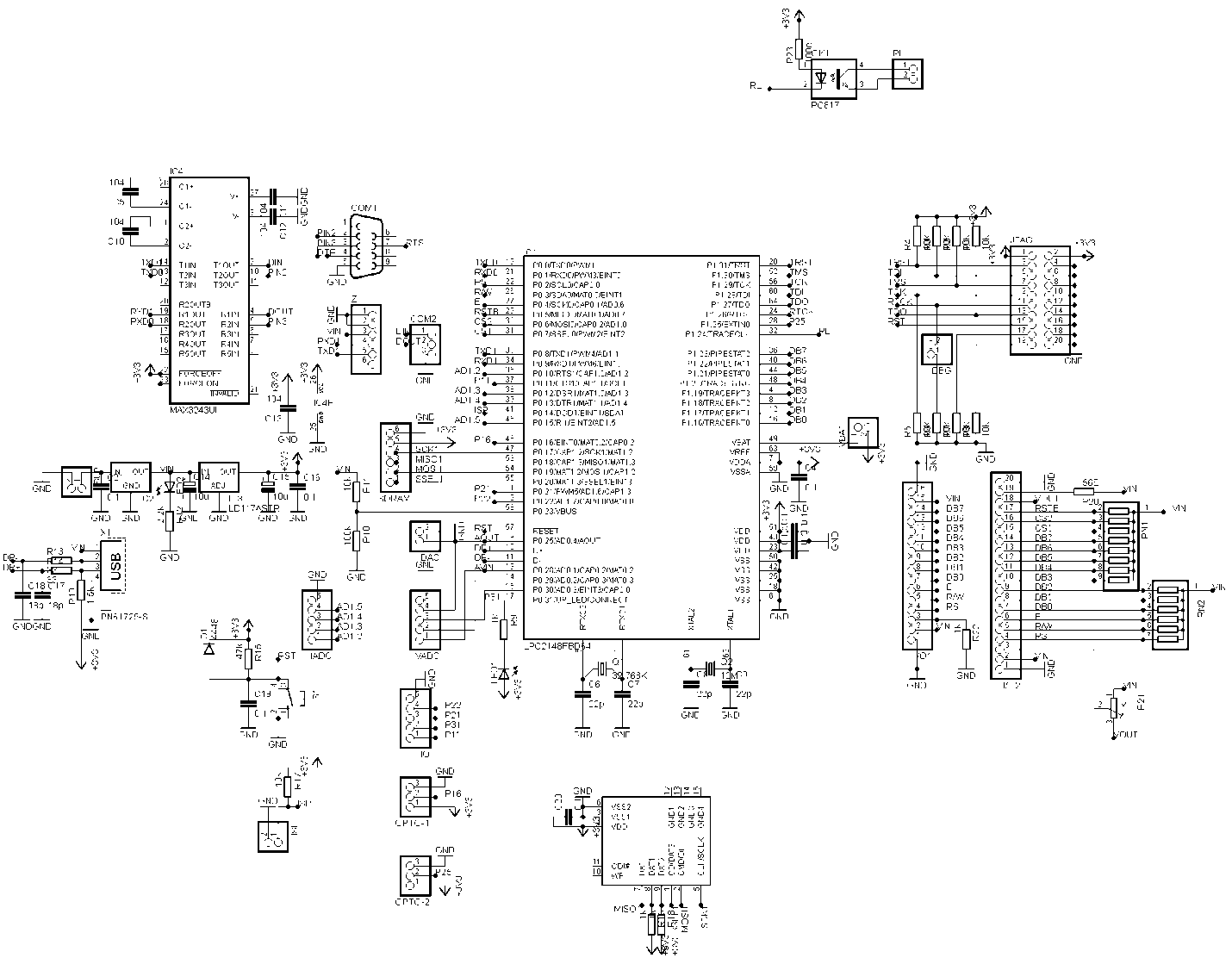
For LINUX machines you may use **http://www.pjrc.com/arm/lpc2k\_pgm/**

• Tool chain:

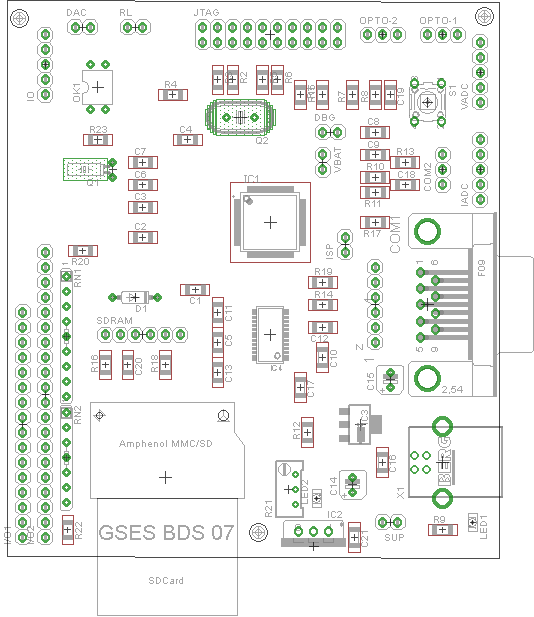
To be able to generate the hex or the binary file the user needs to install the tool chain for ARM based microcontrollers. Any toolchain can be used as long as it is able to generate the necessary files for downloading onto the board. Here are few toolchain suggestions:

* GNUARM ToolChain: **http://winarm.scienceprog.com/winarm-tools/prepare-gnuarm-compiler-toolchain-for-windows.html**
* WINARM : Rowley Crossworks IDE: [**http://www.rowley.co.uk/arm/**](http://www.rowley.co.uk/arm/)

**Board Circuit diagram:**

****

**Board Layout:**

****

**Board Part list:**

Part Value Device Package Library

C1 0.1 C-EUC1206 C1206 resistor

C2 0.1 C-EUC1206 C1206 resistor

C3 0.1 C-EUC1206 C1206 resistor

C4 0.1 C-EUC1206 C1206 resistor

C5 104 C-EUC1206 C1206 resistor

C6 22p C-EUC1206 C1206 resistor

C7 22p C-EUC1206 C1206 resistor

C8 22p C-EUC1206 C1206 resistor

C9 22p C-EUC1206 C1206 resistor

C10 104 C-EUC1206 C1206 resistor

C11 104 C-EUC1206 C1206 resistor

C12 104 C-EUC1206 C1206 resistor

C13 104 C-EUC1206 C1206 resistor

C14 10u CPOL-EUB45181A resistor

C15 10u CPOL-EUB45181A resistor

C16 0.1 C-EUC1206 C1206 resistor

C17 18p C-EUC1206 C1206 resistor

C18 18p C-EUC1206 C1206 resistor

C19 0.1 C-EUC1206 C1206 resistor

C20 0.1 C-EUC1206 C1206 resistor

C21 0.1 C-EUC1206 C1206 resistor

COM1 F09HP F09HP con-subd

COM2 PINHD-1X3 1X03 pinhead

D1 4448 1N4148DO35-7 DO35-7 diode

DAC PINHD-1X2 1X02 pinhead

DBG PINHD-1X2 1X02 pinhead

I/O1 PINHD-1X16 1X16 pinhead

I/O2 PINHD-1X20 1X20 pinhead

IADC PINHD-1X5 1X05 pinhead

IC1 LPC2148FBD64 LPC2148FBD64 TSQFP-64N

IC2 78XXS 78XXS v-reg

IC3 LD117ASTR LD117ASTR SOT223 v-reg

IC4 MAX3243UI MAX3243UI TSSOP28 maxim

IO PINHD-1X5 1X05 pinhead

ISP PINHD-1X2 1X02 pinhead

JTAG PINHD-2X10 2X10 pinhead

LED1 LEDSMT1206 1206 led

LED2 LEDSMT1206 1206 led

OK1 PC817 PC817 DIL04 optocoupler

OPTO-1 PINHD-1X3 1X03 pinhead

OPTO-2 PINHD-1X3 1X03 pinhead

Q1 32.768K CRYSTALTC26H TC26H crystal

Q2 12M CRYSTALHC49U70 HC49U70 crystal

R1 10k R-EU\_R1206 R1206 resistor

R2 10k R-EU\_R1206 R1206 resistor

R3 10k R-EU\_R1206 R1206 resistor

R4 10k R-EU\_R1206 R1206 resistor

R5 10k R-EU\_R1206 R1206 resistor

R6 10k R-EU\_R1206 R1206 resistor

R7 10k R-EU\_R1206 R1206 resistor

R8 10k R-EU\_R1206 R1206 resistor

R9 1k R-EU\_R1206 R1206 resistor

R10 100k R-EU\_R1206 R1206 resistor

R11 10k R-EU\_R1206 R1206 resistor

R12 2.2k R-EU\_R1206 R1206 resistor

R13 33 R-EU\_R1206 R1206 resistor

R14 33 R-EU\_R1206 R1206 resistor

R15 47k R-EU\_R1206 R1206 resistor

R16 1k R-EU\_R1206 R1206 resistor

R17 10k R-EU\_R1206 R1206 resistor

R18 1k R-EU\_R1206 R1206 resistor

R19 1.5k R-EU\_R1206 R1206 resistor

R20 56E R-EU\_R1206 R1206 resistor

R21 R-TRIMM64W RTRIM64W resistor

R22 1k R-EU\_R1206 R1206 resistor

R23 100e R-EU\_R1206 R1206 resistor

RL PINHD-1X2 1X02 pinhead

RN1 G08R SIL9 resistor-sil

RN2 G06R SIL7 resistor-sil

S1 10-XX B3F-10XX switch-omron

SDRAM PINHD-1X6 1X06 pinhead

SUP PINHD-1X2 1X02 pinhead

U$1 SDCARDVAR2 SDCMF-10915W010 sdcmf10915

VADC PINHD-1X5 1X05 pinhead

VBAT PINHD-1X2 1X02 pinhead

X1 PN61729-S con-berg

Z PINHD-1X6 1X06 pinhead

**Experiment List of ARM-7 (LPC2148)**

**Refer LPC2148 Examples Codes Folder**

**Experiment-1:** (code: 1\_Blinky) LED connected on (Port P0.31) of LPC2148

**Experiment-2:** (code: 1\_Blinky\_Input) LED connected on (Port P0.31) of LPC2148, Switch on PORT0.16

**Experiment-3:** (code: 2\_Test\_LCD) 16x2 LCD, 8-bit Mode, RS=PORT0.2, RW=PORT0.3, E=PORT0.4, D0 to D7 = PORT1.16 to PORT1.23

**Experiment-4:** (code: 3\_ADC\_LCD) 16x2 LCD, 8-bit Mode, RS=PORT0.2, RW=PORT0.3, E=PORT0.4, D0 to D7 = PORT1.16 to PORT1.23 & ADC Channel NO-0 (PORT0.28)

**Experiment-4:** (code: 4\_DAC) DAC sine waveform generation and amplification (PORT0.25)

**Experiment-5:** (code: 5\_ADC\_DAC\_LCD) 16x2 LCD, 8-bit Mode, RS=PORT0.2, RW=PORT0.3, E=PORT0.4, D0 to D7 = PORT1.16 to PORT1.23 & ADC Channel NO-0 (PORT0.28) & sine waveform generation and amplification (PORT0.25)

**Experiment-6:** (code: 6\_Serial) Serially communicate String “Hello World” to Computer Serial Terminal using 9600 baud rate, 8-bits, No parity, 1-Stop bit. Use UART-0 of the LPC2148 ARM Processor is working on 60MHz.

**Experiment-7:** (code: 7\_RTC) Initialize Internal RTC of LPC2148 for Seconds, Minutes & Hours and Display the data on LCD of 16x2 LCD, 8-bit Mode, RS=PORT0.2, RW=PORT0.3, E=PORT0.4, D0 to D7 = PORT1.16 to PORT1.23

**Experiment-9:** (code: 8\_Serial\_ADC) read ADC data of Channel No-0 of LPC2148 and transmit serially through UART-0 to Computer Serial Terminal using 9600 baud rate, 8-bits, No parity, and 1-Stop bit.

**Project-1:** (code: 9\_SDHC\_fat32\_3channel) Implement FAT32 file format of 1024 bytes size of each sector on s SDCARD Flash memory which is connected on SPI PORT of LPC2148. The interface signals are SCK (PORT0.17), MISO(PORT0.18), MOSI (PORT0.19), SSEL(PORT0.20). Use LCD 16x2 LCD, 8-bit Mode, RS=PORT0.2, RW=PORT0.3, E=PORT0.4, D0 to D7 = PORT1.16 to PORT1.23 to display the status of SDCARD read, write and error operations. Read ADC 3 Channels data and create a file LOG1.TXT and store the data in three columns along with DATE, TIME.

**Project-2:** (code: L&T/1. Blinky, 2. Serial\_without\_critical, 3\_serial\_with\_semaphore, 4\_soft\_timers) Implement RTOS suing FREE RTOS functions. Demonstrate use of scheduler, semaphores, critical code, and soft timers.

1. LED connected on (Port P0.31) of LPC2148 implement scheduler function of RTOS with a FLAH rate of 1000ms.
2. Create a Multi tasking (two tasksS1, S2) environment using UART-0 where both use the scheduler.
3. Create a Multi tasking (two tasksS1, S2) environment using UART-0 and implement MUTEX semaphore functions.
4. Create a MUTEX for UART-0 and also create two software timers which will generate interrupts at 2000ms & 4000ms.

**Project3-6: (**10\_Serial\_FFT)(11\_Test\_GPS)(12\_Test\_GSM)(13\_Test\_GSM\_GPS) Interface GSM, GPS of SIM 908 to LPC2148 UART-1 and Implement AT commands.

**Project-7:** (14\_DC\_STEP): Interface a Stepper Motor Driver (L293D) to LPC2148 PORTS 0.20 to PORT0.23 and write a program for bidirectional rotation. Connect +12V supply to DC and Stepper Motor Driver.

**Example Programs:**

1. Blink the LED (Port P0.31):

#include <LPC214X.h>

void wait(int count)

{ int j=0,i=0;

for(j=0;j<count;j++)

{ /\* At 60Mhz, the below loop introduces delay of 10 us \*/

for(i=0;i<35;i++); }}

int main()

{ unsigned long i=0;

IO0DIR|=((unsigned long)1<<31);

while(1){

IO0SET=((unsigned long)1<<31);

wait(100000);

IO0CLR=((unsigned long)1<<31);

wait(100000);

i++; }}

1. DAC waveform generation:

#include <LPC214x.H> /\* LPC214x definitions \*/

#include <stdio.h>

#include <math.h>

void delay(int );

int main (void)

{

unsigned char wav[100];

unsigned char ucTemp,buf[16],j=0;

PINSEL1=(1<<19);

for(ucTemp=0;ucTemp<100;ucTemp++)

{

wav[ucTemp]=100\*sin(((float)ucTemp/100)\*3.1415);

wav[ucTemp]=(wav[ucTemp]\*wav[ucTemp])/100;

}

while(j<1000)

{

for(ucTemp=0;ucTemp<100;ucTemp++)

{

DACR=wav[ucTemp]<<8;

}

j++;

}}

1. Serial Communication:

#include <LPC214X.h>

#include "Serial.h"

void wait(int count)

{

int j=0,i=0;

for(j=0;j<count;j++)

{

/\* At 60Mhz, the below loop introduces delay of 10 us \*/

for(i=0;i<35;i++);

}

}

int main()

{

uart0\_init();

IO0DIR|=((unsigned long)1<<31);

while(1){ uart0\_puts("\n\rHello World!!!");

wait(1000);

} }

**Serial.c:**

#include <LPC214x.H> /\* LPC21xx definitions \*/

#include <stdio.h>

#include "Serial.h"

#define CR 0x0D

/\* implementation of putchar (also used by printf function to output data) \*/

int sendchar (int ch) { /\* Write character to Serial Port \*/

if (ch == '\n') {

while (!(U1LSR & 0x20));

U1THR = CR; /\* output CR \*/

}

while (!(U1LSR & 0x20));

return (U1THR = ch);

}

int uart0\_getkey (void) { /\* Read character from Serial Port \*/

while (!(U0LSR & 0x01));

return (U0RBR);

}

int uart1\_getkey (void) { /\* Read character from Serial Port \*/

while (!(U1LSR & 0x01));

return (U1RBR);

}

void uart1\_init()

{

/\* initialize the serial interface

Baud= Pclk \* mulVal

-----------------

16\*(256\*U0BLM+U0DLL)\*(mulVal+divVal)

\*/

PINSEL0 |= 0x00050000; /\* Enable RxD1 and TxD1 \*/

U1LCR = 0x83; /\* 8 bits, no Parity, 1 Stop bit \*/

U1DLL = 71; /\* 9600 Baud Rate @ 15MHz VPB Clock \*/

U1FDR=0x00000083; //MulVal=8 DivVal=3

U1LCR = 0x03; /\* DLAB = 0 \*/

U1FCR=0x01;

}

//---------------- Function for send character 1 time via UART1-----------------------//

void uart1\_putc(char c)

{

while(!(U1LSR & 0x20)); // Wait until UART1 ready to send character

U1THR = c; // Send character

}

//-------------------------- Function for send string via UART1---------------------------------//

void uart1\_puts(char \*p)

{

while(\*p) // Point to character

{

uart1\_putc(\*p++); // Send character then point to next character

}

}

//---------------------------- Function for Initial UART0 ----------------------------------------//

void uart0\_init()

{

/\* initialize the serial interface

Baud= Pclk \* mulVal

-----------------

16\*(256\*U0BLM+U0DLL)\*(mulVal+divVal)

\*/

PINSEL0 |= 0x00000005; /\* Enable RxD0 and TxD0 \*/

U0LCR = 0x83; /\* 8 bits, no Parity, 1 Stop bit \*/

//U0DLL = 71; /\* 9600 Baud Rate @ 15MHz VPB Clock \*/

//U0FDR=0x00000083; /\*MulVal=8 Divide Val=3 \*/

U0DLL=97;

U0LCR = 0x03; /\* DLAB = 0 \*/

}

//----------------------- Function for send character 1 time via UART0-----------------------//

void uart0\_putc(char c)

{

while(!(U0LSR & 0x20)); // Wait until UART0 ready to send character

U0THR = c; // Send character

}

//------------------------- Function for send string via UART1---------------------------------//

void uart0\_puts(char \*p)

{

while(\*p) // Point to character

{

uart0\_putc(\*p++); // Send character then point to next character

}

}

void uart0\_dump(char \*p,unsigned short len,unsigned char mod)

{

unsigned char temp[3];

unsigned short i=0;

for(;i<len;i++){

if(i%mod==0)

uart0\_puts("\n\r");

sprintf((char \*)temp,"%02x ",p[i]);

uart0\_puts((char \*)temp);

}

}

**Serial.h:**

int uart0\_getkey(void);

int uart1\_getkey(void);

void uart1\_init (void);

void uart0\_init (void);

void uart1\_putc (char);

void uart0\_putc (char);

void uart1\_puts (char \*);

void uart0\_puts (char \*);

1. Read ADC display on LCD:

#include <stdio.h>

#include <LPC214x.H> /\* LPC214x definitions \*/

#include "lcd.h"

#include "adc.h"

/\*\* Function Name : wait() \*\*/

void wait(int count)

{ int j=0,i=0;

for(j=0;j<count;j++)

{ /\* At 60Mhz, the below loop introduces delay of 10 us \*/

for(i=0;i<35;i++); } }

/\*\* Function Name : process\_adc() \*\*/

void process\_adc(void)

{ unsigned short adc\_value = 0;

unsigned char buf[16] = {0};

adc\_value = adc\_read(ADC0, CHANNEL\_1);

sprintf((char \*)buf, "ADC:%d ", adc\_value);

lcd\_putstring(LINE1, (char \*)buf); }

/\*\* Function Name : main (void) \*\*/

int main (void)

{

init\_adc0(); // Initialize ADC

init\_lcd(); // Initialize LCD

lcd\_clear(); // clear display

lcd\_putstring(0,"LCD TEST 123");

lcd\_putstring(1,"LPC2148");

wait(900000);

lcd\_clear(); // clear display

while(1)

{ process\_adc(); // Read ADC value and display it on first line of LCD

wait(30000); } }

**LCD.c:**

#include <LPC214x.H> /\* LPC214x definitions \*/

#include "lcd.h"

#define LCD\_BACK\_LIGHT\_TIMEOUT 1000

#define LCD\_BACKLIGHT (1 << 27)

#define LCD\_BACK\_LIGHT\_DIR IO1DIR

#define LCD\_BACK\_LIGHT\_SET IO1SET

#define LCD\_BACK\_LIGHT\_CLR IO1CLR

#define LCD\_DATA\_DIR IO1DIR

#define LCD\_DATA\_SET IO1SET

#define LCD\_DATA\_CLR IO1CLR

#define LCD\_CTRL\_DIR IO0DIR

#define LCD\_CTRL\_SET IO0SET

#define LCD\_CTRL\_CLR IO0CLR

#define LCDRS (1 << 2)

#define LCDRW (1 << 3)

#define LCDEN (1 << 4)

#define LCD\_D4 (1 << 20)

#define LCD\_D5 (1 << 21)

#define LCD\_D6 (1 << 22)

#define LCD\_D7 (1 << 23)

#define LCD\_DATA\_MASK (LCD\_D4 | LCD\_D5 | LCD\_D6 | LCD\_D7)

#define LCD\_BUSY\_FLAG LCD\_D7

#define LCD\_CONTROL\_MASK 0x0000001C

/\*\* Function Name : delay() \*\*/

void delay(int count)

{

int j=0,i=0;

for(j=0;j<count;j++)

{ /\* At 60Mhz, the below loop introduces delay of 10 us \*/

for(i=0;i<35;i++); } }

/\*\* Function Name : wait\_lcd() \*\*/

void wait\_lcd( void )

{

LCD\_CTRL\_CLR |= LCDRS;

LCD\_CTRL\_SET |= LCDRW |LCDEN;

while(IO1PIN & LCD\_BUSY\_FLAG); /\* wait for busy flag to become low \*/

LCD\_CTRL\_CLR |= LCDEN | LCDRW;

LCD\_DATA\_DIR |= LCD\_DATA\_MASK;

delay(100);

}

/\*\* Function Name : lcd\_command\_write() \*\*/

void lcd\_command\_write( unsigned char command )

{

unsigned char temp=0;

unsigned int temp1=0;

temp=command;

temp=(temp>>4)&0x0F;

temp1=(temp<<20)&LCD\_DATA\_MASK;

LCD\_CTRL\_CLR = LCDRS;

LCD\_CTRL\_SET = LCDEN;

LCD\_DATA\_CLR = LCD\_DATA\_MASK;

LCD\_DATA\_SET = temp1;

delay(10000);

LCD\_CTRL\_CLR = LCDEN;

temp=command;

temp&=0x0F;

temp1=(temp<<20)&LCD\_DATA\_MASK;

delay(100\*2);

LCD\_CTRL\_CLR |= LCDRS;

LCD\_CTRL\_SET |= LCDEN;

LCD\_DATA\_CLR = LCD\_DATA\_MASK;

LCD\_DATA\_SET = temp1;

delay(10000);

LCD\_CTRL\_CLR |= LCDEN;

delay(10000);

//wait\_lcd();

}

/\*\* Function Name : set\_lcd\_port\_output() \*\*/

void set\_lcd\_port\_output( void )

{

LCD\_CTRL\_DIR |= ( LCDEN | LCDRS | LCDRW );

LCD\_CTRL\_CLR |= ( LCDEN | LCDRS | LCDRW );

LCD\_DATA\_DIR |= LCD\_DATA\_MASK;

}

/\*\* Function Name : lcd\_clear() \*\*/

void lcd\_clear( void)

{

lcd\_command\_write( 0x01 );

}

/\*\* Function Name : lcd\_gotoxy() \*\*/

int lcd\_gotoxy( unsigned int x, unsigned int y)

{

int retval = 0;

if( (x > 1) && (y > 15) )

{

retval = -1;

} else {

if( x == 0 )

{

lcd\_command\_write( 0x80 + y ); /\* command - position cursor at 0x00 (0x80 + 0x00 ) \*/

} else if( x==1 ){

lcd\_command\_write( 0xC0 + y ); /\* command - position cursor at 0x40 (0x80 + 0x00 ) \*/

} }

return retval;}

/\*\* Function Name : lcd\_data\_write() \*\*/

void lcd\_data\_write( unsigned char data )

{

unsigned char temp=0;

unsigned int temp1=0;

temp=data;

temp=(temp>>4)&0x0F;

temp1=(temp<<20)&LCD\_DATA\_MASK;

LCD\_CTRL\_SET |= LCDEN|LCDRS;

LCD\_DATA\_CLR = LCD\_DATA\_MASK;

LCD\_DATA\_SET = temp1;

delay(10000);

LCD\_CTRL\_CLR |= LCDEN;

temp=data;

temp&=0x0F;

temp1=(temp<<20)&LCD\_DATA\_MASK;

LCD\_CTRL\_SET |= LCDEN|LCDRS;

LCD\_DATA\_CLR = LCD\_DATA\_MASK;

LCD\_DATA\_SET = temp1;

delay(10000);

LCD\_CTRL\_CLR |= LCDEN;

//wait\_lcd();

delay(1000);

}

/\*\* Function Name : lcd\_putchar() \*\*/

void lcd\_putchar( int c )

{

lcd\_data\_write( c );

}

/\*\* Function Name : lcd\_putstring() \*\*/

void lcd\_putstring( unsigned char line, char \*string )

{

unsigned char len = MAX\_CHAR\_IN\_ONE\_LINE;

lcd\_gotoxy( line, 0 );

while(\*string != '\0' && len--)

{

lcd\_putchar( \*string );

string++;

}

}

/\*\* Function Name : lcd\_backlight\_on() \*\*/

void lcd\_backlight\_on()

{

LCD\_BACK\_LIGHT\_DIR |= LCD\_BACKLIGHT;

LCD\_BACK\_LIGHT\_SET |= LCD\_BACKLIGHT;

}

/\*\* Function Name : turn\_off\_lcd\_back\_light() \*\*/

void turn\_off\_lcd\_back\_light\_cb(void)

{

LCD\_BACK\_LIGHT\_DIR |= LCD\_BACKLIGHT;

LCD\_BACK\_LIGHT\_CLR |= LCD\_BACKLIGHT;

}

/\*\* Function Name : init\_lcd() \*\*/

void init\_lcd( void )

{

set\_lcd\_port\_output();

delay(100\*100);

lcd\_command\_write(0x28); /\* 4-bit interface, two line, 5X7 dots. \*/

lcd\_clear() ; /\* LCD clear \*/

lcd\_command\_write(0x02); /\* cursor home \*/

lcd\_command\_write(0x06); /\* cursor move direction \*/

lcd\_command\_write(0x0C) ; /\* display on \*/

lcd\_gotoxy(0, 0);

lcd\_clear();

}

**ADC.c:**

#include "adc.h"

#include <LPC214x.H> /\* LPC214x definitions \*/

/\* Function Name : init\_adc0()

Description : Initialises the ADC0 \*/

void init\_adc0(void)

{

PINSEL1 = (PINSEL1 & ~(3 << 25)) | (1 << 24);

}

/\*\* Function Name :init\_adc1() \*\*/

void init\_adc1(void)

{

}

/\*\* Function Name : adc\_read() \*\*/

unsigned short adc\_read(unsigned char adc\_num, unsigned char ch)

{

unsigned int i=0;

switch(adc\_num)

{

case ADC0:

AD0CR = 0x00200D00 | (1<<ch); // select channel

AD0CR |= 0x01000000; // Start A/D Conversion

do

{

i = AD0GDR; // Read A/D Data Register

} while ((i & 0x80000000) == 0); // Wait for end of A/D Conversion

break;

case ADC1:

AD1CR = 0x00200D00 | (1<<ch); // select channel

AD1CR |= 0x01000000; // Start A/D Conversion

do

{

i = AD1GDR; // Read A/D Data Register

} while ((i & 0x80000000) == 0); // Wait for end of A/D Conversion

break;

}

return (i >> 6) & 0x03FF; // bit 6:15 is 10 bit AD value

}

**ADC.h:**

#ifndef \_ADC\_H

#define \_ADC\_H

#define END\_0F\_CONVERSION\_BIT (1<<31)

#define END\_OF\_CONVERSION(i) (i & END\_0F\_CONVERSION\_BIT)

#define ADC\_VALUE\_MASK 0x03FF

#define ADC\_CHANNEL\_NUMBER\_MASK 0x07

#define CHANNEL\_0 0

#define CHANNEL\_1 1

#define CHANNEL\_2 2

#define CHANNEL\_3 3

#define CHANNEL\_4 4

#define CHANNEL\_5 5

#define CHANNEL\_6 6

#define CHANNEL\_7 7

/\* A/D Control Register \*/

#define AD0\_0 0x00000001

#define AD0\_1 0x00000002

#define AD0\_2 0x00000004

#define AD0\_3 0x00000008

#define AD0\_4 0x00000010

#define AD0\_5 0x00000020

#define AD0\_6 0x00000040

#define AD0\_7 0x00000080

#define AD1\_0 0x00000001

#define AD1\_1 0x00000002

#define AD1\_2 0x00000004

#define AD1\_3 0x00000008

#define AD1\_4 0x00000010

#define AD1\_5 0x00000020

#define AD1\_6 0x00000040

#define AD1\_7 0x00000080

#define CLKDIV\_BIT0 (1<<8)

#define CLKDIV\_BIT1 (1<<9)

#define CLKDIV\_BIT2 (1<<10)

#define CLKDIV\_BIT3 (1<<11)

#define CLKDIV\_BIT4 (1<<12)

#define CLKDIV\_BIT5 (1<<13)

#define CLKDIV\_BIT6 (1<<14)

#define CLKDIV\_BIT7 (1<<15)

#define BURST (1<<16) //to eneble burst mode

#define CLKS\_BIT0 (1<<17)

#define CLKS\_BIT1 (1<<18)

#define CLKS\_BIT2 (1<<19)

#define PDN (1<<21)

#define START\_BIT0 (1<<24)

#define START\_BIT1 (1<<25)

#define START\_BIT2 (1<<26)

#define EDGE (1<<27)

/\* A/D Global Data Register \*/

#define OVERRUN (1L<<30)

#define DONE (1L<<31)

/\* A/D Interrupt Enable Register \*/

#define ADINTEN0 (1<<0)

#define ADINTEN1 (1<<1)

#define ADINTEN2 (1<<2)

#define ADINTEN3 (1<<3)

#define ADINTEN4 (1<<4)

#define ADINTEN5 (1<<5)

#define ADINTEN6 (1<<6)

#define ADINTEN7 (1<<7)

#define ADGINTEN (1<<8)

#define ADC0 0

#define ADC1 1

#define ADC0CHANNELS 8

#define ADC1CHANNELS 8

#define POWER\_DOWN\_ADC0() AD0CR &= ~(PDN)

#define POWER\_UP\_ADC0() AD0CR |= (PDN)

#define POWER\_DOWN\_ADC1() AD1CR &= ~(PDN)

#define POWER\_UP\_ADC1() AD1CR |= (PDN)

//TN\_EVENT EVT\_ADC0;

//TN\_EVENT EVT\_ADC1;

void adc0\_isr(void);

void adc1\_isr(void);

void init\_adc0( void );

void init\_adc1( void );

unsigned short adc\_read(unsigned char adc\_num, unsigned char ch);

#endif