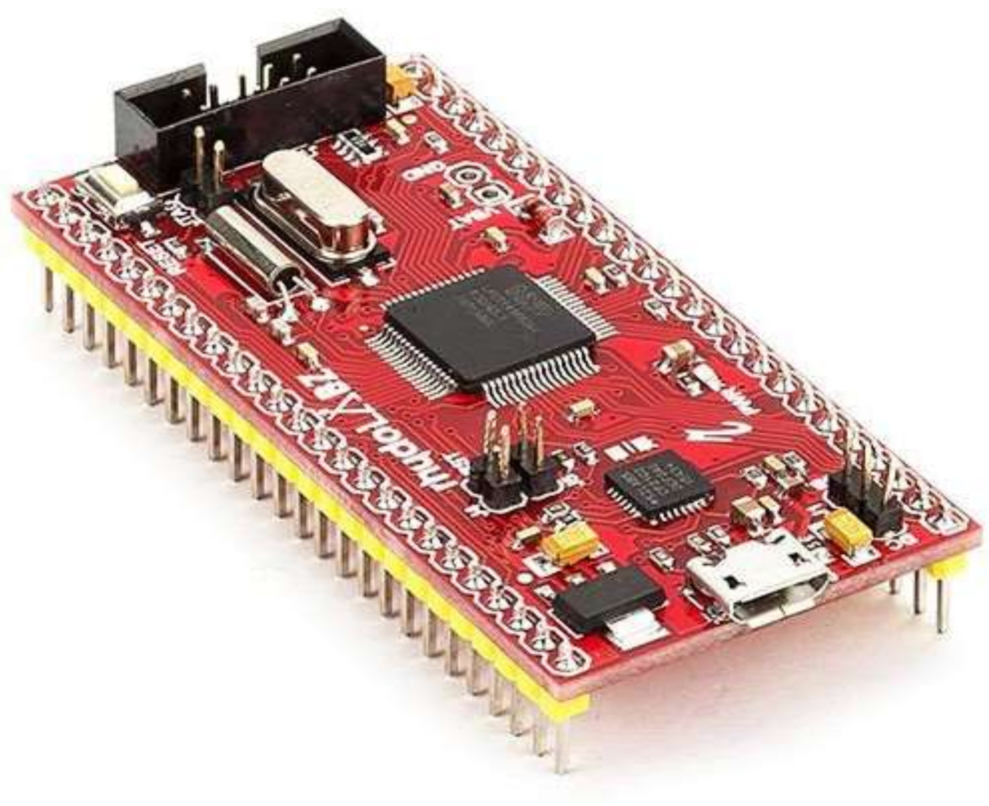
SSG EMBEDDED SOLUTIONS

SSG

info@ssges.co.in

Development Kit
ARM7 Stick - LPC2148
Microcontroller
Documentation



List Of Content

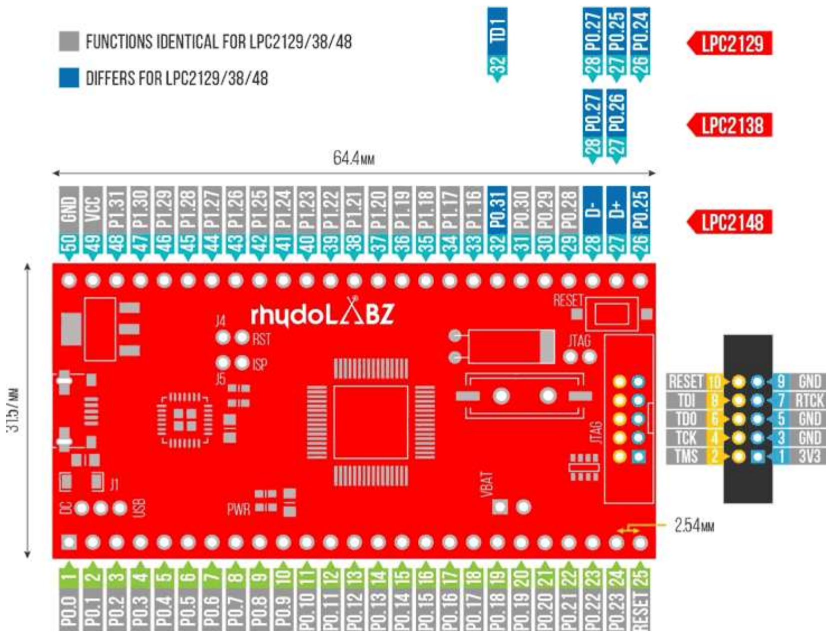
Introduction

Every electronics engineer loves to break electronics things and explore what is present inside it. Recently I opened an LED TV and found **ARM Chip** inside it. ARM based microcontrollers are heavily used in various types of embedded products and systems. They comprise many advanced features that make them powerful and superior to other microcontrollers such as 8051, AVR and PIC. LPC2148 is one of the most commonly used ARM based Microcontroller.

ARM (Advanced RISC Machines) originally known as Acorn RISC Machine is a family of reduced instruction set computing (RISC) architecture for computer processors, configured for various environments. **Arm holdings** is a British company who developed this architecture and licensed it to other companies, who design their own product by using this architecture.

ARM processor is widely found in electronics products such as LED TV, mobile phones, tablets, multimedia devices, gaming devices etc. Even popular electronics company like Apple's mobiles and iPods, Raspberry pi 3 uses ARM architecture and Arm processor in it. Arm Architecture Examples: **ARM7, ARM9, ARM11, CORTEX**.

Pin Configuration



Features

- 16/32-bit ARM7TDMI Microcontroller in a tiny LQFP64 package
- 40 kB of on-chip static RAM and 512 kB of on-chip flash memory. 128-bit wide interface/accelerator enables high-speed 60 MHz operation
- In-System Programming/In-Application Programming (ISP/IAP) via on-chip boot loader software. Single flash sector or full chip erase in 400ms and programming of 256 B in 1 ms
- **USB 2.0 full-speed compliant device controller** with 2 kB of endpoint RAM
- Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input
- ISP (in system programming) or IAP (in application programming) using on-chip boot loader software.

Applications

Embedded System: Consumer electronics, medical medical devices, and automotive control system, due to its compact size and low power consumption.

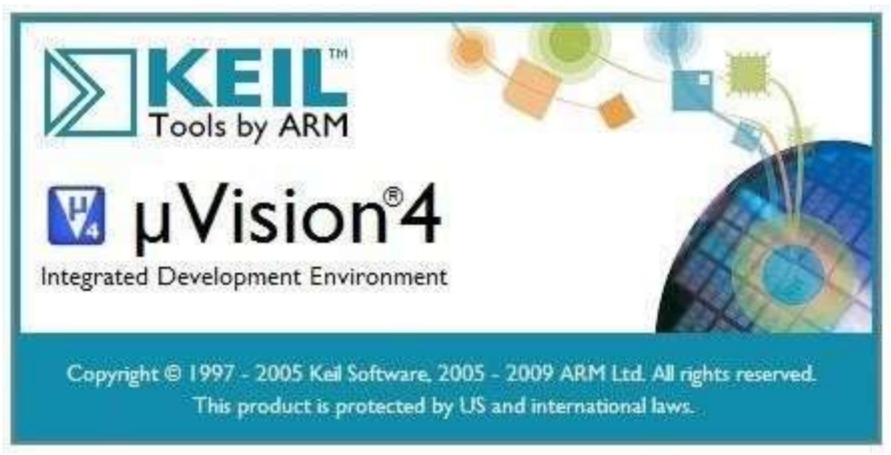
IoT: LPC2148 can collect data from sensor and send it to the cloud or other devices using wireless communication protocol like WIFI, Bluetooth, or MQTT.

Robotics: LPC2148 can serve as brain of robots. It can control motor, sensors, and provide the necessary processing power for robot navigation, manipulation, and communication.

Install Keil uVision4 IDE

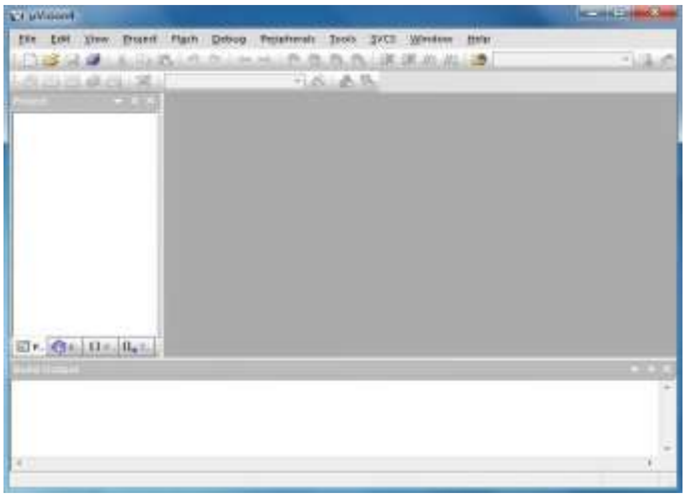
Coding is done using Keil uVision4 IDE (Download).
Procedure

Step1 :
Launch Keil uVision4

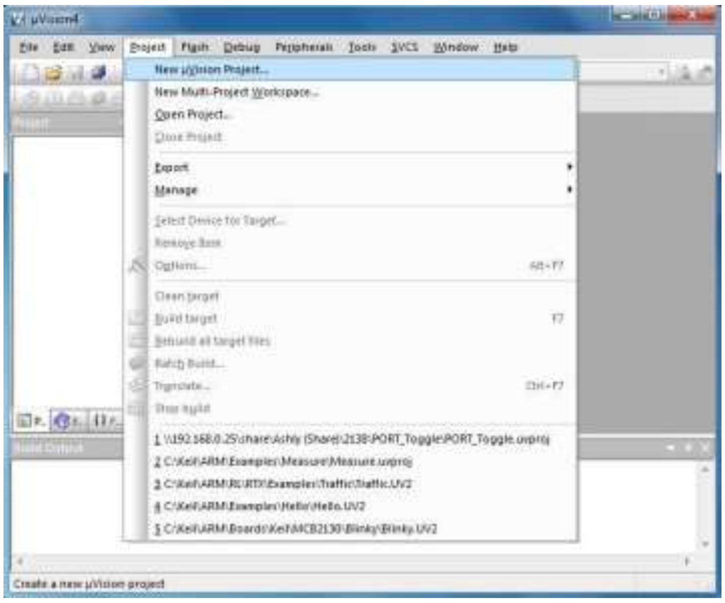


Copyright © 1997 - 2005 Keil Software, 2005 - 2009 ARM Ltd. All rights reserved.
This product is protected by US and international laws.

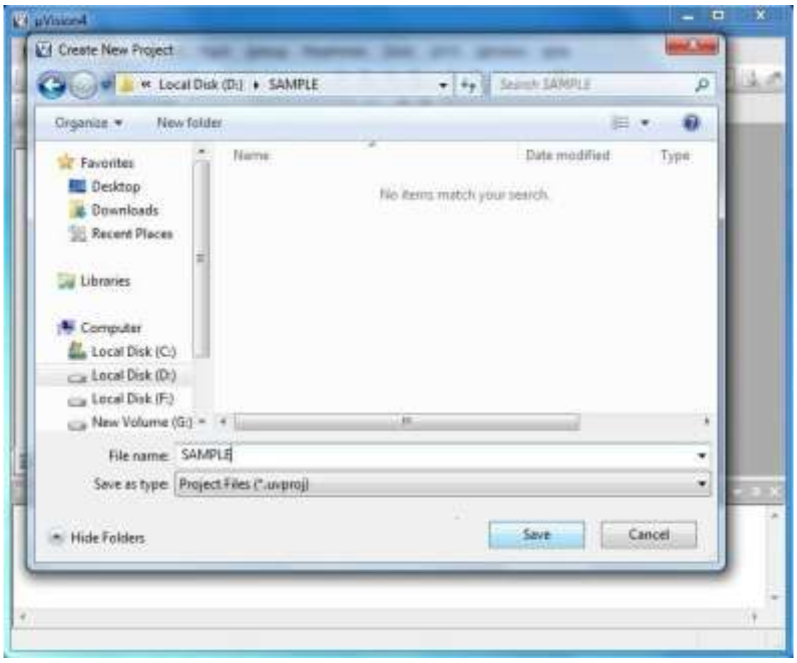
Step 2 : The Keil uvision window opens as shown below


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Step 3 : Create a new project, select *Project > New uVision **Project*** from menu bar


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target...
Remove Item
Options... Alt+F7
Clean target
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 \\192.168.0.25\share\Ashly (Share)\2138\PORT_Toggle\PORT_Toggle.uvproj
2 C:\Keil\ARM\Examples\Measure\Measure.uvproj
3 C:\Keil\ARM\RL\RTX\Examples\Traffic\Traffic.UV2
4 C:\Keil\ARM\Examples\Hello\Hello.UV2
5 C:\Keil\ARM\Boards\Keil\MCB2130\Blinky\Blinky.UV2
Create a new µVision project

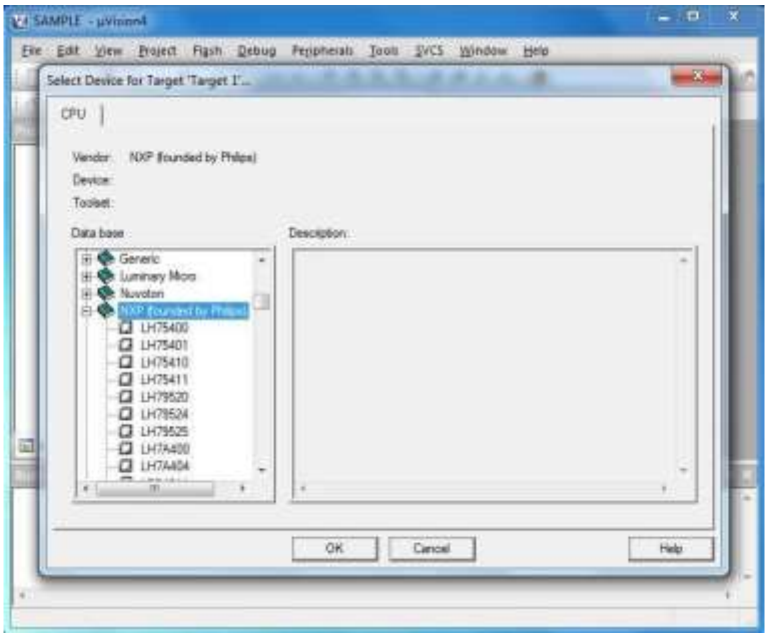
Step 4 : Create a directory for the project, and save the project files with suitable name



File name: SAMPLE

Save as type: Project Files (*.uvproj)

Step 5 : Select the Target Device. Here we are using LPC2138 (NXP>LPC2138) for illustration

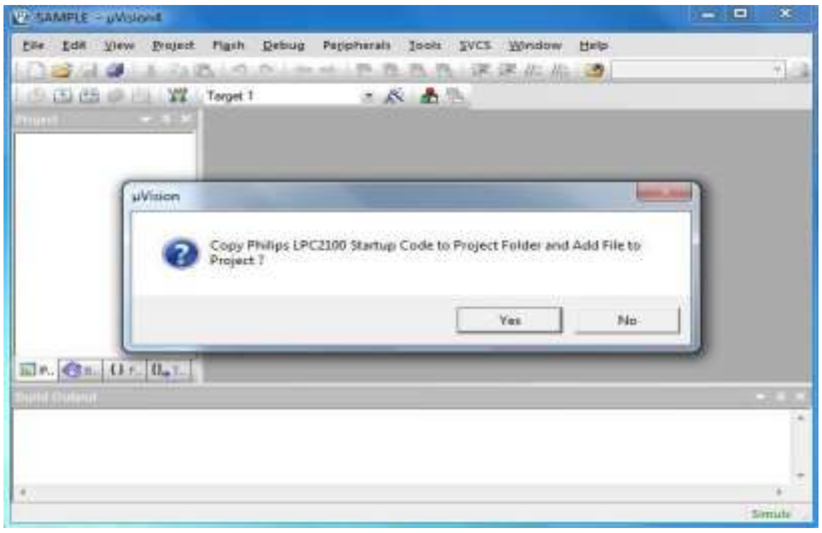

SAMPLE - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Select Device for Target 'Target 1'...
CPU
OK
Cancel
Help

Vendor: NXP (founded by Philips)
Device:
Toolset:

Data base
- Generic
- Luminary Micro
- Nuvoton
- NXP (founded by Philips)
  - LH75400
  - LH75401
  - LH75410
  - LH75411
  - LH79520
  - LH79524
  - LH79525
  - LH7A400
  - LH7A404

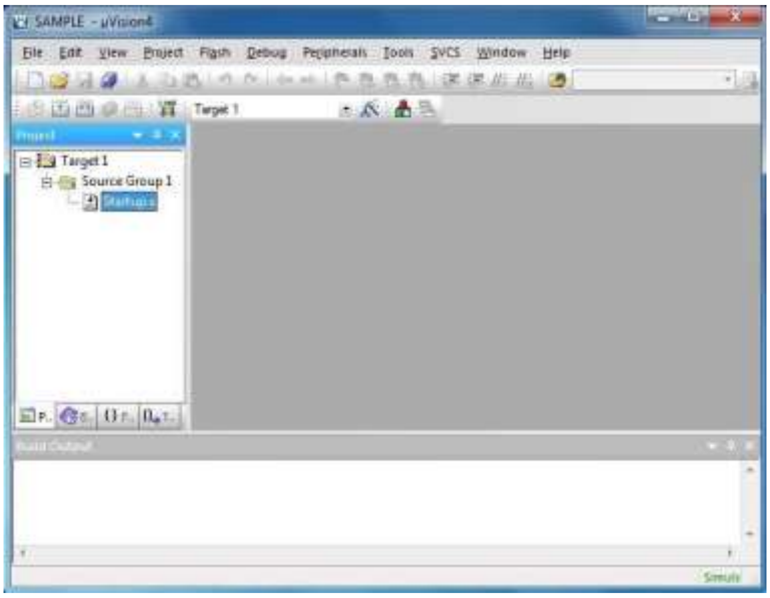
Description:

Step 6 : Click '**Yes**' for the following question to copy and add the Startup code to Project

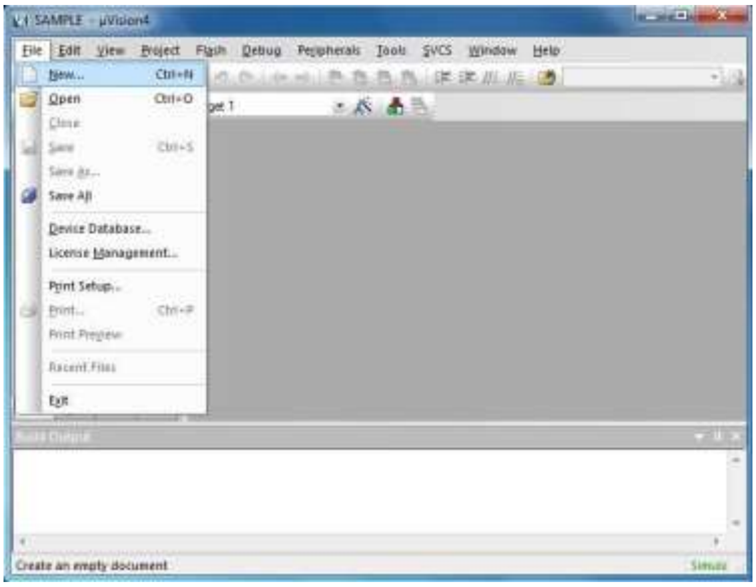

SAMPLE - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
µVision
Yes
No
Simulation

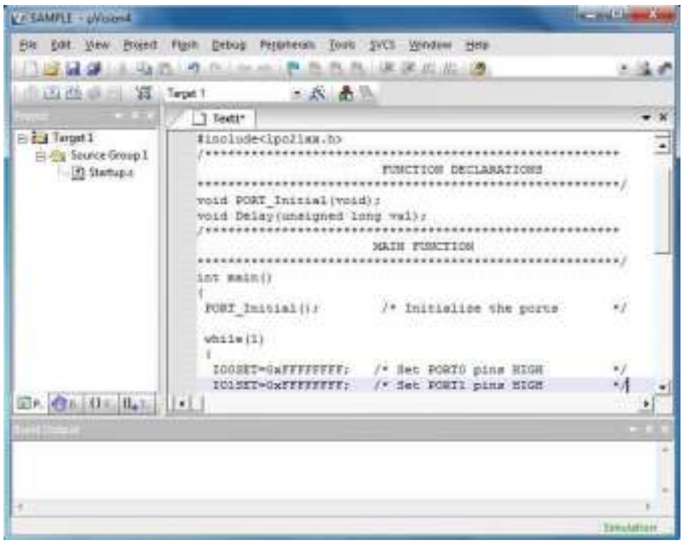
Copy Philips LPC2100 Startup Code to Project Folder and Add File to Project ?

Step 7 : This creates a target to the project

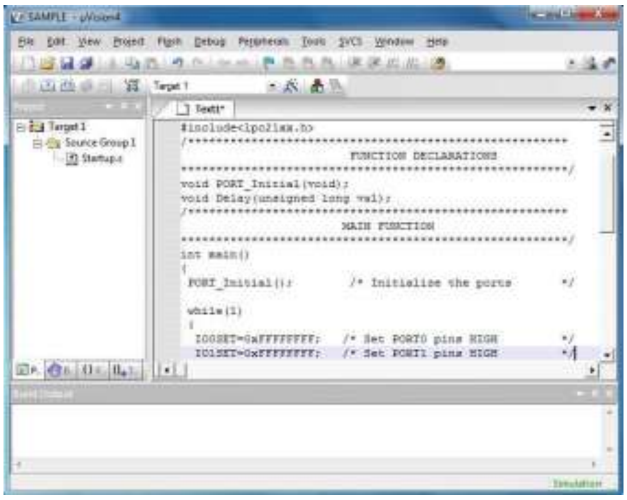


Step 8 : Open a new file (***File > New***) and do the necessary coding



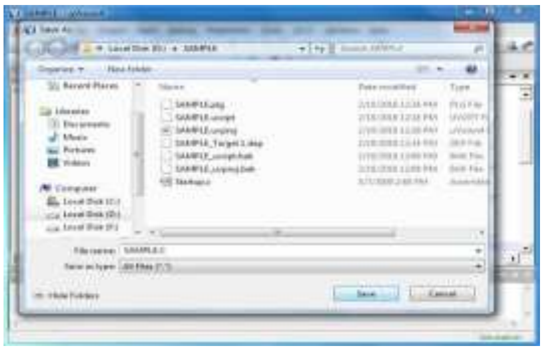


```
#include<lpc21xx.h>
/*********************************************************
                        FUNCTION DECLARATIONS
*********************************************************/
void PORT_Initial(void);
void Delay(unsigned long val);
/*********************************************************
                        MAIN FUNCTION
*********************************************************/
int main()
{
 PORT_Initial();        /* Initialize the ports       */

 while(1)
 {
  IO0SET=0xFFFFFFFF;   /* Set PORT0 pins HIGH         */
  IO1SET=0xFFFFFFFF;   /* Set PORT1 pins HIGH         */
```



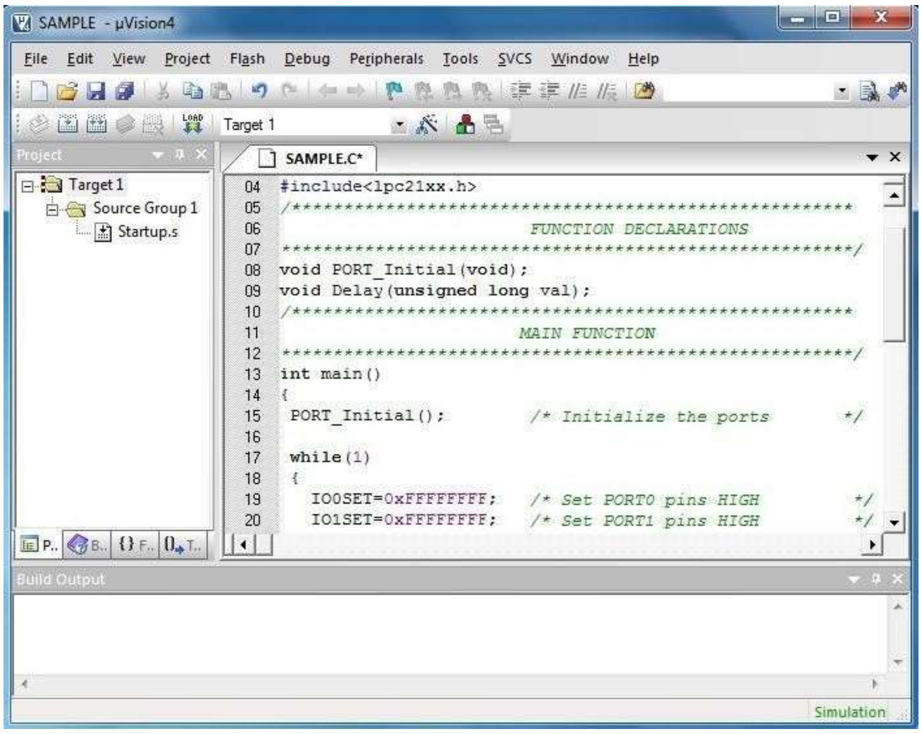
```
#include<lpc21xx.h>
/*********************************************************
                        FUNCTION DECLARATIONS
*********************************************************/
void PORT_Initial(void);
void Delay(unsigned long val);
/*********************************************************
                        MAIN FUNCTION
*********************************************************/
int main()
{
 PORT_Initial();        /* Initialize the ports       */

 while(1)
 {
  IO0SET=0xFFFFFFFF;   /* Set PORT0 pins HIGH         */
  IO1SET=0xFFFFFFFF;   /* Set PORT1 pins HIGH         */
```

Step 9 : Save the file (***File > Save***) with a **.c** extension in the project folder



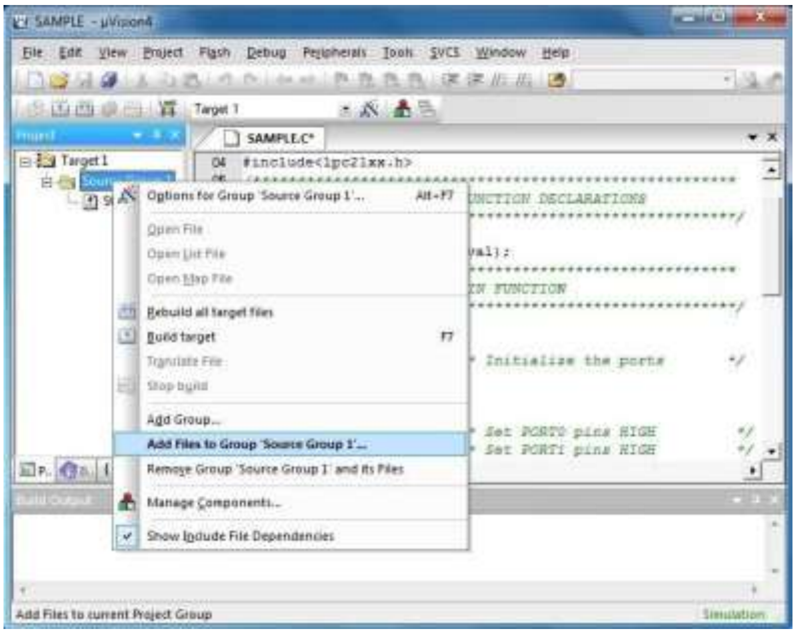
File name: SAMPLE.c

Save as type: All Files (*.*)



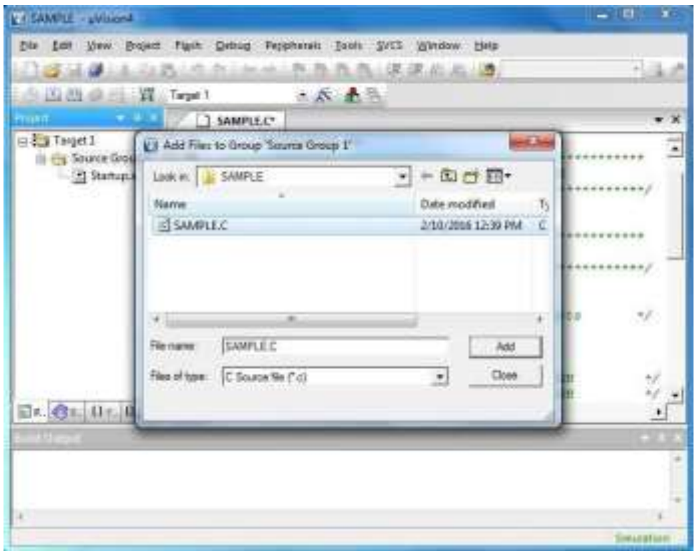
```
#include<lpc21xx.h>
/**********************************************************
                        FUNCTION DECLARATIONS
**********************************************************/
void PORT_Initial(void);
void Delay(unsigned long val);
/**********************************************************
                       MAIN FUNCTION
**********************************************************/
int main()
{
 PORT_Initial();        /* Initialize the ports       */

 while(1)
 {
   IO0SET=0xFFFFFFFF;   /* Set PORT0 pins HIGH        */
   IO1SET=0xFFFFFFFF;   /* Set PORT1 pins HIGH        */
```

Step 10 : Right click ***Source Group 1*** to add C file to source



SAMPLE - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Options for Group 'Source Group 1'... Alt+F7
Open File
Open List File
Open Map File
Rebuild all target files
Build target F7
Translate File
Stop build
Add Group...
Add Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Manage Components...
Show Include File Dependencies
Add Files to current Project Group
Simulation

Step 11 : Select the C file created and cilck Add


Add Files to Group 'Source Group 1'

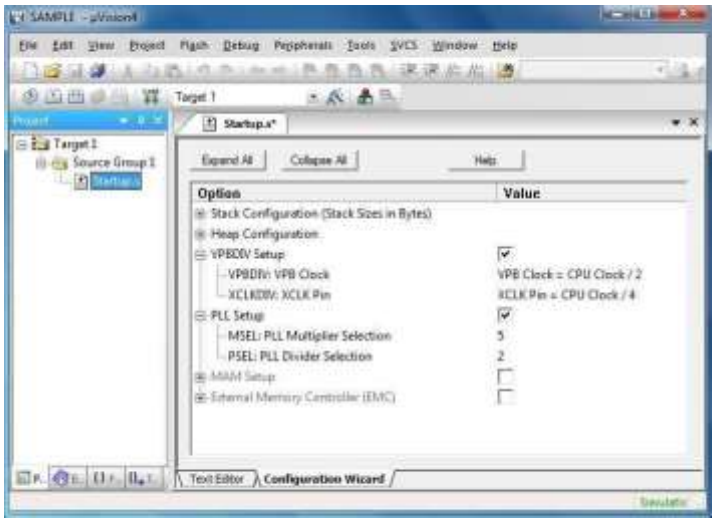
Look in: SAMPLE

Name	Date modified
SAMPLE.C	2/10/2016 12:39 PM

File name: SAMPLE.C

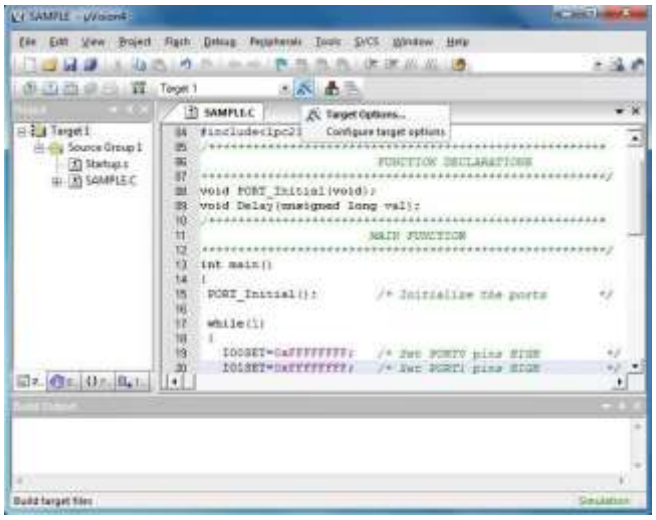
Files of type: C Source file (*.c)

Step 12 : Double click on “**Startup.s**” to open the configuration window and configure as below.
The PLL setup is done for 12MHz crystal. The divider (P) and multiplier (M) are selected such that the ***PLL output is 60MHz*** and and frequency of ***PLL current controlled oscillator is within the specified range of 156 – 320 MHz***. If crystal frequency is changed, then these values must be changed accordingly.


Startup.s*

Option	Value
Stack Configuration (Stack Sizes in Bytes)	
Heap Configuration	
VPBDIV Setup	✓
VPBDIV: VPB Clock	VPB Clock = CPU Clock / 2
XCLKDIV: XCLK Pin	XCLK Pin = CPU Clock / 4
PLL Setup	✓
MSEL: PLL Multiplier Selection	5
PSEL: PLL Divider Selection	2
MAM Setup	☐
External Memory Controller (EMC)	☐

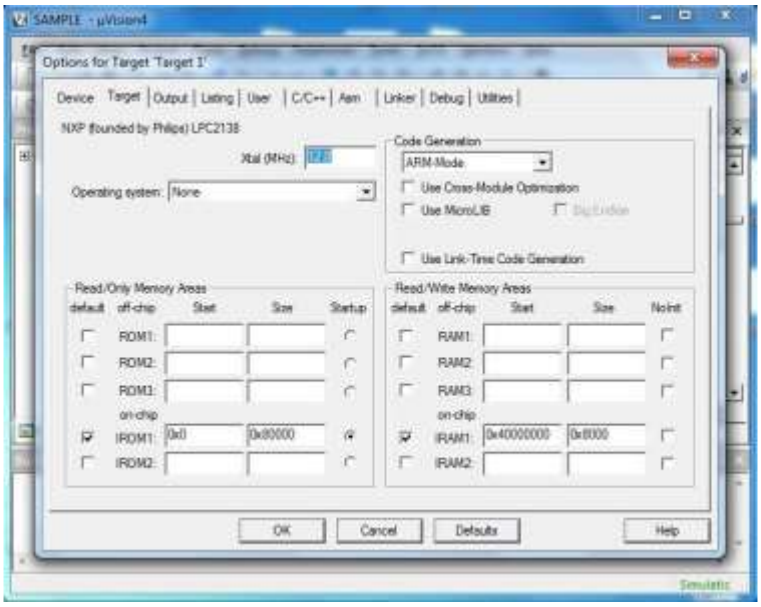
Step 13 : Click the ***icon on ‘Build toolbar’*** to set target file options. You can also do this by ‘***Project > Options for Target ‘Target 1’*** or **right click on Target1**



```
#include<lpc21
/*****************************************************
                FUNCTION DECLARATIONS
*****************************************************/
void PORT_Initial(void);
void Delay(unsigned long val);
/*****************************************************
                MAIN FUNCTION
*****************************************************/
int main()
{
 PORT_Initial();        /* Initialize the ports      */

 while(1)
 {
   IO0SET=0xFFFFFFFF;   /* Set PORT0 pins HIGH       */
   IO1SET=0xFFFFFFFF;   /* Set PORT1 pins HIGH       */
```

Step 14 : Configure **Target**, **Output** and **Linker** options as shown below

Target



NXP (founded by Philips) LPC2138

Xtal (MHz): [illegible]

Operating system: None

ARM-Mode

Use Cross-Module Optimization

Use MicroLIB

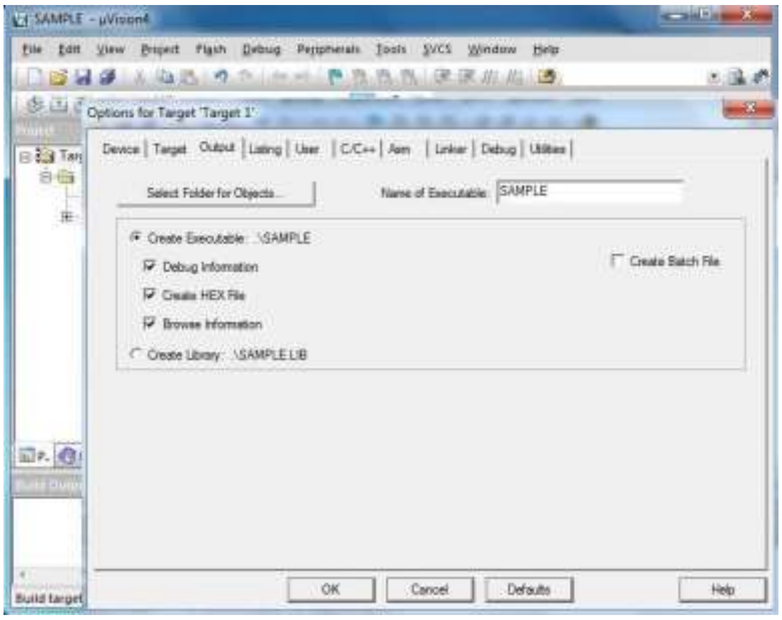
Big Endian

Use Link-Time Code Generation

default	off-chip	Start	Size	Startup
	ROM1:			
	ROM2:			
	ROM3:			
	on-chip			
✓	IROM1:	0x0	0x80000	●
	IROM2:			

default	off-chip	Start	Size	NoInit
	RAM1:			
	RAM2:			
	RAM3:			
	on-chip			
✓	IRAM1:	0x40000000	0x8000	
	IRAM2:			

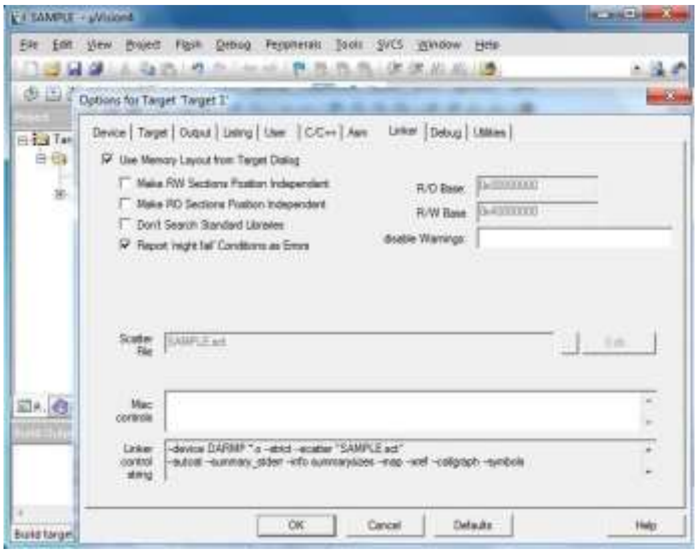
Output


Options for Target 'Target 1' — Output

Name of Executable: SAMPLE

- (●) Create Executable: .\SAMPLE
  - [✓] Debug Information
  - [✓] Create HEX File
  - [✓] Browse Information
- ( ) Create Library: .\SAMPLE.LIB
- [ ] Create Batch File

Linker


Options for Target 'Target 1' — Linker

- [✓] Use Memory Layout from Target Dialog
  - [ ] Make RW Sections Position Independent
  - [ ] Make RO Sections Position Independent
  - [ ] Don't Search Standard Libraries
  - [✓] Report 'might fail' Conditions as Errors

R/O Base: 0x00000000
R/W Base: 0x40000000
disable Warnings:

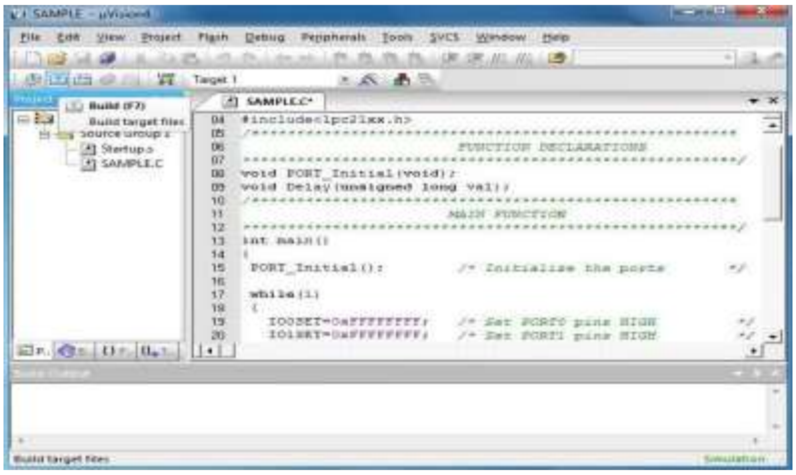
Scatter File: SAMPLE.sct

Misc controls:

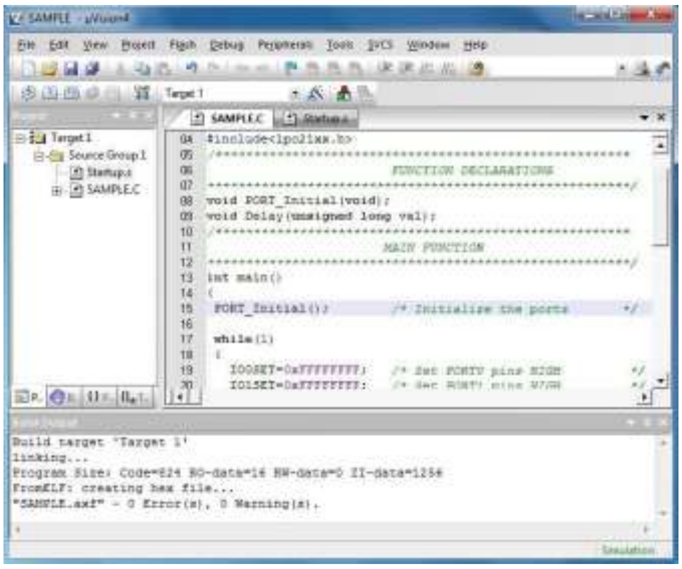
Linker control string:

```
--device DARMP *.o --strict --scatter "SAMPLE.sct"
--autoat --summary_stderr --info summarysizes --map --xref --callgraph --symbols
```

Step 15 : Click the ***build icon*** to build the project. Errors (if any) get listed in the Build output window. Correct them and build again. On successful building, the hex file will be generated in the project folder



```
#include<lpc21xx.h>
/**************************************************************
                         FUNCTION DECLARATIONS
**************************************************************/
void PORT_Initial(void);
void Delay(unsigned long val);
/**************************************************************
                         MAIN FUNCTION
**************************************************************/
int main()
{
 PORT_Initial();          /* Initialize the ports          */

 while(1)
 {
   IO0SET=0xFFFFFFFF;     /* Set PORT0 pins HIGH           */
   IO1SET=0xFFFFFFFF;     /* Set PORT1 pins HIGH           */
```



```
#include<lpc21xx.h>
/**************************************************************
                         FUNCTION DECLARATIONS
**************************************************************/
void PORT_Initial(void);
void Delay(unsigned long val);
/**************************************************************
                         MAIN FUNCTION
**************************************************************/
int main()
{
 PORT_Initial();          /* Initialize the ports          */

 while(1)
 {
   IO0SET=0xFFFFFFFF;     /* Set PORT0 pins HIGH           */
   IO1SET=0xFFFFFFFF;     /* Set PORT1 pins HIGH           */
```

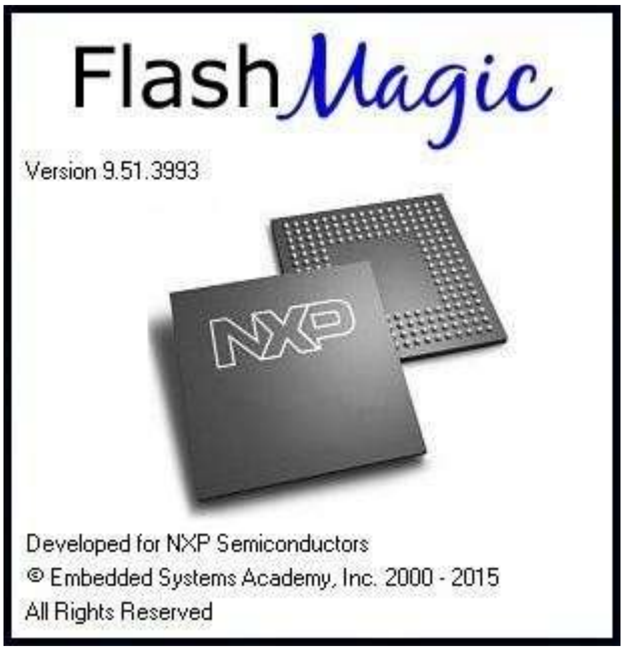
```
Build target 'Target 1'
linking...
Program Size: Code=624 RO-data=16 RW-data=0 ZI-data=1256
FromELF: creating hex file...
"SAMPLE.axf" - 0 Error(s), 0 Warning(s).
```

Once you have completed your program and generated the hex file, its time to flash it into your controller.

Flash Magic Programmer

Flash Magic is a PC tool for programming flash based microcontrollers from NXP using a serial or Ethernet protocol while in the target hardware.

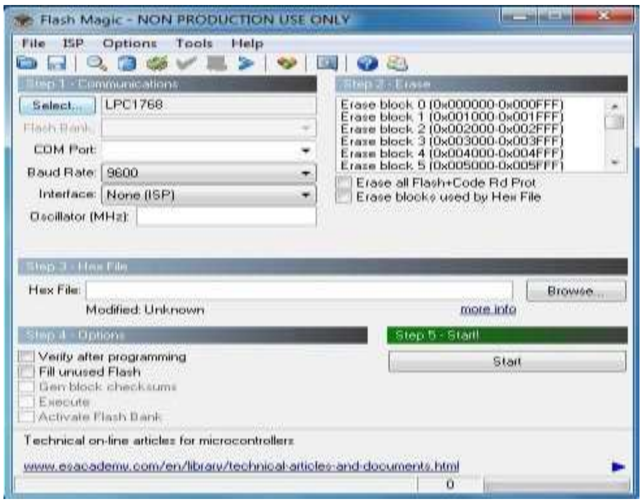
- **Launch Flash Magic**



FlashMagic

Version 9.51.3993

Developed for NXP Semiconductors
© Embedded Systems Academy, Inc. 2000 - 2015
All Rights Reserved



Flash Magic - NON PRODUCTION USE ONLY

Step 1 - Communications

Select... LPC1768
Flash Bank:
COM Port:
Baud Rate: 9600
Interface: None (ISP)
Oscillator (MHz):

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

Erase all Flash+Code Rd Prot
Erase blocks used by Hex File

Step 3 - Hex File

Hex File: Browse...
Modified: Unknown more info

Step 4 - Options

Verify after programming
Fill unused Flash
Gen block checksums
Execute
Activate Flash Bank

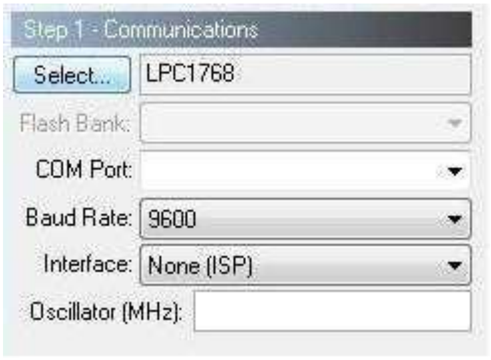
Step 5 - Start!

Start

Technical on-line articles for microcontrollers
www.esacademy.com/en/library/technical-articles-and-documents.html

0

Step 1 : Communications


Step 1 - Communications

Select... LPC1768

Flash Bank:

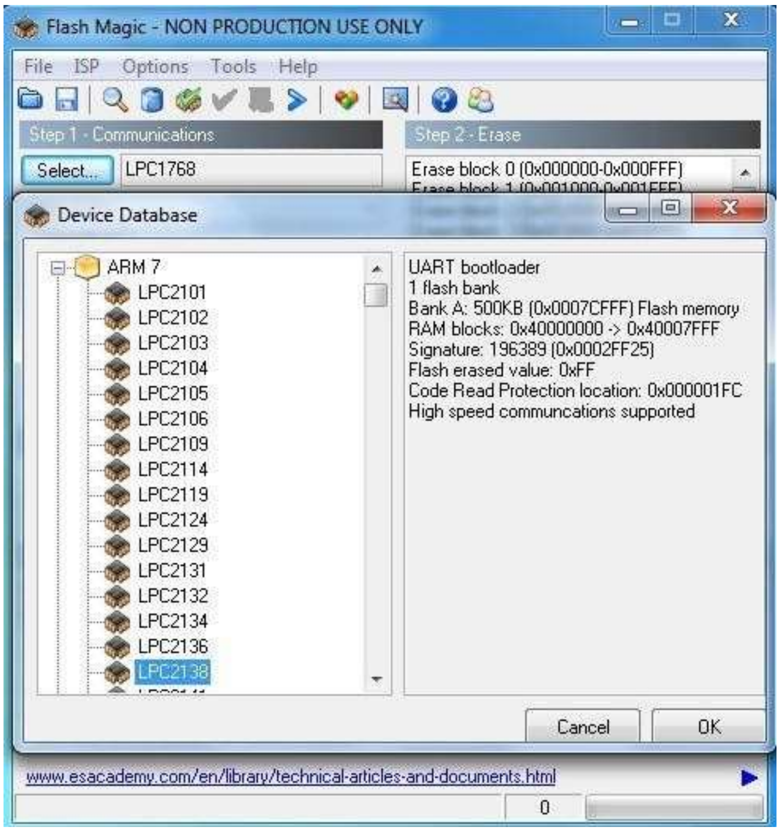
COM Port:

Baud Rate: 9600

Interface: None (ISP)

Oscillator (MHz):

- Select your target device


Flash Magic - NON PRODUCTION USE ONLY
File ISP Options Tools Help
Cancel
OK

Step 1 - Communications

Select... LPC1768

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)

Device Database

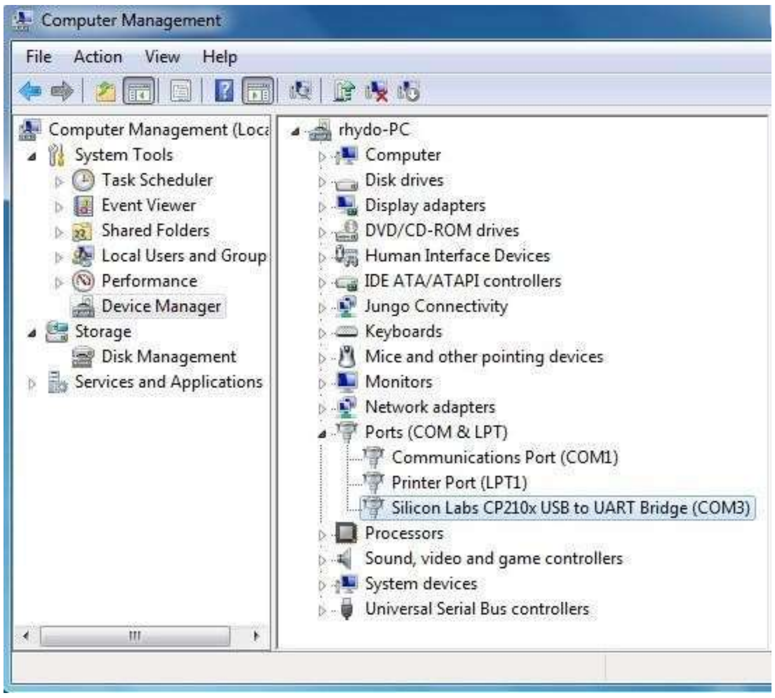
ARM 7: LPC2101, LPC2102, LPC2103, LPC2104, LPC2105, LPC2106, LPC2109, LPC2114, LPC2119, LPC2124, LPC2129, LPC2131, LPC2132, LPC2134, LPC2136, LPC2138

UART bootloader
1 flash bank
Bank A: 500KB (0x0007CFFF) Flash memory
RAM blocks: 0x40000000 -> 0x40007FFF
Signature: 196389 (0x0002FF25)
Flash erased value: 0xFF
Code Read Protection location: 0x000001FC
High speed communcations supported

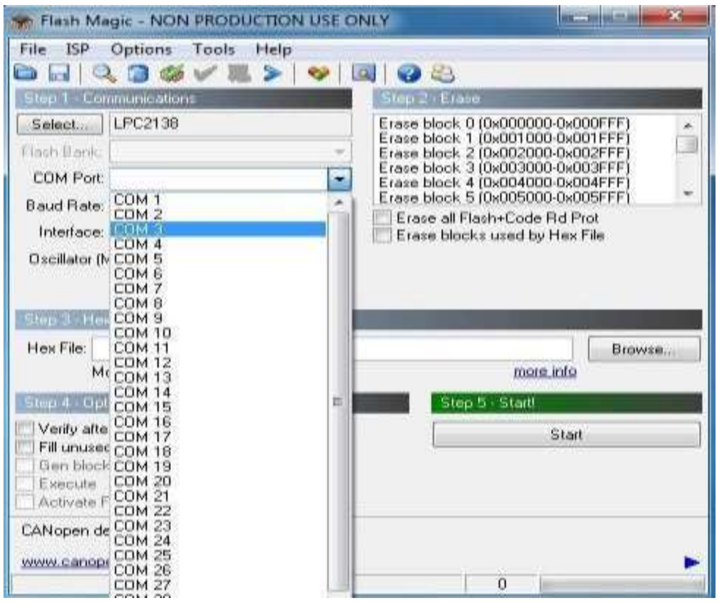
www.esacademy.com/en/library/technical-articles-and-documents.html

0

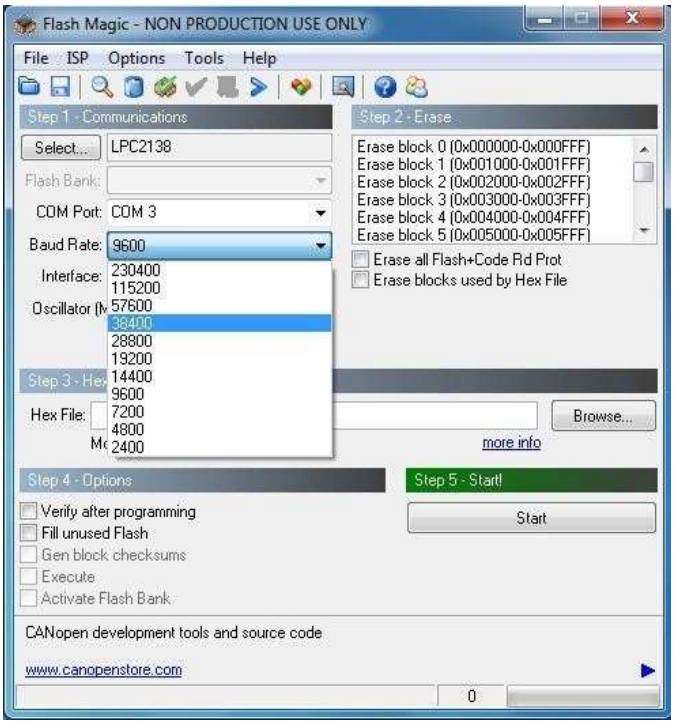
- Check for the COM port in device manager if you are using USB.



- Select your com port.



- Select baud rate for program to the target.



LPC2138

COM Port: COM 3

Baud Rate: 9600

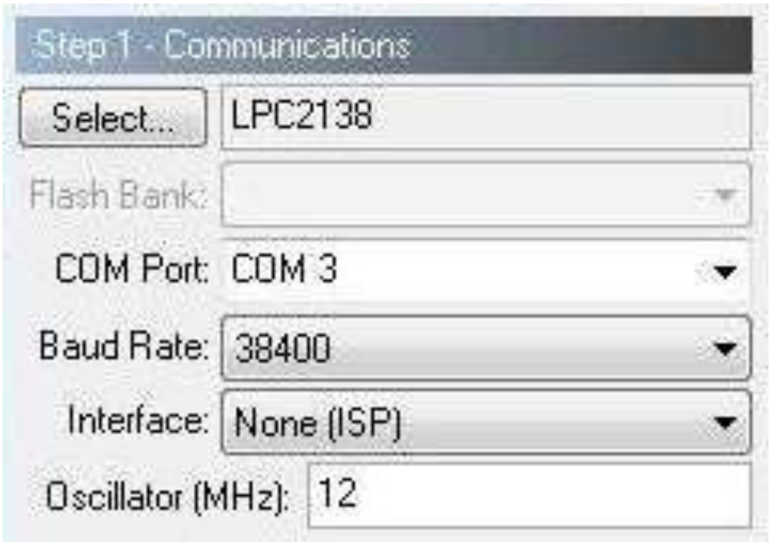
230400
115200
57600
38400
28800
19200
14400
9600
7200
4800
2400

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

Erase all Flash+Code Rd Prot
Erase blocks used by Hex File

Verify after programming
Fill unused Flash
Gen block checksums
Execute
Activate Flash Bank

- Select your interface if you are using DB-9 then it will be None (ISP).
- Give your oscillator frequency in MHz.



LPC2138

COM Port: COM 3

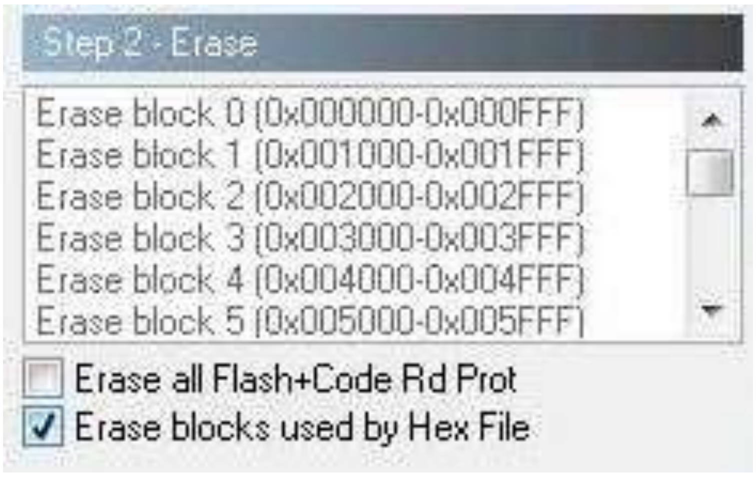
Baud Rate: 38400

Interface: None (ISP)

Oscillator (MHz): 12

Step: Erase

- Tick erase block used hex file.

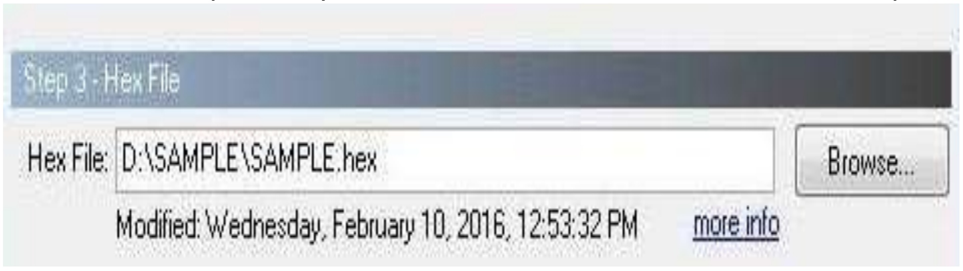

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

☐ Erase all Flash+Code Rd Prot
☑ Erase blocks used by Hex File

Step 3 : Hex file

- Browse the path of your Hex file which is to be loaded on chip.

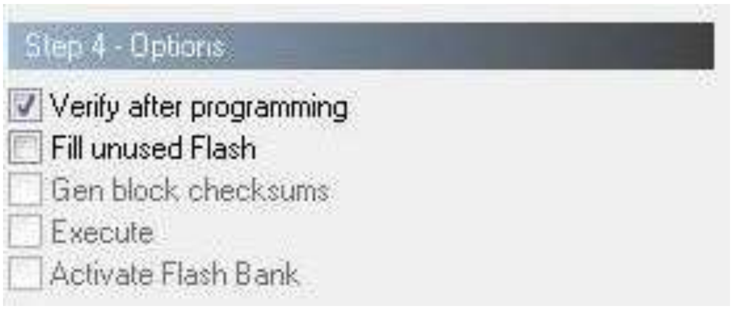

Step 3 - Hex File

Hex File: D:\SAMPLE\SAMPLE.hex

Modified: Wednesday, February 10, 2016, 12:53:32 PM

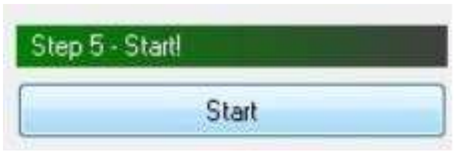
Step 4 : Options

Here always keep Verify after programming option enable by tick mark. You can use another features as well according to your need.

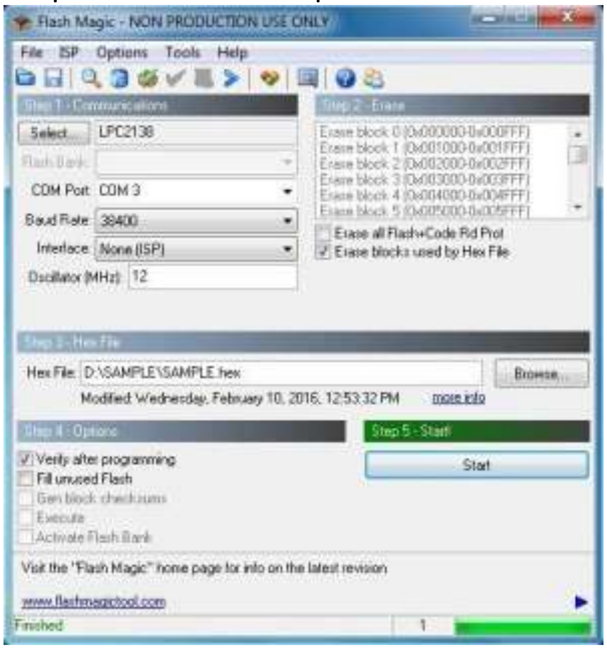

Step 4 - Options

☑ Verify after programming
☐ Fill unused Flash
☐ Gen block checksums
☐ Execute
☐ Activate Flash Bank

Step 5 : Start



Now you are all set to burn your code memory just click on start but and it will start to load hex code in your chip. You can see the process at the bottom.



Flash Magic - NON PRODUCTION USE ONLY

File ISP Options Tools Help

Step 1 - Communications

Select... LPC2138
Flash Bank:
COM Port: COM 3
Baud Rate: 38400
Interface: None (ISP)
Oscillator (MHz): 12

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

- [ ] Erase all Flash+Code Rd Prot
- [x] Erase blocks used by Hex File

Step 3 - Hex File

Hex File: D:\SAMPLE\SAMPLE.hex Browse...
Modified: Wednesday, February 10, 2016, 12:53:32 PM more info

Step 4 - Options

- [x] Verify after programming
- [ ] Fill unused Flash
- [ ] Gen block checksums
- [ ] Execute
- [ ] Activate Flash Bank

Step 5 - Start!

Start

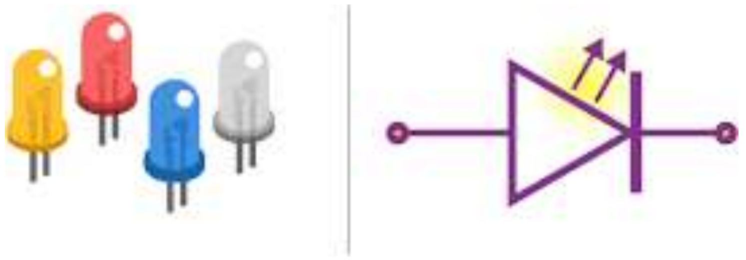
Visit the "Flash Magic" home page for info on the latest revision

www.flashmagictool.com

Finished 1

LED

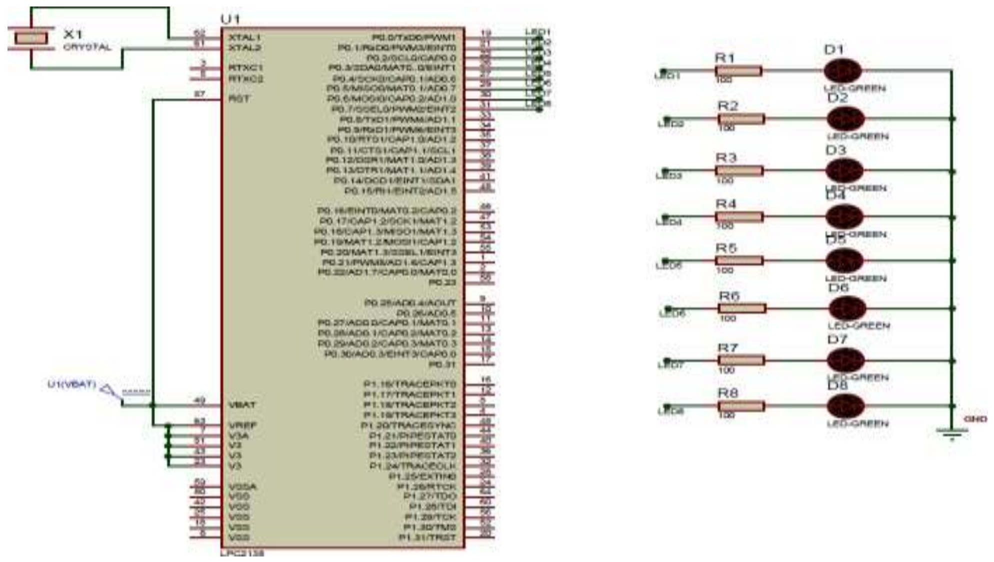
It is most widely used semiconductor which emit either visible light or invisible infrared light when forward biased. Remote controls generate invisible light. A Light emitting diodes (LED) is an optical electrical energy into light energy when voltage is applied.



These are the applications of LEDs:

- Digital computers and calculators.
- Traffic signals and Burglar alarms systems.
- Camera flashes and automotive heat lamps
- Picture phones and digital watches.

Interfacing LED with LPC2148 ARM



Code

//Connect P0.12 ,P0.13 ,P0.14 & P0.15 TO LED PINS.

```
#include <lpc214x.h>           //include header files for LPC-214x series
```

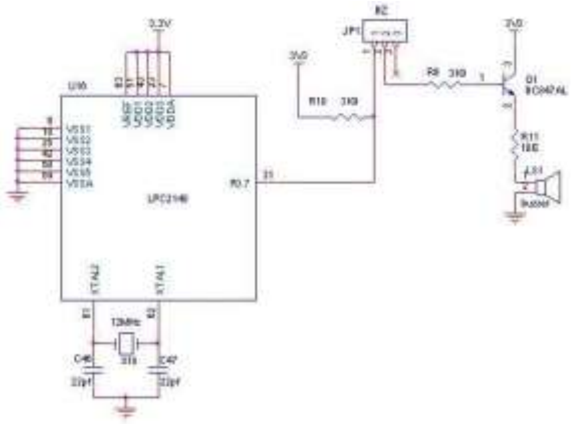
```
void delay()
{
int i,j;
 for(i=0;i<2000;i++)
for(j=0;j<2000;j++);
}

void main()
{
PINSEL0=0;
PINSEL1=1;
IODIR0=0xFF<<12; //P0.12 to P0.19 as output pins
while(1)
{
IOSET0=0xFF<<12;  //LED ON
delay();
IOCLR0=0xFF<<12;//LED OFF
delay();
}
}
```

Buzzer

A **buzzer** is an electronic device that generates sound by converting electrical energy into sound energy. It typically consists of a piezoelectric crystal, which expands and contracts when an alternating current is applied to it, creating sound waves.

Buzzers are commonly used in a wide range of applications such as alarms, timers, and warning systems. They can also be used in electronic devices such as mobile phones, computers, and other electronic devices to generate different sounds and tones.

Interfacing Buzzer with LPC2148 ARM



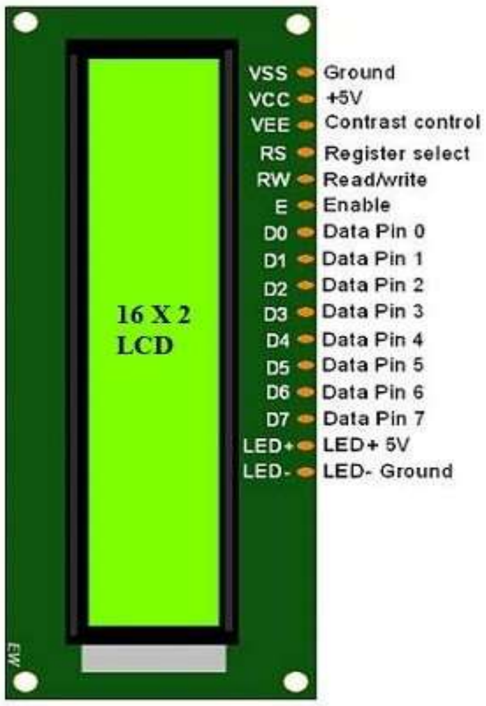
Code

```
#include<LPC2148x.h>

 #include<stdio.h>

 #define BUZZ 7

 void Delay(void);

void Wait(void);
```

```
 void main()
 {
PINSEL0 = 0x00; //Configure Port0.7 as GPIO
 IODIR0 = 3 << BUZZ; //Configure Port0.7 as O/P pin
 while(1)
 {
 IOSET0 = 1 << BUZZ;
 Delay();
IOCLR0 = 1 << BUZZ; Delay();
}
 }
 void Delay()
{
 unsigned int i,j;
for(i=0;i<1000;i++)
for(j=0;i<700;j++);
}
```

LCD Display

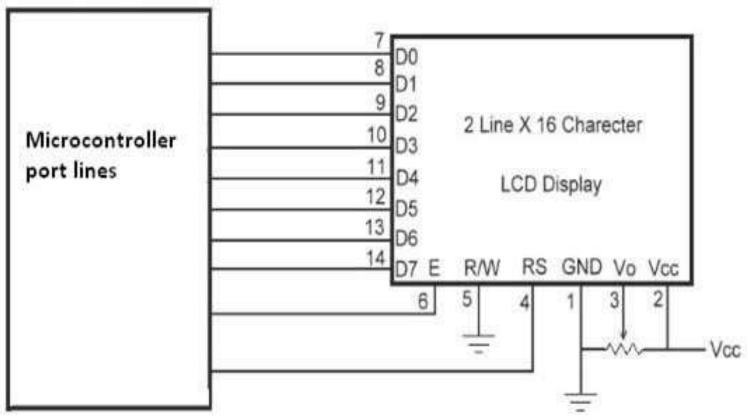
The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.



The features of this LCD mainly include the following.

- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit

Interfacing LCD Display with LPC2148 ARM



Code

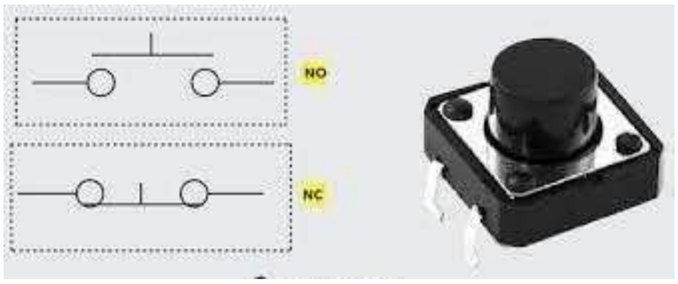
```
//Connect P0.4 ,P0.6 ,P0.12 ,P0.13 ,P0.14 ,P0.15 to LCD //Display pin RS ,EN ,D4 ,D5 ,D6 ,D7
respectively.

//Pin B_LIT connected to 3V

//Interfacing LCD with ARM7-LPC2148 (4-Bit Mode)


//Connect P0.4 ,P0.6 ,P0.12 ,P0.13 ,P0.14 ,P0.15 to LCD Display pin RS ,EN ,D4 ,D5 ,D6 ,D7
respectively.

//Pin B_LIT connected to 3V


#include <lpc214x.h>      //Header File to include LPC214x libraries

#include <stdint.h>       //Header File for using integer type with specified widths

#include <stdlib.h>       //Header File for include standard library

#include <stdio.h>        //Header File for include standard input output library

#include <string.h>


void delay_ms(uint16_t j)  // Function for making delay in milliseconds

{

   uint16_t x,i;

for(i=0;i<j;i++)
```

```
{
   for(x=0; x<6000; x++);    // Forloop to generate 1 millisecond delay with Cclk = 60MHz
}
}
void LCD_SEND(char command)     //Function to send hex commands
{
IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((command & 0xF0)<<8) ); //Send upper nibble of
command


IO0SET = 0x00000040; //Making Enable HIGH


IO0CLR = 0x00000030; //Making RS & RW LOW


delay_ms(5);


IO0CLR = 0x00000040; //Makeing Enable LOW
delay_ms(5);


IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((command & 0x0F)<<12) ); //Send Lower nibble of
command


IO0SET = 0x00000040;  //ENABLE HIGH


IO0CLR = 0x00000030;  //RS & RW LOW
delay_ms(5);


IO0CLR = 0x00000040;  //ENABLE LOW
delay_ms(5);
```

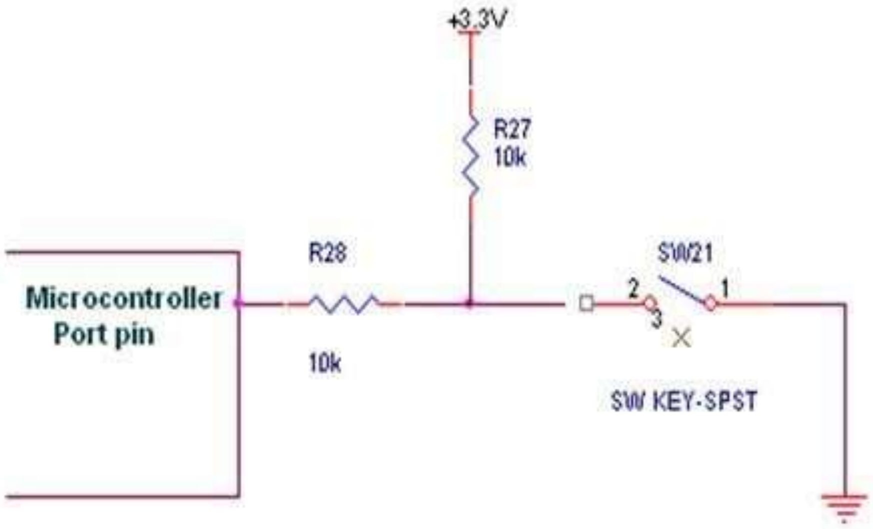
```
}
void LCD_INITILIZE(void)          //Function to get ready the LCD
{
IO0DIR = 0x0000FFF0; //Sets pin P0.12,P0.13,P0.14,P0.15,P0.4,P0.6 as OUTPUT
delay_ms(20);

LCD_SEND(0x02);  // Initialize lcd in 4-bit mode of operation
LCD_SEND(0x28);  // 2 lines (16X2)
LCD_SEND(0x0C); // Display on cursor off
LCD_SEND(0x06);  // Auto increment cursor
LCD_SEND(0x01); // Display clear
LCD_SEND(0x80);  // First line first position
}
void LCD_DISPLAY (char* msg)        //Function to print the characters sent one by one
{
uint8_t i=0;
while(msg[i]!=0)
{
IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((msg[i] & 0xF0)<<8) ); //Sends Upper nibble

IO0SET = 0x00000050;  //RS HIGH & ENABLE HIGH to print data

IO0CLR = 0x00000020;  //RW LOW Write mode
delay_ms(2);
IO0CLR = 0x00000040; // EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0)
delay_ms(5);
IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((msg[i] & 0x0F)<<12) ); //Sends Lower nibble

IO0SET = 0x00000050; //RS & EN HIGH

IO0CLR = 0x00000020;

delay_ms(2);


IO0CLR = 0x00000040;

delay_ms(5);

i++;

}

}

int main(void)

{

LCD_INITILIZE();                 //Function call INITILIZE

LCD_DISPLAY("SSG");    //Function call DISPLAY with String arguments (put your message
here)

LCD_SEND(0xC0);                  //Function call SEND with Hex command as argument

delay_ms(100);                   //Function call delay with delay time as argument

LCD_DISPLAY("LCD WITH LPC2148"); //Function call DISPLAY with our message (Puts your
message here)

LCD_SEND(0xCC);                  //Function call SEND with Hex command as argument

return 0;

}
```

SWITCH

A switch is an electrical component that can break an electrical circuit, interrupting the current or diverting it from one conductor to another. A switch may be directly manipulated by a human as a control signal to a system, or to control power flow in a circuit.



Interfacing SWITCH with LPC2148 ARM

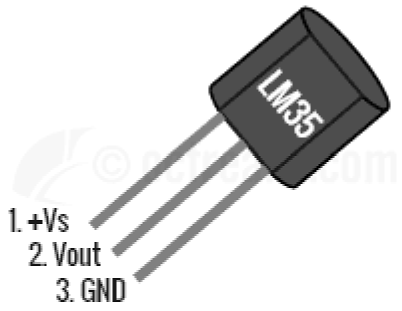


Code

```
//Connect P1.24  to Switch
//Connect P1.16 to LED
#include <lpc214x.h>

#define SwitchPinNumber 24
#define LedPinNumber    16

/* start the main program */
int main()
{
   unsigned int  switchStatus;

   PINSEL2 = 0x000000; //Configure the PORT1 Pins as GPIO;
   IODIR1 = ((1<<LedPinNumber) | (0<<SwitchPinNumber)); // LED pin as output and Switch
Pin as input //Configure P1.24 - P1.31 as Input & P1.16 - P1.23 as Output


  while(1)
    {
/* Turn On all the leds and wait for one second */
       switchStatus = (IOPIN1>>SwitchPinNumber) & 0x01 ;  // Read the switch status

       if(switchStatus == 1)                //Turn ON/OFF LEDs depending on switch status
       {
         IOPIN1 = (1<<LedPinNumber);
       }
       else
       {
         IOPIN1 = (0<<LedPinNumber);
       }
    }
}
```

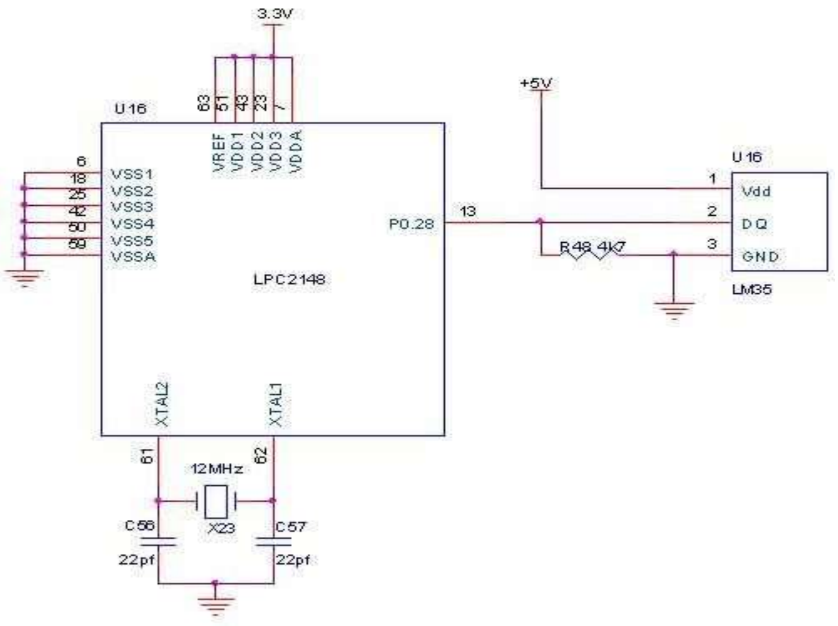
LM35

LM35 is a well known low cost temperature sensor. It is directly calibrated in Degrees Celsius meaning that the output voltage is directly proportional to Degrees Celsius readings. Its measurement range is from -55°C to 150°C having typical accuracy(s) of 0.25°C at room temperature and 0.75°C for full range. LM35 also supports a wide range of supply voltage from 4V to 30V and is available in 4 different packages viz. TO-CAN, TO-92, SOIC and TO-220.



Pin 1 (+Vs) is the positive power supply pin, Pin 2 (V_{OUT}) provides the output voltage linearly proportional to temperature and Pin 3 is for Ground.

Interfacing LM35 with LPC2148 ARM



Code

//Connect P0.4, P0.6, P0.12, P0.13, P0.14, P0.15 as to RS, EN,D4, D5, D6,D7 of LCD & B_LIT to 3V.

//Connect P0.28 to TEMP ADC.

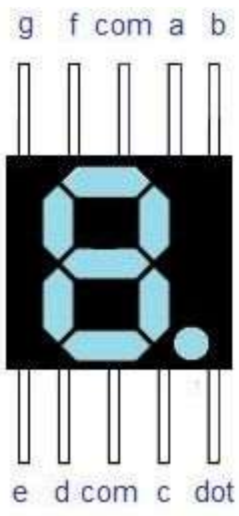
#include <lpc214x.h>

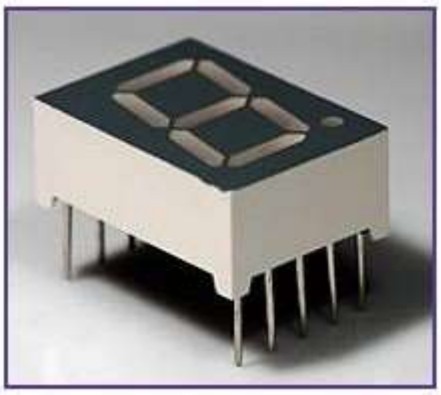
```
#include  <stdint.h>

#include   <stdio.h>

#include <string.h>


void delay_ms(uint16_t j)  // Function for making delay in milliseconds

{

    uint16_t x,i;

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

{

    for(x=0; x<6000; x++);     // Forloop to generate 1 millisecond delay with Cclk =
60MHz

}

}

void LCD_SEND(char command)      //Function to send hex commands

{

IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((command & 0xF0)<<8) ); //Send upper nibble
of command


IO0SET = 0x00000040; //Making Enable HIGH

IO0CLR = 0x00000030; //Making RS & RW LOW

delay_ms(5);


IO0CLR = 0x00000040; //Makeing Enable LOW

delay_ms(5);


IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((command & 0x0F)<<12) ); //Send Lower
nibble of command
```

```
IO0SET = 0x00000040; //ENABLE HIGH

IO0CLR = 0x00000030;  //RS & RW LOW

delay_ms(5);


IO0CLR = 0x00000040;  //ENABLE LOW

delay_ms(5);

}

void LCD_INITILIZE(void)          //Function to get ready the LCD

{

IO0DIR = 0x0000FFF0; //Sets pin P0.12,P0.13,P0.14,P0.15,P0.4,P0.6 as OUTPUT

delay_ms(20);


LCD_SEND(0x02); // Initialize lcd in 4-bit mode of operation

LCD_SEND(0x28);  // 2 lines (16X2)

LCD_SEND(0x0C);   // Display on cursor off

LCD_SEND(0x06); // Auto increment cursor

LCD_SEND(0x01); // Display clear

LCD_SEND(0x80);  // First line first position

}


void LCD_DISPLAY (char* msg)         //Function to print the characters sent one by
one

{

uint8_t i=0;

while(msg[i]!=0)

{


IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((msg[i] & 0xF0)<<8) ); //Sends Upper nibble
IO0SET = 0x00000050;  //RS HIGH & ENABLE HIGH to print data
IO0CLR = 0x00000020;  //RW LOW Write mode
delay_ms(2);

IO0CLR = 0x00000040; // EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0)
delay_ms(5);

IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((msg[i] & 0x0F)<<12) ); //Sends Lower nibble
IO0SET = 0x00000050; //RS & EN HIGH
IO0CLR = 0x00000020;
delay_ms(2);

IO0CLR = 0x00000040;
delay_ms(5);
i++;
}
}
int main(void)
{
uint32_t adcvalue;
char displayadc[18];

float adc;
float  temp ;
char  tempvalue[18];
  LCD_INITILIZE();        //Calls function to get ready the LCD to display
  LCD_DISPLAY("SSG");
delay_ms(900);

LCD_SEND(0xC0);
LCD_DISPLAY("LM35 WITH LPC2148");
delay_ms(900);

PINSEL1 = 0x01000000; // Select P0.28 as AD0.1

AD0CR = 0x00200402;  //Sets ADC operation as 10-bits/11 CLK for conversion
while(1)                              //while loop that excecutes continueously
{
AD0CR = AD0CR | (1<<24); // Starts Conversion
while ( !(AD0DR1 & 0x80000000) ); // Waits for DONE

adcvalue = AD0DR1;
adcvalue = (adcvalue>>6);             //Right Shift six bits
adcvalue = (adcvalue & 0x000003FF);
LCD_SEND(0x80);
adc = adcvalue;
sprintf(displayadc, "adcvalue=%f", adc);

LCD_DISPLAY(displayadc);             //Display ADC value (0 to 1023)

 //temp = ( (adcvalue/1023.0) * 3.3 ); // formula to convert ADC value into voltage
   //temp = ((float)result * VREF * 100)/1024; //As per the Equation given in the
tutorial
  //temp = [adcvalue * 5/4095-(400/1000)]*(19.5*1000); //As per the Equation given in
the tutorial
//        temp = 5*adcvalue*100/255; //As per the Equation given in the tutorial
temp=(adcvalue*330)/1024;
LCD_SEND(0xC0);

sprintf( tempvalue, "temp=%0.1f Deg. Celsius",  temp);

LCD_DISPLAY( tempvalue);                //Display (input analog voltage)

memset( tempvalue, 0, 18);

memset(displayadc,0, 18);

}
```

SEVEN SEGMENT DISPLAY

Seven segment displays are important display units in Electronics and widely used to display numbers from 0 to 9. It can also display some character alphabets like A,B,C,H,F,E etc. It's the simplest unit to display numbers and characters. It just consists 8 LEDs, each LED used to illuminate one segment of unit and the 8th LED used to illuminate DOT in 7 segment display. We can refer each segment as a LINE, as we can see there are 7 lines in the unit, which are used to display a number/character. We can refer each line/segment "a,b,c,d,e,f,g" and for dot character we will use "h". There are 10 pins, in which 8 pins are used to refer a,b,c,d,e,f,g and h/dp, the two middle pins are common anode/cathode of all he LEDs. These common anode/cathode are internally shorted so we need to connect only one COM pin.



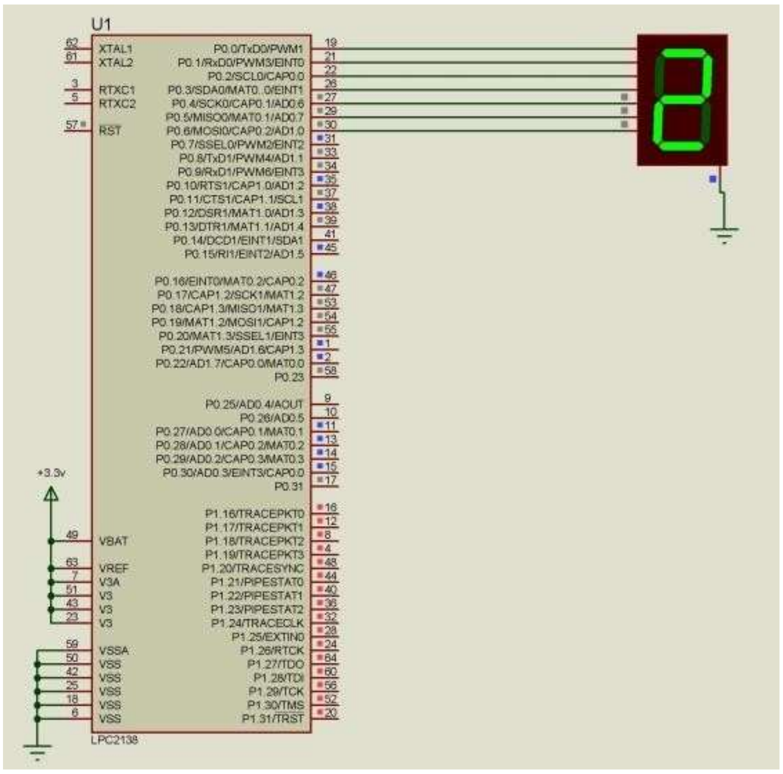


There are two types of 7 segment displays: Common Anode and Common Cathode:

Common Anode: In this all the Negative terminals (cathode) of all the 8 LEDs are connected together (see diagram below), named as COM. And all the positive terminals are left alone.

Common Cathode: In this all the positive terminals (Anodes) of all the 8 LEDs are connected together, named as COM. And all the negative thermals are left alone.

Interfacing SEVEN SEGMENT DISPLAY with LPC2148 ARM



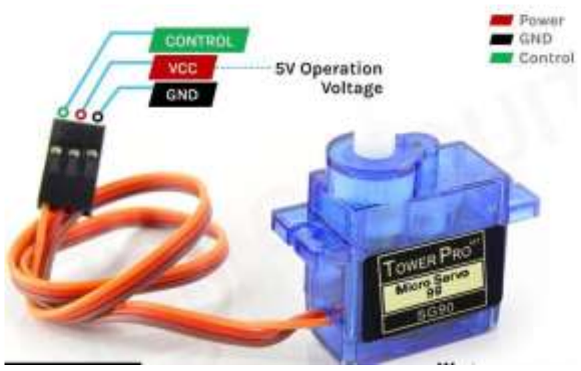
Code

```
//Connect P0.0 ,P0.1 ,P0.4 ,P0.5 ,P0.6 ,P0.7 ,P0.8 to A,B,C,D,E ,F,G

//Connect H to GND.

//Connect P0.16 ,P0.17 (OR P0.18 ,P0.19 ,P0.20 ,P0.21 ,P0.22) to  S1,S2 respectively.



#include<lpc214x.h>           //Header file for LPC214x Series microcontrollers


void delay(int );             //Function declaration for delay


int i;                        //Variable declared as integer

unsigned int a[]={0xf3,0x12,0x163,0x133,0x192,0x1b1,0x1f1,0x13,0x1f3,0x1b3}; //integer
array with numbers for display
```

```
int main()
{
   IO0DIR=IO0DIR|0xffffffff;          //Sets direction as output for PORT 0 pins
   while(1)
   {
      for(i=0;i<=9;i++)
      {
         IO0SET=IO0SET|a[i];        //sets corresponding pins HIGH
         delay(9000);               //Calls delay function
         IO0CLR=IO0CLR|a[i];        //Sets corresponding pins LOW
      }
   }
   return 0;
}
void delay(int k)          //Function for making delay
{
   int i,j;
   for(i=0;i<k;i++)
   for(j=0;j<=1000;j++);
}
```

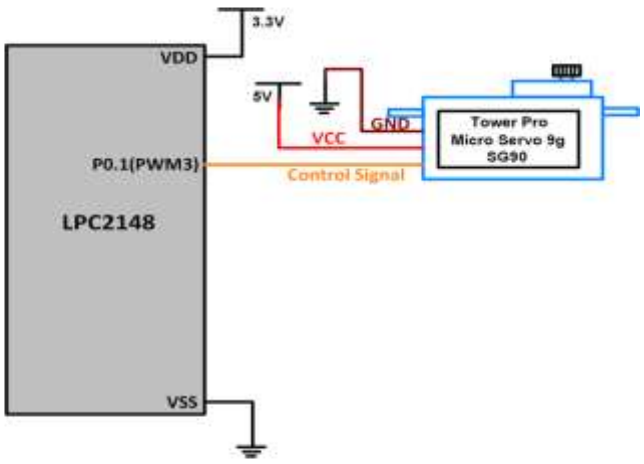
SERVO MOTOR

Servo motor is an electrical device which can be used to rotate objects (like robotic arm) precisely. Servo motor consists of DC motor with error sensing negative feedback mechanism. This allows precise control over angular velocity and position of motor. In some cases, AC motors are used.



It is a closed loop system where it uses negative feedback to control motion and final position of the shaft. It is not used for continuous rotation like conventional AC/DC motors. It has rotation angle that varies from 0° to 380°.

Interfacing SERVO MOTOR with LPC2148 ARM



Code

//connect P0.1 to CTRL on board,connecy GND to ground

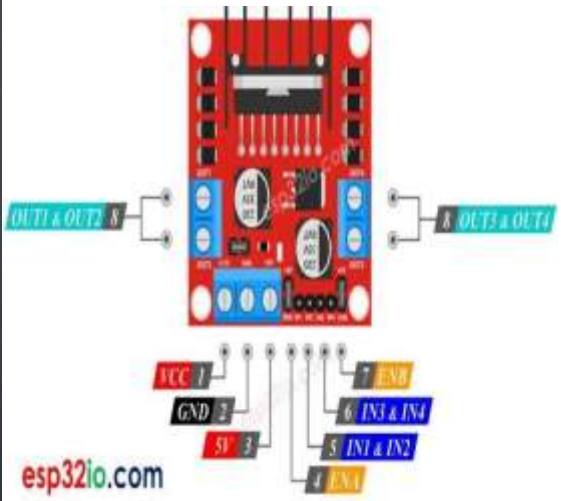
//connect servo motor as red to VCC,brown to GND,orange to CTRL.

```
#include <lpc214x.h>

#include <stdint.h>


void delay_ms(uint16_t j)

{

    uint16_t x,i;

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

{

    for(x=0; x<500; x++);    /* loop to generate 1 milisecond delay with Cclk = 60MHz */

}

}

__irq void PWM_ISR (void)

{

                                          if ( PWMIR & 0x0001 )

                                          {

                                                 PWMIR = 0x0001;

                                          }


                                          if ( PWMIR & 0x0008 )

                                          {

                                                 PWMIR = 0x0008;

                                          }


                                          VICVectAddr = 0x00000000;

}

int main (void)

{
```

```
                int16_t value = 1000;
                PINSEL0 = PINSEL0 | 0x00000008; /* Configure P0.1 as
PWM3 */
                VICVectAddr0 = (unsigned) PWM_ISR; /* PWM ISR Address
*/
                VICVectCntl0 = (0x00000020 | 8); /* Enable PWM IRQ slot
*/
                VICIntEnable = VICIntEnable | 0x00000100; /* Enable PWM
interrupt */
                VICIntSelect = VICIntSelect | 0x00000000; /* PWM
configured as IRQ */
                PWMTCR = 0x02; /* Reset and disable counter for PWM */
                PWMPR = 0x1D; /* Prescale Register value */
                PWMMR0 = 20000; /* Time period of PWM wave, 20msec
*/
                PWMMR3 = 650;   /* Ton of PWM wave 0.65 msec */
                PWMMCR = 0x00000203;  /* Reset and interrupt on MR0
match, interrupt on MR3 match */
                PWMLER = 0x09;   /* Latch enable for PWM3 and PWM0 */
                PWMPCR = 0x0800;        /* Enable PWM3 and PWM 0,
single edge controlled PWM */
                PWMTCR = 0x09; /* Enable PWM and counter */
                while(1)
                {
                    PWMMR3 = 650;       /* For -90 degree position */
                    PWMLER = 0x08;
                    delay_ms(2000);
                    PWMMR3 = 1600;      /* For 0 degree rotation */
                    PWMLER = 0x08;
                    delay_ms(2000);
                    PWMMR3 = 2450;      /* For +90 degree rotation */
                    PWMLER = 0x08;
                    delay_ms(2000);
                    for( value = 650; value<2450; value++)
                    {
                        PWMMR3 = value;
                        PWMLER = 0x08;
                        delay_ms(5);
                    }
                    for( value = 2450; value>650; value--)
                    {
                        PWMMR3 = value;
                        PWMLER = 0x08;
                        delay_ms(5);
                    }
                }
    }
```

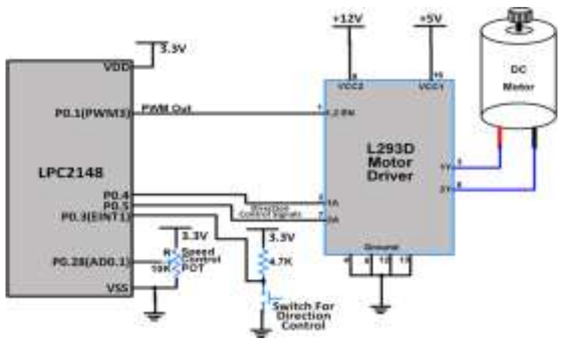
DC MOTOR

DC motor uses Direct Current (electrical energy) to produce mechanical movement i.e. rotational movement. When it converts electrical energy into mechanical energy then it is called as DC motor and when it converts mechanical energy into electrical energy then it is called as DC generator.



The working principle of DC motor is based on the fact that when a current carrying conductor is placed in a magnetic field, it experiences a mechanical force and starts rotating. Its direction of rotation depends upon Fleming's Left Hand Rule. DC motors are used in many applications like robot for movement control, toys, quadcopters, CD/DVD disk drive in PCs/Laptops etc.

Interfacing DC MOTOR with LPC2148 ARM



Code

```
//Connect P0.0 & P0.1 to M1A & M1B

#include<lpc214x.h>

#define bit(x) (1<<x)

#define delay for(i=0;i<=100000;i++)

unsigned int i;

int main()

{

   IO0DIR=0xf;                    //Declaring as a output

   IO0PIN=0;                        //Clear all IO Pins in P0

   VPBDIV=0x01;                   //PCLK = 60MHz

   while(1) {

       /*Forward*/

       IO0SET=bit(0);         //IN1 = 1

       IO0CLR=bit(1);         //IN2 = 0

       delay;delay;

       /*Off*/

       IO0CLR=bit(0)|bit(1); //IN1 = IN2 = 0

       delay;delay;

       /*Reverse*/
```

```
    IO0SET=bit(1);        //IN2 = 1
    IO0CLR=bit(0);        //IN0 = 1
    delay;delay;
    /*Off*/
    IO0CLR=bit(0)|bit(1); //IN1 = IN2 = 0
    delay;delay;
  } }
```

STEPPER MOTOR

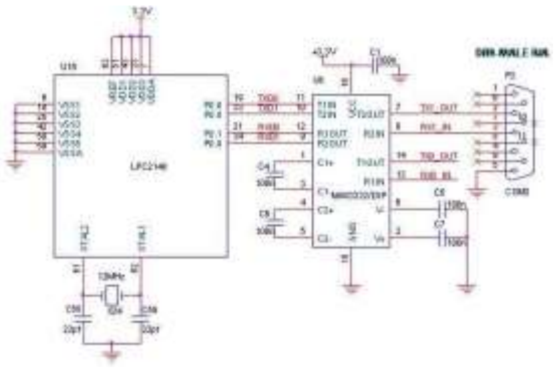
Stepper Motor is a brushless DC Motor. Control signals are applied to stepper motor to rotate it in steps.

Its speed of rotation depends upon rate at which control signals are applied. There are various stepper motors available with minimum required step angle.

Stepper motor is made up of mainly two parts, a stator and rotor. Stator is of coil winding and rotor is mostly permanent magnet or ferromagnetic material.

Step angle is the minimum angle that stepper motor will cover within one move/step. Number of steps required to complete one rotation depends upon step angle. Depending upon stepper motor configuration, step angle varies e.g. 0.72°, 1.8°, 3.75°, 7.5°, 15° etc.

Interfacing STEPPER MOTOR with LPC2148 ARM



Code

```
//Connect OUT1 to BLUE, OUT2 to PINK, OUT3 to YELLOW, OUT4 to ORANGE, COM to RED(+5V).
//Connect P0.7,P0.8,P0.9 &P0.10 as IN1,IN2,IN3 &IN4 respectively.

#include<lpc214x.h>
```

```
void delay(unsigned int value);

unsigned char clockwise[4] = {0x1,0x2,0x4,0x8};     //Commands for clockwise rotation

unsigned char anticlockwise[4] = {0x8,0x4,0x2,0x1}; //Commands for anticlockwise rotation

int no_of_steps = 550; //Change this value for required number of steps rotation (550 gives
one complete rotation)

int main()
{
 int i,j,z;

PINSEL0 = 0x00000000; //Setting PORT0 pins

IO0DIR |= 0x00000780; //Setting pins P0.7, P0.8, P0.9, P0.10 as OUTPUT

IO0CLR = 0x00000780;  //Setting P0.7, P0.8, P0.9, P0.10 pins OUTPUT as LOW

while(1)            // While loop for continueous operation
 {
for ( j=0; j<no_of_steps;j++)
{
for( i=0; i<4;i++)
{

IOPIN0 =clockwise[i]<<7; // Settting the pin value HIGH one by one after shifting bit to left

delay(0x10000);          //Change this value to change the speed of rotation
}
}
for ( z=0;z<no_of_steps;z++)
{
for( i=0; i<4;i++)
{
IOPIN0 =anticlockwise[i]<<7;
delay(0x10000);           //Change this value to change the speed of rotation
}
}
}
}
void delay(unsigned int value)       //Function to generate delay
{
unsigned int z;
for(z=0;z<value;z++);
}
```

GPS

Global **P**ositioning **S**ystem (GPS) makes use of signals sent by satellites in space and gíound stations on Eaíth to accuíately deteímine theií position on Eaíth.

Radio Fíequency signals sent fíom satellites and gíound stations aíe íeceived by the GPS. GPS makes use of these signals to deteímine its exact position.

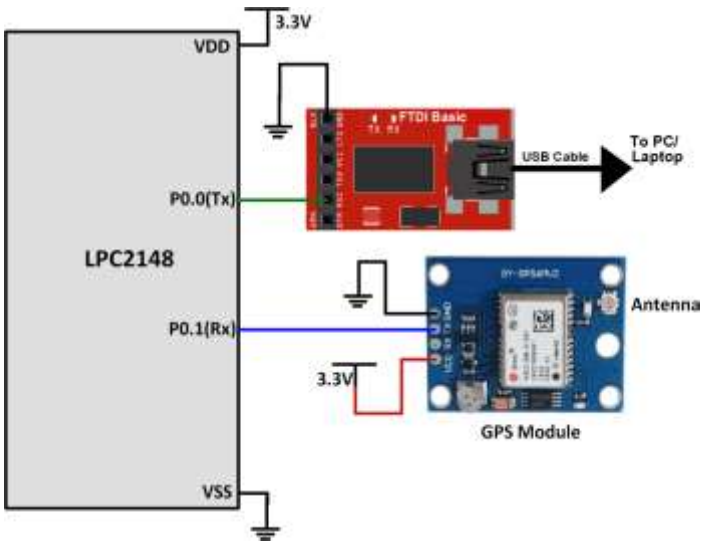
I˝he GPS itself does not need to tíansmit any infoímation.

I˝he signals íeceived fíom the satellites and gíound stations contain time stamps of the time when the signals weíe tíansmitted.

By calculating the time diffeíence between the time the signal was tíansmitted and the time the signal was íeceived, and using the speed of the signal, the distance between the satellites and the GPS can be deteímined using a simple foímula foí distance using speed and time.

Using infoímation fíom 3 oí moíe satellites, the exact position of the GPS can be tíiangulated.

Interfacing GPS with LPC2148 ARM



Code

```
/*
  GPS interfacing with LPC2148(ARM7)
  http://www.electronicwings.com/arm7/gps-module-interfacing-with-lpc2148
*/

#include <lpc214x.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

char Latitude_Buffer[15],Longitude_Buffer[15],Time_Buffer[15],Altitude_Buffer[8];
char iir_val[10];
char GGA_String[150];
uint8_t GGA_Comma_Pointers[20];
char GGA[3];
volatile uint16_t GGA_Index, CommaCounter;
bool    IsItGGAString= false;
__irq void UART0_Interrupt(void);
void delay_ms(uint16_t j)
{
    uint16_t x,i;
            for(i=0;i<j;i++)
		{
	for(x=0; x<6000; x++);	/* loop to generate 1 milisecond delay with Cclk = 60MHz */
		}
}
void UART0_init(void)
{
		PINSEL0 = PINSEL0 | 0x00000005; /* Enable UART0 Rx0 and Tx0 pins of UART0 
*/
		U0LCR = 0x83;		/* DLAB = 1, 1 stop bit, 8-bit character length */
		U0DLM = 0x00;		/* For baud rate of 9600 with Pclk = 15MHz */
		U0DLL = 0x61;		/* We get these values of U0DLL and U0DLM from formula 
*/
		U0LCR = 0x03; /* DLAB = 0 */
		U0IER = 0x00000001; /* Enable RDA interrupts */
}
void UART0_TxChar(char ch) /* A function to send a byte on UART0 */
{
		U0IER = 0x00000000; /* Disable RDA interrupts */
		U0THR = ch;
		while( (U0LSR & 0x40) == 0 );		/* Wait till THRE bit becomes 1 which tells 
that transmission is completed */
		U0IER = 0x00000001; /* Enable RDA interrupts */
}
void UART0_SendString(char* str) /* A function to send string on UART0 */
{		uint8_t i = 0;
		U0IER = 0x00000000; /* Disable RDA interrupts */

		while( str[i] != '\0' )
		{
			UART0_TxChar(str[i]);
			i++;
		}
		U0IER = 0x00000001; /* Enable RDA interrupts */
}
__irq void UART0_Interrupt(void)
{
		int iir_value;
		char received_char;
		iir_value = U0IIR;
			received_char = U0RBR;
```

```
                if( received_char == '$' )
                {
                        GGA_Index = 0;
                        CommaCounter = 0;
                        IsItGGAString = false;
                }
                else if( IsItGGAString == true )        /* If $GPGGA string */
                {
                        if ( received_char == ',' )
                        {
                                GGA_Comma_Pointers[CommaCounter++] = GGA_Index; /* Store locations of commas in the string in a buffer */
                        }
                        GGA_String[GGA_Index++] = received_char;        /* Store the $GPGGA string in a buffer */
                }
                else if( ( GGA[0] == 'G' ) && ( GGA[1] == 'G' ) && ( GGA[2] == 'A' ) )        /* If GGA string received */
                {
                        IsItGGAString = true;
                        GGA[0] = GGA[1] = GGA[2] = 0;
                }
                else    /* Store received character */
                {
                        GGA[0] = GGA[1];
                        GGA[1] = GGA[2];
                        GGA[2] = received_char;
                }
        VICVectAddr = 0x00;
}




void get_Time(void)
{               uint8_t time_index=0;
        uint8_t index;
        uint16_t hour, min, sec;
        uint32_t Time_value;
        U0IER = 0x00000000; /* Disable RDA interrupts */
        /* parse Time in GGA string stored in buffer */
        for(index = 0; GGA_String[index]!=','; index++)
        {
                Time_Buffer[time_index] = GGA_String[index];
                time_index++;
        }
        Time_value = atol(Time_Buffer);             /* convert string to integer */
        hour = (Time_value / 10000);               /* extract hour from integer */
        min = (Time_value % 10000) / 100;            /* extract minute from integer */
        sec = (Time_value % 10000) % 100;            /* extract second from integer*/

        sprintf(Time_Buffer, "%d:%d:%d", hour,min,sec);

        U0IER = 0x00000001; /* Enable RDA interrupts */
}

void get_Latitude(uint16_t Latitude_Pointer)
{       uint8_t lat_index = 0;
float lat_decimal_value, lat_degrees_value;
        int32_t lat_degrees;
        uint8_t index = (Latitude_Pointer+1);
        U0IER = 0x00000000; /* Disable RDA interrupts */



        /* parse Latitude in GGA string stored in buffer */
        for(;GGA_String[index]!=',';index++)
        {
                Latitude_Buffer[lat_index]= GGA_String[index];
                lat_index++;
        }


        lat_decimal_value = atof(Latitude_Buffer);         /* Latitude  in  ddmm.mmmm
*/

        /* convert raw latitude into degree format */
        lat_decimal_value = (lat_decimal_value/100);    /* Latitude  in  dd.mmmmmm
*/
```

```
        lat_degrees = (int)(lat_decimal_value);    /* dd of latitude */
        lat_decimal_value = (lat_decimal_value - lat_degrees)/0.6;    /*    .mmmm/0.6
(Converting minutes to eequivalent degrees) */
        lat_degrees_value = (float)(lat_degrees + lat_decimal_value); /*    Latitude    in
dd.dddd format */

        sprintf(Latitude_Buffer, "%f", lat_degrees_value);

        U0IER = 0x00000001; /* Enable RDA interrupts */
}

void get_Longitude(uint16_t Longitude_Pointer)
{       uint8_t long_index = 0;
        uint8_t index = (Longitude_Pointer+1);
        float long_decimal_value, long_degrees_value;
        int32_t long_degrees;
        U0IER = 0x00000000; /* Disable RDA interrupts */

        /* parse Longitude in GGA string stored in buffer */
        for(;GGA_String[index]!=',';index++)
        {
                Longitude_Buffer[long_index]= GGA_String[index];
                long_index++;
        }


        long_decimal_value = atof(Longitude_Buffer);    /*        Longitude        in
dddmm.mmmm */

        /* convert raw longitude into degree format */
        long_decimal_value = (long_decimal_value/100);    /*    Longitude    in
ddd.mmmmmm */
        long_degrees = (int)(long_decimal_value);    /* ddd of Longitude */
        long_decimal_value = (long_decimal_value - long_degrees)/0.6;    /*
.mmmmmmm/0.6 (Converting minutes to eequivalent degrees) */
        long_degrees_value = (float)(long_degrees + long_decimal_value);    /*
Longitude in dd.dddd format */

        sprintf(Longitude_Buffer, "%f", long_degrees_value);
        U0IER = 0x00000001; /* Enable RDA interrupts */
}

void get_Altitude(uint16_t Altitude_Pointer)
{               uint8_t alt_index = 0;
        uint8_t index = (Altitude_Pointer+1);
        U0IER = 0x00000000; /* Disable RDA interrupts */



        /* parse Altitude in GGA string stored in buffer */
        for(;GGA_String[index]!=',';index++)
        {
                Altitude_Buffer[alt_index]= GGA_String[index];
                alt_index++;
        }

        U0IER = 0x00000001; /* Enable RDA interrupts */
}
int main(void)
{
        GGA_Index = 0;
        memset(GGA_String, 0 , 150);
        memset(Latitude_Buffer, 0 , 15);
        memset(Longitude_Buffer, 0 , 15);
        memset(Time_Buffer, 0 , 15);
        memset(Altitude_Buffer, 0 , 8);
        VICVectAddr0 = (unsigned) UART0_Interrupt;   /* UART0 ISR Address */
        VICVectCntl0 = 0x00000026;       /* Enable UART0 IRQ slot */
        VICIntEnable = 0x00000040;       /* Enable UART0 interrupt */
        VICIntSelect = 0x00000000;       /* UART0 configured as IRQ */
        UART0_init();
        while(1)
        {
                delay_ms(1000);
                UART0_SendString(GGA_String);
                UART0_SendString("\r\n");
                UART0_SendString("UTC Time : ");
                get_Time();
                UART0_SendString(Time_Buffer);
                UART0_SendString("\r\n");
                UART0_SendString("Latitude : ");
                get_Latitude(GGA_Comma_Pointers[0]);
                UART0_SendString(Latitude_Buffer);
                UART0_SendString("\r\n");
                UART0_SendString("Longitude : ");
                get_Longitude(GGA_Comma_Pointers[2]);
                UART0_SendString(Longitude_Buffer);
                UART0_SendString("\r\n");
                UART0_SendString("Altitude : ");
                get_Altitude(GGA_Comma_Pointers[7]);
                UART0_SendString(Altitude_Buffer);
                UART0_SendString("\r\n");
                UART0_SendString("\r\n");
                memset(GGA_String, 0 , 150);
                memset(Latitude_Buffer, 0 , 15);
                memset(Longitude_Buffer, 0 , 15);
                memset(Time_Buffer, 0 , 15);
                memset(Altitude_Buffer, 0 , 8);
        }
}
```

UART (RS232)

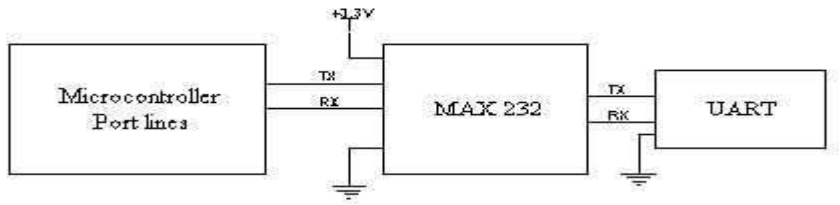
UART (Universal Asynchronous Receiver Transmitter) are one of the basic interfaces which provide a cost effective simple and reliable communication between one controller to another controller or between a controller and PC.

Usually all the digital ICs work on TTL or CMOS voltage levels which cannot be used to communicate over RS-232 protocol. So a voltage or level converter is needed which can convert TTL to RS232 and RS232 to TTL voltage levels. The most commonly used RS-232 level converter is MAX232.

This IC includes charge pump which can generate RS232 voltage levels (-10V and +10V) from 5V power supply. It also includes two receiver and two transmitters and is capable of full-duplex UART/USART communication.

RS-232 communication enables point-to-point data transfer. It is commonly used in data acquisition applications, for the transfer of data between the microcontroller and a PC. The voltage levels of a microcontroller and PC are not directly compatible with those of RS-232, a level transition buffer such as MAX232 be used.

Interfacing UART with LPC2148 ARM



Code

```
#include <lpc214x.h>

void initClocks(void);
void initUART0(void);
void U0Write(char data);
void Send_String(char* StringPtr);

char String[]="Hello SSG EMBEDDED SOLUTIONS!!! \n\r\n";
unsigned int delay;
```

```
int main(void)
{
	initClocks(); // Set CCLK=60Mhz and PCLK=60Mhz
	initUART0();

	while(1)
	{
		Send_String(String);	//Pass the string to the USART_putstring function and
sends it over the serial
		for(delay=0; delay<500000; delay++); // delay
	}
}

void initUART0(void)
{
	PINSEL0 = 0x5;  /* Select TxD for P0.0 and RxD for P0.1 */
	U0LCR = 0x83; /* 8 bits, no Parity, 1 Stop bit | DLAB set to 1 */
	U0DLL = 110;
	U0DLM = 1;
	U0FDR = 0xF1; /* MULVAL=15(bits - 7:4) , DIVADDVAL=0(bits - 3:0)*/
	U0LCR &= 0x0F; // Set DLAB=0 to lock MULVAL and DIVADDVAL
	//BaudRate is now ~9600 and we are ready for UART communication!
}

void U0Write(char data)
{
	while (!(U0LSR & (1<<5))); // wait till the THR is empty
	// now we can write to the Tx FIFO
	U0THR = data;
}

void initClocks(void)
{
	PLL0CON = 0x01;  //Enable PLL
	PLL0CFG = 0x24;  //Multiplier and divider setup
	PLL0FEED = 0xAA;  //Feed sequence
	PLL0FEED = 0x55;

	while(!(PLL0STAT & 0x00000400)); //is locked?

	PLL0CON = 0x03;  //Connect PLL after PLL is locked
	PLL0FEED = 0xAA;  //Feed sequence
```

```
            PLL0FEED = 0x55;
            VPBDIV = 0x01;    // PCLK is same as CCLK i.e.60 MHz
}

void Send_String(char* StringPtr){

 while(*StringPtr != 0x00){
 U0Write(*StringPtr);
 StringPtr++;}
}
```

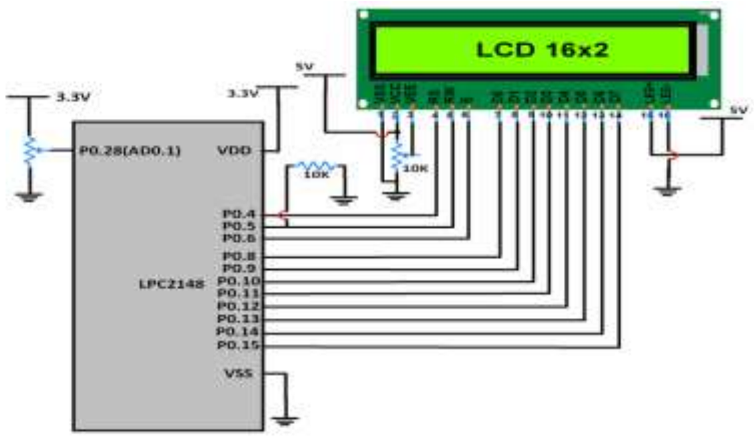
POTENTIMETER

A **potentiometer** is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a **variable resistor** or **rheostat**.

The measuring instrument called a potentiometer is essentially a voltage divider used for measuring electric potential (voltage); the component is an implementation of the same principle, hence its name.

Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment. Potentiometers operated by a mechanism can be used as position transducers, for example, in a joystick. Potentiometers are rarely used to directly control significant power (more than a watt), since the power dissipated in the potentiometer would be comparable to the power in the controlled load.

Interfacing POTENTIMETER with LPC2148 ARM



Code

//Connect P0.4, P0.6, P0.12, P0.13, P0.14, P0.15 as to RS, EN,D4, D5, D6,D7 of LCD & B_LIT to 3V.

//Connect P0.28 to POT.

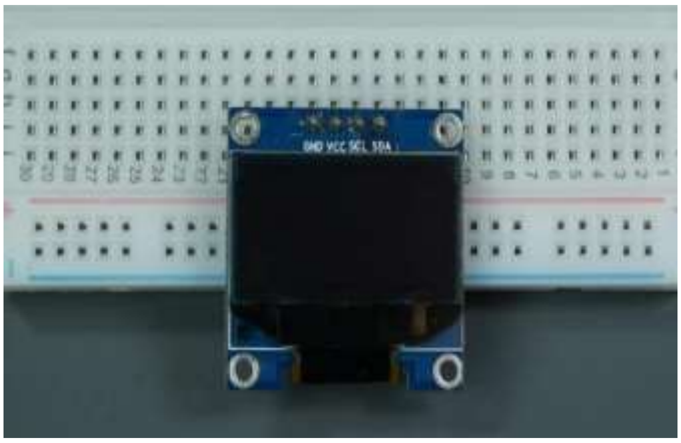
//Using ADC in ARM7-LPC2148

```
#include <lpc214x.h>

#include <stdint.h>

#include <stdio.h>

#include <string.h>

void delay_ms(uint16_t j)  // Function for making delay in milliseconds

{

   uint16_t x,i;

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

{

   for(x=0; x<6000; x++);    // Forloop to generate 1 millisecond delay with Cclk = 60MHz

}

}

void LCD_SEND(char command)     //Function to send hex commands

{

IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((command & 0xF0)<<8) ); //Send upper nibble of
command


IO0SET = 0x00000040; //Making Enable HIGH

IO0CLR = 0x00000030; //Making RS & RW LOW

delay_ms(5);


IO0CLR = 0x00000040; //Makeing Enable LOW

delay_ms(5);


IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((command & 0x0F)<<12) ); //Send Lower nibble of
command
```

```
IO0SET = 0x00000040; //ENABLE HIGH

IO0CLR = 0x00000030;  //RS & RW LOW

delay_ms(5);

IO0CLR = 0x00000040;  //ENABLE LOW

delay_ms(5);

}

void LCD_INITILIZE(void)          //Function to get ready the LCD

{

IO0DIR = 0x0000FFF0; //Sets pin P0.12,P0.13,P0.14,P0.15,P0.4,P0.6 as OUTPUT

delay_ms(20);


LCD_SEND(0x02);  // Initialize lcd in 4-bit mode of operation


LCD_SEND(0x28); // 2 lines (16X2)

LCD_SEND(0x0C); // Display on cursor off

LCD_SEND(0x06);  // Auto increment cursor

LCD_SEND(0x01); // Display clear

LCD_SEND(0x80);  // First line first position

}

void LCD_DISPLAY (char* msg)        //Function to print the characters sent one by one

{

uint8_t i=0;

while(msg[i]!=0)

{

IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((msg[i] & 0xF0)<<8) ); //Sends Upper nibble
IO0SET = 0x00000050;  //RS HIGH & ENABLE HIGH to print data

IO0CLR = 0x00000020;  //RW LOW Write mode

delay_ms(2);


IO0CLR = 0x00000040; // EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0)

delay_ms(5);


IO0PIN = ( (IO0PIN & 0xFFFF00FF) | ((msg[i] & 0x0F)<<12) ); //Sends Lower nibble

IO0SET = 0x00000050; //RS & EN HIGH

IO0CLR = 0x00000020;

delay_ms(2);

IO0CLR = 0x00000040;

delay_ms(5);

i++;

}

}

int main(void)

{

uint32_t adcvalue;

char displayadc[18];

float adc;

float voltage;

char voltvalue[18];


  LCD_INITILIZE();       //Calls function to get ready the LCD to display
  LCD_DISPLAY("SSG");
delay_ms(900);


LCD_SEND(0xC0);
LCD_DISPLAY("ADC WITH LPC2148");
delay_ms(900);


PINSEL1 = 0x01000000; // Select P0.28 as AD0.1
AD0CR = 0x00200402;  //Sets ADC operation as 10-bits/11 CLK for conversion
while(1)                              //while loop that excecutes continueously
{
AD0CR = AD0CR | (1<<24); // Starts Conversion
while ( !(AD0DR1 & 0x80000000) ); // Waits for DONE
adcvalue = AD0DR1;
adcvalue = (adcvalue>>6);            //Right Shift six bits
adcvalue = (adcvalue & 0x000003FF);
LCD_SEND(0x80);
adc = adcvalue;
sprintf(displayadc, "adcvalue=%f", adc);


LCD_DISPLAY(displayadc);            //Display ADC value (0 to 1023)
voltage = ( (adcvalue/1023.0) * 3.3 ); // formula to convert ADC value into voltage


LCD_SEND(0xC0);
sprintf(voltvalue, "Voltage=%.2f V  ", voltage);
LCD_DISPLAY(voltvalue);                //Display (input analog voltage)

memset(voltvalue, 0, 18);

memset(displayadc,0, 18);

}

}
```

OLED Display

OLED is the acronym for **Organic Light Emitting Diode**. OLED is a modern display technology used in a wide range of electronic display devices, such as TVs, monitors, laptops, smartphones, bulletin boards, stadium screens, etc.



OLED displays consist of organic semiconductor compounds that emit a bright light on the passage of electric current through them, and hence it is termed as OLED. Since, OLED displays can emit light on their own, thus they are considered as self-emissive types of display. There is no need of backlight panel with LEDs to illuminate the screen.

The primary advantages of OLED displays include better picture quality, relatively wider viewing angles, greater flexibility in design, compact size, faster response time, and low power consumption.

Interfacing OLED Display with LPC2148 ARM Code

```
#include "system.h"

#include "serial.h"

int main(void)
{
        //mpu_example();
        oled_example();
}
void oled_example(void)
{
        oled_init();
        oled_reset();
        oled_fill(0x55);
        delay_ms(500);
        oled_clear();
        oled_fill(0x00);
        oled_print_xy(0,0,"Lpc2148");
        oled_print_xy(0,30,"SSG");
        while(1);
}
```

GSM

The SIM800C is a Quad-Band GSM/GPRS module in a LCC type which supports GPRS up to 85.6kbps data transfer. It has strong extension capability with abundant interfaces including UART, USB2.0, GPIO etc. The module provides much flexibility and ease of integration for customer's applications

General features

Frequency Band: 850/900/3800/3900MHz

GPRS multi-slot cass: 32/30

Compliment GSM phase 2/2+: Class 4 (2w 850/900MHz) and Class 3 (3w 3800/3900MHz)

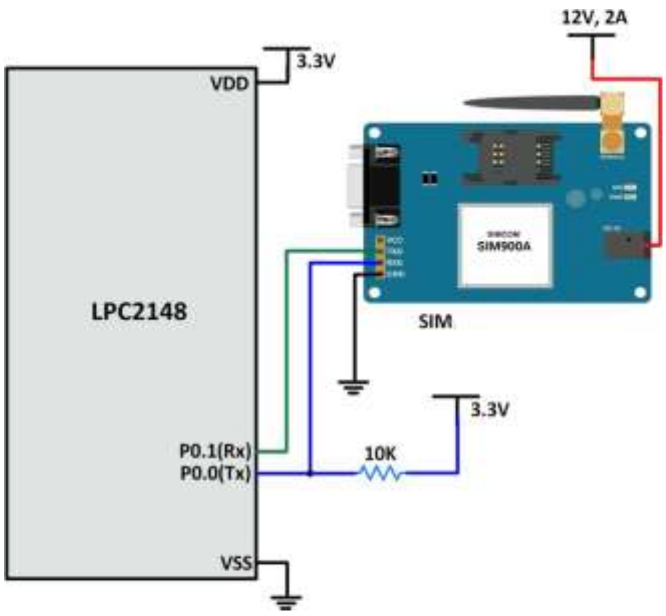
Control via AT commands (3GPP TP 27.007, 27.005 & SIMCom enhanced AT Commands)

Low power comsumption

GPRS mobile station class B, SMS cell boardcast

Embedded TCP/UDP prototcol, FTP/HTTP. Audio record, SSL/TLC, Speech code mode

Interfacing GSM with LPC2148 ARM



Code

```
#include <lpc214x.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

__irq void UART0_Interrupt(void);

void GSM_Begin(void);
void GSM_Calling(char *);
void GSM_HangCall(void);
void GSM_Response(void);
void GSM_Response_Display(void);
void GSM_Msg_Read(int);
bool GSM_Wait_for_Msg(void);
void GSM_Msg_Display(void);
void GSM_Msg_Delete(unsigned int);
void GSM_Send_Msg(char* , char*);
void GSM_Delete_All_Msg(void);

char buff[160];                    /* buffer to store responses and messages */
```

```
bool status_flag = false;          /* for checking any new message */
volatile int buffer_pointer;
char Mobile_no[14];                /* store mobile no. of received message */
char message_received[60];                /* save received message */
int position = 0;           /* save location of current message */

void delay_ms(uint16_t j)
{
   uint16_t x,i;
          for(i=0;i<j;i++)
          {
   for(x=0; x<6000; x++);    /* loop to generate 1 milisecond delay with Cclk = 60MHz */
          }
}

void UART0_init(void)
{
          PINSEL0 = PINSEL0 | 0x00000005;      /* Enable UART0 Rx0 and Tx0 pins of UART0 */
          U0LCR = 0x83; /* DLAB = 1, 1 stop bit, 8-bit character length */
          U0DLM = 0x00;           /* For baud rate of 9600 with Pclk = 15MHz */
          U0DLL = 0x61; /* We get these values of U0DLL and U0DLM from formula */
          U0LCR = 0x03; /* DLAB = 0 */
          U0IER = 0x00000001; /* Enable RDA interrupts */
}

void UART0_TxChar(char ch) /* A function to send a byte on UART0 */
{
          U0IER = 0x00000000; /* Disable RDA interrupts */
          U0THR = ch;
          while( (U0LSR & 0x40) == 0 ); /*  Wait  till  THRE  bit  becomes  1  which  tells  that
transmission is completed */
          U0IER = 0x00000001; /* Enable RDA interrupts */
}

void UART0_SendString(char* str) /* A function to send string on UART0 */
{                   uint8_t i = 0;
          U0IER = 0x00000000; /* Disable RDA interrupts */

          while( str[i] != '\0' )
          {
                    UART0_TxChar(str[i]);
                    i++;
```

```
        }
        U0IER = 0x00000001; /* Enable RDA interrupts */
}

__irq void UART0_Interrupt(void)
{
        buff[buffer_pointer] = U0RBR;           /* copy UDR(received value) to buffer */
        buffer_pointer++;
        status_flag = true;                                             /*    flag    for    new
message arrival */
        VICVectAddr = 0x00;
}

int main(void)
{               bool is_msg_arrived;
        buffer_pointer = 0;

        memset(message_received, 0, 60);
        VICVectAddr0 = (unsigned) UART0_Interrupt;          /* UART0 ISR Address */
        VICVectCntl0 = 0x00000026;  /* Enable UART0 IRQ slot */
        VICIntEnable = 0x00000040; /* Enable UART0 interrupt */
        VICIntSelect = 0x00000000;    /* UART0 configured as IRQ */
        UART0_init();
        //UART0_SendString("GSM Initializing...");
        delay_ms(3000);
        GSM_Begin();  /* check GSM responses and initialize GSM */

        while (1)
                {
                        /*check if any new message received */
                        if(status_flag == true)
                                {
                                        is_msg_arrived       =       GSM_Wait_for_Msg();
        /*check for message arrival*/
                                        if(is_msg_arrived == true)
                                        {
                                                //UART0_SendString("New
message");      /* new message arrived */
                                                delay_ms(1000);
                                                GSM_Msg_Read(position);
        /* read arrived message */
                                                delay_ms(3000);
```

```
                                                        /*check if received message is "call me"
*/
                                                                if(strstr(    message_received,"call
me"))
                                                                        {


        GSM_Calling(Mobile_no);             /* call sender of message */

        //UART0_SendString("Calling...");

        delay_ms(15000);

        GSM_HangCall();             /* hang call */

        //UART0_SendString("Hang Call");

        delay_ms(1000);
                                                                        }

                                                                GSM_Msg_Delete(position);
        /* to save SIM memory delete current message */
                                                                //UART0_SendString("Clear
msg");
                                                                GSM_Response();
                                                                delay_ms(1000);

                                                }

                                        is_msg_arrived = false;
                                        status_flag = false;
                                }
                        UART0_SendString("Waiting for msg");
                        memset(Mobile_no, 0, 14);
                        memset(message_received, 0, 60);
                }
}

void GSM_Begin(void)
{
        while(1)
```

```
		{
			UART0_SendString("ATE0\r\n");			/* send ATE0 to check module is
ready or not */
			delay_ms(500);
			if(strstr(buff,"OK"))
			{
				GSM_Response();			/* get Response */
				memset(buff,0,160);
				break;
			}
			else
			{
				//UART0_SendString("Error");
			}
		}
		delay_ms(1000);

		//UART0_SendString("Text Mode");
		UART0_SendString("AT+CMGF=1\r\n");	/* select message format as text */
		GSM_Response();
		delay_ms(1000);
}

void GSM_Msg_Delete(unsigned int position)
{			char delete_cmd[20];
		buffer_pointer=0;

		sprintf(delete_cmd,"AT+CMGD=%d\r\n",position); /* delete message at specified
position */
		UART0_SendString(delete_cmd);
}

void GSM_Delete_All_Msg(void)
{
		UART0_SendString("AT+CMGDA=\"DEL ALL\"\r\n"); /* delete all messages of SIM */

}

bool GSM_Wait_for_Msg(void)
{
		char msg_location[4];
		int i;
```

```
	delay_ms(500);
	buffer_pointer=0;

	while(1)
	{
		if(buff[buffer_pointer]=='\r' || buff[buffer_pointer]== '\n') /*eliminate "\r \n"
which is start of string */
		{
			buffer_pointer++;
		}
		else
			break;
	}

	if(strstr(buff,"CMTI:"))		/* "CMTI:" to check if any new message received */
	{
		while(buff[buffer_pointer]!= ',')
		{
			buffer_pointer++;
		}
		buffer_pointer++;

		i=0;
		while(buff[buffer_pointer]!= '\r')
		{
			msg_location[i]=buff[buffer_pointer];			/* copy location of
received message where it is stored */
			buffer_pointer++;
			i++;
		}

		/* convert string of position to integer value */
		position = atoi(msg_location);

		memset(buff,0,strlen(buff));
		buffer_pointer=0;

		return true;
	}
	else
	{
		return false;
		}
}
void GSM_Send_Msg(char *num,char *sms)
{
	char sms_buffer[35];
	buffer_pointer=0;
	sprintf(sms_buffer,"AT+CMGS=\"%s\"\r\n",num);
	UART0_SendString(sms_buffer);			/*send command AT+CMGS="Mobile
No."\r */
	delay_ms(200);
	while(1)
	{
		if(buff[buffer_pointer]==0x3e)			/* wait for '>' character*/
		{
			buffer_pointer = 0;
			memset(buff,0,strlen(buff));
			UART0_SendString(sms);			/* send msg to given no. */
			UART0_TxChar(0x1a);		/* send Ctrl+Z then only message will
transmit*/
			break;
		}
		buffer_pointer++;
	}
	delay_ms(300);
	buffer_pointer = 0;
	memset(buff,0,strlen(buff));
	memset(sms_buffer,0,strlen(sms_buffer));
}

void GSM_Calling(char *Mob_no)
{
	char call[20];
	sprintf(call,"ATD%s;\r\n",Mob_no);
	UART0_SendString(call);			/* send command ATD<Mobile_No>; for
calling*/
}

void GSM_HangCall(void)
{
	UART0_SendString("ATH\r\n");			/*send command ATH\r to hang call*/

}
```

```
void GSM_Response(void)
{       int i;
        unsigned int timeout=0;
        int CRLF_Found=0;
        char CRLF_buff[2];
        int Response_Length=0;
        while(1)
        {
                if(timeout>=60000)          /*if timeout occur then return */
                return;
                Response_Length = strlen(buff);
                if(Response_Length)
                {
                        delay_ms(1);
                        timeout++;
                        if(Response_Length==strlen(buff))
                        {
                                for( i=0;i<Response_Length;i++)
                                {
                                        memmove(CRLF_buff,CRLF_buff+1,1);
                                        CRLF_buff[1]=buff[i];
                                        if(strncmp(CRLF_buff,"\r\n",2))
                                        {
                                                if(CRLF_Found++==2)         /*  search  for  \r\n
in string */
                                                {
                                                        GSM_Response_Display();            /*
display response */
                                                        return;
                                                }
                                        }

                                }
                                CRLF_Found = 0;

                        }

                }
                delay_ms(1);
                timeout++;
        }
```

```
		//status_flag = false;
	}

	void GSM_Response_Display(void)
	{
		buffer_pointer = 0;
		while(1)
		{
			if(buff[buffer_pointer]== '\r' || buff[buffer_pointer]== '\n')		/*	search 
	for \r\n in string */
			{
				buffer_pointer++;
			}
			else
				break;
		}


		while(buff[buffer_pointer]!='\r')			/* display response till "\r" */
		{
			UART0_TxChar(buff[buffer_pointer]);

			buffer_pointer++;
		}
		buffer_pointer=0;
		memset(buff,0,strlen(buff));
	}

	void GSM_Msg_Read(int position)
	{
		char read_cmd[10];
		sprintf(read_cmd,"AT+CMGR=%d\r\n",position);
		UART0_SendString(read_cmd);			/*	read	message	at	specified 
	location/position */
		GSM_Msg_Display();		/* display message */
	}

	void GSM_Msg_Display(void)
	{				int i=0;
		delay_ms(500);
		if(!(strstr(buff,"+CMGR")))			/*check for +CMGR response */
		{
        UART0_SendString("No message");
}
else
{
        buffer_pointer = 0;
        while(1)
        {
                if(buff[buffer_pointer]=='\r' || buff[buffer_pointer]== 'n')
/*wait till \r\n not over*/
                {
                        buffer_pointer++;
                }
                else
                break;
        }

        /* search for 1st ',' to get mobile no.*/
        while(buff[buffer_pointer]!=',')
        {
                buffer_pointer++;
        }
        buffer_pointer = buffer_pointer+2;

        /* extract mobile no. of message sender */
        for( i=0;i<=12;i++)
        {
                Mobile_no[i] = buff[buffer_pointer];
                buffer_pointer++;
        }

        do
        {
                buffer_pointer++;
        }while(buff[buffer_pointer-1]!= '\n');



        /* display and save message */
        while(buff[buffer_pointer]!= '\r' && i<31)
        {
                        UART0_TxChar(buff[buffer_pointer]);
                        message_received[i]=buff[buffer_pointer];
                buffer_pointer++;
                i++;
        }

        buffer_pointer = 0;
        memset(buff,0,strlen(buff));
    }
    status_flag = false;
}
```

Interfacing SD CARD with LPC2148 ARM

Note: This program file is given in folder