JX-2148

LPC2148 ARM7-32 bit Microcontroller Education board

1. Kit Contents

In standard package of JX-2148 board include :

1. JX-2148 Education board x 1
2. CX-232 serial port cable x 1
3. AWG#22 wire jumper, 7cm. length x 10
4. Documentation x 1
5. CD-ROM x 1

To run this education board you'll need: DC adaptor +6Vdc 500mA (maximum +9Vdc)

2. JX-2148 board features

● LPC2148 microcontrollers are based on a 32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combine microcontroller with embedded high-speed flash memory 512 kB. A 128-bit wide memory interface and a unique accelerator architecture enable 32-bit code execution at the maximum clock rate

● Standard JTAG connector

● USB 2.0 Full Speed Interface (USB connector type B). JX-2148 board provides a USB interface connector that interfaces ot the on-chip USB peripheral of the LPC2148 device. You may configure the board as self-powered or USB powered device.

● Dual Serial Ports. JX-2148 provides standard DB9 connectors for both of the LPC214x's serial ports. UART-0 for communication and support In-System Programming (ISP), UART-1 for serial communication and select to connect ESD-02 Bluetooth module (optional) by jumpers.

● SD/MMC socket. The JX-2148 provides one SPI module to interface SD/MMC memory socket.

● A PS/2 jack for interface Keyboard or Mouse.

● 2 of push-button switches with resistor pull-up.

● 2 of LED indicator

● Analog Voltage Control for ADC Input. JX-2148 provides an adjustable analog voltage source for testing the A/D converter feature of the LPC2148.

● A small buzzer for sound experiment

● Mini-breadboard 170 points contact.

● 32kHz crytal and +3V battery backup for real-time clock system within MCU.

● +3.3V on-board regulator for MCU and +5V for PS/2 circuit.

● Polarity voltage protection.

3. System requirements

To use the JX-2148 Education Board, the following item must be prepared :

● An IBM-compatible PC with port of the following:

- one unused USB ports to test USB experiment
- at least one unused RS-232 port for In-System Flash Programming via Serial Interface. If have two better, because can download and test communication both.

● Install Windows XP Operating System

● Install µVision3 or Keil ARM tool kit evaluation version. Download at www.keil.com

● Install LPC2000 In-system programming utility software from Philips. Download at www.philips.com and search by keyword "Microcontroller ARM7".

● USB to RS-232 serial port converter. In case the computer does not provide RS-232 port. (Suggess UCON-232. See detail at www.inexglobal.com)

● DC adaptor +9V 500mA recommended.

● A serial cable, 9-pin male to 9-pin female, 1.8 m length, wired one-to-one. In case using both UART in same time. CX-232 cable from INEX recommended

● ESD-02 Bluetooth module if need to make wireless communication.

● PS/2 Keyboard if need to make PS/2 keyboard interface experiment.

● PS/2 Mouse if need to make PS/2 mouse interface experiment.

● USB cable, AB type not over 3m. length for testing USB interface.

● ULINKUSB-JTAG adaptor. Direct contact to www.keil.com.

4. JX-2148 board layout

The figure 1 illustration shows the important interface and hardware components of the JX-2148 board.

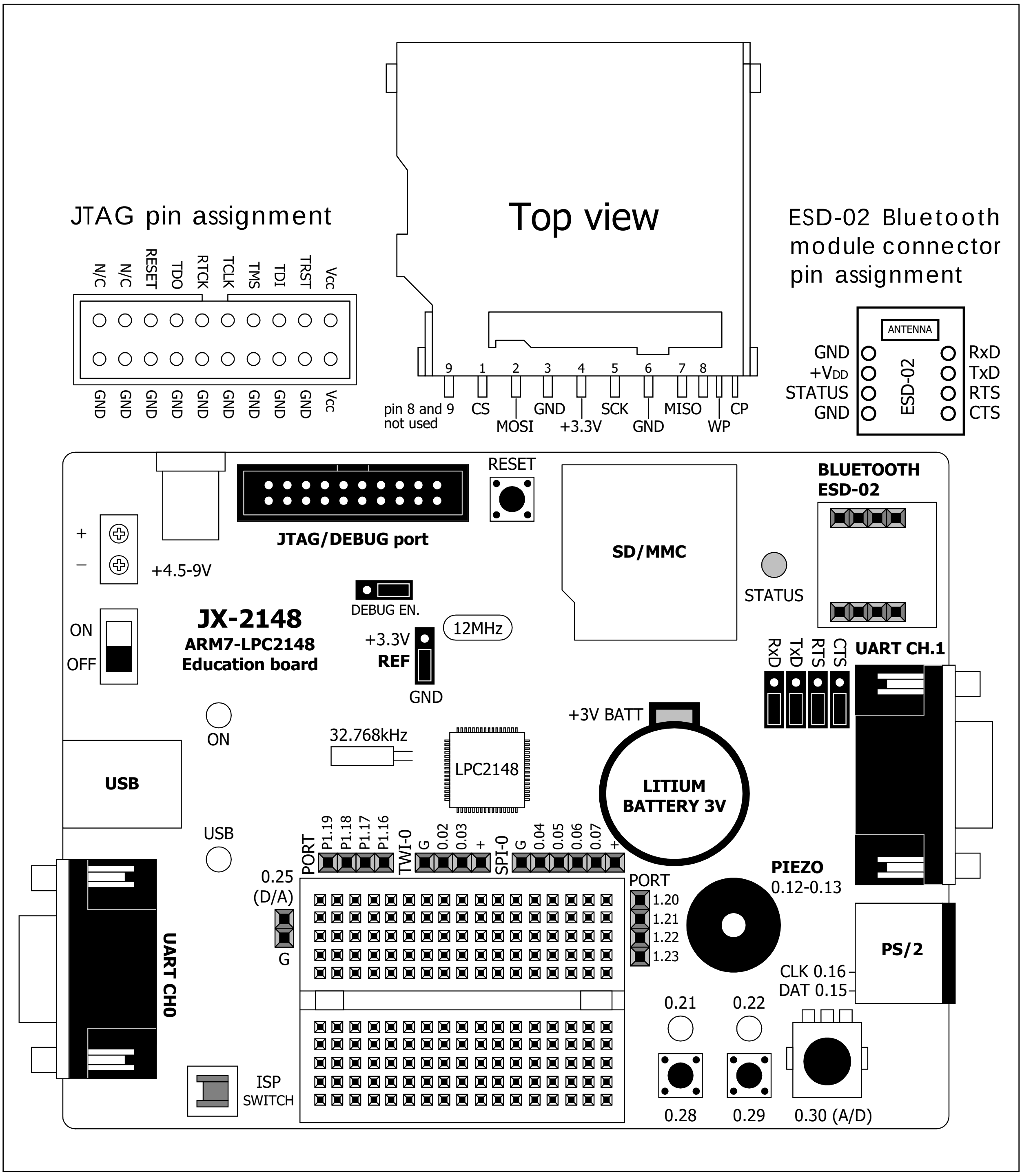


Figure 1 The JX-2148 board layout

5. JX-2148 operation

The operation of JX-2148 board has 3 main sections following :

(1) LPC2148 microcontroller unit

(2) Power supply

(3) Input/Output circuit

Microcontroller unit consist of Philips's LPC2148 and 2 of Clock oscillator circuit; 12MHz main clock oscillator and 32.768kHz for real-time clock. The full schematic can see in the figure 2.

Power supply of JX-2148 has 2 regulator. One is +3.3V. It receives +6 to +16V from external DC adaptor. The 3.3V regulated circuit supplies to the microccontroller unit and many I/O devices. Another one is +5V for supply PS/2 circuit.

Many I/O devices are installed on the JX-2148 board. Includes LED, Push-button switches, Variable resistor for A/D converter circuit, Two of RS-232 serial port interface, PS/2 jack, MMC/SD socket, USB port interface, JTAG interface for many debugger such as *ULINK* from Keil or *Olimex JTAG adaptor* or *Wiggler* from Macraigor, ESD-02 Bluetooth connector, Free I/O microcontroller port and a Mini-breadboard 180 points for construction the experimental circuit.

6. JX-2148 circuit description

The complete shcematic of the JX-2148 board shows in the figure 2. The main device is Philips's LPC2148 ARM7 microcontroller. LPC2148 is assigned to connect many I/O devices as :

P0.00/TxD0 and P0.01/RxD0 are connected to RS-232 serial port inteface at UART0

P0.02/SCL0 and P0.03/SDA0 are connected with I^{2}C bus or Two wire interface (TWI)

P0.04 to P0.07 are connected SPI bus

P0.08/TxD1, P0.09/RxD1, P0.10 and P0.11 are connected with serial port interface circuit; UART1 and ESD-02 Bluetooth module connector. User can select by jumpers. P0.10 pin is connected with CTS pin and P0.11 is connected with RTS pin. See the figure 3.

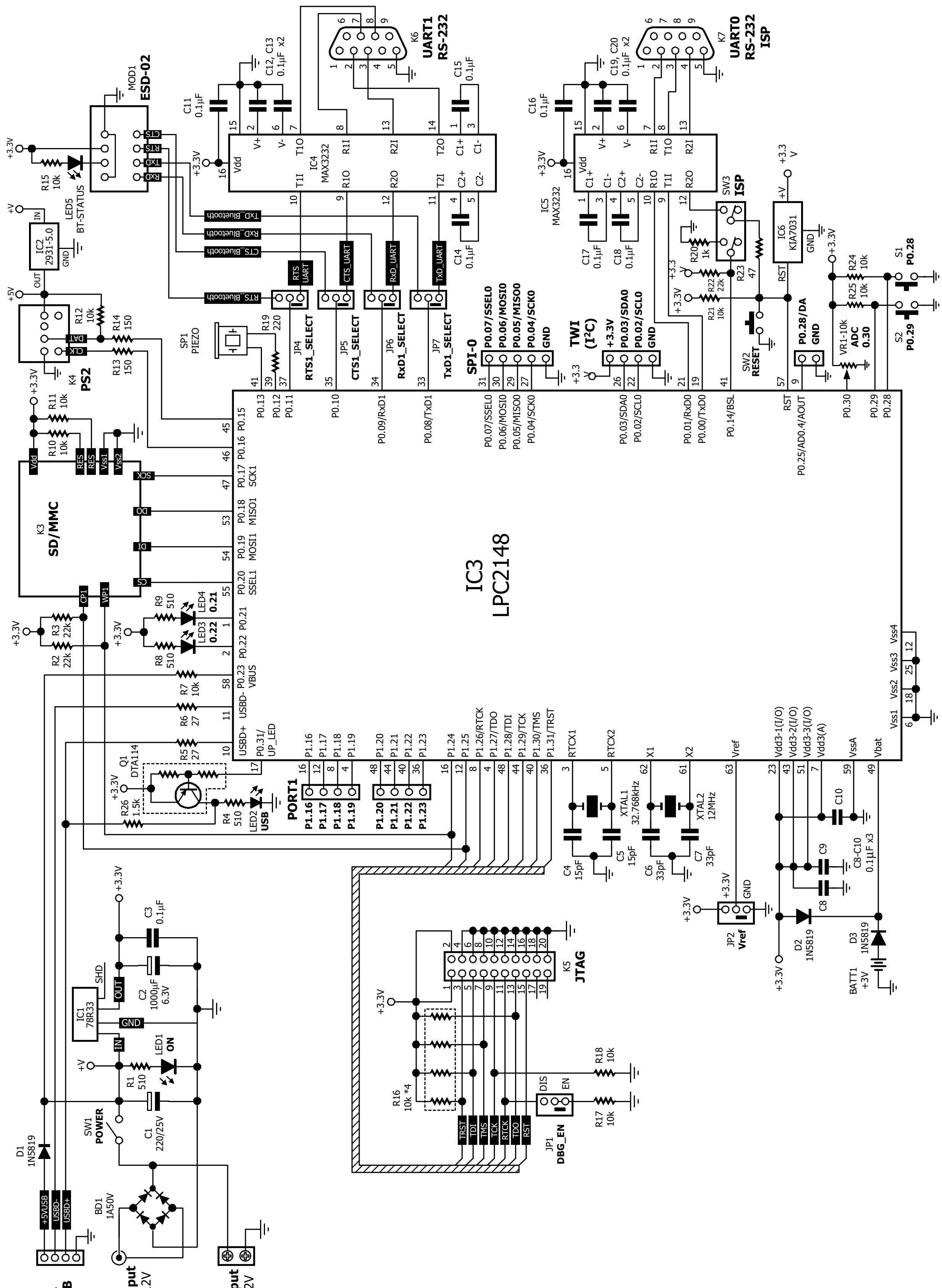


Figure 2 JX-2148 board schematic diagram

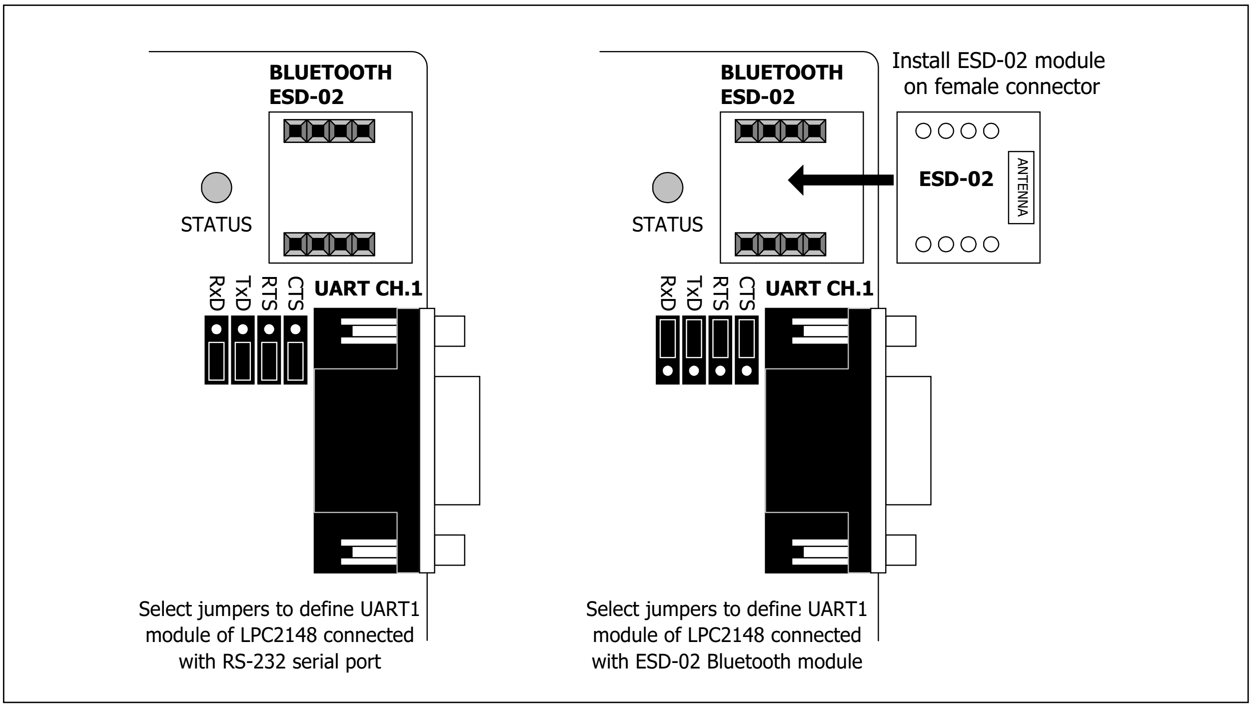


Figure 3 Shows the step of select UART1 to connect RS-232 port and ESD-02 Bluetooth module (ESD-02 is optional device)

P0.12 and P0.13 are connected to buzzer.

P0.15 and P0.16 are connected with PS/2 jack. P0.15 is data pin and P0.16 is clock pin.

P0.17 to P0.20 are SSP module port. They are connected with MMC/ SD socket.

P0.21 and P0.22 are connected with LED in active low.

P0.25 is D/A output of D/A converter module in LPC2148.

P0.28 and P0.29 are connected push-button switches with pull-up resistors.

P0.30 is connected with 10kW variable resistor for testing A/D converter module.

P0.31 is used to control USB port interfacing and connected with REDY connection indicator circuit.

P1.16 to P1.23 are free port.

P1.24 and P1.25 are connected MMC/SD socket to testing card insertion.

P1.26 to P1.31 are assigned as JTAG interface.

In-system flash programming of JX-2148 will work via UART0 module. SW3 is ISP mode switch. Must pess this switch to enter ISP mode.

LPC2148 has each of SPI (Serial Peripheral Interface) and SSP (Synchronous Serial Port) module. THe SSP module can work in SPI mode.On JX-2148 board define SSP to connect with MMC/SD card interface. The SPI module (SPI0) will reserve to connect addition SPI peripheral.

In A/D converter demonstration, on JX-2148 board provides one POT or variable resistor is connected at P0.30 ready to test with programming.

In testing simple I/O port, the JX-2148 board provides 2 of LED that connected with P0.21 and P0.22. About input device, provides 2 of push-button switch to connect with P0.28 and P0.29. Two port pins P0.12 and P0.13 are connected with buzzer to sound generator.

PS/2 interface need +5V. LM2931-5.0 IC is regulator +5V IC. It receive input voltage from main DC adaptor.

USB interface use USBD+ and USBD- pin. They are connected limited current protection resistor. Port 0.23/VBUS is connected with +5V from USB port connector. Interface controlling port is function of P0.31 port pin. In connection must control this pin to logic “0”. Thus, user can control the USB connections via software.

REF jumper : use to select the reference voltage of A/D converter module. Normally connected with +3.3V. If need to use external reference voltage, user can do very easy step. Remove jumper out and connect the external reference voltage with middle pin of REF jumper.

DEBUG EN. jumper : Select to enable degugging via JTAG connector.

7. Software Developmet tools :

7.1 Keil µVision3 evaluation version

The JX-2148 board can develope with any software development such as WinARM with Eclipse IDE or Keil µVision3. However in this documentation will suggess Keil µVision3 evaluation version. Developers can purchase the full version from www.keil.com.

The limitation of Keil mVision3 evaluatiuon version is

- Programs that generate more than 16K Bytes of object code will not compile, assemble, or link.
- The evaluation tools create Symbolic Output Format when the RealView compiler is selected. Fully licensed tools generate standard ELF/DWARF files.
- The debugger supports programs that are 16K Bytes or smaller.
- The RealView Linker does not accept scatter-loading description files for sophisticated memory layouts.
- The RealView Linker restricts the base address for code/constants to to 0xXX000000, 0xXX800000, or 0x00080000 where XX is 00, 01, ..., FF. This allows memory start address like 0x00000000 and 0x12800000.
- It is not possible to generate position independent code or data. The RealView C/C++ Compiler does not generate a listing file.
- The CARM compiler, assembler, and linker are limited to 16K Bytes of object code. Source code may be of any size.
- The GNU ARM tools (compiler, assembler, and so on) that are provided are not limited or restricted in any way.

7.1.1 Download

All steps are introduced in this document can change anytime depend on the owmer website.

(1) Enter to Keil webpage by type URL as www.keil.com following the figure 4. See Software header and select Evaluation Software

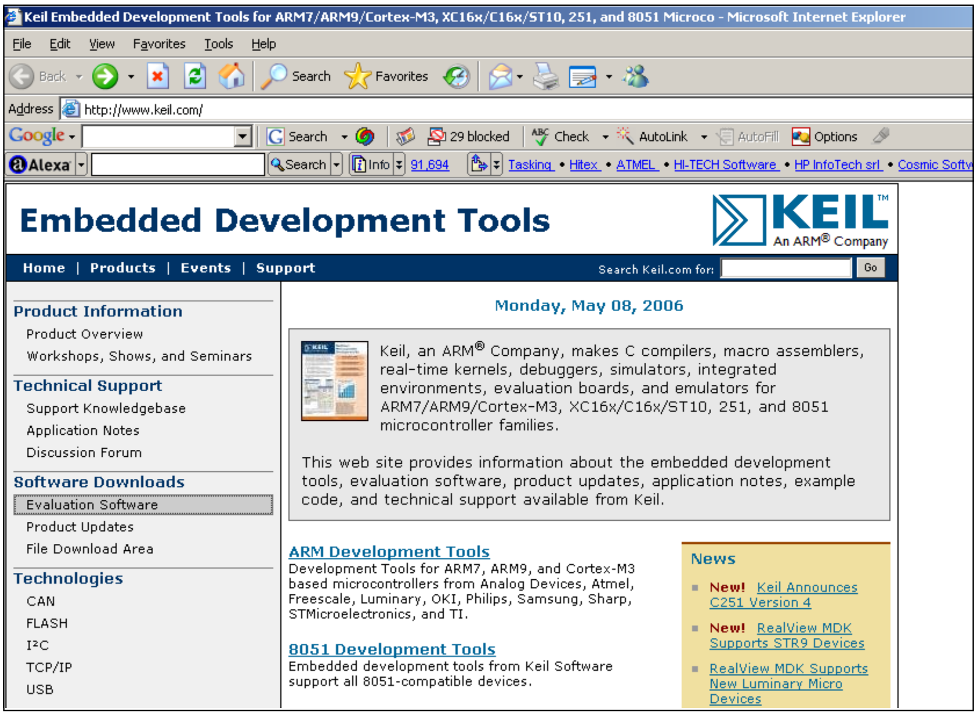


Figure 4 Keil's homepage

(2) After that Evaluation Software page will appear following the figure 5. Select ARM Evaluation Software.

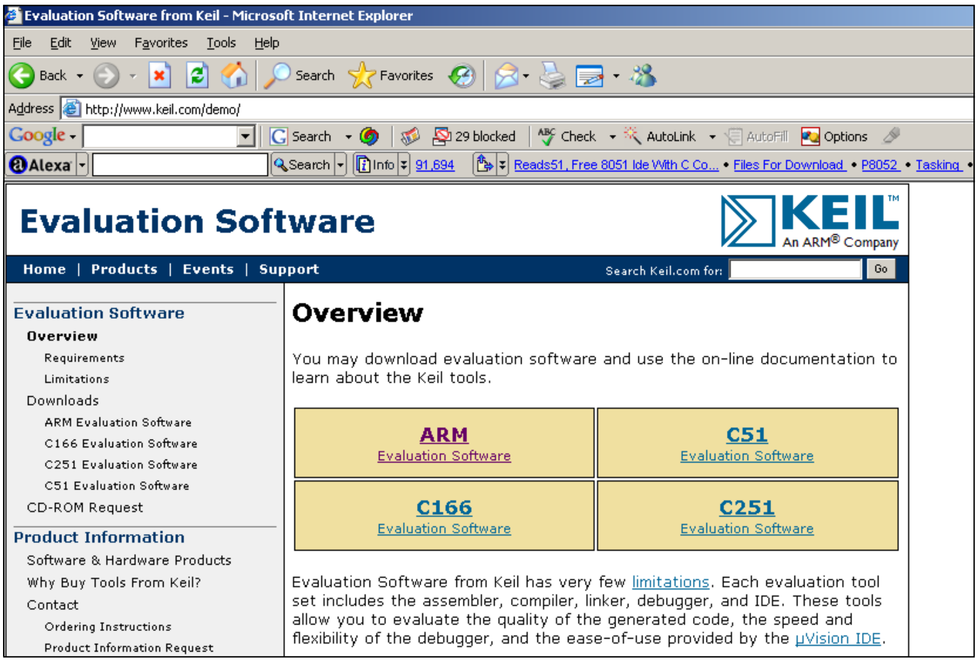


Figure 5 Evaluation Software webpage for download. Select ARM Evaluation Software.

(3) Developers must register before download following the figure 6. After complete, click **Submit** button to confirmation.

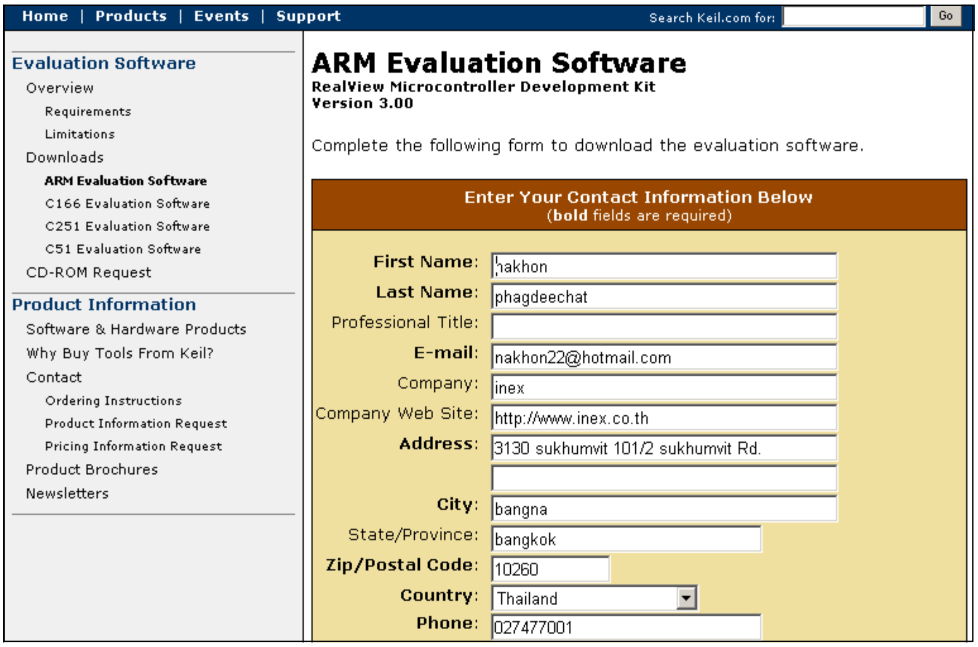


Figure 6 Evaluation Software Registration form.

(4) Download webpage will appear following the figure 7. Click at **RVMDK300A.EXE** file. Its size is 50MB approximation.

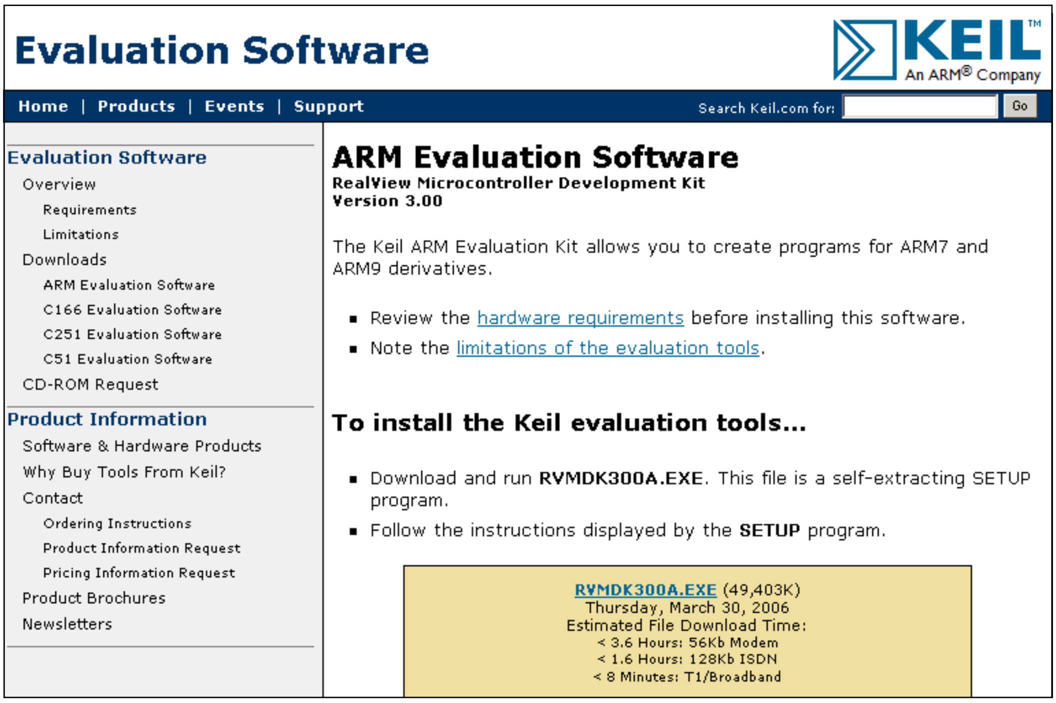


Figure 7 ARM Evaluation Software Download webpage

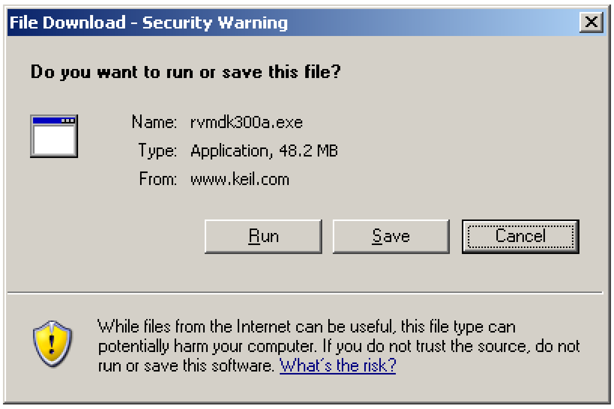


Figure 8 Saving file dialog box.

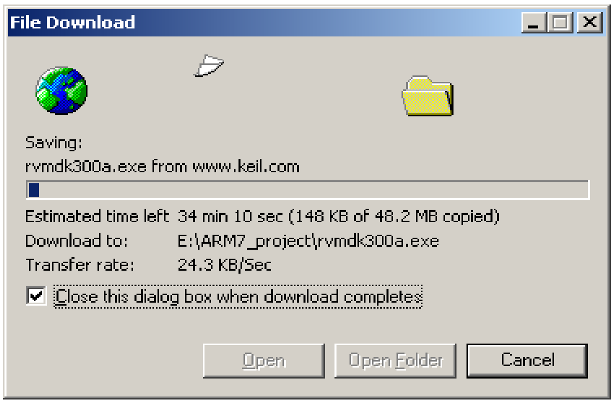


Figure 9 Download software status

(5) Saving file dialog box will appear following the figure 8. Click **Save** button and define path and folder for saving file. After that downloading will begin and shows status following the figure 9.

7.1.2 Installation Keil µVision3 and Preparation

(1) After download complete, double-click at the installation file. THe first installation window will appear following the figure 10. Click **Next** button.

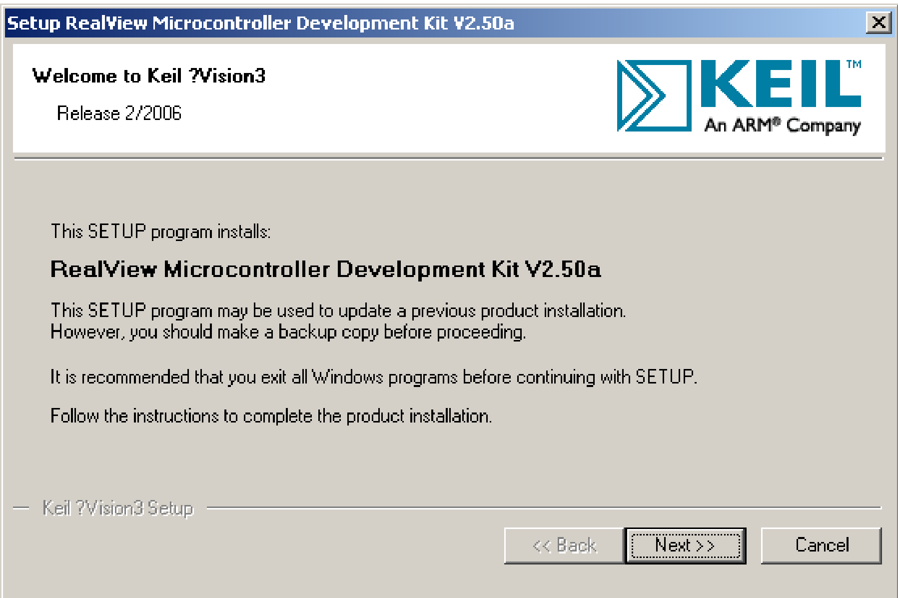


Figure 10 The First Keil µVision3 installationwindow

(2) Enter to License Agreement window following the figure 11. Click at box of message "`I agree to all the terms of the preceding License Agreement`". Click **Next** button.

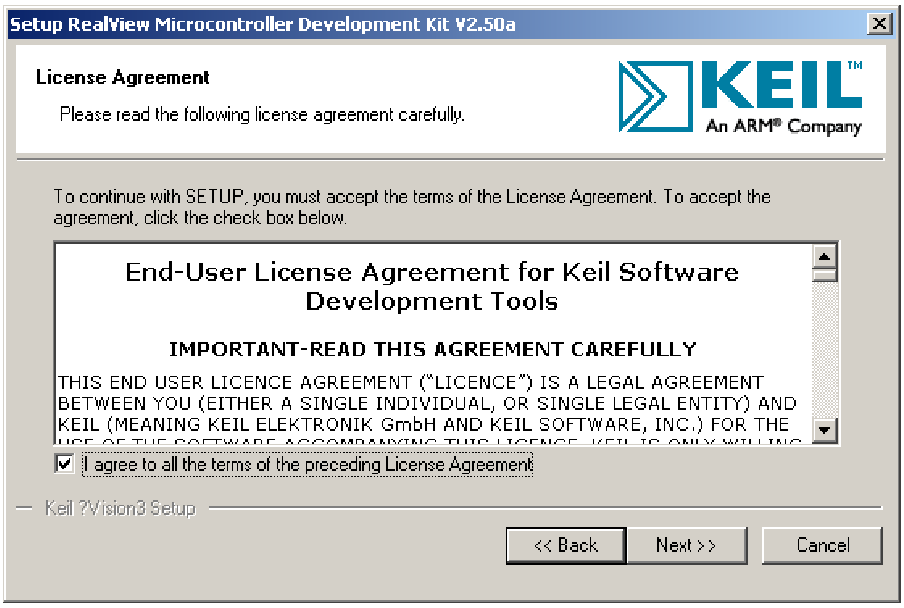


Figure 11 License Agreement window of Keil µVision3

(3) Folder Selection window will appear. Suggest to install into **`Keil`** folder following the figure 12.

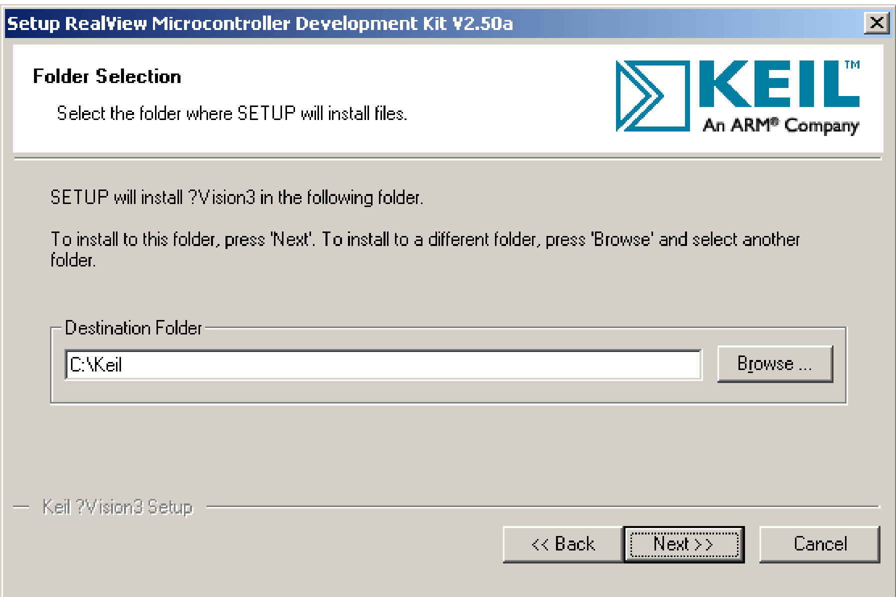


Figure 12 Folder Selection window of Keil µVision3

(4) The Customer Information window will appear following the figure 13. Must to put the user information in the blank box. Click **Next** button.

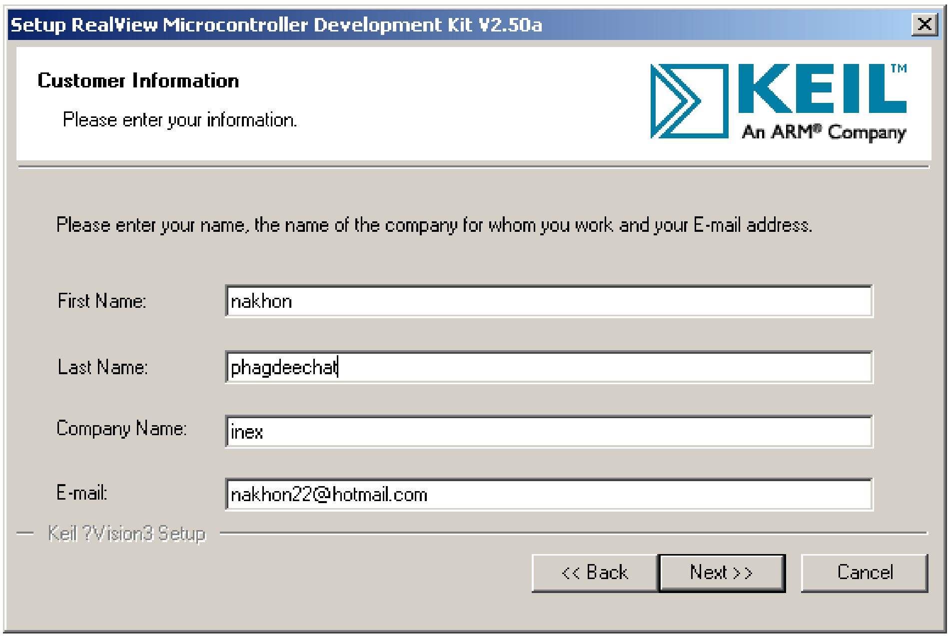


Figure 13 Customer Information window.

(5) Installation software will start and shows status following the figure 14.

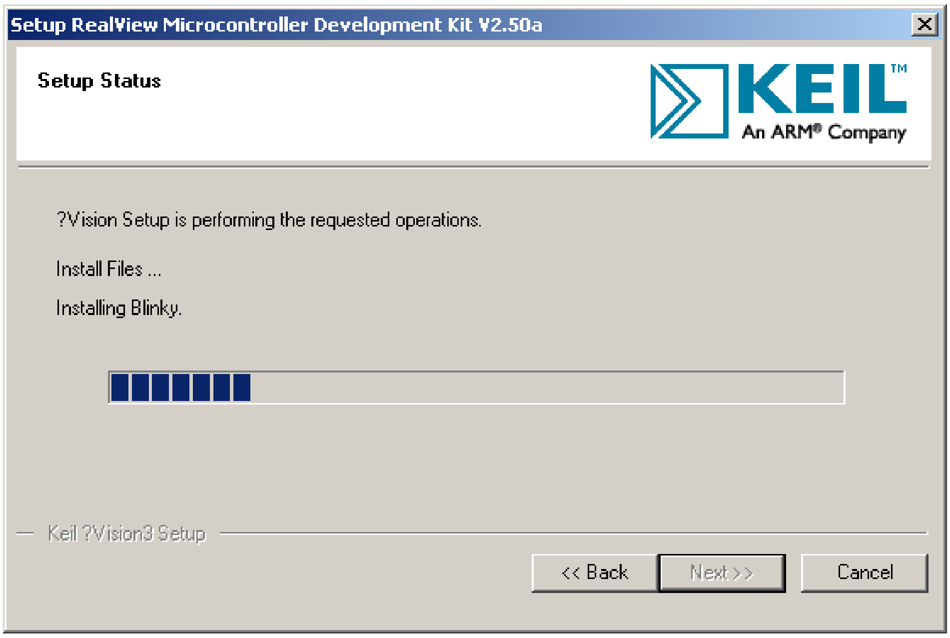


Figure 14 Setup status window shows the installation status.

(6) After installation complete, the window in figure 15 will appear. Click **Finish** button to ending installation.

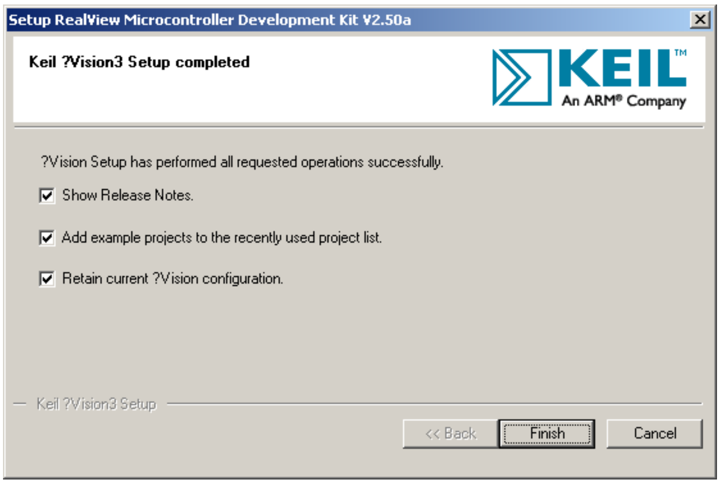


Figure 15 Completed installation window

(7) Run Keil µVision3 program. The main program window will be show following the figure 16.

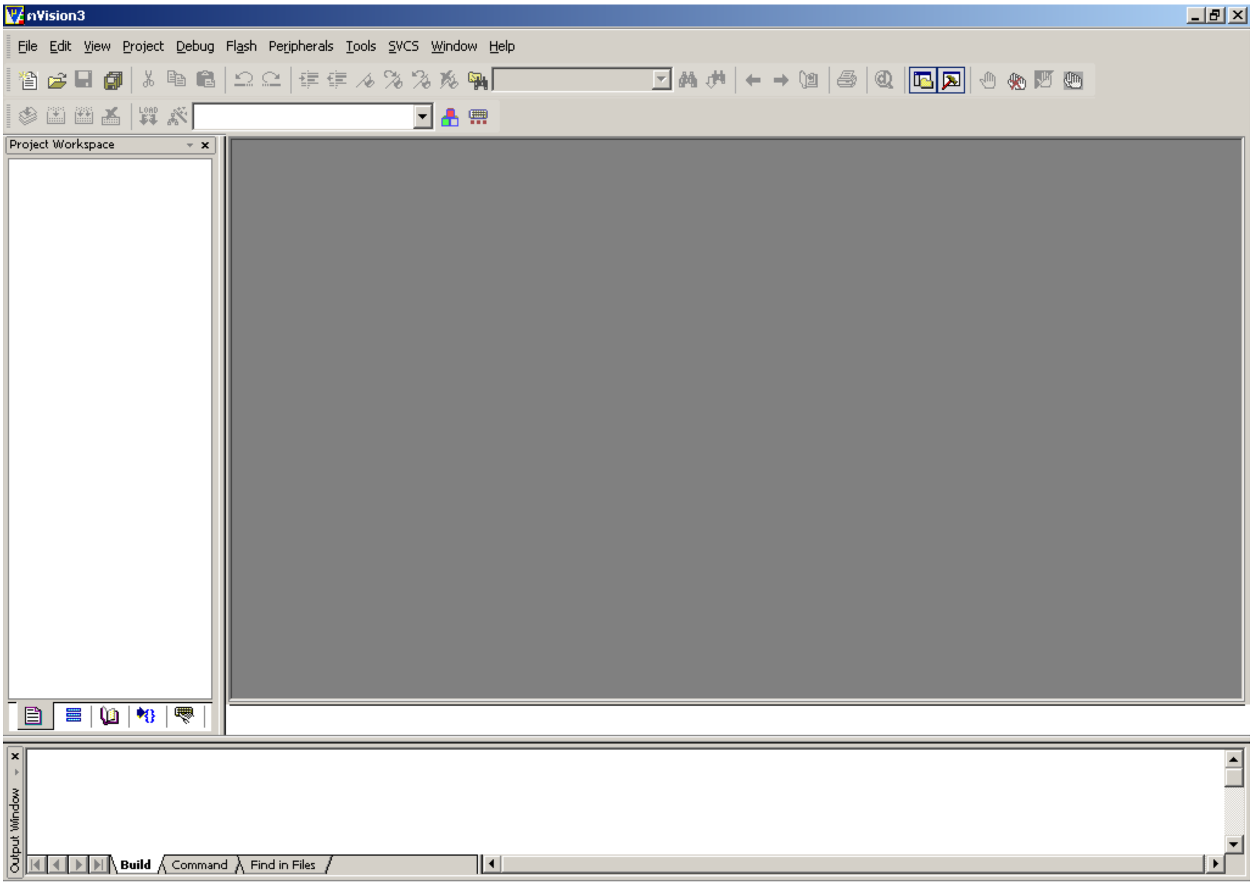


Figure 16 The main window of Keil µVision3

(8) Enter ***Project → Component,Enviroment,Book...*** menu following the figure 17.

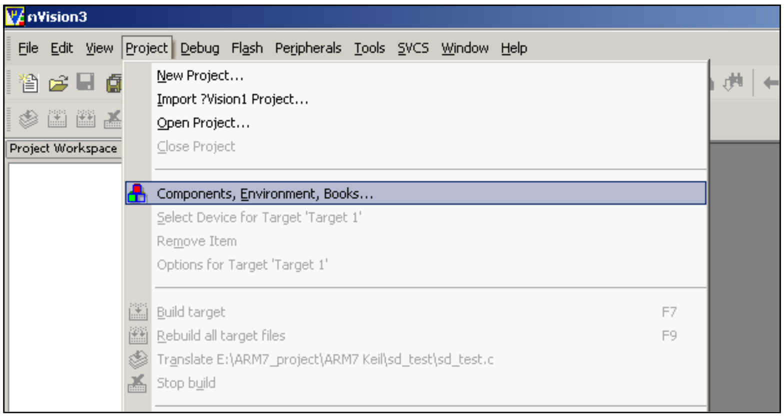


Figure 17 Shows the selection Project → Component,Enviroment,Book...

(9) Component,Enviroment,Book window will appear. At Select ***ARM Development Tools*** menu, select ***Use Keil CARM Compiler*** to select compiler as ***Keil CARM*** following the figure 18. After that developers can develop their own softwares.

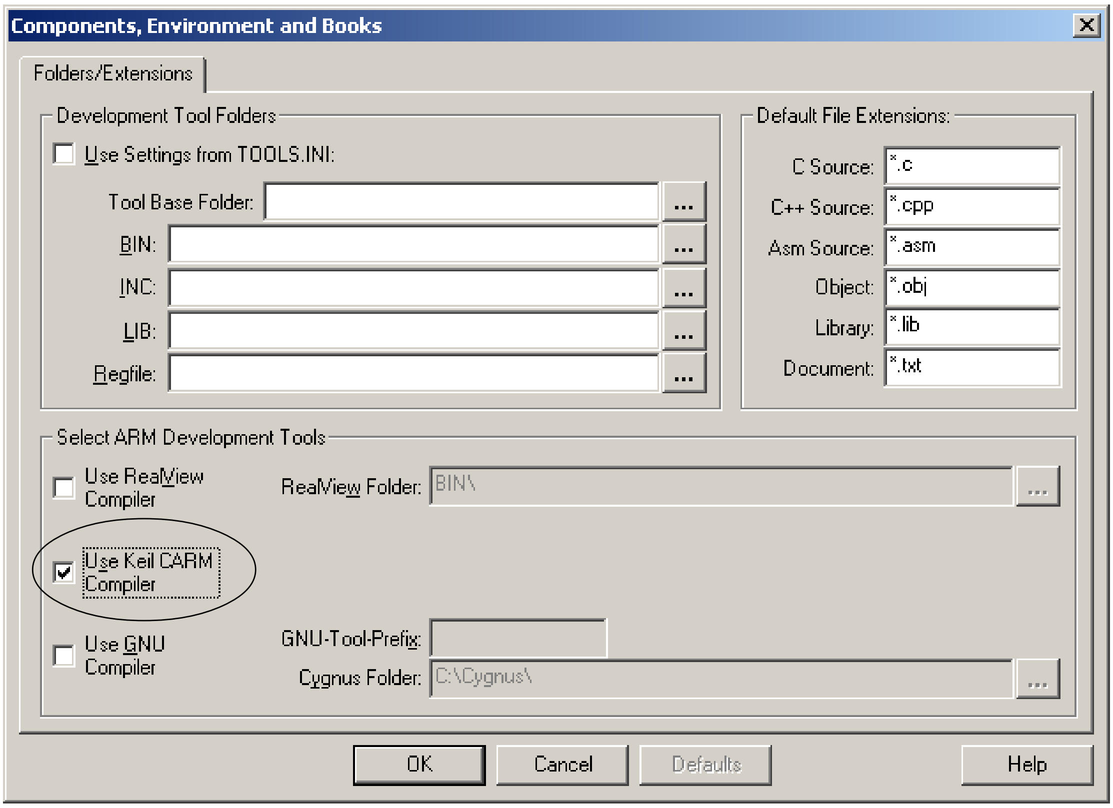


Figure 18 Shows the selection Keil CARM compiler to program development preparation.

7.2 Philips LPC2000 Flash utility

The Philips LPC2000 Flash utility provides In-System Flash Programming (ISP) support for Intel HEX files. The Philips LPC2000 Flash Utility connects the PC's COM port to the serial port UART0 of the JX-2148 Education board. The installation program for this utility is included on the INEX JX-2148 CD-ROM or may be downloaded from the Philips Web Site. The LPC2000 Flash Utility may be run as a stand-alone utility or as an external tool from within the mVision IDE. The version of LPC2000 Flah utility can support LPC2148 is V2.2.3 or higher.

The normal installation file name of Philips LPC2000 Flash utility is Philips Flash Utility Installation.exe. Double-click that file. Accept all confirmation until installation complete.

8. Develop programs

This sections introduce you to the Keil development tools, and take you through the process using them with the JX-2148 board. You'll learn how to use mVision to create, compile, download, and run a program on this board.

Developing programs for the JX-2148 board is easy. The process is:

(1) Creating Application Programs using the µVision IDE and the Keil, GNU or ARM ADS C Compiler.

(2) Download the program to the on-chip Flash of the JX-2148 Board.

8.1 Building C project file

(1) Run Keil µVision3. The main window will appear following the figure 19. If have any project opened, can close by select menu ***Project → Close Project***

(2) Build new project. Enter menu ***Project → New Project***..following the figure 20.

(3) The Create New Project window will appear. Set path of project file, from example set to `C:\` following the figure 21. Make new folder for storing project file in name `led`.

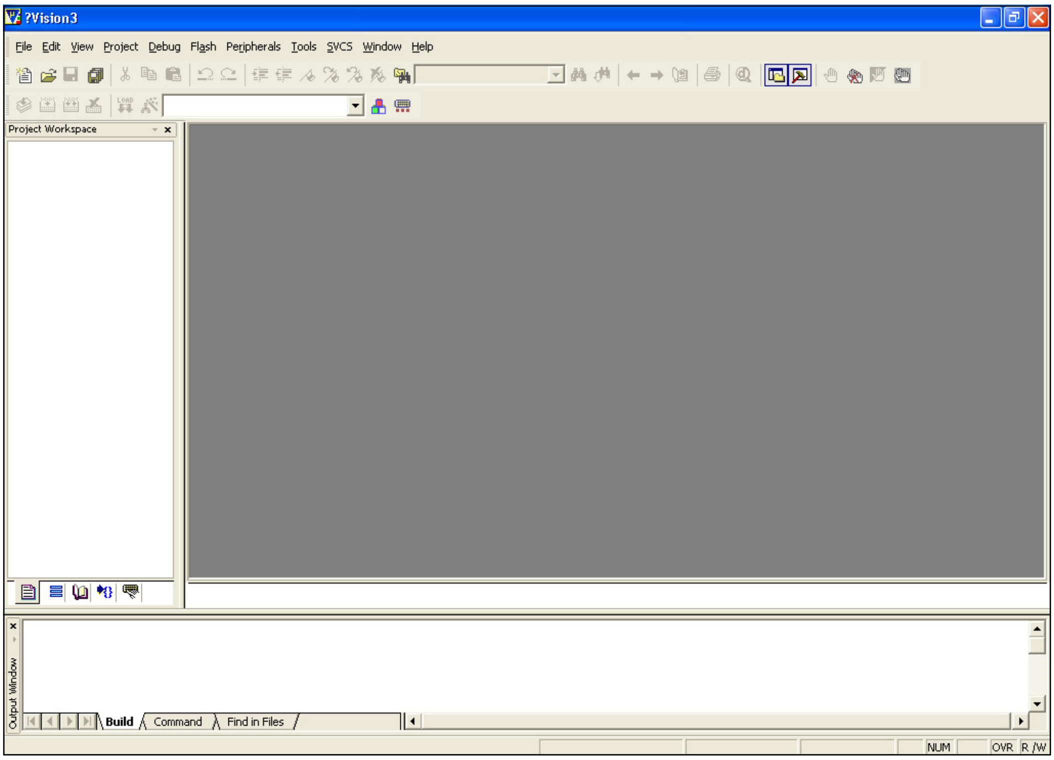


Figure 19 Shows the main window of Keil µVision3

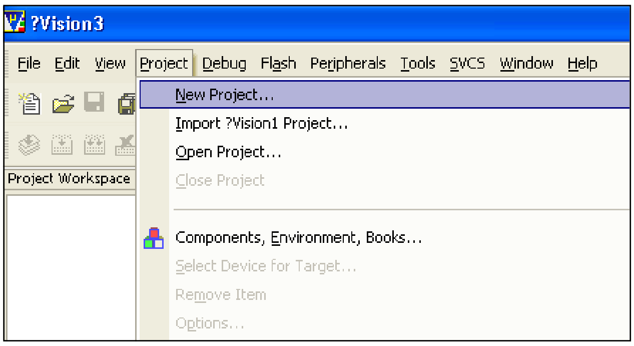


Figure 20 Shows the selection to make the new project.

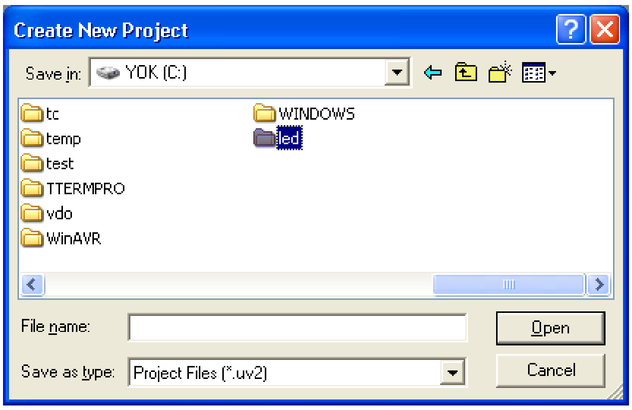


Figure 21 Shows the Create New Project window for making the new project.

(4) Enter to `led` folder. Define the filename as led following the figure 22. Click **Save** button to next step.

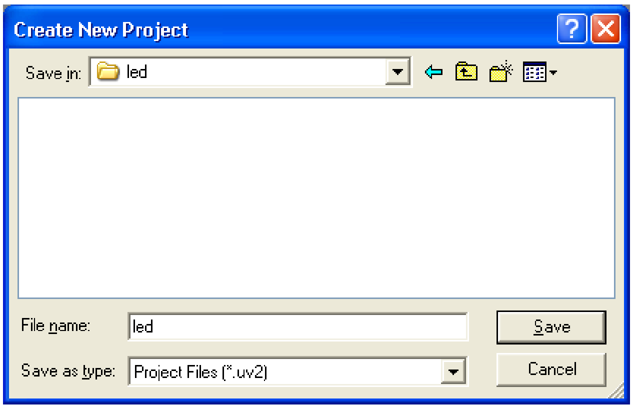


Figure 22 Define the project filename to led at Create New Project window.

(5) The ***Select Device for Target*** window will appear following the figure 23 for selection microcontroller from Manufacturer listing.

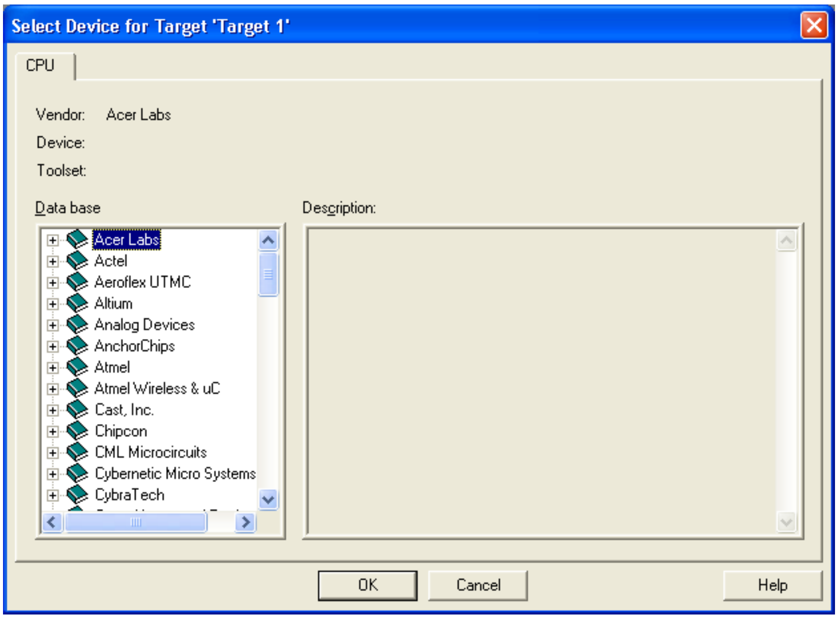


Figure 23 Shows the Select Device for Target window for microcontroller selection.

(6) Select the manufacturer in Data base header to Phillips and select LPC2148 . At Description header is on the right, shows the detail and properties of selected microcontroller following the figure 24. Click **OK** button to next step.

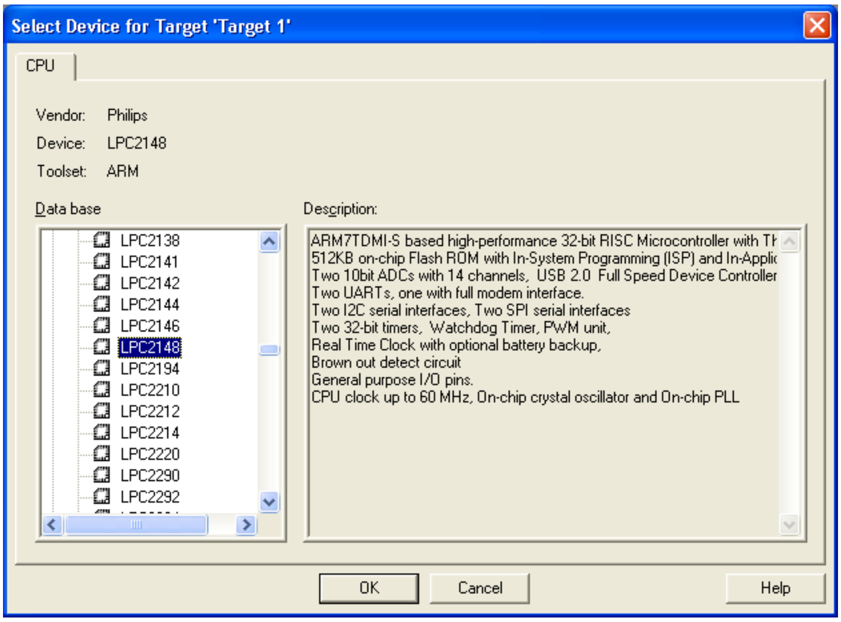


Figure 24 Shows the selection of Philips LPC2148

(7) After that you will see the dialog box ask about copying LPC2100 Startup code into your project following the figure 25. Click **No** button for denying. Developers can copy the Startup code that store in INEX's JX-2148 CD-ROM to this project later.

Figure 25 Asking about copy Startup Code to project.

(8) The project Workspace will appear following the figure 26.

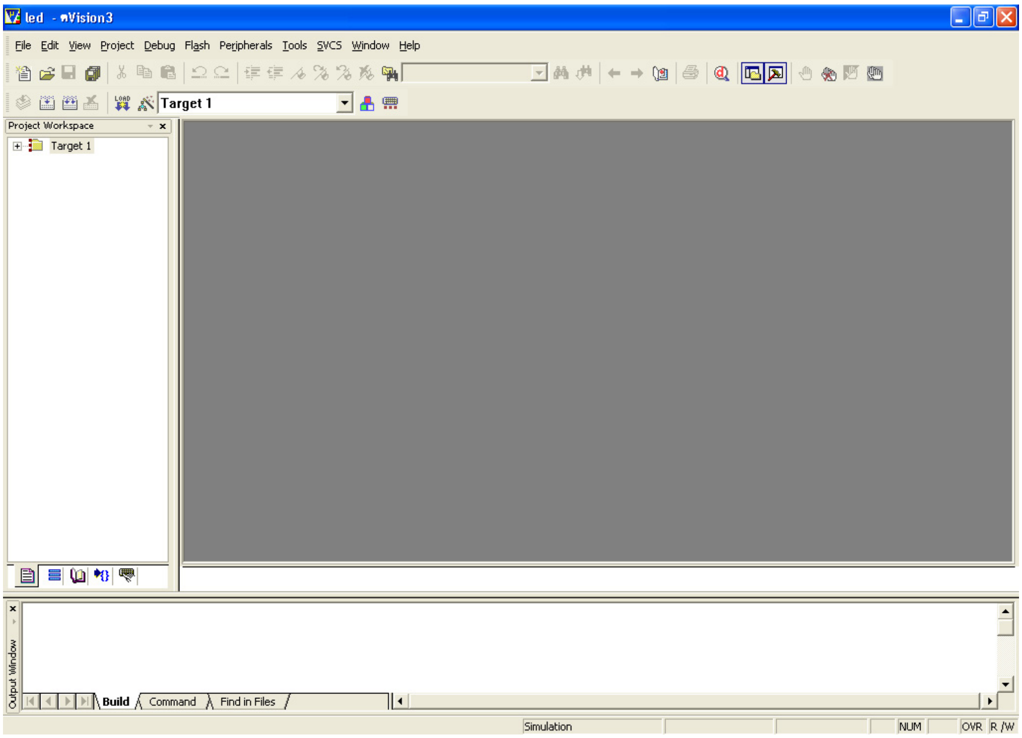


Figure 26 The Project Workspace window

(9) For more comfortable, copy the start up file includes **`main.c`** and **`Startup.s`** from keil_2148_system folder in CD-ROM (or download from www.inexglobal.com ; this start up file INEX modify for support all developers) to path of project. In ths example is **`C:\led`** following the figure 27.

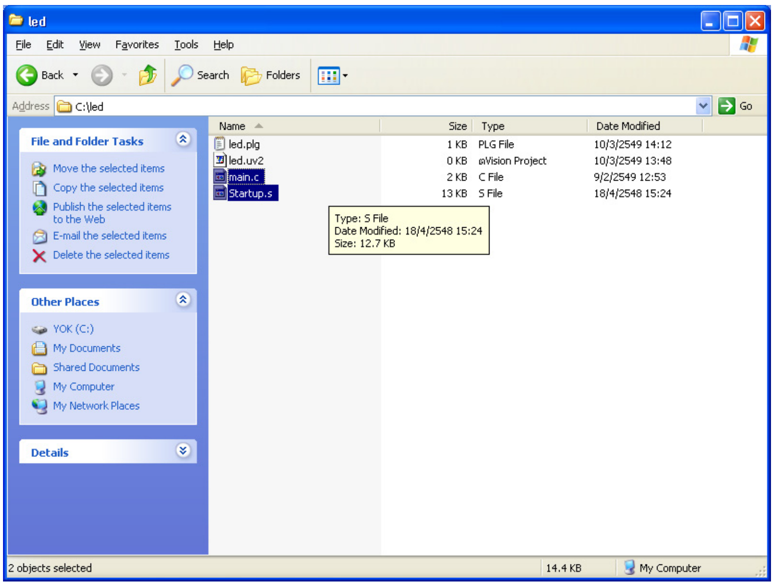


Figure 27 Shows the copying main.c and Startup.s file from keil_2148_system folder in CD-ROM to led Projectd

(10) Click at **+** sign front the ***Target1 project*** for preparing to add **main.c** and **Startup.s** file. These file will be the system files for developing the programs in the future. Then, found ***Source group 1*** file, click right button of mouse for select ***Add Files to Groups*** instruction to add **main.c** and **Startup.s** file into this project following the figure 28.

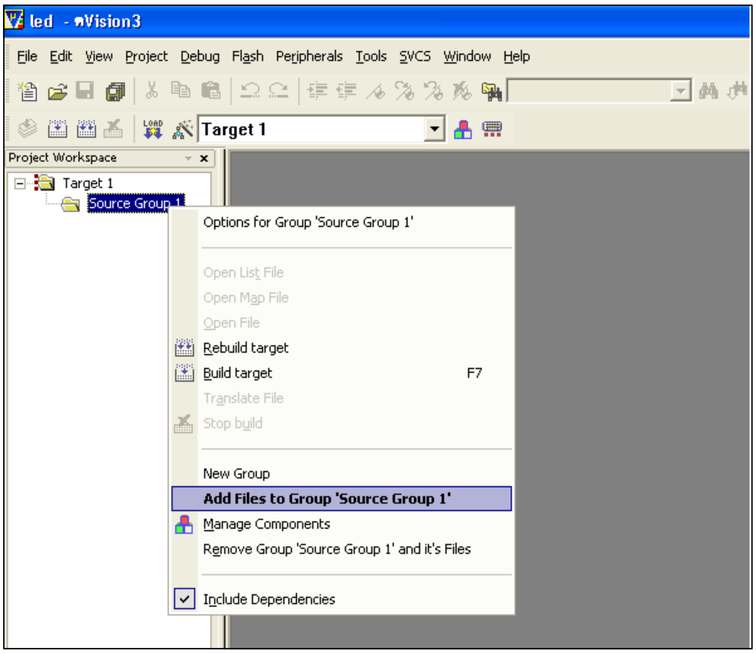


Figure 28 Adding file main.c and Startup.s into the project.

(11) The Add Files to Groups window will appear. Select path in **Look in:** box to **led** project path ; **C:\led**. Select File of type as All Files(*.*) for watching all files and can select **Startup.s** file following the figure 29.

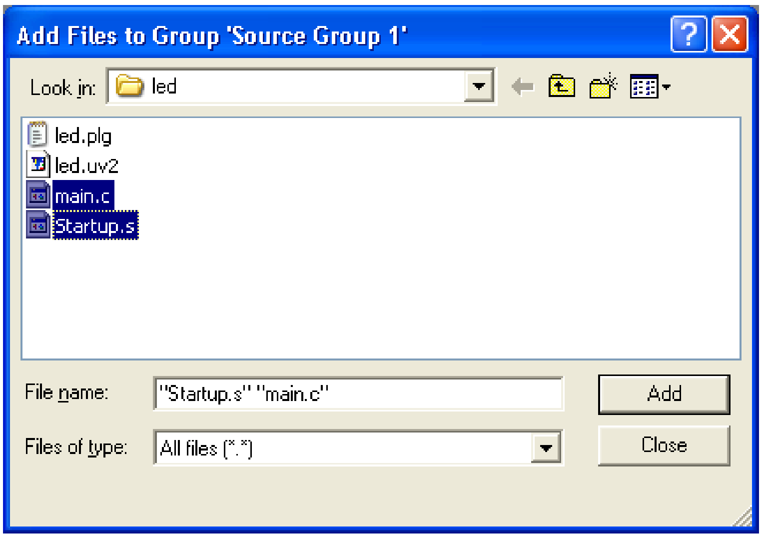


Figure 29 Shows the Add Files to Groups window to adding files.

(12) Press Crtl key and hold it. Click to select **main.c** and **Startup.s** file for adding files into the project. After that click **Add** button once. Both files will add into this project following the figure 30. Click **Close** button to next step.

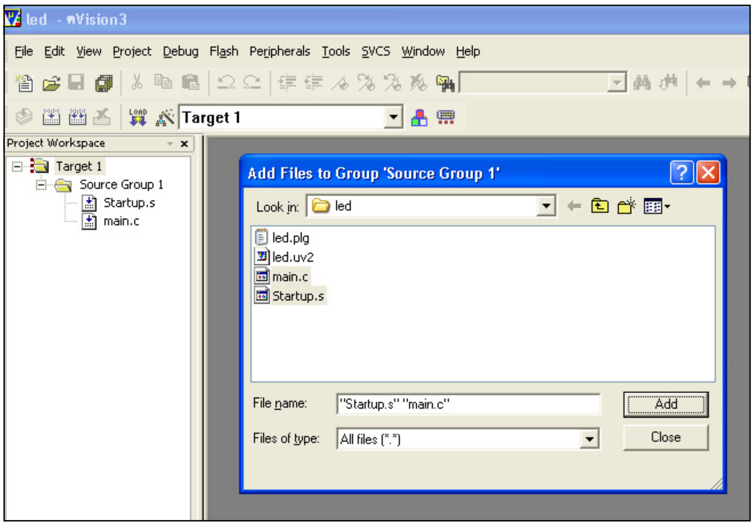


Figure 30 Shows adding file main.c and Startup.s into the project.

(13) Double-click at **main.c** file at ***Project Workspace window***. The editing window will appear. This window is called ***template***. Include init function to initialize the program operation relate with clock oscillator and PLL (Phase Lock Loop) within LPC2148. The main program will declare below. See the figure 31.

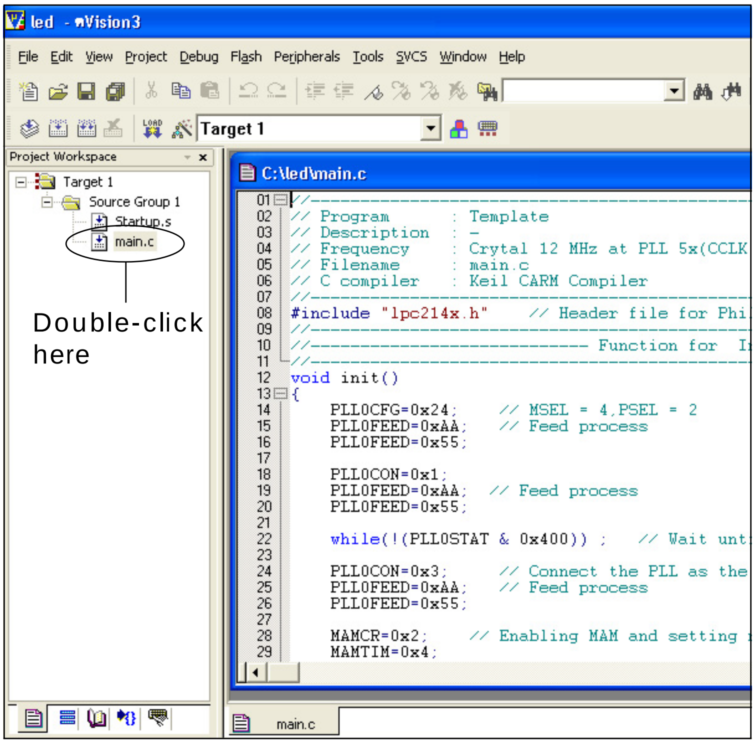


Figure 31 Shows the main.c file that adding into project.

(14) Write the program addition from main program. The contents of program can show in the figure 32.

```
}
//----------------------------------------------------------------------
//------------------------------------ Main Program ------------------
//----------------------------------------------------------------------
void main()
{
    long i;
    init();       // Initialize the system
    SCS = 0x03;  // select the "fast" version of the I/O ports
    FIO0DIR |= 0x00400000;
    while(1)
    {
        FIO0PIN ^= 0x00400000;
        for (i = 0; i < 1000000; i++ );
    }

}
```

Figure 32 Shows the C program that writes into mani program.

(15) Set the project's option to make the target file. Click **Option for Target** button at Tool bar following the figure 33. The steps of setting include :

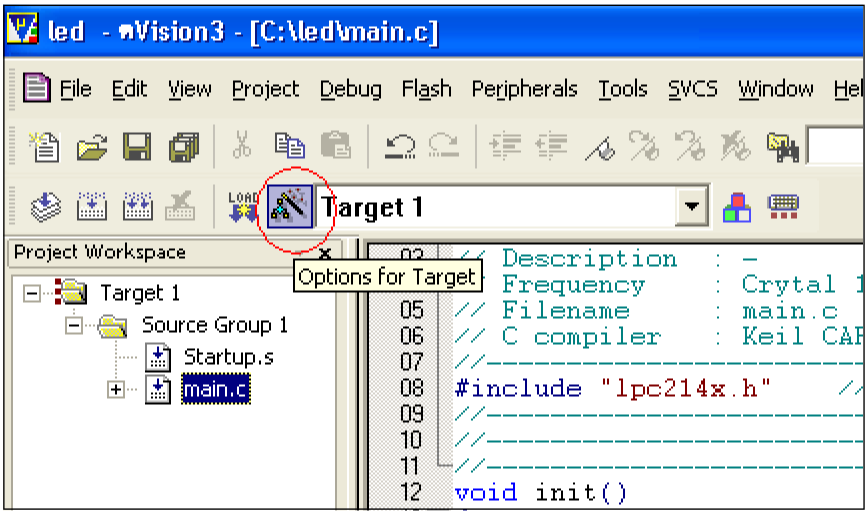


Figure 33 Shows position of Option for Target button at Tool Bar

(15.1) The ***Option for Target window*** will appear following the figure 34 . Select ***Output*** tab.

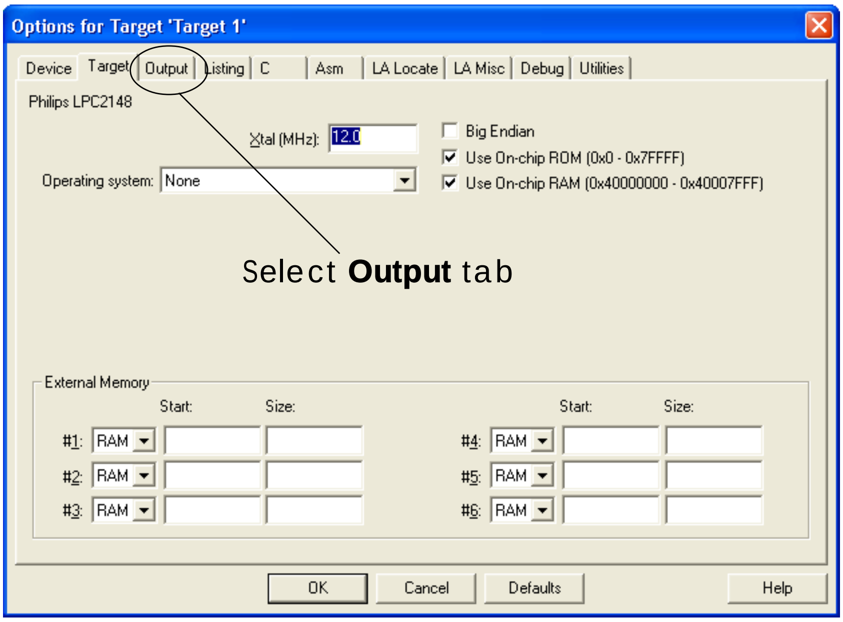


Figure 34 Shows the Option for Target window

(15.2) Click the listing following the figure 35.

- Select ***Create HEX file*** for making **`led.hex`** file after compile.
- Select ***HEX format:*** as *HEX-386*

After that click **OK** button.

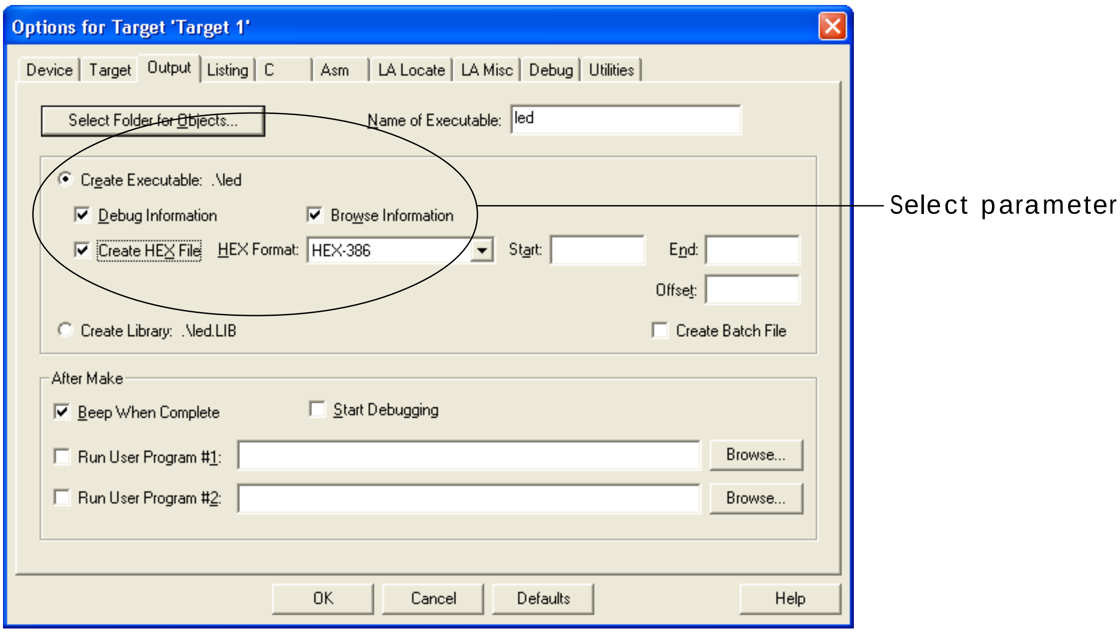


Figure 35 Parameter setting in Output tab of Option for Target window

(16) Compile project by select ***Project → Build Target*** or press ***F7*** key following the figure 36. The result is **`led.hex`** for downloading to MCU later.

Compiling result will show in text format at Output Window bottom section of main window following the figure 37. If compile complete, **`led.hex`** file will generate ans store in same folder of project. In this example is **`C:\led`**.

If deveopers would like to open the old project for develop continue, select at menu bar ***Project → Open Project*** and select path that store the previous project. The project file has **`*.uv2`** file. See the figure 38 for example.

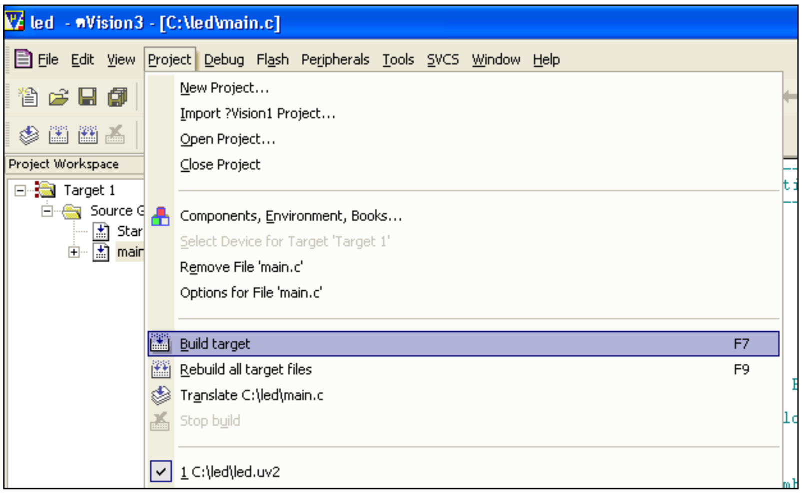


Figure 36 Compiling the program by select at Project → Build Target

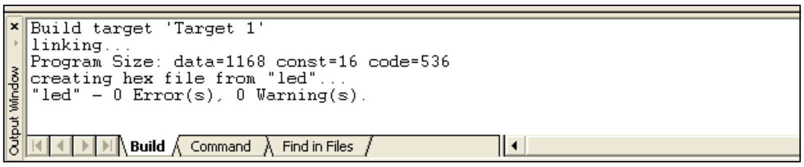


Figure 37 Compiling result at Output window

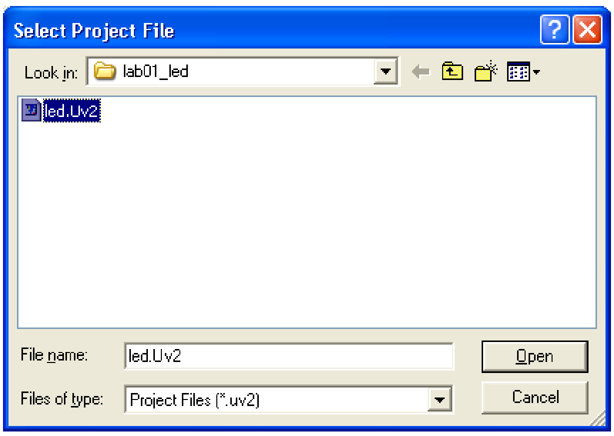


Figure 38 Select the previous file to edit or develop

8.2 Download programs and test

After compile the program, from example the result file is **`led.hex`**. Next step is downloading hex file to LPC2148 microcontroller and run it. Developers can check the operation at P0.22 LED. The downloading procedure is :

(1) Apply the supply voltage to JX-2148 board. Turn-on POWER switch.

(2) Connect download cable to the JX-2148 board and Serial port of computer.

(3) Open the LPC2000 Flash Utility software. The main window will appear following the figure 39.

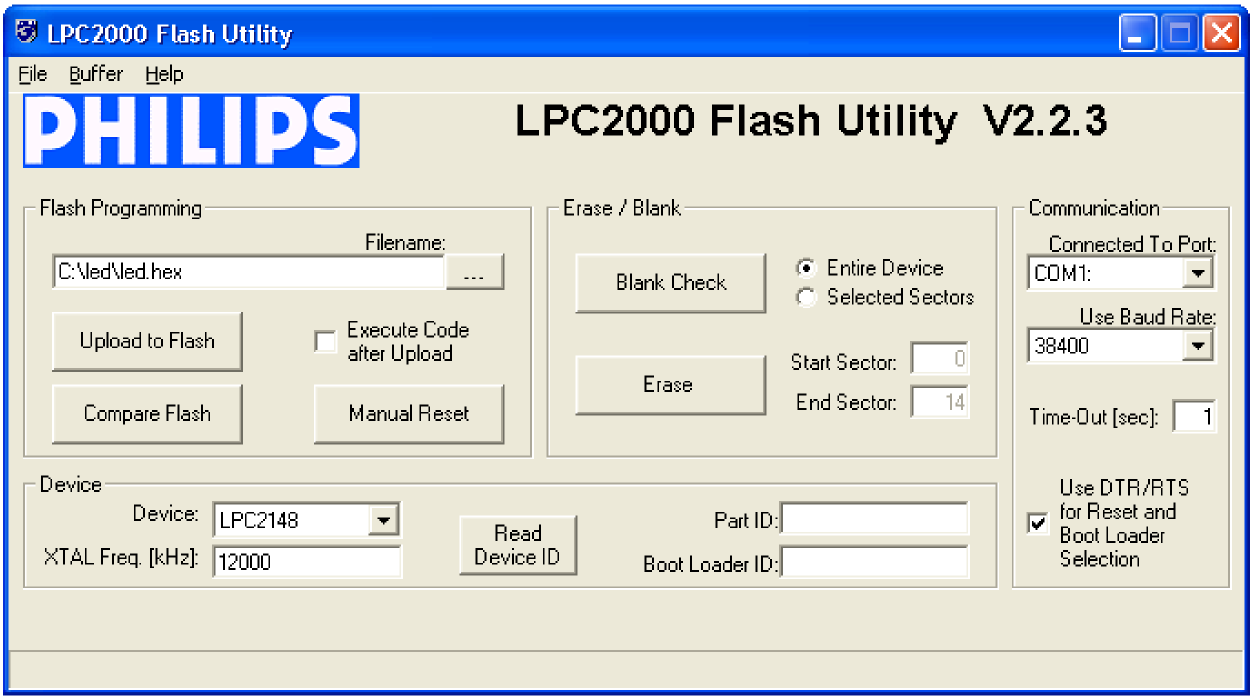


Figure 39 Main window of the LPC2000 Flash Utility

(4) In the first time, developers must set some parameter before

- At Device box set as :
  - **Device** *select to LPC2148*
  - **XTAL Freq. [kHz]** *set to 12000 (12MHz)*
- At Communication box set as :
  - **Connected To Port** *select the serial port is connected*
  - **Use Baud Rate** *set baudrate. 9,600 bps is default.*

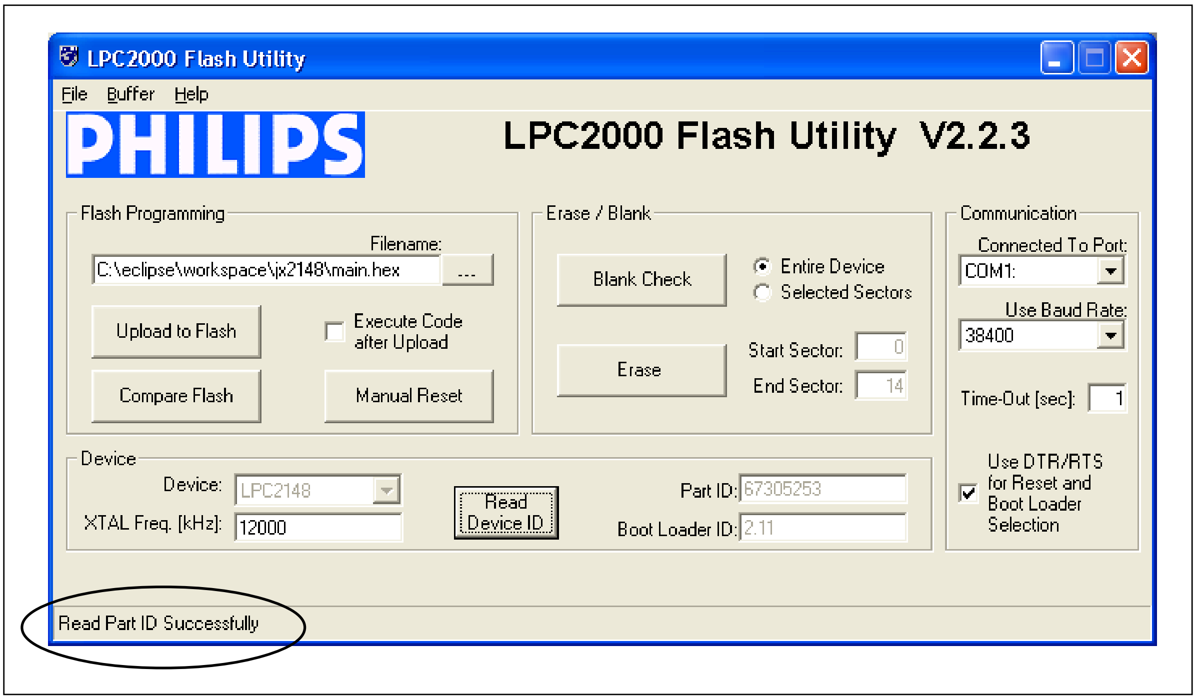


Figure 40 Shows the complete connection message at the status bar

(5) Set the JX-2148 board to ISP mode by press **ISP** SWITCH (its shaft will down) following press **RESET** switch once.

(6) At the main wondow of LPC2000 Flash Utility. Click at **Manual Reset button** once following click **Read Device ID** once too. If the connection is correct, status bar at left down conner will show message ***Read Part ID Successfully*** following the figure 40

If connection fail, the warning window will appear following the figure 41. Click **Read Device ID button** again and see the result. May be back to do in step 5 and 6 again.

Figure 41 Warning message about connection failed

(7) If connection complete, click at Browse button in Filename of Flash Programming. Select **`led.hex`** in path **`C:\led\led.hex`**. Next, click **Upload to Flash button** to download. After downloading complete the message ***`File Upload Successfully Completed`*** will show at the status bar.

(8) Developers can test the program by press ISP switch again to RUN mode (The switch's shaft will release) and press RESET switch once. Observe the operation.

LED at P0.22 blink continuous.

inex
INNOVATIVE EXPERIMENT

9. Sample Experiment

Experiment -1 : Output port

This is simple experiment. This experiment is testing output port function of LPC2148 to drive 2 LED at P0.21 and P0.22.

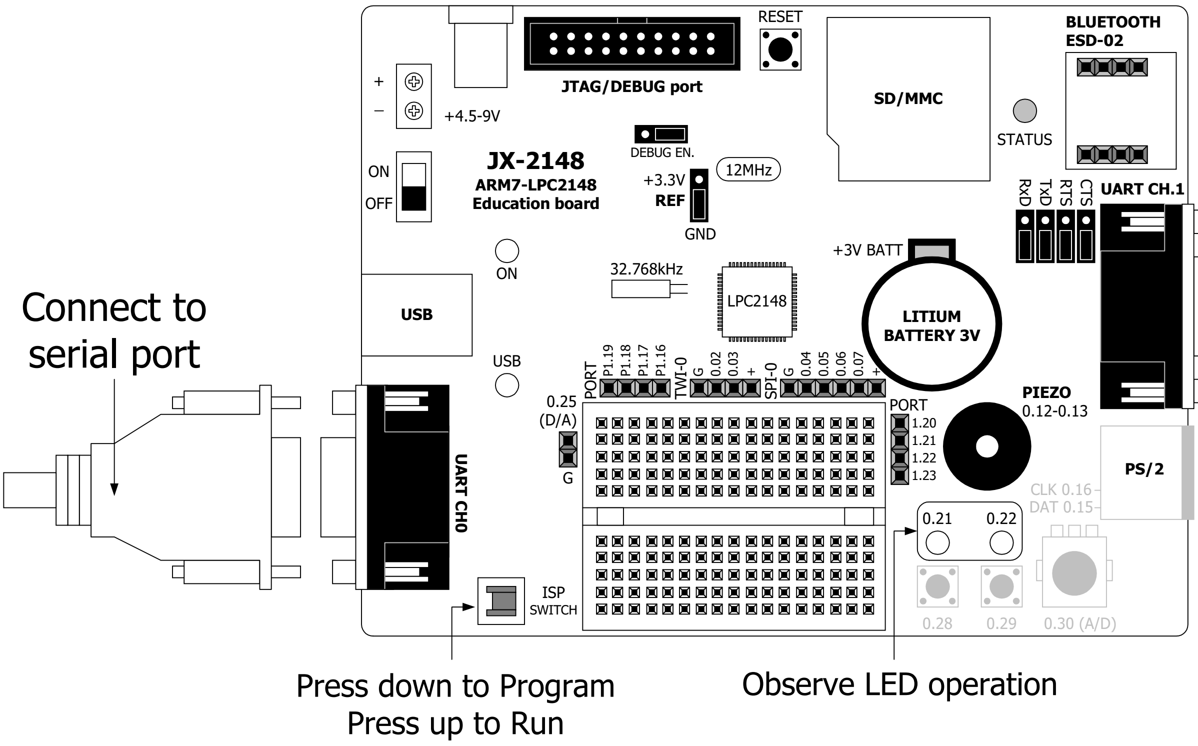
Procedure :

1.1 Build new project, in name **`led`**

1.2 Write the program Listing P1-1. Compile to **`led.hex`** and download to microcontroller.

1.3 Run program. Observe the operation of LED at P0.21 and P0.22

Both LED will blink every 0.5 second.



```
//————————————————————————————————————————————————————————————————————//
// Program     : Example for display LED
// Description : LED Blink toggle at P0.21 and P0.22
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename    : led.c
// C compiler  : Keil CARM Compiler
//————————————————————————————————————————————————————————————————————//
#include "lpc214x.h"          // Header file for Phillips LPC2148 controller
#define LED2_ON FIO0CLR = 0x00200000     // Red led 0.21 on
#define LED2_OFF FIO0SET = 0x00200000    // Red led 0.21 off

//——————————————— Function for  Initial system clock ————————————————//
void init()
{
  PLL0CFG=0x24;               // MSEL = 4,PSEL = 2
  PLL0FEED=0xAA;              // Feed process
  PLL0FEED=0x55;
  PLL0CON=0x1;
  PLL0FEED=0xAA;              // Feed process
  PLL0FEED=0x55;

  while(!(PLL0STAT & 0x400));// Wait until PLL Locked

  PLL0CON=0x3;                // Connect the PLL as the clock source
  PLL0FEED=0xAA;              // Feed process
  PLL0FEED=0x55;

  MAMCR=0x2;                  // Enabling MAM and setting number of clocks used
                              // for Flash memory fetch (4 cclks in this case)
  MAMTIM=0x4;
  VPBDIV=0x02;                // PCLK at 30 MHz
}
//—————————————————— Function delay ———————————————————————//
void delay_ms(long ms)       // delay 1 ms per count @ CCLK 60 MHz
{
  long i,j;
  for (i = 0; i < ms; i++ )
  for (j = 0; j < 6659; j++ );
}
//—————————————————— Main Program ————————————————————————//
void main()
{
  init();                     // Initialize the system
  SCS = 0x03;                 // select the "fast" version of the I/O ports
  FIO0DIR |= 0x00600000;      // Config. pin P0.22 and P0.21 as output
  while (1)                   // Infinite loop
  {
     LED1_ON;                 // Led at P0.22 ON
     LED2_OFF;                // Led at P0.21 OFF
     delay_ms(500);           // Delay 500 ms
     LED2_ON;                 // Led at P0.21 ON
     LED1_OFF;                // Led at P0.22 OFF
     delay_ms(500);           // Delay 500 ms
  }
}
```

Listing P1-1 : **`led.c`** file of **`led`** project for LPC2148 output port experiment (continue)

More information about important function in this program

init Increase processing spped 5 times from 12MHz main crystal to 60MHz with PLL x5. After execute this function, CCLK increase to 60MHz.

delay_ms Delay function in Millisecond unit. This function will refer with 60MHz CCLK and init function.

Note : *All experiment in this manual will use PLL x5 to increase the CCLK to 60MHz. Then developer must exceute* init *function in the top of program always.*

Listing P1-1 : **led.c** file of **led** project for LPC2148 output port experiment (final)

Experiment -2 : Input port

In this experiment introduce using LPC2148 port acts input for reading data from P0.28 and P0.29 that connected with push-button switches and resistor pull-up.

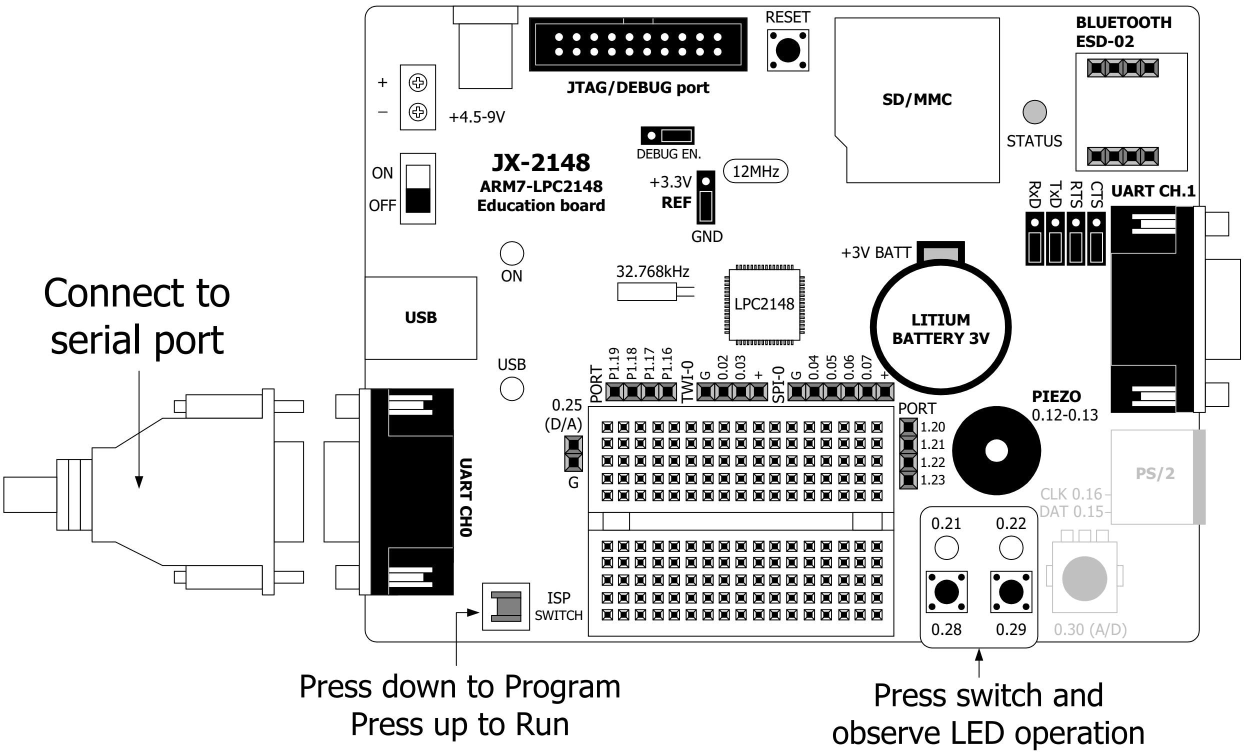
Procedure :

2.1 Build new project, in name **`switch`**

2.2 Write the program Listing P2-1. Compile to **`switch.hex`** and download to microcontroller.

2.3 Run the program. Try to press switch at P0.28 and P0.29. Watch the operation of LED at P0.21 and P0.22.

P0.28 switch will control LED at P0.21 and P0.29 switch will control LED at P0.22



```
//————————————————————————————————————————————————————————————————————————//
// Program     : Example for test switch
// Description : Test switch at P0.28 and P0.29 for toggle LED at P0.21 and P0.22
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename    : switch.c
// C compiler  : Keil CARM Compiler
//————————————————————————————————————————————————————————————————————————//
#include "lpc214x.h"   // Header file for Phillips LPC2148 controller

//————————————— Function for  Initial system clock ————————————————//
void init()
{
  PLL0CFG=0x24;               // MSEL = 4,PSEL = 2
  PLL0FEED=0xAA;              // Feed process
  PLL0FEED=0x55;

  PLL0CON=0x1;
  PLL0FEED=0xAA;              // Feed process
  PLL0FEED=0x55;

  while(!(PLL0STAT & 0x400));// Wait until PLL Locked

  PLL0CON=0x3;                // Connect the PLL as the clock source
  PLL0FEED=0xAA;              // Feed process
  PLL0FEED=0x55;

  MAMCR=0x2;                  // Enabling MAM and setting number of clocks used
                              // for Flash memory fetch (4 cclks in this case)
  MAMTIM=0x4;

  VPBDIV=0x02;                // PCLK at 30 MHz
}

//———————————————— Function Read input P0 ——————————————————//
char inp0(char _bit)
{
  unsigned long c;
  c = 1<<_bit;                // Calculate digit to configuration for input port
  FIO0DIR &= ~c;              // Set input port from parameter _bit
  return((FIO0PIN & c)>>_bit);     // Read and return data bit
}

//———————————————— Main Program ————————————————————————//
void main()
{
  init();                     // Initialize the system
  SCS = 0x03;                 // select the "fast" version of the I/O ports
  FIO0DIR |= 0x00600000;      // P0.21 and P0.22 for output port
  FIO0SET |= 0x00600000;      // OFF led P0.21 and P0.22 for initial
  while (1)                   // Infinite loop
```

Listing P2-1 : **`switch.c`** file of **`switch`** project for LPC2148 input port experiment (continue)

```
  {
    if(inp0(28)==0)                  // Check switch P0.28 push?
    {
      while(inp0(28)==0);            // Wait until release switch P0.28
      FIO0PIN ^= (1<<21);            // Toggle led at P0.21
    }
    if(inp0(29)==0)                  // Check switch P0.29 push?
    {
      while(inp0(29)==0);            // Wait until release switch P0.29
      FIO0PIN ^= (1<<22);            // Toggle led at P0.22
    }
  }
}
```

More information about important function in this program

`inp0` Read data bit from Port 0. The read bit position is agument.

syntax :

```
char inp0(char _bit)
```

parameter :

```
_bit  define bit 0 of Port 0
```

return :

0 or 1 (In case connected switch, switch is pressed - return 0 and exit)

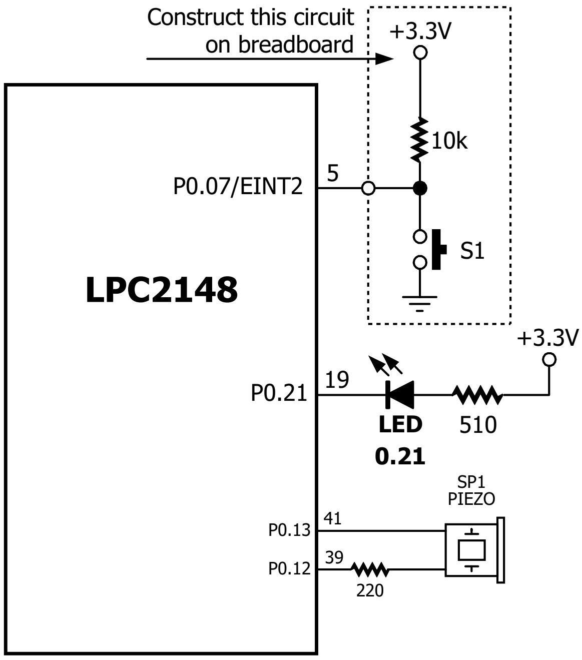
example :

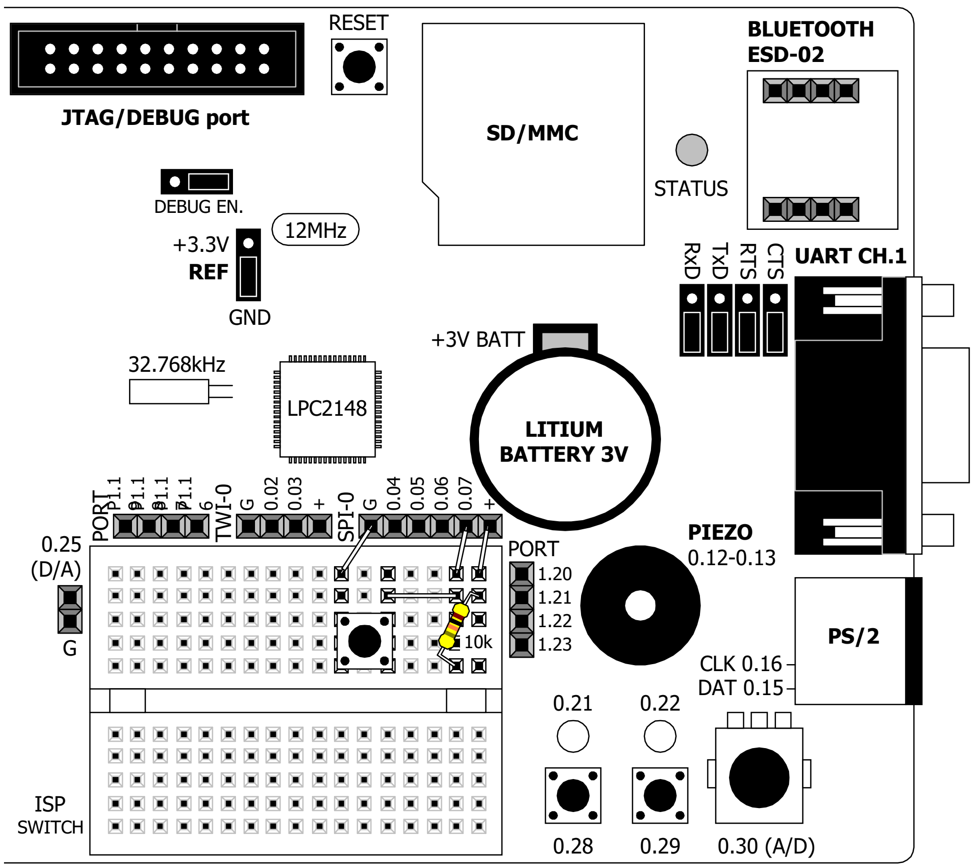
```
char x;
x = inp0(28);
```

Listing P2-1 : **`switch.c`** file of **`switch`** project for LPC2148 input port experiment (final)

Experiment -3 : External Interrupt

This experiment demonstrate how to use External interrupt input port at EINT2 pin (P0.07). Need to construct the addition circuit on breaboard of JX-2148 board below :





Procedure :

3.1 Build new project, in name **ext_int2**

3.2 Write the program Listing P3-1. Compile to **ext_int2.hex** and download to microcontroller.

3.3 Run program. Try to press swtich at P0.07 or EINT2 port pin. Observe the operation of LED at P0.21 and Piezo.

Switch P0.07 pressing is used to control LED at P0.21 and generate sound for informing the external interrupt is happen.

```
//————————————————————————————————————————————————————————————————————————//
// Program     : Example for EINT2(External interrupt2)
// Description : Test interrupt from switch press display by LED at P0.21
//             : toggle and sound beep
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename    : ext_int2.c
// C compiler  : Keil CARM Compiler
//————————————————————————————————————————————————————————————————————————//
#include "lpc214x.h"   // Header file for Phillips LPC2148 controller
#include "sound.h"     // Header file for Phillips LPC2148 controller

//——————————————— Function for  Initial system clock —————————————————//
void init()
{
  PLL0CFG=0x24;                     // MSEL = 4,PSEL = 2
  PLL0FEED=0xAA;                    // Feed process
  PLL0FEED=0x55;

  PLL0CON=0x1;
  PLL0FEED=0xAA;                    // Feed process
  PLL0FEED=0x55;

  while(!(PLL0STAT & 0x400)) ;      // Wait until PLL Locked

  PLL0CON=0x3;                      // Connect the PLL as the clock source
  PLL0FEED=0xAA;                    // Feed process
  PLL0FEED=0x55;

  MAMCR=0x2;                  // Enabling MAM and setting number of clocks used
                              // for Flash memory fetch (4 cclks in this case)
  MAMTIM=0x4;
  VPBDIV=0x02;                      // PCLK at 30 MHz
}
```

Listing P3-1 : **ext_int2.c** file of **ext_int2** project for LPC2148 external interrupt port experiment (continue)

```
//—————————————— Interrupt service routine for EINT2 ———————————————//
void isr_int2(void) __irq
{
    beep();                    // Sound beep 1 time
    FIO0PIN ^= (1<<21);        // Toggle LED at P0.21
    EXTINT |= 0x4;             // Clear interrupt flag EINT2
    VICVectAddr = 0;           // Acknowledge Interrupt
}

//————————————————— Main Program ————————————————————————//
void main()
{
  init();                      // Initialize the system
  SCS = 0x03;                  // select the "fast" version of the I/O ports
  FIO0DIR |= (1<<21);          // Config. output P0.21 connect LED
  FIO0SET |= (1<<21);          // OFF LED
  EXTMODE |= 0x4;              // EINT2 Edge sensitive detection
                               // (Falling edge in this program)
  PINSEL0 = 0xC000;            // Enable EINT2 at P0.7
  VICVectAddr0 = (unsigned)isr_int2;
                               // Register Interrupt service routine name
  VICVectCntl0 = 0x20 | 16;    // EINT2 Interrupt
  VICIntEnable |= 1 << 16;     // Enable EINT2 Interrupt
  while(1);                    // Infinite loop
}
```

More information about important function in this program

beep Generate beep sound one time. This function is add in sound.h library.

syntax :

```
void beep()
```

Listing P4-1 : **`ext_int2.c`** file of **`ext_int2`** project for LPC2148 external interrupt port experiment (final)

Experiment -4 : UART

uart.h library

In this experiment must include an important library file ; uart.h. This library will set the UART module in LPC2148. The source program of this library file in shown in Listing P4-1

Function that added in uart.h library has detail as :

uart1_init

Set the baudrate for UART1 module.

syntax :

```
void uart1_init(unsigned int _baudrate)
```

parameter :

`_baudrate` set baudrate value.

uart1_putc

Send a character to Transmitter buffer of UART1 module.

syntax :

```
void uart1_putc(char c)
```

parameter :

`c` define the transmit character

uart1_puts

Transmit string out to UART1 module

syntax :

```
void uart1_puts(char *p)
```

parameter :

`p` Index the transmit character

uart0_init

Set the baudrate for UART0 module.

syntax :

```
void uart0_init(unsigned int _baudrate)
```

parameter :

`_baudrate` set baudrate value.

uart0_putc

Send a character to Transmitter buffer of UART0 module.

syntax :

```
void uart0_putc(char c)
```

parameter :

`c` define the transmit character

uart0_puts

Transmit string out to UART0 module

syntax :

```
void uart0_puts(char *p)
```

parameter :

`p` Index the transmit character

Note : *For only UART1 module, developers may be execute the many support functions about data communication that added in* **`stdio.h`** *library such as* `printf`, `puts`, `getchar`, `scanf` *etc...*

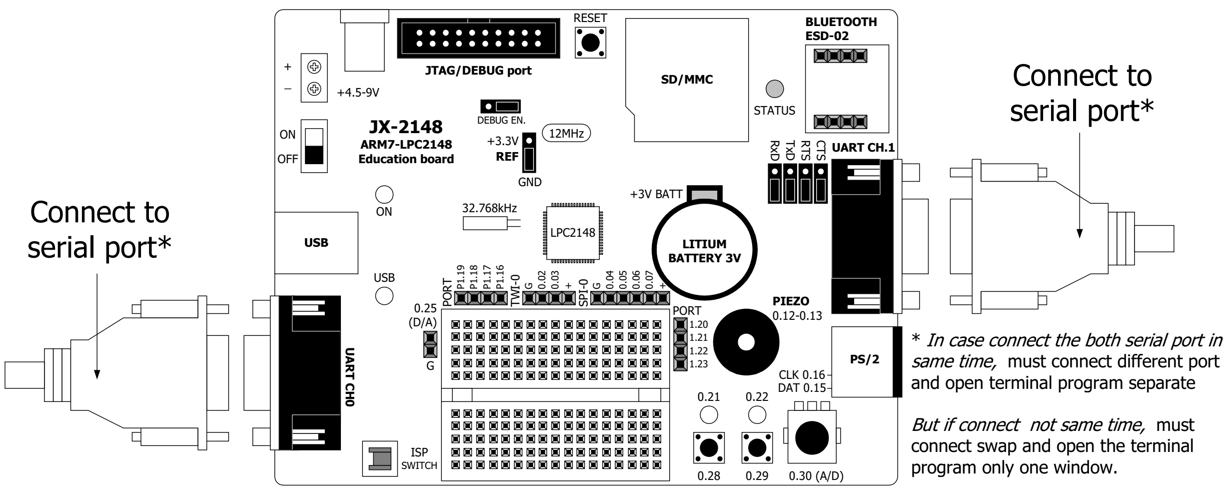
Experiment -4.1 Transmit data with UART0/UART1

In this experiment is demonstration about transmit data from UART0/UART1 module in LPC2148 by using the serila port interface circuit on the JX-2148.

Procedure :

4.1.1 Build new project, in name **`uart_0_1`**

4.1.2 Write the program Listing P4-2. Compile to **`uart_0_1.hex`** and download to microcontroller.



```
//-----------------------------------------------------------------------------------//
// Program       : Library for serial communication(UART)
// Description   : Library for about serial communication UART0 and UART1 of LPC2148
// Frequency     : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename      : uart.h
// C compiler    : Keil CARM Compiler
//-----------------------------------------------------------------------------------//
#define _PCLK 30000000                      // Define PCLK for configuration baudrate
#define uart1_setbaud(x) uart1_init(x)      // Define function name uart1_setbaud equal uart1_init
#define uart0_setbaud(x) uart0_init(x)      // Define function name uart0_setbaud equal uart0_init

//--------------- Function for Initial UART1 --------------------//
void uart1_init(unsigned int _baudrate)
{
  unsigned short u1dl;
  u1dl = _PCLK/(16*_baudrate);              // Calculate for U1DL value
  PINSEL0 |= 0x00050000;                    // Enable rx,tx
  U1LCR = 0x00000083;                       // 8 bit data,1 stop bit,no parity bit
  U1DLL = u1dl & 0xFF;                      // U1DL for low byte
  U1DLM = (u1dl>>8);                        // U1DL for high byte
  U1LCR = 0x00000003;                       // DLAB =0
}
//--------------- 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(unsigned int _baudrate)
{
  unsigned short u0dl;
  u0dl = _PCLK/(16*_baudrate);              // Calculate for U0DL value
  PINSEL0 |= 0x00000005;                    // Enable rx,tx
  U0LCR = 0x00000083;                       // 8 bit data,1 stop bit,no parity bit
  U0DLL = u0dl & 0xFF;                      // U0DL for low byte
  U0DLM = (u0dl>>8);                        // U0DL for high byte
  U0LCR = 0x00000003;                       // 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
  }
}
```

Listing P4-1 : **`uart.h`** The UART library file for LPC2148

```
//————————————————————————————————————————————————————————————————————————//
// Program     : UART example
// Description : Example for test module UART0 and UART1
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename    : uart_0_1.c
// C compiler  : Keil CARM Compiler
//————————————————————————————————————————————————————————————————————————//
#include "lpc214x.h"    // Header file for Phillips LPC2148 controller
#include "uart.h"       // Library for module UART0,UART1(from jx2148_include folder)
#include "stdio.h"      // Library for use printf function(For UART1)

//————————————————— Initial system clock ————————————————————//
void init()
{
  PLL0CFG=0x24;         // MSEL = 4,PSEL = 2
  PLL0FEED=0xAA;        // Feed process
  PLL0FEED=0x55;

  PLL0CON=0x1;
  PLL0FEED=0xAA;        // Feed process
  PLL0FEED=0x55;

  while(!(PLL0STAT & 0x400)) ;      // Wait until PLL Locked

  PLL0CON=0x3;                      // Connect the PLL as the clock source
  PLL0FEED=0xAA;                    // Feed process
  PLL0FEED=0x55;

  MAMCR=0x2;            // Enabling MAM and setting number of clocks used
                        // for Flash memory fetch (4 cclks in this case)
  MAMTIM=0x4;

  VPBDIV=0x02;          // PCLK at 30 MHz
}

//————————————————— Main Program ————————————————————————//
void main()
{
  init();               // Initialize the system
  SCS = 0x03;           // select the "fast" version of the I/O ports
  uart0_init(9600);
          // Initial UART0 @ 9600 bps,8 bit data ,1 stop bit ,no parity bit
  uart1_init(9600);
          // Initial UART1 @ 9600 bps,8 bit data ,1 stop bit ,no parity bit
  while (1)
  {
     uart0_puts("Test UART0\r\n");       // Send string to UART0
     printf("Test UART1\r\n");           // Send string to UART1
  }

}
```

Listing P4-2 : **`uart_0_1.c`** file of **`uart_0_1`** project for LPC2148 UART experiment

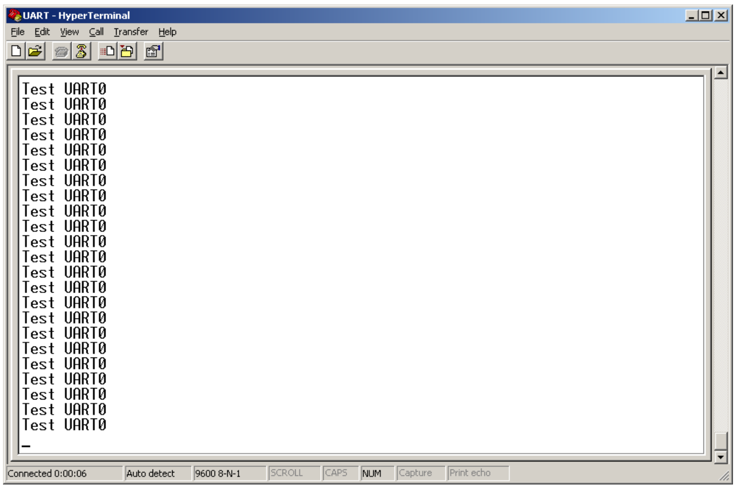


Figure P4-1 Shows UART0 operation at Hyper Terminal window

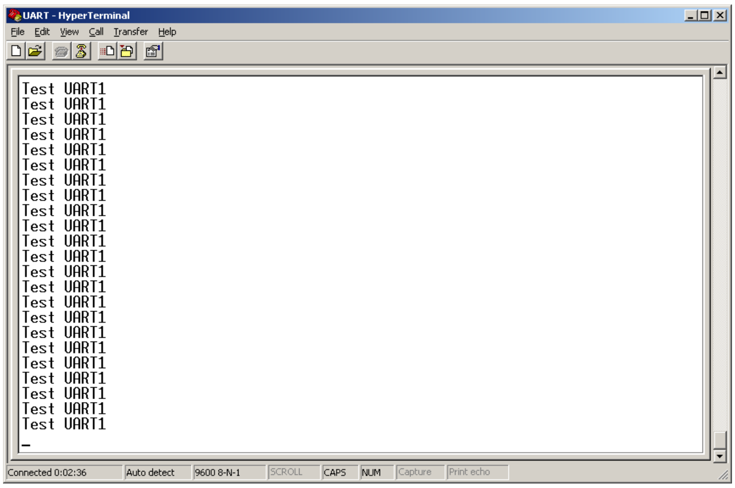


Figure P4-2 Shows UART1 operation at Hyper Terminal window

4.1.3 Open the Terminal program such as Hyper terminal or RS-232 Terminal (download at www.inexglobal.com) to test th e operation. Set baudrate to 9,600 bit per second.

4.1.4 Connect serial port cable from computer RS-232 port to UART CH.0 connector on JX-2148 board. If have more serial port, connect the serial cable to UART CH.1 connector.

4.1.5 Run the program. Watch the result on the terminal program operation.

Message on the Terminal panel will show message :

`Test UART0` *If connect between UART CH.0 with RS-232 serial port*

`Test UART1` *If connect between UART CH.1 with RS-232 serial port*

Experiment - 4.2 : UART operation with interrupt

This experiment demonstrates the UART1 operation with interrupt.

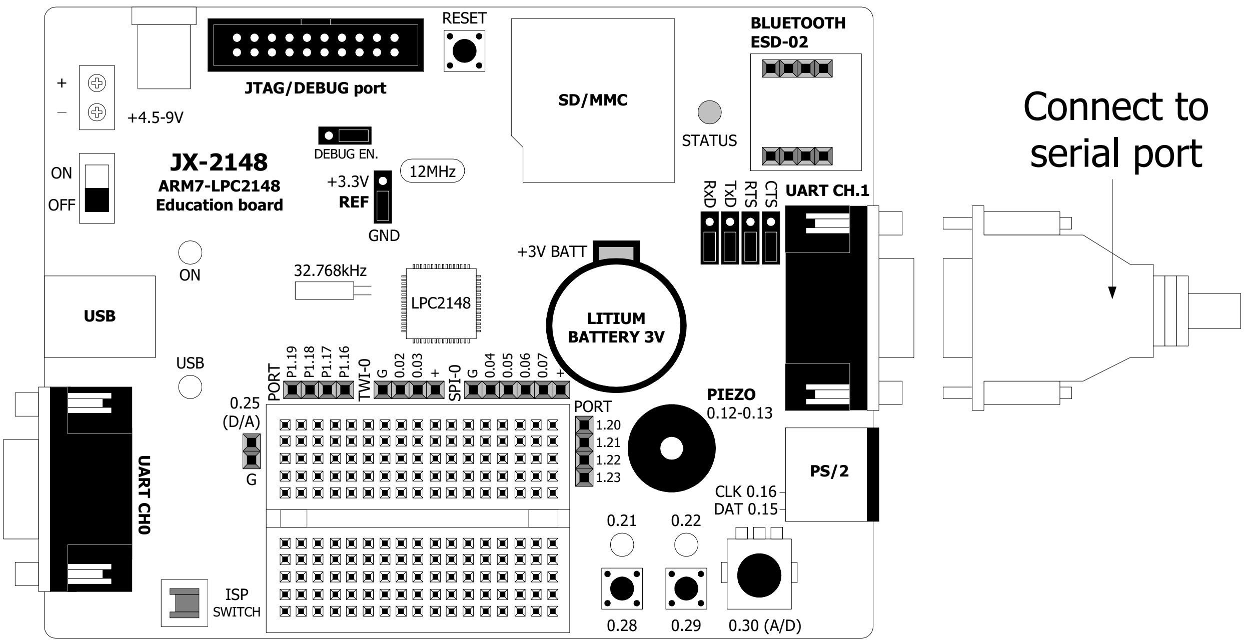
Procedure :

4.2.1 Build new project, in name **`uart1_int.`**

4.2.2 Write the program Listing P4-3. Compile to **`uart1_int.hex`** and download to microcontroller.

4.2.3 Open the Terminal program such as Hyper terminal or RS-232 Terminal to test the operation. Set baudrate to 9,600 bit per second.

4.2.4 Connect serial port cable from computer RS-232 port to UARTCH.1 connector.



4.2.5 Run the program.Watch the result on the terminal program operation.

The termional window shows message below :

`Now test UART1 for echo character`

`Press any key for test!`

Try to press any keyboard button for transmitting that character to LPC2148. This transmit enable echo function (return character back). Thus, transmitted character will send back to serial port and appear at the terminal program window.

```
//————————————————————————————————————————————————————————————————————————————————//
// Program : UART example
// Description : Example for test UART1 interrupt mode
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename : uart_int.c
// C compiler  : Keil CARM Compiler
//————————————————————————————————————————————————————————————————————————————————//
#include "lpc214x.h"   // Header file for Phillips LPC2148 controller
#include "uart.h"      // Library for use module UART0,UART1
#include "stdio.h"     // Library for use puts function(For UART1)

//——————————————— Function for  Initial system clock —————————————————//
void init()
{
  PLL0CFG=0x24;         // MSEL = 4,PSEL = 2
  PLL0FEED=0xAA;        // Feed process
  PLL0FEED=0x55;

  PLL0CON=0x1;
  PLL0FEED=0xAA;        // Feed process
  PLL0FEED=0x55;

  while(!(PLL0STAT & 0x400)) ;      // Wait until PLL Locked

  PLL0CON=0x3;                      // Connect the PLL as the clock source
  PLL0FEED=0xAA;                    // Feed process
  PLL0FEED=0x55;

  MAMCR=0x2;          // Enabling MAM and setting number of clocks used for
                      // Flash memory fetch (4 cclks in this case)
  MAMTIM=0x4;
  VPBDIV=0x02;                      // PCLK at 30 MHz
}

//——————————————— Interrupt service routine for UART1 ————————————————//
void isr_uart1(void) __irq
{
    char msg;
    if(((msg = U1IIR) & 0x01) == 0) // Check status flag communication
    {
        switch (msg & 0x0E)         // Filter message
        {
            case 0x04: while(!(U1LSR & 0x20)); // Receive Data Available
                       U1THR = U1RBR;          // Echo character
                       break;
            case 0x02: break;                  // Their Interrupt
            default:    break;                 // Other
        }
    }
    VICVectAddr = 0;                           // Acknowledge Interrupt
}
```

Listing P4-3 : **`uart1_int.c`** file of **`uart1_int`** project for LPC2148 UART experiment (continue)

```
//—————————————— Main Program ——————————————————//
void main()
{
  init();                 // Initialize the system
  SCS = 0x03;             // select the "fast" version of the I/O ports
  uart1_init(9600);       // Initial UART1 @9600 bps, 8 bit data, 1 stop bit, no parity
  U1IER = 3;              // Enable rx/tx interrupt for UART1
  PINSEL0 |= (1<<18);     // Enable RXD1(from UART1) at P0.9
  VICVectAddr0 = (unsigned)isr_uart1;
                          // Register Interrupt service routine name
  VICVectCntl0 = 0x20 | 7;                    // UART1 Interrupt
  VICIntEnable = 1 << 7;                      // Enable UART1 Interrupt

  puts("Now test UART1 for echo character\n");
                          // Display message for test echo character
  puts("Press any key for test!\n");
  while(1);               // Infinite loop
}
```

Listing P4-3 : **`uart1_int.c`** file of **`uart1_int`** project for LPC2148 UART experiment with interrupt (final)

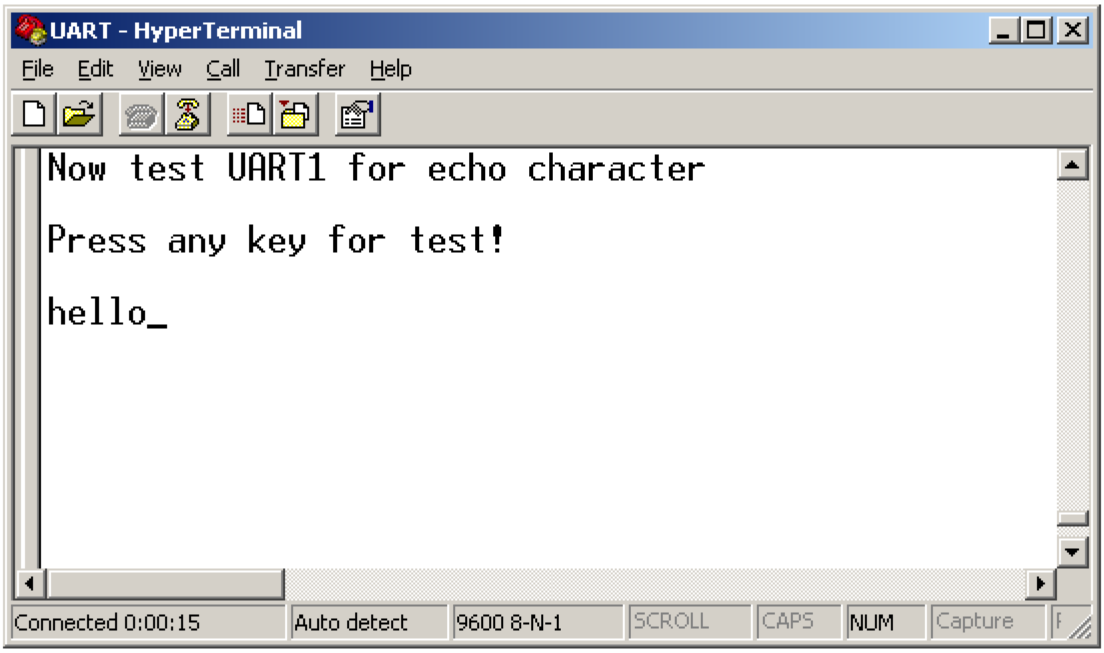


Figure P4-3 Shows the UART1 module data communication function with interrupt

Experiment -5 : A/D converter

This experiment is demonstration of Analog to digital converter module within LPC2148 operation. Selects AD0.3 pin (P0.30). Read the conversion result to display at the terminal program via UART0.

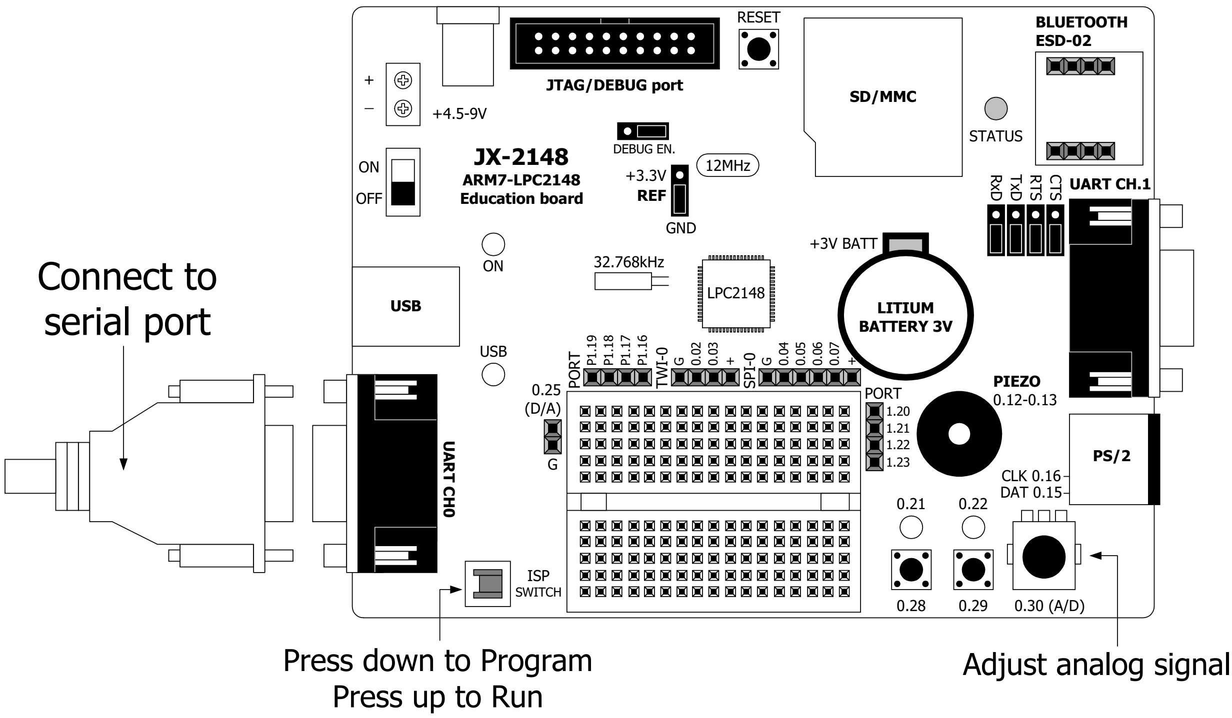
Procedure :

5.1 Build new project, in name **`adc.`**

5.2 Write the program Listing P5-1. Compile to **`adc.hex`** and download to microcontroller.

5.3 Open the Terminal program such as Hyper terminal or RS-232 Terminal to test the operation. Set baudrate to 9,600 bit per second.

5.4 Connect serial port cable from computer RS-232 port to UARTCH.0 connector.



5.5 Run the program.Watch the result on the terminal program operation.

The terminal window shows message below :

`Analog(AD0.3): xxx`

By **xxx** *is the conversion digital data from A/D converter module.*

```
//————————————————————————————————————————————————————————————————————————————————————————//
// Program     : Example Analog to Digital converter
// Description : Display A/D convert value on terminal program(used UART0
//               : communication)
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename    : adc.c
// C compiler  : Keil CARM Compiler
//————————————————————————————————————————————————————————————————————————————————————————//
#include "lpc214x.h"   // Header file for Phillips LPC2148 controller
#include "stdio.h"     // Library for sprintf function
#include "uart.h"      // Library for UART

//—————————————— Function for  Initial system clock ————————————————//
void init()
{
  PLL0CFG=0x24;         // MSEL = 4,PSEL = 2
  PLL0FEED=0xAA;        // Feed process
  PLL0FEED=0x55;

  PLL0CON=0x1;
  PLL0FEED=0xAA;        // Feed process
  PLL0FEED=0x55;

  while(!(PLL0STAT & 0x400)) ;   // Wait until PLL Locked

  PLL0CON=0x3;          // Connect the PLL as the clock source
  PLL0FEED=0xAA;        // Feed process
  PLL0FEED=0x55;

  MAMCR=0x2;            // Enabling MAM and setting number of clocks used for
                        // Flash memory fetch (4 cclks in this case)
  MAMTIM=0x4;
  VPBDIV=0x02;          // PCLK at 30 MHz
}

//—————————————————— Function delay ————————————————————————//
void delay_ms(long ms)       // delay 1 ms per count @ CCLK 60 MHz
{
  long i,j;
  for (i = 0; i < ms; i++ )
  for (j = 0; j < 6659; j++ );
}

//—————————————————— Main Program ————————————————————————//
void main()
{
  int val=0;
  char s[30];                // Buffer for keep string from integer convert
  init();                    // Initialize the system
  SCS = 0x03;                // select the "fast" version of the I/O ports
  uart0_init(9600);          // Initial UART0
```

Listing P5-1 : **`adc.c`** file of **`adc`** project for LPC2148 A/D converter experiment (continue)

```
  PINSEL1  |= 0x10000000;                    // Enable AD0.3 for used
  AD0CR    = 0x00210608;                     // Setup A/D: 10-bit AIN0 @ 3MHz
   while (1)
  {
     AD0CR   |= 0x01000000;                  // Start A/D Conversion
     while ((AD0DR3 & 0x80000000) == 0); // Wait for the conversion to complete
     val = ((AD0DR3 >> 6) & 0x03FF);         // Extract the A/D result
     sprintf(s,"Analog(AD0.3): %d  \r",val);
                                       // Convert string to display analog value
     uart0_puts(s);                    // Display to Terminal program
     delay_ms(200);                    // Delay for display
  }
}
```

Listing P5-1 : `adc.c` file of `adc` project for LPC2148 A/D converter experiment (final)

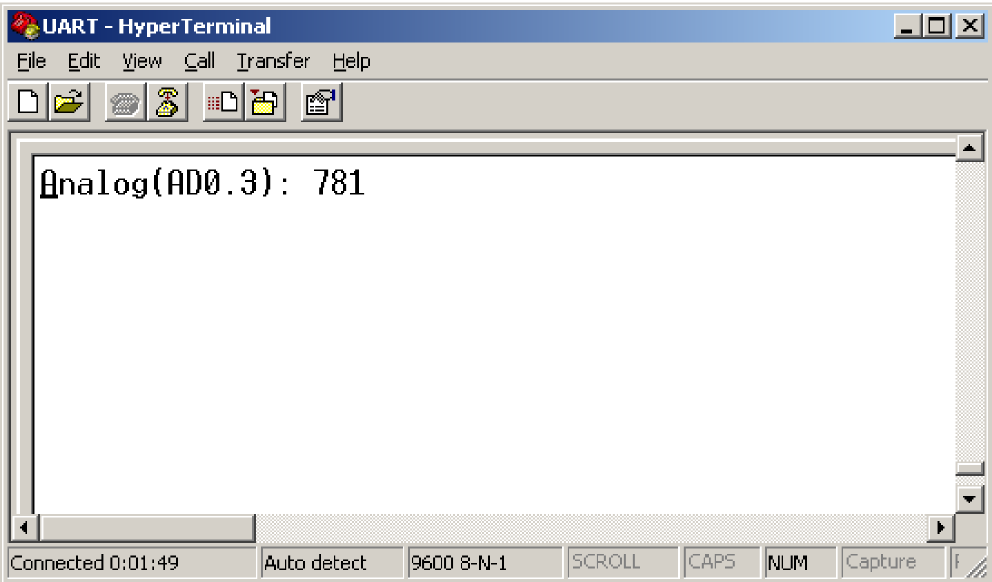


Figure P5-1 Shows the result of A/D converter experiment at the Hyper terminal window.

Experiment - 6 : Real-time Clock in LPC2148

Experiment - 6.1 : RTC interrupt

This experiment is demonstration about set the interrupt time for Real-time clock module within LPC2148. The interrupt will happen 2 cases. The first is interrupt every 1 second. LED at P0.22 will blink following the interrupt event. The second is interrupt at every second unit as 3 such as 12:05:03,12:06:03 or 02:48:03 etc.

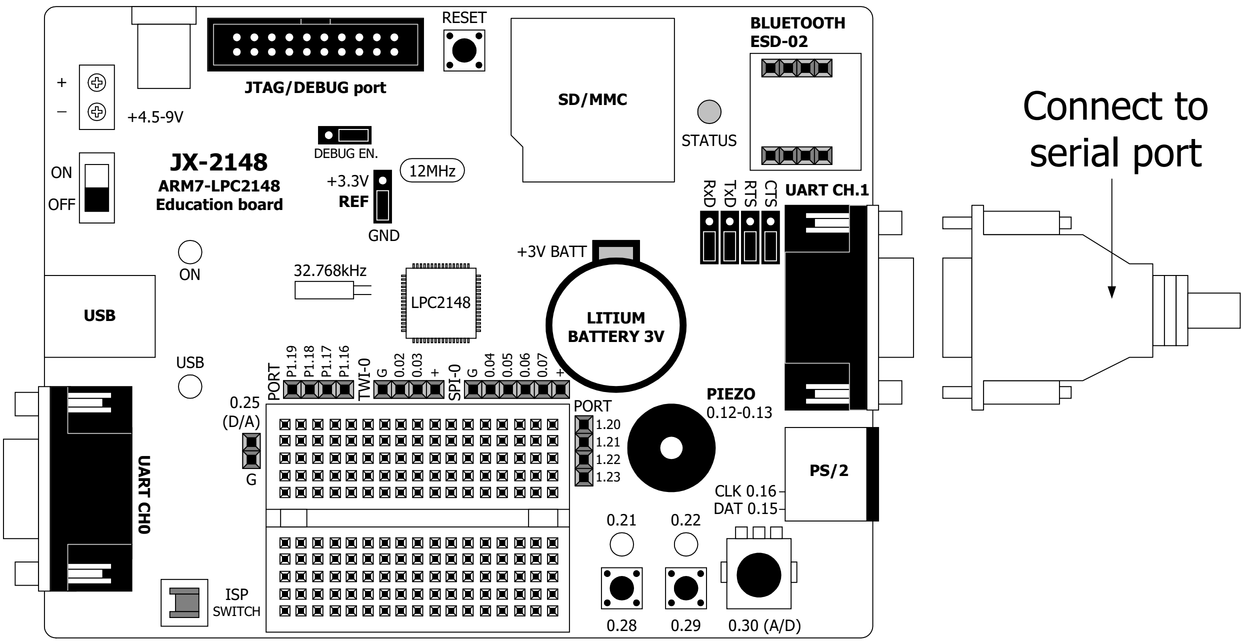
Procedure :

6.1.1 Build new project, in name **`rtc_int.`**

6.1.2 Write the program Listing P6-1. Compile to **`rtc_int.hex`** and download to microcontroller.

6.1.3 Open the Terminal program such as Hyper terminal or RS-232 Terminal to test the operation. Set baudrate to 9,600 bit per second.

6.1.4 Connect serial port cable from computer RS-232 port to UARTCH.1 connector.



```
//————————————————————————————————————————————————————————————————————————————————//
// Program     : Example for Real time clock
// Description : -
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename    : rtc_int.c
// C compiler  : Keil CARM Compiler
//————————————————————————————————————————————————————————————————————————————————//
#include "lpc214x.h"   // Header file for Phillips LPC2148 controller
#include "sound.h"     // Header file for Phillips LPC2148 controller
#include "stdio.h"     // Library for use puts function(For UART1)
#include "uart.h"      // Library for use module UART0,UART1
#define LED1_ON FIO0CLR = 0x00400000     // Red led 0.22 on
#define LED1_OFF FIO0SET = 0x00400000    // Red led 0.22 off
char alarm = 0;                          // Variable for status sound alarm
//—————————————— Function for  Initial system clock ————————————————//
void init()
{
  PLL0CFG=0x24;                          // MSEL = 4,PSEL = 2
  PLL0FEED=0xAA;                         // Feed process
  PLL0FEED=0x55;

  PLL0CON=0x1;
  PLL0FEED=0xAA;                         // Feed process
  PLL0FEED=0x55;
  while(!(PLL0STAT & 0x400)) ;           // Wait until PLL Locked
  PLL0CON=0x3;                           // Connect the PLL as the clock source
  PLL0FEED=0xAA;                         // Feed process
  PLL0FEED=0x55;

  MAMCR=0x2;      // Enabling MAM and setting number of clocks used for Flash
                  // memory fetch (4 cclks in this case)
  MAMTIM=0x4;

  VPBDIV=0x02;                           // PCLK at 30 MHz
}
//————————————————— Function delay ———————————————————————//
void delay_ms(long ms)                   // delay 1 ms per count @ CCLK 60 MHz
{
  long i,j;
  for (i = 0; i < ms; i++ )
  for (j = 0; j < 6659; j++ );
}
//—————————————— Interrupt service routine for UART1 ————————————————//
void isr_rtc(void) __irq
{
    if(ILR & 0x01)                       // Check Interrupt block generate
    {
        LED1_ON;                         // LED on
        delay_ms(100);                   // Delay for Blink LED
        LED1_OFF;                        // LED off
        ILR = 0x01;                      // Clear interrupt flag
    }
    if(ILR & 0x02)
    {
        alarm = 1;                 // Set flag alarm for generate sound beep
        ILR = 0x02;                // Clear interrupt flag
    }
```

Listing P6-1 : **`rtc_int.c`** file of **`rtc_int`** project for LPC2148 RTC experiment (continuous)

```
    VICVectAddr = 0;                // Acknowledge Interrupt
}

//—————————————— Main Program ———————————————————//
void main()
{
  char i=0;
  init();                     // Initialize the system
  SCS = 0x03;                 // select the "fast" version of the I/O ports
  FIO0DIR |= 0x00400000;      // Config. pin P0.22 as output
  uart1_init(9600);
  PREINT = 0x00000392;        // Set RTC Prescaler for PCLK 30 MHz
  PREFRAC = 0x00004380;
  CIIR = 0x00000001;          // Enable seconds counter interrupt
  ALSEC = 0x00000003;               // Set alarm register for 3 seconds
                              // (match when xx:xx:03)
  AMR = 0x000000FE;           // Enable seconds alarm
  CCR = 0x00000001;           // Start RTC
  VICVectAddr13 = (unsigned)isr_rtc;
  VICVectCntl13 = 0x20 | 13;
  VICIntEnable |= (1<<13);    // Enable RTC Interrupt
  while (1)                   // Infinite loop
  {
     if(alarm==1)             // Check seconds alarm match
     {
        beep();               // Sound beep 1 time
        i++;                  // Increment counter for sound
        if(i>10)              // Over 10 time?
        {
           i=0;               // Clear counter
           alarm = 0;         // Clear alarm flag
        }
     }
     printf("TIME: %d:%d:%d   \r  ",HOUR,MIN,SEC);
                              // Display time format hh:mm:ss
     delay_ms(100);           // Delay for display
  }
}
```

More information about important function in this program

The suitable value for **PREINT** and **PREFRAC** register in LPC2148's RTC module can calculate from

PREINT = int(PCLK / 32768) - 1 ....**(Eq. P6-1)**

PREFRAC = PCLK - ((PREINT + 1) x 32768).....................**(Eq. P6-2)**

From Eq. P6-1

PREINT value = int (30000000 / 32768) - 1 = 914 or 0x392

From Eq. P6-2

PREFRAC value = 30000000 - ((914 + 1) x 32768) = 17280 or 0x4380

Listing P6-1 : **`rtc_int.c`** file of **`rtc_int`** project for LPC2148 RTC experiment (final)

6.1.5 Run program. Watch the result on the terminal program and LED at P0.22 operation.

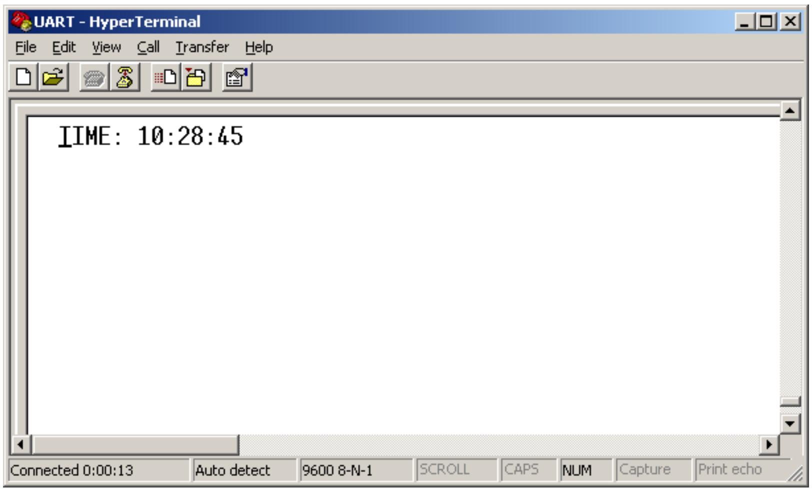
The terminal window shows message below :

TIME: hh:mm:ss

by hh is time in hour unit., mm is time in minute unit and ss is second.

LED at P0.22 will blink in rate 1 second. This operation is defined from the interrupt first case.

In every second at number 3, piezo will drive beep signal 10 times continuous. This operation is defined from the interrupt second case.



However the operation of this experiment RTC module cannot operate continuous without supply voltage*. Because the program set to RTC clock use same souce clock of CPU (****CLKSRC**** bit in CCR register is ‘0’). RTC module will operate when apply supply voltage to CPU only.*

In nthe next experiment will shows the solution of this limitation.

Experiment - 6.2 : Setting time to RTC.

This experiment will demonstrate the setting time value to RTC module. It is writing the value to 3 time registers includes HOUR, MIN and SEC. Input all values with keyboard pass Terminal program via UART1 module.

Procedure :

6.2.1 Build new project, in name **rtc_setup.**

6.2.2 Write the program Listing P6-2. Compile to **rtc_setup.hex** and download to microcontroller. Close LPC2000 Flash Utility program.

```
//------------------------------------------------------------------------//
// Program     : Example for Real time clock
// Description : Example for display via Terminal program time format hh:mm:ss
//               : and user can setup new time by press key '*' on keyboard
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename    : rtc_setup.c
// C compiler  : Keil CARM Compiler
//------------------------------------------------------------------------//
#include "lpc214x.h"   // Header file for Phillips LPC2148 controller
#include "stdio.h"     // Library for use puts function(For UART1)
#include "uart.h"      // Library for use module UART0,UART1
#include "ctype.h"     // Library for isdigit function
#include "stdlib.h"    // Library for atoi function
char key = 0;          // Variable for status sound alarm
//-------------- Function for  Initial system clock ----------------//
void init()
{
  PLL0CFG=0x24;          // MSEL = 4,PSEL = 2
  PLL0FEED=0xAA;         // Feed process
  PLL0FEED=0x55;

  PLL0CON=0x1;
  PLL0FEED=0xAA;         // Feed process
  PLL0FEED=0x55;

  while(!(PLL0STAT & 0x400)) ;      // Wait until PLL Locked

  PLL0CON=0x3;                      // Connect the PLL as the clock source
  PLL0FEED=0xAA;                    // Feed process
  PLL0FEED=0x55;
  MAMCR=0x2;      // Enabling MAM and setting number of clocks used for Flash
                  // memory fetch (4 cclks in this case)
  MAMTIM=0x4;
  VPBDIV=0x02;                      // PCLK at 30 MHz
}
```

Listing P6-2 : **rtc_setup.c** file of **rtc_setup** project for LPC2148 RTC experiment (continuous)

```
//—————————————————— Function delay —————————————————————————//
void delay_ms(long ms)                   // delay 1 ms per count @ CCLK 60 MHz
{
  long i,j;
  for (i = 0; i < ms; i++ )
  for (j = 0; j < 6659; j++ );
}

//——————————————— Interrupt service routine for UART1 ————————————————//
void isr_uart1(void) __irq
{
  char msg;
  if(((msg = U1IIR) & 0x01) == 0)        // Check status flag communication
  {
     switch (msg & 0x0E)                 // Filter message
     {
        case 0x04:  while(!(U1LSR & 0x20));    // Receive Data Available
                    key = U1RBR;
                    break;
        case 0x02:  break;                     // THRE Interrupt
        default:    break;                     // Other
     }
  }
  VICVectAddr = 0;                             // Acknowledge Interrupt
}

//——————————————— Function for setup date/time for Real time clock ——————————//
void rtc_uart1_setup(char *s)
{
  unsigned char tm;                 // Buffer for keep date/time setup value
  char i=0;                         // Variable for loop counter
  for(i=0;i<2;i++)                  // Loop for keep value 2 byte
  {
     while(!isdigit(key));          // Wait for key '0'-'9' only
     if(i==0)
     tm = 10*atoi(&key);            // Keep '1 value
     if(i==1)
     tm = tm+atoi(&key);            // Keep '2 value
     putchar(key);                  // Display key on Terminal program
     key = 0;                       // Clear old key for next key
  }
  *s = tm;                          // Load setup new value
}
//—————————————————— Main Program —————————————————————————//
void main()
{
  init();                     // Initialize the system
  SCS = 0x03;                 // select the "fast" version of the I/O ports
  uart1_init(9600);           // Initial UART1 @9600 bps,8 bit data,1 stop bit,
                              // no parity bit
  U1IER = 3;                  // Enable rx/tx interrupt for UART1
  PINSEL0 |= (1<<18);         // Enable RXD1(from UART1) at P0.9
```

Listing P6-2 : **`rtc_setup.c`** file of **`rtc_setup`** project for LPC2148 RTC experiment (continuous)

```
  VICVectAddr0  =  (unsigned)isr_uart1;     // Register Interrupt service routine name
  VICVectCntl0  = 0x20  |  7;               //  UART1  Interrupt
  VICIntEnable  |= 1  <<  7;                //  Enable  UART1  Interrupt
  CCR  =  0x00000011;      // Start RTC used 32.768 kHz crystal for RTCX1/RTCX2 pin
  while  (1)                                //  Infinite  loop
  {
     printf("TIME:  %d:%d:%d    \r   ",HOUR,MIN,SEC);
                                            // Display time format hh:mm:ss
     delay_ms(100);                         // Delay for display
     if(key=='*')                           // Check key for setup time?
     {
       key  =  0;                     //  Clear  old  key  for  next  key
       printf("\nSet  Time:");        //  Display  message  for  setup  time  at  new
line
       rtc_uart1_setup(&HOUR);        // Wait until user insert new value of HOUR
       uart1_putc(':');               // Display ':' at terminal program
       rtc_uart1_setup(&MIN);         // Wait until user insert new value of MIN
       uart1_putc(':');               // Display ':' at terminal program
       rtc_uart1_setup(&SEC);         // Wait until user insert new value of SEC
       printf("\nTIME:  %d:%d:%d      \r   ",HOUR,MIN,SEC);
                                      // Display new time for setup at new line
     }
  }
}
```

Listing P6-2 : **`rtc_setup.c`** file of **`rtc_setup`** project for LPC2148 RTC experiment (final)

6.2.3 Open the Terminal program such as Hyper terminal or RS-232 Terminal to test the operation. Set baudrate to 9,600 bit per second.

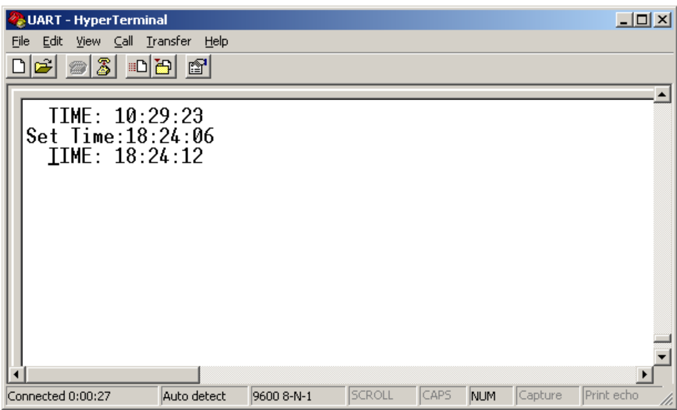
6.2. 4 Connect serial port cable from computer RS-232 port to UARTCH.1 connector.

6.2.5 Run program. Watch the result on the terminal program.

The terminal window shows message below :

`TIME:  hh:mm:ss`

by **`hh`** *is time in hour unit.,* **`mm`** *is time in minute unit and* **`ss`** *is second.*



6.2.6 Press * key for setting time to RTC module. Watch the result on the terminal program.

The new message will appear in the next line :

Set Time:

Wait for data 6 digits. They mean time value in hh,mm and ss. Input the value complete 6 digits. The time value will store in the time registers of RTC module. The terminal will shows the setting time and run continue.

The operation of this experiment RTC module can operate continuous without supply voltage*. Because RTC receive the clock from 32kHz clock oscillator separate CPU clokc.* ***CLKSRC*** *bit in CCR register set to '1'. The time value can store with +3V supply from litium battery on-board.*

Experiment -7 : PS/2 Keyboard interface

This experiment is demonstration about interfacing PS/2 keyboard with LPC2148. The heart of this operaion is **`keyboard.h`** library file. LPC2148 will get the key value and display at the terminal program via UART1.

keyboard.h library for PS/2 keyboard interfacing

The full source program of this library file is shown in Listing P7-1. The description of each function in this library file as :

kb_init

Initialize keyboard clokc pin. In this experiment is EINT0 or P0.16

syntax :

```
void kb_init()
```

kb_getchar

Get the pressed key character that this library support.

syntax :

```
char kb_getchar()
```

return :

Keyboard code or Key value

PS/2 keyboard interface

Keyboard detection of keyboard.h library will detect raising edge interrupt signal at EINT0 or P0.16 pin. This pin will connect to Clock signal of keyboard. If key pressed, status of this pin will down to zero immediately. That is start bit. The figure P7-1 shows the timing diagram of keyboard data bit and clock signal.

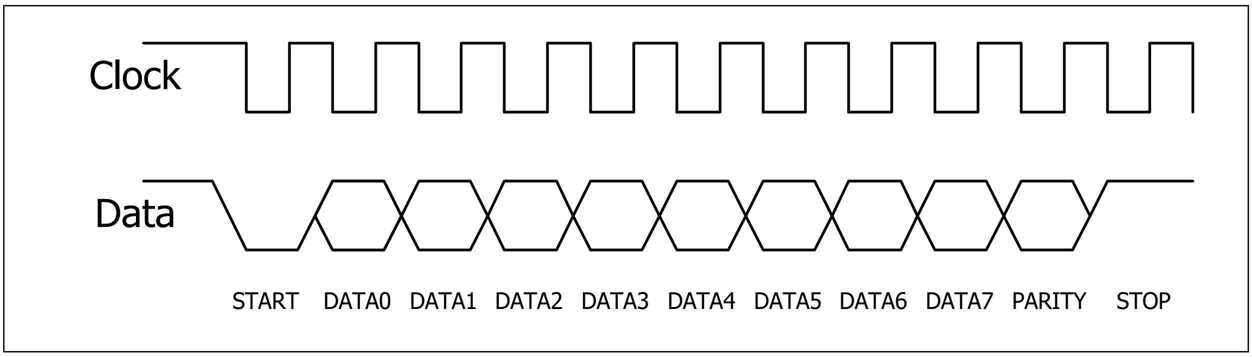


Figure P7-1 PS/2 keyboard signal timing diagram

```
//————————————————————————————————————————————————————//
// Program   : Library for PS/2 Keyboard
// Description  : Detect for input key from PS/2 keyboard
// Frequency : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename : keybord.h
// C compiler   : Keil CARM Compiler
//————————————————————————————————————————————————————//
#include "stdio.h"      // Library for use puts function(For UART1)
#include "ctype.h"       // Library for use toupper function
#define LOWER 0          // Macro for lower case keyboard
#define UPPER 1          // Macro for upper case keyboard
#define CAPS_LOCK_CODE 0x58      // Macro for Caps Lock scan code
unsigned int _code=0;    // Variable for keep scan code
char char_case =LOWER;   // Variable for status character case(lower or upper)

const unsigned char _ascii_key[128] =    // Table for decode
{
    0       // 0x00
    , 0     // 0x01
    , 0     // 0x02
    , 0     // 0x03
    , 0     // 0x04
    , 0     // 0x05
    , 0     // 0x06
    , 0     // 0x07
    , 0     // 0x08
    , 0     // 0x09
    , 0     // 0x0A
    , 0     // 0x0B
    , 0     // 0x0C
    , 11    // 0x0D
    , 0     // 0x0E
    , 0     // 0x0F
    , 0     // 0x10
    , 0     // 0x11
    , 0     // 0x12
    , 0     // 0x13
    , 0     // 0x14
    , 'q'   // 0x15
    , '!'   // 0x16
    , 0     // 0x17
    , 0     // 0x18
    , 0     // 0x19
    , 'z'   // 0x1A
    , 's'   // 0x1B
    , 'a'   // 0x1C
    , 'w'   // 0x1D
    , '@'   // 0x1E
    , 0     // 0x1F
    , 0     // 0x20
    , 'c'   // 0x21
    , 'x'   // 0x22
    , 'd'   // 0x23
    , 'e'   // 0x24
    , '$'   // 0x25
    , '#'   // 0x26
    , 0     // 0x27
    , 0     // 0x28
    , 32    // 0x29
    , 'v'   // 0x2A
    , 'f'   // 0x2B
    , 't'   // 0x2C
    , 'r'   // 0x2D
```

Listing P7-1 Source program of keyboard.h library file (continue)

```
, '%'    // 0x2E
, 0      // 0x2F
, 0      // 0x30
, 'n'    // 0x31
, 'b'    // 0x32
, 'h'    // 0x33
, 'g'    // 0x34
, 'y'    // 0x35
, '^'    // 0x36
, 0      // 0x37
, 0      // 0x38
, 0      // 0x39
, 'm'    // 0x3A
, 'j'    // 0x3B
, 'u'    // 0x3C
, '&'    // 0x3D
, '*'    // 0x3E
, 0      // 0x3F
, 0      // 0x40
, 44     // 0x41
, 'k'    // 0x42
, 'i'    // 0x43
, 'o'    // 0x44
, ')'    // 0x45
, '('    // 0x46
, 0      // 0x47
, 0      // 0x48
, '.'    // 0x49
, '/'    // 0x4A
, 'l'    // 0x4B
, ';'    // 0x4C
, 'p'    // 0x4D
, '-'    // 0x4E
, 0      // 0x4F
, 0      // 0x50
, 0      // 0x51
, 39     // 0x52
, 0      // 0x53
, 0      // 0x54
, '='    // 0x55
, 0      // 0x56
, 0      // 0x57
, 0      // 0x58
, 0      // 0x59
, 13     // 0x5A
, ']'    // 0x5B
, 0      // 0x5C
, 92     // 0x5D
, 0      // 0x5E
, 0      // 0x5F
, 0      // 0x60
, 0      // 0x61
, 0      // 0x62
, 0      // 0x63
, 0      // 0x64
, 0      // 0x65
, 8      // 0x66
, 0      // 0x67
, 0      // 0x68
, '1'    // 0x69
, 0      // 0x6A
, '4'    // 0x6B
, '7'    // 0x6C
, 0      // 0x6D
```

Listing P7-1 Source program of keyboard.h library file (continue)

```
    , 0     // 0x6E
    , 0     // 0x6F
    , '0'   // 0x70
    , '.'   // 0x71
    , '2'   // 0x72
    , '5'   // 0x73
    , '6'   // 0x74
    , '8'   // 0x75
    , 27    // 0x76
    , 0     // 0x77
    , 0     // 0x78
    , '+'   // 0x79
    , '3'   // 0x7A
   , '-'    // 0x7B
    , '*'   // 0x7C
    , '9'   // 0x7D
    , 0     // 0x7E
    , 0     // 0x7F
};

//—————————————————— Function Check input P0 ——————————————————//
char _inp0(char _bit)
{
   unsigned long c;
   c = 1<<_bit;                      // Calculate digit to configuration for input port
   return((FIO0PIN & c)>>_bit);     // Read and return data bit
}

//—————————————————— Function get character from keyboard ————————————//
char kb_getchar()
{
    static unsigned char    temp=0
      ,bk=0;
    unsigned char result;
    while(_code == 0);                // Wait for key code support only
    temp = _code;                     // Keep code detect
    if(bk==2)                         // Not keep code for 0xF0,0xE0,0xE1 sequent
    {
        bk =0;
    }
    if(bk==1)                         // Not keep code for 0xF0,0xE0,0xE1 sequent
    {
        bk =2;
    }
    if(_code==0xF0 || _code==0xE0)    // Check break code and none key support
    {
        bk=1;                         // Start break code sequent
    }
    _code = 0;  // Clear old code
    if((temp !=0xF0 || temp !=0xE0 || temp !=0xE1) && bk==0)  // Check code support?
    {
        if(temp == CAPS_LOCK_CODE)    // Check Caps Lock mode key push?
        {
            char_case ^=1;            // Toggle status Caps Lock mode
        }
        if(char_case==UPPER)          // Check Upper case for character 'A'-'Z'
            result = toupper(_ascii_key[temp]); // Result 'A'-'Z'
        else
            result = _ascii_key[temp];  // Result 'a'-'z'
        return(result);               // Return result
   }
   else
   return(0);
}
```

Listing P7-1 Source program of keyboard.h library file (continue)

```
//―――――――――――――― Interrupt service routine for EINT0 ――――――――――――――//
void isr_int0(void) __irq
{
   unsigned char i;              // Define for counter loop
   if(_inp0(16)==0)              // Check start bit true?
   {
      while(_inp0(16)==0);       // wait for "1" after start bit
      for(i=0;i<10;i++)          // For loop count 10 time(for receive data 8 bit)
      {
         while(_inp0(16)==1);    // wait for "0" after data bit
         _code = _code>>1;       // Shift data bit to right 1 time
         if(_inp0(15))
      _code = _code | 0x8000;    // Config data bit = "1"
         while(_inp0(16)==0);    // wait for "1" after data bit
      }
      while(_inp0(16)==0);       // wait for "1" after stop bit
      _code = _code>>6;
      _code &= 0x00FF;
   }
   EXTINT |= 0x1;                // Clear interrupt flag EINT0
   VICVectAddr = 0;              // Acknowledge Interrupt
 }

//――――――――――――――――― Function initial PS/2 keyboard ――――――――――――――――//
void kb_init()
{
     FIO0DIR &= ~(1<<15);        // Config. output P0.15 DATA pin for keyboard
     FIO0DIR &= ~(1<<16);        // Config. output P0.16 DATA pin for keyboard
     EXTMODE |= 0x1;             // EINT0 Edge sensitive detection(Falling edge)
     PINSEL1 |= 0x01;            // Enable EINT0 at P0.16
     VICVectAddr0 = (unsigned)isr_int0; // Register Interrupt service routine name
     VICVectCntl0 |= (0x20 | 14);       // EINT0 Interrupt
     VICIntEnable |= 1 << 14;           // Enable EINT0 Interrupt
}
```

Listing P7-1 Source program of keyboard.h library file (final)

To start using this library, must execute `kb_init` function in the top of program for setting interrupt operation of EINT0. The function `isr_int0` is interrupt service. It is set the first priority (slot 0) After key pressed, program will jump to this service routien `isr_int0`.

Operation of isr_int0 function is filter 8-bit data to store in `_code` variable and convert to ASCII code by look-up table refer Table P7-1. In one key pressed, the data will appear 2 values. One is pressed key data, another is released key data. For examplr, S key is pressed. Data will be 0x1B. After release, the data will change to 0xF0.

Key detection is operation of `kb_getchar` function. If detect key pressed, this function will return key value (in condition, that key must support by **`keyboard.h`** library). Internal operation of `kb_getchar` function will look over both pressed and releasaed key data (0xE0 and 0xF0). After that bring the key value to look up table for getting ASCII data and retun final data.

Key	Pressed key data	Released key data
A	1C	F0, 1C
B	32	F0, 32
C	21	F0, 21
D	23	F0, 23
E	24	F0, 24
F	2B	F0, 2B
G	34	F0, 34
H	33	F0, 33
I	43	F0, 43
J	3B	F0, 3B
K	42	F0, 42
L	4B	F0, 4B
M	3A	F0, 3A
N	31	F0, 31
O	44	F0, 44
P	4D	F0, 4D
Q	15	F0, 15
R	2D	F0, 2D
S	1B	F0, 1B
T	2C	F0, 2C
U	3C	F0, 3C
V	2A	F0, 2A
W	1D	F0, 1D
X	22	F0, 22
Y	35	F0, 35
Z	1A	F0, 1A
0	45	F0, 45
1	16	F0, 16
2	1E	F0, 1E
3	26	F0, 26
4	25	F0, 25
5	2E	F0, 2E
6	36	F0, 36
7	3D	F0, 3D
8	3E	F0, 3E
9	46	F0, 46

Key	Pressed key data	Released key data
'	0E	F0, 1C
-	4E	F0, 1C
=	55	F0, 1C
\	5D	F0, 1C
BKSP	66	F0, 1C
Space	29	F0, 1C
Tab	0D	F0, 1C
Caps	58	F0, 1C
L Shift	12	F0, 1C
L Ctrl	14	F0, 1C
L GUI	E0, 1F	F0, 1C
L Alt	11	F0, 1C
R Shift	59	F0, 1C
R Ctrl	E0, 14	F0, 1C
R GUI	E0, 27	F0, 1C
R Alt	E0, 11	F0, 1C
Apps	E0, 2F	F0, 1C
Enter	5A	F0, 1C
ESC	76	F0, 1C
F1	05	F0, 1C
F2	06	F0, 1C
F3	04	F0, 1C
F4	0C	F0, 1C
F5	03	F0, 1C
F6	0B	F0, 1C
F7	83	F0, 1C
F8	0A	F0, 1C
F9	01	F0, 1C
F10	09	F0, 1C
F11	78	F0, 1C
F12	07	F0, 1C
PrtScn	E0,12, E0, 7C	F0, 1C
Scroll	7E	F0, 1C
Pause	E1,14,77,E1, F0,14,F0,77	F0, 1C
[	54	F0, 1C
Insert	E0, 70	F0, 1C

Key	Pressed key data	Released key data
Home	E0, 6C	F0, 1C
Page Up	E0, 7D	F0, 1C
Delete	E0, 71	F0, 1C
End	E0, 69	F0, 1C
Page Dwn	E0, 7A	F0, 1C
↑	E0, 75	F0, 1C
←	E0, 6B	F0, 1C
↓	E0, 72	F0, 1C
→	E0, 74	F0, 1C
NUM	77	F0, 1C
KP /	E0, 4A	F0, 1C
KP *	7C	F0, 1C
KP -	7B	F0, 1C
KP +	79	F0, 1C
KP EN	E0, 5A	F0, 1C
KP .	71	F0, 1C
KP 0	70	F0, 1C
KP 1	69	F0, 1C
KP 2	72	F0, 1C
KP 3	7A	F0, 1C
KP 4	6B	F0, 1C
KP 5	73	F0, 1C
KP 6	74	F0, 1C
KP 7	6C	F0, 1C
KP 8	75	F0, 1C
KP 9	7D	F0, 1C
]	5B	F0, 1C
;	4C	F0, 1C
'	52	F0, 1C
,	41	F0, 1C
.	49	F0, 1C
/	4A	F0, 1C

Table P7-1 Keyboard data of PS/2 keyboard

Addition about Caps Lock support, swap character format about lower case and upper case letter. The `kb_getchar` function will return key a as ‘a’ (0x61). If Caps Lock key pressed and press key a again. this function will return ASCII of ‘A’ (0x41) instead but not turn on LED CAps Lock at keyboard.

Procedure :

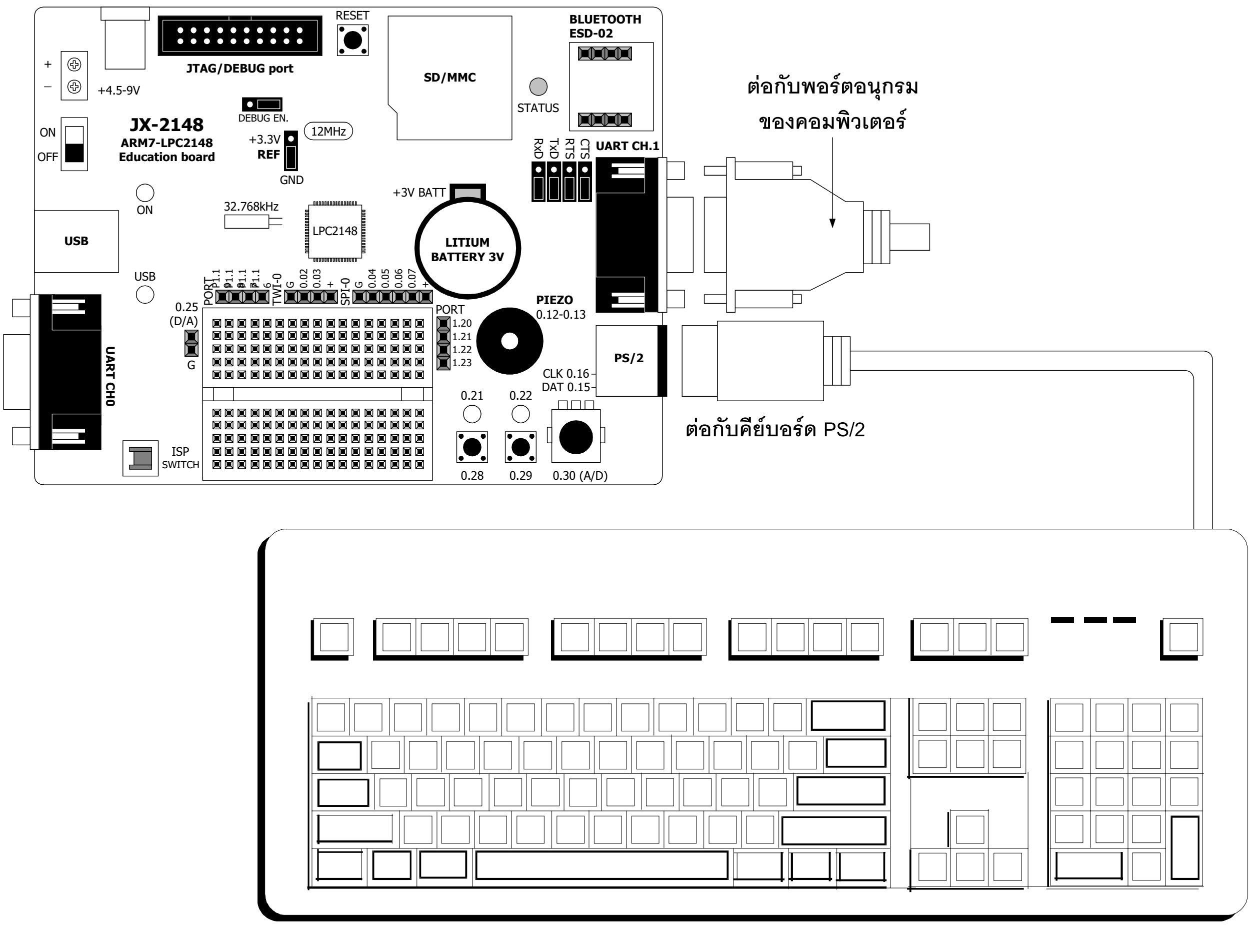
7.1 Build new project, in name **keyboard_ps2.**

7.2 Write the program Listing P7-2. Compile to **keyboard_ps2.hex** and download to microcontroller. Close LPC2000 Flash Utility program.

7.3 Open the Terminal program such as Hyper terminal or RS-232 Terminal to test the operation. Set baudrate to 9,600 bit per second.

7. 4 Connect serial port cable from computer RS-232 port to UARTCH.1 connector.

7.5 Connect PS/2 keyboard at PS/2 connecotr on JX-2148 board.



```
//————————————————————————————————————————————————————————————————————————//
// Program     : Example for decode form PS/2 keyboard
// Description: Get character from PS/2 keyboard and display at terminal program
//             : used UART1 communication
// Frequency   : Crystal 12 MHz at PLL 5x(CCLK = 60 MHz),PCLK = 30 MHz
// Filename    : keyboard_ps2.c
// C compiler : Keil CARM Compiler
//————————————————————————————————————————————————————————————————————————//
#include "lpc214x.h"      // Header file for Phillips LPC2148 controller
#include "sound.h"        // Header file for Phillips LPC2148 controller
#include "uart.h"         // Library for use module UART0,UART1
#include "stdio.h"        // Library for use puts function(For UART1)
#include "keyboard.h"     // Library for use PS/2 keyboard

//—————————————— Function for  Initial system clock ————————————————//
void init()
{
   PLL0CFG=0x24;          // MSEL = 4,PSEL = 2
   PLL0FEED=0xAA;         // Feed process
   PLL0FEED=0x55;

   PLL0CON=0x1;
   PLL0FEED=0xAA;         // Feed process
   PLL0FEED=0x55;

   while(!(PLL0STAT & 0x400)) ;     // Wait until PLL Locked

   PLL0CON=0x3;                     // Connect the PLL as the clock source
   PLL0FEED=0xAA;                   // Feed process
   PLL0FEED=0x55;

   MAMCR=0x2;        // Enabling MAM and setting number of clocks used for
                     // Flash memory fetch (4 cclks in this case)
   MAMTIM=0x4;

   VPBDIV=0x02;      // PCLK at 30 MHz
}

//————————————————— Main Program ————————————————————————//
void main()
{
   init();                // Initialize the system
   SCS = 0x03;            // select the "fast" version of the I/O ports
   kb_init();             // Initial for used PS/2 keyboard
   uart1_init(9600);      // Initial UART1 @9600 bps,8 bit data,1 stop bit,
                          // no parity bit
   while(1)               // Infinite loop
   {
     printf("%c",kb_getchar());     // Get character from keyboard
                                    // and display at terminal program
   }
}
```

Listing P7-1 : **`keyboard_ps2.c`** file of **`keyboard_ps2`** project for LPC2148 RTC experiment

7.6 Run program. Try to type any key on keyboard and watch the result on the terminal program. After that press Caps Lock key and re-type keyboard again. See the different result.