Department of Electronics and Communication Engineering
KPR Institute of Engineering and Technology

KPRIET
Learn Beyond

LABORATORY MANUAL

U21ECG06

EMBEDDED SYSTEMS AND IoT LABORATORY

VISION AND MISSION OF THE INSTITUTION

Vision

To become a premier institute of academic excellence by imparting technical, intellectual and professional skills to students for meeting the diverse need of the industry, society, the nation and the world at large

Mission

- ❖ Commitment to offer value-based education and enhancement of practical skills
- ❖ Continuous assessment of teaching and learning process through scholarly activities
- ❖ Enriching research and innovation activities in collaboration with industry and institute of repute
- ❖ Ensuring the academic process to uphold culture, ethics and social responsibility

VISION AND MISSION OF THE DEPARTMENT

Vision

To be a department of repute for learning and research with state-of-the-art facilities to enable the students to succeed in globally competitive environment.

Mission

The Department of Electronics and Communication Engineering is committed

- ❖ To impart knowledge and skill based education with competent faculty striving for academic excellence
- ❖ To instil research centres in the field, that industry needs, by collaborating with organizations of repute
- ❖ To provide ethical and value based education by promoting activities addressing the societal needs and facilitates lifelong learning

Program Educational Objectives (PEOs)

- ❖ Graduates will possess an adequate knowledge and have successful technical career in Electronics and Communication Engineering or related fields
- ❖ Graduates will possess leadership qualities and demonstrate professional and ethical values.
- ❖ Graduates will continue their life-long professional development through higher education or entrepreneurship

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering Fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

Program Specific Outcomes (PSOs)

A graduate of Electronics and Communication Engineering will be able to

1. **PSO 1**: Graduates will be equipped with microcontroller based system design skills to work as design and verification engineers in the area of Embedded Systems Design.

2. **PSO 2**: (n) Graduates will be able to apply engineering knowledge for the design and implementation of projects pertaining to VLSI Design, Image processing and Communication.

RUBRICS FOR ASSESSMENT

Criteria		Excellent (4 Marks)	Good (3 Marks)	Adequate (2 Marks)	Inadequate (1 Mark)
A. Preparation & Observation	**Criterion #1** Ability to setup and conduct experiments	Able to develop contingency or alternative plans and anticipate problems during experiment.	Able to develop contingency or alternative plans.	Able to use theoretical framework, measurement techniques, testing apparatus or model.	Unable to identify theoretical framework, measurement techniques, testing apparatus or model.
	Criterion #2 Ability to take measurements / readings and present data	Able to formulate, controls and evaluate alternatives of the experiment. Able to evaluate data and relate to engineering phenomena for decision-making.	Able to evaluate data and relate to engineering phenomena for decision-making.	Able to apply constraint and assumption into the experimental design. Able to conduct experiment correctly and collect data.	Unable to discuss experimental processes and protocols
B. Results & Interpretation	**Criterion #3** Ability to analyze the data theoretically and logically to conclude experimental results	Able to combine /organize more than one set of data, interpret data and make meaningful conclusion.	Able to evaluate or compare data and make meaningful conclusion	Able to select and use and apply appropriate techniques or methods to analyse the data.	Unable to select and describe the techniques or methods of analyzing the data.
	Criterion #4 Ability to interpret and discuss any discrepancies between theoretical and experimental results	Able to verify and/or validate several sets of data and relates to engineering phenomena for decision making.	Able to verify and/or validate data and relate to engineering phenomena for decision making.	Able to identify and verify how results relate/differ from theory or previous results	Unable to identify how results relate/differ from theory or previous results.

C. Viva Voce	**Criterion #5** Demonstrate the ability to respond effectively to questions	Able to listen carefully and respond to questions appropriately; is able to explain and interpret results to the teacher	Able to listen carefully and respond to questions appropriately	Misunderstand the questions and does not respond appropriately to the teacher, or has some trouble in answering questions	Unable to listen carefully to questions and does not provide an appropriate answer, or is unable to answer questions

LIST OF EXPERIMENTS

S.NO	NAME OF THE EXPERIMENTS	PAGE NO
01	Interfacing LED to toggle at equal time delay using Arduino.	
02	Interfacing of LED circuit for various intensity levels with different duty cycles using LPC2148 and MSP430	
03	Display a character in a 16x2 LCD using LPC2148.	
04	Stepper motor to rotate in clockwise and anti-clockwise direction using LPC2148.	
05	Real Time Clock using LPC2148	
06	PIR sensor based object detection using LPC2148	
07	IoT based Gas monitoring system using MQ5 sensor	

Prepared by
Mr.T.Venkatesh,AP(Sr.G)
Ms.M.Supriya,AP
Mr.N.Sathish kumar ,AP(Sr.G)

Approved by
HoD/ ECE

CYCLE OF EXPERIMENTS

CYCLE – I

S.NO	NAME OF THE EXPERIMENTS
	Study of ARM evaluation system, Keil and Proteus
01	Interfacing LED to toggle at equal time delay
02	Interface an LED circuit to vary the intensity by varying the duty cycle to 50%, 75% and 100%
03	Measurement of room temperature using LM35
04	Display a character in a 16*2 LCD

CYCLE – II

S.NO	NAME OF THE EXPERIMENTS
05	Stepper motor to rotate in clockwise and anti-clockwise direction
06	Display a character using serial port
07	Interfacing ADC and DAC
08	Real Time Clock
09	Gas monitoring system
10	Smart power saving system for home automation
11	Mini Project

SAMPLE VIVA QUESTIONS

1. What is the instruction set used by ARM7?
2. How many registers are there in ARM7?
3. ARM7 has an in-built debugging device?
4. What is the capability of ARM7 f instruction for a second?
5. What are the profiles for ARM architecture?
6. List the types of ADC and DAC
7. Define resolution.
8. Summarize the features of Conversion time in ADC.
9. What is the function of Sample-and-hold circuits in analog-to digital converters?
10. Why are internal ADCs preferred over external ADCs?
11. What is Burst conversion mode?
12. What is settling time?
13. How would you define real time clock?
14. List the applications of real time clock.
15. What is the baud rate of serial port ?
16. Compare serial communication and parallel communication.
17. Enumerate the different modes of communication.
18. Through which port the date and time is displayed in RTC?
19. What is a serial port?
20. List the registers used to transfer data in serial port.
21. Why LM35 is used to Measure Temperature?
22. Compare the difference between LM 34 and LM 35 sensors?
23. What is the operating temperature range in LM35?
24. How many pins are available in LM35?
25. What is the main function of analog pin in LPC 2148?
26. What is GSM module SIM900A?
27. What is GSM module for?
28. Which is better GSM or CDMA?

29. What is the difference between SIM900 and SIM900A?
30. What are the types of GSM?
31. Whats is LTE?
32. What are Bluetooth modules?
33. What is the function of Bluetooth module?
34. How many types of Bluetooth module are there?
35. Why we use HC-05 Bluetooth module?
36. What is the longest range of Bluetooth?
37. What is HC-06 Bluetooth module?
38. What is the difference between Bluetooth HC-05 and HC 06?
39. What is MQ Series in gas sensor?
40. What is the advantages of MQ Series gas sensors?
41. What is MQ 6 gas sensor?
42. What is the use of MQ2 sensor?
43. Does a gas sensor measure co2?
44. What are the different types of gas sensors?
45. What is the difference between smart home and home automation?
46. What is the function of ESP8266?
47. List some of the devices that can be automated in home?
48. What Makes a smart home smart?
49. What are the disadvantages of smart homes?
50. Can a smart home be hacked?

EX.NO : 01	INTERFACING LED TO TOGGLE AT EQUAL TIME DELAY USING ARDUINO
DATE :	

AIM :

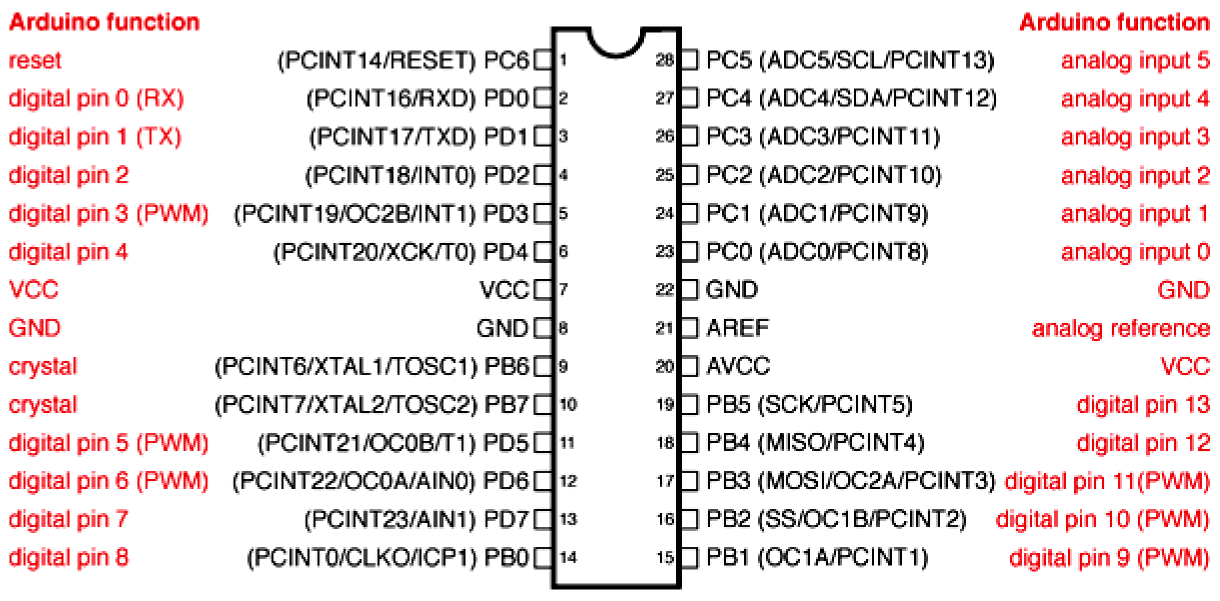
To toggle LED at equal time delay by interfacing it with Arduino.

SOFTWARES/HARDWARES REQUIRED:

S.No	Description	Specification	Quantity
1	Breadboard	-	1
2	Arduino Uno	-	1
3	LED	-	1
4	Resistor	330Ω	1
5	Connecting wires	-	As Required

THEORY:

Arduino is an open-source board that has a Microchip ATmega328P microcontroller on it. Along with ATmega328P MCU IC, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. This microcontroller has a set of Digital & Analog input and output pins. The operating voltage of the board is 5V. It has 14 digital I/O pins & 6 Analog input pins. The clock frequency of the microcontroller is 16 MHz.



Arduino function
reset (PCINT14/RESET) PC6 1
digital pin 0 (RX) (PCINT16/RXD) PD0 2
digital pin 1 (TX) (PCINT17/TXD) PD1 3
digital pin 2 (PCINT18/INT0) PD2 4
digital pin 3 (PWM) (PCINT19/OC2B/INT1) PD3 5
digital pin 4 (PCINT20/XCK/T0) PD4 6
VCC VCC 7
GND GND 8
crystal (PCINT6/XTAL1/TOSC1) PB6 9
crystal (PCINT7/XTAL2/TOSC2) PB7 10
digital pin 5 (PWM) (PCINT21/OC0B/T1) PD5 11
digital pin 6 (PWM) (PCINT22/OC0A/AIN0) PD6 12
digital pin 7 (PCINT23/AIN1) PD7 13
digital pin 8 (PCINT0/CLKO/ICP1) PB0 14
Arduino function
28 PC5 (ADC5/SCL/PCINT13) analog input 5
27 PC4 (ADC4/SDA/PCINT12) analog input 4
26 PC3 (ADC3/PCINT11) analog input 3
25 PC2 (ADC2/PCINT10) analog input 2
24 PC1 (ADC1/PCINT9) analog input 1
23 PC0 (ADC0/PCINT8) analog input 0
22 GND GND
21 AREF analog reference
20 AVCC VCC
19 PB5 (SCK/PCINT5) digital pin 13
18 PB4 (MISO/PCINT4) digital pin 12
17 PB3 (MOSI/OC2A/PCINT3) digital pin 11(PWM)
16 PB2 (SS/OC1B/PCINT2) digital pin 10 (PWM)
15 PB1 (OC1A/PCINT1) digital pin 9 (PWM)

Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Arduino Uno Pinout Configuration

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 – 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

<u>INTERFACING OF LED WITH ARDUINO:</u>

Each digital pin of the Arduino is outputting 5v DC at 40mA, and most LEDs require a voltage of 2v and a current of 35mA. Anything higher than 2 volts may damage the LED. Therefore, a resistor is used that will reduce the 5v to 2v and the current from 40mA to 35mA.

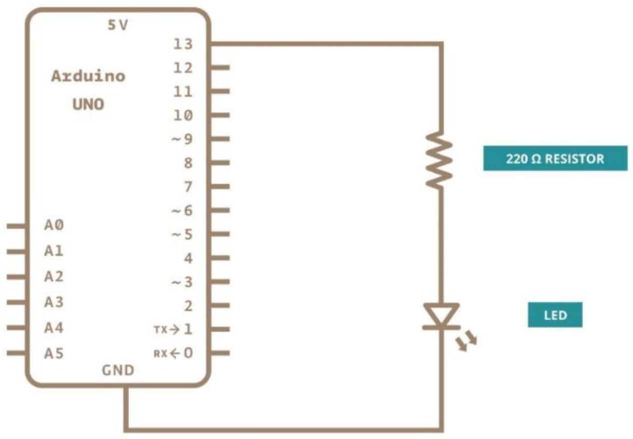


Figure 1: LED interface with ARDUINO

In the circuit diagram, one 220-ohm resistor is connected in series with the LED. This resistor is also called a current-limiting resistor. The Anode of the LED (the longer pin) is connected to one end of the resistor, and the cathode (the shorter pin) is connected to the ground. The other end of the resistor is connected to the Arduino pin. Now connect the Arduino circuit to the computer via a USB cable. Write the program in Arduino IDE. When the program is uploaded, the microcontroller on the Arduino board executes the program, and the LED will blink according to the code written.

PROGRAM

```
int LEDpin = 13;
int delayT = 1000;
void setup() {
  // put your setup code here, to run once:
pinMode(LEDpin, OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(LEDpin, HIGH);
delay(delayT);
digitalWrite(LEDpin, LOW);
delay(delayT);
}
```

WORKING PROCEDURE

setup() and loop() are two fundamental Arduino functions for controlling the behaviour of the board. The Arduino framework automatically calls these functions, which form the foundation of any Arduino program.

The setup() function is only called once when the Arduino board boots up or is reset. Its goal is to set pin modes, initialize variables, and execute any other necessary setup tasks before the main loop begins. This function can be used to configure settings that should only be changed once over the board's lifespan.

The loop() function is the heart of an Arduino program. After the setup() function is executed, the loop() function starts running repeatedly until the Arduino is powered off or reset. It contains the main code that performs the desired tasks, controls the board, user input. Whatever is included in the loop() function will be executed in a continuous loop, allowing the Arduino to perform its intended functions continuously.

In the code, two integers are declared, **LEDpin** and **delayT. LEDpin** represents the pin number of the Arduino where LEDs need to be connected, and **delayT** is an integer variable for the delay() function. The delay() function accepts values in milliseconds.

PROCEDURES:

- Open Arduino IDE.
- Click File, select new project. Type the program, then choose the desired location and give the name of the project and click Save.
- Select Board as Arduino UNO and desired Port and press OK
- Compile the program by clicking Sketch and select compile. Compilation is done by clicking Ctrl+R or by using shortcut √
- After compilation, check any error present in the program and correct if any error is present and compile again till no errors.
- Click Sketch and select Upload to load the code in Arduino
- Do the connections as per the above circuit in bread board and check the LED toggling.
- Check Verify after Programming.
- Connect the Embedded Kit to PC via USB Cable.
- Click Start and check for the software to finish uploading.
- Press Reset button in the embedded kit and check the relative output in the Embedded Kit.

RESULT:

Thus the interfacing of LED with Arduino Microcontroller was done and the program to toggle LED at equal time delay was executed successfully.

EX.NO : 02	LED INTENSITY CONTROL USING LPC2148
DATE :	

AIM :

To control the intensity of LED by generating PWM signal using ARM 7-LPC2148 Microcontroller.

SOFTWARES/HARDWARES REQUIRED:

S.No	Description	Specification	Quantity
1	ARM Development Kit	LPC 2148	1
2	Keil µVision3 IDE	-	-
3	Flash Magic	-	-
4	USB Cable	-	1

THEORY:

Pulse Width Modulation (PWM) is very useful technique for controlling analog circuits with processor's digital outputs. PWM is used in a wide variety of applications ranging from measurement and communications to power control. PWM is a famous technique to generate a signal of varying duty cycle.

Registers of PWM in LPC2148:

PWMPR (PWM Prescale Register): The 32-bit register which hold the maximum value of prescale counter after which it reset

PWMPC (PWM Prescale Counter) : The 32-bit PC is a counter which is incremented to the value stored in PWMPR (Prescale Register) when value in PWMPR is reached, The PWMTC (Timer Counter) is incremented and PWMPC is cleared.

PWMTC (PWM Timer Counter): This is 32-bit Timer Counter which gets incremented whenever PWMPC Prescale Counter value reaches to its maximum value as specified in PWMPR

PWMTCR: PWM Timer Control Register– PWMTCR is used to control the timer counter functions. The Timer Counter can be disable or reset through the PWMTCR.

PWMMCR : PWM Match Control Register– The PWMMCR is used to control if an interrupt is generated and if the PWMTC is reset when match occurs.

PWMMR0-PWMMR6 : PWM Match Register 0- PWM Match Register 6– PWMMR0-6 can be enabled through PWMMCR to reset the PWMTC, stop both the PWMTC and PWMPC, and/or generate an interrupt when it matches the PWMTC. In addition, a match between

PWMMR0-PWMMR6 and the PWMTC sets all PWM outputs that are single edge mode and sets PWM1 if it is in double-edge mode.

PWMIR : PWM Interrupt Register– The PWMIR can be written to clear interrupt. The PWMIR can be read to identify which of the possible interrupt sources are pending.

PWMLER : PWM Latch Enable Register– Enables use of new PWM Match values

PWMPCR : PWM Control Register– Enables PWM outputs and selects PWM channel types as either single
edge or double edge controlled.

PINSEL0 : - PINSEL0 is used to configure PORT0 pins P0.0 to P0.15.

PINSEL0 register

Register bits	30-31	28-29	26-27	24-25	22-23	20-21	18-19	16-17	14-15	12-13	10-11	8-9	6-7	4-5	2-3	0-1
Port pins	P0.15	P0.14	P0.13	P0.12	P0.11	P0.10	P0.9	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

Register Value	Functions
00	Default Function= GPIO
01	TXD0 (Transmit Data for UART0)
10	PWM1.1 (Pulse Width Modulation output)
11	Reserved

PROCEDURES FOR KEIL AND FLASH MAGIC:

- Open Keil uVison 4.
- Click Project in tool bar then select New uVision Project, then choose the desired location and give the name of the project and click Save.
- In the pop-up menu, under device, select Legacy device database and then select LPC2148 in the drop-down menu and click OK.
- Again, in the pop up menu click Yes to Copy the Startup code to project folder.
- Click File ->New to create a new file and save it as main.c .
- Copy the required Header Files to your project folder.
- In project workspace bar in left, open yourProject: name -> Target 1 -> Source Group 1.
- Right click on the Source Group 1 folder and select Add Existing Files to Group 'Source Group 1'... and select the created main.c file and the required header files.
- Write the required program in main.c file and save it.
- Right click on Target 1 in left side project workspace and select Options for Target

‘Target 1’.
- In the pop up menu, select Output tab and check Create Hex File. Also select Linker tab and check Use Memory Layout from Target Dialog and click OK.
- Compile and Built the program.
- Open Flash Magic.
- Click Select Device and select ARM7 -> LPC2148.
- Set the correct COM PORT where Embedded Kit is connected.
- Set the Baud Rate as 9600 and Oscillator (MHz) as 12.00.
- Check Erase all Flash+Code Rd Prot.
- Click Browse and select the required HEX File of your program from your project folder.
- Check Verify after Programming.
- Connect the Embedded Kit to PC via USB Cable.
- Click Start and check for the software to finish uploading.
- Press Reset button in the embedded kit and check the relative output in the Embedded Kit.

SOURCE CODE:

```
#include <LPC214X.H>

int main(void)
{
        PINSEL0=2;                  // P0.0 is configured as PWM1 output
        PWMPC=0;                    //PWM Pre scale counter value is 0; it is not used
        PWMPR=0;                    //PWM Pres scale register value is 0; it is not used
        PWMMR0 = 12000000;// Match register 0 has count for period; in this case approx. 1 sec.
                                    // Peripheral clock is 12 MHz
        PWMMR1 = 1200000;           // On time approx. 0.45 sec
        PWMMCR=0X2;         // when match occurs with the value in Match Register 0, reset

        //PWMLER = 0x3;             // Enable PWM Match 0 Latch and PWM Match1 Latch
                                    // to transfer the values to shadow registers
        PWMPCR = 1<<9;              // to enable PWM1 output (9th bit)
        PWMTCR = 0x9;               // to enable Timer counter (0th bit) and PWM(3rd bit)
        while(1);                   // idling
}
// to vary the on time, the content of PWMMR1 is to be changed.  Maximum time is period
// which is stored in PWMMR0
```

RESULT:

Thus the the intensity of LED is controlled by generating PWM signal using ARM 7-LPC2148 Microcontroller.

EX.NO : 03	DISPLAY A CHARACTER IN A 16*2 LCD
DATE :	

AIM:

To interface the LCD with ARM 7 – LPC2148 Microcontroller and to display a character in LCD.

SOFTWARES/HARDWARES REQUIRED:

S.No	Description	Specification	Quantity
1	ARM Development Kit	LPC 2148	1
2	Keil µVision3 IDE	-	-
3	Flash Magic	-	-
4	USB Cable	-	1

THEORY:

LCDs (Liquid Crystal Displays) are used for displaying status or parameters in embedded systems. LCD 16x2 is a 16-pin device which has 8 data pins (D0-D7) and 3 control pins (RS, RW, EN). The remaining 5 pins are for supply and backlight for the LCD. The control pins help us configure the LCD in command mode or data mode. They also help configure read mode or write mode and also when to read or write. LCD 16x2 can be used in 4-bit mode or 8-bit mode depending on the requirement of the application. In order to use it, certain commands to the LCD need to be send in command mode and once the LCD is configured according to need, the required data will be send in data mode to display on it.

Steps for Sending Data:

- step1: Send the character to LCD.
- step2: Select the Data Register by making RS high.
- step3: Select Write operation making RW low.
- step4: Send a High-to-Low pulse on Enable PIN with some delay_us.

Steps for Sending Command:

- step1: Send the I/P command to LCD.
- step2: Select the Control Register by making RS low.
- step3: Select Write operation making RW low.
- step4: Send a High-to-Low pulse on Enable PIN with some delay_us.

SOURCE CODE:

```
#include <lpc214x.h>
#include <string.h>
void delay(void)                    // delay routine
{
        unsigned int i;
        i = 0xfff;
        while (i--);
}
void write(unsigned char data)
{
        IOSET0 = data <<16;
                         // data is shifted left by 16 bits for sending to data bits (P0.23 - P0.16)
        delay();                    // delay is called
        IOSET1 = 1 << 31;     // enable bit is set to 1
        delay();                    // delay is called
        IOCLR1 = 1 << 31;     // enable bit is made to 0
        delay();                    // delay is called
        IOCLR0 = data <<16; // data lines are cleared
}


void cmd_write(unsigned char data)
{
        IOCLR1 = 1 << 30;     // make RS as 0
        IOCLR1 = 1 <<31;      // make enable bit as 0
        write(data);          // write routine is called
}

void data_write(unsigned char data)
{
        IOSET1 = 1 << 30;     // make RS as 1
        IOCLR1 = 1 <<31;      // make enable bit as 0
        write(data);          // write routine is called
}
void lcd_init()
{
        cmd_write(0x38);      // 5/8 matrix and 8 bit data
        cmd_write(0xe);       // first byte of the initialization sequence
        cmd_write(0x1);       // clear the display
        cmd_write(0x6);       // the cursor moves right when a data is entered
}
void disp(unsigned char *msg)          // display message routine, * denotes the address
pointer of the msg array
{
        unsigned int i,j;
        j = strlen(msg);                 // function to find the length of the message
```

```
          for (i = 0; i<j; i ++)       // for the string length, for loop is repeated
                    {
                    data_write (msg[i]); // each character is written
                    }
}
int main(void)                         // main program
{
          PINSEL1 = 0 ;                          // P0.16 to P0.31 are I/O
          IODIR0 = 0xff << 16;  // P0.16 to P0.23 are output lines (data lines)
          PINSEL2 = 0;           // P1 is I/O
          IODIR1 = 0x3 <<30;    // make P1.30 and P1.31 as output lines
          lcd_init();            // LCD initialization routine is called
          delay();                         // delay routine is called
          cmd_write(0x80);       // cursor is in I row and I column
          disp ("    Welcome to      ");     // message is displayed in first row
          cmd_write (0xc0);      // cursor is in II row and I column
          disp ("Electronics Dept"); // message is displayed in second row
          while (1) ;            // idling
}
```

<u>**Registers used in the program**</u>

1.**IOxPIN (GPIO Port Pin value register):** This is a 32-bit wide register. This register is used to read/write the value on Port (PORT0/PORT1). But care should be taken while writing. Masking should be used to ensure write to the desired pin.
2. **IOxSET (GPIO Port Output Set register) :** This is a 32-bit wide register. This register is used to make pins of Port (PORT0/PORT1) HIGH. Writing one to specific bit makes that pin HIGH. Writing zero has no effect.

3. **IOxDIR (GPIO Port Direction control register) :** This is a 32-bit wide register. This register individually controls the direction of each port pin. Setting a bit to '1' configures the corresponding pin as an output pin. Setting a bit to '0' configures the corresponding pin as an input pin.

4. **IOxCLR (GPIO Port Output Clear register) :** This is a 32-bit wide register. This register is used to make pins of Port LOW. Writing one to specific bit makes that pin LOW. Writing zeroes has no effect.

5. PINSEL0 : - PINSEL0 is used to configure PORT0 pins P0.0 to P0.15.
6. **PINSEL1 : -** PINSEL1 is used to configure PORT0 pins P0.16 to P0.31
7. **PINSEL2 : -** PINSEL2 is used to configure PORT1 pins P1.16 to P1.31

<u>**WORKING PROCEDURE**</u>

1. Configure the Port0 pins P0.16 to P0.31 and Port 1 pins as Input/Output pins using PINSEL1 = 0 and PINSEL2 = 0 respectively.

2. Make P0.16 to P0.23 as output lines (data lines) using command IODIR0 = 0xff << 16 and make P1.30 and P1.31 as output lines using IODIR1 = 0x3 <<30;
3. Initialize the LCD by calling LCD initialization routine (void lcd_init())
4. After certain delay produced by delay routine (void delay(void)), cursor is placed in I row and I column by calling subroutine void cmd_write(unsigned char data)
5. Message is displayed in first row by calling subroutine void disp(unsigned char *msg)
6. Now cursor is placed in II row and I column by calling subroutine void cmd_write(unsigned char data)
7. Then Message is displayed in second row by calling subroutine void disp(unsigned char *msg)
8. Repeat the program.

RESULT:

Thus the interfacing of LCD with ARM 7 - LPC2148 Microcontroller is done and a character is displayed in LCD.

EX.NO : 04	INTERFACING OF STEPPER MOTOR USING LPC2148
DATE :	

AIM:

To interface the stepper motor with ARM 7 – LPC2148 Microcontroller and to rotate in clockwise and anticlockwise directions using LPC2148.

SOFTWARES/HARDWARES REQUIRED:

S.No	Description	Specification	Quantity
1	ARM Development Kit	LPC 2148	1
2	Keil µVision3 IDE	-	-
3	Flash Magic	-	-
4	IMIK Evaluation board	-	1
5	Stepper motor	-	1
6	Connecting wires	-	As Required

THEORY:

A stepper motor is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation. Every revolution of the stepper motor is divided into a discrete number of steps, and the motor must be sent a separate pulse for each step.

Interfacing the stepper motor with ARM7 LPC2148:

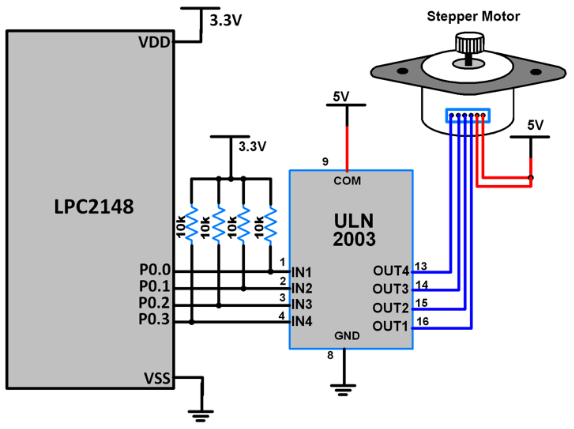


Figure 3: Interfacing diagram of stepper motor with ARM 7-LPC2148

In this the stepper motor is connected with Microcontroller output port pins through a ULN2803A array. So, when the microcontroller is giving pulses with particular frequency to ULN2803A, the motor is rotated in clockwise or anticlockwise.

Controlling a stepper motor using LPC2148 Primer Board. It works by turning ON & OFF a four I/O port lines generating at a particular frequency.The ARM7 LPC2148 Primer board has four numbers of I/O port lines, connected with I/O Port lines (P1.16 – P1.19) to rotate the stepper motor. ULN2803 is used as a driver for port I/O lines, drivers output connected to stepper motor, connector provided for external power supply if needed.

PROCEDURES:

- Open Keil uVison 4.
- Click Project in tool bar then select New uVision Project, then choose the desired location and give the name of the project and click Save.
- In the pop-up menu, select NXP, then select LPC2148 in the drop-down menu and click OK.
- Again, in the pop up menu click Yesto Copy the Startup code to project folder.
- Click File ->New to create a new file and save it as main.c .
- Copy the required Header Files to your project folder.
- In project workspace bar in left, open yourProject: name -> Target 1 -> Source Group 1.
- Right click on the Source Group 1 folder and select Add Existing Files to Group 'Source Group 1'... and select the created main.c file and the required header files.

  Write the required program in main.c file and save it.

- Right click on Target 1 in left side project workspace and select Options for Target 'Target 1'.
- In the pop up menu, select Output tab and check Create Hex File. Also select Linker tab and check Use Memory Layout from Target Dialog and click OK.
- Compile and Built the program.
- Open Flash Magic.
- Click Select Device and select ARM7 -> LPC2148.
- Set the correct COM PORT where Embedded Kit is connected.
- Set the Baud Rate as 9600 and Oscillator (MHz) as 12.00.
- Check Erase all Flash+Code Rd Prot.
- Click Browse and select the required HEX File of your program from your project folder.
- Check Verify after Programming.
- Connect the Embedded Kit to PC via USB Cable.
- Click Start and check for the software to finish uploading.
- Press Reset button in the embedded kit and check the relative output in the Embedded Kit.

SOURCE CODE:

```
 #include<LPC214x.h>
#include<stdio.h>
#define  SW1 24            //SW1 (P1.24)
#define  SW2 25            //SW2 (P1.25)
#define  SW3 26            // SW3    (P1.26)
#define  COIL_A 16         //change the Stepper Motor Port!

voidmotor_cw(void);
void motor_ccw(void);
void delay(int);
{
unsigned char i = 0;
PINSEL2&=  0xFFFFFFF3;        //P1.16 – P1.31 as GPIO IODIR1=0x000F0000;


                                                    //P1.16 – P1.19 as Output
while(1)
{
if(!(IOPIN1 & (1<<SW1)))           // switch SW1 ON/OFF
{
motor_cw();
}
else if (!IOPIN1 & (1<<SW2))                 // switch SW2 ON/OFF
{
motor_ccw();
}
else  if(!(IOPIN1  & (1<<SW3)))              // switch SW3 ON/OFF
{
while(i<12)
{
motor_cw(); i++;
}}
else if(i=0);
}}

Void delay(int n)
{
int j;
for(i=0;i<n;i++)
{
for(j=0;j<0x3FF0;j++)
{
}}}
Void motor_ccw(void)
{
unsigned int i=0;
while (STEP[i] !='\0')
{
IOSET1 = STEP[i] << COIL_A;
delay();
IOCLR1 = STEP[i] << COIL_A;
delay(); I++;
}}

void motor_cw(void)
{
Int i=7;


while(i>=0)
{
IOSET1 = STEP[i] << COIL_A;
Delay();
IOCLR1 = STEP[i] << COIL_A;
delay();
i--;
}}
```

APPLICATIONS:

- As the stepper motor are digitally controlled using an input pulse, they are suitable for use with computer controlled systems.
- They are used in numeric control of machine tools.
- Used in tape drives, floppy disc drives, printers and electric watches.
- The stepper motor also use in X-Y plotter and robotics.

RESULT:

Thus the interfacing of the stepper motor with ARM 7 – LPC2148 Microcontroller is done and it is rotated in clockwise and anti-clockwise directions.

Ex. N0.: 5	**REAL TIME CLOCK**
DATE :	

AIM:

To implement real time clock LPC2148 microcontroller.

SOFTWARES/HARDWARES REQUIRED:

S.No	Description	Specification	Quantity
1	ARM Development Kit	LPC 2148	1
2	Keil µVision3 IDE	-	-
3	Flash Magic	-	-
4	IMIK Evaluation board	-	1

THEORY:

A real-time clock (RTC) is a computer clock that keeps track of the current time. Although the term often refers to the devices in personal computers, servers and embedded systems, RTCs are present in almost any electronic device which needs to keep accurate time. It demonstrates the principle behind the interfacing of RTC with ARM LPC2148. A better, cost-effective alternative is to implement the RTC functionality into a microcontroller ARM LPC2148 that performs other tasks as well. RTC is used even when the system is not in operation. RTC is inbuilt part of any electronics device.

PROCEDURES:

- Open Keil uVison 4.
- Click Project in tool bar then select New uVision Project, then choose the desired location and give the name of the project and click Save.
- In the pop-up menu, select NXP, then select LPC2148 in the drop-down menu and click OK.
- Again, in the pop up menu click Yesto Copy the Startup code to project folder.
- Click File ->New to create a new file and save it as main.c .
- Copy the required Header Files to your project folder.
- In project workspace bar in left, open yourProject: name -> Target 1 -> Source Group 1.
- Right click on the Source Group 1 folder and select Add Existing Files to Group
- 'Source Group 1'... and select the created main.c file and the required header files.
- Write the required program in main.c file and save it.
- Right click on Target 1 in left side project workspace and select Options for Target 'Target 1'.
- In the pop up menu, select Output tab and check Create Hex File. Also select Linker tab and check Use

Memory Layout from Target Dialog and click OK.

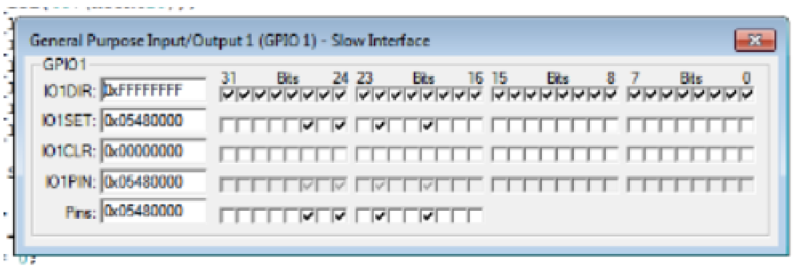
- Compile and Built the program.
- Open Flash Magic.
- Click Select Device and select ARM7 -> LPC2148.
- Set the correct COM PORT where Embedded Kit is connected.
- Set the Baud Rate as 9600 and Oscillator (MHz) as 12.00.
- Check Erase all Flash+Code Rd Prot.
- Click Browse and select the required HEX File of your program from your project folder.
- Check Verify after Programming.
- Connect the Embedded Kit to PC via USB Cable.
- Click Start and check for the software to finish uploading.
- Press Reset button in the embedded kit and check the relative output in the Embedded Kit.

SOURCE CODE:

```
#include<lpc21xx.h>  // header file for LPC21XX series
#define  rs (1<<24)    // register select pin
#define  rw (1<<25)    // read write pin
#define  en (1<<26)    // enable pin
void  delay(int  j )       // Time delay function in milli seconds
{
int i; for(;j;j--)
for(i=6000;i;i--);
}
void data_lcd(char  ch)          // Function to send data to LCD
{
int i =0; i = ch;
i = i<<16;
IOPIN1 &=(0XFF00FFFF); IOPIN1 |= i;
IOSET1 = rs; IOCLR1 = rw; IOSET1 = en;
delay(2); IOCLR1  = en;
}
void cmd_lcd(char  ch)          // Function to send command to LCD
{
int i =0; i = ch;
i = i<<16;
IOPIN1 &=(0XFF00FFFF); IOPIN1 |= i;
IOCLR1 = rs; IOCLR1 = rw;

IOSET1 = en;
delay(2); IOCLR1  = en;
}
void init_lcd()  / Funtion to Initialize LCD
{
cmd_lcd(0x38);         // for using 8-bit 2 row mode and 5x7 Dots of LCD
cmd_lcd(0x01);         // clear screen
cmd_lcd(0x06);         // display ON
cmd_lcd(0x0c);         // force cursor to beginning of second row
cmd_lcd(0x80);         // clear screen
}
void str_lcd(char  *str)          // Function to display it in LCD
{
while(*str) data_lcd(*str++);
}
void time(void)           // function to perfom the operation of clock
{
cmd_lcd(0x80); str_lcd("HH:MM:SS"); cmd_lcd(0xc0); data_lcd(48+(HOUR/10)); data_lcd(48+(HOUR%10));
data_lcd(':'); data_lcd(48+(MIN/10)); data_lcd(48+(MIN%10)); data_lcd(':'); data_lcd(48+(SEC/10));
data_lcd(48+(SEC%10));
}
void SetTime(void)     // function to initialize RTC
{
CCR = 0x02; HOUR = 0;
MIN = 0;
SEC = 0; CCR = 0x11;
}
int  main(void)
{
SetTime();
PINSEL2 = 0X00000000;                  // select PORT1 as GPIO mode
IODIR1 = 0XFFFFFFFF;       // make PORT1 pin as Output mode init_lcd();
while  (1)        // Repeat(loop) forever
{
time();
}
}
```

OUTPUT:



```
General Purpose Input/Output 1 (GPIO 1) - Slow Interface
GPIO1
          31   Bits   24 23   Bits   16 15   Bits   8 7   Bits   0
IO1DIR:  0xFFFFFFFF
IO1SET:  0x05480000
IO1CLR:  0x00000000
IO1PIN:  0x05480000
Pins:    0x05480000
```

RESULT:

Thus the real time clock is implemented using LPC2148.

Ex. No: 7	GAS MONITORING SYSTEM
Date:	

AIM:

To monitor LPG gas leakage by interfacing gas sensor with LPC2148 microcontroller.

SOFTWARES/HARDWARES REQUIRED:

S.No	Description	Specificatio n	Quantity
1	ARM Development Kit	LPC 2148	1
2	Keil µVision3 IDE	-	-
3	Flash Magic	-	-
4	IMIK Evaluation board	-	1
5	Sensor Module	Gas	1
6	LCD Module	-	1

THEORY:

Gas sensor is used to detect the gas leakage. The electrical properties of the sensor would change with variations in gas concentration. Sensitive material of MQ-6 gas sensor is SnO2, which with lower conductivity in clean air. When the target combustible gas exist, the sensor's conductivity is higher along with the gas concentration rising. MQ-6 gas sensor has high sensitivity to Propane, Butane and LPG, also response to Natural gas. The sensor could be used to detect different combustible gas, especially Methane; it is with low cost and suitable for different applications

PROCEDURE:

- Open Keil uVison 4.
- Click Project in tool bar then select New uVision Project, then choose the desired location and give the name of the project and click Save.
- In the pop-up menu, select NXP, then select LPC2148 in the drop-down menu and click
- OK.
- Again, in the pop up menu click Yes to Copy the Startup code to project folder.
- Click File ->New to create a new file and save it as main.c .
- Copy the required Header Files to your project folder.
- In project workspace bar in left, open your Project: name -> Target 1 -> Source Group 1.
- Right click on the Source Group 1 folder and select Add Existing Files to Group 'Source Group 1'... and select the created main.c file and the required header files.
- Write the required program in main.c file and save it.
- Right click on Target 1 in left side project workspace and select Options for Target 'Target 1'.

- In the pop up menu, select Output tab and check Create Hex File. Also select Linker
- tab and check Use Memory Layout from Target Dialog and click OK.
- Compile and Built the program.

- Open Flash Magic.
- Click Select Device and select ARM7 -> LPC2148.
- Set the correct COM PORT where Embedded Kit is connected.
- Set the Baud Rate as 9600 and Oscillator (MHz) as 12.00.
- Check Erase all Flash+Code Rd Prot.
- Click Browse and select the required HEX File of your program from your project folder.
- Check Verify after Programming.
- Connect the Embedded Kit to PC via USB Cable.
- Click Start and check for the software to finish uploading.
- Press Reset button in the embedded kit and check the relative output in the Embedded Kit.

SOURCE CODE:

```
* Gas Sensor Interfacing with LPC2148
 * Done by EmbeTronicX
 */
#include<lpc214x.h>
#define bit(x) (1<<x)
#define delay for(i=0;i<7000;i++);

#define GAS (IO1PIN & (1<<24))

unsigned int i;

void lcd_int(void);
void dat(unsigned char);
void cmd(unsigned char);
void string(unsigned char *);

void main()
{
   IO0DIR =0XFFF;
   IO1DIR = 0x0;
   lcd_int();
   cmd(0x80);
   string("EMBETRONICX.COM ");
   while(1) {
         if(GAS) {
               string("Gas Detected");
         }
         delay;delay;
         cmd(0x01);
      }
}

void lcd_int()
{
   cmd(0x38);
   cmd(0x0c);
   cmd(0x06);
   cmd(0x01);
   cmd(0x80);
}

void cmd(unsigned char a)
```

```
{
   IO0PIN&=0x00;
   IO0PIN|=(a<<0);
   IO0CLR|=bit(8);                //rs=0
   IO0CLR|=bit(9);                //rw=0
   IO0SET|=bit(10);               //en=1
   delay;
   IO0CLR|=bit(10);               //en=0
}

void dat(unsigned char b)
{
   IO0PIN&=0x00;
   IO0PIN|=(b<<0);
   IO0SET|=bit(8);                //rs=1
   IO0CLR|=bit(9);                //rw=0
   IO0SET|=bit(10);               //en=1
   delay;
   IO0CLR|=bit(10);               //en=0
}

void string(unsigned char *p)
{
   while(*p!='\0') {
      dat(*p++);
   }
}
```

RESULT:

Thus the leakage of gas is monitored using ARM LPC2148.

Ex. No: 10	HOME AUTOMATION USING LPC2148
Date:	

AIM:

To implement home automation using Bluetooth and LPC2148.

SOFTWARES/HARDWARES REQUIRED:

S.No	Description	Specification	Quantit y
1	ARM Development Kit	LPC 2148	1
2	Keil µVision3 IDE	-	-
3	Flash Magic	-	-
4	IMIK Evaluation board	-	1
5	Bluetooth module	-	1

THEORY:

Bluetooth has a wide range of applications like data transfer, wireless music, device control etc. The reason Bluetooth is used in such applications is its ease of implementation, low power consumption and the most important thing is its availability i.e. almost all mobile phone (whether it may a simple feature phone or an advanced smart phone) is equipped with Bluetooth. ARM 7 based LPC2148 is one of the successful and popular 32 – bit microcontrollers. It has many on – chip features. One of the major applications of Bluetooth module is in the field of Home Automation and Smart Home systems, where a Bluetooth enabled device like a smart phone or tablet can be used to control the smart home application.

SCHEMATIC DIAGRAM:

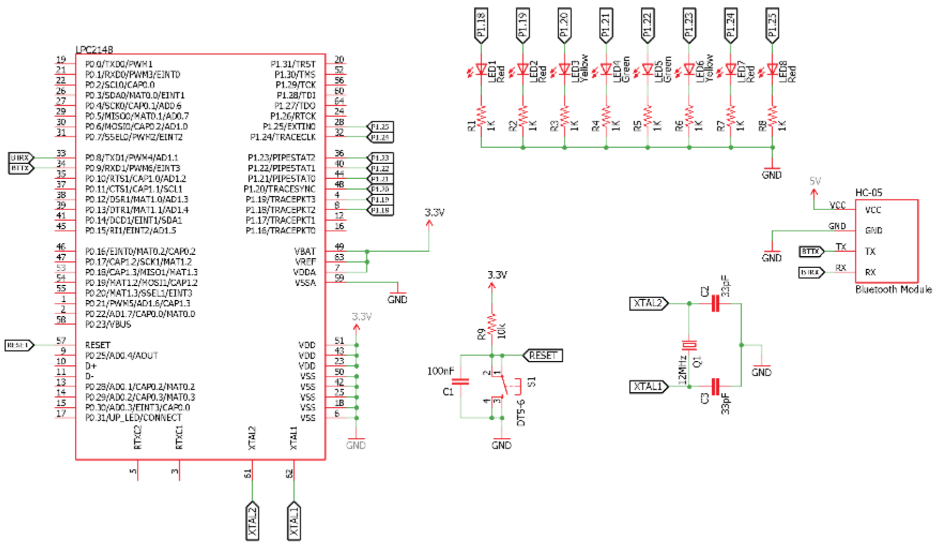


Figure 6: Interfacing Bluetooth module with LPC2148

PROCEDURES:

- Open Keil uVison 4.
- Click Project in tool bar then select New uVision Project, then choose the desired location and give the name of the project and click Save.
- In the pop-up menu, select NXP, then select LPC2148 in the drop-down menu and click
- OK.
- Again, in the pop up menu click Yes to Copy the Startup code to project folder.
- Click File ->New to create a new file and save it as main.c .
- Copy the required Header Files to your project folder.
- In project workspace bar in left, open your Project: name -> Target 1 -> Source Group 1.
- Right click on the Source Group 1 folder and select Add Existing Files to Group 'Source Group 1'... and select the created main.c file and the required header files.
- Write the required program in main.c file and save it.
- Right click on Target 1 in left side project workspace and select Options for Target 'Target 1'.
- In the pop up menu, select Output tab and check Create Hex File. Also select Linker
- tab and check Use Memory Layout from Target Dialog and click OK.
- Compile and Built the program.
- Open Flash Magic.
- Click Select Device and select ARM7 -> LPC2148.
- Set the correct COM PORT where Embedded Kit is connected.
- Set the Baud Rate as 9600 and Oscillator (MHz) as 12.00.
- Check Erase all Flash+Code Rd Prot.
- Click Browse and select the required HEX File of your program from your project folder.
- Check Verify after Programming.
- Connect the Embedded Kit to PC via USB Cable.
- Click Start and check for the software to finish uploading.
- Press Reset button in the embedded kit and check the relative output in the Embedded Kit.

SOURCE CODE:

```
#include <lpc214x.h>

#define CLK_DIV (15-1)
#define PowerUP (1<<21)
#define Sel_ADC_Cha (1<<1)
#define ADC_START (1<<24)
#define ADC_DONE (1<<31)

void initPLL(void);
void initPWM(unsigned int periodPWM);
void delay(int);

void initPLL (void)
{
  PLL0CON = 0x01;
  PLL0CFG = 0x24;
  PLL0FEED = 0xAA;
  PLL0FEED = 0x55;
  while(!(PLL0STAT & 0x00000400));
  PLL0CON = 0x03;
  PLL0FEED = 0xAA;
  PLL0FEED = 0x55;
  VPBDIV = 0x01;
}

void delay (int d)
{
unsigned int i,j;
  for(j=0;j<d;j++)
  {
    for(i=0;i<10;i++);
  }
}

void initPWM(unsigned int periodPWM)
{
  PINSEL0 = 0x00000002;
  PWMTCR = (1<<1);
  PWMPR = 0X00;
  PWMMCR = (1<<0)|(1<<1);
  PWMMR0 = periodPWM;
  PWMLER = (1<<0);
  PWMTCR = (1<<0) | (1<<3);

}

int main()
{
  int dutycycle;
  unsigned short int adcResult;
  initPLL();
  PINSEL1 = 0x01000000;
  AD0CR = ((CLK_DIV<<8) | PowerUP);
  initPWM(255);
  PWMPCR |= (1<<9);
  while(1)
  {
    AD0CR |= Sel_ADC_Cha;
    delay(10);
    AD0CR |= ADC_START;
    while( (AD0DR1 & ADC_DONE) == 0 );
    adcResult = (AD0DR1>>6) & 0x3ff;
    AD0CR &= ~(ADC_START);
    dutycycle = adcResult/4;
    PWMMR1 = dutycycle;
    PWMLER |= (1<<1);
  }
}
```

<u>RESULT:</u>

Thus home automation is implemented using Bluetooth module and LPC2148 microcontroller.

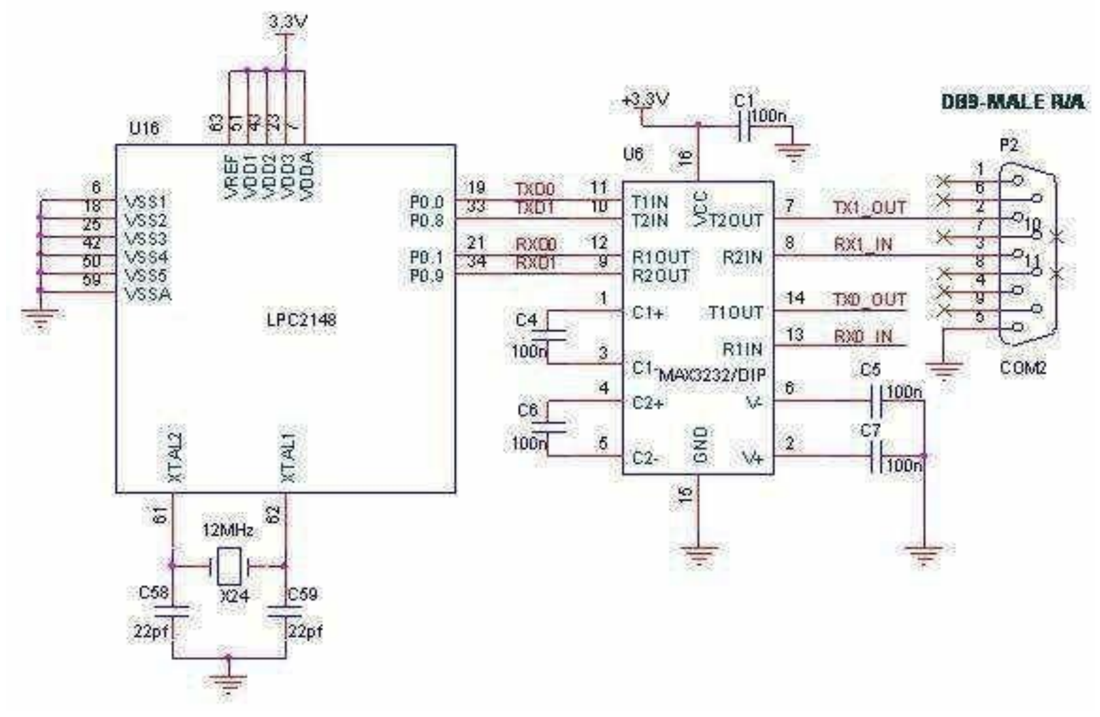
MINI PROJECT

INTERFACING BLUETOOTH MODULE

AIM :

To write an embedded C code for interfacing Bluetooth Module with ARM CORTEX M3-LPC2148.

CIRCUIT DIAGRAM:



Description

HC-05 module is an easy-to-use Bluetooth SPP (Serial Port Protocol) module, designed for a transparent wireless serial connection setup. The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication. This serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with a complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single-chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature).

The Bluetooth module HC-05 is a MASTER/SLAVE module. By default the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device but can accept connections. The master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish a connection between MCU and GPS, PC to your embedded project, etc.

Procedure:

Give +3.3V power supply to LPC2148 Primer Board; connect the 5V adapter with Bluetooth module which is connected with the LPC2148 Primer Board. There are two Bluetooth modules are required. One is connected with LPC2148 Primer Board; other one is connected with PC. First connect the serial cable between LPC2148 Primer board & PC. Then open the Hyper Terminal screen, select which port you are using and set the default settings. Now the screen should show some text messages.

If the messages are correctly displayed in Hyper Terminal, then only connect the Bluetooth modules in LPC2148 Primer Board UART0 & PC. If you are not reading any data from UART0, then you just check the jumper connections & just check the serial cable is working. Otherwise you just check the code with debugging mode in Keil.

Program:

```
#defineCR0x0D
#include      <LPC21xx.H>
void  init_serial  (void);  int
putchar (int ch);
int  getchar  (void);
unsigned       char
test; int main(void)
{
char *Ptr = "*** UART0 Demo ***\n\n\rType Characters to be echoed!!\n\n\r";
VPBDIV        =        0x02;   //        Divide  Pclk     by       two init_serial();
while(1)
{
while (*Ptr) { putchar(*Ptr++);
} putchar(getchar()); // Echo terminal
}
}
void init_serial (void)
{
PINSEL0 = 0x00000005; // Enable RxD0 and TxD0 U0LCR = 0x00000083; //8 bits,  no Parity, 1
Stop bit
U0DLL = 0x000000C3; //9600 Baud Rate @ 30MHz VPB Clock U0LCR = 0x00000003;
}
int putchar (int ch)
{
if (ch == '\n')
{
while(!(U0LSR & 0x20)); U0THR = CR;
}
While (!(U0LSR & 0x20));
return (U0THR= ch);
}
int getchar (void)
{
while !(U0LSR & x01)); return (U0RBR);

}
```

RESULT

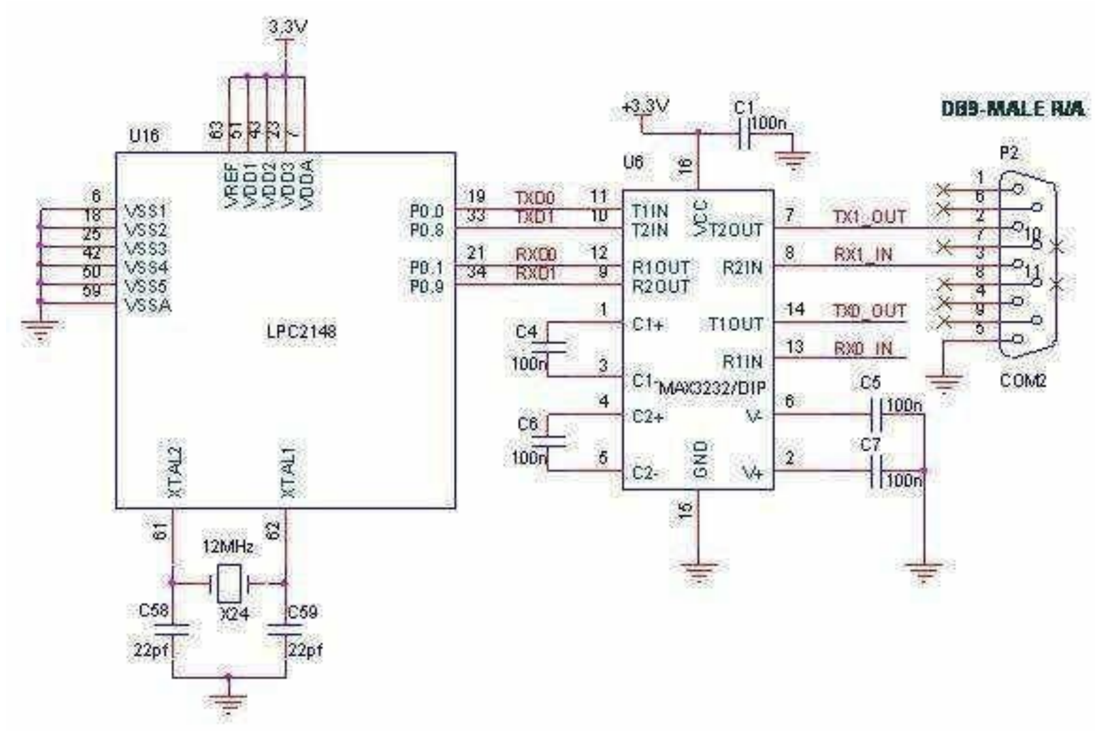
Thus the Bluetooth Module is interfaced with ARM CORTEX M3-LPC2148 and verified successfully.

INTERFACING GPS MODULE

AIM:

To write an embedded C code for interfacing GSM Module with ARM CORTEX M3-LPC2148.

CIRCUIT DIAGRAM:



Description

Global **P**ositioning **S**ystem (GPS) makes use of signals sent by satellites in space and ground stations on Earth to accurately determine their position on Earth. Radio Frequency signals sent from satellites and ground stations are received by the GPS. GPS makes use of these signals to determine its exact position. The GPS itself does not need to transmit any information. The signals received from the satellites and ground stations contain time stamps of the time when the signals were transmitted.

By calculating the time difference between the time the signal was transmitted and the time the signal was received, and using the speed of the signal, the distance between the satellites and the GPS can be determined using a simple formula for distance using speed and time. Using information from 3 or more satellites, the exact position of the GPS can be triangulated. The GPS receiver module uses UART communication to communicate with controller or PC terminal.

Procedure:

Give +3.3V power supply to LPC2148 Primer Board; connect the +5V adapter with GSM module which is connected with LPC2148 Primer Board through UART0. Open the Hyper Terminal screen, select which port you are using and set the default settings. Now the screen should show some text messages.

The following Commands and sequence of events performed for sending text message to a mobile phone through GSM Modem interfaced with microcontroller:

First select the text mode for SMS by sending the following AT Command to GSM Modem: AT+CMGF = 1 . This command configures the GSM modem in text mode.

Send the following AT Command for sending SMS message in text mode along with mobile number to the GSM Modem : AT+CMGS =+923005281046 . This command sends the mobile number of the recipient mobile to the GSM modem.

Send the text message string ("hello!") to the GSM Modem This is a test message from UART".

If you not reading any text from UART0, then you just check the jumper connections & just check the serial cable is working. Otherwise you just check the code with debugging mode in Keil. If you want to see more details about debugging just see the videos in below link.

If you not reading any text from UART0, then you just check the jumper connections & just check the serial cable is working. Otherwise you just check the code with debugging mode in Keil. If you want to see more details about debugging just see the videos in below link.

Program:

```
#define CR 0x0D #include <LPC21xx.H> #include <stdio.h>
void getstring(unsigned char *);
int getchar (void) /* Read character from Serial  Port */ void status_ok(void);
void     Serial_Init(void); void delay(unsigned int n); void main(void)
{
unsigned        int
cnt=0x80,m; char xx; Serial_Init();  delay(50);
while(1)
{
printf("AT\r");   //     AT    COMMAND    FOR    INITIALING    status_ok();
printf("AT+IPR=9600\r"); // AT COMMAND FOR BAUD RATE
status_ok(); printf("AT+CMGR=2\r"); // Reading the message detail // at Index 1 with phone number, data and time status_ok(); delay(250); printf("ATD9790550124;\r");//AT COMMAND FOR CALL DIALING
delay(250); status_ok(); delay(500); delay(500);
delay(500); delay(500); delay(500); delay(500);
printf("ATH\r"); // AT COMMAND FOR CALL DISCONNECTING delay(250);
status_ok(); delay(500); delay(500);
printf("ATDL\r"); // AT COMMAND FOR REDIALING delay(250); status_ok(); delay(500);
delay(500);
printf("ATH\r"); // AT COMMAND FOR ANSWERING THE CALL
delay(250); status_ok();
delay(500); delay(500);
}
}
void getstring(unsigned char *array)
{
unsignedchar temp=0, i=0;
do
{ temp = getchar();
*array++ = temp;
}
while((temp != '\r') && (temp != '\n'))
*array = '\0';
}
int getchar (void) /* Read character from Serial Port */
{
while(!(U0LSR&0x01)
);
return (U0RBR);
}
void status_ok(void)
{
getstring(y); while(!(strstr(y,"OK"))) getstring(y);
pointr=   strstr(y,"OK");   lcd_cmd(0xc0);   lcd_data(*pointr++);   lcd_data(*pointr);   delay(500);
lcd_cmd(0x01);
}
void Serial_Init(void)
{
PINSEL0 |= 0X00000005; //Enable Txd0 and Rxd0 U0LCR = 0x00000083; //8-bit data, no parity,
1- stop bit U0DLL = 0x00000061;   //for    Baud rate=9600,DLL=82                 U0LCR
 = 0x00000003;
//DLAB = 0; }
void delay(unsigned int n)
{
unsigned         int       i,j; for(i=0;i<n;i++)
{ for(j=0;j<12000;j++)
}}
```

RESULT

Thus the GSM Module is interfaced with ARM CORTEX M3-LPC2148 and verified successfully.