EX1 Study of PIC MICRO CONTROLLER SYSTEM

Aim: Study of PIC Microcontroller

What is PIC Microcontroller?

PIC (Programmable Interface Controller) microcontrollers are the worlds smallest microcontroller that can be programmed to carry out a huge range of tasks. These controllers are found in many electronic devices such as phones, computer control systems, alarm systems, embedded systems, etc.

Architecture of PIC Microcontroller

The PIC microcontroller architecture comprises of CPU, I/O, ports, memory organization, A/D converter, timers/counters, interrupts, serial communication, oscillator and CCP module which are discussed in detailed below

CPU (Central Processing Unit)

Its not different from other microcontrollers CPU and the PIC microcontroller CPU consists of the ALU, CU, MU and accumulator, etc. Arithmetic logic unit is mainly used for arithmetic operations and to take logical decisions.

Architecture of PIC microcontroller :

Memory Organization

The memory module in the PIC microcontroller architecture consists of RAM (Random access memory), ROM (Read Only memory) and STACK.

RAM

Ram is an unstable memory which is used to store the data the data temporarily in its registers.

General Purpose Register (GPR)

These registers are used for general purpose only as the name implies. For example, if we want to multiply two numbers.

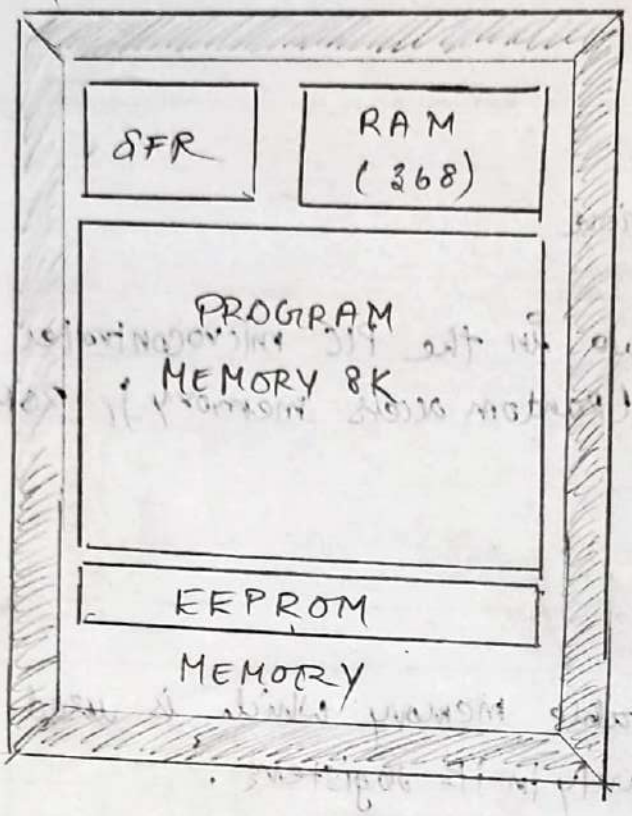
Special Function Registers

These registers are used for special purposes only as the name SFR implies. These registers will perform accordingly to the functions assigned to them and they can't be used as registers.

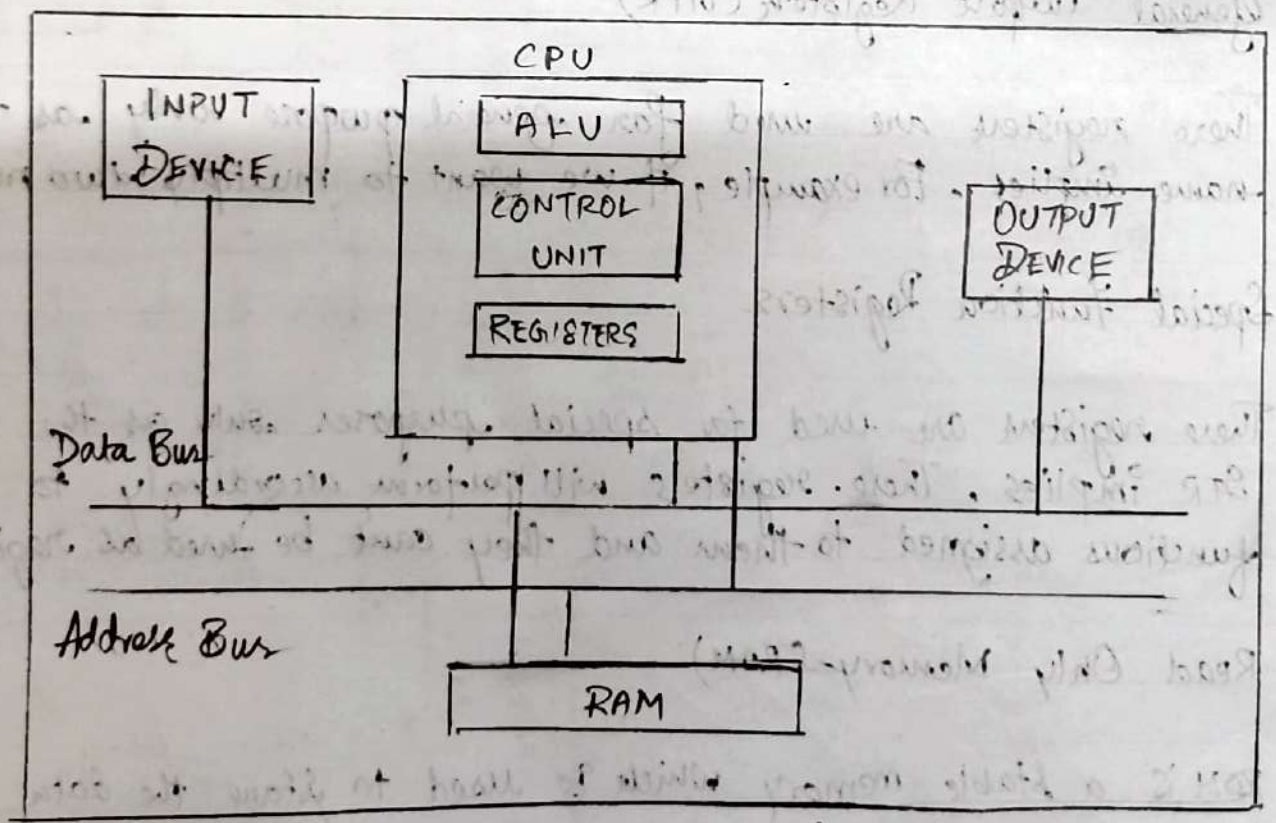
Read Only Memory (ROM)

ROM is a stable memory which is used to store the data permanently. In PIC microcontroller architecture, the ROM stores the instructions or program.

Memory Organization :-



BUS :



Electrically Erasable Programmable Read Only Memory (EEPROM)

In the normal ROM, we can write the program for only once, we can't use again the microcontroller for multiple times.

Flash Memory

Flash memory is also programmable read only memory (PROM) in which we can read, write and erase the program thousands of times.

Stack

When an interrupt occurs, the first PIC microcontroller has to execute the interrupt and the existing process address.

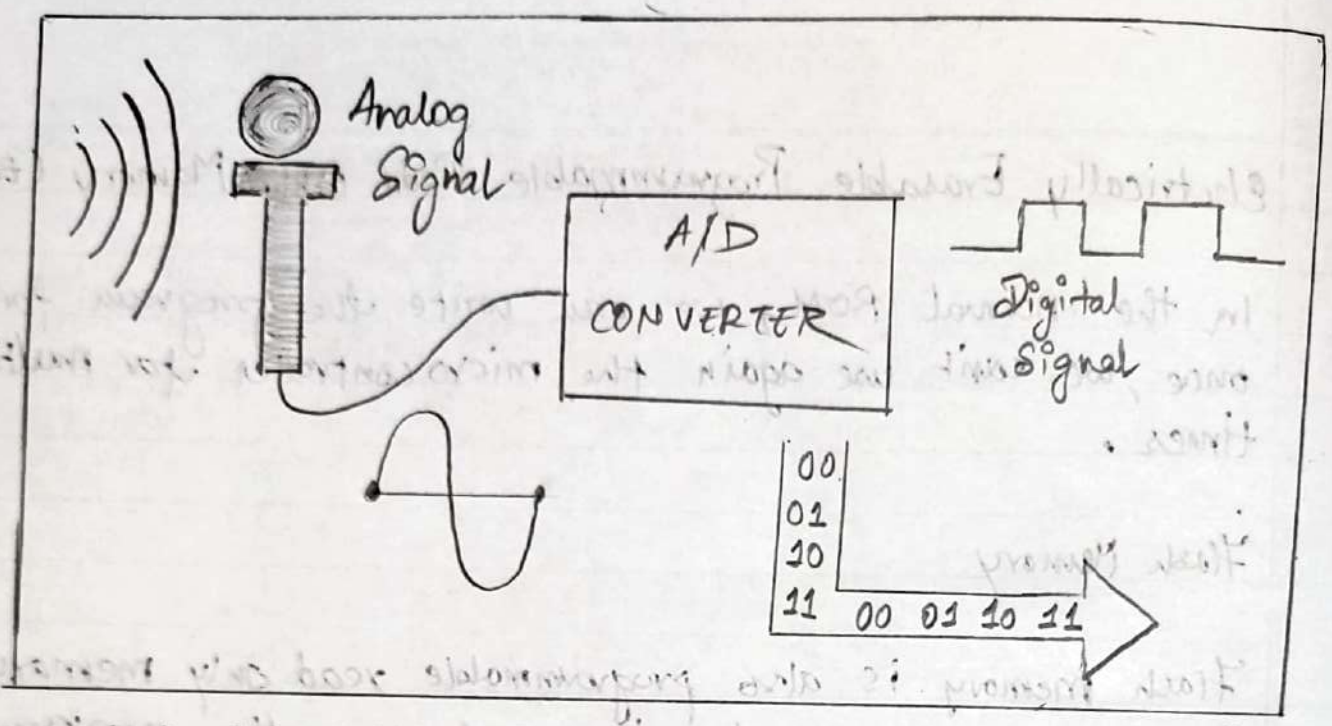
I/O Ports:

This series of PIC 16 consists of five ports A, B, C, D & E. Port A is 16 bit, B is 8 bit, C is 8 bit, D is 8 bit, E is 3 bit.

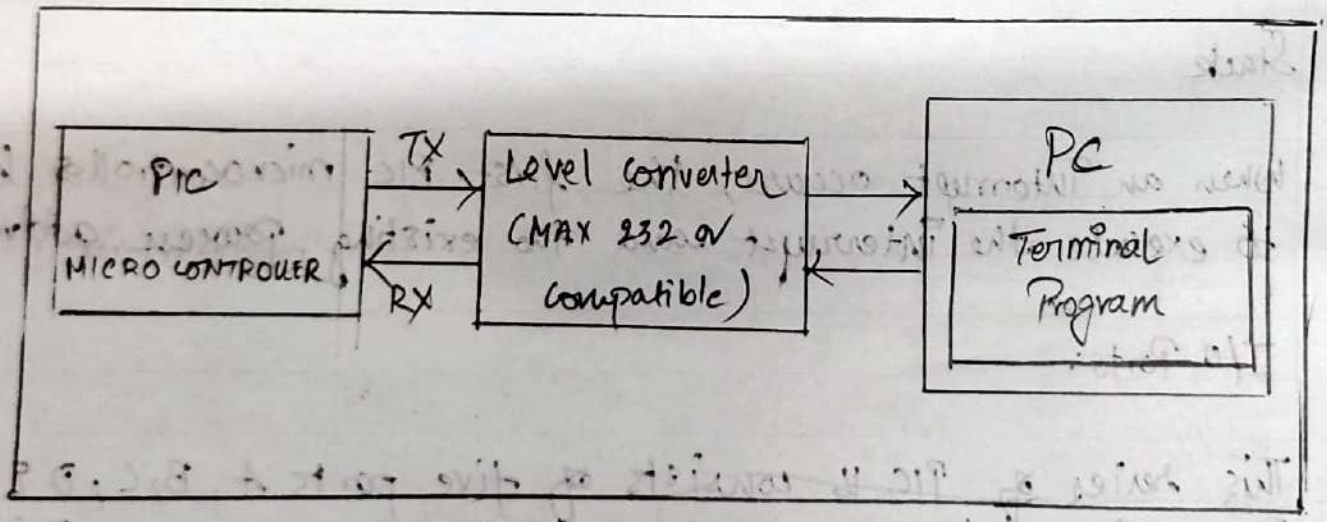
Bus

Bus is used to transfer and receive the data from one peripheral to another. It's classified into Data bus and Address bus.

A/D CONVERTER



Serial Communication



Advantages of PIC microcontroller:

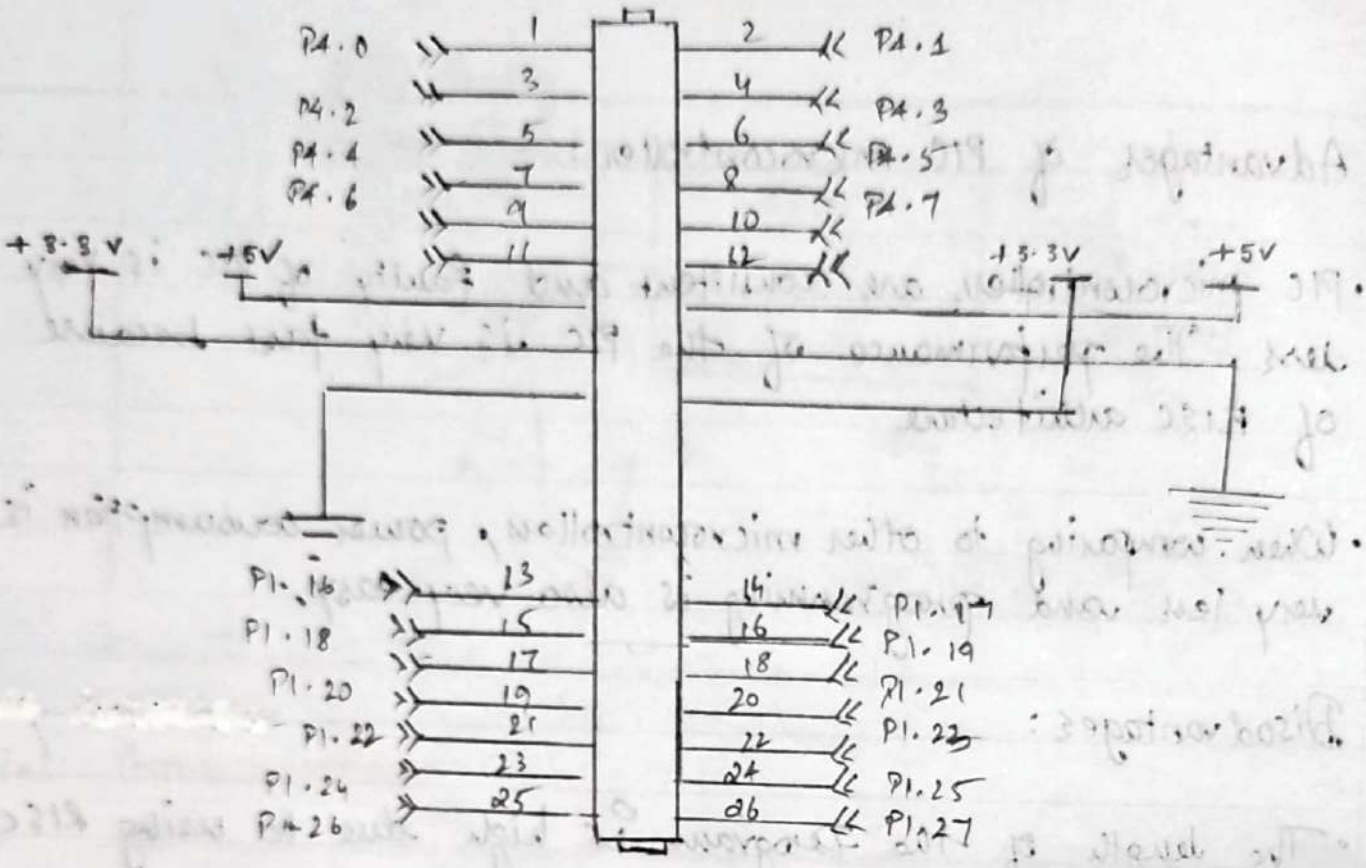
- PIC microcontrollers are consistent and faulty of PIC is very less. The performance of the PIC is very fast because of RISC architecture
- When comparing to other microcontrollers, power consumption is very less and programming is also very easy.

Disadvantages:

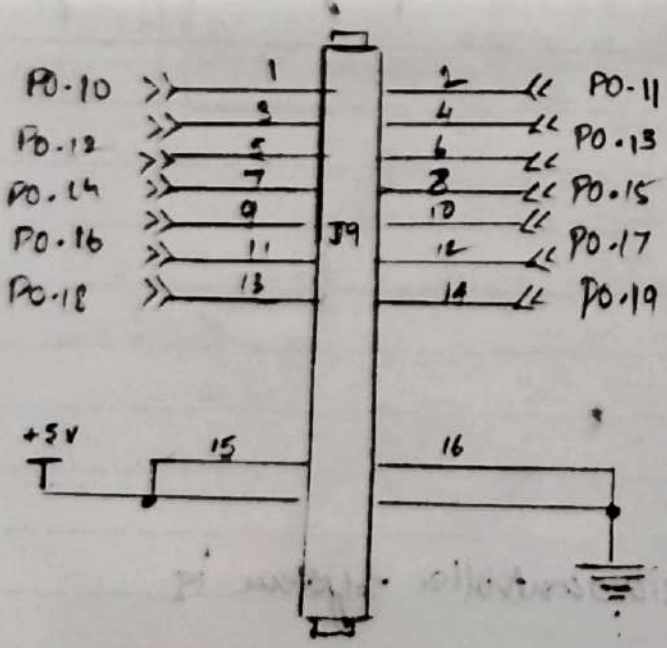
- The length of the program is high due to using RISC architecture (35 instructions)
- One single accumulator is present and program memory isn't accessible.

Result: Thus the study of PIC microcontroller system is done successfully.

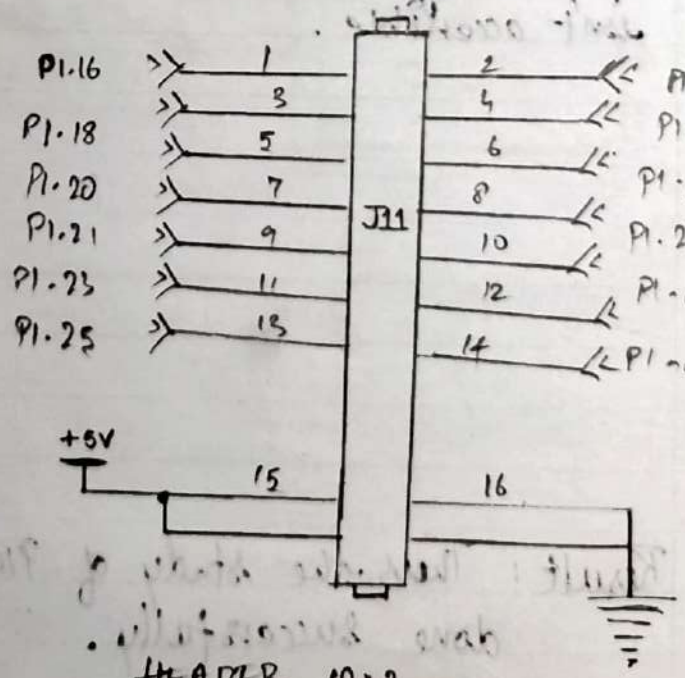
Extension Conn



GPIO Connectors



HEADER 10X2



HEADER 10x2

Ex no 2 Study of ARM microcontroller system

Aim: To study arm evaluation system lpc 4088.

Theory:

Arm Development Board:

The ARM cortex m4 development board has the following hardware features:

- 8 nos. Point leds (logic output)
- 8 nos. Digital input (slide switch)
- 2 nos. Analog input (potentiometer)
- 2x16 character lcd interface
- Temperature sensor (lm35)
- Internal rtc with battery - backup
- 1 no. Uart (rs232)
- 1 no. Usb uart
- Usb 2.0 device (virtual port)
- DAC output
- Interrupt study, reset button
- spi / i2c expansion connector
- sd card interface
- can interface (optional)

Jumper settings

Jumper	Description
Jp3	lcd enable : place a jumper
J6	Adc selection: place a jumper for the required channel. this will connect lpc408 pins with pot or lm35.
Jp7	i2c enable
Jp10	connects the buzzer with p0.26 or rtc alarm output pin
Jp11	Rts and dtr termination, jumper not needed
Jp13	Ft232 enable : powers the ft232 ic

Config dip switch (SW29)

Dip switch pin	Pin	Description
5v	1-step	Power supply for stepper motor
5v	2-lcd	Power supply for lcd
3v	3-eep	Power supply for i2c eeprom
Zigbee	4-zbe	Power supply for zigbee
Ft232	5 & 6	Connects ft232 with p0.0 & p0.1
Ft232	7 & 8	Connects ft232 with p0.2 & p0.3 (default)

Connectors:

40 pin frc box type connector: Instead of terminating each port separately, this connector has all the ports pins. So more 10 lines can be taken by using single cable. The ports are arranged as shown in the following figures. Similarly, two number of 20x2 connector gives access to various port lines.

Features of ARM microcontroller:

- 16-bit / 32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
- 8 kb to 40 kb of on-chip static RAM & 32 kb to 512 kb of on-chip flash memory; 128-bit wide interface / accelerator enables high-speed 60 MHz operation.
- Single power supply chip with POR and BOD circuits:

Brief overview of ARM7 Architecture

The ARM7 core is based on the von Neumann Architecture with 32 bit data bus that carries both instruction and data. Data can be of 8-bits, 16 bits, 32 bits. It has following features:

- Instruction pipeline
- Memory format
- Operation modes
- Coprocessor
- Debugging feature

Power supply to the peripherals

Peripheral	Switch	Pin	Description
Lpc4088	Sw1	-	Turn on the switch Sw1 towards usb ext
Ft232	Jp13	-	Place a jumper at jp13
Stepper	Sw29	1	Turn on the dip switch pin
Led	Sw29	2	Turn on the dip switch pin
I2C eeprom	Sw29	3	Turn on the dip switch pin
Zigbee	Sw29	4	Turn on the dip switch pin

Instruction pipeline:

The ARMv7 core uses a three stage pipeline to increase the flow of instructions to the processor. This allows multiple simultaneous operations to take place and continuous operations and memory systems. There instructions are executed in 3 stages:

- Fetch
- Decode
- Execute

Memory Formats:

ARM7 has 4 basic types of cycle:

- Internal
- Non sequential
- Sequential
- Coprocessor transfer

The ARM processor supports the following data types:

- Word, 32-bit
- Half word, 16-bit
- Byte, 8-bit

Operating modes:

- User mode - normal ARM program execution mode and used for executing most application programs.
- Fast Interrupt (FIQ) - mode supports data transfer or channel process to allow very fast interrupt
- Interrupt (IRQ) - used for general purpose interrupt handling.
- Supervisor (SVC) - Protected mode for Operating System
- Abort (ABT) - mode is entered after a data or instruction fetch is aborted.
- Undefined (UND) - mode is entered when an undefined instruction is executed.
- System (SYS) - is a priviledged user mode for the operating system.

Coprocessor:

Typical coprocessor contains:

- An Instruction pipeline
- Instruction decode logic
- Handshake logic
- A register bank
- Special processing logic with its own data path.

Creating a project in KEIL

Step 1: Open Keil and the environment will be as following

Step 2: Select project → new uvision project

Step 3: Create a project folder

Step 4: Name the project

Step 5: Select the processor and click ok

Step 6: Click 'yes' to the following message box

Step 7: Create a new source file by selecting file → new

Step 8: Write the required source code

Step 9: Save this file with ".c" extension

Step 10: Right click on source group and select all files group.

Step 11: Select the source code file, click add then click close

Step 12: Navigate to the folder "c:\keil\arm\Startup\nxp\lpc407x_8x_177x_8x" and copy the file "system_lpc407x_8x_8x.c"

Step 13: Paste it in project directory

Step 14: Right click on source group once again and select add files to group

Step 15: Now add this "system_lpc407x_8x_177x_8x.c" file to your project

Step 16: Build the project

Step 17: Right click "target 1" on the project tab and select options for target

Step 18: On the target tab, select the floating point hardware as "not used".

Step 19: Select output tab and put a check mark on "create hex file". The name in the text box "name of executable" will be used to name the hex file.

Step 20: Click rebuild

PROGRAMMING IN FLASH MAGIC

Hardware connection	Turn on dip switch sw2a pins 7 and 8 Supply power to the board using sw1 Press and hold sw4 (eint0) and press reset to enter into boot-loader mode

Programming in FLASH MAGIC

Five steps programming :—

Step 1 : Communications :
a) Click select ...
b) Expand arm cortex from device database
c) Scroll down and select your ic.
d) Click ok.

Step 2 — erase
put a check mark on erase blocks used by hex file checkbox

Step 3 : hex file
1. Click browse ...
2. Navigate to the project folder, select the hex file to be loaded and click open.

Step 4 : Options
1. Put a check mark on each options that is required for the project.
2. Generally verify : after Programming is required to cross check the burned hex file with loaded hex file and the remaining options left unchecked.

Step 5: Start

1. Click start. This will start programming the chip

2. See the status bar for current status

3. Press reset after programming

Result :- Thus the study of ARM microcontroller System is done successfully.

Ex. No: 1

FLASHING OF LEDS USING ARM PROCESSOR

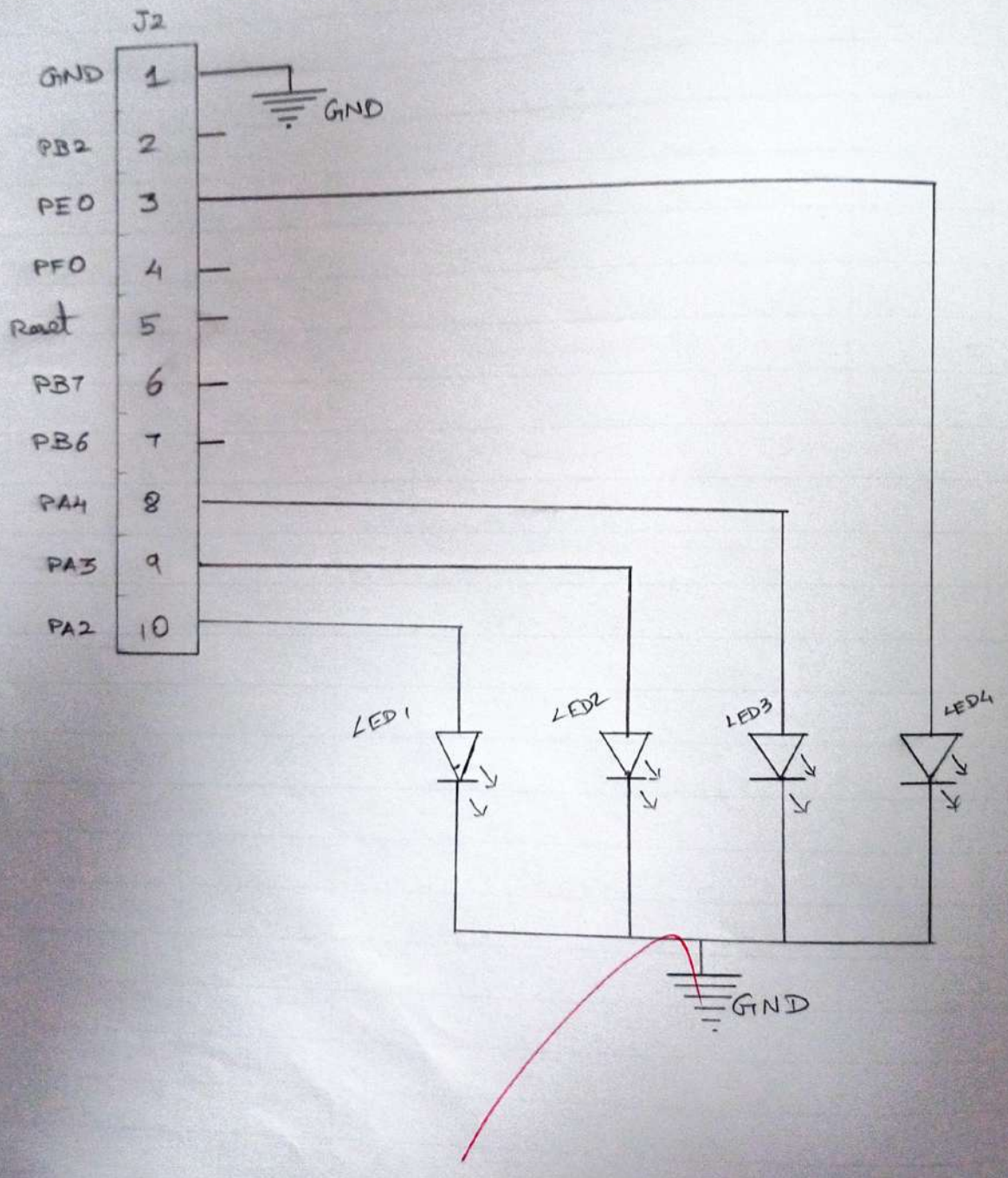
AIM:

To perform an experiment on Flashing of LED by interfacing with ARM processor.

PROCEDURE:

1. Open Keil Software in PC and create a project
2. Select the processor - ARM LPC2148 and click OK
3. Create a new source file
4. Write the required source code and save
5. Add files to source group
6. Build the project
7. Debug the project, and select peripheral GPIO show interface
8. Compile and run the project.
9. Create Hex file and rebuild the project.
10. Now, connect LPC2148 kit via USB port
11. Open FlashMagic to interface with LPC2148.
12. Using FlashMagic Download the hex file and start execution
13. Once executed successfully, check the output in the kit.

CIRCUIT DIAGRAM:



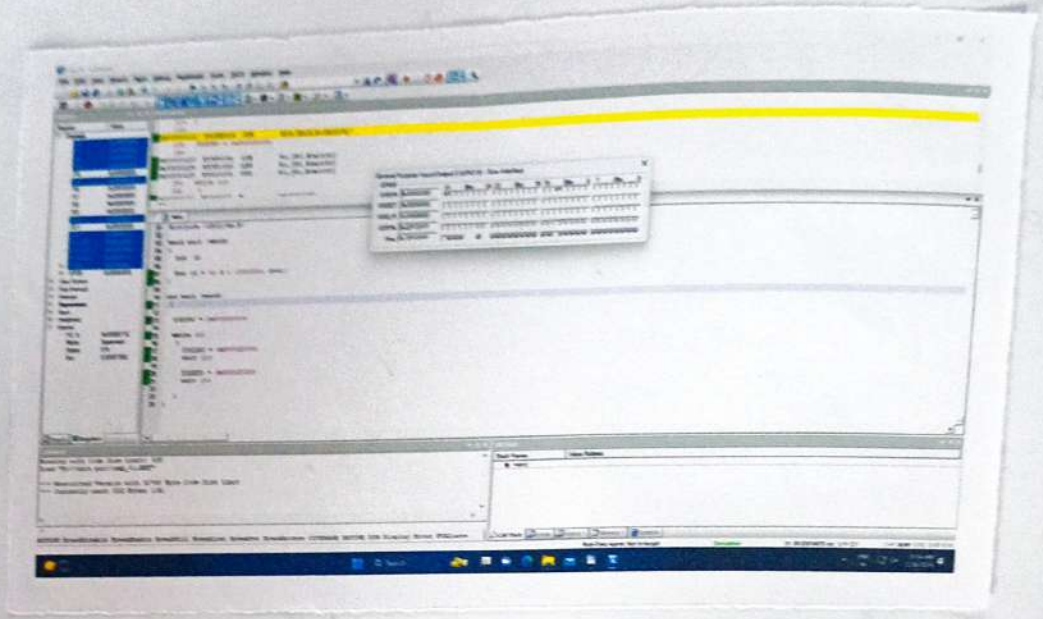
PROGRAM:

```
#include <LPC214x.H>
void wait(void)
{
    int d;
    for(d=0; d<1000000; d++);
}

int main(void)
{
    IODIR0 = 0x80002000;

    while(1) {
      IOCR0 = 0x80002000;
      wait();
      IOSET0 = 0x80002000;
      wait(); }
}
```

RESULT:

Thus the flashing of LED's using ARM processor is performed and verified.

OUTPUT:



Ex. No: 2. Interfacing Digital to Analog Convertor with ARM Processor

AIM:

To convert digital signal by interfacing with ARM processor.

COMPONENTS REQUIRED:

ARM Trainer Kit, Mini USB cable

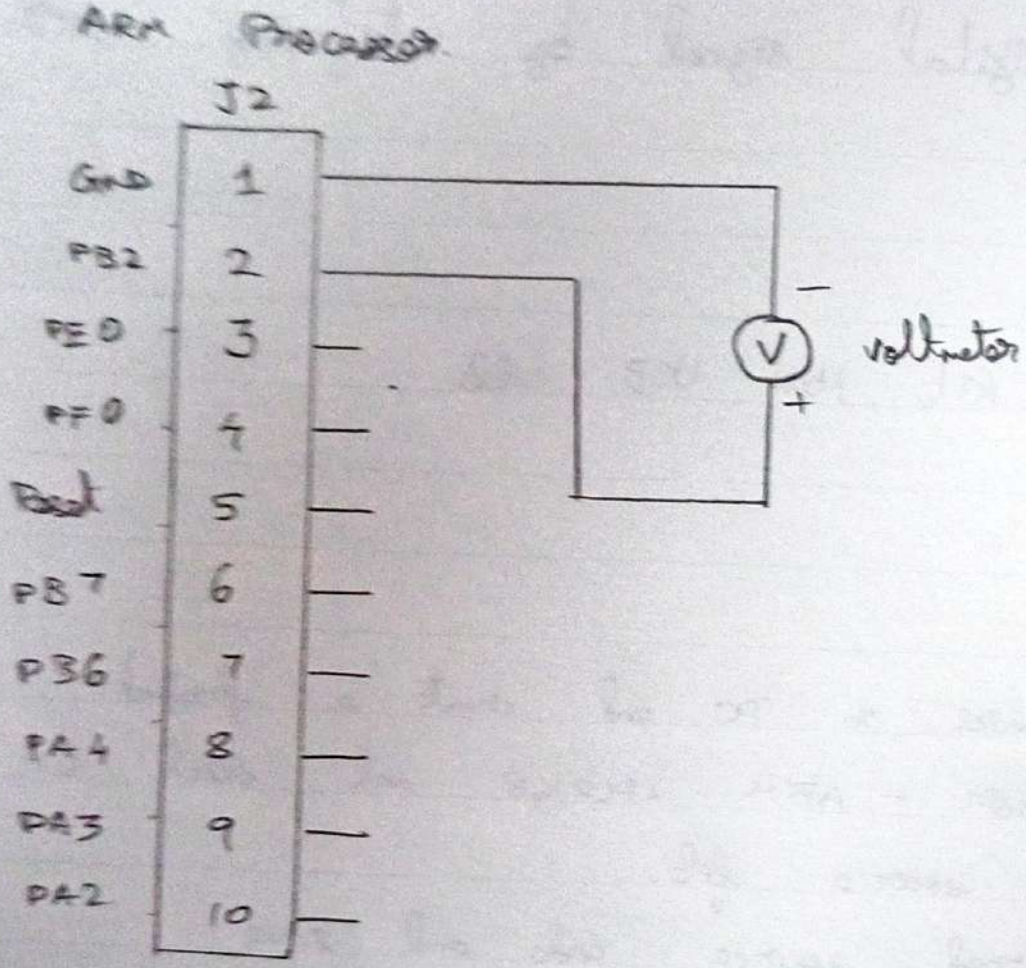
PROCEDURE:

1. Open Keil Software in PC and create a project.
2. Select the processor - ARM LPC2148 and click OK.
3. Create a new source file.
4. Write the required source code and save
5. Add files to source group
6. Build the project
7. Debug the project, and select peripheral GPIO Slow Interface
8. Compile and run the project
9. Create Hex file and rebuild the target
10. Now, Connect LPC2148 kit via USB port
11. Open Flash Magic to interface with LPC2148.
12. Using Flash Magic Download the hex file and start execution.
13. Once executed successfully, check for output in the kit.

PROGRAM:

```
#include <LPC214x.H>
#include "ADC_Driver.c"
#include "lcd.c"
#include <stdio.h>
void delay (int n) {
    int i,j;
    for (i=1; i<=n; i++)
    for (j=0; j<=10000; j++);
}
int main (void)
{
    char *temp = "0000";
    unsigned int adc_val;
    unsigned int temp;
    unsigned char buf[4] = {0,0,0,0};
    ADC_Init();
    lcd_init();
    clrscr(10);
    fprintf("LPC2148_ADC_TEST", 0,0);
        wait();

    while (1)
    {
        adc_val = ADC_ReadChannel();
        split_numbers(adc_val);
        goto xy(0, 1);
```

CIRCUIT DIAGRAM:



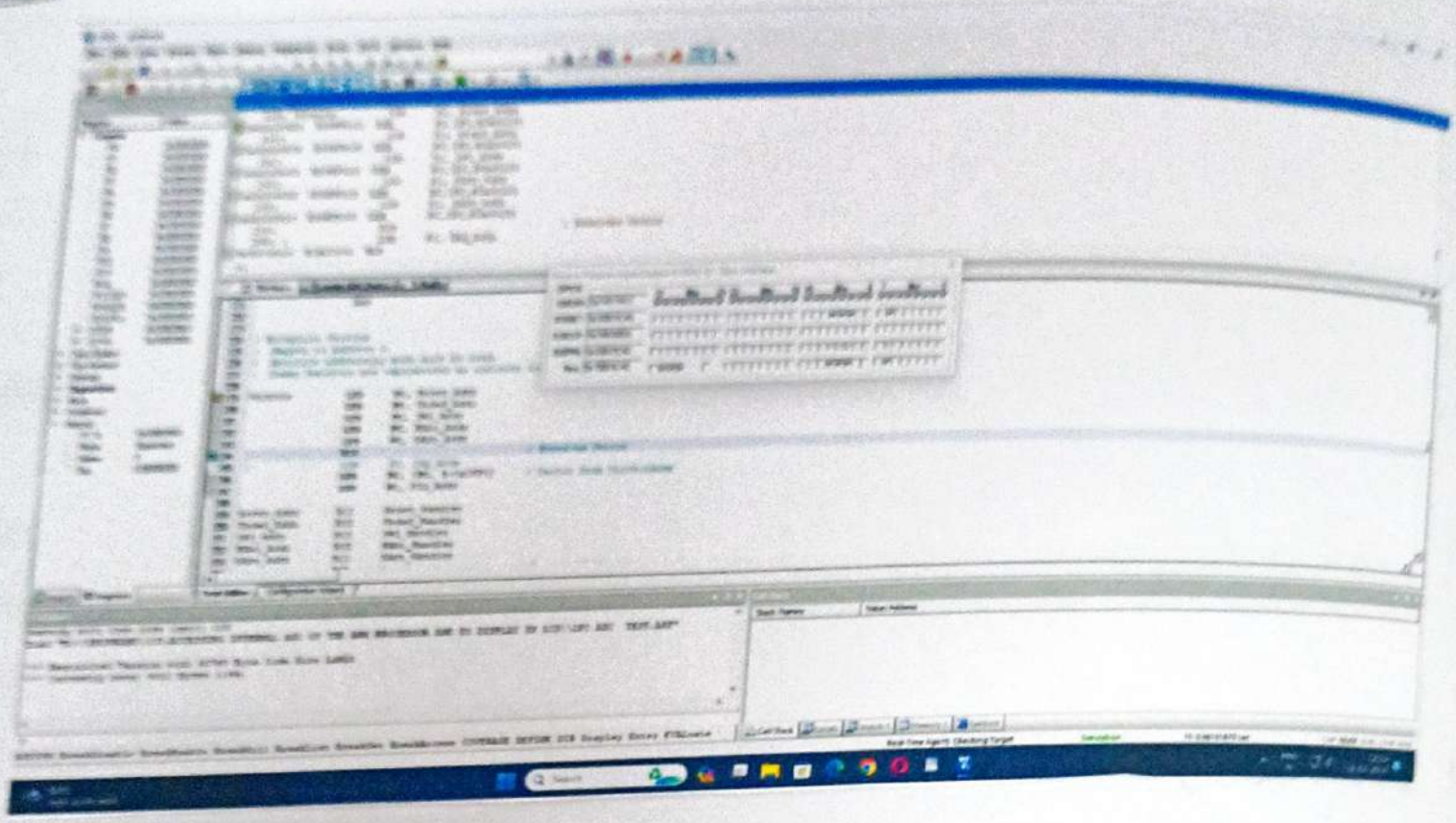
```
lcddat(thousands+0x30);
lcddat(hundreds+0x30);
lcddat(tens+0x30);
lcddat(ones+0x30);
}
```

3.

RESULT:

Thus the Conversion of Digital to Analog is performed and verified with ARM processor successfully.

OUTPUT :



Ex. No: 3. Interface LED and PWM and to verify the Output for the ARM7

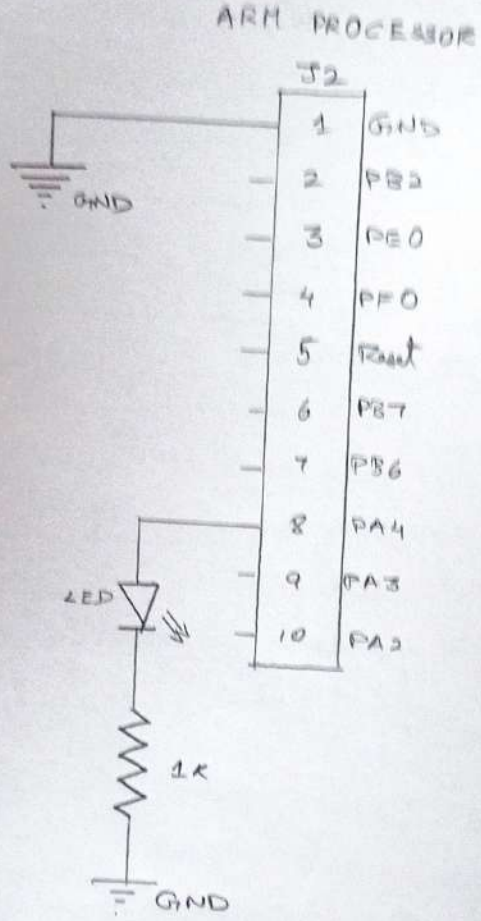
AIM:

To switch ON and OFF the LED by interfacing with the ARM processor.

PROCEDURE:

1. Open Keil Software in PC and create a project
2. Select the processor - ARM LPC2148 and click ok
3. Create a new source file
4. Write the required source code and save
5. Add files to source code and save
6. Build the project
7. Debug the project, and select Peripheral GPIO slow interface
8. Compile and run the project
9. Create Hex file and rebuild the target
10. Now, connect LPC2148 kit via USB port
11. Open FlashMagic to interface with LPC2148.
12. Using FlashMagic download the hex file and start execution
13. Once executed successfully, check the output in the kit.

CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
#define PLOCK 0x00000400
#define PWMPRESCALE 60

void initPWM(void);
void initClocks(void);
void setupPLL0(void);
void feedseq(void);
void ConnectPLL0(void);

int main(void) {
    initClocks();
    initPWM();
    while(1){ }
}

void initPWM(void) {

PINSEL1 |= 0x00000400;
PWMPR = 60-1;
PWMPCR = 0x00002000;
PWMMCR = (1<<1);
PWMMR0 = 10000;
PWMMR5 = 2500;
PWMLER = 0x00000021;
PWMTCR = 0x00000002;
```

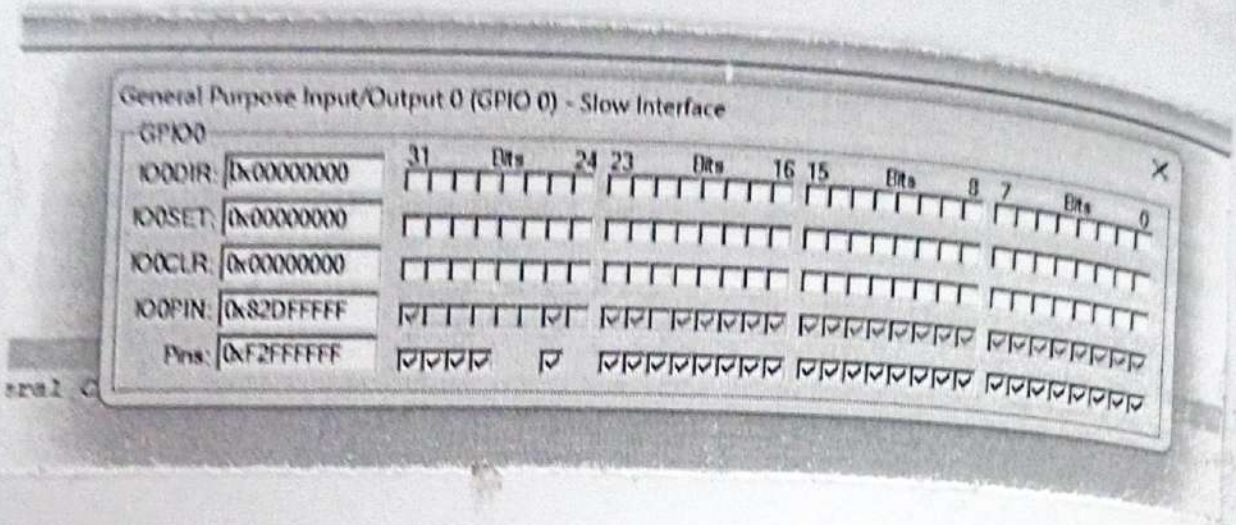
```
PWMTCR = 0x00000009;
}

void initClocks(void) {

setupPLL0();
feedSeq();
connectPLL0();
feedSeq();
VPBDIV = 0x01;
}

void setupPLL0(void) {

PLL0CON = 0x01;
PLL0CFG = 0x24;
}

void feedSeq(void) {

PLL0FEED = 0xAA
PLL0FEED = 0x55;
}
```

```
void connectPLL0(void) {

while (!(PLL0STAT & PLOCK));
PLL0CON = 0x03;
}
```

RESULT:

Thus the interfacing of LED with ARM processor is done successfully.

OUTPUT:


General Purpose Input/Output 0 (GPIO 0) - Slow Interface
GPIO0
IO0DIR: 0x00000000
IO0SET: 0x00000000
IO0CLR: 0x00000000
IO0PIN: 0x82DFFFFF
Pins: 0xF2FFFFFF
31 Bits 24
23 Bits 16
15 Bits 8
7 Bits 0

Ex No: 6 Interfacing Real Time Clock with ARM Processor

AIM:

To perform the experiment to study the use the internal RTC module by interfacing with ARM processor.

PROCEDURE:

1. Open Keil Software in PC and Create a project.
2. Select the processor - ARM LPC2148 and click ok.
3. Create a new source file
4. Write the required source code and save
5. Add files to source group.
6. Build the project
7. Debug the project, and select peripheral GPIO Slow interface
8. Compile and run the project
9. Create Hex file and rebuild the target
10. Now, connect LPC2148 kit via USB port
11. Open FlashMagic to interface with LPC2148
12. Using FlashMagic download the hex file and start execution
13. Once executed successfully, check for output in the kit

PROGRAM:

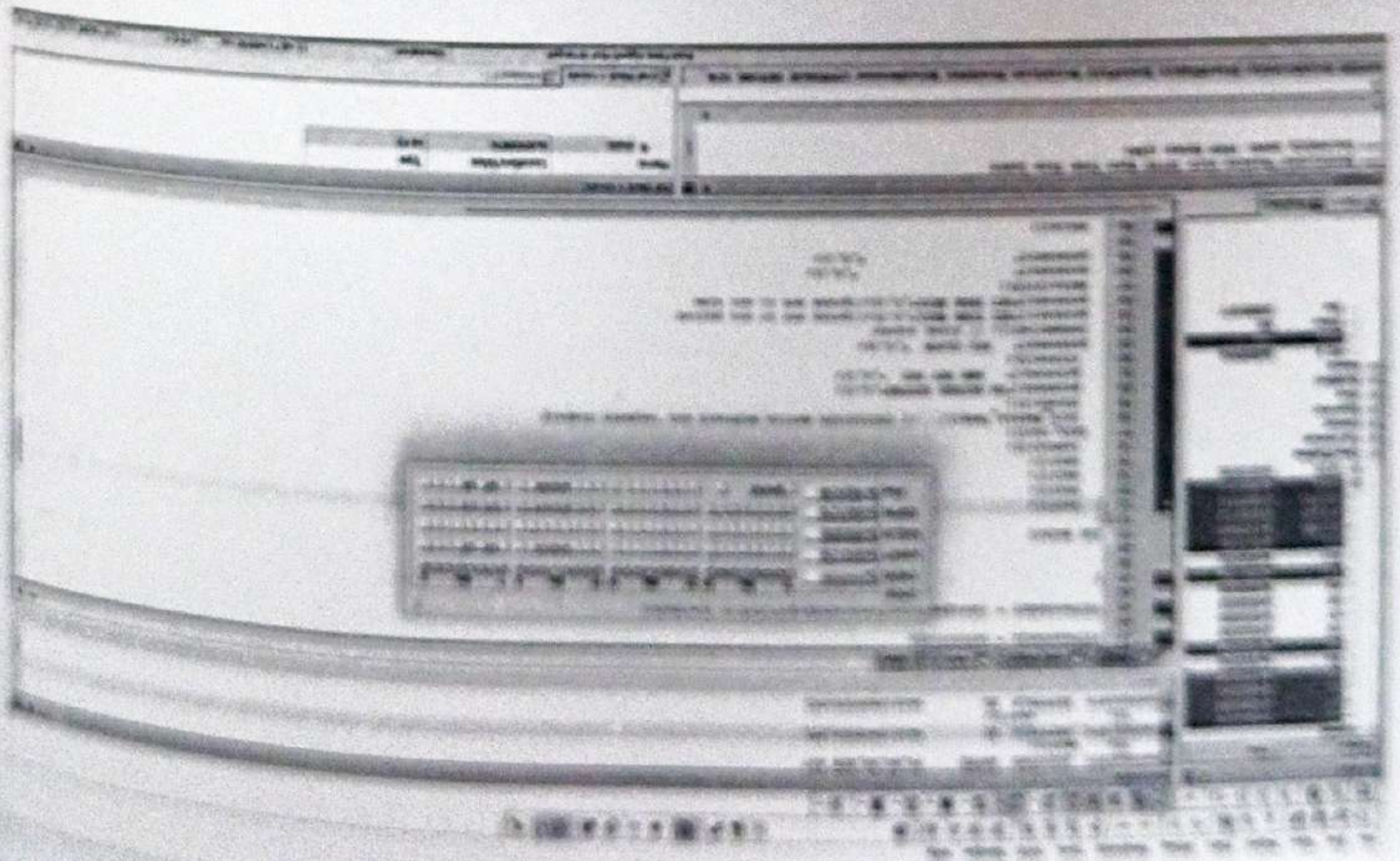
```
#include <LPC214x.h>
#include "lcd.h"
#include "mat_prog.h"

unsigned int mnt, dt, dtyr;
unsigned int hrs_f, min_f, sec_f, i, j;
unsigned int beg;
char En_alarm_enter, alarm_enter_mode, char_count;
char hr[2];
char mine[2];
char secs[2];
void set_dt (void);
void delay (unsigned int x) {
        int i;
        while (x--) {
            for (i=0; i<=2000; i++);
        }
}
unsigned char flag=0;
void rtc_int (void) __irq {
ILR = 0x01;
flag = 1;
VICVectAddr = 0x00000000;
}
```

```
void init_rtc()
{
ILR = 0x01;
CCR = 0x13;
CCR = 0x11;
CIIR = 0x01;
VICIntEnable = 0x00002000;
VICVectCntl0 = 0x0000002D;
VICVectAddr0 = (unsigned) rtc_int;
}

int main()
{
    wait();
    lcdInit();
    init_rtc();
    init_Matrix_Keyboard();
    clrscr(2);
    printstr("SM Micro System", 0,0);
    printstr("RTC CLOCK", 0, 1);
    lcdcmd(0x01);
    set_time_delay(0, 1);
    delay(2000);
    while(1){
        dt = DOM;
        mnt = MONTH;
        dtyr = DAY;
        hrs_fr = HOUR;
        min_fr = MIN;
        sec_fr = SEC;
```

```
key = catch_key();
if (key != 0) {
    if (key == 13) {
        clrscr();
        printstr("SET TIME", 0, 1);
        delay(10);
        set_dt();
        DOM = (hrs[0]*10) + hrs[1];
        MONTH = (mins[0]*10) + mins[1];
        DAY = (secs[0]*10) + secs[1];
        hrs[0] = hrs[1] = mins[0] = mins[1] = sec[0] = secs[1] = 0;
    }
}
gotoxy(0,0);
printstr("DATE:", 0, 0);
split_numbers(dt);
lcddat(temp + 0x30);
lcddat("TIME=", 0, 1);
}
}
```

RESULT:

Thus the interfacing of Real Time Clock with ARM processor is performed.



Ex 4

INTERFACING KEYBOARD AND LCD

Aim: To perform experiment on 4x4 matrix Keypad and display its result in LCD by interfacing with ARM processor.

Components Required:

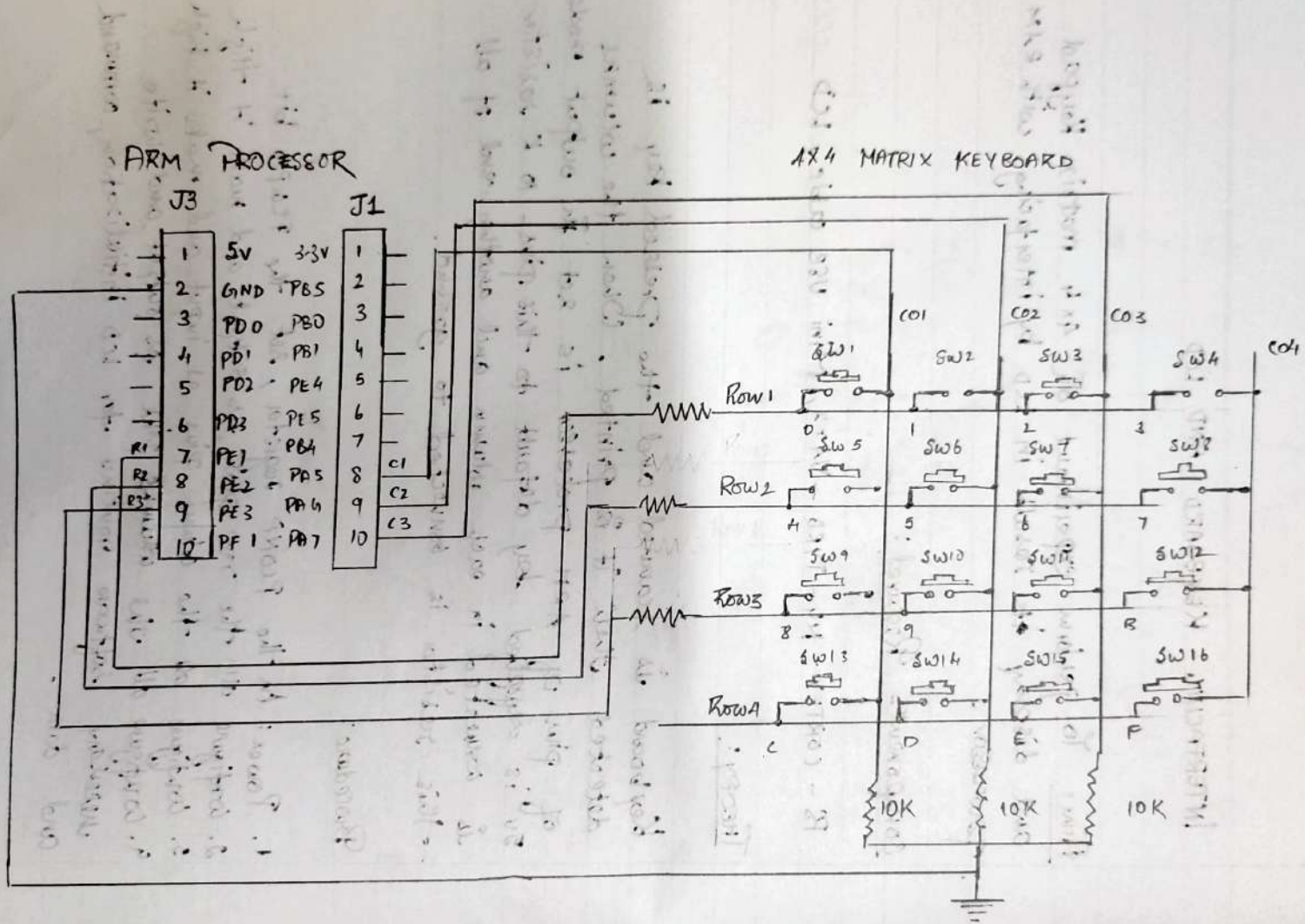
FS - CORTEX - M4 - TYRO - V4+2, Mini USB cable, LCD

THEORY:

Keyboard is scanned and the pressed key is detected then it is printed. When the columns of pins of ARM processor is set in output mode, 5V is supplied by default to this pins. 10 K resistor is connected in each column and another end of all this resistor is connected to ground.

Procedure:

1. Power: In the PCONP register, set the PCGPIO bit
2. Configure all the row pins as output and make it high
3. Configure all the column pins as input and make it high
4. Configure all the LCD pins as output and write necessary software routines for LCD Initialization, command and data.


ARM PROCESSOR
4X4 MATRIX KEYBOARD
J3
J1
5V
3.3V
GND
PD0
PD1
PD2
PD3
PE1
PE2
PE3
PF1
PB5
PB0
PB1
PE4
PE5
PB4
PA5
PA6
PA7
R1
R2
R3
C1
C2
C3
Row1
Row2
Row3
Row4
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
SW13
SW14
SW15
SW16
10K
10K
10K

5. Make a row pin zero and Check all four column lines for zero.
6. If any key is pressed, the corresponding column will get zero.
7. Read the Key value and display it in LCD.

- Open Keil Software in PC and create a project.
- Select the processor - ARM LPC2148 & click ok
- Create a new source file
- Write the required source code & save.
- Add files to source group
- Build the project
- Debug the project, and select - peripheral GPIO & low interface
- Compile and run the project
- Create Hex file and rebuild the target
- Now, Connect LPC2148 Kit via USB port
- Open flash magic to interface with LPC2148.
- Using flash magic download the hex file and start the execution

Program :

Program :

Result : Thus the Keyboard interfacing with ARM processor is done and pressed key is verified successfully.

Ex 8 Interrupt Performance characteristics of ARM & FPGA

Aim: To compare the performance of FGPA and CORTEX-M4 using interrupts.

COMPONENTS REQUIRED:

ARM Trainer kit, 2 nos. of Mini USB cable, one FPGA stick board

THEORY

Two interrupt ports are available in ARM processor. Two interrupt ports are PUSH1, PUSH2. PUSH1 & linked with PF4 & PUSH2 is linked with PF0, so external switch is to be connected with PF4 to trigger the PUSH1 interrupt, & PF0 for PUSH2. SW1 & SW2 switch are present in the ARM target board, SW1 can be used instead of connecting external switch to PF4 to trigger PUSH1 interrupt.

Procedure:

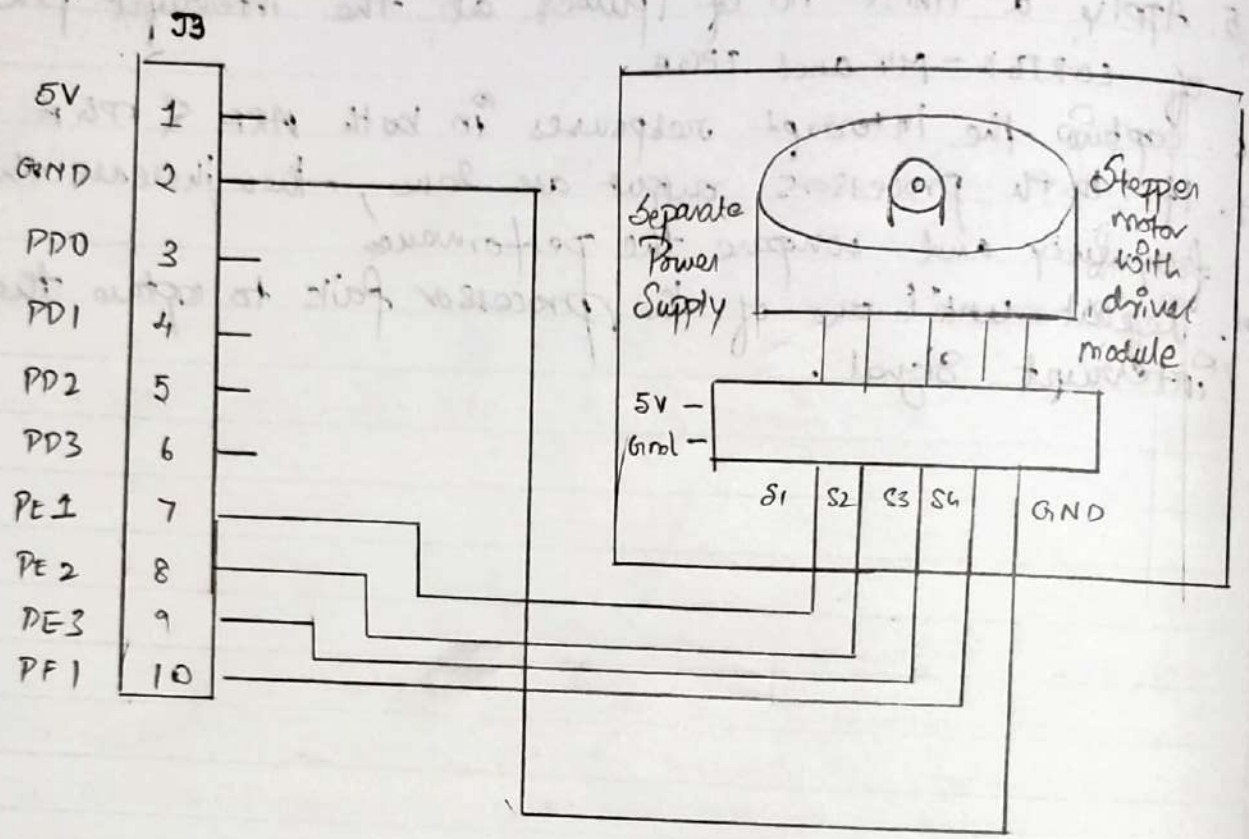
1. Configure the External Interrupt of CORTEX-4.
2. Configure pins P4.0 to P4.7 as GPIO pins using IOCON registers IOCON-P4-0 to IOCON-P4-7
3. Configure the pins P4.0 to P4.7 as output pins using their direction registers
4. Configure Interrupt in FPGA.

5. Apply a Finite no of pulses at the interrupt pins of CORTEX-M4 and FPGA
6. Capture the interrupt responses in both ARM & FPGA
7. If both processors output are same, then increase the frequency and compare the performance
8. Repeat until one of the processor fails to capture the interrupt signal.

Result: Thus the implementation of interrupt performance of ARM and FPGA is performed.

CIRCUIT DIAGRAM:

ARM PROCESSOR



Ex 9

CIR

Ex 9 Interfacing Stepper Motor with ARM processor

Aim: To run a Stepper motor by interfacing with ARM processor.

COMPONENTS REQUIRED:

1. Power: In the PCONP register, Set the PCGPIO bit
2. Configure pins P3.23 to P3.26 as GPIO pins using IOCON register IOCON_P3_23 to IOCON_P3_26
3. Configure the pins P3.23 to P3.24 as output pins using their direction registers.
4. Send the Stepper sequence via IO lines
   - Open Keil Software in PC and create a project.
   - Select the processor - ARM LPC2148 and click OK
   - Create a new source file
   - Write the required source code & Save.
   - Add files to Source group
   - Build the project
   - Debug the project, and select peripheral GPIO Slow Interface
   - Compile and run the project
   - Create Hex file and rebuild the target.
   - Now, Connect LPC2148 kit via USB port
   - Open Flash Magic to Interface with LPC 2148
   - Using Flash Magic Download the Hex file and start execution.
   - Once executed successfully, Check for output in the Kit.

Theory:

A stepper motor is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation. Every revolution for the stepper motor is divided into a discrete number of steps, and the motor must be sent a separate pulse for each step. Sequence 1100, 0110, 0011, 1001 sent to stepper from arm processor to rotate in forward direction, sequence is reversed to rotate motor in reverse direction.

Result: Thus the stepper motor speed control is performed by interfacing stepper with ARM processor.

Ex 10

TURN AN LED ON & OFF WITH ARDUINO

Aim: To turn an LED light on & off with Arduino UNO

Procedure:

1. Connect LED in the breadboard
2. Connect GND to GND
3. Connect Digital PIN 7 to Positive Connection
4. Now type the code in ARDUINO IDE & verify it.
5. Connect the USB of ARDUINO to the system.
6. Now upload the code in ARDUINO to the System.
7. Check whether the LED is blinking or not.

ARDUINO INSTALLATION STEPS:

1. First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

2. Download Arduino IDE Software

3. Power up your board.

4. Launch Arduino IDE.

5. Open your first project

6. Select your Arduino board

7. Select your serial port

8. Upload the program to your board.

CODING:

```
int led1 = 7; void setup (){
 // pinMode (pin, mode);
 pinMode (7, OUTPUT); }
void loop()
{
  digitalWrite (led1, HIGH);
  delay (200);
  digitalWrite (led1, LOW);
  delay (200);
}
```

Result: Thus Turn on & Off LED light with Arduino is executed and performed Successfully.

Expno. 11 Read a Switch, print the state out to the ARDUINO SERIAL MONITOR

Aim: To read a Switch, print the state out to the Arduino serial monitor

Procedure:

1. Take a arduino uno board, breadboard, button, jumper wires (male to male) and pointer wire (to connect arduino board to PC).

2. Connect the circuit as shown in below figures.

3. Write a program in Arduino uno desktop editor IDE. By opening new sketch.

4. Save file, compile it & upload.

CODING:

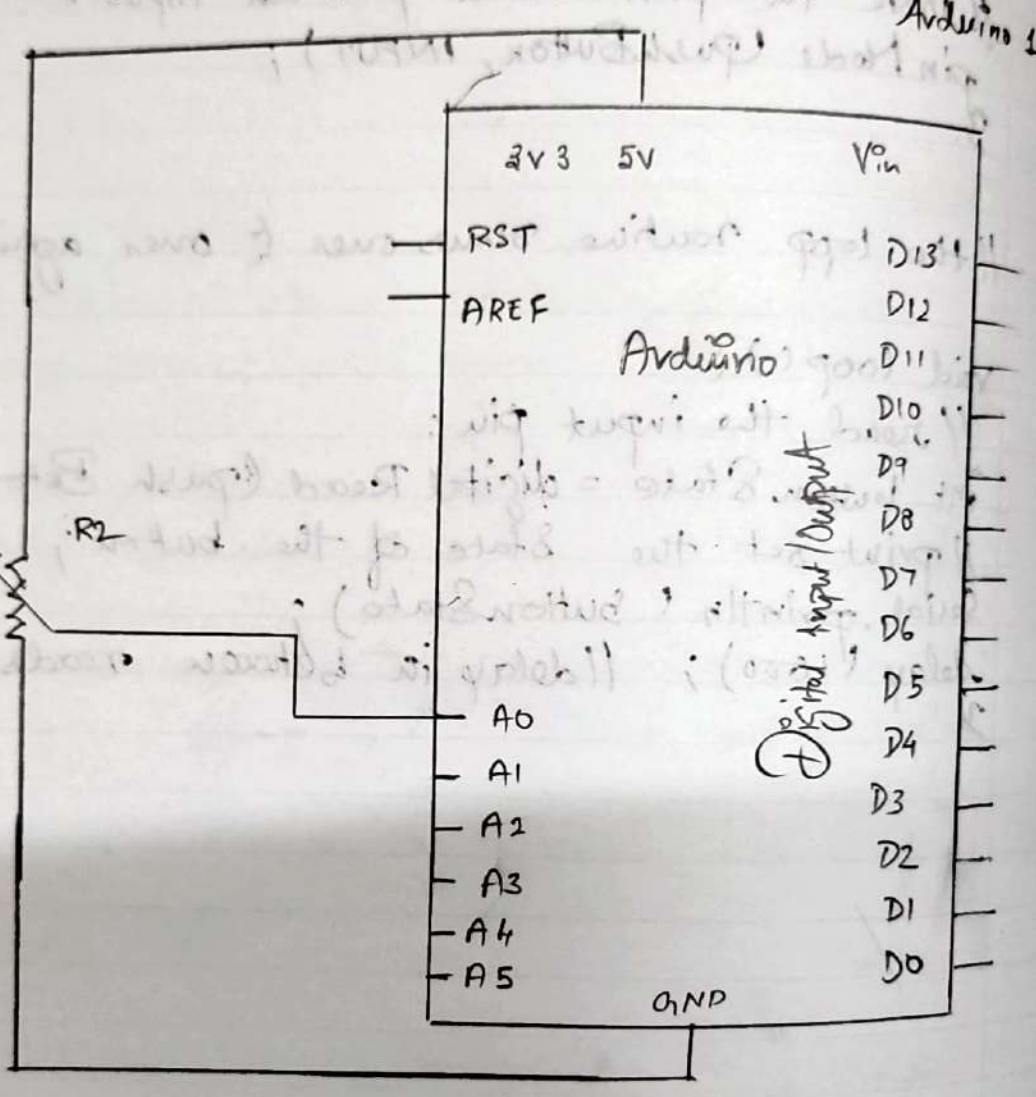
```
//digital pin 2 has a pushbutton attached to it. Give it a name:
  int pushButton = 2;
//the setup routine runs once when you press reset:
  void setup() {
```

```
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
// make the pushbutton's pin an input:
pinMode(pushButton, INPUT);
}

// the loop routine runs over & over again forever:

void loop() {
  // read the input pin:
int buttonState = digitalRead(pushButton);
// print out the state of the button;
Serial.println(buttonState);
delay(1000); // delay in between reads for stability
}
```

Result: To read a switch, print the state out to the Arduino serial monitor if compiled & executed successfully.

Ex 12

CIRCUIT DIAGRAM:



Ex 18 Read an analog input and print the voltage to the Arduino Serial monitor

Aim: To read an analog input and print the voltage to the Arduino Serial monitor.

Procedure:

1. Apparatus is arduino uno board, jumper wires (male to male), printer wire to connect board to PC and 10K potentiometer.
2. Connect centre pin of the potentiometer to A0.
3. Connect outside pin of the potentiometer to +5V
4. Connect GND to GND
5. Upload the code in Arduino board using Arduino IDE.
6. Open the Serial monitor for the output.

Program:

```
// Analog Read with Serial Monitor

void setup() {

// the setup routine runs once when you press reset
Serial.begin(9600); // initialize serial communication at
9600 bits per second }

void loop() {

// the loop routine runs one & one again forever
int sensorValue = analogRead(A0); // read the input on
analog pin 0
Serial.println(sensorValue); // print out the value you
read delay(10); // delay in between reads for
stability }
```

Result: To read an analog input & print the voltage to the Arduino serial monitor has been executed & performed successfully.

Ex no. 13 Detect Knock with a piezo element Sensor

Aim: To detect knock with a piezo element using Arduino uno.

Procedure:

1. Apparatus is arduino uno board, jumper wires (male to male), printer wire to connect board to PC and Piezo electric sensor.

2. Connect the circuit, by seeing the schematic diagram.

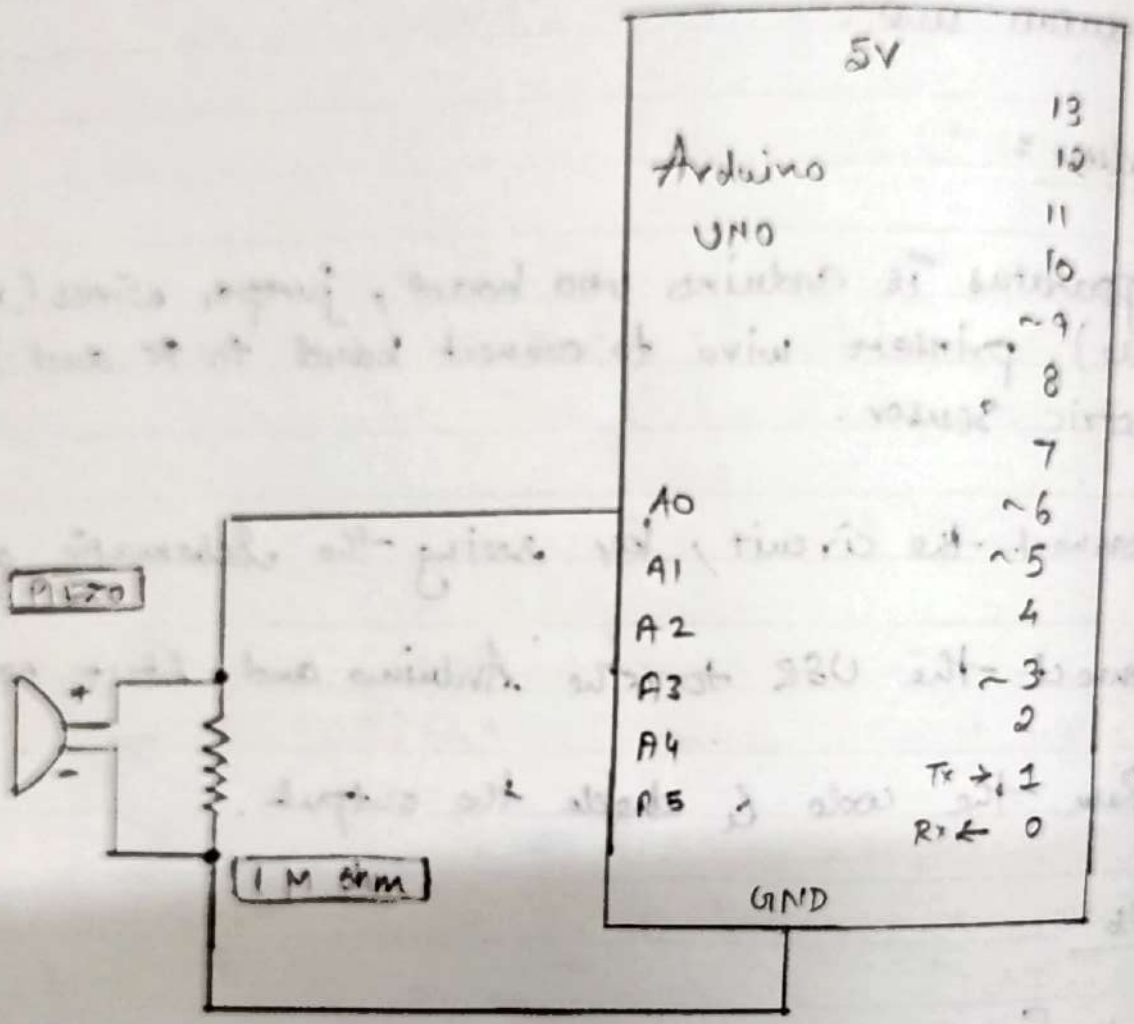
3. Connect the USB to the Arduino and start compiling.

4. Run the code & check the output.

CODE:

```
/*
Knock Sensor
This sketch reads a piezo element to detect a
knocking sound.
It reads an analog pin & compares the result to a
set-threshold.
If the result is greater than the threshold, it writes
"knock" to the serial port, & toggles the LED on pin 13.
```

CIRCUIT:


5V
Arduino
UNO
13
12
11
10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
A0
A1
A2
A3
A4
A5
GND
PIEZO
+
-
1 M ohm

The circuit:

- positive connection of the piezo attached to analog in 0
- negative connection of the piezo attached to ground
- 1 megaohm resistor attached from analog in 0 to ground.

// these constants won't change:

```
const int ledPin = 13; // LED connected to digital pin 13
const int knockSensor = A0; // the piezo is connected to
analog pin 0 const int threshold = 100;
// threshold value to decide when the detected sound is
a knock or not
// these variables will change:
int sensorReading = 0; // variable to store the value read
from the sensor pin
int ledState = LOW; // variable used to store the last LED
status, to toggle the light

void setup() {
  pinMode(ledPin, OUTPUT); // declare the ledPin as
OUTPUT Serial.begin(9600);
}
```

```
void loop() {

// read the sensor and store it in the variable sensor
   Reading: sensorReading = analogRead (knock Sensor);

// if the sensor reading is greater than the threshold
: if (sensorReading = analogRead (knock Sensor);
// toggle the status of the ledPin:

ledState = ! led State;

// update the LED pin itself:
  digital Write (ledPin, led State);
// send the string "knock!" back to the computer,
   followed by newline Serial.println ("Knock!");
}

delay (100); // delay to avoid overloading the serial port
buffer

}
```

Result: To detect knocks with a piezo element sensor using Arduino uno is performed & executed successfully.