LABORATARY MANUAL

EMBEDDED LAB

THE NEOTIA UNIVERSITY

ज्ञानम् आत्म प्रदीपाय

DEPARTMENT

OF

ROBOTICS & AUTOMATION ENGINEERING

EMBEDDED LABORATORY

LIST OF EXPERIMENTS

1. Study of ARM evaluation system………………………………………… 1

2. LED Interfacing……………………………………………………….. 5

3. Flashing of LEDs…………………………………………………… 7

4. ADC Interfacing…………………………………………………… 9

5. DAC Interfacing…………………………………………………… 11

6. DC Motor Interfacing……………………………………………….13

7. Interrupt………………………………………………………….. 15

8. Keyboard Interfacing……………………………………………….. 18

9. LCD Interfacing…………………………………………………… 21

10. Temperature Sensor Interfacing……………………………………….. 23

11. Stepper Motor Interfacing………………………………………… 25

12. EPROM Interfacing ………………………………………………… 27

13. Real Time Clock Interfacing………………………………………... 30

SYLLABUS

EMBEDDED LABORATORY **L T P C 0 0 3 2**

OBJECTIVES:
The student should be made to:

- Learn the working of ARM processor
- Understand the Building Blocks of Embedded Systems
- Learn the concept of memory map and memory interface
- Know the characteristics of Real Time Systems
- Write programs to interface memory, I/Os with processor
- Study the interrupt performance

LIST OF EXPERIMENTS

1. Study of ARM evaluation system
2. Interfacing ADC and DAC.
3. Interfacing LED and PWM.
4. Interfacing real time clock and serial port.
5. Interfacing keyboard and LCD.
6. Interfacing EPROM and interrupt.
7. Mailbox.
8. Interrupt performance characteristics of ARM and FPGA.
9. Flashing of LEDS.
10. Interfacing stepper motor and temperature sensor.
11. Implementing zigbee protocol with ARM.

OUTCOMES:
At the end of the course, the student should be able to:

- Write programs in ARM for a specific Application
- Interface memory and Write programs related to memory operations
- Interface A/D and D/A convertors with ARM system
- Analyse the performance of interrupt
- Write programmes for interfacing keyboard, display, motor and sensor.
- Formulate a mini project using embedded system

LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS (3 students per batch)
1. Embedded trainer kits with ARM board 10 No.s
2. Embedded trainer kits suitable for wireless communication 10 No.s
3. Adequate quantities of Hardware, software and consumables

1. STUDY OF ARM EVALUATION SYSTEM

AIM :

To study the ARM Evaluation processor TM4C123GXL.

TM4C123GXL LanchPad :

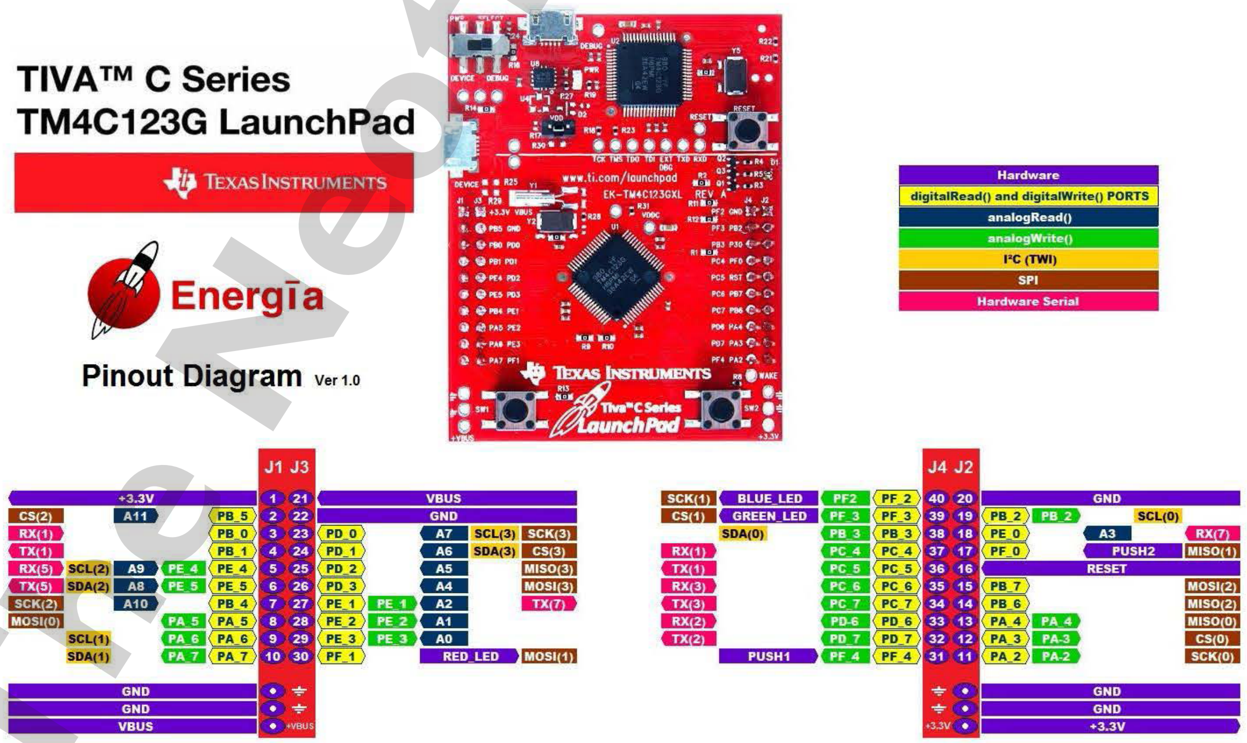
TM4C123GXL is the 32-bit ARM Cortex – M4 80 MHz CPU, 256 kB Flash / 32 kB EEPROM, 12-bit SAR adc, Comparators, Timers, DMA, I2C, UART and Integrated Full & Low speed USB 2.0.

SOFTWARE:

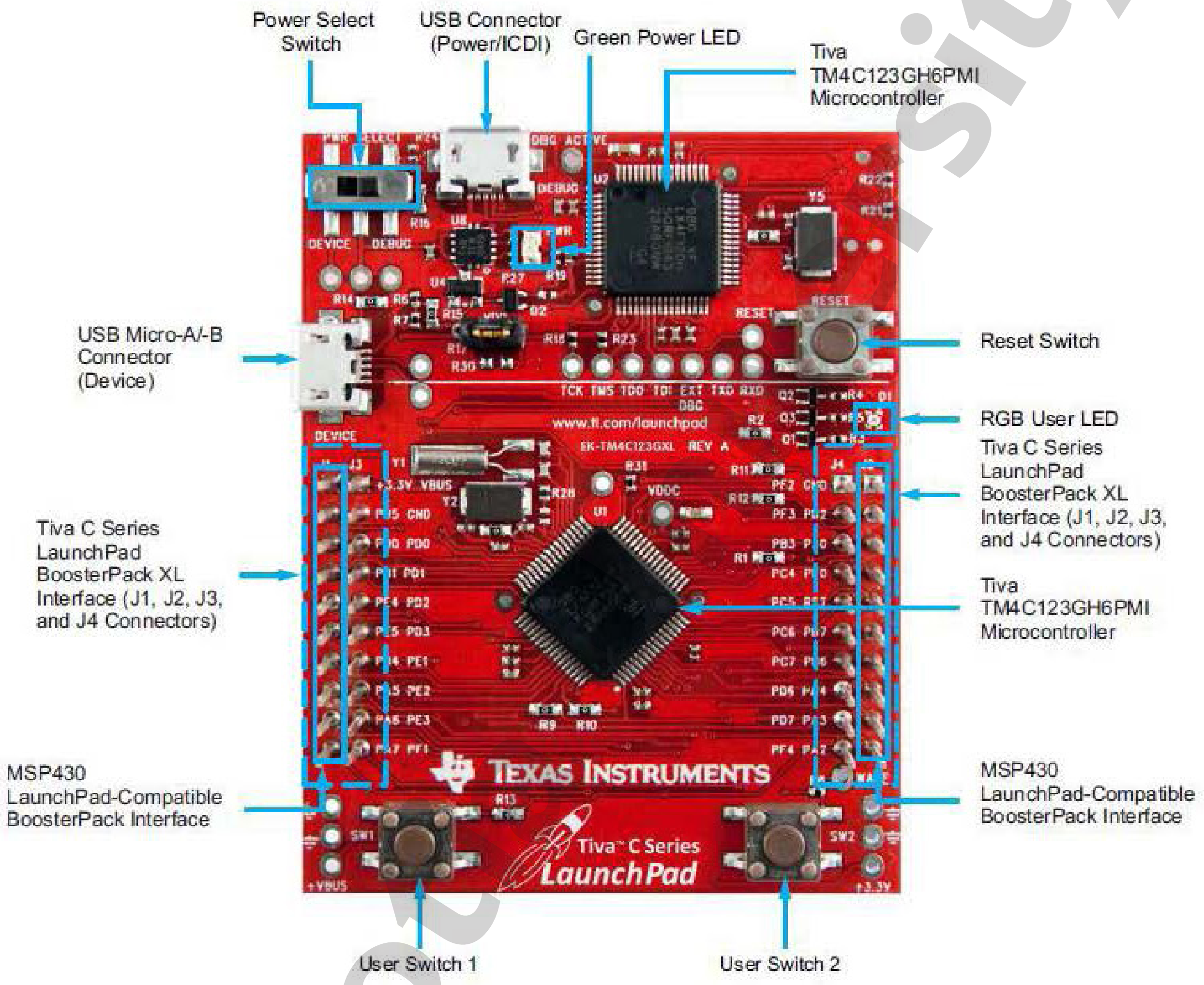
Energia is the open source software to program for TM4C123GXL. This is IDE (Integrated Development Environment) for programming, compiling and debugging. Software has facility to compile the program, upload the program into the target board.

Serial monitor is useful to debug the code in real time response of hardware and composed circuit.

TM4C123GXL PIN OUT DIAGRAM:


TIVA™ C Series
TM4C123G LaunchPad
Texas Instruments
Energia
Pinout Diagram Ver 1.0
Hardware
digitalRead() and digitalWrite() PORTS
analogRead()
analogWrite()
I²C (TWI)
SPI
Hardware Serial
J1 J3
+3.3V 1 21 VBUS
CS(2) A11 PB_5 2 22 GND
RX(1) PB_0 3 23 PD_0 A7 SCL(3) SCK(3)
TX(1) PB_1 4 24 PD_1 A6 SDA(3) CS(3)
RX(5) SCL(2) A9 PE_4 PE_4 5 25 PD_2 A5 MISO(3)
TX(5) SDA(2) A8 PE_5 PE_5 6 26 PD_3 A4 MOSI(3)
SCK(2) A10 PB_4 7 27 PE_1 PE_1 A2 TX(7)
MOSI(0) PA_5 PA_5 8 28 PE_2 PE_2 A1
SCL(1) PA_6 PA_6 9 29 PE_3 PE_3 A0
SDA(1) PA_7 PA_7 10 30 PF_1 RED_LED MOSI(1)
GND
GND
VBUS +VBUS
J4 J2
SCK(1) BLUE_LED PF2 PF_2 40 20 GND
CS(1) GREEN_LED PF_3 PF_3 39 19 PB_2 PB_2 SCL(0)
SDA(0) PB_3 PB_3 38 18 PE_0 A3 RX(7)
RX(1) PC_4 PC_4 37 17 PF_0 PUSH2 MISO(1)
TX(1) PC_5 PC_5 36 16 RESET
RX(3) PC_6 PC_6 35 15 PB_7 MOSI(2)
TX(3) PC_7 PC_7 34 14 PB_6 MISO(2)
RX(2) PD-6 PD_6 33 13 PA_4 PA_4 MISO(0)
TX(2) PD_7 PD_7 32 12 PA_3 PA-3 CS(0)
PUSH1 PF_4 PF_4 31 11 PA_2 PA-2 SCK(0)
GND
GND
+3.3V

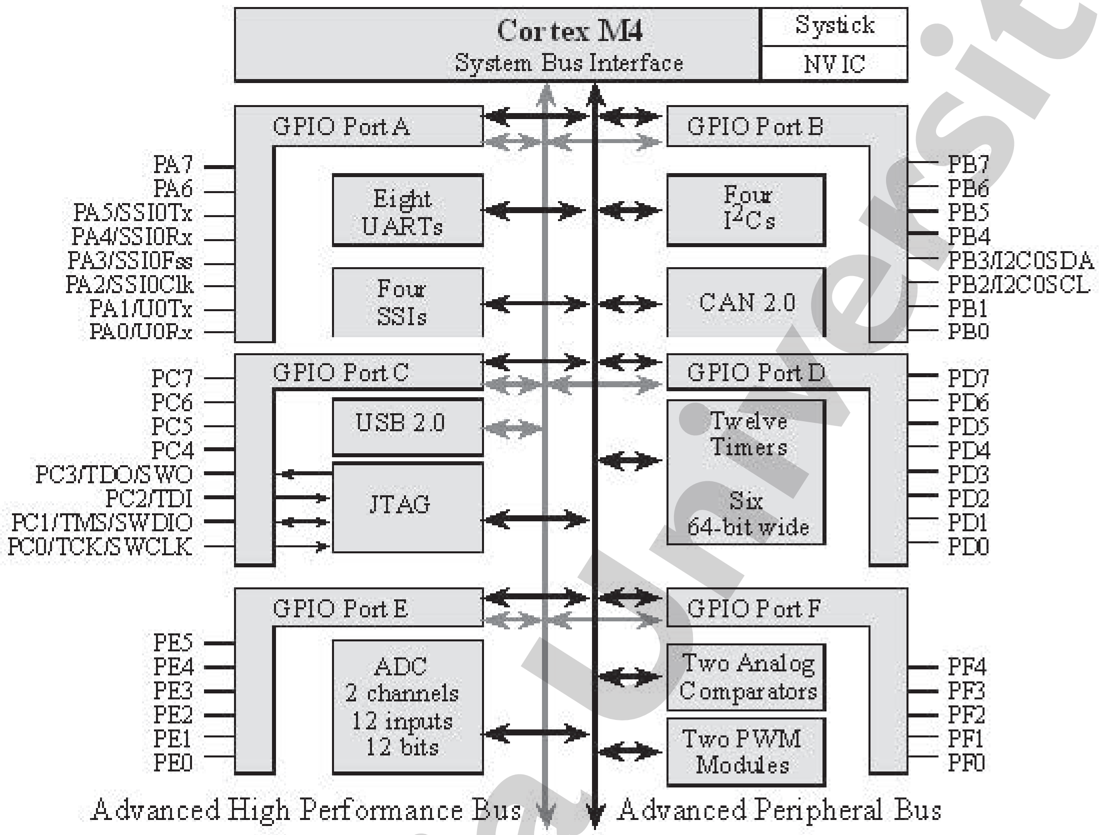
TM4C123GXL – OVER VIEW



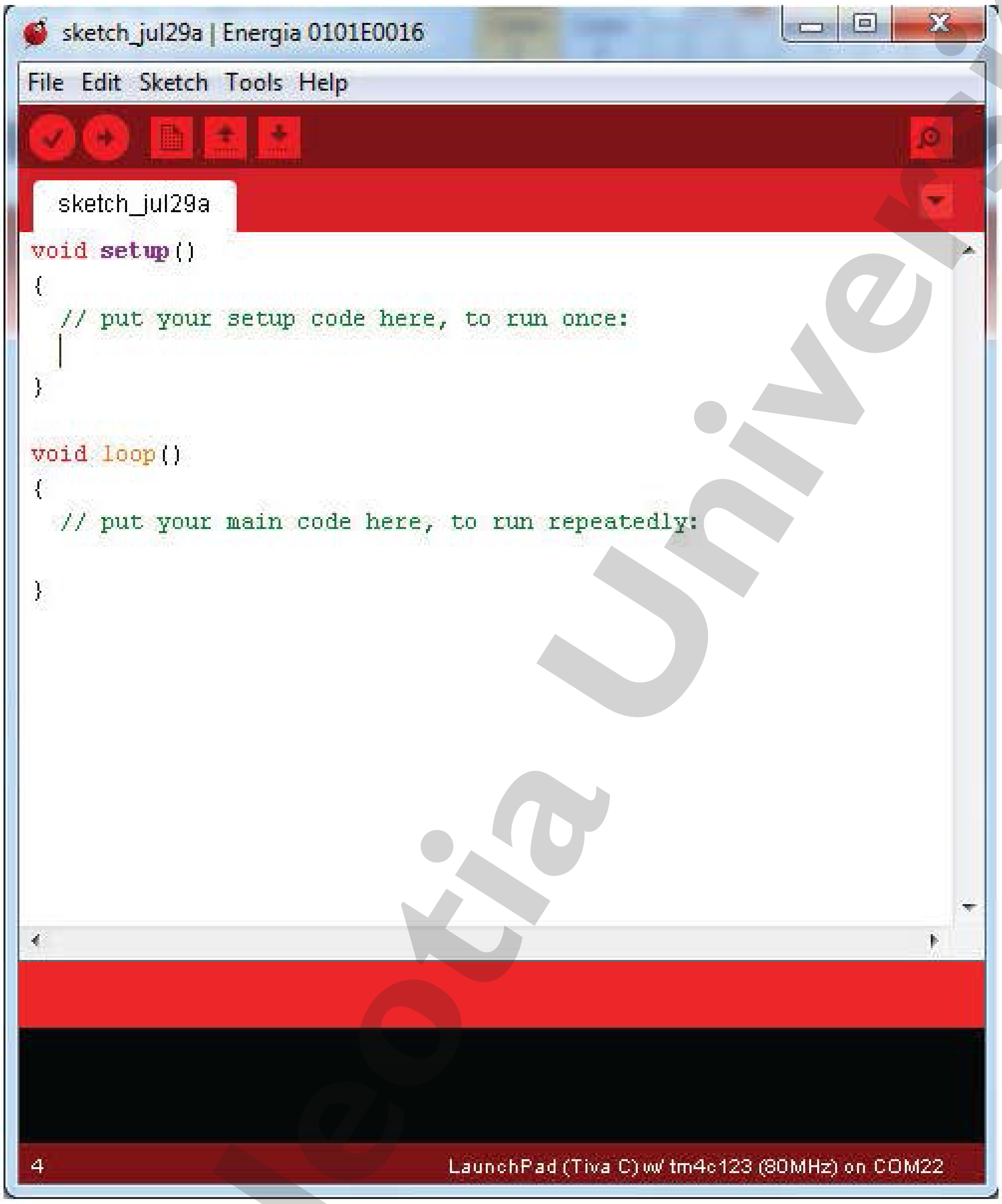
Power Select Switch
USB Connector (Power/ICDI)
Green Power LED
Tiva TM4C123GH6PMI Microcontroller
USB Micro-A/-B Connector (Device)
Reset Switch
RGB User LED
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva TM4C123GH6PMI Microcontroller
MSP430 LaunchPad-Compatible BoosterPack Interface
MSP430 LaunchPad-Compatible BoosterPack Interface
User Switch 1
User Switch 2

Tiva C Series TM4C123G LaunchPad Evaluation Board

TM4C123GXL – INTERNAL STRUCTURE DIAGRAM



ENERGIA SOFTWARE SCREEN:


sketch_jul29a | Energia 0101E0016
File Edit Sketch Tools Help
sketch_jul29a

```
void setup()
{
  // put your setup code here, to run once:

}

void loop()
{
  // put your main code here, to run repeatedly:

}
```

```
4                                   LaunchPad (Tiva C) w/ tm4c123 (80MHz) on COM22
```

RESULT :

Thus the study of ARM Evaluation is completed.

2. LED INTERFACING

AIM

To interface LED with ARM processor.

APPARTUS REQUIRED

1. ARM Processor EK -TM4C123GXL.
2. Computer with energia software.
3. LED.
4. Resister 1K.

THEORY

ARM processor is used for LED ON and OFF. Digital out of processor is connected to the LED. When digital out is HIGH, LED gets ON, When digital out is LOW, LED gets OFF. Delay is used in the program in between LED on and off to view the led change of state from ON to OFF.

PROCEDURE

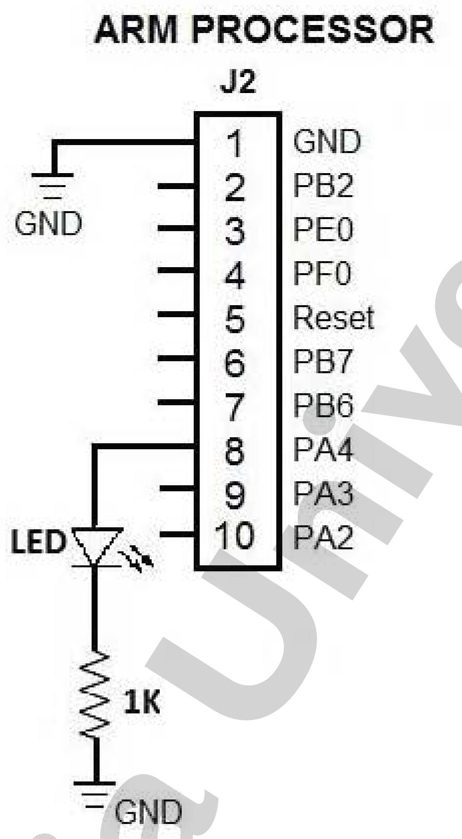
1. Connections are given as per the circuit diagram.
2. Upload energia program into the ARM processor.
3. Observe the LED ON and OFF like blinking led.

PROGRAM

```
void setup()
{
      pinMode(PA_4, OUTPUT);
}

void loop()
{
      digitalWrite(PA_4,HIGH);
      delay(100);
      digitalWrite(PA_4,LOW);
      delay(100);
}
```

INPUT	OUTPUT
PORT PA_4, HIGH	LED ON
PORT PA_4, LOW	LED OFF

CIRCUIT DIAGRAM : LED INTERFACING



RESULT

Thus the interfacing of LED with ARM processor is done successfully

3. FLASHING OF LED

AIM

To perform the flashing of LEDs with ARM processor

APPARATUS REQUIRED

1.ARM processor -TM4123GXL
2.Separate 5V power Supply
3.LEDs – 5nos
4.Bread Board
5.connecting wires

THEORY

5 LEDs are connected with Arm processor by individual digital pin. Program is written to ON LEDs one by one then off all the LEDs at a time, then ON the LEDs one by one, the same loop is operated. So the flashing of leds is done.

PROCEDURE

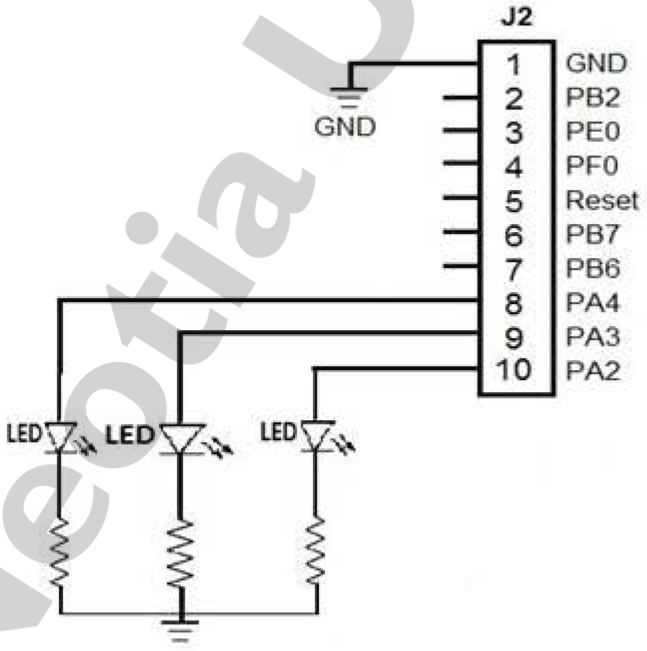
1.Connections are given as per the circuit diagram
2.energia program is loaded into the ARM processor
3.Output is observed in LEDs
4.LEDs are gets ON one by one, then at a time all the LEDs are turned OFF, then LEDs are get ON one by one loop is contined

PROGRAM

```
int tonDelay=1000;
int toffDelay=500;
int LED1=PA_2;
int LED2=PA_3;
int LED3=PA_4;

    void setup()
{
 pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
    pinMode(LED3,OUTPUT);
}
```

```
void loop()
{
  digitalWrite(LED1,HIGH);
  delay (tonDelay);
  digitalWrite(LED2,HIGH);
  delay (tonDelay);
  digitalWrite(LED3,HIGH);
  delay (tonDelay);
  digitalWrite(LED1,LOW);
  digitalWrite(LED2,LOW);
  digitalWrite(LED3,LOW);
  delay(toffDelay);

}
```

CIRCUIT DIAGRAM : FLASHING OF LED



RESULT

Thus the flashing of LEDs by ARM processor is performed and verified.

4. ADC INTERFACING WITH ARM PROCESSOR

AIM

To interface convert Analog signal in digital form using ADC in ARM processor.

APPARATUS REQUIED

1. ARM processor, EK-TM4C123GXL
2. Computer with energia software.
3. 1kΩ Resistor – 5 nos
4. Bread Board.
5. Connecting wires

THEORY

Five 1kΩ resistors are connected in series. 5V is supplied and the circuit is closed by the ground. 5V is applied across 5 1kΩ resistors, so each resistor drops 1v, totally 5v. Test point in ground, it produces 0 analog voltage, From ground after the first resistor produces 1 analog volt, after the second resistor produces 2 analog volt, after the third resistor produces 3 analog volt, after the fourth resistor produces 4 analog volt, all these test point voltage nodes are connected with the analog IN pin of ARM processor one by one. Program is written to read analog value in analog pin using analog read syntax and this received scale value in program is converted by multiplication factor and can be displayed as 1,2,3,4 volt as digital numbers. Finally analog voltage across the resistors is displayed as digital voltage value numbers in ARM processor serial monitor.

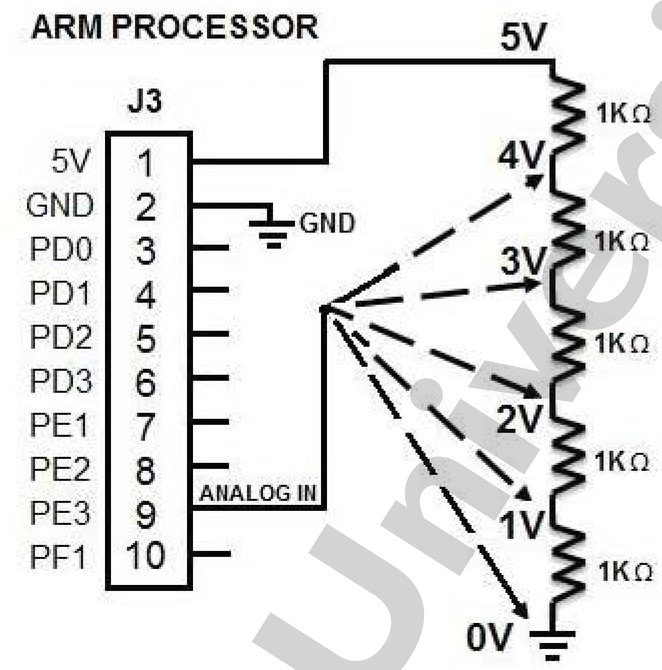
PROCEDURE

1. Connections are given as per the circuit diagram.
2. ADC energia program is loaded in ARM processor.
3. Open the serial monitor in energia
4. In circuit connect analog pin to test point - gnd, 1, 2, 3, 4.
5. Observe the respective digital voltage values in serial monitor as 0,1,2,3,4

PROGRAM

```
int analogPin = PE_2;
int volt;
int val = 0;
void setup()
{
        Serial.begin(9600);
}
void loop()
{
        val = analogRead(analogPin);
        Serial.print("scale value = ");
        Serial.println(val);
        Volt = val/1000;
        Serial.print(volt);
        Serial.println(" volts");
        delay(1000);
}
```

CIRCUIT DIAGRAM : **ADC INTERFACING WITH ARM PROCESSOR**



INPUT ANALOG VALUE (Mutimeter)	OUTPUT DIGITAL VALUE (energia serial monitor)
GND	0
1	1
2	2
3	3
4	4

RESULT

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

5. DAC INTERFACING WITH ARM PROCESSOR

AIM

To interface and convert Digital to Analog using DAC in ARM processor.

APPARATUS REQUIED

1. ARM processor, EK-TM4C123GXL
2. Computer with energia software.
3. Voltmeter (0-20)v or Multimeter
4. Connecting wires

THEORY

Digital scale value in program is uploaded to ARM processor. This value is passed to digital to analog converting pin using analog write program line, so equivalent digital value is obtained at the output of that pin. This is monitored by Voltmeter connected across the same pin with ground.

PROCEDURE

1. Connections are given as per the circuit diagram.
2. DAC energia program is loaded in ARM processor.
3. In circuit connect digital to analog pin to + terminal of voltmeter
4. Another end of voltmeter to be connected with ground.
5. Observe the respective analog voltage in voltmeter.

<u>PROGRAM</u> – FIXED VALUE

```
int digitalscalevalue =85;
int analogpin=PB_2;
void setup()
{
Serial.begin(9600);
  pinMode(analogpin, OUTPUT);
}
void loop()
{
 analogWrite(analogpin,  digitalscalevalue);
 Serial.println( digitalscalevalue);
}
```

INPUT	OUTPUT
85	1 volt
140	2 volt
240	3 volt

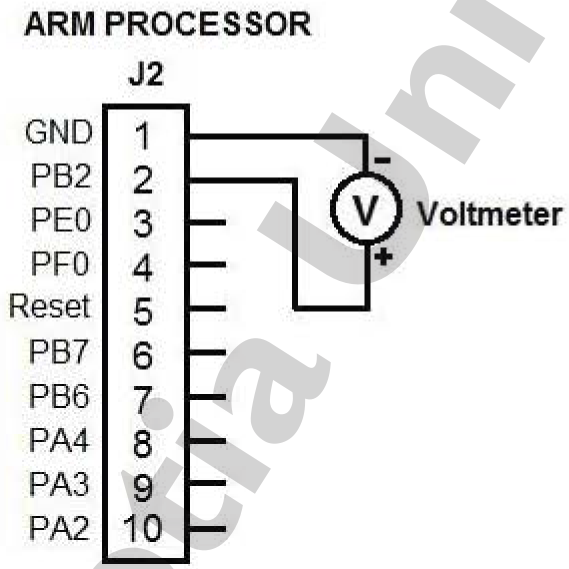
<u>PROGRAM</u> - VARIABLE VALUE

```
int digitalscalevalue =0;
int addvalue = 5;
int analogpin=PB_2;
void setup()
{
 pinMode(analogpin, OUTPUT);
}
```

INPUT	OUTPUT
5	0.1 volt
10	0.2 volt
15	0.3 volt
.	.
.	.
255	3.3 volt

```
void loop()
{
  analogWrite(analogpin,digitalscalevalue);
  delay(30);
  digitalscalevalue = digitalscalevalue +
addvalue;
  if (digitalscalevalue >255 )
  {
    digitalscalevalue =0;
  }
}
```

CIRCUIT DIAGRAM : DAC INTERFACING WITH ARM PROCESSOR



RESULT

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

6. DC MOTOR SPEED CONTROL (PWM)

AIM

To interface the DC motor with ARM processor and perform the speed control.

APPARTUS REQUIRED

5. ARM Processor, EK-TM4C123GXL.
6. BC547.
7. 12 V DC Motor.
8. 12 V Power supply.
9. LED.
10. Resister 1K.

THEORY

Pulse width modulation is digital signal changing high and low with respect to time. The time is varying for high state and low state. The low state is less time dc motor is low speed. The high is higher then low state dc motor run high speed.

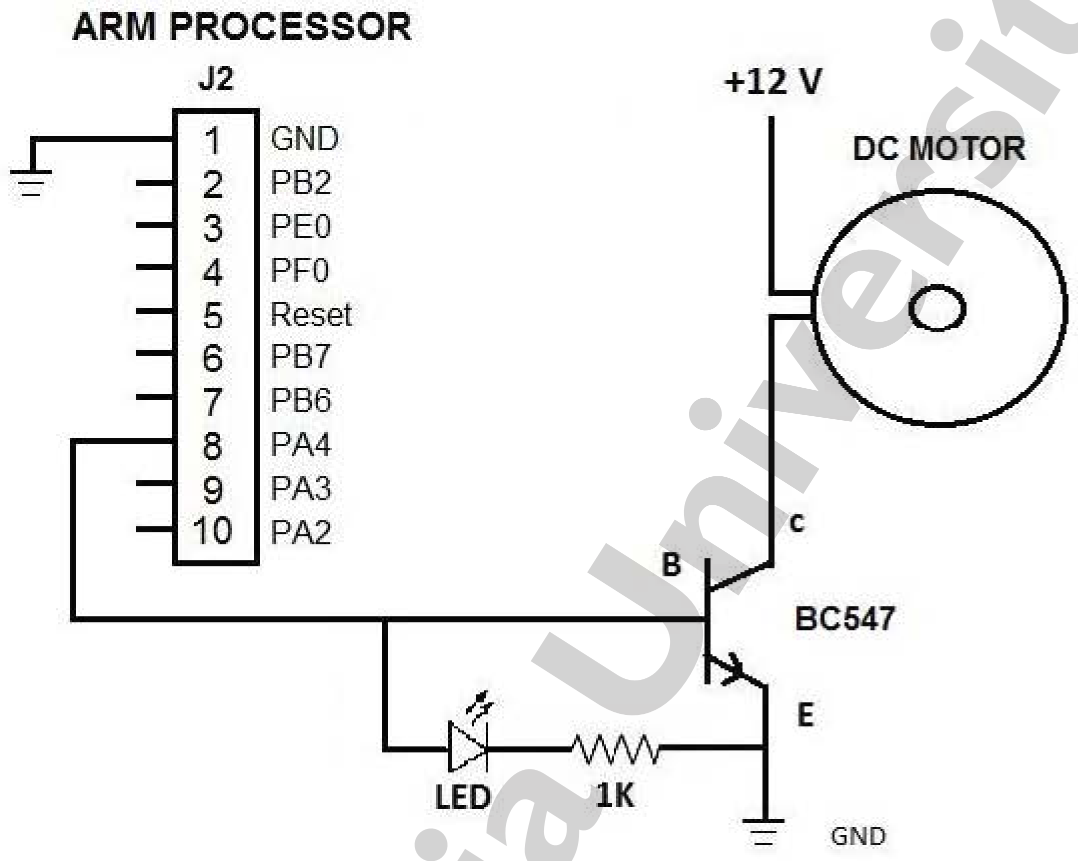
PROCEDURE

1. Connections are given as per the Diagram.
2. For **LOW** speed : Set ON time 100ms and OFF time 10ms.
3. Upload energia programe to ARM processor.
4. Observe motor speed.
5. For **HIGH** speed : Set ON time 10ms and OFF time 100ms.
4. Upload energia programe to ARM processor.
4. Observe motor speed.

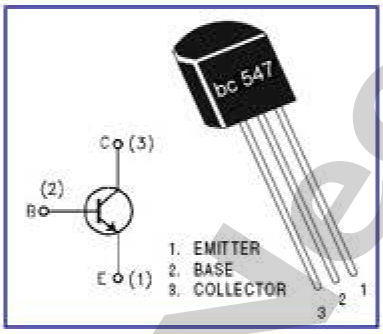
PROGRAM

```
void setup()
{
  // put your setup code here, to run once:
  pinMode(PA_4, OUTPUT);
}

void loop()
{
  digitalWrite(PA_4,HIGH);
  delay(10);
  digitalWrite(PA_4,LOW);
  delay(100);
}
```

CIRCUIT DIAGRAM: DC MOTOR SPEED CONTROL (PWM)



BC547 PIN DETAIL



INPUT:
For **LOW** speed:
HIGH time = 10ms
LOW time = 100ms

OUTPUT:
For **HIGH** speed:
HIGH time = 100 ms
LOW time = 10ms**CIRCUIT DIAGRAM**

RESULT

Thus the interfacing of DC motor with ARM processor is done successfully and speed control of DC motor is performed.

7. IMPLEMENTING INTERRUPT IN ARM PROCESSOR

AIM

To implement interrupt in ARM processor.

APPARATUS REQUIRED

1. ARM processor, EK-TM4C123GXL
2. Computer with energia software.
3. LED – 3 nos
4. Bread Board.
5. Connecting wires

THEORY

Two interrupt ports are available in ARM processor. Two interrupt ports are PUSH1, PUSH2. PUSH1 is linked with PF4 and PUSH2 is linked with PF0, so external switch is to be connected with PF4 to trigger the PUSH1 interrupt, and PF0 for PUSH2.

SW1 and SW2 switch are present in the ARM target board, SW1 can be used instead of connecting external switch to PF4 to trigger PUSH1 interrupt. SW2 can be used instead of connecting external switch to PF0 to trigger PUSH2 interrupt.

These interrupts are initiated on RISING or FALLING of signal which is to provided through switches and as mentioned in program in “attachInterrupt” function.

PROCEDURE

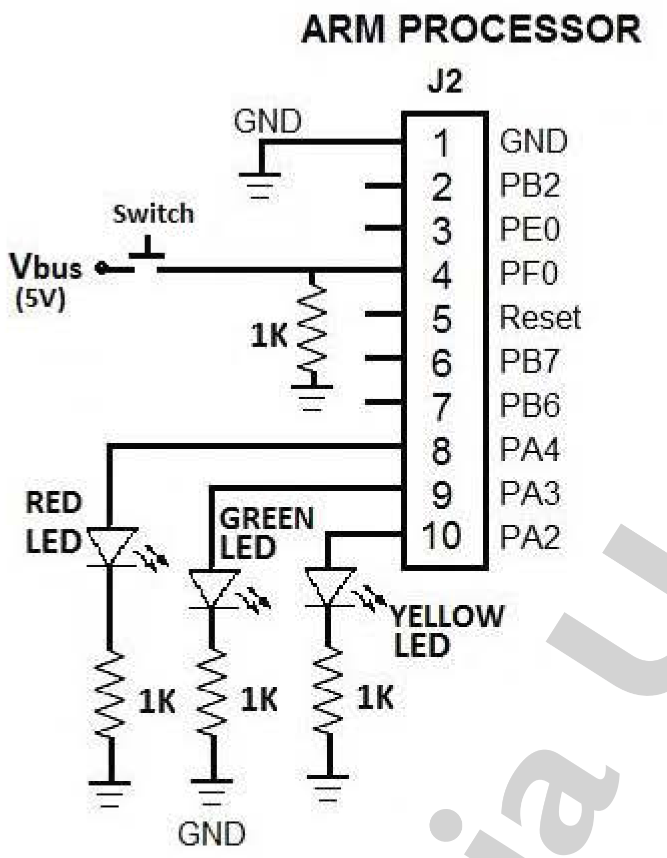
1. Connections are given as per the circuit diagram.
2. Program is uploaded to ARM processor.
3. Output is verified in LEDs.
4. The main program execution is observed in Red and Green Leds continuously blinking.
5. When pressing External interrupt switch Yellow Led is glowing for some time due to interrupt is initiated.

PROGRAM

```
int ledRed=PE_1;
int ledGreen=PE_2;
int ledYellow=PE_3;
int interrCount=0;

void setup()
{
  pinMode(ledRed, OUTPUT);
  pinMode(ledGreen, OUTPUT);
  pinMode(ledYellow, OUTPUT);
```

```
   digitalWrite(ledRed, LOW);
  digitalWrite(ledGreen, LOW);
  digitalWrite(ledYellow, LOW);

  pinMode(PUSH2, INPUT_PULLUP);

  attachInterrupt(PUSH2, interruptBlink, RISING);
}

void loop()
{
  interrCount++;

  digitalWrite(ledRed, HIGH);
  digitalWrite(ledGreen, LOW);
  delay(300);
  digitalWrite(ledRed, LOW);
  digitalWrite(ledGreen, HIGH);
  delay(300);

  if ( interrCount == 10 )
  {
    interrCount = 0;
    digitalWrite(ledYellow, LOW);
  }

}

void interruptBlink()
{
  digitalWrite(ledYellow, HIGH);
}
```

CIRCUIT DIAGRAM : **INTERRUPT**



INPUT

HIGH and LOW signal to Red and Green Led
HIGH signal to Yellow Led when interrupt is triggered

OUTPUT

Red and Green Led continuously blinking
Yellow led blinking when External interrupt switch is pressed.

RESULT

Thus the implementation of interrupt is performed in ARM processor.

8. KEYBOARD INTERFACING WITH ARM PROCESSOR

AIM

To interface keyboard with ARM processor

APPARTUS REQUIRED

1. ARM Processor EK -TM4C123GXL.
2. Computer with energia software.
3. 3x3 matrix keyboard.

THEORY

Keyboard is scanned and the pressed key is detected then it is printed. When the columns pins of ARM processor is set in output mode, 5v is supplied by default to this pins. 10k resistor is connected in each column and another end of all this resistor is connected to ground. To start scanning, Column1 is set as high, remaining columns set as low, so 5v is available in column1 alone. When scanning all the rows, if any key is pressed in row1 will received 5v in row1, so column1 row1 key is the pressed key, it is assumed as value of this position is 1, so 1 is printed. If row2 key is pressed, 5v in column1 is received in row2, so the pressed key is column1 row2, it is assumed as 4 as it is the 3x3 matrix, this process is continued to row3. Next step column2 is enabled by setting it as high and all the rows are scanned and it key values are assumed like 2,5,8 if it is pressed.

PROCEDURE

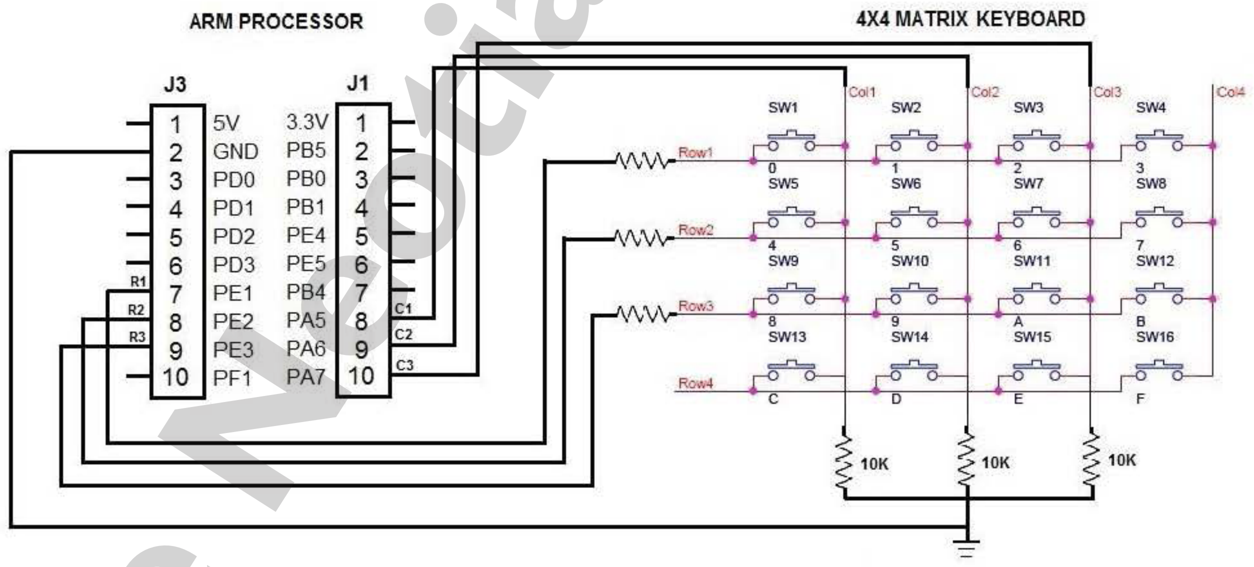
1. Connect 3 digital lines from ARM processor to row lines of keyboard
2. Connect another 3 digital lines from ARM processor to Columns line of keyboard
3. Upload energia program to processor.
4. Open serial monitor in computer.
5. Press any key in keyboard and observe that the same key is displayed in serial monitor.

PROGRAM

```
int C1= PA_5;
int C2= PA_6;
int C3= PA_7;

int R1= PE_1;
int R2 = PE_2;
int R3 = PE_3;

void setup()
{
 pinMode(C1, OUTPUT); // c1
 pinMode(C2, OUTPUT);  // c2
 pinMode(C3, OUTPUT);  // c3
```

```
  pinMode(R1, INPUT);  // r1
  pinMode(R2, INPUT);  //r2
  pinMode(R3, INPUT);  //r3
  Serial.begin(9600);
}

void loop()
{
        delay(150);

        // C1- column 1 scanning
        digitalWrite(C1, HIGH);
        digitalWrite(C2, LOW);
        digitalWrite(C3, LOW);

        if(digitalRead(R1) == HIGH & digitalRead(C1) == HIGH)
        {
                Serial.println("1");
        }
        if(digitalRead(R2)==HIGH & digitalRead(C1) == HIGH)
        {
                Serial.println("2");
        }
        if(digitalRead(R3)==HIGH & digitalRead(C1) == HIGH)
        {
                Serial.println("3");
        }

        // C2 - column 2 scanning
        digitalWrite(C1, LOW);
        digitalWrite(C2, HIGH);
        digitalWrite(C3, LOW);

         if(digitalRead(R1)==HIGH & digitalRead(C2) == HIGH)
        {
                 Serial.println("4");
        }

        if(digitalRead(R2)==HIGH & digitalRead(C2) == HIGH)
        {
                Serial.println("5");
        }

        if(digitalRead(R3)==HIGH & digitalRead(C2) == HIGH)
         {
                Serial.println("6");
        }

        // C3 - column 3 scanning
        digitalWrite(C1, LOW);
        digitalWrite(C2, LOW);
         digitalWrite(C3, HIGH);

        if(digitalRead(R1)==HIGH & digitalRead(C3) == HIGH)
        {
                Serial.println("7");
        }

        if(digitalRead(R2)==HIGH & digitalRead(C3) == HIGH)
        {
                Serial.println("8");
        }

        if(digitalRead(R3)==HIGH & digitalRead(C3) == HIGH)
        {
                Serial.println("9");
        }
}
```

CIRCUITDIAGRAM: KEYBOARD INTERFACING



INPUT

Key to be pressed in Keyboard
If 2 is pressed

OUTPUT

2 – is displayed in serial monitor

RESULT

Thus the keyboard interfacing with ARM processor is done and pressed key is verified successfully.

9. LCD INTERFACING WITH ARM PROCESSOR

AIM

To interface LCD with ARM processor and display the text.

APPARATUS REQUIRED

1. ARM processor - EK-TM4C123GXL
2. Computer with energia software
3. LCD 16x2
4. 10kΩ variable resistor – 1.
5. Separate 5V supply.

THEORY

VSS – GND
VDD – 5V (separate supply not from ARM processor)
V0 –3 wires 10kΩ pot, top wire to 5v, bottom wire to ground, middle wire to V0
RS – to arm processor digital pin
RW – to GND
E – to arm processor digital pin
D4,D5,D6,D7 – to arm processor digital pin

To glow up internal light of LCD
A – Anode to 5V supply
K – Cathode to GND

V0 – connected to 10kΩ pot, this is used to change brightness of LCD display.
ARM processor send RS-Reset and E-Enable signal to LCD display.
Then it sends the 4 bit digital data to LCD, to display text.

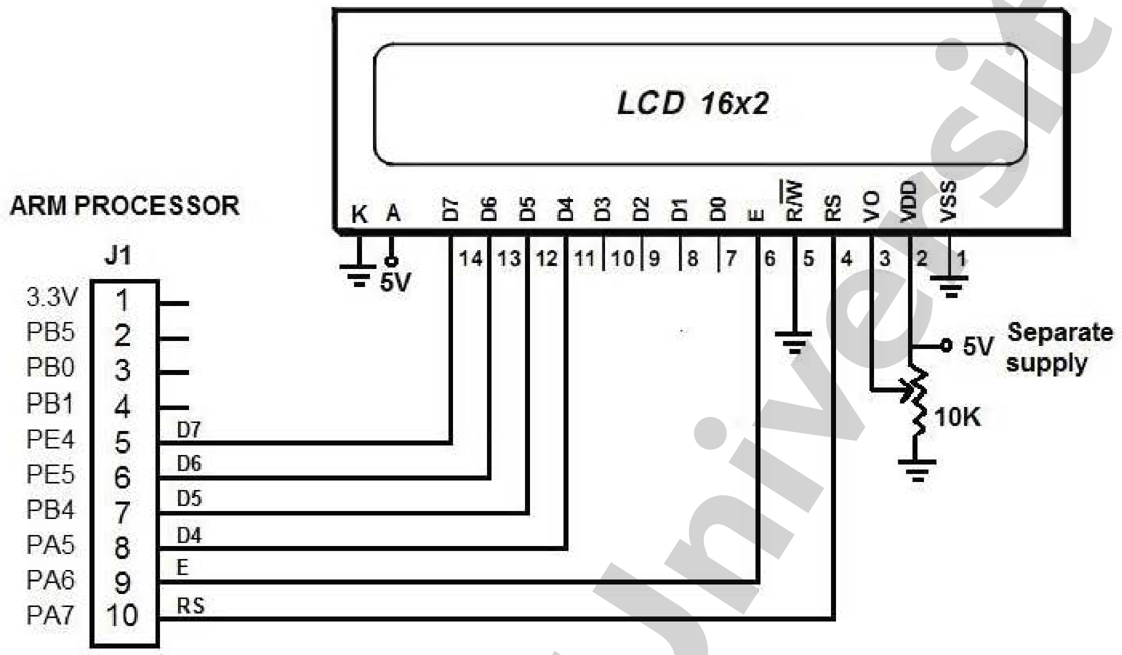
PROCEDURE

1. Connection are given as per the circuit diagram.
2. Separate 5V supply to be used in LCD display, ARM processor GND to connected to GND of separate 5v power supply.
3. Varying 10kΩ pot in V0 brightness of LCD is tuned.
4. LCD display program in energia is loaded to the ARM processor.
5. Observe the text displayed in LCD display

<u>**Note:**</u>

sketch=>add file+>liquid crystal.h &liquid crystal.cpp

CIRCUIT DIAGRAM: LCD INTERFACING



INPUT
"LCD TEST" in row0
"EMBEDED LAB" in row1

OUTPUT
LCD TEST
EMBEDED LAB

PROGRAM

```
#include <LiquidCrystal.h>
int RS =  PA_7;
int EN = PA_6;
int D4 = PA_5;
int D5 = PB_4;
int D6 = PE_5;
int D7 = PE_4;
LiquidCrystal lcd(RS,EN,D4,D5,D6,D7);
void setup()
{
    lcd.begin(16, 2);
 }
void loop()
{
    lcd.setCursor(0, 0);
    lcd.print("LCD TEST");
    lcd.setCursor(0, 1);
    lcd.print("EMBEDED LAB");
}
```

RESULT

Thus the interfacing LCD display with ARM processor is done and text is displayed in LCD.

10. TEMPERATURE SENSOR WITH ARM PROCESSOR

AIM

To interface temperature sensor with ARM processor and display the temperature.

APPARATUS REQUIRED

1. ARM processor, EK-TM4C123GXL
2. Computer with energia software.
3. Temperature sensor – LM35
4. Bread Board.
5. Connecting wires

THEORY

LM35 temperature sensor has 3 pin 5v,temp out, GND. Out pin of sensor connected to analog pin of ARM processor. In energia program is written to read analog pin value and this scale value is multiplied with factor to get temperature value in Celcius. It can be observed in serial monitor of energia

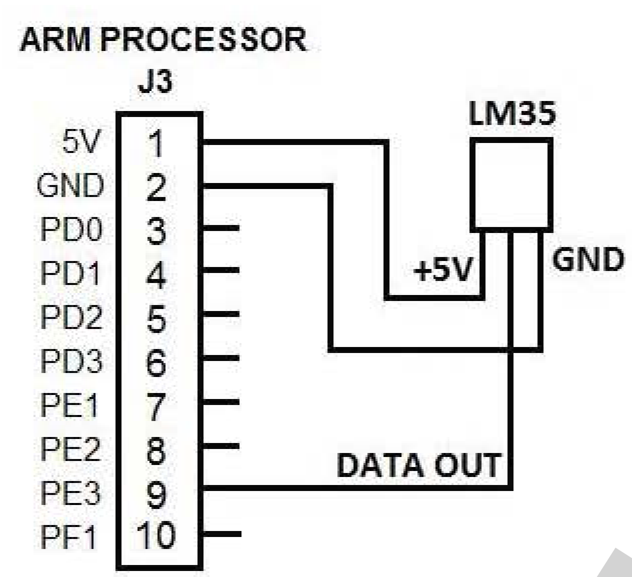
PROCEDURE

1. Connections are as per the circuit diagram.
2. Present room temperature is displayed in serial monitor.
3. Using heating medium temperature is increased around the LM35 temperature sensor.
4. Observe the increase in temperature in serial monitor.

PROGRAM

```
float tempC;
int reading;
int tempPin =PE_3;
int INTERNAL;

void setup()
{
analogReference(INTERNAL);
Serial.begin(9600);
}

void loop()
{
reading = analogRead(tempPin);
tempC = reading / 9.31;
Serial.println(tempC);
delay(1000);
}
```

CIRCUIT DIAGRAM: TEMPERATURE SENSOR



INPUT
Apply Heat to Temperature sensor

OUTPUT
Increase in temperature is observed in serial monitor

```
32
35
37
40
```

RESULT
Thus the interfacing of temperature sensor is done and temperature changes are observed.

11. STEPPER MOTOR INTERFACING

AIM

To perform a speed control of stepper motor using ARM processor.

APPARATUS REQUIRED:

1. ARM processor TM4123GXL
2. Stepper motor with driver module
3. Separate 5V power supply

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. 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.

PROCEDURE:

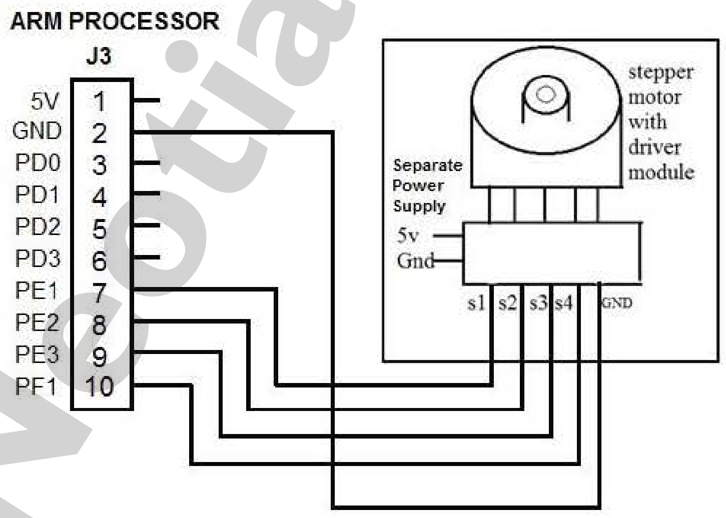
1. ARM processor USB is plugged with computer.
2. Digital output pins of ARM processor connected to the stepper motor driver.
3. Stepper motor sequence is programmed in energia.
4. Program is uploaded to ARM processor.
5. LED will glow in the given sequence.
6. Stepper motor rotation is observed.

PROGRAM:

```
int S1 = PE_1;
int S2 = PE_2;
int S3 = PE_3;
int S4 = PF_1;

void setup()
{
      pinMode(S1,OUTPUT);
      pinMode(S2,OUTPUT);
      pinMode(S3,OUTPUT);
      pinMode(S4,OUTPUT);
      int i = 5; // 5 meant for 5 ms, speed operation
}

void loop()
{
        //Sequence 1100
      digitalWrite(S1,HIGH);
      digitalWrite(S2,HIGH);
      digitalWrite(S3,LOW);
      digitalWrite(S4,LOW);
      delay(i);
```

```
        // sequence 0110
        digitalWrite(S1,LOW);
        digitalWrite(S2,HIGH);
        digitalWrite(S3,HIGH);
        digitalWrite(S4,LOW);
        delay(i);

        // sequence 0011
        digitalWrite(S1,LOW);
        digitalWrite(S2,LOW);
        digitalWrite(S3,HIGH);
        digitalWrite(S4,HIGH);
        delay(i);

        // sequence 1001
        digitalWrite(S1,HIGH);
        digitalWrite(S2,LOW);
        digitalWrite(S3,LOW);
        digitalWrite(S4,HIGH);
        delay(i);
}
```

CIRCUIT DIAGRAM: STEPPER MOTOR INTERFACING



INPUT/OUTPUT

1. For forward rotation sequence 1100, 0110, 0011, 1001 is used.
2. For reverse rotation sequence 1100, 1001, 0011, 0110 is used.
3. For speed operation, delay is set to 5ms.
4. For low speed operation, delay is set to 50ms.

RESULT:

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

12. EPROM INTERFACING

AIM:

To write and read data from EEPROM interfaced with ARM processor.

Apparatus required:

1. ARM processor TM4123GXL
2. 24C02 - EPROM
3. 1k resistor – 2 nos

EPROM:

24C02C is a 2K bit Serial Electrically Erasable PROM with a voltage range of 4.5V to 5.5V. It has a single block of 256 x 8-bit memory, It means 256 address location each has the capacity to store 8 bit, 256x8 = 2048 bit.

PROCEDURE:

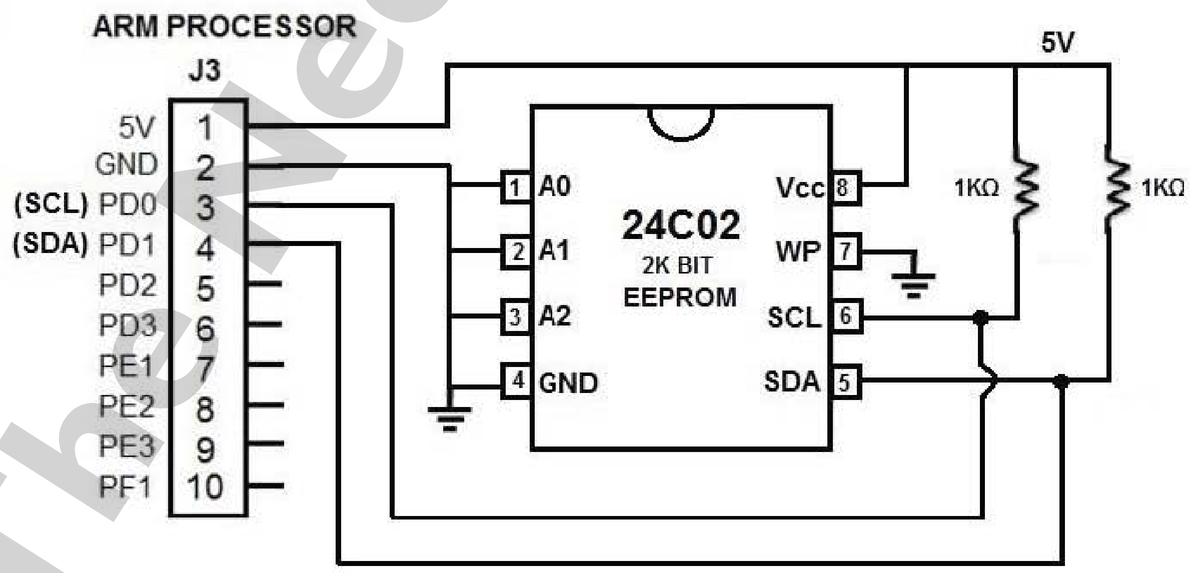
1. ARM processor USB is plugged with computer.
2. SDA(5), SCL(6)pin of EPROM is connected with the SDA(PD_1), SCL(PD_0) pin of ARM processor.
3. Pull up resistors each 1k ohms to be connected in SDA, SCL and ended with common 5v.
4. WP of EPROM to be connected in GND to enable both write and read operation in EPROM.
5. Load the program into ARM processor, open the serial monitor.
6. Setup part of program writes the data in EPROM memory from 0 to 256 address.
7. Loop part of the program reads the data from EPROM memory from 0 to 256 address again and again.
8. Writing data and reading data output can be viewed in serial monitor of energia.

PROGRAM:

```
#include <Wire.h>

void setup()
{
   Wire.begin();
   Serial.begin(9600);
   Serial.println("Writen to memory!");
   for(int i = 0; i < 100; i++)
   {
     eeprom_i2c_write(B01010000, i, 'A'+i);
     Serial.print(i);
     Serial.print(" - ");
     Serial.print('A'+i);
     Serial.print("\n");
     delay(100);
   }
}
```

```
void loop()
{
    Serial.println("Reading from memory!");
    for(int i = 0; i < 7; i++)
    {
      byte r = eeprom_i2c_read(B01010000, i);
      Serial.print(i);
      Serial.print(" - ");
      Serial.print(r);
      Serial.print("\n");
      delay(500);
    }
}

void eeprom_i2c_write(byte deviceaddress, byte memory_addr, byte data)
{
    Wire.beginTransmission(deviceaddress);
    Wire.write(memory_addr);
    Wire.write(data);
    Wire.endTransmission();
}

byte eeprom_i2c_read(int deviceaddress, int memory_addr)
{
    Wire.beginTransmission(deviceaddress);
    Wire.write(memory_addr);
    Wire.endTransmission();

    Wire.requestFrom(deviceaddress, 1);
    if(Wire.available())
      return Wire.read();
    else
      return 0xFF;
}
```

CIRCUIT DIAGRAM:



INPUT:

MEMORY ADDRESS	DATA (WRITING)
0	65
1	66
2	67
3	68
.	.
.	.
.	.
256	

OUTPUT:

MEMORY ADDRESS	DATA (READING)
0	65
1	66
2	67
3	68
.	.
.	.
.	.
256	

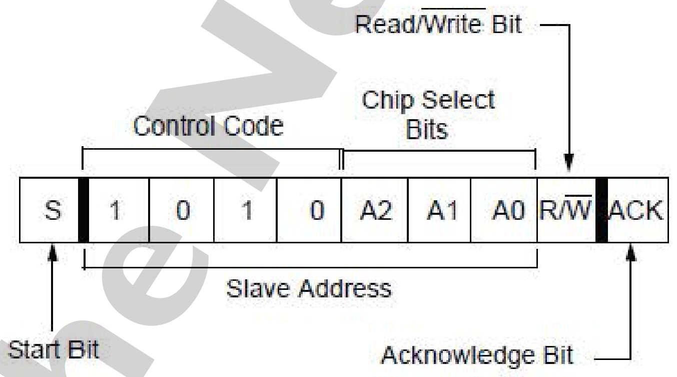
SDA (Serial Data)
This is a bidirectional pin used to transfer addresses and data into and data out of the device. It is an open drain terminal; therefore, the SDA bus requires a pullup resistor to VCC (typical 10 k for 100 kHz, 2 k for 400 kHz).

SCL (Serial Clock)
This input is used to synchronize the data transfer from and to the device.

A0, A1, A2
The levels on these inputs are compared with the corresponding bits in the slave address. The chip is selected if the compare is true. Up to eight 24C02C devices may be connected to the same bus by using different Chip Select bit combinations. If A0, A1, A1 are 0,0,0, it is the first device, so combination of three line upto 1,1,1 eight devices can be connected.

WP
This is the hardware write-protect pin. It must be tied to VCC or GND. If tied to Vcc, the hardware write protection is enabled. If the WP pin is tied to GND the hardware write protection is disabled.

DEVICE ADDRESSING:



RESULT:
Thus the writing and reading the data from EEPROM with ARM processor is performed

13. RTC INTERFACING WITH ARM

AIM:

To interface RTC (Real Time Clock – DS1307) with ARM processor.

Apparatus required:

1. ARM processor TM4123GXL
2. RTC – DS1307 with 3v battery
3. 1k resistor – 2 nos

RTC DS1307:

Real-time clock (RTC) counts seconds, minutes, hours, day of the week, date of the month, month, and year with leap-year. Full binary-coded decimal (BCD) clock/calendar and 56-byte, battery-backed, nonvolatile (NV) RAM for data storage. Two-wire serial interface are SDA, SCL. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

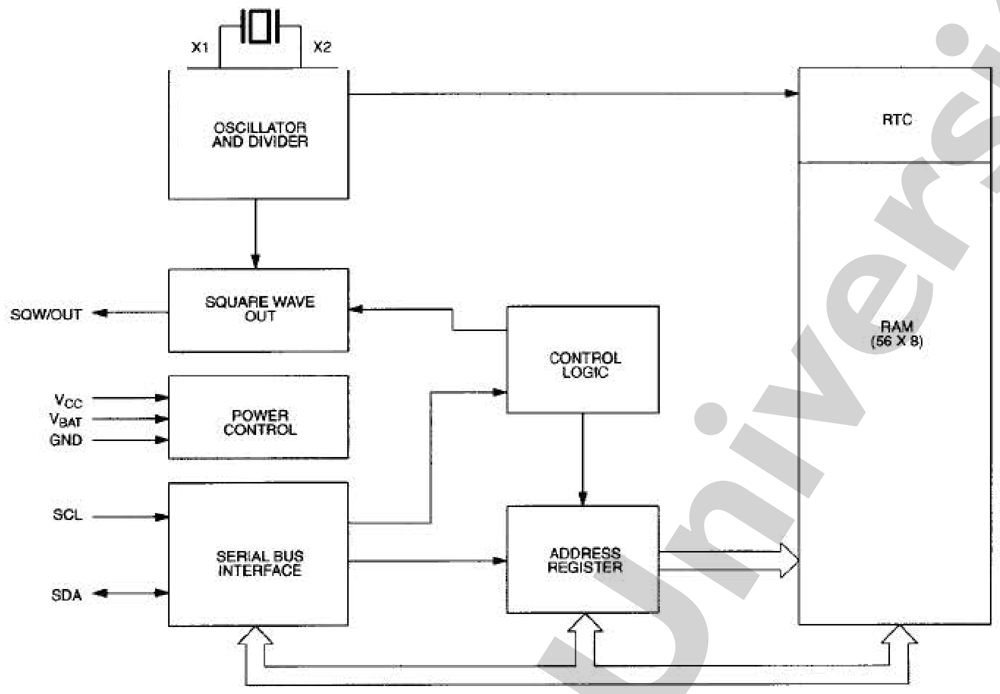
PROCEDURE:

1. ARM processor USB is plugged with computer.
2. SDA(5), SCL(6) pin of DS1307 is connected with the SDA(PD_1), SCL(PD_0) pin of ARM processor.
3. Pull up resistors each 1k ohms to be connected in SDA, SCL and ended with common 5v.
4. Load the program into ARM processor, open the serial monitor in energia.
5. Setup part of program has settime function, this sets the seconds, minutes, hours, day, date, month and year. This to be done only once when new RTC is used, then settime to be comments because every time of program upload should not set time.
6. Loop part of the program reads the seconds, minutes, hours, day, date, month and year from RTC memory from 0 to 7 address again and again.
7. Reading calendar values can be viewed in serial monitor of energia.

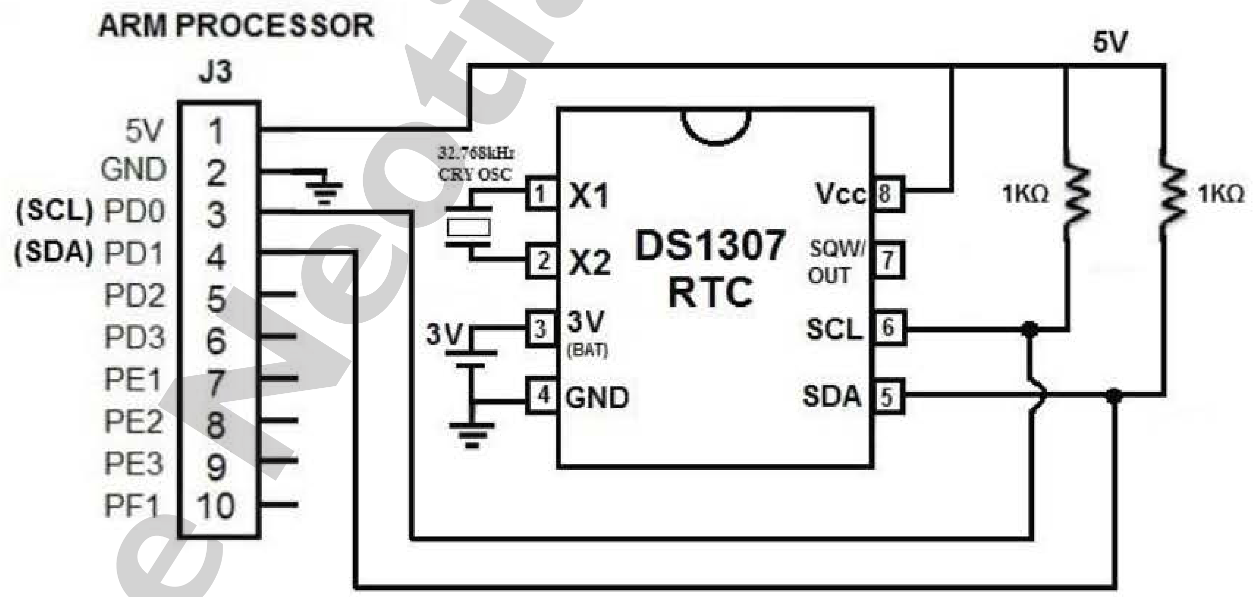
RTC MEMORY ORGANISATION:

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM
		24	PM/AM							00–23
03H	0	0	0	0	0	DAY			Day	01–07
04H	0	0	10 Date		Date				Date	01–31
05H	0	0	0	10 Month	Month				Month	01–12
06H	10 Year				Year				Year	00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H–FFH

DS1307 BLOCK DIAGRAM



CIRCUIT DIAGRAM:



INPUT:

setDS1307time(40,07,11,6,20,8,16);
feed parameters in program line as
seconds, minutes, hours, day, date, month, year
(this is sample values, current time value to be set)
40 – seconds
07 – minutes
11 – hours
6 – day (FRIDAY)
20 – date
8 – month
16 - year

OUTPUT:

```
11:07:40 20/8/16 Day of week: Friday
11:07:41 20/8/16 Day of week: Friday
11:07:42 20/8/16 Day of week: Friday
11:07:43 20/8/16 Day of week: Friday
11:07:44 20/8/16 Day of week: Friday
11:07:45 20/8/16 Day of week: Friday
.
.
.
.
```

PROGRAM:

```
#include "Wire.h"
#define DS1307_I2C_ADDRESS 0x68
void setup()
{
        Wire.begin();
        Serial.begin(9600);
        // set the initial time here:
        // DS1307 seconds, minutes, hours, day, date, month, year
        //setDS1307time(40,07,11,6,20,8,16);
}
void loop()
{
          displayTime(); // display the real-time clock data on the Serial Monitor,
          delay(1000); // every second
}

// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val)
{
          return( (val/10*16) + (val%10) );
}

// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
         return( (val/16*10) + (val%16) );
}

void setDS1307time(byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth, byte
month, byte year)
{
          // sets time and date data to DS1307
          Wire.beginTransmission(DS1307_I2C_ADDRESS);
          Wire.write(0); // set next input to start at the seconds register
          Wire.write(decToBcd(second)); // set seconds
        Wire.write(decToBcd(minute)); // set minutes
        Wire.write(decToBcd(hour)); // set hours
        Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday, 7=Saturday)
        Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)
        Wire.write(decToBcd(month)); // set month
        Wire.write(decToBcd(year)); // set year (0 to 99)
        Wire.endTransmission();
}

void readDS1307time(byte *second, byte *minute, byte *hour, byte *dayOfWeek, byte *dayOfMonth, byte
*month, byte *year)
{
        Wire.beginTransmission(DS1307_I2C_ADDRESS);
        Wire.write(0); // set DS1307 register pointer to 00h
        Wire.endTransmission();
        Wire.requestFrom(DS1307_I2C_ADDRESS, 7);
        // request seven bytes of data from DS1307 starting from register 00h
        *second = bcdToDec(Wire.read() & 0x7f);
        *minute = bcdToDec(Wire.read());
        *hour = bcdToDec(Wire.read() & 0x3f);
        *dayOfWeek = bcdToDec(Wire.read());
        *dayOfMonth = bcdToDec(Wire.read());
        *month = bcdToDec(Wire.read());
        *year = bcdToDec(Wire.read());
}

void displayTime()
{
       byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
        // retrieve data from DS1307
        readDS1307time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month,  &year);
        // send it to the serial monitor
        Serial.print(hour, DEC);
        // convert the byte variable to a decimal number when displayed
        Serial.print(":");
        if (minute<10)
        {
       Serial.print("0");
        }
        Serial.print(minute, DEC);
        Serial.print(":");
        if (second<10)
        {
            Serial.print("0");
        }
        Serial.print(second, DEC);
       Serial.print(" ");
        Serial.print(dayOfMonth, DEC);
        Serial.print("/");
        Serial.print(month, DEC);
        Serial.print("/");
        Serial.print(year, DEC);
        Serial.print(" Day of week: ");
        switch(dayOfWeek)
        {
        case 1:
            Serial.println("Sunday");
            break;
          case 2:
            Serial.println("Monday");
            break;
          case 3:
            Serial.println("Tuesday");
            break;
          case 4:
            Serial.println("Wednesday");
            break;
          case 5:
            Serial.println("Thursday");
            break;
          case 6:
            Serial.println("Friday");
            break;
          case 7:
            Serial.println("Saturday");
            break;
        }
}
```

SIMPLE PROGRAM - RTC

```
#include <Wire.h>
# define dev_add  0x68
void setup()
{
   Wire.begin();
   Serial.begin(9600);
   //    "Writen to memory!"
 // Writing date in memory, one time is enough, if the date to be changed, It can be written again.
 //    eeprom_i2c_write(dev_add, 0, decToBcd(21));
 //    eeprom_i2c_write(dev_add, 1, decToBcd(36));
 //    eeprom_i2c_write(dev_add, 2, decToBcd(2));
 //    eeprom_i2c_write(dev_add, 3, decToBcd(3));
 //    eeprom_i2c_write(dev_add, 4, decToBcd(31));
 //    eeprom_i2c_write(dev_add, 5, decToBcd(8));
 //    eeprom_i2c_write(dev_add, 6, decToBcd(16));
}

void loop()
{
  byte sec = eeprom_i2c_read(dev_add, 0);
  sec = bcdToDec(sec);
  byte minute = eeprom_i2c_read(dev_add, 1);
  minute = bcdToDec(minute);
  byte hour = eeprom_i2c_read(dev_add, 2);
  hour = bcdToDec(hour);
  byte day = eeprom_i2c_read(dev_add, 3);
  day = bcdToDec(day);
 byte date = eeprom_i2c_read(dev_add, 4);
 date = bcdToDec(date);
 byte month = eeprom_i2c_read(dev_add, 5);
 month = bcdToDec(month);
 byte year = eeprom_i2c_read(dev_add, 6);
 year = bcdToDec(year);
 Serial.println("HOUR : MIN : SEC  DATE/MONTH/YEAR , DAY");
 Serial.println(String(hour) + " : " + minute + " : " + sec + "  " + date + "/" + month + "/" + year + " , " + day);
 delay(5000);
}
byte decToBcd(byte val)
{
 return( (val/10*16) + (val%10) );
}

byte bcdToDec(byte val)
{
 return( (val/16*10) + (val%16) );
}

void eeprom_i2c_write(byte deviceaddress, byte memory_addr, byte data)
{
   Wire.beginTransmission(deviceaddress);
   Wire.write(memory_addr);
   Wire.write(data);
   Wire.endTransmission();
}

byte eeprom_i2c_read(int deviceaddress, int memory_addr)
{
   Wire.beginTransmission(deviceaddress);
   Wire.write(memory_addr);
   Wire.endTransmission();

   Wire.requestFrom(deviceaddress, 1);
   if(Wire.available())
     return Wire.read();
   else
     return 0xFF;
}
```

Note: Writing date in memory, one time is enough, if the date to be changed can be written again.

TEST data stored and retrieved from RTC memory in RTC Memory order

Function	Address	BCD Retrieved	Decimal Converted
Sec	0	21	33
Min	1	24	17
Hour	2	18	12
Day	3	4	4
Date	4	49	31
Month	5	8	8
Year	6	22	16

<u>OUTPUT:</u>

Date retrieved to display in Date Time format order:

	DATE	MONTH	YEAR	HOUR	MIN	SEC	DAY
	31	8	16	12	17	33	Wed
Address	4	5	6	2	1	0	3
Retrieved	49	8	22	18	24	21	4
BCD TO DEC	**31**	**8**	**16**	**12**	**17**	**33**	**4**

RESULT:

Thus the interfacing of Real Time Clock with ARM processor is performed and calendar value is displayed.