|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust (R.)

BGS INSTITUTE OF TECHNOLOGY

[Affiliated to VTU, Belgaum; Approved by AICTE, New Delhi and Recognized by Govt. of Karnataka)

BG Nagara – 571 448 (Bellur Cross)
Nagamangala Taluk, Mandya District

Practical Record

Name : VINUTHA.C

Branch : ECE Sem : 6th 'B'

USN : 16EC116

Subject : Embedded controller

CET Code : 142

|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust (R.)

BGS INSTITUTE OF TECHNOLOGY

[Affiliated to VTU, Belgaum; Approved by AICTE, New Delhi and Recognized by Govt. of Karnataka)

BG Nagara – 571 448 (Bellur Cross)
Nagamangala Taluk, Mandya District

Certificate

This is to certify that Mr/Ms. VINUTHA.C

USN: 4BW.16EC116 has satisfactorily completed the course of experiments in Embedded Controller Laboratory (Course Code: 15ECL67) prescribed by the Visvesvaraya Technological University, Belagavi for VI Semester, BE. Electronics and Communication Engineering, of this College in the year 2018 - 2019

Record Marks : 12 Test Marks : 08

IA Marks : 20

Date : 22/5/19

B.S.Balaji
Staff Incharge

Head of the Department

INDEX

Name of the Student Vinutha.C Class 5th Sem B Sec

Expt. No.	Date	Tittle of the Experiment	Page No.	Marks obtained a	b	c	d	Total	Sign. of the Staff
~~01.~~	~~25~~/02/19	INTRODUCTION	01-08						
01.	~~25~~/02/19	(a) ALP to multiply 2 16 bit nos	09	2	4	2	4	12	
02.	15/02/19	(b) ALP to find the sum of first 10 integer	10						
		PART-B							
01.	01/03/19	Interface a simple switch & display its status through	11-12	2	4	2	4	12	
		relay, buzzer & LED.	13-14						
02.	15/3/19	Interface DC motor & rotate		2	4	2	4	12	
		a) Clockwise & b) anticlockwise	15						
		c) Both clockwise & anticlockwise	16						
		d) Both clockwise & anticlockwise using switches.	17						
		e) Rotation of motor clockwise using switch.	18						
		f) Rotation of motor anticlockwise using switch.	19						
03.	29/3/19	Interface of Steppee motor & rotate		2	4	2	4	12	
		a) Clockwise direction	20						
		b) Anticlockwise direction.	21						
		c) Both clockwise & anticlockwise direction.	22-23						
		d) Both clockwise & anticlockwise using switches.	24-25						
		e) Rotate at 180°	26						
		f) Rotate motor at 90°	27						
		g) Rotate at 360°	28						
		h) Rotate motor using keys.	29-31						
04.	5/4/19	Hexadigit on 7 segment LED.		2	4	2	4	12	
		(a) from 0-f	32						
		(b) from f-0.	33						

INDEX

Name of the Student ..... Class ..... Sem. .....

Expt. No.	Date	Tittle of the Experiment	Page No.	Marks obtained					Sign. of the Staff
				a	b	c	d	Total	
		C> Display B&SIT	34						
05.	03/05/19	(a) DAC - square wave (b) Triangular wave.	35 36	2	4	2	4	12	
06.	03/05/19	Use an external interrupt to toggle LED ON/OFF	37	2	4	2	4	12	
07.	03/05/19	"HELLO WORLD" message using UART.	38-39	2	4	2	4	12	
08.	10/05/19	PWM module of dutycycle 0-100	40-41	2	4	2	4	12	
	10/05/19	PWM module of dutycycle 50-100.	42-43						
09.	10/05/19	4x4 Hexa Keypad.	44-46.	2	4	2	4	12	
								(12)	
			Valued						
								22/5/19	

Name of Experiment ... INTRODUCTION ..... Date : 25/02/2019 .....

Experiment No. ..... Page No. 01

INTRODUCTION TO ARM CORTEX-M3 PROCESSOR :-

The requirements for higher performance micro controller has been driven globally by the industry's changing needs; for ex- microcontrollers are required to handle more work increasing a products frequency or power.

Microcontrollers are becoming increasingly connected whether by USB, ethernet or wireless radios & hence the processing needed to support these communication channels & advanced peripherals are growing.

General application complexity is on the rise due to more sophosticated convergency of functionality.

ARM Cortex-M3 processor the 1st of the Cortex generation of processor released by ARM.

It address the requirement of 32 bit embedded processor like:

→ Greater performance efficiency.
→ low power consumption
→ Enhanced determinism or improved code density
→ Ease of Use
→ low cost & duration
→ Wide choice of development tools

Name of Experiment ..... Date : .....

Experiment No. ..... Page No. 02

Cortex-M3 processor build on the success of ARM7 processor to deliver that significally easy to program & debug & yet deliver a high processing capabilities.

ARCHITECTURE OF ARM-CORTEX-M3 PROCESSOR:

ARM cortex-M3 is a 32 bit µp. It has a 32 bit data bus, 32 bit register bank & 32 bit memory interface.

It has a Hardvard architecture, which means a seperate instruction & data bus. It allows instruct ions & data access to takeplace at same time.

The instruction & data bus have the same memory space.

For Complex application which requires memory system features where the Cortex-M3 processor has an optional memory protection unit [MPU] & if its requi red external cache can be used.

It supports both little endian & big endian memory system.

It includes a no of fixed internal debugging compo nents. Those components provide debugging operation supports & features such as breakpoint & watchpoint

It has various units like registers, operation mode, Built-in-vectored interrupt controller, memory map, Bus interface, memory protection unit, instruction set, low power consumption, debug supports.

EVOLUTION OF ARM CORTEX-M3 PROCESSOR:

ARM was formed in 1990 as advanced RISC machine & a joint venture of apple computer, Acorn computer group, VLSI technology.

In 1991, Arm introduced the ARM6 processor family & VLSI became the initial license.

Additional companion including Texas instruments, NEC sharp & ST microelectronics, licensed the ARM processor design.

Application of ARM processor into mobile phone, Computer, hard disk, personal digit assistance, home entertainment system etc.

THUMB-2 TECHNOLOGY:

The Thumb-2 technology extended the thumb instruction set architecture into a highly efficient & powerful instruction set.

It delivers significant benefits in terms of ease of use code size, performance.

The relationship b/w thumb instⁿ set is thumb-2 technology & the traditional Thumb.

It is a superset of the processor 16 bit thumb instⁿ set with additional 16 bit instruction along side 32-bit instruction.

In 2003, ARM announced the thumb-2 instⁿ set which is a new superset of thumb instⁿ that contain both 16 bit & 32 bit instruction.

It focus on small memory system devices such as microcontroller & reducing the size of processor.

Cortex-M3 supports only the unaligned data access, a feature previously available only in high.

ADVANTAGES OF ARM CORTEX-M3 PROCESSOR:

The processor was developed to address the demands of digital signal control application.

It offers high efficiency signal processing functionality with low power, low cost & ease to use its benefits

It also satisfies the emerging category of flexible solutions specially targeting the motor control automotive, power management, embedded, audio & industrial automation markets.

APPLICATIONS OF ARM-CORTEX-M3 PROCESSOR:

1. low cost microcontrollers: It is commonly used in consumer products from toys to electrical appliances.
   * The lower power, high performance & ease of use enable embedded developers to migrate to 32 bit system & develop products with ARM architecture.

2. Automotive:- The Cortex-M3 processor has very high performance efficiency & low tendancy allowing it to be used in real time system.

3. Data communicaton: The processor's low power & high efficiency coupled with instruction in thumb-2 for bit-field manipulation make the Cortex-M3 ideal for many communication application such as bluetooth & ZigBee.

4. Industrial Control: In industrial control application, simplicity, fast response & reliability are key factors.

The Cortex M3 provides low interrupt latency & enhanced fault handling features.

5. Consumer products:- In many consumer products, high performance microprocessor is used.
* CORTEX-M3 is a small processor is highly effecient & supports a MPU enabling complex-software to execute while providing robust memory protection.

Introduction to Instruction sets:-

Cortex-M3 processor supports the thumb-2 instn set. It allows 32 bit instn & 32 bit, 16 bit instructions to be used together for high code density & high efficiency.

It is flexible & powerful yet ease to use.

To get the best of both instn set there is a overhead (in terms of both execution time & instruction space to switch between the state (ARM) & thumb codes might used to be compiled seperately in different files.

It increases the complexity & reducing maximum efficiency of Core.

INTRODUCTION TO THE INTERRUPTS:-

Cortex-M3 processor implements a new expectation model, enabling very efficient exception handling.

It has a number of system exception plus a number of external interrupt [IRQs] (external interrupt i/p)

There is no fast interrupts

It supports nested interrupts (a higher priority interrupt can override or point a lower priority interrupt handler) behave just like FIQ.

KEIL µVISION DEVELOPMENT KIT:-

Keil µVision is the complete software development environment for wide range of ARM Cortex M3 based microcontroller devices

It includes µVision IDE, debug gec, ARM C/C++ compiler, essential middleware components

It supports all silicon vendors with over 4000 devices & is easy to learn & use.

The integrated µVision editor includes all standard

features of modern source code editor is also available.

EXPLANATION ABOUT LPC1768 KIT:-

LPC1768 is a header board designed for cortex M3 based LPC1768 from NXP the board is a basic to pin outboard with options of onboard power & USB device, remaining IO pins are taken out on 2.54 mm berg connector. The board has standard JTAG connectivity for debug/programming.

FEATURES:-

* Controller: Cortex-M3 based LPC1768 from NXP package LQFP100
* Clock speed: 12MHz for controller, 32.768 KHz for internal RTC.
* Onboard CAN transreceivers with CAN port thruberg stick
* Power: On board reset & ISP switches.
  * JTAG connectivity option
  * Test LED Via IO pin
  * On board USB device
  * Can be USB or external powerable.

Name of Experiment Part-A Multiplication Date : 15/02/19

Experiment No. 01 Page No. 09

PART-A.

Conduct the following study experiments to learn ALP using ARM Cortex M3 registers using an evaluation board & the required software tool.

a) ALP to Multiply two 16-bit binary numbers.

```
AREA armEX, CODE, READONLY
 EXPORT-- main
 ENTRY
--main LDR R8,= SOURCE
       LDR R0, [R8]
       LDR R1, [R8, #4]
       MUL R7, R0, R1
   Store
     LDR R6, = dest
     STR R7, [R6]

     ST B ST
AREA dest, DATA, READWRITE
 SPACE 32
AREA SOURCE, CODE, READONLY
 DCD 0X02, 0X03.

End.
```

OUTPUT:-

Before execution	After execution
a) R_1 0X00000000 R_8 0X00000000	R_1 = 0X00000037 R_8 = 0X0000026C Address: 0X0000026C 0X0000026C : 0A0000
b) R_1 0X00000000 R_8 0X00000000	R_1 0X0000004E R_8 0X0000026E Address: 0X0000026C 0X0000026C: 0C0000

Name of Experiment: Sum of 10 integers Date: 15/02/19

Experiment No. ..... Page No. 10

(b) ALP to find the sum of first 10 integer.

```
AREA PROGRAM, CODE, READONLY
EXPORT -- main
ENTRY
-- main LDR R8, = SOURCE
        LDR R0, R[8]
        mov R1, #0X00

LOOP
        ADD R1, R0
        SUB R0, #0X01
        CMP R0,#0X00
        BNE LOOP

ST B ST
AREA dest, DATA, READWRITE
SPACE 32
AREA SOURCE, DATA, READONLY
DCD 0XA

end.
```

Sign

15/3/19.

: OUTPUT :-

Before Execution	After execution
② R0 0X00000000 R1 0X00000000	R0 0X00000002 R1 0X00000003 R6 0X10000000 R7 0X00000006 Address: 0X10000000 0X10000000: 06
R0 0X00000000 R1 0X00000000	R0 0X00000004 R1 0X00000003 R6 0X10000000 R7 0X0000000C Address: 0X10000000 0X10000000: 0C

Name of Experiment ..... LED ..... Date : 01/03/19

Experiment No. ..... 1.(a) .....

PART - B.

Conduct the following experiments on a ARM Cortex M3 evaluation based using evaluation version of embedded version and Keil microvision tool.

01. Interface a simple switch and display status through buzzer, LED and Relay.

a) LED:-

```
#include<LPC17xx.H>
void delay(unsigned int count)
{
unsigned int j=0; i=0;
for(j=0; j<count; j++)
{
for(i=0; i<12000; i++);
}
}
int main(void)
{
unsigned int del=900;
delay(10000);
LPC_GPIO0->FIODIR = 0x000000F0;
while(1)
{
LPC_GPIO0->FIOSET = 0x000000E0;
```

```
delay (del);
}
}
LPC_GPIO0 ---> FIODIR = 0x000FF0;
while(1)
{
LPC_GPIO0 ---> FIOSET = 0x00000FF0;
 delay (del);
LPC_GPIO0 ---> FIOCLR = 0x00000FF0;
delay (del);
}
}
```

Output:-

LED blinks at E → 1110 for delay of 200

Name of Experiment ...Relay & buzzer... Date : ...01/02/19...

Experiment No. ...1(b)... Page No. 18

To Display the status of LED Relay & buzzer :-

```
#include <LPC17xx.h>
#define KEY1 (1<<1H)
#define KEY2 (1<<15)
#define KEY3 (1<<16)
#define LED (1<<4)
#define RELAY (1<<25)
#define BUZZ (1<<25)
#define KEY_PIN LPC_GPIO1->FIOPIN
int main(void)
{
LPC_GPIO0->FIODIR |= LED;
LPC_GPIO0->FIOCLR = LED;
LPC_GPIO0->FIODIR |= RELAY;
LPC_GPIO0->FIOCLR = RELAY;
LPC_GPIO3->FIODIR |= BUZZ;
LPC_GPIO3->FIOCLR = BUZZ;
LPC_GPIO1->FIODIR = ~(KEY1|KEY2|KEY3);
LPC_GPIO1->FIOSET = (KEY1|KEY2|KEY3);
 while(1)
{
if(!(KEY_PIN & KEY1))
  LPC_GPIO0->FIOSET = LED;
  else
  LPC_GPIO0->FIOCLR = LED;
 if(!(KEY_PIN & KEY2))
 LPC_GPIO0->FIOSET = RELAY;
 else
```

BGSIT

Name of Experiment ..... Date : .....

Experiment No. ..... Page No. 14

```
LPC_GPIO0 -> FIOCLR = RELAY;
if (!(KEY_PIN & KEY3))
    LPC_GPIO3 -> FIOSET = BUZZ;
  else
    LPC_GPIO3 -> FIOCLR = BUZZ;
  }
}
```

13/3/19

Output:-

LED (P0.11) is ON when KEY1 (P1.14) is pressed.

RELAY (D21) when KEY2 (P1.15) is pressed

BUZZER is ON when KEY3 (P1.16) is pressed.

(2) Interface the DC-motor & rotate it in

a. clockwise direction :

```
#include<LPC17xx.h>
int main(void)
{
LPC_GPIO4 -> FIODIR = 0x30000000;
while(1)
{
LPC_GPIO4 -> FIOPIN = 1<<29;
LPC_GPIO2 -> FIOPIN = 0x00000100;
}
}
```

b. Anticlockwise direction :-

```
#include<LPC17xx.h>
int main(void)
{
LPC_GPIO4 -> FIODIR = 0x30000000;
while(1)
{
LPC_GPIO4 -> FIOPIN = 1<<28;
LPC_GPIO2 -> FIOPIN = 0x00000100;
}
}
```

Output:-

DC motor rotates in clockwise direction

Output:-

DC motor rotates in anticlockwise direction.

Name of Experiment ..... using delay ..... Date : 15/03/19

Experiment No. ..... 2(c) .....

c)

Both clockwise and anticlockwise direction with LPC1768 using delay.

```
#include <LPC17xx.h>
void delay(unsigned int count)
{
unsigned int i=0; j=0;
for(j=0; j<count; j++)
{
for(i=0; i<12000; i++);
}
}
int main(void)
{
unsigned int del=200;
delay(1000);
LPC_GPIO4->FIODIR = 0X30000000;
while(1)
{
LPC_GPIO4->FIOPIN = 1<<29;
LPC_GPIO2->FIOPIN = 0x00000100;
delay(del);
LPC_GPIO4->FIOPIN = 1<<28;
LPC_GPIO2->FIOPIN = 0x00000100;
delay(del);
}
}
}
```

Output :-

Motor rotates in both clockwise and anticlockwise direction.

Name of Experiment ..... Switches ..... Date : ..... 15/02/19 .....

Experiment No. ... 2(d) .....

Interface DC motor with LPC1768 & rotate both clockwise and anticlockwise using switches

```
#include <LPC17xx.H>
# define KEY1 (1<<14)
# define KEY2 (1<<15)
# define KEY_PIN LPC_GPIO1 -> FIOPIN
int main(void)
{
LPC_GPIO4 -> FIODIR = 0X30000000;
while(1)
{
if(!(KEY_PIN & KEY1))
{
LPC_GPIO4 -> FIOPIN = 1<<29;
LPC_GPIO2 -> FIOPIN = 0X00000100;
}
if(!(KEY_PIN & KEY2))
{
LPC_GPIO4 -> FIOPIN = 1<<28
LPC_GPIO2 -> FIOPIN = 0X00000100;
}
}
}
```

Output:

when KEY-1 (p.14) is pressed, motor rotates in clockwise direction

when KEY-2 (p.15) is pressed, motor rotates in anticlockwise direction

Name of Experiment ..... clockwise ..... Date : 15/3/19

Experiment No. 2(c) ..... Page No. 18

Interface DC motor with LPC1768 & rotate clockwise using switches.

```
#include<LPC17xx.h>
#define KEY1 (1<<10)
#define KEY_PIN LPC_GPIO1 -> FIOPIN
int main(void)
{
LPC_GPIO4 -> FIODIR = 0x30000000;
while(1)
{
if(!(KEY_PIN & KEY1))
{
LPC_GPIO4 -> FIOPIN = 1<<29;
LPC_GPIO2 -> FIOPIN = 0x00000100;
}
}
}
```

Output:

When KEY-1 (P.1.4) is pressed, motor rotates in clockwise direction.

Name of Experiment ..... Anti clockwise ..... Date : 15/02/19

Experiment No. 2(f) Page No. 19

Interface DC motor with LPC1768 & rotate anti clockwise using switch:

```
#include<LPC17xx.h>
#define KEY1 (1<<11)
#define KEY_PIN LPC_GPIO1 -> FIOPIN
int main(void)
{
LPC_GPIO4 -> FIODIR = 0x30000000;
while(1)
 {
if(!(KEY_PIN & KEY1))
{
LPC_GPIO4 -> FIOPIN = 1<<29;
LPC_GPIO2 -> FIOPIN = 0x00000100;
}
}
}
```

12/12

Sem

15/3/19

Output:

when KEY1 (P.14) is pressed, motor rotates in anticlockwise direction.

Name of Experiment ..... Stepper motor ..... Date : 29/03/19 .....

Experiment No. ..... 3(a) .....

To Interface stepper motor and rotate in the clockwise direction.

```
#include<LPC17XX.H>
#define MOTOR_CTRL_DIR
void delay()
{
unsigned int j=0; i=0;
for(j=0; j<15; j++)
{
for(i=0; i<12000; i++);
}
}
int main(void)
{
LPC_GPIO1 -> FIODIR = 0x03C00000;
while(1)
{
LPC_GPIO1 -> FIOPIN = k<<25;
delay();
LPC_GPIO1 -> FIOPIN = 1<<24;
delay();
LPC_GPIO1 -> FIOPIN = 1<<23;
delay();
LPC_GPIO1 -> FIOPIN = 1<<22;
delay();
}
}
```

Name of Experiment ...Anticlockwise... Date : ...29/03/19...

Experiment No. ...3b... Page No. 21

To interface steppermotor & rotate in anticlockwise direction.

```
#include<LPC17xx.h>
#define MOTOR_CTRL_DIR
void delay()
{
unsigned int j=0, i=0;
for(j=0; j<15; j++)
{
for(i=0; i<12000; i++);
}
}
int main(void)
{
LPC_GPIO1 -> FIODIR = 0x03C00000;
while(1);
{
LPC_GPIO1 -> FIOPIN = 1<<22;
delay();
LPC_GPIO1 -> FIOPIN = 1<<23;
delay();
LPC_GPIO1 -> FIOPIN = 1<<24;
delay();
LPC_GPIO1 -> FIOPIN = 1<<25;
delay();
}
}
```

Name of Experiment ....Both clockwise & anticlockwise.... 29/03/19.....

Experiment No. ...3C.....

To interface stepper motor & rotate both clockwise & anticlockwise direction.

```
#include <LPC17xx.h>
#define MOTOR_CTRL_DIR
void delay()
{
unsigned int j=0, i=0;
for(j=0; j<20; j++)
{
for(i=0; i<12000; i++);
}
}
int main(void)
{
 int k=0;
unsigned int count=0;
LPC_GPIO1 -> FIODIR = 0x03C00000;
while(count<=100)
{
for(k=0; k<50; k++)
{
LPC_GPIO1 -> FIOPIN = 1<<25;
 delay();
LPC_GPIO1 -> FIOPIN = 1<<24;
 delay();
LPC_GPIO1 -> FIOPIN = 1<<23;
 delay();
LPC_GPIO1 -> FIOPIN = 1<<22;
```

```
delay();
count = count+4;
}
for(k=0; k<50; k++)
{
LPC_GPIO1 -> FIOPIN = 1<<22;
delay();
LPC_GPIO1 -> FIOPIN = 1<<23;
delay();
LPC_GPIO1 -> FIOPIN = 1<<24;
delay();
LPC_GPIO1 -> FIOPIN = 1<<25;
delay();
count = count+4;
}
}
}
```

Name of Experiment ..... Switches ..... Date : 29/03/19

Experiment No. ..... 3d ..... Page No. 24

To Interface stepper motor & rotate both clockwise & anticlockwise using switches:

```
#include <LPC17xx.h>
#define MOTOR_CTRL_DIR
#define KEY1 (1<<14)
#define KEY2 (1<<15)
#define KEY_PIN LPC_GPIO1->FIOPIN
 void delay()
{
unsigned int j=0, i=0;
for(j=0; j<20; j++)
{
for(i=0; i<12000; i++);
 }
  }
int main(void)
{
LPC_GPIO1->FIODIR = 0X03C00000;
while(1)
{
if(!(KEY_PIN & KEY1))
  {
LPC_GPIO1->FIOPIN = 1<<25;
 delay();
LPC_GPIO1->FIOPIN = 1<<24;
delay();
LPC_GPIO1->FIOPIN = 1<<23;
 delay();
LPC_GPIO1->FIOPIN = 1<<22;
```

BGSIT

```
delay();
}
if((KEY==PIN)&&(KEY2))
{
LPC_GPIO1 -> FIOPIN = 1<<22;
delay();
LPC_GPIO1 -> FIOPIN = 1<<23;
delay();
LPC_GPIO1 -> FIOPIN = 1<<24;
delay();
LPC_GPIO1 -> FIOPIN = 1<<25;
delay();
}
}
}
```

Name of Experiment ... angle of 180° Date : 29/02/19

Experiment No. 3f Page No. 27

f. To interface Stepper motor & rotate it at an angle of 180°

```
#include<LPC17xx.h>
#define MOTOR_CTRL_DIR
void delay()
{
unsigned int j=0, i=0;
for(j=0; j<20; j++)
{
for(i=0; i<12000; j++);
}
}
int main(void)
{
unsigned int count=0;
LPC_GPIO1->FIODIR=0x03C00000;
while(count<=100)
{
LPC_GPIO1->FIOPIN=1<<25;
delay();
LPC_GPIO1->FIOPIN=1<<24;
delay();
LPC_GPIO1->FIOPIN=1<<23;
delay();
LPC_GPIO1->FIOPIN=1<<22;
delay();
count=count+4;
}
}
```

Name of Experiment ..... angle of 360° ..... Date : ..... 29/02/19 .....

Experiment No. ..... 3g ..... Page No. 28

3g. To Interface stepper motor and rotate it at an angle of 360°

```
#include<LPC17xx.h>
#define MOTOR_CTRL_DDR
void delay()
{
unsigned int j=0; i=0;
for(j=0; j<20; j++)
{
for(i=0; i<12000; i++);
}
}
int main(void)
{
unsigned int count=0;
LPC_GPIO1 -> FIODIR = 0x03C00000;
while(count<=200)
{
LPC_GPIO1 -> FIOPIN = 1<<25;
delay();
LPC_GPIO1 -> FIOPIN = 1<<24;
delay();
LPC_GPIO1 -> FIOPIN = 1<<23;
delay();
LPC_GPIO1 -> FIOPIN = 1<<22;
delay();
Count = Count+4;
}
}
```

BGSIT

Name of Experiment .Controlling using switches Date : 29/03/19

Experiment No. (3g) Page No. 29

Q. Rotation of Stepper motor and controlling using switches.

```
#include<LPC17xx.h>
#define MOTOR_CTRL_DIR LPC_GPIO1->FIODIR
#define MOTOR_CTRL_SET LPC_GPIO1->FIOSET
#define MOTOR_CTRL_CLR LPC_GPIO1->FIOCLR
#define MOTOR_MASK 0x03C00000
#define KEY_DIR LPC_GPIO1->FIODIR
#define KEY_SET LPC_GPIO1->FIOSET
#define KEY_CLR LPC_GPIO1->FIOCLR
#define KEY_PIN LPC_GPIO1->FIOPIN
#define KEY_START (1<<14)
#define KEY_STOP (1<<15)
#define KEY_INC (1<<16)
#define KEY_DEC (1<<17)
#define KEY_CW (1<<18)
#define KEY_CCW (1<<19)
void delay()
{
unsigned int j=0,i=0;
for(j=0;j<10;j++)
 {
 for(i=0;j<12000;i++);
 }
}
void motor_write(uint32_t data)
{
uint32_t temp;
```

BGSIT

```
temp = (data << 22) & MOTOR_MASK;
MOTOR_CTRL_CLR |= MOTOR_MASK;
MOTOR_CTRL_SET |= temp;
}
int main (void)
{
unsigned int del = 10;
uint32_t stpval = 0x01;
unsigned char dir = 0;
unsigned char run = 1;
MOTOR_CTRL_DIR |= MOTOR_MASK;
Motor_write (stpval);
while(1)
{
if (!(KEY_PIN & KEY_START))
   run = 1;
if (!(KEY_PIN & KEY_STOP))
   run = 0;
if (!(KEY_PIN & KEY_CW))
   dir = 0;
if (!(KEY_PIN & KEY_CCW))
   dir = 1;
if (!(KEY_PIN & KEY_ENC))
  if (del != 0)  del = del - 1;
 if (run == 1)
{
 if (dir == 0)
{
```

```
if(stpval==8)
stpval=1;
else
stpval<<=1;
}
else
{
if(stpval==1)
stpval=8;
else
stpval>>=1;
}
motor_write(stpval);
delay(del);
}
}
}
```

12/12

14/19

Output:

when KEY-1 (p.14) is pressed, motor start rotating

when KEY-2 (p.15) is pressed, motor stops rotating

when KEY-3 (p.16) is pressed, motor speed increases

when KEY-4 (p.17) is pressed, motor speed decreases

when KEY-5 (p.18) is pressed, motor rotates in clockwise direction

when KEY-6 (p.19) is pressed, motor rotates in anticlockwise direction

(b)

from f-0:-

```
#include <LPC17xx.H>
unsigned char data7[] = {0x88, 0xEB, 0x4C, 0x49, 0x08,
        0x19, 0x18, 0xCB, 0x8, 0x9, 0xA, 0x38, 0x00, 0x68,
        0x16, 0x1E };
  int main(void)
{
 unsigned int i, j, count = 0x0F;
LPC_GPIO2 -> FIODIR = 0x000000FF;
LPC_GPIO2 -> FIOPIN = 0x000000FF;
LPC_GPIO1 -> FIODIR = 1<<26;
LPC_GPIO1 -> FIOPIN = ~(1<<26);
while (1)
{
 if(count>0x0F)
 count = 0;
for (i=0; i<10000; i++)
{
LPC_GPIO2 -> FIOPIN = data7[count];
LPC_GPIO1 -> FIOSET = 0x04000000;
for(j=0; j<500; j++);
LPC_GPIO1 -> FIOCLR = 0x04000000;
}
--count;
}
}
```

Output

g	f	a	b	p	c	d	e	Hexadigit	7segment
0	0	0	1	1	1	1	0	0X88	
0	0	0	1	1	1	0	0	0XEB	e
0	0	1	0	1	0	0	0	0X4C	d
1	0	0	1	1	1	0	0	0X49	c
0	0	1	1	1	0	0	0	0X2B	b
0	0	0	0	1	0	1	0	0X19	a
0	0	0	0	1	0	0	1	0X18	9
0	0	0	0	1	0	0	0	0XCB	8
1	1	0	0	1	0	1	1	0X8	7
0	0	0	1	1	0	0	0	0X9	6
0	0	0	1	1	0	0	1	0XA	5
0	0	1	0	1	0	1	1	0X38	4
0	1	0	0	1	0	0	1	0X9C	3
0	1	0	0	1	1	0	0	0X68	2
1	1	1	0	1	0	1	1	0X1C	1
1	0	0	0	1	0	0	0	0X1E	0

4C.

7 segment LED Interface to display BGSIT-ECE:

```
#include<LPC17XX.H>
unsigned char data 7[] = {0x38, 0x9, 0x19, 0xBE,
                          0x3C, 0x7F, 0x1C, 0x9C, 0x1C};
int main (void)
{
 unsigned int i, j;
unsigned int count=0;
LPC_GPIO2 -> FIODIR = 0x000000FF;
LPC_GPIO2 -> FIOPIN = 0x000000FF;
LPC_GPIO1 -> FIODIR = (1<<26);
LPC_GPIO1 ->FIOPIN = ~(1<<26);
while(1)
{
if(count>0) count=0;
for(i=0; i<10000; i++)
{
LPC_GPIO2 -> FIOPIN = data 7[count];
LPC_GPIO1 -> FIOSET = 0x04000000;
for(j=0; j<500; j++);
LPC_GPIO1 -> FIOCLR = 0x04000000;
}
count++;
}
}
```

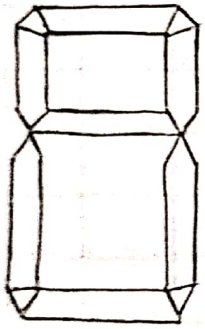
12/12 Sec

5/11/19

Output :-

g	f	a	b	p	c	d	e	Hexadigit	7segment
0	0	1	1	1	0	0	0	0x38	b
0	0	0	0	1	0	0	1	0x09	g
0	0	0	1	1	0	0	1	0x19	s
0	0	0	1	1	0	0	1	0x19	s
1	0	1	1	1	1	1	0	0xbe	i
0	0	1	1	1	1	0	0	0x3C	t
0	0	0	1	1	1	0	0	0x1C	e
1	0	0	1	1	1	0	0	0x9C	c
0	0	0	1	1	1	0	0	0x1C	

Theory :-



A 7segment display is a form of electronic display device for displaying decimal numerals that is an alternative to this more complex dot matrix display.

Seven segment display are widely used in digital clocks electronic meters, basic calculators and electronic devices that display numerical information.

Name of Experiment ....Square wave.....
Experiment No. (5)
Date : 02/05/19
Page No. 35

Interface DAC and generate square wave and Triangular wave.

(5a) To generate square wave using DAC:

```
#include <LPC17xx.h>
int main (void)
{
uint32_t m;
LPC_PINCON -> PINSEL1 = 0x00200000;
while(1)
{
LPC_DAC -> DACR = (512<<6);
for(m=12000; m>1; m--);
LPC_DAC -> DACR = (1023<<6);
for(m=12000; m>1; m--);
}
}
```

Output:

Theoretical output:-

W.K.T,

$$V_{out} = \frac{V_{ref} \times Valmax}{2^N}$$

$$V_{out} = \frac{V_{ref} \times Valmax}{2^{10}}$$

$$= \frac{3.3 \times 1023}{1024}$$

$$\langle V_{out} = 3.3V \rangle$$

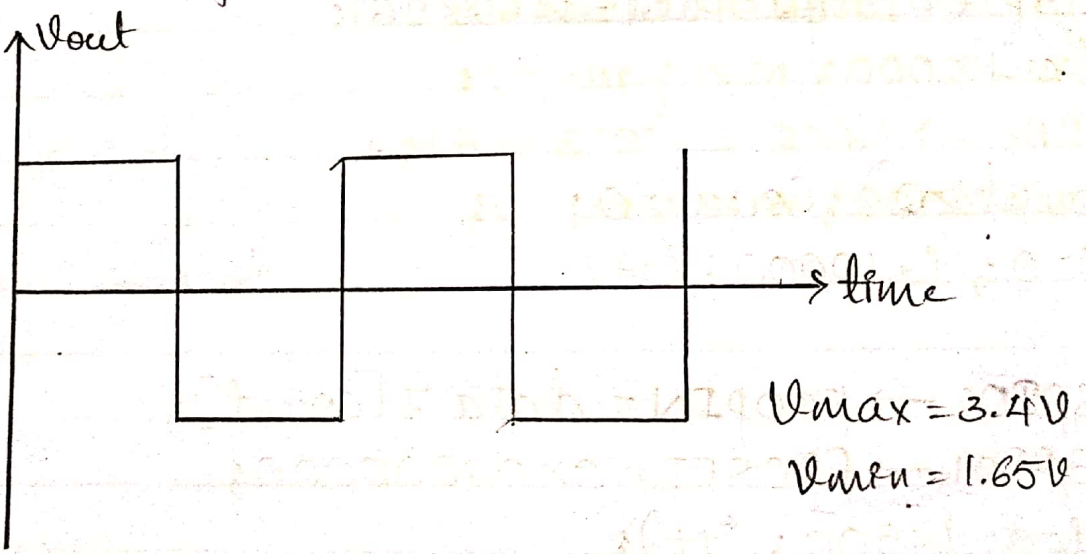
$$V_{min} = \frac{V_{ref} \times Valmin}{2^N}$$

$$= \frac{V_{ref} \times Valmin}{2^{10}}$$

$$= \frac{3.3 \times 512}{1024}$$

$$\langle V_{min} = 1.65V \rangle$$

Practical output:



$V_{max} = 3.4V$

$V_{min} = 1.65V$

Name of Experiment ... triangle wave ... Date : 03/05/19

Experiment No. ... 5(b) ... Page No. 36

5b

To generate triangle wave using DAC:-

```
#include <LPC17xx.H>
int main (void)
{
uint 32_t i=0;
uint 32_t m;
LPC_PINCON -> PINSEL 1 = 0X00200000;
while (1)
{
for (i=0; i<1023; i++)
{
LPC_DAC -> DACR = (i<<16);
for (m=12000; m>1; m--);
}
for (i=1023; i>0; i--)
{
LPC_DAC -> DACR = (i<<6);
for (m=12000; m>1; m--);
}
}
}
```

12/12

Sec 27/4/19

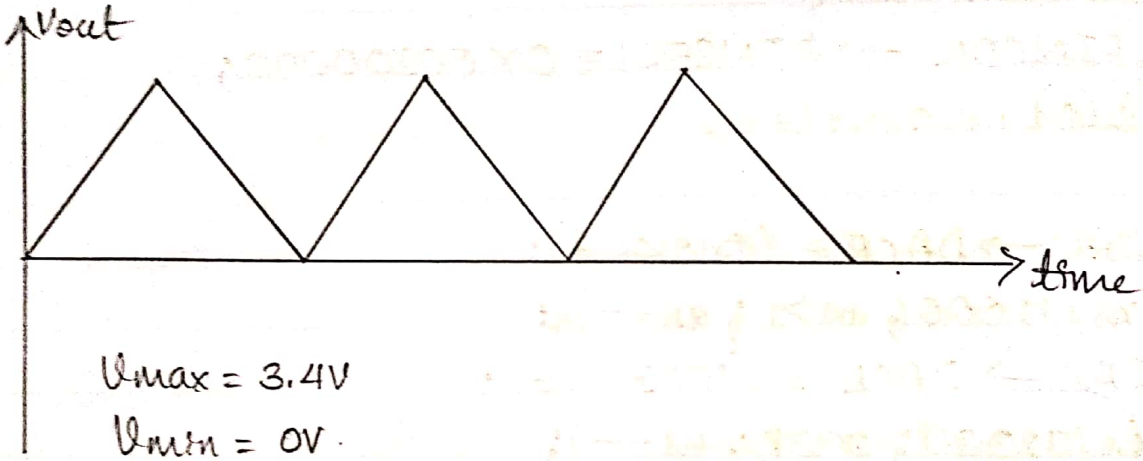
Output :-

Theoretical output :

$$V_{out} = \frac{V_{ref} \times V_{al}}{2^N} = \frac{3.3 \times 1023}{1024}$$

$$V_{out} = V_{max} = 3.3V$$

Practical output :-



$V_{max} = 3.4V$

$V_{min} = 0V$

Name of Experiment ..... toggle an LED ..... Date : 03/05/19

Experiment No. ..... 06 .....

6. Demonstrate the use of an external interrupt to toggle an LED ON/OFF:

```
#include<LPC17xx.h>
void EINT0_IRQHandler(void)
{
LPC_SC->EXTINT= (1<<0);
lpc_GPIO0->FIOPIN^= (1<<4);
}
void EINT1_IRQHandler(void)
{
LPC_SC->EXTINT= (1<<1);
LPC_GPIO0->FIOPIN^= (1<<5);
}
int main()
{
LPC_SC->EXTINT= (1<<0)|(1<<1);
LPC_PINCON->PINSEL4= (1<<20)|(1<<22);
LPC_SC->EXTMODE = (1<<0)|(1<<1);
LPC_SC->EXTPOLAR = (1<<0)|(1<<1);
LPC_GPIO0->FIODIR = ((1<<4)|(1<<5));
LPC_GPIO0->FIOPIN = 0x00;
NVIC_EnableIRQ(EINT0_IRQn);
NVIC_EnableIRQ(EINT1_IRQn);
while(1)
{
}
}
```

12/12 Sun 3/5/19

Output:-

when interrupt 0 is pressed p.4 ON
when interrupt 1 is pressed p.5 ON

Theory:-

External Interrupt:- An external interrupt in a computer system. Interrupt that happens as a result of outside interference. whether that's from the user, from peripherals, from other hardware device or through a network. These are different than internal interrupts that happens automatically as the machine reads through program instruction.

External interrupt control in ARM LPC1768:-

The external interrupt input of the LPC1768 is a very important functionality of software. It is one of the many function of the symbol control but that are not related to a specific peripheral device. These are 4 pins that can be set to functions as a external interrupt.

The external interrupt mode register is a 32bit register that is used to select the level of edges sensitivity of the respective interrupt pins.

Name of Experiment ..... HELLO WORLD ..... Date : ..02/05/19.....

Experiment No. ..07..... Page No. 38

07. Display "HELLO WORLD" message using internal UART

```
#include <LPC17xx.h>
#define FOSC 12000000
#define FCCLK (FOSC*8)
#define FCCO (FCCLK*3)
#define FPCLK (FCCLK/4)
#define UART0_BPS 9600
int UART0_SendByte (int UCData)
{
while(!(LPC_UART0->LSR & 0x20));
return (LPC_UART0->THR = UCData);
}
void UART0_SendString(unsigned char*s)
{
while (*s!=0)
UART0_SendByte (*s++);
}
int main(void)
{
unsigned int ref;
unsigned int Usfdiv;
LPC_PINCON->PINSEL0|=(1<<4);
LPC_PINCON->PINSEL0|=(1<<6);
LPC_UART0->LCR=0x83;
Usfdiv=(FPCLK/16)/UART0_BPS;
LPC_UART0->DLM=Usfdiv/256;
LPC_UART0->DLL=Usfdiv%256;
```

BGSIT

```
LPC_UART0 -> LCR = 0x03;
LPC_UART0 -> FCR = 0x06;
UART0_SendString ("HELLO WORLD");
UART0_SendByte (0x0D);
UART0_SendByte (0x0A);
while(1)
{
while(!(LPC_UART0 -> LSR & 0x01));
 rep = LPC_UART0 -> RBR;
while(!(LPC_UART0 -> LSR & 0x20));
 LPC_UART0 -> THR = rep;
}
}
```

Theory:- UART module & register. LPC1768 has 4-UART's numbering 0-3. Similarly, the pins are also named as RxD0-RxD3 & TxD0-TxD3. As the LPC1768 pins are multiplexed for multiple functionalities first they have to be configured as UART pins.

Below table shows the multiplexed UART pins.

Port pin	Pin number	PINSEL FUNC_0	PINSEL FUNC_1	PINSEL FUNC_2	PINSEL FUNC_3
P0.02	98	GPIO	TXD0	AD0[7]	
P0.03	99	GPIO	RXD0	AD0[6]	
P2.0	48	GPIO	PWM[1]	TXD1	
P2.1	49	GPIO	PWM[0]	RXD1	
P0.10	62	GPIO	TXD2	SDA	MAT3[0]
P0.11	63	GPIO	RXD2	SCL2	MAT3[1]
P0.0	82	GPIO	CAN1_Rx	TXD3	SDD1
P0.1	85	GPIO	CAN1_Tx	RXD3	SCL1

Output:-

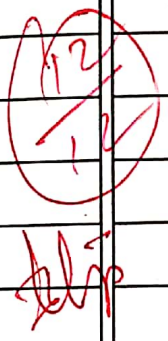
Input:- HELLO WORLD

Output:- HELLO WORLD.

Using the internal PWM module of LPC1768 generate PWM and vary its duty cycle 0 to 100.

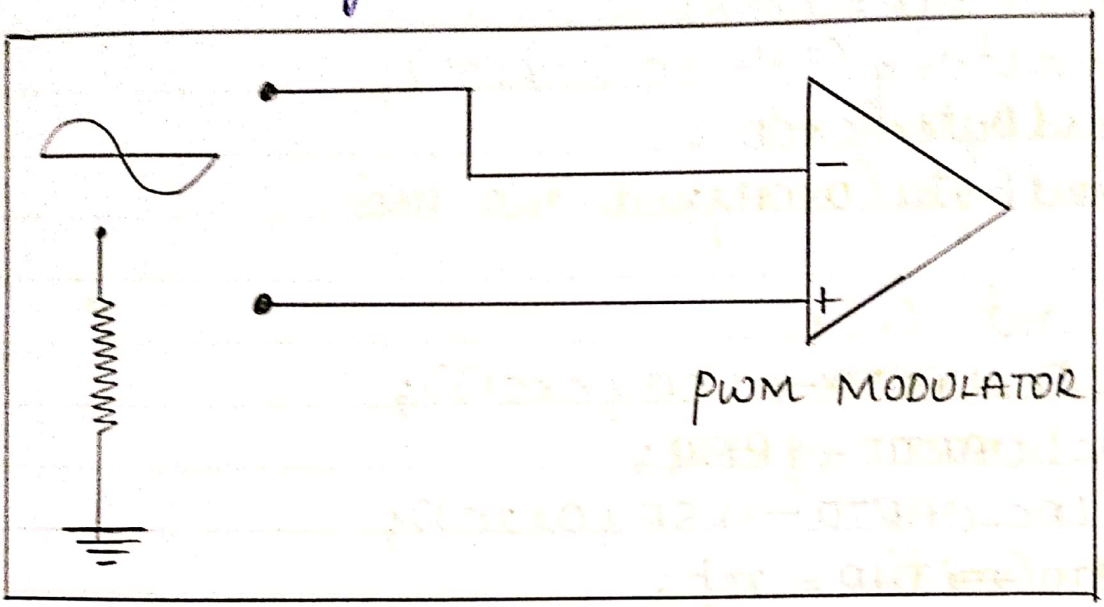
```
#include<lpc17xx.h>
void delay_ms(unsigned int ms)
{
unsigned int i,j;
for(i=0; i<ms; i++)
for(j=0; j<50000; j++);
}
int main(void)
{
int dutycycle;
LPC_PINCON->PINSEL7 = 3<<20;
LPC_PWM1->TCR = (1<<0);
LPC_PWM1->PR = 0X0;
LPC_PWM1->MCR = (1<<1);
LPC_PWM1->MR0 = 100;
LPC_PWM1->MR3 = 0;
LPC_PWM1->LER = 1<<3;
LPC_PWM1->PCR = 1<<11;
while(1)
{
for(dutycycle=0; dutycycle<100; dutycycle++)
{
LPC_PWM1->MR3 = dutycycle;
LPC_PWM1->LER = 1<<3;
  delay_ms(5);
```

```
{
for (dutycycle=100; dutycycle>0; dutycycle--)
{
LPC_PWM1->MR3 = dutycycle;
LPC_PWM1->LER = 1<<3;
 delay_ms(5);
}
}
}
```



THEORY:-

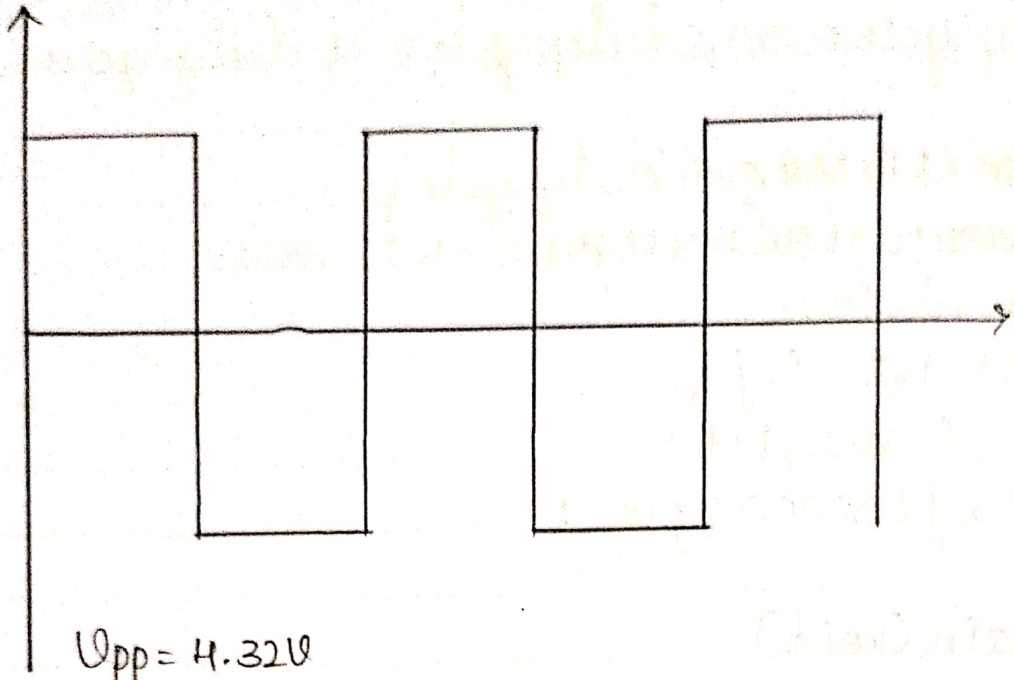
Circuit diagram:-



Pulse width modulation

Pulse width modulation is a modulation technique used to encode a message into a pulsing signal although this modulation technique can be used to encode information for transmission. This makes use it allows the control of the power supplied to electrical device specially to initial load such as motor.

Output :-



$V_{pp} = 4.32V$

$V_{avg} = 2.35V$

freq = 2.47.5kHz

Dutycycle = 67.73%

Ripetime = 1.572 μsec.

Name of Experiment ....PWM.... Date : ...10/05/19....

Experiment No. ....8(b).... Page No. 42

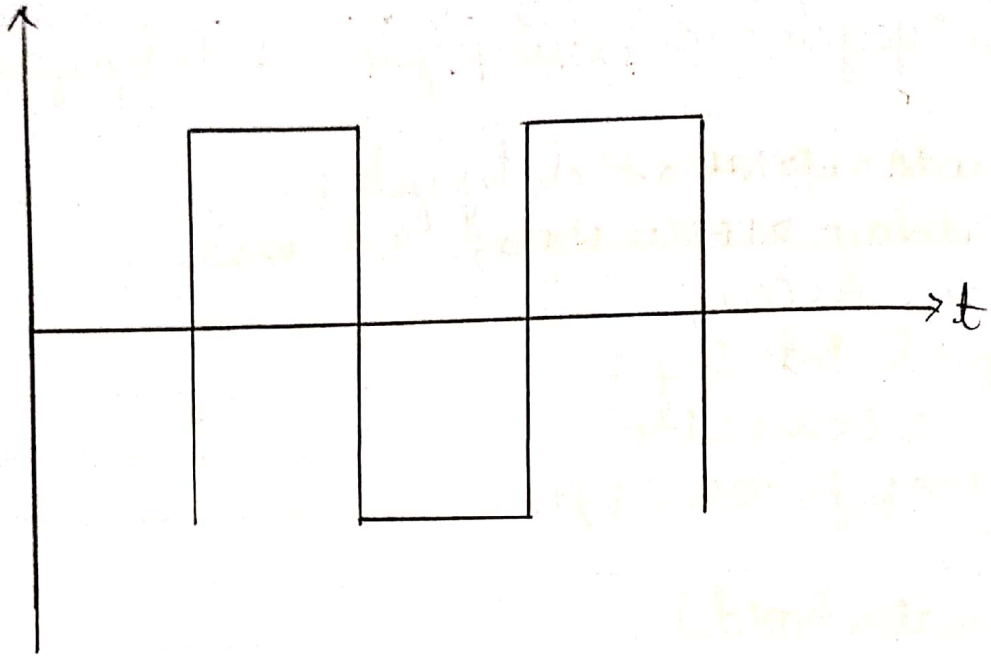
b) Using the PWM module of LPC1768 generate PWM & vary its dutycycle 50-100.

```
#include<LPC17xx.H>
void delay_ms(unsigned int ms)
{
unsigned int i, j;
for(i=0; i<ms; i++)
for(j=0; j<50000; j++)
}
int main(void)
{
int dutycycle;
LPC_PINCON->PINSEL3 = 3<<20;
LPC_PWM1->TCR = (1<<0);
LPC_PWM1->PR = 0x0;
LPC_PWM1->MCR = (1<<1);
LPC_PWM1->MR0 = 100;
LPC_PWM1->MR3 = 50;
LPC_PWM1->LER = 1<<3;
LPC_PWM1->PCR = 1<<11;
while(1)
{
for(dutycycle=0; dutycycle<100; dutycycle++)
{
LPC_PWM1->MR3 = dutycycle;
LPC_PWM1->LER = 1<<3;
delay_ms(5);
}
```

BGSIT

```
for(dutycycle=100; dutycycle>0; dutycycle--)
{
LPC_PWM1->MR3 = dutycycle;
LPC_PWM1->LER = 1<<3;
delay_ms(5);
}
}
```

Output:-



$V_{pp} = 2.56V$

$V_{avg} = 1.57V$

freq = 247.5 KHz

Duty cycle = 56.15%

Rise time = 1.57 μsec

Name of Experiment ....HEX Keypad..... Date : ...10/05/19.....

Experiment No. ..09.(a)..... Page No. 44

Interface a 4x4 hex keypad & display the keycode on 7segment display.

```
#include<LPC17xx.h>
#include "lcd.h"
#define col1 (1<<14)
#define col2 (1<<15)
#define col3 (1<<16)
#define col4 (1<<17)
#define Row1 (1<<18)
#define Row2 (1<<19)
#define Row3 (1<<20)
#define Row4 (1<<21)
#define COLMASK (col1|col2|col3|col4)
#define ROWMASK (Row1|Row2|Row3|Row4)
#define KEYDIR LPC_GPIO1->FIODIR
#define KEYSET LPC_GPIO1->FIOSET
#define KEY_CLR LPC_GPIO1->FIOCLR
#define KEY_PIN LPC_GPIO1->FIOPIN
void col_write (unsigned char data)
{
unsigned int temp=0;
temp=(data<<14)&COLMASK;
KEY_CLR1=COLMASK;
KEY_SET|=temp;
}
int main(void)
{
```

```
unsigned char KEY, i;
unsigned char rval[] = {0x7, 0xB, 0xD, 0xE, 0x0};
unsigned char KeypadMatrix[] = {'C', '8', '4', '0',
                                'D', '9', '5', '1', 'E', 'A', '6', '2', 'F',
                                'B', '7', '3'};

int LCD();
LPC_GPIO1->FIODIR1 = (COL1|COL2|COL3|COL4);
KEY_DIR = ~(ROWMASK);
lcd_putstring16(0, "Press HEX_KEYS..");
lcd_putstring16(1, "Key pressed = ");
LPC_GPIO3->FIODIR = 0x02000000;
  KEY=0;
for(i=0; i<4; i++)
{
col_write(rval[i]);
if(!(KEY_PIN & ROW1))
  break;
  KEY++;
if(!(KEY_PIN & ROW2))
  break;
  KEY++;
if(!(KEY_PIN & ROW3))
  break;
  KEY++;
if(!(KEY_PIN & ROW4))
  break;
  KEY++;
}
```

```
if(KEY==0X10)
lcd_putstring16(1,"Keypressed=");
else
lcd_gotoxy(1,14);
lcd_putchar(Keypad Matrix[KEY]);
}
}
}
```

Name of Experiment ....Hex Keypad..... Date : ....10/05/19.....

Experiment No. ....9(b)..... Page No.

Interface a 4x4 Hexkeypad & display the key code on 7 segment display.

```
#include<LPC17xx.h>
#define dig1 (1<<26)
unsigned char data 7[] = {0x88, 0xeb, 0x4c,
    0x2b, 0x19, 0x18, 0xcb, 0x9, 0x0, 0x38, 0x9c,
    0x68, 0x1c, 0x1e};
#define COL1 (1<<14)
#define COL2 (1<<15)
#define COL3 (1<<16)
#define COL4 (1<<17)
#define ROW1 (1<<18)
#define ROW2 (1<<19)
#define ROW3 (1<<20)
#define ROW4 (1<<21)
#define COLMASK (COL1|COL2|COL3|COL4)
#define ROWMASK (ROW1|ROW2|ROW3|ROW4)

#define KEY_PIN LPC_GPIO1->FIOPIN
void col_write (unsigned char data)
{
  unsigned int temp=0;
  temp=(data<<14) & COLMASK;
  LPC_GPIO1->FIOCLR1 = COLMASK;
  LPC_GPIO1->FIOSET1 = temp;
}
int main (void)
{
```

Name of Experiment ..... Date : .....

Experiment No. ..... Page No.

```
if(KEY==0X10)
LPC_GPIO2 -> FIOPIN = 0Xff;
  else
{
LPC_GPIO2 -> FIOPIN = data =[KeypadMatrix[key]];
}
}
}
```

Results:-

4X4 keypad is interfaced and 7 segment display is displayed.

12/5/19.