EE447 Pre Liminary-Work Lab # 3

Q1. Write a C function that sends a GPIO port the necessary signals to demonstrate the Full Step Mode in both directions (clockwise or counterclockwise).

Ans. The initfunc() function initializes the GPIO pins and the SysTick timer. It configures specific GPIO pins (PB4-PB7) as outputs and sets up the SysTick timer to generate interruptions every millisecond. These interrupts will be used to keep track of steps and control the motor's movement. The SysTick_Handler() function is an interrupt service routine. It increments the mcount variable each time the SysTick interrupt occurs. This variable acts as a counter to keep track of the motor's position within a step cycle.

The stepfulldir() function controls the motor movement based on the mcount value and the specified direction. It sets specific GPIO pins high or low to activate the motor coils in a particular sequence. This sequence determines the steps taken by the motor and hence the direction of rotation. In the main() function, the system is initialized using initfunc(), and then it enters an infinite loop where stepfulldir() is continuously called with an argument indicating a clockwise rotation. This setup ensures that the motor rotates continuously in one direction depending on the value in the dir variable.

Code:

```
A#include "TM4C123GH6PM.h"

// Code for Clockwise and anti Clock wise Direction
int mcount=0;

void initfunc(void)
    {
        SYSCTL→RCGCGPIO |= 0x2F; // turn on bus clock for GPIOB
        GPIOB→DIR       |= 0xF0; // set PB4-PB7 pin as a digital output pin
        GPIOB→DEN       |= 0xFF; // Enable PB0-PB7 pin as a digital pin
        GPIOB→AFSEL     &=~0xFF;   // Disable Alternate Function
        SysTick→LOAD     = 4000; // one milli second delay relaod value
        SysTick→CTRL     = 3;    // enable counter , interrupt and select system bus clock
        SysTick→VAL      = 0;      // initialize current value register
  }

void SysTick_Handler(void)
    {
        mcount++;
    }

void stepfulldir(int dir)
    {
    if (dir==1) // Check for Clockwise or Anticlockwise
        {
            if (mcount == 7) // Check for count and set corresponding LED high and all others
LOW
                {
                    GPIOB→DATA &= ~0xF0;
                    GPIOB→DATA |= 0x10;
                }
            if (mcount == 5)
                {
                    GPIOB→DATA &= ~0xF0;
```

```
                    GPIOB->DATA |= 0x20;
                }
            if (mcount == 3)
                {
                    GPIOB->DATA &= ~0xF0;
                    GPIOB->DATA |= 0x40;
                }
            if (mcount == 1)
                {
                    GPIOB->DATA &= ~0xF0;
                    GPIOB->DATA |= 0x80;
                }
            if (mcount > 8)
                {
                    mcount = 0;
                }
        }
    else
        {
            if (mcount == 1)
                {
                    GPIOB->DATA &= ~0xF0;
                    GPIOB->DATA |= 0x10;
                }
            if (mcount == 3)
                {
                    GPIOB->DATA &= ~0xF0;
                    GPIOB->DATA |= 0x20;
                }
            if (mcount == 5)
                {
                    GPIOB->DATA &= ~0xF0;
                    GPIOB->DATA |= 0x40;
                }
            if (mcount == 7)
                {
                    GPIOB->DATA &= ~0xF0;
                    GPIOB->DATA |= 0x80;
                }
            if (mcount > 8)
                {
                    mcount = 0;
                }
        }
}
int main()
    {
        initfunc();
```

```
        while(1) // Run Continuously
        {
            stepfulldir(1); // 1 for Clockwise , 0 for Anti Clock wise
        }
    }
```

Q2. The following diagram indicates the connections between the TM4C123G, ULN2003A's board, 4x4 Keypad Module and stepper motor.

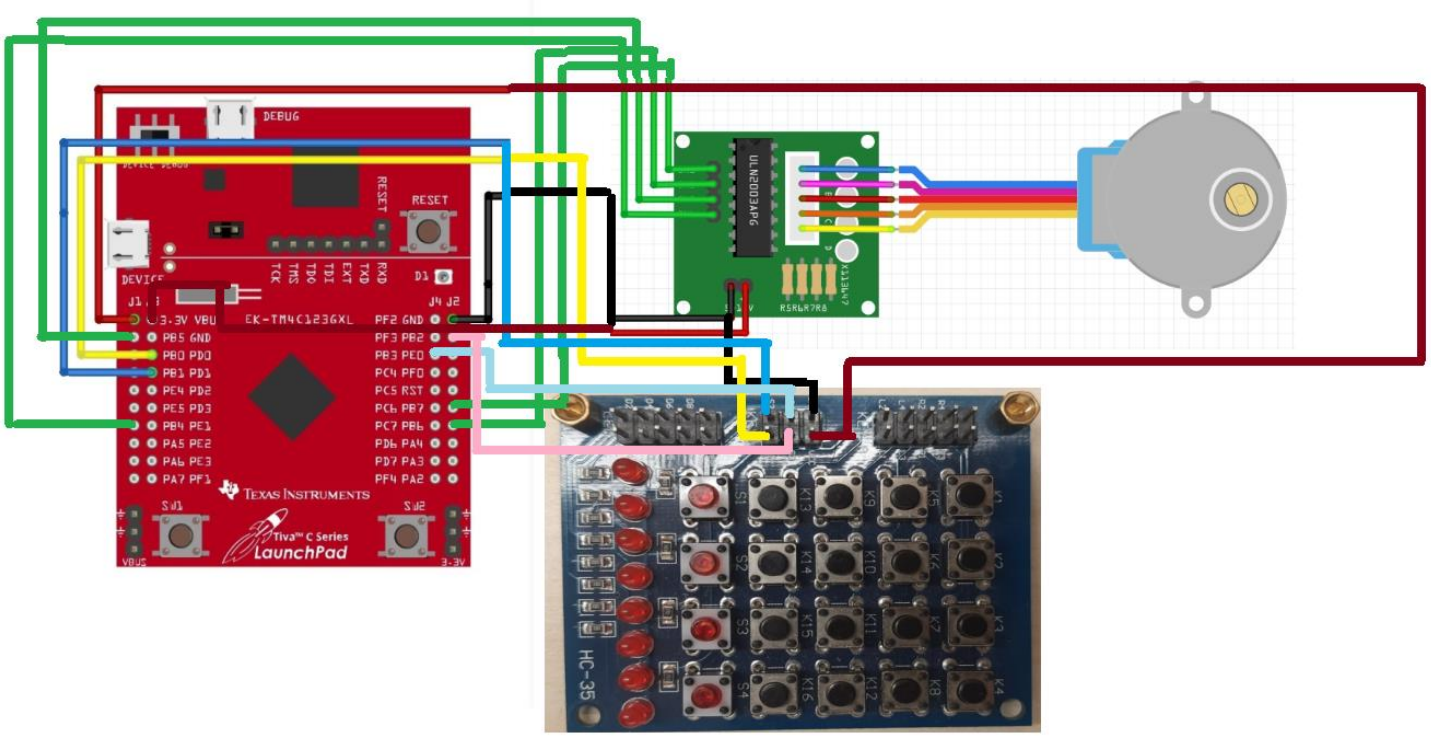

Fig 2. Swithes S1 and S2 are used (S3 and S4 unused)

The table below shows the connections in tabular form that can be traced in the picture:

From Pin	To Pin
PB0	S1-Keypad
PB1	S2-Keypad
PB2	S3-Keypad
PB3	S4-Keypad
PB4	IN1-ULN2003A
PB5	IN2-ULN2003A
PB6	IN3-ULN2003A
PB7	IN4-ULN2003A
VBUS	V+, ULN2003A
VCC	VCC- Keypad
GND	GND Keypad and ULN2003

EE447 Pre Liminary-Work Lab # 3

Code:

```
#include "TM4C123GH6PM.h"
#define BUTTONNUMBER 4
#define INPUTMASK 0x0F
int mcount=0;

void initfunc(void)
{
    SYSCTL→RCGCGPIO |= 0x2F;
    GPIOB→DIR       |= 0xF0;
    GPIOB→DEN       |= 0xFF;
    GPIOB→AFSEL     &=~0xFF;


    SysTick→LOAD     = 5000;
    SysTick→CTRL     = 3;
    SysTick→VAL      = 0;
}
void SysTick_Handler(void)
{
     mcount++;
}

unsigned char getbutton(void)
{
    unsigned char input;
    return GPIOB→DATA & INPUTMASK;
}



unsigned char debounced_button()
{
  unsigned char button,prevbutton;
    button = 0x8F;
    while(1)
    {
        int syscount;
        prevbutton = getbutton();
        syscount = mcount;

        while(syscount == mcount)
        {
            button = getbutton();
        }
        if (button == prevbutton)
        {
```

```
            return button;
        }
    }
    }


void stepfulldir(int dir)
{
    if (dir==1)
    {
        if (mcount == 7)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x10;
        }
        if (mcount == 5)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x20;
        }
        if (mcount == 3)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x40;

        }
        if (mcount == 1)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x80;
        }
        if (mcount > 8)
        {
            mcount = 0;
        }
    }
    else
    {
        if (mcount == 1)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x10;
        }
        if (mcount == 3)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x20;
        }
```

```
        if (mcount == 5)
        {
            GPIOB→DATA &= ~0xF0;
            GPIOB→DATA |= 0x40;
        }
        if (mcount == 7)
        {
            GPIOB→DATA &= ~0xF0;
            GPIOB→DATA |= 0x80;
        }
        if (mcount > 8)
        {
            mcount = 0;
        }
    }
}
int main()
{
    initfunc();
    while(1)
    {
        unsigned char real_button;
        real_button = debounced_button();
        if (real_button == 0x0D)
        {
            stepfulldir(1);
        }
        if (real_button == 0x0E)
        {
            stepfulldir(0);
        }
    }
}
```

Explanation:

The initialization function initfunc() remains similar to the previous code, setting up the GPIO pins and configuring the SysTick timer to generate interrupts.The SysTick_Handler() function increments the mcount variable, just as before, to keep track of the motor's position within a step cycle.

A new function, getbutton(), reads the status of the buttons connected to the GPIO pins and returns the button's state. It masks the input to obtain only the lower 4 bits (representing 4 buttons) of the GPIOB→DATA register. The debounced_button() function is introduced to handle button debouncing, a technique used to ensure clean and stable button inputs. It reads the button state using getbutton() and waits for changes while continuously checking the mcount value. Once a change is detected and remains consistent for a short duration, it considers the button input as stable and returns the debounced button state.

EE447 Pre Liminary-Work Lab # 3

The stepfulldir() function, similar to the previous code, controls the motor's movement based on the mcount value and the specified direction. However, instead of relying on a constant direction, it now takes input from the debounced button presses. In the main() function, it continuously checks for debounced button inputs using debounced_button(). Depending on the specific button press, it calls stepfulldir() with either a clockwise (1) or anti-clockwise (0) direction for motor rotation.

Q3. The following code can vary the speed as well as change the direction of the stepper motor

Code:

```
#include "TM4C123GH6PM.h"

#define BUTTONNUMBER 4
#define INPUTMASK 0x0F
int mcount=0;

void initfunc(void)
{
    SYSCTL→RCGCGPIO |= 0x2F; // turn on bus clock for GPIOB
    GPIOB→DIR       |= 0xF0; // set PB4-PB7 pin as a digital output pin
    GPIOB→DEN       |= 0xFF; // Enable PB0-PB7 pin as a digital pin
    GPIOB→AFSEL     &=~0xFF; // Disable Alternate Function
    SysTick→LOAD     = 5000; // one milli second delay relaod value
    SysTick→CTRL     = 3;    // enable counter , interrupt and select system bus clock
    SysTick→VAL      = 0;    // initialize current value register
}
void SysTick_Handler(void)
{
    mcount++;
}

unsigned char getbutton(void) // function for getting push button input
{
    unsigned char input;
    return GPIOB→DATA & INPUTMASK;
}



unsigned char debounced_button() // debouncing problem solving
{
  unsigned char button,prevbutton;
    button = 0x8F;
    while(1)
    {
        int syscount;
        prevbutton = getbutton();
        syscount = mcount;

        while(syscount == mcount)
        {
```

```
            button = getbutton();
        }
        if (button == prevbutton)
        {
            return button;
        }
    }
}

unsigned char released_button() // function to check whether the button has released or not
{
unsigned char button,release;
    button = debounced_button();
    while(1)
    {
        release = getbutton();
        if (release == 0x0F) // by comparing all swithes to high
        {
            return button;
        }
    }
}


void stepfulldir(int dir)
{
    if (dir==1) // Check for Clockwise or Anticlockwise
    {
        if (mcount == 7)  // Check for count and set corresponding LED high and all others
LOW
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x10;
        }
        if (mcount == 5)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x20;

        }
        if (mcount == 3)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x40;

        }
        if (mcount == 1)
        {
```

```
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x80;


        }
        if (mcount > 8)
        {
            mcount = 0;
        }
    }
    else
    {
        if (mcount == 1)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x10;
        }
        if (mcount == 3)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x20;
        }
        if (mcount == 5)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x40;
        }
        if (mcount == 7)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x80;
        }
        if (mcount > 8)
        {
            mcount = 0;
        }
    }
}
int main()
{
    initfunc();
    int trigger = 2;
while(1)
    {
        unsigned char real_button;
        real_button = released_button(); // check for which push button pressed and then pass
condition
        if (real_button == 0x0D)
```

```
        {
            trigger = 1;
        }
        if (real_button == 0x0E)
        {
            trigger = 0;
        }
        if (trigger != 2)
        {
            stepfulldir(trigger);
        }
    }
}
```

Explanation: The initfunc() initializes GPIO pins to control the motor and sets up a SysTick timer for interrupt handling, enabling precise motor control based on step counts. The SysTick_Handler() increments mcount upon each interrupt, essential for tracking the motor's position within a step cycle. The getbutton() function reads the state of push buttons connected to GPIO pins. It masks the input to isolate the lower 4 bits, representing individual buttons. The debounced_button() function addresses button bouncing, ensuring stable input by repeatedly checking for consistent button states before returning the debounced button status.

An additional function, released_button(), detects when a button is released by waiting until all buttons are in a high state (0x0F). This mechanism ensures the system only responds to released button events, preventing accidental or unstable inputs. In the main() function, the code continuously monitors button inputs using released_button(). Depending on the pressed button's status (real_button), it adjusts a trigger variable to indicate the direction for the stepfulldir() function, responsible for controlling the motor's rotation.

The stepfulldir() function interprets the trigger value to control the motor's rotation direction, switching between clockwise and anti-clockwise sequences based on this variable. It adjusts GPIO outputs to activate specific motor coils, facilitating rotation in the desired direction.

Q4. The following diagram indicates the connections between the TM4C123G, ULN2003A’s board, 4x4 Keypad Module and stepper motor.

EE447 Pre Liminary-Work Lab # 3

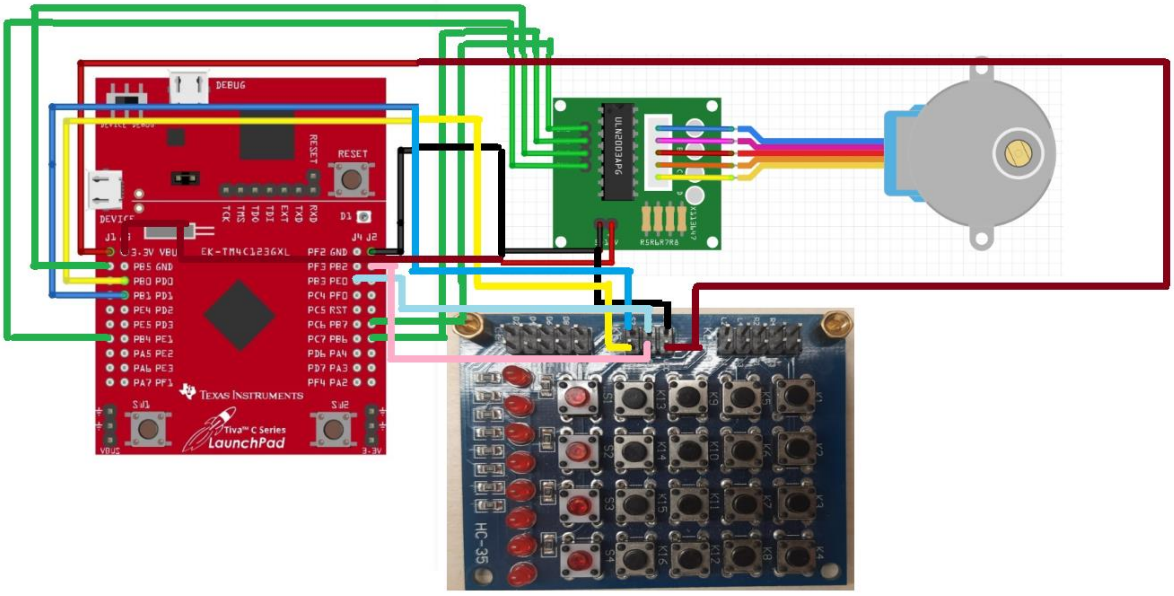

Fig 2. Swithes S3 and S4 are now used (Previously unused)

The table below shows the connections in tabular form that can be traced in the picture:

From Pin	To Pin
PB0	S1-Keypad
PB1	S2-Keypad
PB2	S3-Keypad
PB3	S4-Keypad
PB4	IN1-ULN2003A
PB5	IN2-ULN2003A
PB6	IN3-ULN2003A
PB7	IN4-ULN2003A
VBUS	V+, ULN2003A
VCC	VCC- Keypad
GND	GND Keypad and ULN2003

Q5. The following code implements all the desired requirements by adjusting speed and rotation.

Code:
```
#include "TM4C123GH6PM.h"

#define BUTTONNUMBER 4
#define INPUTMASK 0x0F
int mcount=0;

void initfunc(void)
{
    SYSCTL->RCGCGPIO |= 0x2F; // turn on bus clock for GPIOB
    GPIOB->DIR       |= 0xF0; // set PB4-PB7 pin as a digital output pin
    GPIOB->DEN       |= 0xFF; // Enable PB0-PB7 pin as a digital pin
    GPIOB->AFSEL     &=~0xFF; // Disable Alternate Function
```

```
    SysTick→LOAD     = 5000; // one milli second delay relaod value
    SysTick→CTRL     = 3;    // enable counter , interrupt and select system bus clock
    SysTick→VAL      = 0;    // initialize current value register
}
void SysTick_Handler(void)
{
    mcount++;
}
void initspeedslow(void) // Increasing speed function by loading twice the value in Systick
{
    SysTick→LOAD     = 8000;
    SysTick→CTRL     = 3;
    SysTick→VAL      = 0;
}
void initspeedfast(void)
{
    SysTick→LOAD     = 4000; // Increasing speed function by loading half the value in
Systick
    SysTick→CTRL     = 3;
    SysTick→VAL      = 0;
}


unsigned char getbutton(void) // function for getting push button input
{
    unsigned char input;
    return GPIOB→DATA & INPUTMASK;
}

unsigned char debounced_button() // debouncing problem solving
{
  unsigned char button,prevbutton;
    button = 0x8F;
    while(1)
    {
        int syscount;
        prevbutton = getbutton();
        syscount = mcount;

        while(syscount == mcount)
        {
            button = getbutton();
        }
        if (button == prevbutton)
        {
            return button;
        }
    }
```

```
}

unsigned char released_button() // function to check whether the button has released or not
{
unsigned char button,release;
    button = debounced_button();
    while(1)
    {
        release = getbutton();
        if (release == 0x0F) // by comparing all swithces to high
        {
            return button;
        }
    }
}


void stepfulldir(int dir)
{
    if (dir==1) // Check for Clockwise or Anticlockwise
    {
        if (mcount == 7)  // Check for count and set corresponding LED high and all others
LOW
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x10;
        }
        if (mcount == 5)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x20;

        }
        if (mcount == 3)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x40;

        }
        if (mcount == 1)
        {
            GPIOB->DATA &= ~0xF0;
            GPIOB->DATA |= 0x80;


        }
        if (mcount > 8)
        {
```

```
            mcount = 0;
        }
    }
    else
    {
        if (mcount == 1)
        {
            GPIOB→DATA &= ~0xF0;
            GPIOB→DATA |= 0x10;
        }
        if (mcount == 3)
        {
            GPIOB→DATA &= ~0xF0;
            GPIOB→DATA |= 0x20;
        }
        if (mcount == 5)
        {
            GPIOB→DATA &= ~0xF0;
            GPIOB→DATA |= 0x40;
        }
        if (mcount == 7)
        {
            GPIOB→DATA &= ~0xF0;
            GPIOB→DATA |= 0x80;
        }
        if (mcount > 8)
        {
            mcount = 0;
        }
    }
}
int main()
{
    initfunc();
    int trigger = 2;
    while(1)
    {
        unsigned char real_button;
        real_button = released_button();
        if (real_button == 0x07)
        {
            initspeedfast(); // increase the speed
        }
        if (real_button == 0x0B)
        {
            initspeedslow(); // decrease the speed
        }
        if (real_button == 0x0D)
```

```
        {
            trigger = 1; // run the motor in Clockwise
        }
        if (real_button == 0x0E)
        {
            trigger = 0; // run the motor in anticlockwise
        }
        if (trigger != 2)
        {
            stepfulldir(trigger);
        }
    }
}
```

Explanation: This code enhances the previous stepper motor control program by introducing features to adjust the motor's speed and change its direction based on push-button inputs. The TM4C123GH6PM microcontroller is employed to handle motor control through GPIO pins, incorporating button inputs for dynamic speed modification and directional changes. In the main() function, the code continuously monitors button inputs using released_button(). Based on the status of the pressed button (real_button), it either adjusts the motor's speed using the initspeedslow() or initspeedfast() functions or sets a trigger variable for the stepfulldir() function, which controls the motor's direction.

Flowchart :

Start → Initialization → Continuous Loop → Button Input Check -Button Pressed?→

Yes → Check Button Value -Button Value→

0x07 → Increase Speed → Continuous Loop

0x0B → Decrease Speed → Continuous Loop

0x0D or 0x0E → Set Motor Direction → Motor Direction Control → Continuous Loop

No → Button Input Check

Motor Direction Control -Check Trigger Value→

Clockwise or Anti-clockwise → Motor Direction Control → Continuous Loop

Increase Speed → Continuous Loop

Decrease Speed → Continuous Loop

End