EE690
Embedded Systems Design

EK-TM4C123: Toggling of GPIO Pin

Group11: EE23DP001 Aditya Shirodkar, EE23MS006 Prasanth Pithanisetty

Aim: To toggle a GPIO pin of the EK-TM4C123 and verify its operation using an oscilloscope.

Procedure:
1. Determine the clock frequency received by the GPIO pin and verify the same
2. Determine the number of clock cycles required to toggle a GPIO pin with delay
3. Calculate the delay to be given for generating pulses of 1KHz frequency
4. Verify the output by observing the generated waveforms using an oscilloscope.

Documents Referred:
1. TM4C123GH6PM microcontroller datasheet
2. Cortex-M4 Technical Reference Manual

Determination of Clock Frequency received by GPIO pins:
On referring to the datasheet for the TM4C123GH6PM microcontroller, it is seen that the microcontroller uses the **Precision Internal Oscillator (PIOSC)** (Pg 219) during and following Power On Reset (POR) (Pg 214). This PIOSC provides a **16MHz** clock with ±1% accuracy with calibration and ±3% accuracy across temperature.

The reception of 16MHz clock frequency from the PIOSC to the GPIO pins is also proved as elaborated below:
From the TM4C123GH6PM microcontroller datasheet, the main clock tree diagram (Fig 5-5, page 222) depicts the architecture of the various clocks present in the system.

Assumption: The PIOSC supplies 16MHz to the GPIO pins without any alteration.

A simple code for toggling of a GPIO pin without delay on port B is written, and the assembly code thus generated is observed. Then, using the Cortex-M4 technical reference manual, the number of clock cycles necessary to execute toggling of the GPIO pin is also noted.

Code for toggling GPIO pin without delay:

```
#include <stdint.h>

#include "tm4c123gh6pm.h"

int main(void)

{

    SYSCTL_RCGCGPIO_R |= 0x02; // enable clock to GPIOB

    GPIO_PORTB_DIR_R |= 0x02; // set PB1 as output

    GPIO_PORTB_DEN_R |= 0x02; // enable digital function for PB1

    int i,x;

    x=SYSCTL_RCC_R;

    while(1)

    {

        GPIO_PORTB_DATA_R = 0x02;  //Pin On

        GPIO_PORTB_DATA_R= 0x00;   //Pin Off

    }

}
```

It is seen that toggling of the GPIO pin requires a total of 11 clock cycles. Given the number of clock cycles and the clock frequency, the time period of the pulses generated would theoretically be:

$$Time\ Period\,(s) = \frac{Number\ of\ Clock\ Cycles}{Clock\ Frequency}$$

Or,

$$Frequency\,(Hz) = \frac{Clock\ Frequency\,(Hz)}{Clock\ Cycles}$$

For 11 clock cycles and a clock clock frequency of 16MHz, the time period is calculated to be 687.5ns. On connecting the GPIO pin being toggled to an oscilloscope, it is observed that the generated pulses have a time period of 691.7ns (Fig. 1), thus confirming that the GPIO pins receive 16MHz clock frequency.

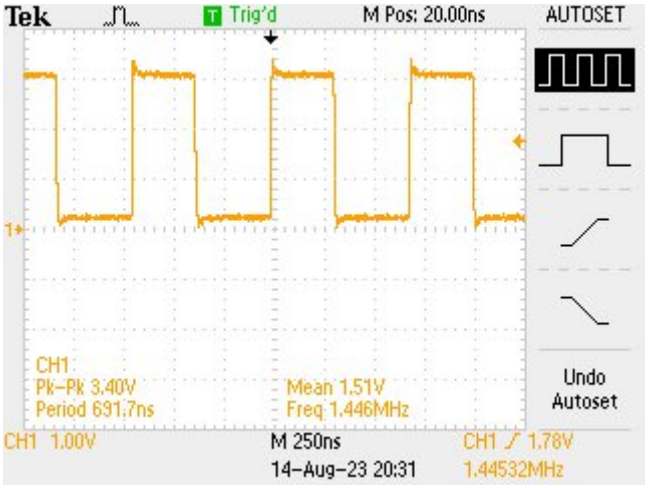


Figure 1: GPIO Pin Toggling without delay

Similarly, a code is written to execute a for loop delay, with a single count, and it is found that execution of the same requires 11 clock cycles. Therefore, to produce pulses of 1KHz frequency, the number of counts required in the for loop delay is given by

$$1KHz = \frac{16MHz}{11 + 10(2x)}$$

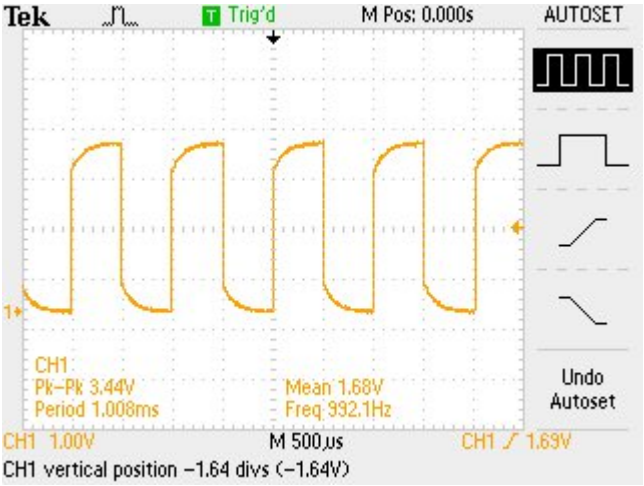
Here, "x" refers to the number of counts in the delay for-loop function. Upon calculating, it is found that a count of 800 should provide pulses of 1KHz on the GPIO pin.

Code for Toggling GPIO Pin at 1KHz:

```
#include <stdint.h>

#include "tm4c123gh6pm.h"


int main(void)

{

    SYSCTL_RCGCGPIO_R |= 0x02; // enable clock to GPIOB

    GPIO_PORTB_DIR_R |= 0x02; // set PB1 as output

    GPIO_PORTB_DEN_R |= 0x02; // enable digital function for PB1

    int i,x;

    x=SYSCTL_RCC_R;

    while(1)

    {

        GPIO_PORTB_DATA_R = 0x02; //PB1 On
```

```
            for(i = 0; i <800; i++){}// delay for 1 ms
            GPIO_PORTB_DATA_R= 0x00; //PB! Off
            for(i = 0; i <800; i++){}// do nothing for 1 ms
    }
}
```

Upon connecting the GPIO pin to the oscilloscope, it is observed that the pulses generated have a frequency of 992Hz, thus satisfactorily verifying the calculations and expected results.



Results: GPIO pin toggling was conducted successfully to achieve pulses of desired frequency on the pin. The results were observed using an oscilloscope and the calculations were thus verified.

Conclusion: The GPIO pins receive 16MHz clock signals through the PIOSC. For using the 80MHz main oscillator, software based enabling of the same is necessary. Pulses of desired frequency can be generated on GPIO pins using a rudimentary for-loop delay, with minor inaccuracies. For accurate generation of pulses, PWM modules are preferred.