Lab 1 **(G10) Daniel EE23DP003**

Lokesh Kumar EE23MT012

Write a "while" loop to toggle a pin at a periodic interval and verify using an oscilloscope.

1) C code for toggling pin PB1 every 0.5 millisecond:

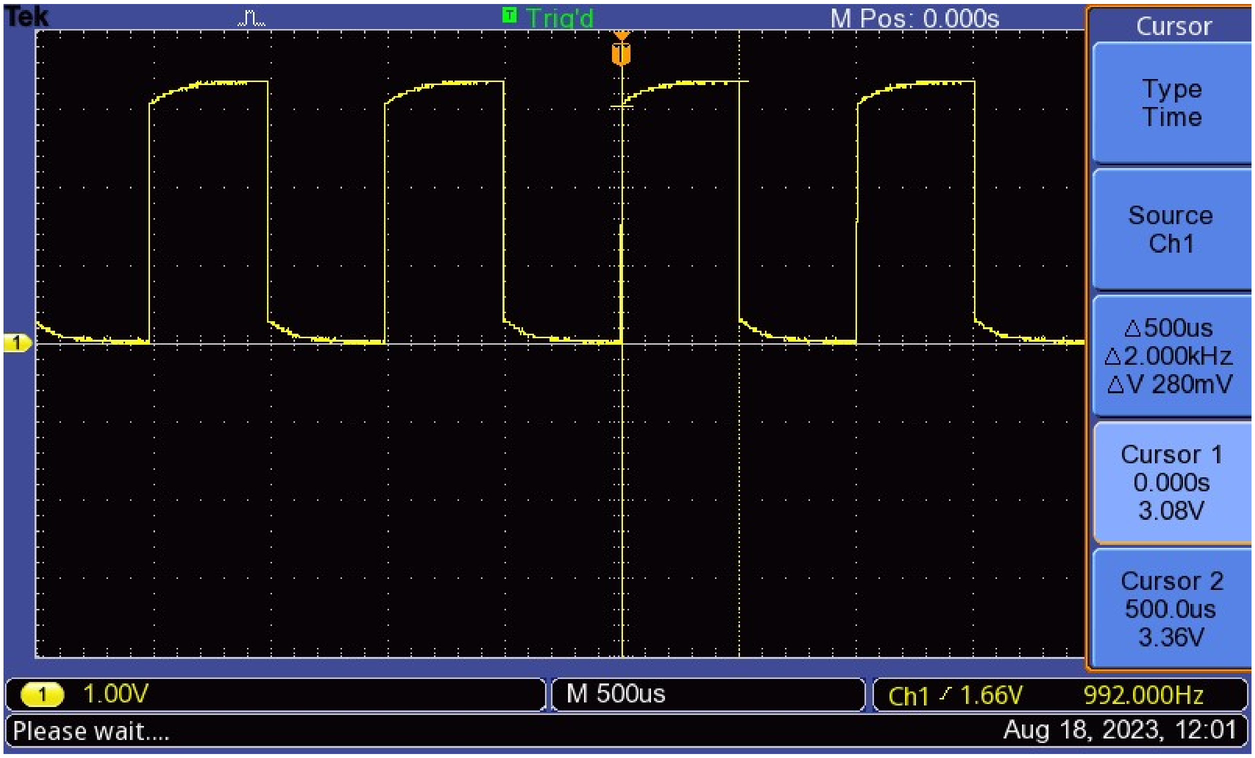
```
#include <stdint.h>
#include "tm4c123gh6pm.h"
int main(void)
{
    SYSCTL_RCGCGPIO_R |= 0x02;   // Enable clock to GPIOB
    GPIO_PORTB_DIR_R |= 0x02;   //  Set PB1 as output
    GPIO_PORTB_DEN_R |= 0x02;  //   Enable digital function for PB1

    int i;
    while(1)
    {
        GPIO_PORTB_DATA_R = 0x02;          // Set port PB1
            for(i = 0; i <800; i++){}     //  Wait for 0.5 millisecond
        GPIO_PORTB_DATA_R = 0x00;        //   Clear port PB1
            for(i = 0; i <800; i++){}   //    Wait for 0.5 millisecond
    }

}
```

2) Oscilloscope Waveform:

Switching Frequency: **1 KHz**, Ton=Toff= **0.5 ms**



M Pos: 0.000s

Cursor

Type
Time

Source
Ch1

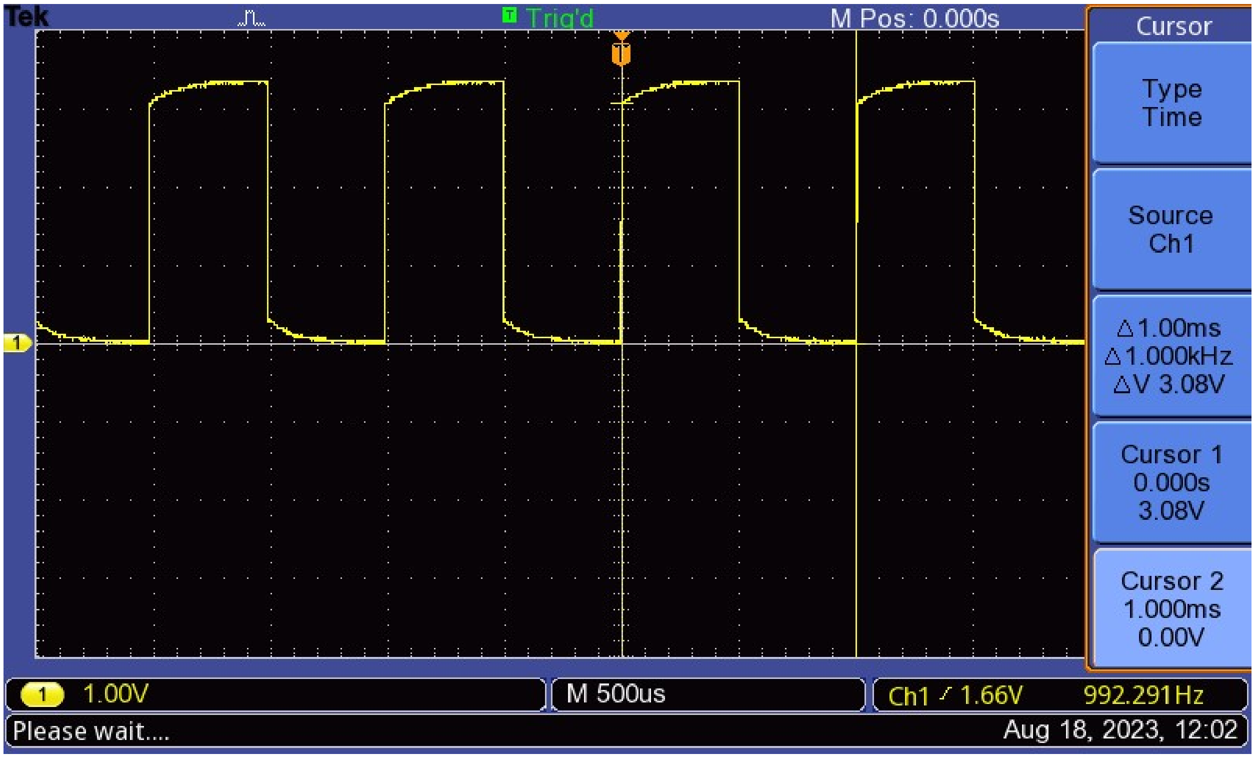
△500us
△2.000kHz
△V 280mV

Cursor 1
0.000s
3.08V

Cursor 2
500.0us
3.36V

1 1.00V M 500us Ch1 ⁄ 1.66V 992.000Hz

Please wait.... Aug 18, 2023, 12:01



M Pos: 0.000s

Cursor

Type
Time

Source
Ch1

△1.00ms
△1.000kHz
△V 3.08V

Cursor 1
0.000s
3.08V

Cursor 2
1.000ms
0.00V

1 1.00V M 500us Ch1 ⁄ 1.66V 992.291Hz

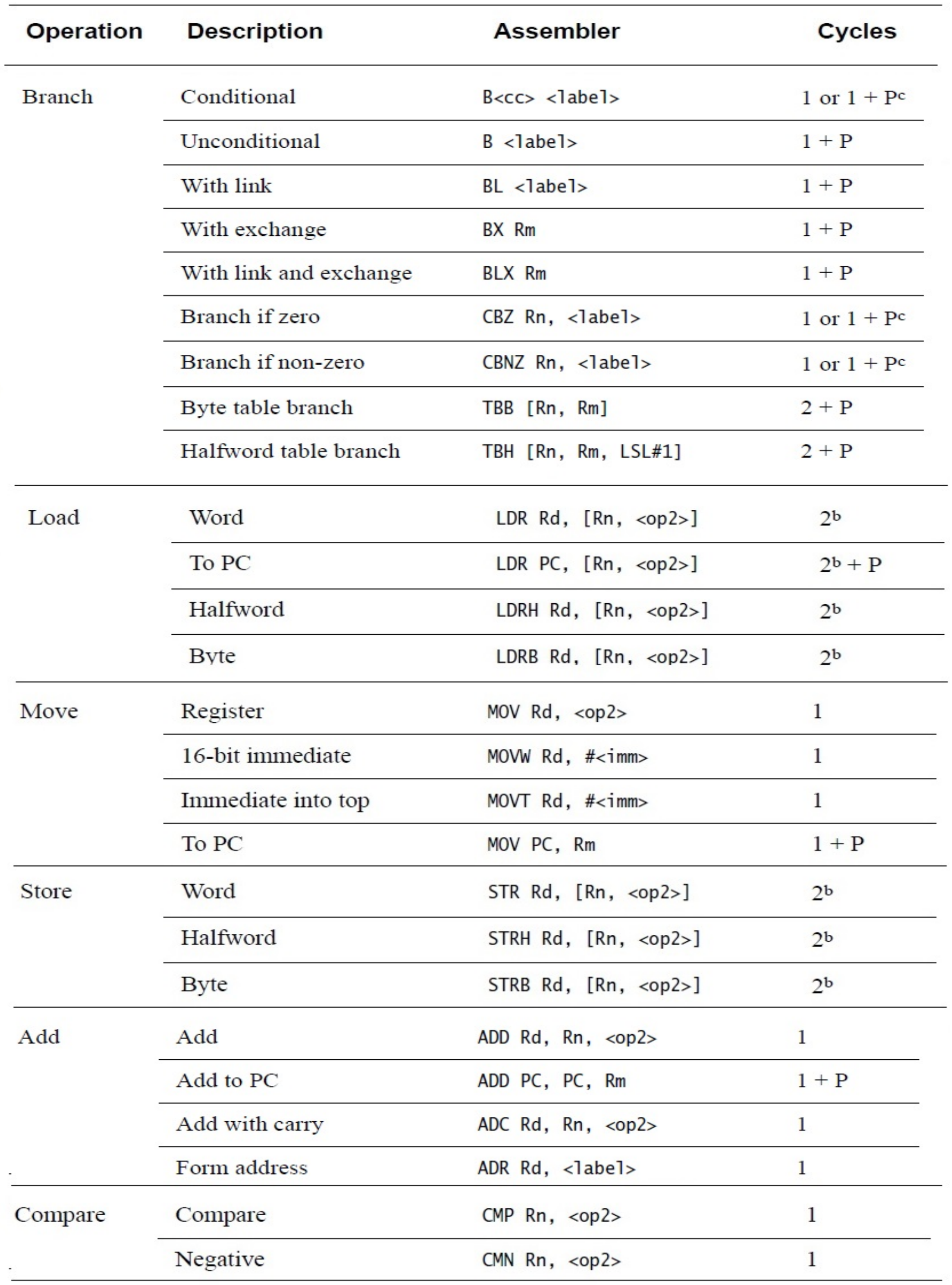
Please wait.... Aug 18, 2023, 12:02

3) Expected and actual frequency:

The default system clock frequency of the TM4C123GH6PM board is 16MHz.

The number of CPU clock cycles taken by each statement in the main program is obtained by viewing the disassembly window in CCS which contains assembly instructions and then referring to the Cortex M4 Technical Reference Manual (Chapter 3 - Programmers Model, Pages 4-12).

Table 3-1 Cortex-M4 instruction set summary (continued)

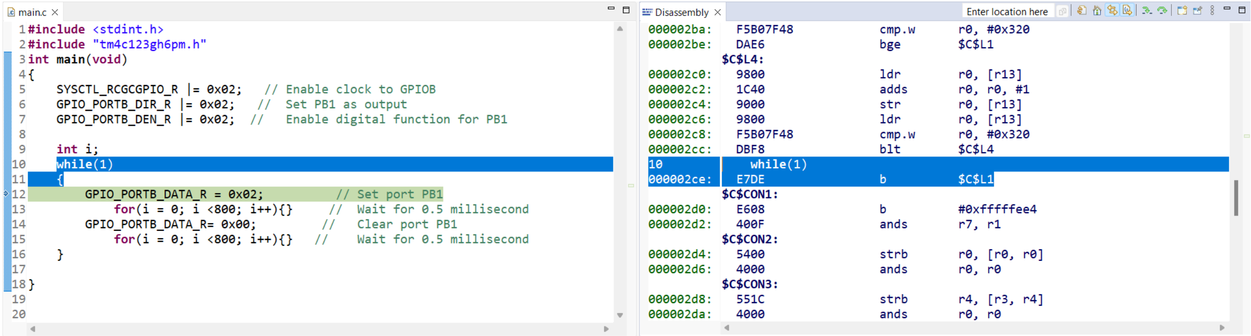


Operation	Description	Assembler	Cycles
Branch	Conditional	`B<cc> <label>`	1 or 1 + P^{c}
	Unconditional	`B <label>`	1 + P
	With link	`BL <label>`	1 + P
	With exchange	`BX Rm`	1 + P
	With link and exchange	`BLX Rm`	1 + P
	Branch if zero	`CBZ Rn, <label>`	1 or 1 + P^{c}
	Branch if non-zero	`CBNZ Rn, <label>`	1 or 1 + P^{c}
	Byte table branch	`TBB [Rn, Rm]`	2 + P
	Halfword table branch	`TBH [Rn, Rm, LSL#1]`	2 + P
Load	Word	`LDR Rd, [Rn, <op2>]`	2^{b}
	To PC	`LDR PC, [Rn, <op2>]`	2^{b} + P
	Halfword	`LDRH Rd, [Rn, <op2>]`	2^{b}
	Byte	`LDRB Rd, [Rn, <op2>]`	2^{b}
Move	Register	`MOV Rd, <op2>`	1
	16-bit immediate	`MOVW Rd, #<imm>`	1
	Immediate into top	`MOVT Rd, #<imm>`	1
	To PC	`MOV PC, Rm`	1 + P
Store	Word	`STR Rd, [Rn, <op2>]`	2^{b}
	Halfword	`STRH Rd, [Rn, <op2>]`	2^{b}
	Byte	`STRB Rd, [Rn, <op2>]`	2^{b}
Add	Add	`ADD Rd, Rn, <op2>`	1
	Add to PC	`ADD PC, PC, Rm`	1 + P
	Add with carry	`ADC Rd, Rn, <op2>`	1
	Form address	`ADR Rd, <label>`	1
Compare	Compare	`CMP Rn, <op2>`	1
	Negative	`CMN Rn, <op2>`	1

To determine the number of iterations required for the ‘for’ loop to produce a delay of 0.5ms (ON time delay or OFF time delay):

In the disassembly window, the assembly instructions can be seen along with the cycles for each statement in the main program.

The **while(1)** loop executes a branch instruction (b) that takes **2** clock cycles.

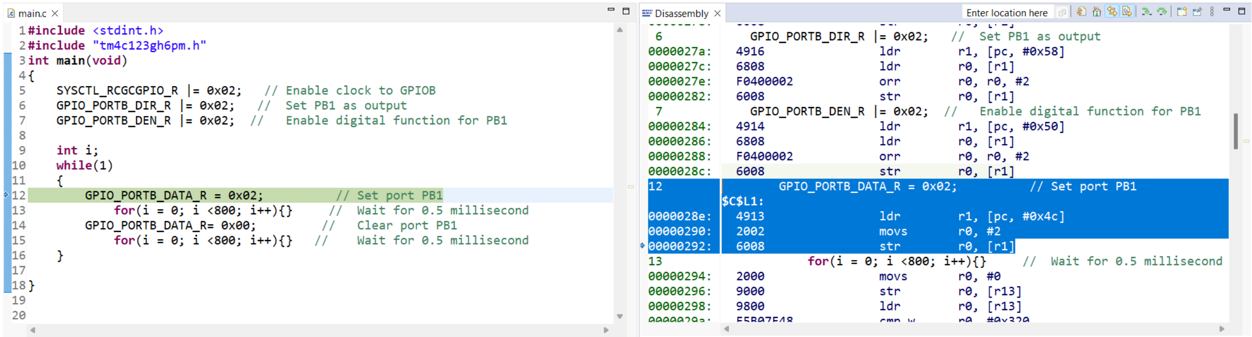


```
#include <stdint.h>
#include "tm4c123gh6pm.h"
int main(void)
{
    SYSCTL_RCGCGPIO_R |= 0x02;   // Enable clock to GPIO
    GPIO_PORTB_DIR_R |= 0x02;   //  Set PB1 as output
    GPIO_PORTB_DEN_R |= 0x02;  //   Enable digital function for PB1

    int i;
    while(1)
    {
        GPIO_PORTB_DATA_R = 0x02;           // Set port PB1
            for(i = 0; i <800; i++){}      //  Wait for 0.5 millisecond
        GPIO_PORTB_DATA_R= 0x00;          //   Clear port PB1
            for(i = 0; i <800; i++){}    //    Wait for 0.5 millisecond
    }
}
```

```
000002ba:   F5B07F48            cmp.w     r0, #0x320
000002be:   DAE6                bge       $C$L1
          $C$L4:
000002c0:   9800                ldr       r0, [r13]
000002c2:   1C40                adds      r0, r0, #1
000002c4:   9000                str       r0, [r13]
000002c6:   9800                ldr       r0, [r13]
000002c8:   F5B07F48            cmp.w     r0, #0x320
000002cc:   DBF8                blt       $C$L4
10              while(1)
000002ce:   E7DE                b         $C$L1
          $C$CON1:
000002d0:   E608                b         #0xffffee4
000002d2:   400F                ands      r7, r1
          $C$CON2:
000002d4:   5400                strb      r0, [r0, r0]
000002d6:   4000                ands      r0, r0
          $C$CON3:
000002d8:   551C                strb      r4, [r3, r4]
000002da:   4000                ands      r0, r0
```

The **GPIO_PORTB_DATA_R** command implements the following instructions: ldr: 2 cycles, mov: 1 cycle and str: 2 cycles for a total of **5** clock cycles.

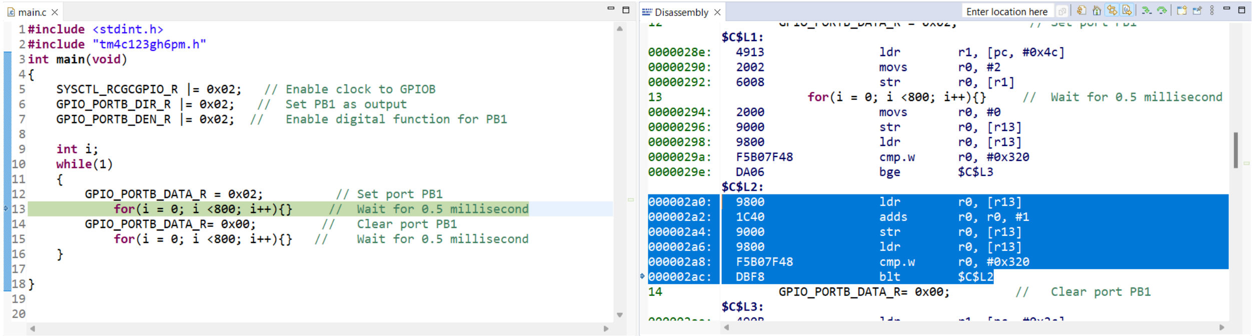


```
#include <stdint.h>
#include "tm4c123gh6pm.h"
int main(void)
{
    SYSCTL_RCGCGPIO_R |= 0x02;   // Enable clock to GPIO
    GPIO_PORTB_DIR_R |= 0x02;   //  Set PB1 as output
    GPIO_PORTB_DEN_R |= 0x02;  //   Enable digital function for PB1

    int i;
    while(1)
    {
        GPIO_PORTB_DATA_R = 0x02;           // Set port PB1
            for(i = 0; i <800; i++){}      //  Wait for 0.5 millisecond
        GPIO_PORTB_DATA_R= 0x00;          //   Clear port PB1
            for(i = 0; i <800; i++){}    //    Wait for 0.5 millisecond
    }
}
```

```
 6              GPIO_PORTB_DIR_R |= 0x02;   //  Set PB1 as output
0000027a:   4916                ldr       r1, [pc, #0x58]
0000027c:   6808                ldr       r0, [r1]
0000027e:   F0400002            orr       r0, r0, #2
00000282:   6008                str       r0, [r1]
 7              GPIO_PORTB_DEN_R |= 0x02;  //   Enable digital function for PB1
00000284:   4914                ldr       r1, [pc, #0x50]
00000286:   6808                ldr       r0, [r1]
00000288:   F0400002            orr       r0, r0, #2
0000028c:   6008                str       r0, [r1]
12                  GPIO_PORTB_DATA_R = 0x02;           // Set port PB1
          $C$L1:
0000028e:   4913                ldr       r1, [pc, #0x4c]
00000290:   2002                movs      r0, #2
00000292:   6008                str       r0, [r1]
13                      for(i = 0; i <800; i++){}     //  Wait for 0.5 millisecond
00000294:   2000                movs      r0, #0
00000296:   9000                str       r0, [r13]
00000298:   9800                ldr       r0, [r13]
```

Each iteration of the **‘for’** loop implements the following instructions: ldr: 2 cycles, adds: 1 cycle, str: 2 cycles, ldr: 2 cycles, cmp: 1 cycles, blt: 2 cycles for a total of **10** clock cycles per ‘for’ loop iteration.



```
#include <stdint.h>
#include "tm4c123gh6pm.h"
int main(void)
{
    SYSCTL_RCGCGPIO_R |= 0x02;   // Enable clock to GPIO
    GPIO_PORTB_DIR_R |= 0x02;   //  Set PB1 as output
    GPIO_PORTB_DEN_R |= 0x02;  //   Enable digital function for PB1

    int i;
    while(1)
    {
        GPIO_PORTB_DATA_R = 0x02;           // Set port PB1
            for(i = 0; i <800; i++){}      //  Wait for 0.5 millisecond
        GPIO_PORTB_DATA_R= 0x00;          //   Clear port PB1
            for(i = 0; i <800; i++){}    //    Wait for 0.5 millisecond
    }
}
```

```
          $C$L1:
0000028e:   4913                ldr       r1, [pc, #0x4c]
00000290:   2002                movs      r0, #2
00000292:   6008                str       r0, [r1]
13                      for(i = 0; i <800; i++){}     //  Wait for 0.5 millisecond
00000294:   2000                movs      r0, #0
00000296:   9000                str       r0, [r13]
00000298:   9800                ldr       r0, [r13]
0000029a:   F5B07F48            cmp.w     r0, #0x320
0000029e:   DA06                bge       $C$L3
          $C$L2:
000002a0:   9800                ldr       r0, [r13]
000002a2:   1C40                adds      r0, r0, #1
000002a4:   9000                str       r0, [r13]
000002a6:   9800                ldr       r0, [r13]
000002a8:   F5B07F48            cmp.w     r0, #0x320
000002ac:   DBF8                blt       $C$L2
14                  GPIO_PORTB_DATA_R= 0x00;          //   Clear port PB1
          $C$L3:
```

Considering that **Clock Cycles = Execution Time Period * CPU Clock Frequency**, we get:

$$7 + 10X = 0.5\text{ms} * 16\text{ MHz}$$

Where 7 clock cycles is taken by the while(1) loop and GPIO_PORTB_DATA_R statements. The ‘for’ loop takes 10 clock cycles per iteration.

Solving the above equation, where X is the number of iterations of the ‘for’ loop to produce a 0.5 ms delay results in the value of X=799.3.

Taking **800** iterations in each ‘for’ loop, produces an expected delay of 0.5 ms on the GPIO pin after each toggle statement. The expected switching frequency on pin PB1 is 1/(1ms) = **1 KHz** with (Ton=Toff= 0.5ms) which matches with the actual switching frequency of 1 KHz seen in the oscilloscope waveform.