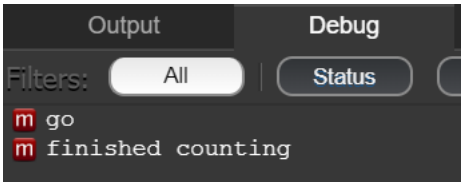
Rajbir Kahlon

Lab 4 Assignment

Exercise 4.1

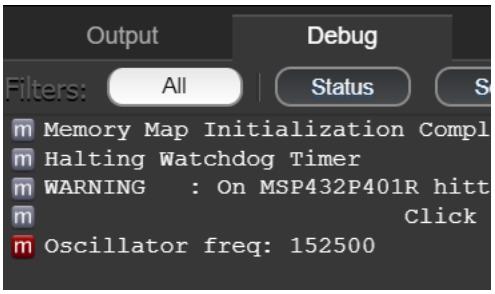


This is the result in shows the time delay starting to count and finishing. Function was called as delay_ms(10000) and compared it to 10 seconds on a real clock. It was accurate. Code is in appendix.

Exercise 4.2

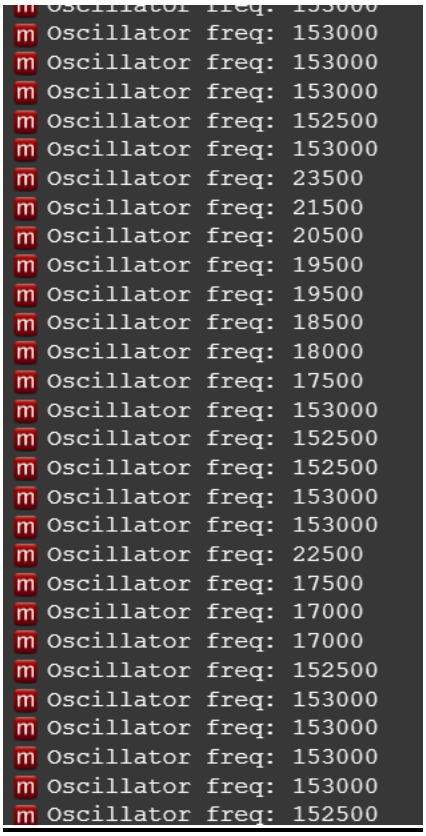
This is the output stream generated. Code is in appendix.

Exercise 4.3



This was the oscillator frequency produced. Code is in appendix.

Exercise 4.4



The oscillator frequency was usually about 100,000 closer to 150,000. When the pin was touched, it went to a lot lower to under 20,000. These rough estimates were used to trigger the LED too as seen in the code. The code is in the appendix.

Appendix

Exercise 4.1

/* --COPYRIGHT--,BSD

* Copyright (c) 2017, Texas Instruments Incorporated

* All rights reserved.

*

* Redistribution and use in source and binary forms, with or without

* modification, are permitted provided that the following conditions

* are met:

*

* * Redistributions of source code must retain the above copyright

* notice, this list of conditions and the following disclaimer.

/**

* MSP432 DMA - CRC32 calculation using DMA

```
*
* Description: This code examples shows how to use the DMA module on MSP432
* to feed data into the CRC32 module for a CRC32 signature calculation. This
* use case is particularly useful when the user wants to calculate the CRC32
* signature of a large data array (such as a firmware image) but still wants
* to maximize power consumption. After the DMA transfer is setup, a software
* initiation occurs and the MSP432 device is put to sleep. Once the transfer
* completes, the DMA interrupt occurs and the CRC32 result is placed into
* a local variable for the user to examine.
*
*             MSP432P401
*            ------------------
*         /|\|                  |
*          | |                  |
*          --|RST               |
*            |                  |
*            |                  |
*            |                  |
*            |                  |
*
*******************************************************************************
/
/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
```

```
#include <string.h>
#include <stdbool.h>



void delay_ms(uint32_t count);


bool tick;
volatile uint32_t val;


int main(void)
{
   Timer32_initModule(TIMER32_0_BASE, TIMER32_PRESCALER_1, TIMER32_32BIT,
TIMER32_FREE_RUN_MODE);
   Timer32_startTimer(TIMER32_0_BASE, true);


   /* Halting the Watchdog */
   MAP_WDT_A_holdTimer();


   tick = true;
   printf("go\n");
   delay_ms(10000);
   printf("finished counting\n");
}


void delay_ms(uint32_t count) {
   Timer32_startTimer(TIMER32_0_BASE,true);
   Timer32_setCount(TIMER32_BASE,UINT32_MAX);
   val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);
```

```
    while(tick) {
        val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);
        if(val >= (count*3000))
            tick = false;
    }
}
```

Exercise 4.2

```
/* --COPYRIGHT--,BSD
 * Copyright (c) 2017, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS"
```

```
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
/******************************************************************************
 * MSP432 DMA - CRC32 calculation using DMA
 *
 * Description: This code examples shows how to use the DMA module on MSP432
 * to feed data into the CRC32 module for a CRC32 signature calculation. This
 * use case is particularly useful when the user wants to calculate the CRC32
 * signature of a large data array (such as a firmware image) but still wants
 * to maximize power consumption. After the DMA transfer is setup, a software
 * initiation occurs and the MSP432 device is put to sleep. Once the transfer
 * completes, the DMA interrupt occurs and the CRC32 result is placed into
 * a local variable for the user to examine.
 *
 *            MSP432P401
```

```
 *            -----------------
 *         /|\|                 |
 *          | |                 |
 *          --|RST              |
 *            |                 |
 *            |                 |
 *            |                 |
 *            |                 |
 *
 ******************************************************************************
 /
 /* DriverLib Includes */
 #include <ti/devices/msp432p4xx/driverlib/driverlib.h>

 /* Standard Includes */
 #include <stdint.h>
 #include <stdlib.h>
 #include <stdio.h>

 #include <string.h>
 #include <stdbool.h>



 void delay_ms(uint32_t count);

 bool tick, state;
 volatile uint32_t val;
```

```
int main(void)
{
    Timer32_initModule(TIMER32_0_BASE, TIMER32_PRESCALER_1, TIMER32_32BIT,
TIMER32_FREE_RUN_MODE);
    Timer32_startTimer(TIMER32_0_BASE, true);


    /* Halting the Watchdog */
    MAP_WDT_A_holdTimer();


    tick = true;
    printf("go\n");
    delay_ms(10000);
    printf("finished counting\n");


    CAPTIO0CTL |= (1 << 8);
    CAPTIO0CTL |= 0b0100 << 4;
    CAPTIO0CTL |= 0b0001 << 1;


    while(true) {
        state = CAPTIO0CTL & 0x200;
        printf("%d", state);
        delay_ms(50);
        fflush(stdout);
    }
}


void delay_ms(uint32_t count) {
    Timer32_startTimer(TIMER32_0_BASE,true);
    Timer32_setCount(TIMER32_BASE,UINT32_MAX);
```

```
    val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);

    while(tick) {
        val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);
        if(val >= (count*3000))
            tick = false;
    }
}
```

Exercise 4.3

```
/* --COPYRIGHT--,BSD
 * Copyright (c) 2017, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
```

```
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
/******************************************************************************
 * MSP432 DMA - CRC32 calculation using DMA
 *
 * Description: This code examples shows how to use the DMA module on MSP432
 * to feed data into the CRC32 module for a CRC32 signature calculation. This
 * use case is particularly useful when the user wants to calculate the CRC32
 * signature of a large data array (such as a firmware image) but still wants
 * to maximize power consumption. After the DMA transfer is setup, a software
 * initiation occurs and the MSP432 device is put to sleep. Once the transfer
 * completes, the DMA interrupt occurs and the CRC32 result is placed into
 * a local variable for the user to examine.
```

```
 *
 *             MSP432P401
 *           ------------------
 *         /|\|                 |
 *          | |                 |
 *          --|RST              |
 *            |                 |
 *            |                 |
 *            |                 |
 *            |                 |
 *
*****************************************************************************
/
/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

#include <string.h>
#include <stdbool.h>
#include <time.h>

void delay_ms(uint32_t count);

const Timer_A_ContinuousModeConfig continuousModeConfig = {
```

```
    TIMER_A_CLOCKSOURCE_INVERTED_EXTERNAL_TXCLK,
    TIMER_A_CLOCKSOURCE_DIVIDER_1,
    TIMER_A_TAIE_INTERRUPT_ENABLE,
    TIMER_A_DO_CLEAR
};

bool tick, state;
volatile uint32_t val, tim1, tim2;

int main(void)
{
    Timer32_initModule(TIMER32_0_BASE, TIMER32_PRESCALER_1, TIMER32_32BIT,
TIMER32_FREE_RUN_MODE);
    Timer32_startTimer(TIMER32_0_BASE, true);

    /* Halting the Watchdog */
    MAP_WDT_A_holdTimer();

    /*tick = true;
    printf("go\n");
    delay_ms(10000);
    printf("finished counting\n");
    */
    CAPTIO0CTL |= (1 << 8);
    CAPTIO0CTL |= 0b0100 << 4;
    CAPTIO0CTL |= 0b0001 << 1;

    /*while(true) {
        state = CAPTIO0CTL & 0x200;
```

```
        printf("%d", state);
        delay_ms(50);
        fflush(stdout);
    } */

    MAP_Timer_A_configureContinuousMode(TIMER_A2_BASE, &continuousModeConfig);
    MAP_Timer_A_startCounter(TIMER_A2_BASE, TIMER_A_CONTINUOUS_MODE);
    MAP_Timer_A_clearTimer(TIMER_A2_BASE);
    delay_ms(2);
    tim1 = Timer_A_getCounterValue(TIMER_A2_BASE);
    tim2 = tim1/.002; //2ms
    printf("Oscillator freq: %d \n", tim2);
}

void delay_ms(uint32_t count) {
    Timer32_startTimer(TIMER32_0_BASE,true);
    Timer32_setCount(TIMER32_BASE,UINT32_MAX);
    val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);

    while(tick) {
        val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);
        if(val >= (count*3000))
            tick = false;
    }
}
```

Exercise 4.4

```
/* --COPYRIGHT--,BSD
 * Copyright (c) 2017, Texas Instruments Incorporated
```

```
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
/******************************************************************************
 * MSP432 DMA - CRC32 calculation using DMA
 *
 * Description: This code examples shows how to use the DMA module on MSP432
 * to feed data into the CRC32 module for a CRC32 signature calculation. This
 * use case is particularly useful when the user wants to calculate the CRC32
 * signature of a large data array (such as a firmware image) but still wants
 * to maximize power consumption. After the DMA transfer is setup, a software
 * initiation occurs and the MSP432 device is put to sleep. Once the transfer
 * completes, the DMA interrupt occurs and the CRC32 result is placed into
 * a local variable for the user to examine.
 *
 *             MSP432P401
 *           ------------------
 *         /|\|                  |
 *          | |                  |
 *          --|RST               |
 *            |                  |
 *            |                  |
 *            |                  |
 *            |                  |
```

```
 *
******************************************************************************
/
/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

#include <string.h>
#include <stdbool.h>
#include <time.h>

void delay_ms(uint32_t count);

const Timer_A_ContinuousModeConfig continuousModeConfig = {
    TIMER_A_CLOCKSOURCE_INVERTED_EXTERNAL_TXCLK,
    TIMER_A_CLOCKSOURCE_DIVIDER_1,
    TIMER_A_TAIE_INTERRUPT_ENABLE,
    TIMER_A_DO_CLEAR
};

bool tick, state;
volatile uint32_t val, tim1, tim2;

int main(void)
```

```
{
    MAP_GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN0);
    Timer32_initModule(TIMER32_0_BASE, TIMER32_PRESCALER_1, TIMER32_32BIT,
TIMER32_FREE_RUN_MODE);
    Timer32_startTimer(TIMER32_0_BASE, true);

    /* Halting the Watchdog */
    MAP_WDT_A_holdTimer();

    /*tick = true;
    printf("go\n");
    delay_ms(10000);
    printf("finished counting\n"); */

    CAPTIO0CTL |= (1 << 8);
    CAPTIO0CTL |= 0b0100 << 4;
    CAPTIO0CTL |= 0b0001 << 1;
/*
while(true) {
        state = CAPTIO0CTL & 0x200;
        printf("%d", state);
        delay_ms(50);
        fflush(stdout);
    } */

    while(true) {
    MAP_Timer_A_configureContinuousMode(TIMER_A2_BASE, &continuousModeConfig);
    MAP_Timer_A_startCounter(TIMER_A2_BASE, TIMER_A_CONTINUOUS_MODE);
    MAP_Timer_A_clearTimer(TIMER_A2_BASE);
```

```
    delay_ms(2);
    tim1 = Timer_A_getCounterValue(TIMER_A2_BASE);
    tim2 = tim1/.002; //2ms
    if(tim2>100000) {
        MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);
    }
    if(tim2<100000) {
        MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN0);
    }
    printf("Oscillator freq: %d \n", tim2);
    }
}

void delay_ms(uint32_t count) {
    Timer32_startTimer(TIMER32_0_BASE,true);
    Timer32_setCount(TIMER32_BASE,UINT32_MAX);
    val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);

    while(tick) {
        val = UINT32_MAX-Timer32_getValue(TIMER32_BASE);
        if(val >= (count*3000))
            tick = false;
    }
}
```