**Angelica Semenec**

**012179180**

**Lab 4**

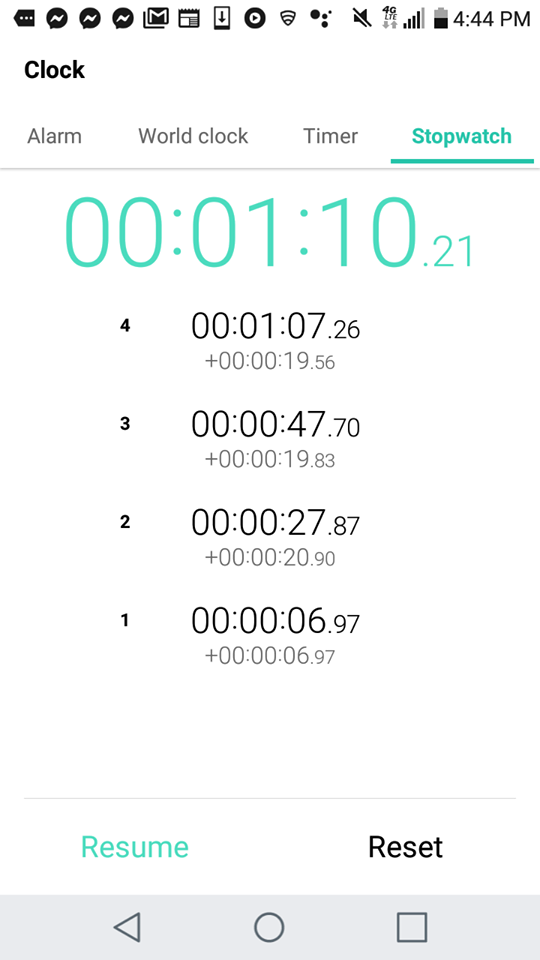
**GPIO capacitive-touch and Timer\_A module on MSP432 MCU**

**Purpose**

The purpose of this lab is to gain an understanding of the MSP432’s GPIO ports and their capacitive-touch functionality. This also covers some of the basic functionality of the Timer\_A module.

**Exercise 1**

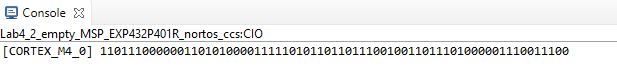
|  |
| --- |
| **main.c** |
| #include <ti/devices/msp432p4xx/driverlib/driverlib.h>  #include <stdint.h>  #include <stdbool.h>  #include <stdio.h>  void delay\_ms(uint32\_t count); //Delay function  uint32\_t t0, t1, cycles;  int main(void)  {  /\* Stop Watchdog \*/  MAP\_WDT\_A\_holdTimer();  /\* Set up Timer32 \*/  MAP\_Timer32\_initModule(TIMER32\_0\_BASE, TIMER32\_PRESCALER\_1, TIMER32\_32BIT,  TIMER32\_FREE\_RUN\_MODE);  MAP\_Timer32\_startTimer(TIMER32\_0\_BASE, 0);  /\* Configure output pins \*/  MAP\_GPIO\_setAsOutputPin(GPIO\_PORT\_P1, GPIO\_PIN0);  MAP\_GPIO\_setAsOutputPin(GPIO\_PORT\_P2, GPIO\_PIN1);  /\* Toggle lights \*/  while(1)  {  delay\_ms(2000);  MAP\_GPIO\_toggleOutputOnPin(GPIO\_PORT\_P1, GPIO\_PIN0);  MAP\_GPIO\_toggleOutputOnPin(GPIO\_PORT\_P2, GPIO\_PIN1);  }  }  /\* Delay number of milliseconds \*/  void delay\_ms(uint32\_t count)  {  t0 = MAP\_Timer32\_getValue(TIMER32\_0\_BASE); //Get current count  cycles = count \* (MAP\_CS\_getMCLK()/1000); //Calculate number of cycles  do{  t1 = MAP\_Timer32\_getValue(TIMER32\_0\_BASE);  }while((t0 - t1) < cycles); //Check cycle count  return;  } |



The figure above shows times taken with a stopwatch per every 10 toggles of the system. Multiple laps were recorded to get a more accurate reading of the time. Taking the average of these yields 1 toggle per every 2.0097 seconds, which is in accordance with the count (2000 ms or 2 s) set in the above code.

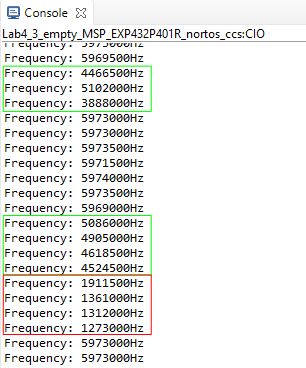
**Exercise 2**

|  |
| --- |
| **main.c** |
| /\* DriverLib Includes \*/  #include <ti/devices/msp432p4xx/driverlib/driverlib.h>  /\* Standard Includes \*/  #include <stdint.h>  #include <stdbool.h>  #include <stdio.h>  void delay\_ms(uint32\_t count);  uint32\_t t0, t1, cycles;  int main(void)  {  /\* Stop Watchdog \*/  MAP\_WDT\_A\_holdTimer();  /\* Set up Timer32 \*/  MAP\_Timer32\_initModule(TIMER32\_0\_BASE, TIMER32\_PRESCALER\_1, TIMER32\_32BIT,  TIMER32\_FREE\_RUN\_MODE);  MAP\_Timer32\_startTimer(TIMER32\_0\_BASE, 0);  CAPTIO0CTL |= (1 << 8); // Enable CAPTIO  CAPTIO0CTL |= 0b0100 << 4; // Choose Port 4  CAPTIO0CTL |= 0b0001 << 1; // Choose Pin 1  while(1){  printf("%d", (bool)(CAPTIO0CTL & 0x200)); //Read state of capacitor  fflush(stdout);  delay\_ms(10);  }  }  /\* Delay number of milliseconds \*/  void delay\_ms(uint32\_t count)  {  t0 = MAP\_Timer32\_getValue(TIMER32\_0\_BASE); //Get current count  cycles = count \* (MAP\_CS\_getMCLK()/1000); //Calculate number of cycles  do{  t1 = MAP\_Timer32\_getValue(TIMER32\_0\_BASE);  }while((t0 - t1) < cycles); //Check cycle count  return;  } |



**Exercise 3**

|  |
| --- |
| **main.c** |
| /\* DriverLib Includes \*/  #include <ti/devices/msp432p4xx/driverlib/driverlib.h>  /\* Standard Includes \*/  #include <stdint.h>  #include <stdbool.h>  #include <stdio.h>  void delay\_ms(uint32\_t count);  uint32\_t t0, t1, cycles;  uint16\_t timer\_A\_cycles;  int main(void)  {  /\* Stop Watchdog \*/  MAP\_WDT\_A\_holdTimer();  /\* Set up Timer32 \*/  MAP\_Timer32\_initModule(TIMER32\_0\_BASE, TIMER32\_PRESCALER\_1, TIMER32\_32BIT,  TIMER32\_FREE\_RUN\_MODE);  MAP\_Timer32\_startTimer(TIMER32\_0\_BASE, 0);  /\* Setting up the oscillator at port 4, pin 1 \*/  CAPTIO0CTL |= (1 << 8); // Enable CAPTIO  CAPTIO0CTL |= 0b0100 << 4; // Choose Port 4  CAPTIO0CTL |= 0b0001 << 1; // Choose Pin 1  /\* Setting up Timer\_A module \*/  Timer\_A\_ContinuousModeConfig timer\_continuous\_obj; // Declare Timer\_A object  timer\_continuous\_obj.clockSource = TIMER\_A\_CLOCKSOURCE\_INVERTED\_EXTERNAL\_TXCLK; //INCLK  timer\_continuous\_obj.clockSourceDivider = TIMER\_A\_CLOCKSOURCE\_DIVIDER\_1; //No division  timer\_continuous\_obj.timerInterruptEnable\_TAIE = TIMER\_A\_TAIE\_INTERRUPT\_DISABLE; //No interrupt  timer\_continuous\_obj.timerClear = TIMER\_A\_DO\_CLEAR; //Clear timer and disable during configuration  MAP\_Timer\_A\_configureContinuousMode(TIMER\_A2\_BASE, &timer\_continuous\_obj); //Configure mode  MAP\_Timer\_A\_startCounter(TIMER\_A2\_BASE, TIMER\_A\_CONTINUOUS\_MODE); //Start the timer  while(1){  MAP\_Timer\_A\_clearTimer(TIMER\_A2\_BASE); //Clear timer to count up from 0  delay\_ms(2); //Timer delay for frequency  timer\_A\_cycles = MAP\_Timer\_A\_getCounterValue(TIMER\_A2\_BASE); //Read the counter  delay\_ms(500); //Print delay for console output  printf("Frequency: %dHz\n", (int)(timer\_A\_cycles\*500)); //Frequency of Timer\_A  }  }  /\* Delay number of milliseconds \*/  void delay\_ms(uint32\_t count)  {  t0 = MAP\_Timer32\_getValue(TIMER32\_0\_BASE); //Get current count  cycles = count \* (MAP\_CS\_getMCLK()/1000); //Calculate number of cycles  do{  t1 = MAP\_Timer32\_getValue(TIMER32\_0\_BASE);  }while((t0 - t1) < cycles); //Check cycle count  return;  } |



The console output shows the frequency of the clock when it was changed slightly (green border), when it was changed drastically (red border) and the frequency with no human intervention (no border). To change the frequency slightly, the pin was touched gently. This produced less contact with the capacitive pin, which limited the number of electrons travelling from the pin to the finger. The amount of space allowing the transfer was minimized. To change the frequency drastically, the pin was pressed harder or the side of the pin instead of the tip was touched, which gave it more contact with the finger. More contact means that there is more surface area touching the pin. This allows for a larger transfer of electrons in a given unit of time, reducing the frequency of the clock more drastically than when it was touched gently.

**Exercise 4**

|  |
| --- |
| **main.c** |
| /\* DriverLib Includes \*/  #include <ti/devices/msp432p4xx/driverlib/driverlib.h>  /\* Standard Includes \*/  #include <stdint.h>  #include <stdbool.h>  #include <stdio.h>  void delay\_ms(uint32\_t count);  uint32\_t t0, t1, cycles;  uint\_fast8\_t Port = GPIO\_PORT\_P1;  uint\_fast16\_t Pin = GPIO\_PIN0;  int main(void)  {  /\* Stop Watchdog \*/  MAP\_WDT\_A\_holdTimer();  /\* Set up Timer32 \*/  MAP\_Timer32\_initModule(TIMER32\_0\_BASE, TIMER32\_PRESCALER\_1, TIMER32\_32BIT,  TIMER32\_FREE\_RUN\_MODE);  MAP\_Timer32\_startTimer(TIMER32\_0\_BASE, 0);  /\* Setting up the oscillator at port 4, pin 1 \*/  CAPTIO0CTL |= (1 << 8); // Enable CAPTIO  CAPTIO0CTL |= 0b0100 << 4; // Choose Port 4  CAPTIO0CTL |= 0b0001 << 1; // Choose Pin 1  /\* Setting up Timer\_A module \*/  Timer\_A\_ContinuousModeConfig timer\_continuous\_obj; // Declare Timer\_A object  timer\_continuous\_obj.clockSource = TIMER\_A\_CLOCKSOURCE\_INVERTED\_EXTERNAL\_TXCLK; //INCLK  timer\_continuous\_obj.clockSourceDivider = TIMER\_A\_CLOCKSOURCE\_DIVIDER\_1; //No division  timer\_continuous\_obj.timerInterruptEnable\_TAIE = TIMER\_A\_TAIE\_INTERRUPT\_DISABLE; //No interrupt  timer\_continuous\_obj.timerClear = TIMER\_A\_DO\_CLEAR; //Clear timer and disable during configuration  MAP\_Timer\_A\_configureContinuousMode(TIMER\_A2\_BASE, &timer\_continuous\_obj); //Configure mode  MAP\_Timer\_A\_startCounter(TIMER\_A2\_BASE, TIMER\_A\_CONTINUOUS\_MODE); //Start the timer    MAP\_GPIO\_setAsOutputPin(Port, Pin); //Set up red LED  while(1){  MAP\_Timer\_A\_clearTimer(TIMER\_A2\_BASE); //Clear timer to count up from 0  delay\_ms(2); //Timer delay for frequency  if((uint32\_t)(MAP\_Timer\_A\_getCounterValue(TIMER\_A2\_BASE)/20) < 5) //if frequency < threshold  {  MAP\_GPIO\_setOutputHighOnPin(Port, Pin); //Turn on LED  }  else  {  MAP\_GPIO\_setOutputLowOnPin(Port, Pin); //Turn off LED  }  }  }  /\* Delay number of milliseconds \*/  void delay\_ms(uint32\_t count)  {  t0 = MAP\_Timer32\_getValue(TIMER32\_0\_BASE); //Get current count  cycles = count \* (MAP\_CS\_getMCLK()/1000); //Calculate number of cycles  do{  t1 = MAP\_Timer32\_getValue(TIMER32\_0\_BASE);  }while((t0 - t1) < cycles); //Check cycle count  return;  } |

In the above code, an LED on the board will turn on when it is touched and will turn off when there is no contact. Timer\_A is reset inside the *while* loop and then the main function is delayed for 2 ms. After the delay, the program then checks if the frequency of the clock at that instance is below a specified threshold. For this implementation, the threshold is 5MHz. The average value for the frequency of the oscillator when there is no contact with the pin was 5.9MHz. The threshold is high and so a minimal amount of contact will reduce the frequency beneath its value and turn on the LED. To calculate the current frequency of the oscillator, the number of cycles is divided by the delay (0.002 s) and converted to MHz (106). The calculation is simplified in the implemented code. When the measured frequency is below the specified threshold (the pin is being touched), the LED is turned on by setting the output pin to high. When the frequency is above the specified threshold (the pin is not being touched), the LED is turned off by setting the output pin to low. This algorithm is repeated in the *while* loop until the program is closed and so it continuously polls the frequency.