Electrical and Computer Engineering
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas

CE6302: Microprocessor and Embedded Systems

Dr. Tooraj Nikoubin

MCU Timer

Submitted by:
Muripa Uppaluri (mxu220008)
Chandanam Sai Nived (sxc210186)

Table of Contents

MCU Timer

1. Introduction

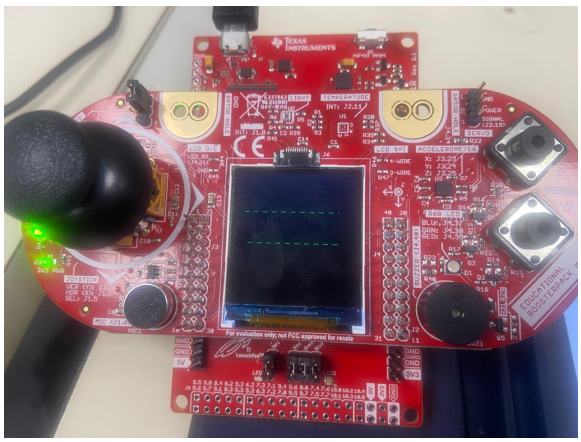
In this lab, we will write the program to control the PWM on the MSP-EXP432P4111 board. There are multiple timers and PWM modules on the MSP-EXP432P4111 board. In this experiment, we will use timer0 to output PWM signals. The square wave should be 10Hz with 50% duty cycle. We implemented this in three timer output modes (mode2: toggle/reset, mode4: toggle and mode7: reset/set) under two counting modes (up mode and up/down mode).

2. Procedure

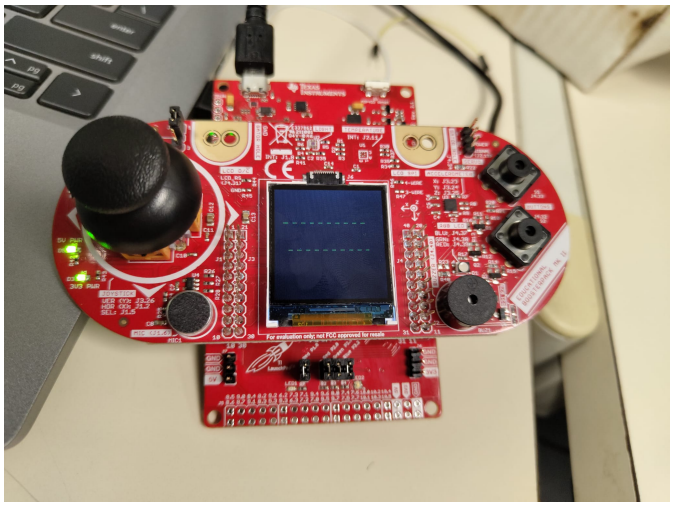
MSP432P4111 contains one PWM module and four PWM generator blocks. The PWM output could be directly connected to the ports. The Ports 2.4, 2.5, 2.6, and 2.7 could be the output of PWM with timer 0.

To modify the period of the PWM outputs. We could change the ”Reset” node (TACCR0). The ACLK is 32.768kHz. To get the PWM output in 3.2kHz, the TACCR0 will be set to 10.The ”toggle” (TACCR1) node is to determine the duty cycle. Under the specific TACCR1, The duty cycle is related to the timer output function selected (CCTL).

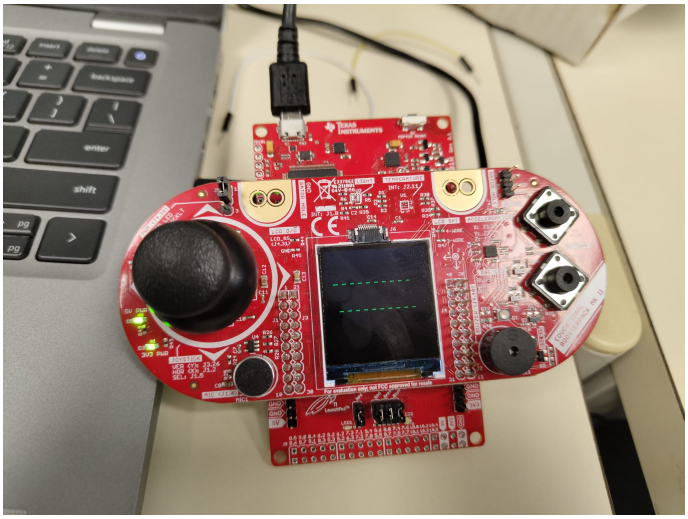
2.1 Counter: UP
Output mode: Toggle/Reset
#define COUNT 32768/10
#define COUNT2 32768/10/2



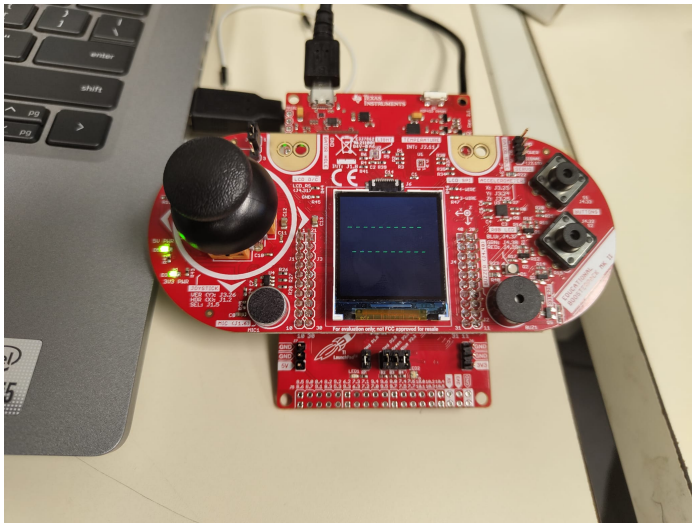
2.2 Counter: UP
Output mode: Toggle
#define COUNT 32768/20
#define COUNT2 32768/20/2



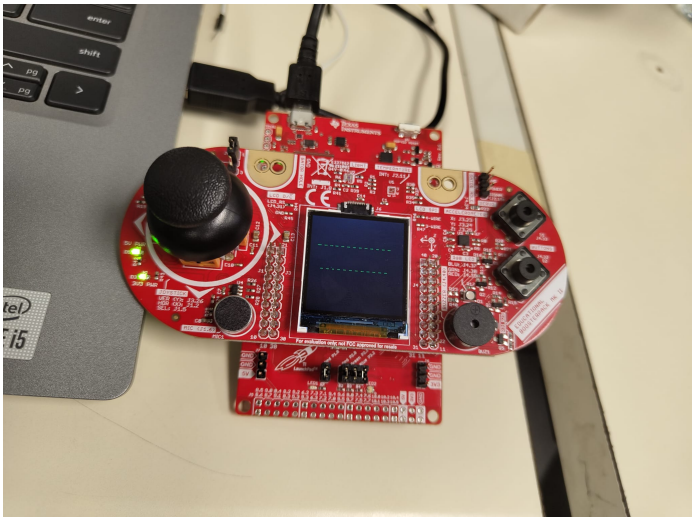
2.3 Counter: UP
Output mode: Reset/Set
#define COUNT 32768/10
#define COUNT2 32768/20



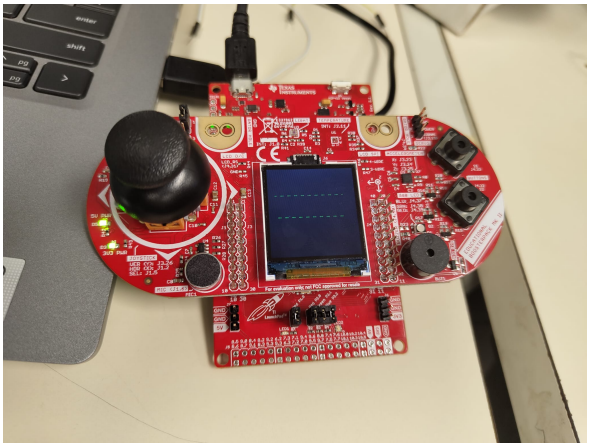
2.4 Counter: UPDOWN
Output mode: Toggle
#define COUNT 32768/20
#define COUNT2 32768/40



2.5 Output mode: Toggle/Reset
#define COUNT 32768/25
#define COUNT2 32768/25/2



2.6 Counter: UPDOWN
Output mode: Reset/Set
#define COUNT 32768/20
#define COUNT 2 0



How to calculate the values of TACCR0 and TACCR1?

50% of duty cycle & 10Hz

Count = 2x cycles

Count2 = x cycles

$$2x \times \frac{1}{32\,kHz} = \frac{1}{10}$$

$$x = \frac{32kHz}{20}$$

$$2x = \frac{32kHz}{10}$$

3. Code

```
#include <SPI.h>
#include <LCD_screen.h>
#include <LCD_screen_font.h>
#include <LCD_utilities.h>
#include <Screen_HX8353E.h>
#include <Terminal12e.h>
#include <Terminal6e.h>
#include <Terminal8e.h>
// Define screen
Screen_HX8353E myScreen;
#define MSP432P4111
#include "msp.h"
#include <energia.h>
#include "SPI.h"
#include <stdint.h>
#include <stdio.h>

#define MAXDOTS 100
int pwm_data[MAXDOTS];
void LCD_plot(void);


//FOR Timer_A0
#define UP 0x0010
#define UPDOWN 0x0030
#define CONT 0x0020
#define HALT 0x0000
//32kHz ACLK (32768)
#define ACLK 0x0100

#define COUNT  32768/5
#define COUNT2 32768/5/2

void setup() {
  WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;   // stop watchdog timer
  myScreen.begin();
  myScreen.setPenSolid(true);
  pinMode(74, INPUT_PULLUP);  ////74 -> P1.4 pushbotton
  pinMode(34, INPUT_PULLUP);////34 -> p2_3
```

```
  ////**********  please modify the following code  ***********////

  ////setting up timer_A0 connected to ACLK in UP mode(counting modes)
  TIMER_A0->CTL = (UP | ACLK);

  TIMER_A0->CCR[0] = 1600;
  ////setting up P2.5 as output mode.
  P2->DIR =  BIT5;
  ////setting up P2.5 connected to timer0.
  P2->SEL0 |= BIT5;
  P2->SEL1 &= ~(BIT5);
  ////setting up timer output mode as toggle(mode 3) (connected to P2.5)
  TIMER_A0->CCTL[2] = TIMER_A_CCTLN_OUTMOD_4;
  ////setting up duty cycle (connected to P2.5)
  TIMER_A0->CCR[2] = 800;

  ////**********  please modify the above code  ***********////
}
void loop() {
  LCD_plot();
  delay(0.1);
}
void LCD_plot(void){
  if(digitalRead(74)==0){
    myScreen.dRectangle(5, 30, 10+100, 50, blackColour);
    while(digitalRead(34)==1){continue;}
    for(int loops=0;loops<MAXDOTS;loops++){
      int buttonState = digitalRead(34);
      // print out the state of the button:
      pwm_data[loops] = buttonState;
      Serial.println(buttonState);
      delay(10);////10ms
    }
    for(int i=0;i<MAXDOTS;i++) myScreen.dRectangle(10+(i), 40+ (pwm_data[i]<<5),
1, 1, greenColour);
  }
}

//void TA0_0_IRQHandler(void){
//}
```

4. Conclusion

The 50% duty cycle square wave is generated and displayed.