Lab 4: PWM and USBUART

Instructor's Guide

Lab Introduction

This lab introduces a simple control signal for power or lighting and a method for communicating with the PSoC over USB. This is the first lab that doesn't introduce concepts that will be used in future labs, but introduces concepts that may be used in the final project.

Note that some of the content of this lab was cut due to the initial lab being too long. This content has been placed into the incomplete Lab 7.

Preparation

There are two things that need to be put on the computers used for this lab.

PuTTY

PuTTY is an SSH terminal that can also be a serial terminal. This program is necessary for communicating with the PSoC with USBUART. If desired, another serial terminal application could be used instead. To get PuTTY, visit here: http://www.putty.org/

Note: either download putty.exe or the Windows installer. Ensure the version installed is at least 0.61, as 0.60 (and maybe earlier) had a bug that caused serial connections to hang.

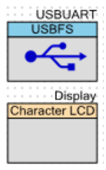
USBUART Driver

The USBUART component creates a driver that needs to be installed in order for the PSoC to be recognized by the computer. This driver has been provided with this document, as it should be identical to the one generated with the default USBUART settings. Note that this driver is unsigned, so there are some complications when installing this on Windows 8 and 8.1. See Part 3 of Procedure Part A for instructions for how to install this driver. Those instructions are provided to the student in case they are using their personal computer to complete the lab.

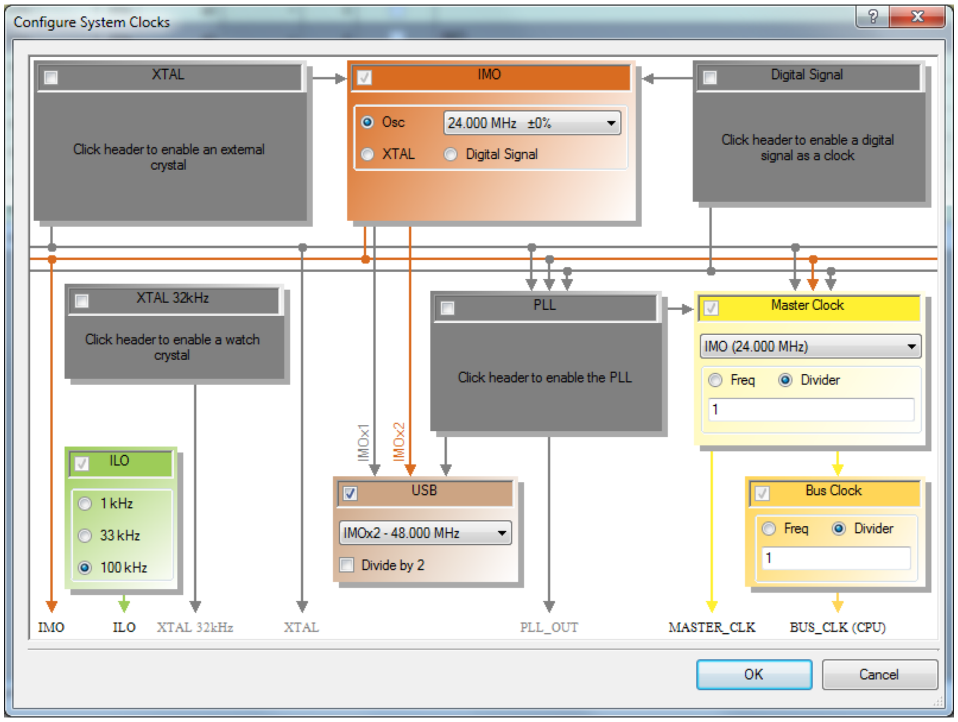
Instructor Review

Part A

Here is what the schematic should look like:



This lab requires modification to how the system clocks are configured in order to work because the USB requires a fairly stable 48 MHz clock. There are two ways to do this, and the first way is depicted in the lab itself. To access the dialog for changing the system clocks, go to the Clocks tab in the DWR file in the project. Select one of the system clocks and click Edit Clock… and see if it looks like one of these two images. If the dialog looks different, the most important things are that ILO is 100 kHz and the USB box has a 48 MHz clock as an input, or a 98 MHz clock input with the "Divide by 2" box checked.


Configure System Clocks
XTAL
Click header to enable an external crystal
IMO
Digital Signal
Click header to enable a digital signal as a clock
XTAL 32kHz
Click header to enable a watch crystal
PLL
Click header to enable the PLL
Master Clock
ILO
USB
Bus Clock
IMOx1
IMOx2
IMO
ILO
XTAL 32kHz
XTAL
PLL_OUT
MASTER_CLK
BUS_CLK (CPU)
OK
Cancel

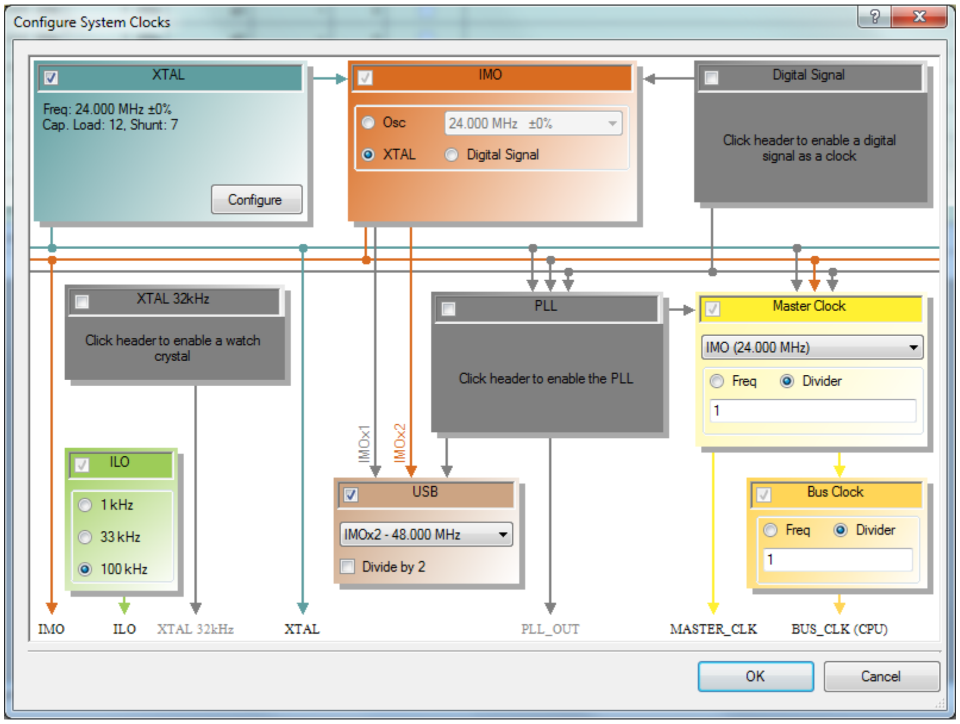
IMO: [✓ Osc] 24.000 MHz ±0%; XTAL; Digital Signal

Master Clock: IMO (24.000 MHz); Freq [✓ Divider] 1

ILO: 1 kHz; 33 kHz; [✓ 100 kHz]

USB: IMOx2 - 48.000 MHz; Divide by 2 (unchecked)

Bus Clock: Freq [✓ Divider] 1



XTAL: Freq: 24.000 MHz ±0%
Cap. Load: 12, Shunt: 7

IMO: Osc 24.000 MHz ±0%; [✓ XTAL]; Digital Signal

Master Clock: IMO (24.000 MHz); Freq [✓ Divider] 1

ILO: 1 kHz; 33 kHz; [✓ 100 kHz]

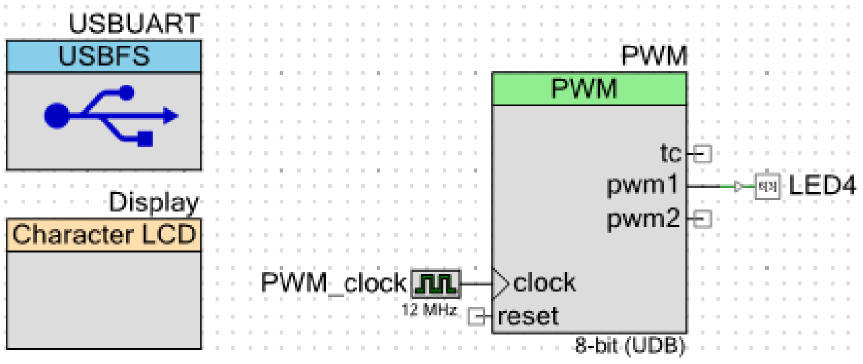
USB: IMOx2 - 48.000 MHz; Divide by 2 (unchecked)

Bus Clock: Freq [✓ Divider] 1

The solution for the firmware is provided in entirety for this part of the lab.

Part B

This part requires an update to the schematic. Here is what the updated schematic should look like:



Here is a sample firmware solution:

```c
#include <project.h>
#include "stdio.h"
#include "inttypes.h"

#define FALSE 0
#define TRUE 1

//The size of the USB buffer
#define BUF_SIZE 64

//The maximum number of digits to accept from the terminal/USB UART
#define NUMBUF_MAX 3

#define CHAR_NULL '\0'
#define CHAR_BACKSP 0x7F
#define CHAR_ENTER 0x0D
#define LOW_DIGIT '0'
#define HIGH_DIGIT '9'

//Prints the current brightness of the LED to the display
void printBrightnesses(uint8 led4Bright)
{
    Display_ClearDisplay();
    Display_Position(0, 0);
    Display_PrintString("LED4: ");
    Display_PrintNumber(led4Bright);
}
```

```
//Utility that writes a character to the USB UART
void usbPutChar(char c)
{
    /* Wait till component is ready to send more data to the PC */
    while(USBUART_CDCIsReady() == 0);
    /* Send data back to PC */
    USBUART_PutChar(c);
}

int main()
{
    //Create and initialize local variables
    uint8 usbStarted = FALSE;
    uint8 usbBuffer[BUF_SIZE];
    uint16 usbBufCount;
    char numBuf[NUMBUF_MAX+1] = "\0\0\0";
    uint8 numBufCount = 0;
    uint8 led4Bright = 0;

    CyGlobalIntEnable;

    /* Start USBFS Operation with 3V operation */
    USBUART_Start(0, USBUART_3V_OPERATION);

    /* Start LCD */
    Display_Start();

    //Initialize PWM
    PWM_Start();

    //Set the initial brightness of the LED
    PWM_WriteCompare1(led4Bright);

    //Print the LED brightness
    printBrightnesses(led4Bright);

    for(;;)
    {
        if(!usbStarted)
        {
            /* Wait for Device to enumerate */
            if(USBUART_GetConfiguration())
            {
                /* Enumeration is done, enable OUT endpoint for
                receive data from Host */
                USBUART_CDC_Init();
                usbStarted = TRUE;
            }
        }
        else
        {
            /* Check for input data from PC */
            if(USBUART_DataIsReady() != 0)
            {
                /* Read received data and re-enable OUT endpoint */
                usbBufCount = USBUART_GetAll(usbBuffer);
                int i;
                //Process each character individually
                for(i=0; i<usbBufCount; ++i)
                {
                    //If the character is a digit and there is still room in
                    //  the number buffer, place it in the buffer
                    if(usbBuffer[i] >= LOW_DIGIT &&
                       usbBuffer[i] <= HIGH_DIGIT &&
                       numBufCount < NUMBUF_MAX)
                    {
                        numBuf[numBufCount] = usbBuffer[i];
                        ++numBufCount;
                        usbPutChar(usbBuffer[i]);
                    }
                    //If the character is backspace and there are characters
                    //  in the number buffer, remove a character from the
                    //  buffer (make it null to mark the end)
                    else if(usbBuffer[i] == CHAR_BACKSP && numBufCount > 0)
                    {
                        --numBufCount;
                        numBuf[numBufCount] = CHAR_NULL;
                        usbPutChar(CHAR_BACKSP);
                    }
                    //If the character is enter and there are characters in
                    //  the number buffer, convert the number buffer to a
                    //  uint8, update the brightness, and print it.
                    //  Also, send a CRLF to the terminal so the user can
                    //  start typing on the next line.
                    else if(usbBuffer[i] == CHAR_ENTER && numBufCount > 0)
                    {
                        sscanf(numBuf, "%" SCNu8, &led4Bright);
                        PWM_WriteCompare1(led4Bright);
                        printBrightnesses(led4Bright);
                        while(USBUART_CDCIsReady() == 0);
                        USBUART_PutCRLF();
                        int x;
                        for(x = 0; x<NUMBUF_MAX; ++x)
                        {
                            numBuf[x] = CHAR_NULL;
                        }
                        numBufCount = 0;
                    }
                }
            }
        }
    }
}
```