# DMX512 Controller Receiver

Generated by Doxygen 1.8.15

## Chapter 1

# DMX-512 Controller Receiver

For CSE 4342: Embedded II Spring 2019
Instructor: Dr. Jason Losh
Made by: Satej Mhatre, Matthew Hilliard

# Chapter 2

# DMX512

This project uses the TM4C123GXL Launchpad from Texas Instruments, which features a TM4C123GH6PM ARM processor. This processor is a feature-packed processor and in this project, we attempt to explore some of the functionality it offers, by setting it up as a DMX-512 Controller and Receiver. The Launchpad is also connected to a SN75HVD12 RS-485 transceiver, which allows the DMX data to be effortlessly used by 512 devices, by connecting it to the data lines in parallel over a long distance. The Launchpad sends signals to the transceiver by utilizing the UART1 module on the TM4C123GH6PM processor. Similar setups are made where one module is a controller and the rest are devices. For the code, a reference manual is attached with this project report which contains descriptions of all variables and functions and what they are used for.

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

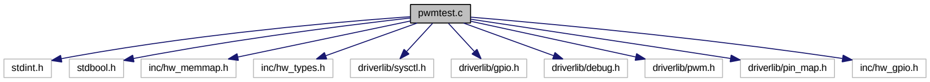
# Chapter 4

# File Documentation

## 4.1 pwmtest.c File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
```

Include dependency graph for pwmtest.c:



### Macros

- #define PWM_FREQUENCY 55

### Functions

- int maein (void)

### 4.1.1 Macro Definition Documentation

#### 4.1.1.1 PWM_FREQUENCY

```
#define PWM_FREQUENCY 55
```

Definition at line 12 of file pwmtest.c.

### 4.1.2 Function Documentation

#### 4.1.2.1 maein()

```
int maein (
            void  )
```

Definition at line 14 of file pwmtest.c.

## 4.2 pwmtest.c

```
00001 #include <stdint.h>
00002 #include <stdbool.h>
00003 #include "inc/hw_memmap.h"
00004 #include "inc/hw_types.h"
00005 #include "driverlib/sysctl.h"
00006 #include "driverlib/gpio.h"
00007 #include "driverlib/debug.h"
00008 #include "driverlib/pwm.h"
00009 #include "driverlib/pin_map.h"
00010 #include "inc/hw_gpio.h"
00011
00012 #define PWM_FREQUENCY 55
00013
00014 int maein(void)
00015 {
00016     volatile uint32_t ui32Load;
00017     volatile uint32_t ui32PWMClock;
00018     volatile uint8_t ui8Adjust;
00019     ui8Adjust = 83;
00020
00021     SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
00022     SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
00023
00024     SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
00025     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
00026
00027     GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
00028     GPIOPinConfigure(GPIO_PD0_M1PWM0);
00029
00030     ui32PWMClock = SysCtlClockGet() / 64;
00031     ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
00032     PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
00033     PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);
00034
00035     PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
00036     PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
00037     PWMGenEnable(PWM1_BASE, PWM_GEN_0);
00038
00039     while(1)
00040     {
00041     }
00042 }
```

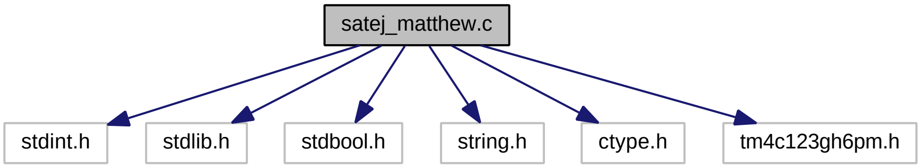
## 4.3 README.md File Reference

## 4.4 satej_matthew.c File Reference

File containing everything for the DMX Controller Receiver Project.
For CSE 4342: Embedded II Spring 2019
Instructor: Dr. Jason Losh

```
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <ctype.h>
#include "tm4c123gh6pm.h"
```

Include dependency graph for satej_matthew.c:



### Macros

- #define RED_LED (∗((volatile uint32_t ∗)(0x42000000 + (0x400253FC-0x40000000)∗32 + 1∗4)))
- #define GREEN_LED (∗((volatile uint32_t ∗)(0x42000000 + (0x400253FC-0x40000000)∗32 + 3∗4)))
- #define BLUE_LED (∗((volatile uint32_t ∗)(0x42000000 + (0x400253FC-0x40000000)∗32 + 2∗4)))
- #define PUSH_BUTTON (∗((volatile uint32_t ∗)(0x42000000 + (0x400253FC-0x40000000)∗32 + 4∗4)))
- #define PUSH_BUTTON2 (∗((volatile uint32_t ∗)(0x42000000 + (0x400253FC-0x40000000)∗32 + 0∗4)))
- #define GREEN_LED_MASK 8
- #define RED_LED_MASK 2
- #define BLUE_LED_MASK 4
- #define PUSH_BUTTON_MASK 16
- #define PUSH_BUTTON2_MASK 1
- #define delay4Cycles() __asm(" NOP\n NOP\n NOP\n NOP")
- #define delay1Cycle() __asm(" NOP\n")
- #define delay6Cycles() __asm(" NOP\n NOP\n NOP\n NOP\n NOP\n NOP\n")

### Functions

- void animationRamp ()

  *Function to enable ramping animation.*
- void clearStr ()

  *Function to clear command, arg1, and arg2 arrays.*
- char getcUart0 ()

*Blocking function that returns with serial data once the buffer is not empty.*

- void getModeEE ()

  *Function to get the launchpad mode from EEPROM.*
- char ∗ intToChar (uint16_t x)

  *Function to convert integer to character for UART0.*
- bool isLetter (char c)

  *Function to check if character is letter.*
- bool isNumber (char c)

  *Function to check if character is number.*
- uint8_t main ()

  *Runs everything.*
- void printCommandList ()

  *Function to print available commands to user.*
- void putcUart1 (uint8_t i)

  *Function to send characters to UART0.*
- void Uart0Isr ()

  *Function to handle UART0 interrupts.*
- void waitMicrosecond (uint32_t us)

  *Function to wait for specified microseconds.*
- void wooone ()

  *Function to set all DMX values to 255.*
- void putsUart0 (char ∗str)

  *Blocking function that writes a string when the UART buffer is not full.*
- void changeTimer1Value (uint32_t us)

  *Function to change load value of Timer1.*
- void initHw ()

  *Function to initialize all required hardware functions.*
- void Uart1Isr ()

  *Function to Handle Interrupts from UART1.*
- void Timer2ISR ()

  *Function to Handle Interrupts from Timer2.*
- void Timer1ISR ()

  *Function to handle TIMER1 interrupts.*
- void putcUart0 (char c)

  *Blocking function that writes a serial character when the UART buffer is not full.*
- void EEWRITE (uint16_t B, uint16_t offSet, uint16_t val)

  *Function to write to EEPROM to set address.*
- void clearDMX ()

  *Function to clear DMX data bins.*
- uint8_t parseCommand ()

  *Function to parse commands from UART0 and execute functions or set flags.*
- void sweepServo ()

  *Function to sweep servo.*

### Variables

- char command [20]
- char arg1 [20]
- char arg2 [20]
- int8_t enteringField = 0
- int8_t pos = 0
- uint16_t maxAddress = 512
- uint8_t continuous = 0
- uint16_t DMXMode = 0
- uint16_t deviceModeAddress = 0
- uint8_t prevRX = 0
- uint8_t rxError = 0
- uint16_t rxState = 0
- float seconds = 0
- int upR
- int upG
- int upB
- int goR
- int goG
- int goB
- float secondsTrigger = 0.0
- uint16_t dimStart = 0
- uint16_t dimEnd = 0
- float dimValue = 0
- uint8_t woo = 0
- int servoDir = 0
- char ch [3]
- uint8_t vall = 8
- uint8_t incr = 1
- uint16_t program
- uint16_t Address
- uint16_t opMode
- uint16_t setval
- uint8_t mode = 0
- uint8_t dmxData [512]
- uint8_t RGBMode = 0

## 4.4.1 Detailed Description

File containing everything for the DMX Controller Receiver Project.
For CSE 4342: Embedded II Spring 2019
Instructor: Dr. Jason Losh

#### Author

Satej Mhatre, Matthew Hilliard

#### Date

1 May 2019

**Hardware Target:**

Target Platform: EK-TM4C123GXL Evaluation Board
Target uC: TM4C123GH6PM
System Clock: 40 MHz

**Hardware configuration:**

Red LED:
PF1 drives an NPN transistor that powers the red LED
Blue LED:
PF2 drives an NPN transistor that powers the green LED
Green LED:
PF3 drives an NPN transistor that powers the green LED
UART Interface:
U0TX (PA1) and U0RX (PA0) are connected to the 2nd controller
U1TX (PA1) and U1RX (PA0) are used for DMX Data Transmit and Receive
Other Interface:
PD0, PD1, PD2, PD3 is connected to a mux that reads the value from a DIP switch
PF1, PF2, PF3 are also configured as PWM outputs to control servos and LEDs on-board.
To Do:
PD6, PD7 will be connected to a ESP8266-01 that will serve a webpage for UART communication so that launchpad can be controlled without physically using a USB cable.
The USB on the 2nd controller enumerates to an ICDI interface and a virtual COM port
Configured to 115,200 baud, 8N1

Definition in file satej_matthew.c.

## 4.4.2 Macro Definition Documentation

### 4.4.2.1 BLUE_LED

```
#define BLUE_LED (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 2*4)))
```

Bit banding for PORTF2 Blue LED

Definition at line 59 of file satej_matthew.c.

### 4.4.2.2 BLUE_LED_MASK

```
#define BLUE_LED_MASK 4
```

GPIO PORTF Blue LED Mask

Definition at line 74 of file satej_matthew.c.

### 4.4.2.3 delay1Cycle

```
#define delay1Cycle( ) __asm(" NOP\n")
```

Delaying for 1 cycle

Definition at line 85 of file satej_matthew.c.

### 4.4.2.4 delay4Cycles

```
#define delay4Cycles( ) __asm(" NOP\n NOP\n NOP\n NOP")
```

Delaying for 4 cycles

Definition at line 83 of file satej_matthew.c.

### 4.4.2.5 delay6Cycles

```
#define delay6Cycles( ) __asm(" NOP\n NOP\n NOP\n NOP\n NOP\n NOP\n")
```

Delaying for 6 cycles

Definition at line 87 of file satej_matthew.c.

### 4.4.2.6 GREEN_LED

```
#define GREEN_LED (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 3*4)))
```

Bit banding for PORTF3 GREEN LED

Definition at line 56 of file satej_matthew.c.

### 4.4.2.7 GREEN_LED_MASK

```
#define GREEN_LED_MASK 8
```

GPIO PORTF Green LED Mask

Definition at line 68 of file satej_matthew.c.

#### 4.4.2.8 PUSH_BUTTON

```
#define PUSH_BUTTON (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 4*4)))
```

Bit banding for PORTF4 PushButton 1

Definition at line 62 of file satej_matthew.c.

#### 4.4.2.9 PUSH_BUTTON2

```
#define PUSH_BUTTON2 (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 0*4)))
```

Bit banding for PORTF0 PushButton 0

Definition at line 65 of file satej_matthew.c.

#### 4.4.2.10 PUSH_BUTTON2_MASK

```
#define PUSH_BUTTON2_MASK 1
```

GPIO PORTF Push Button 2 Mask

Definition at line 80 of file satej_matthew.c.

#### 4.4.2.11 PUSH_BUTTON_MASK

```
#define PUSH_BUTTON_MASK 16
```

GPIO PORTF Push Button 1 Mask

Definition at line 77 of file satej_matthew.c.

#### 4.4.2.12 RED_LED

```
#define RED_LED (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 1*4)))
```

Bit banding for PORTF1 Red LED

Definition at line 53 of file satej_matthew.c.