Pomona College

Computer Science

Fall 2024: CSCI 181RT

Real-Time Systems in the Real World

Lecture 22

Tuesday, November 12, 2024
Edmunds Hall 105
2:45 PM - 4:00 PM

Professor Jennifer DesCombes

Agenda

- Go Backs
- Discussion on Assignments
- Software Package Structure - Functional & Data Privacy
- Lab #11 Preview
- Look Ahead
- Assignment
- Action Items

Go Backs

- General?
- Announcements
  - Lab Tomorrow (13 November) Will End at 3:30
  - No Lecture on Tuesday, 26 November 2024
- Action Item Status
  - AI240910-2: Find recommended book on computer architecture.
  - AI240924-1: At what point as a development team grows does it make sense to have dedicated software and integration testers?
  - AI241024-1: Provide documentation on how to disable compiler optimization.
  - AI241107-1: Generate drawing showing location of Task Test Points on evaluation board.

Discussion on Readings

- The Soul Of A New Machine
  - Chapters 5 and 6: Midnight Programmer, Flying Upside Down

Discussion on Readings

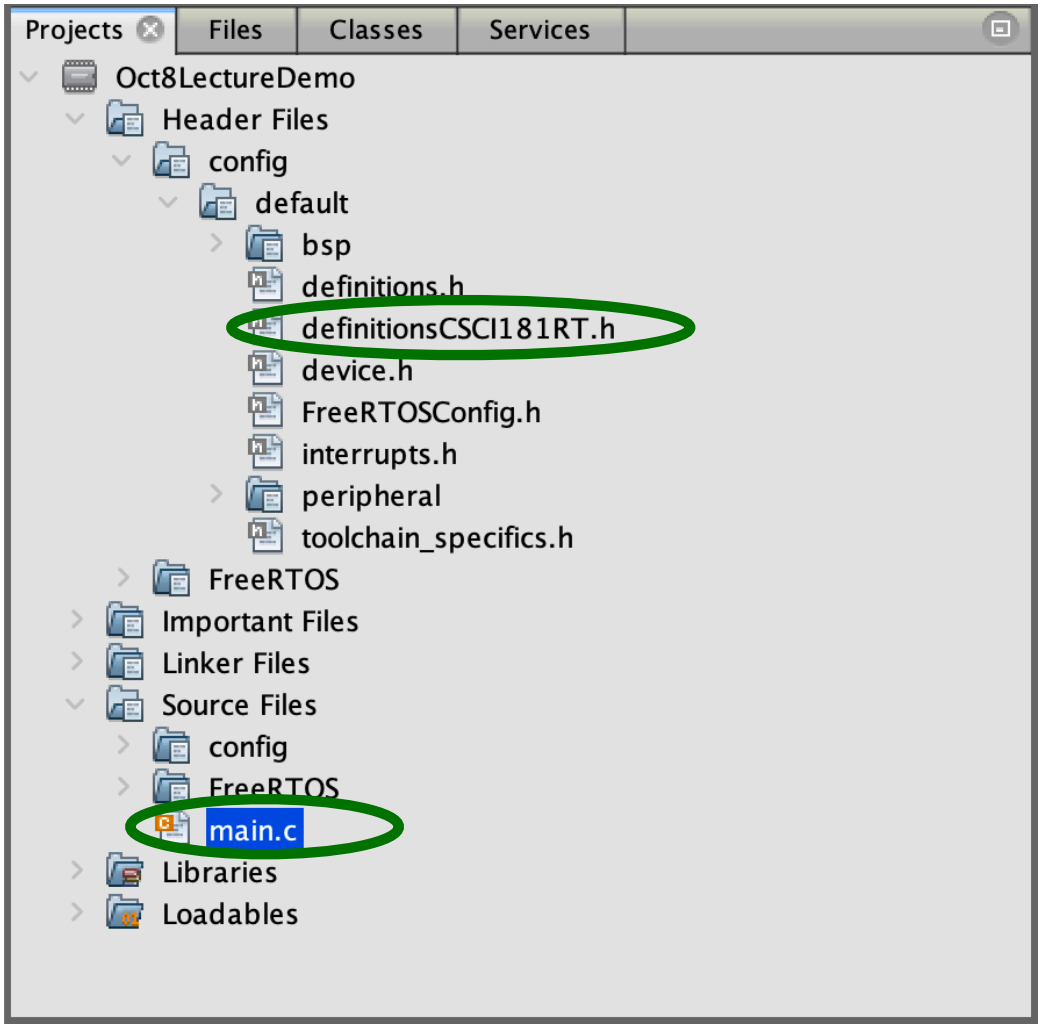
- The Soul Of A New Machine
  - Chapters 5 and 6: Midnight Programmer, Flying Upside Down

Software Package Structure - Functional & Data Privacy

- Arrangement of Files and Folders
- Arrangement of Code in Project Viewer
- Minimize Code Interdependency / Knowledge
- Reduce Risk of Inadvertent Edits / Corruption
- Maximize Reuse of Functions
- Maximize Understanding of Each File

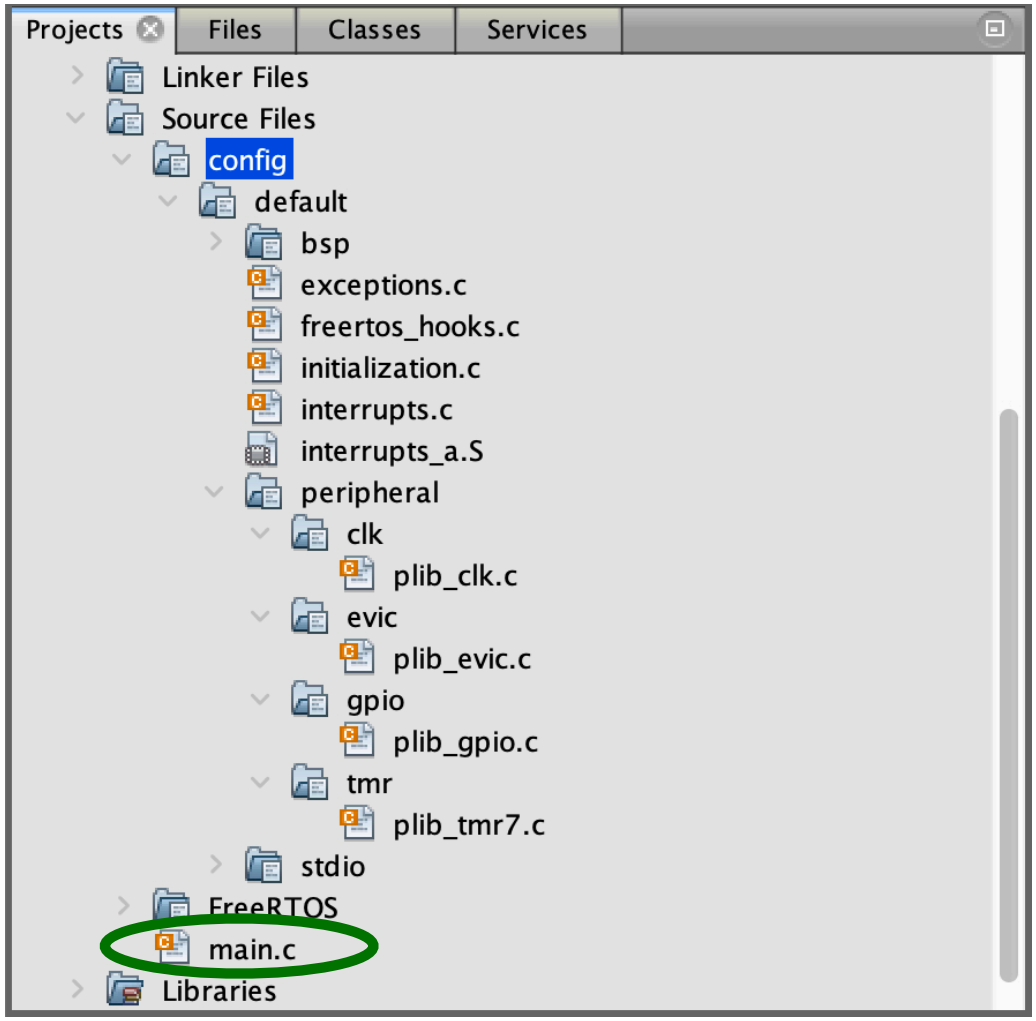
Software Package Structure - MPLab Structure

- Base Structure of Distributed RTOS Lab Template in MPLab
- Added Files:
  - main
  - definitionsCSCI181RT.h
- All Code in Single File



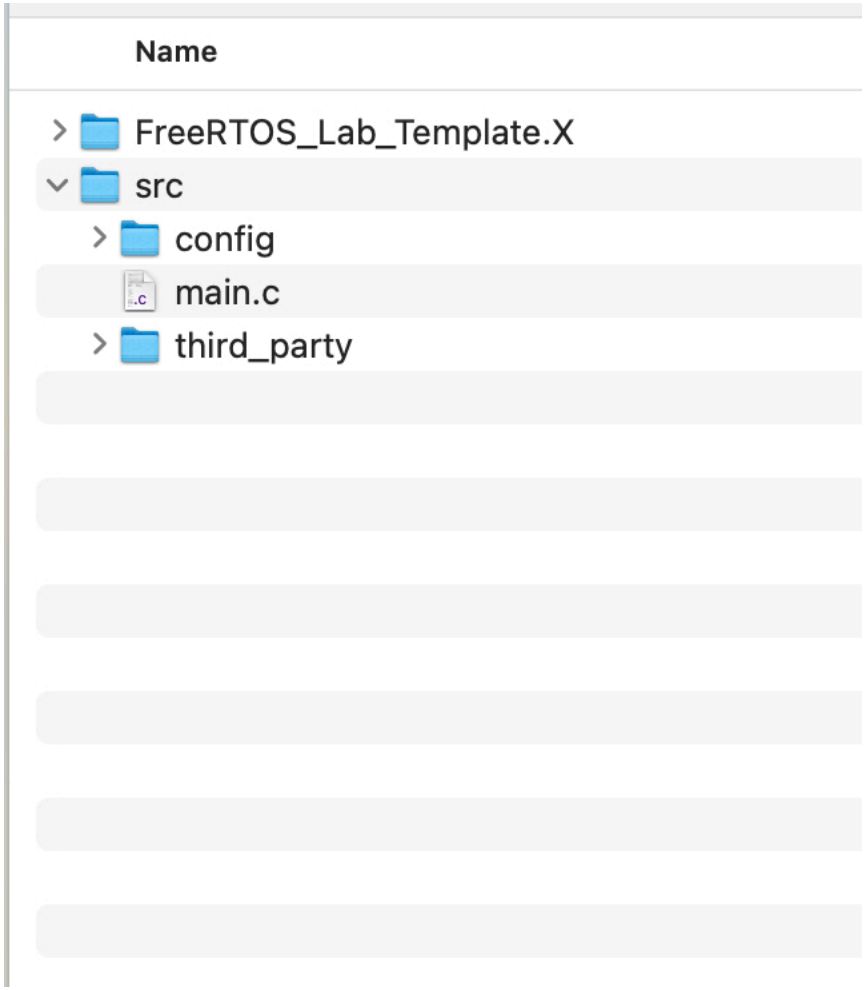
Software Package Structure - MPLab Structure

- Microchip Code Files Partitioned / Divided by Functionality
- Include Files are Also Similarly Partitioned



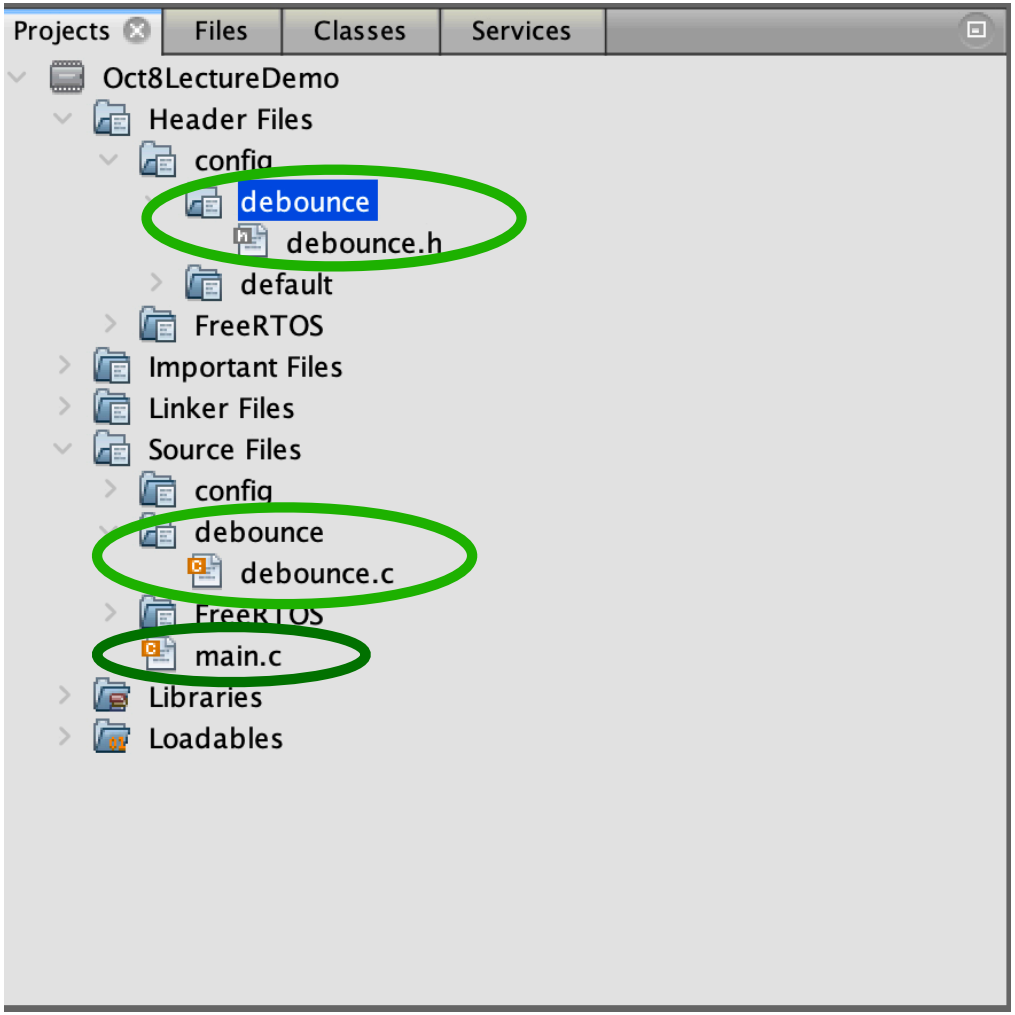
Software Package Structure - Host Structure

- File Folder Structure on Host Computer
- Not Necessarily Same as MPLab File Structure



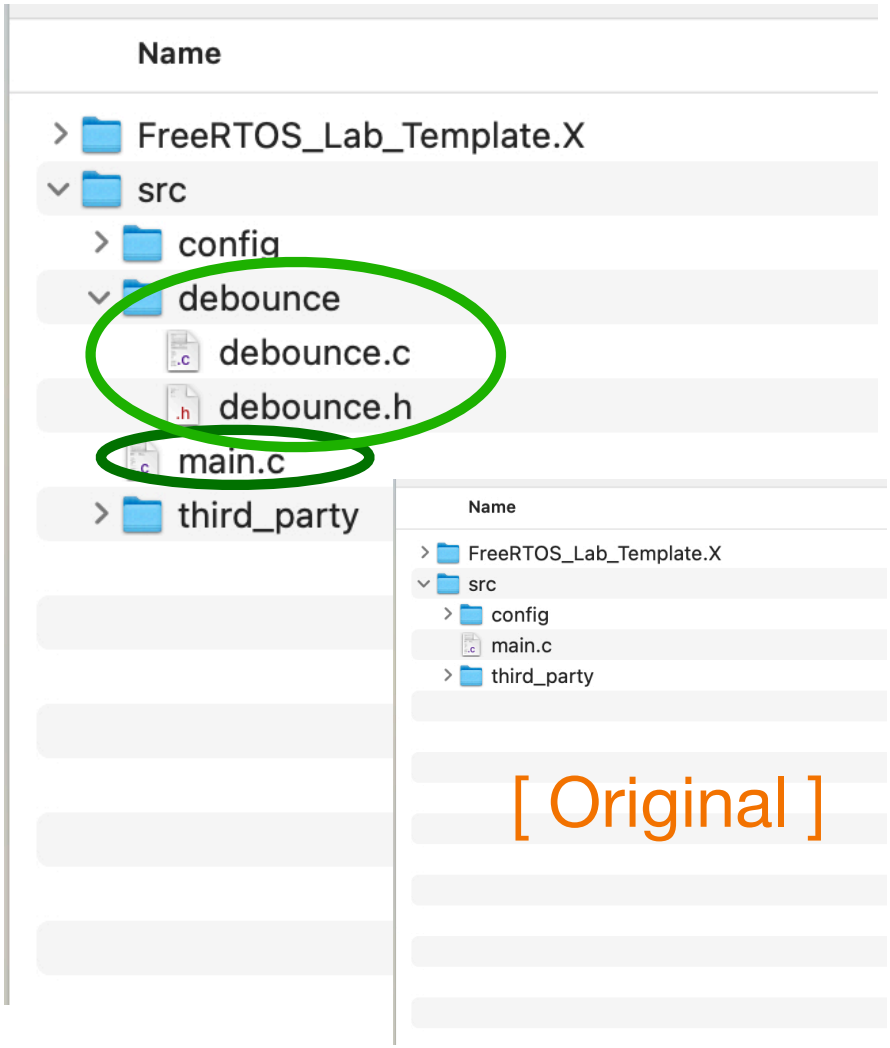
Software Package Structure - MPLab Structure

- Debounce Function Used as Case Study
- Logical Folders Used Within MPLab (right click)
- Not related to Physical Location of Source File
-



Software Package Structure - Host Structure

- File Folder Structure on Host Computer
- Not Necessarily Same as MPLab File Structure
- Both .c and .h in Same Folder for Portability
- Other Files Could Be Included
  - Test Files
  - Functional Descriptive Files
  - ???



Software Package Structure - main.c

- Original main.c Includes and Data Types
- “definitionsCSCI181RT.h” is only added file
- typedef for Structures Used Within main.c

```c
#include <stddef.h>
#include <stdbool.h>
#include <stdlib.h>
#include "definitions.h"
#include "definitionsCSCI181RT.h"
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
```

```c
//
// Type Declarations
//
```

```c
typedef struct {
    int buttonValue;
    int buttonSameCount;
    bool buttonStatusChange;
} debounceStruct;
```

```c
typedef struct {
    int pwmLevelMinCount;
    int pwmLevelMaxCount;
    int pwmCountMaximum;
    int pwmCount;
} pwmStruct;
```

Software Package Structure - main.c

- Improved main.c Includes and Data Types
- “debounce.h” is added
  - Note Folder Directory Listing
- The typedef for ‘debounceStruct’ has been removed from main.c

```
#include <stddef.h>
#include <stdbool.h>
#include <stdlib.h>
#include "definitions.h"
#include "definitionsCSCI181RT.h"
#include "debounce/debounce.h"

#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
```

```
//
// Type Declarations
//

typedef struct {
    int pwmLevelMinCount;
    int pwmLevelMaxCount;
    int pwmCountMaximum;
    int pwmCount;
} pwmStruct;
```

Software Package Structure - main.c

- Improved main.c Includes and Types - Side-by-side

```c
#include <stddef.h>
#include <stdbool.h>
#include <stdlib.h>
#include "definitions.h"
#include "definitionsCSCI181RT.h"
#include "debounce/debounce.h"

#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
```

```c
//
// Type Declarations
//

typedef struct {
    int pwmLevelMinCount;
    int pwmLevelMaxCount;
    int pwmCountMaximum;
    int pwmCount;
} pwmStruct;
```

```c
#include <stddef.h>
#include <stdbool.h>
#include <stdlib.h>
#include "definitions.h"
#include "definitionsCSCI181RT.h"
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
```

[Original]

```c
//
// Type Declarations
//

typedef struct {
    int buttonValue;
    int buttonSameCount;
    bool buttonStatusChange;
} debounceStruct;

typedef struct {
    int pwmLevelMinCount;
    int pwmLevelMaxCount;
    int pwmCountMaximum;
    int pwmCount;
} pwmStruct;
```

Software Package Structure - main.c

- Original main.c Function Prototypes
- Includes Prototype for ‘debounceInput’ Method

```
//
// Method Definitions
//

void vP1Task(void *pvParameters);     // Highest Priority Task (P1)
void vP2Task(void *pvParameters);     // Task Priority 2 (P2)
void vP3Task(void *pvParameters);     // Task Priority 3 (P3)
void vP4Task(void *pvParameters);     // Task Priority 4 (P3)
void dummyTask(void *pvParameters);   // Lowest Priority Task
void monitorGPIO1PinState( void );
ledStates pwmProcessing( int currentLevel, pwmStruct *pwmStructure );
void debounceInput( int currentInput, debounceStruct *inputStructure );
void levelLED3Adjust( void );
```

Software Package Structure - main.c

- Improved main.c Function Prototypes
- Removes Prototype for ‘debounceInput’ Method
- Function Prototype for this Method Now Contained within “debounce.h’

```
//
// Method Definitions
//

void vP1Task(void *pvParameters);     // Highest Priority Task (P1)
void vP2Task(void *pvParameters);     // Task Priority 2 (P2)
void vP3Task(void *pvParameters);     // Task Priority 3 (P3)
void vP4Task(void *pvParameters);     // Task Priority 4 (P3)
void dummyTask(void *pvParameters);   // Lowest Priority Task
void monitorGPIO1PinState( void );
ledStates pwmProcessing( int currentLevel, pwmStruct *pwmStructure );
void levelLED3Adjust( void );
```

Software Package Structure - main.c

- Improved main.c Function Prototypes - Side-by-side

```
//
// Method Definitions
//

void vP1Task(void *pvParameters);     // Highest Priority Task (P1)
void vP2Task(void *pvParameters);     // Task Priority 2 (P2)
void vP3Task(void *pvParameters);     // Task Priority 3 (P3)
void vP4Task(void *pvParameters);     // Task Priority 4 (P3)
void dummyTask(void *pvParameters);   // Lowest Priority Task
void monitorGPIO1PinState( void );
ledStates pwmProcessing( int currentLevel, pwmStruct *pwmStructure );
void levelLED3Adjust( void );
```

[Original]

```
//
// Method Definitions
//

void vP1Task(void *pvParameters);     // Highest Priority Task (P1)
void vP2Task(void *pvParameters);     // Task Priority 2 (P2)
void vP3Task(void *pvParameters);     // Task Priority 3 (P3)
void vP4Task(void *pvParameters);     // Task Priority 4 (P3)
void dummyTask(void *pvParameters);   // Lowest Priority Task
void monitorGPIO1PinState( void );
ledStates pwmProcessing( int currentLevel, pwmStruct *pwmStructure );
void debounceInput( int currentInput, debounceStruct *inputStructure );
void levelLED3Adjust( void );
```

Software Package Structure - main.c

- Original main.c and improved main.c Have Identical Global Variables
- Globals Required When Multiple Tasks Share Data
  - Other Data Sharing Schemes Could be Implemented
  - Limitations on Data Integrity

```
//
// Global Data Declarations
//

// Global data used by fiveMSProcessing()
int gIntInputSignalState;
int gIntLevelControl;

// Global data used by PWM code
pwmStruct gPWMLED3;
pwmStruct gPWMLEDRGB;

// Global data used by button de-bounce code
debounceStruct gDebounceButton1;
debounceStruct gDebounceButton2;
debounceStruct gDebounceButton3;
debounceStruct gDebounceButton4;
```

Software Package Structure - main.c

- Original main.c Variable Initialization
- main.c Needs to Utilize / Understand Structure Elements

```
// Initialize global data
gIntInputSignalState = 1;
gIntLevelControl = levelInitialCount;
gPWMLED3.pwmLevelMaxCount = levelMaxCount;
gPWMLED3.pwmLevelMinCount = levelMinCount;
gPWMLED3.pwmCount = 0;
gPWMLED3.pwmCountMaximum = levelMaxCount;
gPWMLEDRGB.pwmLevelMaxCount = levelMaxCount;
gPWMLEDRGB.pwmLevelMinCount = levelMinCount;
gPWMLEDRGB.pwmCount = 0;
gPWMLEDRGB.pwmCountMaximum = levelMaxCount;
gDebounceButton1.buttonValue = 0;
gDebounceButton1.buttonSameCount = 0;
gDebounceButton1.buttonStatusChange = false;
gDebounceButton2.buttonValue = 0;
gDebounceButton2.buttonSameCount = 0;
gDebounceButton2.buttonStatusChange = false;
gDebounceButton3.buttonValue = 0;
gDebounceButton3.buttonSameCount = 0;
gDebounceButton3.buttonStatusChange = false;
gDebounceButton4.buttonValue = 0;
gDebounceButton4.buttonSameCount = 0;
gDebounceButton4.buttonStatusChange = false;
```

Software Package Structure - main.c

- Improved main.c Variable Initialization
- main.c **DOES NOT** Need to Utilize / Understand Structure Elements

```
// Initialize global data
gIntInputSignalState = 1;
gIntLevelControl = levelInitialCount;
gPWMLED3.pwmLevelMaxCount = levelMaxCount;
gPWMLED3.pwmLevelMinCount = levelMinCount;
gPWMLED3.pwmCount = 0;
gPWMLED3.pwmCountMaximum = levelMaxCount;
gPWMLEDRGB.pwmLevelMaxCount = levelMaxCount;
gPWMLEDRGB.pwmLevelMinCount = levelMinCount;
gPWMLEDRGB.pwmCount = 0;
gPWMLEDRGB.pwmCountMaximum = levelMaxCount;

initDebounceInput( &gDebounceButton1,     // Pointer to de-bounce data structure
                   3 );                   // Number of sequential states for de-bounce
initDebounceInput( &gDebounceButton2,     // Pointer to de-bounce data structure
                   3 );                   // Number of sequential states for de-bounce
initDebounceInput( &gDebounceButton3,     // Pointer to de-bounce data structure
                   3 );                   // Number of sequential states for de-bounce
initDebounceInput( &gDebounceButton4,     // Pointer to de-bounce data structure
                   3 );                   // Number of sequential states for de-bounce
```

Software Package Structure - main.c

- Improved main.c Variable Initialization - Side-by-side

```
// Initialize global data
gIntInputSignalState = 1;
gIntLevelControl = levelInitialCount;
gPWMLED3.pwmLevelMaxCount = levelMaxCount;
gPWMLED3.pwmLevelMinCount = levelMinCount;
gPWMLED3.pwmCount = 0;
gPWMLED3.pwmCountMaximum = levelMaxCount;
gPWMLEDRGB.pwmLevelMaxCount = levelMaxCount;
gPWMLEDRGB.pwmLevelMinCount = levelMinCount;
gPWMLEDRGB.pwmCount = 0;
gPWMLEDRGB.pwmCountMaximum = levelMaxCount;

initDebounceInput( &gDebounceButton1,     // Pointer to de-bounce data s
                   3 );                   // Number of sequential states
initDebounceInput( &gDebounceButton2,     // Pointer to de-bounce data s
                   3 );                   // Number of sequential states for de-bounce
initDebounceInput( &gDebounceButton3,     // Pointer to de-bounce data structure
                   3 );                   // Number of sequential states for de-bounce
initDebounceInput( &gDebounceButton4,     // Pointer to de-bounce data structure
                   3 );                   // Number of sequential states for de-bounce
```

```
// Initialize global data
gIntInputSignalState = 1;
gIntLevelControl = levelInitialCount;
gPWMLED3.pwmLevelMaxCount = levelMaxCount;
gPWMLED3.pwmLevelMinCount = levelMinCount;
gPWMLED3.pwmCount = 0;
gPWMLED3.pwmCountMaximum = levelMaxCount;
gPWMLEDRGB.pwmLevelMaxCount = levelMaxCount;
gPWMLEDRGB.pwmLevelMinCount = levelMinCount;
gPWMLEDRGB.pwmCount = 0;
gPWMLEDRGB.pwmCountMaximum = levelMaxCount;
gDebounceButton1.buttonValue = 0;
gDebounceButton1.buttonSameCount = 0;
gDebounceButton1.buttonStatusChange = false;
gDebounceButton2.buttonValue = 0;
gDebounceButton2.buttonSameCount = 0;
gDebounceButton2.buttonStatusChange = false;
gDebounceButton3.buttonValue = 0;
gDebounceButton3.buttonSameCount = 0;
gDebounceButton3.buttonStatusChange = false;
gDebounceButton4.buttonValue = 0;
gDebounceButton4.buttonSameCount = 0;
gDebounceButton4.buttonStatusChange = false;
```

[Original]

Software Package Structure - main.c

- Original main.c and improved main.c Have Identical Global Calls to ‘debounceInput’ Method

```
// De-bounce pushbutton inputs
debounceInput( GPIO_PinRead( PushButton_1 ), &gDebounceButton1 );
debounceInput( GPIO_PinRead( PushButton_2 ), &gDebounceButton2 );
debounceInput( GPIO_PinRead( PushButton_3 ), &gDebounceButton3 );
debounceInput( GPIO_PinRead( PushButton_4 ), &gDebounceButton4 );
```

Software Package Structure - main.c

- Original main.c Accessing of Pushbutton States
- main.c Needs to Utilize / Understand Structure Elements

```
// Decrease level by pressing button 3
if (gDebounceButton3.buttonStatusChange == true)
{
    if (gDebounceButton3.buttonValue == 0)  // NOTE: button pushed
    {
        gIntLevelControl -= 1;
        if (gIntLevelControl < levelMinCount)
        {
            gIntLevelControl = levelMinCount;
        }
    }
}

// Increase level by pressing button 4
if (gDebounceButton4.buttonStatusChange == true)
{
    if (gDebounceButton4.buttonValue == 0)  // NOTE: button pushed
    {
        gIntLevelControl += 1;
        if (gIntLevelControl > levelMaxCount)
        {
            gIntLevelControl = levelMaxCount;
        }
    }
}
```

Software Package Structure - main.c

- Improved main.c Accessing of Pushbutton States
- main.c **DOES NOT** Need to Utilize / Understand Structure Elements

```
// Decrease level by pressing button 3
if (getDebounceStatusChange( &gDebounceButton3 ) == true)
{
    if (getDebounceValue( &gDebounceButton3 ) == 0)  // NOTE: butt
    {
        gIntLevelControl -= 1;
        if (gIntLevelControl < levelMinCount)
        {
            gIntLevelControl = levelMinCount;
        }
    }
}

// Increase level by pressing button 4
if (getDebounceStatusChange( &gDebounceButton4 ) == true)
{
    if (getDebounceValue( &gDebounceButton4 ) == 0)  // NOTE: butt
    {
        gIntLevelControl += 1;
        if (gIntLevelControl > levelMaxCount)
        {
            gIntLevelControl = levelMaxCount;
        }
    }
}
```

Software Package Structure - main.c

- Improved main.c Accessing of Pushbutton States - Side-by-side

```c
// Decrease level by pressing button 3
if (getDebounceStatusChange( &gDebounceButton3 ) == true)
{
    if (getDebounceValue( &gDebounceButton3 ) == 0)  // NOTE: butt
    {
        gIntLevelControl -= 1;
        if (gIntLevelControl < levelMinCount)
        {
            gIntLevelControl = levelMinCount;
        }
    }
}

// Increase level by pressing button 4
if (getDebounceStatusChange( &gDebounceButton4 ) == true)
{
    if (getDebounceValue( &gDebounceButton4 ) == 0)  // NOTE: butt
    {
        gIntLevelControl += 1;
        if (gIntLevelControl > levelMaxCount)
        {
            gIntLevelControl = levelMaxCount;
        }
    }
}
```

```c
// Decrease level by pressing button 3
if (gDebounceButton3.buttonStatusChange == true)
{
    if (gDebounceButton3.buttonValue == 0)  // NOTE: button pushed
    {
        gIntLevelControl -= 1;
        if (gIntLevelControl < levelMinCount)
        {
            gIntLevelControl = levelMinCount;
        }
    }
}

// Increase level by pressing button 4
if (gDebounceButton4.buttonStatusChange == true)
{
    if (gDebounceButton4.buttonValue == 0)  // NOTE: button pushed
    {
        gIntLevelControl += 1;
        if (gIntLevelControl > levelMaxCount)
        {
            gIntLevelControl = levelMaxCount;
        }
    }
}
```

[Original]

Software Package Structure - main.c

- So Why Bother?
- Debounce Code Can be Enhanced / Modified Without Impacting Code Users
- Reduce Risk of Inadvertent Edits / Corruption
- Maximize Reuse of Functions
- Maximize Understanding of Each File

```c
// NOTE: do not access the individual elements of the structure
// as the type, name, and location of these elements may change as
// improvements are made to the code. Use the access methods to
// access information about the de-bounced pushbutton.
typedef struct {
    int buttonEndCount;
    int buttonValue;
    int buttonSameCount;
    bool buttonStatusChange;
    int buttonTime;
} debounceStruct;
```

```c
/*
//  Initialize data within the debounce structure
*/
void initDebounceInput( debounceStruct *inputStructure, int debounceMaxCount )
{

    inputStructure->buttonEndCount = debounceMaxCount;
    inputStructure->buttonValue = 0;
    inputStructure->buttonSameCount = 0;
    inputStructure->buttonStatusChange = false;
    inputStructure->buttonTime = 0;

}
```

[Code Compiled and Executed Properly Without Modifying Other Files]

Software Package Structure - main.c

- What Additional Partitioning Should Be Performed?

```
//
// Method Definitions
//

void vP1Task(void *pvParameters);     // Highest Priority Task (P1)
void vP2Task(void *pvParameters);     // Task Priority 2 (P2)
void vP3Task(void *pvParameters);     // Task Priority 3 (P3)
void vP4Task(void *pvParameters);     // Task Priority 4 (P3)
void dummyTask(void *pvParameters);   // Lowest Priority Task
void monitorGPIO1PinState( void );
ledStates pwmProcessing( int currentLevel, pwmStruct *pwmStructure );
void levelLED3Adjust( void );
```

Lab 11 Preview

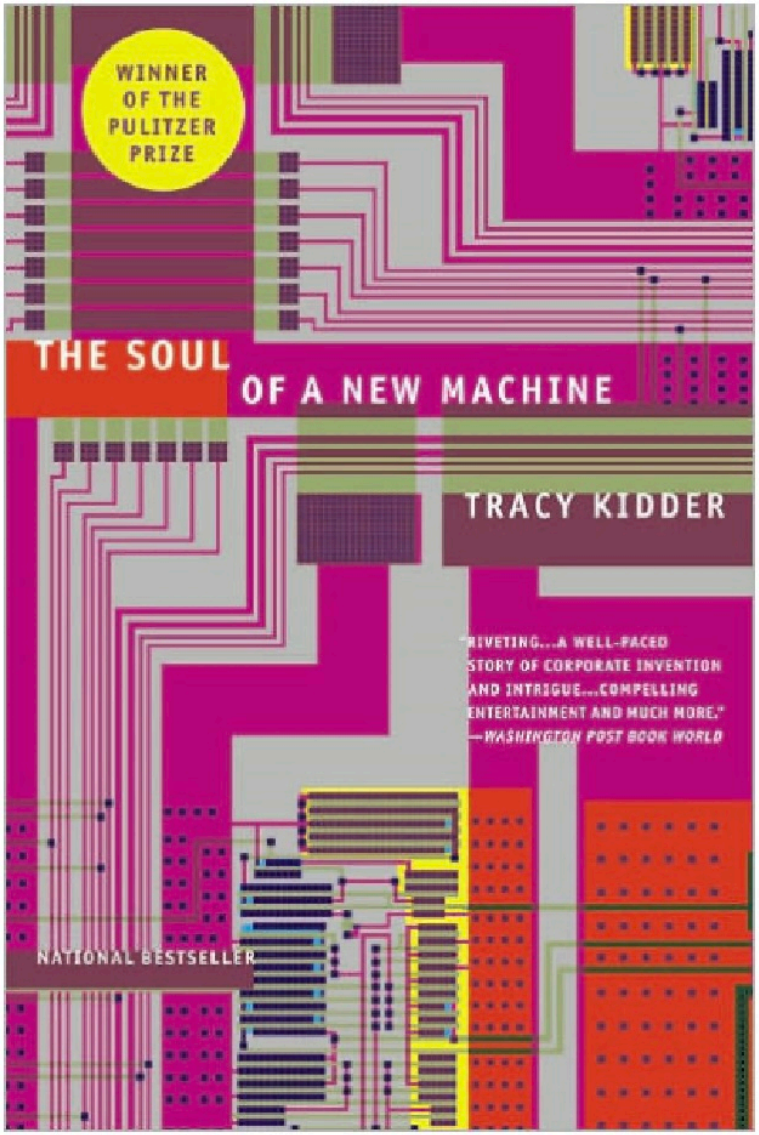
- Use Newest Version of Template with RTOS
  - Improved Partitioning with Functional & Data Privacy
- Incorporate Hardware Input Capture Function
  - Digital Input via Function Generator
  - Assess Capture Rate Improvement
- Lab Will End 30 Minutes Early

Look Ahead

- Discussion on Reading
- Review of Lab 11
- FPGA Functionality and Incorporation with RTOS

Assignment - Readings

- The Soul Of A New Machine
  - Chapters 7: La Machine
  - Send Me Discussion Topics by 10:00 AM on Thursday, November 14, 2024.



Action Items and Discussion

AI#:	Owner	Slide #	Document	Action