Project Management

Last updated 11/28/22

Project Management

- Workspace
  - Code Composer/Eclipse uses the concept of a workspace to manage projects
  - Create a single workspace
    - No Spaces in the path/names
    - Located in a place you can find it (outside of the eclipse install)
    - I use "workspace_class#_ccstudio

      e.g

      workspace_ee2931_ccstudio

Project Management

- File inclusion
  - As a developer who has spent a lot of time developing code, I might want to allow you to use the functions in my library without giving you access to source code
  - I could give you compiled code, and the linker can include the compiled code into your final executable code
    - But you cannot see the functions and how to use them
    - The compiler cannot see the function prototypes and will generate lots of errors
  - To resolve this, I break my code into 2 parts
    - Header files – visible to you
    - Source files – ultimately these are compiled and unreadable by you

Project Management

- .c files and .h files
  - .c files are used to store C code
    - Project code
    - Library code (collected functions)
  - .h files are used to store prototypes and constants
    - Function prototypes
    - Constants

Project Management

- General software development process
  - Develop code using libraries from other sources along with your code
  - The owners of the libraries want you to be able to use the functions in the library but may not want you to be able to see the implementation
    - Provide x.h files with the prototypes (declarations) of all the functions
      - Allows you to see the format and documentation of the functions
      - Allows your code to compile without the actual x.c files
    - Provide a compiled version of the code (xx.lib)
  - Your code #includes the library x.h file

Project Management

- General software development process
  - When you ‘build’ your project
    - All of the non-excluded .c files in the project get compiled
      - This is why you can only have one file with a main function
      - The included x.h files allow the compiler to know what functions are coming from elsewhere
      - Compile → assemble → machine code (10110100101010)
    - The Linker then arranges all the compiled functions from all the .c files along with any pre-compiled libraries so they can be used in your program
      - Creates a single executable file

Project Management

- Header Files
  - xxxx.h
  - Store prototypes and constants
    - Constants
      - Pin / Bit numbers and names (msp.h)
    - Structure definitions
    - Enumerated types
    - Function declarations (prototypes)

    - Wrapped in an “include guard” to prevent including the code multiple times

Project Management

- Header File - Include guard
  - Prevents the same code from being included multiple times

```
#ifndef MYFILENAME_H
#define MYFILENAME_H
…
declarations
…
#endif
```

Check to see if the constant MYFILENAME_H has not been defined – #ifndef

If it is not defined,
    create the constant - #define
    execute the commands between #define
        and #endif

If it has been defined
    skip to #endif

All caps used for the constant
Based on .h file name with dot replaced by _

Constant is not initialized or set

Project Management

- Header File - Inclusion
  - Header files are #included into the .c file using the module
  - Optionally they can be included into the related module .c file

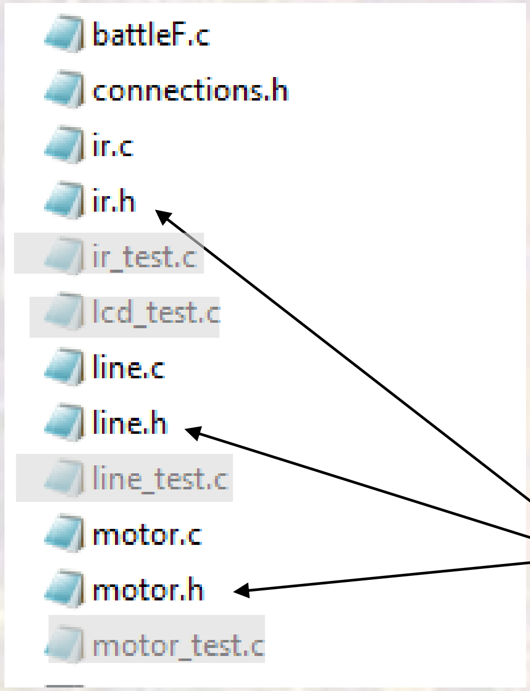
Note – c system header files are
enclosed in angled brackets < >

user defined header files are
enclosed in double quotes " "

Project Management

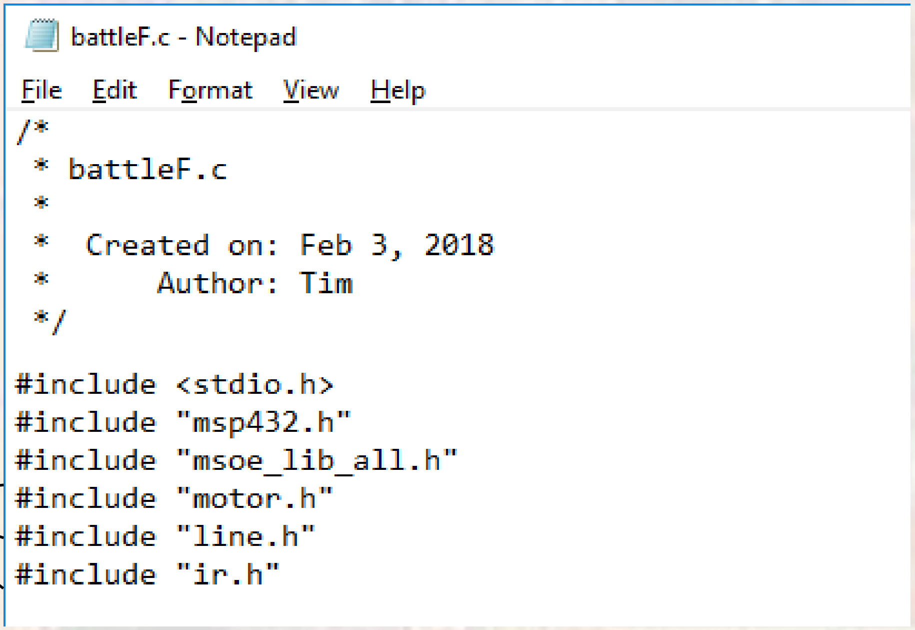
- Header File – Inclusion
  - Sumo bot example

Project Files



battleF.c
connections.h
ir.c
ir.h
ir_test.c
lcd_test.c
line.c
line.h
line_test.c
motor.c
motor.h
motor_test.c

Top level program
battleF.c

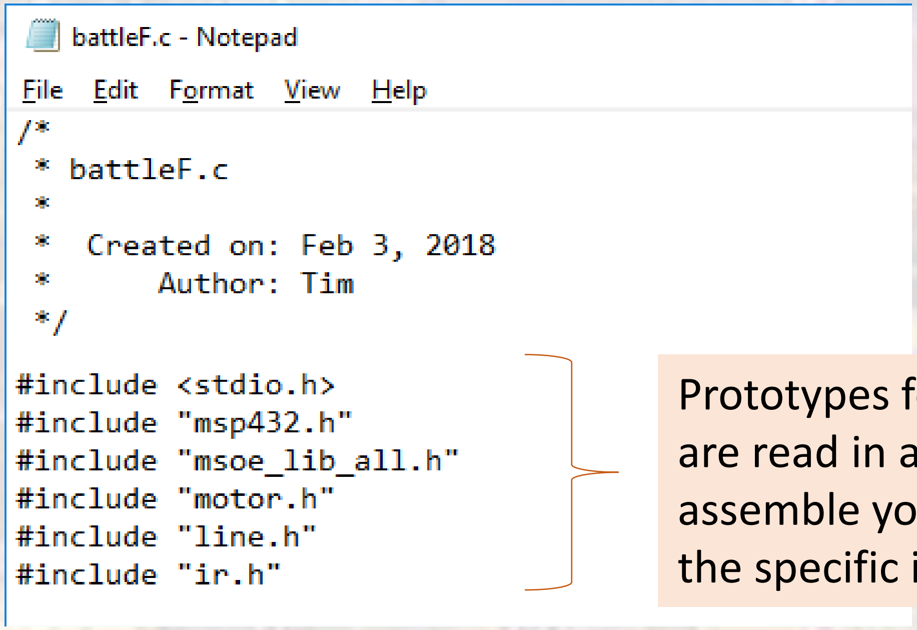


```
/*
 * battleF.c
 *
 *  Created on: Feb 3, 2018
 *      Author: Tim
 */

#include <stdio.h>
#include "msp432.h"
#include "msoe_lib_all.h"
#include "motor.h"
#include "line.h"
#include "ir.h"
```

Project Management

- Build
  - Sumo bot example – compile/assemble

Top level program
battleF.c



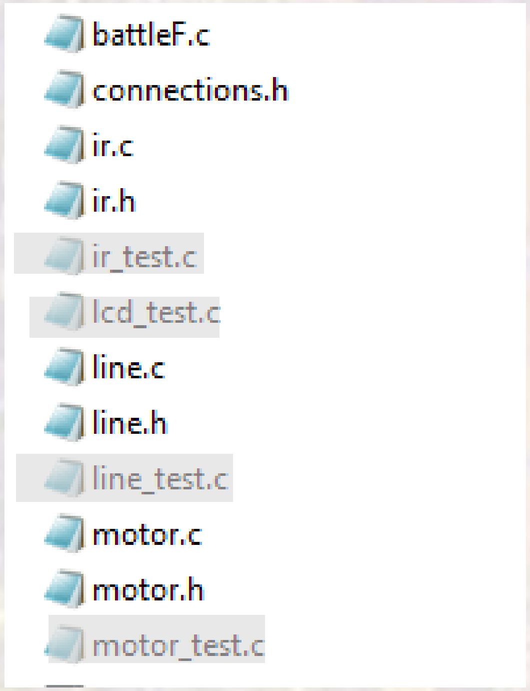
```
/*
 * battleF.c
 *
 *  Created on: Feb 3, 2018
 *      Author: Tim
 */

#include <stdio.h>
#include "msp432.h"
#include "msoe_lib_all.h"
#include "motor.h"
#include "line.h"
#include "ir.h"
```

Prototypes for all the functions in these files are read in and the compiler can compile and assemble your top-level code – without knowing the specific implementation of the functions

Project Management

- Build
  - Sumo bot example – compile/assemble

Project Files



```
battleF.c
connections.h
ir.c
ir.h
ir_test.c
lcd_test.c
line.c
line.h
line_test.c
motor.c
motor.h
motor_test.c
```

Each of the non-excluded .c files is compiled independently (including your top level)

Project Management

- Build
  - Sumo bot example - linker

The machine code from each compiled/assembled .c file is combined along with the compiled/assembled code from standard libraries and MSOE_LIB to create an executable file

Note: The MSOE_LIB includes readable .c files. These files are not the files used during build. I have already compiled/assembled the MSOE_LIB files and included them in the distribution (zip file) under the Debug directory

When you add Debug/MSOE_LIB.lib to your linker path during library installation, you point the linker to the already compiled/assembled MSOE_LIB files

Project Management

- Header File – Inclusion
  - Sumo bot example

```c
/*
 * ir.h
 *
 *  Created on: Jan 17, 2018
 *      Author: Tim
 */

#ifndef IR_H_
#define IR_H_

//////////////////////////////////////
//
// IR_setup()
//    Sets up the pins for the IR rx/tx
//    Uses 2 IR transmitters and 2 IR receivers
//
// Transmitters are IR diodes and require one pin each
// L tx - P10.5
// R tx - P10.5     -  common output
//
// Sensors require Vcc, gnd, and 1 output
// L rx - P10.2
// R rx - P10.3
//
//////////////////////////////////////
void ir_setup(void);

//////////////////////////////////////
//
// check_ir(l_ptr, r_ptr)
//    modifies the pointers based on sensor output values
//
//////////////////////////////////////
void check_ir(int * left, int * right);

#endif /* IR_H_ */
```

IR .c file

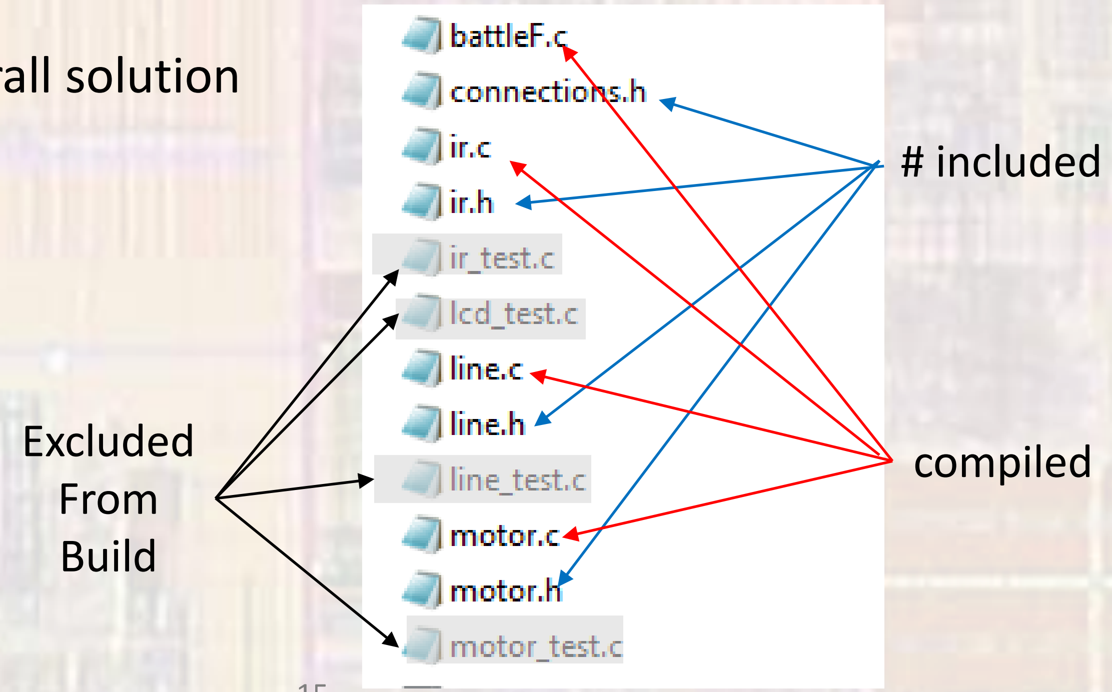
```c
/*
 * ir.c
 *
 *  Created on: Dec 28, 2017
 *      Author: johnsontimoj
 */

#include "msp432.h"
#include "msoe_lib_all.h"
#include <stdio.h>
#include "ir.h"

////////////////////////////////////
//
// IR sensor routines
//
// 1) IR_setup
//    Sets up the pins for the IR rx/tx
//    Sets up the 38KHz PWM signal - TimerA3
//    Sets up the PWM envelope signal - TimerA2
//
// 2) check_IR
```

```c
void ir_setup(void){
    //
    // setup pins
    //
    // tx outputs         P10.5
    P10->SEL0 |= 0x20;
    P10->SEL1 &= ~0x20;
    P10->DIR |= 0x20;
```

© tj

Project Management

- Project Build
  - The IDE (Code Composer /Eclipse)
    - Includes all the files “included” in the top level file (the one containing main), and all files “included” in those files
      - This gives the compiler a complete set of function/object prototypes
    - Compiles all the .c files in the project that have not been excluded from the build
    - Builds the overall solution


battleF.c
connections.h
ir.c
ir.h
ir_test.c
lcd_test.c
line.c
line.h
line_test.c
motor.c
motor.h
motor_test.c
# included
compiled
Excluded From Build