

Panel 1

Prior to Le12

# Writing Functions

ME430 Mechatronics

1

Panel 2

## Writing Functions

passing parameters

return types

prototypes

Writing a function together for `addTwoNumbers`

Writing a function by yourself for `integerExp`

2

Panel 3

### Passing parameters to a function

```
/******  
 * Additional Helper functions  
*****/  
  
/*****  
 * Function:      void sample(void)  
 * Input Variables:  
 * Output return:  
 * Overview:      You change this template of a function if needed  
*****/  
void sampleFunction(){  
    // Some function that does a specific task  
}
```

3

Panel 4

### Passing parameters to a function

```
void sampleFunction(int a){  
    a = 55;  
}  
  
void sampleFunction(char a){  
    a = 55;  
}
```

4

Panel 5

### Passing parameters to a function

**Note:** Global variables defined outside of any function don't actually need to be passed. They are visible everywhere.

```
void sampleFunction(int a) {  
    a = 55;  
}
```

```
void sampleFunction(char a) {  
    a = 55;  
}
```

### Usually you use the variable to control other things

```
void sampleFunction(char b) {  
    int y;  
    y = 5*b;  
  
    // Do stuff with y or b within function  
}
```

5

Panel 6

### Return types from a function

```
int sampleFunction() {  
    int y;  
    y = 22;  
  
    return y;  
}
```

```
char sampleFunction() {  
    char y;  
    y = 22;  
  
    return y;  
}
```

6

Panel 7

## Return types from a function

Can only return 1 variable!

```
long float sampleFunction(){  
    char y;  
    float f;  
    y = 22;  
    f = 1.25;  
  
    return y, f;  
}
```

7

Panel 8

## Example: rand() was a function that returned a value

---

### rand

---

**Function:** Generate a pseudo-random integer.

**Include:** `stdlib.h`

**Prototype:** `int rand( void );`

**Remarks:** Calls to this function return pseudo-random integer values in the range [0,32767]. To use this function effectively, you must seed the random number generator using the `srand()` function. This function will always return the same sequence of integers when identical seed values are used.

**Return Value:** A psuedo-random integer value.

**File Name:** `rand.asm`

8

Panel 9

Just cause a function has a return value, you're not required to store it

---

## printf

---

**Function:** Formatted string output to stdout.

**Include:** `stdio.h`

**Prototype:** `int printf (const rom char *fmt, ...);`

**Remarks:** The `printf` function formats output, passing the characters to `stdout` via the `putc` function. The format string is processed as described for the `fprintf` function.

**Return Value:** `printf` returns EOF if an error occurs, otherwise returns the number of characters output.

**Filename:** `printf.c`

**Code Example:**

```
#include <stdio.h>
void main (void)
{
    /* will output via stdout (_H_USART by default) */
    printf ("Hello, World!\n");
}
```

9

Panel 10

## Function prototypes

```
/** Local Function Prototypes *****/
void sampleFunction(void);
```

---

```

/*****
 * Additional Helper functions
 *****/

/*****
 * Function:      void sample(void)
 * Input Variables:
 * Output return:
 * Overview:      You change this template of a function if needed
 *****/
void sampleFunction(){
    // Some function that does a specific task
}
```

10

Panel 11

## Function Prototypes

### If a function only had input parameters

```
void sampleFunction(char);
```

```
void sampleFunction(int);
```

### If a function only had return values

```
int sampleFunction(void);
```

```
char sampleFunction(void);
```

These are just examples, make sure you can generalize.

11

Panel 12

### Example of a slightly more complex function

```
/** Local Function Prototypes *****/
float yValue(float, float, float);

/*****
 * Function:          yValue
 * Input Variables:  b = intercept of line
 *                  m = slope of line
 *                  x = x value of function y
 * Output return:    y = y value of line at location x
 * Overview:         Return the value of y based on a line
 *****/
float yValue(float m, float x, float b){
    float y;
    y = m*x + b;

    return y;
}
```

12

Panel 13

Your turn **addTwoNumbers**

Your function should be called **addTwoNumbers**

Receives two integers

Returns an integer

The return value is simply the sum of the two inputs

**Make sure to add the function prototype as well!!!**

13

Panel 14

```
/** Local Function Prototypes *****/  
  
/*****  
* Function:      addTwo Numbers  
* Input Variables: a = first integer of sum  
*                b = second integer of sum  
* Output return: y = Result of a + b  
* Overview:     Returns the sum  
*****/  
  
      
      
      
}
```

14

Panel 15

Now that we've got a function how do we use it?

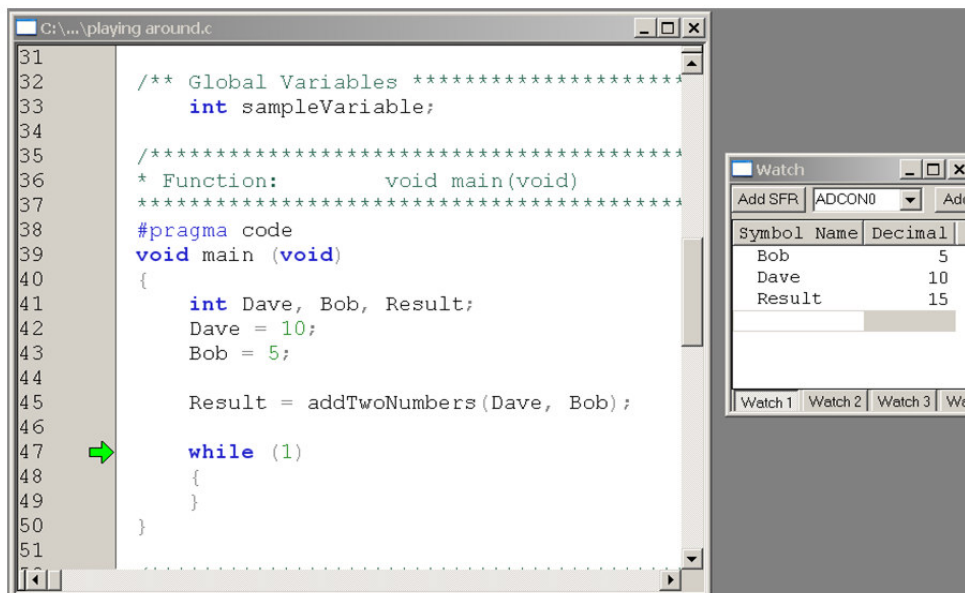
```
void main (void)
{
    int Dave, Bob, Result;
    Dave = 10;
    Bob = 5;

    Result = addTwoNumbers (Dave, Bob);
}
```

15

Panel 16

Go play in MPLAB



The screenshot shows the MPLAB IDE interface. The main window displays a C program with the following code:

```
31
32  /** Global Variables ****
33  int sampleVariable;
34
35  /**
36  * Function: void main(void)
37  ****
38  #pragma code
39  void main (void)
40  {
41  int Dave, Bob, Result;
42  Dave = 10;
43  Bob = 5;
44
45  Result = addTwoNumbers (Dave, Bob);
46
47  while (1)
48  {
49  }
50
51
```

A green arrow points to line 47. To the right, the Watch window is open, showing the following table:

Symbol Name	Decimal
Bob	5
Dave	10
Result	15

16

Panel 17

**Simplified syntax**

```
void main (void)
{
    int Result;
    Result = addTwoNumbers (2, 3);
}

int addTwoNumbers (int a, int b) {
    return a+b;
}
```

17

Panel 18

**Your turn integerExponent****Your function should be called integerExponent**

Receives two inputs

- int base = The number for the base in  $base^x$
- char exp = The exponent in  $base^{exp}$

Returns a long

- long = The result of  $base^{exp}$

The return value is simply the expected result if integers were used in exponentiation. Note that ^ for integers becomes exclusive OR. So we made our own function!

**Hint:**

- Make sure to add the function prototype
- Make sure it works for the 0<sup>th</sup> power  $a^0 = 1$
- Try using a well constructed for loop!

18

Panel 19

This question will be #1 on the quiz

Use the function to load an array with 14 elements  
The elements should be  $\text{exponents}[i] = 5^i$

```

37 #pragma code
38 void main (void)
39 {
40     int Dave, Bob, Result, x;
41     char i;
42     long arrayFives[14];
43     Dave = 10;
44     Bob = 5;
45     x = 5;
46
47     Result = addTwoNumbers(Dave, Bob);
48
49     for(i=0;i<14;i++)
50     {
51         arrayFives[i] = integerExp(x,i);
52     }
53
54     while (1)
55     {
56         // This area loops forever
57     }
58 }
59

```

Symbol Name	Decimal
Bob	5
Dave	10
Result	15
arrayFives	
[0]	1
[1]	5
[2]	25
[3]	125
[4]	625
[5]	3125
[6]	15625
[7]	78125
[8]	390625
[9]	1953125
[10]	9765625
[11]	48828125
[12]	244140625
[13]	1220703125

Loaded C:\Rose\ME430 - Mechatronics\Winter 0708\Week 4\Writing Functions\writing functions.cof.  
BUILD SUCCEEDED: Mon Dec 17 23:25:28 2007

19

Panel 20

## Multiple .c and .h files

You can have as many .c and .h files as you like within a project.  
However you can only have 1 function named **main**.

All the .c and .h files should be in the same folder with the project.

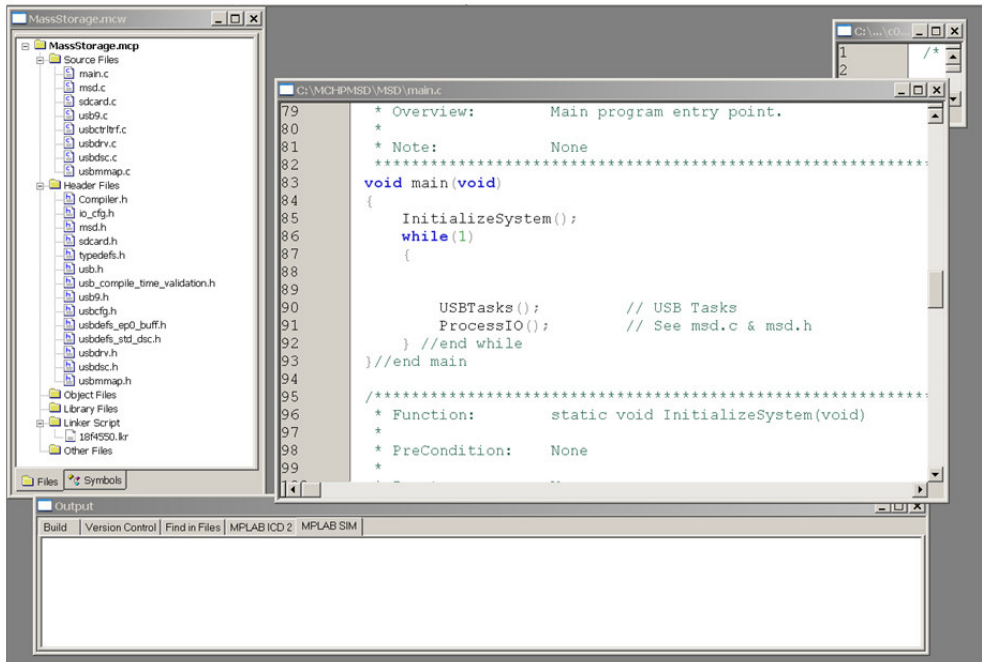
Some unfamiliar syntax, but easy to use a template

In this class you're not required to use multiple files but I wanted to make you aware that you could and big project almost ALWAYS use different files for different tasks.

20

Panel 21

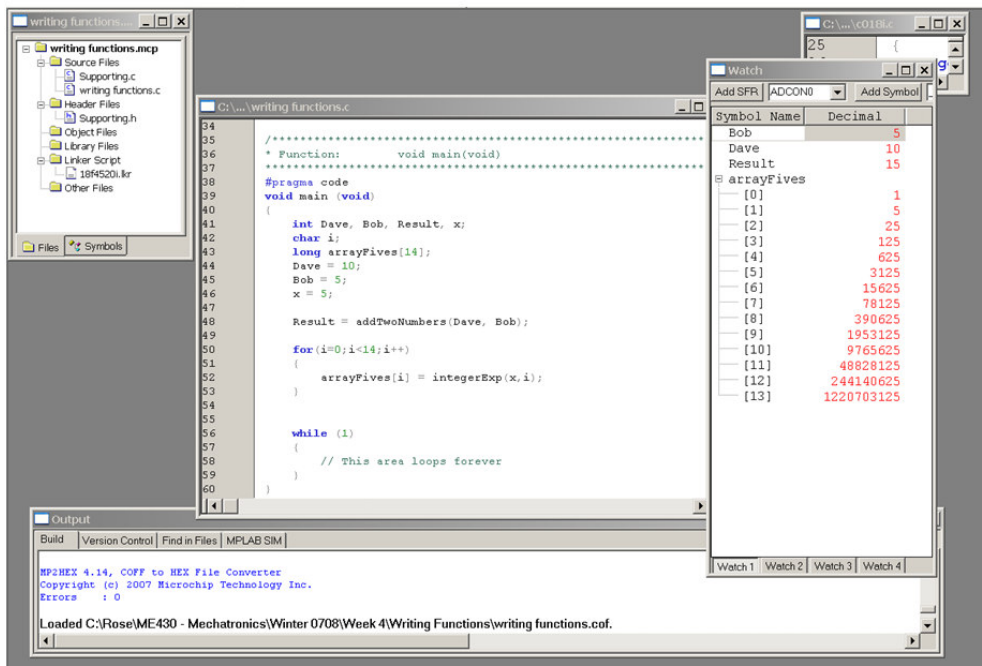
Sample image from a BIGGER project (lots of files)



21

Panel 22

The files get added, the functions and prototypes move but Main looks the same



22

Panel 23

## Supporting header file

```

/*****
 * FileName:      Supporting.h
 * Processor:    18F4520
 * Compiler:     MCC18
 *
 * Author        Date        Comment
 *-----
 * David Fisher  9/25/07
 *****/
#ifndef __SUPPORTING_H__
#define __SUPPORTING_H__

/*****
      External Definitions
 *****/

// Set defines here that are used in both supporting and main
#define EXAMPLE 100

/*****
 * Function:     integerExp
 * Input Variables:  base and exp where output=base^exp
 * Output return:  Resulting output from exponentiation operation
 * Overview:     Performs exponentiation for integers
 *****/
extern long integerExp(int, char);

/*****
 * Function:     exampleFunction
 * Input Variables:
 * Output return:
 * Overview:
 *****/
extern void exampleFunction(void);

#endif

```

23

Panel 24

## Supporting source file

```

/*****
 * FileName:      Supporting.C
 * Processor:    18F4520
 * Compiler:     MCC18
 *
 * Author        Date        Comment
 *-----
 * David Fisher  9/25/07
 *****/
#include <p18f4520.h>
#include "Supporting.h"

/** Variables visible within this .c file only *****/
unsigned char privateVariable;

/*****
      Function Prototypes
 *****/
long integerExp(int, char);
void exampleFunction(void);

/*****
 * Function:     exampleFunction
 * Input Variables:
 * Output return:
 * Overview:
 *****/
void exampleFunction()
{
    // Code for function goes here
}

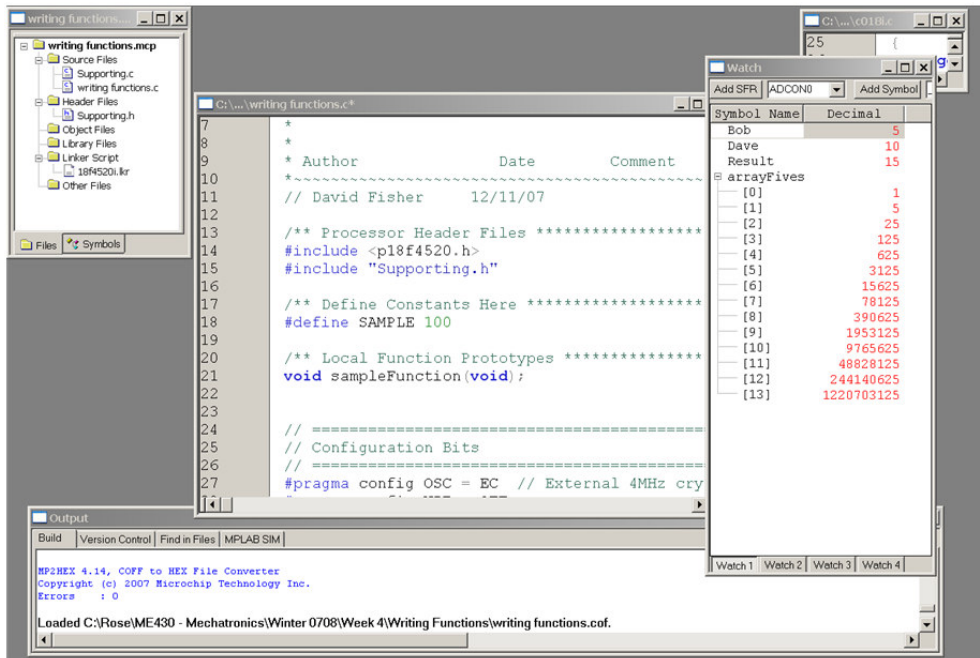
/*****
 * Function:     integerExp
 * Input Variables:  base and exp where output=base^exp
 * Output return:  Resulting output from exponentiation operation
 * Overview:     Performs exponentiation for integers
 *****/
long integerExp(int base, char exp)
{
    // Your code goes here for this function
}

```

24

Panel 25

Add the #include file using " " instead of < > since it's in the folder not the path



25

Panel 26

### Modular Programming using a Supporting file:

- Download the blank template Supporting.c and Supporting.h files off the website Calendar under today's date
- Cut your integerExp from file with main in it and put it into Supporting.c with a prototype in Supporting.h
- Add the #include "Supporting.h"
- Add the extern prototype in Supporting.h (with comments)

### If you want make a few other functions in Supporting

- Sending an int and getting back the ASCII character for each digit:
  - char charHundredsDigit(int);
  - char charTensDigit(int);
  - char charOnesDigit(int);

26