Compiler, Arrays and Pointers

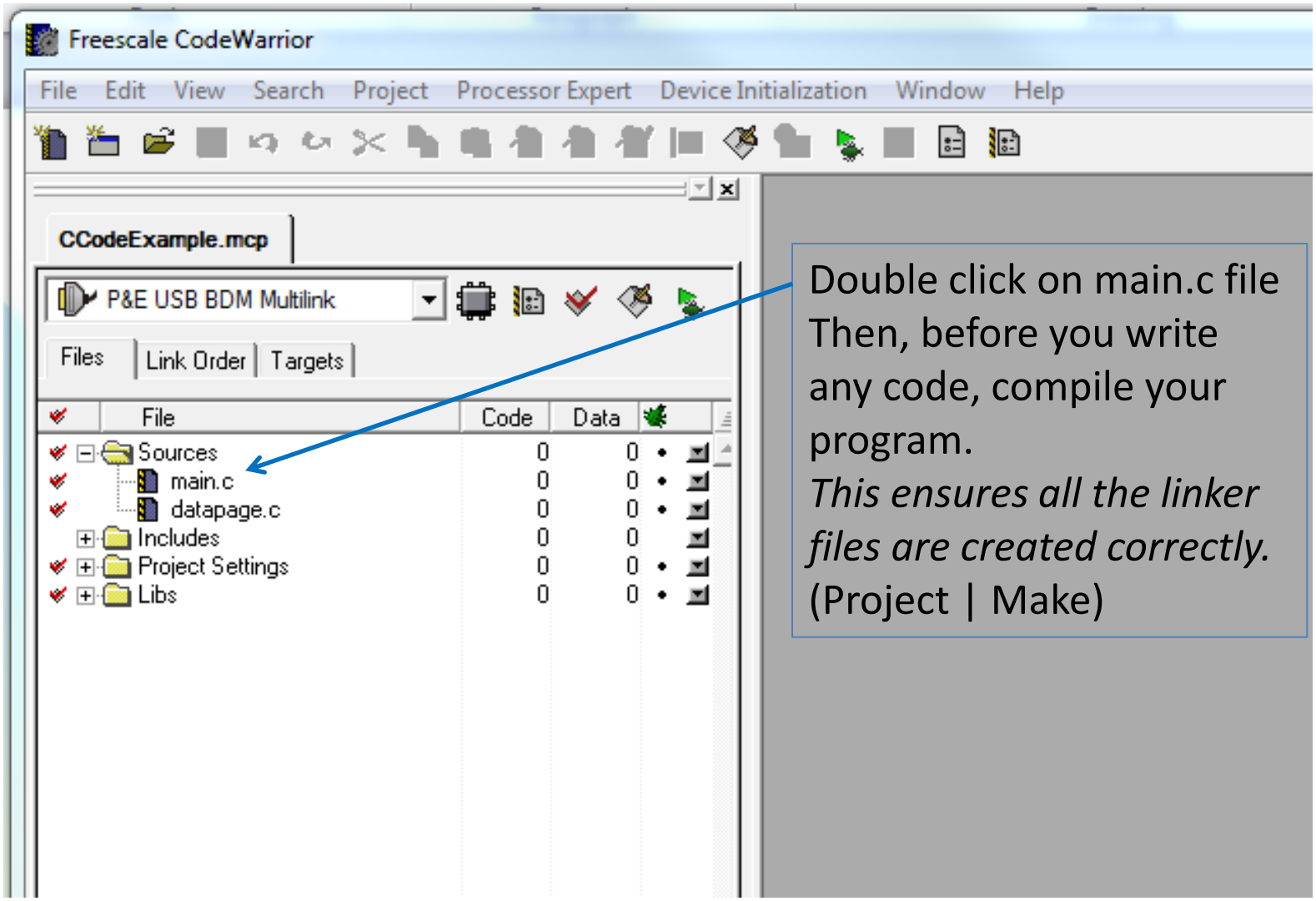
ECET 20900

IUPUI

Outline

- CodeWarrior IDE & Program Organization
- More C Topics
  - Arrays
  - Pointers
- Practice Writing C Functions

C and Code Warrior IDE



Double click on main.c file
Then, before you write any code, compile your program.
This ensures all the linker files are created correctly.
(Project | Make)

C and Code Warrior IDE

Files | Link Order | Targets

File	Code	Data
Sources	0	0
main.c	0	0
datapage.c	0	0
Includes	0	0
derivative.h	0	0
MC9S12XEP100.h	0	0
Project Settings	0	0
Libs	7881	2019

main.c

Path: E:\HW06\Sources\main.c

```c
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */




void main(void) {
  /* put your own code here */


   EnableInterrupts;


  for(;;) {
    _FEED_COP(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}
```

Line 1 Col 1

C and Code Warrior IDE

main.c

Path: E:\HW06\Sources\main.c

```c
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */



void main(void) {
  /* put your own code here */


   EnableInterrupts;


  for(;;) {
    _FEED_COP(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}
```

Line 1 Col 1

Put your function prototypes here

Put all function after main function

Program organization:

- Uninitialized and initialized variables first
- Executable code after all variable declarations.

C and Code Warrior IDE

Variable declarations mixed in with executable code will produce a syntax error.

```
void main(void) {
  /* put your own code here */

  int x;
  x = 0xc5;
  int answer;
  answer = x + 4;

  EnableInterrupts;


  for(;;) {



    _FEED_COP(); /* feeds the d
  } /* loop forever */
  /* please make sure that you
}
```

Errors & Warnings

2 | 0 | 0 | Errors and wa

Error : C2801: '}' missing

main.c line 16

Error : Compile failed

Path: I:\BkUP\209 Fall 201

```
  int answer;
  answer = x + 4;

  EnableInterrupts;


  for(;;) {
```

The correct way to declare variables in CodeWarrior IDE: Variable declaration first Then executable code.

```
void main(void) {
  /* put your own code here */

  int x;
  int answer;
  x = 0xc5;
  answer = x + 4;

  EnableInterrupts;


  for(;;) {
```

```
void main(void) {
  /* put your own code here */

  int x = 0xc5;
  int answer;
  answer = x + 4;

  EnableInterrupts;


  for(;;) {
```

This may cause error.

C and Code Warrior IDE: proper code formatting

```c
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */

//Put your commented header here

//For now: Put your function prototypes here


//uninitialized Global variables go next

//initialized Global variables go next

void main(void)
{
  // declare uninitialized local variables
  // declare and initialize local variables next
  /* call functions and write statements that require
     a one-time process: like initialize data, ports….
  */
  EnableInterrupts;   /* you may delete this line. */


  for(;;)
  {

  /* call functions and write statements that require
     a continuous process: like checking a dip switch bit-state
  */


    _FEED_COP(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}
```

/** For now, write your function declarations outside of the void main(void){ }, but still inside the main.c file

later we will write our function declarations in another file, not in the main.c file. **/

C and Code Warrior IDE: proper code formatting

```
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */

//Put your commented header here

//For now: Put your function prototypes here
void MyNewFunction(unsigned char);

void main(void)
{
  // declare uninitialized local variables
  unsigned char val;

  for(;;)
  {

   /* call functions and write statements that require
      a continuous process: like checking a dip switch bit-state
      val = 0x05;
      MyNewFunction(unsigned char val);
   */

      _FEED_COP(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}

/***** For now write your function declarations outside of the void main(void){ } ******/
void MyNewFunction(unsigned char someVar)
{
   someVar++:
}
```

Function prototype

Function receives one unsigned char passed to it and returns nothing

Arrays

- CodeWarrior IDE & Program Organization
- More C Topics
  - Arrays
  - Pointers
- Practice Writing C Functions

Arrays

- All high-level computer languages have arrays
  - For convenient access of related data items like:
    - character strings, bitmapped images, or memory blocks

```
//declare array of 10 integers, indexed 0...9
unsigned int stuff[10];
```

- The name "stuff" is actually a memory pointer to the start of the array.
  - Remember *list* in assembly language? ☺
    - This is the same as an assembly list, but now we can data type the memory to hold specific value types

Arrays

Example: Write list of data values to an I/O Port

The size declarator is not necessary when a list is explicitly assigned

```
void main(void)
{
    unsigned char index;
    unsigned char dataArray[5]={0x10, 0x20, 0x30, 0x40, 0x50};
    DDRC = 0xff;

    for(index=0; index<5; ++index)
    {
        PORTC = dataArray[index];
        Delay( );    //call to Delay function
    }
}
```

* Of course, to see the LEDs light, a delay algorithm must be written

Character Arrays

- Note that C compiler automatically terminates any character string with a null char (has a value of zero)
  - In the example below, memory starting at location charArray would have 41h,42h,43h,44h,00h

```
// volatile unsigned char PortC@0x04;
void main()
{
    //send array contents to PORTC
    unsigned char charArray[] = {"ABCD"};
    unsigned char index=0;
    DDRC = 0xff;

    while(charArray[index]!=0)
    {
        PORTC = charArray[index];
        ++index;
    }
}
```

We will not have to write this statement: This declaration and assignment is done for us in the included .h files provided in a CodeWarrior program..

C array of characters to create a string

C language does not have a string data type…

- So, you must do one of the following.

```
char *p = “This is a string";
char p2[ ] = “This is a string";
char p3[17] = “This is a string";  //why 17? ..Not a good practice
```

Pointers

- CodeWarrior IDE & Program Organization
- More C Topics
  - Arrays
  - Pointers
- Practice Writing C Functions

C Pointers

- Pointer is a variable that holds the address of a variable
- It first must be declared and then it can be used to point.

```
char*  ptrc;   // ptrc points to a character
char  *ptrc;   // this says the same thing as previous instruction
//now ptrc can be used any place a character could
```

- Example:

```
int a = 3, *b, c;     // integer a is assigned a value of 3
                      // b is an integer pointer,
                      // c is an integer
b = &a;               // b is assigned the address of a
c = *b;               // c gets a copy of value stored at a
```

- In the MC9S12, pointers are 2 bytes long
  - Can you describe why?

C Pointers & Arrays

- Pointers allow
  - Easy access to array elements
  - Access to variables and arrays from within a function

```
unsigned char *y, x[100];
y = x;      //since x points to 1st element in the array: x[0],
            //now y points to 1st element of array: y -> x[0]
y = &x[0]; //another way of saying the same thing
y++;       //now y points to 2nd element of the array: y -> x[1]
y +=98; //now y points to last (99th) element: y -> x[99]
```

- Compiler knows how to increment a pointer
  - depending on the size of the datatype:
    - 1 for a char, 2 for int, 4 for long or float, 8 for double
  - And NOW you know why the MC9S12 has an auto increment feature for index addressing: ldd 2,x+ ;(assembly)

Passing Arrays to Functions

```
/*function prototype (char* or char x[ ] synonymous)*/
void ReadArray(char*);

void main()
{
    char charStr[ ] = {"ABCD0123"};
    ReadArray(charStr);
}

void ReadArray(char theStr[ ])
{
    char value;
    value = theStr[2];
     /* value is now 'C'  */
}
```

```
//this would be the same function header
void ReadArray(char *theStr)
{
    char value;
    value = theStr[2];
}
```

Returning a character "String"

```
char* ReturnsArray();   //function prototype

void main()
{
  char* ptrData;     //can point to any char data type
  ptrData = ReturnsArray(); //function call
  LCDDisplay(ptrData);
  LCDDisplay("XYZ");
}

char* ReturnsArray()
{
  static char charArray[] = {"ABCD0123"};
  return(charArray);
}
```

Remember
static:
this variable retains its value between calls to a function
Has scope inside a function
Gets initialized to zero if not explicitly initialized;
otherwise, gets the explicitly assigned value only once

Practice Writing C Functions

- CodeWarrior IDE & Program Organization
- More C Topics
  - Arrays
  - Pointers
- Practice Writing C Functions

Exercise 1

Write a C language subroutine named CoutBit6 that is passed a character array and returns an integer.

The character array passed is 4 characters long initialized with the values {0x08,0x04,0xc0,0x20}.

The subroutine should count the number of characters in the array that have bit 6 set(one) and return the count as an integer.

The routine should only check bit 6, other bits in the character array can be any value.

Exercise 1: Answer

```
int CountBit6(char *passedArray)
{
  int index;
  int bit6Counter = 0;
  for(index=0; index<4; index++)
  {
    if(passedArray[index] & 0x40)
     bit6Counter += 1;
  }
  return bit6Counter;
}
```

Exercise 1: Answer continued

```
//This is how the entire program might look
int CountBit6(char*);  //function prototype

void main(void)
{
  int returnedCount = 0;
  char arrayVals[]= {0x08,0x04,0xC0,0x20};
  returnedCount = CountBit6(arrayVals);
}
int CountBit6(char *passedArray)
{
  int index;
  int bit6Counter = 0;
  for(index=0; index<4; index++)
  {
    if(passedArray[index] & 0x40)
     bit6Counter += 1;
  }
  return bit6Counter;
}
```

Exercise 2

- Write a simple C program to convert degrees from Celsius to Fahrenheit. Use unsigned chars only.

Exercise 2: Answer (one of many possible)

```
#define FREEZING    32
#define MAXC        124
#define MAXF        255
void main()
{
  unsigned char cd, fd; //celsius, fahr degrees
  cd = 20;              //arbitrary example value

  if (cd > MAXC) //then fd won't fit in a byte
  {
       fd = MAXF; //clamp it at the max
  }
  else
  {
       fd = 9*cd / 5 + FREEZING;
  }
}
```

Arguments to a Function

```
void TheSubroutine(int, char);    //function prototype

void main(void )
{
      int x = 5;
      char y = 2;
      TheSubroutine(x, y);              //function call w/arguments passed
  }

  //function declared w/two parameters
  void TheSubroutine(int firstArgument, char secondArgument)
  {
      firstArgument = firstArgument + (int)secondArgument);
  }
```

Cast operator

Function Call

Freescale CodeWarrior

File Edit View Search Project Processor Expert Device Initialization Window Help

CCodeExample.mcp

Full Chip Simulation

Files | Link Order | Targets

File	Code	Data
Sources	0	0
main.c	0	0
datapage.c	0	0
Includes	0	0
derivative.h	0	0
MC9S12XEP100.h	0	0
Project Settings	0	0
Startup Code	0	0
Linker Files	0	0
Libs	7881	2019

main.c

Path: F:\2014 Fall\2014 ECET 209_Fall\Labs\Lab 8\Lab 8\CCodeExample\Sources\main.c

```c
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */

//Put your function prototypes here
void myNewFunction(unsigned char);


void main(void) {
  /* put your own code here */
   unsigned char val;
   DDRC = 0xff;


   EnableInterrupts;

  for(;;) {
    val = 0x05;
    myNewFunction(val);

    _FEED_COP(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}

void myNewFunction(unsigned char secondVal)
{
  secondVal = secondVal +2;

}
```

Summary

- CodeWarrior IDE & Program Organization
- More C Topics
  - Arrays
  - Pointers
- Practice Writing C Functions