

Panel 1

Prior to Le09

Introduction to C Programming

ME430 Mechatronics

1

Panel 2

Introduction to C programming

Talking about the template

- Comments
- #include
- #pragma Configuration bits
- #define
- Function prototypes
- Global Variables
- Main
- Additional Functions
- (interrupts) - later in course

Constants vs. Variables and their locations

Control structures

- if statements (in their various forms)
- switch statements
- for loops
- while loops

2

Panel 3

Basic template.c file

Comments

// Single line comments

/*
Comment out multiple lines
*/

```

.....
* FileName: (change filename of template).c
* Processor: PIC18F420
* Compiler: MPLAB C18 v.3.04
*
* This file does the following...
*
*
* Creation and Revision:
* Author      Date      Comments
* (Your name here)
.....
/** Header Files .....
#include <pic18f420.h>
.....
/** Configuration Bits .....
#pragma config OSC = EC // EC = External 4MHz Crystal for PICDEM board only
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOREN = OFF
.....
/** Define Constants Here .....
#define SAMPLE 100
.....
/** Local Function Prototypes .....
void sampleFunction(void);
.....
/** Global Variables .....
int sampleVariable = 0;
.....
* Function: void main(void)
.....
#pragma code
void main(void)
{
    // This area happens once
    // Good for initializing and things that need to happen once
    while (1)
    {
        // This area loops forever
    }
}
.....
* Additional Helper Functions
.....
* Function: void sample(void)
* Input Variables: none
* Output Returns: none
* Overview: Use a comment block like this before functions
.....
void sampleFunction()
{
    // Some function that does a specific task
}
.....
    
```

3

Panel 4

#include section for PIC variables, library header files, and other headers

```

/** Header Files .....
#include <pic18f420.h>
.....
Configuration Bits setup the mode of the PIC
- You will never change WDT, LVP, BOREN
- Later you will change OSC = EC to OSC = INTIO67

/** Configuration Bits .....
#pragma config OSC = EC // EC = External 4MHz Crystal for PICDEM board only
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOREN = OFF
    
```

4

Panel 5

```

#defines      String replacement for constants

/** Define Constants Here *****/
#define SAMPLE 100

```

5

Panel 6

Function prototypes

```

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

/*****
 * Additional Helper Functions
 *****/

/*****
 * Function:      void sample(void)
 * Input Variables: none
 * Output Return: none
 * Overview:      Use a comment block like this before functions
 *****/
void sampleFunction()
{
    // Some function that does a specific task
}

```

6

Panel 7

Global Variables

```

/** Global Variables *****/
int sampleVariable;

```

7

Panel 8

Main Function

```

/*****
 * Function:      void main(void)
 *****/
#pragma code
void main (void)
{
    // This area happens once
    // Good for initializing and things that need to happen once

    while (1)
    {
        // This area loops forever
    }
}

```

8

Panel 9

Include the Standard library file - stdlib.h

if the sample functon returned a char rewrite the function prototype here

9

Panel 10

Global vs Local variables

```

.....
* FileName: (change filename of template).c
* Processor: PIC18F4120
* Compiler: KEIL C51 v. 5.06
* This file does the following ...
*
*
* Creation and Revisions:
* Author:
* (Your name here) Date Comments
.....
/** Header File .....
#include "SAMPLE.h"
.....
/** Configuration Bits .....
#pragma config OSC = EC // EC = External 4MHz Crystal for PIC18M board only
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOREN = OFF
.....
/** Define Constants Here .....
#define SAMPLE 100
.....
/** Local Function Prototypes .....
void sampleFunction(void);
.....
/** Global Variables .....
int sampleVariable = 0;
.....
* Function: void main(void)
* Input Variables: none
* Output Return: none
* Overview: Use a comment block like this before functions
.....
void main(void)
{
    // This area happens once
    // Good for initializing and things that need to happen once
    while (1)
    {
        // This area loops forever
    }
}
.....
* Additional Helper Functions
.....
/** .....
* Function: void sample(void)
* Input Variables: none
* Output Return: none
* Overview: Use a comment block like this before functions
.....
void sampleFunction()
{
    // Do a function that does a specific task
}
.....
    
```

10

Panel 11

Just declaring variables

```

// Just declaring variables
char x;
unsigned char y;
int Dave, Steve, a, b, c, d, e, f, g, h;
long Bob;
float Steve;
char arrayBob[10];
    
```

Initializing the variables when declaring them

```

// Initialized data
int x = 0;
float f = 1.253;
char message1[] = "Handy string syntax"
int numlist[] = {1, 1, 2, 3, 5, 8, 13, 21, 34};
    
```

Defining variables sepearte from their declarations

```


// Defining data that is already declared
x = 3;
y = 34;
arrayBob[0] = 10;
arrayBob[1] = 9;
x = rand();
    
```

11


Panel 12

Variable scope

Global Variables



Local Variables



Let's go practice!

12

Panel 13

Draw arrows to where that goes

Declaring a local variable within Main function
int y;

Changing the value of a variable within sampleFunction

Creating global variable
char x = 10;

Changing the value of sampleVariable within the main function

```

*****
* FileName: (change filename of template).c
* Processor: PIC18F4320
* Compiler: MPLAB C18 v.3.04
* This file does the following...
*
*
* Creation and Revision:
* Author      Date      Comments
* (Your name here)
*****
** Header Files *****/
#include "pic18f4320.h"
** Configuration Bits *****/
#define CONFIG_8051 // 8051
#define CONFIG_8052 // 8052
#define CONFIG_8053 // 8053
#define CONFIG_8054 // 8054
#define CONFIG_8055 // 8055
#define CONFIG_8056 // 8056
#define CONFIG_8057 // 8057
#define CONFIG_8058 // 8058
#define CONFIG_8059 // 8059
#define CONFIG_8060 // 8060
#define CONFIG_8061 // 8061
#define CONFIG_8062 // 8062
#define CONFIG_8063 // 8063
#define CONFIG_8064 // 8064
#define CONFIG_8065 // 8065
#define CONFIG_8066 // 8066
#define CONFIG_8067 // 8067
#define CONFIG_8068 // 8068
#define CONFIG_8069 // 8069
#define CONFIG_8070 // 8070
#define CONFIG_8071 // 8071
#define CONFIG_8072 // 8072
#define CONFIG_8073 // 8073
#define CONFIG_8074 // 8074
#define CONFIG_8075 // 8075
#define CONFIG_8076 // 8076
#define CONFIG_8077 // 8077
#define CONFIG_8078 // 8078
#define CONFIG_8079 // 8079
#define CONFIG_8080 // 8080
#define CONFIG_8081 // 8081
#define CONFIG_8082 // 8082
#define CONFIG_8083 // 8083
#define CONFIG_8084 // 8084
#define CONFIG_8085 // 8085
#define CONFIG_8086 // 8086
#define CONFIG_8087 // 8087
#define CONFIG_8088 // 8088
#define CONFIG_8089 // 8089
#define CONFIG_8090 // 8090
#define CONFIG_8091 // 8091
#define CONFIG_8092 // 8092
#define CONFIG_8093 // 8093
#define CONFIG_8094 // 8094
#define CONFIG_8095 // 8095
#define CONFIG_8096 // 8096
#define CONFIG_8097 // 8097
#define CONFIG_8098 // 8098
#define CONFIG_8099 // 8099
#define CONFIG_8100 // 8100
** Define Constants Here *****/
#define SAMPLE 100
** Local Function Prototypes *****/
void sampleFunction(void);
** Global Variables *****/
int sampleVariable = 0;
*****
* Function: void main(void)
* Description: This is the main function
* Parameters: None
* Return: None
* Comments: This is the main function
*****
void main(void)
{
    // This area happens once
    // Good for initializing and things that need to happen once
    while (1)
    {
        // This area loops forever
    }
}
*****
* Additional Helper Functions
*****
** Function: void sample(void)
* Description: This is the sample function
* Parameters: None
* Return: None
* Comments: This is the sample function
*****
void sampleFunction()
{
    // Some function that does a specific task
}

```

13

Panel 14

if statements

- Basic if statement
- If statement with multile lines
- Multiple conditions for an if
(The difference between & vs. &&)
- If - else
- If - else if - else if - else if
- The switch statement

14

Panel 15

Basic if statements

```

// Basic if statement
if (Bob == 10)
    Dave = 5;

// BAD if statement
if (Bob = 10)
    Dave = 5;

// BAD if statement
if(Bob == 10);
    Dave = 5;

// Multiple lines if statement
if (Bob == 11)
{
    Dave = 5;
    PORTB = 0x03;
}

// BAD multiple lines if statement
if (Bob == 11)
    Dave = 5;
    PORTB = 0x03;

```

15

Panel 16

Multiple conditions

```

// Multiple conditions
if ( (Bob == 11) && (x < 3) )
{
    Dave = 5;
    PORTB = 0x03;
}

// BAD statement
if ( 2 < x < 6 )
{
    Dave = 5;
    PORTB = 0x03;
}

// Conditional Statements == != < > >=
// Logical Operators && ||

```

16

Panel 17

| | Logical Operators (result in a 0 or 1) | Bitwise Operators (maintain bit size) | Value after bitwise operation |
|--------|--|--|-------------------------------------|
| Equals | <code>==</code> <code>if(x == 5)</code> | <code>=</code> <code>char x = 5;</code> | [Redacted] |
| AND | <code>&&</code> <code>if(x == 5 && y == 3)</code> | <code>&</code> <code>x = 0b00000101 & 0b00000011;</code> | [Redacted] |
| OR | <code> </code> <code>if(x == 5 y == 3)</code> | <code> </code> <code>x = 0b00000101 0b00000011;</code> | [Redacted] |
| NOT | <code>!</code> <code>if(x != 5)</code> | <code>~</code> <code>y = ~x;</code> <small>(assume x equals value from prior question)</small> | [Redacted] |

17

Panel 18

| if - else statement | if - else if - else if statement |
|---|---|
| <pre>// if - else statement if (Bob == 12) { Dave = 5; PORTB = 0x03; } else { Dave = 3; PORTB = 0x01; }</pre> | <pre>// if - else if - else if - else statement if (Bob == 12) { Dave = 5; PORTB = 0x03; } else if (Bob == 13) { Dave = 4; PORTB = 0x02; } else { Dave = 3; PORTB = 0x01; }</pre> |

18

Panel 19

Write an if else statement the sets PORTB to 3 if x is less than or equal to 10 and greater than 0, PORTB to 2 if x is less than or equal to 20 but greater than 10, PORTB to 1 if x is greater than 20, and PORTB to 0 if none of the above are true

19

Panel 20

Switch statements

```
// Switch statement
switch (Bob)
{
    case 3:
        Dave = 3;
        break;
    case 4:
        Dave = 4;
        break;
    case 5:
        Dave = 50;
        break;
    default:
        Dave = 0;
        break;
}
```

20

Panel 21

for loops

Standard for loop syntax

```
// Three sections of the FOR loop
for( i=0 ; i<10 ; i++)
{
    arrayB[i] = i * 10;
    x = x + i;
}
```

21

Panel 22

while loops

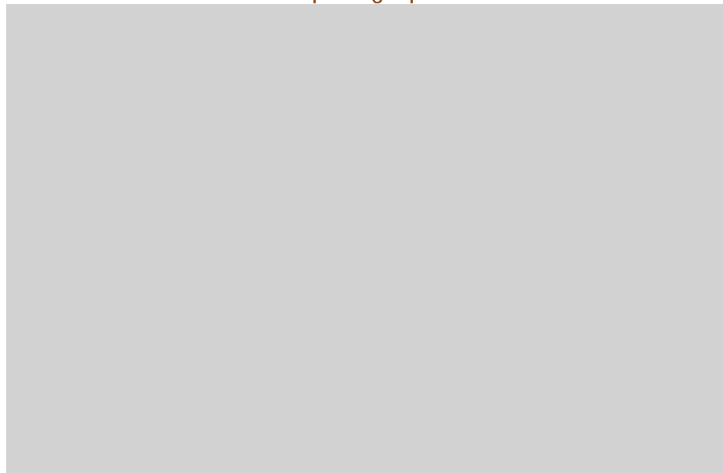
```
// A while loop that runs forever
while (1)
{
    // This area loops forever
}
```

```
// A while loop implementing a for loop
Bob = 20;
while ( Bob < 35)
{
    printf("Bob is %d years old",Bob);
    Bob++;
}
```

22

Panel 23

Write a loop that makes x count down from 12 to 0.
Print the value of x within the loop using a printf statement.



23