EE 458 Introduction to C and Code Composer Studio

Ishaan Bhimani

EE458 Winter 2022

Agenda:

- Intro to C for EE458 (not comprehensive)
  - Syntax
  - Variables and defines
  - Types (ints, floats)
  - Header files
  - Functions
  - Variable scoping
  - Loops/conditionals
- Intro to CCS
  - Standard CCS Code walkthrough
    - Register access
    - Interrupts
    - Watch Window
    - Debug

Syntax:

- Very similar to Java
- Assume students have used Python. Differences below:
  - Semicolons required
  - Type required when declaring/initializing
  - Comments denoted with “//” or “/* ….. */”
  - Whitespace does not matter, brackets do

Variables

- Variable must be declared before definition. Can be done simultaneously with initialization.
- Example:

```
int j; // declare (value of j is garbage)
j = 1; // define (change value to 1)

int k = 2; // initialize k to value of 1
```

Types

- In EE458 we will use mostly integers and floats.
- Integers (ints) can be unsigned or signed, and do not have decimal (will round down)
- Ints have different sizes (in bits). Make sure we know the correct size by using standard sizes:
  - Example: uint16_t is a 16 bit (2 byte) unsigned integer
- Important to understand sizes to make sure the max/min values of the data type not exceeded (see next slides for limits)
  - For EE458 you will likely not see this problem

Integer Types

The following table provides the details of standard integer types with their storage sizes and value ranges −

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	8 bytes or (4bytes for 32 bit OS)	-9223372036854775808 to 9223372036854775807
unsigned long	8 bytes	0 to 18446744073709551615

Floating-Point Types

The following table provide the details of standard floating-point types with storage sizes and value ranges and their precision −

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Data Structures

- Arrays:
  - Of any data type
  - 0 indexed
  - Initialized to garbage!

```
type array_name [array_size];
```

- Structs (aside)
  - Kind of like objects, but without functions

```
struct StructTag {
  type field_1;
    …
  type field_n;
} structName;

Access:
structName.field_1 = value;
```

Define statements

- NOT a variable, just a value
- Equivalent to find and replace at compile time
- Example:

#define pi 3.14

- This will replace all standalone “pi” words with “3.14” at compile time. Can now treat “pi” as a constant.
- Parentheses important if #define statement is complex, otherwise weird bugs can occure

Header Files

- Basically like “import” in Python
- C has standard libraries, included as standard headers
- Can also make custom header files

```
#include <standard_header.h>
#include “custom_header.h”
```

Functions:

- Functions must either be defined before use, or declared at the top of the file and defined elsewhere (2nd option best practice);
- Params passed by VALUE
- Function declaration:

```
ret_type function_name(params);
```

- Function definition:

```
ret_type function_name(params) {
  return return_val; //return_val is of type ret_type
}
```

Variable Scoping

- Similar to Python
- Global variables accessible anywhere
- Local variables accessible only within the function.
- Functions are pass by value:

```
int i = 1;
int j = increment_by_10(i); // increments arg by 10
printf(“%d \n”, i); //prints 1
printf(“%d \n”, j); //prints 11
```

Loops

- Similar to Python

```
while (condition) {
   // do stuff
}
int i;
for(i = 0; i < 10; i++) {
   // do stuff for 10 iterations
}
```

Code Sample Walkthrough

- Code available on Canvas

Code Composer Studio

- Texas Instrument IDE for their embedded platforms
- We are using a C2000 F280049C (real-time DSP).
- Things to note about CCS:
  - Sample code available, we will start from a base sample
  - Register access available, we will modify values at the register level
    - Note: registers in embedded systems often have special functions. Register holds data for read and write. Based on what the register holds, the microcontroller may behave differently.
  - No printing to console (there is no console!). Instead use the watch window and debug
  - We will use an interrupt (code in the interrupt service routine is called every time the interrupt event occurs, in our case, on a Timer. Basically, this means this code section runs every periodically without us calling it!)

CCS Walkthrough of Sample Code

- Can download using instructions from Canvas