CMPE-013/L

Introduction to “C” Programming

Maxwell James Dunne
Metaprogramming: The C Preprocessor

Directives
Constants/Macros
Conditionals
Debugging
Preprocessor

Preprocessor stage

C Source File

C Compiler

Preprocessor

Compiler

C Header Files

Assembly Source File

 Maxwell James Dunne

CMPE-013/L: “C” Programming
Preprocessor

Operation of

- Preprocessor operates on all sources files before they're pass to the compiler
- Processes special *preprocessor directives* specified in the code
- Final text of the source file after all preprocessor directives are processed is then compiler
Preprocessor Directives

**Definition**

**Preprocessor Directives** are parts of the code that give special instructions to the compiler. They always begin with a `#` at the beginning of the line, and are used to direct the compiler with a number of specific commands.

- **Groups:**
  - `#defines`: constants, macros
  - Conditionals

- **Usage:**
  - Code organization
  - Debugging
## Preprocessor Directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define</code></td>
<td>Define a preprocessor macro.</td>
</tr>
<tr>
<td><code>#elif</code></td>
<td>Alternatively include some text based on the value of another expression, if the previous <code>#if</code>, <code>#ifdef</code>, <code>#ifndef</code>, or <code>#elif</code> test failed.</td>
</tr>
<tr>
<td><code>#else</code></td>
<td>Alternatively include some text, if the previous <code>#if</code>, <code>#ifdef</code>, <code>#ifndef</code>, or <code>#elif</code> test failed.</td>
</tr>
<tr>
<td><code>#endif</code></td>
<td>Terminate conditional text.</td>
</tr>
<tr>
<td><code>#error</code></td>
<td>Produce a compile-time error with a designated message.</td>
</tr>
<tr>
<td><code>#if</code></td>
<td>Conditionally include text, based on the value of an expression.</td>
</tr>
<tr>
<td><code>#ifdef</code></td>
<td>Conditionally include text, based on whether a macro name is defined.</td>
</tr>
<tr>
<td><code>#ifndef</code></td>
<td>Conditionally include text, based on if a name is not a defined macro.</td>
</tr>
<tr>
<td><code>#include</code></td>
<td>Insert text from another source file.</td>
</tr>
<tr>
<td><code>#line</code></td>
<td>Reset the line number for compiler output</td>
</tr>
<tr>
<td><code>#pragma</code></td>
<td>Allows for extending preprocessor directives beyond what's in the standard</td>
</tr>
<tr>
<td><code>#</code></td>
<td>Null directive</td>
</tr>
<tr>
<td><code>#warning</code></td>
<td>Emits a warning described by the rest of the line</td>
</tr>
</tbody>
</table>
Preprocessor Directives

Text substitution using `#define`

- Defines a text substitution label

**Syntax**

```
#define label text
```

- Each instance of `label` will be replaced with `text` by the preprocessor unless `label` is inside a string
- `text` is optional
- Uses no memory

**Example**

```
#define PI 3.14159
#define MOL 6.02E23
#define MCU "PIC32MX320F128H"
#define PI_2 2 * PI
#define __STDC_ISO_649_H__
```
Preprocessor Directives

Text substitution using `#define`

- Labels must be valid identifiers

Example

```
#define 0 1
#define _WRONG
#define __WRONG
#define RIGHT
```
Preprocessor Directives

Text substitution using `#define`

• Text goes until the end of the line
  – Unless newline is escaped with a `\`

Example
```
#define true false
#define true \false
```

• Constants can be nested

Example
```
#define OLED_NUM_LINES (OLED_DRIVER_PIXEL_ROWS \ / ASCII_FONT_HEIGHT)
```
# Preprocessor Directives

## Predefined constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FILE</strong></td>
<td>Full path of current file</td>
</tr>
<tr>
<td><strong>LINE</strong></td>
<td>The current line in the file</td>
</tr>
<tr>
<td><strong>DATE</strong></td>
<td>The current date as a string, like &quot;Jan 27 2014&quot;</td>
</tr>
<tr>
<td><strong>TIME</strong></td>
<td>The current time as a string, like &quot;17:20:50&quot;</td>
</tr>
<tr>
<td><strong>func</strong></td>
<td>The current function as a string, like &quot;main&quot;</td>
</tr>
<tr>
<td><strong>DEBUG</strong></td>
<td>When debugging is specified in MPLAB X, not part of the standard!</td>
</tr>
</tbody>
</table>
Preprocessor Directives

#undef

Syntax

```markdown
#undef LABEL
```

- Deletes a macro definition
- Allows you to change a macro
  - Error when macros are redefined otherwise

Example

```c
#define M_PI 3.14
#undef M_PI
#define M_PI 3.141592653589793238462643383279502884197
```
Preprocessor Directives

Argument Macros

- Create a function-like macro

**Syntax**

```c
#define LABEL(arg₁, ..., argₙ) code
```

- The `code` must fit on a single line or use '\\' to split lines
- Text substitution used to insert arguments into `code`
- Each instance of `LABEL()` will be expanded into `code`
- This is not the same as a C function! No stack allocation.

**Example**

```c
#define MIN(x, y) ((x) < (y) ? (x) : (y))
#define SQUARE(x) ((x) * (x))
#define SWAP(x, y) { (x) ^= (y); (y) ^= (x); (x) ^= (y); }
```
Preprocessor Directives

Argument Macros – Side Effects

Example

#define SQUARE(x) x * x

Extreme care must be exercised when using macros. Consider the following use of the above macro:

i = 5;
a = SQUARE(i + 3);
Preprocessor Directives

Argument Macros – Side Effects

Example

```c
#define SQUARE(x) ((x)*(x))
```

Extremely care must be exercised when using macros. Consider the following use of the above macro:
```
i = 5;
a = SQUARE(i++);
```

```
(i++) * (i++)
```
#define ABS(x) (((x) > 0) ? (x) : (-x))
#define NORM1(x, y) (ABS((x)) + ABS((y)))

int x = NORM1(5, 6.6);

int x = ((((5) > 0)?(5):(-5)) + ((((6.6) > 0)?(6.6):(-6.6))));
Macros with `define

Emulating functions

- Functions provide useful features:
  - Encapsulation
  - Evaluate as an expression
  - Return values
Preprocessor Directives

Emulating functions

- For encapsulation

Example

```c
#define LABEL(arg1, ..., argn) {
    ...
    }
```

- Code blocks forces all code in the macro to execute in the same context
  - Also allows for temporary variables within the macros
#define INIT() TRISA = 5; LATA = 5;

if (beginStartup)
    INIT();

    LATA = 5;
Preprocessor Directives

Emulating functions

Example

```c
#define INIT() {TRISA = 5; LATA = 5;};

if (beginStartup)
    INIT();
else
    ...
```
Preprocessor Directives

Emulating functions

- For encapsulation with expression-ness

Example

```c
#define LABEL(arg1, ..., argn) do {
    ...
    } while (0)
```

- Code blocks forces all code in the macro to execute in the same context
  - Also allows for temporary variables within the macros
- `while`-statement allows for semi-colon termination
  - Generates a single statement
Preprocessor Directives

Emulating functions

• To "return" values, just have the statement evaluate to a value

Example

```c
#define LABEL(arg1, ..., argn) VALUE
```
Preprocessor Directives
Stringification of macro values

Example

```
#define VERSION 6.3
#define TEXTIFY(x) #x

printf("%s", TEXTIFY(VERSION));
```

```
6.3
```
Preprocessor Directives

Stringification of macro values

- You need another layer of indirection

Example

```c
#define TEXTIFY(x) TEXTIFY_HELPER(x)
#define TEXTIFY_HELPER(x) #x
#define MAJOR_VER 1
#define MINOR_VER 3
#define VERSION_STRING TEXTIFY(MAJOR_VER) \ 
    "." \ 
    TEXTIFY(MINOR_VER)

printf("%s", TEXTIFY(VERSION));
```

1.3
Preprocessor Directives

Token concatenation

• To combine argument with existing token to generate identifiers

Example

```c
#define DEBUGIFY(x) x ## _DEBUG

printf("%s", DEBUGIFY(asdf));
```
Preprocessor Directives
Conditional compilation

• Control what code actually gets compiled
  – Already seen this with header guards

Example

```c
#ifndef BUTTONS_H
#define BUTTONS_H

...
#endif
```
Preprocessor Directives
Conditional compilation

- Family of if-statements
  - #if
  - #ifdef
  - #ifndef
- Ended with #endif
- #if is the general case
  - #ifdef/#ifndef only check if a macro has been defined
Preprocessor Directives

Emulating functions

Example

```c
#include INIT

#include 0

#include defined(_WIN32)

#include defined(__unix__) && !defined(__APPLE__)

#include __STDC_VERSION__ > 199409L
```
Preprocessor Directives

Conditional compilation

- `#ifdef text`
  - Same as `#if defined(...)`
- `#ifndef text`
  - Same as `#if !defined(...)`
- `#elif text`
  - Else-if, follows same rules as `#if`
- `#else`
- `#endif`
Preprocessor Directives

Unit testing

- Conditionally compile in test code

Example

```c
int main(void)
{
    // Initialization code

    #if 0
    // Test code
    #endif

    // Main program
}
```
Preprocessor Directives

Fatal errors

- Output location of failure and stop running

Example

```c
#define FATAL_ERROR()  
   do {  
       printf("FATAL ERROR at %s:%s():%d\n",  
                __FILE__, __func__, __LINE__);  
       TRISE = 0;  
       LATE = 0xFF;  
   } while (1);
```
Preprocessor Directives

Forcing compilation errors/warnings

• `#warning text`
  – Outputs compilation warning

• `#error text`
  – Outputs compilation error

Example

```c
#if __STDC_VERSION__ < 199901
#error "Must be compiled with C99 or greater"
#endif
```
main() { 
  if(event) 
    count += 1; 
    event = TRUE; 
}
8 samples

All CBuF0 ... 7

223

0 - 1023

1023

1021
Bounce

- Digital I/O
- A/D
- Timers
- Debouncing

LED5.h
Button event

\[
\begin{array}{c}
\text{low} \\
\text{high}
\end{array}
\]

\[\frac{1}{T}\]
Low

Lost event \neq \text{low}

High

Lost event \neq \text{High}
One button work
<open and paste
if (a[0] == 0) ee a[1] == 1...

[a] = BTN1

a <= 1;

[a1] = BTN1;

1

10

1x
while (1) ≤
if (event) ≤
  ≤
  3