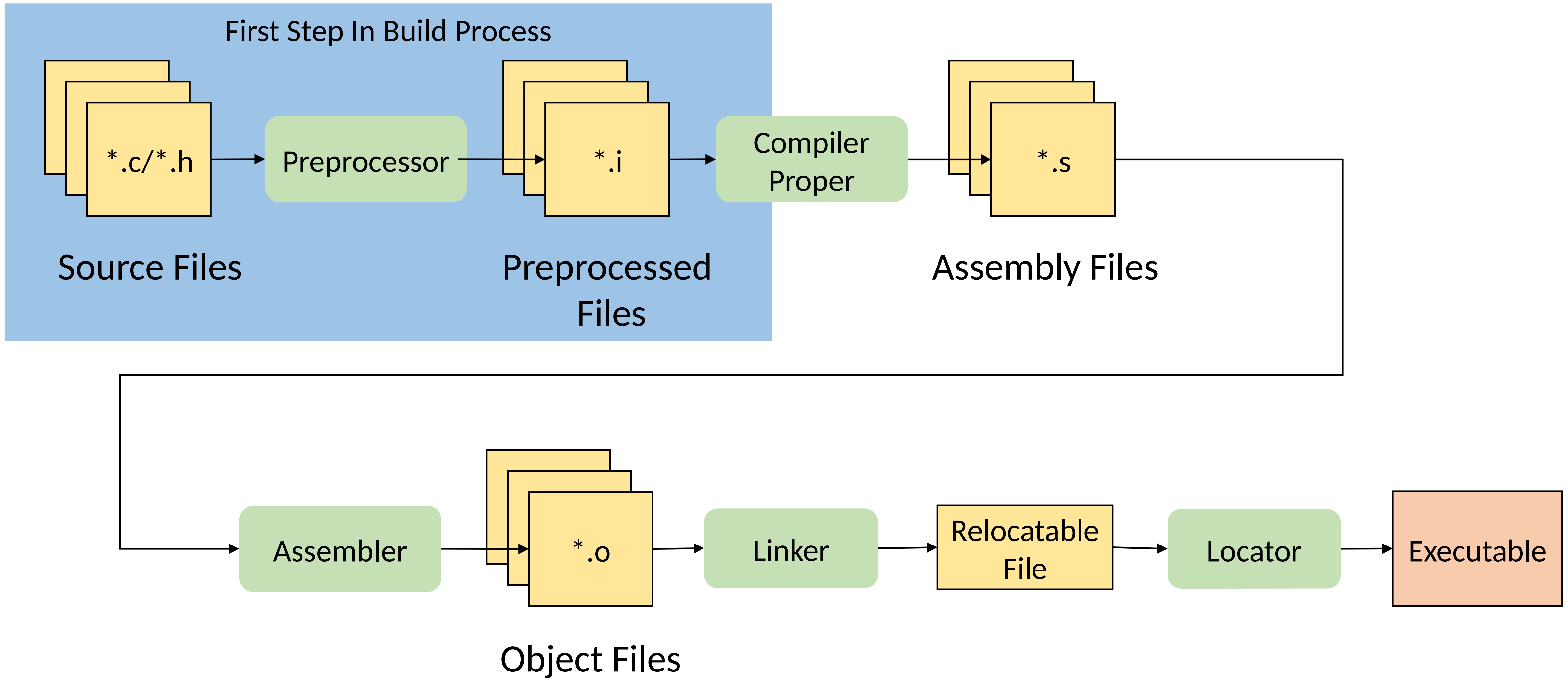
Embedded Software Essentials

The Preprocessor

C1 M2 V3

Copyright

The Preprocess [S1.2.3.1]

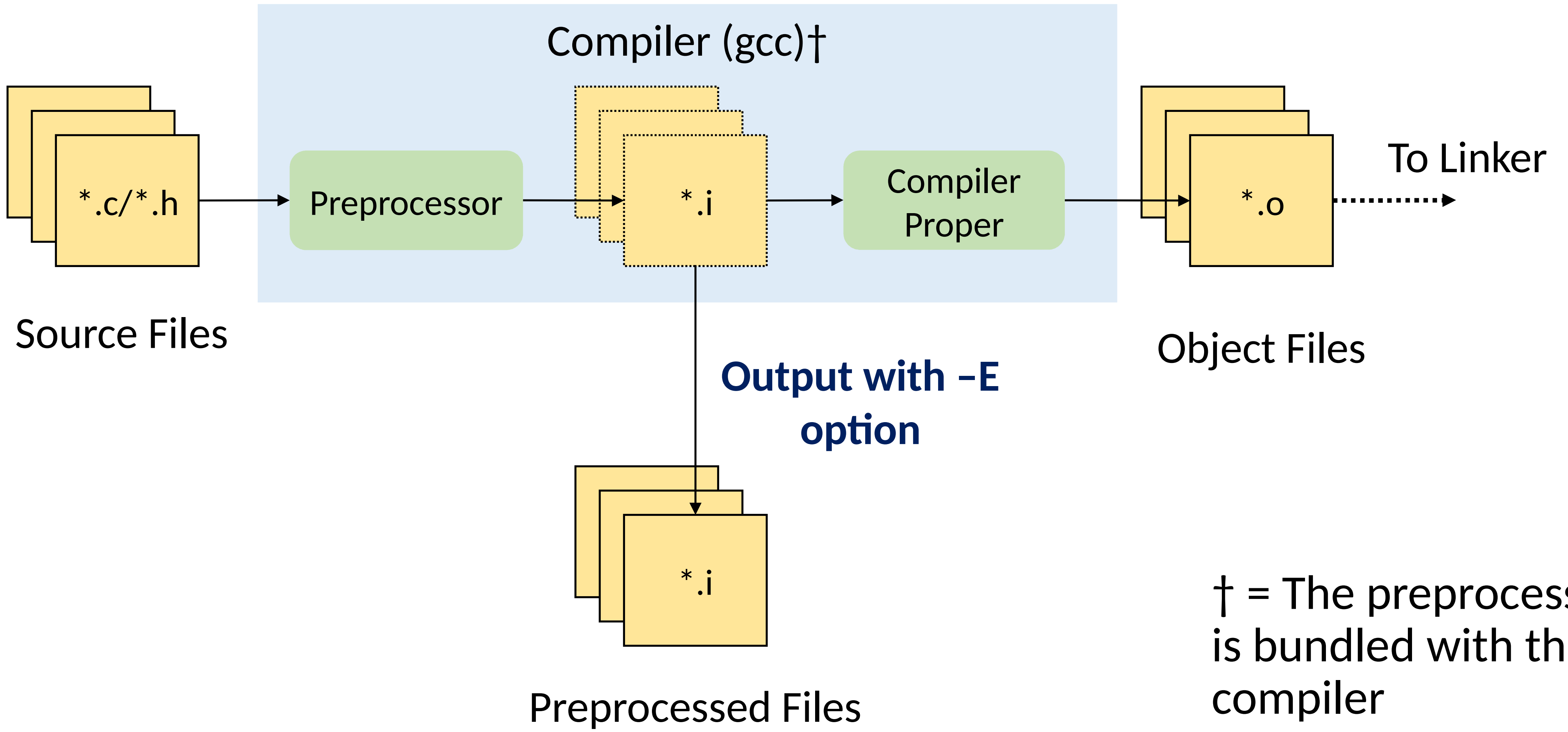


Preprocessor Directives [S1.2.3.2]

- Special keywords used by the preprocessor before compilation
  - **Compile Time switches**
- Directives start with ‘**#**’ sign

- Important Directives
  - #define, #undef
  - #ifndef, #ifdef, #endif
  - #include
  - #warning, #error
  - #pragma

Preprocessor’s Role **[S1.2.3.3.a]**



† = The preprocessor is bundled with the compiler

Preprocessed Output [S1.2.3.3.b]

- Stop after preprocessing
- Output the preprocessed file to a ***.i** extension

```
$ gcc -E -o main.i main.c
```

#define as a Constant [S1.2.3.4]

- Used for defining constants, features or macro functions

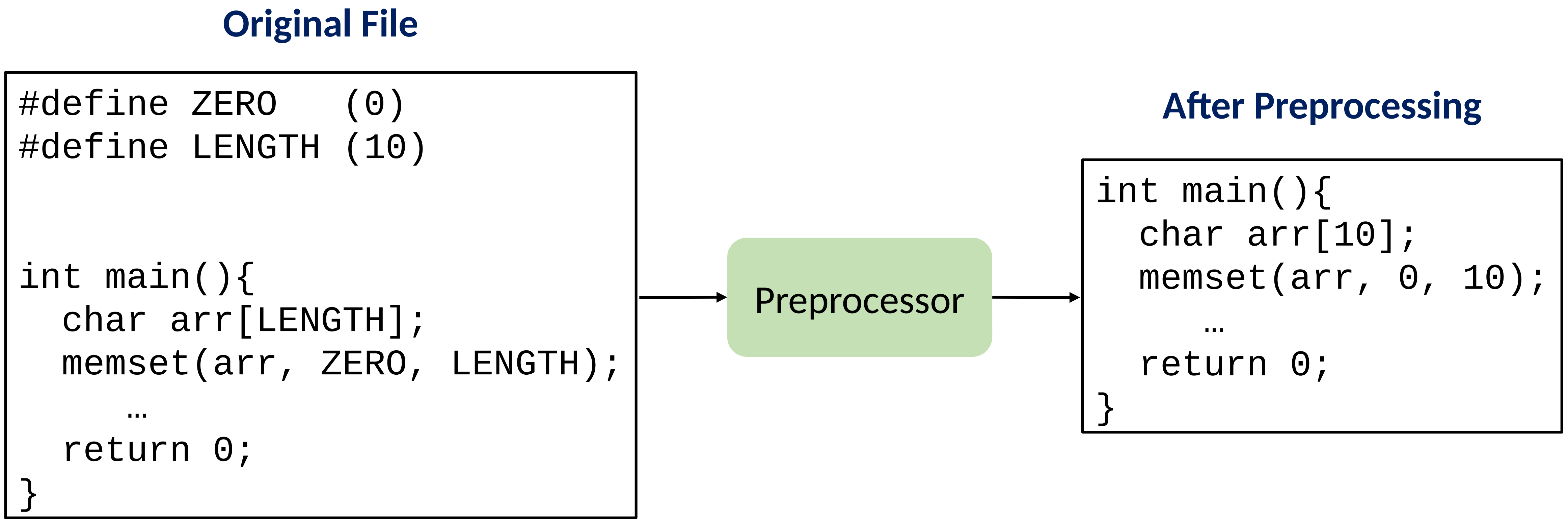
```
#define <MACRO-NAME> <MACRO-VALUE>
```

- Constant Examples:

```
#define LENGTH      (10)
#define NO_ERROR    (0)
#define ERROR       (1)

/* Macro defined as another macro */
#define UART_ERROR  ERROR
```

Macro Substitution [S1.2.3.5a]

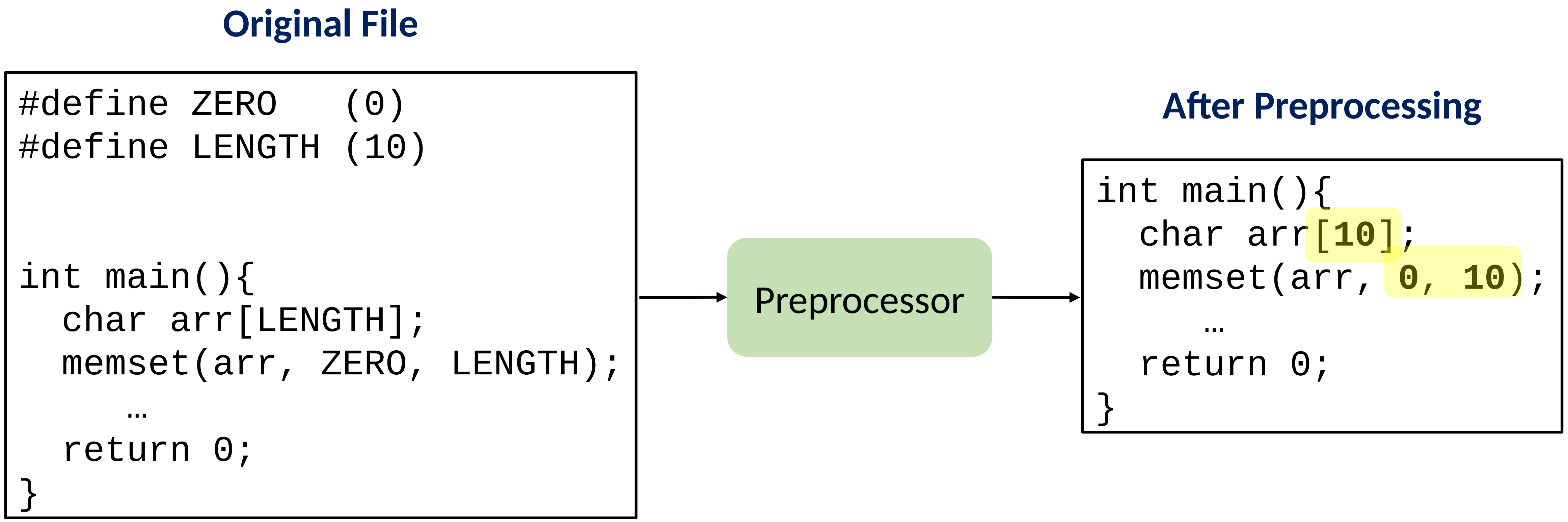


```
#define ZERO   (0)
#define LENGTH (10)


int main(){
  char arr[LENGTH];
  memset(arr, ZERO, LENGTH);
     …
  return 0;
}
```

```
int main(){
  char arr[10];
  memset(arr, 0, 10);
     …
  return 0;
}
```

Macro Substitution [S1.2.3.5b]



```
#define ZERO   (0)
#define LENGTH (10)


int main(){
  char arr[LENGTH];
  memset(arr, ZERO, LENGTH);
     …
  return 0;
}
```

```
int main(){
  char arr[10];
  memset(arr, 0, 10);
     …
  return 0;
}
```

#define as a Macro Function [S1.2.3.6]

- Provide Macro Function name, parameters, and operation

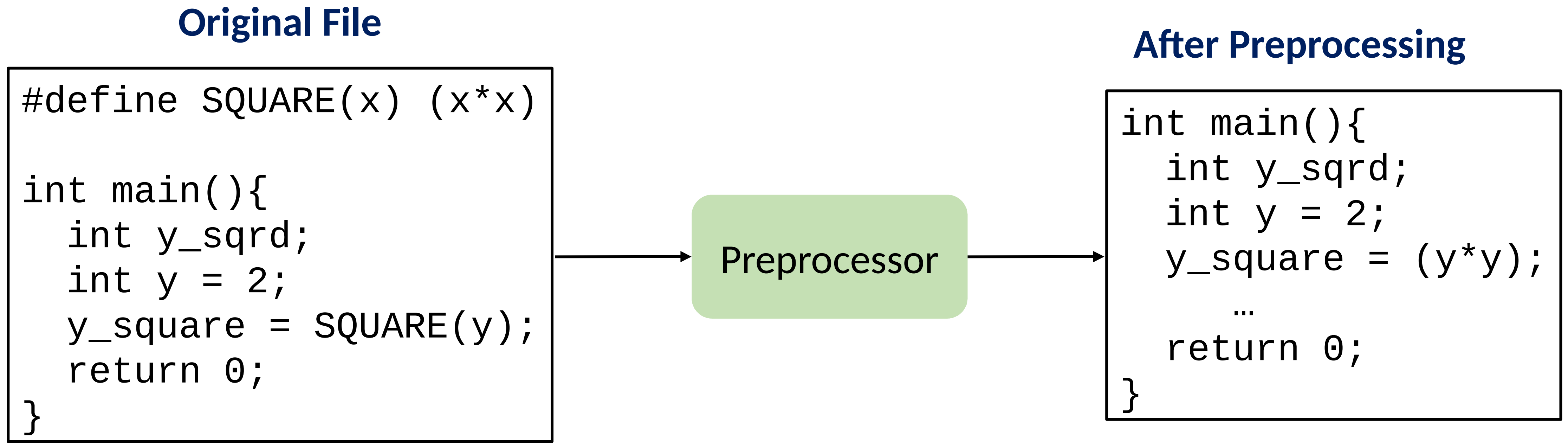
```
#define <MACRO-FUNCTION>(<PARAMS>) (<OPERATION>)
```

- Constant Examples:

```
#define SQUARE(x) (x*x)
    …
int y_sqrd;
int y = 2;
y_square = SQUARE(y);
```

⇨ **y_square will equal 4**

Macro Function Substitution [S1.2.3.7a]



Original File

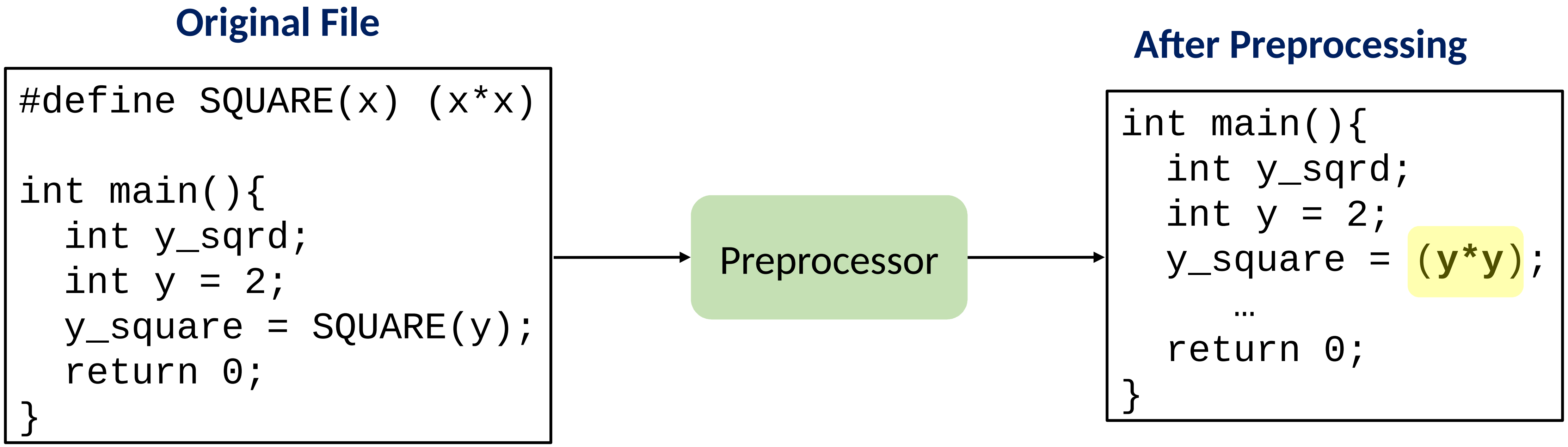
```
#define SQUARE(x) (x*x)

int main(){
  int y_sqrd;
  int y = 2;
  y_square = SQUARE(y);
  return 0;
}
```

After Preprocessing

```
int main(){
  int y_sqrd;
  int y = 2;
  y_square = (y*y);
     …
  return 0;
}
```

Macro Function Substitution [S1.2.3.7b]

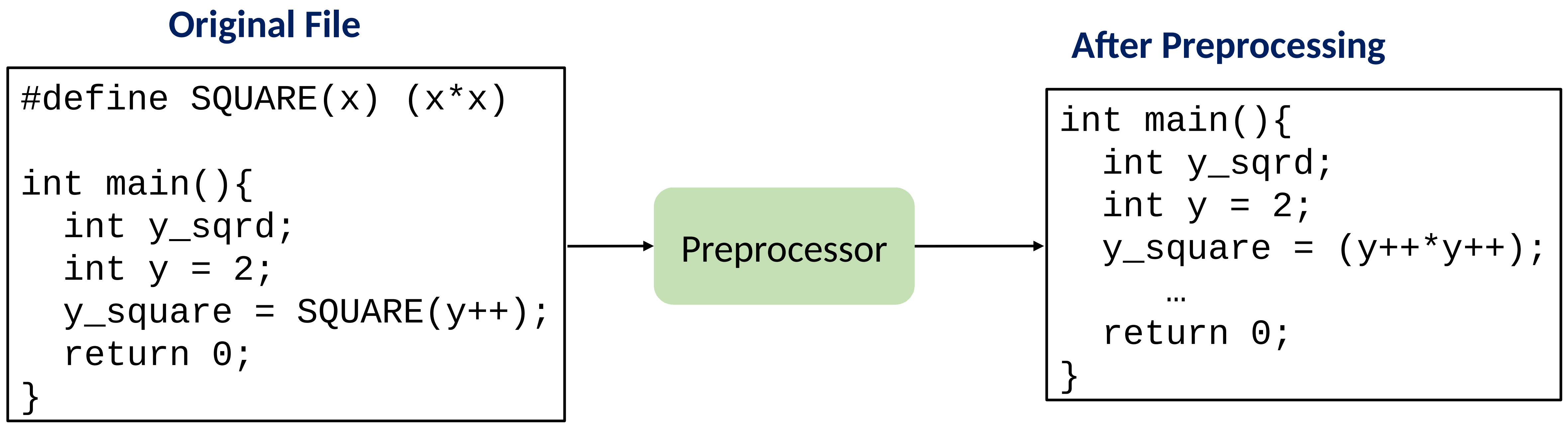


```
#define SQUARE(x) (x*x)

int main(){
  int y_sqrd;
  int y = 2;
  y_square = SQUARE(y);
  return 0;
}
```

```
int main(){
  int y_sqrd;
  int y = 2;
  y_square = (y*y);
     …
  return 0;
}
```

Macro Function Issues [S1.2.3.8a]



Original File

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

int main(){
  int y_sqrd;
  int y = 2;
  y_square = SQUARE(y++);
  return 0;
}
```

After Preprocessing

```
int main(){
  int y_sqrd;
  int y = 2;
  y_square = (y++*y++);
     …
  return 0;
}
```

Macro Function Issues [S1.2.3.8b]

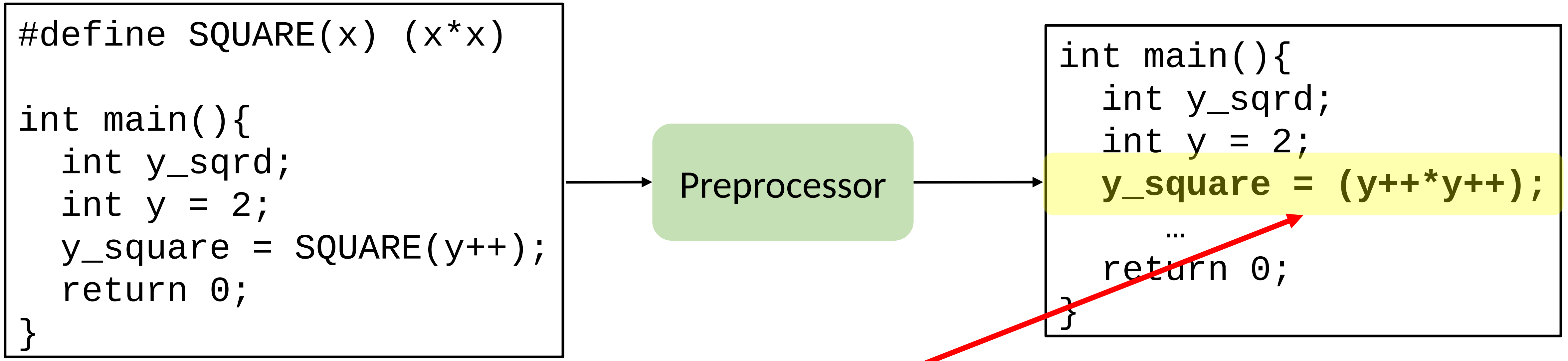
After the line y_square executes:

y = 4

y_square = 6

Original File

After Preprocessing



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

int main(){
  int y_sqrd;
  int y = 2;
  y_square = SQUARE(y++);
  return 0;
}
```

```
int main(){
  int y_sqrd;
  int y = 2;
  y_square = (y++*y++);
     …
  return 0;
}
```

Undefined Behavior!!!

#define/#undef as a Feature [S1.2.3.9]

- Directive used for Boolean Compilation Conditions

```
#define <FEATURE-NAME>
```

- Constant Examples:

```
/* Define feature for the MSP */
#define MSP_PLATFORM

#define TEN      (10)
/* Undefine the Constant TEN */
#undef  TEN
```

```
#define KL25_PLATFORM
/* Undefine the Feature */
#undef KL25_PLATFORM
```

#if-else Directives [S1.2.3.10]

- Conditionally compile blocks of code
  - #ifdef
  - #ifndef
  - #elif
  - #else
  - #endif – End of block **(required)**

- Useful for debugging

- **“Turn Off”** Large amounts of code

```
#define COMPILE_CODE
…
#ifdef COMPILE_CODE
     // Code will be compiled
#endif
```

```
#define DO_NOT_COMPILE_CODE
…
#undef DO_NOT_COMPILE_CODE
#ifdef DO_NOT_COMPILE_CODE
     // Code will be NOT be compiled
#endif
```

#if-else & #define Directives [S1.2.3.11]

```
int main(void){

#ifdef ( KL25_PLATFORM ) && ( ! MSP_PlATFROM )
  kl25_initialize();
#elif ( MSP_PlATFROM ) && ( ! KL25_PLATFORM )
  msp_initialize();
#else
  #error “Please specify one platform target”
#endif

  /* More code here */

  return 0;
}
```

#include Directive [S1.2.3.12]

- Includes software defined in other files
- Declarations get copied into file
- Include file from local directory

```
#include “uart.h”
```

- Include file from a library path or include path:

```
#include <stdio.h>
```

#include Directive [S.1.2.3.13a]

my_file.c

```
#include “my_file.h”
char arr[LENGTH];

void clear(char * ptr, int size){
 int i;
  for(i = 0, i < size, i++){
    ptr[i] = 0;
  }
}
```

my_file.h

```
#define LENGTH (10)
void clear(char * ptr, int size);
```

Preprocessed

my_file.i

```
void clear(char * ptr, int size);

char arr[10];

void clear(char * ptr, int size){
  int i;
  for(i = 0, i < size, i++){
    ptr[i] = 0;
  }
}
```

#include Directive [S.1.2.3.13b]

my_file.c

```
#include “my_file.h”
char arr[LENGTH];

void clear(char * ptr, int size){
 int i;
  for(i = 0, i < size, i++){
    ptr[i] = 0;
  }
}
```

my_file.h

```
#define LENGTH (10)
void clear(char * ptr, int size);
```

Preprocessed

$\longrightarrow$

my_file.i

```
void clear(char * ptr, int size);

char arr[10];

void clear(char * ptr, int size){
  int i;
  for(i = 0, i < size, i++){
    ptr[i] = 0;
  }
}
```

#pragma [S1.2.3.14]

- Gives a specific instruction to the compiler
  - Controls compilation from software instead of command line

- Implementation/Compiler specific ␣ Unrecognized pragmas will be ignored

- Adds options to compiler for specific function

```
#pragma GCC push_options
```

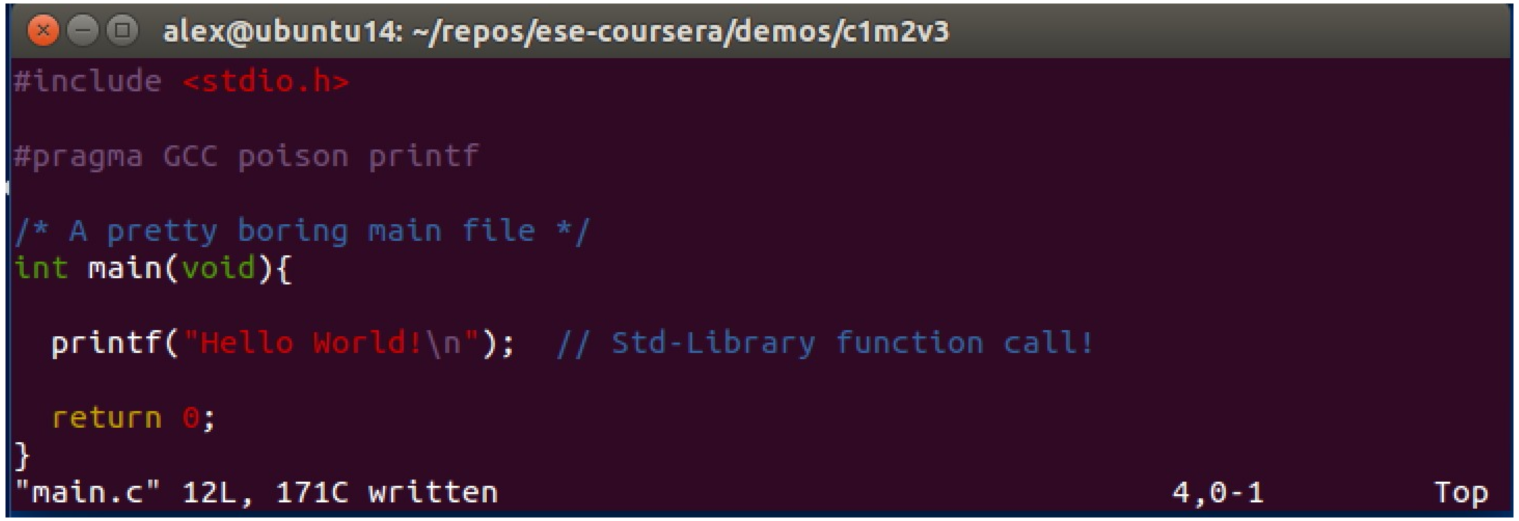
- Causes an error during compilation if code uses these functions

```
#pragma GCC poison printf sprint fprintf
```

- Compile a function with a specific architecture

```
#pragma GCC target (“arch=armv6”)  -or-
                   (“cpu=cortex-m0plus”)
```

Pragma Compile Failure [S1.2.3.15]



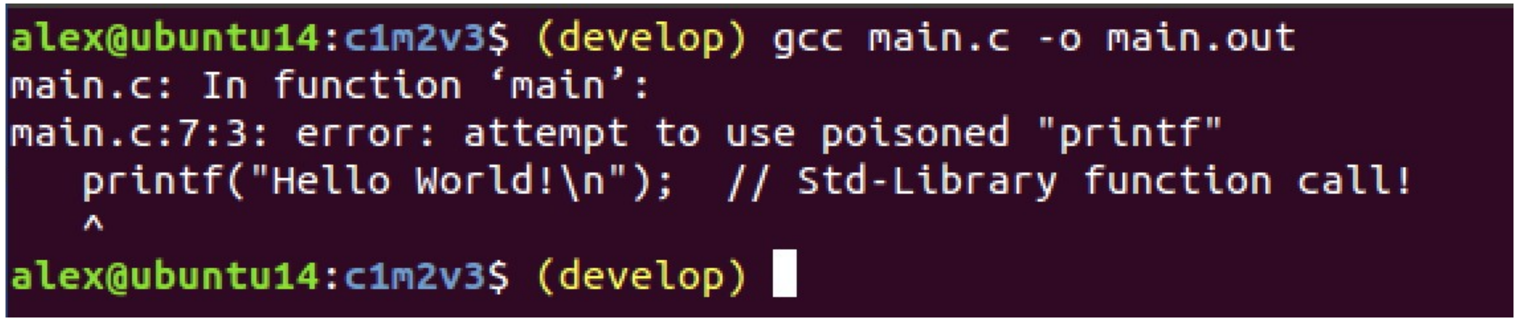
```
#include <stdio.h>

#pragma GCC poison printf

/* A pretty boring main file */
int main(void){

  printf("Hello World!\n");  // Std-Library function call!

  return 0;
}
"main.c" 12L, 171C written                                        4,0-1         Top
```



```
alex@ubuntu14:c1m2v3$ (develop) gcc main.c -o main.out
main.c: In function ‘main’:
main.c:7:3: error: attempt to use poisoned "printf"
   printf("Hello World!\n");  // Std-Library function call!
   ^
alex@ubuntu14:c1m2v3$ (develop) 
```

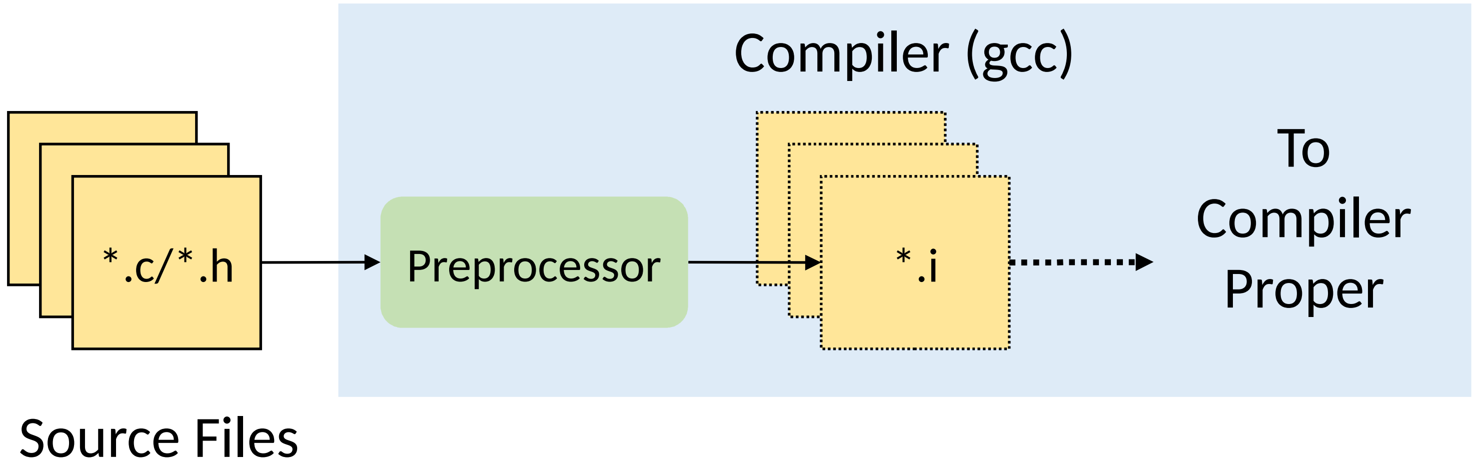
Compile Time Switch [S1.2.3.16]

- Condition provided at Compile time to dictate **WHAT** should be compiled
  - Uses combination of #if-else and #define directives

```
#if defined ( KL25Z_PLATFORM ) && ! defined ( MSP_PLATFROM )
  kl25_initialize();
#elif ( MSP_PLATFROM ) && ( ! KL25Z_PLATFORM )
  msp_initialize();
#else
  #error “Please specify one platform target”
#endif
```

Compile Time Switch [S1.2.3.17]

- Condition provided at Compile time to dictate **WHAT** should be compiled
  - Uses combination of #if-else and #define directives

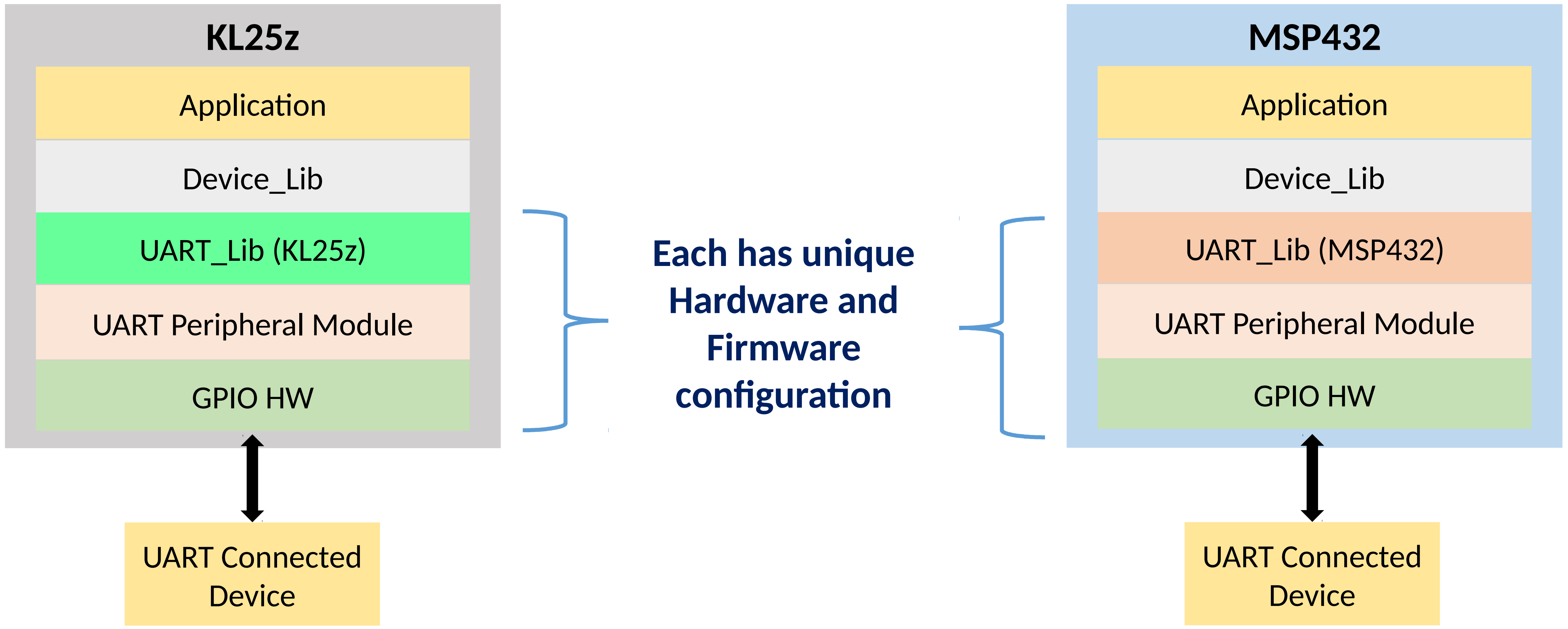


Add extra option to gcc command to define Macro

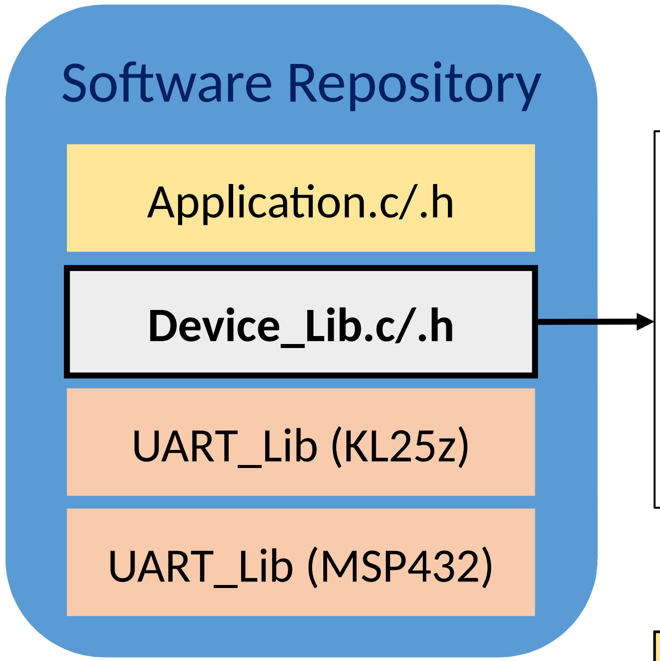
```
–D<MACRO-NAME>
```

```
$ gcc –DMSP_PLATFORM –o main.out main.c
```

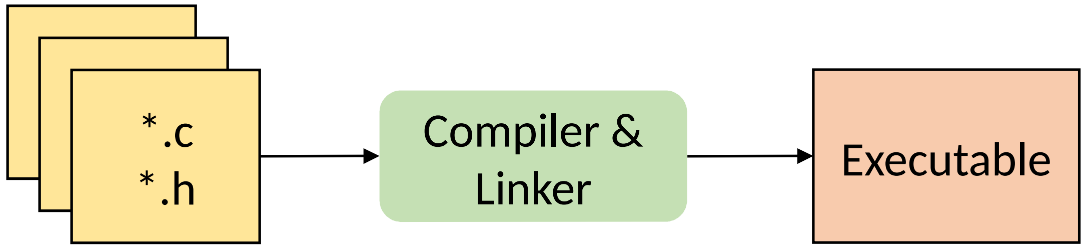
Compile Time Switch [S1.2.3.18a]

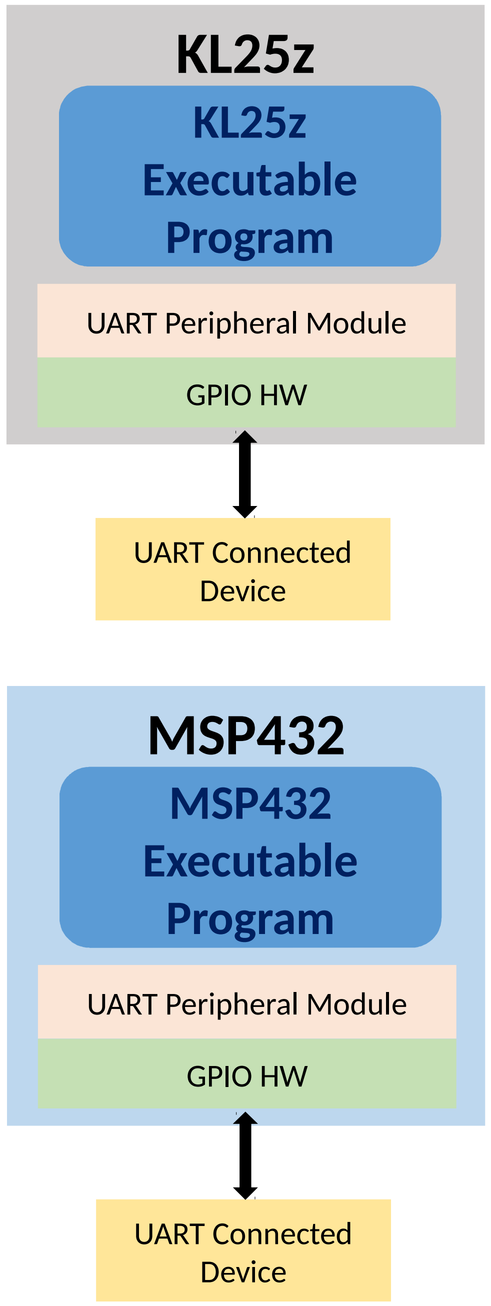


Compile-Time Switch [S1.2.3.1

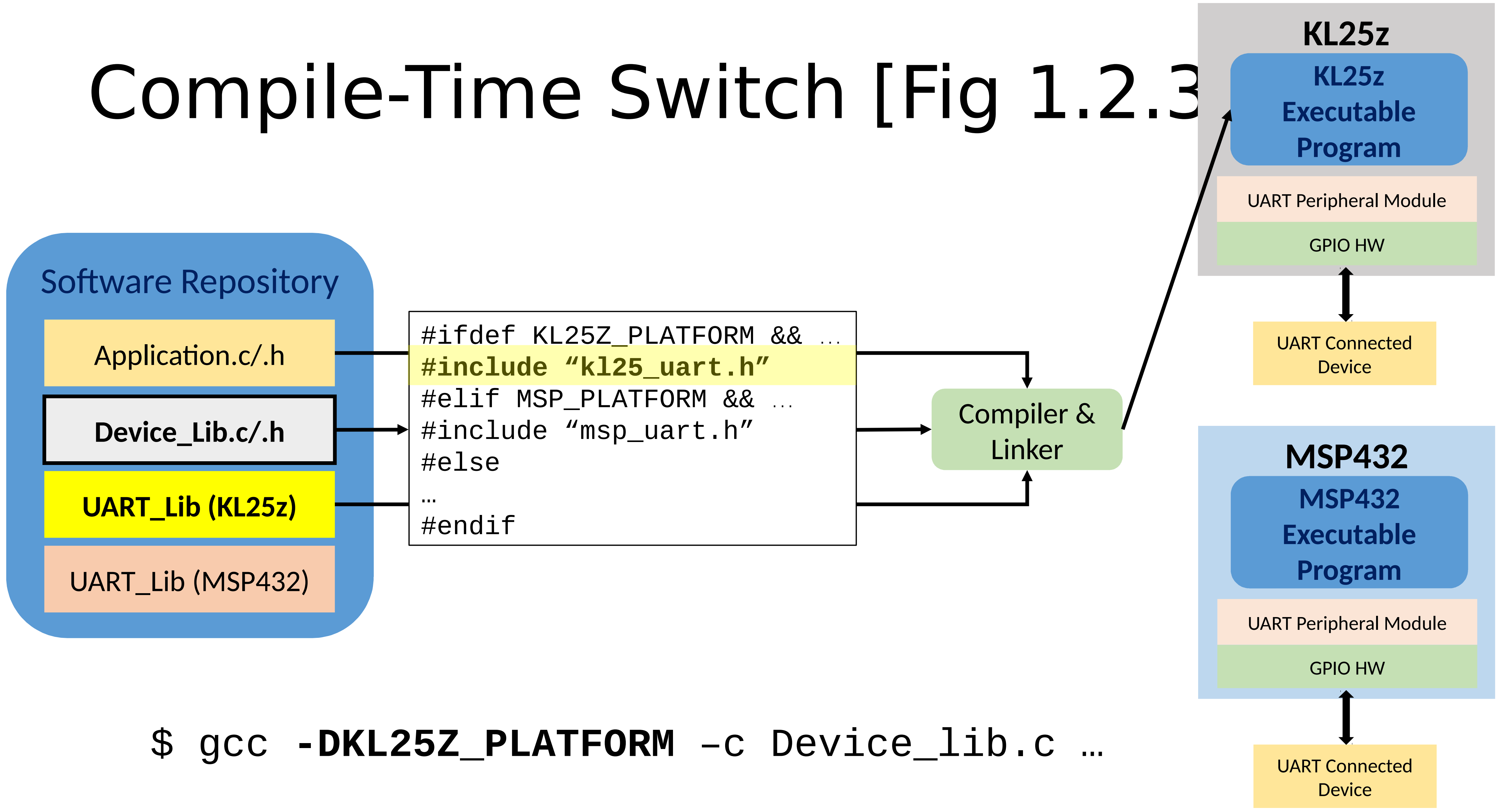


```
#ifdef KL25Z_PLATFORM && …
#include “kl25_uart.h”
#elif MSP_PLATFORM && …
#include “msp_uart.h”
#else
…
#endif
```





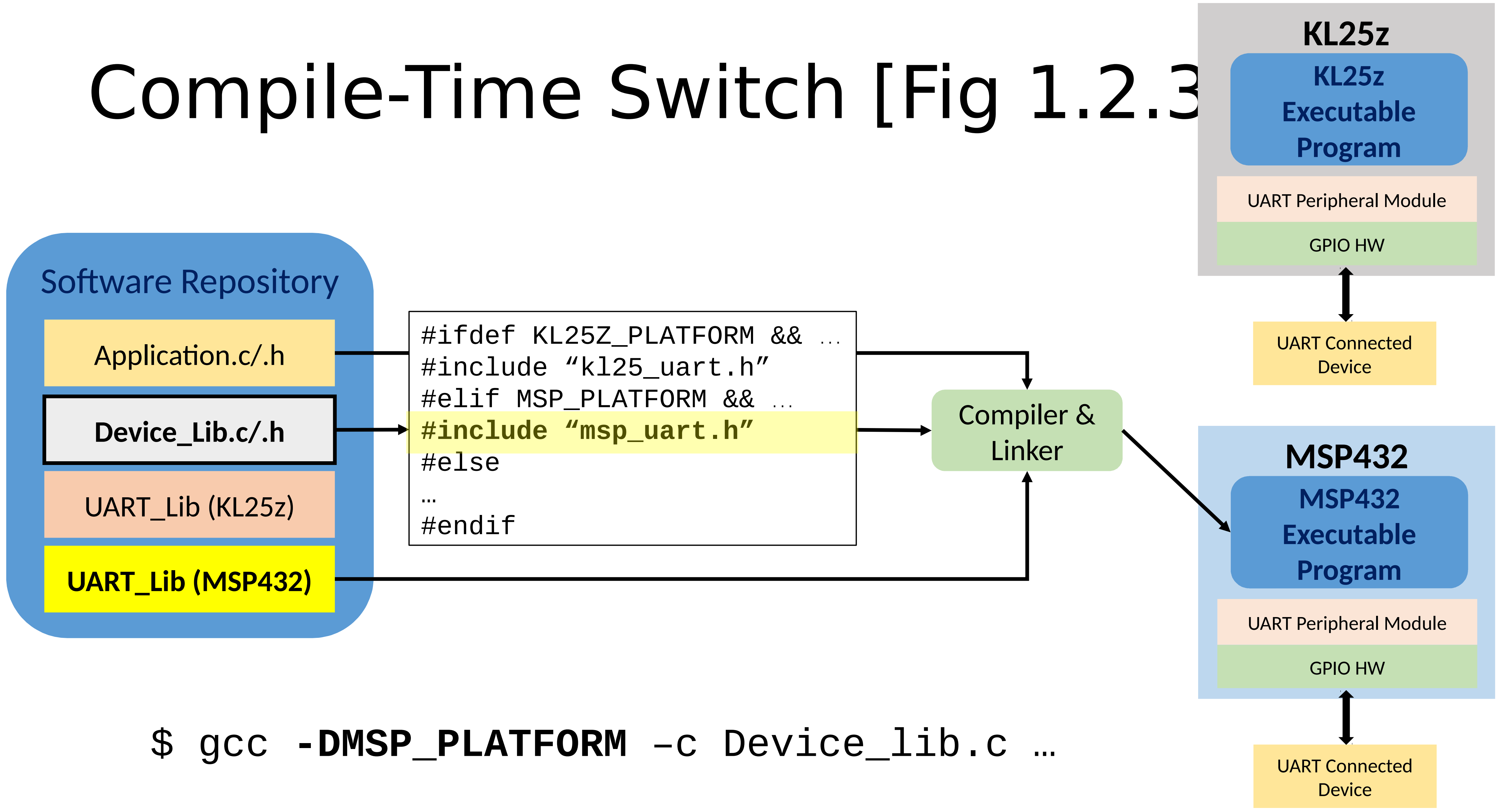
Compile-Time Switch [Fig 1.2.3



```
#ifdef KL25Z_PLATFORM && …
#include “kl25_uart.h”
#elif MSP_PLATFORM && …
#include “msp_uart.h”
#else
…
#endif
```

```
$ gcc -DKL25Z_PLATFORM –c Device_lib.c …
```

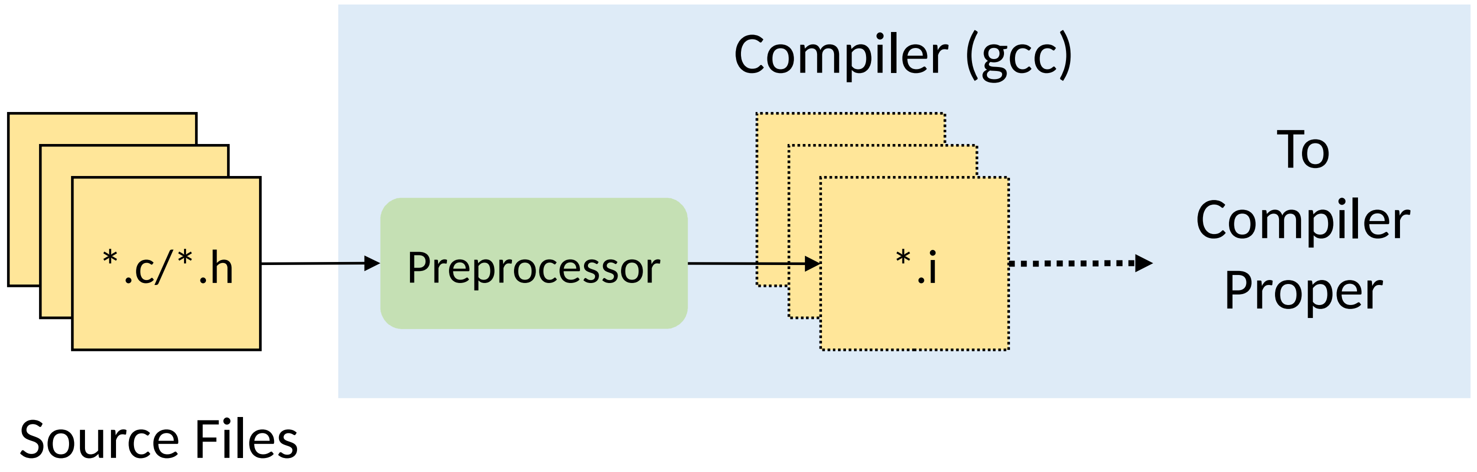
Compile-Time Switch [Fig 1.2.3



```
#ifdef KL25Z_PLATFORM && …
#include “kl25_uart.h”
#elif MSP_PLATFORM && …
#include “msp_uart.h”
#else
…
#endif
```

```
$ gcc -DMSP_PLATFORM –c Device_lib.c …
```

Preprocessor Command Line Define **[Unused]**



Add extra option to gcc command line to define Macro → `-D<`**`MACRO-NAME`**`>`

```
$ gcc -DMSP_PLATFORM -o main.out main.c
```