Embedded Software Essentials

Creating Header and Implementation Files

C1 M2 V4

Copyright

Software Modules and Libraries [S1.2.4.1]

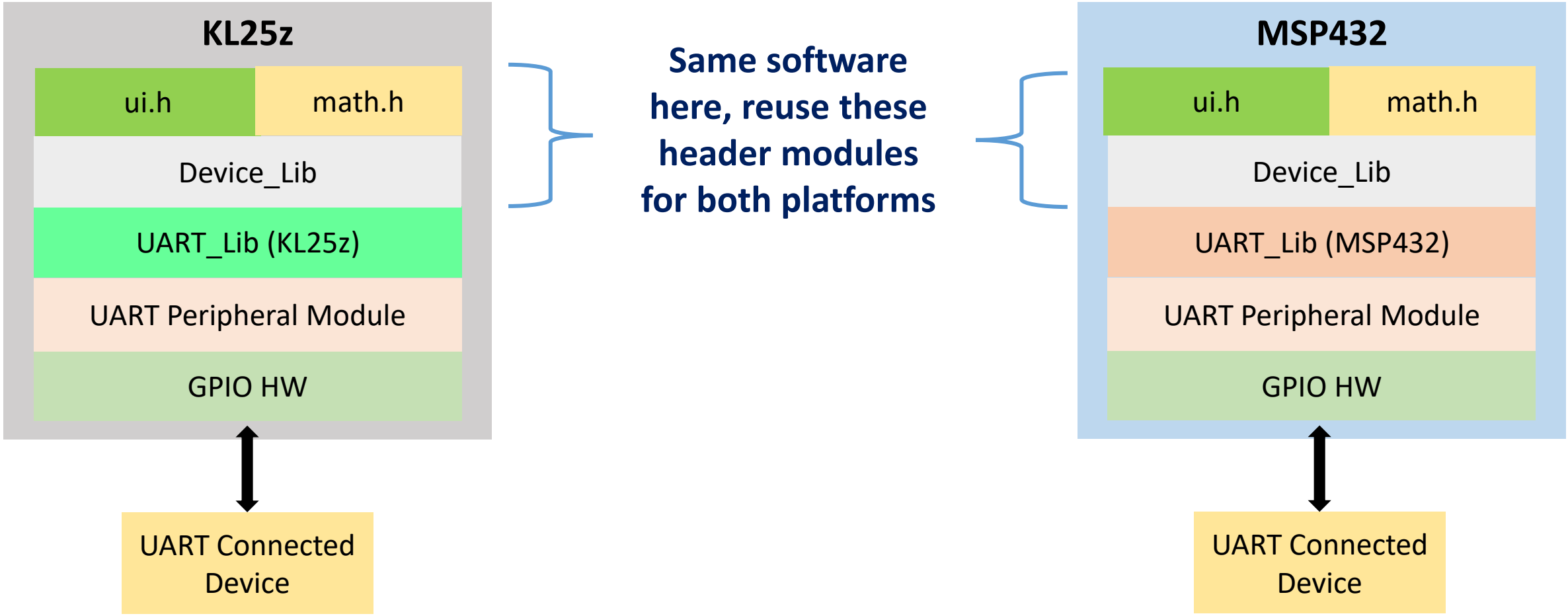
- **Libraries** – Collection of software (precompiled or direct source)

- **Modules** – Software organization that each module has encapsulated certain functionality within a library
  - Create portable code!

```
#include <math.h>
```

Performs math operations like square root

Code Reuse [S1.2.4.1]



Creating Modules [S1.2.4.3]

- **Implementation files (*.c):** Contains the function definitions or the actual implementation details

- **Header files (*.h):** Contain the function declarations, macros, & derived data type definitions (structs, enums)

memory.c

```
#include "memory.h"
char memzero(char * src, int length){
  int i;
  for(i = 0; i < length; i++){
    *src++ = 0;
  }
}
```

memory.h

```
#ifndef __MEMORY_H__
#define __MEMORY_H__

char memzero(char * src, int length);

#endif /* __MEMORY_H__ */
```

Include Guards [S.1.2.4.5a]

- Top of Header file contains a ***#ifndef*** statement
  - Protects against redundant includes

main.c

```
#include "memory.h"
#include "memory.h"

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

memory.h

```
#ifndef __MEMORY_H__
#define __MEMORY_H__

char memzero(char * src, int length);

#endif /* __MEMORY_H__ */
```

Include Guards [S.1.2.4.5b]

- Top of Header file contains a ***#ifndef*** statement
  - Protects against redundant includes

main.c

```
#include "memory.h"
#include "memory.h"

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

memory.h

```
#ifndef __MEMORY_H__
#define __MEMORY_H__

char memzero(char * src, int length);

#endif /* __MEMORY_H__ */
```

These include guards protect the main file from repeated declarations

Include Guards [S.1.2.4.5c]

- Top of Header file contains a ***#ifndef*** statement
  - Protects against redundant includes

main.c

```
#include "memory.h"
#include "memory.h"

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

memory.h

```
/* No Include guard */

char memzero(char * src, int length);
```

Include Guards [S.1.2.4.5d]

- Top of Header file contains a ***#ifndef*** statement
  - Protects against redundant includes

memory.h

```
/* No Include guard */


char memzero(char * src, int length);
```

main.c

```
char memzero(char * src, int length);
char memzero(char * src, int length);


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

This causes a compile error for duplicate declarations of the memzero function

Include Guards [S.1.2.4.5e]

- Top of Header file contains a ***#ifndef*** statement
  - Protects against redundant includes

main.c

```
#include "memory.h"
#include "memory.h"

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

memory.h

```
#ifndef __MEMORY_H__
#define __MEMORY_H__

char memzero(char * src, int length);

#endif /* __MEMORY_H__ */
```

Include Guards [S.1.2.4.5f]

- Top of Header file contains a ***#ifndef*** statement
  - Protects against redundant includes

main.c

```
char memzero(char * src, int length);


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

memory.h

```
#ifndef __MEMORY_H__
#define __MEMORY_H__

char memzero(char * src, int length);

#endif /* __MEMORY_H__ */
```

No error here because only one declaration is used

Pragma Once [S.1.2.4.6a]

- #pragma once
  - One-line Include guard
  - Non-standard
- **Not Portable!**

memory.h

```
#pragma once

char memzero(char * src, int length);
```

main.c

```
#include "memory.h"
#include "memory.h"

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

Pragma Once [S.1.2.4.6b]

- #pragma once:
  - One-line Include guard
  - Non-standard
- **Not Portable!**

memory.h

```
#pragma once

char memzero(char * src, int length);
```

main.c

```
char memzero(char * src, int length);


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

No Error Here because only one declaration is used

Header Files [S1.2.4.7a]

- Header files are the **interface**
- Anything you want to give access to, put in header file
- Make Informative function comments in header File
  - Function Description
  - **Inputs**: type and description
  - **Return**: type and description

```
#ifndef __MEMORY_H__
#define __MEMORY_H__
/*************************************
 *  memzero() - Takes a pointer to a
 *       location in memory and sets
 *       the contents to zero for
 *       a length bytes.
 *  char * src: Pointer starting byte
 *  int length: Number of bytes to zero
 *  char (return): Success or Failure of
 *                 operation
 *************************************/
char memzero(char * src, int length);

#endif /* __MEMORY_H__ */
```

Header Files [S1.2.4.7b]

- Header files are the **interface**
- Anything you want to give access to, put in header file
- Make Informative function comments in header File
  - Function Description
  - **Inputs**: type and description
  - **Return**: type and description

```
#ifndef __MEMORY_H__
#define __MEMORY_H__
/*************************************
 *  memzero() - Takes a pointer to a
 *       location in memory and sets
 *       the contents to zero for
 *       a length bytes.
 *  char * src: Pointer starting byte
 *  int length: Number of bytes to zero
 *  char (return): Success or Failure of
 *                 operation
 *************************************/
char memzero(char * src, int length);

#endif /* __MEMORY_H__ */
```

Including Precompiled Libraries [S1.2.4.8a]

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include “thirdparty.h”

int main(){

  /* Some Code here */

  return 0;
}
```

Standard libraries that come precompiled with your compiler toolchain

Potential Third Party library

Including Precompiled Libraries [S1.2.4.8b]

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include “thirdparty.h”

int main(){

  /* Some Code here */

  return 0;
}
```

Questions you Should Ask:

If library is precompiled:

- **Is it compiled for my architecture?**
- **Was this designed to be optimized for my architecture?**

If you have full library source code:

- **What software features does this use?**
- **What other code does this include?**

String and Stdio Libraries [S1.2.4.9a]

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

int main(){

  /* Some Code here */
  memmove(dest_ptr, src_ptr, length);
  printf("Done Moving %d Bytes!", length);
  /* Other Code here */

  return 0;
}
```

String and Stdio Libraries [S1.2.4.9b]

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

These libraries are likely already optimized… but only for the Instruction set Architectures (ISA) but not for the platform!

```
int main(){

  /* Some Code here */
  memmove(dest_ptr, src_ptr, length);
  printf("Done Moving %d Bytes!", length);
  /* Other Code here */

  return 0;
}
```

String and Stdio Libraries [S1.2.4.9c]

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

int main(){

  /* Some Code here */
  memmove(dest_ptr, src_ptr, length);
  printf("Done Moving %d Bytes!", length);
  /* Other Code here */

  return 0;
}
```

These libraries are likely already optimized… but only for the Instruction set Architectures (ISA) but not for the platform!

Is there hardware offloading that can increase performance?

String and Stdio Libraries [S1.2.4.9d]

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

int main(){

  /* Some Code here */
  memmove(dest_ptr, src_ptr, length);
  printf("Done Moving %d Bytes!", length);
  /* Other Code here */

  return 0;
}
```

These libraries are likely already optimized… but only for the Instruction set Architectures (ISA) but not for the platform!

Is there hardware offloading that can increase performance?

What internal memory requirements does this require?

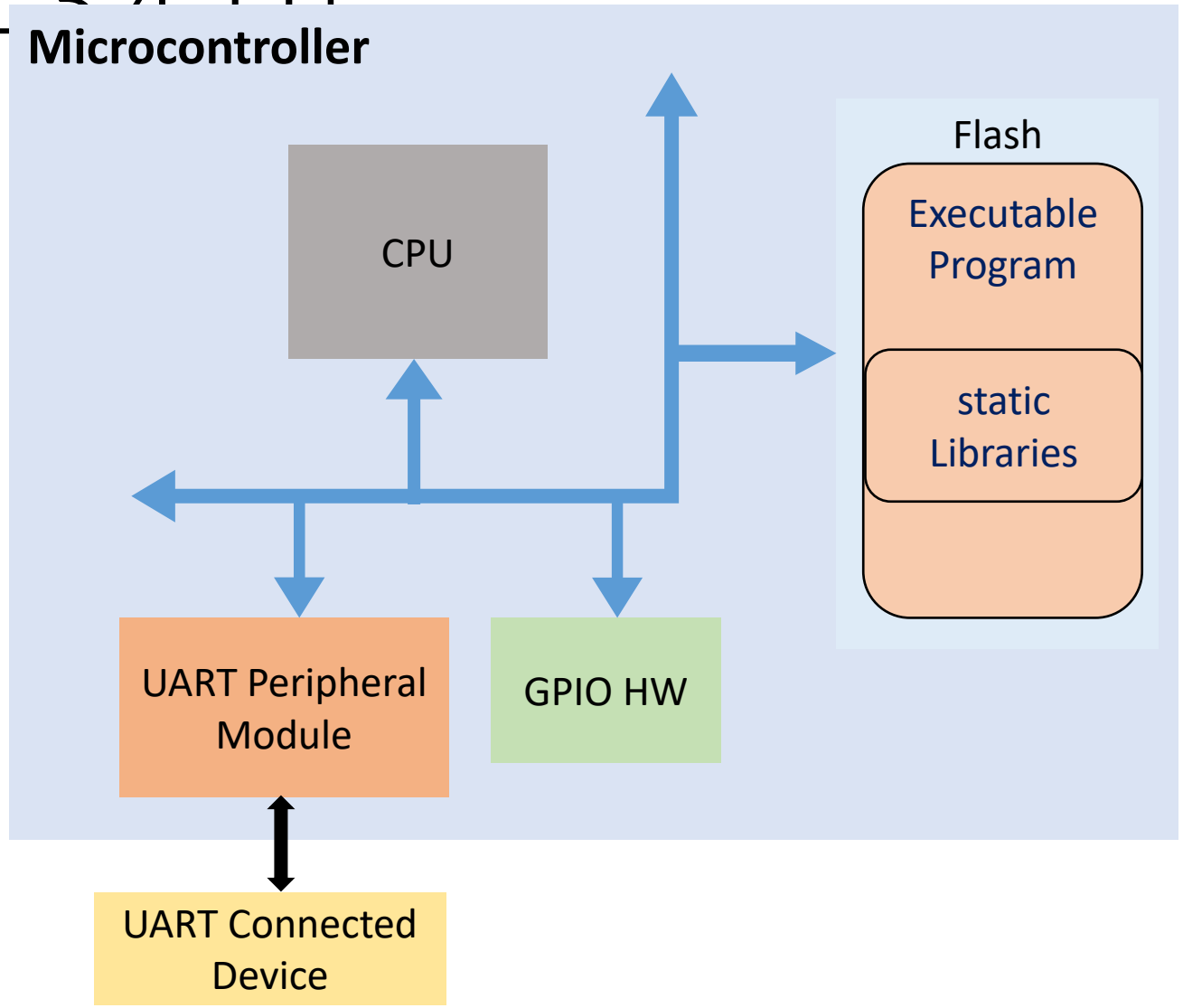
Compiled Libraries [S1.2.4.10]

- **Static Libraries:** Directly linked into your output executable
  - Installed with the program image as part of the executable
  - Create using archiver

- **Shared libraries:** Linked dynamically at runtime with your executable
  - Pre-installed onto target
  - Used for applications with an operating systems
  - Create with **shared** flag

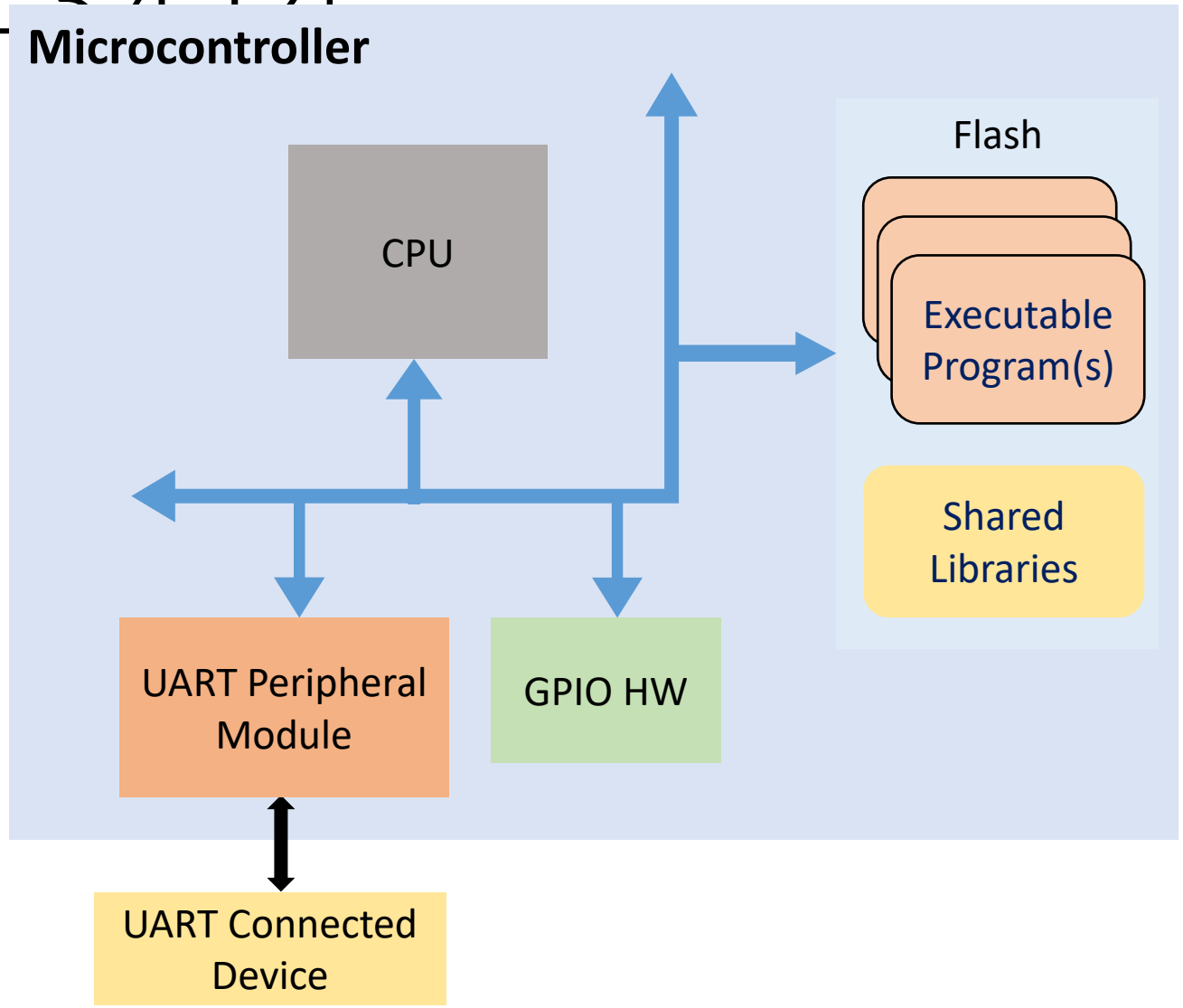
Compiled Libraries [S1.3.4.11]

- Picture of a static library and the installed executable
- The static library is built into the executable image



Compiled Libraries [S1.3.4.12]

- Picture of a dynamic library and the installed executable
- Your executable is placed in separate regions then the libraries.

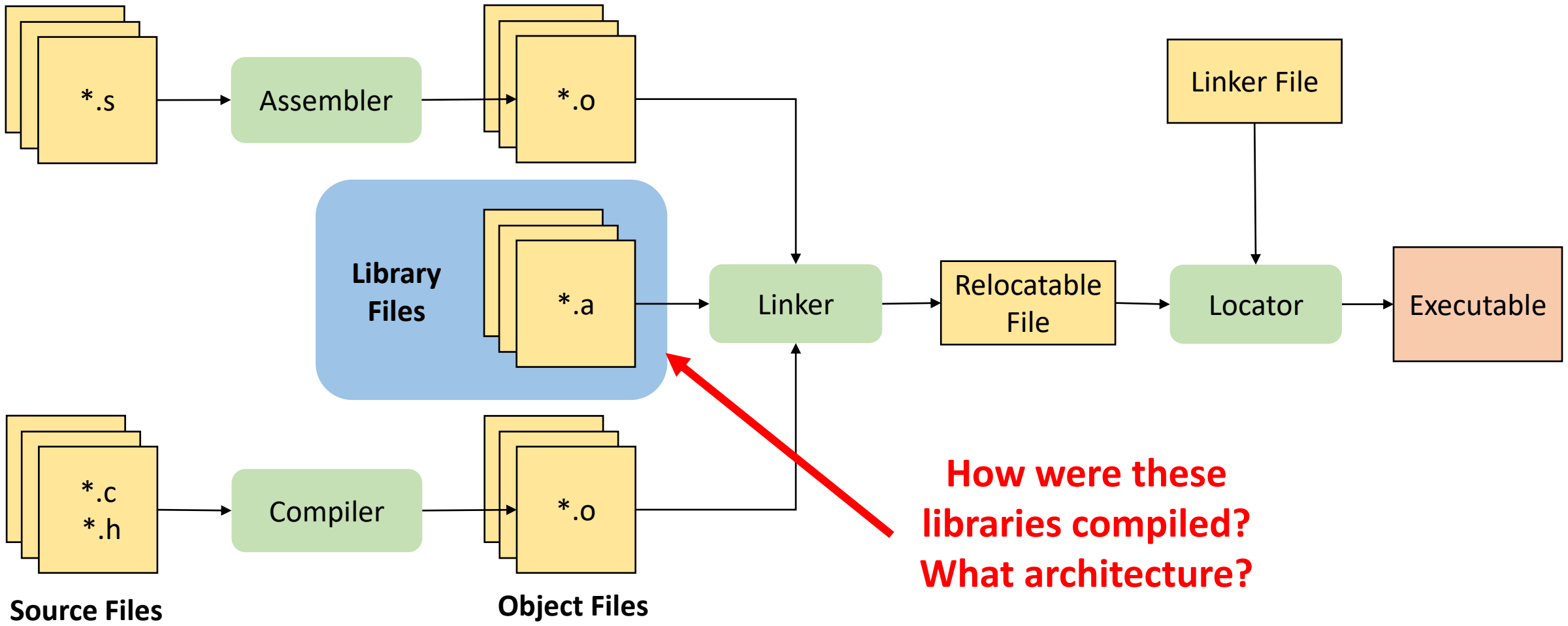


Header Files [S1.2.4.7a]

- Header files are the **interface**
- Anything you want to give access to, put in header file
- Make Informative function comments in header File
  - Function Description
  - **Inputs**: type and description
  - **Return**: type and description

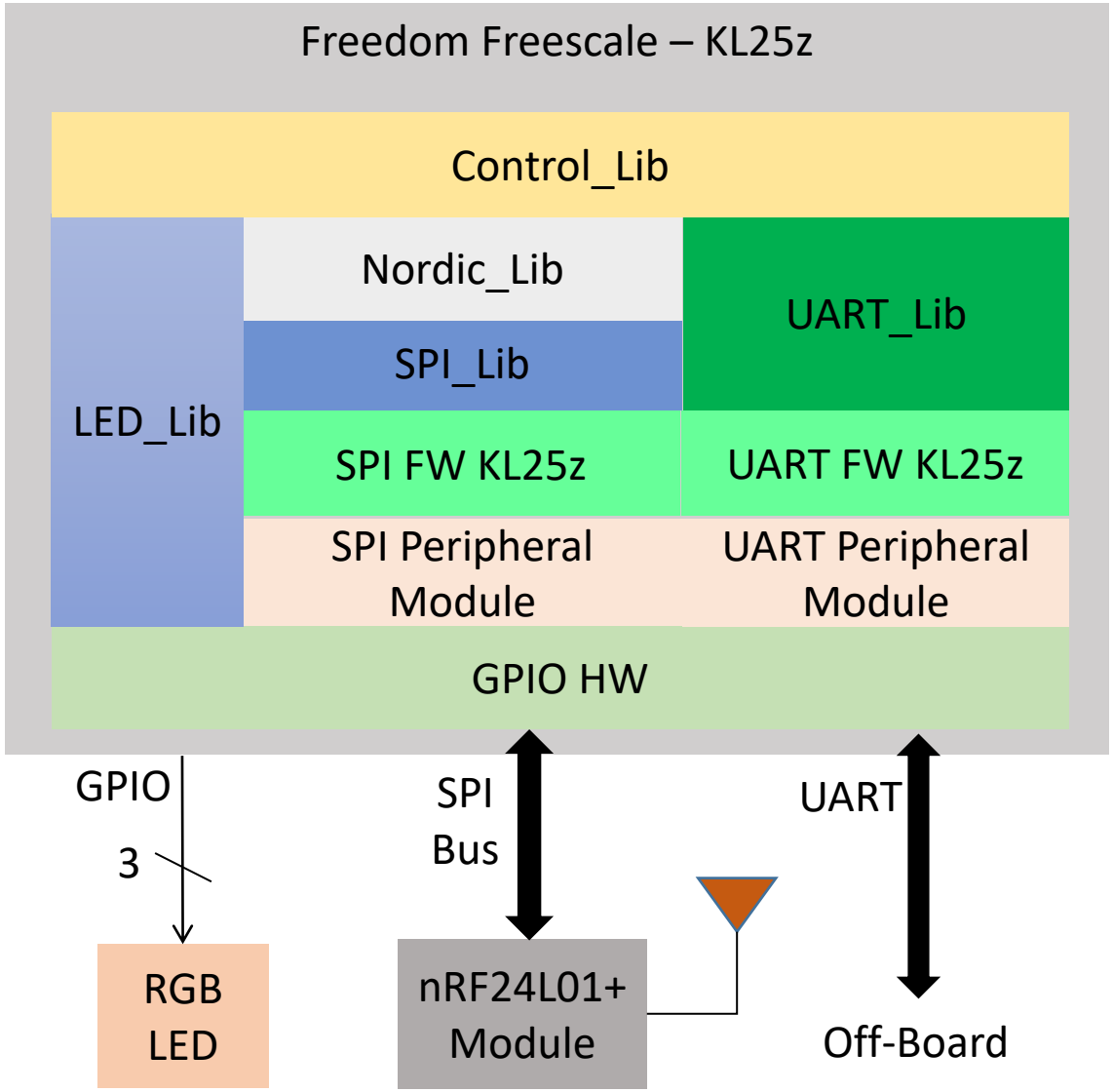
```
#ifndef __MEMORY_H__
#define __MEMORY_H__
/*************************************
 *  memzero() - Takes a pointer to a
 *       location in memory and sets
 *       the contents to zero for
 *       a length bytes.
 *  char * src: Pointer starting byte
 *  int length: Number of bytes to zero
 *  char (return): Success or Failure of
 *                 operation
 *************************************/
char memzero(char * src, int length);

#endif /* __MEMORY_H__ */
```

Typical Build Process [S1.2.2.x]



Module Design [S1.2.4.13]

- Where do the logical boundaries exist?
- What have architecture dependencies?
- What have platform dependencies?



Portable Header Interface [S1.2.4.14a]

main.c

```
#include “platform.h”

int main(void) {

 platform_initialize();

 /* More code here */

 return 0;
}
```

platform.h

```
#ifndef __PLATFORM_H__
#define __PLATFORM_H__


#ifdef ( KL25_PLATFORM ) && ( ! MSP_PlATFROM )
#include “kl25_platform.h”
#elif ( MSP_PlATFROM ) && ( ! KL25_PLATFORM )
#include “msp_platform.h”
#else
#error “Please specify one platform target”
#endif

#endif /* __PLATFORM_H__ */
```

Portable Header Interface [S1.2.4.14b]

platform.h

```
#ifndef __PLATFORM_H__
#define __PLATFORM_H__


#ifdef ( KL25_PLATFORM ) && ( ! MSP_PlATFROM )
#include “kl25_platform.h”
#elif ( MSP_PlATFROM ) && ( ! KL25_PLATFORM )
#include “msp_platform.h”
#else
#error “Please specify one platform target”
#endif

#endif /* __PLATFORM_H__ */
```

msp_platform.h

```
#ifndef __MSP_PLATFORM_H__
#define __MSP_PLATFORM_H__

initialize();

#endif /* __MSP_PLATFORM_H__ */
```

kl25_platform.h

```
#ifndef __KL25_PLATFORM_H__
#define __KL25_PLATFORM_H__

initialize();

#endif /* __KL25_PLATFORM_H__ */
```

Portable Header Interface [S1.2.4.xc]

platform.h

```
#ifndef __PLATFORM_H__
#define __PLATFORM_H__


#ifdef ( KL25_PLATFORM ) && ( ! MSP_PlATFROM )
#include “kl25_platform.h”
#elif ( MSP_PlATFROM ) && ( ! KL25_PLATFORM )
#include “msp_platform.h”
#else
#error “Please specify one platform target”
#endif

#endif /* __PLATFORM_H__ */
```

msp_platform.h

```
#ifndef __MSP_PLATFORM_H__
#define __MSP_PLATFORM_H__

initialize();

#endif /* __MSP_PLATFORM_H__ */
```

kl25_platform.h

```
#ifndef __KL25_PLATFORM_H__
#define __KL25_PLATFORM_H__

initialize();

#endif /* __KL25_PLATFORM_H__ */
```

```
$ gcc -DKL25_PLATFORM main.c …
```