ESS Exercise 1
Example Solution

Dipl.-Ing. Florian Wilde

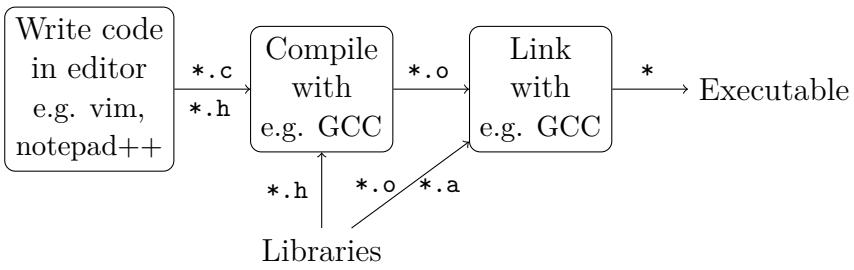
October 21, 2021

I can explain it to you
but I cannot understand it for you.

T-Shirt print

1 Hello World

1.1 Sketch what steps are necessary from a blank command line until a running program



1.2 Perform the steps you just sketched

1.2.1 Write the code

Use an editor of your choice to create a file named `helloworld.c` with the following content:

```
1 #include <stdio.h>
2
3 int main(int argc, char* argv[]) {
```

```
4   puts("Hello␣World");
5   return 0;
6 }
```

Note: The arguments to `main()` are optional. For bare metal (i.e. without OS) embedded systems in particular, there is no calling function for your `main()` who could pass any arguments. The return type and statement, though, are required in order to obey the C standard.

1.2.2 Compile and link your code

Enter the following on the command line:

```
1 gcc -o helloworld helloworld.c
```

Of course this is a minimal working example (MWE). If you look into the Makefile supplied with the Linux toolchain, you see how many options one can respectively has to set depending on the complexity of the project.

1.2.3 Run your program

Enter the following on the command line:

```
1 ./helloworld
```

2 Build Steps For Embedded Systems

2.1 Build the example project using the supplied makefile

Change to the folder called `example_project`, then a simple `make` would suffice to build the project. Because we want to retrace the build steps from the command line output, we first enter `make clean` to remove any output products of the build process. So the whole project is rebuild without reusing previously compiled and still up-to-date parts.

2.2 Retrace the build steps

While your project builds, something similar to this should appear in your console:

```
1 mkdir -p build
2 cp -pu example_project.ld build/example_project.ld | true
3 cp -n --preserve=timestamps /opt/XMClib/XMC_Peripheral_Library_v2.1.8/»
    «CMSIS/Infineon/XMC4500_series/Source/GCC/XMC4500x1024.ld build/»
    «example_project.ld
4 echo "building build/example_project.o"
5 building build/example_project.o
6 arm-none-eabi-gcc -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/»
    «Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/Infineon»
    «/XMC4500_series/Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2»
    «.1.8/XMCLib/inc/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib»
```

```
    « -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Common -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Device -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Common -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core/XMC4000 -»
    «DXMC4500_F100x1024 -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp»
    «-d16 -mthumb -g3 -gdwarf-2 -c -Wa,-adhlns="build/example_project.o.»
    «lst" -std=gnu99 -O0  -Wall -ffunction-sections -o build/»
    «example_project.o example_project.c
7 echo "building lib_build/xmc_gpio.o"
8 building lib_build/xmc_gpio.o
9 arm-none-eabi-gcc -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/»
    «Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/Infineon»
    «/XMC4500_series/Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2»
    «.1.8/XMCLib/inc/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib»
    « -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Common -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Device -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Common -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core/XMC4000 -»
    «DXMC4500_F100x1024 -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp»
    «-d16 -mthumb -g3 -gdwarf-2 -c -Wa,-adhlns="lib_build/xmc_gpio.o.lst»
    «" -std=gnu99 -O0  -Wall -ffunction-sections -o lib_build/xmc_gpio.o»
    « /opt/XMClib/XMC_Peripheral_Library_v2.1.8/XMCLib/src/xmc_gpio.c
10 echo "building lib_build/xmc4_gpio.o"
11 building lib_build/xmc4_gpio.o
12 arm-none-eabi-gcc -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/»
    «Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/Infineon»
    «/XMC4500_series/Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2»
    «.1.8/XMCLib/inc/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib»
    « -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Common -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Device -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Common -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core/XMC4000 -»
    «DXMC4500_F100x1024 -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp»
    «-d16 -mthumb -g3 -gdwarf-2 -c -Wa,-adhlns="lib_build/xmc4_gpio.o.»
    «lst" -std=gnu99 -O0  -Wall -ffunction-sections -o lib_build/»
    «xmc4_gpio.o /opt/XMClib/XMC_Peripheral_Library_v2.1.8/XMCLib/src/»
    «xmc4_gpio.c
13 echo "building lib_build/system_XMC4500.o"
14 building lib_build/system_XMC4500.o
15 arm-none-eabi-gcc -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/»
    «Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/Infineon»
    «/XMC4500_series/Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2»
    «.1.8/XMCLib/inc/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib»
    « -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Common -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Device -I/opt/»
```

```
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Common -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core/XMC4000 -»
    «DXMC4500_F100x1024 -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp»
    «-d16 -mthumb -g3 -gdwarf-2 -c -Wa,-adhlns="lib_build/system_XMC4500»
    «.o.lst" -std=gnu99 -O0  -Wall -ffunction-sections -o lib_build/»
    «system_XMC4500.o /opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/»
    «Infineon/XMC4500_series/Source/system_XMC4500.c
16 echo "building lib_build/syscalls.o"
17 building lib_build/syscalls.o
18 arm-none-eabi-gcc -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/»
    «Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/Infineon»
    «/XMC4500_series/Include/ -I/opt/XMClib/XMC_Peripheral_Library_v2»
    «.1.8/XMCLib/inc/ -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib»
    « -I/opt/XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Common -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Class/Device -I/opt/»
    «XMClib/XMC_Peripheral_Library_v2.1.8/USBLib/Common -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core -I/opt/XMClib/»
    «XMC_Peripheral_Library_v2.1.8/USBLib/Core/XMC4000 -»
    «DXMC4500_F100x1024 -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp»
    «-d16 -mthumb -g3 -gdwarf-2 -c -Wa,-adhlns="lib_build/syscalls.o.lst»
    «" -std=gnu99 -O0  -Wall -ffunction-sections -o lib_build/syscalls.o»
    « /opt/XMClib/XMC_Peripheral_Library_v2.1.8/Newlib/syscalls.c
19 arm-none-eabi-gcc --specs=nosys.specs -Tbuild/example_project.ld -»
    «nostartfiles -L/opt/XMClib/XMC_Peripheral_Library_v2.1.8/CMSIS/Lib/»
    «GCC/ -Wl,-Map,"build/main.elf.map" -mcpu=cortex-m4 -mthumb -g3 -»
    «gdwarf-2 -o build/main.elf build/example_project.o lib_build/»
    «xmc_gpio.o lib_build/xmc4_gpio.o lib_build/startup_XMC4500.o »
    «lib_build/system_XMC4500.o lib_build/syscalls.o
20 arm-none-eabi-size build/main.elf
21    text    data     bss     dec     hex filename
22    2624    1072    2076    5772    168c build/main.elf
23 arm-none-eabi-objdump -h -S build/main.elf > build/main.lst
24 rm lib_build/xmc_gpio.o lib_build/xmc4_gpio.o lib_build/system_XMC4500»
    «.o lib_build/syscalls.o build/example_project.o
```

Lines 4-18 are from compiling the start-up files, libraries and `main.c`. So entirely the step 'cross-compiler'. Notice the many `-I` options, which specify directories to search for header files to be included, among them the device header `XMC4500.h` and the header files for the GPIO driver. The `mkdir` commands create subfolders for output products or library code, and are just for proper structuring the file locations.

Line 19 links all the `*.o` files together into `bin/main.elf`. You can spot the difference in output product and input files – or notice that it is the only invocation of `gcc` which is given the linker description file via `-Tbuild/example_project.ld`.

Lines 20-22 are created by a call to `size`, which shows how large certain sections are. It is just for information and not necessary for the build process, but helpful to check whether your code actually fits into the chosen uC.

Line 23 shows the creation of the *.lst file

Line 24 shows how intermediate files are removed, this step is optional and only included to save some disk space.

2.3 Look into the files and decide which files are human readable

***.o** are compiled but unlinked versions of your source files, not human readable

***.elf** are compiled and linked programs, ready to execute on the architecture they are build for – as long as an OS loads them. If you enabled it in the compiler options, this file also contains debug information, so the debugger can show you which line of your source file the processor currently executes. ELF stands for "executable and linkable format", if you are interested, have a look in the Wikipedia entry:

`https://en.wikipedia.org/wiki/Executable_and_Linkable_Format`

***.hex** is the pure machine code together with address information to which address the machine instructions should be copied by the programmer, but no architecture or other meta data as in the `*.elf` file is contained. It is binary encoded in hex digits, so technically human readable, but you probably cannot recognize the assembler instructions in this file. This file format also has a Wikipedia entry:

`https://en.wikipedia.org/wiki/Intel_HEX`

***.lst** is a human readable copy of parts of the `*.elf` file. What to put in here is set by options to `objdump`, but usually it's

- Section headers (where `.data`, `.bss`, etc. are located and how large they are)
- Disassembly of the `.text` section interleaved with the C instructions it was compiled from. Be aware that optimization (`-O` options) might shift code around, merge it together with other code or even remove it if it does not affect the result. So as long as you switch them off, the compiled assembler code might look terribly inefficient but as soon as you enable optimization, you can no longer trace what the compiler made out of your C instructions. It takes some experience to deal with that.