

*Learn in depth*

**Assignment-2**

**Lesson-2**

*Name : Ayat Mohamed*

## ■ C-codes

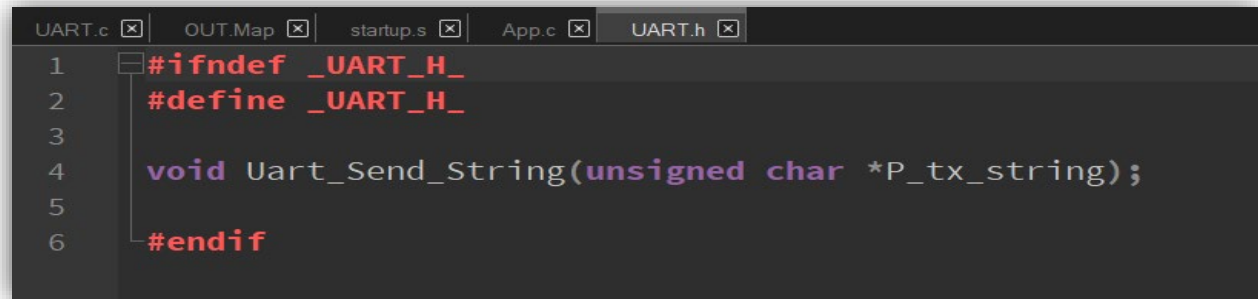
### ■ App.c

```
UART.c x OUT.Map x startup.s x App.c x
1  #include "UART.h"
2
3  unsigned char str_buffer[100] = "Learn_in_depth : Ayat mohamed";
4  unsigned char const str_buffer2[100] = "Learn_in_depth : Ayat mohamed";
5  void main ()
6  {
7      Uart_Send_String(str_buffer);
8  }
```

### ■ UART.c

```
UART.c x OUT.Map x startup.s x App.c x
1  #include "UART.h"
2
3  #define UART0DR *((volatile unsigned int*)((unsigned int*)0x101f1000))
4
5  void Uart_Send_String(unsigned char *P_tx_string)
6  {
7      while(*P_tx_string != '\0')
8      {
9          UART0DR = (unsigned int)(*P_tx_string);
10         P_tx_string++;
11     }
12 }
```

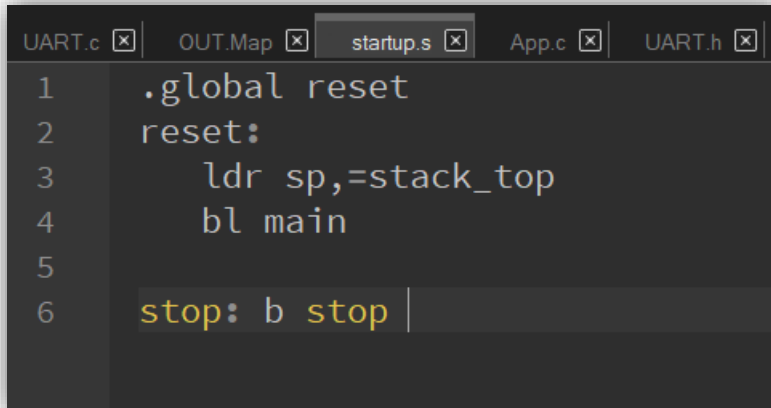
## ■ *UART.h*



A screenshot of a code editor with five tabs: UART.c, OUT.Map, startup.s, App.c, and UART.h. The UART.h tab is active, showing the following code:

```
1  #ifndef _UART_H_
2  #define _UART_H_
3
4  void Uart_Send_String(unsigned char *P_tx_string);
5
6  #endif
```

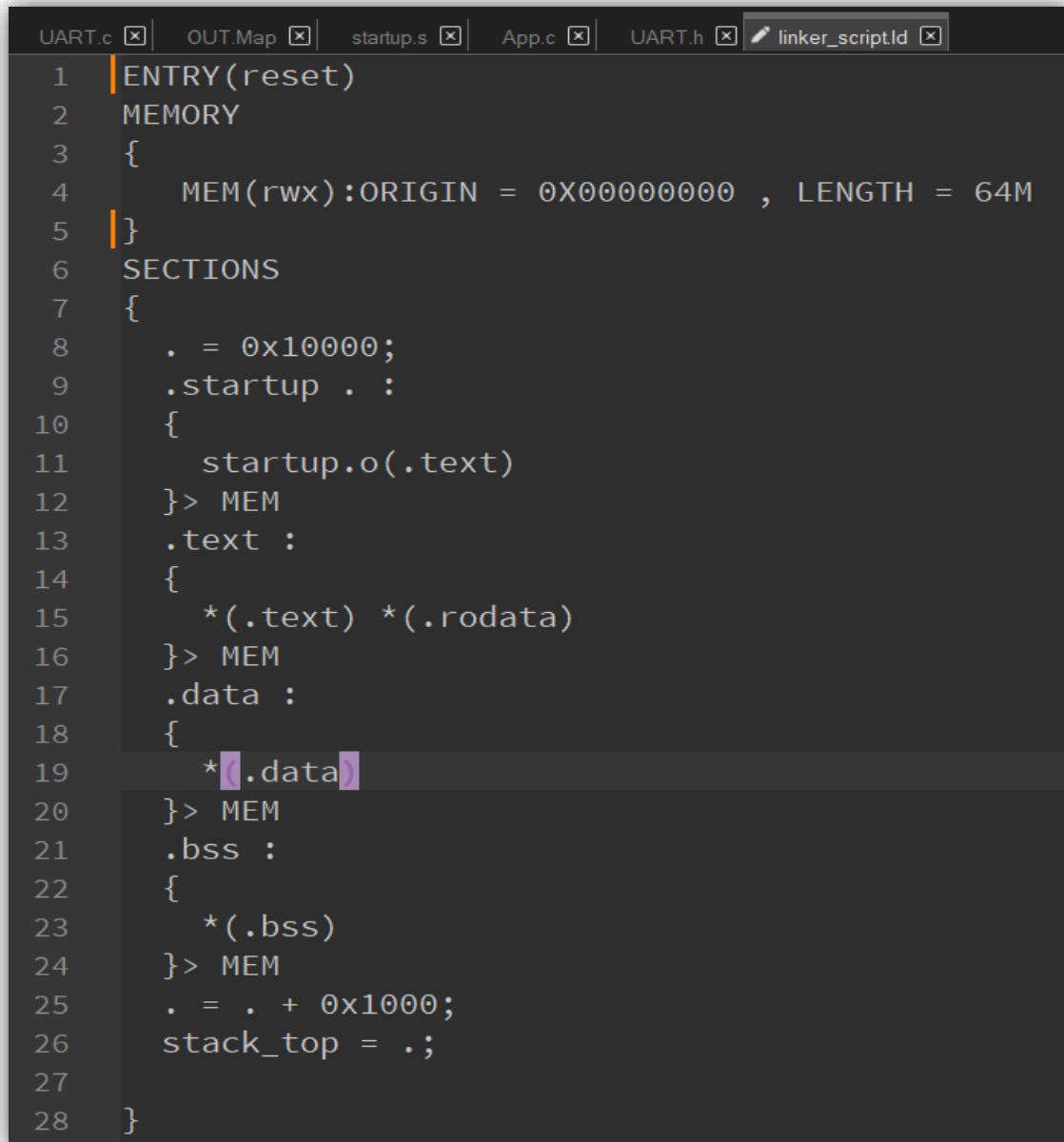
## ■ *Startup.s*



A screenshot of a code editor with five tabs: UART.c, OUT.Map, startup.s, App.c, and UART.h. The startup.s tab is active, showing the following assembly code:

```
1  .global reset
2  reset:
3      ldr sp,=stack_top
4      bl main
5
6  stop: b stop
```

## ■ *Linker\_script.ld*



```
1 ENTRY(reset)
2 MEMORY
3 {
4     MEM(rwx):ORIGIN = 0X00000000 , LENGTH = 64M
5 }
6 SECTIONS
7 {
8     . = 0x10000;
9     .startup . :
10    {
11        startup.o(.text)
12    }> MEM
13    .text :
14    {
15        *(.text) *(.rodata)
16    }> MEM
17    .data :
18    {
19        *(.data)
20    }> MEM
21    .bss :
22    {
23        *(.bss)
24    }> MEM
25    . = . + 0x1000;
26    stack_top = .;
27
28 }
```

- *get obj\_file form App.c UART.c included UART.h*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ export PATH=../../../../../../units/UNIT_3/LESSON2/ARM/bin/:$PATH

Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s:5: Warning: end of file not at end of a line; newline inserted

Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s UART.c -o UART.o

Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s App.c -o App.o
```

- *Linking all objects*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-ld.exe -T linker_script.ld App.o UART.o startup.o -o learn-in-de
pth.elf
```

- *Create a binary file(objcopy)*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
```

## ➤ Section for each obj\_file

### ■ App.o

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-objdump.exe -h App.o

App.o:      file format elf32-littlearm

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          0000001c  00000000  00000000  00000034  2**2
                CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data          00000064  00000000  00000000  00000050  2**2
                CONTENTS, ALLOC, LOAD, DATA
  2 .bss           00000000  00000000  00000000  000000b4  2**0
                ALLOC
  3 .rodata        00000064  00000000  00000000  000000b4  2**2
                CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment       0000007f  00000000  00000000  00000118  2**0
                CONTENTS, READONLY
  5 .ARM.attributes 00000032  00000000  00000000  00000197  2**0
                CONTENTS, READONLY
```

### ■ UART.o

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-objdump.exe -h UART.o

UART.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          00000054  00000000  00000000  00000034  2**2
                CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data          00000000  00000000  00000000  00000088  2**0
                CONTENTS, ALLOC, LOAD, DATA
  2 .bss           00000000  00000000  00000000  00000088  2**0
                ALLOC
  3 .comment       0000007f  00000000  00000000  00000088  2**0
                CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  00000107  2**0
                CONTENTS, READONLY
```

## ■ *Startup.o*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:      file format elf32-littlearm

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
 0 .text          00000010  00000000  00000000  00000034  2**2
   CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data          00000000  00000000  00000000  00000044  2**0
   CONTENTS, ALLOC, LOAD, DATA
 2 .bss           00000000  00000000  00000000  00000044  2**0
   ALLOC
 3 .ARM.attributes 00000022  00000000  00000000  00000044  2**0
   CONTENTS, READONLY
```

## ■ *Learn-in-depth.elf*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-objdump.exe -h learn-in-depth.elf

learn-in-depth.elf:      file format elf32-littlearm

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
 0 .startup       00000010  00010000  00010000  00010000  2**2
   CONTENTS, ALLOC, LOAD, READONLY, CODE
 1 .text          000000d4  00010010  00010010  00010010  2**2
   CONTENTS, ALLOC, LOAD, READONLY, CODE
 2 .data          00000064  000100e4  000100e4  000100e4  2**2
   CONTENTS, ALLOC, LOAD, DATA
 3 .ARM.attributes 0000002e  00000000  00000000  00010148  2**0
   CONTENTS, READONLY
 4 .comment       0000007e  00000000  00000000  00010176  2**0
   CONTENTS, READONLY
```

➤ *show symbol tables for :*

■ *App.o*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-nm.exe App.o
00000000 T main
00000000 D str_buffer
00000000 R str_buffer2
          U Uart_Send_String
```

■ *UART.o*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-nm.exe UART.o
00000000 T Uart_Send_String
```

■ *Startup.o*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-nm.exe startup.o
          U main
00000000 T reset
          U stack_top
00000008 t stop
```

■ *Learn-in-depth.elf*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Lesson_2 (master)
$ arm-none-eabi-nm.exe learn-in-depth.elf
00010010 T main
00010000 T reset
00011148 D stack_top
00010008 t stop
000100e4 D str_buffer
00010080 T str_buffer2
0001002c T Uart_Send_String
```



➤ *burn binary file on board using  
qemu simulator*

```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d  
iploma/C_programming/Unit_3/Lesson_2 (master)  
$ E:/KEROLOS_Diploma/units/UNIT_3/LESSON2/qemu/qemu-system-arm -M versatilepb -m  
128M -nographic -kernel learn_in_depth.elf  
Learn_in_depth : Ayat mohamed|
```