

Learn in depth
Embedded C - Lesson 2
Lab_1 - UART
Abdullah Mohamed Kortam

Main.c Code

```
hh.c  d;d.c  uart.c  uart.h  main.c ×
1  #include <stdio.h>
2  #include "uart.h"
3
4  unsigned char string[100]="Learn-in-depth:<Abdullah>";
5  void main()
6  {
7      UART_SEND_STRING(string);
8  }
9
```

Main.o objdump

```
mkm@DESKTOP-580JK14 MINGW64 ~/Desktop/Lab
$ arm-none-eabi-objdump.exe -h main.o

main.o:      file format elf32-littlearm

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          00000018  00000000  00000000  00000034  2**2
    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data           00000064  00000000  00000000  0000004c  2**2
    CONTENTS, ALLOC, LOAD, DATA
  2 .bss            00000000  00000000  00000000  000000b0  2**0
    ALLOC
  3 .comment        00000012  00000000  00000000  000000b0  2**0
    CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  000000c2  2**0
    CONTENTS, READONLY
```

UART.c

```
hh.c d;d.c uart.c x uart.h main.c
1 #include "uart.h"
2 #define UART0DR *((volatile unsigned int *)((unsigned int*)0x101f1000))
3
4 void UART_SEND_STRING(unsigned char* p_string)
5 {
6     while(p_string != '\0')
7     {
8         UART0DR = (unsigned int)(*p_string);
9         p_string ++;
10    }
11 }
```

UART.h

```
hh.c d;d.c uart.c uart.h x main.c
1 #ifndef _UART_H_
2 #define _UART_H_
3 void UART_SEND_STRING(unsigned char* p_string);
4 #endif // _UART_H_
```

UART.o objdump

```
mkm@DESKTOP-580JK14 MINGW64 ~/Desktop/Lab
$ 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          0000004c  00000000  00000000  00000034  2**2
    CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data          00000000  00000000  00000000  00000080  2**0
    CONTENTS, ALLOC, LOAD, DATA
  2 .bss           00000000  00000000  00000000  00000080  2**0
    ALLOC
  3 .comment       00000012  00000000  00000000  00000080  2**0
    CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  00000092  2**0
    CONTENTS, READONLY
```

Linker_script.ld

```
ENTRY(reset)
```

```
MEMORY
```

```
{  
    Mem (rwx) : ORIGIN =0x00000000 , LENGTH =64M  
}
```

```
SECTIONS
```

```
{  
    . = 0x10000;  
    .startup :  
    {  
        startup.o(.text)  
    }>Mem  
  
    .text :  
    {  
        *(.text) *(.rodata)  
    }>Mem  
  
    .data :  
    {  
        *(.data)  
    }>Mem  
  
    .bss :  
    {  
        *(.bss) *(COMMON)  
    }>Mem  
  
    . = . + 0x1000 ;  
    stack_top = . ;  
}
```

Abdullah.elf objdump

```
mkm@DESKTOP-580JK14 MINGW64 ~/Desktop/Lab
$ arm-none-eabi-objdump.exe -h abdullah.elf

abdullah.elf:      file format elf32-littlearm

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .startup      00000010  00000000     00000000     00008000  2**2
CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         00000064  00000010     00000010     00008010  2**2
CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  00000074     00000074     00008074  2**2
CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000     00000000     000080d8  2**0
CONTENTS, READONLY
  4 .comment      00000011  00000000     00000000     00008106  2**0
CONTENTS, READONLY
```

Qemu run

```
mkm@DESKTOP-580JK14 MINGW64 ~/Desktop/1
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel abdullah.bin
Learn-in-depth: <Abdullah>
```

Memory configuration

Memory Configuration

Name	Origin	Length	Attributes
Mem	0x00000000	0x04000000	xrw
default	0x00000000	0xffffffff	

Linker script and memory map

```

                                0x00010000                . = 0x10000

.startup                      0x00000000                0x10
startup.o(.text)
.text                         0x00000000                0x10 startup.o
                                0x00000000                reset

.text                         0x00000010                0x64
*(.text)
.text                         0x00000010                0x18 main.o
                                0x00000010                main
.text                         0x00000028                0x4c uart.o
                                0x00000028                UART_SEND_STRING
*(.rodata)

.glue_7                      0x00000074                0x0
.glue_7                      0x00000000                0x0 linker stubs

.glue_7t                     0x00000074                0x0
.glue_7t                     0x00000000                0x0 linker stubs

.vfp11_veneer                0x00000074                0x0
.vfp11_veneer                0x00000000                0x0 linker stubs

.v4_bx                       0x00000074                0x0
.v4_bx                       0x00000000                0x0 linker stubs

.iplt                        0x00000074                0x0
.iplt                        0x00000000                0x0 startup.o

.rel.dyn                     0x00000074                0x0
.rel.iplt                    0x00000000                0x0 startup.o

.data                        0x00000074                0x64
*(.data)
.data                        0x00000074                0x0 startup.o
.data                        0x00000074                0x64 main.o
                                0x00000074                string
.data                        0x000000d8                0x0 uart.o

.igot.plt                    0x000000d8                0x0
.igot.plt                    0x00000000                0x0 startup.o

.bss                         0x000000d8                0x0
*(.bss)
.bss                         0x000000d8                0x0 startup.o
.bss                         0x000000d8                0x0 main.o
.bss                         0x000000d8                0x0 uart.o
*(COMMON)

                                0x000010d8                . = (. + 0x1000)
                                0x000010d8                stack_top = .

LOAD main.o
LOAD uart.o

```