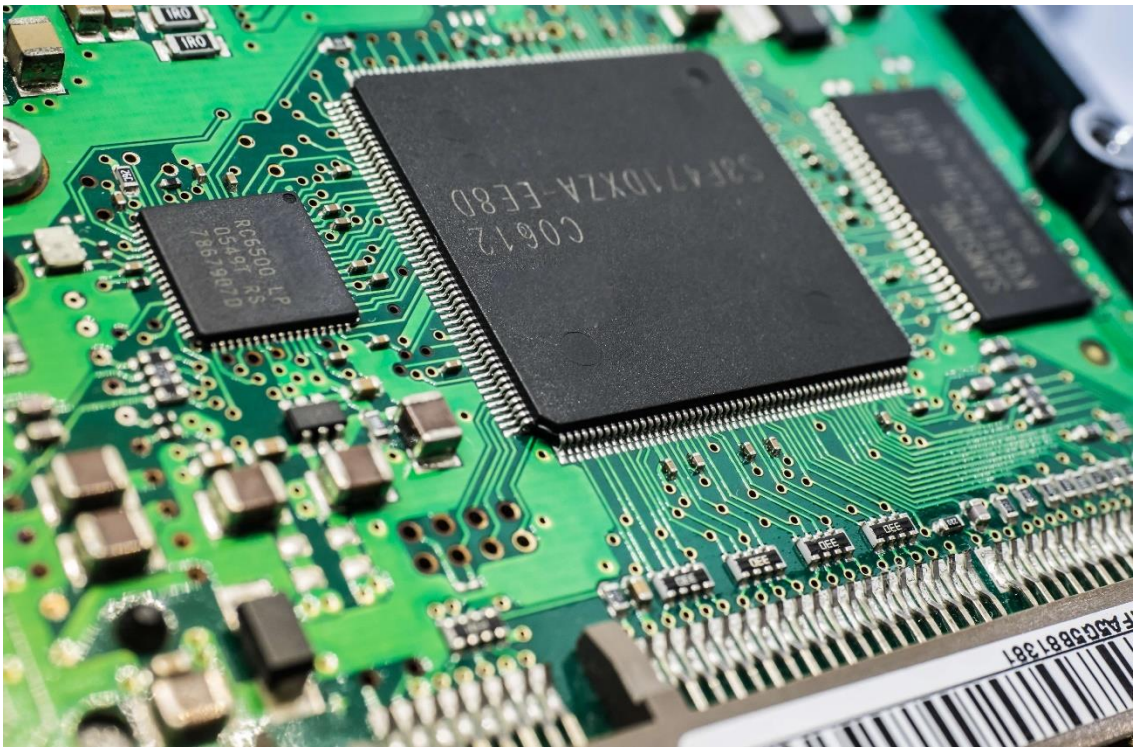
Mastering Embedded Systems
Learn in depth

Unit 3 Lesson 2

Lab 1



Submitted by: Omar Shawky Mohamed

app.c code:

This code takes a global string called buffer and send it through the uart through the UART_Send_String() API.

```c
#include "uart.h"

unsigned char Buffer[100] = "Omar Shawky";

void main(void)
{
    UART_Send_String(Buffer);
}
```

uart.c code:

This code contains the address of the uart data register and the definition of the UART_Send_String() function that sends the characters of the string through the uart data register till it reaches the null character.

```c
#include "uart.h"
//Registers to use
#define UART0DR *( ( volatile unsigned int* const ) ((unsigned int*)0x101f1000) )

void UART_Send_String(unsigned char * P_tx_string)
{
    while(*P_tx_string != '\0')
    {
        UART0DR = (unsigned int) *P_tx_string;
        P_tx_string++;
    }
}
```

uart.h code:

This code contains only a prototype for the UART_Send_String() function.

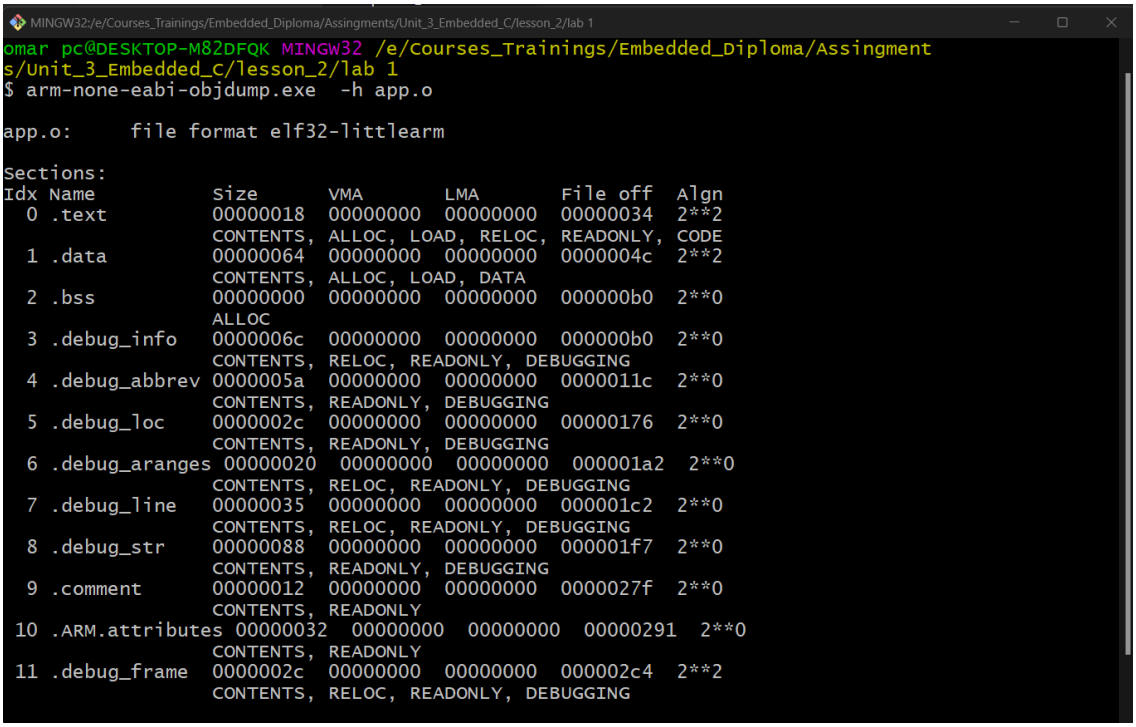
```c
//Protection from multiple declarations
#ifndef UART_H_
#define UART_H_

// Prototypes
void UART_Send_String(unsigned char * P_tx_string) ;

#endif
```

Compiling and assembling the .c files with debugging info:

Displaying section headers for app.o and uart.o with debugging info:



```
MINGW32:/e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1

omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedde
d_C/lesson_2/lab 1
$ 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         00000050  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000084  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000084  2**0
                  ALLOC
  3 .debug_info   0000005c  00000000  00000000  00000084  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 00000051  00000000  00000000  000000e0  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000002c  00000000  00000000  00000131  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  0000015d  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000003d  00000000  00000000  0000017d  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000091  00000000  00000000  000001ba  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000024b  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000032  00000000  00000000  0000025d  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000028  00000000  00000000  00000290  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

we note two things. First is that both the VMA and LMA of all sections are set to zero as these sections hasn't been relocated yet. The relocation happens after linking through the linker script. Second the is .rodata section since there is no constant global variables in all .c files.

Displaying section headers for app.o and uart.o without debugging info:

```
omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedde
d_C/lesson_2/lab 1
$ 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         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
```

```
MINGW32:/e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1

omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedde
d_C/lesson_2/lab 1
$ 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         00000050  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000084  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000084  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000084  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  00000096  2**0
                  CONTENTS, READONLY
```

Disassembly of app.o and uart.o:

```
omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ arm-none-eabi-objdump.exe  -D app.o

app.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <main>:
   0:   e92d4800        push    {fp, lr}
   4:   e28db004        add     fp, sp, #4
   8:   e59f0004        ldr     r0, [pc, #4]    ; 14 <main+0x14>
   c:   ebfffffe        bl      0 <UART_Send_String>
  10:   e8bd8800        pop     {fp, pc}
  14:   00000000        andeq   r0, r0, r0

Disassembly of section .data:

00000000 <Buffer>:
   0:   72616d4f        rsbvc   r6, r1, #5056   ; 0x13c0
   4:   61685320        cmnvs   r8, r0, lsr #6
   8:   00796b77        rsbseq  r6, r9, r7, ror fp
        ...

Disassembly of section .comment:

00000000 <.comment>:
   0:   43434700        movtmi  r4, #14080      ; 0x3700
   4:   4728203a                        ; <UNDEFINED> instruction: 0x4728203a
   8:   2029554e        eorcs   r5, r9, lr, asr #10
   c:   2e372e34        mrccs   14, 1, r2, cr7, cr4, {1}
  10:   Address 0x00000010 is out of bounds.


Disassembly of section .ARM.attributes:

00000000 <.ARM.attributes>:
   0:   00003141        andeq   r3, r0, r1, asr #2
   4:   61656100        cmnvs   r5, r0, lsl #2
   8:   01006962        tsteq   r0, r2, ror #18
   c:   00000027        andeq   r0, r0, r7, lsr #32
  10:   4d524105        ldfmie  f4, [r2, #-20]  ; 0xffffffec
  14:   45363239        ldrmi   r3, [r6, #-569]!        ; 0x239
  18:   00532d4a        subseq  r2, r3, sl, asr #26
  1c:   01080506        tsteq   r8, r6, lsl #10
  20:   04120109        ldreq   r0, [r2], #-265 ; 0x109
  24:   01150114        tsteq   r5, r4, lsl r1
  28:   01180317        tsteq   r8, r7, lsl r3
  2c:   011a0119        tsteq   sl, r9, lsl r1
  30:   Address 0x00000030 is out of bounds.
```

```
omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ arm-none-eabi-objdump.exe  -D uart.o

uart.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <UART_Send_String>:
   0:   e52db004        push    {fp}            ; (str fp, [sp, #-4]!)
   4:   e28db000        add     fp, sp, #0
   8:   e24dd00c        sub     sp, sp, #12
   c:   e50b0008        str     r0, [fp, #-8]
  10:   ea000006        b       30 <UART_Send_String+0x30>
  14:   e59f3030        ldr     r3, [pc, #48]   ; 4c <UART_Send_String+0x4c>
  18:   e51b2008        ldr     r2, [fp, #-8]
  1c:   e5d22000        ldrb    r2, [r2]
  20:   e5832000        str     r2, [r3]
  24:   e51b3008        ldr     r3, [fp, #-8]
  28:   e2833001        add     r3, r3, #1
  2c:   e50b3008        str     r3, [fp, #-8]
  30:   e51b3008        ldr     r3, [fp, #-8]
  34:   e5d33000        ldrb    r3, [r3]
  38:   e3530000        cmp     r3, #0
  3c:   1afffff4        bne     14 <UART_Send_String+0x14>
  40:   e28bd000        add     sp, fp, #0
  44:   e8bd0800        ldmfd   sp!, {fp}
  48:   e12fff1e        bx      lr
  4c:   101f1000        andsne  r1, pc, r0

Disassembly of section .comment:

00000000 <.comment>:
   0:   43434700        movtmi  r4, #14080      ; 0x3700
   4:   4728203a                        ; <UNDEFINED> instruction: 0x4728203a
   8:   2029554e        eorcs   r5, r9, lr, asr #10
   c:   2e372e34        mrccs   14, 1, r2, cr7, cr4, {1}
  10:   Address 0x00000010 is out of bounds.


Disassembly of section .ARM.attributes:

00000000 <.ARM.attributes>:
   0:   00003141        andeq   r3, r0, r1, asr #2
   4:   61656100        cmnvs   r5, r0, lsl #2
   8:   01006962        tsteq   r0, r2, ror #18
   c:   00000027        andeq   r0, r0, r7, lsr #32
  10:   4d524105        ldfmie  f4, [r2, #-20]  ; 0xffffffec
  14:   45363239        ldrmi   r3, [r6, #-569]!        ; 0x239
  18:   00532d4a        subseq  r2, r3, sl, asr #26
  1c:   01080506        tsteq   r8, r6, lsl #10
  20:   04120109        ldreq   r0, [r2], #-265 ; 0x109
  24:   01150114        tsteq   r5, r4, lsl r1
  28:   01180317        tsteq   r8, r7, lsl r3
  2c:   011a0119        tsteq   sl, r9, lsl r1
  30:   Address 0x00000030 is out of bounds.
```

Startup.s with hard coded stack pointer value:

```
.global reset
reset:
    ldr sp , = 0x11000
    bl main
stop:
    b stop
```

Startup.o objdump:

```
omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ 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         0000000c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000040  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000040  2**0
                  ALLOC
  3 .ARM.attributes 00000022  00000000  00000000  00000040  2**0
                  CONTENTS, READONLY
  4 .debug_line   00000039  00000000  00000000  00000062  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_info   00000085  00000000  00000000  0000009b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_abbrev 00000014  00000000  00000000  00000120  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  00000138  2**3
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

Basic linkerscript:

This linker script also has a counter variable that automatically calculate the stack top without the need to hardcode it in the startup code.

```
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 = .;
}
```

Adding a constant variable to app.c:

We notice after adding a constant variable in the app.c file a .rodata section is generated when we display the objdump of the app.o .

```
omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ 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         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 .rodata       00000064  00000000  00000000  000000b0  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      00000012  00000000  00000000  00000114  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000032  00000000  00000000  00000126  2**0
                  CONTENTS, READONLY
```

Dumping the .elf file after linking through the previous linkerscript:

```
omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ 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      0000000c  00010000  00010000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         000000cc  0001000c  0001000c  0000800c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  000100d8  000100d8  000080d8  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  0000813c  2**0
                  CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  0000816a  2**0
                  CONTENTS, READONLY
  5 .debug_line   00000039  00000000  00000000  0000817b  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_info   00000085  00000000  00000000  000081b4  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_abbrev 00000014  00000000  00000000  00008239  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_aranges 00000020  00000000  00000000  00008250  2**3
                  CONTENTS, READONLY, DEBUGGING
```

Symbols before and after relocating:

```
MINGW32:/e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1

omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ arm-none-eabi-nm.exe app.o
00000000 D Buffer
00000000 R Buffer2
00000000 T main
         U UART_Send_String

omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ arm-none-eabi-nm.exe uart.o
00000000 T UART_Send_String

omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
00000008 t stop

omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ arm-none-eabi-nm.exe learn_in_depth.elf
000100d8 D Buffer
00010074 T Buffer2
0001005c T main
00010000 T reset
0001113c D stack_top
00010008 t stop
0001000c T UART_Send_String
```

Exporting a map file for .elf:

```
omar pc@DESKTOP-M82DFQK MINGW32 /e/Courses_Trainings/Embedded_Diploma/Assingments/Unit_3_Embedded_C/lesson_2/lab 1
$ arm-none-eabi-ld.exe  -T  linkerscript.ld -Map=out.map startup.o uart.o app.o -o learn_in_depth.elf
```

E: > Courses_Trainings > Embedded_Diploma > Assingments > Unit_3_Embedded_C > lesson_2 > lab 1 > out.map

```
1
2   Memory Configuration
3
4   Name             Origin             Length             Attributes
5   Mem              0x00000000         0x04000000         xrw
6   *default*        0x00000000         0xffffffff
7
8   Linker script and memory map
9
10                   0x00010000                . = 0x10000
11
12  .startup         0x00010000        0xc
13   startup.o(.text)
14   .text           0x00010000        0xc startup.o
15                   0x00010000                reset
16
17  .text            0x0001000c        0xcc
18   *(.text)
19   .text           0x0001000c        0x50 uart.o
20                   0x0001000c                UART_Send_String
21   .text           0x0001005c        0x18 app.o
22                   0x0001005c                main
23   *(.rodata)
24   .rodata         0x00010074        0x64 app.o
25                   0x00010074                Buffer2
26
27  .glue_7          0x000100d8         0x0
28   .glue_7         0x00000000         0x0 linker stubs
29
30  .glue_7t         0x000100d8         0x0
31   .glue_7t        0x00000000         0x0 linker stubs
32
33  .vfp11_veneer    0x000100d8         0x0
34   .vfp11_veneer   0x00000000         0x0 linker stubs
35
```

Qemu output: