

Embedded c Lesson 2 Lab 1

This lab aims to create a bare-metal software application from scratch without using IDE.

Application :

To send a string using UART protocol

Micro-Controller used:

- VersatilePB micro-controller chip based on arm926ej-s micro-processor.
 - Using it in qemu.
-
- In this lab I will make this project from scratch including startup, linker script and C codes.
 - Compile and link with arm-none-eabi cross tool chain.
 - Run the application using qemu.

Source codes: (app.c – uart.c – uart.h)

1- app.c

```
app.c
1  #include "uart.h"
2
3
4  uint8 String_Buffer [100]= "Learn-In-Depth : Ibrahim Shokry";
5  void main (void)
6  {
7      Uart_Send_String (String_Buffer);
8  }
```

2- uart.c

```
uart.c
1  #include "uart.h"
2
3  #define UART0DR *(vuint32_t*) ((uint32_t)0x101f1000)
4
5
6  void Uart_Send_String (uint8 *P_tx_string)
7  {
8      while (*P_tx_string != '\0')
9      {
10         UART0DR = (uint32_t) *P_tx_string;
11         P_tx_string++;
12     }
13 }
```

3- Uart.h

```
uart.h
1  #ifndef UART_H_
2  #define UART_H_
3
4  #include "platforms_type.h"
5
6  void Uart_Send_String (uint8 *P_tx_string);
7
8  #endif
```

Startup

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

Linker script

```
linker_script.ld
1  ENTRY(reset)
2
3  MEMORY
4  {
5      Mem (rwx) : ORIGIN = 0x00000000, LENGTH = 64M
6  }
7
8  SECTIONS
9  {
10     . = 0x10000;
11     .startup . :
12     {
13         startup.o(.text)
14     }> Mem
15     .text :
16     {
17         *(.text) *(.rodata)
18     }> Mem
19     .data :
20     {
21         *(.data)
22     }> Mem
23     .bss :
24     {
25         *(.bss) *(COMMON)
26     }> Mem
27     . = . + 0x1000 ;
28     stack_top = . ;
29 }
```

To get app.o

```
Shokry@DESKTOP-SHDL77I MINGW64 /e/MEC/embedded/diploma K S/unit 3/lec 2/llab
$ arm-none-eabi-gcc.exe -c -I. -mcpu=arm926ej-s app.c -o app.o
```

To get uart.o

```
Shokry@DESKTOP-SHDL77I MINGW64 /e/MEC/embedded/diploma K S/unit 3/lec 2/llab
$ arm-none-eabi-gcc.exe -c -I. -mcpu=arm926ej-s uart.c -o uart.o
```

To get startup.o

```
Shokry@DESKTOP-SHDL77I MINGW64 /e/MEC/embedded/diploma K S/unit 3/lec 2/llab
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted
```

To link and get .elf file and map file

```
Shokry@DESKTOP-SHDL77I MINGW64 /e/MEC/embedded/diploma K S/unit 3/lec 2/llab
$ arm-none-eabi-ld.exe -T linker_script.ld -Map=output.map startup.o app.o
uart.o -o learn-in-depth.elf
```

To get .bin or .hex file

```
Shokry@DESKTOP-SHDL77I MINGW64 /e/MEC/embedded/diploma K S/unit 3/lec 2/llab
$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
```

To execute the app on qemu (simulation app)

```
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
Learn-In-Depth : Ibrahim Shokry
```

Symbols

app.o symbols

```
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D String_Buffer
          U Uart_Send_String
```

uart.o symbols

```
Shokry@DESKTOP-SHDL77I MINGW64 /e/MEC
$ arm-none-eabi-nm.exe uart.o
00000000 T Uart_Send_String
```

startup.o symbols

```
Shokry@DESKTOP-SHDL77I MINGW64 /e/MEC
$ arm-none-eabi-nm.exe startup.o
          U main
00000000 T reset
          U stack_top
00000008 t stop
```

The all-project symbols after linking and resolving

```
$ arm-none-eabi-nm.exe learn-in-depth.elf
00010010 T main
00010000 T reset
000110dc D stack_top
00010008 t stop
00010078 D String_Buffer
00010028 T Uart_Send_String
```

Header sections:

app.o sections

```
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
```

uart.o sections

```
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
```

startup.o sections

```
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
```

The all-project sections after linking and relocating

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

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .startup      00000010  00010000  00010000  00008000  2**2
CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         00000068  00010010  00010010  00008010  2**2
CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  00010078  00010078  00008078  2**2
CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  000080dc  2**0
CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  0000810a  2**0
CONTENTS, READONLY
```