By Alfred Fayez

LAB 1 Report

Debugging Using GDB, Creating Linker_Script, Startup.s, & Startup.c For a Toggle LED Program & Simulating On Proteus

Contents

Table of Figures

By Alfred Fayez

1. Debugging Using GDB:

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1_with_
debug
$ qemu-system-arm -M versatilepb -s -S -m 128M -nographic -kernel learn-in-depth
.bin
```

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1_with_
debug
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\Embedded\Unit_3\lesson_2\assignment\lab1_with_debug\lear
n-in-depth.elf...done.
(gdb) |
```

Snippet 1 Starting GDB

Note: Symbols are successfully read by GDB

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1_with_
debug
$ qemu-system-arm -M versatilepb -s -S -m 128M -nographic -kernel learn-in-depth
.bin
```

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1_with_
debug
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\Embedded\Unit_3\lesson_2\assignment\lab1_with_debug\lear
n-in-depth.elf...done.
(gdb) display/3i $pc
(gdb) b reset
Breakpoint 1 at 0x10004: file startup.s, line 5.
(gdb) b main
Breakpoint 2 at 0x10018: file app.c, line 8.
(gdb) |
```

Snippet 2 adding breakpoints

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1_with_
debug
$ qemu-system-arm -M versatilepb -s -S -m 128M -nographic -kernel learn-in-depth
.bin
```

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\Embedded\Unit_3\lesson_2\assignment\lab1_with_debug\lear
n-in-depth.elf...done.
(gdb) display/3i $pc
(gdb) b reset
Breakpoint 1 at 0x10004: file startup.s, line 5.
(gdb) b main
Breakpoint 2 at 0x10018: file app.c, line 8.
(gdb) si
(gdb) The program is not being run.
target remote localhost:1234
Remote debugging using localhost:1234
0x00000000 in ?? ()
1: x/3i $pc
=> 0x0:  mov     r0, #0
   0x4:  ldr     r1, [pc, #4]    ; 0x10
   0x8:  ldr     r2, [pc, #4]    ; 0x14
(gdb) |
```

Snippet 3 connecting to the QEMU simulation

By Alfred Fayez

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1_with_
debug
$ qemu-system-arm -M versatilepb -s -S -m 128M -nographic -kernel learn-in-depth
.bin
```

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Breakpoint 1, reset () at startup.s:5
5                       bl      main
1: x/3i $pc
=> 0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      b       0x10008 <stop>
   0x1000c <stop+4>:    andeq   r1, r0, r0
(gdb) c
Continuing.

Breakpoint 2, main () at app.c:8
8               UART_Send_string (string_buffer);
1: x/3i $pc
=> 0x10018 <main+8>:    ldr     r0, [pc, #4]    ; 0x10024 <main+20>
   0x1001c <main+12>:   bl      0x10028 <UART_Send_string>
   0x10020 <main+16>:   pop     {r11, pc}
(gdb) l
3
4       unsigned char string_buffer[100] = "Learn-in-depth: Alfred Fayez";
5       unsigned char const string_buffer2[100] = "Learn-in-depth: Alfred Fayez"
;
6       void main()
7       {
8               UART_Send_string (string_buffer);
9       }(gdb)
```

Snippet 4 transitioning throw breakpoints

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1_with_
debug
$ qemu-system-arm -M versatilepb -s -S -m 128M -nographic -kernel learn-in-depth
.bin
```

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
7       {
8               UART_Send_string (string_buffer);
9       }(gdb) si
UART_Send_string (p_tx_string=0x0) at UART.c:6
6       {
1: x/3i $pc
=> 0x10028 <UART_Send_string>:
    push        {r11}           ; (str r11, [sp, #-4]!)
   0x1002c <UART_Send_string+4>:        add     r11, sp, #0
   0x10030 <UART_Send_string+8>:        sub     sp, sp, #12
(gdb) l
1       // Alfred Fayez
2       #include "UART.h"
3       //uart register
4       #define UART0DR *((volatile unsigned int*)((unsigned int*)0x101f1000))
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++;
(gdb) b UART.c:10
Breakpoint 3 at 0x1004c: file UART.c, line 10.
(gdb)
```

Snippet 5 add breakpoint in the loop

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1_with_
debug
$ qemu-system-arm -M versatilepb -s -S -m 128M -nographic -kernel learn-in-depth
.bin
L
```

```
MINGW64:/e/Embedded/Unit_3/lesson_2/assignment/lab1_with_debug
1       // Alfred Fayez
2       #include "UART.h"
3       //uart register
4       #define UART0DR *((volatile unsigned int*)((unsigned int*)0x101f1000))
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++;
(gdb) b UART.c:10
Breakpoint 3 at 0x1004c: file UART.c, line 10.
(gdb) c
Continuing.

Breakpoint 3, UART_Send_string (
    p_tx_string=0x100dc <string_buffer> "Learn-in-depth: Alfred Fayez")
    at UART.c:10
10                      p_tx_string++;
1: x/3i $pc
=> 0x1004c <UART_Send_string+36>:       ldr     r3, [r11, #-8]
   0x10050 <UART_Send_string+40>:       add     r3, r3, #1
   0x10054 <UART_Send_string+44>:       str     r3, [r11, #-8]
(gdb)
```

Snippet 6 start printing each character

2. Writing Startup Code and Extracting its Object File and Analyzing it

```
MINGW64:/e/Embedded/Unit_3/lesson_3/assignment/lab2_startup_s

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_3/assignment/lab2_start
up_s
$ 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         00000008  00000000  00000000  00000034  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000003c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000003c  2**0
                  ALLOC
  3 .vectors      00000050  00000000  00000000  0000003c  2**0
                  CONTENTS, RELOC, READONLY
  4 .debug_line   0000003b  00000000  00000000  0000008c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_info   00000026  00000000  00000000  000000c7  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_abbrev 00000014  00000000  00000000  000000ed  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  00000108  2**3
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000004e  00000000  00000000  00000128  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .ARM.attributes 00000021  00000000  00000000  00000176  2**0
                  CONTENTS, READONLY

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_3/assignment/lab2_start
up_s
$
```

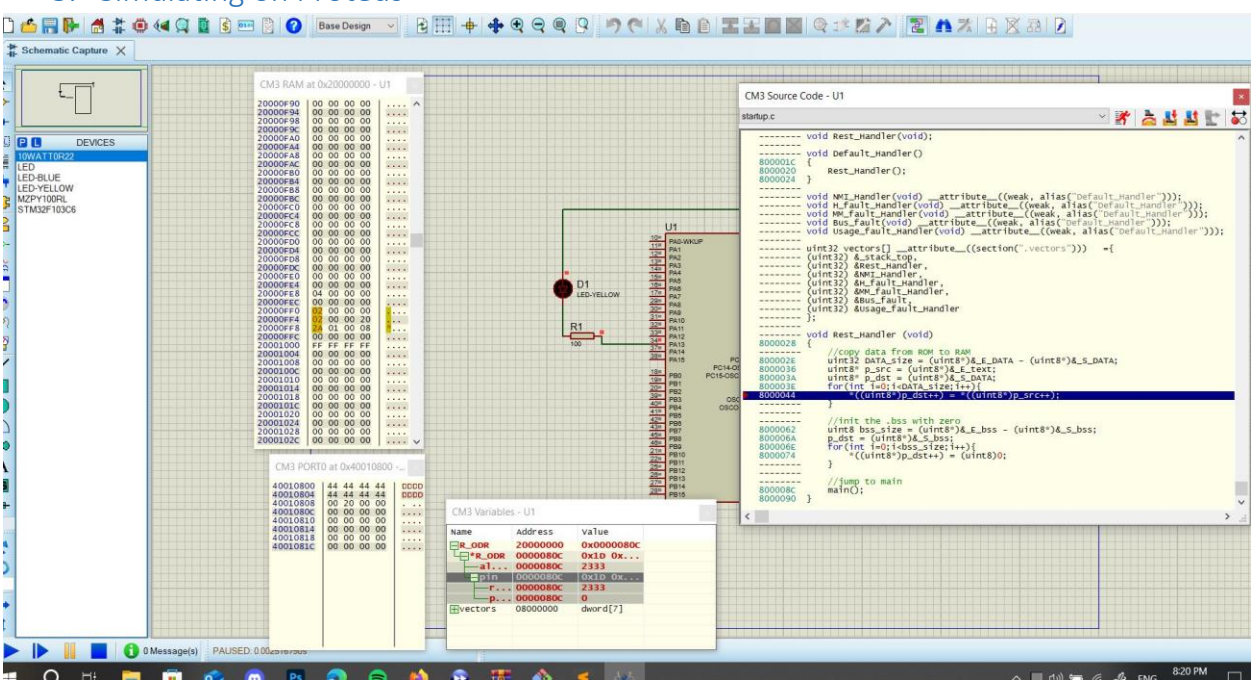
Snippet 7 startup.o sections from startup.s

By Alfred Fayez

```
MINGW64:/e/Embedded/Unit_3/lesson_3/assignment/lab2_startup_c

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_3/assignment/lab2_start
up_c
$ 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         00000090  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000c4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000c4  2**0
                  ALLOC
  3 .vectors      0000001c  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .debug_info   000001c3  00000000  00000000  000000e0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000d6  00000000  00000000  000002a3  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    0000007c  00000000  00000000  00000379  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  000003f5  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   00000207  00000000  00000000  00000415  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    000001dc  00000000  00000000  0000061c  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      0000007c  00000000  00000000  000007f8  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000050  00000000  00000000  00000874  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 12 .ARM.attributes 00000033  00000000  00000000  000008c4  2**0
                  CONTENTS, READONLY

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_3/assignment/lab2_startup_c
$
```

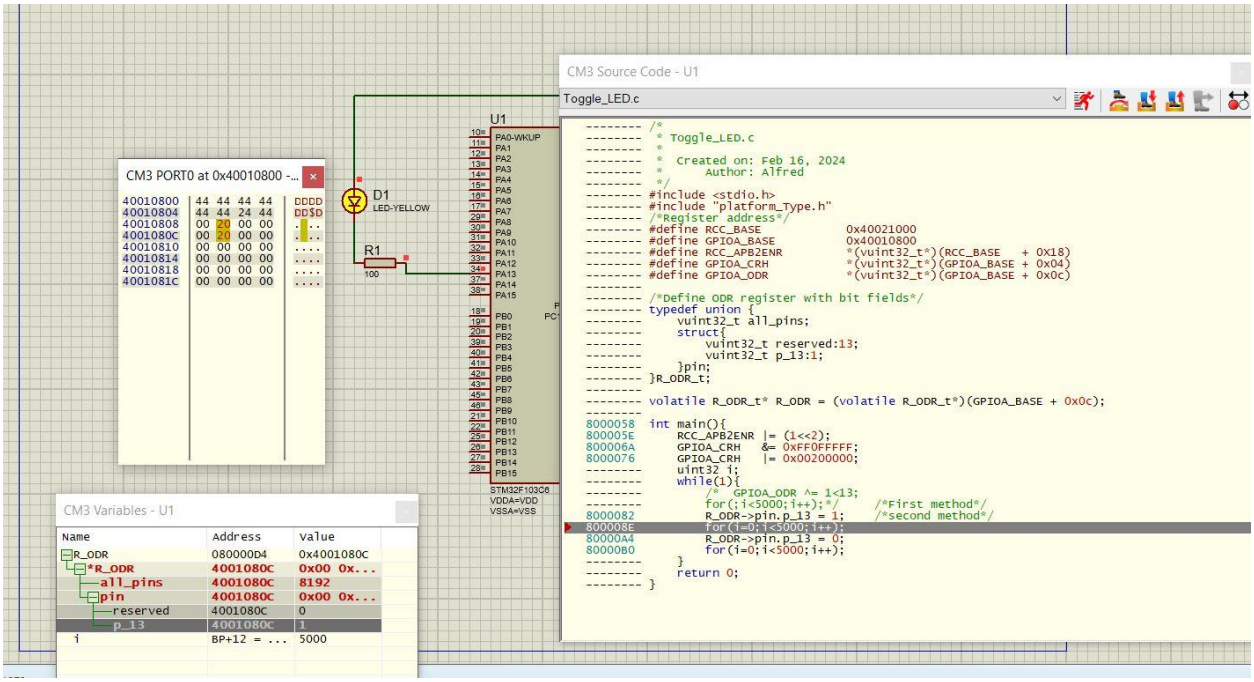
Snippet 8 startup.o sections from startup.c

3. Simulating on Proteus

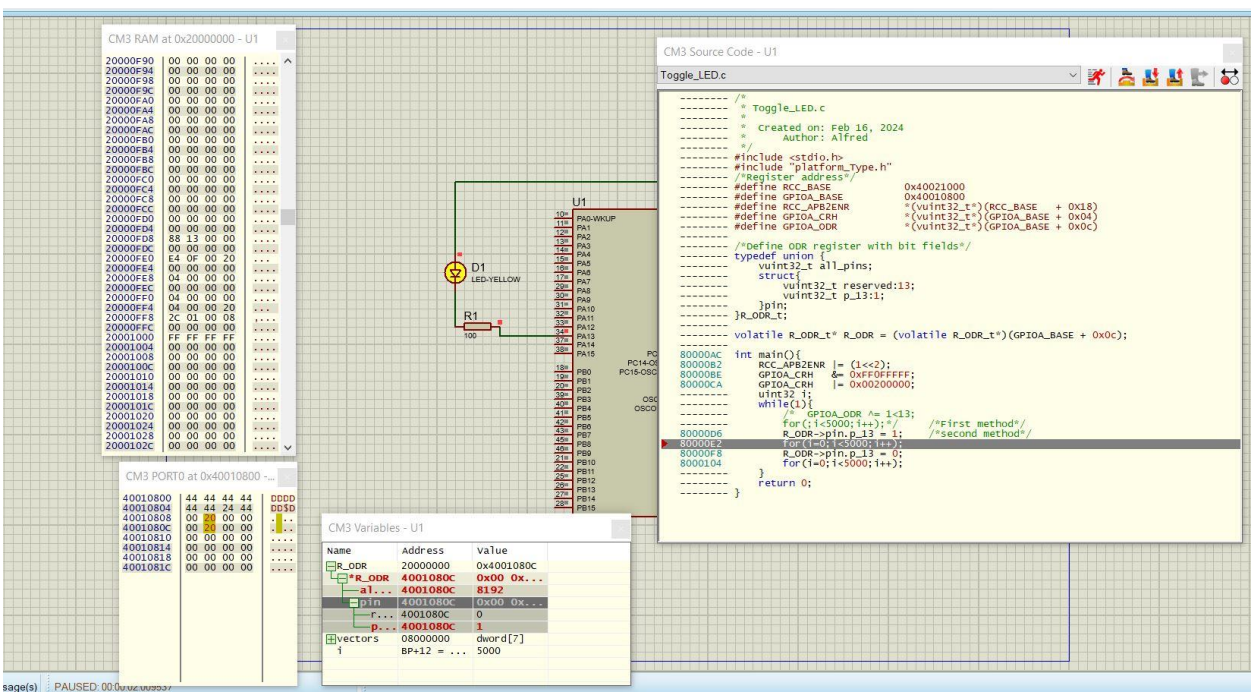


Snippet 9 debug data copying from ROM to SRAM

By Alfred Fayez



Snippet 10 simulation on proteus with startup.s



Snippet 11 simulation on proteus with startup.c

4. Symbol Table Of The Object Files & The .elf File

```
MINGW64:/e/Embedded/Unit_3/lesson_3/assignment/Embedded-Online-Dipl...

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_3/assignment/Embedded-O
nline-Diploma/Unit_3/lesson_3_assignment/lab2_(Toggle a led on STM32 using start
up.s) (main)
$ arm-none-eabi-nm.exe Toggle_LED.o Toggle_LED.elf

Toggle_LED.o:
00000000 T main
00000000 D R_ODR

Toggle_LED.elf:
08000050 t _reset
08000058 T main
080000d4 D R_ODR
08000056 t Vector_handler
```

Snippet 12 Symbol tables with startup.s

```
MINGW64:/e/Embedded/Unit_3/lesson_3/assignment/Embedded-Online-Dipl...

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_3/assignment/Embedded-O
nline-Diploma/Unit_3/lesson_3_assignment/lab2_(Toggle a led on STM32 using start
up.c) (main)
$ arm-none-eabi-nm.exe Toggle_LED.o Toggle_LED.elf

Toggle_LED.o:
00000000 T main
00000000 D R_ODR

Toggle_LED.elf:
20000004 B _E_bss
20000004 D _E_DATA
08000128 T _E_text
20000004 B _S_bss
20000000 D _S_DATA
20001004 B _stack_top
0800001c W Bus_fault
0800001c T Default_Handler
0800001c W H_fault_Handler
080000ac T main
0800001c W MM_fault_Handler
0800001c W NMI_Handler
20000000 D R_ODR
08000028 T Rest_Handler
0800001c W Usage_fault_Handler
08000000 T vectors
```

Snippet 13 Symbol tables with startup.c