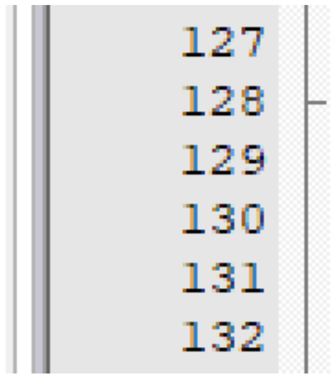
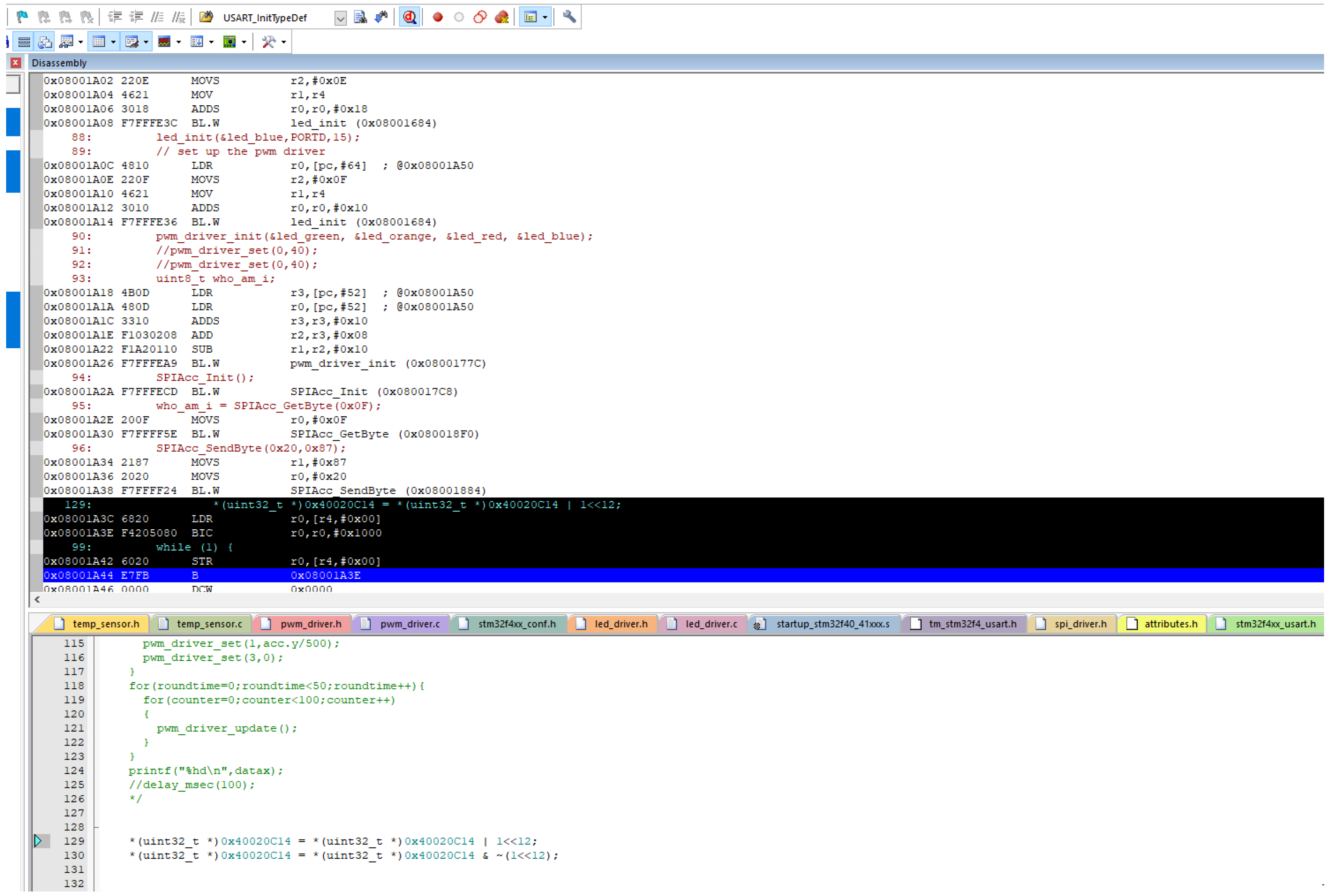
ESS_Lab_1

- **Task 3(b)**: Volatile
- **Task 7:** Redirect printf() via ETM
- **Exercise 3:** Unit Test

Volatile

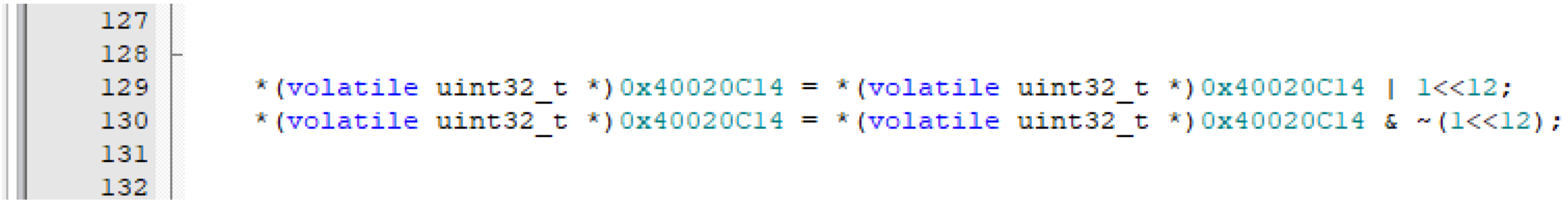


```
127
128
129     *(uint32_t *)0x40020C14 = *(uint32_t *)0x40020C14 | 1<<12;
130     *(uint32_t *)0x40020C14 = *(uint32_t *)0x40020C14 & ~(1<<12);
131
132
```

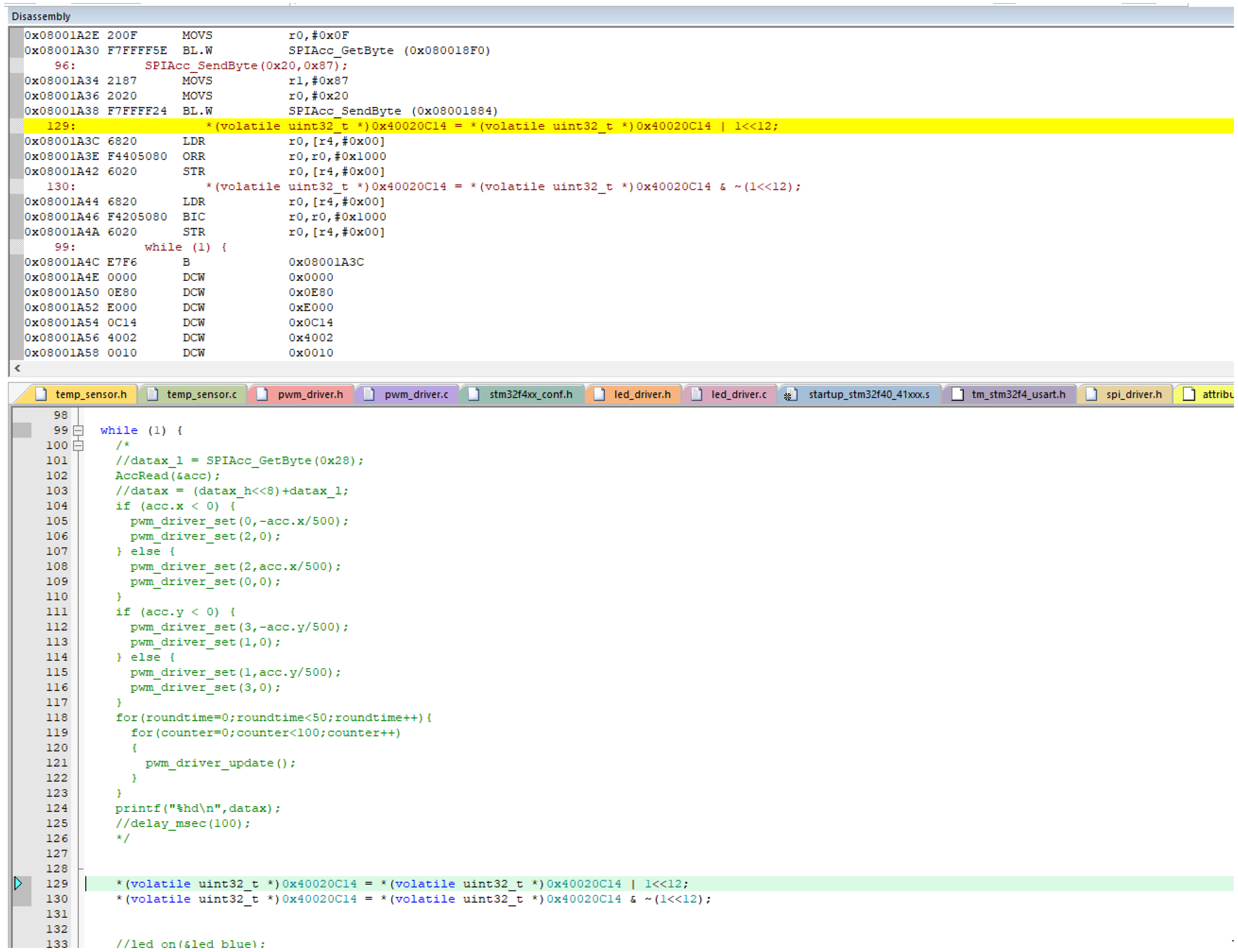


```
0x08001A02 220E      MOVS          r2,#0x0E
0x08001A04 4621      MOV           r1,r4
0x08001A06 3018      ADDS          r0,r0,#0x18
0x08001A08 F7FFFE3C  BL.W          led_init (0x08001684)
    88:         led_init(&led_blue,PORTD,15);
    89:         // set up the pwm driver
0x08001A0C 4810      LDR           r0,[pc,#64]  ; @0x08001A50
0x08001A0E 220F      MOVS          r2,#0x0F
0x08001A10 4621      MOV           r1,r4
0x08001A12 3010      ADDS          r0,r0,#0x10
0x08001A14 F7FFFE36  BL.W          led_init (0x08001684)
    90:         pwm_driver_init(&led_green, &led_orange, &led_red, &led_blue);
    91:         //pwm_driver_set(0,40);
    92:         //pwm_driver_set(0,40);
    93:         uint8_t who_am_i;
0x08001A18 4B0D      LDR           r3,[pc,#52]  ; @0x08001A50
0x08001A1A 480D      LDR           r0,[pc,#52]  ; @0x08001A50
0x08001A1C 3310      ADDS          r3,r3,#0x10
0x08001A1E F1030208  ADD           r2,r3,#0x08
0x08001A22 F1A20110  SUB           r1,r2,#0x10
0x08001A26 F7FFFEA9  BL.W          pwm_driver_init (0x0800177C)
    94:         SPIAcc_Init();
0x08001A2A F7FFFECD  BL.W          SPIAcc_Init (0x080017C8)
    95:         who_am_i = SPIAcc_GetByte(0x0F);
0x08001A2E 200F      MOVS          r0,#0x0F
0x08001A30 F7FFFF5E  BL.W          SPIAcc_GetByte (0x080018F0)
    96:         SPIAcc_SendByte(0x20,0x87);
0x08001A34 2187      MOVS          r1,#0x87
0x08001A36 2020      MOVS          r0,#0x20
0x08001A38 F7FFFF24  BL.W          SPIAcc_SendByte (0x08001884)
   129:                 *(uint32_t *)0x40020C14 = *(uint32_t *)0x40020C14 | 1<<12;
0x08001A3C 6820      LDR           r0,[r4,#0x00]
0x08001A3E F4205080  BIC           r0,r0,#0x1000
    99:         while (1) {
0x08001A42 6020      STR           r0,[r4,#0x00]
0x08001A44 E7FB      B             0x08001A3E
0x08001A46 0000      DCW           0x0000
```

```
115       pwm_driver_set(1,acc.y/500);
116       pwm_driver_set(3,0);
117     }
118     for(roundtime=0;roundtime<50;roundtime++){
119       for(counter=0;counter<100;counter++)
120       {
121         pwm_driver_update();
122       }
123     }
124     printf("%hd\n",datax);
125     //delay_msec(100);
126     */
127
128
129     *(uint32_t *)0x40020C14 = *(uint32_t *)0x40020C14 | 1<<12;
130     *(uint32_t *)0x40020C14 = *(uint32_t *)0x40020C14 & ~(1<<12);
131
132
```

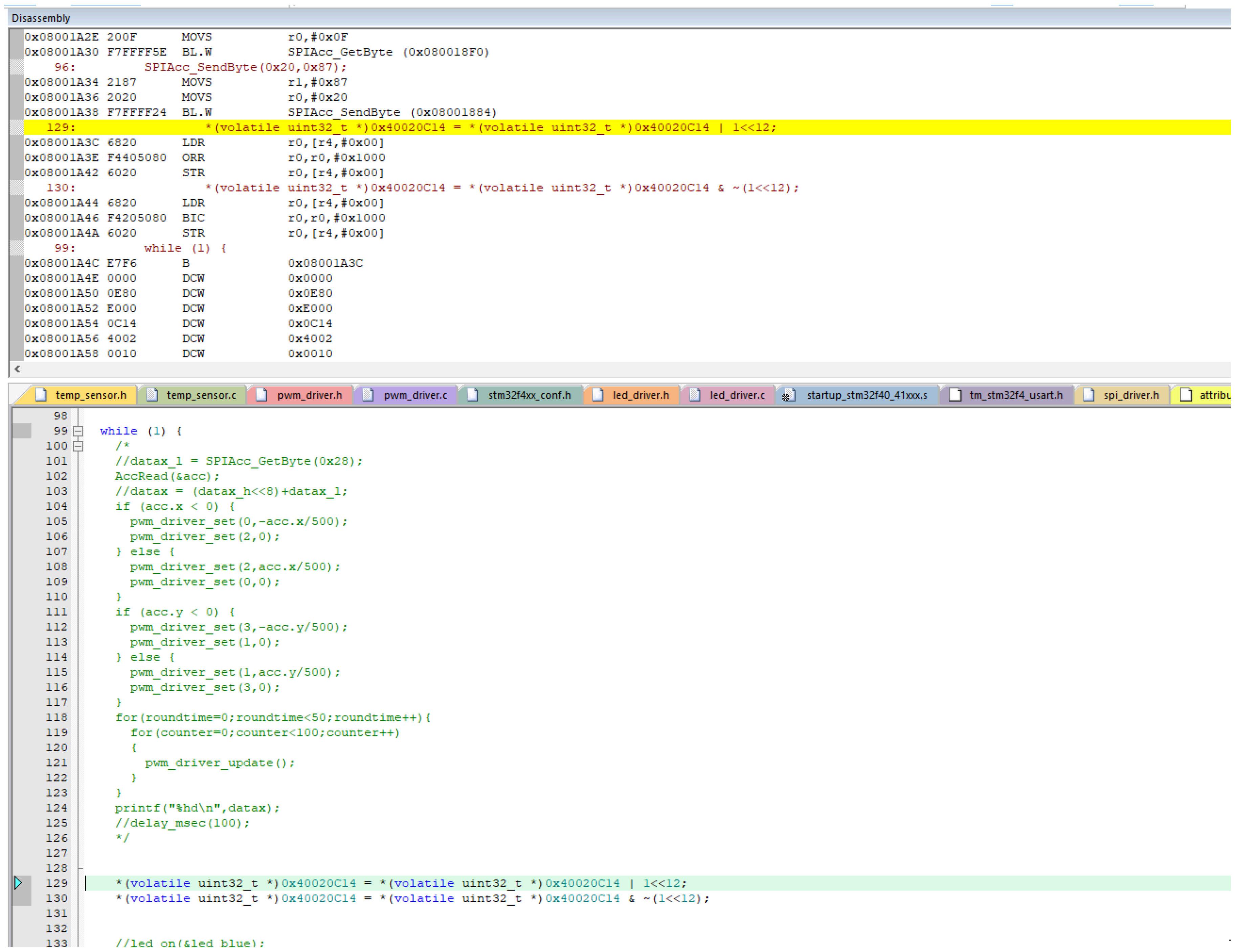


```
127
128
129     *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 | 1<<12;
130     *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 & ~(1<<12);
131
132
```



```
0x08001A2E 200F      MOVS          r0,#0x0F
0x08001A30 F7FFFF5E  BL.W          SPIAcc_GetByte (0x080018F0)
    96:          SPIAcc_SendByte(0x20,0x87);
0x08001A34 2187      MOVS          r1,#0x87
0x08001A36 2020      MOVS          r0,#0x20
0x08001A38 F7FFFF24  BL.W          SPIAcc_SendByte (0x08001884)
   129:                    *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 | 1<<12;
0x08001A3C 6820      LDR           r0,[r4,#0x00]
0x08001A3E F4405080  ORR           r0,r0,#0x1000
0x08001A42 6020      STR           r0,[r4,#0x00]
   130:                    *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 & ~(1<<12);
0x08001A44 6820      LDR           r0,[r4,#0x00]
0x08001A46 F4205080  BIC           r0,r0,#0x1000
0x08001A4A 6020      STR           r0,[r4,#0x00]
    99:          while (1) {
0x08001A4C E7F6      B             0x08001A3C
0x08001A4E 0000      DCW           0x0000
0x08001A50 0E80      DCW           0x0E80
0x08001A52 E000      DCW           0xE000
0x08001A54 0C14      DCW           0x0C14
0x08001A56 4002      DCW           0x4002
0x08001A58 0010      DCW           0x0010
```

```
 98
 99   while (1) {
100     /*
101     //datax_l = SPIAcc_GetByte(0x28);
102     AccRead(&acc);
103     //datax = (datax_h<<8)+datax_l;
104     if (acc.x < 0) {
105       pwm_driver_set(0,-acc.x/500);
106       pwm_driver_set(2,0);
107     } else {
108       pwm_driver_set(2,acc.x/500);
109       pwm_driver_set(0,0);
110     }
111     if (acc.y < 0) {
112       pwm_driver_set(3,-acc.y/500);
113       pwm_driver_set(1,0);
114     } else {
115       pwm_driver_set(1,acc.y/500);
116       pwm_driver_set(3,0);
117     }
118     for(roundtime=0;roundtime<50;roundtime++){
119       for(counter=0;counter<100;counter++)
120       {
121         pwm_driver_update();
122       }
123     }
124     printf("%hd\n",datax);
125     //delay_msec(100);
126     */
127
128
129     *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 | 1<<12;
130     *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 & ~(1<<12);
131
132
133     //led_on(&led_blue);
```

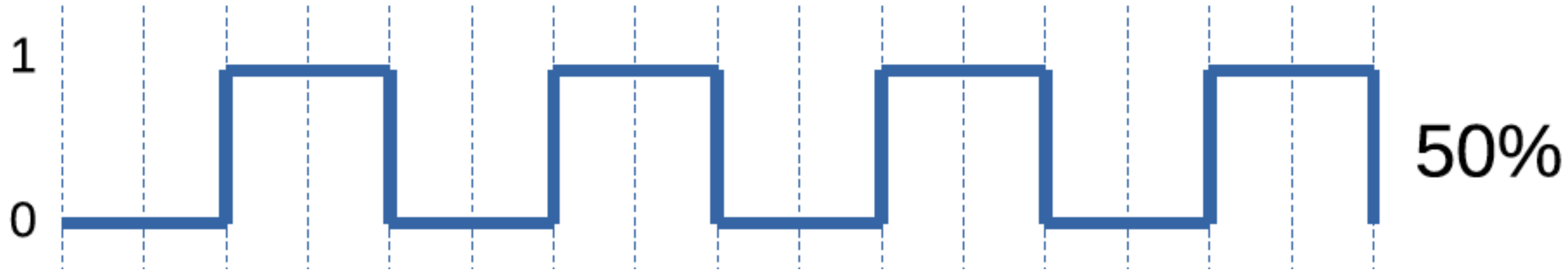


Disassembly

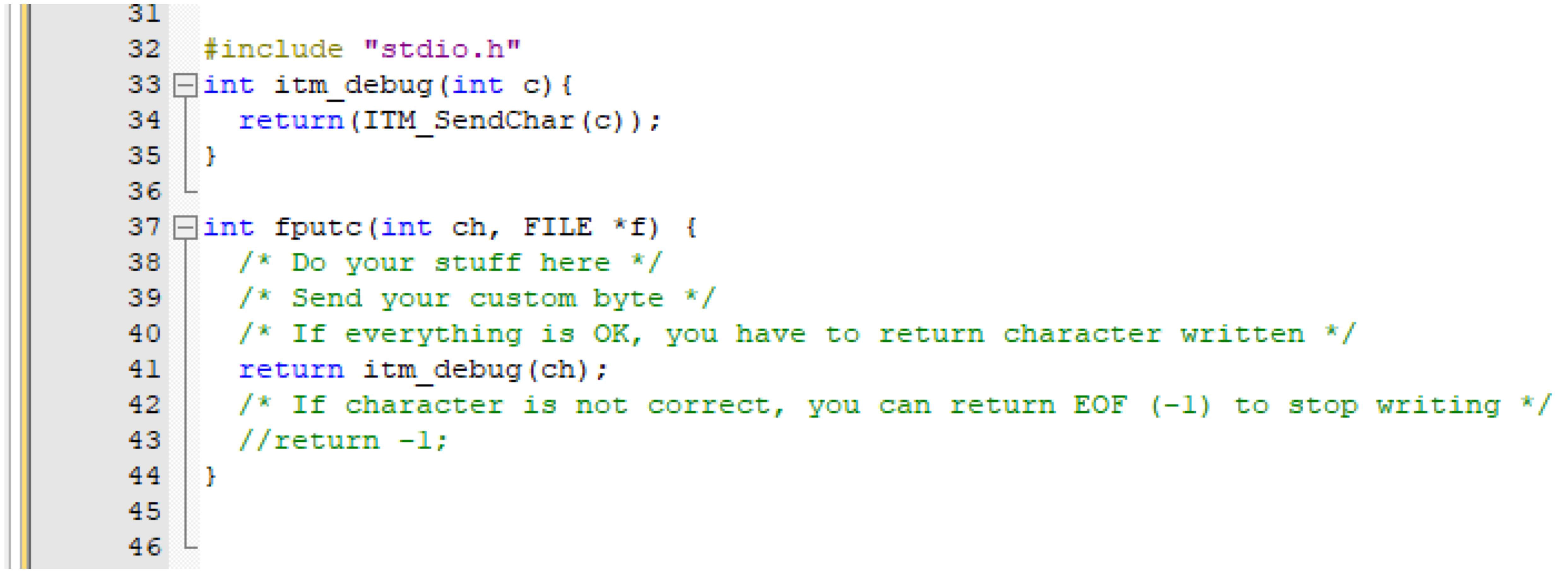
```
0x08001A2E 200F      MOVS          r0,#0x0F
0x08001A30 F7FFFF5E  BL.W          SPIAcc_GetByte (0x080018F0)
    96:         SPIAcc_SendByte(0x20,0x87);
0x08001A34 2187      MOVS          r1,#0x87
0x08001A36 2020      MOVS          r0,#0x20
0x08001A38 F7FFFF24  BL.W          SPIAcc_SendByte (0x08001884)
   129:              *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 | 1<<12;
0x08001A3C 6820      LDR           r0,[r4,#0x00]
0x08001A3E F4405080  ORR           r0,r0,#0x1000
0x08001A42 6020      STR           r0,[r4,#0x00]
   130:              *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 & ~(1<<12);
0x08001A44 6820      LDR           r0,[r4,#0x00]
0x08001A46 F4205080  BIC           r0,r0,#0x1000
0x08001A4A 6020      STR           r0,[r4,#0x00]
    99:         while (1) {
0x08001A4C E7F6      B             0x08001A3C
0x08001A4E 0000      DCW           0x0000
0x08001A50 0E80      DCW           0x0E80
0x08001A52 E000      DCW           0xE000
0x08001A54 0C14      DCW           0x0C14
0x08001A56 4002      DCW           0x4002
0x08001A58 0010      DCW           0x0010
```

temp_sensor.h | temp_sensor.c | pwm_driver.h | pwm_driver.c | stm32f4xx_conf.h | led_driver.h | led_driver.c | startup_stm32f40_41xxx.s | tm_stm32f4_usart.h | spi_driver.h | attribu

```
 98
 99  while (1) {
100    /*
101    //datax_l = SPIAcc_GetByte(0x28);
102    AccRead(&acc);
103    //datax = (datax_h<<8)+datax_l;
104    if (acc.x < 0) {
105      pwm_driver_set(0,-acc.x/500);
106      pwm_driver_set(2,0);
107    } else {
108      pwm_driver_set(2,acc.x/500);
109      pwm_driver_set(0,0);
110    }
111    if (acc.y < 0) {
112      pwm_driver_set(3,-acc.y/500);
113      pwm_driver_set(1,0);
114    } else {
115      pwm_driver_set(1,acc.y/500);
116      pwm_driver_set(3,0);
117    }
118    for(roundtime=0;roundtime<50;roundtime++){
119      for(counter=0;counter<100;counter++)
120      {
121        pwm_driver_update();
122      }
123    }
124    printf("%hd\n",datax);
125    //delay_msec(100);
126    */
127
128
129    *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 | 1<<12;
130    *(volatile uint32_t *)0x40020C14 = *(volatile uint32_t *)0x40020C14 & ~(1<<12);
131
132
133    //led_on(&led_blue);
```



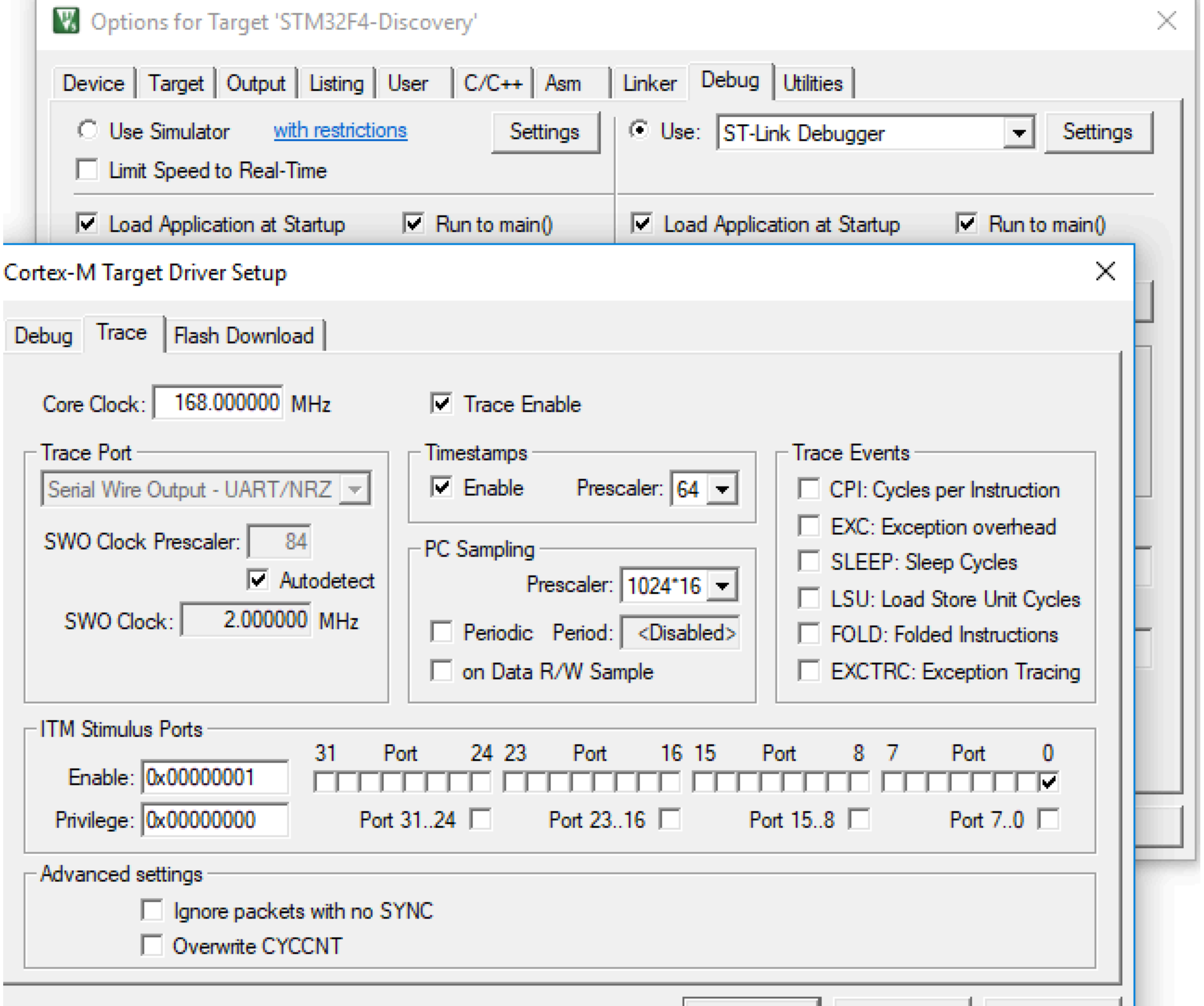
printf() redirection



```c
#include "stdio.h"
int itm_debug(int c){
  return(ITM_SendChar(c));
}

int fputc(int ch, FILE *f) {
  /* Do your stuff here */
  /* Send your custom byte */
  /* If everything is OK, you have to return character written */
  return itm_debug(ch);
  /* If character is not correct, you can return EOF (-1) to stop writing */
  //return -1;
}

```



Options for Target 'STM32F4-Discovery'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

( ) Use Simulator with restrictions Settings
[ ] Limit Speed to Real-Time
[✓] Load Application at Startup [✓] Run to main()

(●) Use: ST-Link Debugger Settings
[✓] Load Application at Startup [✓] Run to main()

Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Core Clock: 168.000000 MHz
[✓] Trace Enable

Trace Port
Serial Wire Output - UART/NRZ
SWO Clock Prescaler: 84
[✓] Autodetect
SWO Clock: 2.000000 MHz

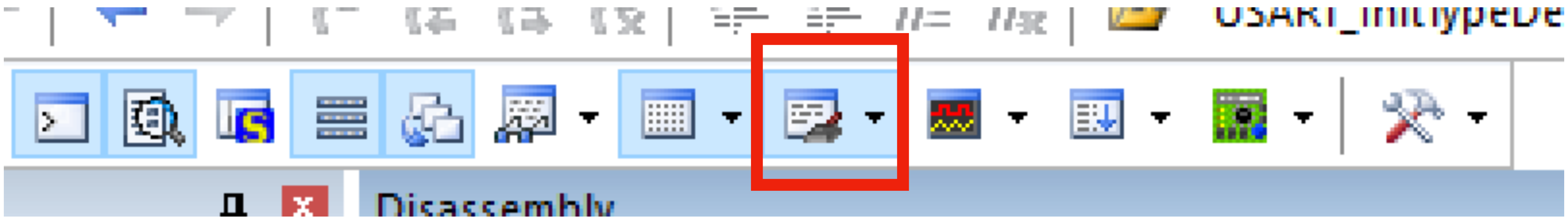
Timestamps
[✓] Enable Prescaler: 64

PC Sampling
Prescaler: 1024*16
[ ] Periodic Period: <Disabled>
[ ] on Data R/W Sample

Trace Events
[ ] CPI: Cycles per Instruction
[ ] EXC: Exception overhead
[ ] SLEEP: Sleep Cycles
[ ] LSU: Load Store Unit Cycles
[ ] FOLD: Folded Instructions
[ ] EXCTRC: Exception Tracing

ITM Stimulus Ports
31 Port 24 23 Port 16 15 Port 8 7 Port 0
Enable: 0x00000001 [ ] [✓]
Privilege: 0x00000000
Port 31..24 [ ] Port 23..16 [ ] Port 15..8 [ ] Port 7..0 [ ]

Advanced settings
[ ] Ignore packets with no SYNC
[ ] Overwrite CYCCNT


USART_InitTypeDe
Disassembly

Unit Test

- Exercise 3 (Question 2)
- Visual Studio Code
- Speaker: Mogan