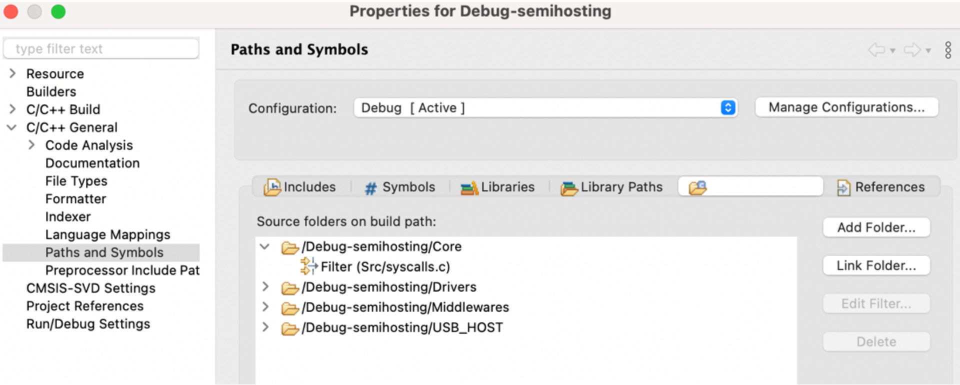
Configuring Semihosting for STM32CubeIDE

Author: Lawrence Ong
Created: 12 Jan 2021
Updated: 15 Jan 2022

Semihosting allows the board to “execute” some instruction sets via the host computer.

Configuration

1. Go to an STM32CubeIDE project.
2. Close the <project name>.ioc window. Go to the C/C++ perspective , double-click on <project name>/Core/Src/main.c to open main.c
3. Go to Project -> Properties; C/C++ General -> Paths and Symbols. Click on the Source Location tab. Under /<project name>/Core -> Filter, Click “Edit Filter”. Add, browse, and select Src -> syscalls.c. Click Apply. You will get this:

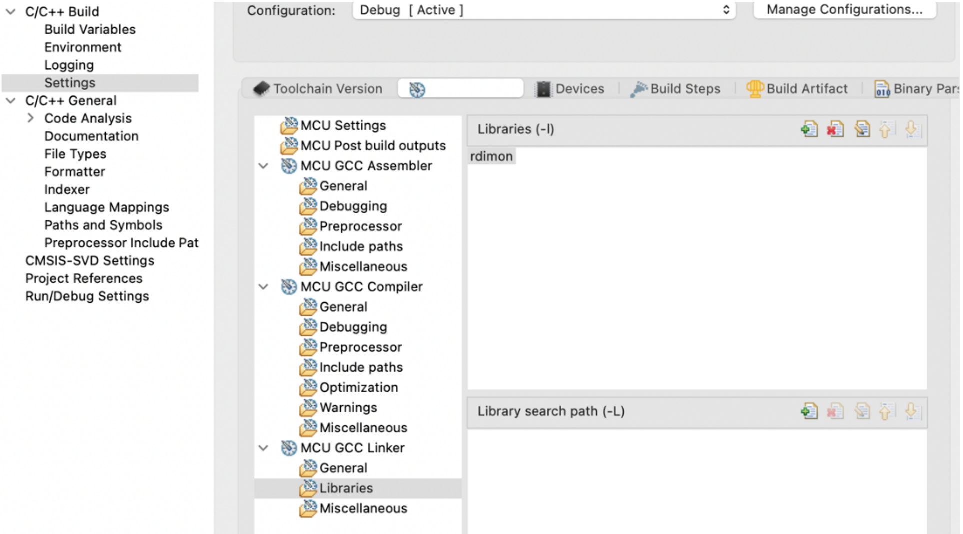


Configuration: Debug [Active]

Source folders on build path:

```
/Debug-semihosting/Core
    Filter (Src/syscalls.c)
/Debug-semihosting/Drivers
/Debug-semihosting/Middlewares
/Debug-semihosting/USB_HOST
```

4. Then, go into C/C++ Build -> Settings. Under the Tool Settings tab -> MCU GCC Linker > Libraries. In the libraries pane, click the Add… button and enter “rdimon” (without the quote). Then click Apply. You will get this:

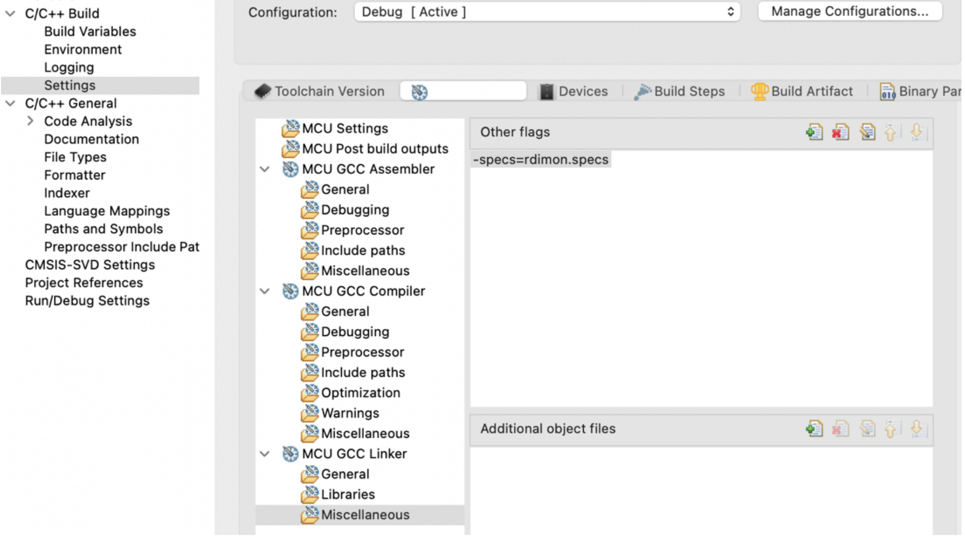


Configuration: Debug [Active]

Libraries (-l)

rdimon

5. Still on C/C++ Build -> Settings , go to Tool Settings -> MCU GCC Linker -> Miscellaneous, under the Other flags pane, click the Add… button and enter “-specs=rdimon.specs” (without the quote). Click Apply, and you will get this:



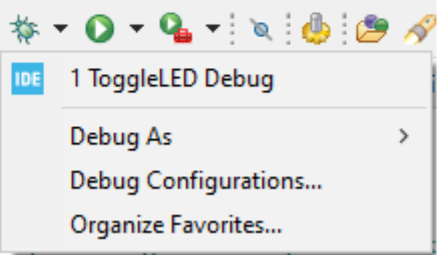
Configuration: Debug [Active]

Other flags

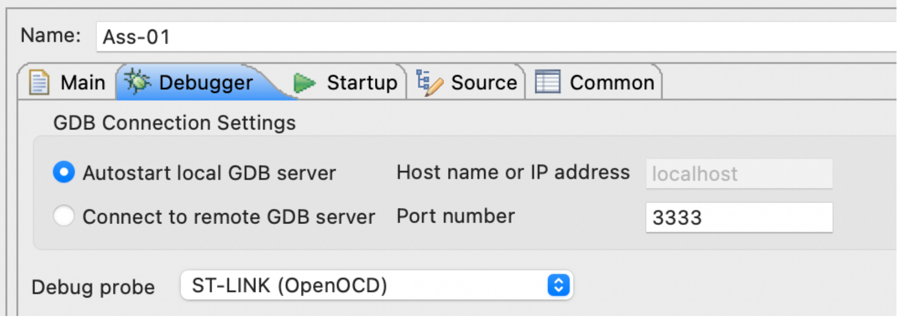
-specs=rdimon.specs

6. Click Apply and Close.
7. Connect the board’s CN1 port to the computer’s USB port.
8. Click Debug to run the project in the Debug mode. Then click Run -> Terminate.

9. Go to Debug Configurations (click the down arrow next to the green bug)



10. Go to the Debugger tab. Make sure that project name is correct. Change the Debug probe to ST-LINK (OpenOCD). Click Apply.



Name: Ass-01

Main | Debugger | Startup | Source | Common

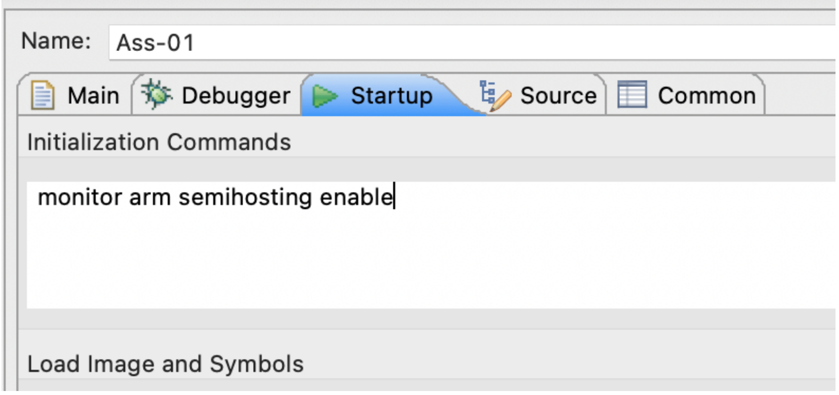
GDB Connection Settings

● Autostart local GDB server — Host name or IP address: localhost

○ Connect to remote GDB server — Port number: 3333

Debug probe: ST-LINK (OpenOCD)

11. In the Startup tab, put the following in Initialization Commands:
    monitor arm semihosting enable
    Click Apply and Close.



Name: Ass-01

Main | Debugger | Startup | Source | Common

Initialization Commands

```
monitor arm semihosting enable
```

Load Image and Symbols

12. From Project Explorer, open <project name>/Core/Src/main.c. Go to before `int main(void)`, add the following line in the USER CODE 0 section:

```
/* USER CODE BEGIN PFP */
extern void initialise_monitor_handles(void);
/* USER CODE END PFP */
```

13. In `int main(void)` before `while (1)`, add the following line inthe USER CODE 1 section:

```
int main(void)
{
  /* USER CODE BEGIN 1 */
  initialise_monitor_handles();
  /* USER CODE END 1 */
```

14. This completes the semihosting setup.

Testing

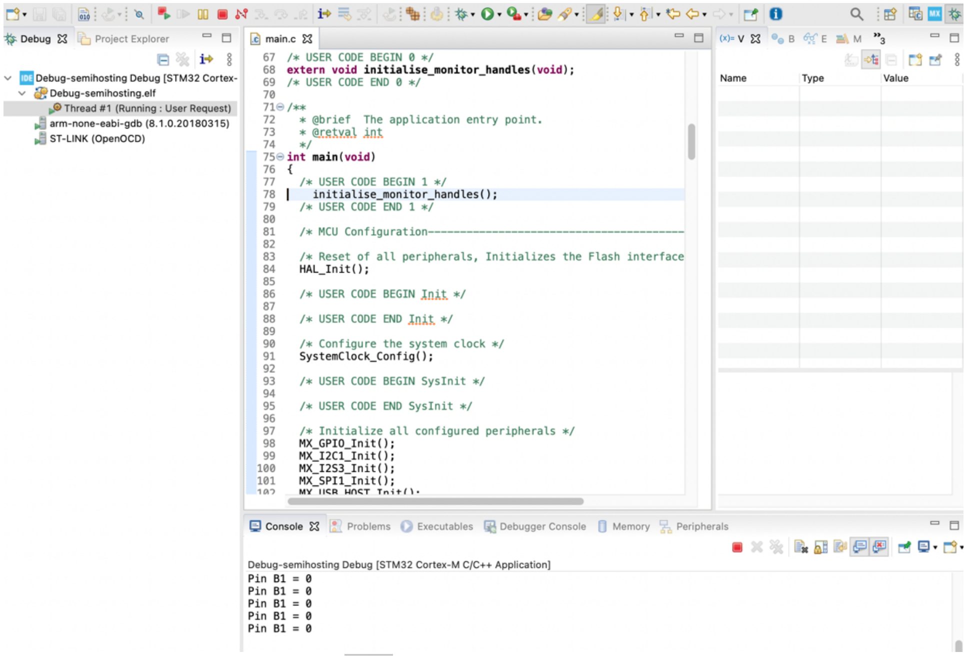
1. Insider the `while (1)` loop, add the following within the USER CODE WHILE section:

```
/* USER CODE BEGIN WHILE */
while (1)
{
    /* Define a variable to check in debugging */
    uint32_t pin_state;
    /* The variable pin_state will be the state of the pin B1 */
    pin_state = HAL_GPIO_ReadPin(B1_GPIO_Port, B1_Pin);
    /* print the state to console every second*/
    printf("Pin B1 = %lu\n", pin_state);
    HAL_Delay(1000);
  /* USER CODE END WHILE */
```

2. Before the main () function, add the following within USER CODE Includes:

```
/* USER CODE BEGIN Includes */
// for printf
#include <stdio.h>
/* USER CODE END Includes */
```

3. Note: In OpenOCD's semihosting implementation, every string must be terminated with the newline character (\n) before the string appears on the OpenOCD console.
4. Click Build to confirm that there is no error.
5. Click Debug .
6. Click Resume . The status of B1 pin will be printed in Console every second as shown below. Press the blue B1 User button and the pin state will change to 1. Otherwise, it will be zero.



```
/* USER CODE BEGIN 0 */
extern void initialise_monitor_handles(void);
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
    initialise_monitor_handles();
  /* USER CODE END 1 */

  /* MCU Configuration----------------------------------------

  /* Reset of all peripherals, Initializes the Flash interface
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S3_Init();
  MX_SPI1_Init();
  MX_USB_HOST_Init();
```

Debug-semihosting Debug [STM32 Cortex-M C/C++ Application]

```
Pin B1 = 0
Pin B1 = 0
Pin B1 = 0
Pin B1 = 0
Pin B1 = 0
```

7. When done, click Terminate .

References:
https://shawnhymel.com/1840/how-to-use-semihosting-with-stm32/
https://community.st.com/s/article/how-to-use-semihosting-with-stm32cubeide-and-stm32