STM32F407 Bare-Metal Driver Development

v1.0

Generated by Doxygen 1.15.0

Chapter 1

STM32 Driver Development

This repository contains **bare-metal driver development projects** for the **STM32F407 Discovery Board**, written entirely from scratch without using HAL or STM32Cube libraries.
Each project focuses on a specific hardware feature or peripheral of the STM32 microcontroller, helping to build a deep understanding of low-level embedded system design.

1.1 Repository Structure

Folder	Description
000_Hello_World	Basic GPIO configuration — toggle an LED using software delay.
001_STM_CLOCK_ENABLE	Demonstrates clock enabling for GPIO peripherals and verifies clock using MCO pin on a logic analyzer.
002_User_button_interrupt	Implements an **external interrupt (EXTI)** on the user button (PA0) to toggle LEDs using an ISR.
Resource	Reference materials, schematics, and supporting documentation.

1.2 Hardware Used

- **Microcontroller:** STM32F407VG (ARM Cortex-M4)
- **Board:** STM32F4 Discovery Kit
- **Core Clock:** 16 MHz (HSE)
- **Tools Required:**
  - STM32CubeIDE or any ARM-GCC-based IDE
  - ST-Link debugger/programmer
  - Logic analyzer (optional, for clock verification)

1.3 Development Approach

All drivers are written **bare-metal**, meaning:

- Direct register access (no HAL or StdPeriph libraries)
- Complete control of clock gating, GPIO modes, and interrupt configuration
- Written in **C** with startup assembly for vector table mapping

1.4 Concepts Covered So Far

- GPIO initialization and configuration
- RCC (Reset and Clock Control) fundamentals
- External interrupt (EXTI) setup using SYSCFG
- NVIC configuration and interrupt masking
- Debouncing techniques in ISRs
- Understanding the STM32 bus architecture (AHB/APB)

1.5 Upcoming Modules

Planned future modules include:

- SPI Master/Slave driver implementation
- I2C driver (polling + interrupt-based)
- UART communication
- Timer-based LED blinking (SysTick & TIM2)
- ADC/DAC configuration and data acquisition
- DMA-controlled data transfers

1.6 Learning Objective

This repository is a **step-by-step journey** toward mastering STM32 microcontrollers through:

- Building reusable driver-level code
- Understanding how ARM Cortex-M4 interfaces with peripherals
- Learning to read and apply the STM32 Reference Manual and Datasheet

1.7 Author

Yuvraj Singh Rathore
IIT Bombay Postgraduate | Scientist, DRDO
Bangalore, India
Interested in Embedded Systems, VLSI, and Radar Communication Systems

1.8 License

1.8.1 If you find this repository helpful, don't forget to star it on GitHub!

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 3

Directory Hierarchy

3.1 Directories

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Topic Documentation

5.1 Memory Segmentation

Base Addresses various memory and buses.

Macros

- #define FLASH_BASEADDR 0x08000000UL
- #define SRAM1_BASEADDR 0x20000000UL
- #define SRAM SRAM1_BASEADDR
- #define SRAM2_BASEADDR 0x2001C000UL
- #define SRAM3_BASEADDR 0x20020000UL
- #define ROM_BASEADDR 0x1FFF0000UL
- #define PERIPHERAL_BASEADDR 0x40000000UL
- #define APB1_PERIPHERAL_BASEADDR 0x40000000UL
- #define APB2_PERIPHERAL_BASEADDR 0x40010000UL
- #define AHB1_PERIPHERAL_BASEADDR 0x40020000UL
- #define AHB2_PERIPHERAL_BASEADDR 0x50000000UL

5.1.1 Detailed Description

Base Addresses various memory and buses.

5.1.2 Macro Definition Documentation

5.1.2.1 AHB1_PERIPHERAL_BASEADDR

```
#define AHB1_PERIPHERAL_BASEADDR 0x40020000UL
```
AHB1 Peripheral base address
Definition at line 36 of file stm32f407xx.h.

5.1.2.2 AHB2_PERIPHERAL_BASEADDR

```
#define AHB2_PERIPHERAL_BASEADDR 0x50000000UL
```
AHB2 Peripheral base address
Definition at line 37 of file stm32f407xx.h.

5.1.2.3 APB1_PERIPHERAL_BASEADDR

```
#define APB1_PERIPHERAL_BASEADDR 0x40000000UL
```
APB1 Peripheral base address
Definition at line 34 of file stm32f407xx.h.

5.1.2.4 APB2_PERIPHERAL_BASEADDR

```
#define APB2_PERIPHERAL_BASEADDR 0x40010000UL
```
APB2 Peripheral base address
Definition at line 35 of file stm32f407xx.h.

5.1.2.5 FLASH_BASEADDR

```
#define FLASH_BASEADDR 0x08000000UL
```

Note

> In STM32, Flash memory is called “Main Memory” because it’s the primary executable memory region containing your user application.

Definition at line 20 of file stm32f407xx.h.

5.1.2.6 PERIPHERAL_BASEADDR

```
#define PERIPHERAL_BASEADDR 0x40000000UL
```
Peripheral base address
Definition at line 33 of file stm32f407xx.h.

5.1.2.7 ROM_BASEADDR

```
#define ROM_BASEADDR 0x1FFF0000UL
```

Note

> The ROM is called "System Memory" because it holds the ST factory bootloader, used only for system-level functions like firmware programming or DFU — not for normal application execution.

Definition at line 28 of file stm32f407xx.h.

5.1.2.8 SRAM

```
#define SRAM SRAM1_BASEADDR
```
SRAM base address (It is nothing but SRAM1)
Definition at line 25 of file stm32f407xx.h.

5.1.2.9 SRAM1_BASEADDR

```
#define SRAM1_BASEADDR 0x20000000UL
```
SRAM1 base address
Definition at line 24 of file stm32f407xx.h.

5.1.2.10 SRAM2_BASEADDR

```
#define SRAM2_BASEADDR 0x2001C000UL
```
SRAM2 base address
Definition at line 26 of file stm32f407xx.h.

5.1.2.11 SRAM3_BASEADDR

```
#define SRAM3_BASEADDR 0x20020000UL
```
SRAM3 base address
Definition at line 27 of file stm32f407xx.h.

5.2 AHB1 Bus Peripheral Base Addresses

Base addresses of peripherals connected to AHB1 Bus.

Macros

- #define GPIOA_BASEADDR 0x40020000UL
- #define GPIOB_BASEADDR 0x40020400UL
- #define GPIOC_BASEADDR 0x40020800UL
- #define GPIOD_BASEADDR 0x40020C00UL
- #define GPIOE_BASEADDR 0x40021000UL
- #define GPIOF_BASEADDR 0x40021400UL
- #define GPIOG_BASEADDR 0x40021800UL
- #define GPIOH_BASEADDR 0x40021C00UL
- #define GPIOI_BASEADDR 0x40022000UL
- #define GPIOJ_BASEADDR 0x40022400UL
- #define GPIOK_BASEADDR 0x40022800UL
- #define CRC_BASEADDR 0x40023000UL
- #define RCC_BASEADDR 0x40023800UL
- #define FLASH_IF_REG_BASEADDR 0x40023C00UL
- #define BKPSRAM_BASEADDR 0x40024000UL
- #define DMA1_BASEADDR 0x40026000UL
- #define DMA2_BASEADDR 0x40026400UL
- #define ETHERNET_MAC_BASEADDR 0x40028000UL
- #define DMA2D_BASEADDR 0x4002B000UL
- #define USB_OTG_HS_BASEADDR 0x40040000UL

5.2.1 Detailed Description

Base addresses of peripherals connected to AHB1 Bus.

5.2.2 Macro Definition Documentation

5.2.2.1 BKPSRAM_BASEADDR

```
#define BKPSRAM_BASEADDR 0x40024000UL
```
BKPSRAM Peripheral base address
Definition at line 61 of file stm32f407xx.h.

5.2.2.2 CRC_BASEADDR

```
#define CRC_BASEADDR 0x40023000UL
```
Cyclic Redundancy Check (CRC) Peripheral base address
Definition at line 58 of file stm32f407xx.h.

5.2.2.3 DMA1_BASEADDR

```
#define DMA1_BASEADDR 0x40026000UL
```
Direct Memory Access 1 (DMA1) Peripheral base address
Definition at line 62 of file stm32f407xx.h.

5.2.2.4 DMA2_BASEADDR

```
#define DMA2_BASEADDR 0x40026400UL
```
Direct Memory Access 2 (DMA2) Peripheral base address
Definition at line 63 of file stm32f407xx.h.

5.2.2.5 DMA2D_BASEADDR

```
#define DMA2D_BASEADDR 0x4002B000UL
```
Chrom-Art Accelerator Controller (DMA2D) Peripheral base address
Definition at line 65 of file stm32f407xx.h.

5.2.2.6 ETHERNET_MAC_BASEADDR

```
#define ETHERNET_MAC_BASEADDR 0x40028000UL
```
ETHERNET MAC Peripheral base address
Definition at line 64 of file stm32f407xx.h.

5.2.2.7 FLASH_IF_REG_BASEADDR

```
#define FLASH_IF_REG_BASEADDR 0x40023C00UL
```
Flash interface register Peripheral base address
Definition at line 60 of file stm32f407xx.h.

5.2.2.8 GPIOA_BASEADDR

```
#define GPIOA_BASEADDR 0x40020000UL
```
GPIOA Peripheral base address
Definition at line 47 of file stm32f407xx.h.

5.2.2.9 GPIOB_BASEADDR

```
#define GPIOB_BASEADDR 0x40020400UL
```
GPIOB Peripheral base address
Definition at line 48 of file stm32f407xx.h.

5.2.2.10 GPIOC_BASEADDR

```
#define GPIOC_BASEADDR 0x40020800UL
```
GPIOC Peripheral base address
Definition at line 49 of file stm32f407xx.h.

5.2.2.11 GPIOD_BASEADDR

```
#define GPIOD_BASEADDR 0x40020C00UL
```
GPIOD Peripheral base address
Definition at line 50 of file stm32f407xx.h.

5.2.2.12 GPIOE_BASEADDR

```
#define GPIOE_BASEADDR 0x40021000UL
```
GPIOE Peripheral base address
Definition at line 51 of file stm32f407xx.h.

5.2.2.13 GPIOF_BASEADDR

```
#define GPIOF_BASEADDR 0x40021400UL
```
GPIOF Peripheral base address
Definition at line 52 of file stm32f407xx.h.

5.2.2.14 GPIOG_BASEADDR

```
#define GPIOG_BASEADDR 0x40021800UL
```
GPIOG Peripheral base address
Definition at line 53 of file stm32f407xx.h.

5.2.2.15 GPIOH_BASEADDR

```
#define GPIOH_BASEADDR 0x40021C00UL
```
GPIOH Peripheral base address
Definition at line 54 of file stm32f407xx.h.

5.2.2.16 GPIOI_BASEADDR

```
#define GPIOI_BASEADDR 0x40022000UL
```
GPIOI Peripheral base address
Definition at line 55 of file stm32f407xx.h.

5.2.2.17 GPIOJ_BASEADDR

```
#define GPIOJ_BASEADDR 0x40022400UL
```
GPIOJ Peripheral base address
Definition at line 56 of file stm32f407xx.h.

5.2.2.18 GPIOK_BASEADDR

```
#define GPIOK_BASEADDR 0x40022800UL
```
GPIOK Peripheral base address
Definition at line 57 of file stm32f407xx.h.

5.2.2.19 RCC_BASEADDR

```
#define RCC_BASEADDR 0x40023800UL
```
Reset and Clock Control (RCC) Peripheral base address
Definition at line 59 of file stm32f407xx.h.

5.2.2.20 USB_OTG_HS_BASEADDR

```
#define USB_OTG_HS_BASEADDR 0x40040000UL
```
USB On-The-Go High Speed (USB OTG HS) Peripheral base address
Definition at line 66 of file stm32f407xx.h.

5.3 AHB2 Bus Peripheral Base Addresses

Base Addresses of peripheral hanging on AHB2 Bus.

Macros

- #define USB_OTG_FS_BASEADDR 0x50000000UL
- #define DCMI_BASEADDR 0x50050000UL
- #define CRYP_BASEADDR 0x50060000UL
- #define HASH_BASEADDR 0x50060400UL
- #define RNG_BASEADDR 0x50060800UL
- #define FSMC_CR_BASEADDR 0xA0000000UL

5.3.1 Detailed Description

Base Addresses of peripheral hanging on AHB2 Bus.

5.3.2 Macro Definition Documentation

5.3.2.1 CRYP_BASEADDR

```
#define CRYP_BASEADDR 0x50060000UL
```
Cryptographic Processor (CRYP) Peripheral base address
Definition at line 77 of file stm32f407xx.h.

5.3.2.2 DCMI_BASEADDR

```
#define DCMI_BASEADDR 0x50050000UL
```
Digital Camera Interface (DCMI) Peripheral base address
Definition at line 76 of file stm32f407xx.h.

5.3.2.3 FSMC_CR_BASEADDR

```
#define FSMC_CR_BASEADDR 0xA0000000UL
```
Flexible Memory Controller (FMC) Peripheral base address
Definition at line 80 of file stm32f407xx.h.

5.3.2.4 HASH_BASEADDR

```
#define HASH_BASEADDR 0x50060400UL
```
Hash Processor (HASH) Peripheral base address
Definition at line 78 of file stm32f407xx.h.

5.3.2.5 RNG_BASEADDR

```
#define RNG_BASEADDR 0x50060800UL
```
Random Number Generator (RNG) Peripheral base address
Definition at line 79 of file stm32f407xx.h.

5.3.2.6 USB_OTG_FS_BASEADDR

```
#define USB_OTG_FS_BASEADDR 0x50000000UL
```
USB On-The-Go Full Speed (USB OTG FS) Peripheral base address
Definition at line 75 of file stm32f407xx.h.

5.4 APB1 Bus Peripheral Base Addresses

Base Addresses of peripheral hanging on APB1 Bus.

Macros

- #define TIM2_BASEADDR 0x40000000UL
- #define TIM3_BASEADDR 0x40000400UL
- #define TIM4_BASEADDR 0x40000800UL
- #define TIM5_BASEADDR 0x40000C00UL
- #define TIM6_BASEADDR 0x40001000UL
- #define TIM7_BASEADDR 0x40001400UL
- #define TIM12_BASEADDR 0x40001800UL
- #define TIM13_BASEADDR 0x40001C00UL
- #define TIM14_BASEADDR 0x40002000UL
- #define RTC_BKP_REG_BASEADDR 0x40002800UL
- #define WWDG_BASEADDR 0x40002C00UL
- #define IWDG_BASEADDR 0x40003000UL
- #define I2S2ext_BASEADDR 0x40003400UL
- #define SPI2_I2S2_BASEADDR 0x40003800UL
- #define SPI3_I2S3_BASEADDR 0x40003C00UL
- #define I2S3ext_BASEADDR 0x40004000UL
- #define USART2_BASEADDR 0x40004400UL
- #define USART3_BASEADDR 0x40004800UL
- #define UART4_BASEADDR 0x40004C00UL
- #define UART5_BASEADDR 0x40005000UL
- #define I2C1_BASEADDR 0x40005400UL
- #define I2C2_BASEADDR 0x40005800UL
- #define I2C3_BASEADDR 0x40005C00UL
- #define CAN1_BASEADDR 0x40006400UL
- #define CAN2_BASEADDR 0x40006800UL
- #define PWR_BASEADDR 0x40007000UL
- #define DAC_BASEADDR 0x40007400UL
- #define UART7_BASEADDR 0x40007800UL
- #define UART8_BASEADDR 0x40007C00UL

5.4.1 Detailed Description

Base Addresses of peripheral hanging on APB1 Bus.

5.4.2 Macro Definition Documentation

5.4.2.1 CAN1_BASEADDR

```
#define CAN1_BASEADDR 0x40006400UL
```
Controlled Area Network 1 (CAN1) Peripheral base address
Definition at line 112 of file stm32f407xx.h.

5.4.2.2 CAN2_BASEADDR

```
#define CAN2_BASEADDR 0x40006800UL
```
Controlled Area Network 2 (CAN2) Peripheral base address
Definition at line 113 of file stm32f407xx.h.

5.4.2.3 DAC_BASEADDR

```
#define DAC_BASEADDR 0x40007400UL
```
Digital to Analog Converter (DAC) Peripheral base address
Definition at line 115 of file stm32f407xx.h.

5.4.2.4 I2C1_BASEADDR

```
#define I2C1_BASEADDR 0x40005400UL
```
Inter-integrated circuit 1 (I2C1) Peripheral base address
Definition at line 109 of file stm32f407xx.h.

5.4.2.5 I2C2_BASEADDR

```
#define I2C2_BASEADDR 0x40005800UL
```
Inter-integrated circuit 2 (I2C2) Peripheral base address
Definition at line 110 of file stm32f407xx.h.

5.4.2.6 I2C3_BASEADDR

```
#define I2C3_BASEADDR 0x40005C00UL
```
Inter-integrated circuit 3 (I2C3) Peripheral base address
Definition at line 111 of file stm32f407xx.h.

5.4.2.7 I2S2ext_BASEADDR

```
#define I2S2ext_BASEADDR 0x40003400UL
```
I2S2ext Peripheral base address
Definition at line 101 of file stm32f407xx.h.

5.4.2.8 I2S3ext_BASEADDR

```
#define I2S3ext_BASEADDR 0x40004000UL
```
I2S3ext Peripheral base address
Definition at line 104 of file stm32f407xx.h.

5.4.2.9 IWDG_BASEADDR

```
#define IWDG_BASEADDR 0x40003000UL
```
Independent Watch Dog (IWDG) Peripheral base address
Definition at line 100 of file stm32f407xx.h.

5.4.2.10 PWR_BASEADDR

```
#define PWR_BASEADDR 0x40007000UL
```
Power Controller (PWR) Peripheral base address
Definition at line 114 of file stm32f407xx.h.

5.4.2.11 RTC_BKP_REG_BASEADDR

```
#define RTC_BKP_REG_BASEADDR 0x40002800UL
```
Real Time Counter Backup Register Peripheral base address
Definition at line 98 of file stm32f407xx.h.

5.4.2.12 SPI2_I2S2_BASEADDR

```
#define SPI2_I2S2_BASEADDR 0x40003800UL
```
Serial Peripheral Interface 2 (SPI2/I2S2) Peripheral base address
Definition at line 102 of file stm32f407xx.h.

5.4.2.13 SPI3_I2S3_BASEADDR

```
#define SPI3_I2S3_BASEADDR 0x40003C00UL
```
Serial Peripheral Interface 3 (SPI3/I2S3) Peripheral base address
Definition at line 103 of file stm32f407xx.h.

5.4.2.14 TIM12_BASEADDR

```
#define TIM12_BASEADDR 0x40001800UL
```
Basic Timer 12 (TIM12) Peripheral base address
Definition at line 95 of file stm32f407xx.h.

5.4.2.15 TIM13_BASEADDR

```
#define TIM13_BASEADDR 0x40001C00UL
```
Basic Timer 13 (TIM13) Peripheral base address
Definition at line 96 of file stm32f407xx.h.

5.4.2.16 TIM14_BASEADDR

```
#define TIM14_BASEADDR 0x40002000UL
```
Basic Timer 14 (TIM14) Peripheral base address
Definition at line 97 of file stm32f407xx.h.

5.4.2.17 TIM2_BASEADDR

```
#define TIM2_BASEADDR 0x40000000UL
```
Basic Timer 2 (TIM2) Peripheral base address
Definition at line 89 of file stm32f407xx.h.

5.4.2.18 TIM3_BASEADDR

```
#define TIM3_BASEADDR 0x40000400UL
```
Basic Timer 3 (TIM3) Peripheral base address
Definition at line 90 of file stm32f407xx.h.

5.4.2.19 TIM4_BASEADDR

```
#define TIM4_BASEADDR 0x40000800UL
```
Basic Timer 4 (TIM4) Peripheral base address
Definition at line 91 of file stm32f407xx.h.

5.4.2.20 TIM5_BASEADDR

```
#define TIM5_BASEADDR 0x40000C00UL
```

Basic Timer 5 (TIM5) Peripheral base address
Definition at line 92 of file stm32f407xx.h.

5.4.2.21 TIM6_BASEADDR

```
#define TIM6_BASEADDR 0x40001000UL
```

Basic Timer 6 (TIM6) Peripheral base address
Definition at line 93 of file stm32f407xx.h.

5.4.2.22 TIM7_BASEADDR

```
#define TIM7_BASEADDR 0x40001400UL
```

Basic Timer 7 (TIM7) Peripheral base address
Definition at line 94 of file stm32f407xx.h.

5.4.2.23 UART4_BASEADDR

```
#define UART4_BASEADDR 0x40004C00UL
```

Universal Synchronous Asynchronous Receiver Transmitter 4 (USART4) Peripheral base address
Definition at line 107 of file stm32f407xx.h.

5.4.2.24 UART5_BASEADDR

```
#define UART5_BASEADDR 0x40005000UL
```

Universal Synchronous Asynchronous Receiver Transmitter 5 (USART5) Peripheral base address
Definition at line 108 of file stm32f407xx.h.

5.4.2.25 UART7_BASEADDR

```
#define UART7_BASEADDR 0x40007800UL
```

Universal Asynchronous Receiver Transmitter 7 (UART7) Peripheral base address
Definition at line 116 of file stm32f407xx.h.

5.4.2.26 UART8_BASEADDR

```
#define UART8_BASEADDR 0x40007C00UL
```

Universal Asynchronous Receiver Transmitter 8 (UART8) Peripheral base address
Definition at line 117 of file stm32f407xx.h.

5.4.2.27 USART2_BASEADDR

```
#define USART2_BASEADDR 0x40004400UL
```

Universal Synchronous Asynchronous Receiver Transmitter 2 (USART2) Peripheral base address
Definition at line 105 of file stm32f407xx.h.

5.4.2.28 USART3_BASEADDR

```
#define USART3_BASEADDR 0x40004800UL
```

Universal Synchronous Asynchronous Receiver Transmitter 3 (USART3) Peripheral base address
Definition at line 106 of file stm32f407xx.h.

5.4.2.29 WWDG_BASEADDR

```
#define WWDG_BASEADDR 0x40002C00UL
```

Window Watch Dog (WWDG) Peripheral base address
Definition at line 99 of file stm32f407xx.h.

5.5 APB2 Bus Peripheral Base Addresses

Base Addresses of peripheral hanging on APB2 Bus.

Macros

- #define TIM1_BASEADDR 0x40010000UL
- #define TIM8_BASEADDR 0x40010400UL
- #define USART1_BASEADDR 0x40011000UL
- #define USART6_BASEADDR 0x40011400UL
- #define ADC123_BASEADDR 0x40012000UL
- #define SDIO_BASEADDR 0x40012C00UL
- #define SPI1_BASEADDR 0x40013000UL
- #define SPI4_BASEADDR 0x40013400UL
- #define SYSCFG_BASEADDR 0x40013800UL
- #define EXTI_BASEADDR 0x40013C00UL
- #define TIM9_BASEADDR 0x40014000UL
- #define TIM10_BASEADDR 0x40014400UL
- #define TIM11_BASEADDR 0x40014800UL
- #define SPI5_BASEADDR 0x40015000UL
- #define SPI6_BASEADDR 0x40015400UL
- #define SAI1_BASEADDR 0x40015800UL
- #define LCD_TFT_BASEADDR 0x40016800UL

5.5.1 Detailed Description

Base Addresses of peripheral hanging on APB2 Bus.

5.5.2 Macro Definition Documentation

5.5.2.1 ADC123_BASEADDR

```
#define ADC123_BASEADDR 0x40012000UL
```
Analog to Digital Converter 1,2,3 (ADC) Peripheral base address
Definition at line 129 of file stm32f407xx.h.

5.5.2.2 EXTI_BASEADDR

```
#define EXTI_BASEADDR 0x40013C00UL
```
External interrupt/event controller Peripheral base address
Definition at line 134 of file stm32f407xx.h.

5.5.2.3 LCD_TFT_BASEADDR

```
#define LCD_TFT_BASEADDR 0x40016800UL
```
LCD TFT Controller (LCD_TFT) Peripheral base address
Definition at line 141 of file stm32f407xx.h.

5.5.2.4 SAI1_BASEADDR

```
#define SAI1_BASEADDR 0x40015800UL
```
Serial audio interface (SAI) Peripheral base address
Definition at line 140 of file stm32f407xx.h.

5.5.2.5 SDIO_BASEADDR

```
#define SDIO_BASEADDR 0x40012C00UL
```
Secure digital input/output interface (SDIO) Peripheral base address
Definition at line 130 of file stm32f407xx.h.

5.5.2.6 SPI1_BASEADDR

```
#define SPI1_BASEADDR 0x40013000UL
```
Serial Peripheral Interface 1 (SPI1) Peripheral base address
Definition at line 131 of file stm32f407xx.h.

5.5.2.7 SPI4_BASEADDR

```
#define SPI4_BASEADDR 0x40013400UL
```
Serial Peripheral Interface 4 (SPI4) Peripheral base address
Definition at line 132 of file stm32f407xx.h.

5.5.2.8 SPI5_BASEADDR

```
#define SPI5_BASEADDR 0x40015000UL
```
Serial Peripheral Interface 5 (SPI5) Peripheral base address
Definition at line 138 of file stm32f407xx.h.

5.5.2.9 SPI6_BASEADDR

```
#define SPI6_BASEADDR 0x40015400UL
```
Serial Peripheral Interface 6 (SPI6) Peripheral base address
Definition at line 139 of file stm32f407xx.h.

5.5.2.10 SYSCFG_BASEADDR

```
#define SYSCFG_BASEADDR 0x40013800UL
```
System configuration controller (SYSCFG) Peripheral base address
Definition at line 133 of file stm32f407xx.h.

5.5.2.11 TIM10_BASEADDR

```
#define TIM10_BASEADDR 0x40014400UL
```
Basic Timer 10 (TIM10) Peripheral base address
Definition at line 136 of file stm32f407xx.h.

5.5.2.12 TIM11_BASEADDR

```
#define TIM11_BASEADDR 0x40014800UL
```
Basic Timer 11 (TIM11) Peripheral base address
Definition at line 137 of file stm32f407xx.h.

5.5.2.13 TIM1_BASEADDR

```
#define TIM1_BASEADDR 0x40010000UL
```
Basic Timer 1 (TIM1) Peripheral base address
Definition at line 125 of file stm32f407xx.h.

5.5.2.14 TIM8_BASEADDR

```
#define TIM8_BASEADDR 0x40010400UL
```
Basic Timer 8 (TIM8) Peripheral base address
Definition at line 126 of file stm32f407xx.h.

5.5.2.15 TIM9_BASEADDR

```
#define TIM9_BASEADDR 0x40014000UL
```
Basic Timer 9 (TIM9) Peripheral base address
Definition at line 135 of file stm32f407xx.h.

5.5.2.16 USART1_BASEADDR

```
#define USART1_BASEADDR 0x40011000UL
```

Universal Synchronous Asynchronous Receiver Transmitter 1 (USART1) Peripheral base address
Definition at line 127 of file stm32f407xx.h.

5.5.2.17 USART6_BASEADDR

```
#define USART6_BASEADDR 0x40011400UL
```

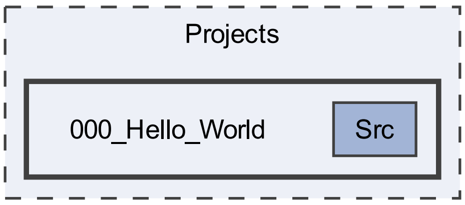
Universal Synchronous Asynchronous Receiver Transmitter 6 (USART6) Peripheral base address
Definition at line 128 of file stm32f407xx.h.

Chapter 6

Directory Documentation

6.1 Projects/000_Hello_World Directory Reference

Directory dependency graph for 000_Hello_World:

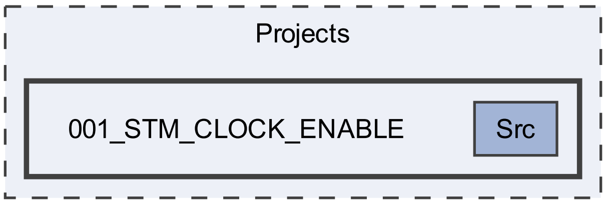


Directories

- directory Src

6.2 Projects/001_STM_CLOCK_ENABLE Directory Reference

Directory dependency graph for 001_STM_CLOCK_ENABLE:

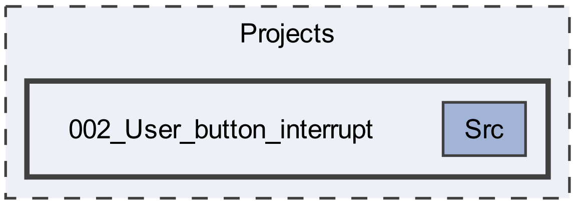


Directories

- directory Src

6.3 Projects/002_User_button_interrupt Directory Reference

Directory dependency graph for 002_User_button_interrupt:



Directories

- directory Src

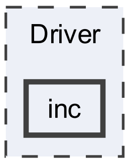
6.4 Driver Directory Reference

Directories

- directory inc

6.5 Driver/inc Directory Reference

Directory dependency graph for inc:



Files

- file stm32f407xx.h

  : Header file for STM32F407xx MCUS

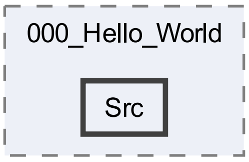
6.6 Projects Directory Reference

Directories

- directory 000_Hello_World
- directory 001_STM_CLOCK_ENABLE
- directory 002_User_button_interrupt

6.7 Projects/000_Hello_World/Src Directory Reference

Directory dependency graph for Src:



Files

- file main.c

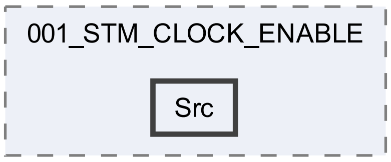
  : Main program body
- file syscalls.c

  STM32CubeIDE Minimal System calls file.
- file sysmem.c

  STM32CubeIDE System Memory calls file.

6.8 Projects/001_STM_CLOCK_ENABLE/Src Directory Reference

Directory dependency graph for Src:



Files

- file main.c

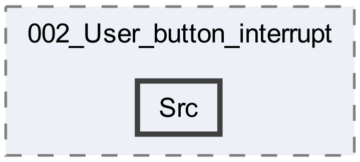
  : Main program body
- file syscalls.c

  STM32CubeIDE Minimal System calls file.
- file sysmem.c

  STM32CubeIDE System Memory calls file.

6.9 Projects/002_User_button_interrupt/Src Directory Reference

Directory dependency graph for Src:



Files

- file main.c

  : Main program body
- file syscalls.c

  STM32CubeIDE Minimal System calls file.
- file sysmem.c

  STM32CubeIDE System Memory calls file.

Chapter 7

File Documentation

7.1 Driver/inc/stm32f407xx.h File Reference

: Header file for STM32F407xx MCUS

Macros

- #define FLASH_BASEADDR 0x08000000UL
- #define SRAM1_BASEADDR 0x20000000UL
- #define SRAM SRAM1_BASEADDR
- #define SRAM2_BASEADDR 0x2001C000UL
- #define SRAM3_BASEADDR 0x20020000UL
- #define ROM_BASEADDR 0x1FFF0000UL
- #define PERIPHERAL_BASEADDR 0x40000000UL
- #define APB1_PERIPHERAL_BASEADDR 0x40000000UL
- #define APB2_PERIPHERAL_BASEADDR 0x40010000UL
- #define AHB1_PERIPHERAL_BASEADDR 0x40020000UL
- #define AHB2_PERIPHERAL_BASEADDR 0x50000000UL
- #define GPIOA_BASEADDR 0x40020000UL
- #define GPIOB_BASEADDR 0x40020400UL
- #define GPIOC_BASEADDR 0x40020800UL
- #define GPIOD_BASEADDR 0x40020C00UL
- #define GPIOE_BASEADDR 0x40021000UL
- #define GPIOF_BASEADDR 0x40021400UL
- #define GPIOG_BASEADDR 0x40021800UL
- #define GPIOH_BASEADDR 0x40021C00UL
- #define GPIOI_BASEADDR 0x40022000UL
- #define GPIOJ_BASEADDR 0x40022400UL
- #define GPIOK_BASEADDR 0x40022800UL
- #define CRC_BASEADDR 0x40023000UL
- #define RCC_BASEADDR 0x40023800UL
- #define FLASH_IF_REG_BASEADDR 0x40023C00UL
- #define BKPSRAM_BASEADDR 0x40024000UL
- #define DMA1_BASEADDR 0x40026000UL
- #define DMA2_BASEADDR 0x40026400UL
- #define ETHERNET_MAC_BASEADDR 0x40028000UL
- #define DMA2D_BASEADDR 0x4002B000UL
- #define USB_OTG_HS_BASEADDR 0x40040000UL
- #define USB_OTG_FS_BASEADDR 0x50000000UL
- #define DCMI_BASEADDR 0x50050000UL
- #define CRYP_BASEADDR 0x50060000UL
- #define HASH_BASEADDR 0x50060400UL

- #define RNG_BASEADDR 0x50060800UL
- #define FSMC_CR_BASEADDR 0xA0000000UL
- #define TIM2_BASEADDR 0x40000000UL
- #define TIM3_BASEADDR 0x40000400UL
- #define TIM4_BASEADDR 0x40000800UL
- #define TIM5_BASEADDR 0x40000C00UL
- #define TIM6_BASEADDR 0x40001000UL
- #define TIM7_BASEADDR 0x40001400UL
- #define TIM12_BASEADDR 0x40001800UL
- #define TIM13_BASEADDR 0x40001C00UL
- #define TIM14_BASEADDR 0x40002000UL
- #define RTC_BKP_REG_BASEADDR 0x40002800UL
- #define WWDG_BASEADDR 0x40002C00UL
- #define IWDG_BASEADDR 0x40003000UL
- #define I2S2ext_BASEADDR 0x40003400UL
- #define SPI2_I2S2_BASEADDR 0x40003800UL
- #define SPI3_I2S3_BASEADDR 0x40003C00UL
- #define I2S3ext_BASEADDR 0x40004000UL
- #define USART2_BASEADDR 0x40004400UL
- #define USART3_BASEADDR 0x40004800UL
- #define UART4_BASEADDR 0x40004C00UL
- #define UART5_BASEADDR 0x40005000UL
- #define I2C1_BASEADDR 0x40005400UL
- #define I2C2_BASEADDR 0x40005800UL
- #define I2C3_BASEADDR 0x40005C00UL
- #define CAN1_BASEADDR 0x40006400UL
- #define CAN2_BASEADDR 0x40006800UL
- #define PWR_BASEADDR 0x40007000UL
- #define DAC_BASEADDR 0x40007400UL
- #define UART7_BASEADDR 0x40007800UL
- #define UART8_BASEADDR 0x40007C00UL
- #define TIM1_BASEADDR 0x40010000UL
- #define TIM8_BASEADDR 0x40010400UL
- #define USART1_BASEADDR 0x40011000UL
- #define USART6_BASEADDR 0x40011400UL
- #define ADC123_BASEADDR 0x40012000UL
- #define SDIO_BASEADDR 0x40012C00UL
- #define SPI1_BASEADDR 0x40013000UL
- #define SPI4_BASEADDR 0x40013400UL
- #define SYSCFG_BASEADDR 0x40013800UL
- #define EXTI_BASEADDR 0x40013C00UL
- #define TIM9_BASEADDR 0x40014000UL
- #define TIM10_BASEADDR 0x40014400UL
- #define TIM11_BASEADDR 0x40014800UL
- #define SPI5_BASEADDR 0x40015000UL
- #define SPI6_BASEADDR 0x40015400UL
- #define SAI1_BASEADDR 0x40015800UL
- #define LCD_TFT_BASEADDR 0x40016800UL

7.1.1 Detailed Description

: Header file for STM32F407xx MCUS

Author

: Yuvraj Singh Rathore

Date

: Nov 2025

Version

: 1.0

: Provides APIs for all the peripheral connected to the processor
Definition in file stm32f407xx.h.

7.2 stm32f407xx.h

Go to the documentation of this file.

```
#ifndef INC_STM32F407XX_H_
#define INC_STM32F407XX_H_


#define FLASH_BASEADDR  0x08000000UL //This is also called as Main Memory
#define SRAM1_BASEADDR  0x20000000UL
#define SRAM            SRAM1_BASEADDR
#define SRAM2_BASEADDR  0x2001C000UL
#define SRAM3_BASEADDR  0x20020000UL
#define ROM_BASEADDR    0x1FFF0000UL
#define PERIPHERAL_BASEADDR         0x40000000UL
#define APB1_PERIPHERAL_BASEADDR    0x40000000UL
#define APB2_PERIPHERAL_BASEADDR    0x40010000UL
#define AHB1_PERIPHERAL_BASEADDR    0x40020000UL
#define AHB2_PERIPHERAL_BASEADDR    0x50000000UL



#define GPIOA_BASEADDR              0x40020000UL
#define GPIOB_BASEADDR              0x40020400UL
#define GPIOC_BASEADDR              0x40020800UL
#define GPIOD_BASEADDR              0x40020C00UL
#define GPIOE_BASEADDR              0x40021000UL
#define GPIOF_BASEADDR              0x40021400UL
#define GPIOG_BASEADDR              0x40021800UL
#define GPIOH_BASEADDR              0x40021C00UL
#define GPIOI_BASEADDR              0x40022000UL
#define GPIOJ_BASEADDR              0x40022400UL
#define GPIOK_BASEADDR              0x40022800UL
#define CRC_BASEADDR                0x40023000UL
#define RCC_BASEADDR                0x40023800UL
#define FLASH_IF_REG_BASEADDR       0x40023C00UL
#define BKPSRAM_BASEADDR            0x40024000UL
#define DMA1_BASEADDR               0x40026000UL
#define DMA2_BASEADDR               0x40026400UL
#define ETHERNET_MAC_BASEADDR       0x40028000UL
#define DMA2D_BASEADDR              0x4002B000UL
#define USB_OTG_HS_BASEADDR         0x40040000UL



#define USB_OTG_FS_BASEADDR         0x50000000UL
#define DCMI_BASEADDR               0x50050000UL
#define CRYP_BASEADDR               0x50060000UL
#define HASH_BASEADDR               0x50060400UL
#define RNG_BASEADDR                0x50060800UL
#define FSMC_CR_BASEADDR            0xA0000000UL



#define TIM2_BASEADDR            0x40000000UL
#define TIM3_BASEADDR            0x40000400UL
#define TIM4_BASEADDR            0x40000800UL
#define TIM5_BASEADDR            0x40000C00UL
#define TIM6_BASEADDR            0x40001000UL
```

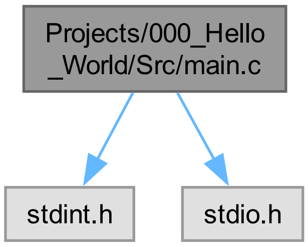
```
#define TIM7_BASEADDR            0x40001400UL
#define TIM12_BASEADDR           0x40001800UL
#define TIM13_BASEADDR           0x40001C00UL
#define TIM14_BASEADDR           0x40002000UL
#define RTC_BKP_REG_BASEADDR     0x40002800UL
#define WWDG_BASEADDR            0x40002C00UL
#define IWDG_BASEADDR            0x40003000UL
#define I2S2ext_BASEADDR         0x40003400UL
#define SPI2_I2S2_BASEADDR       0x40003800UL
#define SPI3_I2S3_BASEADDR       0x40003C00UL
#define I2S3ext_BASEADDR         0x40004000UL
#define USART2_BASEADDR          0x40004400UL
#define USART3_BASEADDR          0x40004800UL
#define UART4_BASEADDR           0x40004C00UL
#define UART5_BASEADDR           0x40005000UL
#define I2C1_BASEADDR            0x40005400UL
#define I2C2_BASEADDR            0x40005800UL
#define I2C3_BASEADDR            0x40005C00UL
#define CAN1_BASEADDR            0x40006400UL
#define CAN2_BASEADDR            0x40006800UL
#define PWR_BASEADDR             0x40007000UL
#define DAC_BASEADDR             0x40007400UL
#define UART7_BASEADDR           0x40007800UL
#define UART8_BASEADDR           0x40007C00UL


#define TIM1_BASEADDR            0x40010000UL
#define TIM8_BASEADDR            0x40010400UL
#define USART1_BASEADDR          0x40011000UL
#define USART6_BASEADDR          0x40011400UL
#define ADC123_BASEADDR          0x40012000UL
#define SDIO_BASEADDR            0x40012C00UL
#define SPI1_BASEADDR            0x40013000UL
#define SPI4_BASEADDR            0x40013400UL
#define SYSCFG_BASEADDR          0x40013800UL
#define EXTI_BASEADDR            0x40013C00UL
#define TIM9_BASEADDR            0x40014000UL
#define TIM10_BASEADDR           0x40014400UL
#define TIM11_BASEADDR           0x40014800UL
#define SPI5_BASEADDR            0x40015000UL
#define SPI6_BASEADDR            0x40015400UL
#define SAI1_BASEADDR            0x40015800UL
#define LCD_TFT_BASEADDR         0x40016800UL





















#endif /* INC_STM32F407XX_H_ */
```

7.3 Projects/000_Hello_World/Src/main.c File Reference

: Main program body

```
#include <stdint.h>
#include <stdio.h>
```

Include dependency graph for main.c:



Functions

- int main (void)

7.3.1 Detailed Description

: Main program body

Author

: Auto-generated by STM32CubeIDE

Attention

Definition in file main.c.

7.3.2 Function Documentation

7.3.2.1 main()

```
int main (
          void )
```

Definition at line 26 of file main.c.

```
00027 {
00028     /* Loop forever */
00029     printf("Hello World!!!\n");
00030     for(;;);
00031 }
```

7.4 main.c

Go to the documentation of this file.

```
00001
00018
00019 #include <stdint.h>
00020 #include <stdio.h>
00021
00022 #if !defined(__SOFT_FP__) && defined(__ARM_FP)
00023   #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU
      before use."
```

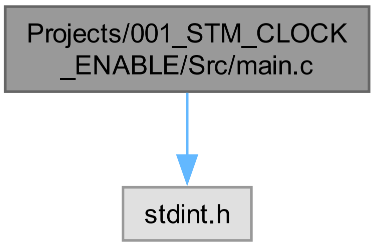
```
#endif

int main(void)
{
    /* Loop forever */
    printf("Hello World!!!\n");
    for(;;);
}
```

7.5 Projects/001_STM_CLOCK_ENABLE/Src/main.c File Reference

: Main program body

`#include <stdint.h>`

Include dependency graph for main.c:



Macros

- #define RCC_BASE_ADDRESS 0x40023800UL
- #define RCC_CR (RCC_BASE_ADDRESS + 0x00)
- #define RCC_CFGR (RCC_BASE_ADDRESS + 0x08)
- #define RCC_AHB1ENR (RCC_BASE_ADDRESS + 0x30)
- #define GPIOA_BASE_ADDRESS 0x40020000UL
- #define GPIOA_MODER (GPIOA_BASE_ADDRESS + 0x00)
- #define GPIOA_AFRL_ADDRESS (GPIOA_BASE_ADDRESS + 0x20)

Functions

- int main (void)

7.5.1 Detailed Description

: Main program body

Author

: Auto-generated by STM32CubeIDE

Attention

Definition in file main.c.

7.5.2 Macro Definition Documentation

7.5.2.1 GPIOA_AFRL_ADDRESS

```
#define GPIOA_AFRL_ADDRESS (GPIOA_BASE_ADDRESS + 0x20)
```

Definition at line 37 of file main.c.

7.5.2.2 GPIOA_BASE_ADDRESS

```
#define GPIOA_BASE_ADDRESS 0x40020000UL
```

Definition at line 33 of file main.c.

7.5.2.3 GPIOA_MODER

```
#define GPIOA_MODER (GPIOA_BASE_ADDRESS + 0x00)
```

Definition at line 35 of file main.c.

7.5.2.4 RCC_AHB1ENR

```
#define RCC_AHB1ENR (RCC_BASE_ADDRESS + 0x30)
```

Definition at line 31 of file main.c.

7.5.2.5 RCC_BASE_ADDRESS

```
#define RCC_BASE_ADDRESS 0x40023800UL
```

Definition at line 25 of file main.c.

7.5.2.6 RCC_CFGR

```
#define RCC_CFGR (RCC_BASE_ADDRESS + 0x08)
```

Definition at line 29 of file main.c.

7.5.2.7 RCC_CR

```
#define RCC_CR (RCC_BASE_ADDRESS + 0x00)
```

Definition at line 27 of file main.c.

7.5.3 Function Documentation

7.5.3.1 main()

```
int main (
               void )
```

Definition at line 39 of file main.c.

```
{
     /* Loop forever */
     //0. Enabling the HSE Clock
     uint32_t* pRCC_CR = (uint32_t*) RCC_CR;
     *pRCC_CR |= (1«16);

     //1. Setting the MCO1 Clock as HSI
     uint32_t* pRCC_CFGR_REG = (uint32_t*) RCC_CFGR;
     *pRCC_CFGR_REG &= ~(3«21);
     *pRCC_CFGR_REG |= (2«21);
     *pRCC_CFGR_REG |= (6«24);
     //*pRCC_CFGR_REG |= (3«21);

     //2. Enabling the GPIOA RCC
     uint32_t* pRCC_AHB1ENR_REG = (uint32_t*) RCC_AHB1ENR;
     *pRCC_AHB1ENR_REG |= (1«0);

     //3. Setting PA8 to alternate functionality AF0
     uint32_t* pGPIOA_MODER_REG = (uint32_t*) GPIOA_MODER;
     *pGPIOA_MODER_REG |= (2«16);

     //4. Setting the alternate function of GPIO PA8
     uint32_t* pGPIOA_AFRL = (uint32_t*) GPIOA_AFRL_ADDRESS;
     *pGPIOA_AFRL |= 0;

//  for(;;);

}
```

7.6 main.c

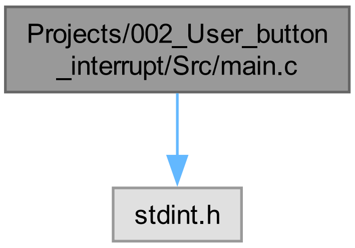
Go to the documentation of this file.

```


#include <stdint.h>

#if !defined(__SOFT_FP__) && defined(__ARM_FP)
  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
#endif

#define RCC_BASE_ADDRESS 0x40023800UL

#define RCC_CR (RCC_BASE_ADDRESS + 0x00)

#define RCC_CFGR (RCC_BASE_ADDRESS + 0x08)

#define RCC_AHB1ENR (RCC_BASE_ADDRESS + 0x30)

#define GPIOA_BASE_ADDRESS 0x40020000UL

#define GPIOA_MODER (GPIOA_BASE_ADDRESS + 0x00)

#define GPIOA_AFRL_ADDRESS (GPIOA_BASE_ADDRESS + 0x20)

int main(void)
{
    /* Loop forever */
    //0. Enabling the HSE Clock
    uint32_t* pRCC_CR = (uint32_t*) RCC_CR;
    *pRCC_CR |= (1«16);

    //1. Setting the MCO1 Clock as HSI
    uint32_t* pRCC_CFGR_REG = (uint32_t*) RCC_CFGR;
    *pRCC_CFGR_REG &= ~(3«21);
    *pRCC_CFGR_REG |= (2«21);
    *pRCC_CFGR_REG |= (6«24);
    //*pRCC_CFGR_REG |= (3«21);

    //2. Enabling the GPIOA RCC
    uint32_t* pRCC_AHB1ENR_REG = (uint32_t*) RCC_AHB1ENR;
    *pRCC_AHB1ENR_REG |= (1«0);

    //3. Setting PA8 to alternate functionality AF0
    uint32_t* pGPIOA_MODER_REG = (uint32_t*) GPIOA_MODER;
    *pGPIOA_MODER_REG |= (2«16);

    //4. Setting the alternate function of GPIO PA8
    uint32_t* pGPIOA_AFRL = (uint32_t*) GPIOA_AFRL_ADDRESS;
    *pGPIOA_AFRL |= 0;

//  for(;;);

}
```

7.7 Projects/002_User_button_interrupt/Src/main.c File Reference

: Main program body

```
#include <stdint.h>
```
Include dependency graph for main.c:



Macros

- #define RCC_BASE_ADDRESS 0x40023800UL
- #define RCC_AHB1ENR (RCC_BASE_ADDRESS + 0x30)
- #define GPIOA_BASE_ADDRESS 0x40020000UL
- #define GPIOD_BASE_ADDRESS 0x40020C00UL
- #define GPIOA_MODER (GPIOA_BASE_ADDRESS + 0x00)
- #define GPIOD_MODER (GPIOD_BASE_ADDRESS + 0x00)
- #define GPIOA_PUPDR (GPIOA_BASE_ADDRESS + 0x0C)
- #define GPIOA_IDR (GPIOA_BASE_ADDRESS + 0x10)
- #define GPIOD_ODR (GPIOD_BASE_ADDRESS + 0x14)
- #define RCC_APB2ENR (RCC_BASE_ADDRESS + 0x44)
- #define EXTI_BASE_ADDRESS 0x40013C00UL
- #define EXTI_IMR (EXTI_BASE_ADDRESS + 0x00)
- #define EXTI_RTSR (EXTI_BASE_ADDRESS + 0x08)
- #define EXTI_PR (EXTI_BASE_ADDRESS + 0x14)
- #define SYSCFG_BASE_ADDRESS 0x40013800UL
- #define SYSCFG_EXTICR1 (SYSCFG_BASE_ADDRESS + 0x08)
- #define NVIC_BASE_ADDRESS 0xE000E100UL
- #define NVIC_ISER0 (NVIC_BASE_ADDRESS)

Functions

- int main (void)
- void EXTI0_IRQHandler (void)

7.7.1 Detailed Description

: Main program body

Author

: Auto-generated by STM32CubeIDE

Attention

Definition in file main.c.

7.7.2 Macro Definition Documentation

7.7.2.1 EXTI_BASE_ADDRESS

```
#define EXTI_BASE_ADDRESS 0x40013C00UL
```
Definition at line 43 of file main.c.

7.7.2.2 EXTI_IMR

```
#define EXTI_IMR (EXTI_BASE_ADDRESS + 0x00)
```
Definition at line 44 of file main.c.

7.7.2.3 EXTI_PR

```
#define EXTI_PR (EXTI_BASE_ADDRESS + 0x14)
```
Definition at line 46 of file main.c.

7.7.2.4 EXTI_RTSR

```
#define EXTI_RTSR (EXTI_BASE_ADDRESS + 0x08)
```
Definition at line 45 of file main.c.

7.7.2.5 GPIOA_BASE_ADDRESS

```
#define GPIOA_BASE_ADDRESS 0x40020000UL
```
Definition at line 29 of file main.c.

7.7.2.6 GPIOA_IDR

```
#define GPIOA_IDR (GPIOA_BASE_ADDRESS + 0x10)
```
Definition at line 37 of file main.c.

7.7.2.7 GPIOA_MODER

```
#define GPIOA_MODER (GPIOA_BASE_ADDRESS + 0x00)
```
Definition at line 32 of file main.c.

7.7.2.8 GPIOA_PUPDR

```
#define GPIOA_PUPDR (GPIOA_BASE_ADDRESS + 0x0C)
```
Definition at line 35 of file main.c.

7.7.2.9 GPIOD_BASE_ADDRESS

```
#define GPIOD_BASE_ADDRESS 0x40020C00UL
```
Definition at line 30 of file main.c.

7.7.2.10 GPIOD_MODER

```
#define GPIOD_MODER (GPIOD_BASE_ADDRESS + 0x00)
```
Definition at line 33 of file main.c.

7.7.2.11 GPIOD_ODR

```
#define GPIOD_ODR (GPIOD_BASE_ADDRESS + 0x14)
```
Definition at line 39 of file main.c.

7.7.2.12 NVIC_BASE_ADDRESS

```
#define NVIC_BASE_ADDRESS 0xE000E100UL
```
Definition at line 51 of file main.c.

7.7.2.13 NVIC_ISER0

```
#define NVIC_ISER0 (NVIC_BASE_ADDRESS)
```

Definition at line 52 of file main.c.

7.7.2.14 RCC_AHB1ENR

```
#define RCC_AHB1ENR (RCC_BASE_ADDRESS + 0x30)
```

Definition at line 27 of file main.c.

7.7.2.15 RCC_APB2ENR

```
#define RCC_APB2ENR (RCC_BASE_ADDRESS + 0x44)
```

Definition at line 41 of file main.c.

7.7.2.16 RCC_BASE_ADDRESS

```
#define RCC_BASE_ADDRESS 0x40023800UL
```

Definition at line 25 of file main.c.

7.7.2.17 SYSCFG_BASE_ADDRESS

```
#define SYSCFG_BASE_ADDRESS 0x40013800UL
```

Definition at line 48 of file main.c.

7.7.2.18 SYSCFG_EXTICR1

```
#define SYSCFG_EXTICR1 (SYSCFG_BASE_ADDRESS + 0x08)
```

Definition at line 49 of file main.c.

7.7.3 Function Documentation

7.7.3.1 EXTI0_IRQHandler()

```
void EXTI0_IRQHandler (
            void )
```

Definition at line 122 of file main.c.

```
00122                          {
00123     volatile uint32_t *pGPIOD_ODR = (uint32_t*) GPIOD_ODR;
00124     volatile uint32_t *pGPIOA_IDR = (uint32_t*) GPIOA_IDR;
00125     volatile uint32_t* pEXTI_IMR = (uint32_t*)EXTI_IMR;
00126     volatile uint32_t* pEXTI_PR = (uint32_t*)EXTI_PR;
00127 
00128     // Clear the interrupt flag
00129     //*******Most important Clear the pending interrupt in the EXTI peripheral
00130     // To clear it we have to write 1 to it.
00131     *pEXTI_PR |= (1«0);
00132     //Mask the interrupt
00133     *pEXTI_IMR &= ~(1«0);
00134     for (int i = 0; i < 100000; i++)
00135         ;
00136     //Debounce while press
00137     *pGPIOD_ODR ^= (1 « 12);
00138     *pGPIOD_ODR ^= (1 « 13);
00139     *pGPIOD_ODR ^= (1 « 14);
00140     *pGPIOD_ODR ^= (1 « 15);
00141     while ((*pGPIOA_IDR) & 1)
00142         ;
00143     // Wait for the button hold high
00144     for (int i = 0; i < 100000; i++)
00145         ;
00146     // Debounce after release
00147     //Unmask the interrupt
00148     *pEXTI_IMR |= (1«0);
00149 
00150 
00151 }
```

7.7.3.2 main()

```
int main (
              void )
```

Definition at line 54 of file main.c.

```
00054                 {
00055     /* Loop forever */
00056     //1. Enable GPIOA peripheral
00057     volatile uint32_t *pRCC_AHB1ENR = (uint32_t*) RCC_AHB1ENR;
00058     *pRCC_AHB1ENR |= (1 « 0);
00059 
00060     //2. Set pin PA0 as the input
00061     volatile uint32_t *pGPIOA_MODER = (uint32_t*) GPIOA_MODER;
00062     *pGPIOA_MODER &= ~(3 « 0); //Setting PA0 as input
00063     //uint32_t *pGPIOA_IDR = (uint32_t*) GPIOA_IDR;
00064 
00065     //3. Enable GPIOD peripheral
00066     *pRCC_AHB1ENR |= (1 « 3);
00067 
00068     //4. Setup Pin PD12 as output
00069     volatile uint32_t *pGPIOD_MODER = (uint32_t*) GPIOD_MODER;
00070     *pGPIOD_MODER |= (1 « 24); //Setting PD12 as output
00071     *pGPIOD_MODER |= (1 « 26); //Setting PD13 as output
00072     *pGPIOD_MODER |= (1 « 28); //Setting PD14 as output
00073     *pGPIOD_MODER |= (1 « 30); //Setting PD15 as output
00074 
00075 
00076     //5. Enable the interrupt peripheral peripheral
00077     volatile uint32_t* pRCC_APB2ENR = (uint32_t*)RCC_APB2ENR;
00078     *pRCC_APB2ENR |= (1«14);
00079 
00080     //6. Set EXT0 to pin PA0
00081     volatile uint32_t* pSYSCFG_EXTICR1 = (uint32_t*)SYSCFG_EXTICR1;
00082     *pSYSCFG_EXTICR1 &= ~(15«0);
00083 
00084     //7. Unmask EXTI0
00085     volatile uint32_t* pEXTI_IMR = (uint32_t*)EXTI_IMR;
00086     *pEXTI_IMR |= (1«0);
00087 
00088     //8. Setting EXTI_RTSR for rising edge trigger
00089     volatile uint32_t* pEXTI_RTSR = (uint32_t*)EXTI_RTSR;
00090     *pEXTI_RTSR |= (1«0);
00091 
00092     //9. Enable the IRQ corresponding to the EXTI0 in the Cortex M4 processor
00093     volatile uint32_t* pNVIC_ISER0 = (uint32_t*)NVIC_ISER0;
00094     *pNVIC_ISER0 |= (1«6);
00095 
00096 
00097     //10. Toggle the LED on button press
00098     volatile uint32_t *pGPIOD_ODR = (uint32_t*) GPIOD_ODR;
00099     *pGPIOD_ODR |= (1 « 12);
00100     *pGPIOD_ODR |= (1 « 14);
00101 //  *pGPIOD_ODR |= (1«12); //Setting PA0 as input
00102 
00103     while(1);
00104 //  while (1) {
00105 //      if ((*pGPIOA_IDR) & 1) {
00106 //          for (int i = 0; i < 100000; i++)
00107 //              ; //Debounce while press
00108 //          *pGPIOD_ODR ^= (1 « 12);
00109 //          *pGPIOD_ODR ^= (1 « 13);
00110 //          *pGPIOD_ODR ^= (1 « 14);
00111 //          *pGPIOD_ODR ^= (1 « 15);
00112 //          while ((*pGPIOA_IDR) & 1)
00113 //              ; // Wait for the button hold high
00114 //          for (int i = 0; i < 100000; i++)
00115 //              ; // Debounce after release
00116 //      }
00117 //  }
00118     for (;;)
00119         ;
00120 }
```

7.8 main.c

Go to the documentation of this file.

```
00001 
00018 
00019 #include <stdint.h>
00020 
00021 #if !defined(__SOFT_FP__) && defined(__ARM_FP)
```

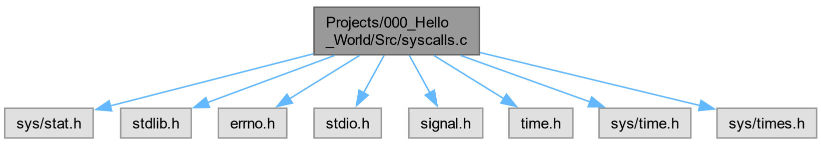
```
  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
#endif

#define RCC_BASE_ADDRESS 0x40023800UL

#define RCC_AHB1ENR  (RCC_BASE_ADDRESS + 0x30)

#define GPIOA_BASE_ADDRESS 0x40020000UL
#define GPIOD_BASE_ADDRESS 0x40020C00UL

#define GPIOA_MODER (GPIOA_BASE_ADDRESS + 0x00)
#define GPIOD_MODER (GPIOD_BASE_ADDRESS + 0x00)

#define GPIOA_PUPDR (GPIOA_BASE_ADDRESS + 0x0C)

#define GPIOA_IDR  (GPIOA_BASE_ADDRESS + 0x10)

#define GPIOD_ODR (GPIOD_BASE_ADDRESS + 0x14)

#define RCC_APB2ENR (RCC_BASE_ADDRESS + 0x44)

#define EXTI_BASE_ADDRESS 0x40013C00UL
#define EXTI_IMR (EXTI_BASE_ADDRESS + 0x00)
#define EXTI_RTSR (EXTI_BASE_ADDRESS + 0x08)
#define EXTI_PR (EXTI_BASE_ADDRESS + 0x14)

#define SYSCFG_BASE_ADDRESS 0x40013800UL
#define SYSCFG_EXTICR1 (SYSCFG_BASE_ADDRESS + 0x08)

#define NVIC_BASE_ADDRESS 0xE000E100UL
#define NVIC_ISER0 (NVIC_BASE_ADDRESS)

int main(void) {
    /* Loop forever */
    //1. Enable GPIOA peripheral
    volatile uint32_t *pRCC_AHB1ENR = (uint32_t*) RCC_AHB1ENR;
    *pRCC_AHB1ENR |= (1 « 0);

    //2. Set pin PA0 as the input
    volatile uint32_t *pGPIOA_MODER = (uint32_t*) GPIOA_MODER;
    *pGPIOA_MODER &= ~(3 « 0); //Setting PA0 as input
    //uint32_t *pGPIOA_IDR = (uint32_t*) GPIOA_IDR;

    //3. Enable GPIOD peripheral
    *pRCC_AHB1ENR |= (1 « 3);

    //4. Setup Pin PD12 as output
    volatile uint32_t *pGPIOD_MODER = (uint32_t*) GPIOD_MODER;
    *pGPIOD_MODER |= (1 « 24); //Setting PD12 as output
    *pGPIOD_MODER |= (1 « 26); //Setting PD13 as output
    *pGPIOD_MODER |= (1 « 28); //Setting PD14 as output
    *pGPIOD_MODER |= (1 « 30); //Setting PD15 as output


    //5. Enable the interrupt peripheral peripheral
    volatile uint32_t* pRCC_APB2ENR = (uint32_t*)RCC_APB2ENR;
    *pRCC_APB2ENR |= (1«14);

    //6. Set EXT0 to pin PA0
    volatile uint32_t* pSYSCFG_EXTICR1 = (uint32_t*)SYSCFG_EXTICR1;
    *pSYSCFG_EXTICR1 &= ~(15«0);

    //7. Unmask EXTI0
    volatile uint32_t* pEXTI_IMR = (uint32_t*)EXTI_IMR;
    *pEXTI_IMR |= (1«0);

    //8. Setting EXTI_RTSR for rising edge trigger
    volatile uint32_t* pEXTI_RTSR = (uint32_t*)EXTI_RTSR;
    *pEXTI_RTSR |= (1«0);

    //9. Enable the IRQ corresponding to the EXTI0 in the Cortex M4 processor
    volatile uint32_t* pNVIC_ISER0 = (uint32_t*)NVIC_ISER0;
    *pNVIC_ISER0 |= (1«6);


    //10. Toggle the LED on button press
    volatile uint32_t *pGPIOD_ODR = (uint32_t*) GPIOD_ODR;
    *pGPIOD_ODR |= (1 « 12);
    *pGPIOD_ODR |= (1 « 14);
//  *pGPIOD_ODR |= (1«12); //Setting PA0 as input

    while(1);
//  while (1) {
//      if ((*pGPIOA_IDR) & 1) {
//          for (int i = 0; i < 100000; i++)
//              ; //Debounce while press
//          *pGPIOD_ODR ^= (1 « 12);
//          *pGPIOD_ODR ^= (1 « 13);
//          *pGPIOD_ODR ^= (1 « 14);
//          *pGPIOD_ODR ^= (1 « 15);
//          while ((*pGPIOA_IDR) & 1)
//              ; // Wait for the button hold high
//          for (int i = 0; i < 100000; i++)
//              ; // Debounce after release
//      }
//  }
    for (;;)
        ;
}

void EXTI0_IRQHandler(void) {
    volatile uint32_t *pGPIOD_ODR = (uint32_t*) GPIOD_ODR;
    volatile uint32_t *pGPIOA_IDR = (uint32_t*) GPIOA_IDR;
    volatile uint32_t* pEXTI_IMR = (uint32_t*)EXTI_IMR;
    volatile uint32_t* pEXTI_PR = (uint32_t*)EXTI_PR;

    // Clear the interrupt flag
    //*******Most important Clear the pending interrupt in the EXTI peripheral
    // To clear it we have to write 1 to it.
    *pEXTI_PR |= (1«0);
    //Mask the interrupt
    *pEXTI_IMR &= ~(1«0);
    for (int i = 0; i < 100000; i++)
        ;
    //Debounce while press
    *pGPIOD_ODR ^= (1 « 12);
    *pGPIOD_ODR ^= (1 « 13);
    *pGPIOD_ODR ^= (1 « 14);
    *pGPIOD_ODR ^= (1 « 15);
    while ((*pGPIOA_IDR) & 1)
        ;
    // Wait for the button hold high
    for (int i = 0; i < 100000; i++)
        ;
    // Debounce after release
    //Unmask the interrupt
    *pEXTI_IMR |= (1«0);


}
```

7.9 Projects/000_Hello_World/Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Include dependency graph for syscalls.c:



Macros

- #define DEMCR *((volatile uint32_t*) 0xE000EDFCU)
- #define ITM_STIMULUS_PORT0 *((volatile uint32_t*) 0xE0000000)

- #define ITM_TRACE_EN ∗((volatile uint32_t∗) 0xE0000E00)

Functions

- void ITM_SendChar (uint8_t ch)
- int __io_putchar (int ch) __attribute__((weak))
- int __io_getchar (void)
- void initialise_monitor_handles ()
- int _getpid (void)
- int _kill (int pid, int sig)
- void _exit (int status)
- __attribute__ ((weak))
- int _close (int file)
- int _fstat (int file, struct stat ∗st)
- int _isatty (int file)
- int _lseek (int file, int ptr, int dir)
- int _open (char ∗path, int flags,...)
- int _wait (int ∗status)
- int _unlink (char ∗name)
- int _times (struct tms ∗buf)
- int _stat (char ∗file, struct stat ∗st)
- int _link (char ∗old, char ∗new)
- int _fork (void)
- int _execve (char ∗name, char ∗∗argv, char ∗∗env)

Variables

- char ∗∗ environ = __env

7.9.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

Attention

Definition in file syscalls.c.

7.9.2 Macro Definition Documentation

7.9.2.1 DEMCR

```
#define DEMCR ∗((volatile uint32_t∗) 0xE000EDFCU )
```
Definition at line 42 of file syscalls.c.

7.9.2.2 ITM_STIMULUS_PORT0

```
#define ITM_STIMULUS_PORT0 ∗((volatile uint32_t∗) 0xE0000000 )
```
Definition at line 45 of file syscalls.c.

7.9.2.3 ITM_TRACE_EN

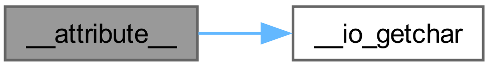
```
#define ITM_TRACE_EN *((volatile uint32_t*) 0xE0000E00 )
```
Definition at line 46 of file syscalls.c.

7.9.3 Function Documentation

7.9.3.1 __attribute__()

```
__attribute__ (
                (weak) )
```
Definition at line 99 of file syscalls.c.

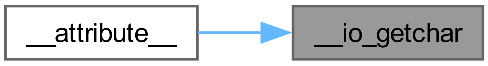
```
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }

  return len;
}
```
Here is the call graph for this function:



7.9.3.2 __io_getchar()

```
int __io_getchar (
                void )  [extern]
```
Definition at line 68 of file syscalls.c.

```
                  { 0 };
```
Here is the caller graph for this function:



7.9.3.3 __io_putchar()

```
int __io_putchar (
                int ch)  [extern]
```

7.9.3.4 _close()

```
int _close (
            int file)
```

Definition at line 125 of file syscalls.c.

```
{
  (void)file;
  return -1;
}
```

7.9.3.5 _execve()

```
int _execve (
            char * name,
            char ** argv,
            char ** env)
```

Definition at line 202 of file syscalls.c.

```
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

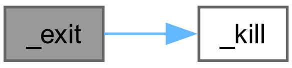
7.9.3.6 _exit()

```
void _exit (
            int status)
```

Definition at line 93 of file syscalls.c.

```
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
```

Here is the call graph for this function:



7.9.3.7 _fork()

```
int _fork (
            void )
```

Definition at line 196 of file syscalls.c.

```
{
  errno = EAGAIN;
  return -1;
}
```

7.9.3.8 _fstat()

```
int _fstat (
            int file,
            struct stat * st)
```

Definition at line 132 of file syscalls.c.

```
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
```

7.9.3.9 _getpid()

```
int _getpid (
            void )
```

Definition at line 80 of file syscalls.c.

```
{
  return 1;
}
```

7.9.3.10 _isatty()

```
int _isatty (
            int file)
```

Definition at line 139 of file syscalls.c.

```
{
  (void)file;
  return 1;
}
```

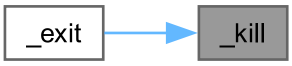
7.9.3.11 _kill()

```
int _kill (
            int pid,
            int sig)
```

Definition at line 85 of file syscalls.c.

```
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
```

Here is the caller graph for this function:



7.9.3.12 _link()

```
int _link (
            char * old,
            char * new)
```

Definition at line 188 of file syscalls.c.

```
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
```

7.9.3.13 _lseek()

```
int _lseek (
              int file,
              int ptr,
              int dir)
```

Definition at line 145 of file syscalls.c.

```
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
```

7.9.3.14 _open()

```
int _open (
              char * path,
              int flags,
               ...)
```

Definition at line 153 of file syscalls.c.

```
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
```

7.9.3.15 _stat()

```
int _stat (
              char * file,
              struct stat * st)
```

Definition at line 181 of file syscalls.c.

```
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
```

7.9.3.16 _times()

```
int _times (
              struct tms * buf)
```

Definition at line 175 of file syscalls.c.

```
{
  (void)buf;
  return -1;
}
```

7.9.3.17 _unlink()

```
int _unlink (
              char * name)
```

Definition at line 168 of file syscalls.c.

```
{
  (void)name;
  errno = ENOENT;
  return -1;
}
```

7.9.3.18 _wait()

```
int _wait (
              int * status)
```

Definition at line 161 of file syscalls.c.

```
{
  (void)status;
  errno = ECHILD;
  return -1;
}
```

7.9.3.19 initialise_monitor_handles()

```
void initialise_monitor_handles ()
```

Definition at line 76 of file syscalls.c.

```
00077 {
00078 }
```

7.9.3.20 ITM_SendChar()

```
void ITM_SendChar (
              uint8_t ch)
```

Definition at line 48 of file syscalls.c.

```
00049 {
00050
00051     //Enable TRCENA
00052     DEMCR |= ( 1 « 24);
00053
00054     //enable stimulus port 0
00055     ITM_TRACE_EN |= ( 1 « 0);
00056
00057     // read FIFO status in bit [0]:
00058     while(!(ITM_STIMULUS_PORT0 & 1));
00059
00060     //Write to ITM stimulus port0
00061     ITM_STIMULUS_PORT0 = ch;
00062 }
```

7.9.4 Variable Documentation

7.9.4.1 environ

```
char** environ = __env
```

Definition at line 72 of file syscalls.c.

7.10 syscalls.c

Go to the documentation of this file.

```
00001
00022
00023 /* Includes */
00024 #include <sys/stat.h>
00025 #include <stdlib.h>
00026 #include <errno.h>
00027 #include <stdio.h>
00028 #include <signal.h>
00029 #include <time.h>
00030 #include <sys/time.h>
00031 #include <sys/times.h>
00032
00033
00035 //                  Implementation of printf like feature using ARM Cortex M3/M4/ ITM functionality
00036 //                  This function will not work for ARM Cortex M0/M0+
00037 //                  If you are using Cortex M0, then you can use semihosting feature of openOCD
00039
00040
00041 //Debug Exception and Monitor Control Register base address
00042 #define DEMCR                   *((volatile uint32_t*) 0xE000EDFCU )
00043
00044 /* ITM register addresses */
00045 #define ITM_STIMULUS_PORT0      *((volatile uint32_t*) 0xE0000000 )
00046 #define ITM_TRACE_EN            *((volatile uint32_t*) 0xE0000E00 )
00047
00048 void ITM_SendChar(uint8_t ch)
00049 {
00050
00051     //Enable TRCENA
00052     DEMCR |= ( 1 « 24);
00053
00054     //enable stimulus port 0
00055     ITM_TRACE_EN |= ( 1 « 0);
00056
00057     // read FIFO status in bit [0]:
00058     while(!(ITM_STIMULUS_PORT0 & 1));
00059
00060     //Write to ITM stimulus port0
00061     ITM_STIMULUS_PORT0 = ch;
```

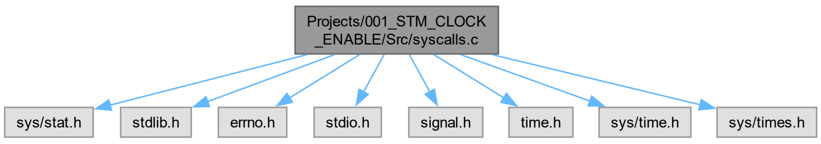
```c
}



/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));


char *__env[1] = { 0 };
char **environ = __env;


/* Functions */
void initialise_monitor_handles()
{
}

int _getpid(void)
{
  return 1;
}

int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}

void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}

__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }

  return len;
}

__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
//    __io_putchar(*ptr++);
      ITM_SendChar(*ptr++);
  }
  return len;
}

int _close(int file)
{
  (void)file;
  return -1;
}


int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}

int _isatty(int file)
{
  (void)file;
  return 1;
}

int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}

int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}

int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}

int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}

int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}

int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}

int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}

int _fork(void)
{
  errno = EAGAIN;
  return -1;
}

int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

7.11 Projects/001_STM_CLOCK_ENABLE/Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Include dependency graph for syscalls.c:



Functions

- int __io_putchar (int ch) __attribute__((weak))
- int __io_getchar (void)
- void initialise_monitor_handles ()
- int _getpid (void)
- int _kill (int pid, int sig)
- void _exit (int status)
- __attribute__ ((weak))
- int _close (int file)
- int _fstat (int file, struct stat ∗st)
- int _isatty (int file)
- int _lseek (int file, int ptr, int dir)
- int _open (char ∗path, int flags,...)
- int _wait (int ∗status)
- int _unlink (char ∗name)
- int _times (struct tms ∗buf)
- int _stat (char ∗file, struct stat ∗st)
- int _link (char ∗old, char ∗new)
- int _fork (void)
- int _execve (char ∗name, char ∗∗argv, char ∗∗env)

Variables

- char ∗∗ environ = __env

7.11.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

Attention

Definition in file syscalls.c.

7.11.2 Function Documentation

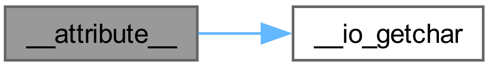
7.11.2.1 __attribute__()

```
__attribute__ (
                (weak) )
```

Definition at line 67 of file syscalls.c.

```
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }

  return len;
}
```

Here is the call graph for this function:



7.11.2.2 __io_getchar()

```
int __io_getchar (
                void )  [extern]
```

Definition at line 36 of file syscalls.c.

```
                      { 0 };
```

7.11.2.3 __io_putchar()

```
int __io_putchar (
                int ch)  [extern]
```

7.11.2.4 _close()

```
int _close (
                int file)
```

Definition at line 92 of file syscalls.c.

```
{
  (void)file;
  return -1;
}
```

7.11.2.5 _execve()

```
int _execve (
                char * name,
                char ** argv,
                char ** env)
```

Definition at line 169 of file syscalls.c.

```
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

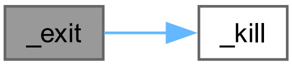
7.11.2.6 _exit()

```
void _exit (
            int status)
```

Definition at line 61 of file syscalls.c.

```
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
```

Here is the call graph for this function:



7.11.2.7 _fork()

```
int _fork (
            void )
```

Definition at line 163 of file syscalls.c.

```
{
  errno = EAGAIN;
  return -1;
}
```

7.11.2.8 _fstat()

```
int _fstat (
            int file,
            struct stat * st)
```

Definition at line 99 of file syscalls.c.

```
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
```

7.11.2.9 _getpid()

```
int _getpid (
            void )
```

Definition at line 48 of file syscalls.c.

```
{
  return 1;
}
```

7.11.2.10 _isatty()

```
int _isatty (
            int file)
```

Definition at line 106 of file syscalls.c.

```
{
  (void)file;
  return 1;
}
```

7.11.2.11 _kill()

```
int _kill (
            int pid,
            int sig)
```

Definition at line 53 of file syscalls.c.

```
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
```

7.11.2.12 _link()

```
int _link (
            char * old,
            char * new)
```

Definition at line 155 of file syscalls.c.

```
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
```

7.11.2.13 _lseek()

```
int _lseek (
            int file,
            int ptr,
            int dir)
```

Definition at line 112 of file syscalls.c.

```
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
```

7.11.2.14 _open()

```
int _open (
            char * path,
            int flags,
             ...)
```

Definition at line 120 of file syscalls.c.

```
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
```

7.11.2.15 _stat()

```
int _stat (
            char * file,
            struct stat * st)
```

Definition at line 148 of file syscalls.c.

```
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
```

7.11.2.16 _times()

```
int _times (
            struct tms * buf)
```

Definition at line 142 of file syscalls.c.

```
{
  (void)buf;
  return -1;
}
```

7.11.2.17 _unlink()

```
int _unlink (
            char * name)
```

Definition at line 135 of file syscalls.c.

```
{
  (void)name;
  errno = ENOENT;
  return -1;
}
```

7.11.2.18 _wait()

```
int _wait (
            int * status)
```

Definition at line 128 of file syscalls.c.

```
{
  (void)status;
  errno = ECHILD;
  return -1;
}
```

7.11.2.19 initialise_monitor_handles()

```
void initialise_monitor_handles ()
```

Definition at line 44 of file syscalls.c.

```
{
}
```

7.11.3 Variable Documentation

7.11.3.1 environ

```
char** environ = __env
```

Definition at line 40 of file syscalls.c.

7.12 syscalls.c

Go to the documentation of this file.

```


/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>


/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));


char *__env[1] = { 0 };
char **environ = __env;
```

```c


/* Functions */
void initialise_monitor_handles()
{
}

int _getpid(void)
{
  return 1;
}

int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}

void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}

__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }

  return len;
}

__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}

int _close(int file)
{
  (void)file;
  return -1;
}


int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}

int _isatty(int file)
{
  (void)file;
  return 1;
}

int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}

int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}

```

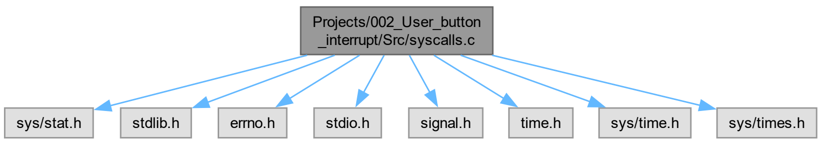
```
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}

int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}

int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}

int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}

int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}

int _fork(void)
{
  errno = EAGAIN;
  return -1;
}

int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

7.13 Projects/002_User_button_interrupt/Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Include dependency graph for syscalls.c:



Macros

- #define DEMCR ∗((volatile uint32_t∗) 0xE000EDFCU)
- #define ITM_STIMULUS_PORT0 ∗((volatile uint32_t∗) 0xE0000000)
- #define ITM_TRACE_EN ∗((volatile uint32_t∗) 0xE0000E00)

Functions

- void ITM_SendChar (uint8_t ch)
- int __io_putchar (int ch) __attribute__((weak))
- int __io_getchar (void)
- void initialise_monitor_handles ()
- int _getpid (void)
- int _kill (int pid, int sig)
- void _exit (int status)
- __attribute__ ((weak))
- int _close (int file)
- int _fstat (int file, struct stat ∗st)
- int _isatty (int file)
- int _lseek (int file, int ptr, int dir)
- int _open (char ∗path, int flags,...)
- int _wait (int ∗status)
- int _unlink (char ∗name)
- int _times (struct tms ∗buf)
- int _stat (char ∗file, struct stat ∗st)
- int _link (char ∗old, char ∗new)
- int _fork (void)
- int _execve (char ∗name, char ∗∗argv, char ∗∗env)

Variables

- char ∗∗ environ = __env

7.13.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

Attention

Definition in file syscalls.c.

7.13.2 Macro Definition Documentation

7.13.2.1 DEMCR

```
#define DEMCR ∗((volatile uint32_t∗) 0xE000EDFCU )
```
Definition at line 42 of file syscalls.c.

7.13.2.2 ITM_STIMULUS_PORT0

```
#define ITM_STIMULUS_PORT0 *((volatile uint32_t*) 0xE0000000 )
```

Definition at line 45 of file syscalls.c.

7.13.2.3 ITM_TRACE_EN

```
#define ITM_TRACE_EN *((volatile uint32_t*) 0xE0000E00 )
```

Definition at line 46 of file syscalls.c.

7.13.3 Function Documentation

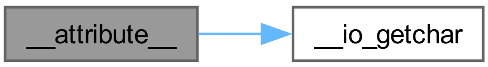
7.13.3.1 __attribute__()

```
__attribute__ (
              (weak) )
```

Definition at line 99 of file syscalls.c.

```
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }

  return len;
}
```

Here is the call graph for this function:



7.13.3.2 __io_getchar()

```
int __io_getchar (
              void ) [extern]
```

Definition at line 68 of file syscalls.c.

```
                  { 0 };
```

7.13.3.3 __io_putchar()

```
int __io_putchar (
              int ch) [extern]
```

7.13.3.4 _close()

```
int _close (
              int file)
```

Definition at line 125 of file syscalls.c.

```
{
  (void)file;
  return -1;
}
```

7.13.3.5 _execve()

```
int _execve (
            char * name,
            char ** argv,
            char ** env)
```

Definition at line 202 of file syscalls.c.

```
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

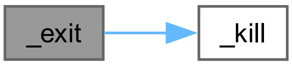
7.13.3.6 _exit()

```
void _exit (
            int status)
```

Definition at line 93 of file syscalls.c.

```
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
```

Here is the call graph for this function:



7.13.3.7 _fork()

```
int _fork (
            void )
```

Definition at line 196 of file syscalls.c.

```
{
  errno = EAGAIN;
  return -1;
}
```

7.13.3.8 _fstat()

```
int _fstat (
            int file,
            struct stat * st)
```

Definition at line 132 of file syscalls.c.

```
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
```

7.13.3.9 _getpid()

```
int _getpid (
            void )
```

Definition at line 80 of file syscalls.c.

```
00081 {
00082   return 1;
00083 }
```

7.13.3.10 _isatty()

```
int _isatty (
            int file)
```

Definition at line 139 of file syscalls.c.

```
00140 {
00141   (void)file;
00142   return 1;
00143 }
```

7.13.3.11 _kill()

```
int _kill (
            int pid,
            int sig)
```

Definition at line 85 of file syscalls.c.

```
00086 {
00087   (void)pid;
00088   (void)sig;
00089   errno = EINVAL;
00090   return -1;
00091 }
```

7.13.3.12 _link()

```
int _link (
            char * old,
            char * new)
```

Definition at line 188 of file syscalls.c.

```
00189 {
00190   (void)old;
00191   (void)new;
00192   errno = EMLINK;
00193   return -1;
00194 }
```

7.13.3.13 _lseek()

```
int _lseek (
            int file,
            int ptr,
            int dir)
```

Definition at line 145 of file syscalls.c.

```
00146 {
00147   (void)file;
00148   (void)ptr;
00149   (void)dir;
00150   return 0;
00151 }
```

7.13.3.14 _open()

```
int _open (
            char * path,
            int flags,
             ...)
```

Definition at line 153 of file syscalls.c.

```
00154 {
00155   (void)path;
00156   (void)flags;
00157   /* Pretend like we always fail */
00158   return -1;
00159 }
```

7.13.3.15 _stat()

```
int _stat (
        char * file,
        struct stat * st)
```

Definition at line 181 of file syscalls.c.

```
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
```

7.13.3.16 _times()

```
int _times (
        struct tms * buf)
```

Definition at line 175 of file syscalls.c.

```
{
  (void)buf;
  return -1;
}
```

7.13.3.17 _unlink()

```
int _unlink (
        char * name)
```

Definition at line 168 of file syscalls.c.

```
{
  (void)name;
  errno = ENOENT;
  return -1;
}
```

7.13.3.18 _wait()

```
int _wait (
        int * status)
```

Definition at line 161 of file syscalls.c.

```
{
  (void)status;
  errno = ECHILD;
  return -1;
}
```

7.13.3.19 initialise_monitor_handles()

```
void initialise_monitor_handles ()
```

Definition at line 76 of file syscalls.c.

```
{
}
```

7.13.3.20 ITM_SendChar()

```
void ITM_SendChar (
        uint8_t ch)
```

Definition at line 48 of file syscalls.c.

```
{

    //Enable TRCENA
    DEMCR |= ( 1 « 24);

    //enable stimulus port 0
    ITM_TRACE_EN |= ( 1 « 0);

    // read FIFO status in bit [0]:
    while(!(ITM_STIMULUS_PORT0 & 1));

    //Write to ITM stimulus port0
    ITM_STIMULUS_PORT0 = ch;
00062 }
```

7.13.4 Variable Documentation

7.13.4.1 environ

`char** environ = __env`

Definition at line 72 of file syscalls.c.

7.14 syscalls.c

Go to the documentation of this file.

```
00001
00022
00023 /* Includes */
00024 #include <sys/stat.h>
00025 #include <stdlib.h>
00026 #include <errno.h>
00027 #include <stdio.h>
00028 #include <signal.h>
00029 #include <time.h>
00030 #include <sys/time.h>
00031 #include <sys/times.h>
00032
00033
00035 //                  Implementation of printf like feature using ARM Cortex M3/M4/ ITM functionality
00036 //                  This function will not work for ARM Cortex M0/M0+
00037 //                  If you are using Cortex M0, then you can use semihosting feature of openOCD
00039
00040
00041 //Debug Exception and Monitor Control Register base address
00042 #define DEMCR                   *((volatile uint32_t*) 0xE000EDFCU )
00043
00044 /* ITM register addresses */
00045 #define ITM_STIMULUS_PORT0      *((volatile uint32_t*) 0xE0000000 )
00046 #define ITM_TRACE_EN            *((volatile uint32_t*) 0xE0000E00 )
00047
00048 void ITM_SendChar(uint8_t ch)
00049 {
00050
00051     //Enable TRCENA
00052     DEMCR |= ( 1 « 24);
00053
00054     //enable stimulus port 0
00055     ITM_TRACE_EN |= ( 1 « 0);
00056
00057     // read FIFO status in bit [0]:
00058     while(!(ITM_STIMULUS_PORT0 & 1));
00059
00060     //Write to ITM stimulus port0
00061     ITM_STIMULUS_PORT0 = ch;
00062 }
00063
00064
00065
00066 /* Variables */
00067 extern int __io_putchar(int ch) __attribute__((weak));
00068 extern int __io_getchar(void) __attribute__((weak));
00069
00070
00071 char *__env[1] = { 0 };
00072 char **environ = __env;
00073
00074
00075 /* Functions */
00076 void initialise_monitor_handles()
00077 {
00078 }
00079
00080 int _getpid(void)
00081 {
00082   return 1;
00083 }
00084
00085 int _kill(int pid, int sig)
00086 {
00087   (void)pid;
00088   (void)sig;
00089   errno = EINVAL;
00090   return -1;
00091 }
```

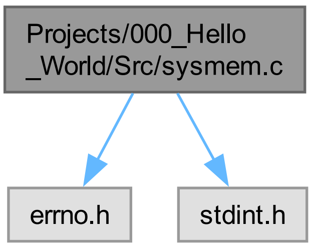
```

void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}

__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }

  return len;
}

__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
//    __io_putchar(*ptr++);
      ITM_SendChar(*ptr++);
  }
  return len;
}

int _close(int file)
{
  (void)file;
  return -1;
}


int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}

int _isatty(int file)
{
  (void)file;
  return 1;
}

int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}

int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}

int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}

int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}

int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}

int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}

int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}

int _fork(void)
{
  errno = EAGAIN;
  return -1;
}

int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

7.15 Projects/000_Hello_World/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

Include dependency graph for sysmem.c:



Functions

- void ∗ _sbrk (ptrdiff_t incr)

  _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

Variables

- static uint8_t ∗ __sbrk_heap_end = NULL

7.15.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

Definition in file sysmem.c.

7.15.2 Function Documentation

7.15.2.1 _sbrk()

```
void * _sbrk (
             ptrdiff_t incr)
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ############################################################################
* #   .data   #   .bss   #       newlib heap       #          MSP stack         #
* #           #          #                         # Reserved by _Min_Stack_Size #
* ############################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

Pointer to allocated memory

Definition at line 53 of file sysmem.c.

```
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
  const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;

  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }

  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }

  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;

  return (void *)prev_heap_end;
}
```

7.15.3 Variable Documentation

7.15.3.1 __sbrk_heap_end

`uint8_t* __sbrk_heap_end = NULL [static]`

Pointer to the current high watermark of the heap usage

Definition at line 30 of file sysmem.c.

7.16 sysmem.c

Go to the documentation of this file.

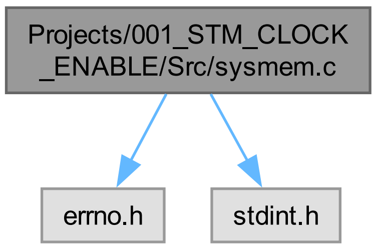
```
00001
00022
00023 /* Includes */
00024 #include <errno.h>
00025 #include <stdint.h>
00026
00030 static uint8_t *__sbrk_heap_end = NULL;
00031
00053 void *_sbrk(ptrdiff_t incr)
00054 {
00055   extern uint8_t _end; /* Symbol defined in the linker script */
00056   extern uint8_t _estack; /* Symbol defined in the linker script */
00057   extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
00058   const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
00059   const uint8_t *max_heap = (uint8_t *)stack_limit;
00060   uint8_t *prev_heap_end;
00061
00062   /* Initialize heap end at first call */
00063   if (NULL == __sbrk_heap_end)
00064   {
00065     __sbrk_heap_end = &_end;
00066   }
00067
00068   /* Protect heap from growing into the reserved MSP stack */
00069   if (__sbrk_heap_end + incr > max_heap)
00070   {
00071     errno = ENOMEM;
00072     return (void *)-1;
00073   }
00074
00075   prev_heap_end = __sbrk_heap_end;
00076   __sbrk_heap_end += incr;
00077
00078   return (void *)prev_heap_end;
00079 }
```

7.17 Projects/001_STM_CLOCK_ENABLE/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

Include dependency graph for sysmem.c:



Functions

- void * _sbrk (ptrdiff_t incr)

  _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

Variables

- static uint8_t * __sbrk_heap_end = NULL

7.17.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

Definition in file sysmem.c.

7.17.2 Function Documentation

7.17.2.1 _sbrk()

```
void * _sbrk (
            ptrdiff_t incr)
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ############################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack          #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ############################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

Pointer to allocated memory

Definition at line 53 of file sysmem.c.

```
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
  const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
00060   uint8_t *prev_heap_end;
00061
00062   /* Initialize heap end at first call */
00063   if (NULL == __sbrk_heap_end)
00064   {
00065     __sbrk_heap_end = &_end;
00066   }
00067
00068   /* Protect heap from growing into the reserved MSP stack */
00069   if (__sbrk_heap_end + incr > max_heap)
00070   {
00071     errno = ENOMEM;
00072     return (void *)-1;
00073   }
00074
00075   prev_heap_end = __sbrk_heap_end;
00076   __sbrk_heap_end += incr;
00077
00078   return (void *)prev_heap_end;
00079 }
```

7.17.3 Variable Documentation

7.17.3.1 __sbrk_heap_end

`uint8_t* __sbrk_heap_end = NULL  [static]`

Pointer to the current high watermark of the heap usage

Definition at line 30 of file sysmem.c.

7.18 sysmem.c

Go to the documentation of this file.

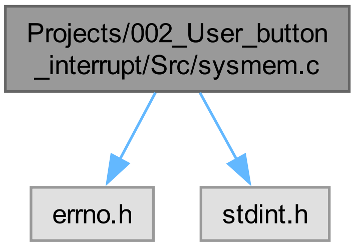
```
00001
00022
00023 /* Includes */
00024 #include <errno.h>
00025 #include <stdint.h>
00026
00030 static uint8_t *__sbrk_heap_end = NULL;
00031
00053 void *_sbrk(ptrdiff_t incr)
00054 {
00055   extern uint8_t _end; /* Symbol defined in the linker script */
00056   extern uint8_t _estack; /* Symbol defined in the linker script */
00057   extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
00058   const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
00059   const uint8_t *max_heap = (uint8_t *)stack_limit;
00060   uint8_t *prev_heap_end;
00061
00062   /* Initialize heap end at first call */
00063   if (NULL == __sbrk_heap_end)
00064   {
00065     __sbrk_heap_end = &_end;
00066   }
00067
00068   /* Protect heap from growing into the reserved MSP stack */
00069   if (__sbrk_heap_end + incr > max_heap)
00070   {
00071     errno = ENOMEM;
00072     return (void *)-1;
00073   }
00074
00075   prev_heap_end = __sbrk_heap_end;
00076   __sbrk_heap_end += incr;
00077
00078   return (void *)prev_heap_end;
00079 }
```

7.19 Projects/002_User_button_interrupt/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

Include dependency graph for sysmem.c:



Functions

- void ∗ _sbrk (ptrdiff_t incr)

    _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

Variables

- static uint8_t ∗ __sbrk_heap_end = NULL

7.19.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

Definition in file sysmem.c.

7.19.2 Function Documentation

7.19.2.1 _sbrk()

```
void * _sbrk (
            ptrdiff_t incr)
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ############################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack         #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ############################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

Pointer to allocated memory

Definition at line 53 of file sysmem.c.

```
00054 {
00055   extern uint8_t _end; /* Symbol defined in the linker script */
00056   extern uint8_t _estack; /* Symbol defined in the linker script */
00057   extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
00058   const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
00059   const uint8_t *max_heap = (uint8_t *)stack_limit;
00060   uint8_t *prev_heap_end;
00061
00062   /* Initialize heap end at first call */
00063   if (NULL == __sbrk_heap_end)
00064   {
00065     __sbrk_heap_end = &_end;
00066   }
00067
00068   /* Protect heap from growing into the reserved MSP stack */
00069   if (__sbrk_heap_end + incr > max_heap)
00070   {
00071     errno = ENOMEM;
00072     return (void *)-1;
00073   }
00074
00075   prev_heap_end = __sbrk_heap_end;
00076   __sbrk_heap_end += incr;
00077
00078   return (void *)prev_heap_end;
00079 }
```

7.19.3 Variable Documentation

7.19.3.1 __sbrk_heap_end

```
uint8_t* __sbrk_heap_end = NULL  [static]
```

Pointer to the current high watermark of the heap usage
Definition at line 30 of file sysmem.c.

7.20 sysmem.c

Go to the documentation of this file.

```
00001
00022
00023 /* Includes */
00024 #include <errno.h>
00025 #include <stdint.h>
00026
00030 static uint8_t *__sbrk_heap_end = NULL;
00031
00053 void *_sbrk(ptrdiff_t incr)
00054 {
00055   extern uint8_t _end; /* Symbol defined in the linker script */
00056   extern uint8_t _estack; /* Symbol defined in the linker script */
00057   extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
00058   const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
00059   const uint8_t *max_heap = (uint8_t *)stack_limit;
00060   uint8_t *prev_heap_end;
00061
00062   /* Initialize heap end at first call */
00063   if (NULL == __sbrk_heap_end)
00064   {
00065     __sbrk_heap_end = &_end;
```

```
  }

  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }

  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;

  return (void *)prev_heap_end;
}
```

7.21 README.md File Reference