

HPMicro 高性能 MCU

HPM SDK 使用指南

| | |
|----------------------------|----|
| 1 简介 | 4 |
| 2 开发前的准备 | 5 |
| 2.1 硬件环境准备 | 5 |
| 2.2 软件环境准备 | 7 |
| 2.3 开发调试环境搭建 | 8 |
| 3 HPM SDK 介绍 | 8 |
| 3.1 SDK ENV 组成结构 | 9 |
| 3.2 HPM SDK 核心软件包架构 | 10 |
| 4 HPM SDK 驱动模块简介 | 11 |
| 4.1 HPM SDK 驱动文件结构 | 11 |
| 4.2 HPM SDK 驱动 API | 13 |
| 4.3 HPM SDK 数据结构 | 15 |
| 4.4 HPM SDK 宏定义 | 17 |
| 4.5 HPM SDK 函数接口定义 | 17 |
| 4.6 HPM SDK API 应用示例 | 18 |
| 6 HPM SDK 中间件 | 19 |
| 7 HPM SDK 应用例程 | 19 |
| 8 HPM SDK 常见问题 | 25 |

版本:

| 日期 | 版本号 | 说明 |
|-----------|-----|----|
| 2022-4-25 | 1.0 | 初版 |
| | | |

1 简介

本文将为先楫半导体的高性能 MCU 配套的 HPM SDK（以下可简称 SDK）提供基本的说明，为用户提供 HPM SDK 入门指导。

HPM SDK 是一个基于宽松使用许可（BSD 3-Clause）完全开源的综合性软件支持包，可帮助用户在使用基于 RISC-V 内核的先楫半导体 MCU 时能简化和加快应用开发。

HPM SDK 包括第三方工具软件、硬件驱动软件、集成实时操作系统、中间件、参考例程以及说明文档等。HPM SDK 支持用 Cmake 为 Segger 和 GCC 提供示例项目，并同时集成进 RT Thread、Zephyr 等开源项目中。

用户在先楫半导体官网平台选择所需要的 MCU、评估板和可选组件后即可下载获得对应的 SDK 支持包。

HPM SDK 主要由以下组件组成：

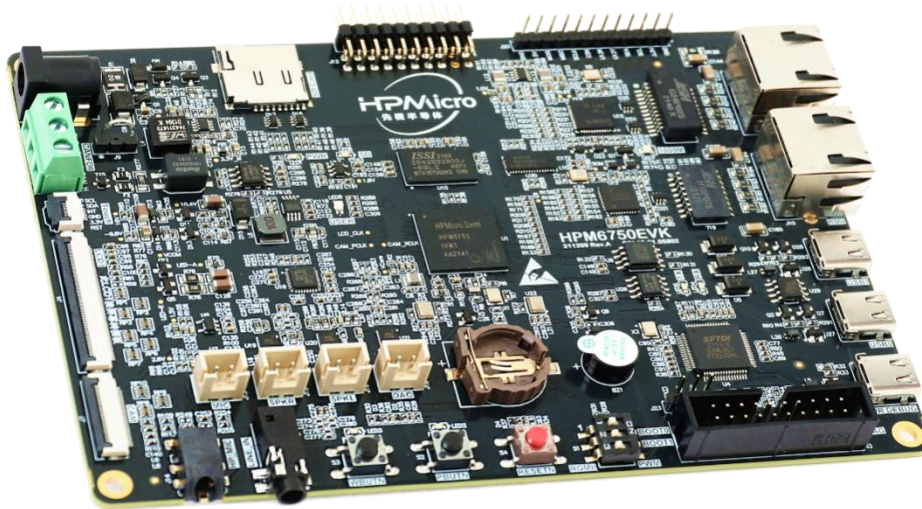
- RISC-V 启动文件和设备头文件及标准库。
- 开源外设驱动程序提供无状态、高性能、易于使用的 API。
- 通信外设的驱动程序还包括用于高性能数据传输的高级通信 API。
- 预集成的实时操作系统(RTOS)内核：FreeRTOS。
- 先楫半导体和合作伙伴提供的软件支持组件，包括：
 - 图形和人机界面
 - 音频和语音处理
 - 电机控制
 - 信息安全
 - 存储
 - 无线和有线通信
- 展示外设驱动程序、实时操作系统包驱动程序、中间件和 RTOS 的用法的示例软件

2 开发前的准备

2.1 硬件环境准备

(1) HPM67xx 系列 MCU 的开发板: HPM6750 EVK 开发板和 HPM6750 EVK mini 开发板。

HPM6750EVK 开发板如下图:

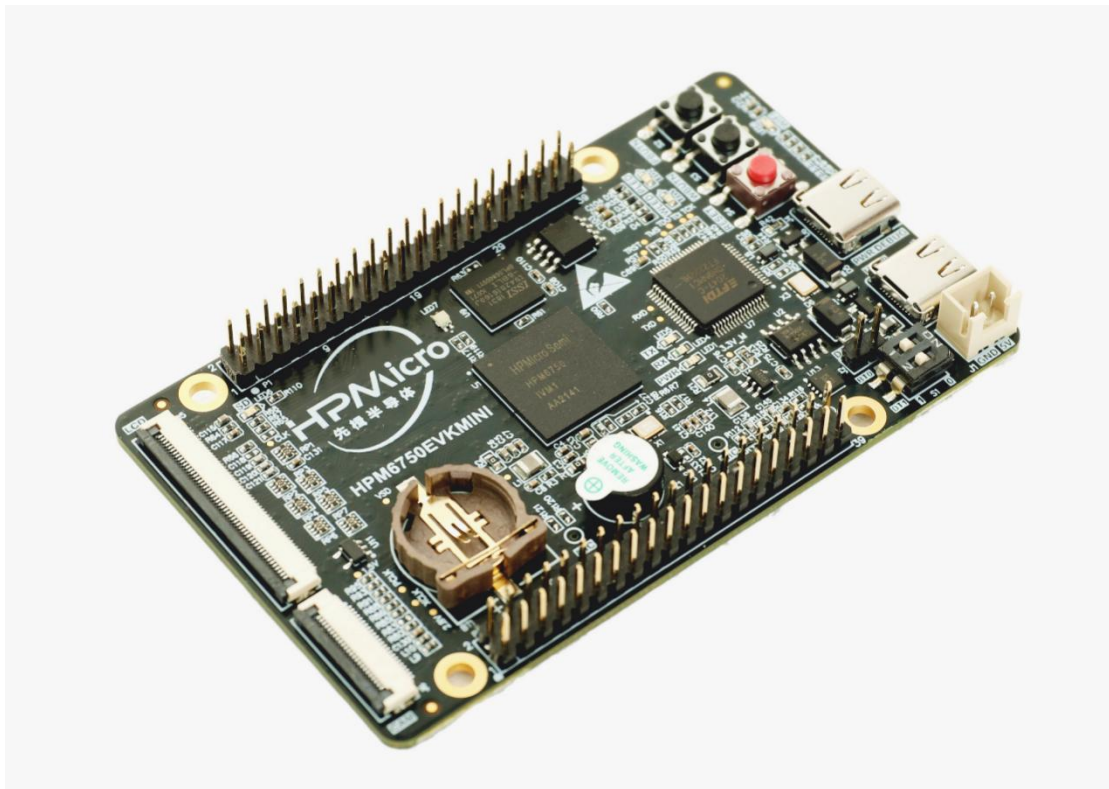


HPM6750EVK 板上硬件资源:

- HPM6750IVM 微控制器 (主频 816Mhz, 2MB 片上内存)
- 板载存储
 - 256Mb SDRAM
 - 128Mb Quad SPI NOR Flash
- 显示/摄像头
 - LCD 接口
 - 摄像头(DVP)接口
- 以太网
 - 1000 Mbits PHY
 - 100 Mbits PHY
- USB
 - USB type C (USB 2.0 OTG) connector x3

- 音频
 - Line in
 - Mic
 - Speaker
 - DAO
- 其他
 - TF 卡槽
 - FT2232
 - 蜂鸣器
 - RGB LED
 - CAN
- 扩展口
 - 电机控制

HPM6750 EVK mini 开发板如下图:



HPM6750EVKmini 板上硬件资源:

- HPM6750IVM 微控制器 (主频 816Mhz, 2MB 片上内存)
- 板载存储
 - 128Mb SDRAM

- 64Mb Quad SPI NOR Flash
- 显示/摄像头
 - LCD 接口
 - 摄像头(DVP)接口
- WiFi
 - RW007
- USB
 - USB type C (USB 2.0 OTG) connector x2
- 音频
 - Mic
 - DAO
- 其他
 - TF 卡槽
 - FT2232
 - 蜂鸣器
 - RGB LED
- 扩展口
 - ART-PI

(2) 一条 USB-TypeC 线缆，用于给 HPM6750 EVK mini 开发板供电，以及调试。

(3) 一台 PC 主机或者笔记本电脑。

2.2 软件环境准备

为了更好的进行 HPM 6700 系列 MCU 的开发，建议您应当至少具备以下开发环境：

- Windows 使用 Windows 7 以上版本。
- Linux 建议使用 Ubuntu 18 以上的 LTS 版本。
- 安装 Segger Embedded Studio，使用 RISC_V_v620a_win_x64 或者以上版本。
- 运行 sdk_env\tools\FTDI_InstallDriver.exe 以安装可用于调试的 FT2232 驱动。

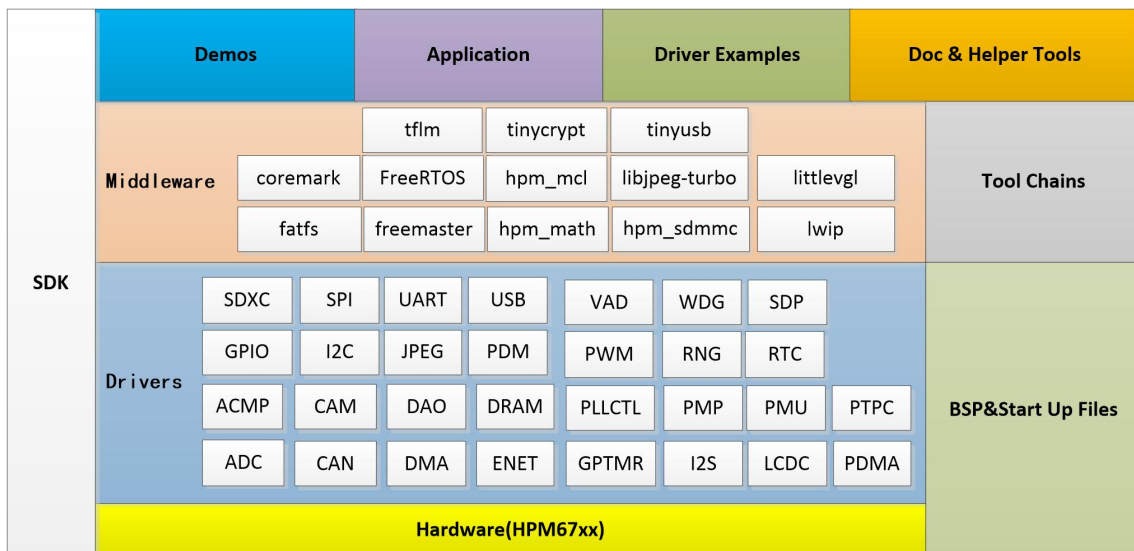
2.3 开发调试环境搭建

具体开发板的连接, 请参考硬件连接指南部分, 请确保开发板被正确设置后再进行下面的步骤软件开发调试, 以 HPM6750EVKMini 为例, 开发环境连接示意如下:



3 HPM SDK 介绍

HPM 软件开发工具包(HPM SDK)是一套用于先楫半导体微控制器的软件, 包括底层外设驱动程序、组件、中间件和集成的 RTOS 支持。除了这些组件之外, HPM SDK 还提供了相关的演示和示例, 以及文档, 以帮助用户评估 HPM 产品并有效地构建他们的应用程序。



Hardware: 基于 RISC-V 内核的高性能 MCU HPM6750, 该芯片拥有最大 2M 字节的连续片上 RAM, 并集成了丰富的存储接口, 如 SDRAM, Quad SPI NOR

flash, SD/eMMC 卡。同时它也提供多种音视频接口包括 LCD 显示, 像素 DMA, 摄像头以及 I2S 音频接口。

Drivers: 抽象访问、操作硬件寄存器的 C 函数驱动接口。

Middleware: 在驱动程序的基础上增加功能, 例如通信栈、图形库、通用算法、操作系统等。

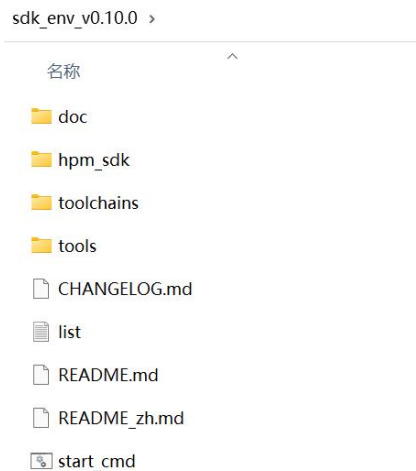
通过浏览器打开以下文件, 可快速方便的查询 HPM SDK 的详细介绍。

[" sdk_env_v0.10.0/hpm_sdk/doc/output/sdk_doc/zh/latest/html/doc/README_zh.html"](https://sdk_env_v0.10.0/hpm_sdk/doc/output/sdk_doc/zh/latest/html/doc/README_zh.html)



3.1 SDK ENV 组成结构

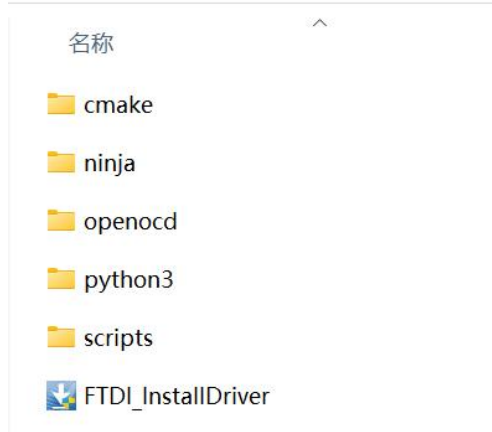
整个 SDK ENV 组成结构如下图所示:



- **doc**: 主要是先楫半导体官方开发的用户指导手册, 包含 HPM6750EVK 和 HPM6750EVK mini 的用户指导手册。
- **hpm_sdk**: HPM SDK 的核心软件包。

- **toolchains** : 编译工具链, 主要为 rv32imadc-ilp32d-x86_64-w64-mingw32。
- **tools**:主要第三方软件和相应的处理脚本, 如下图:

sdk_env_v0.10.0 > tools

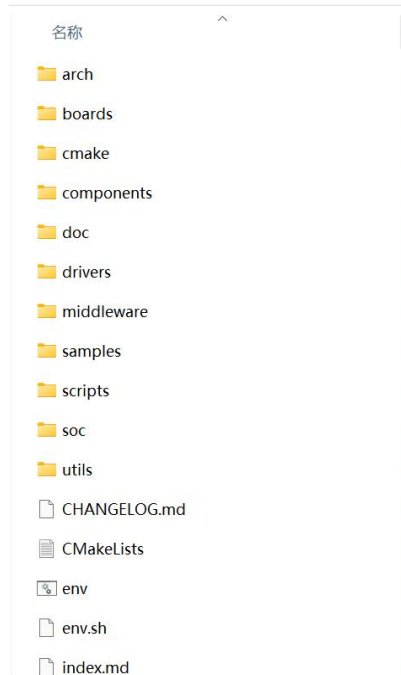


- **其他**: SDK 运行环境配置脚本和说明文件。

3.2 HPM SDK 核心软件包架构

SDK 核心软件包主要是支持完成应用软件开发的各种组件, 包含了驱动、板级支持文件、中间件、soc 定义文件、实例、处理脚步、帮助文件等。如下图所示

sdk_env_v0.10.0 > hpm_sdk >



各个目录的主要内容:

- **arch**: 与 risc-v 架构相关的操作接口。

- **boards:** 与开发板相关的硬件定义和接口，主要为 HPM6750 EVK 和 HPM6750 EVK mini 相关的板卡硬件信息。
- **cmake:** 与 cmake 和工程管理相关配置文件。
- **components:** 常用板载模块驱动、处理逻辑和控制接口，包括 camera、codec、debug_console、enet_phy、serial_nor、touch、usb 等。
- **doc:** SDK 支持文档和软件 API 的支持文档。
- **drivers:** soc 片上硬件模块驱动、处理逻辑和控制接口。
- **middleware:** 常见算法库、图形库、组件库、RTOS 等。
- **samples:** 各种应用例程。
- **script:** 与工程管理、编译相关的处理脚本。
- **soc:** soc 片上资源寄存器定义。
- **utils:** 其他通用处理组件。
- **其他:** 处理脚步和说明文件。

4 HPM SDK 驱动模块简介

4.1 HPM SDK 驱动文件结构

HPM SDK 的驱动模块目录结构如下图所示，包括了头文件目录 inc 和源代码目录 src。

| | drivers |
|-----|----------------------|
| 1. | └inc |
| 2. | hpm_acmp_drv.h |
| 3. | hpm_adc12_drv.h |
| 4. | hpm_adc16_drv.h |
| 5. | hpm_bacc_drv.h |
| 6. | hpm_bkey_drv.h |
| 7. | hpm_butn_drv.h |
| 8. | hpm_cam_drv.h |
| 9. | hpm_can_drv.h |
| 10. | hpm_common.h |
| 11. | hpm_dao_drv.h |
| 12. | hpm_display_common.h |
| 13. | hpm_dmamux_drv.h |
| 14. | hpm_dma_drv.h |
| 15. | hpm_dram_drv.h |
| 16. | hpm_enet_drv.h |
| 17. | hpm_gpiom_drv.h |

| | | |
|-----|--|--------------------------|
| 18. | | hpm_gpio_drv.h |
| 19. | | hpm_gptmr_drv.h |
| 20. | | hpm_hall_drv.h |
| 21. | | hpm_i2c_drv.h |
| 22. | | hpm_i2s_common.h |
| 23. | | hpm_i2s_drv.h |
| 24. | | hpm_jpeg_drv.h |
| 25. | | hpm_lcdc_drv.h |
| 26. | | hpm_mbx_drv.h |
| 27. | | hpm_mchtmr_drv.h |
| 28. | | hpm_mono_drv.h |
| 29. | | hpm_pcfg_drv.h |
| 30. | | hpm_pdma_drv.h |
| 31. | | hpm_pdm_drv.h |
| 32. | | hpm_pllctl_drv.h |
| 33. | | hpm_pmon_drv.h |
| 34. | | hpm_pmp_drv.h |
| 35. | | hpm_ppor_drv.h |
| 36. | | hpm_psec_drv.h |
| 37. | | hpm_ptpc_drv.h |
| 38. | | hpm_pwm_drv.h |
| 39. | | hpm_qei_drv.h |
| 40. | | hpm_rng_drv.h |
| 41. | | hpm_romapi_xpi_def.h |
| 42. | | hpm_romapi_xpi_nor_def.h |
| 43. | | hpm_romapi_xpi_ram_def.h |
| 44. | | hpm_rtc_drv.h |
| 45. | | hpm_sdp_drv.h |
| 46. | | hpm_sdxs_drv.h |
| 47. | | hpm_spi_drv.h |
| 48. | | hpm_synt_drv.h |
| 49. | | hpm_trgm_drv.h |
| 50. | | hpm_uart_drv.h |
| 51. | | hpm_usb_drv.h |
| 52. | | hpm_vad_drv.h |
| 53. | | hpm_wdg_drv.h |
| 54. | | |
| 55. | | ↳src |
| 56. | | hpm_acmp_drv.c |
| 57. | | hpm_adc12_drv.c |
| 58. | | hpm_adc16_drv.c |
| 59. | | hpm_cam_drv.c |
| 60. | | hpm_can_drv.c |
| 61. | | hpm_dao_drv.c |

| | |
|-----|------------------|
| 62. | hpm_dma_drv.c |
| 63. | hpm_dram_drv.c |
| 64. | hpm_enet_drv.c |
| 65. | hpm_gpio_drv.c |
| 66. | hpm_gptmr_drv.c |
| 67. | hpm_i2c_drv.c |
| 68. | hpm_i2s_drv.c |
| 69. | hpm_jpeg_drv.c |
| 70. | hpm_lcdc_drv.c |
| 71. | hpm_pdma_drv.c |
| 72. | hpm_pdm_drv.c |
| 73. | hpm_pllctl_drv.c |
| 74. | hpm_pmp_drv.c |
| 75. | hpm_pmu_drv.c |
| 76. | hpm_ptpc_drv.c |
| 77. | hpm_pwm_drv.c |
| 78. | hpm_rng_drv.c |
| 79. | hpm_rtc_drv.c |
| 80. | hpm_sdp_drv.c |
| 81. | hpm_sdx_c_drv.c |
| 82. | hpm_spi_drv.c |
| 83. | hpm_uart_drv.c |
| 84. | hpm_usb_drv.c |
| 85. | hpm_vad_drv.c |
| 86. | hpm_wdg_drv.c |

soc 的外设(peripheral device)驱动源代码在 “\hpm_sdk\drivers\src” 目录下, 每一个外设(如 CAN、SPI、UART 等)为一个独立的源文件, 文件命名形式:

hpm_<peripheral device>_drv.c

soc 的外设(peripheral device)驱动头文件在 “\hpm_sdk\drivers\inc” 目录下, 每一个外设(如 CAN、SPI、UART 等)为一个独立的头文件, 文件命名形式:

hpm_<peripheral device>_drv.h

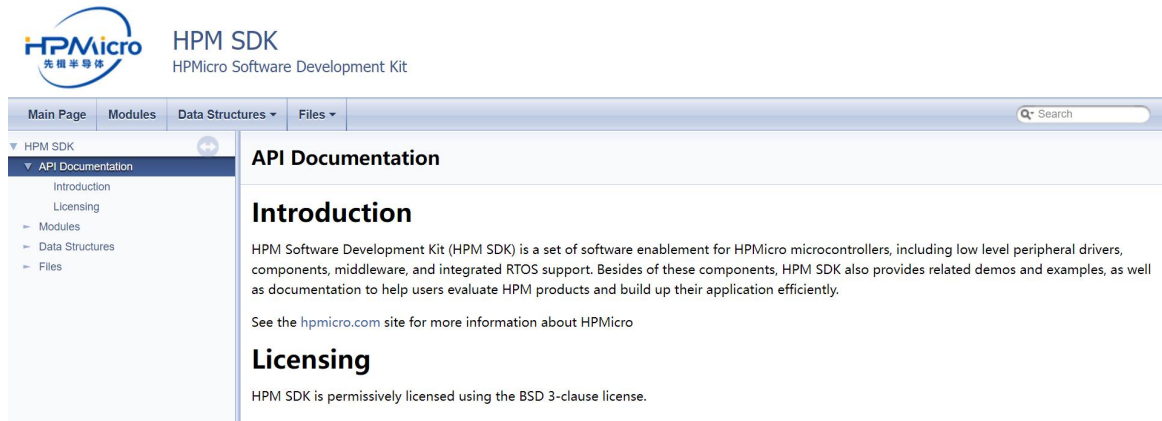
4.2 HPM SDK 驱动 API

HPM SDK 提供了丰富的外设接口, 主要的外设 API 见下表:

| | |
|----------------------------|------------------------|
| ACMP driver APIs | LCD driver APIs |
| ADC12 driver APIs | MBX driver APIs |
| ADC16 driver APIs | MCHTMR driver APIs |
| BACC driver APIs | MONO driver APIs |
| BKEY driver APIs | OTP driver APIs |
| BUTN driver APIs | PDM driver APIs |
| CAM driver APIs | PDMA driver APIs |
| CAN driver APIs | PLIC driver APIs |
| CLOCK driver APIs | PLLCTL driver APIs |
| COMMON driver APIs | PTPC driver APIs |
| DAO driver APIs | PWM driver APIs |
| DMA driver APIs | QEI driver APIs |
| DMAMUX driver APIs | ROM APIs |
| DRAM driver APIs | RTC driver APIs |
| Display_common driver APIs | SDP driver APIs |
| Enet driver APIs | SDXC driver APIs |
| GPIO driver APIs | SPI driver APIs |
| GPIOM driver APIs | SYSCTL driver APIs |
| GPTMR driver APIs | TRGM driver APIs |
| HALL driver APIs | UART driver APIs |
| I2C driver APIs | USB driver APIs |
| I2S common driver APIs | VAD driver APIs |
| I2S driver APIs | WDG driver APIs |
| INTERRUPT driver APIs | XPI NOR driver APIs |
| JPEG driver APIs | XPI RAM driver APIs |
| L1CACHE driver APIs | <i>XPI driver APIs</i> |

通过浏览器可以打开以下文件，方便查询 HPM SDK 中 API 相关的介绍。

["sdk_env_v0.10.0/hpm_sdk/doc/output/api_doc/html/index.html"](http://sdk_env_v0.10.0/hpm_sdk/doc/output/api_doc/html/index.html)



4.3 HPM SDK 数据结构

HPM SDK 中数据结构具备按照以下规则命名：

- (1) 外设模块寄存器数据结构定义

外设模块名_Type

比如 ADC 的寄存器定义如下：

```

1.  typedef struct {
2.     __RW uint32_t CONFIG[12]; /* 0x0 - 0x2C: */
3.     __RW uint32_t TRG_DMA_ADDR; /* 0x30: */
4.     __R uint8_t RESERVED0[972]; /* 0x34 - 0x3FF: Reserved */
5.     __R uint32_t BUS_RESULT[19]; /* 0x400 - 0x448: */
6.     __R uint8_t RESERVED1[180]; /* 0x44C - 0x4FF: Reserved */
7.     __RW uint32_t BUF_CFG0; /* 0x500: */
8.     __R uint8_t RESERVED2[764]; /* 0x504 - 0x7FF: Reserved */
9.     __RW uint32_t SEQ_CFG0; /* 0x800: */
10.    __RW uint32_t SEQ_DMA_ADDR; /* 0x804: */
11.    __R uint32_t SEQ_WR_ADDR; /* 0x808: */
12.    __RW uint32_t SEQ_DMA_CFG; /* 0x80C: */
13.    __RW uint32_t SEQ_QUE[16]; /* 0x810 - 0x84C: */
14.    __R uint8_t RESERVED3[944]; /* 0x850 - 0xBFF: Reserved */
15.    struct {
16.        __RW uint32_t PRD_CFG; /* 0xC00: */
17.        __RW uint32_t PRD_THSHD_CFG; /* 0xC04: */
18.        __R uint32_t PRD_RESULT; /* 0xC08: */
19.        __R uint8_t RESERVED0[4]; /* 0xC0C - 0xC0F: Reserved */
20.    } PRD_CFG[19];
21.    __R uint8_t RESERVED4[720]; /* 0xD30 - 0xFFF: Reserved */
22.    __RW uint32_t SAMPLE_CFG[19]; /* 0x1000 - 0x1048: */
23.    __R uint8_t RESERVED5[184]; /* 0x104C - 0x1103: Reserved */
24.    __RW uint32_t CONV_CFG1; /* 0x1104: */
    
```

```
25.  __RW uint32_t ADC_CFG0; /* 0x1108: */
26.  __R uint8_t RESERVED6[4]; /* 0x110C - 0x110F: Reserved */
27.  __RW uint32_t INT_STS; /* 0x1110: */
28.  __RW uint32_t INT_EN; /* 0x1114: */
29.  __R uint8_t RESERVED7[232]; /* 0x1118 - 0x11FF: Reserved */
30.  __RW uint32_t ANA_CTRL0; /* 0x1200: */
31.  __RW uint32_t ANA_CTRL1; /* 0x1204: */
32.  __R uint8_t RESERVED8[8]; /* 0x1208 - 0x120F: Reserved */
33.  __RW uint32_t ANA_STATUS; /* 0x1210: */
34. } ADC12_Type;
```

(2) 外设模块通用配置数据结构定义

外设模块名_config_t

比如 ADC 的通用配置数据结构定义如下:

```
1. typedef struct {
2.     uint8_t ch;
3.     uint8_t diff_sel;
4.     uint8_t res;
5.     uint8_t sample_cycle_shift;
6.     uint8_t conv_mode;
7.     uint8_t wait_dis;
8.     uint16_t thshdh;
9.     uint16_t thshdl;
10.    uint32_t sample_cycle;
11.    uint32_t adc_clk_div;
12.    bool sel_sync_ahb;
13.    bool adc_ahb_en;
14. } adc12_config_t;
```

(3) 外设模块工作模式数据结构定义

外设模块名_模式_config_t

比如 CAN 的 filter 配置数据结构定义如下:

```
1. typedef struct {
2.     uint16_t index;
3.     can_filter_mode_t mode;
4.     bool enable;
5.     uint32_t code;
6.     uint32_t mask;
7. } can_filter_config_t;
```


各个外设模块的配置方式不同，对应的数据结构也会有相应的差异，具体各个外设模块的数据结构定义与说明可以参考 SDK 的说明指导文档，文档路径：

["sdk_env_v0.10.0/hpm_sdk/doc/output/api_doc/html/annotated.html"](sdk_env_v0.10.0/hpm_sdk/doc/output/api_doc/html/annotated.html)

注意：所有硬件外设相关的命名和数据手册的寄存器命名一致，地址偏移一致，请勿随意修改。

4.4 HPM SDK 宏定义

(1) 外设模块工作模式数据结构定义

`#define 外设模块名_操作 外设模块名_操作`

比如 CAN 的事件宏定义如下：

```
1. #define CAN_EVENT_RECEIVE (CAN_RTIF_RIF_MASK)
2. #define CAN_EVENT_RX_BUF_OVERRUN (CAN_RTIF_ROIF_MASK)
3. #define CAN_EVENT_RX_BUF_FULL (CAN_RTIF_RFIF_MASK)
4. #define CAN_EVENT_RX_BUF_ALMOST_FULL (CAN_RTIF RAFIF_MASK)
5. #define CAN_EVENT_TX_PRIMARY_BUF (CAN_RTIF_TPIF_MASK)
6. #define CAN_EVENT_TX_SECONDARY_BUF (CAN_RTIF_TSIF_MASK)
7. #define CAN_EVENT_ERROR (CAN_RTIF_EIF_MASK)
8. #define CAN_EVENT_ABORT (CAN_RTIF_AIF_MASK )
```

4.5 HPM SDK 函数接口定义

HPM SDK 中函数的命名基本使用英文缩写来表达函数的完成功能，遵循动宾结构的命名法则，函数名中动词在前，并在命名前加入函数的前缀，外设模块操作函数接口定义规则如下

`函数返回值 外设模块名_目标词_动词_宾语(函数参数)`

比如 RTC 的函数定义如下：

```
1. RTC Functions
2. hpm_stat_t rtc_config_time (RTC_Type *base, time_t time)
3.     Configure the RTC time.
4.
5. hpm_stat_t rtc_config_alarm (RTC_Type *base, rtc_alarm_config_t *config)
6.     Configure RTC Alarm.
7.
8. time_t rtc_get_time (RTC_Type *base)
9.     Get the time returned by RTC module.
10.
```

```
11. static void rtc_enable_alarm_interrupt (RTC_Type *base, uint32_t index, bool enable)
12.     Enable RTC alarm interrupt.
13.
14. static void rtc_clear_alarm_flag (RTC_Type *base, uint32_t index)
15.     Clear RTC alarm flag.
16.
17. static bool rtc_is_alarm_flag_asserted (RTC_Type *base, uint32_t index)
18.     Check whether RTC alarm flag is set or not.
```

4.6 HPM SDK API 应用示例

下面以 PWM 的初始化为示例，说明驱动模块的数据结构以及 API 的使用方式：

```
1.     pwm_cmp_config_t cmp_config = {0 };
2.     pwm_config_t pwm_config = {0};
3.
4.     pwm_stop_counter(ptr);
5.     pwm_get_default_pwm_config(ptr, &pwm_config);
6.     pwm_get_default_cmp_config(ptr, &cmp_config);
7.
8.     pwm_config.enable_output = false;
9.     pwm_config.dead_zone_in_half_cycle = 0;
10.    pwm_config.invert_output = false;
11.
12.    /* reload and start counter */
13.    pwm_set_reload(ptr, 0, reload);
14.    pwm_set_start_count(ptr, 0, 0);
15.
16.    cmp_config.mode = pwm_cmp_mode_output_compare;
17.    cmp_config.cmp = cmp_initial_zero ? 0 : reload + 1;
18.    cmp_config.update_trigger = pwm_shadow_register_update_on_modify;
19.
20.    /* config initial compare value which should take affect immediately */
21.    pwm_config_cmp(ptr, cmp_index, &cmp_config);
22.
23.    /* update trigger type so that compare value will be updated on hardware event (RELOAD) */
24.    cmp_config.update_trigger = pwm_shadow_register_update_on_hw_event;
```

```

25. /* * config pwm as output driven by cmp */
26.   if (status_success != pwm_setup_waveform(ptr, pin, &pwm_config, c
      mp_index, &cmp_config, 1)) {
27.       printf("failed to setup waveform\n");
28.       while(1);
29.   }
30.
31.   /* * config hw event */
32.   cmp_config.cmp = reload - 1;
33.   cmp_config.update_trigger = pwm_shadow_register_update_on_hw_even
      t;
34.   pwm_load_cmp_shadow_on_match(ptr, hw_event_cmp, &cmp_config);

```

6 HPM SDK 中间件

HPM SDK 提供大部分通用的中间件，支持用户更便捷的完成更多复杂的功能，HPM SDK 提供的中间如下：



7 HPM SDK 应用例程

HPM SDK 提供了丰富应用例程，帮助用户快速熟悉、掌握 SDK 内容，包括驱动、API、开发环境的使用方法。

| 例程 | 描述 |
|-------------|--------------------------------|
| audio_codec | 音频解码例程 |
| coremark | Coremark 例程，获取 CoreMark 性能评分 |
| dhrystone | Dhrystone 例程，获取 Dhrystone 性能评分 |
| drivers | 包含了 SDK 所有驱动的应用参考例程 |
| dsp | DSP 功能相关的应用例程 |
| freemaster | 嵌入式系统数据可视化工具 Freemaster 例程 |

| | |
|--------------|-----------------------------|
| hello_world | 第一个 SDK 使用入门例程 |
| jpeg | jpeg 功能相关的应用例程 |
| littlevgl | littlevgl 功能相关的应用例程 |
| lwip | Lwip 功能相关的应用例程 |
| motor_ctrl | 电机控制功能相关的应用例程 |
| multicore | 双核同步运行功能相关的应用例程 |
| openocd_algo | openocd 功能相关的应用例程 |
| rgb_led | RGB 流水灯功能相关的应用例程 |
| rom_api | ROM 操作功能相关的应用例程 |
| rtos | FreeRTOS 相关的应用例程 |
| tflm | TensorflowLiteMicro 相关的应用例程 |
| tinycrypt | TinyCrypt 加密工具相关的应用例程 |
| tinyusb | tinyusb 功能相关的应用例程 |
| touch_panel | 触摸功能相关的应用例程 |

HPM SDK 中各个例程的描述也可以在路径找到：

["hpm_sdk/doc/output/sdk_doc/zh/latest/html/samples/index_zh.html"](http://hpm_sdk/doc/output/sdk_doc/zh/latest/html/samples/index_zh.html)

The screenshot shows the HPM SDK documentation website. On the left, there is a dark blue sidebar with the HPMicro logo and a search bar. Below the search bar, a list of samples is displayed: 1. Hello World, 2. Audio Codec, 3. Core Mark, 4. Touch Panel, 5. RTOS 例程, 6. 驱动, 7. Freemaster, 8. 电机控制, 9. JPEG, 10. Tinycrypt. On the right, the main content area shows the 'Hello World' sample details, including a sub-list: 1.1. 概述, 1.2. 硬件设置, 1.3. 运行现象.

下面以 SDK 中 “rgb_led” 例程，来示范 SDK 的应用例程的使用方法，依

据《HPM6750EVKMINI_UG》中的章节“3.4 sdk_env/Segger Embedded Studio For RISC V 使用快速指南”中操作步骤，可以很便捷完成新建“rgb_led”工程。

(1) 新建“rgb_led”工程

首先通过点击 `start_cmd.bat` 文件来运行命令行，并进入到路径“`sdk_env_v0.10.0\hpm_sdk\samples\rgb_led`”，然后再命令行窗口执行命令“`generate_project -b hpm6750evkmini`”，就会生成 SES 工程。

```

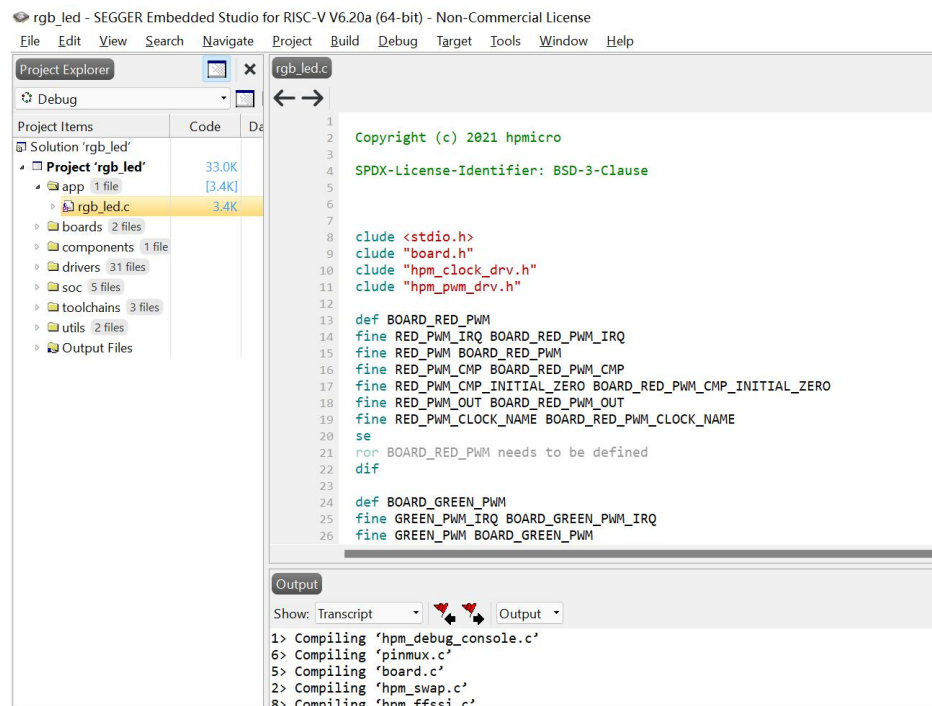
Microsoft Windows [版本 10.0.22000.613]
(c) Microsoft Corporation. 保留所有权利。

E:\sdk_env_v0.10.0>cd hpm_sdk\samples\rgb_led

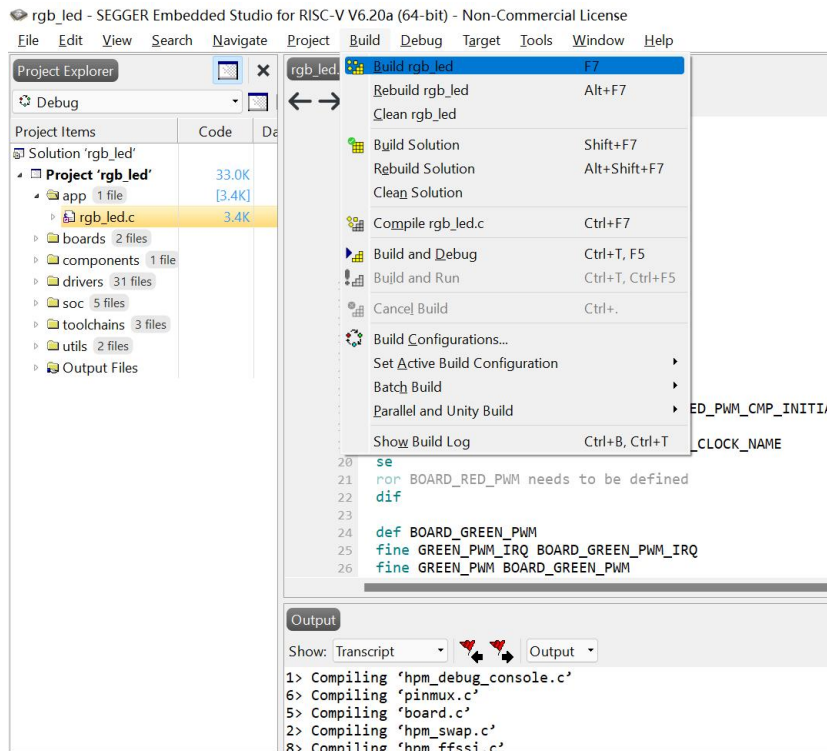
E:\sdk_env_v0.10.0\hpm_sdk\samples\rgb_led>generate_project -b hpm6750evkmini
hpm6750evkmini
-- Application: E:/sdk_env_v0.10.0/hpm_sdk/samples/rgb_led
-- Board: hpm6750evkmini
-- Found toolchain: gnu (E:/sdk_env_v0.10.0/toolchains/rv32imadc-rlp32d-x86_64-w64-mingw32)
-- The C compiler identification is GNU 11.1.0
-- The CXX compiler identification is GNU 11.1.0
-- The ASM compiler identification is GNU
-- Found assembler: E:/sdk_env_v0.10.0/toolchains/rv32imadc-rlp32d-x86_64-w64-mingw32/bin/riscv32-unknown-elf-gcc.exe
-- Segger linker script: E:/sdk_env_v0.10.0/hpm_sdk/soc/HPM6750/toolchains/segger/ram.icf
-- Segger Embedded Studio Project: E:/sdk_env_v0.10.0/hpm_sdk/samples/rgb_led/hpm6750evkmini_build/segger_embedded_studio/rgb_led.emProject
-- Configuring done
-- Generating done
-- Build files have been written to: E:/sdk_env_v0.10.0/hpm_sdk/samples/rgb_led/hpm6750evkmini_build
    
```

(2) 编译“rgb_led”工程

进入“`rgb_led\hpm6750evkmini_build\segger_embedded_studio`”，然后，双击“`rgb_led.emProject`”文件，可以打开“`rgb_led`”的 SES 工程。使用 Segger Embedded Studio 打开 `rgb_led` 工程即可进行编译。

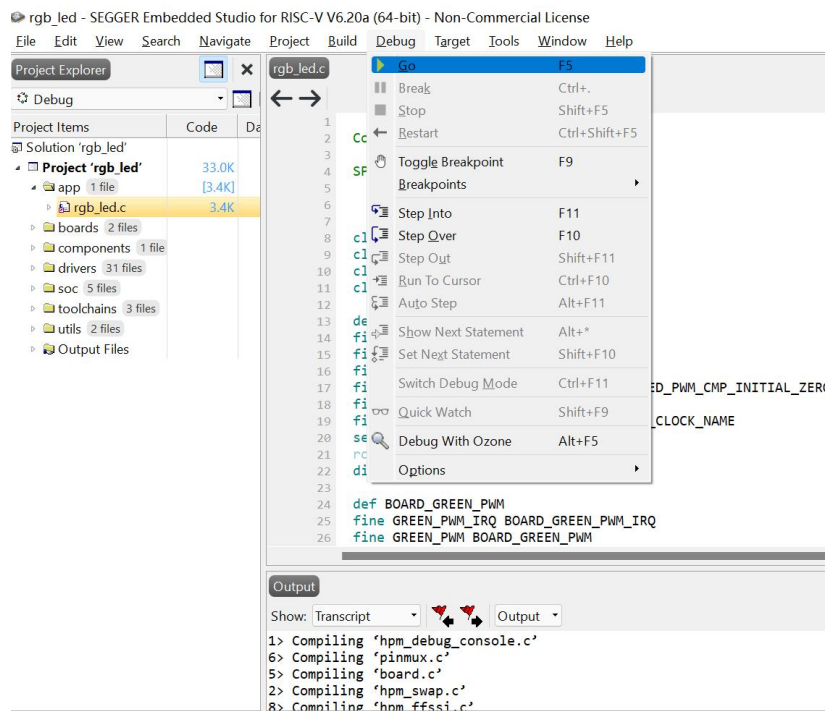


通过“`Project>Build reb_led`”或者用快捷键 `F7` 来编译工程。



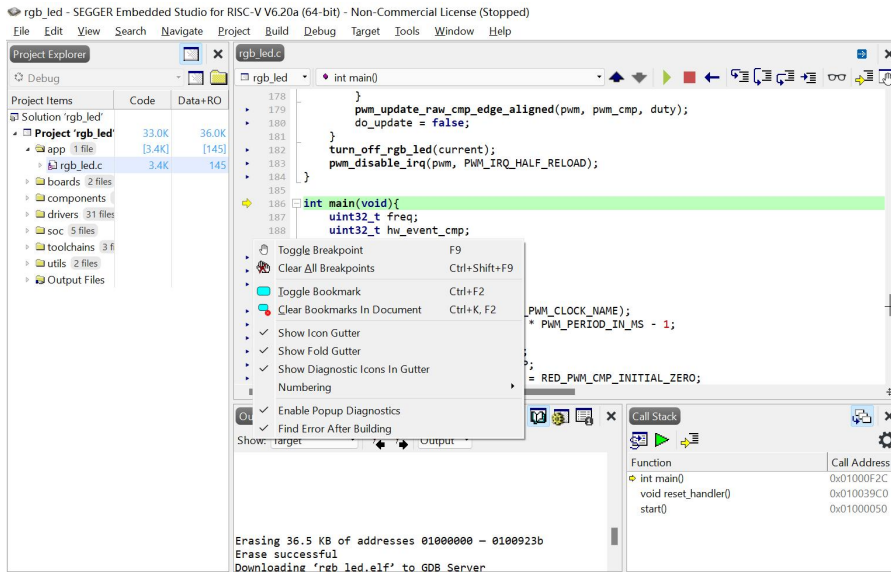
(3) 下载调试 “rgb_led” 工程

使用 Segger Embedded Studio 进行 rgb_led 调试。通过 “Debug>Go” 或者用快捷键 **F5** 来调试工程。



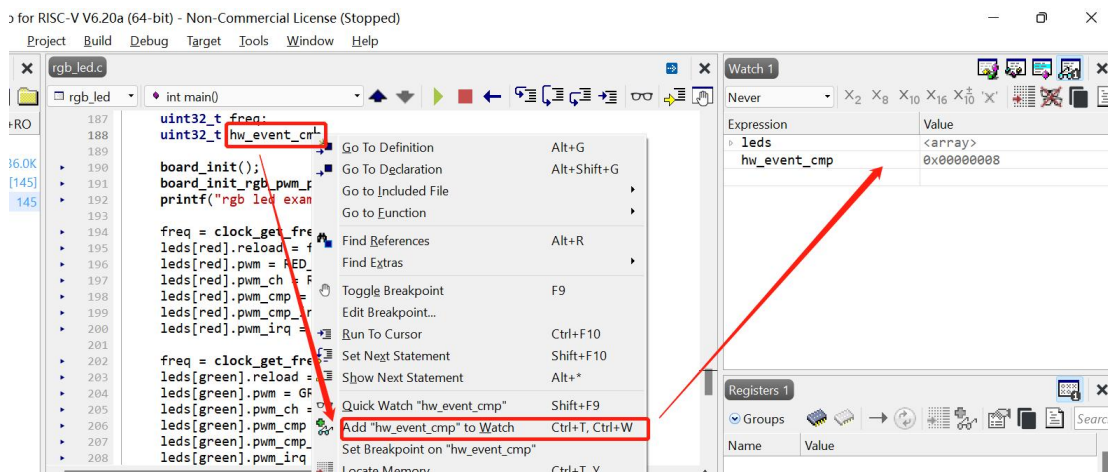
(4) 调试 “rgb_led” 工程：断点使用。

在相应的文件编辑器窗口，通过在相应行点击鼠标右键，选中菜单项 “Toggle Breakpoint” 或者使用 **F9**，可以打上断点。

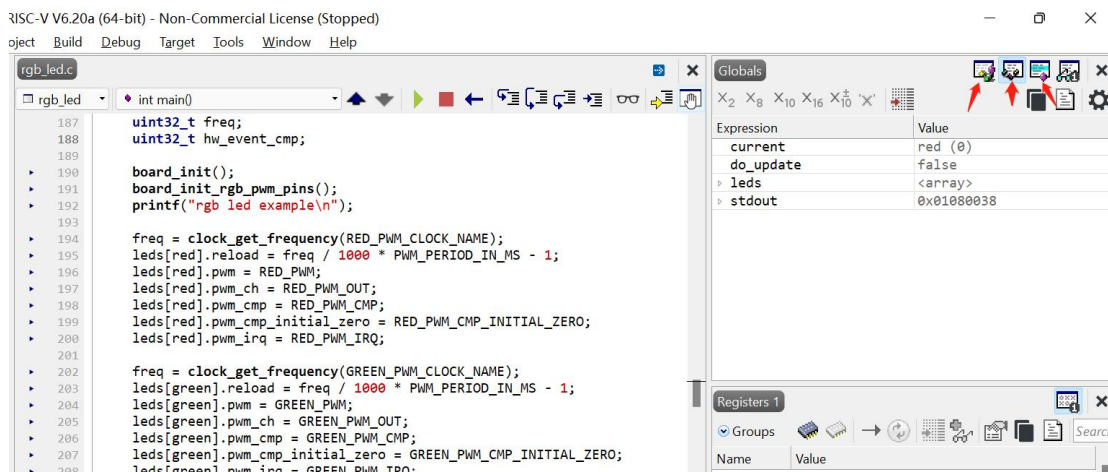


(5) 调试“rgb_led”工程：观察内存。

在相应的变量上点击右键，选中菜单项“Add hw_event_cmp to Watch”或者使用快捷键“Ctrl+T, Ctrl+W”，可以将相应的变量增加到内存观察窗口。

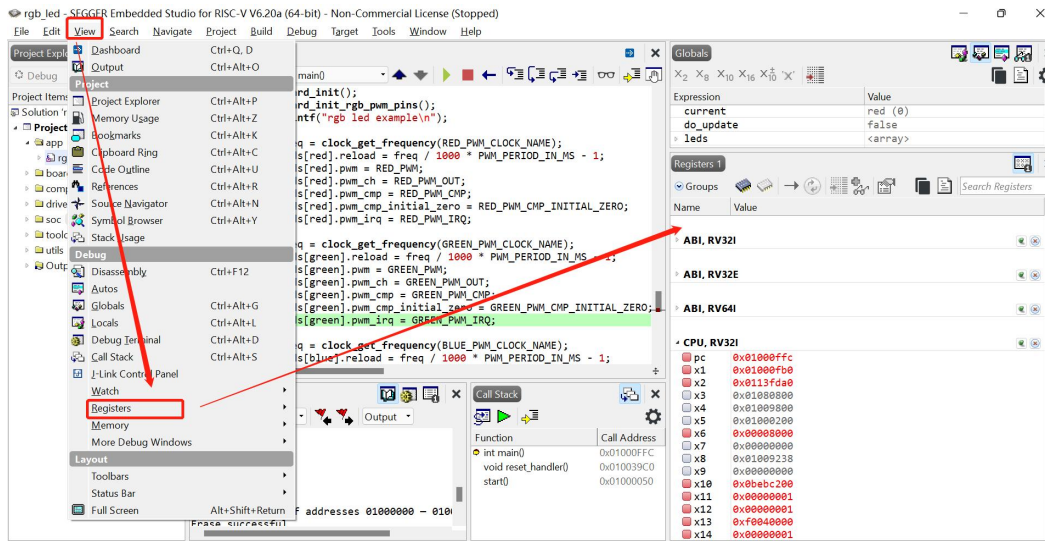


还能通过内存窗口控件标签，来切换观察变量的类型。可以查看局部变量、全局变量、自定义内存等。



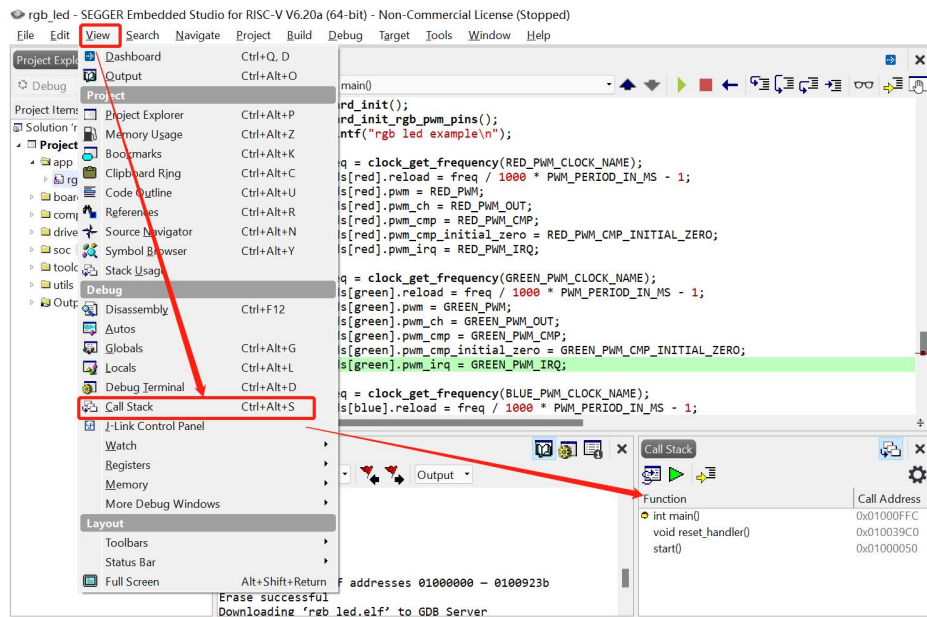
(6) 调试“rgb_led”工程：监测寄存器。

通过 “View>Registers” 来显示寄存器窗口，进而观察程序调试过程中，各种寄存器中相应值的变化情况。



(7) 调试 “rgb_led” 工程：函数调用栈。

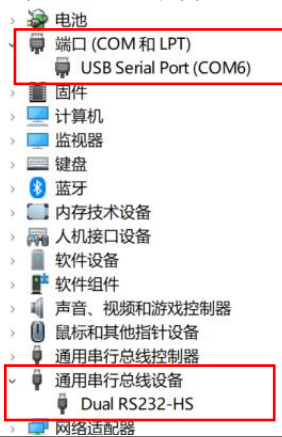
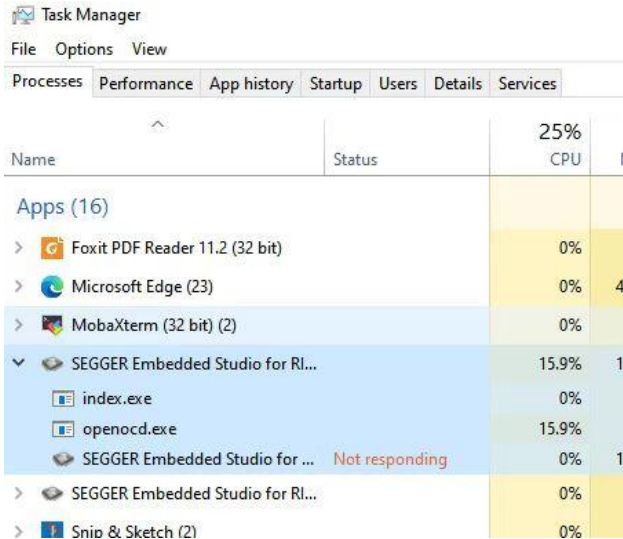
通过 “View>Call Stack” 或者快捷键 “Ctrl+Alt+S” 来显示函数调用栈窗口，进而观察程序调试过程中，函数调用情况。



更详细的 SES 使用方法，可以参考 SES 的帮助手册。



8 HPM SDK 常见问题

| 问题描述 | 解决方法 |
|---|--|
| Debug 无法连接 EVK, 无法烧写, Flash 内程序可以运行。 | EVK 上均设计有 BOOT 模式开关, 用户需要保证开关位于正确的组合, 才能进行调试。 Boot 模式选择在 SDK 中的 Doc 目录下的 EVK 使用说明书有详细的描述。 |
| SES 无法正确编译 | 请确认下载的 SES 的版本为 Risc-V 版本。 |
| SES 中无法输入中文, 中文乱码 | 不建议在 SES 中使用中文注释。 |
| SES 中提示 GDB server 出错 | 检查是否正确安装 FT2232 驱动, 正确安装后, 设备管理器中应存在 1 个串口和一个 USB 设备如下:  |
| <p>基于 windows 的操作系统, 在 SES 中通过 register 窗口访问未开 clock 模块寄存器, 会出现 SES 挂死。</p> <p>现象:</p> <ol style="list-style-type: none"> 1, 用户打开了寄存器窗口, 2, 在 download 加 debug 过程中, 寄存器窗口里有需要初始化 clk 的 soc 寄存器。 | <ol style="list-style-type: none"> 1、 在 Windows 的任务管理器找到挂死的 SES 进程。  2、 从挂死的 SES 进程中找到关联的 openocd 进程, 并将其结束运行。 |

The image shows two screenshots related to task management and GDB server configuration.

The top screenshot is a Windows Task Manager window showing a list of processes. The 'openoc' process is highlighted, and a context menu is open with 'End task' circled in red.

| Name | Status | CPU | Memory | Disk | Network |
|-----------------------------------|--------|-------|----------|--------|---------|
| Apps (16) | | | | | |
| Foxit PDF Reader 11.2 (32 bit) | | 0% | 5.9 MB | 0 MB/s | 0 Mbps |
| Microsoft Edge (23) | | 0% | 439.1 MB | 0 MB/s | 0 Mbps |
| MobaXterm (32 bit) (2) | | 0% | 2.5 MB | 0 MB/s | 0 Mbps |
| SEGGGER Embedded Studio for RI... | | 18.0% | 128.4 MB | 0 MB/s | 0 Mbps |
| index.exe | | 0% | 4.1 MB | 0 MB/s | 0 Mbps |
| openoc | | 18.0% | 3.5 MB | 0 MB/s | 0 Mbps |
| SEGGGER | | 0% | 120.9 MB | 0 MB/s | 0 Mbps |
| SEGGGER | | 0% | 4.6 MB | 0 MB/s | 0 Mbps |

3、 重启开发板。

4、 更改 SES 中 GDB server 设置, 将 “Reset and Stop Command” 的选项改为 “reset init”, 保存设置后, SES 可以重新调试

The bottom screenshot shows the 'Debug' options in SEGGER Embedded Studio. The 'Reset and Stop Command' is set to 'reset init', which is circled in red.

| Option | Value |
|--------------------------------------|--|
| Host | localhost |
| Type | Custom |
| GDB Server Command Line | C:\repos\rd_ems\tools\openoc\openoc.exe -f \$@HPM_LDK_BIOS\boards\openoc\boards\R2232... |
| Auto Start GDB Server | Yes |
| Port | 2331 |
| Reset and Stop Command | reset init |
| Ignore Checksum Errors | No |
| Allow Memory Access During Execution | Enabled |
| Register Access | General and Individual |
| Breakpoint Types | Software Only |
| Log File | None |
| Target Ick File | None |
| Connect Timeout | 5 seconds |
| Read Timeout | 60 seconds |
| Write Timeout | 300 seconds |