



GigaDevice

VSCODE环境搭建

VSCOED

Visual Studio Code 是一款免费、开源、跨平台的代码编辑器，支持几乎任何编程语言。在嵌入式开发方面，相比KEIL、IAR：

优点：

- 1) 跨平台，可同时接入ARMCC和GCC；
- 2) 丰富的插件生态，可加入各种强大的功能，实现类似KEIL的效果；
- 3) 友好的代码编辑页面和AI插件功能；
- 3) 轻量化设计，资源占用低，不会像KEIL一样突然崩溃；

缺点：相比于KEIL、IAR的零配置，VSCODE需要自行配置从工具链到仿真烧录等各种功能。

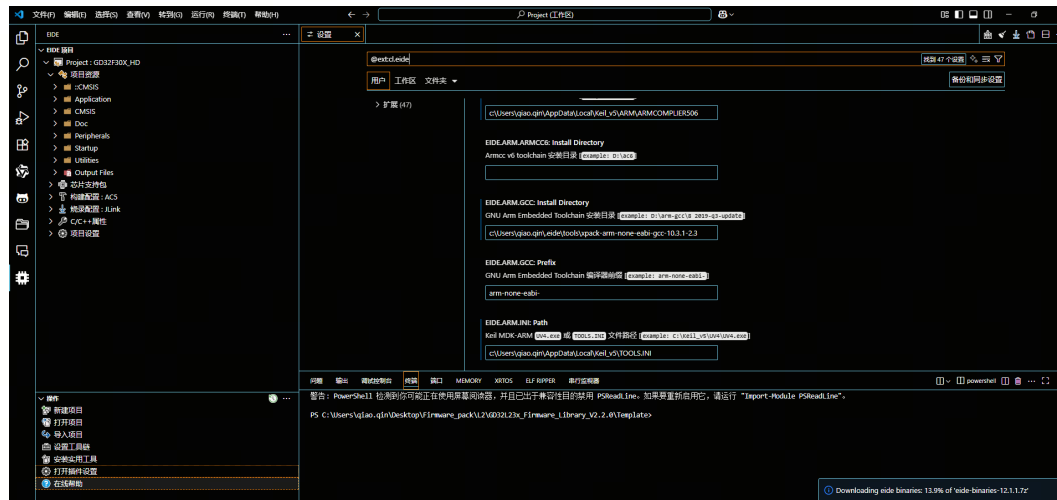
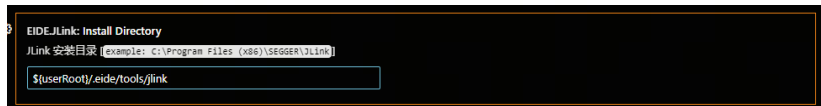
安装EIDE插件

Embedded IDE是一个专门用于单片机开发的VSCODE插件，支持8051/STM8/Cortex-M/MIPS/RISC-V。支持KEIL5/IAR/Eclipse项目导入，支持多种工具链的代码构建和编译。



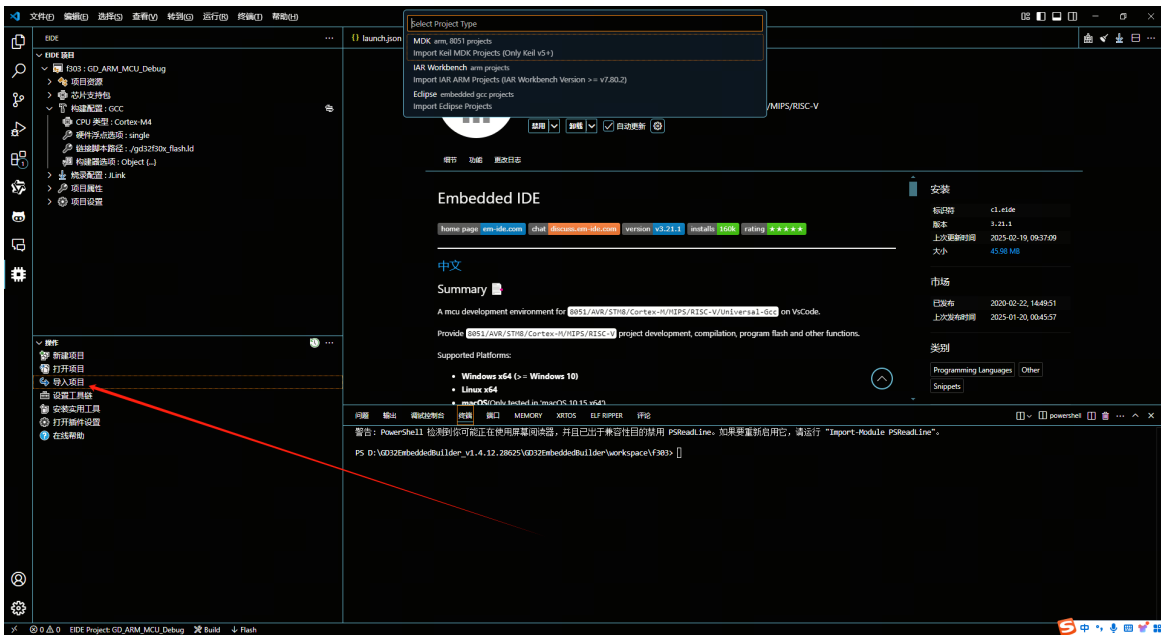
设置工具链

在插件设置中配置编译器和烧录器的路径。

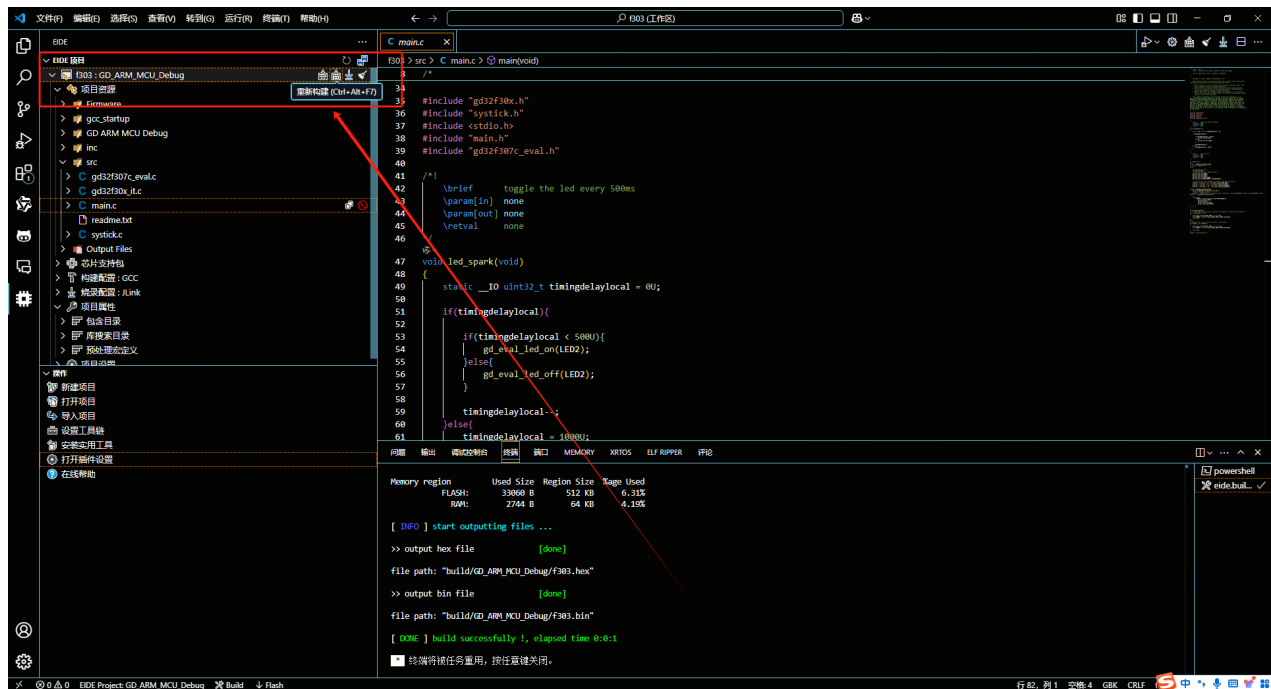


导入项目

插件支持导入MDK、IAR、eclipse工程，导入的工程默认分别使用MDK的ARMCC编译器、IAR的ICCARM编译器、eclipse的arm-none-eabi-gcc编译器。MDK和IAR工程切换为GCC编译需要修改启动文件和链接脚本。

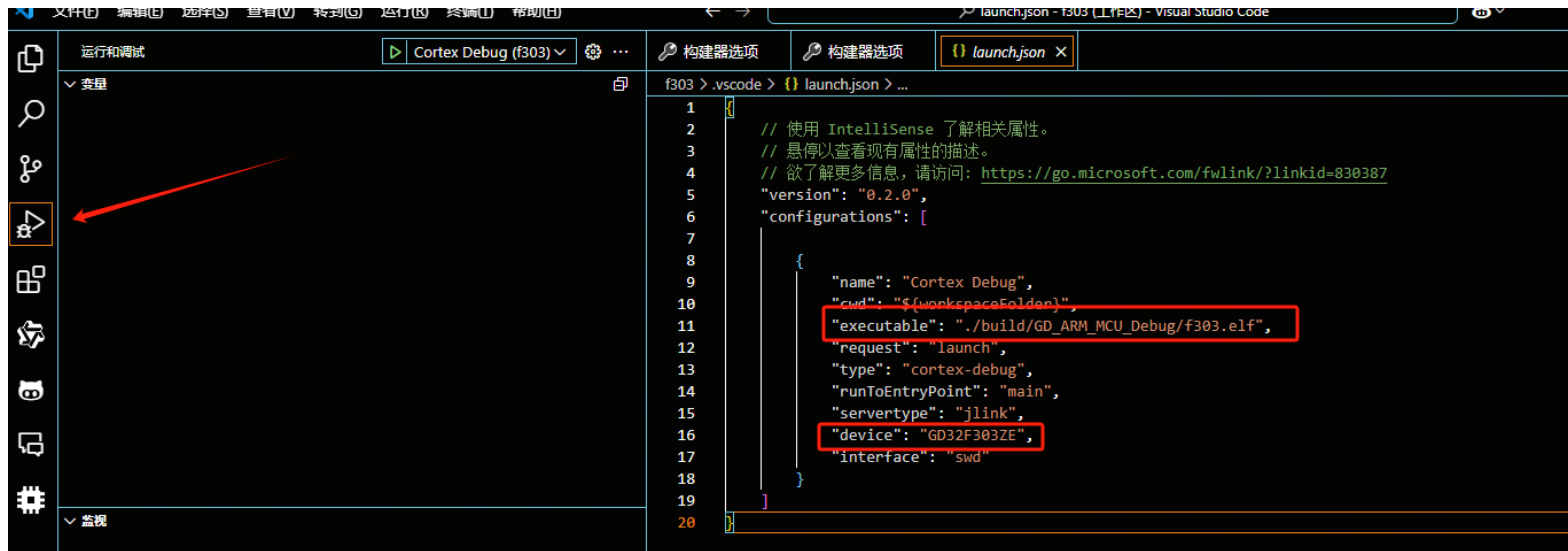
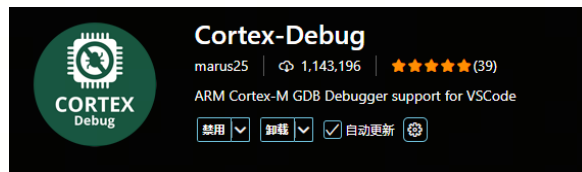


编译-下载

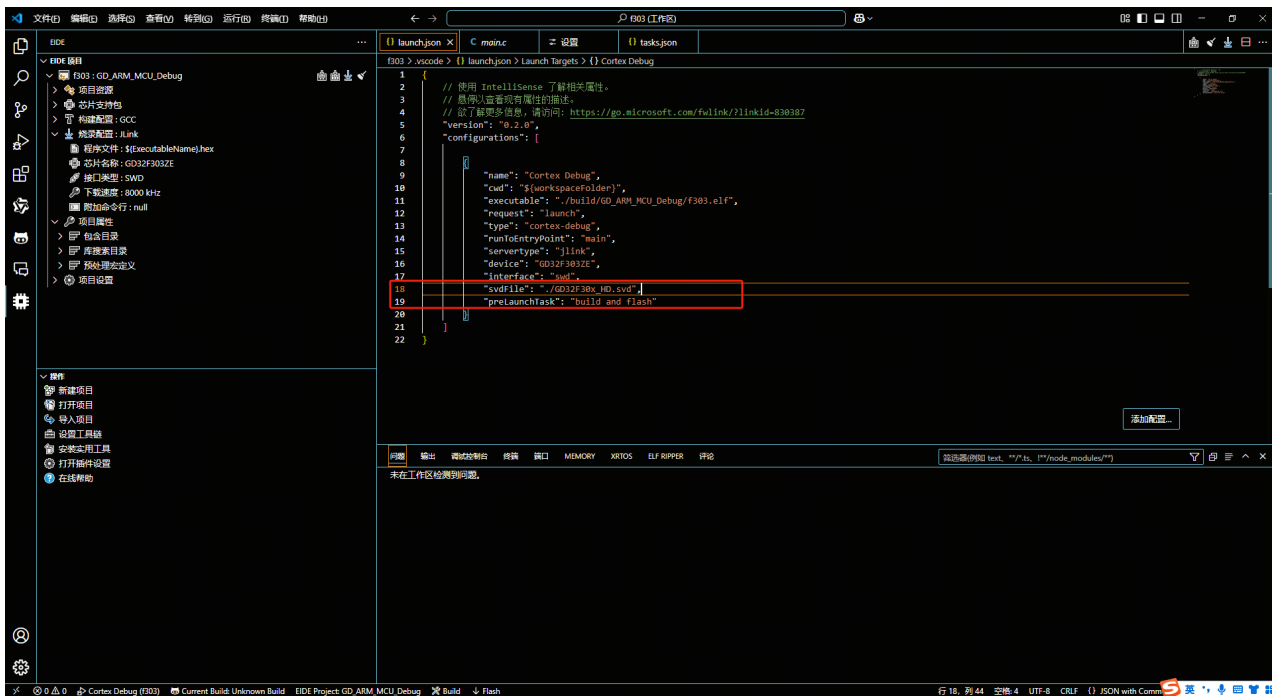


调试配置

调试功能需要借助Cortex-Debug插件，插件会自动生成配置脚本如下，需要配置相应的axf或elf文件路径和设备名。



添加SVD文件和预执行任务

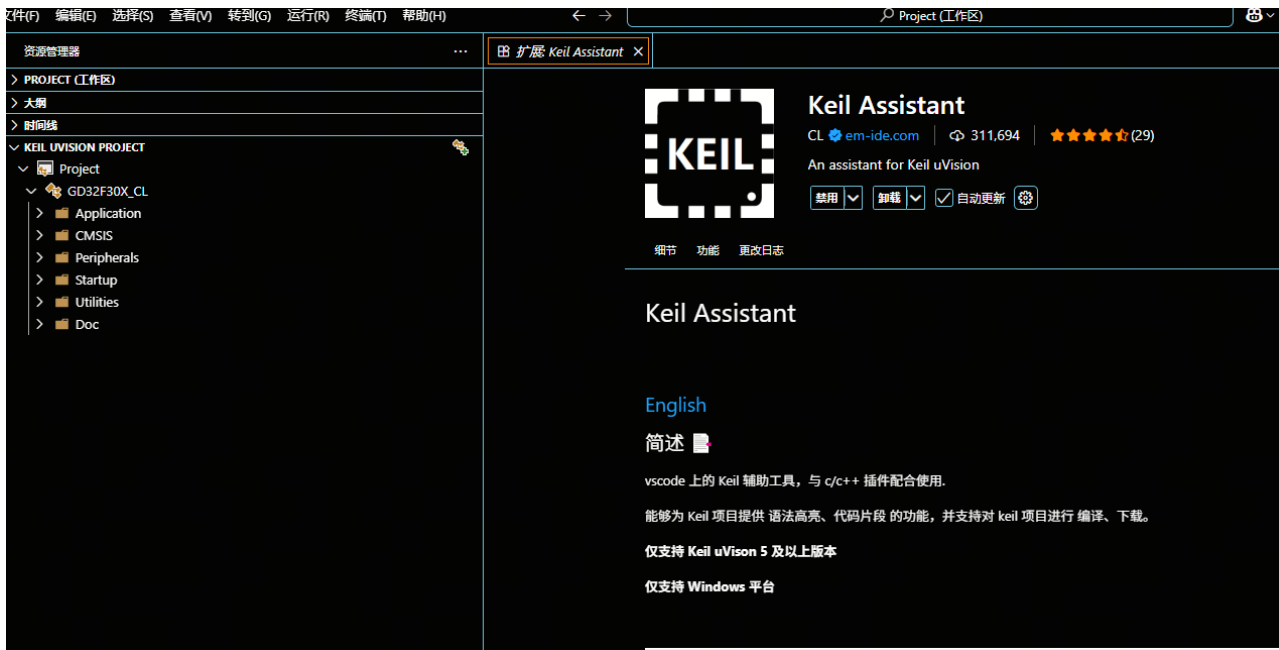


添加文件夹至工作区



插件-Keil Assistant

如果不想繁琐的配置，只需要vscode作为代码编辑器和插件平台，可以使用Keil Assistant。只需要配置UV5路径后，可以打开已有的KEIL工程，插件会自动调用该工程的配置，使用ARMCC进行编译、烧录和调试。但是不能对工程和编译器进行配置，也不能使用GCC编译。



插件-外设寄存器查看



运行和调试 | Cortex Debug (GD32F30x_Firmware_Library_V3.0.v... | C main.c | 扩展 RTOS Views | launchjson | C gd32f30x_itc | C gd32f30x_itk

变量

- Local
 - __result = 0
 - dma_init_struct = {...}
- Global
- Static: c:/Users/qiao.qin/Desktop/Firmware_pack/F3/GD32F30x_Firmware_Library_V3.0
- Registers

监视

- 通用寄存器
 - main@0x88009ec | _vmain.c (75)

断点

- CORTEX LIVE WATCH
- XPERIPHERALS
 - ADC0 @ 0x40012400
 - ADC1 @ 0x40012800
 - ADC2 @ 0x40013c00
 - AFIO @ 0x40010000
 - BKP @ 0x40006c00
 - CAN @ 0x40006400
 - CRC @ 0x40023000
 - CTC @ 0x4000c800
 - DAC @ 0x40007400
 - DBG @ 0xe0042000
 - DMA0 @ 0x40020000
 - DMA1 @ 0x40020400
 - EXMC @ 0xa0000000
 - EXTI @ 0x40010400
 - FMC @ 0x40022000
 - FWDTG @ 0x40033000
 - GPIOA @ 0x40010800
 - GPIOB @ 0x40010c00
 - GPIOC @ 0x40011000
 - GPIOD @ 0x40011400
 - GPIOE @ 0x40011800
 - GPIOF @ 0x40011c00
 - GPIOG @ 0x40012000

```
GD32F30x_Firmware_Library_V3.0 | Template > C main.c > main(void)
54 void i2c_config(void);
55 void nvic_config(void);
56 ErrStatus memory_compare(uint8_t *src, uint8_t *dst, uint16_t length);
57
58
59 /*
60  *brief      main function
61  *param[in]  none
62  *param[out] none
63  *retval     none
64  */
65
66 int main(void)
67 {
68     dma_parameter_struct dma_init_struct;
69
70     gd_eval_led_init(LED2);
71     gd_eval_led_init(LED3);
72
73     /* initialize DMA complete flags */
74     dma_ch4_complete = 0;
75     dma_ch5_complete = 0;
76
77     rcu_config();
78     gpio_config();
79     i2c_config();
80     nvic_config();
81
82     /* initialize DMA channel4 */
83     dma_deinit(DMA0, DMA_CH4);
84     dma_struct_para_init(&dma_init_struct);
```

问题 输出 调试控制台 终端 窗口 MEMORY XRTOS ELF RIPPER

0x08000000 | + | 图标 | 图标 | 图标 | 图标 | Status: Debugger attached, stopped

00000000	03-00	07-04	0b-08	0f-0c	13-10	17-14	1b-18	1f-1c
0000000000000000	20004a0	08000149	08000253	0800024b	0800024f	0800019f	080003a5	08000000
0000000000000000	08000300	00000000	00000000	08000257	08000249	00000000	08000255	08000259
0000000000000000	08000162	08000162	08000162	08000162	08000162	08000162	08000162	08000162
0000000000000000	08000163	08000163	08000163	08000163	08000163	08000163	08000195	080001c9
0000000000000000	08000209	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000000000000	08000163	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000000000000	08000163	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000000000000	08000163	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000000000100	08000163	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000000000120	08000163	08000163	08000163	08000163	d0f8dfdf	f81af000	47004800	080009d5
0000000000000140	8000f2af	200004a0	47004806	47004806	e7fe07fe	e7fe07fe	e7fe07fe	e7fe07fe
0000000000000160	e7fe07fe	08000254	08000131	40b7a006	68e00006	0301f040	00070804	34104708
0000000000000180	d3f642ac	ffd8f7ff	08000f84	08000fa4	e7feb000	2202b510	480a2103	f97af000
00000000000001a0	2202b120	48072103	f96cf000	21032208	f0004804	b120f96f	00004801	00004801
00000000000001c0	bd10f96f	40020000	2202b510	480c2104	f960f000	2202b138	48092104	f952f000
00000000000001e0	49082001	22087008	48052104	f952f000	22080120	48022104	f941f000	08000b10

插件-内存查看



```
C main.c x launchjson C gd32f30x_itc C gd32f30x_it.h
GD32F30x_Firmware_Library_V3.0.1 > Template > C main.c > main(void)
35 #include "gd32f30x.h"
36 #include "gd32f30x_itc.h"
37 #include <stdio.h>
38
39 /*
40  *brief      main function
41  *param[in]  none
42  *param[out] none
43  *retval     none
44  */
45 int main(void)
46 {
47     /* enable GPIO clock */
48     rcu_periph_clock_enable(RCU_GPIOA);
49
50     /* enable USART clock */
51     rcu_periph_clock_enable(RCU_USART0);
52
53     /* connect port to USARTx_Tx */
54     gpio_init(GPIOA, GPIO_MODE_AF_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_9);
55
56     /* connect port to USARTx_Rx */
57     gpio_init(GPIOA, GPIO_MODE_IN_FLOATING, GPIO_OSPEED_50MHZ, GPIO_PIN_10);
58
59     /* USART configure */
60     usart_deinit(USART0);
61     usart_baudrate_set(USART0, 115200);
62     usart_receive_config(USART0, USART_RECEIVE_ENABLE);
63     usart_transmit_config(USART0, USART_TRANSMIT_ENABLE);
64 }

```

问题 输出 调试控制台 终端 窗口 MEMORY XRTOS ELF RIPPER

0x08000000 Status: Debugger attached, stopped

03-00	07-04	0b-08	0f-0c	13-10	17-14	1b-18	1f-1c
0000000008000000	20000408	08000149	08000151	08000153	08000155	08000157	08000159
0000000008000020	08000000	08000000	0800015b	0800015d	08000000	0800015f	08000161
0000000008000040	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000008000060	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000008000080	08000163	08000163	08000163	08000163	08000163	08000163	08000163
00000000080000a0	08000163	08000163	08000163	08000163	08000163	08000163	08000163
00000000080000c0	08000163	08000163	08000163	08000163	08000163	08000163	08000163
00000000080000e0	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000008000100	08000163	08000163	08000163	08000163	08000163	08000163	08000163
0000000008000120	08000163	08000163	08000163	08000163	d010f8df	f81af000	47004800
0000000008000140	8000f3af	20000408	47804806	47004806	e7fee7fe	e7fee7fe	e7fee7fe
0000000008000160	e7fee7fe	08000191	08000131	4d074c06	68e0e006	0301f040	9007e894
0000000008000180	d3f642ac	f0d01fff	08000008	08000908	494eb51c	f4405800	49440070
00000000080001a0	f0406800	40440001	b4f00908	68004848	0007f000	d0f92800	63404845
00000000080001c0	d1662988	2800bf00	e0029001	1c409801	98019001	d3f92850	6340484d
00000000080001e0	00f0f020	98009000	0800f040	49389800	60489800	90012000	9801e002

插件-串口工具



```
EIDE
C main.c x 田 扩展 Serial Monitor {} launch.json C gd32f30x_itc C gd32f30x_itx
> EIDE 项目
GD32F30x Firmware Library_V3.0.1 > Template > C main.c > ...
  操作
  新建项目
  打开项目
  导入项目
  设置工具箱
  安装实用工具
  打开插件设置
  在线帮助
  串行监视器: 串行监视器
  + 打开其他监视器
  监视模式 Serial 查看模式 文本
  端口 COM18 - USB-SERIAL-CH340 (COM18)
  波特率 115200
  行尾 无 停止监视
  35 #include "gd32f30x.h"
  36 #include "gd32f30x_itc.h"
  37 #include <stdio.h>
  38
  39 /*
  40  * \brief      main function
  41  * \param[in] none
  42  * \param[out] none
  43  * \retval    none
  44  */
  45 int main(void)
  46 {
  47     /* enable GPIO clock */
  48     rcu_periph_clock_enable(RCU_GPIOA);
  49
  50     /* enable USART clock */
  51     rcu_periph_clock_enable(RCU_USART0);
  52
  53     /* connect port to USARTx Tx */
  54     gpio_init(GPIOA, GPIO_MODE_AF_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_9);
  55
  56     /* connect port to USARTx Rx */
  57     gpio_init(GPIOA, GPIO_MODE_IN_FLOATING, GPIO_OSPEED_50MHZ, GPIO_PIN_10);
  58
  59     /* USART configure */
  60     usart_deinit(USART0);
  61     usart_baudrate_set(USART0, 115200);
  62     usart_receive_config(USART0, USART_RECEIVE_ENABLE);
  63     usart_transmit_config(USART0, USART_TRANSMIT_ENABLE);
  64     usart_enable(USART0);
  65
  66     printf("\n\n\r usart transmit test example!\n\n\r");
  67     while(1);
  68 }
  69
  70 /* retarget the C library printf function to the USART */
  71 int fputc(int ch, FILE *f)
  72 {
  73     usart_data_transmit(USART0, (uint8_t)ch);
  74     while(RESET == usart_flag_get(USART0, USART_FLAG_TBE));
  75     return ch;
  76 }
  77
```

插件-RTOS Views

RTOS Views 预览

mcu-debug | 906.137 | ★★★★★(3)

RTOS views for microcontrollers

禁用 卸载 自动更新

细节 功能 更改日志 依赖项

```
186
187
188     ret = bind(sockfd, (struct sockaddr *)&clt_addr, sizeof(clt_addr));
189     if(ret < 0){
190         lwip_close(sockfd);
191         sockfd = -1;
192         continue;
193     }
194
195     /* connect */
196     ret = connect(sockfd, (struct sockaddr *)&svr_addr, sizeof(svr_addr));
197     if(ret < 0){
198         lwip_close(sockfd);
199         sockfd = -1;
200         continue;
201     }
202
203     while(-1 != sockfd){
204         /* receive packets, and limit a reception to MAX_BUF_SIZE bytes */
205         recvnum = recv(sockfd, buf, MAX_BUF_SIZE, 0);
206         if(recvnum <= 0){
207             lwip_close(sockfd);
208             sockfd = -1;
209             break;
210         }
211         send(sockfd, buf, recvnum, 0);
212     }
```

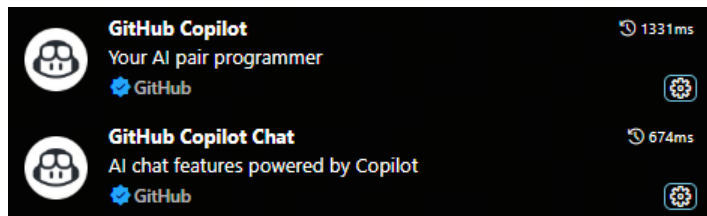
问题 输出 调试控制台 终端 窗口 MEMORY XRTOS ELF RIPPER

RTOS Views: Session Name: "Cortex Debug", FreeRTOS detected.

Thread ID	Address	Task Name	Status	Prio	Stack rity	Stack Start	Stack Top	Stack End	Stack Size	Stack Used	Stack Free	Stack Peak	Runtime
2	0x20004508	IDLE	READY	0,0	0x200042f8	0x200044b8	0x????????	???	???	448	???	???	??,??%
3	0x20004a58	Tmr_Svc	SUSPENDED	2,2	0x20004640	0x200049e8	0x????????	???	???	936	???	???	??,??%
4	0x20005b90	TCP/IP	BLOCKED	6,6	0x20004b88	0x20005ad0	0x????????	???	???	3912	???	???	??,??%
5	0x200061d0	ETHERNETIF_INPU	BLOCKED	7,7	0x20005c50	0x20006158	0x????????	???	???	1288	???	???	??,??%
6	0x20006a98	HELLO	RUNNING	5,5	0x20006290	0x20006870	0x????????	???	???	1504	???	???	??,??%
7	0x200073d8	TCP_CLIENT	READY	5,5	0x20006bd0	0x20007238	0x????????	???	???	1640	???	???	??,??%
8	0x20007d18	UDP	SUSPENDED	5,5	0x20007510	0x20007b08	0x????????	???	???	1528	???	???	??,??%
9	0x20008060	LED	BLOCKED	2,2	0x20007e50	0x20008000	0x????????	???	???	432	???	???	??,??%

Data collected at 2025-07-21T08:51:16.323Z in 870 ms

Copilot



模式	功能性
内联建议/代码补 ⤴	简化编码，同时保持流程。
询问 (ASK)	提供基于聊天的问答助手，用于理解代码、概念，以及获取代码片段
编辑 (Edit)	通过解释自然语言提示直接修改代码，应用更改到多个文件。
代理 (Agent)	充当自主的AI助手，能够执行多步任务，包括编写、测试和调试代码



Copilot - 指令文件

1. 开启设置中的Use Instruction Files
2. 在工作区的根目录中创建文件.github/copilot-instructions.md
3. 将Markdown格式的说明添加到文件中。例如：
 - 1) 必须使用C++编写代码
 - 2) 每个函数接口都必须有详细的函数说明
 - 3) 所有函数接口都必须使用GD_为开头
 - 4) 所有代码必须经过测试并确保无错误

GitHub · Copilot · Chat · Code Generation: Use Instruction Files

控制是否将来自 `.github/copilot-instructions.md` 的代码说明添加到 Copilot 请求。

注意: 请提供简短且精确的指令。指令不佳可能会降低 Copilot 的质量和效果。 [详细了解](#)如何自定义 Copilot。

Copilot - 代理模式

Mermaid语言

```
C main.c x LIN执行流程图.md x
MDK-ARM > LIN执行流程图.md > # LIN通信执行流程图 > ## 3. 主循环流程 (lin_loop)
1 # LIN通信执行流程图
2
3
4 ## 1. 系统初始化流程
5
6 mermaid
7 graph TD
8   A[系统启动] --> B[SEGGER RTT Init RTT调试初始化]
9   B --> C[systick config 系统滴答定时器配置]
10  C --> D[param_init 参数初始化]
11  D --> E[lin_all_init LIN外设初始化]
12  E --> F[timeout_config 超时配置]
13  F --> G[timer_config 定时器配置]
14  G --> H[timer_enable 使能定时器]
15  H --> I[进入主循环 lin_loop]
16
17 ## 2. LIN参数初始化流程 (param_init)
18
19 mermaid
20 graph TD
21   A[param_init开始] --> B[遍历所有LIN节点]
22   B --> C[设置节点基本参数]
23   C --> D[初始化消息列表数组]
24   D --> E[计算PID包含校验位]
25   E --> F[配置发送任务ID-接收数据]
26   F --> G[配置接收任务ID映射]
27   G --> H[设置预期接收数据]
28   H --> I[初始化完成]
29
30
```



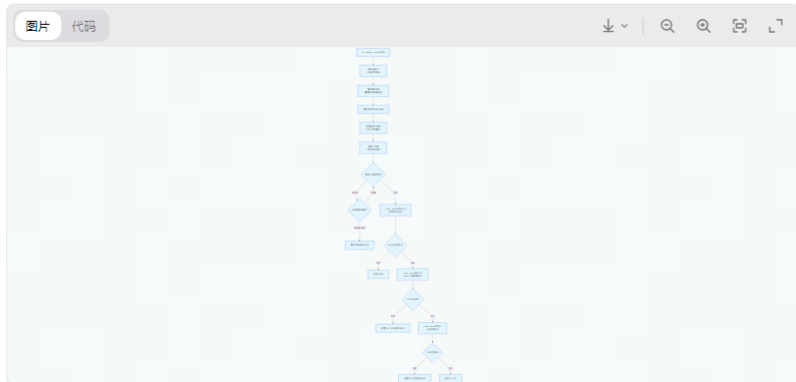
Copilot – 代理模式

```
95 \param[out] none
96 \retval none
97 \note
98     执行步骤:
99     1. 根据当前LIN通信阶段处理接收到的数据
100    2. 验证SYNC、PID、数据和校验和的正确性
101    3. 更新LIN状态机和错误标志
102    4. 处理帧接收完成后的数据拷贝
103
104
105 void lin_interrupt_caller(lin_typedef_enum lin, uint8_t rx_data)
106 {
107     uint8_t i;
108
109     /* LIN状态机推进到下一个阶段 */
110     g_lin_node[lin].lin_phase++;
111     switch(g_lin_node[lin].lin_phase){
112     case PHASE_IDLE:
113     case PHASE_BREAK:
114         /* 空闲和中断帧阶段无需处理 */
115         break;
116     case PHASE_SYNC:
117         /* check if the received/readback SYNC is correct or not */
118         /* 检查接收/回读的SYNC是否正确 */
119         if(0x55!= rx_data){
120             g_lin_node[lin].msg_readback.error = SYNC_ERR;
121             g_lin_node[lin].status |= STATUS_SYNC_ERR;
122             g_timeout_lin[lin][1]=0;
123             g_timeout lin[lin][1]=0; // 停止超时间数
124         }
125     }
```

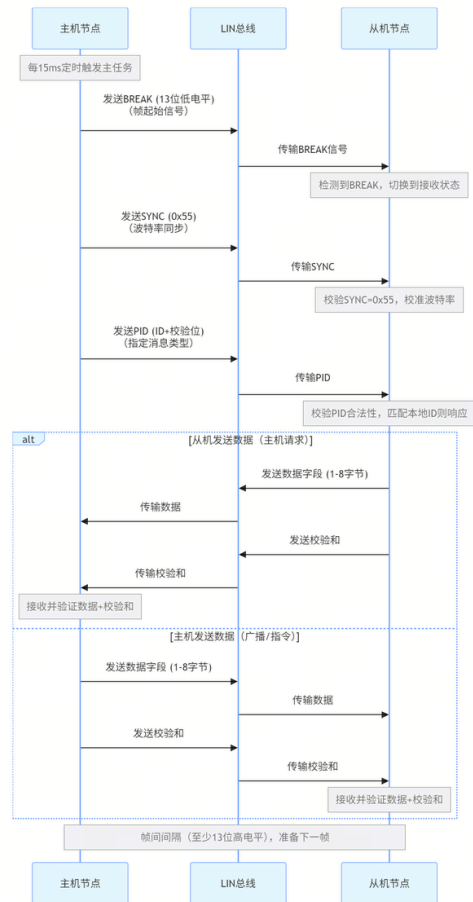
Copilot – 代理模式



4. LIN 消息头发送流程图 (lin_header_transmit)



5. 接收中断处理流程图 (lin_interrupt_caller)



Copilot – 代理模式

现在工程的功能是IC+DMA传输，现在要求加入dma传输完成中断，在中断服务函数中清除中断标志位，中断服务函数即gd32f30x_itc.c中

我来帮您在当前的IC+DMA传输工程中添加DMA传输完成中断功能，首先让我分析一下当前的代码结构，然后在 `gd32f30x_itc.c` 中添加相应的中断服务函数。

让我先查看一下相关的头文件，了解需要的函数声明：

- ✓ 读取 `gd32f30x_it.h`，行 1 到 50
- ✓ 读取 `gd32f30x_it.h`，行 51 到 63

现在我需要在 `gd32f30x_it.c` 中添加DMA传输完成中断服务函数。根据 `main.c` 中的代码，我们使用了DMA0的通道4和通道5：

- DMA0_CH4: 用于I2C接收 (peripheral to memory)
- DMA0_CH5: 用于I2C发送 (memory to peripheral)

我需要添加对应的中断服务函数。首先在头文件中添加DMA0_channels5_irq_handler的声明：

```
C gd32f30x_it.h +2-0
```

现在在 `gd32f30x_it.c` 中添加DMA中断服务函数。我需要在文件末尾添加这些函数：

```
C gd32f30x_it.c +42-0
```

```
现在需要修改 main.c 文件来启用DMA中断，需要在DMA配置完成后启用中断，并在NVIC中配置中断优先级：
```

```
C main.c +1-0
```

```
C main.c +1-0
```

已更改 3 个文件

- C gd32f30x_itc.Template
- C gd32f30x_it.h.Template
- C main.c.Template

添加上下文 C gd32f30x_itc.当前文件

在代理模式下编辑工作区中的文件

Agent Claude Sonnet 4

```
GD32F30x_Firmware_Library_V3.0.1 > Template > C gd32f30x_itc.c ...
170 void DMA0_Channel4_IRQHandler(void)
171 {
172     if(dma_flag_get(DMA0, DMA_CH4, DMA_FLAG_FTF)) {
173     }
174 }
175
176 /* check DMA0 channel4 transfer error flag */
177 if(dma_flag_get(DMA0, DMA_CH4, DMA_FLAG_ERR)) {
178     /* clear DMA0 channel4 transfer error flag */
179     dma_flag_clear(DMA0, DMA_CH4, DMA_FLAG_ERR);
180 }
181 }
182
183 /*!
184 \brief this function handles DMA0_Channel5_IRQHandler interrupt
185 \param[in] none
186 \param[out] none
187 \retval none
188 */
189 void DMA0_Channel5_IRQHandler(void)
190 {
191     /* check DMA0 channel5 transfer complete flag */
192     if(dma_flag_get(DMA0, DMA_CH5, DMA_FLAG_FTF)) {
193         /* clear DMA0 channel5 transfer complete flag */
194         dma_flag_clear(DMA0, DMA_CH5, DMA_FLAG_FTF);
195         /* set transfer complete flag */
196         dma_ch5_complete = 1;
197     }
198 }
199
200 /* check DMA0 channel5 transfer error flag */
201 if(dma_flag_get(DMA0, DMA_CH5, DMA_FLAG_ERR)) {
202 }
203 }
204
```

问题 输出 调试控制台 终端 MEMORY XRTOS ELF RIPPER

```
FPS28= 2FFC4E79, FPS29= 9926878B, FPS30= 67CDD2CD, FPS31= BBA35B0B
FPS32= 00000000
Downloading file [C:\Users\qiao.qin\Desktop\Firmware_pack\F3\GD32F30x_Firmware_Library_V3.0.1\Template\Weil5_project\build\GD32F30x_HD\Project.hex]...
J-Link: Flash download: Bank 0 @ 0x00000000: 1 range affected (6144 bytes)
J-Link: Flash download: Total time needed: 0.400s (Prepare: 0.077s, Compare: 0.007s, Erase: 0.174s, Program: 0.124s, Verify: 0.001s, Restore: 0.014s)
O.K.
Reset delay: 0 ms
Reset type: NDRMW: Resets core & peripherals via SYSRESETREQ & VECTRESET bit.
Reset: Halt core after reset via DEMCR.VC_CORERESET.
Reset: Reset device via AIBCR.SYSRESETREQ.
Script processing completed.
终端将被任务重用，按任意键关闭。
```



GigaDevice

感谢聆听

