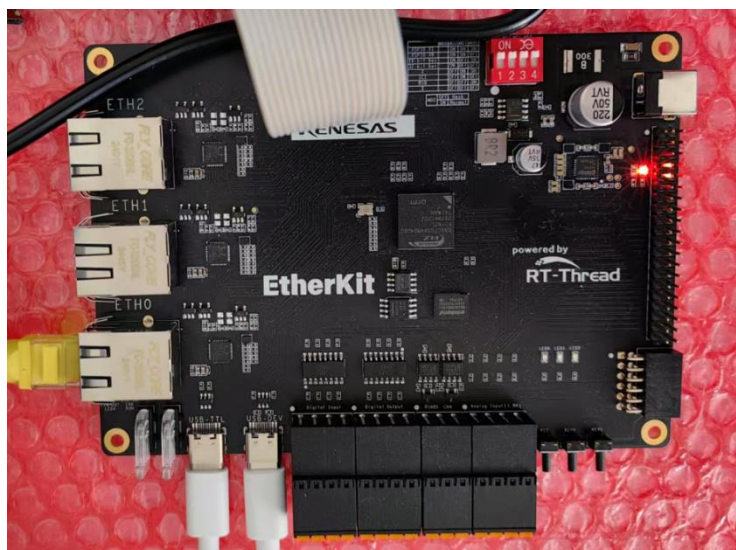


RZN2L SPI 操作手册-----基于 Etherkit 开发板

简介

本应用笔记介绍了基于 RZ/N2L Etherkit 开发板的定时器 SPI 的操作。分别介绍 IDE IAR 和 E2studio 软件下的操作。



开发工具

- IDE: IAR EW for Arm 9.50.2
E2studio 2024-01.1
- FSP: RZ/N2L FSP V2.0
- 仿真器: Jlink V12

实验材料

- Etherkit 开发板
- Jlink 仿真器, 需支持瑞萨 R52 内核

实验部分

1.硬件设置及软件安装	2
2.IAR 环境工程介绍	3
3.E2studio 环境工程介绍	9

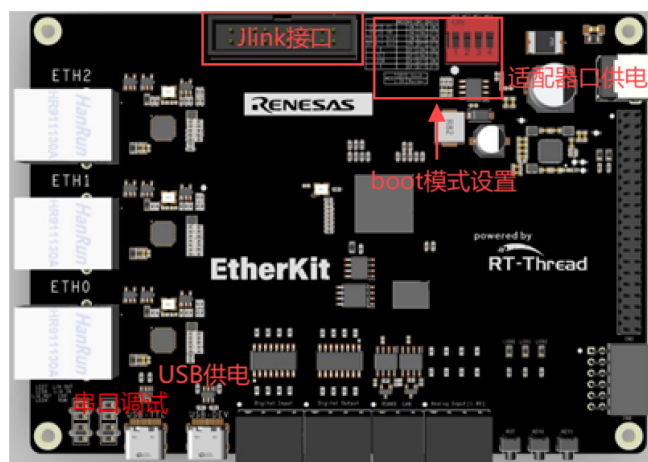
1 .硬件设置及软件安装

本节 EtherKit 开发板硬件设置。

1.1

开发板设置：

- 供电：可选 USB 供电或适配器供电
- Boot 模式设置：推荐 xSPI0 x1 boot mode
- Jlink v12

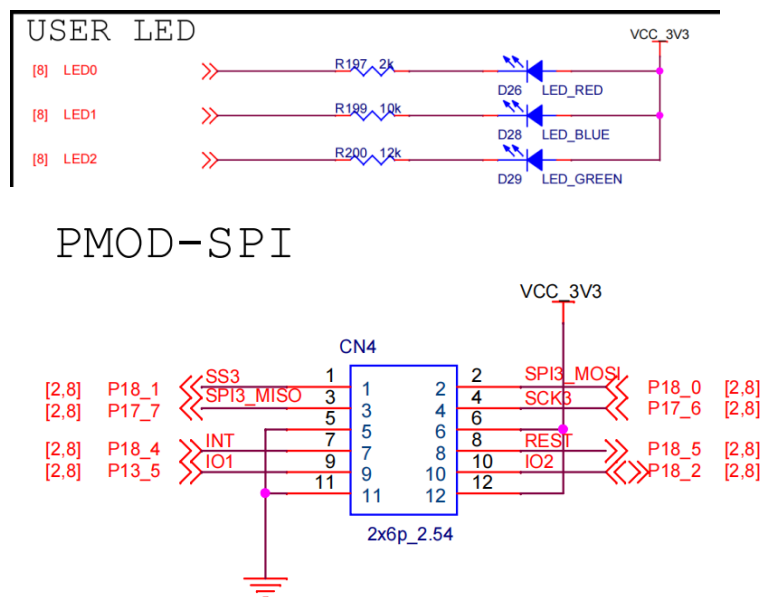


1.2

硬件原理图：

本实验用到 LED0，对应 P143 引脚。

板载资源 PMOD 接口，连接到 N2L 芯片的 SCI_SPI3；



本节完

2 .IAR 环境工程介绍

本节介绍 IAR 环境下 GPT 工程介绍。

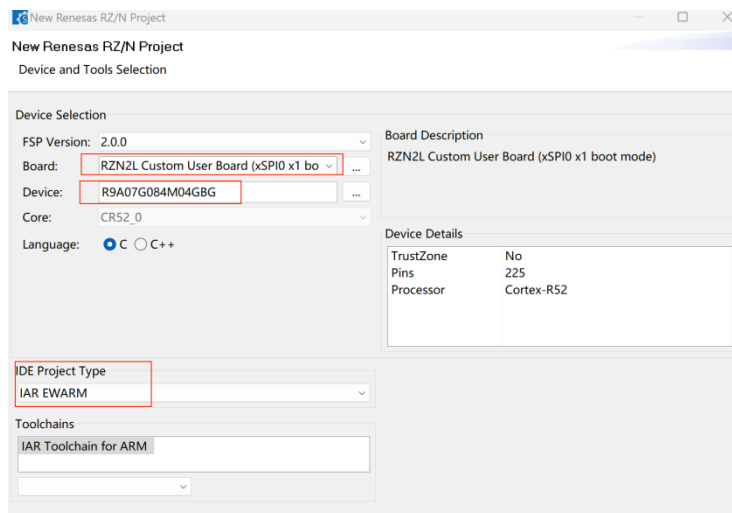
2.1

● 打开 FSP 新建工程：SPI

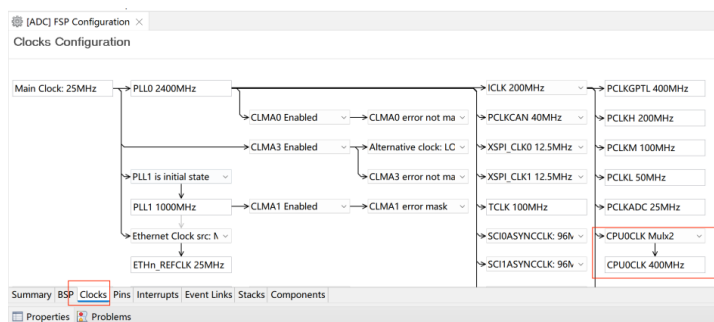
1. Boot 模式选择 RZN2L Custom User Board (XSPI0 x1 boot mode)

芯片型号：R9A07G084M04GBG

IDE Project Type 选择 IAR EWARM

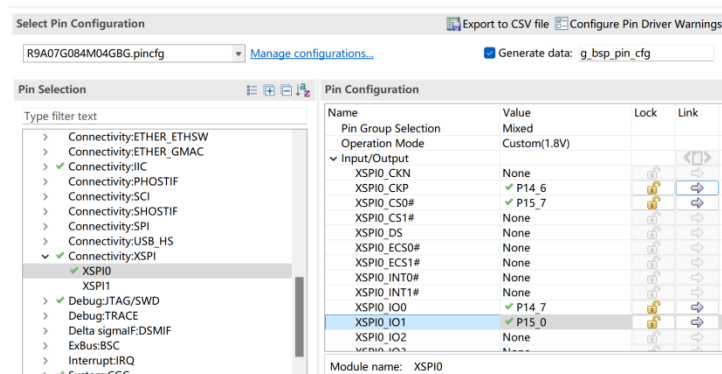


1. 主时钟设置 400MHZ



2. 配置 XSPI0

Pin Configuration



3. P143 引脚配置

P143 设置输出模式，初始电平 0

Select Pin Configuration Export to CSV file Configure Pin Driver Warnings

R9A07G084M04GBG.pincfg [Manage configurations...](#) ☒ Generate data: g_bsp_pin_cfg

Pin Selection
Type filter text

- > P12
- > P13
- > P14
 - P14_0
 - P14_1
 - P14_2
 - P14_3**
 - P14_4
 - P14_5
 - P14_6
 - P14_7
- > P15
- > P16

Pin Configuration

Name	Value
Comment	
Mode	Output mode (Low & Not Into Input)
Pull up/down	None
Output Type	CMOS
Drive Capacity	Low
Region	Safety
Schmitt Trigger	None
Slew Rate	Slow
Input/Output	
P14_3	GPIO

Module name: P14_3
Port Capabilities: ETHER, ETH0, ETH0_COL, ETHER, ETH2, ETH2_COL

Pin Function Pin Number

Summary BSP Clocks **Pins** Interrupts Event Links Stacks Components

4. 配置 SCI3

R9A07G084M04GBG.pincfg [Manage configurations...](#) ☒ Generate data: g_bsp_pin_cfg

Pin Selection
Type filter text

- > Connectivity:ETHER_ESC
- > Connectivity:ETHER_ETH
- > Connectivity:ETHER_ETHSW
- > Connectivity:ETHER_GMAC
- > Connectivity:IIC
- > Connectivity:PHOSTIF
- > Connectivity:SCI
 - SCI0
 - SCI1
 - SCI2
 - SCI3**
 - SCI4
 - SCI5
- > Connectivity:SHOSTIF
- > Connectivity:SPI
- > Connectivity:USB_HS
- > Connectivity:XSPI

Pin Configuration

Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Simple SPI mode		
Input/Output			
SS3	None		
MISO3	P17_7		
SCK3	P17_6		
	None		
MOSI3	P18_0		

Module name: SCI3
Usage: In initial register value, when CCR0.TE bit is 0, and the terminal fun

Pin Function Pin Number

Summary BSP Clocks **Pins** Interrupts Event Links Stacks Components

5. 添加 new Stack: SPI

New Stack---Connectivity--SPI(r_sci_spi), 删除 DMA

Threads New Thread Remove HAL/Common Stacks New Stack

HAL/Common

- g_ioport I/O Port (r_ioport)
- Memory config check
- g_spi3 SPI (r_sci_spi)

Objects New Object Remove

Summary BSP Clocks Pins Interrupts Event Links **Stacks** Components

Properties Problems

g_spi0 SPI (r_sci_spi)

Property	Value
Multiplex Interrupt	Disabled
Module g_spi0 SPI (r_sci_spi)	
Name	g_spi3
Channel	3
Operating Mode	Master
Clock Phase	Data sampling on odd edge, data variation on even edge
Clock Polarity	Low when idle
Bit Order	MSB First
Clock Source	SCINASYNCCLK
Callback	sci_spi_callback
Receive Interrupt Priority	Priority 12
Transmit Interrupt Priority	Priority 12
Transmit End Interrupt Priority	Priority 12
Error Interrupt Priority	Priority 12
Bitrate	2000000
Pins	
CTS_RTS_SS3#	None
RXD_MISO3	P17_7
SCK3	P17_6
TXD_MOSI3	P18_0

6. 点击: Generate Project Content 生成代码

Generate Project Content

New Stack > Extend Stack > Remove

2.2

7. 打开生成的代码

- 仿真器由 Ijet 切换为 Jlink
- 编写用户代码：将 P17_7 和 P18_0 短接，进行 SPI 回环测试，自发自收。

```
hal_entry.c x hal_data.c startup_core.c main.c
hal_entry()
7 static volatile bool g_transfer_complete = false;
8 void sci_spi_callback (spi_callback_args_t * p_args)
9 {
10     if (SPI_EVENT_TRANSFER_COMPLETE == p_args->event)
11     {
12         g_transfer_complete = true;
13     }
14 }
15 uint8_t tx_buffer[15];
16 uint8_t rx_buffer[15];
17 /*
18  * main() is generated by the FSP Configuration editor and is used to generate threads if an RTOS is used. This function
19  * is called by main() when no RTOS is used.
20  */
21 void hal_entry(void)
22 {
23     /* TODO: add your own code here */
24     uint16_t i;
25     R_SCI_SPI_Open (&g_spi3_ctrl, &g_spi3_cfg);
26     __asm volatile ("cpsie i");
27     __asm volatile ("isb");
28     for(i=0;i<12;i++)
29     {
30         tx_buffer[i] = i + 1;
31     }
32     g_transfer_complete = false;
33     R_SCI_SPI_WriteRead (&g_spi3_ctrl, tx_buffer, rx_buffer, 10, SPI_BIT_WIDTH_8_BITS);
34     while (false == g_transfer_complete)
35     {
36     }
37     while(1)
38     {
39     }
40 }
```

回调函数

用户代码

- Rebuild All---编译工程 无报错

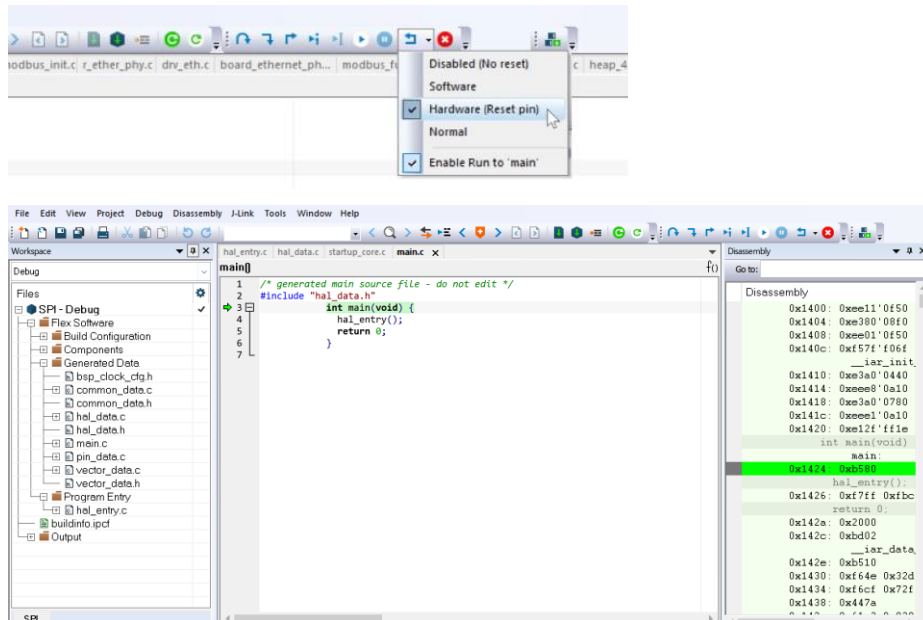
Build

Messages

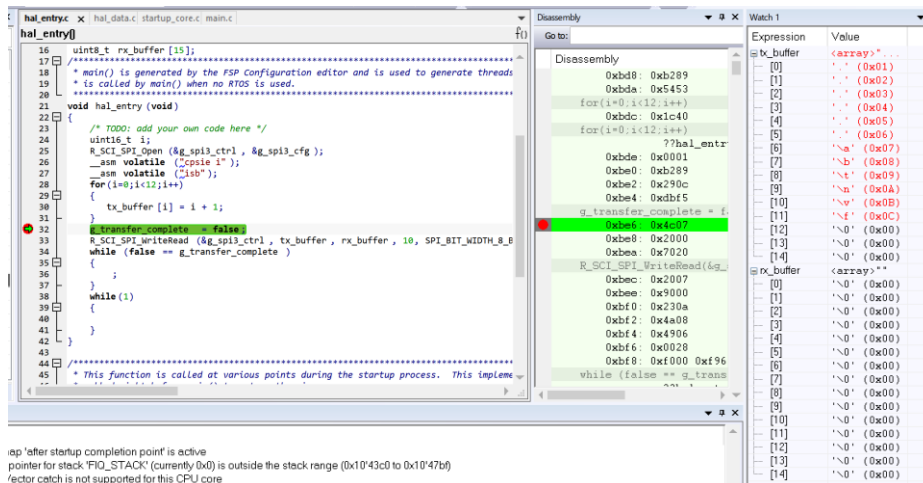
Total number of errors: 0
Total number of warnings: 0
Resolving dependencies...
Build succeeded

2.3

- Download and Debug --- 下载程序
- 下载工程到开发板，进入仿真界面
- Debug 复位设置为 Hardware



- 设置断点，程序运行之断点处，watch 查看 tx_buffer 和 rx_buffer 值。



- 全速运行代码，然后点击暂停：

The screenshot displays a development environment with three main panels:

- Source Code (Left):** Shows the `hal_entry` function in `hal_data.c`. The code includes a loop that reads data from the SPI controller into the `rx_buffer` array. Line 32, `g_transfer_complete = false;`, is highlighted in red.
- Disassembly (Middle):** Shows the assembly code corresponding to the C code. The instruction `hex02: 0x07fe` is highlighted in green, which corresponds to the `while(1)` loop in the C code.
- Watch Window (Right):** Shows the state of the `rx_buffer` array. The expression is `@rx_buffer` and the value is an array of 16 elements, all currently set to `0x00`.

Below the code panels, a status bar provides additional information:

```

'ector catch is not supported for this CPU core
t hit Code @ hal_data.c:32.5 type: default (hardware)
pointer for stack 'FIQ_STACK' (currently 0x0) is outside the stack range (0x10'43cd to 0x10'47bf)
'ector catch is not supported for this CPU core
  
```

可以看到 `rx_buffer` 中接收到 SPI 发送来的数据。

本节完

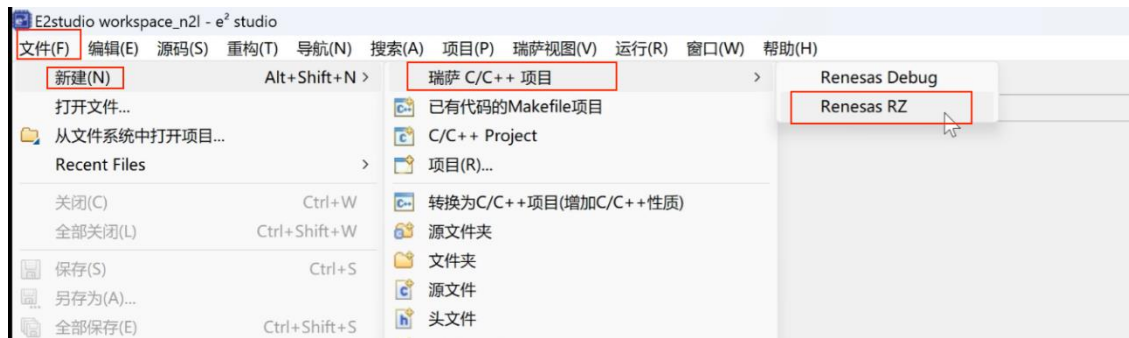
3 .E2studio 环境工程介绍

本节介绍使用 E2studio 环境创建 IIC 工程。

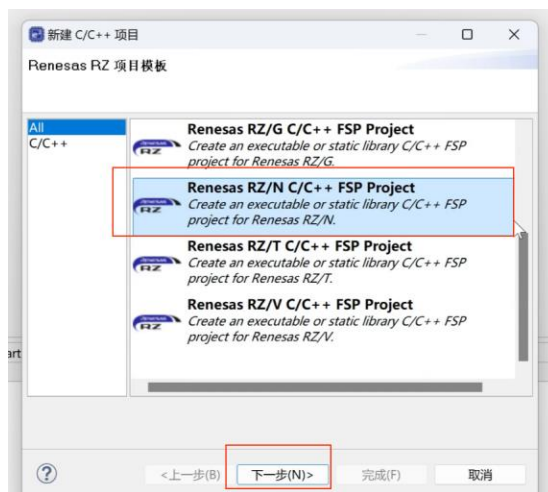
3.1

● 打开 E2studio, 新建工程

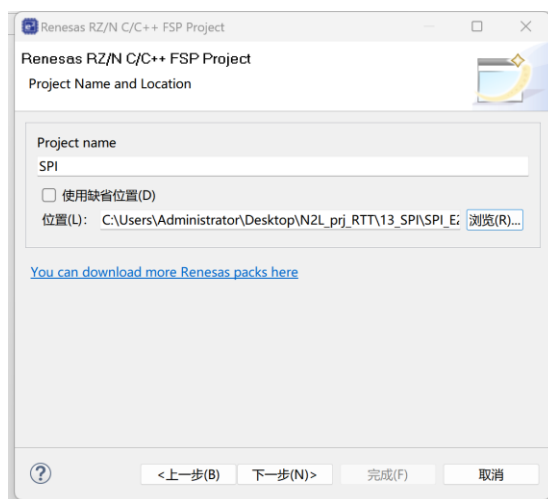
1. 选择 文件--新建--瑞萨 C/C++项目--Renesas RZ:



2. 选择 Renesas RZ/N C/C++ FSP Project

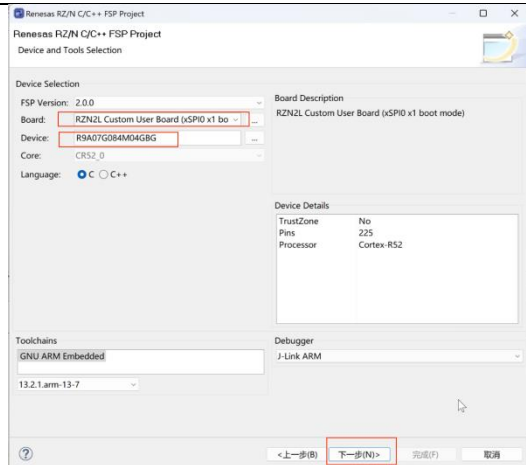


3. 设置项目名称: SPI, 选择工程保存路径

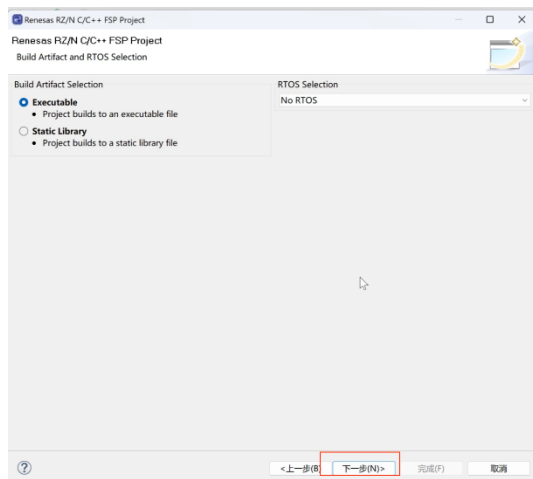


4. Boot 模式选择 RZN2L Custom User Board (XSPI0 x1 boot mode)

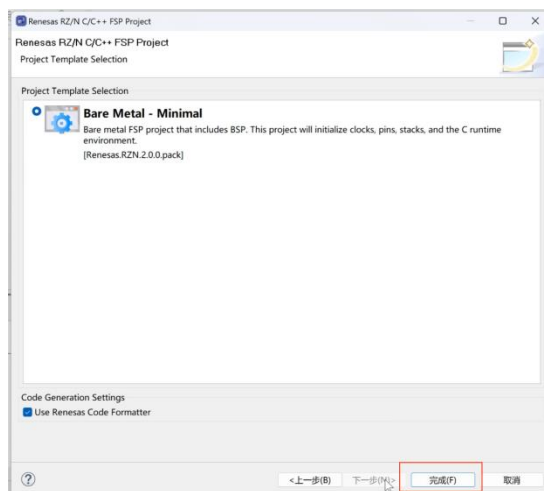
芯片型号: R9A07G084M04GBG



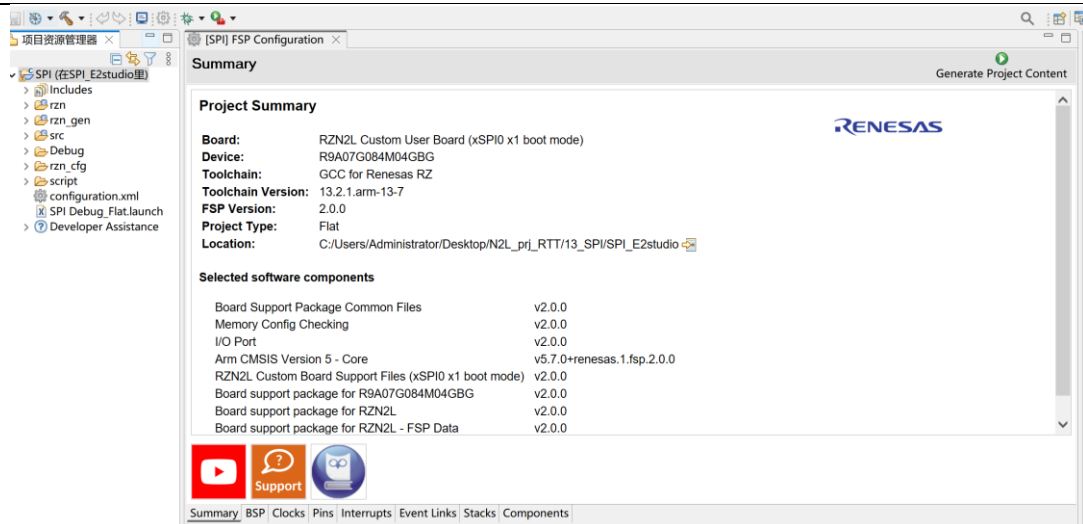
5. 直接下一步



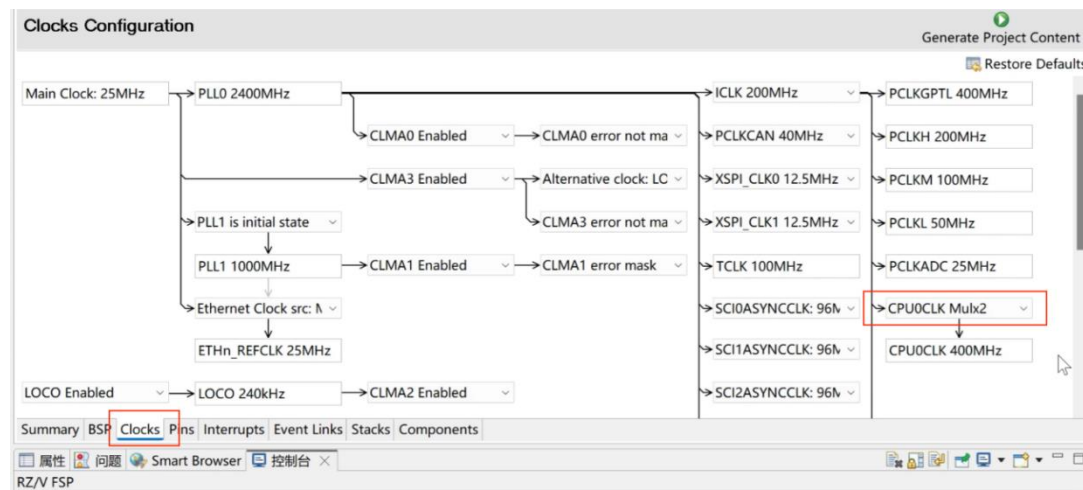
6. 点击 完成



7. 进入到工程界面



8. 设置时钟 CPU0CLK 400MHZ



3.2

9. 配置 XSPI0

The screenshot shows the 'Select Pin Configuration' window with 'R9A07G084M04GBG.pincfg' selected. The 'Pin Selection' tree on the left has 'Connectivity:XSPI' expanded, with 'XSPI0' selected. The 'Pin Configuration' table on the right shows the following settings:

Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Custom(1.8V)		
Input/Output			
XSPI0_CKN	None		
XSPI0_CKP	✓ P14_6		
XSPI0_CS0#	✓ P15_7		
XSPI0_CS1#	None		
XSPI0_DS	None		
XSPI0_ECS0#	None		
XSPI0_ECS1#	None		
XSPI0_INT0#	None		
XSPI0_INT1#	None		
XSPI0_IO0	✓ P14_7		
XSPI0_IO1	✓ P15_0		

Module name: XSPI0

10. P143 引脚配置

P143 设置输出模式，初始电平 0

The screenshot shows the 'Select Pin Configuration' window with 'R9A07G084M04GBG.pincfg' selected. The 'Pin Selection' tree on the left has 'P14' expanded, with 'P14_3' selected. The 'Pin Configuration' table on the right shows the following settings:

Name	Value
Comment	
Mode	Output mode (Low & Not Into Input)
Pull up/down	None
Output Type	CMOS
Drive Capacity	Low
Region	Safety
Schmitt Trigger	None
Slew Rate	Slow
Input/Output	
P14_3	✓ GPIO

Module name: P14_3
Port Capabilities: ETHER, ETH0, ETH0, COL, ETHER, ETH0, ETH0, COL

11. 配置 SCI3

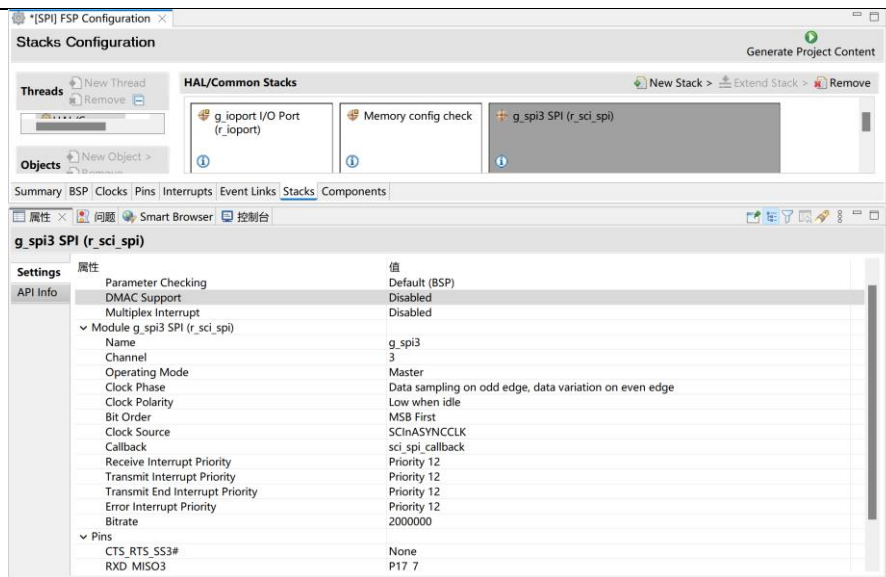
The screenshot shows the 'Pin Configuration' window with 'R9A07G084M04GBG.pincfg' selected. The 'Pin Selection' tree on the left has 'Connectivity:SCI' expanded, with 'SCI3' selected. The 'Pin Configuration' table on the right shows the following settings:

Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Simple SPI mode		
Input/Output			
SS3	None		
MISO3	✓ P17_7		
SCK3	✓ P03_7		
MOSI3	✓ P18_0		

Module name: SCI3
Usage: In initial register value, when CCR0.TE bit is 0, and the terminal function is TXDn, the terminal outputs high.

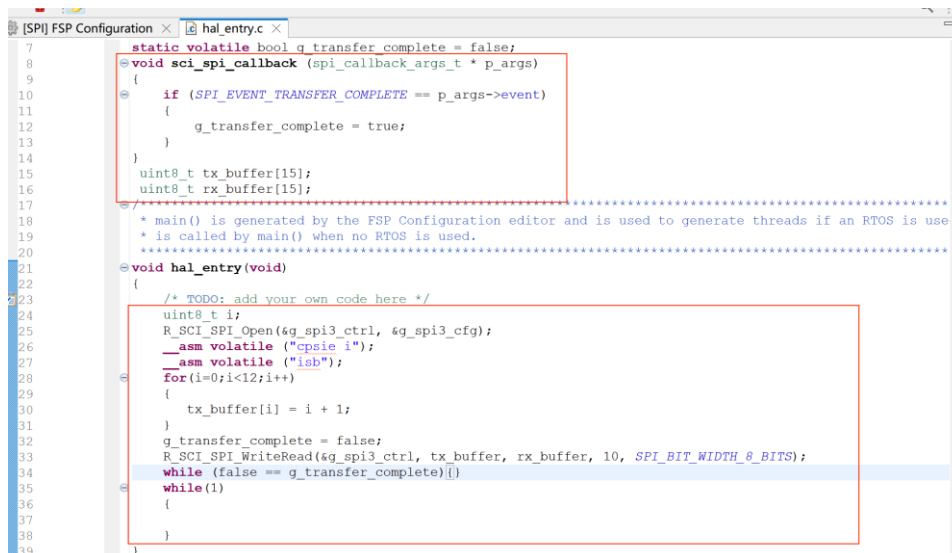
12. 添加 new Stack: SPI

New Stack---Connectivity--SPI(r_sci_spi) 删除默认的 DMA



13. 点击：Generate Project Content 生成代码

编写用户代码



3.3

- 下载代码到开发板

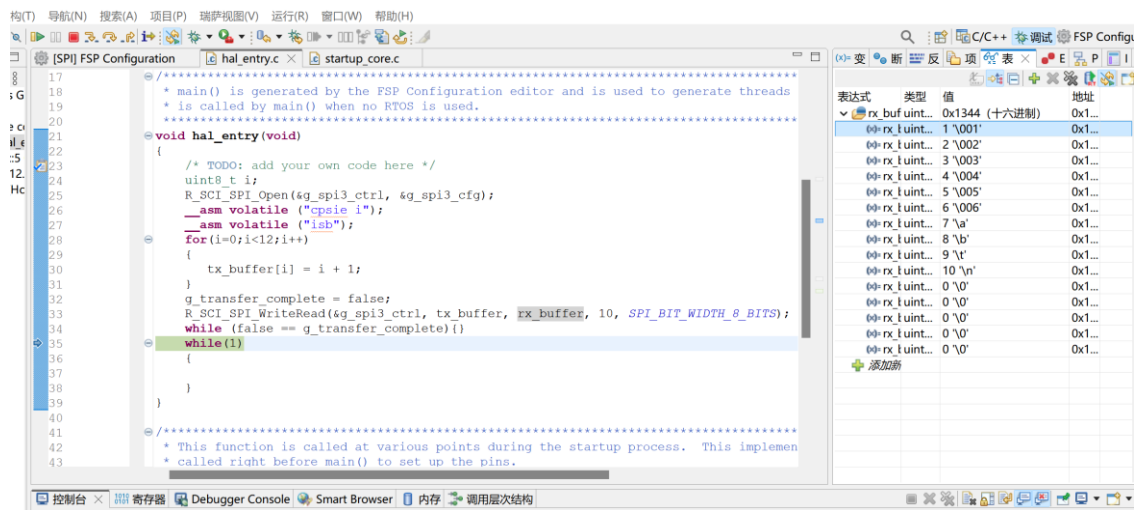
编译代码，无报错。

E2studio 在线仿真下载，全速运行

3.4

➤ 全速运行代码。

暂停仿真，watch 窗口观察 rx_buffer 收到 SPI 发送的数据。



本节完