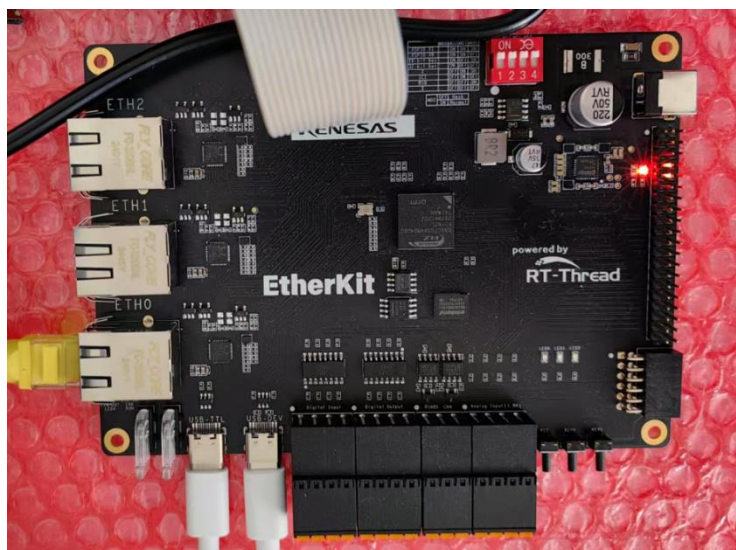


RZN2L CANFD 操作手册-----基于 Etherkit 开发板

简介

本应用笔记介绍了基于 RZ/N2 Etherkit 开发板的定时器 CANFD 的操作。分别介绍 IDE IAR 和 E2studio 软件下的操作。



开发工具 <ul style="list-style-type: none"> • IDE: IAR EW for Arm 9.50.2 E2studio 2024-01.1 • FSP: RZ/N2 FSP V2.0 • 仿真器: Jlink V12 	实验材料 <ul style="list-style-type: none"> • Etherkit 开发板 • Jlink 仿真器, 需支持瑞萨 R52 内核 • Usb 转 CANFD 模块
--	---

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

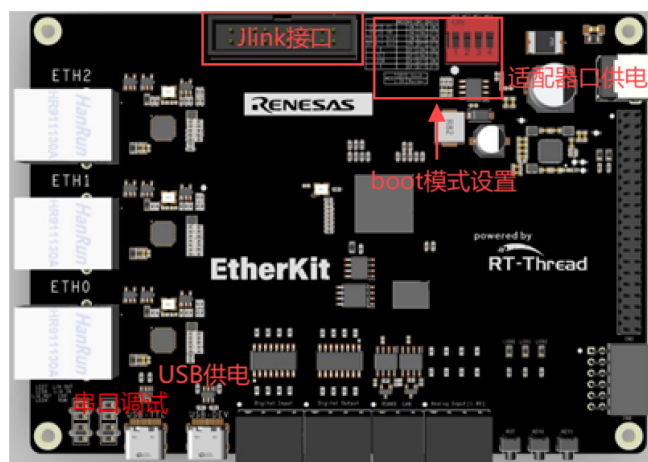
1 .硬件设置及软件安装

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

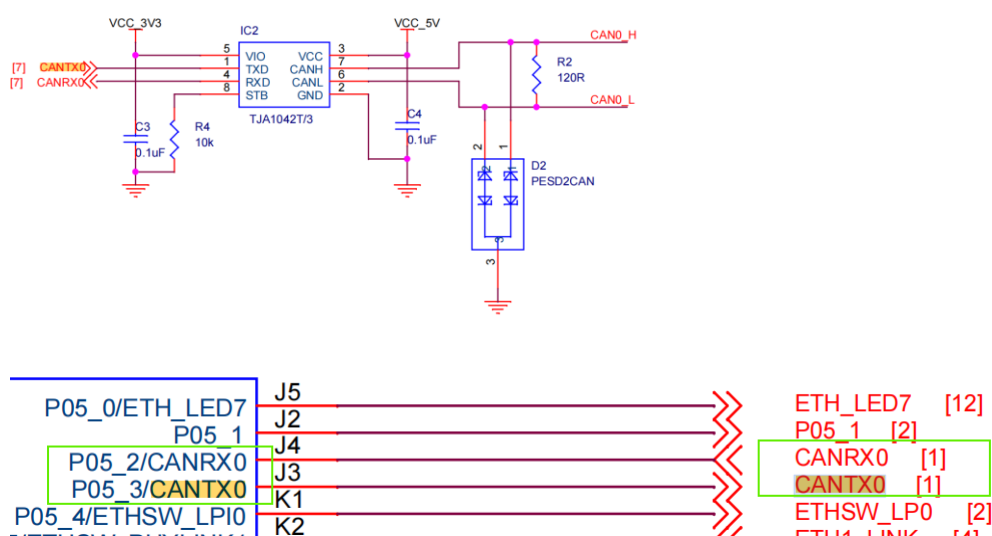


1.2

硬件原理图：

板载资源 CAN0 CAN1 接口，本实验用 CAN0：

CAN0



本节完

2 .IAR 环境工程介绍

本节介绍 IAR 环境下 CANFD 工程介绍。

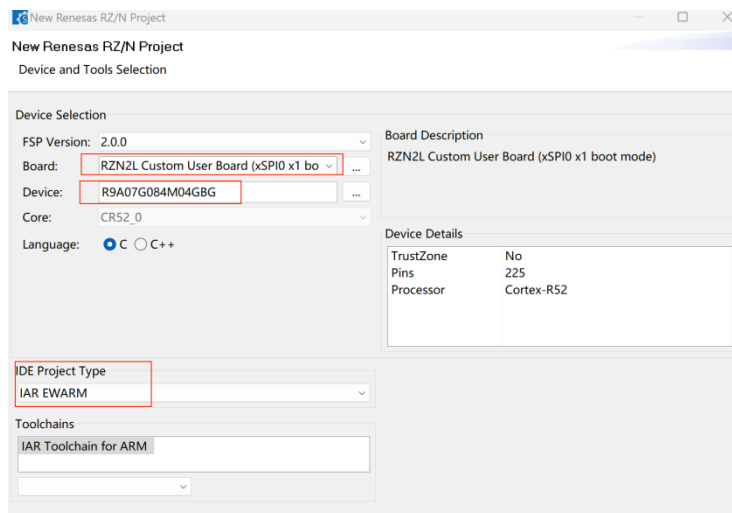
2.1

● 打开 FSP 新建工程：CANFD

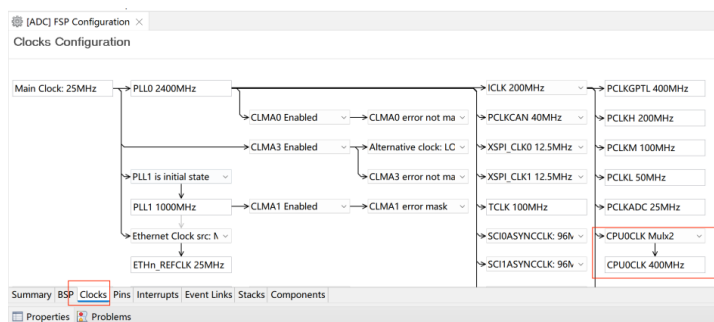
1. Boot 模式选择 RZN2L Custom User Board (XSPI0 x1 boot mode)

芯片型号：R9A07G084M04GBG

IDE Project Type 选择 IAR EWARM

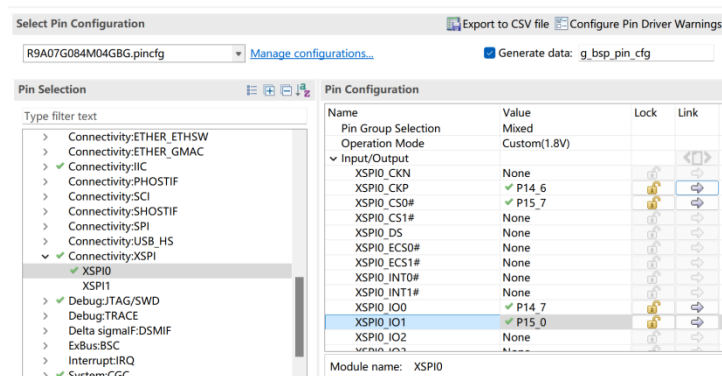


1. 主时钟设置 400MHZ



2. 配置 XSPI0

Pin Configuration



3. P143 引脚配置

P143 设置输出模式，初始电平 0,做为指示灯。

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 CAN0

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

- > P24
- > Other Pins
- > Peripherals
 - > Connectivity:CANFD
 - CANFD0**
 - CANFD1
 - > Connectivity:ETHER_ESC
 - > Connectivity:ETHER_ETH
 - > Connectivity:ETHER_ETHSW
 - > Connectivity:ETHER_GMAC
 - > Connectivity:IIC
 - > Connectivity:PHOSTIF
 - > Connectivity:SCI

Pin Configuration

Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Enabled		
Input/Output			
CANRX0	P05_2		
CANRXDP0	None		
CANTX0	P05_3		
CANTXDP0	None		

Module name: CANFD0

Pin Function Pin Number

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

Properties Problems

5. 添加 new Stack: CANFD

New Stack---Connectivity--CAN FD(r_canfd)

Stacks Configuration

Threads
New Thread Remove

- HAL/Common
 - g_ioport I/O Port (r_ioport)
 - Memory config check
 - g_canfd0 CAN FD (r_canfd)

HAL/Common Stacks
New Stack >

- g_ioport I/O Port (r_ioport)
- Memory config check
- g_canfd0 CAN FD (r_canfd)**

Objects New Object > Remove

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

Properties Problems

g_canfd0 CAN FD (r_canfd)

Settings	Property	Value
> Global Error Interrupt	> Flexible Data (FD)	
	> Reception	
	> Parameter Checking	Default (BSP)
	> Multiplex Interrupt	Disabled
	> Transmission Priority	Buffer Number
DLC Check		Disabled
	Clock Source	PCLKCAN
Module g_canfd0 CAN FD (r_canfd)	> General	
	> Bitrate	
	> Interrupts	
	> Transmit Interrupts	
	> Channel Error Interrupts	
	> Filter List Array	p_canfd0 afl
Pins		

6. 波特率这里设置 500000, 根据需要设置

Threads

HAL/Common

- g_ioport I/O Port (r_ioport)
- Memory config check
- g_canfd0 CAN FD (r_canfd)

Objects

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

Properties Problems

g_canfd0 CAN FD (r_canfd)

Settings	Property	Value
	DLC Check	Disabled
	Clock Source	PCLKCAN
	Module g_canfd0 CAN FD (r_canfd)	
	General	
	Name	g_canfd0
	Channel	0
	Bitrate	
	Automatic	
	Nominal Rate (bps)	500000
	FD Data Rate (bps)	500000
	Sample Point (%)	75
	Manual	
	Delay Compensation	Enable
	Interrupts	
	Transmit Interrupts	
	Channel Error Interrupts	

发送中断使能

Threads

HAL/Common

- g_ioport I/O Port (r_ioport)
- Memory config check
- g_canfd0 CAN FD (r_canfd)

Objects

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

Properties Problems

g_canfd0 CAN FD (r_canfd)

Settings	Property	Value
	Channel	0
	Bitrate	
	Automatic	
	Nominal Rate (bps)	500000
	FD Data Rate (bps)	500000
	Sample Point (%)	75
	Manual	
	Delay Compensation	Enable
	Interrupts	
	Transmit Interrupts	
	TXMB 0	<input checked="" type="checkbox"/>
	TXMB 1	<input type="checkbox"/>
	TXMB 2	<input type="checkbox"/>
	TXMB 3	<input type="checkbox"/>
	TXMB 4	<input type="checkbox"/>
	TXMB 5	<input type="checkbox"/>

7. 点击: Generate Project Content 生成代码

Generate Project Content

New Stack > Extend Stack > Remove

2.2

8. 打开生成的代码

- 仿真器由 Ijet 切换为 Jlink
- 编写用户代码，上电 CAN 发送一帧数据，随后 CAN 接收到一帧数据，发送接收到的数据。

```

22 void hal_entry(void)
23 {
24     /* TODO: add your own code here */
25     __asm volatile ("cpsie i");
26
27     /* Initialize the CAN module */
28     fsp_err_t err = R_CANFD_Open(&g_canfd0_ctrl, &g_canfd0_cfg);
29     /* Handle any errors. This function should be defined by the user. */
30     handle_error(err);
31
32     /*----- Send data from CAN CH0 (Use Classical CAN frame)-----*/
33     /* Setup frame to write to CAN ID 0x20 */
34     g_can_tx0_frame.id = CAN_EXAMPLE_ID;
35     g_can_tx0_frame.id_mode = CAN_ID_MODE_STANDARD;
36     g_can_tx0_frame.type = CAN_FRAME_TYPE_DATA;
37     g_can_tx0_frame.data_length_code = 8;
38     g_can_tx0_frame.options = 0;
39     /* Write some data to the transmit frame */
40     for (uint32_t i = 0; i < 8; i++)
41     {
42         g_can_tx0_frame.data[i] = (uint8_t)(64 - i);
43     }
44
45     /* Send data on the bus */
46     R_CANFD_Write(&g_canfd0_ctrl, CANFD_TX_MB_0, &g_can_tx0_frame);
47

```

- Rebuild All---编译工程 无报错

```

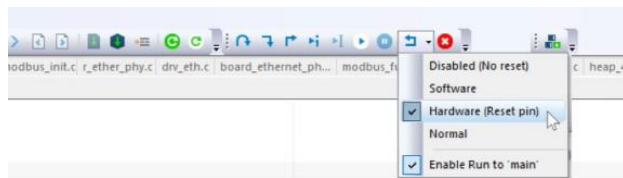
Build
Messages

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



打开 can 上位机调试助手，第一帧数据之后，上位机发送数据，然后开发板返回接收到的数据。

HCANView 通道0

打开文件 保存数据 实时保存 清空数据 存储配置 波特率检测 在线配置 离线配置 工程曲线

设备配置

选择通道: 0
波特率: Kbd 500
工作模式: 正常模式
☒ 离线唤醒 ☒ 自动重传 ☒ 接通电阻
复位 暂停 关闭

接收窗口

序号	时间标识 (us)	传输方向	帧类型	帧格式	帧ID	数据长度	帧数据
1	0628.351...	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29
2	0009.968...	发送	标准	数据	00000020	8	29 23 24 25 26 27 28 29
3	0000.000...	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29
4	0000.941...	发送	标准	数据	00000020	8	29 23 24 25 26 27 28 29
5	0000.000...	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29
6	0000.636...	发送	标准	数据	00000020	8	29 23 24 25 26 27 28 29
7	0000.000...	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29
8	0000.637...	发送	标准	数据	00000020	8	29 23 24 25 26 27 28 29
9	0000.000...	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29

接收显示设置

☒ 间隔时间 ☐ 连续时间
☒ 发送帧 ☐ 错误帧
☐ 统计模式 ☒ 滚动显示
数据显示格式: 16进制

本节完

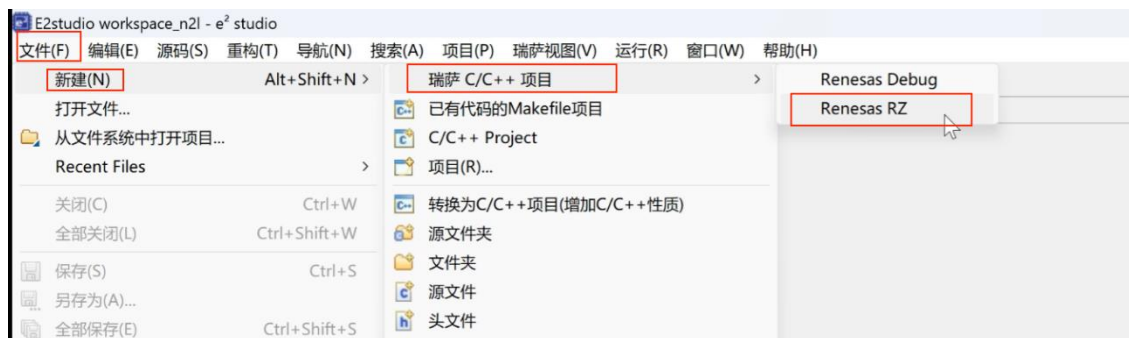
3 .E2studio 环境工程介绍

本节介绍使用 E2studio 环境创建 CANFD 工程。

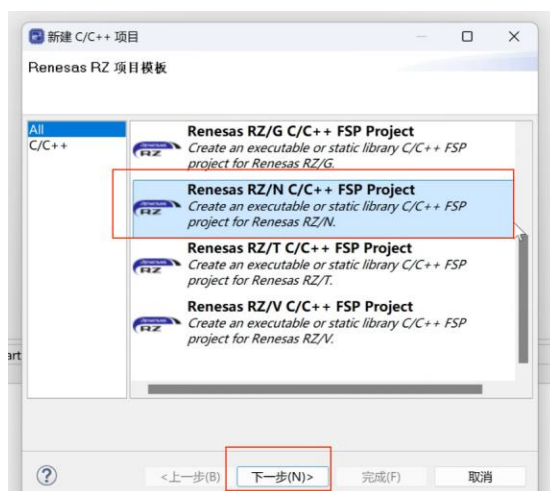
3.1

● 打开 E2studio, 新建工程

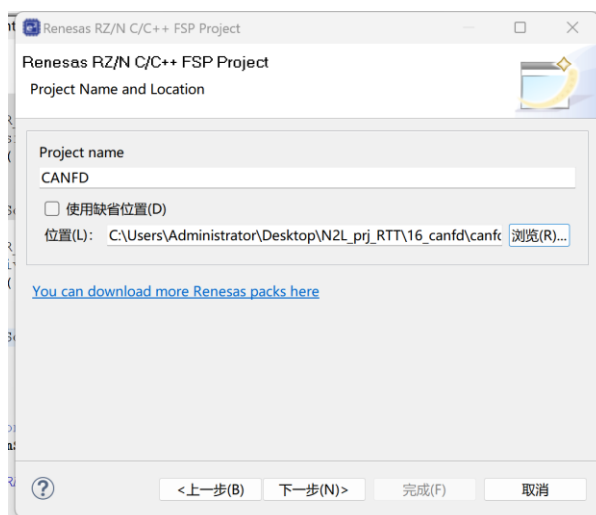
1. 选择 文件--新建--瑞萨 C/C++项目--Renesas RZ:



2. 选择 Renesas RZ/N C/C++ FSP Project

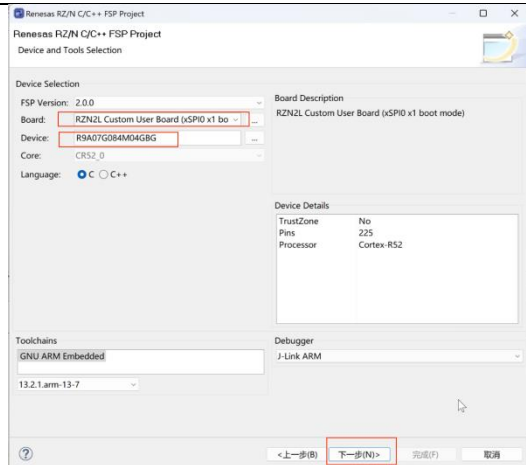


3. 设置项目名称: CANFD, 选择工程保存路径

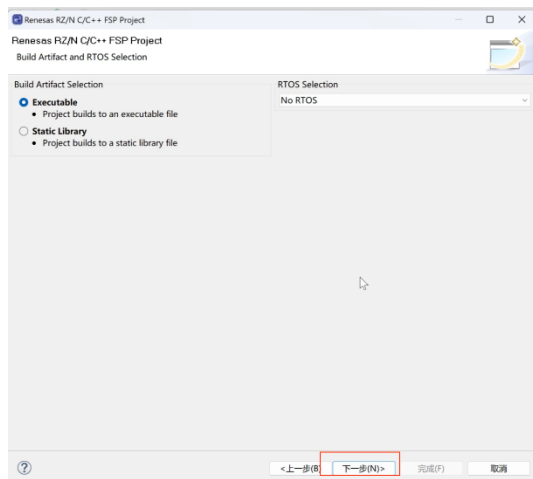


4. Boot 模式选择 RZN2L Custom User Board (XSPI0 x1 boot mode)

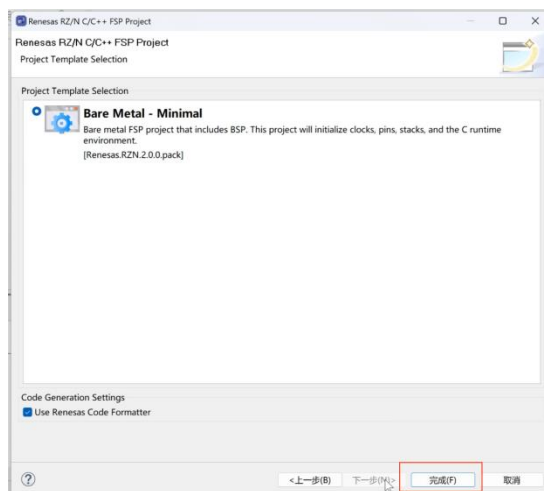
芯片型号: R9A07G084M04GBG



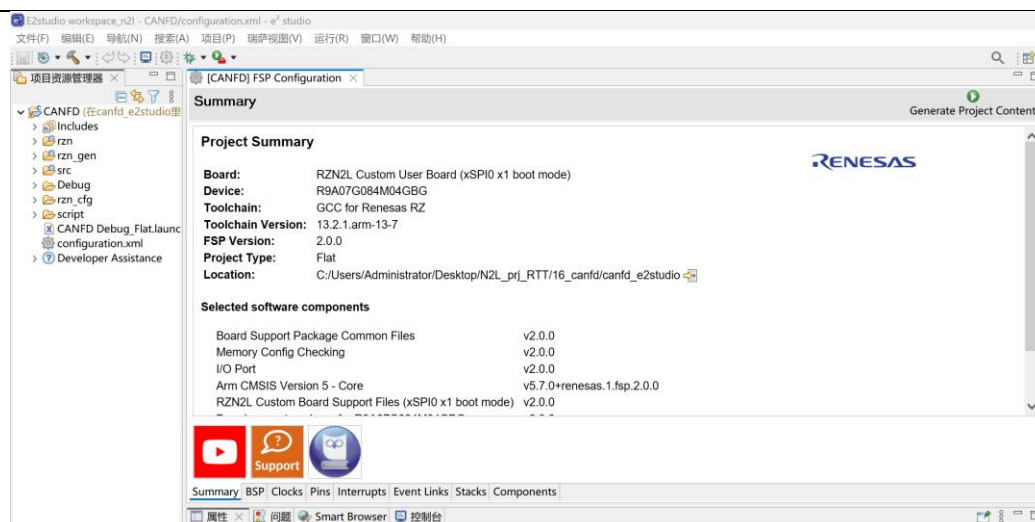
5. 直接下一步



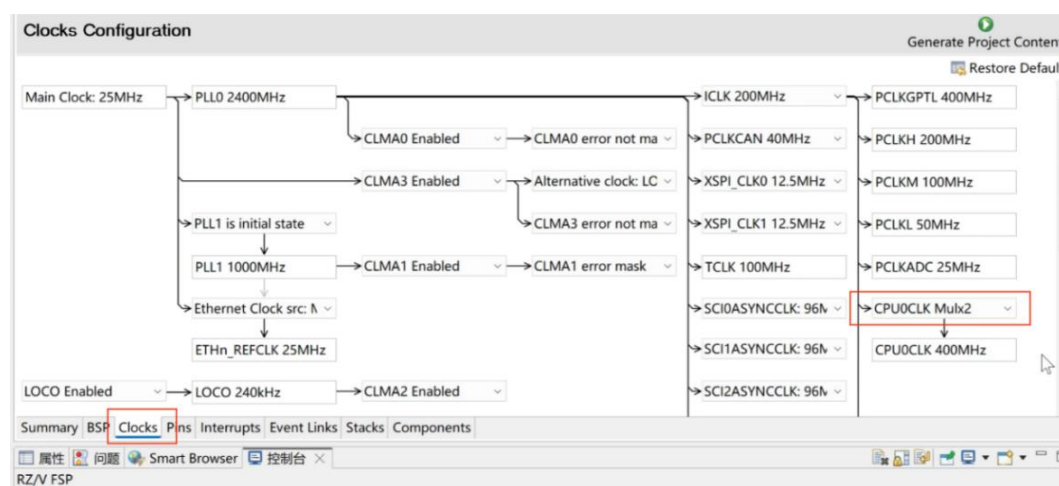
6. 点击 完成



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 for R9A07G084M04GBG.pincfg. The 'Pin Selection' list 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		
XSPI0_IO2	None		

Module name: XSPI0

10. P143 引脚配置

P143 设置输出模式，初始电平 0, 做为指示灯

The screenshot shows the 'Select Pin Configuration' window for R9A07G084M04GBG.pincfg. The 'Pin Selection' list 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_ETH1: ETH1_COL

Pin Function | Pin Number
Summary | BSP | Clocks | Pins | Interrupts | Event Links | Stacks | Components

11. 配置 CANFD0

The screenshot shows the 'Select Pin Configuration' window for R9A07G084M04GBG.pincfg. The 'Pin Selection' list on the left has 'Connectivity:CANFD' expanded, with 'CANFD0' selected. The 'Pin Configuration' table on the right shows the following settings:

Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Enabled		
Input/Output			
CANRX0	✓ P05_2		
CANRXDP0	None		
CANTX0	✓ P05_3		
CANTXDP0	None		

Module name: CANFD0

Pin Function | Pin Number
Summary | BSP | Clocks | Pins | Interrupts | Event Links | Stacks | Components

12. 添加 new Stack: CANFD

New Stack---Connectivity--CAN FD(r_canfd)

波特率设置 500000

Stacks Configuration

Threads: New Thread, Remove

HAL/Common Stacks: New Stack >

Objects: New Object >, Remove

Summary: BSP, Clocks, Pins, Interrupts, Event Links, Stacks, Components

属性, 问题, Smart Browser, 控制台

g_canfd0 CAN FD (r_canfd)

属性	值
DLC Check	Disabled
Clock Source	PCLKCAN
Module g_canfd0 CAN FD (r_canfd)	
General	
Name	g_canfd0
Channel	0
Bitrate	
Automatic	
Nominal Rate (bps)	500000
FD Data Rate (bps)	500000
Sample Point (%)	75
Manual	
Delay Compensation	Enable
Interrupts	
Transmit Interrupts	
Channel Error Interrupts	
Filter List Array	p_canfd0_afl
Pins	

发送中断使能

Threads: New Thread, Remove

HAL/Common Stacks: New Stack >

Objects: New Object >, Remove


Summary: BSP, Clocks, Pins, Interrupts, Event Links, Stacks, Components

Properties, Problems

g_canfd0 CAN FD (r_canfd)

Property	Value
Module g_canfd0 CAN FD (r_canfd)	
General	
Bitrate	
Interrupts	
Transmit Interrupts	
TXMB 0	<input checked="" type="checkbox"/>
TXMB 1	<input type="checkbox"/>
TXMB 2	<input type="checkbox"/>
TXMB 3	<input type="checkbox"/>
TXMB 4	<input type="checkbox"/>

13. 点击: Generate Project Content 生成代码



Generate Project Content

14.

➤ 编写用户代码, 上电 CAN 发送一帧数据, 随后 CAN 接收到一帧数据, 发送接收到的数据。

```

22
23
24 void hal_entry(void)
25 {
26     /* TODO: add your own code here */
27     __asm volatile ("cpsie i");
28
29     /* Initialize the CAN module */
30     fsp_err_t err = R_CANFD_Open(&g_canfd0_ctrl, &g_canfd0_cfg);
31     /* Handle any errors. This function should be defined by the user. */
32     handle_error(err);
33
34     /*----- Send data from CAN CH0 (Use Classical CAN frame)-----*/
35     /* Setup frame to write to CAN ID 0x20 */
36     g_can_tx0_frame.id = CAN_EXAMPLE_ID;
37     g_can_tx0_frame.id_mode = CAN_ID_MODE_STANDARD;
38     g_can_tx0_frame.type = CAN_FRAME_TYPE_DATA;
39     g_can_tx0_frame.data_length_code = 8;
40     g_can_tx0_frame.options = 0;
41     /* Write some data to the transmit frame */
42     for (uint32_t i = 0; i < 8; i++)
43     {
44         g_can_tx0_frame.data[i] = (uint8_t)(64 - i);
45     }
46
47     /* Send data on the bus */
48     err = R_CANFD_Write(&g_canfd0_ctrl, CANFD_TX_MB_0, &g_can_tx0_frame);
49     handle_error(err);

```

CDT Build Console [CANFD]
arm-none-eabi-size --format=berkeley "CANFD.elf"
text data bss dec hex filename
6860 6068 27396 40324 9d84 CANFD.elf
10:51:37 Build Finished. 0 errors, 0 warnings. (took 1s.894ms)

➤ 编译工程，无报错

3.3

打开 can 上位机调试助手，全速运行代码，收到第一帧数据之后，上位机发送数据，然后开发板返回接收到的数据。

序号	时间标识 (us)	传输方向	帧类型	帧格式	帧ID	数据长度	帧数据
1	0628.351....	接收	标准	数据	00000020	8	40 3F 3E 3D 3C 3B 3A 39
2	0009.968....	发送	标准	数据	00000020	8	29 23 24 25 26 27 28 29
3	0000.000....	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29
4	0000.941....	发送	标准	数据	00000020	8	29 23 24 25 26 27 28 29
5	0000.000....	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29
6	0000.636....	发送	标准	数据	00000020	8	29 23 24 25 26 27 28 29
7	0000.000....	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29
8	0000.637....	发送	标准	数据	00000020	8	29 23 24 25 26 27 28 29
9	0000.000....	接收	标准	数据	00000020	8	29 23 24 25 26 27 28 29

本节完