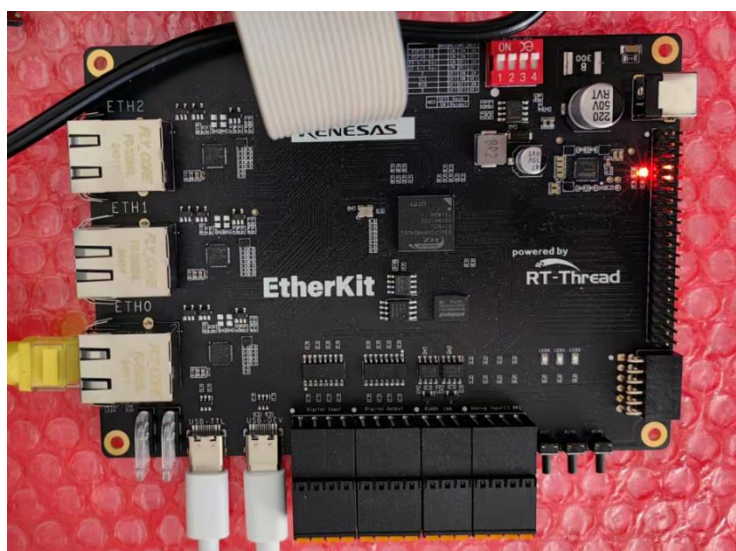


RZN2L Modbus 例程操作手册-----基于 Etherkit 开发板

简介

本应用笔记介绍了基于 RZ/N2 Etherkit 开发板的 Modbus 例程操作。分别介绍 IDE IAR 和 E2studio 软件下的操作，及从官方 SDK 移植过来需要修改的内容。

本例由官方例程 RZN2L_Modbus_RSK_rev0200 中 modbus_tcp 移植而来。



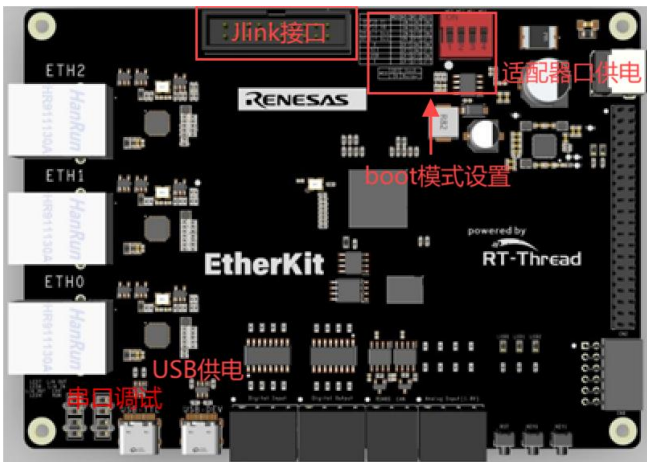
开发工具 <ul style="list-style-type: none"> • IDE: IAR EW for Arm 9.50.2 E2studio 2024-01.1 • FSP: RZ/N2 FSP V2.0 • 仿真器: Jlink V12 	实验材料 <ul style="list-style-type: none"> • Etherkit 开发板 • Jlink 仿真器，需支持瑞萨 R52 内核
--	--

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	4
4.连接 ModbusDemoApplication 测试	5
5.官方 SDK 移植到 Etherkit 修改位置	6

1 .硬件设置及软件安装

本节 EtherKit 开发板硬件设置。

<p>1.1</p>	<p>开发板设置：</p> <ul style="list-style-type: none"> ● 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 ● 准备网线连接电脑网口，ETH0 连接电脑网口  <p>The image shows the EtherKit development board with several components highlighted by red boxes and labels: 'Jlink接口' (Jlink interface) at the top center, '适配器口供电' (Adapter port power supply) at the top right, 'boot模式设置' (Boot mode setting) in the center, 'USB供电' (USB power supply) at the bottom left, and '串口调试' (Serial port debugging) at the bottom left. The board also features Ethernet ports labeled ETH0, ETH1, and ETH2, and is powered by RT-Thread.</p>
<p>1.2</p>	<p>软件安装：</p> <ul style="list-style-type: none"> ● 安装 IAR EW for Arm 9.50.2 ● 安装 FSP 2.0: RZN2L: setup_rznfsp_v2_0_0_rzsc_v2024-01.1.exe ● 安装 E2studio : RZN2L: setup_rznfsp_v2_0_0_e2s_v2024-01.1.exe

本节完

2 .IAR 环境工程介绍

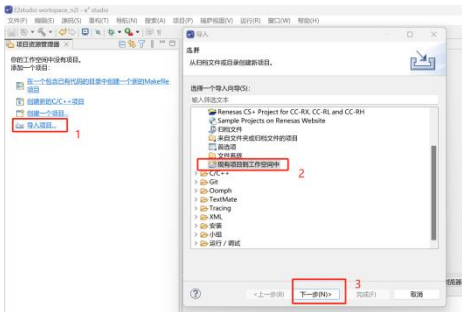
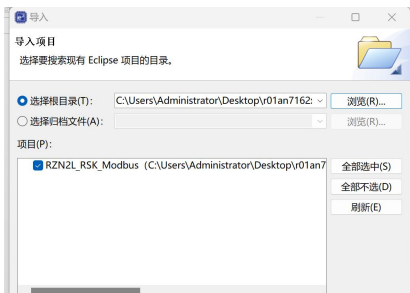
本节介绍 IAR 环境下 modbus 工程。

2.1	<ul style="list-style-type: none"> ● 打开工程: RZN2L_RSK_Modbus <div data-bbox="284 434 683 591"> <div>RZN2L_RSK_Modbus.ewd</div> <div>RZN2L_RSK_Modbus.ewp</div> <div>RZN2L_RSK_Modbus.ewt</div> <div>RZN2L_RSK_Modbus</div> </div> <p>如果使用 FSP 重新生成了代码， 从 script 文件夹复制 linker 文件 “fsp_xxx_xxx.icf” 替换工程下链接文件 modbus_tcp_(lwip)\script\fsp_xxx_xxx.icf copy to modbus_tcp_(lwip)\project\ewarm\script\fsp_xxx_xxx.icf 根据 boot 模式选择 icf 文：. RAM mode: <i>fsp_ram_execution.icf</i> xSPI mode: <i>fsp_xspio_boot.icf</i></p>
2.2	<ul style="list-style-type: none"> ● Rebuild All---编译工程 无报错 <div data-bbox="256 1070 1134 1317"> <div>Build</div> <div> <div>Messages</div> <div>.postbuild 50% Generating Smart Bundle100% Generating Smart Bundle</div> <div>Total number of errors: 0 Total number of warnings: 153 Resolving dependencies... Build succeeded</div> </div> </div>
2.3	<ul style="list-style-type: none"> ● Download and Debug ---下载程序 <ol style="list-style-type: none"> 1. 开发板 ETH0 网口接电脑 2. Jlink 正确连接，板子上电 3. 下载工程到开发板，进入仿真界面 4. Debug 复位设置为 Hardware <div data-bbox="252 1653 890 1832"> </div> <ol style="list-style-type: none"> 5. 全速运行代码 6. 打开上位机 ModbusDemoApplication，连接开发板测试

本节完

3 .E2studio 环境工程介绍

本节介绍使用 E2studio 环境 Ethernet 工程。

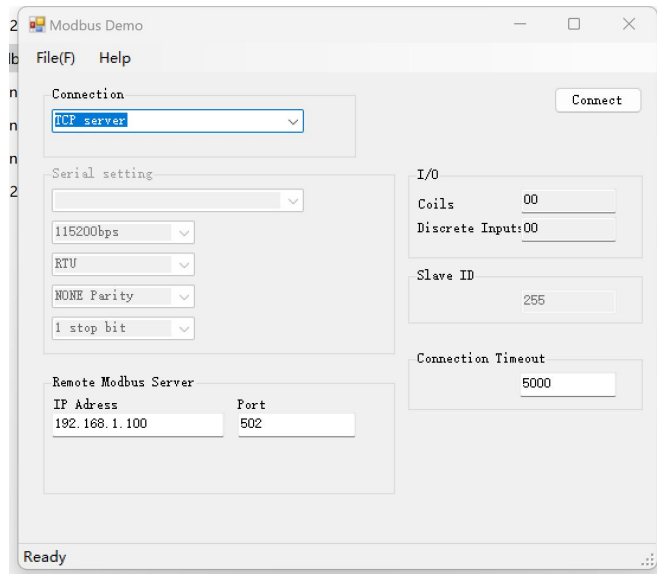
<p>3.1</p>	<ul style="list-style-type: none"> ● 打开 E2studio，导入工程 <p>1. 选择 文件--导入--下一步:</p>  <p>2. 浏览--指定到工程文件夹---完成</p> 
<p>3.2</p>	<ul style="list-style-type: none"> ● 编译工程，无报错 <p>如果使用 FSP 重新生成了代码， 从 script 文件夹复制 linker 文件“fsp_xxx_xxx.ld”替换工程下链接文件 modbus_tcp_(lwip)\script\fsp_xxx_xxxld copy to modbus_tcp_(lwip)\project\ewarm\script\fsp_xxx_xxx.ld</p> <p>根据 boot 模式选择 icf 文：.</p> <p>RAM mode: <i>fsp_ram_execution.ld</i></p> <p>xSPI mode: <i>fsp_xspio_boot.ld</i></p>
<p>3.3</p>	<ul style="list-style-type: none"> ● 下载代码到开发板，两种方式 <p>1. 通过 j-flash 下载代码</p> <p>2. E2studio 在线仿真下载，全速运行，关掉仿真</p> <p>3. 复位开发板，让程序运行</p>
<p>3.4</p>	<ul style="list-style-type: none"> ● 打开上位机 ModbusDemoApplication，连接开发板测试

本节完

4 .连接 ModbusDemoApplication 测试

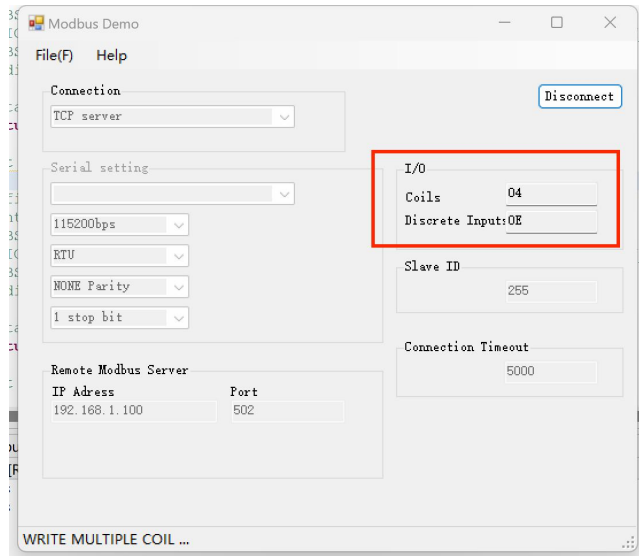
4.1

- 启动 ModbusDemoApplication, 如下配置, 点击 Connect 按钮



4.2

- 链接后, 操作用户按键观察 UI 界面数字变化, 及开发板 led 灯状态。



本节完

5 .官方 SDK 移植到 Etherkit 修改位置

5.1

● E2studio 环境

1. PHY0 配置修改

Summary BSP Clocks Pins Interrupts Event Links Stacks Components

问题 控制台 属性 x Smart Browser Smart Manual 内存 包含浏览器 搜索 调试 调用层次结构

g_ether_phy0 Ethernet (r_ether_phy)

属性	值
User Own Target	Enabled
Module g_ether_phy0 Ethernet (r_ether_phy)	
Name	g_ether_phy0
Channel	0
PHY-LSI Address	1
PHY-LSI Reset Completion Timeout	0x00020000
Flow Control	Disable
Port Type	Ethernet
Phy LSI type	User own PHY
Port Custom Init Function	NULL
Select MDIO type	GMAC
Auto Negotiation	ON
Speed	100M
Duplex	FULL
Reset Driver	12

PHY1 配置修改

BSP Clocks Pins Interrupts Event Links Stacks Components

控制台 属性 x Smart Browser Smart Manual 内存 包含浏览器 搜索 调试 调用层次结构

r_phy1 Ethernet (r_ether_phy)

属性	值
KSZ8041 Target	Disabled
User Own Target	Enabled
Module g_ether_phy1 Ethernet (r_ether_phy)	
Name	g_ether_phy1
Channel	1
PHY-LSI Address	2
PHY-LSI Reset Completion Timeout	0x00020000
Flow Control	Disable
Port Type	Ethernet
Phy LSI type	User own PHY
Port Custom Init Function	NULL
Select MDIO type	GMAC
Auto Negotiation	ON
Speed	100M
Duplex	FULL

2. Pin 脚配置修改

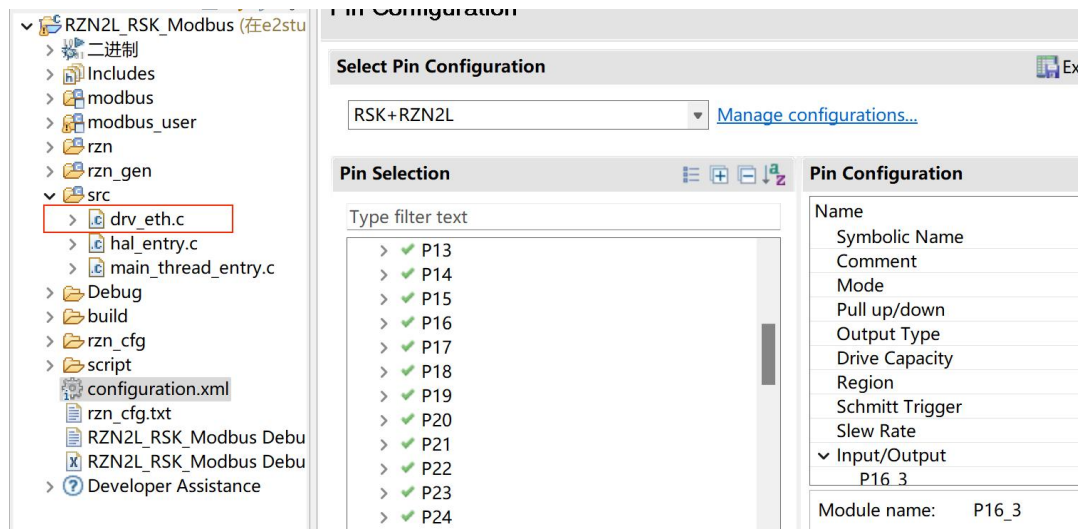
用户 LED 灯 P140 P141 P143 配置为输出

用户按键 P142 P163 配置为输入

修改后, 重新生成代码, 替换 linker 文件 fsp_xspi0_boot.ld。

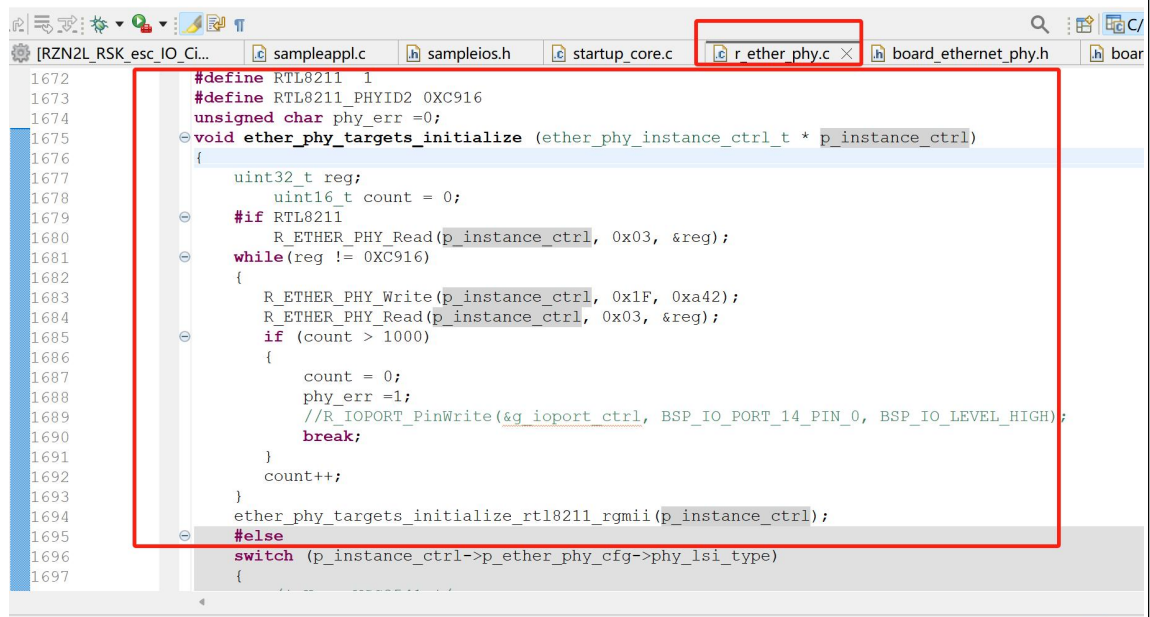
3. RTL8211 驱动文件添加

将 drv_eth.c 添加至工程文件夹 src 下，添加到工程：

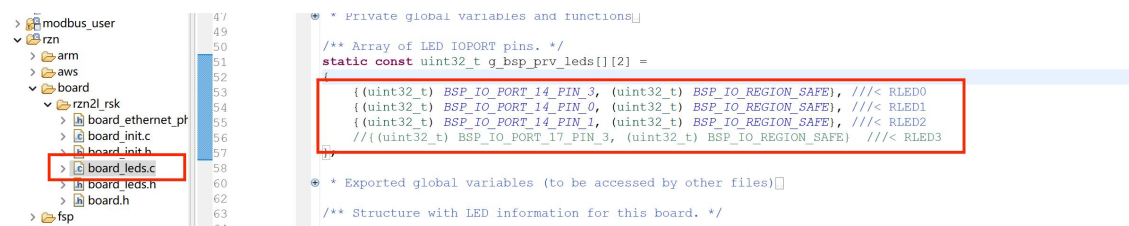


4. PHY 初始化代码修改

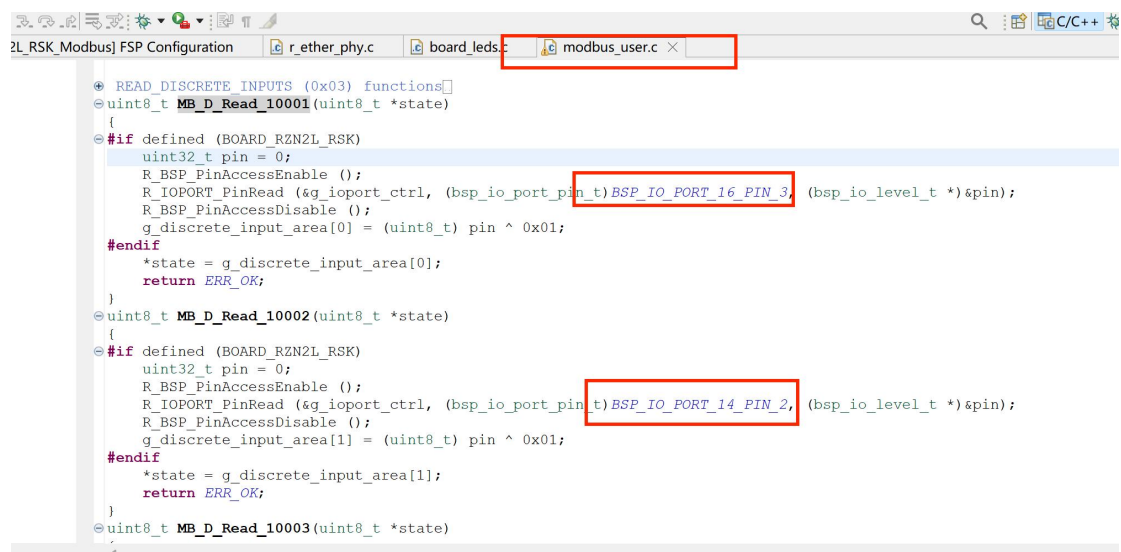
如下图，修改 r_ether_phy.c 文件，添加 RTL8211 代码



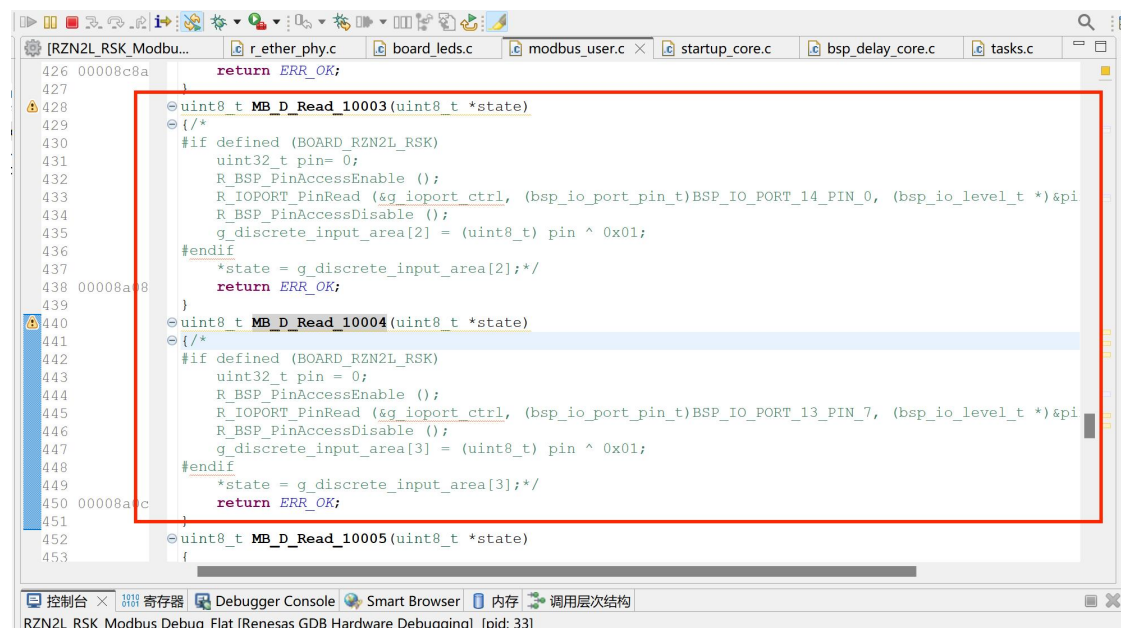
5. 用户 LED，定义修改



6. 用户按键代码修改



- 屏蔽如下代码：



5.2

● IAR 环境

1. PHY0 配置修改

The screenshot shows the IAR IDE interface with the 'FreeRTOS+TCP Stacks' component selected. The 'g_ether_phy0 Ethernet (r_ether_phy)' component is highlighted with a red box. Below it, the 'g_ether_selector0 Ethernet (r_ether_selector)' component is also visible. The 'Properties' window for 'g_ether_phy0 Ethernet (r_ether_phy)' is open, showing the following configuration:

Property	Value
KSZ8041 Target	Disabled
User Own Target	Enabled
Module g_ether_phy0 Ethernet (r_ether_	
Name	g_ether_phy0
Channel	0
PHY-LSI Address	1
PHY-LSI Reset Completion Timeout	0x00020000
Flow Control	Disable
Port Type	Ethernet
Phy LSI type	User own PHY
Port Custom Init Function	ether_phy_targets_initialize_rtl8211_rgmii
Select MDIO type	GMAC
Auto Negotiation	ON
Speed	100M
Duplex	FULL
Reset Port	13
Reset Pin	4

2. PHY1 配置修改

The screenshot shows the IAR IDE interface with the 'FreeRTOS+TCP Stacks' component selected. The 'g_ether_phy1 Ethernet (r_ether_phy)' component is highlighted with a red box. Below it, the 'g_ether_selector1 Ethernet' component is also visible. The 'Properties' window for 'g_ether_phy1 Ethernet (r_ether_phy)' is open, showing the following configuration:

Property	Value
KSZ8041 Target	Disabled
User Own Target	Enabled
Module g_ether_phy1 Ethernet (r_ether_	
Name	g_ether_phy1
Channel	1
PHY-LSI Address	2
PHY-LSI Reset Completion Timeout	0x00020000
Flow Control	Disable
Port Type	Ethernet
Phy LSI type	User own PHY
Port Custom Init Function	ether_phy_targets_initialize_rtl8211_rgmii
Select MDIO type	GMAC
Auto Negotiation	ON
Speed	100M
Duplex	FULL
Reset Port	13

3. Pin 脚配置修改

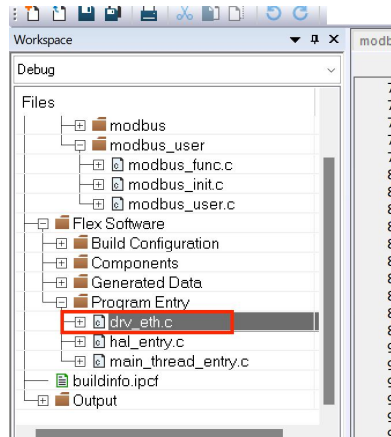
用户 LED 灯 P140 P141 P143 配置为输出

用户按键 P142 P163 配置为输入

修改后，重新生成代码，替换 linker 文件 fsp_xsapi0_boot.icf。

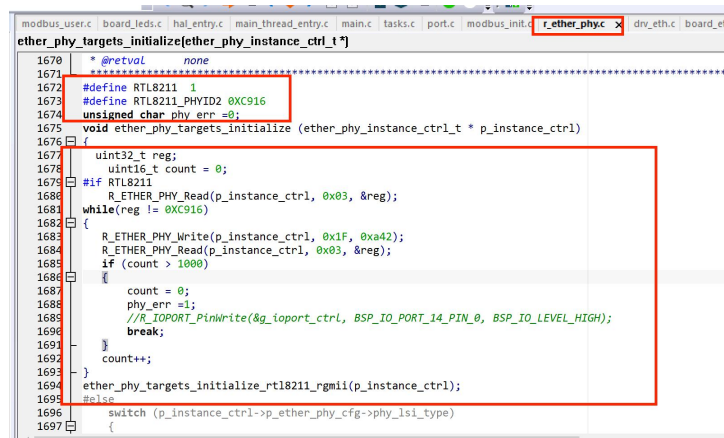
4. RTL8211 驱动文件添加

将 drv_eth.c 添加至工程文件夹 src 下，添加到工程：

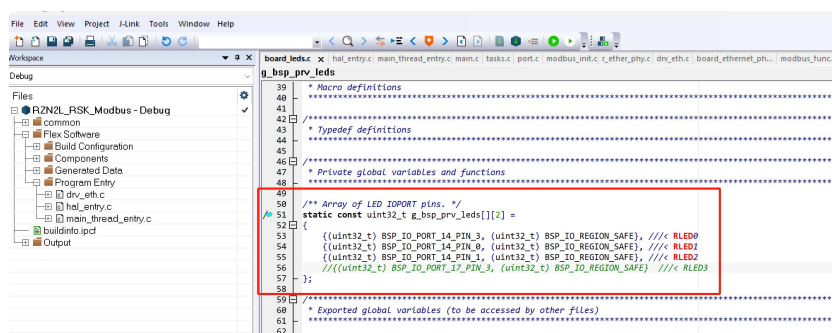


5. PHY 初始化代码修改

如下图，修改 r_ether_phy.c 文件，添加 RTL8211 代码



6. 用户 LED 引脚定义修改



7. 用户按键代码修改

● 修改按键引脚

```

401  /*****
402  READ_DISCRETE_INPUTS (0x03) functions
403  */
404  uint8_t MB_D_Read_10001(uint8_t *state)
405  {
406      #if defined (BOARD_RZN2L_RSK)
407          uint32_t pin = 0;
408          R_BSP_PinAccessEnable ();
409          R_IOPORT_PinRead (&g_ioport_ctrl, (bsp_io_port_pin_t)BSP_IO_PORT_16_PIN_3, (bsp_io_level_t *)&pin);
410          R_BSP_PinAccessDisable ();
411          g_discrete_input_area[0] = (uint8_t) pin ^ 0x01;
412      #endif
413      *state = g_discrete_input_area[0];
414      return ERR_OK;
415  }
416  uint8_t MB_D_Read_10002(uint8_t *state)
417  {
418      #if defined (BOARD_RZN2L_RSK)
419          uint32_t pin = 0;
420          R_BSP_PinAccessEnable ();
421          R_IOPORT_PinRead (&g_ioport_ctrl, (bsp_io_port_pin_t)BSP_IO_PORT_14_PIN_2, (bsp_io_level_t *)&pin);
422          R_BSP_PinAccessDisable ();
423          g_discrete_input_area[1] = (uint8_t) pin ^ 0x01;
424      #endif
425      *state = g_discrete_input_area[1];
426      return ERR_OK;
427  }
428  uint8_t MB_D_Read_10003(uint8_t *state)

```

● 屏蔽如下代码

```

425      *state = g_discrete_input_area[1];
426      return ERR_OK;
427  }
428  uint8_t MB_D_Read_10003(uint8_t *state)
429  {
430      #if defined (BOARD_RZN2L_RSK)
431          uint32_t pin = 0;
432          R_BSP_PinAccessEnable ();
433          R_IOPORT_PinRead (&g_ioport_ctrl, (bsp_io_port_pin_t)BSP_IO_PORT_14_PIN_0, (bsp_io_level_t *)&pin);
434          R_BSP_PinAccessDisable ();
435          g_discrete_input_area[2] = (uint8_t) pin ^ 0x01;
436      #endif
437      *state = g_discrete_input_area[2];
438      return ERR_OK;
439  }
440  uint8_t MB_D_Read_10004(uint8_t *state)
441  {
442      #if defined (BOARD_RZN2L_RSK)
443          uint32_t pin = 0;
444          R_BSP_PinAccessEnable ();
445          R_IOPORT_PinRead (&g_ioport_ctrl, (bsp_io_port_pin_t)BSP_IO_PORT_13_PIN_7, (bsp_io_level_t *)&pin);
446          R_BSP_PinAccessDisable ();
447          g_discrete_input_area[3] = (uint8_t) pin ^ 0x01;
448      #endif
449      *state = g_discrete_input_area[3];
450      return ERR_OK;
451  }
452  uint8_t MB_D_Read_10005(uint8_t *state)

```

本节完