CC3200 实现 TCP Client 双向通讯笔记

1、资料说明

如下目录

①网络说明文档

cc3200 开发文档\网络部分说明\Internet on chip

②OTA 库文档

cc3200 开发文档\库文件\CC3200 Simplelink OTA Extlib API User's Guide

③网络接口文档

cc3200 开发文档\库文件\simplelink_api

2、概念

正式开讲之前，先介绍几个概念，有主意我们接下来的说明。

AP(Access Point):无线接入点，这个概念特别广，在这里，用大白话说，你可以把 CC3200 当做一个无线路由器，这个路由器的特点不能插入网线，没有接入 Internet，只能等待其他设备的链接，并且智能接入一个设备。类似于点对点模式啦。

STA(Station)：任何一个接入无线 AP 的设备都可以称为一个站点。大白话说也就是平时接入路由器的设备，哈哈。

SSID(Service Set Identifier):SSID,每个无线 AP 都应该有一个标示用于用户识别，SSID 就是这个用于用户识别的的名字，也就是我们经常说到的 wifi 名。

BSSID:每一个网络设备都有其用于识别的物理地址，这个东西呢就叫 MAC 地址，这个东西一般情况下出厂会有一个默认值，可更改，也有其固定的命名格式，也是设备识别的标识符。这个 BSSID 呢是针对设备说的，对于 STA 的设备来说，拿到 AP 接入点的 MAC 地址就是这个 BSSID。

RSSI:这个理解起来更简单，就是通过 STA 扫描到 AP 站点的信号强度。

2. 工程导入

2.1 导入的工程

①ti_rtos_config:TI RTOS 配置工程库

②oslib：因为 TI 的工程能同时支持 TI rtos 和 Free RTOS,所以又在此基础上写了一个通用的 os 库，我个人感觉有些东西好用，有些东西不好用，到时候再说

③simplelink：网络连接库，想要使用 CC3200 的网络必须包含这个库，别无他法，至少我现在没有发现其他的方式。

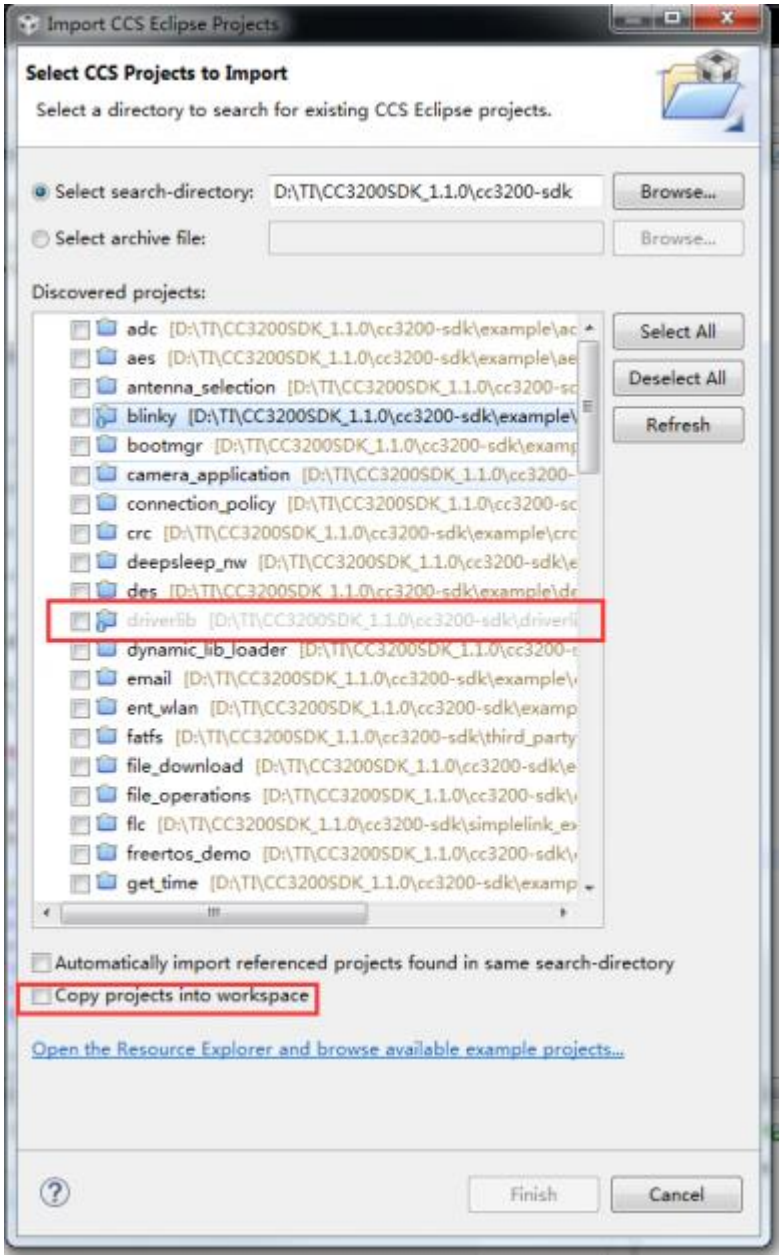
④tcp_socket：这个就是我们要调试的程序主题。

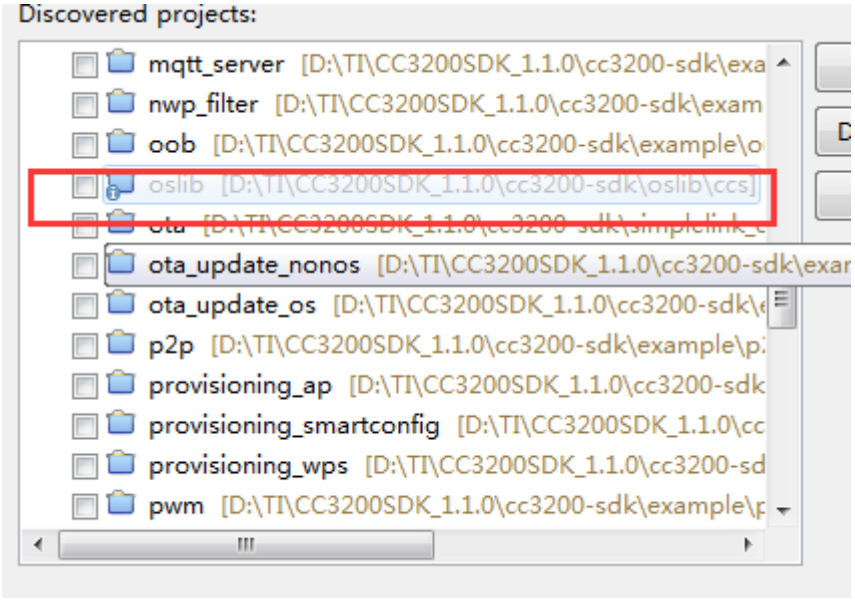
⑤Driverlib：这个是 GPIO 和串口用的库（在这个工程里）。

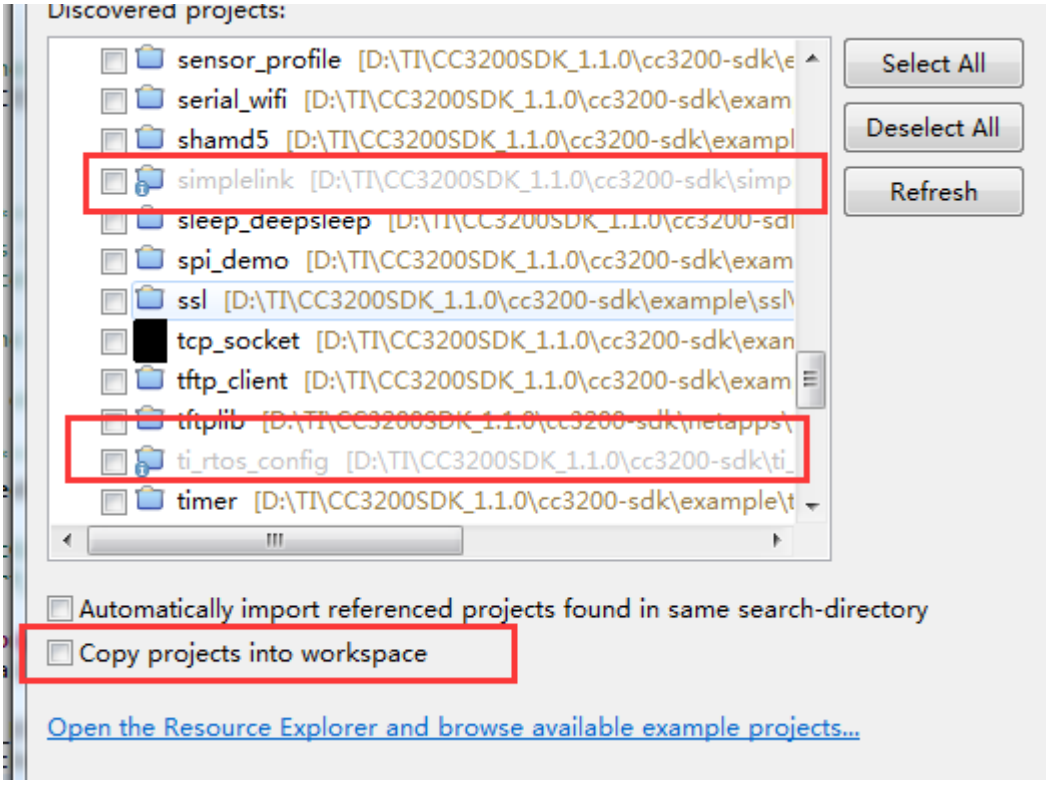
2.2 导入过程

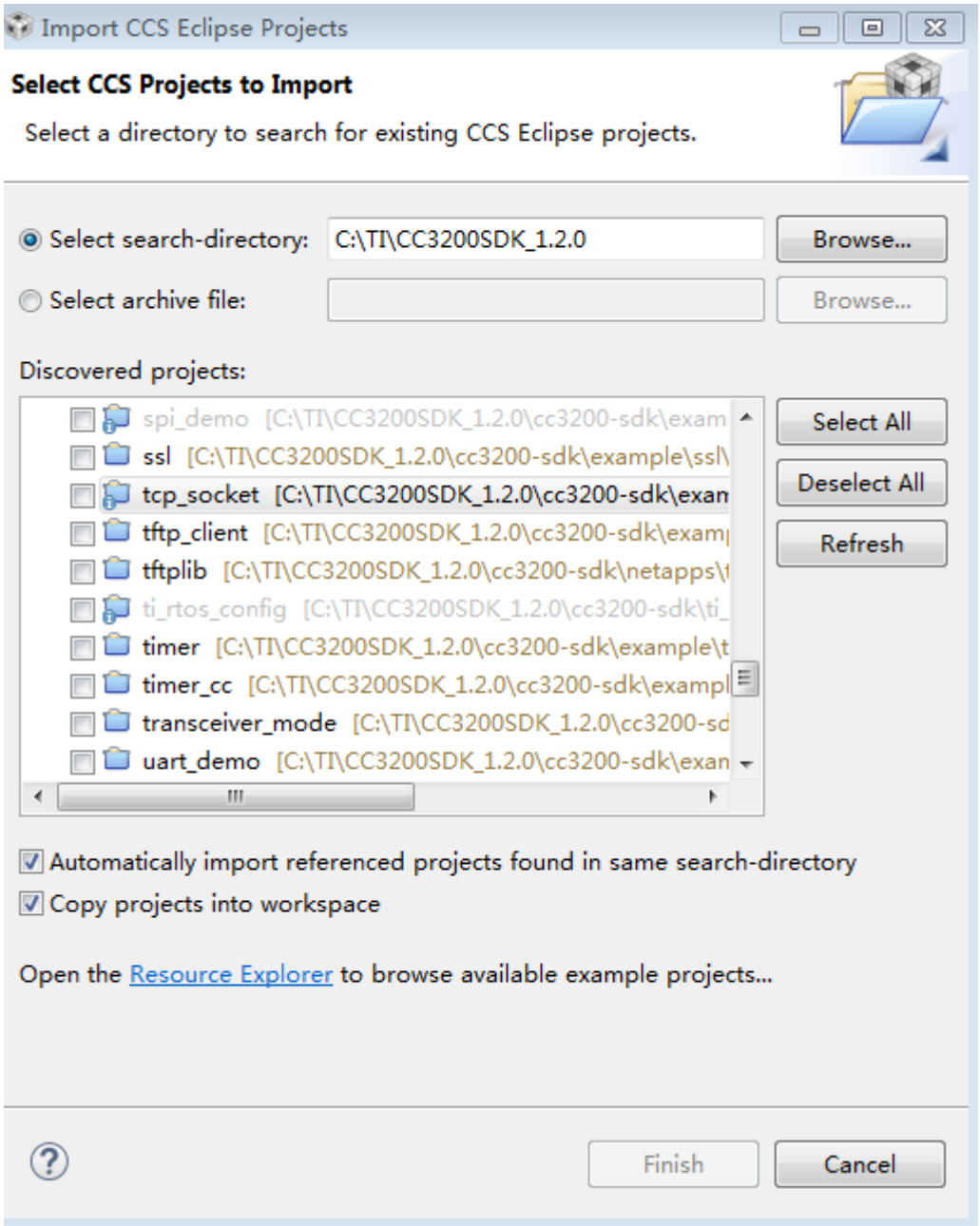
导入过程上一贴已经讲过了，所以就不多讲了，主要注意的是

①②③的导入不要 copy，④的导入 copy 就行。看图

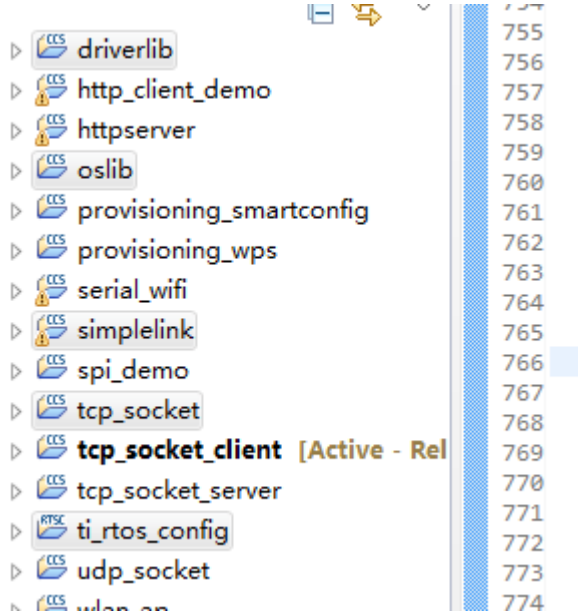

Import CCS Eclipse Projects
Select CCS Projects to Import
Select a directory to search for existing CCS Eclipse projects.
Select search-directory:
D:\TI\CC3200SDK_1.1.0\cc3200-sdk
Browse...
Select archive file:
Browse...
Discovered projects:
adc [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\ac
aes [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\ae
antenna_selection [D:\TI\CC3200SDK_1.1.0\cc3200-sc
blinky [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\
bootmgr [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\examp
camera_application [D:\TI\CC3200SDK_1.1.0\cc3200-
connection_policy [D:\TI\CC3200SDK_1.1.0\cc3200-sc
crc [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\crc
deepsleep_nw [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\e
des [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\de
driverlib [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\driverli
dynamic_lib_loader [D:\TI\CC3200SDK_1.1.0\cc3200-s
email [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\
ent_wlan [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\examp
fatfs [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\third_party
file_download [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\e
file_operations [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\
flc [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\simplelink_e
freertos_demo [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\
get_time [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\examp
Select All
Deselect All
Refresh
Automatically import referenced projects found in same search-directory
Copy projects into workspace
Open the Resource Explorer and browse available example projects...
Finish
Cancel


Discovered projects:
mqtt_server [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\exa
nwp_filter [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\exam
oob [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\o
oslib [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\oslib\ccs]
ota_update_nonos [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\exar
ota_update_os [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\
p2p [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\p
provisioning_ap [D:\TI\CC3200SDK_1.1.0\cc3200-sdk
provisioning_smartconfig [D:\TI\CC3200SDK_1.1.0\cc
provisioning_wps [D:\TI\CC3200SDK_1.1.0\cc3200-sd
pwm [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\p

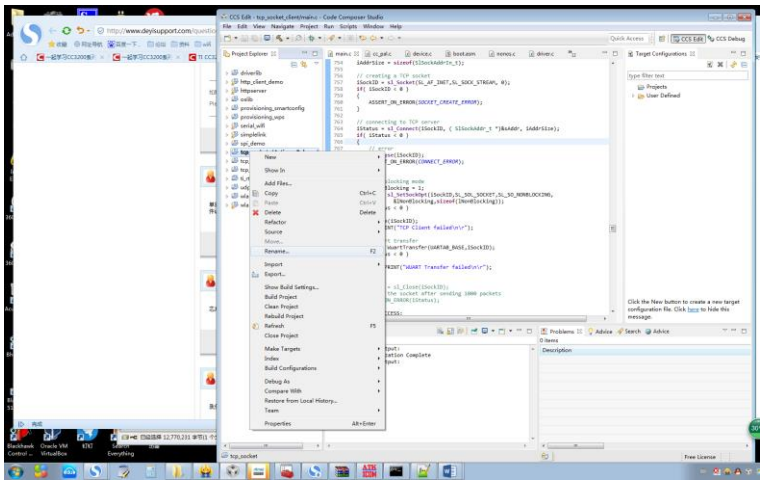

Discovered projects:
sensor_profile [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\e
serial_wifi [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\exam
shamd5 [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\exampl
simplelink [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\simp
spi_demo [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\exam
ssl [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\ssl
tcp_socket [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\exan
tftp_client [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\exam
ti_rtos_config [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\ti_
timer [D:\TI\CC3200SDK_1.1.0\cc3200-sdk\example\t
Select All
Deselect All
Refresh
Automatically import referenced projects found in same search-directory
Copy projects into workspace
Open the Resource Explorer and browse available example projects...


Import CCS Eclipse Projects
Select CCS Projects to Import
Select a directory to search for existing CCS Eclipse projects.
Select search-directory:
C:\TI\CC3200SDK_1.2.0
Browse...
Select archive file:
Browse...
Discovered projects:
spi_demo [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\exam
ssl [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\example\ssl\
tcp_socket [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\exan
tftp_client [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\exam
tftplib [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\netapps\
ti_rtos_config [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\ti_
timer [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\example\t
timer_cc [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\exampl
transceiver_mode [C:\TI\CC3200SDK_1.2.0\cc3200-sd
uart_demo [C:\TI\CC3200SDK_1.2.0\cc3200-sdk\exan
Select All
Deselect All
Refresh
Automatically import referenced projects found in same search-directory
Copy projects into workspace
Open the Resource Explorer to browse available example projects...
Finish
Cancel

导入完成，如下图所示



将 tcp_socket,更名为 tcp_socket_client

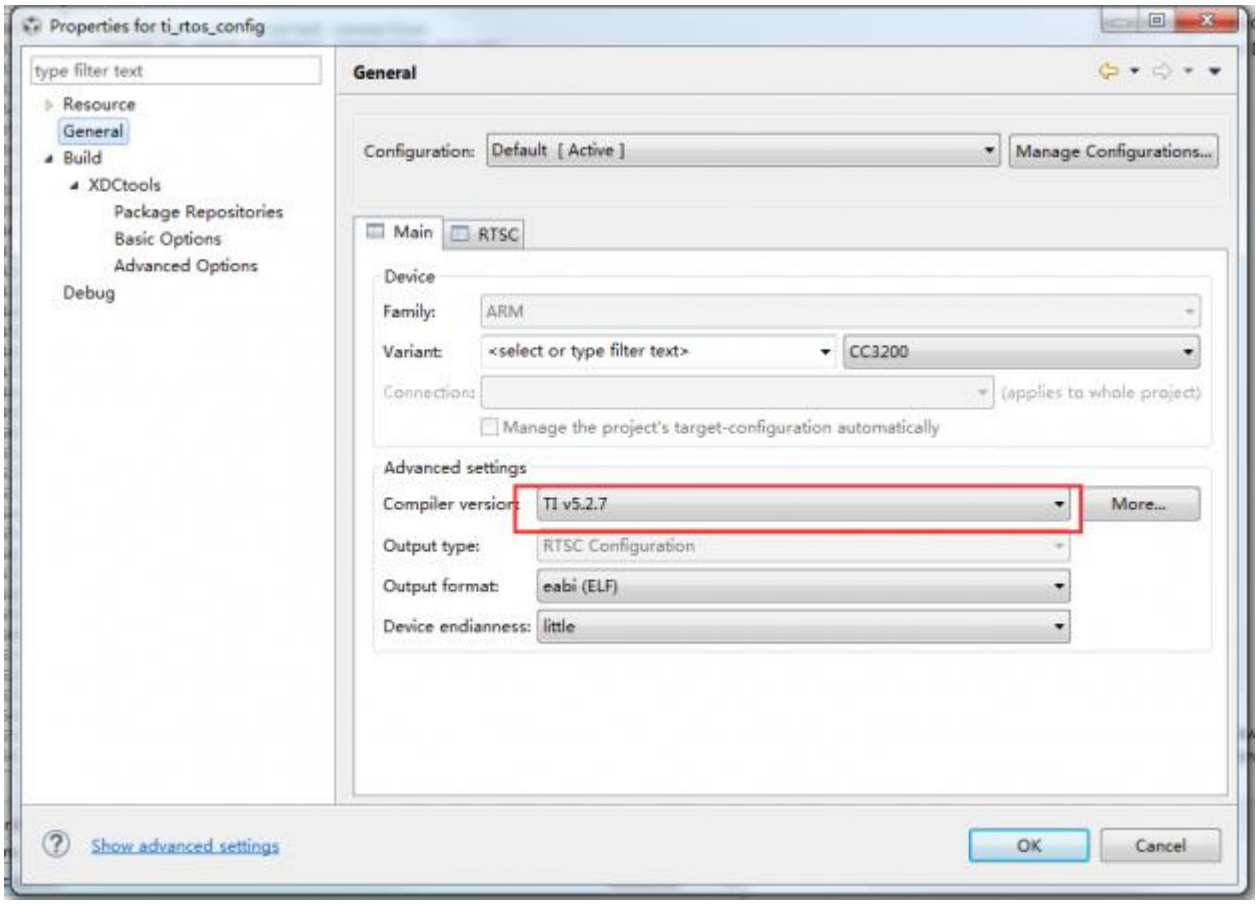


2.3 编译工程

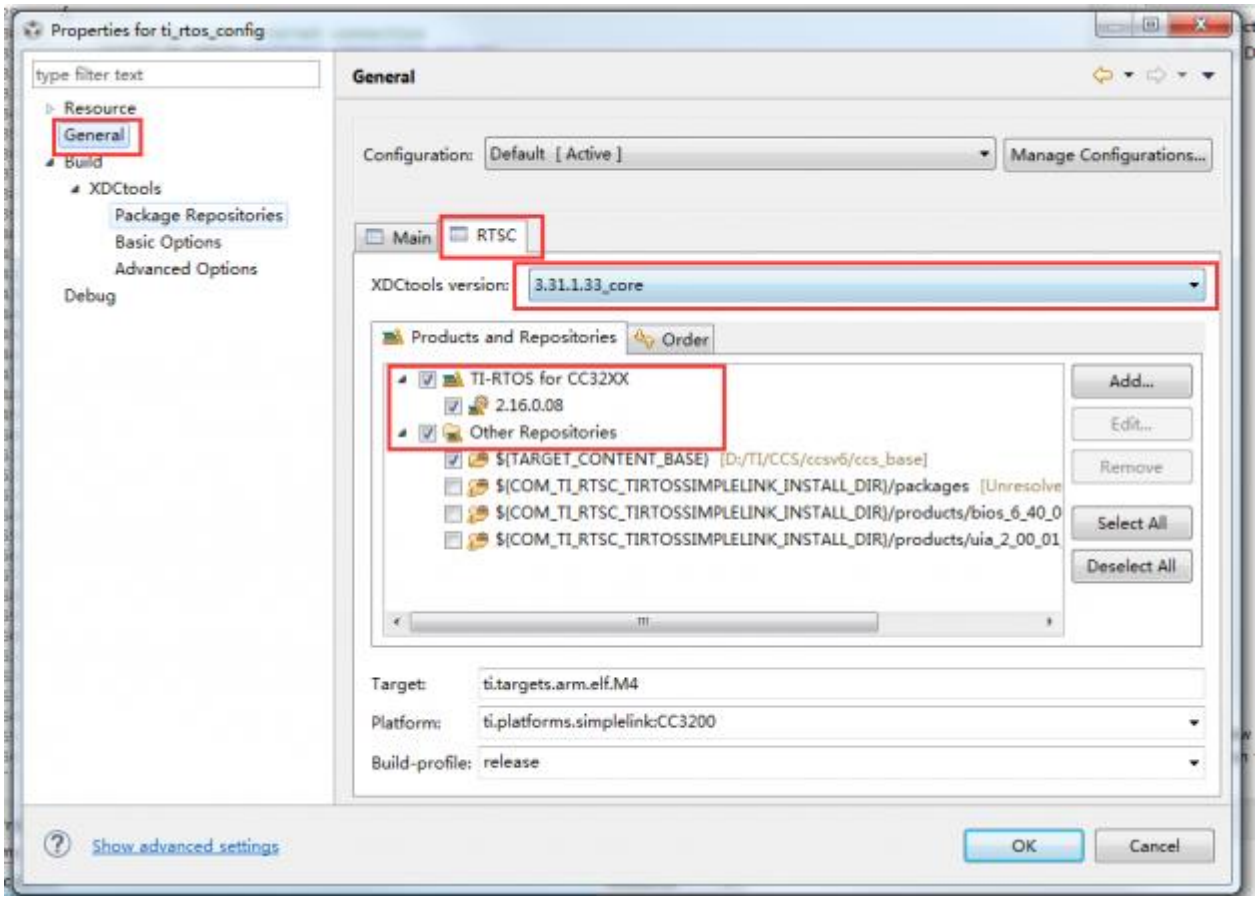
编译的顺序一定要严格按照下边的顺序，因为每个库之间是有相互依赖关系的,不按照套路编译，就会出错

1、编译 ti_rtos_config

右击 ti_rtos_config 选择 Propertise


Properties for ti_rtos_config
type filter text
Resource
General
Build
XDCtools
Package Repositories
Basic Options
Advanced Options
Debug
General
Configuration: Default [Active]
Manage Configurations...
Main
RTSC
Device
Family: ARM
Variant: <select or type filter text>
CC3200
Connection:
(applies to whole project)
Manage the project's target-configuration automatically
Advanced settings
Compiler version: TI v5.2.7
More...
Output type: RTSC Configuration
Output format: eabi (ELF)
Device endianness: little
Show advanced settings
OK
Cancel

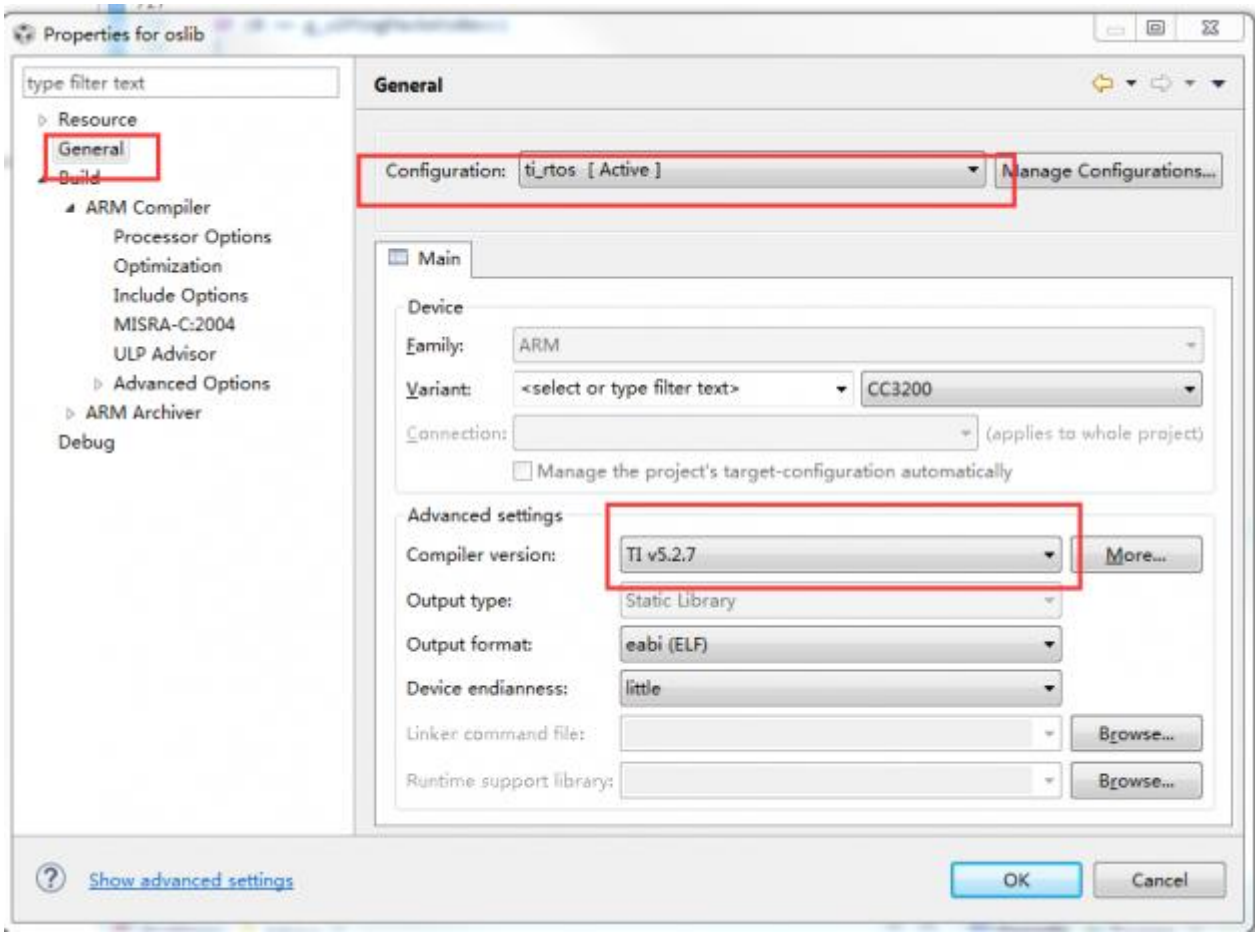
选择最新的编译器

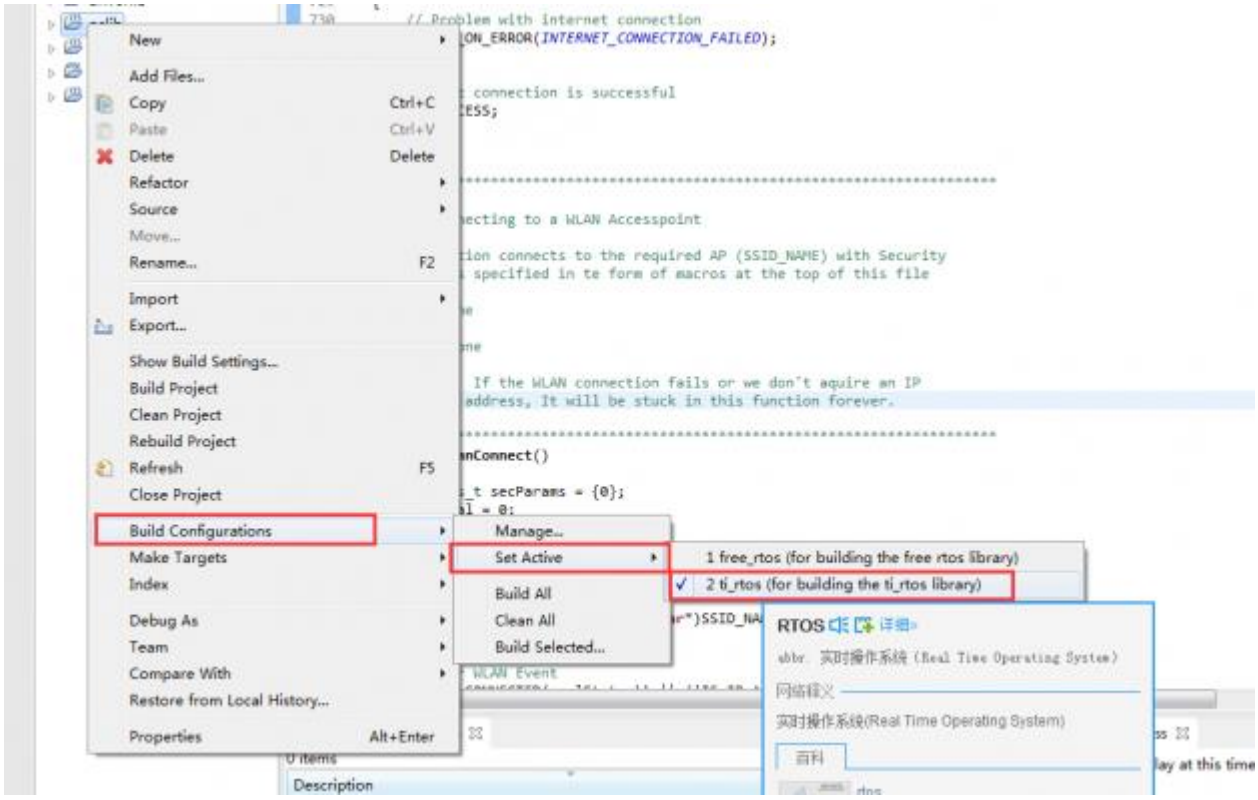


按照图中选择，然后编译，没有报错和警告证明编译完成

2、编译 oslib

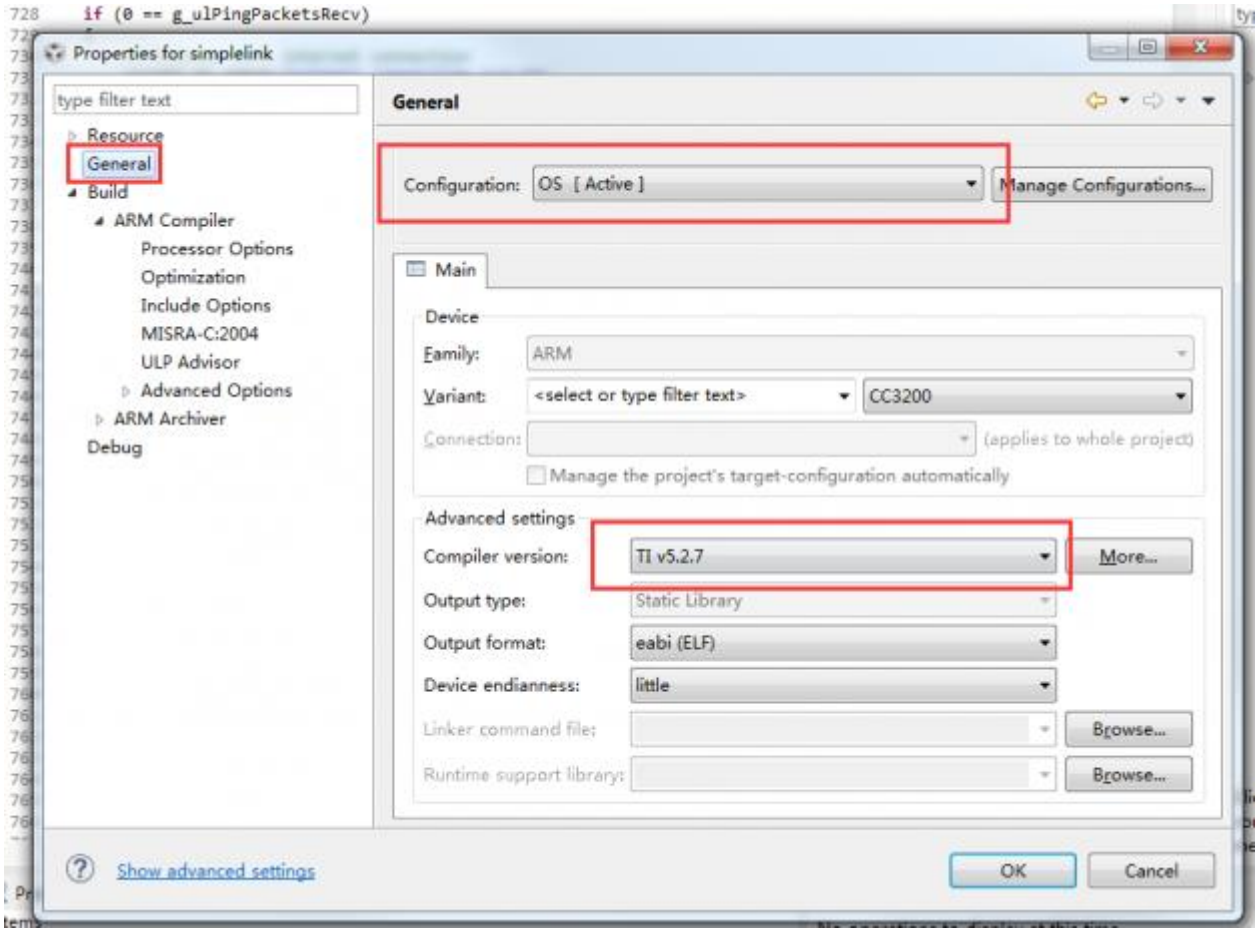
同样右击点击 Propertise，选择最近的编译器，如果不是最新的应该会给警告，但是没有问题。

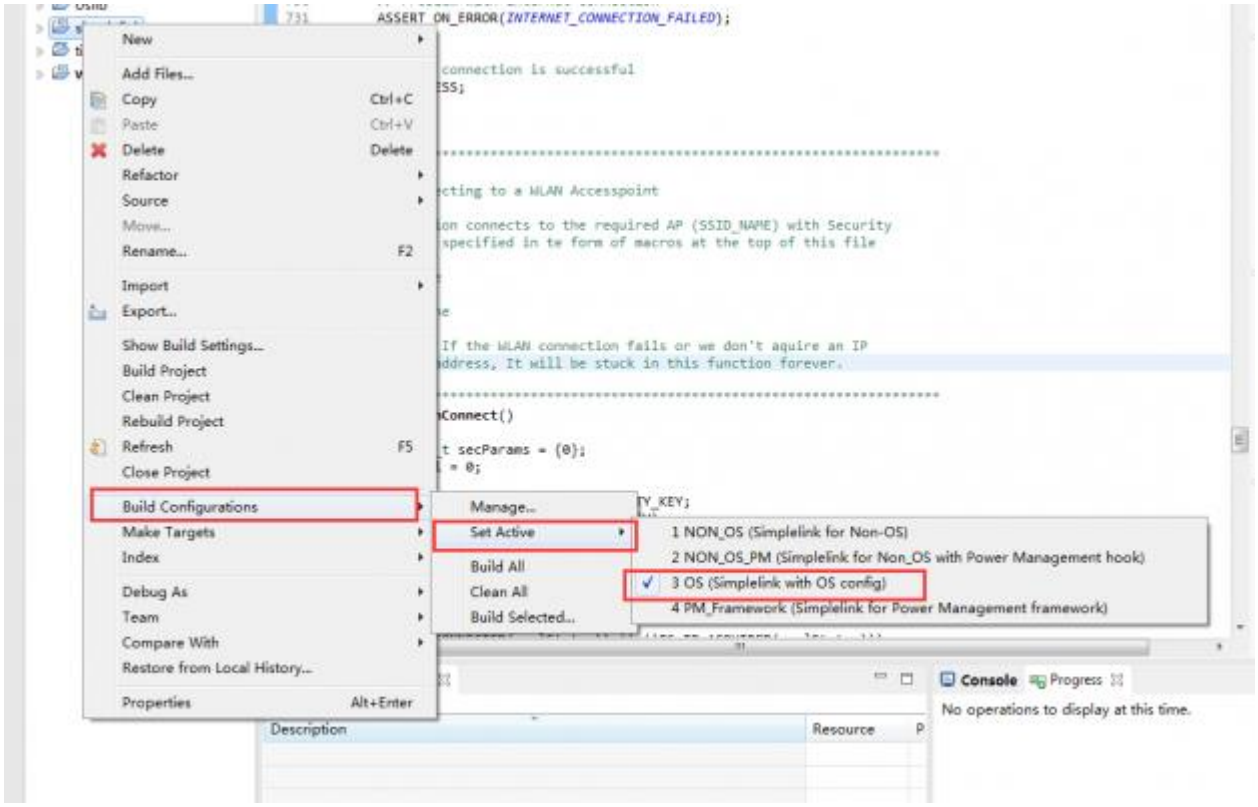




然后编译，如果没有警告和错误，证明编译完成

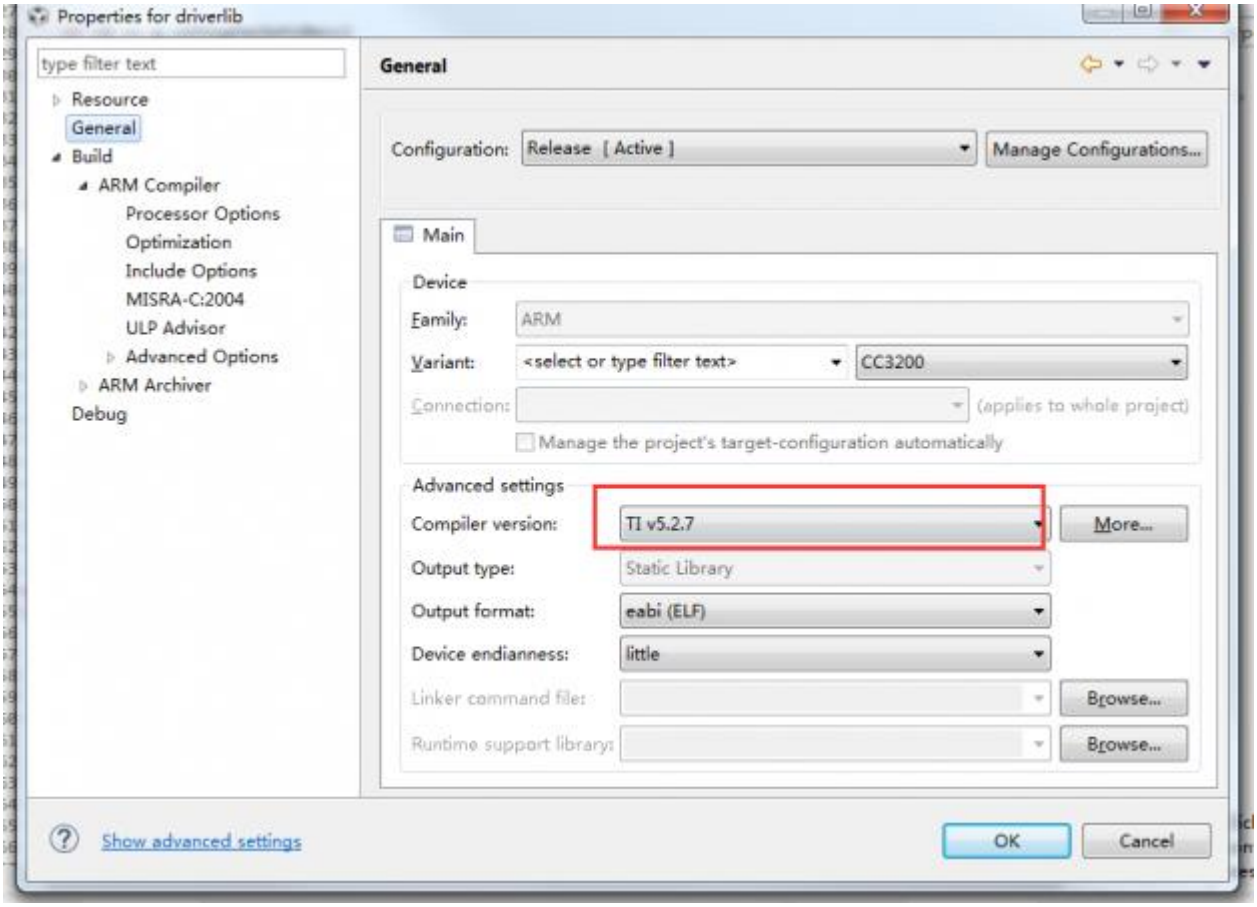
3、编译 simplelink





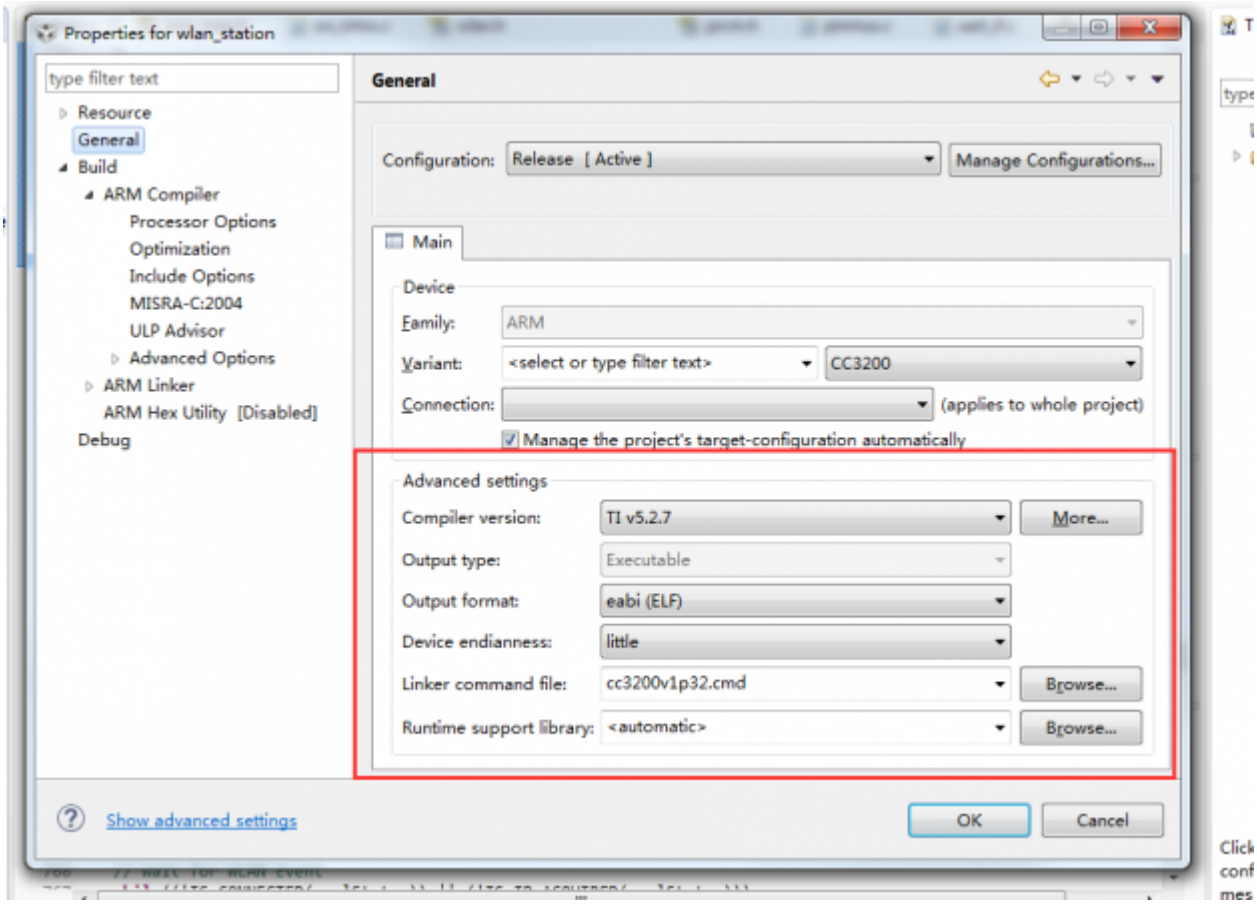
同样的编译没有错误，就算编译完成

4、编译 driverlib



同样设置完成后编译，如果没有错误就算编译完成

5、编译 tcp_socket_client

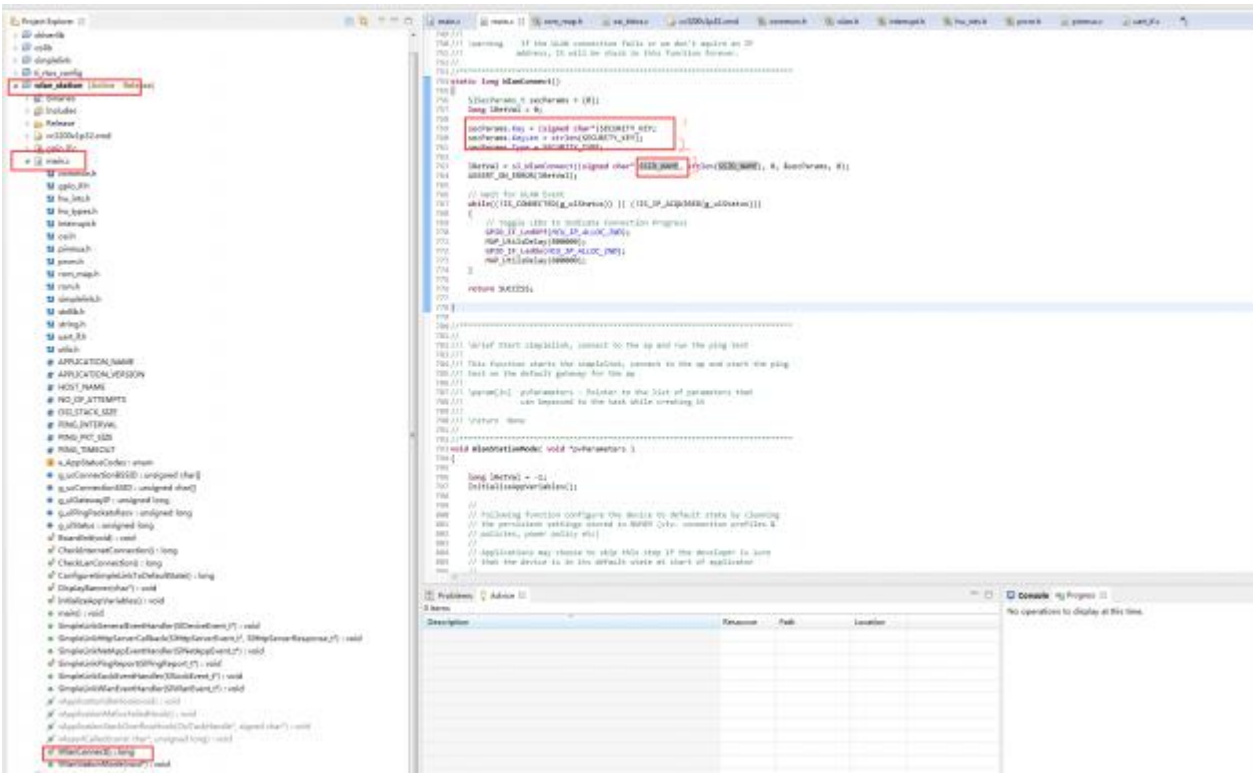


如果没有出现任何错误，就证明编译完成，

3、修改参数

下边我们就让 cc3200 的设备连接到相应的网络上

修改参数



SECURITY_KEY：是你要连接的无线 AP 的名称　路由的密码

SECURITY_TYPE：要连接的无线 AP 的加密方式

SSID_NAME：要连接的 AP 的名称　路由的名字

3.1 更改 BsdTcpClient 函数

BsdTcpClient 函数更改为如下：

```
SlSockAddrIn_t  sAddr;
    int             iAddrSize;
    int             iSockID;
    int             iStatus;

    //filling the TCP server socket address
    sAddr.sin_family = SL_AF_INET;
    sAddr.sin_port = sl_Htons((unsigned short)usPort);
    sAddr.sin_addr.s_addr = sl_Htonl((unsigned int)g_ulDestinationIp);

    iAddrSize = sizeof(SlSockAddrIn_t);

    // creating a TCP socket
    iSockID = sl_Socket(SL_AF_INET,SL_SOCK_STREAM, 0);
    if( iSockID < 0 )
    {
```

```
        ASSERT_ON_ERROR(SOCKET_CREATE_ERROR);
    }

    // connecting to TCP server
    iStatus = sl_Connect(iSockID, ( SlSockAddr_t *)&sAddr, iAddrSize);
    if( iStatus < 0 )
    {
        // error
        sl_Close(iSockID);
        ASSERT_ON_ERROR(CONNECT_ERROR);
    }

    //set nonblocking mode
    long lNonBlocking = 1;
    iStatus = sl_SetSockOpt(iSockID,SL_SOL_SOCKET,SL_SO_NONBLOCKING,
            &lNonBlocking,sizeof(lNonBlocking));
    if( iStatus < 0 )
    {
      sl_Close(iSockID);
      UART_PRINT("TCP Client failed\n\r");
    }
    //wifi uart transfer
    iStatus = WuartTransfer(UARTA0_BASE,iSockID);
    if( iStatus < 0 )
    {
        UART_PRINT("WUART Transfer failed\n\r");
    }
    return SUCCESS;
```

3.2 增加 WuartTransfer 函数

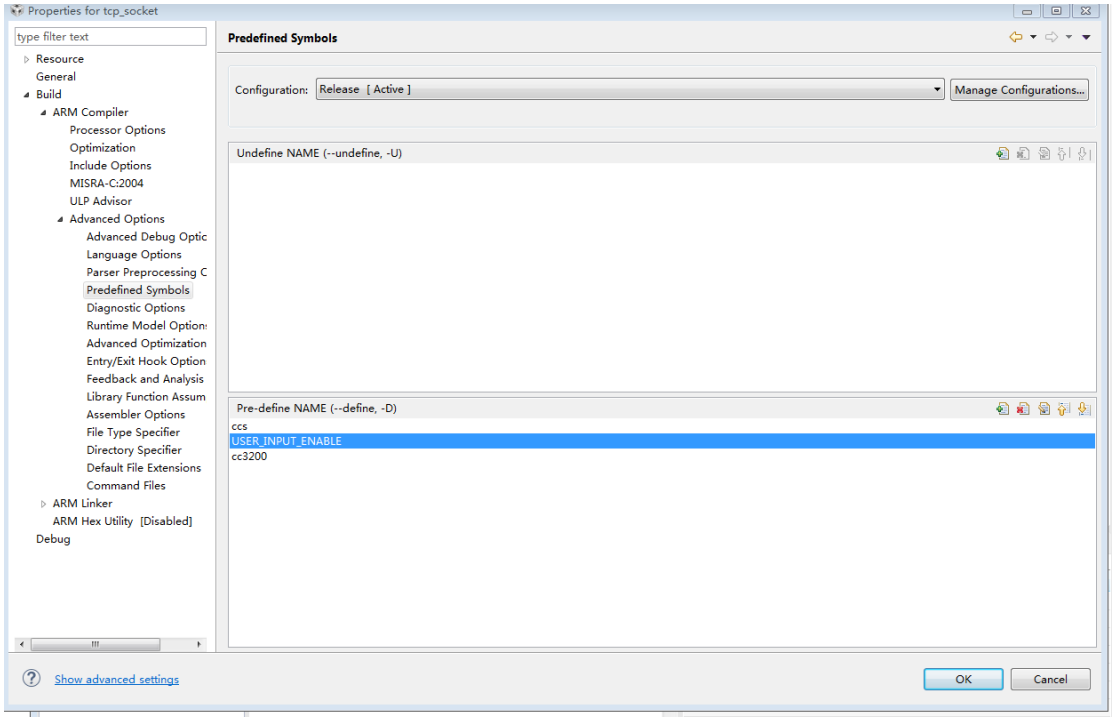
```
int WuartTransfer(unsigned long ulBase, int iSockID)
{
    char    cTxBuf[100],i;
    char    cRxBuf[100];
    char    cGetChar;
    int     iStatus,iStatusRecv;
    int     iCounter = 0;

    //UART_PRINT("\r\n\send:");
    while(1)
    {
        //uart get char
        cGetChar = MAP_UARTCharGetNonBlocking(ulBase);
```

```
if(cGetChar != 0xff)
{
    //uart display back
    MAP_UARTCharPut(ulBase,cGetChar);
    //save char of recv
    cTxBuf[iCounter++] = cGetChar;
    //enter 0x0d  or esc  0x1b
    if((cGetChar == 0x0d)||(cGetChar == 0x1b))
    {
        //display back and save  0x0a
        MAP_UARTCharPut(ulBase,0x0a);
        cTxBuf[iCounter++] = 0x0a;
        //tcp send
        iStatus = sl_Send(iSockID, cTxBuf, iCounter, 0);
        if(iStatus <= 0)
        {
            ASSERT_ON_ERROR(sl_Close(iSockID));
            UART_PRINT("发送数据失败\n\r:");
            break;
        }
        if(cGetChar == 0x0d)
        {
            UART_PRINT("send1:");
            iCounter = 0;
        }
        else
            break;
    }
}
//recv tcp
iStatus = sl_Recv(iSockID,cRxBuf,100,0);
if(iStatus > 0)
{
    iStatusRecv = sl_Send(iSockID, cRxBuf, iStatus, 0);
    if(iStatusRecv <= 0)
    {
        ASSERT_ON_ERROR(sl_Close(iSockID));
        UART_PRINT("发送数据失败\n\r:");
        break;
    }
    for(i = 0; i < iStatus; i++ )
    {
        UART_PRINT("send2: %2c \n\r", cRxBuf[i]);
    }
```

```
        }
    }
    return(iStatus);
}
```

3.3 更改 Properties 中的 predefined Symbols 项

删除 USER_INPUT_ENABLE 宏定义

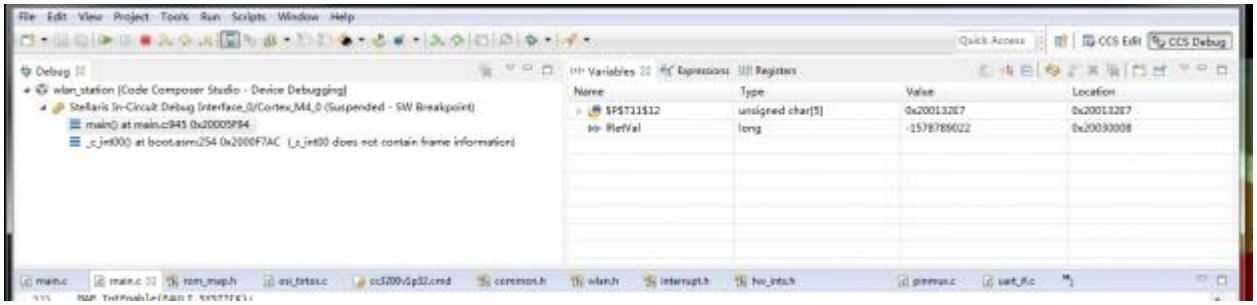


3.4 删除 main 函数中 BsdTcpServer 相关

```
lRetVal = BsdTcpServer(PORT_NUM);
    if(lRetVal < 0)
    {
        UART_PRINT("TCP Server failed\n\r");
        LOOP_FOREVER();
    }
```

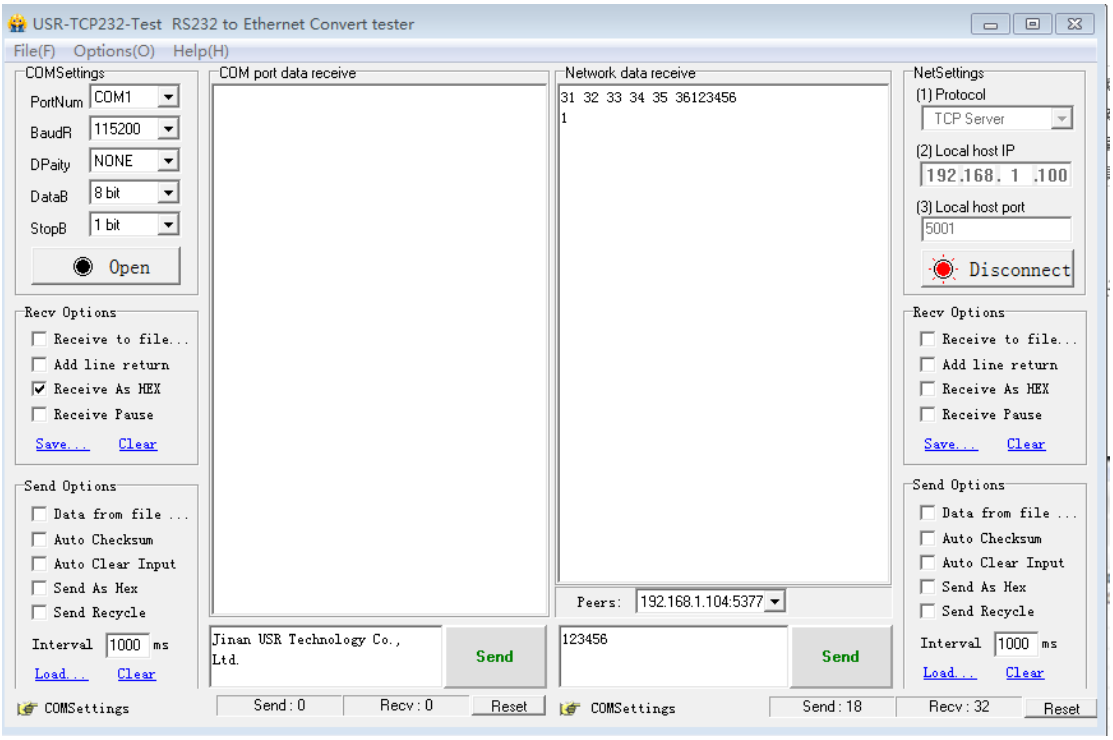
4 编译下载

然后重新编译一下，没有出现错误就把程序下载到板子里



打开串口助手

打开 TCP 测试软件



点击串口调试工具是发送，和 TCP 测试工具“send”，从以上两图可以看到输出。

注意：

1.在线调试时，需要注意 IRetVal = sl_Start(0, 0, 0);

全速运行会卡在 sl_Start 函数中的 spi_Read_CPU 函数内的

```
    while(ulCnt--)
    {

        while(!( HWREG(ulStatusReg)& MCSPI_CH0STAT_TXS ));
        HWREG(ulTxReg) = 0xFFFFFFFF;
        UtilsDelay(300*80/3);
        while(!( HWREG(ulStatusReg)& MCSPI_CH0STAT_RXS ));
        *ulDataIn = HWREG(ulRxReg);
        ulDataIn++;
        UtilsDelay(300*80/3);
    }
```

不论是把这段函数注销还是删除，都会继续执行这段代码，不解。重启电脑，板子从上电，CCS 重启都是这样的结果。

但有时能通过，这个问题困扰我 2 天；据说这是 TI 的 API 的 BUG，目前没有解决，下载到板子运行，没有问题。

2.运行前，需要先打开 TCP 测试工具。