Windows 下 STM32 单片机的 eclipse 编译环境搭建

英创公司开发的 ETA321 单片机模块，是基于 STM32F103RC 单片机设计的实时任务处理单元模块。

ETA321 模块作为客户端，使用 USB 与英创公司的 ARM 工控主板进行数据传输、通讯，根据工控主板应用程序发出来的指令，执行相应的实时任务处理，如实现：电机控制、数据采集、高速 D/A 输出、状态采集保护等。另外，用户可以基于英创公司构建好的 USB 通讯结构，利用 KEIL、IAR、eclipse 等集成编译环境，在 ETA321 模块上编写自己特殊的应用程序。英创公司推荐使用 eclipse 编译环境。

由于 eclipse 软件编译工具是开源、免费的工具，在 windows 环境下编译 linux 应用软件的用户也比较多，因此在 windows 环境中，搭建 eclipse 的 STM32 编译环境，就变得非常有意义：可以很好地与 linux 编译环境进行整合；不用担心开发软件的版权问题；eclipse 编译环境也能支持各种调试工具。

因此，使用 eclipse 编译环境完全替换 KEIL、IAR 等需要授权才能使用工具，是非常好的选择。但是搭建基于 eclipse 的 STM32 的开发环境，需要好几个软件与插件，且相应的版本需要能兼容，才能正确搭建好开发环境。

为了协助客户快速搭建 eclipse 编译环境，英创公司编写了这篇文章，详细说明了整个 eclipse 环境搭建、工程建立、目标代码烧写、以及使用 JLink 进行软件调试的方法。

所需要的软件或插件如下：

- JAVA
- eclipse C/C++
- gcc-arm-none-eabi，交叉编译工具链
- CDT，对 cortex-M 的支持的编译工具以及对 J-Link 调试工具的支持
- J-LINK 驱动

开发/烧写 STM32 所需要的软件/工具：

- STM32 Flashloader 串口烧写工具

为了方便客户更加简单、方便地搭建 ETA321 的开发环境，英创公司下载了所需的工具软件：JAVA、eclipse C/C++、gcc-arm-none-eabi、STM32-Flashloader 等，可以直接利用。

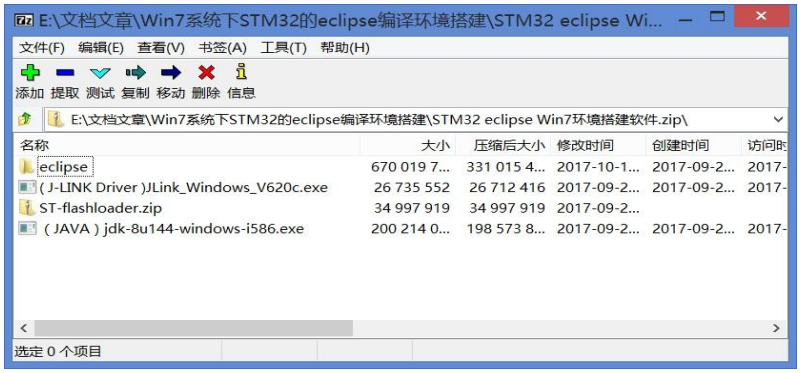

图 1、eclipse 编译环境软件工具包

下面则详细地描述 eclipse 编译环境搭建方法。

一、安装 eclipse 编译环境

1、安装 JAVA

打开工具包中的（JAVA）jdk-8u144-windows-i586.exe 进行默认安装即可。

图 2、启动并安装 JAVA

图 3、JAVA 安装中...

图 4、JAVA 安装完成

现阶段，不建议安装 JAVA 9.0 版本。因为要实现 JAVA 9.0 对 eclipse 的支持，需要额外的补丁包，所以相对麻烦一些。

2、安装 eclipse

复制 eclipse 文件夹到开发用的计算机中（任意位置均可），如：C:\program file(x86)目录（ 如果系统是 32 bit，则 C:\program file ）。

图 5、将 eclipse 复制到 Program Files(x86)目录下

3、设置交叉编译工具链的环境变量

从 eclipse 目录下，找到交叉编译工具链 gcc-arm-none-eabi 的路径，并复制下来。

图 6、gcc-arm-none-eabi 工具链目录路径

打开“系统属性”->“高级”配置页面，再点击“环境变量”，进入环境变量设置页面。

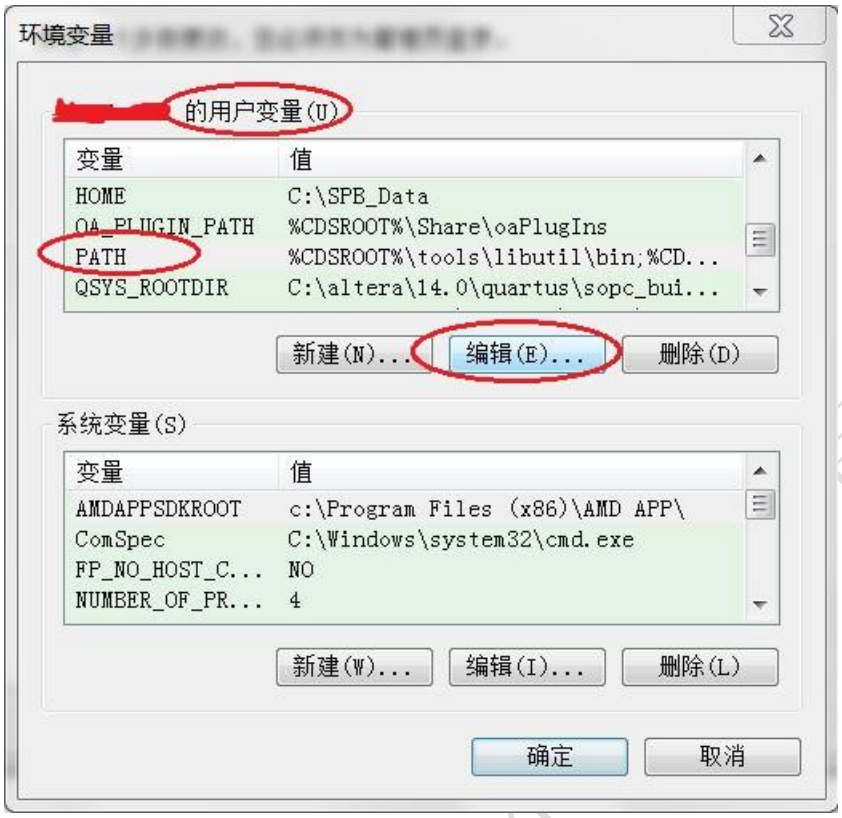

图 7、环境变量设置页面

在用户环境变量中找到 PATH 项，点击编辑，将工具链的路径添加到 PATH 参数中，点击确认退出。

图 8、在用户环境变量的 PATH 变量名中，添加工具链的路径

环境变量添加完成后，可以从系统的 CMD 命令提示符窗口中，输入 path 指令，检查环境变量是否添加成功。

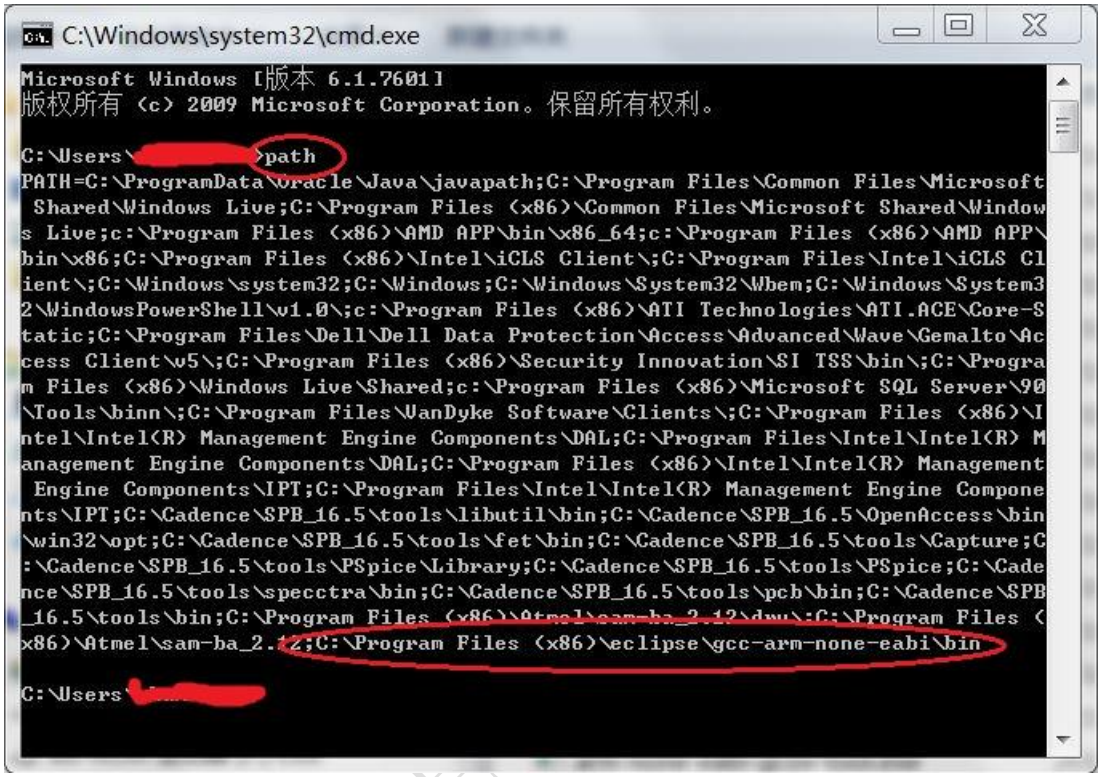

图 9、检查工具连的环境变量

4、安装 CDT（C/C++ Development Tooling）

进入 eclipse 目录，双击 eclipse.exe 启动 eclipse 软件，这时设置 eclipse 的工作区路径，可以系统中的任意路径均可。设置好以后，点击“launch”。

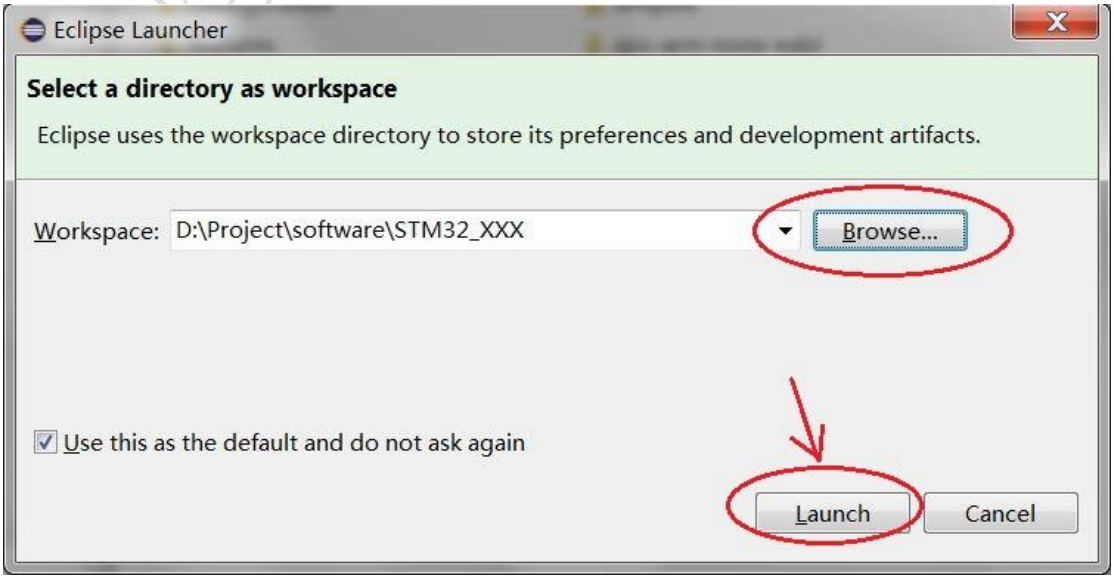

图 10、设置 eclipse 工作区

软件启动完成后，在 eclipse 的“help”菜单中，选择“install New software”

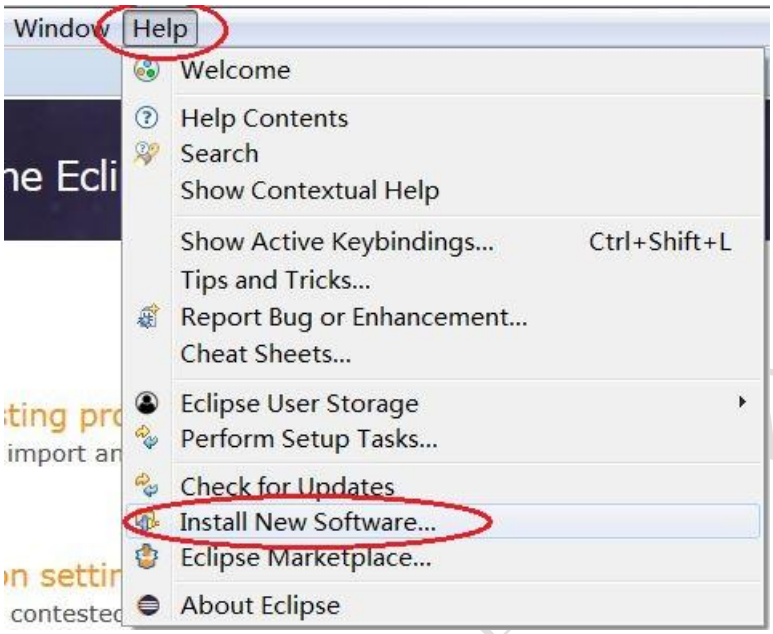

图 11、选中 help->Install New Software

然后在“Work With:”中输入链接“http://gnu-mcu-eclipse.netlify.com/v4-neon-updates”并回车，将会列出该 CDT 所包含的所有工具列表。在列出来的选项中，全部选择，点击“next”。

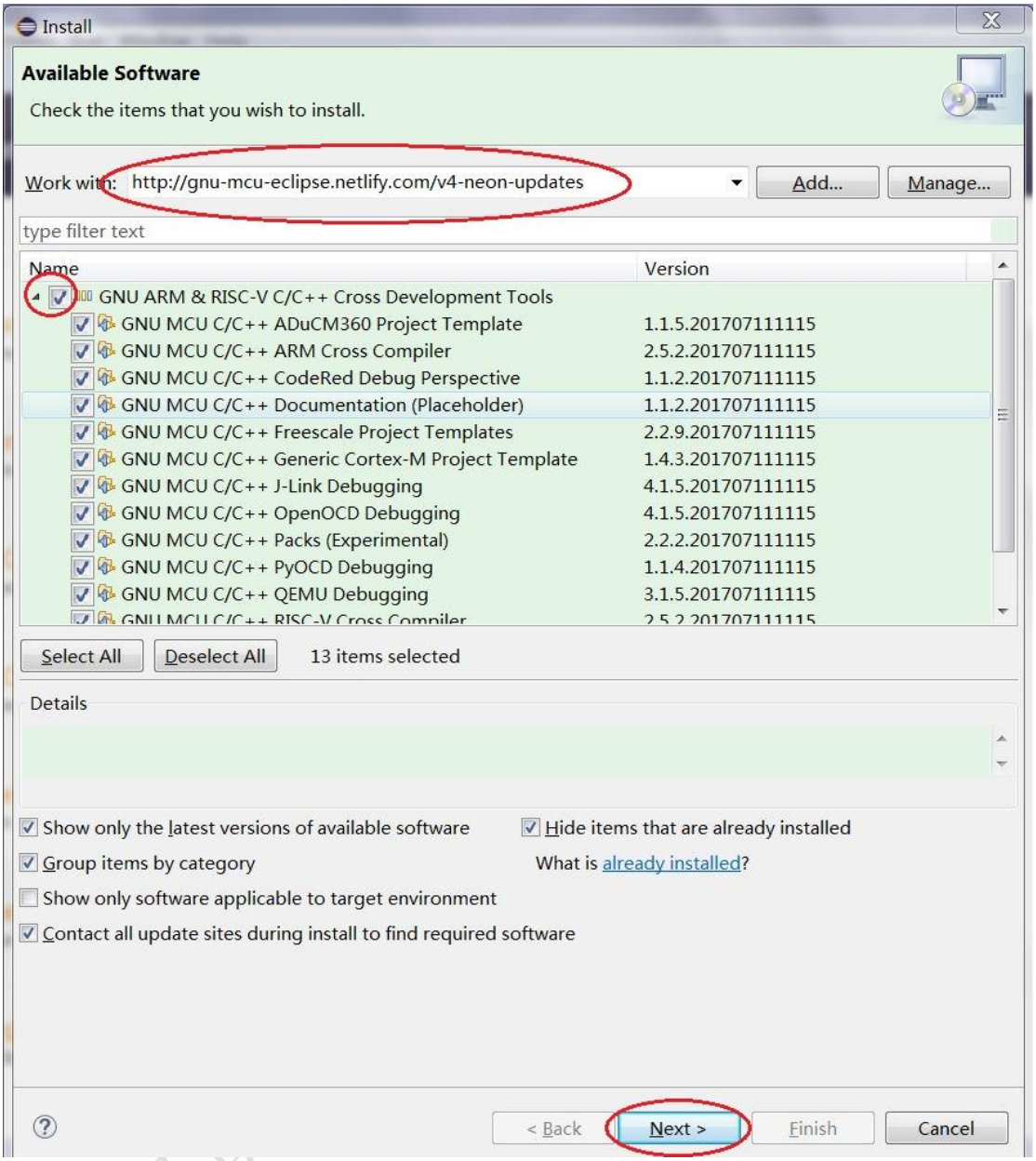

图 12、选中所有列表

最后接受协议，点击“finish”开始安装 CDT。

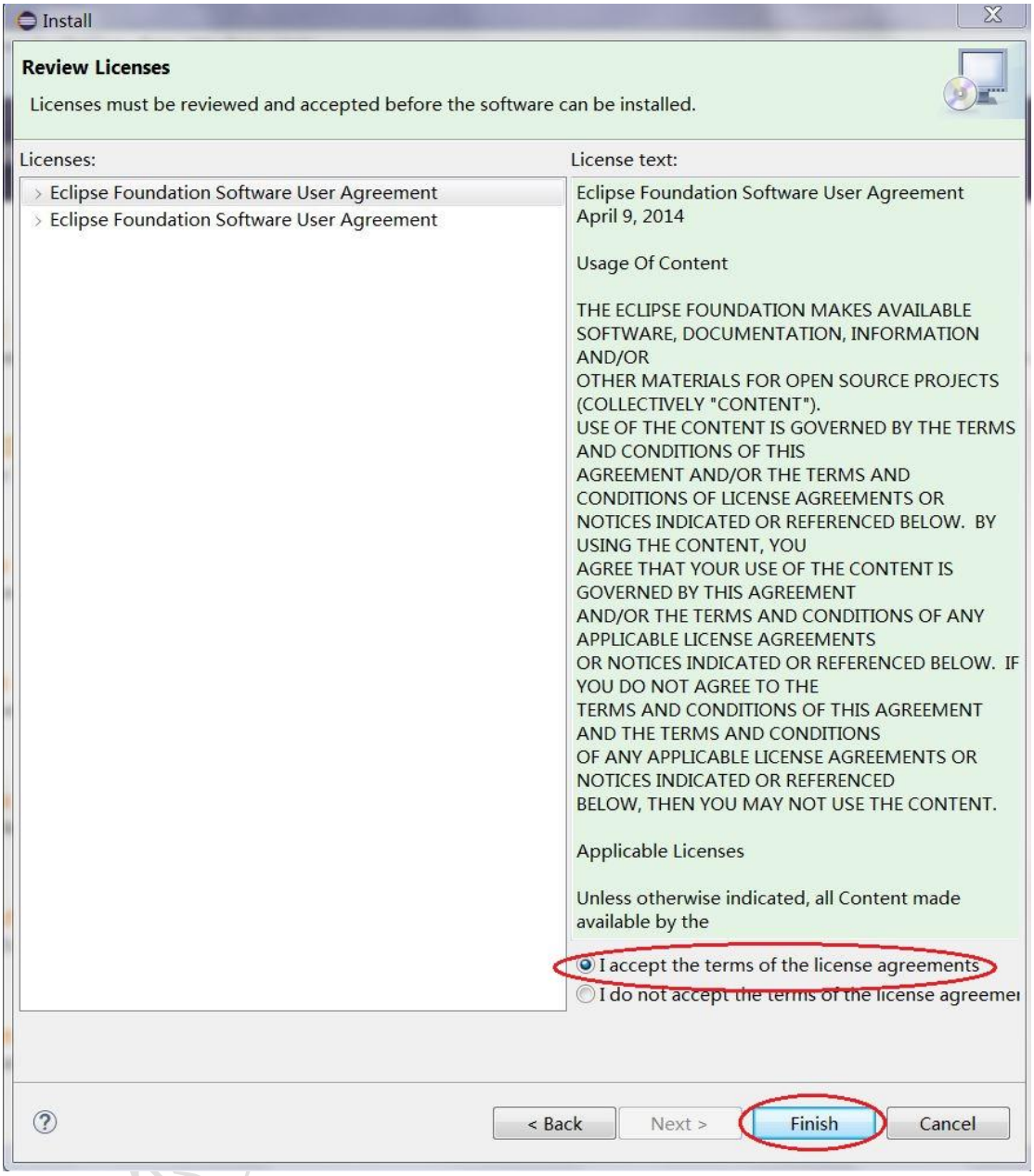

图 13、接受 license 后开始安装

在 eclipse 的右下角可以看到安装进度，双击小的进度条，则可以打开安装信息。在安装过程中，如果有安全警告，点击“Install anyway”继续完成安装。

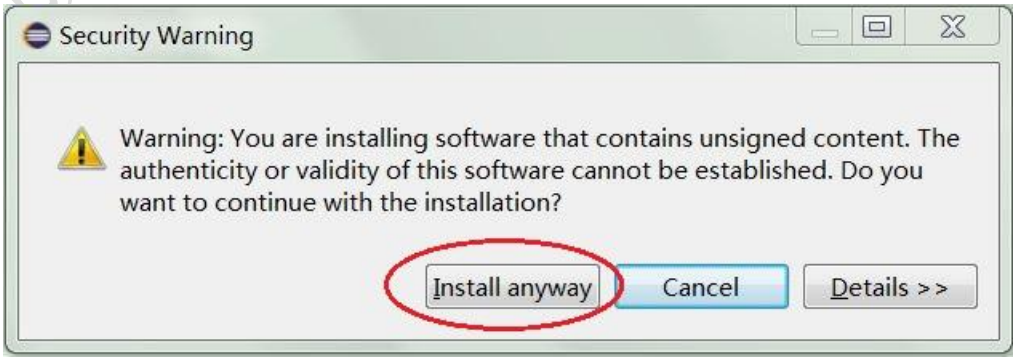

图 14、安装过程中的安全警告

当 CDT 安装完后，会提示软件重启，这时点击“Restart Now”，重新启动 eclipse。

图 15、CDT 安装完成，需要重新启动软件

到此，在 Windows7 系统下，STM32 的 eclipse 编译环境已经搭建完成，接下来就可以建立第一个应用程序工程了。

二、建立 eclipse 应用工程程序

eclipse 编译环境安装完成以后，就可以开始编写应用程序了，我们以 ETA321 模块上的 LED 灯 D1 闪烁为例子，举例说明建立简单的应用工程。

1、启动 eclipse，从 file->new 中选择 C/C++ Project。

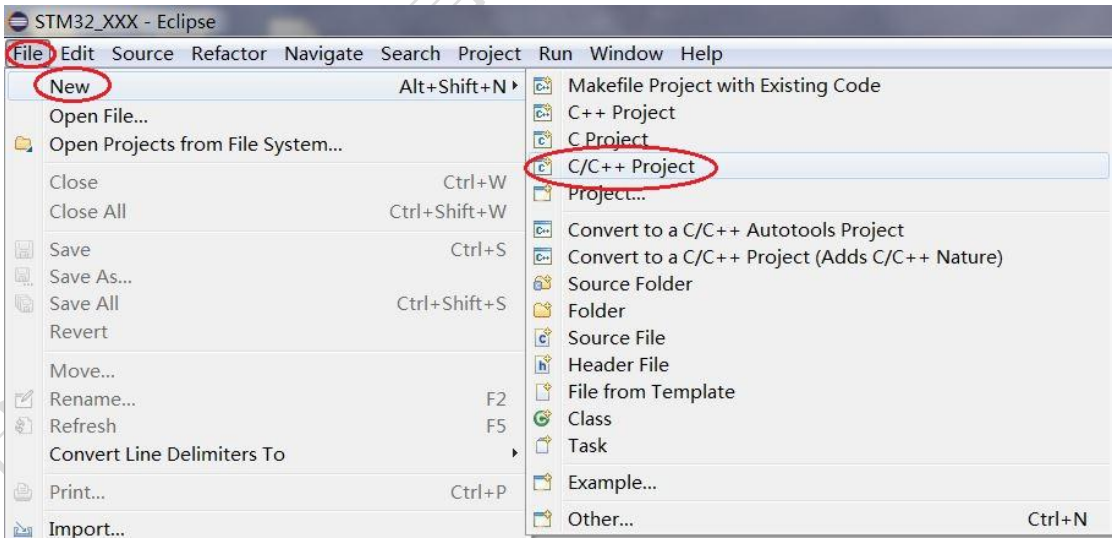

图 16、建立工程

2、选择 C++ Managed Build，点击“next”。

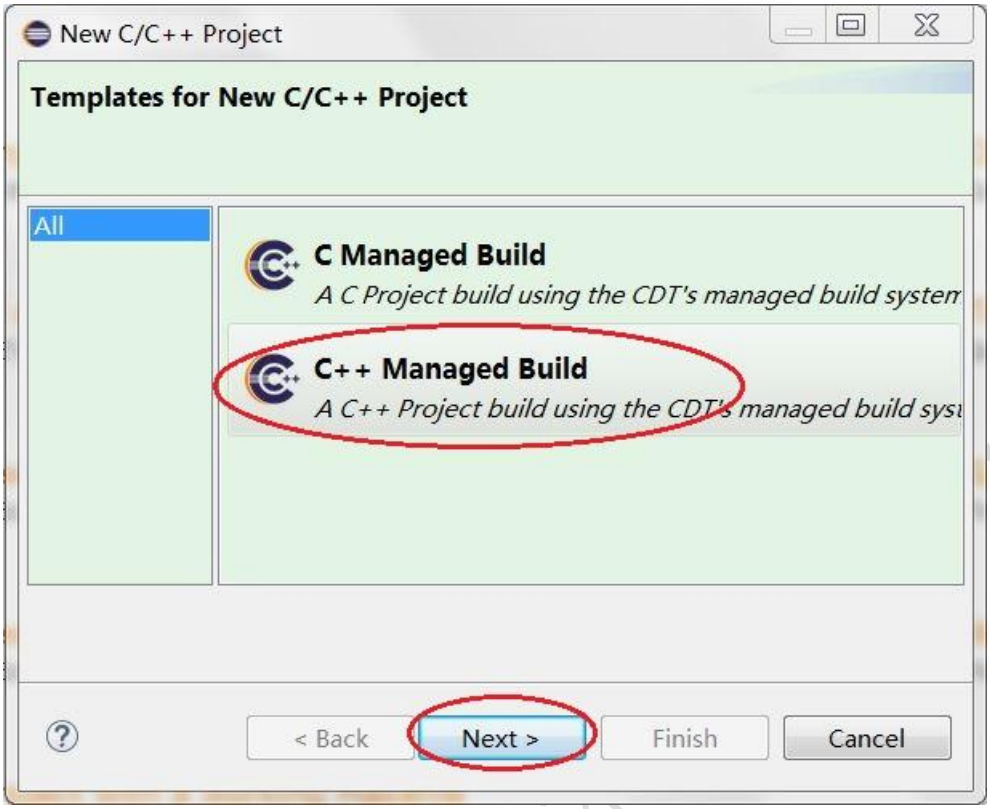

图 17、新工程类型选择

3、输入工程名，以及选择 Project type 为“STM32F10x C/C++ project”，右侧 Toolchains 选择为“ARM Cross GCC”，点击“next”。

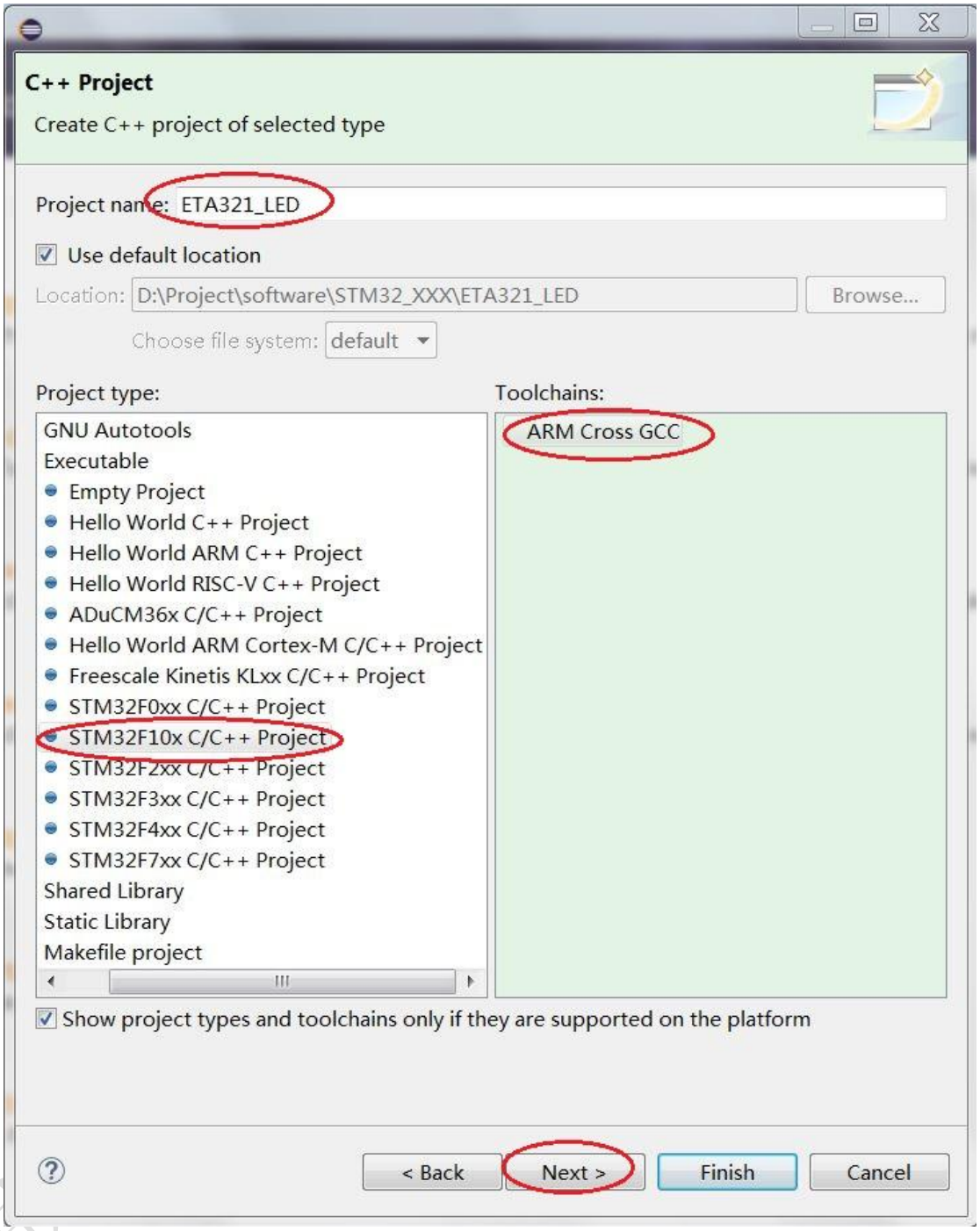

图 18、工程编译参数选择

4、设置芯片属性，ETA321 采用的是高密度器件，Flash 大小为 256KB，RAM 大小为 48KB，根据该参数进行设置，点击“next”。

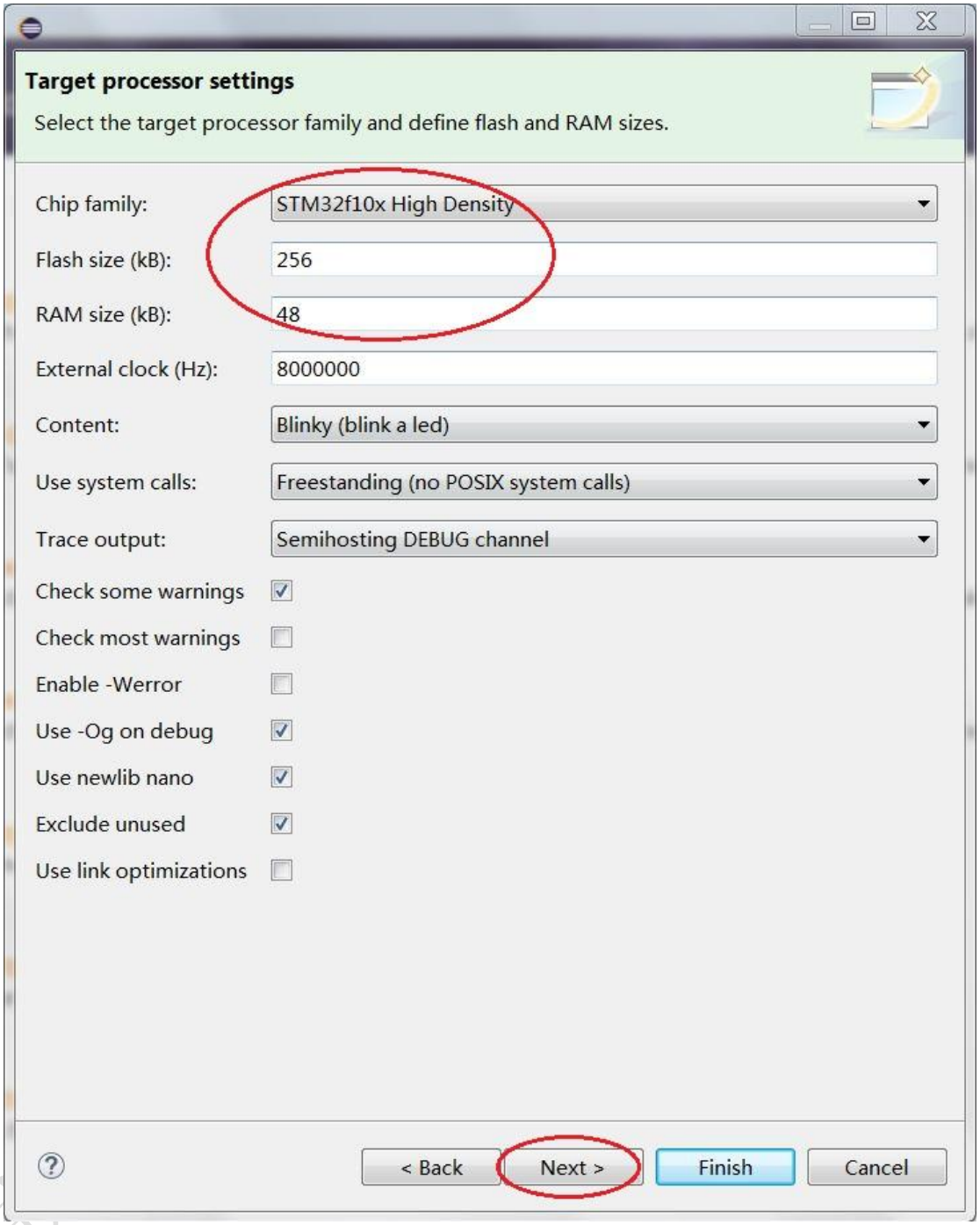

图 20、配置 ETA321 芯片参数

5、设置工程所用的文件夹，可以默认即可，点击“next”。

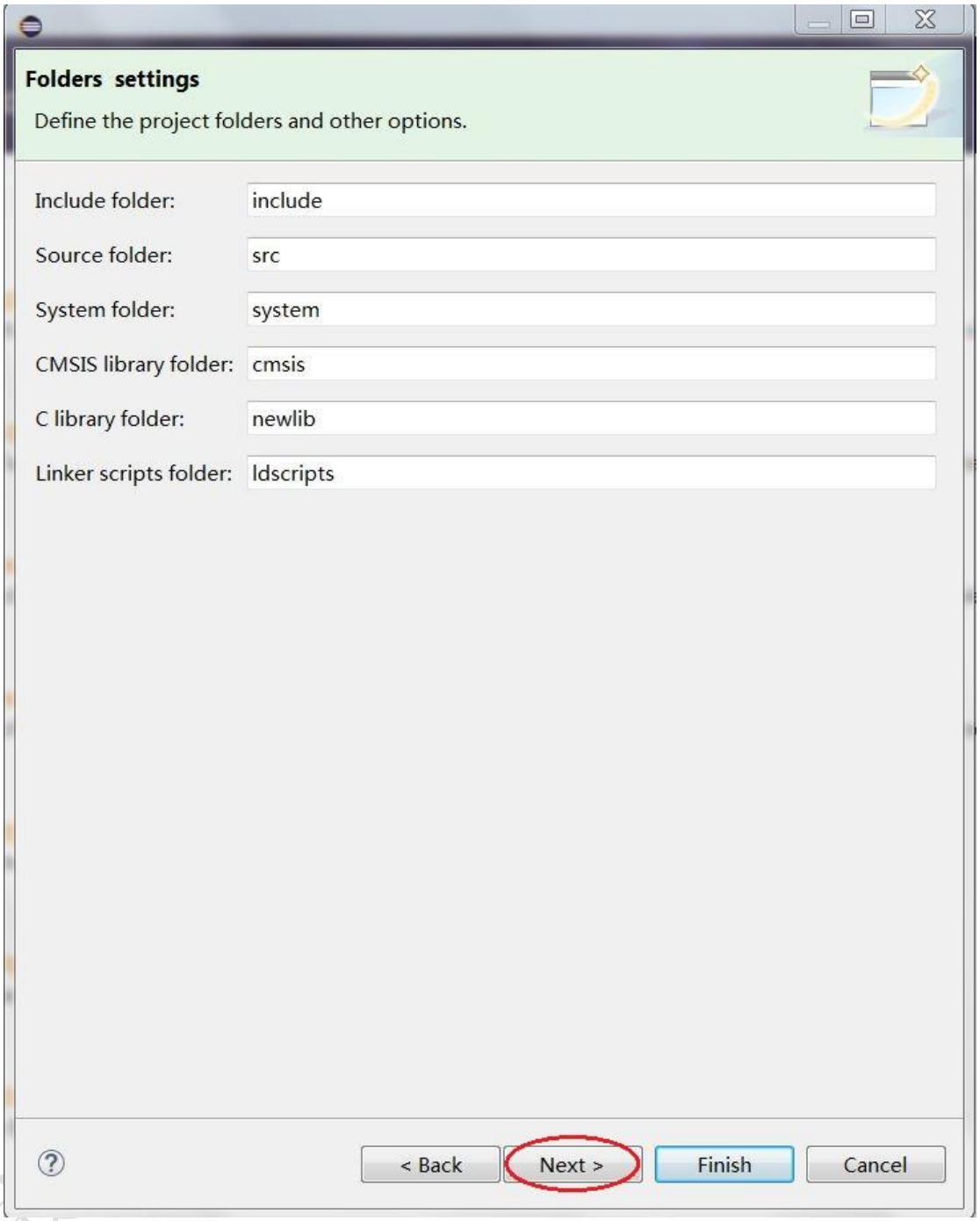

图 20、配置工程所使用的文件夹

6、选择工程中需要编译的项目，一般情况下,“Debug”与“Release”都会默认选择，点击“next”。

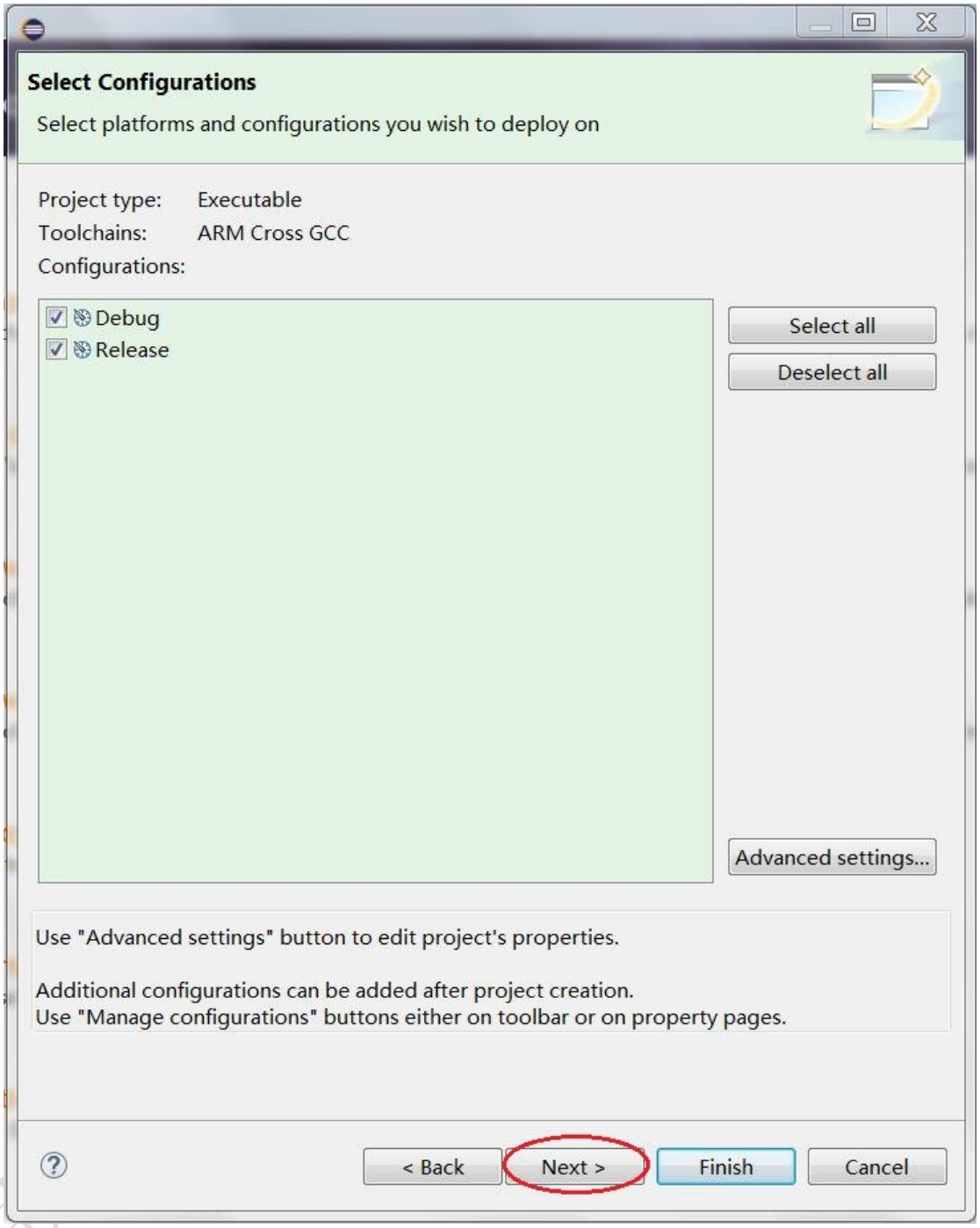

图 21、编译结果项目选择

7、设置工具链。如果设置了工具链环境变量且系统重启生效，则“Toolchain path”会自动填写完成。如果这里为空，可以点击“Browse”按钮，手动选择工具链的目录即可，即上面所说的工具链路径，最后点击“finish”完成工程建立。

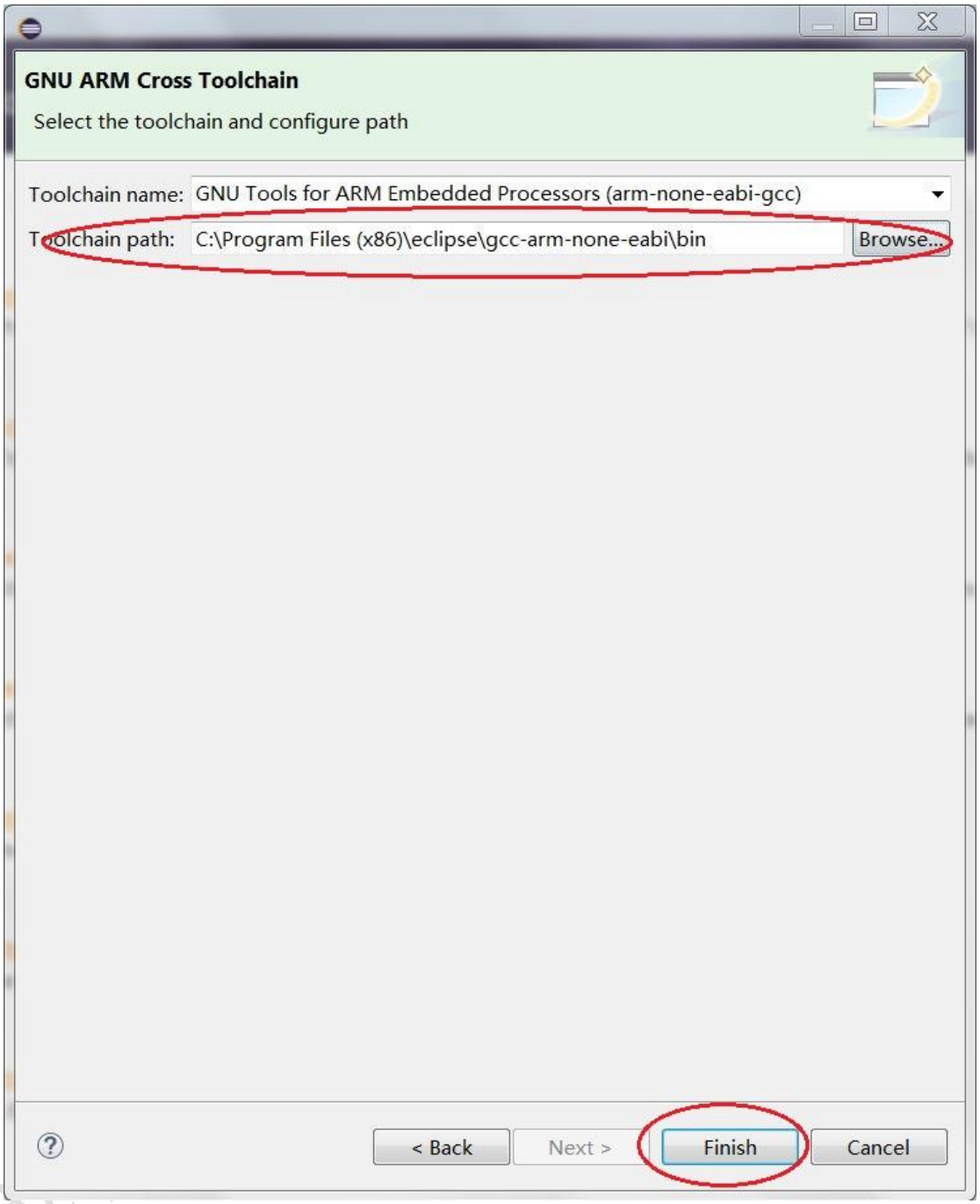

图 22、配置工具链路径

8、工程建立完成后，需要进行编译工具的配置。我们提供的 eclipse 工具包，有两个编译工具可以选择：CDT Internal Builder 或 Gnu Make Builder，使用其中一个即可。配置编译工具的方法如下：

首先，在 eclipse 环境中，选中所建立的工程，点击菜单中的“project”-> "Properties"，打开配置页面。

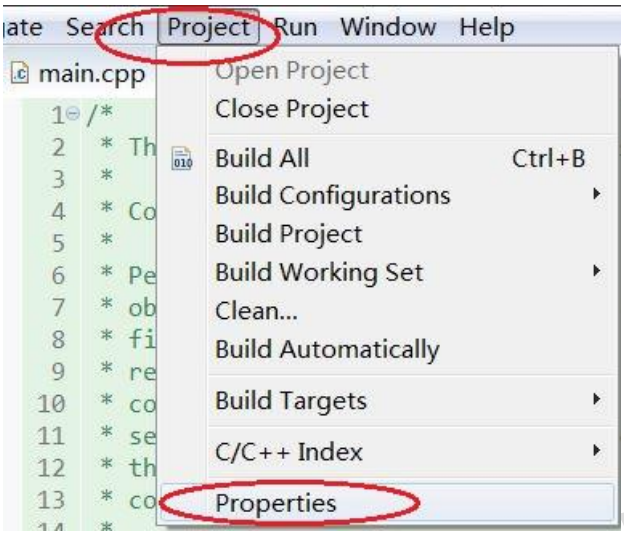

图 23、进入工程属性配置

a)、选择 CDT Internal Builder 编译工具

在弹出的配置页面中，展开左侧的“C/C++ Build”选项，点击“Tool Chain Editor”,在右边的参数“Current Builder”下拉列表中，选择 CDT Internal Builder，然后点击“Apply and Close”完成设置。

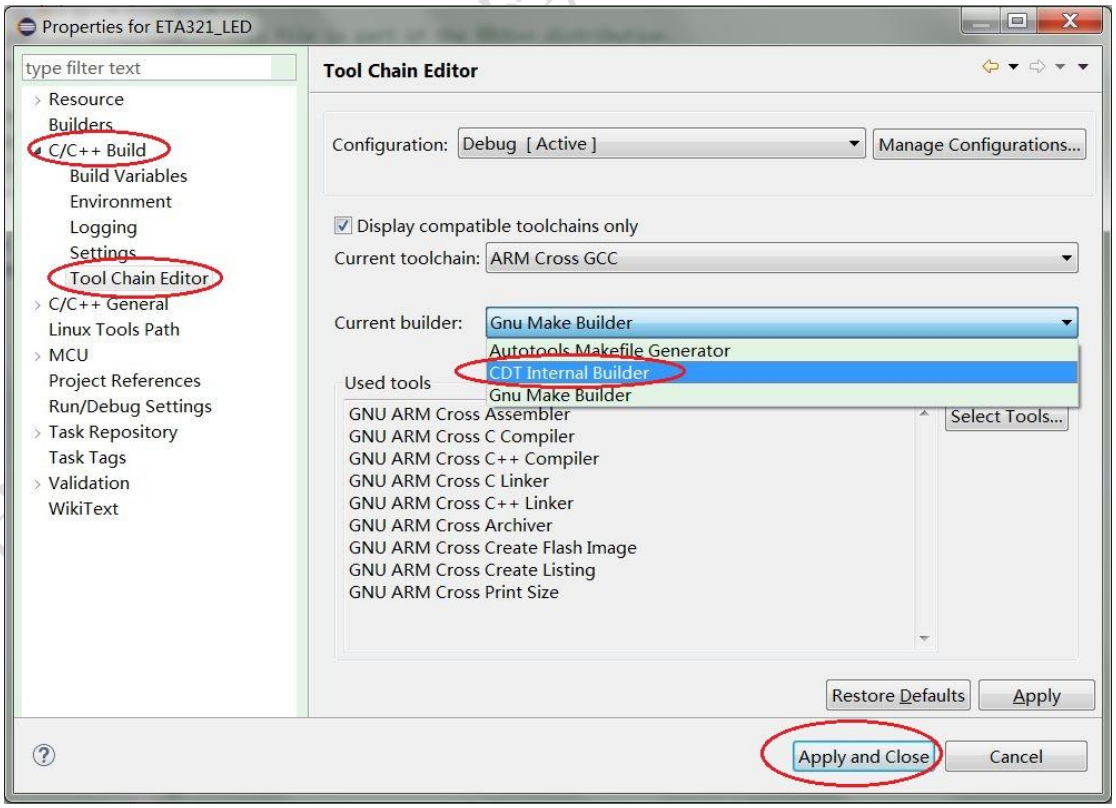

图 24、选择编译工具为“CDT Internal Builder”

b)、选择 Gnu Make Builder 编译工具

从上图中可以看出，eclipse 也可以使用 Gnu Make Builder 编译工具。在我们的 eclipse

工具包中，已经包含了该编译工具，在 eclipse 文件夹中的“GNU MCU Eclipse”就是该工具包，因此可以配置使用该编译工具。使用该工具的配置方法如下。

首先确认编译工具的路径并复制下来。

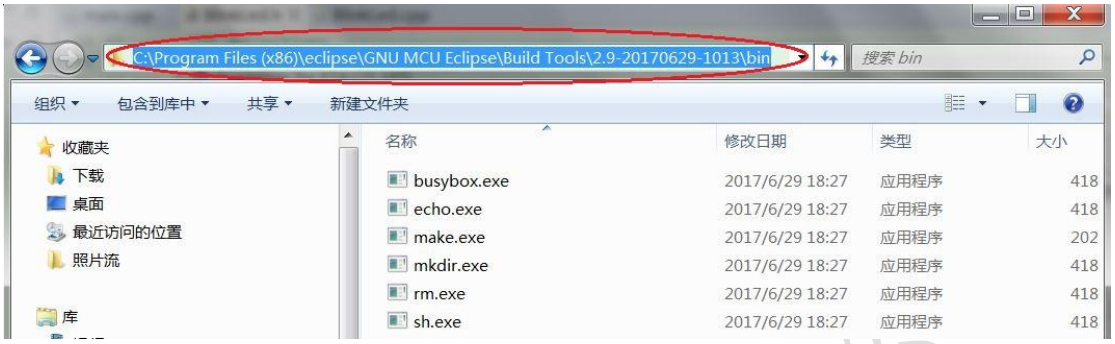

图 25、GNU Make Builder 路径确认

点击菜单中的“project”-> "Properties"，选中左侧的“C/C++ Build”下“Environment”，会在右侧列出相应的环境变量参数。单击选择右侧的“PATH”项，再点击“Edit”按钮，进行参数配置。

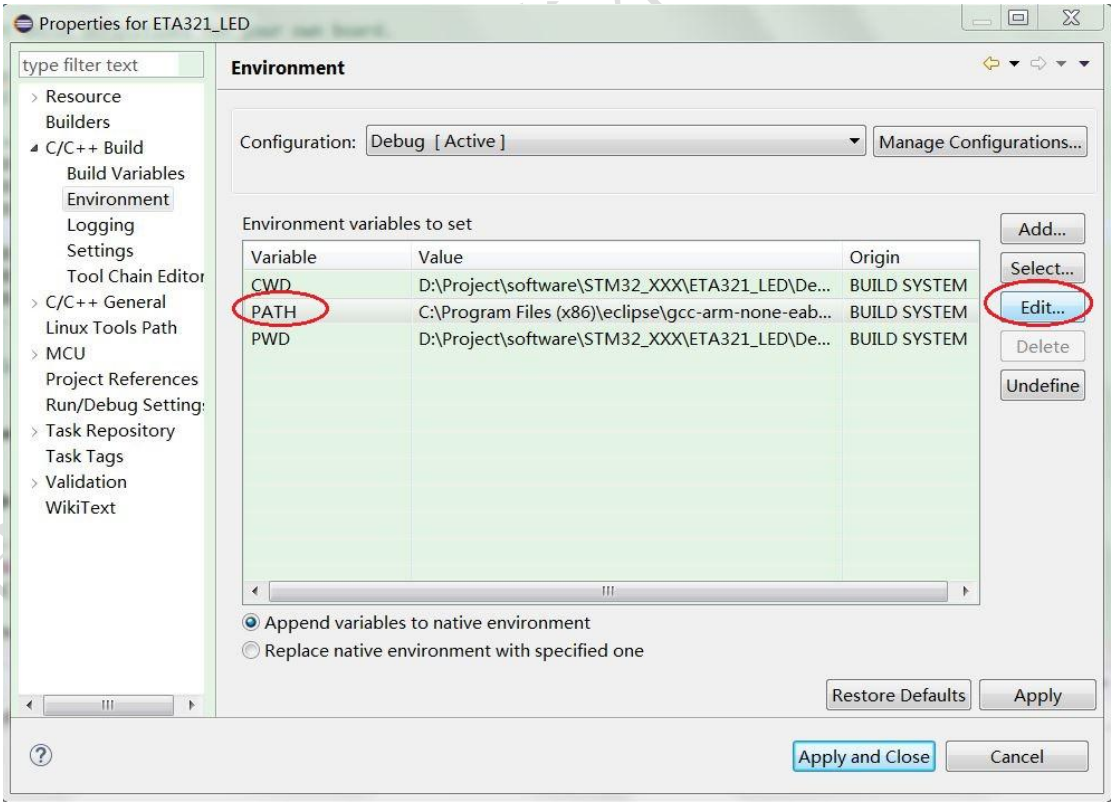

图 26、选择“Environment”配置项

在弹出的“Edit variable”参数修改页面中，添加 GNU Make Builder 工具的路径到 PATH

环境变量中。

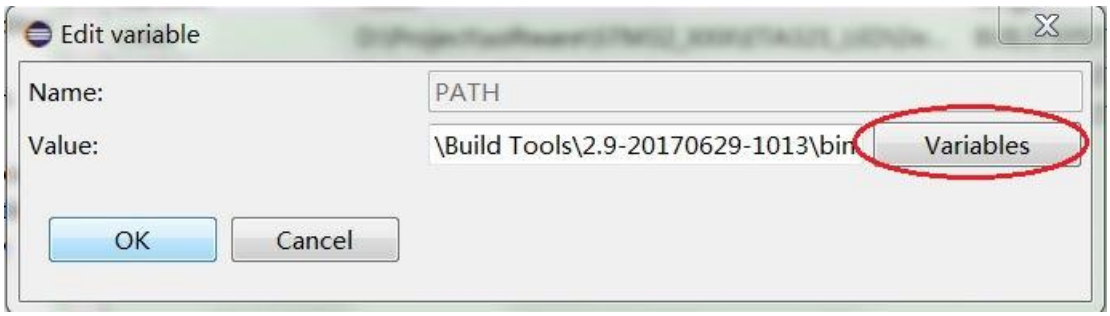

图 27、在 PATH 参数中，添加 GNU Make Builder 工具路径

最后，在左侧的“C/C++ Build”选项，点击“Tool Chain Editor”,在右边的参数“Current Builder”下拉列表中，选择“Gnu Make Builder”，然后点击“Apply and Close”完成设置。

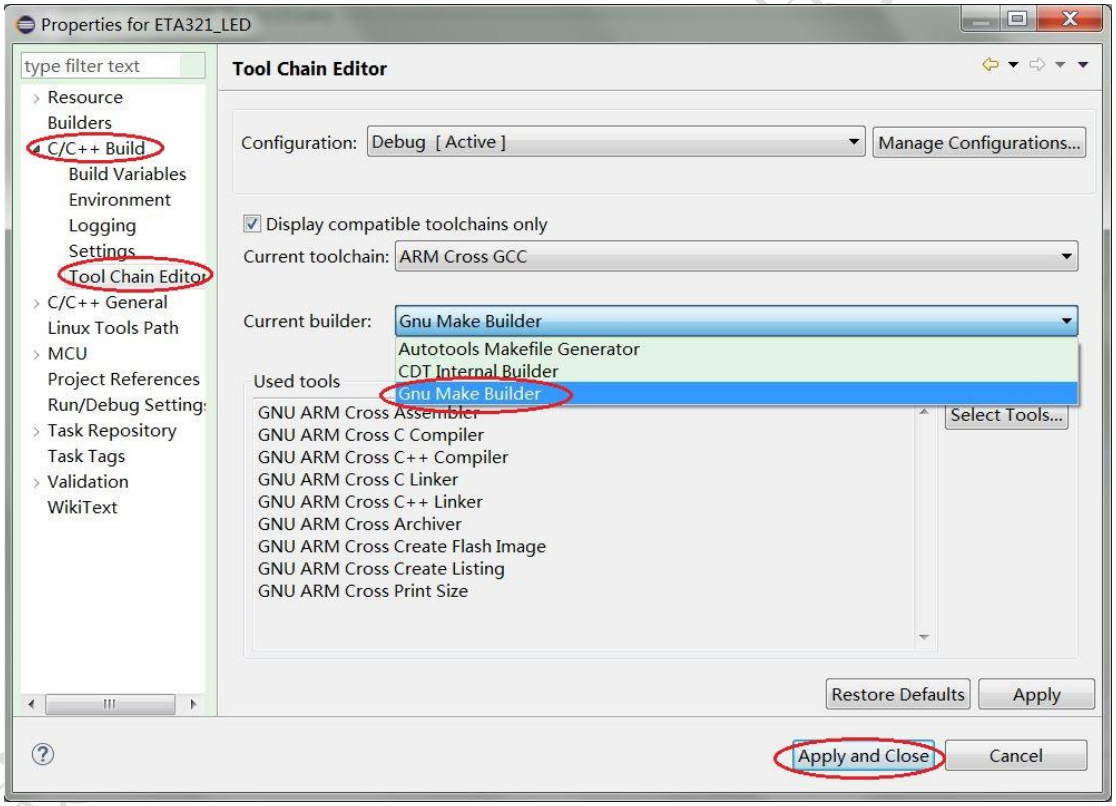

图 28、选择编译工具为“Gnu Make Builder”

10、由于 ETA321 模块上的 LED 灯使用的是 PB2 引脚，与建立的工程中默认使用的 GPIO 不同，所以需要修改代码，与 ETA321 的 GPIO 对应。需要修改的定义文件为 BlinkLed.h 文件，其中的 BLINK_JPORT_NUMBER 参数更改为 1，BLINK_PIN_NUMBER 参数更改为 2。

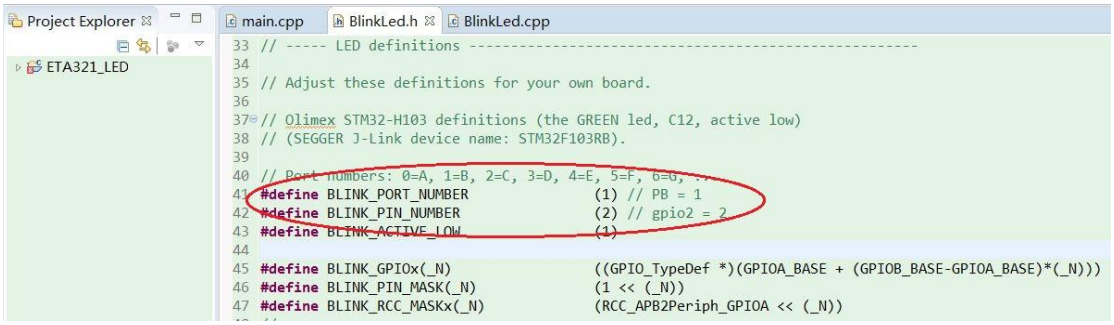

图 29、修改代码中的 GPIO 引脚定义为 PB2

11、接下来，就可以编译工程了。右键单击工程名，在弹出的选项中，点击“Build Project”，开始编译工程。

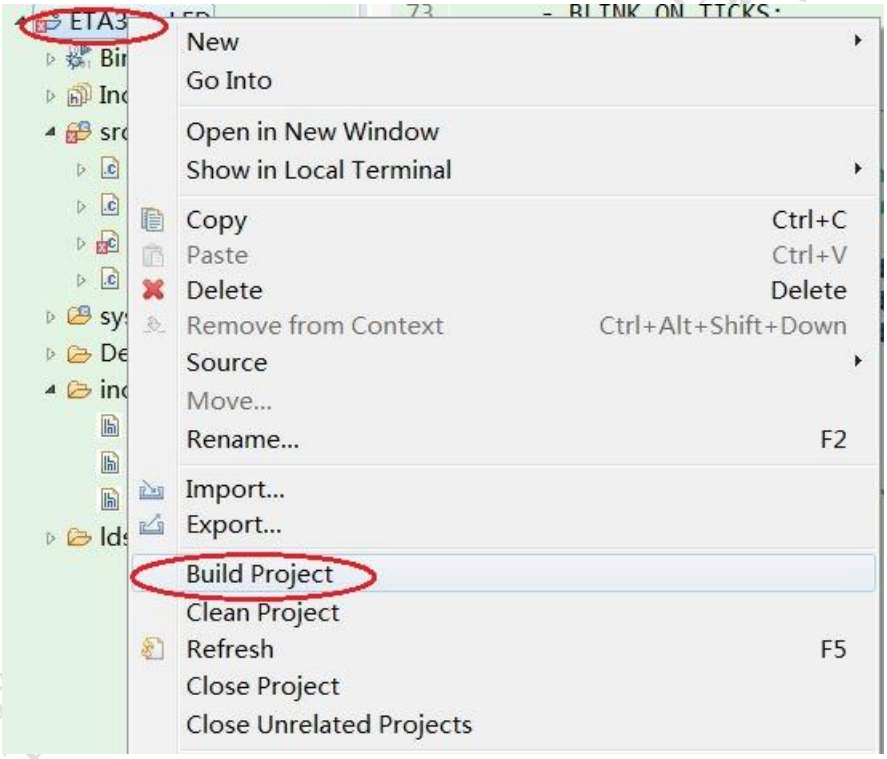

图 30、编译工程

编译成功后，将会生成 hex 文件与 elf 文件

图 31、工程编译成功后的信息

三、使用 flashloader 工具烧写 ETA321 应用程序

如果安装好了 Flashloader 工具，这时，可以找到工程目录下编译出来的 HEX 文件，利用 Flashloader 工具将其烧写到 ETA321 中，查看程序运行结果。烧写方法如下：

1、拔动开关到 ETA321 模块上“3.3V”端，使 ETA321 处于下载状态，即：ETA321 模块上的红色 LED 灯点亮。用 USB 线连接 ETA321 到 PC 机的 USB 端口。如果是首次进行连接，需要安装 USB 驱动，等一点时间，直到驱动自动安装完成。

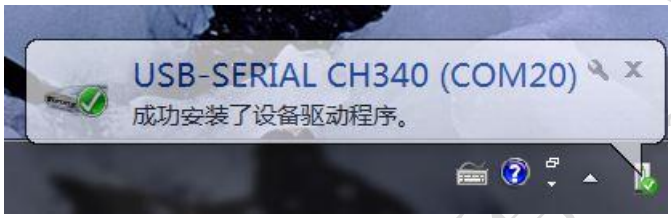

图 32、首次安装 USB 驱动

2、从“设备管理器”中，查看 ETA321 模块虚拟的串口编号，即 USB-SERIAL CH340 虚拟串口编号。

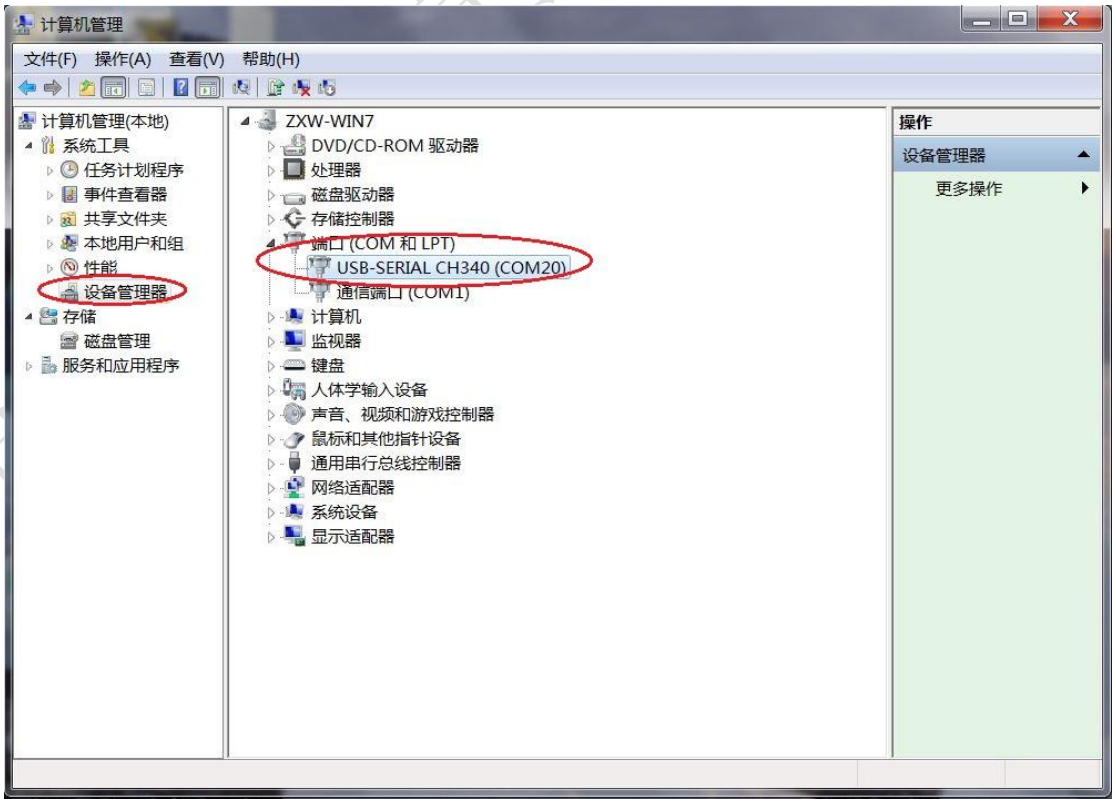

图 33、检查、确认 ETA321 串口驱动及串口编号

3、启动 Flashloader，“Port Name”端口选择 CH340 对应的串口编号，校验选择“none”。

图 34、Flashloader 通讯参数

如果确认这里的参数都正确，但是又不能连接到芯片，则需要重新检查一下 CH340 虚拟串口的“端口设置”属性，并将波特率（位/秒）更改到 115200 即可。

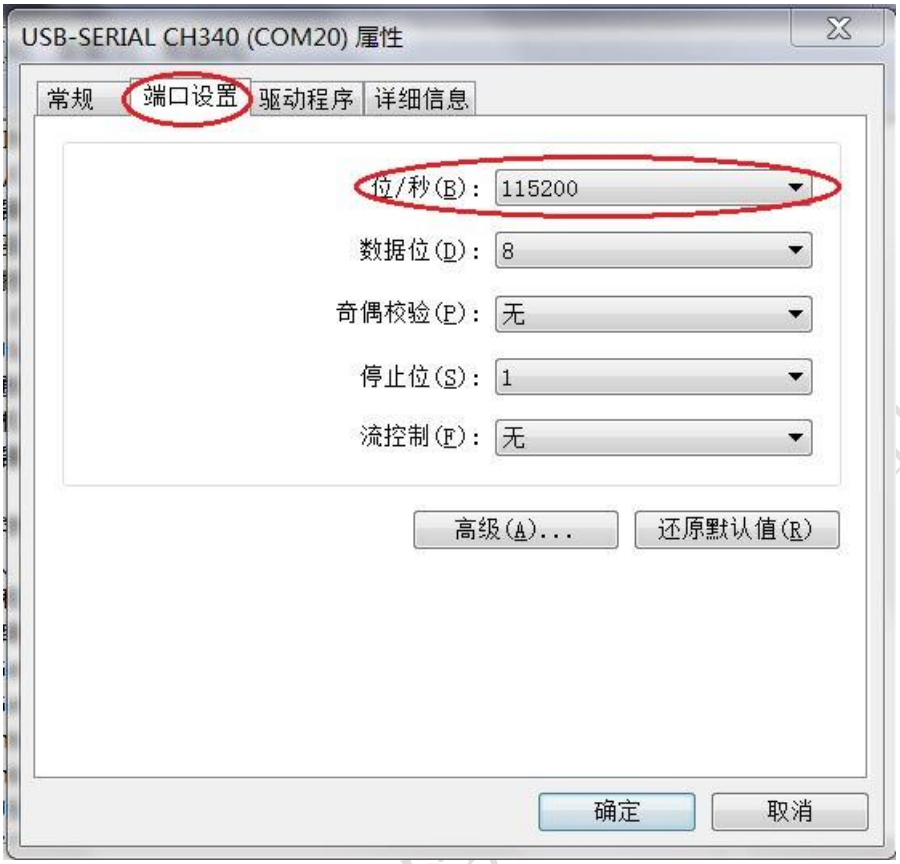

图 35、ETA321 虚拟串口参数属性

4、点击下一步，Flashloader 会读取到芯片相关的存贮器信息，再下一步，就可以到烧写界面。

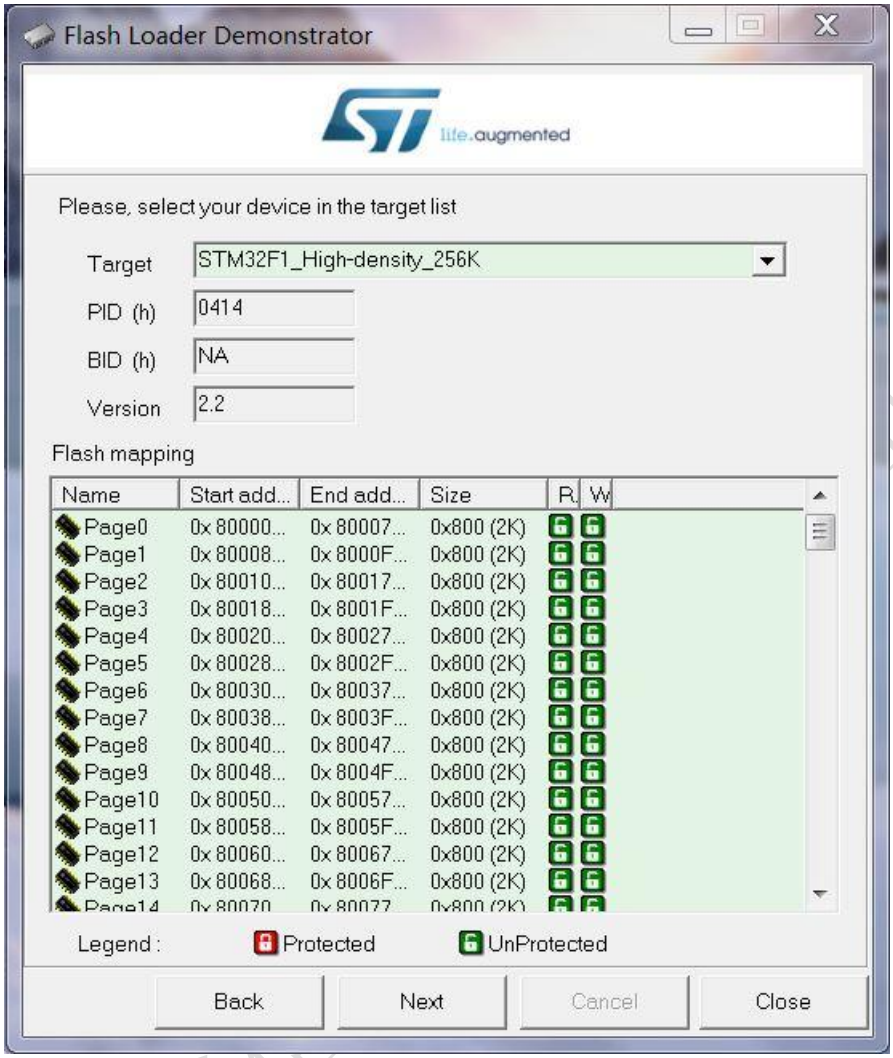

图 36、Flashloader 工具读取到的芯片参数

5、选中“Download to device”单选按钮，并将“Verify after download”选择上，即下载完以后进行校验。

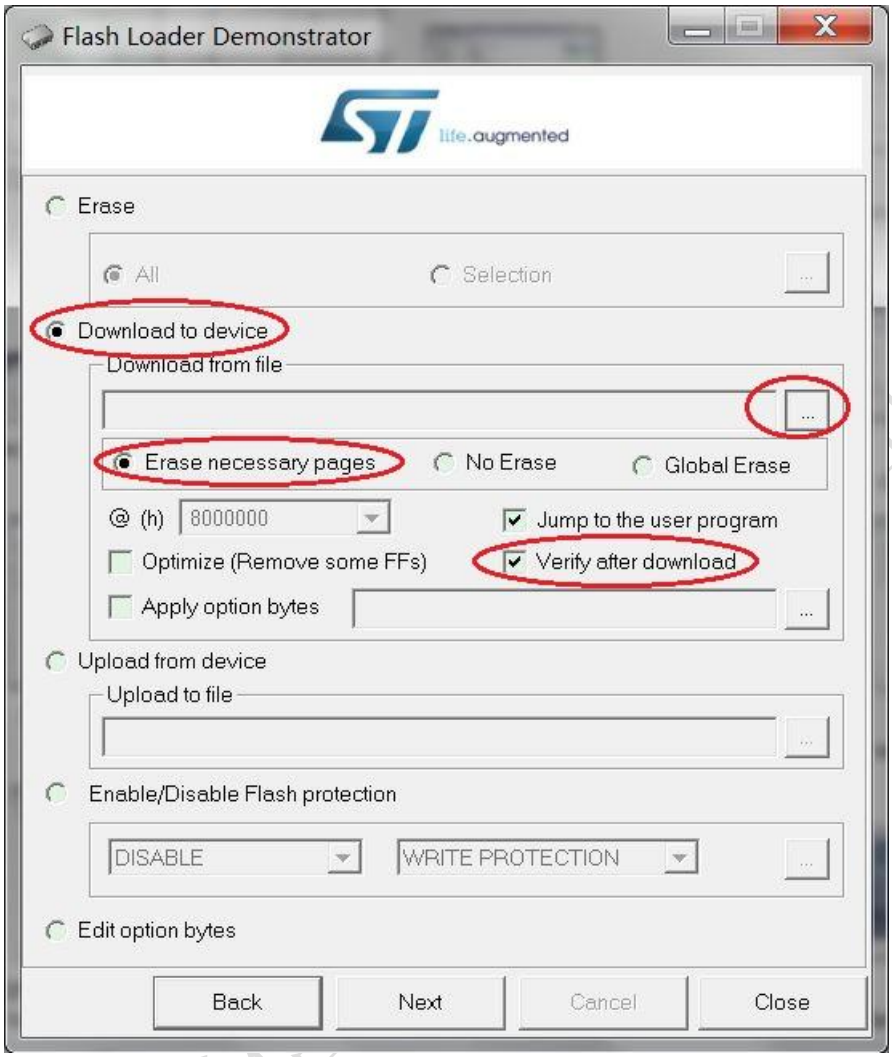

图 37、烧写界面

点击“Download from file”右侧的按钮，在弹出的页面中，选择需要烧写的 hex 文件。需要注意，这里要指定文件类型，Flashloader 软件默认是*.s19，要改为*.hex，否则在文件夹中看不到相应的 hex 文件。

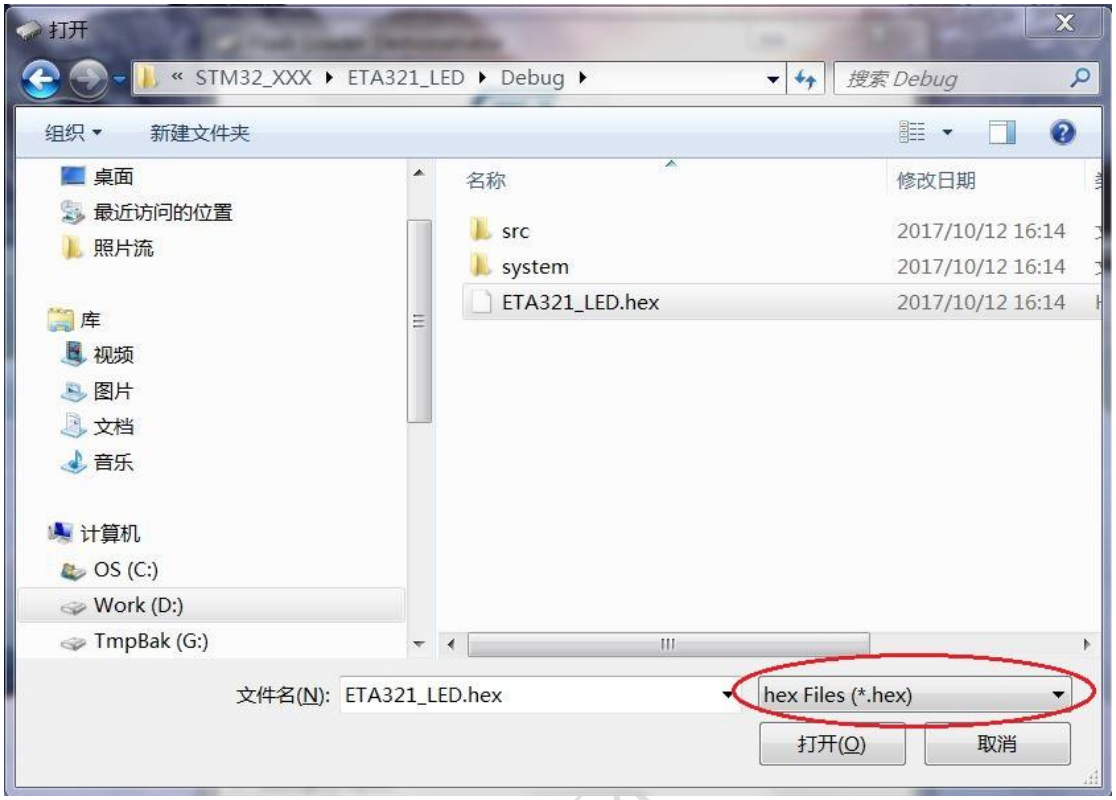

图 38、选择烧写文件

选择好文件后，点击“打开”按钮，返回烧写页面，再点击“NEXT”便开始烧写。进度条显示绿色，即烧写成功。

图 39、文件烧写成功

四、利用 J-Link 调试程序代码

首先安装 J-Link 的驱动，在我们提供的软件工具包中也有：JLink_Windows_V620c.exe。安装完成后，就可以使用 Jlink 进行软件调试了。在调试的时候，ETA321 要处于运行状态，即：拔动 ETA321 模块上的开关到 S2 端，红色 LED 不亮。

使用 JLink 进行调试之前，先用 JLink GDB Server 工具进行芯片连测试，以确认 Jlink 在正常工作。连接测试流程如下：

1、使用 JLink 的 SWD 信号接口，连接到 ETA321 的 CN3 插针上，并给 ETA321 上电，且 ETA321 处于运行状态。

2、启动 GDB Server 工具

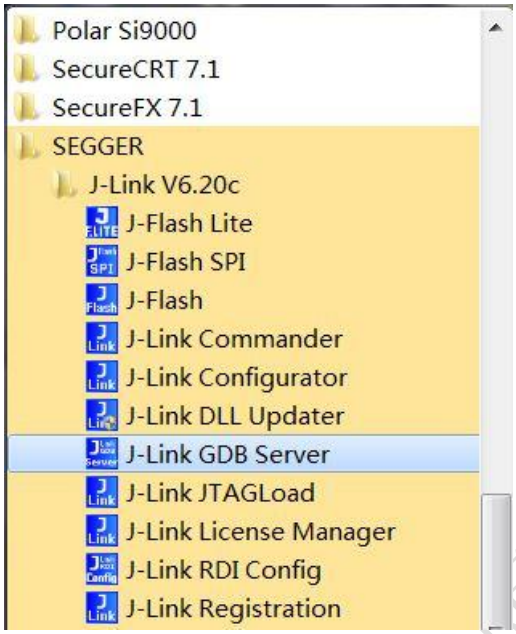

图 40、启动 J-Link GDB Server

3、首先选择器件，GDB 启动起来后，点击“Target Device”右侧的按钮，弹出“Target device settings”页面。在“Manufacturer”中选择“ST”，在“Core”中选择“Cortex-M3”，然后在下面的列表中，选中“STM32F103RC”，最后点击“OK”。

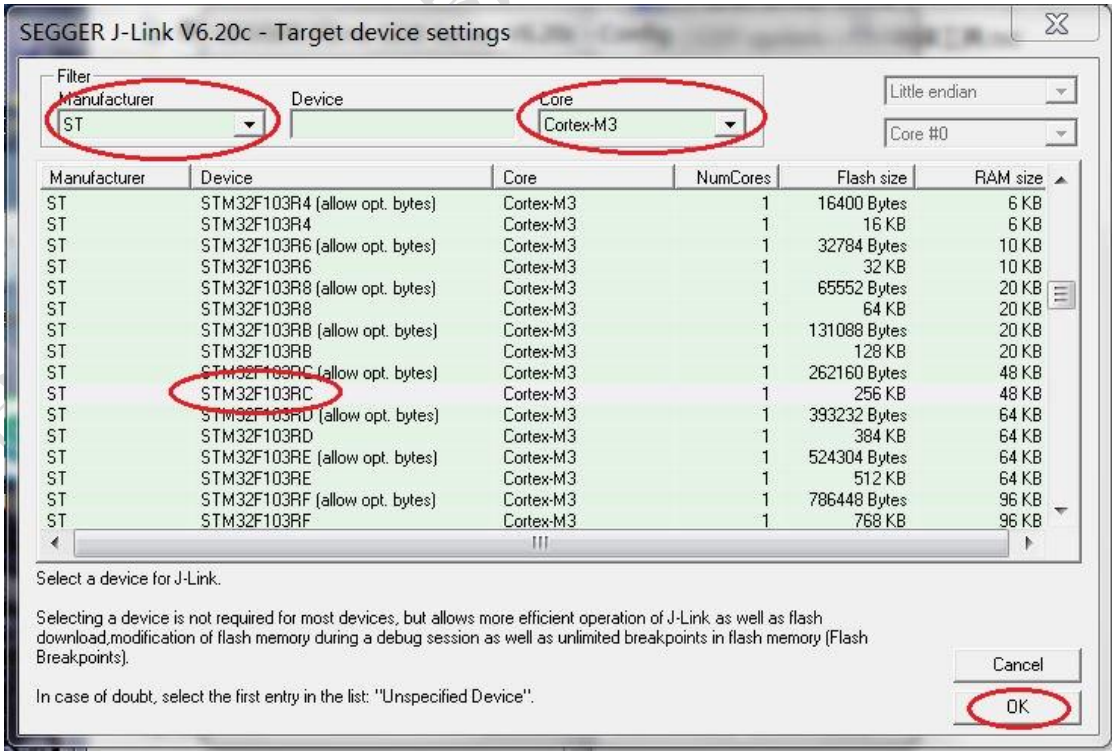

图 41、选择目标器件

4、器件选择完以后，还需要配置 GDB Server 连接参数：使用 USB 连接 J-Link，SWD 接口连接目标芯片，1000Khz 速率，最后点击“OK”进行芯片连接。

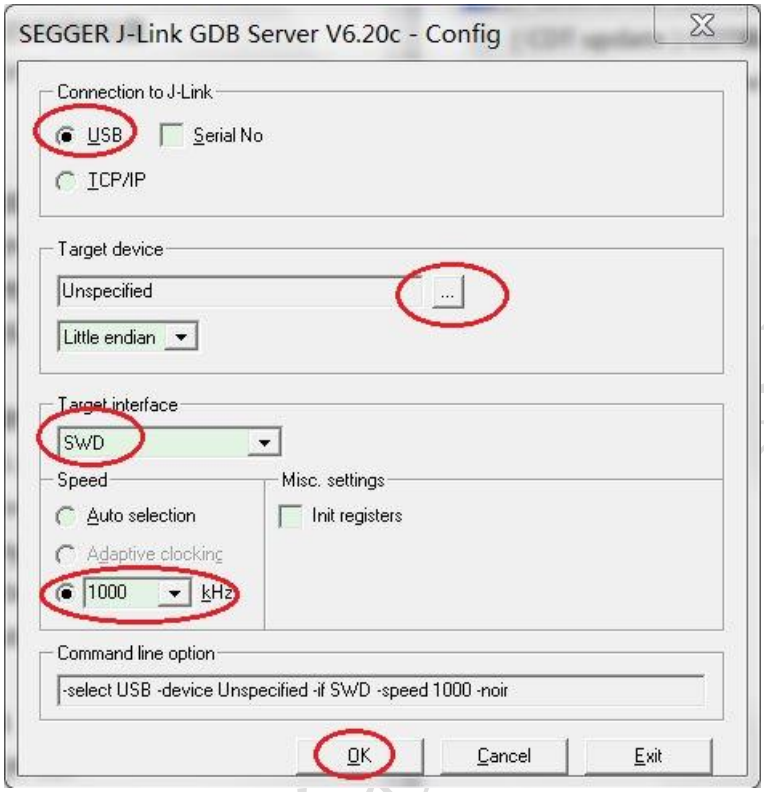

图 42、配置 GDB 参数

5、连接成功以后，J-Link 项与 CPU 项显示绿色状态，并能读出 CPU 的工作电压。

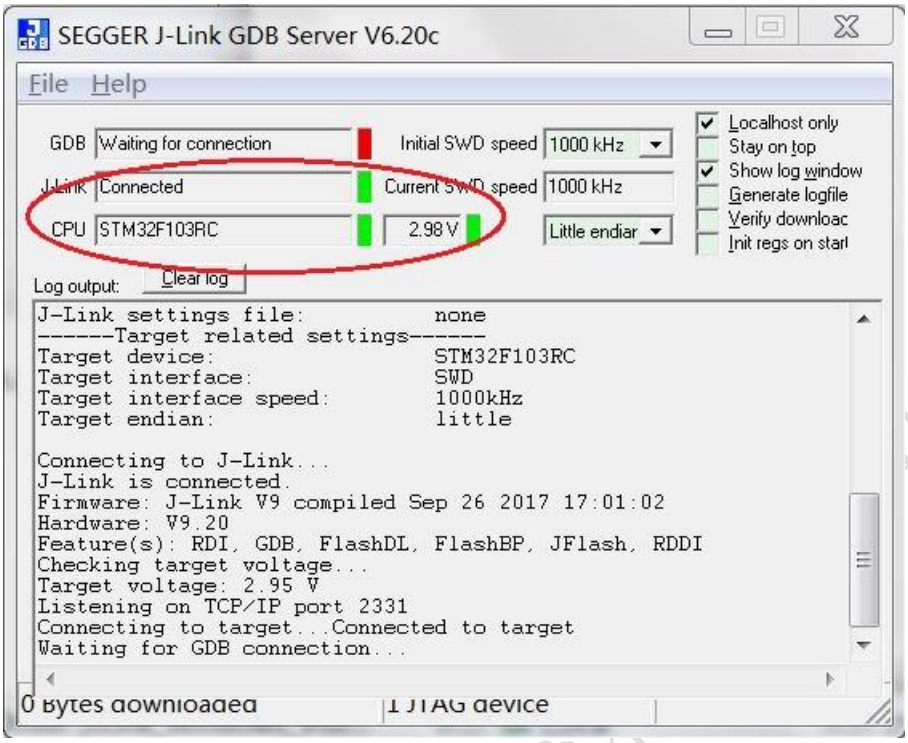

图 43、连接成功

到这时，说明 JLINK 的驱动以及硬件连接环境均正是正常，就可以在 eclipse 环境中，利用 Jlink 进行程序调试了。调试方法如下：

1、启动 eclipse，打开之前已经编译通过的工程文件 ETA321_LED。

2、从菜单栏上选择“RUN”下面的“Debug Configurations”。

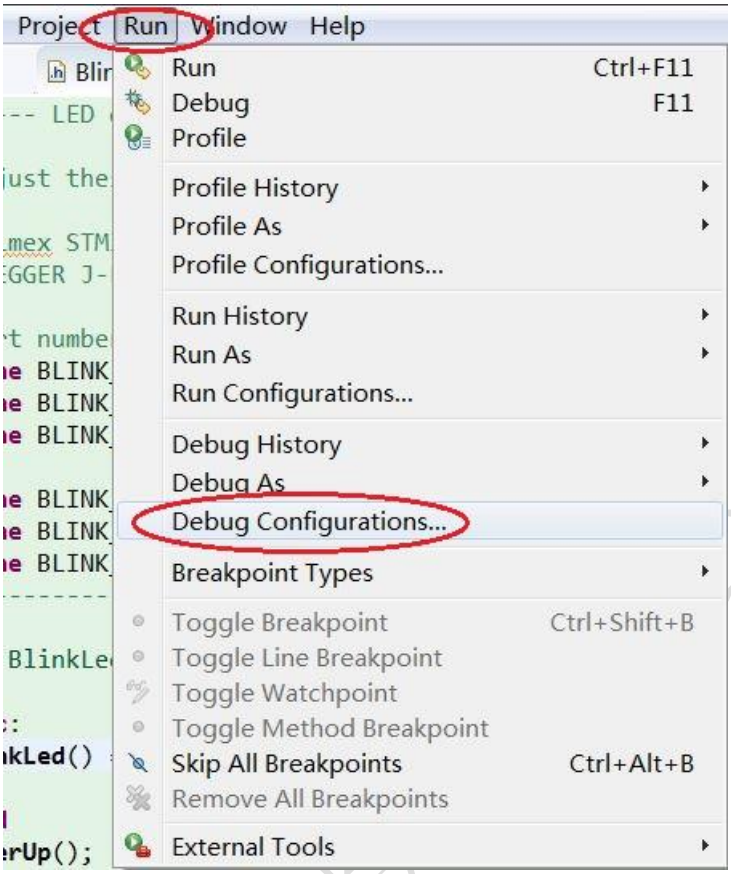

图 44、选择 Debug Configurations

3、在左边列表中，选中“GDB SEGGER J-Link Debugging”，再点鼠标右键，在弹出的菜单上，点击“New”，在该项目下建立一个新的调试项目。

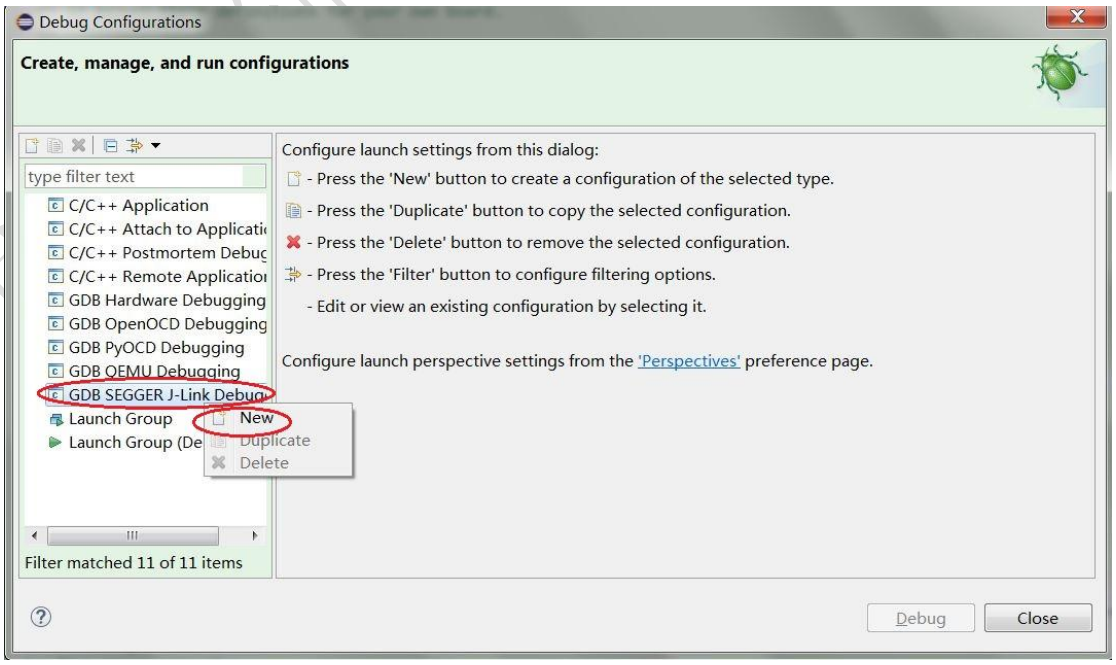

图 45、新建调试项目

4、点击新建立的调试项目，展示出右侧的配置页面，这里主要需要配置/检查“Debugger”与“Startup”这两个配置页面。

在“Startup”配置页面中，去掉如下图中红圈中的两项，并点击“Apply”，以确认应用。

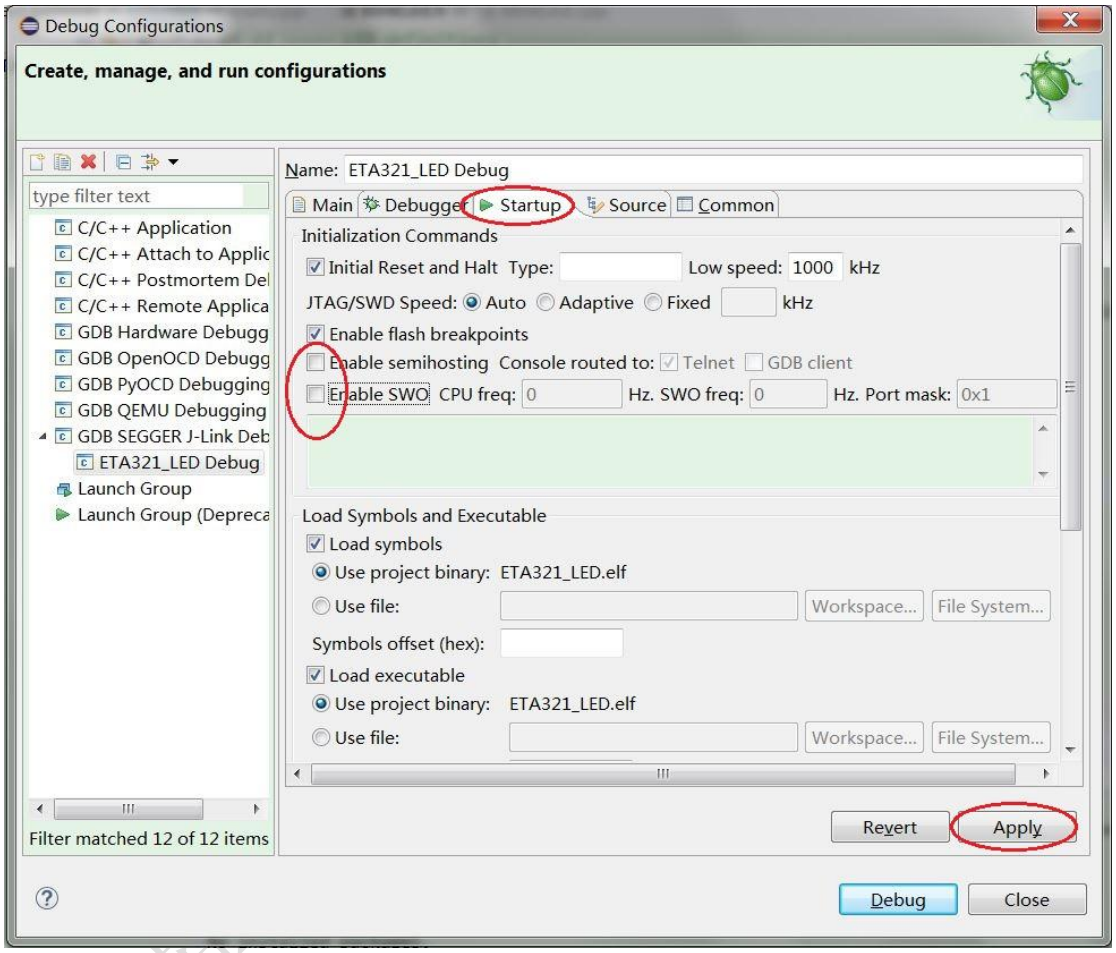

图 46、配置 Startup

在“Debugger”配置页面中，需要检查多项参数：

1、检查“Actual executable”参数指向 JLinkGDBServerCL.exe；

2、并输入“Device name”为“STM32F103RC”；

3、 “Endianness” 选择为“Little”；

4、“Connection”选择为“USB”；

5、“Interface”选择为“SWD”；

6、“Initial speed”配置为 1000KHz。

确认参数正确以后，点击“Apply”应用配置参数。最后点击“Debug”进入调试。

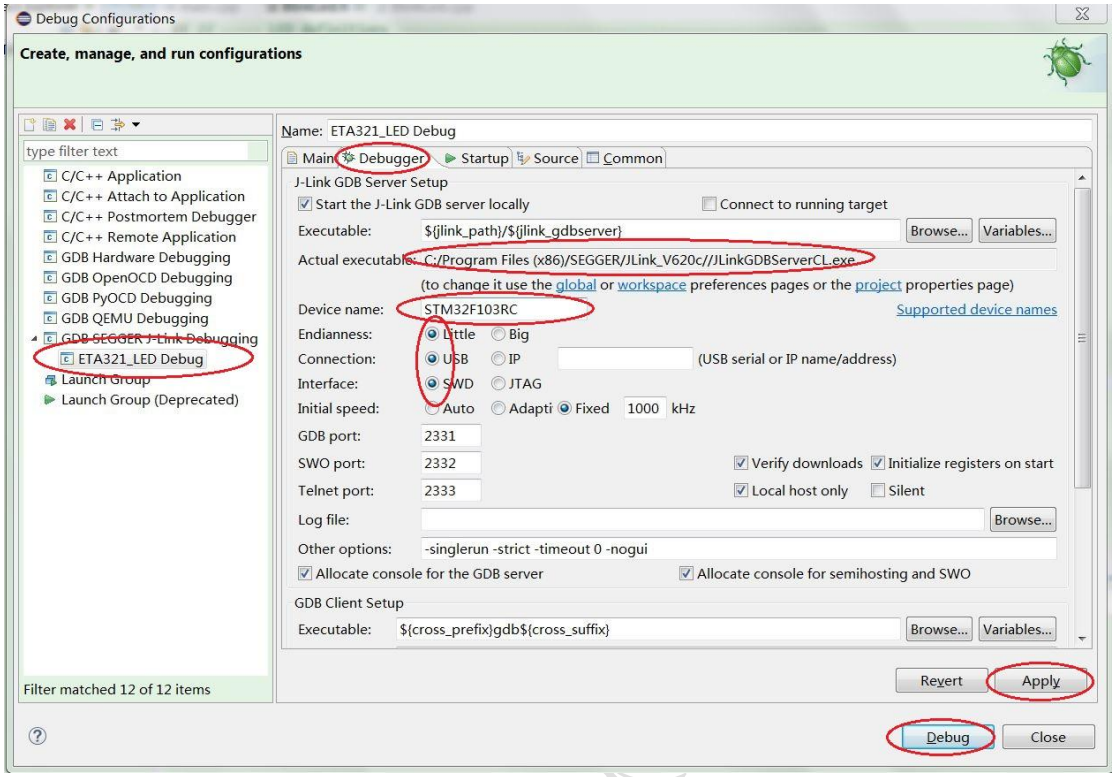

图 47、配置 Debugger

7、启动调试以后，程序入口停止在 main()函数，这时可以按 F6 进行单步运行、F5 可以进入函数内部跟踪运行。