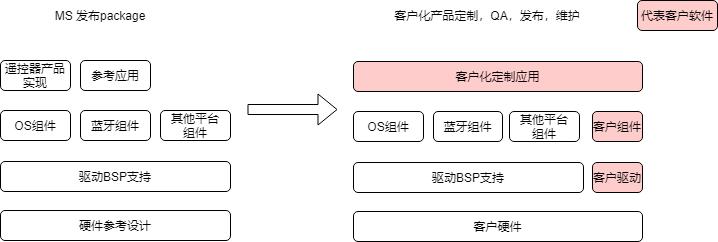
# Mooresilicon Cygnus SDK用户指南

## 概述

MS蓝牙SOC芯片支持BLE（低功耗蓝牙）5.2， 主频最高96Mhz， 内置96K RAM和512K SPI Flash。芯片的主要应用场景包括 蓝牙遥控器等智能家居 以及 IOT设备。MS 软件SDK平台的目的是帮助客户快速开发和生产基于MS蓝牙SOC芯片的产品，平台和客户产品定制关系如下图所示：

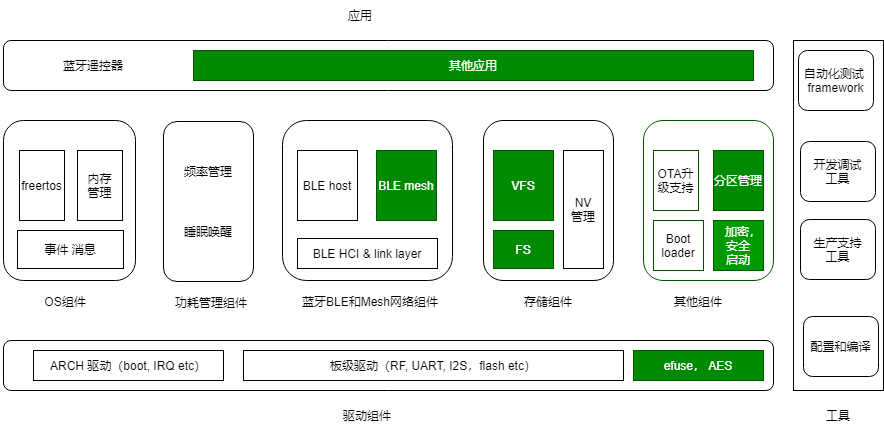


MS 软件 SDK平台支持如下特性：

1. 支持基本功能集： 包括SOC的所有外设驱动，BLE protocol stack (controller & host)，BLE application （HID，Battery等常用 profile），功耗管理等功能；
2. 满足客户基于平台做二次开发的要求：提供清晰的接口，易用的开发环境，高效的debug手段，完整的生产工具和参考文档；
3. 提供稳定的各种扩展软件组件（模块），以适应客户对不同产品的需求：提供基于BLE的MESH网络，RTOS （FreeRTOS），基于flash的通用文件系统等；
4. 满足可扩展（可以支持将来不同型号的SOC ，容易集成更多的软件模块），可配置（例如满足不同客户对memory size的要求 而进行组件的裁剪）等要求；

## 系统框架

SDK平台的软件框架图如下：



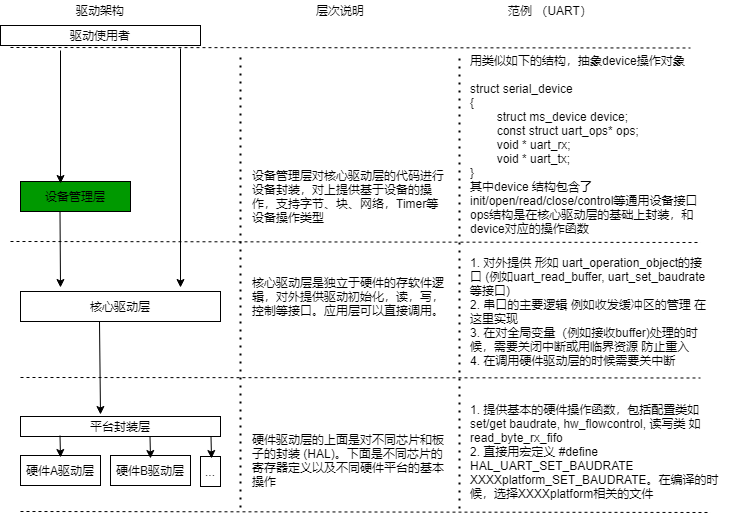
1. 扁平化层次管理。包括驱动，OS，所有的独立软件包都是组件(component)形式管理，对外提供组件的接口；
2. Flatbuild，app和组件打包在一起编译。各个组件都编译成独立的lib image。在配置和编译阶段可以通过配置工具配置需要的组件以及组件的参数；
3. 分阶段实现。白色组件为高优先级 (目标22年Q3 prototype)，绿色部分为低优先级实现。第一个产品蓝牙遥控器基于高优先级组件开发；

## 驱动组件

驱动的整体框架分层 设备管理层，核心驱动层 以及 硬件驱动层 3层。调用驱动的组件可以调用核心驱动层接口。硬件驱动层需要处理对不同芯片和平台的抽象封装（HAL），以便对核心驱动层提供固定的接口。驱动的逻辑必须在核心驱动层完成。

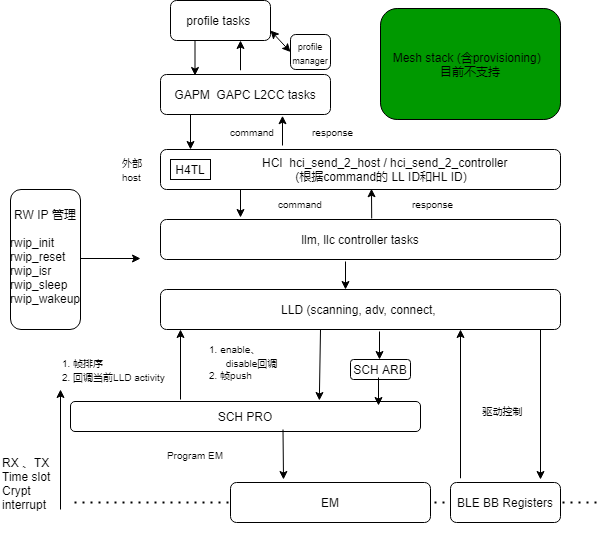
引入设备管理层的目的是将来方便客户已有的app移植，作为 低优先级实现。设备管理层做成 可以通过宏定义打开/关闭；

不同芯片/平台的代码 在不同目录分开管理，通过make PLATFORM = XXXX 隔开。核心驱动层调用统一的 HAL硬件操作接口；



## 蓝牙组件

* Host部分包括所有的 profile，GATT，GAPM(profile 管理模块）以及 Controller的LLM, LLC等control部分以 task方式运作。Task之间使用ke\_msg\_send传递消息。
* Controller 的LLD，schedule 通过中断方式运作。通过BLE BB的 rwip\_ist入口 处理所有的RX/TX/TimeSlot/ FR等中断。中断callback到当前活跃的LLD activity
* HCI主要是在 Host和Controller之间传递消息，所有发给HCI的command需要注明target id。HCI可以通过H4TL和串口驱动外接host。
* 采用ble广播的方式就行信息发送和接受的蓝牙组网 mesh，作为 低优先级实现。



作为BLE SOC软件平台，CEVA BLE SW是平台最重要的模块之一。

CEVA 是个典型的前后台系统：蓝牙host（gap, profile等模块）以及low layer （link layer manager, link layer control等模块）都是以 background “task”方式运行。其他对时间敏感的模块 （low layer driver, scheduler， arbitrator）以中断方式 （ rwip\_isr ）进行。为了支持用户添加profile ，提供了 prf\_env相关的操作 对 profile tasks 进行管理。

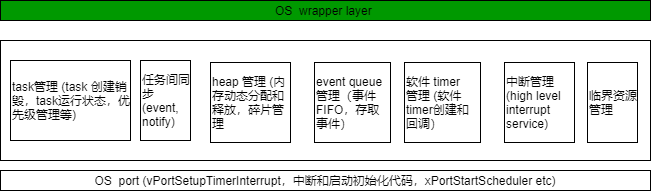
系统以 mainloop为主循环，从KE event queue获取事件，并根据 task id分发到对应的“task”。在这里根据 message id 选择对应的 message handler处理。处理之后如果 产生新的消息需要其他“task”处理，push新消息到事件队列。

CEVA定义的“task”，包含了task id， 该task维护的状态变迁，该task需要处理的事件和对应的handler组成。本质上是一些message handler 的集合，和mainloop跑在同一个大循环里面。

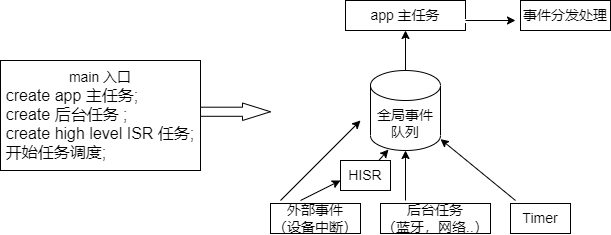


## 操作系统

SDK支持FreeRTOS操作系统，系统框架如下。其中OS wrapper layer 作为低优先级实现，以便将来支持不同的RTOS:



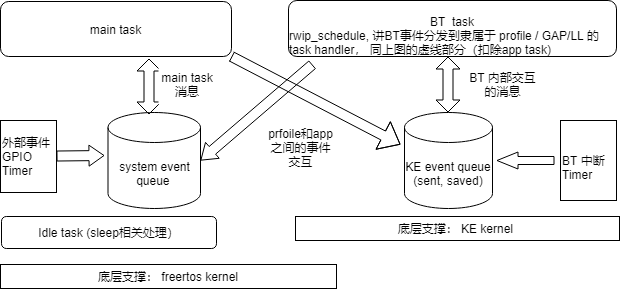
引入多任务RTOS之后的系统整体运作框架如下所示，在中断处理比较复杂的情况下，为了减少关中断事件，中断服务程序ISR只做最基本的操作。ISR触发HISR (high level ISR），在HISR做驱动逻辑处理之后 传递事件到事件队列。



## 蓝牙组件与操作系统的运行关系

运行过程如下图所示：

1. SDK平台提供3个任务。 Main task是app task，负责类似蓝牙遥控器app。BT task是管理 CEVA目前background “task” 的管理模块。 在没有事件处理的情况下, idle task被唤醒（FreeRTOS内核实现）。 在idle task hook (vApplicationIdleHook) 中处理睡眠相关的事情 （判断进入低功耗状态）。
2. 蓝牙task会轮询KE event queue ，和之前一样，根据 rwip\_configure.h定义的 task id和message id找到message handler 并处理。蓝牙task主体是 ke\_event\_schedule， 处理 KE\_EVENT\_TIMER（通用timer） 和 KE\_EVENT\_KE\_MESSAGE。
3. 蓝牙往app发送消息的时候，需要发送到 system event queue （需要改造目前 profile发给app的消息）
4. App往蓝牙发消息的时候，需要发送到 KE event queue （参考CEVA app\demo\src 目录 hid, battery 往对应profile的消息发送 ）
5. 蓝牙用到的arbiter timer，alarm timer保持之前的处理方式。给app以及profile使用的通用 timer （co\_time），也可以考虑保留 （risc-v 的timer只用来支持蓝牙之外的事情）
6. 底层的支撑包括 内存管理不需要变化。



## 功耗管理

功耗管理的系统框图如图所示：

绿色部分（模块别power domain控制）芯片不支持，暂不实现

1. 最底层是芯片支撑下的软件功耗feature；
2. 系统的3种工作模式，依赖这些软件功耗feature的组合实现；
3. Power Management模块对外提供接口，主要内容应当包含：

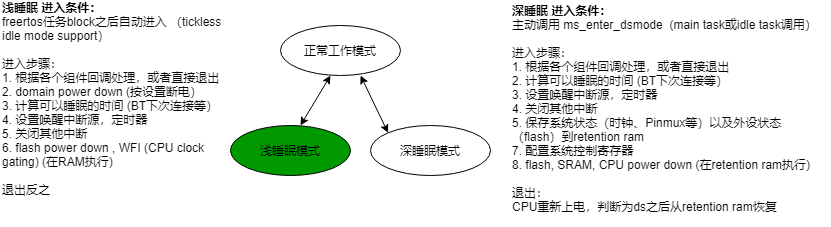
* 设置当前工作模式，缺省为正常工作模式
* 设置/获取 所有可以控制的power domain的状态
* 设置/获取当前工作模式下CPU工作频率
* 设置/获取对CPU资源的特殊要求，例 lock 最低运行频率
* 各个组件注册 浅睡眠和深睡眠前后的call back函数
* 进入睡眠之前设置唤醒源
* 获取当前的电池电量



工作状态切换示意图 （浅睡眠模式作为 低优先级支持）

浅睡眠的使用场景：复杂应用，频繁短时间睡眠

深睡眠的使用场景：应用场景单一（类似蓝牙遥控器），功耗敏感

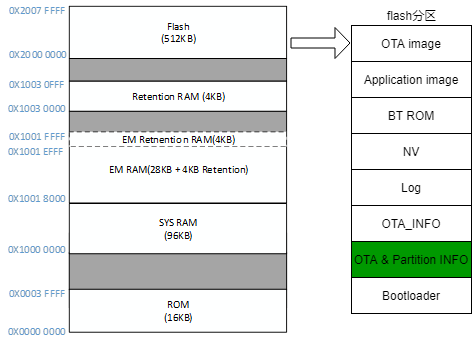


## 分区管理

SOC片内支持16K片内ROM， 512K SPI flash，96K System RAM，32K 蓝牙专用的EM RAM 以及 4KRetentionRAM。考虑到客户的具体项目，比如可能不需要支持OTA，或者需要更多的数据存储，SDK平台需要支持Flash可配置的分区管理。这部分作为低优先级支持。

为了支持flash分区管理需要完成以下事项：

* 在配置工具提供分区配置管理，生成partition info bin
* partition info bin 含分区名,类型（数据/程序）,偏移量, 大小
* partition info bin 固定烧入在紧跟bootloader 的分区
* 烧入程序支持对 bin /data大小的检查和错误报告



## 系统启动

系统启动采用三段式启动过程，如下图所示：

1. ROM程序 被固化在MS BT SOC内部的 ROM 中，它从 flash 起始位置 加载二级引导程序 bootloader。
2. Bootloader 从 flash 中加载分区表, 根据当前OTA info判断并跳转到正确的应用程序。
3. 应用程序启动，判断先前是否deep sleep状态。如果是的话，从retention ram恢复参数。应用程序初始化，FreeRTOS 任务创建，进入app main。
4. 为了满足IOT的一些特殊场景，要支持以下功能（低优先级）:

* Deep sleep stub，允许系统从DS恢复后快速处理简单事务；
* Flash加密，secure boot支持 以及 上面说的可配置分区关；

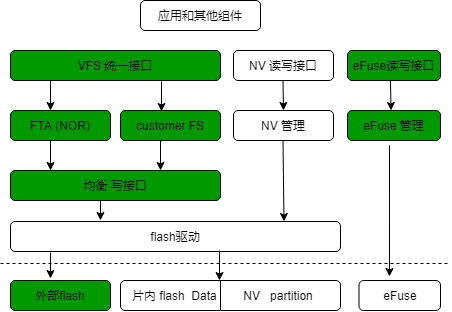


## 存储管理

存储的总体原则：文件系统适合存储较大数据量的二进制数据，NV适合存储较小数据量的参数类型的数据。考虑到当前遥控器项目存储数据不多，文件系统部分作为低优先级实现。

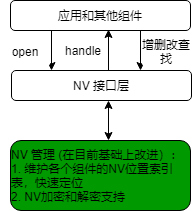
1. 应用层统一调用VFS接口，平台提供的FAT或者客户希望使用的FS可以挂接到VFS
2. 为了避免过度使用norflash的某一个扇区，文件系统调用norflash均衡写接口，确保用作文件存储区域的norflash磨损均衡
3. NV 数据保存在NV partition，通过NV管理模块来进行索引管理
4. MS BT SOC芯片提供 1kbit efuse区域，用于芯片信息（chip id, 电压调整），加密 (flash encrypt, secure boot)，用户应用（mac address etc)。SDK平台提供eFuse管理模块，方便用户使用。 这部分作为低优先级实现。

存储管理框架如下图所示：



Nv的实现方案：

1. 用应用和不同组件的NV handler做空间划分
2. 对操作频繁的模块，维护NV tag索引表
3. 考虑支持NV加密

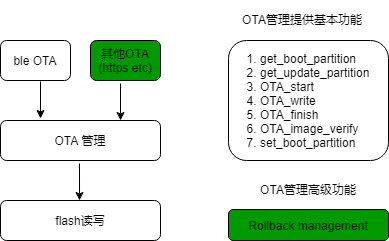


## OTA升级

为了持续提供新功能和修补漏洞，MS SDK平台提供OTA升级功能。OTA管理的框架和主要功能如图所示。

OTA包括：OTA管理，ble，https等升级渠道对核心OTA管理的封装。核心的OTA升级功能，包括获取当前的boot 分区，OTA升级过程管理，image 校验，设置当前的boot分区。OTA管理还涉及到一些高级功能，例如防止回刷功能，升级之后的回退功能等作为低优先级实现。

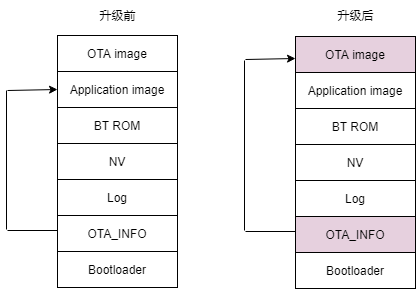
ble OTA在OTA管理模块的基础上做一次封装，在ble OTA service中调用。通过ble OTA service下面的 characteristic完成firmware版本，OTA patch版本，OTA patch size等信息的交互以及 OTA data的收发。收到的OTA数据交给核心OTA管理层处理。 类似的， 在核心功能支持下，后续可以支持其他途径的OTA升级例如 https。



OTA升级流程如图所示：

为了支持OTA，系统的分区必须包含应用image，OTA image和 OTA\_INFO分区。Bootloader在加载正确的应用之前，到OTA\_INFO区域获取 Application和OTA两个分区当前版本的序列号，以及上一次升级完成的状态，通过这些信息跳到正确的image执行。

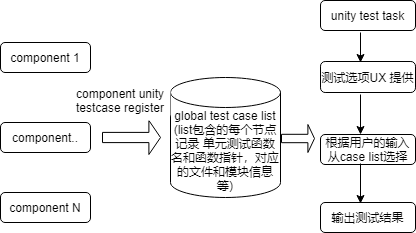
OTA升级只对应用image有效。当前蓝牙image和应用 image是分开编译，因此不会升级蓝牙image。如果要升级蓝牙image，需要和应用放在一起编译。用户在配置分区的时候也需要适当的修改（去掉蓝牙ROM分区，扩大应用和OTA分区）。



## 自动化测试

MS SDK提供单元测试框架，满足客户二次软件开发的质量诉求。通过高效的自动化测试来快速发现新引入的问题，确保平台的基本功能正确。单元测试框架基于unity。主要原因是 unity支持跨平台，遵从C标准，提供了丰富的断言，兼容多种嵌入式C编译器的支持。

单元测试系统框图如下图所示：



所有的component（包括platform, freertos, BT..)都在component目录下包含test测试目录，里面有单元测试的代码。所有对外提供的接口需要有单元测试覆盖。

Tools 目录下有单元测试工具。这里是所有单元测试的入口app\_main （调用到test main)。跑起来之后会 提供测试选项，按照输入的测试选择选择单元测试用例。

典型测试代码如下：

#include "unity.h"

#include "file\_to\_test.h"

void setUp(void) {

// 在这里配置东西

}

void tearDown(void) {

// 在这里清理东西

}

void test\_function\_should\_doBlahAndBlah(void) {

//单元测试内容

}

void test\_function\_should\_doAlsoDoBlah(void) {

//单元测试内容

}

int main(void) {

UNITY\_BEGIN();

RUN\_TEST(test\_function\_should\_doBlahAndBlah);

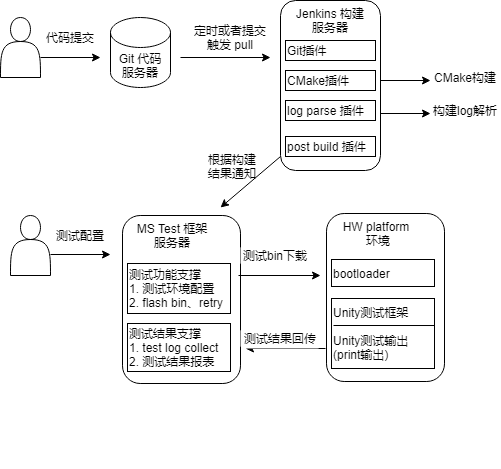
RUN\_TEST(test\_function\_should\_doAlsoDoBlah);

return UNITY\_END();

}

CI持续（定时触发或者代码提交触发）从Git仓库获取最新代码，自动触发构建，自动下载到开发板，完成单元测试之后，发出测试报告。在每天上班的时候可以拿到当前最新代码的测试报告。

CI整体的框架如下图，需要搭建Jenkins构建服务器，MS测试框架服务器。



流程说明：

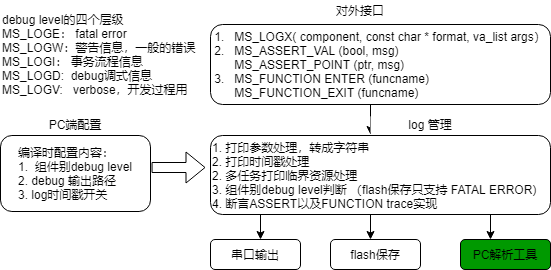
1. 通过Jenkins的Git插件，定时触发代码下载。例如设置为凌晨1点的定时触发。
2. Jenkins 使用CMake插件进行构建
3. 通过log parse插件，在Jenkin console上看到编译过程和编译结果
4. 在编译成功（看log parse结果)之后，post build插件通知测试服务器，并传送test bin
5. 测试服务器对硬件板子进行复位（通过和bootloader的通信），下 test bin
6. 执行test bin，执行unity测试框架的断言。 unity通过串口回传每个测试用例的执行结果
7. 测试服务器收集测试结果并生成当天的测试报表

## 可维可测

高效的开发调式工具对软件开发效率至关重要。MS SDK平台提供log/断言，内存检查，RAM DUMP等手段帮助软件开发人员快速查找问题。

### Log和断言的支持

* 提供通用的 log打印函数 MS\_LOGx，该函数支持按照组件别设置的五级debug level打印。 其中 fatal error等级支持把调式信息写到flash。在release编译版本，fatal error将会引起SW reset 同时将重要信息写入flash。
* 提供FUNCTINO TRACE，支持输出当前所在的文件名和函数名，等级相当于普通调式信息。
* 提供断言，对指针是否为空和数据比较结果是否为真的断言，等级相当于fatal error。
* 提供PC解析工具，用于对时间敏感的组件。设置debug输出到PC解析工具之后，log format的处理放到PC端处理 (低优先级实现)



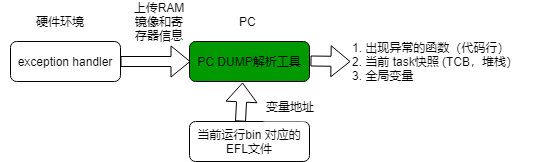
### 内存检查

内存损坏是软件开发中出现的比较普遍但是又比较难查找的问题。SDK平台提供统一的内存分配和释放的接口，作为低优先级支持 。SDK平台提供统一的内存管理支持如下功能：

* 内存泄漏检查，可以具体到函数级（调用malloc的函数）
* 内存越界检查，可以检查到使用的内存超过了分配的大小

### RAM DUMP

为了快速查找一些不易复现的问题，平台支持RAM DUMP的解析。在编译配置阶段如果不打开DUMP支持，异常处理函数只是向串口打印当前的寄存器（PC指针，函数参数等寄存器）。如果打开DUMP支持（作为低优先级支持)，可以通过PC DUMP支持上传当前的RAM镜像，结合ELF文件给出更多的信息。



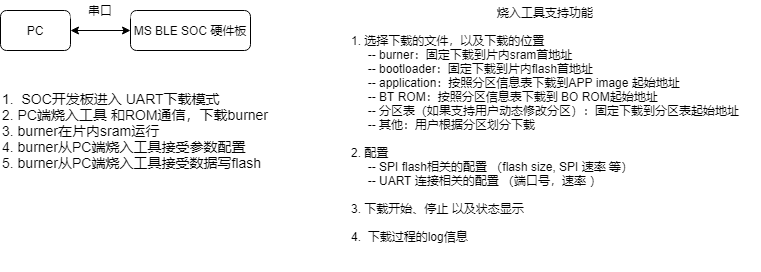
## 生产自动化

生产支持工具主要包括三部分：

* PC烧入工具；
* 用于硬件自检和校准的PC测试工具；
* 以及和测试工具配套的设备端的工程模式。

### 软件镜像烧录

烧入工具的流程和功能如下所示。产线用的和开发用的烧入工具用同一个PC程序。如果配置为产线用的话，增加一拖多的烧入功能。为了保证ROM的逻辑足够简单，烧入过程中，PC端的参数配置解析和实际烧入由burner完成。



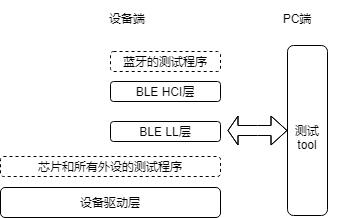
### 产测自动化

在生产过程中需要工程测试，以确保芯片，硬件以及核心基本功能是正常工作的。

通过固定GPIO的状态判断，在开机的时候直接进入工程测试模式。工程模式下，硬件板上的工程测试软件和PC端测试软件通信，完成并显示测试的结果。对芯片和外设的测试包括如下：

* 芯片的信息读取 （ eFuse 相关的操作正常）
* Flash，sram的读写
* RF 射频 （BT RF，需要PC 测试tool协助）
* 其他需要支持的设备

在HCI层需要有蓝牙快速配对等蓝牙基本功能测试

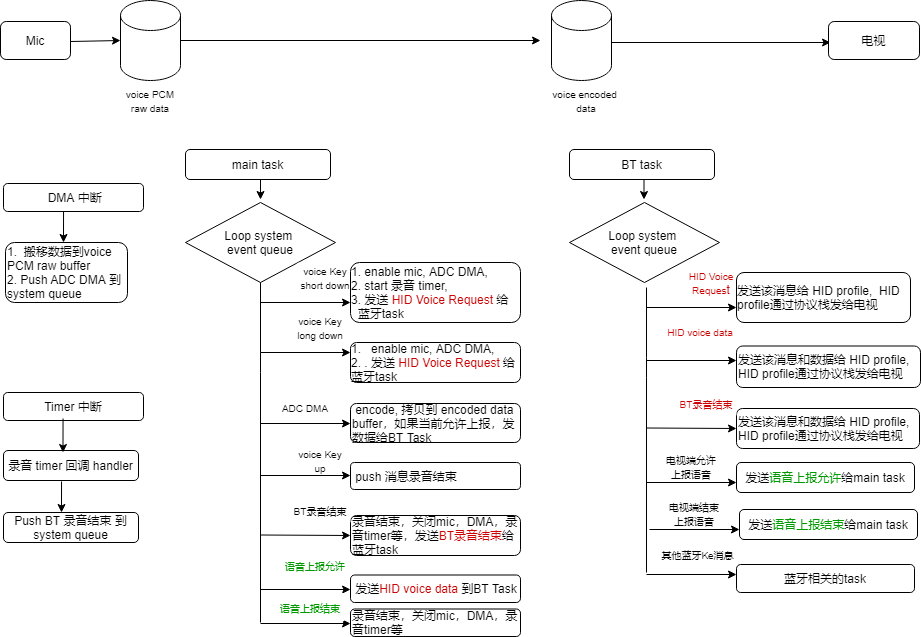


## 典型场景应用—蓝牙遥控器语音发送

场景说明：支持短按/长按 录音键 发起语音发送。短按情况下超时停止录音，长按情况下 按键抬起停止录音。收到电视端允许录音消息后开始录音，收到电视端停止录音消息时停止录音。

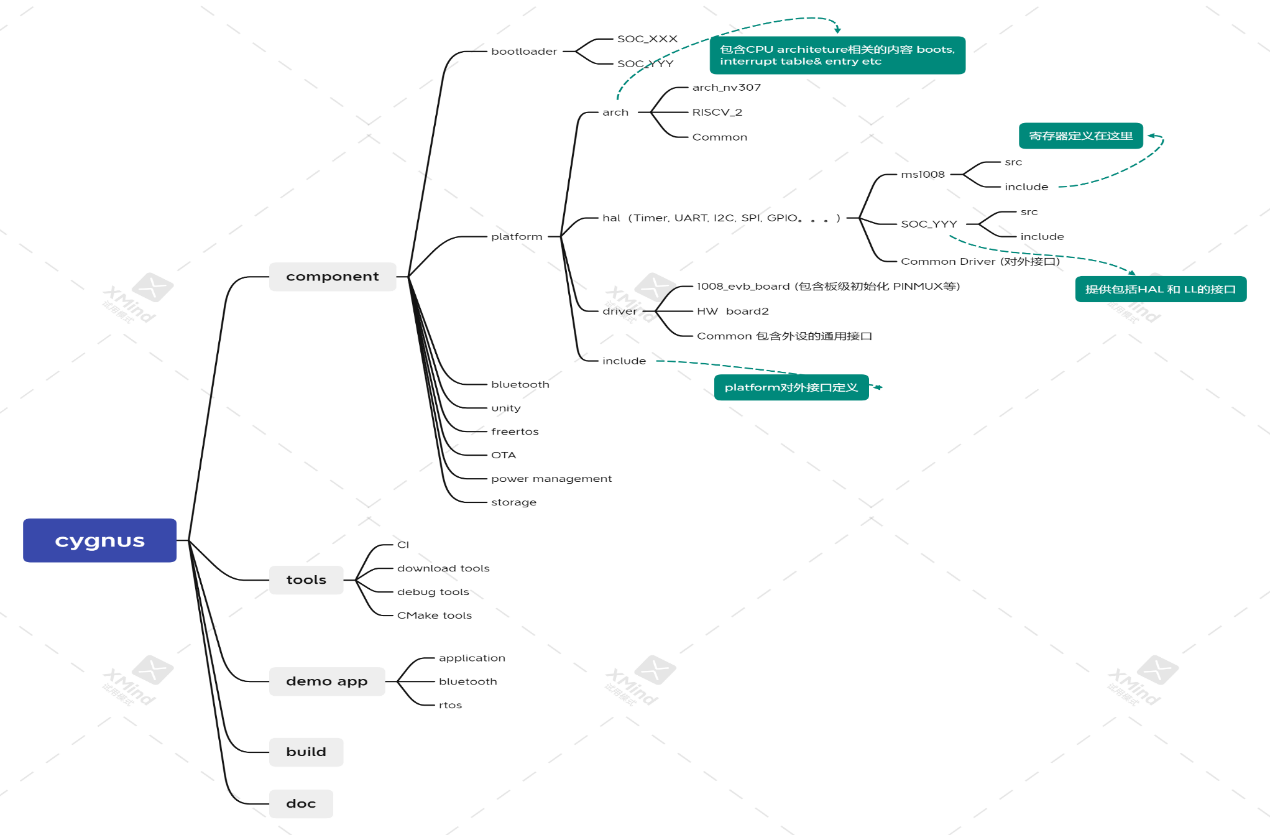
考虑到目前的编码不太复杂，PCM raw data到 SBC encoded code可以在 main task做。图中红色的消息是 main task发给BT task，绿色的消息是BT task发给main task 。

按照之前TCL蓝牙遥控器的设计，蓝牙键盘和语音传输是用HID 服务的 report characteristic进行。这些Report Characteristic的UUID都是0x2A4D，通过对应描述符的Report ID和 Report Type区分语音数据通道、控制通道、蓝牙键值通道。



## 目录树

扁平化目录设置，所有的软件包都在component目录下。典型的客户项目可以参考 app目录的内容，这些 app是用平台软件component 开发的演示功能。 在编译阶段Component会编译成独立的静态库文件。Doc包含Doxygen自动化生成的以及必要开放给客户的设计文档 (application note etc) 。



## 版本获取

### Windows版本获取

版本使用git管理。需要做如下配置：

1. 安装Git Bash；
2. 运行Git Bash，配置添加自己的邮箱和姓名

git config --global user.email “[you@mooresilicon.com.cn](mailto:you@mooresilicon.com.cn)”

git config --global user.name “your name”

1. 生成RSA密钥对

ssh-keygen.exe -o

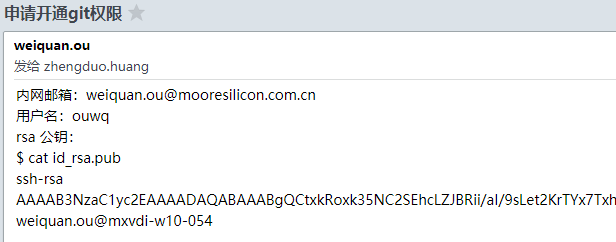
~/.ssh目录下生成两个RSA密钥对文件



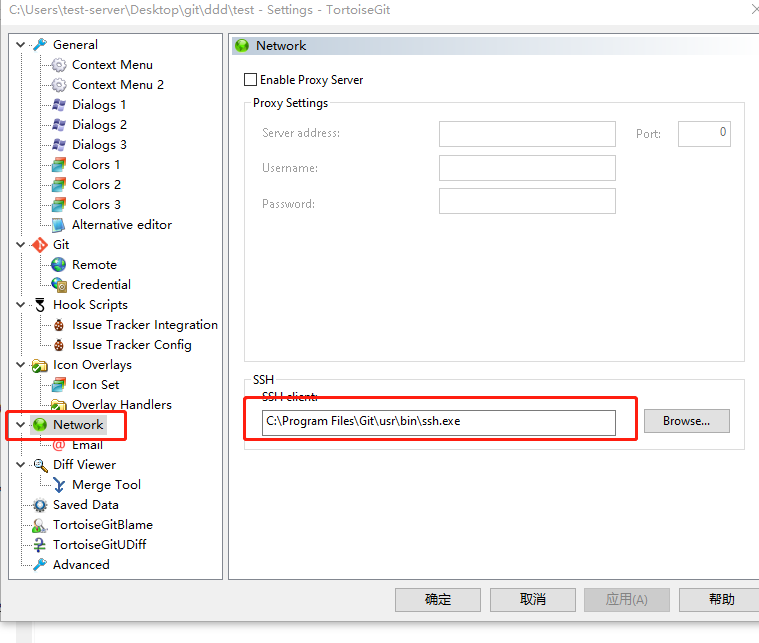
id\_rsa 私钥

id\_rsa.pub 公钥

1. 发邮件给软件总监黄工申请git权限，附上邮箱/用户名/RSA公钥



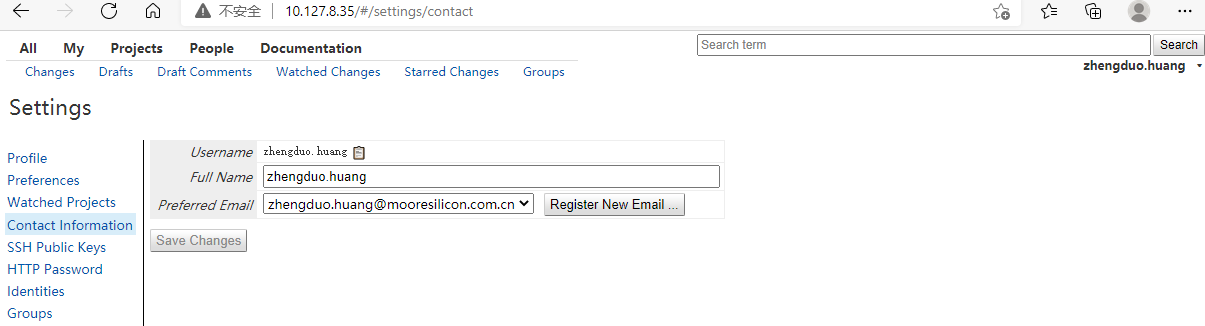
1. 安装TortoiseGit，设置network，必须设置ssh.exe(git安装路径)

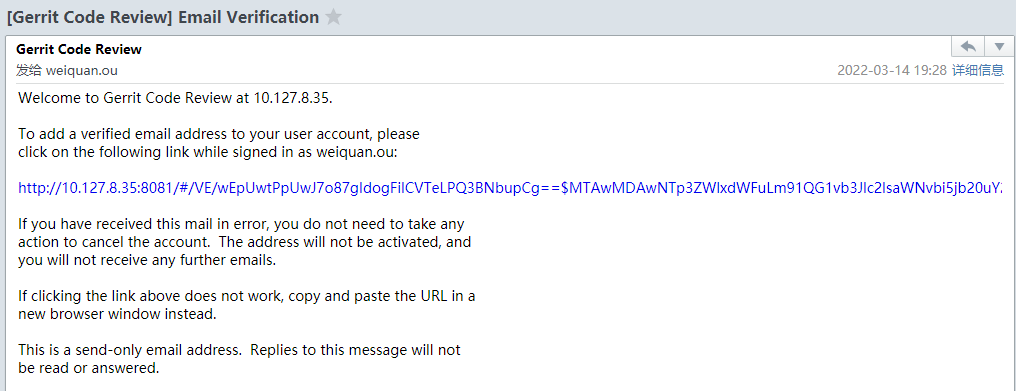
[](http://10.127.8.32/lib/exe/fetch.php?media=newbie:2.png)

1. TortoiseGit clone获取版本

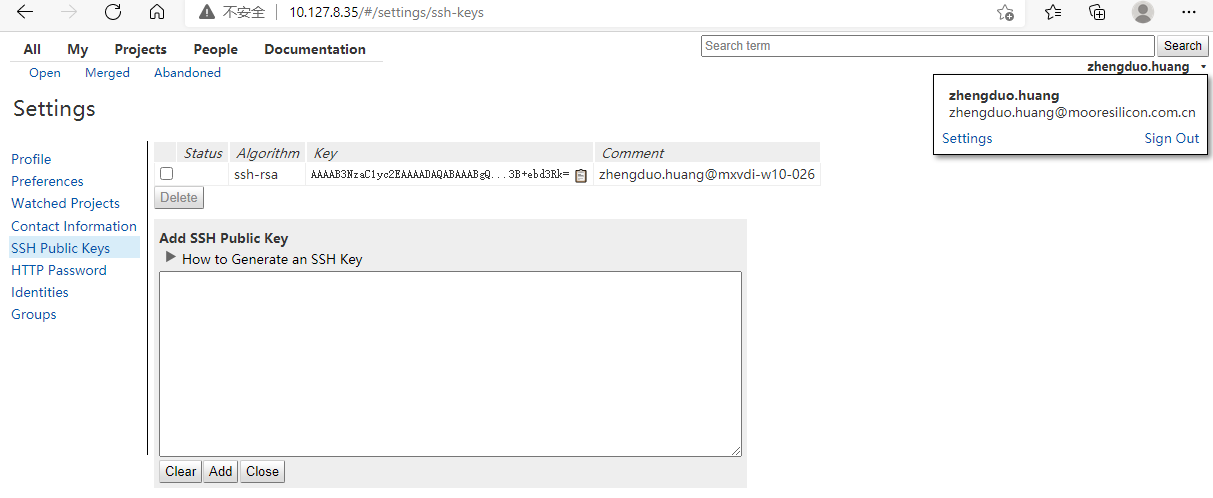
版本路径ssh://git@10.127.8.16/remote/depot/git/cygnus

1. 配置Gerrit
   1. 登录Gerrit服务器，用户名和密码，需向软件总监黄工申请
   2. 点击“Contact Information”进入Email Register页面，输入自己的邮箱账户（此邮箱需要与自己的Git配置一致）。Gerrit服务器会向你的邮箱发送激活邮件。点击激活。





* 1. 配置公钥，页面“SSH Public Key” 设置git 公钥



### Linux版本获取

1. 通过ssh [用户名@10.127.8.38](mailto:用户名@10.127.8.38)登录，初始密码123456

登录后可通过”passwd 用户名”修改密码

注：如果没有Linux帐号请向软件总监黄工申请



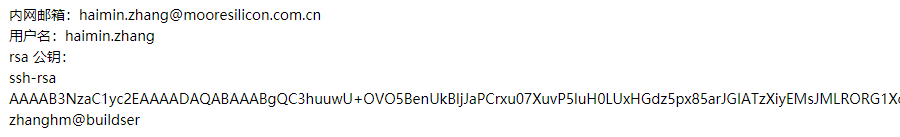
1. git配置添加自己的邮箱和姓名，生成RSA密钥对

git config --global user.email “[you@mooresilicon.com.cn](mailto:you@mooresilicon.com.cn)”

git config --global user.name “your name”

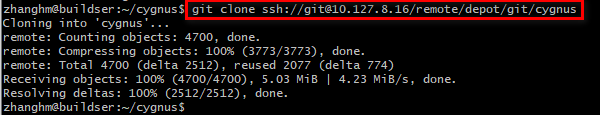
ssh-keygen -o

1. 发邮件给软件总监黄工申请git权限，附上邮箱/用户名/RSA公钥



1. 源码下载

git clone ssh://git@10.127.8.16/remote/depot/git/cygnus



## 配置编译

SDK平台支持Windows操作系统的命令行编译。基于cygwin环境，通过CMake生成makefile，再用make编译。

### 支持三套程序的编译

* 芯片测试程序
* SDK测试程序

为了两套程序可复用hal、register 以及部分公共组件。在SDK里增加芯片测试程序的目录。下图红色部分编译的是SKD测试程序。绿色部分编译的是芯片测试程序。

[](http://10.127.8.32/lib/exe/detail.php?id=system%3A%E5%A6%82%E4%BD%95%E7%94%A8cmake%E7%BC%96%E8%AF%91cygnus_sdk%E5%B9%B3%E5%8F%B0&media=system:%E7%BC%96%E8%AF%91%E8%AF%B4%E6%98%8E_sdk%E6%B5%8B%E8%AF%95.png)

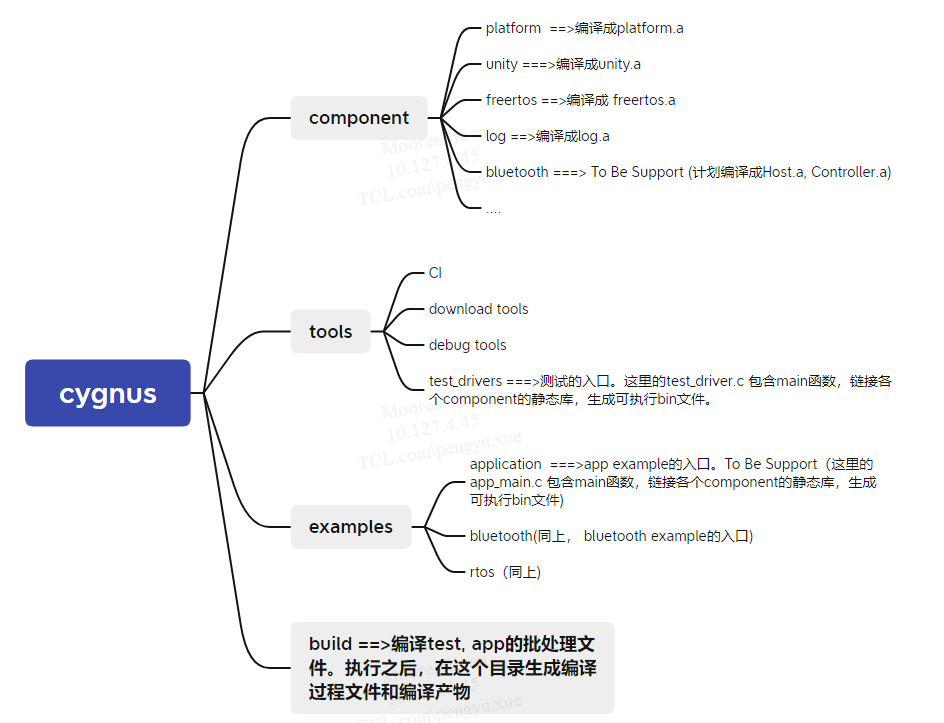
[](http://10.127.8.32/lib/exe/detail.php?id=system%3A%E5%A6%82%E4%BD%95%E7%94%A8cmake%E7%BC%96%E8%AF%91cygnus_sdk%E5%B9%B3%E5%8F%B0&media=system:%E7%BC%96%E8%AF%91%E8%AF%B4%E6%98%8E_%E8%8A%AF%E7%89%87%E6%B5%8B%E8%AF%95.png)

为了便于芯片测试程序的开发，hal根目录（即 SDK平台驱动的逻辑部分）也包含到芯片验证程序。芯片测试的逻辑不同于 SDK平台驱动的逻辑，可以复用的主要是hal封装和寄存器操作部分。但是考虑到SDK平台驱动可能会对芯片测试有用，暂都包含进来。

### 编译方案

* 各个组件编译生成静态库文件。
* 对cygnus SDK平台 tools/test\_drivers 目录下面的测试程序 作为程序的入口，链接需要的库文件，生成可执行文件。以后在example下面的 各个范例代码也会按照这个方式进行编译。
* 对芯片测试程序，components/plantform/hal/ms1008verify 目录下面的测试程序 作为程序的入口。该目录以后不会发布给客户。

SDK测试平台为例，以下是编译方案（芯片测试程序 只链接 log和unity库）：

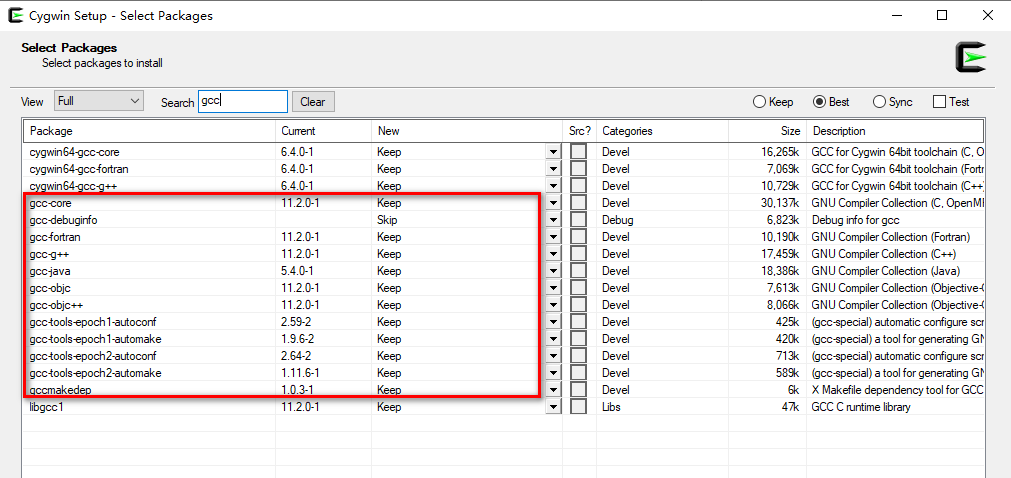
[](http://10.127.8.32/lib/exe/fetch.php?cache=&media=system:%E7%BC%96%E8%AF%91%E6%95%B4%E4%BD%93%E6%80%9D%E8%B7%AFupdate.png)

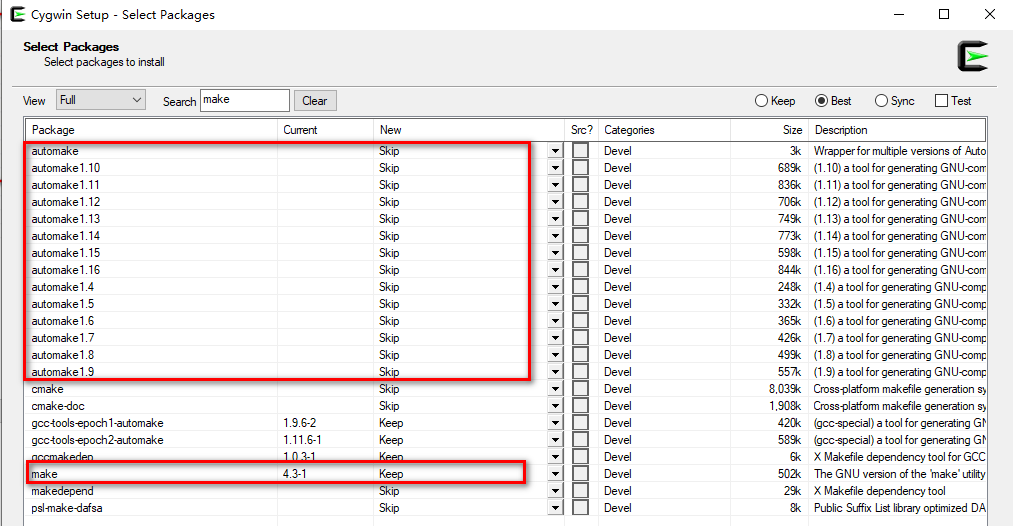
### Windows编译环境安装

1. Windows安装CMake
2. Windows安装Cygwin

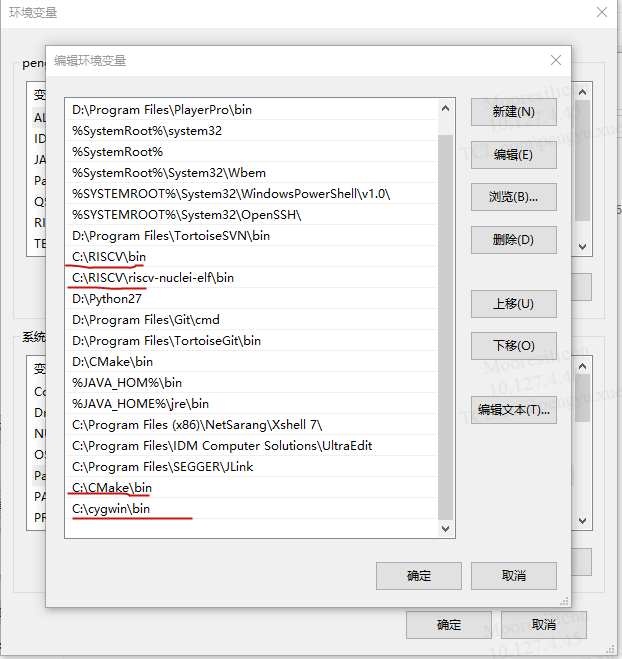
**Cygwin安装cygwin注意事项：**

1. **选择从本地安装，安装default package。**
2. **在default基础上加上gcc 和 make**
3. **不要选择安装automake，否则编译时会出错**





1. Windows环境变量里面增加CMake，cygwin，以及RISCV交叉编译器所在的目录

[](http://10.127.8.32/lib/exe/fetch.php?cache=&media=system:%E7%8E%AF%E5%A2%83%E5%8F%98%E9%87%8F%E8%AE%BE%E7%BD%AE.png)

### 编译执行

进入build目录，运行 build\_drivertest.bat 即可编译。注意：

1. 修改，增加，删除 源文件的时候，不需要修改CMake脚本，重新跑一下build\_drivertest.bat 即可。
2. 某个组件，比如plaform组件，有可能需要增加新的 include文件路径。 这种情况下需要修改该组件的CMake脚本, 在include\_directories下面添加。
3. 当前各个组件在编译成库的时候，会扫描该目录下所有的.c .cpp .S 文件。因此如果是不需要编译进来的冗余文件，请不要放在component目录，或者通过宏定义来选择编译。

### 支持芯片和板子的类型

SDK平台的驱动设计 和 platform 目录结构 已经考虑到 CPU Core (arch)，ble芯片 (SOC)，开发板 (Driver) 这三个层级的更换。

CPU core目前都是 riscv300系列，无需配置。 板子： FPGA evb板(FPGA) 和 芯片evb板(1008) 和遥控器产品板( To Be Supported)。芯片板和遥控器板要等7月份芯片流片回来才能拿到。

SOC芯片： 已经流片 tapeout的芯片 (MS1008) 以及 目前正在开发的芯片(MS1008\_V2)。

不同板子之间差异的代码放在 driver目录下的 board子目录。

不同SOC芯片之间差异的代码放在 hal 目录下的 soc子目录。

在运行编译批处理的时候，需要输入板子名字和芯片名字，根据板子和芯片 取选择不同的目录编译。 如：

$ ./build\_drivertest.bat FPGA MS1008

如果没有输入参数，会提示错误。

$ ./build\_drivertest.bat

please select board name FPGA or 1008

please select soc name MS1008 or MS1008\_V2

build error, need board and soc name

build end

### 目标文件

编译成功后，输出的目标文件在build目录

test\_driver\_sdk.bin 文件大小54k左右，软件程序，烧录在Flash运行。

注：如果在Linux编译服务器上编译，可通过以下命令将目标文件下载到Windows主机上

scp [用户名@10.127.8.38:/data/](mailto:用户名@10.127.8.38:/data/)用户名/cygnus/cygnus/build/\*.bin ./



## 开发调试

当前基于FPGA平台调试。待回片后增加DEMO板调试说明。

### FPGA开发调试

#### 环境准备

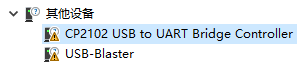
使用ALTERA FPGA开发板。使用JTAG接口加载FPGA文件和软件版本。使用串口调试。

##### 软件环境

1. 安装Quartus II 14.1

用于FPGA bitfile加载。

注意：软件自带安装USB设备驱动程序。如果出现设备驱动异常，请联系系统管理员开通瘦客户端权限。



1. 安装 J-Link和J-Flash

J-Link用于bootrom加载

J-Flash用于软甲版本在Flash加载。需要进行如下配置：

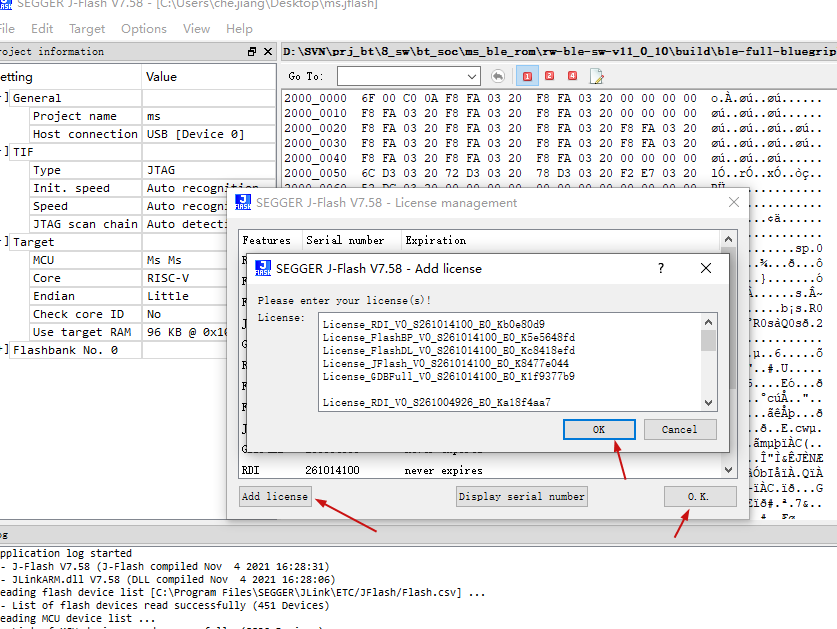
1. 添加licence

点击help->Licenses进入licenses配置。Licenses文件在：

svn://10.127.8.17/prj\_bt/8\_sw/bt\_soc/ms\_ble\_tools/flashloader\_RV32\_1126/J\_License.txt

或者wiki: http://10.127.8.32/doku.php?id=newbie:jlink





1. 替换JLinkDevices.xml

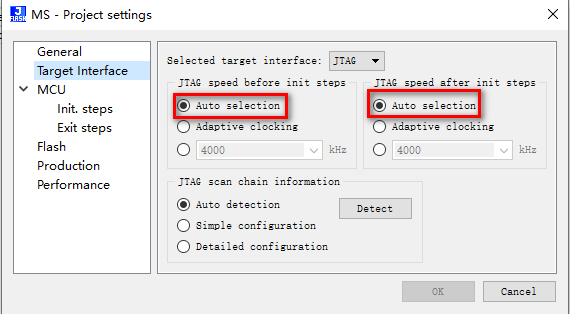
把JLinkDevices.xml替换到Jlink的安装目录，如C:\Program Files\SEGGER\Jlink

1. 创建elf文件

* 在C:\Program Files\SEGGER\JLink\Devices\目录下创建MS目录
* 把Flashloader.elf和Flashloader\_gd.elf放在C:\Program Files\SEGGER\JLink\Devices\MS目录
* elf路径:

ssh://git@10.127.8.16/remote/depot/git/8\_sw/bt\_soc/ms\_ble\_tools/flashloader\_RV32

* 时钟配置成Auto selection，否则会烧录失败



1. 串口终端工具XCOM V2.3

安装串口驱动CP210x\_Windows\_Drivers

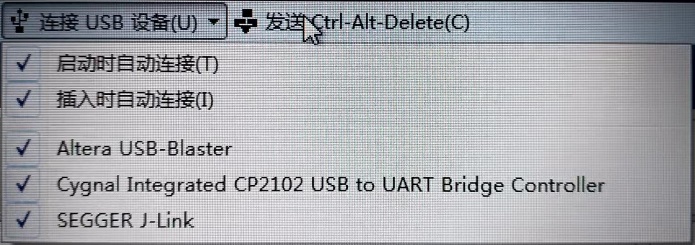
XCOM V2.3是绿色软件，直接运行即可

##### 硬件环境

硬件环境如下图所示：



PC使用内网瘦客户端，通过USB HUB连接JTAG仿真器，和串口转换器。以上设备按硬件原理图与FPGA板做跳线连接。连接正常后，瘦客户端有如下显示：



FPGA板uart和jtag管脚连接关系如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 接口 | FUNC | 芯片PIN管脚 | FPGA板PJ10转接管脚 |
| UART0 | UART0\_TXD | P14 | 17 |
| UART0\_RXD | P15 | 18 |
| UART0\_CTS | P16 | 19 |
| UART0\_RTS | P17 | 20 |
| JTAG | TDI | P12 | 15 |
| TDO | P13 | 16 |
| TCK | P25 | 28 |
| TMS | P26 | 31 |

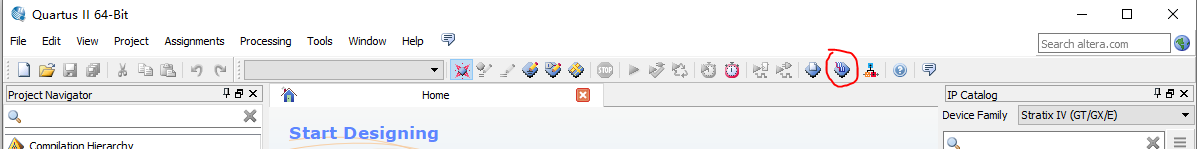
更详细的管脚复用关系，可以参考：

svn://10.127.8.17/DOC/BT\_SOC/arch/spec/MS\_BLE\_arch\_V1.0.xlsx

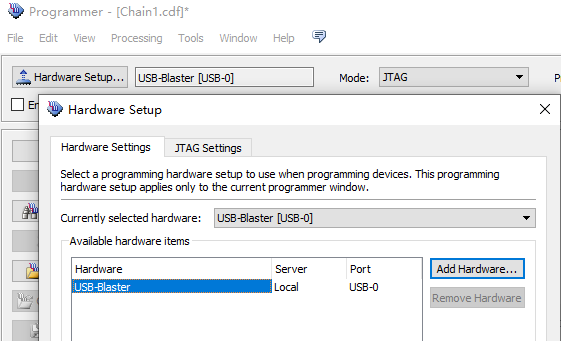
svn://10.127.8.17/DOC/BT\_SOC/fpga/平台资料/TerASIC/TR4/ALTERA引脚分配图.xlsx

#### FPGA bitfile加载

运行Quartus II 14.1 ，选择“tools->programmer”，或者点击快捷图标



点击“Hardware Setup”选择相应的USB设备。

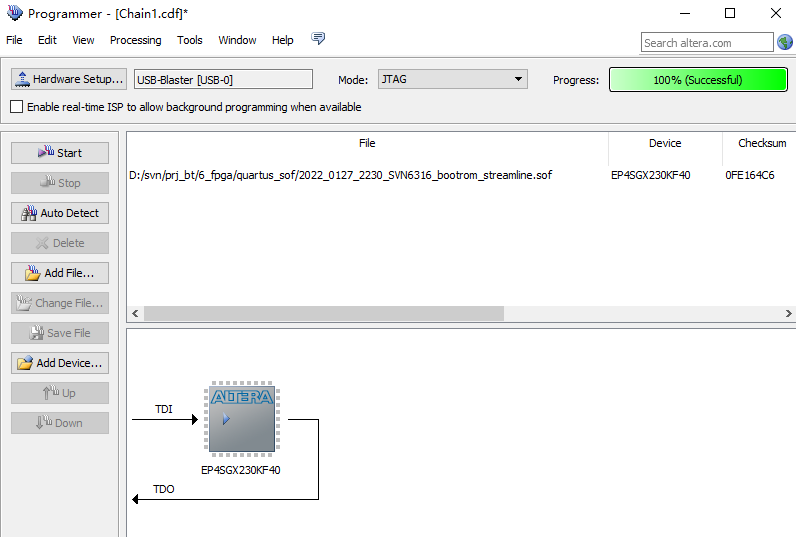


加入bitfile文件，点击“Start”开启加载。待加载完成后，FPGA开始工作，可以被连接。

注：bitfile文件请从svn上下载，路径为：

svn://10.127.8.17/prj\_bt/6\_fpga/quartus\_sof

svn://10.127.8.17/prj\_bt\_to2/6\_fpga/6\_fpga\_bitfile



#### J-Link加载bootrom

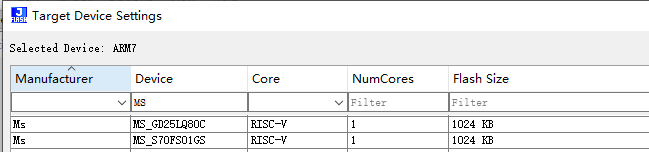
用J-Link把固化程序加载到芯片内ROM，具体步骤如下：

1. 运行J-Link Commander

注意: 请检查电压参数是否正常

1. 命令行输入 “connect”，连接JTAG
2. Device选择MS\_XXX，初次使用可输入“?”，进入列表选择。根据板子的Flash类型选择

注：当用FPGA板用的是MS\_GD25LQ80C



1. 选择JTAG连接，连接速率等参数使用默认值
2. 命令行输入loadbin xxx.bin 0



Jlink连接不上一般会有以下几种原因:

1. Bitfile问题导致，可能会出现以下的几种的现象：

* RV32识别成RV64
* 旧版本正常，Bitfile更换新版本异常(可能会因RTL版本导致)

1. 运行在Flash/RAM程序跑飞导致

* 默认是bootsel是从Flash启动，如果烧录之前连接不上Jlink，可能是Flash残留程序跑飞导致。
* 解决办法:建议切换Bootsel到Bootrom启动，直到烧录完成后切换到Flash启动模式。

Bootsel芯片管脚与FPGA板管脚跳线关系：

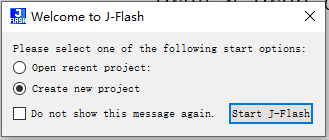
|  |  |  |
| --- | --- | --- |
| FUNC | 芯片PIN管脚 | FPGA板PJ10跳线管脚 |
| Bootsel  拉低：Flash启动  拉高：bootrom启动 | P00 | 1 |

#### J-Flash加载软件版本

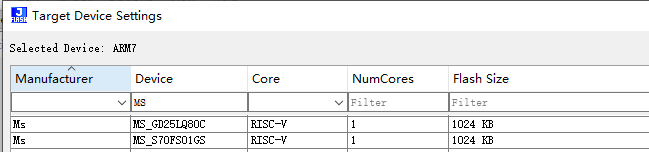
用J-Flash把软件版本加载到Flash，具体步骤如下：

1. 建立工程

打开JFlash后有引导界面。也可以点击File->New Project 。



Target Device选择MS\_XXX，根据板子的Flash型号选择。



1. 点击target->connect连接
2. 烧录：

* 把bin文件拖动到J-Flash
* 点击target->production programming
* 等待烧录完成

1. 测试烧录模式：

* 点击target->test->generate test data可以生成测试数据
* 测试数据烧录进flash

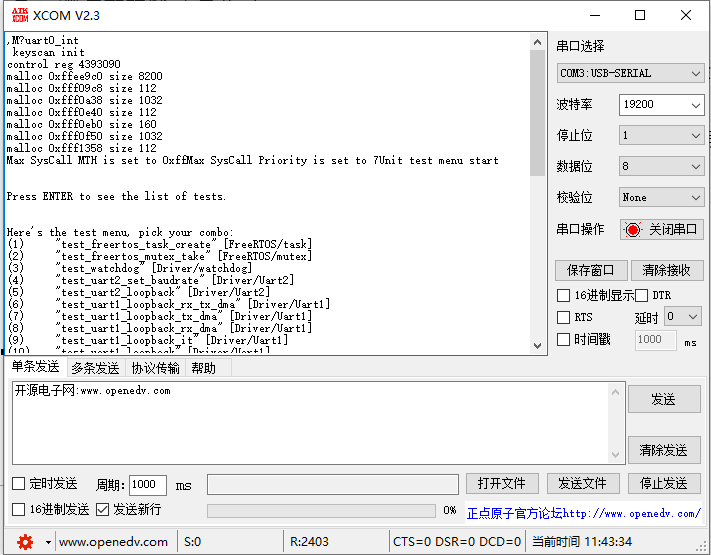
1. 擦除sector

* Target->manual programing->erase sector可以把所有sector擦除一遍

1. 从flash读取

* Target->manual programing->read back->select sector可以把所有sector的数据读回来。

重新Flash启动，可在串口调试。



#### 逻辑分析仪

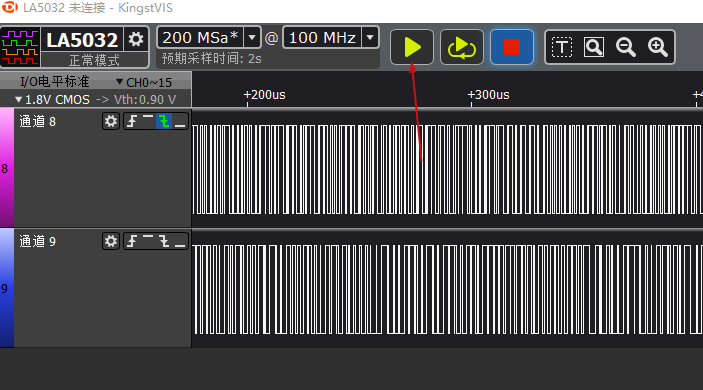
1. 打开Kingst VIS软件；
2. 确保逻分需要cap的信号已经连接；

具体需要采样的信号对应的通道可以和FPGA硬件平台同事进行确认。

1. 选择对应cap信号所使用的通道等配置，如下图的ch8和ch9；



1. 选择红色箭头处进行采样，等待采样完成；

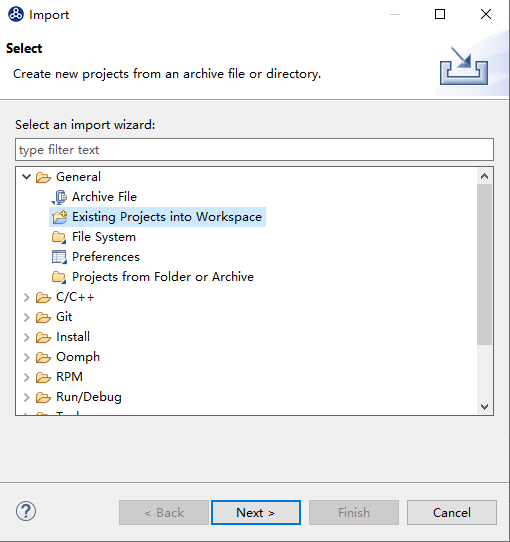
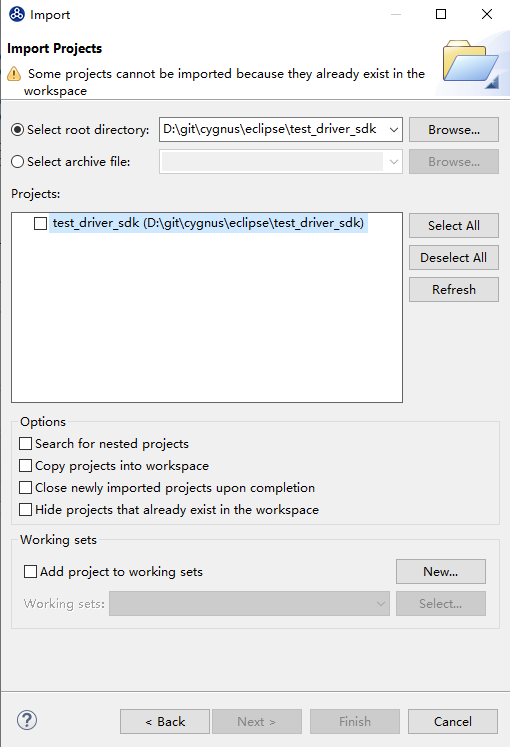
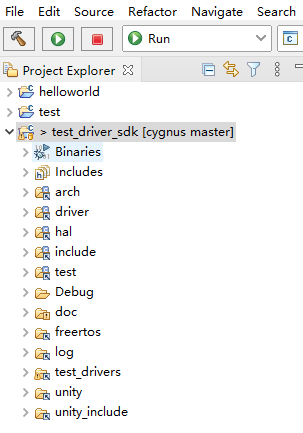


#### IDE开发模式

使用芯来 NucleiStudio，压缩包解压即可运行。

1. 导入SDK工程

SDK的eclipse目录已包含IDE必须的工程文件。在菜单栏中选择“File—>import”， 选择“Existing Project into WorkSpace”后，点击“Next”。输入SDK根目录，IDE识别工程成功。点击“Finish”。SDK工程导入成功。

1. 编译SDK

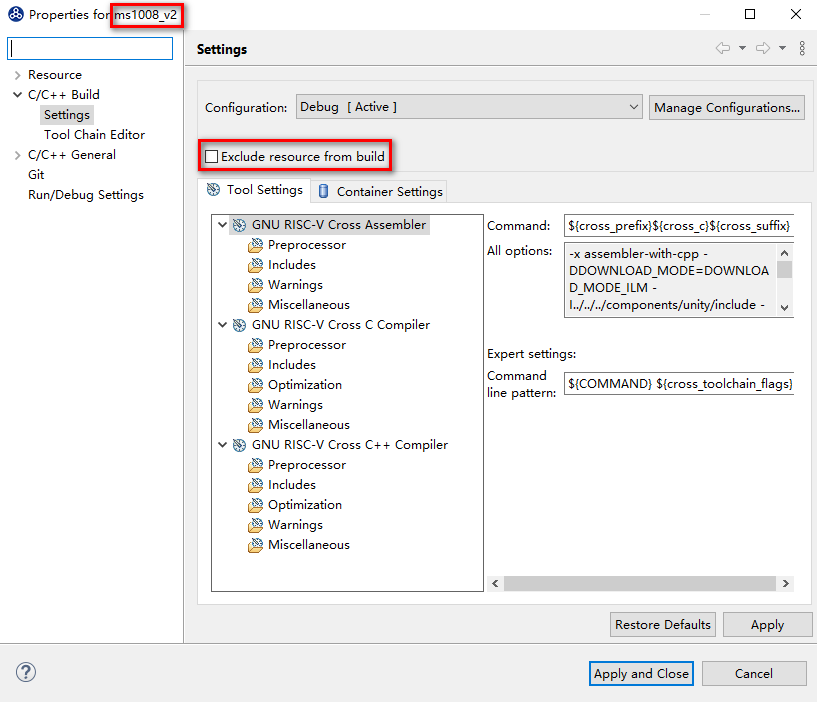
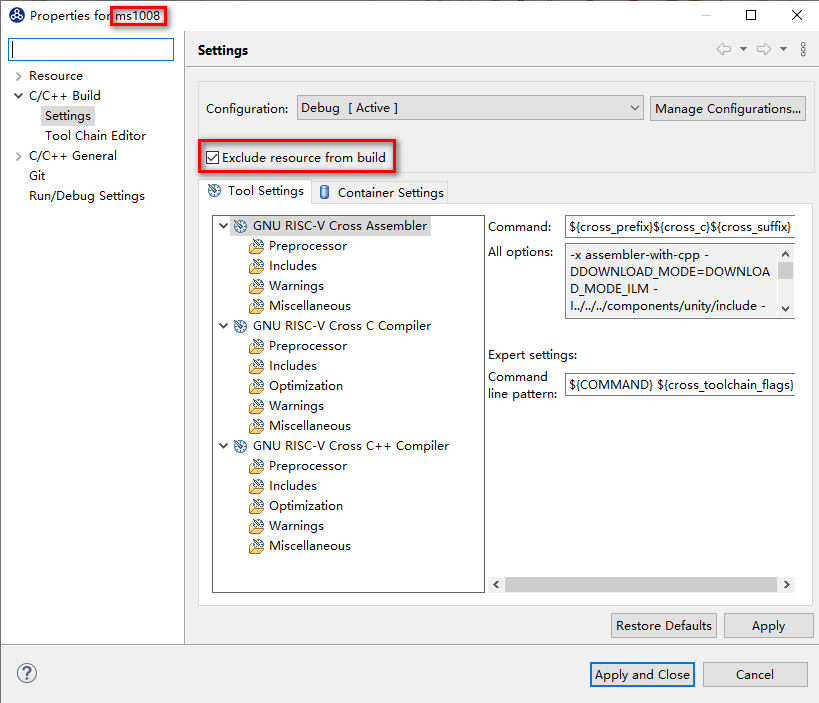
在 Project Explorer 栏中选中SDK项目，单击鼠标右键，选择“Clean Project” 可清除编译。选择“Build Project”进行编译。

**注：如采用MS1008\_V2板进行IDE调试，编译时需要将工程配置成MS1008\_V2之后再进行编译，方法如下：**

1. 配置hal层编译选项；

在Project Explorer中去掉hal/ms1008，添加hal/ms1008\_v2进行编译

方法：右键对应目录->Properties->C/C++ Build->Settings->Exclude resource from buld

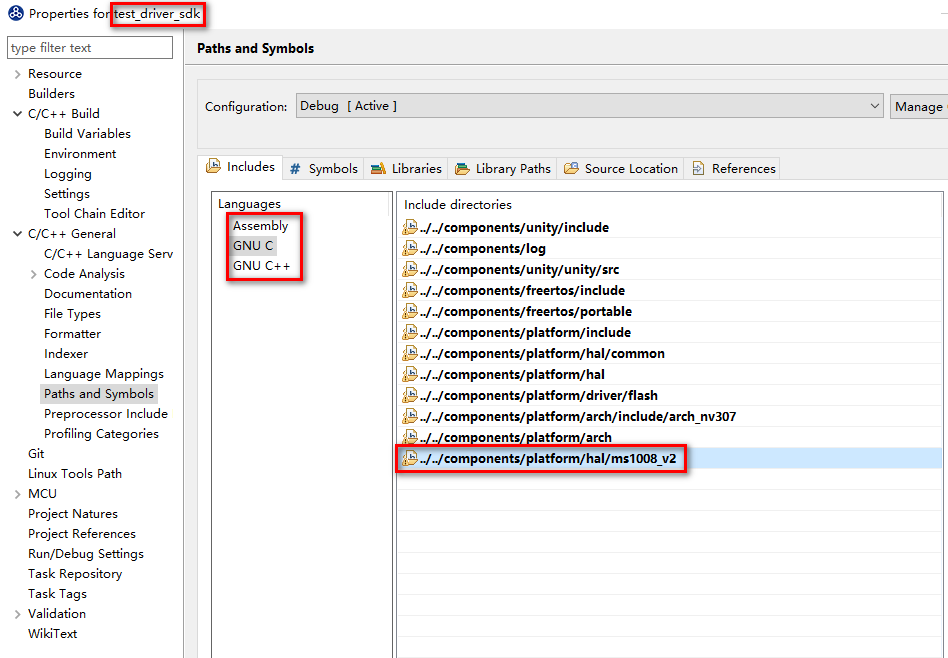
 

1. 配置include路径；

在Project Explorer中将工程的include路径配置为ms1008\_v2

方法：右键对应工程 -> Properties -> C/C++ General -> Paths and Symbols -> Includes

Assembly, GNU C和GNU C++三个选项里都要同时修改

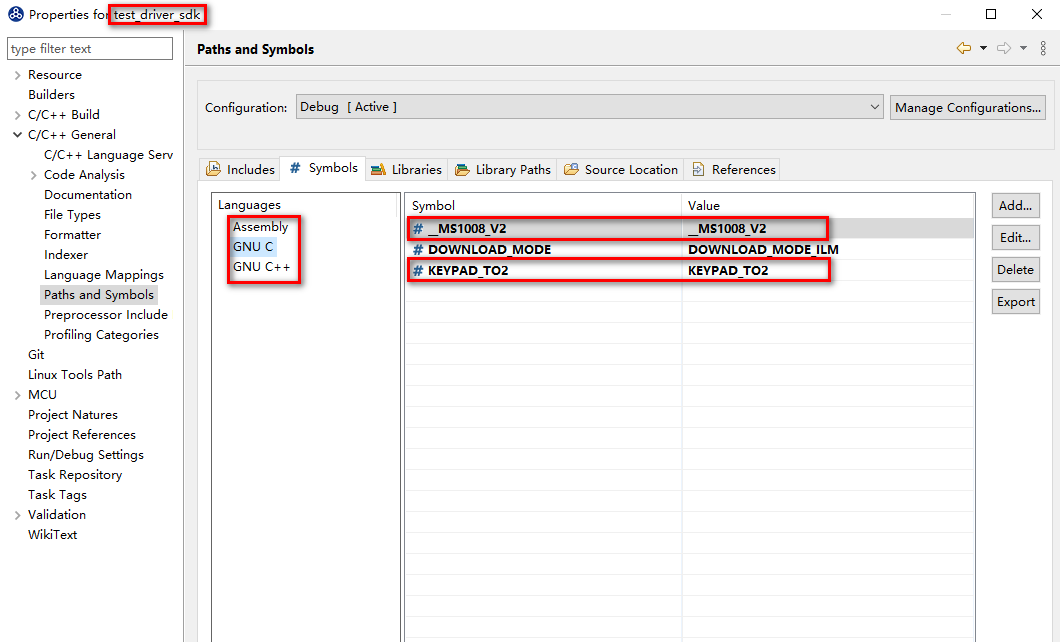


1. 配置编译宏；

在Project Explorer工程中增加MS1008\_V2开发板的编译宏

方法：右键对应工程 -> Properties -> C/C++ General -> Paths and Symbols -> Symbols

Assembly, GNU C和GNU C++三个选项里都要同时修改

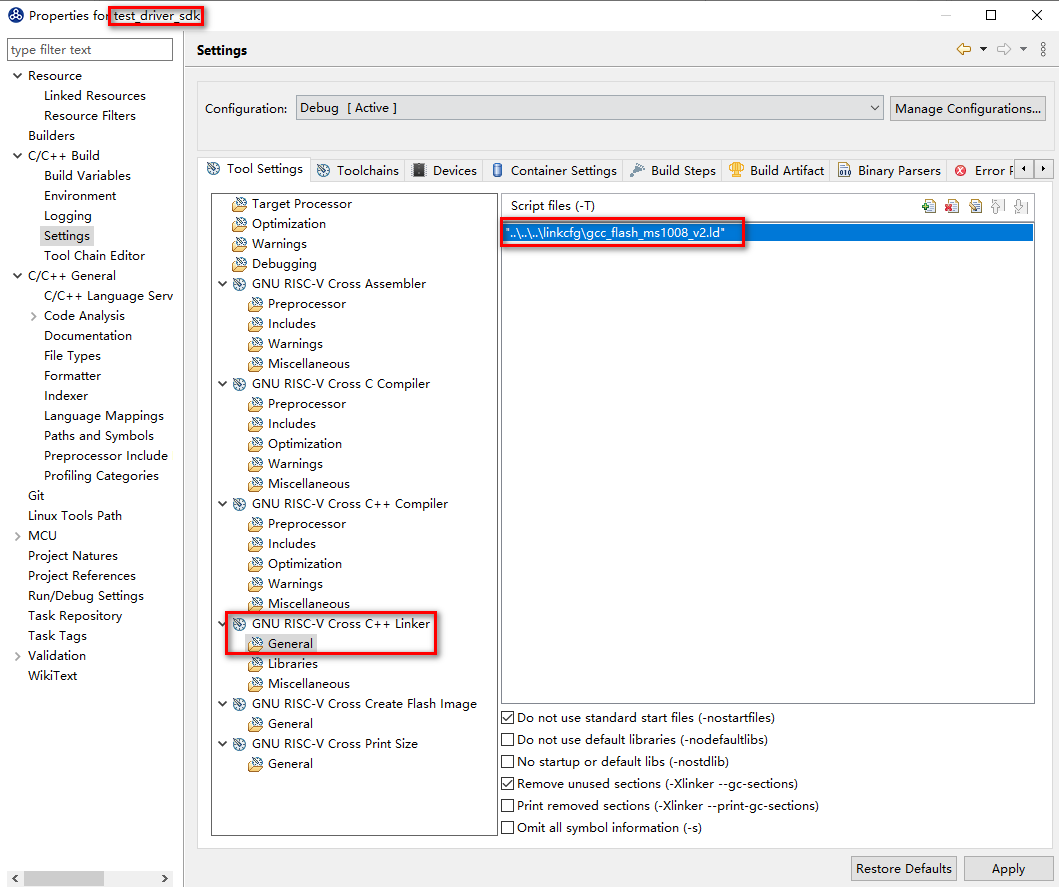


1. 配置link脚本；

在Project Explorer中将工程的link脚本更新为gcc\_flash\_ms1008\_v2.ld

方法：右键对应工程 -> Properties -> C/C++ Build -> Settings -> Tool Settings ->

GNU RISC-V Cross C++ Linker -> General -> Script files

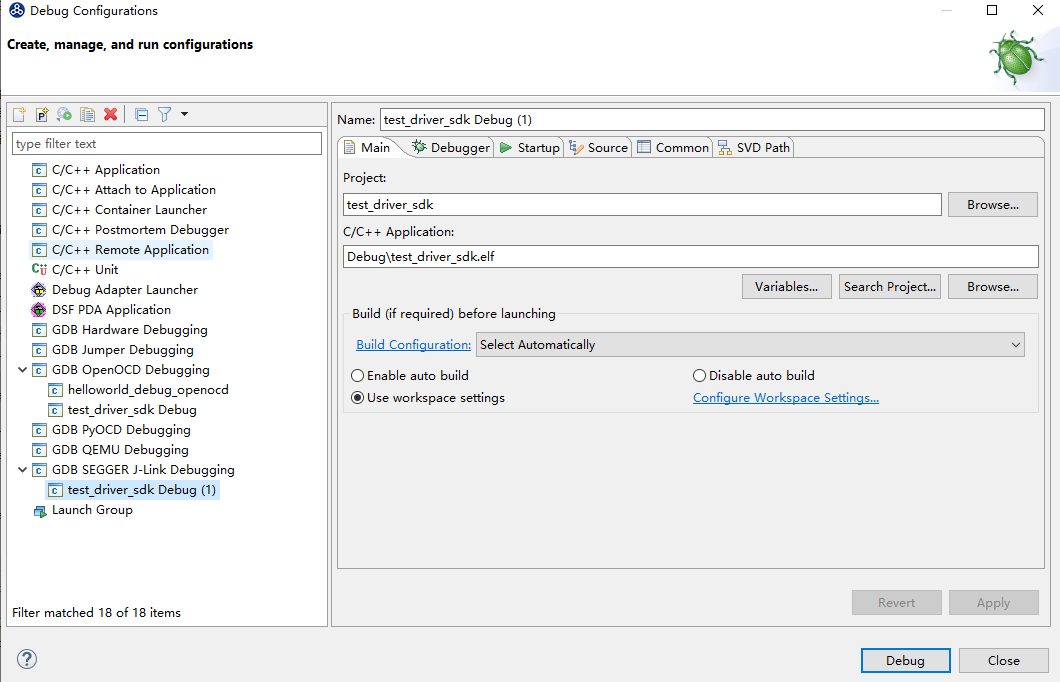


1. SDK Debug

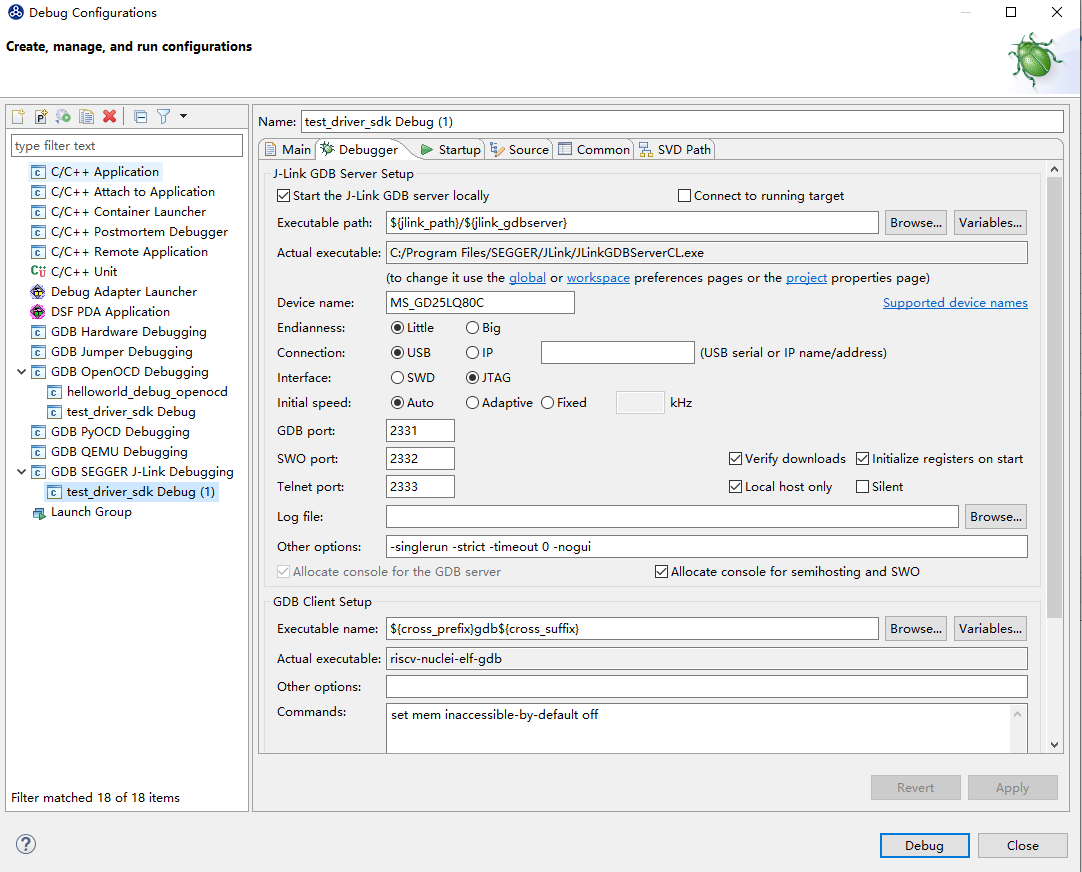
使用J-LINK连接FPGA板Debug。IDE做如下配置：

在菜单栏中选择“Run—>Debug Configurations”

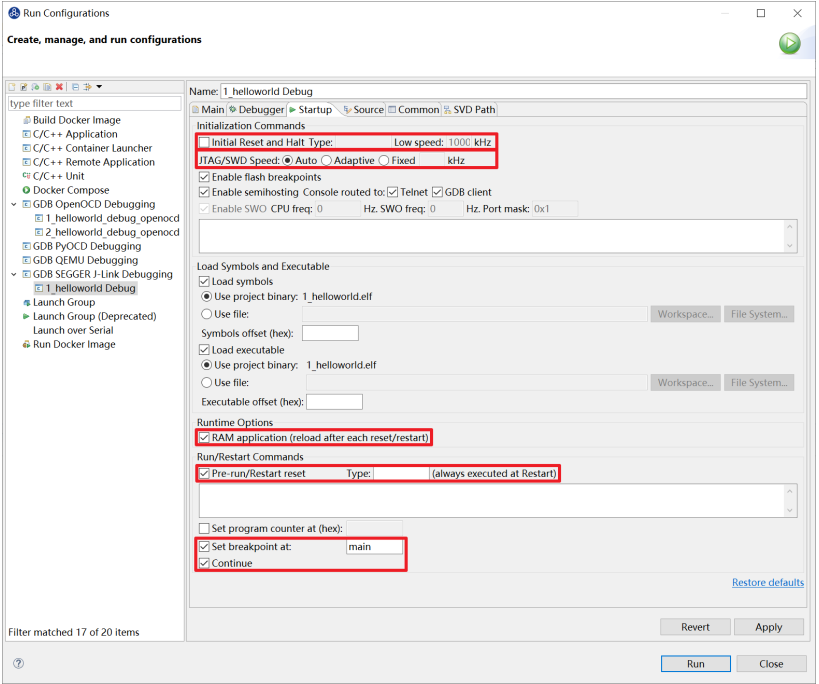
在弹出的窗口中， 如果没有当前工程的调试设置内容， 右键单击“GDB SEGGER J-Link Debugging”，选择“New Configuration”，将会为本项目新建出一个调试项目“test\_driver\_sdk Debug (1)”。 确保“Project”是当前需要调试的工程，“C/C++ Application”中选择了正确的需要调试的 ELF 文件。



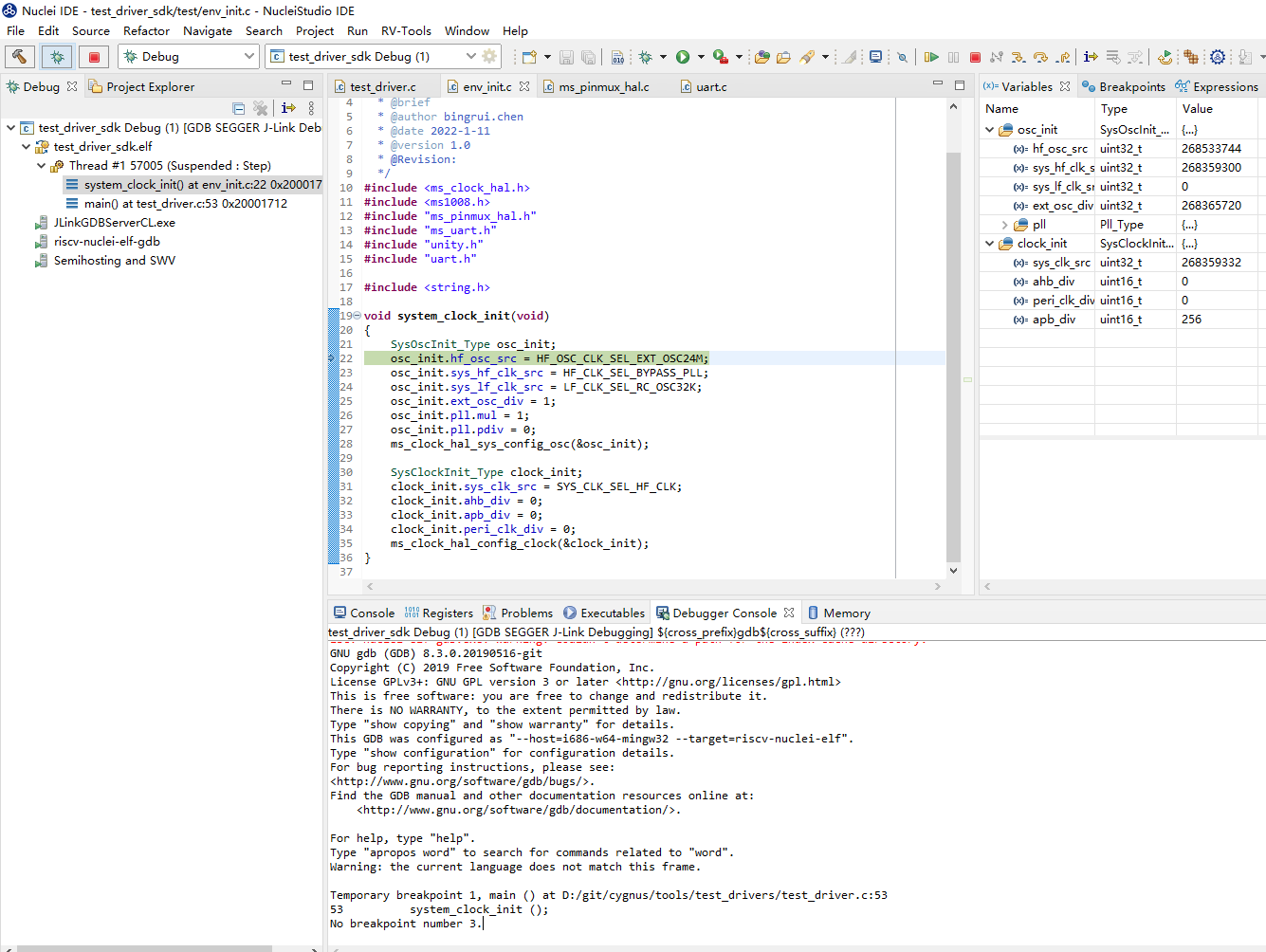
打开 Debugger 栏目，确保 “Start the J-Link GDB server locally”被选中。正确指向 JLinkGDBServerCL.exe 的路径。Device Name是MS\_GD25LQ80C。选择“Interface”为 JTAG，“initial speed”为 Auto。确认与使用的 GDB 设置一致。点击 “Apply”保存。



打开“Startup”栏目，确保 JTAG/SWD Speed 为 Auto， “set Breakpoint at main”，“Continue”， “Pre-run/Restart reset” 和“RAM application” 选项被勾选， 并且取消勾选“Initial Reset and Halt”选项。



下拉框选中 Debug，之后左侧图标会变为甲虫图标，单击即可进入调试模式并下载程序进入开发板中。如果下载成功，并且会启动调试界面。可以使用单步跟踪，设置断点，查看函数变量。在控制台可以输入调试命令。



1. 下载SDK到板子

点击下拉框切换至运行模式，此时左侧图标会切换为绿色运行按键，单击即可将程序下载至开发板并运行。