GR551x GCC用户手册

版本： **1.7**

发布日期： **2020-12-23**

深圳市汇顶科技股份有限公司

商标声明

GOODiX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经GOODIX书面批准，不得将GOODIX的产品用作生命维持系统中的关键组件。在GOODIX知识产权保护下，不得暗中或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田保税区腾飞工业大厦B座2层、13层

电话：+86-755-33338828　　传真：+86-755-33338099

网址：www.goodix.com

前言

编写目的

本文档介绍了在Linux环境下，使用GCC + Makefile的方式构建GR551x的命令行交叉编译开发环境，旨在帮助用户快速进行GR551x SDK应用的二次开发。

读者对象

本文适用于以下读者：

- GR551x用户
- GR551x开发人员
- GR551x测试人员
- 文档工程师

版本说明

本文档为第5次发布，对应的产品系列为GR551x。

修订记录

版本	日期	修订内容
1.0	2019-12-08	首次发布
1.3	2020-03-16	更新了“2.3 安装Python”章节Python版本号
1.5	2020-05-30	增加“2.7 下载程序”章节中GProgram for Linux描述
1.6	2020-06-30	基于SDK刷新版本
1.7	2020-12-23	删除常见问题“链接时发生Conflicting CPU architectures错误”

目录

1 简介

GCC（GNU Compiler Collection）是由GNU开发的一套开源编译器集，也是跨平台软件的编译器。GCC支持Linux操作系统。arm-none-eabi-gcc交叉编译器基于GCC，支持ARM系列CPU的指令集，适合GR551x芯片使用。

在软件开发中，Make是一个自动化构建工具，它根据Makefile文件的内容完成工程源代码文件的自动编译和链接。Makefile定义了使用编译器进行众多工程源文件编译、链接的规则，同时也可以调用执行系统命令。

本文将介绍在Linux发行版Ubuntu开发环境下，使用GCC + Makefile构建GR551x应用开发环境的方法，并给出示例供用户参考。

在进行操作前，建议参考以下文档。

表 1-1 文档参考

名称	描述
GR551x开发者指南	GR551x软硬件介绍、快速使用及资源总览
J-Link用户指南	J-Link使用说明：www.segger.com/downloads/jlink/UM08001_JLink.pdf
Bluetooth Core Spec v5.1	Bluetooth官方标准核心规范5.1： https://www.bluetooth.com/specifications/bluetooth-core-specification/
Bluetooth GATT Spec	Bluetooth Profile和Service的详细信息：www.bluetooth.com/specifications/gatt
GCC	GCC用户手册：https://launchpad.net/gcc-arm-embedded
Makefile	Makefile开发指南：https://www.gnu.org/software/make/manual/make.html

2 构建编译环境

本章节介绍基于Linux（Ubuntu）构建GR551x的交叉编译及开发环境。

构建GR551x的交叉编译及开发环境，需依赖以下工具：

- gcc-arm-none-eabi交叉编译器：用于GR551x可执行目标代码的交叉编译。
- Python：用于GR551x应用工程的脚本执行环境的构建。
- J-Link：用于GR551x固件程序的调试、烧录等。

2.1 准备工作

在构建编译环境之前，用户需要完成如下准备。

表 2-1 软件准备

名称	描述
Ubuntu	推荐使用Ubuntu 16.04及之后的LTS版本（32 bit或64 bit均可）
gcc-arm-none-eabi	下载版本：gcc-arm-none-eabi-5_4-2016q3-linux.tar.bz2 下载地址：https://launchpad.net/gcc-arm-embedded/+download
Python	版本：python 3.x + 下载地址：www.python.org/downloads
J-Link	版本：J-Link Software and Documentation pack for Linux, DEB installer 下载地址：www.segger.com/downloads/jlink/ **说明：** • 选择和用户所用的Ubuntu系统兼容的版本。 • 使用J-Link 6.10a及之后的版本。

说明:

- 建议使用Ubuntu 16.04及之后的LTS版本。由于GCC软件的运行依赖于Ubuntu的环境，建议将本文推荐的GCC版本和Ubuntu版本配合使用。
- 如果选用其他Linux发行版本，可能会产生环境依赖问题。

2.2 安装GCC

在Linux上编译ARM程序前，用户需要安装交叉编译器gcc-arm-none-eabi。下面章节将详细描述GCC的安装步骤。

2.2.1 下载GCC

访问网址https://launchpad.net/gcc-arm-embedded/+download，下载版本为*gcc-arm-none-eabi-5_4-2016q3-linux.tar.bz2*的安装包。

该安装包基于32-bit架构构建。如用户需要64-bit版本，可从https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads进行下载，Linux 64-bit Tarball版本。

2.2.2 安装

安装包为免编译版本，将其解压到合适的目录位置。

使用以下命令解压安装包：

```
tar xf gcc-arm-none-eabi-5_4-2016q3-linux.tar.bz2
```

2.2.3 设置环境变量

在环境变量PATH中增加路径，请用户按照实际情况添加相应的路径。以下仅为示例。

- Root用户

```
echo “export PATH=$PATH:/home/goodix/gcc-arm-none-eabi-5_4-2016q3/bin” >> /etc/bash.bashrc
source /etc/bash.bashrc
```

- 非Root用户

```
echo “export PATH=$PATH:/home/goodix/gcc-arm-none-eabi-5_4-2016q3/bin” >> ~/.bashrc
source ~/.bashrc
```

2.2.4 测试GCC安装结果

GCC安装完成后，可通过以下命令测试GCC是否安装成功。

```
arm-none-eabi-gcc -v
```

在Terminal上打印出以下信息，表示GCC安装成功。

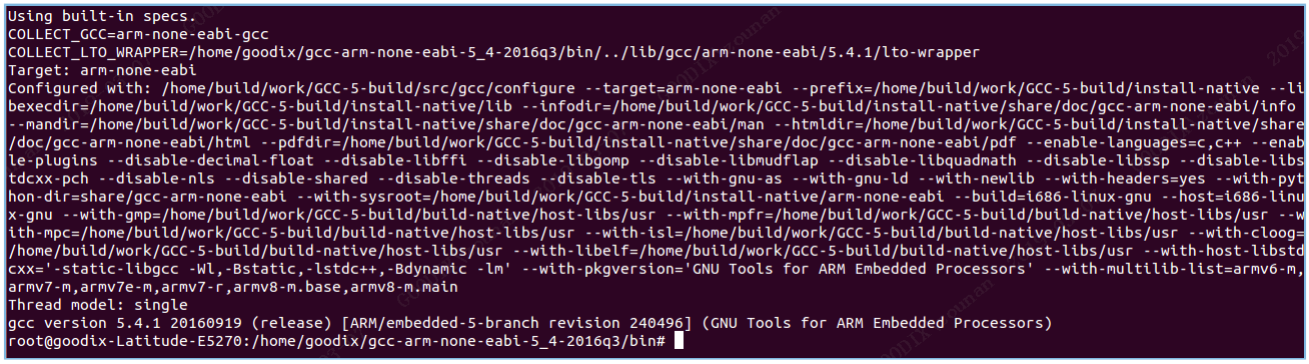

图 2-1 GCC安装结果显示

说明:

- Ubuntu版的arm-none-eabi-gcc编译器自身不区分32-bit / 64-bit。
- 部分Ubuntu发行版在执行`arm-none-eabi-*`系列命令时，可能出现错误提示“no such file or directory”，原因是缺少三方依赖库ia32-libs，执行如下安装命令即可：

  ```
  sudo apt-get install lib32ncurses5
  sudo apt-get install lib32z1
  ```

2.3 安装Python

1. 访问网址：www.python.org/downloads，下载并安装Python3。要求下载的Python版本和用户使用的Ubuntu系统兼容。

   输入以下命令安装Python3：

```
sudo apt-get install python3
```

2. 运行Python。

```
python
```

3. 如果Python安装成功，将查看到Python的版本信息。

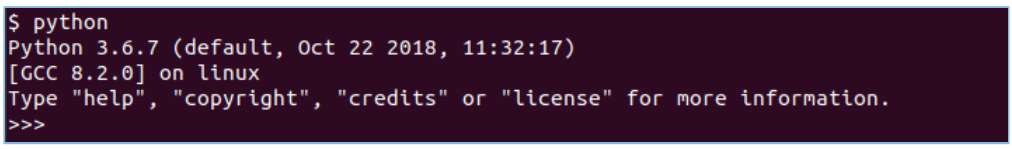

图 2-2 Python安装结果显示

说明:

本示例安装的Python版本为3.6.7。在Terminal上打印出：Python 3.6.7，表示Python安装成功。

2.4 安装J-Link

访问网址：www.segger.com/downloads/jlink/，在Segger官网下载J-Link for Linux。

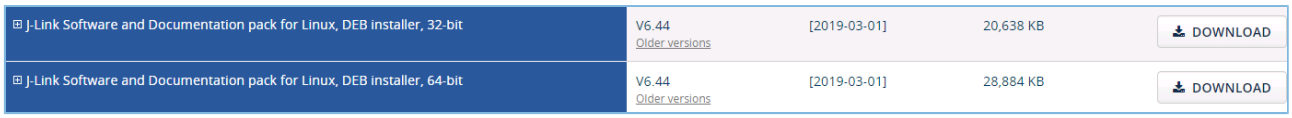

图 2-3 Segger官网J-Link下载

在Ubuntu下安装J-Link for Linux的DEB包。

说明:

- J-Link版本需和用户的Ubuntu系统兼容。
- J-Link使用6.10a及以上版本。
- 安装完成后，在命令行执行JLinkExe命令，正常情况下可开始使用J-Link。如果安装J-Link后不能正常使用，请检查环境变量是否添加成功。

2.5 开发板连接测试

在完成J-Link安装后，即可连接开发板测试。

2.5.1 连接开发板

使用Micro USB 2.0数据线连接GR5515 Starter Kit开发板与计算机。

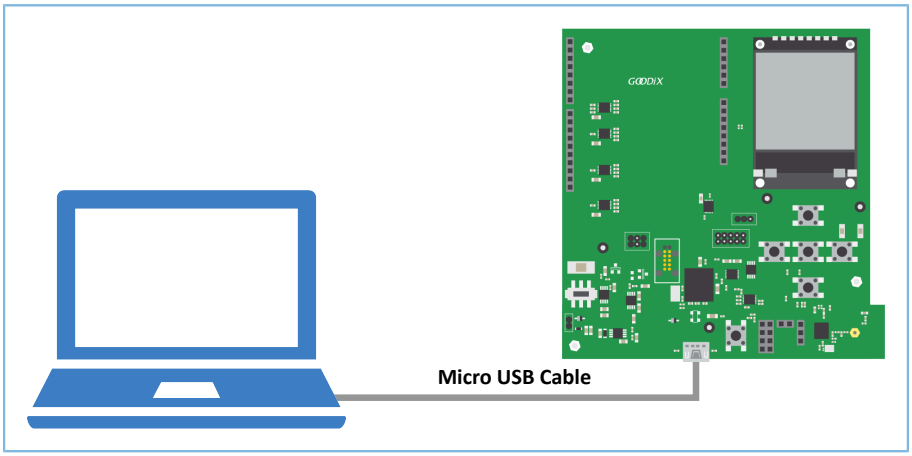

图 2-4 硬件连接示意图

2.5.2 使用J-Link连接开发板

确认J-Link已加入环境变量，在计算机终端界面依次键入命令（#之后的文本为命令注释）：

```
JLinkExe          # 命令行调用启动Jlink工具
connect           # 使用connect命令连接开发板，执行本命令前请保证已接入开发板
CORTEX-M4         # 指定CPU内核型号，如J-Link工具正常识别,可以直接回车
S                 # 选择硬件链接调试接口，S代表SWD
4000              # 指定SWD通讯速率，单位kHz,这里设置为4000
```

界面出现“Cortex-M4 identified”时，表示PC通过J-Link成功连接了GR5515 Starter Kit开发板。

```
Connecting to J-Link via USB...O.K.
Firmware: J-Link OB-SAM3U128 V3 compiled Sep 21 2017 14:14:50
Hardware version: V3.00
S/N: 483060523
VTref = 3.300V


Type "connect" to establish a target connection, '?' for help
J-Link>connect
Please specify device / core. <Default>: CORTEX-M4
Type '?' for selection dialog
Device>
Please specify target interface:
  J) JTAG (Default)
  S) SWD
TIF>s
Specify target interface speed [kHz]. <Default>: 4000 kHz
Speed>
Device "CORTEX-M4" selected.


Connecting to target via SWD
Found SW-DP with ID 0x2BA01477
Scanning AP map to find all available APs
AP[1]: Stopped AP scan as end of AP map has been reached
AP[0]: AHB-AP (IDR: 0x24770011)
Iterating through AP map to find AHB-AP to use
AP[0]: Core found
AP[0]: AHB-AP ROM base: 0xE00FF000
CPUID register: 0x410FC241. Implementer code: 0x41 (ARM)
Found Cortex-M4 r0p1, Little endian.
FPUnit: 6 code (BP) slots and 2 literal slots
CoreSight components:
ROMTbl[0] @ E00FF000
ROMTbl[0][0]: E000E000, CID: B105E00D, PID: 000BB00C SCS-M7
ROMTbl[0][1]: E0001000, CID: B105E00D, PID: 003BB002 DWT
ROMTbl[0][2]: E0002000, CID: B105E00D, PID: 002BB003 FPB
ROMTbl[0][3]: E0000000, CID: B105E00D, PID: 003BB001 ITM
ROMTbl[0][4]: E0040000, CID: B105900D, PID: 000BB9A1 TPIU
Cortex-M4 identified.
J-Link>
```

图 2-5 J-Link连接成功

2.6 编译SDK应用示例工程

本节通过应用示例工程ble_app_hrs，对Makefile的生成、使用和编译等进行介绍。

说明:

SDK_Folder为GR551x SDK的根目录。

2.6.1 Makefile文件

目前GR551x SDK默认提供了ble_app_hrs示例工程的makefile文件，方便用户进行参考和测试验证。其他示例工程的makefile文件需要使用脚本工具进行生成。

ble_app_hrs示例工程的Makefile文件的参考路径为：

`SDK_Folder\projects\ble\ble_peripheral\ble_app_hrs\make_gcc`。

Makefile：make编译规则文件，用于执行GCC命令（编译、链接）和各种操作系统命令，以及定义一系列规则，比如编译器属性、文件编译顺序、编译及链接规则、目标依赖关系等。通过执行make操作，生成可执行文件。

2.6.2 生成Makefile文件

GR551x SDK开发包中的应用示例工程，默认使用Keil uVision5 IDE环境进行编译构建，如果用户希望使用GCC工具链编译构建ble_app_hrs之外的应用示例工程，可使用*keil2makefile.py*脚本工具，将Keil的工程文件* .uvprojx转换为Makefile。

*keil2makefile.py*使用说明如下：

1. *keil2makefile.py*工具文件默认位于`SDK_Folder\tools\gcc`目录下（请注意不是*keil2make.py*）。
2. 为保证转换后的Makefile引用的源文件和头文件路径正确，已约束*keil2makefile.py*脚本在使用时需要与*.uvprojx文件位于同一个目录。
3. 将*keil2makefile.py*文件拷贝到目标应用工程的Keil_5目录下。以ble_app_hrs为例，将脚本拷贝至`SDK_Folder\projects\ble\ble_peripheral\ble_app_hrs\keil_5`。

```
$ ls
ble_app_hrs.uvoptx  ble_app_hrs.uvprojx  keil2makefile.py
```

图 2-6 拷贝keil2makefile.py至Keil_5目录

4. 从命令行切换到目标路径，执行如下命令。其中，“$(project_name)”表示Keil工程文件名，如ble_app_hrs，后同。

```
python keil2makefile.py $(project_name).uvprojx [-C/-D]
```

5. 如果Keil源工程文件存在多个构建目标（Target）时，在转换过程中，脚本工具会提示用户选择期望构建的目标，一般情况选择第一个（非Test）目标（Target）即可。

```
# python keil2makefile.py ble_app_hrs.uvprojx
>>> Transfer project : ble_app_hrs.uvprojx
>>> ALERT: ignore group gr_drivers because it's provided in sdk lib ...
>>> ALERT: ignore group gr_drivers because it's provided in sdk lib ...
>>> Find more than one compile target, Please select which one to generate makefile ?
++++++ 0 : GR5515_SK / ble_app_hrs
++++++ 1 : gr551x_auto_test / ble_app_hrs

>>> Enter the selected order : 0
>>> The goal project name : GR5515_SK / ble_app_hrs
>>> OS type: Linux
>>> Generate Makefile Successfully, located at ../make_gcc/Makefile
```

图 2-7 选择期望构建的目标

6. 转换成功后的Makefile文件，放置于与Keil_5目录平行的make_gcc目录，用户可进入此目录进行查看。

2.6.3 修改Makefile配置项

我们为Makefile提供了编译和链接的一组默认参数，用户可以根据项目的具体情况，对编译参数按需进行修改，但建议谨慎修改，以避免引起工程编译失败。

2.6.4 执行Make编译

1. 进入目标示例工程的Makefile文件放置目录，以ble_app_hrs为例，位于：

   SDK_Folder\projects\ble\ble_peripheral\ble_app_hrs\make_gcc

2. 通过系统的命令行工具进入Makefile目录，输入make命令即可自动编译：

```
make
```

如果命令行打印输出以下类似的信息（不同工程具体信息不相同），表示编译成功：

```
COMMENTS USET DEFAULT APP NAME
add pad byte count=8
size of the boot information struct     =0X00000018
the address of the boot info           =0X01000000
--------------------the boot information-----------------------------
boot_info.load_addr                    =0X01002000
boot_info.run_addr                     =0X01002000
bin size of the input file             =0X0003CFD0
the check_image_sum of the input file  =0X017C6133
the cmd of the input file              =0X00000000
xqspi_speed                            =0X00000000
code_copy_mode                         =0X00000000
system_clk                             =0X00000000
check image                            =0X00000000
boot delay flag                       =0X00000001
--------------------the boot information-----------------------------
bin file to hex file success!
gen info ok...
gen header ok...
gen app.bin with header ok...
gen app.hex ok...
gen app+info.hex ok...
rm -rf out/info.* out/header.* out/*.tmp out/*.lds out/ble_app_hrs.bin
=========================================
du -h out/ble_app_hrs_app.bin
244K    out/ble_app_hrs_app.bin
=========================================
```

图 2-8 编译成功显示

编译构建成功后，在out目录下输出*$(project_name)_app.bin*和*$(project_name)_app.hex*文件以及存放编译过程文件的文件夹lst和obj。

```
$ ls out
ble_app_hrs_app.bin  ble_app_hrs_app.hex  lst  obj
```

图 2-9 make输出文件

2.7 下载程序

在make_gcc目录下，可使用GProgrammer for Linux图形化界面烧写工具进行程序下载。

在下载SDK示例工程前，需要确保安装好GProgrammer for Linux，并且添加其所在路径到环境变量中，安装步骤如下：

1. 选择安装目录，解压GProgrammer for Linux的绿色版安装文件*GProgrammer_linux_x64_1.2.6.tar.bz2*，解压后的文件目录如图 2-10所示。

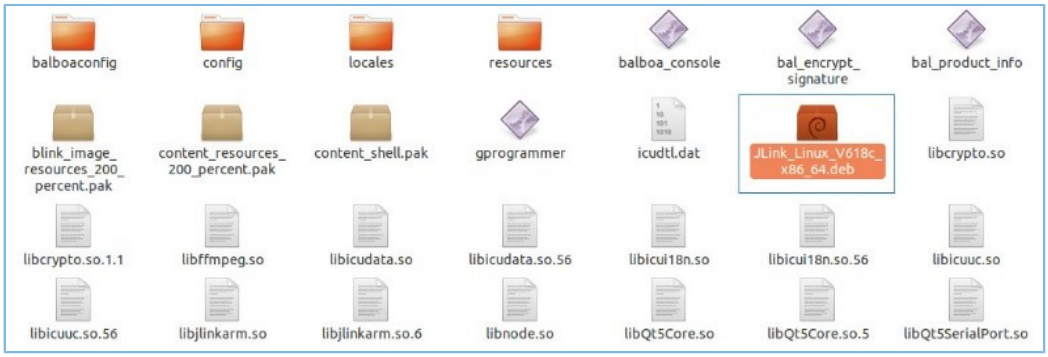

图 2-10 解压后的文件目录

2. 若用户需要安装JLink驱动，请双击目录下的*JLink_Linux_V618c_x86_64.deb*完成JLink驱动的安装，然后再启动软件安装。

3. 打开终端，使用`cd`命令进入到解压目录，输入命令`sudo ./gprogrammer`，根据提示输入密码后，即可启动GProgrammer软件。

   至此，在Linux环境下构建GR551x应用工程的交叉编译环境已完成。用户可以对GR551x SDK中的示例工程进行修改、编译、下载、测试等操作。

2.8 构建新应用工程

如果用户需基于GR551x开发新的应用，可参考以下步骤：

1. 用户可以根据编程习惯自由构建应用工程基础框架，也可选用如下方式构建新应用工程的基础框架。

   - 减法构建模式：在`SDK_Folder\projects`示例工程中寻找需求近似的应用工程，目录名修改为目标应用工程名，并更新keil工程文件名，保留工程需要继续引用的文件，移除不再使用的文件。使用*keil2makefile.py*脚本工具，生成新应用工程的Makefile初始文件。
   - 加法构建模式：参考模板应用工程的目录结构，构建新应用工程的目录结构，复制已存在的Makefile文件（如ble_app_hrs工程下），保留Makefile公共配置部分，移除源文件、头文件设置，后续再按工程实际进行设置。

2. 按项目需要对新应用工程进行源码开发，可增加、删除、修改源文件或头文件。

3. 根据新工程的文件依赖关系，修改Makefile中包含的源文件和头文件引用。

4. 根据实际项目需要，修改编译和链接等参数。

5. 执行make命令进行交叉编译生成.hex/.bin文件。用户可将.hex/.bin文件下载至GR5515 SK开发板进行测试验证。

3 常见问题

本章描述了在使用GCC示例时，可能出现的问题、原因及处理方法。

3.1 ble_tools.gcc运行错误

- 问题描述

  Make后提示“ble_tools.gcc运行错误”。

- 问题分析

  用户无可执行权限。

- 处理方法

  使用命令`chmod +x ble_tools.gcc`对***ble_tools.gcc***赋予执行权限。