技 术 文 件

文件名称：模组测试仪开发环境搭建指南

文件编号：

版　　本：

拟 制　Hardy

审 核　

批 准　

修改记录

文件编号	版本号	拟制人/ 修改人	拟制日期/ 修改日期	更改理由	主要更改内容 （写要点即可）
	V1.0	Hardy	2017/11/27	首版。	文档框架。
注：文件第一次归档时，“更改理由”、“主要更改内容”栏写“无”。					

目 录

1 引言

1.1 编写目的

本文档描述了使用辰卓科技模组测试仪（以下简称测试仪）进行二次开发的开发环境搭建过程。

1.2 文档约定

本文档遵循以下约定：

a) 使用的模板是：“技术文档模板 V1.0”。

b) 文档的密级分为：机密、秘密、内部公开、公开发布。

c) 表头文字使用了 20%灰度背景。

d) 插图一律“嵌入”于描述正文中，而非“浮于文字上方”。

e) 用同号、同体但加粗的文字来强调需要读者重视的内容。

f) 表格和图片统一使用“题注”，编号方式采用“包含章节号”，标签方式采用“图表”，表格的题注放于表格的上方，图片的题注放于图片的下方，其它选项使用默认值。

g) 文中引用题注时，统一使用“交叉引用”，引用内容“只有标签和编号”。

2 术语、定义和缩略语

2.1 术语、定义

本文使用的专用术语、定义见表 1。

表 1 术语、定义

术语/定义	英文	说　　明

2.2 缩略语

本文使用的专用缩略语见表 2。缩略语按其第 1 个字母顺序排列。

表 2 缩略语

缩略语	英文原文	中文含义

3 相关文档

本文涉及的相关文档表 3

表 3 相关文档

文件编号	文件名称	版本号	说明
			上游文档
			上游文档
			引用文档
			引用文档
			引用文档
			引用文档
			引用文档

4 编译环境

辰卓科技模组测试仪是使用与模组终端平台一致的硬件平台，其硬件架构遵循冯诺依曼计算机的组织结构，包括基于 CPU+GPU 的 SOC 处理器、存储器、输入输出设备。软件系统是基于 Ubuntu 的 Linux 操作系统，包括了驱动、内核、文件系统及应用程序等部分。

在测试仪上进行软件二次开发，主要是编译出能运行在 ARM Linux 平台下的二进制目标代码，主要包括可执行程序和动态库。

4.1 直接在测试仪上开发

测试仪是一台装有 Ubuntu 系统的 ARM 单板电脑（Single Board PC），在测试仪上预装了 QT 软件包和 gcc 编译器。开发时，可直接将源代码放到测试仪上，直接在测试仪上编辑、编译、调试和运行。

在测试仪上开发的基本流程与一台装有 Ubuntu 系统的 x86 个人电脑一样。大部分常用软件库及工具包已经预装，如果还需要其他的软件库或工具包，可以使用 apt 软件管理器在线安装，或下载离线软件包安装，或下载源码包编译。

测试仪默认的用户名和密码均为：ubuntu。

如果需要使用图形用户界面，将测试仪外接一个带 HDMI 接口的显示器即可，然后接 USB 接口的键盘鼠标。登录后，即可以看到 Ubuntu 图形用户界面。

如果不需要使用图形用户界面，可以使用网络方式联机，使用 ssh、VNC 等远程登录工具登录到测试仪（登录前先获取测试仪的 IP，目前测试仪支持主机名登录，主机名即为测试仪的序列号，如 527LC001）。如果使用 ssh 等终端登录，成功后，就能看到经典的 Linux 终端用户界面；如果使用 VNC 登录，成功后，则可以看到 Ubuntu 桌面。

4.2 在 PC 主机上交叉编译

为了方便在 PC 上编译出能在 ARM 平台下运行的二进制目标代码，当前常用的技术是交叉编译。通常称交叉编译器所在的机器为主机（Host），称编译出来的目标代码要运行的机器为目标机（Target）。

4.2.1 主机环境搭建

通常情况下，主机就是一台装有任意 Linux 发行版本的 PC 电脑。这里，我们以基于 x64 硬件架构和 Ubuntu 发行版的 Linux 操作系统电脑为例。

我们的 Ubuntu 操作系统可以是直接安装在物理 PC 机上。如果我们是使用的 Windows 操作系统，则需要安装双系统或在虚拟机中安装 Ubuntu 系统。

下面假设我们有一台 Windows 7 操作系统的 PC 机作为程序开发的主机。若物理机上有 Linux 系统，则略过前两节。

4.2.1.1 安装虚拟机

虚拟机软件有很多种，常用的有 VMWare 或 Virtual Box。

4.2.1.2 安装 Ubuntu 操作系统

装好虚拟机后，就可以在虚拟机中安装 Ubuntu 操作系统了。Ubuntu 操作系统当前推荐使用 Ubuntu 16.04 LTS。

4.2.1.3 安装交叉编译器

交叉编译器使用 gcc-linaro-5.4.1-2017.01-x86_64_aarch64-linux-gnu。这里选用的是 2017 年 1 月发布的基于 gcc5.4.1 的版本，理论上如果没有用新版本编译器的新特性，应该是向下兼容的，如 4.8.2 版本。

官方下载链接为：

x64 位环境：

http://releases.linaro.org/components/toolchain/binaries/5.4-2017.01/aarch64-linux-gnu/gcc-linaro-5.4.1-2017.01-x86_64_aarch64-linux-gnu.tar.xz

x32 位环境：

http://releases.linaro.org/components/toolchain/binaries/5.4-2017.01/aarch64-linux-gnu/gcc-linaro-5.4.1-2017.01-i686_aarch64-linux-gnu.tar.xz

下载完成后，这是一个软件压缩包。解压后即可以使用。

```
$ tar Jxvf *.tar.xz
```

4.3 程序部署及数据共享

目标机程序部署或数据共享有很多种方式。

4.3.1 U 盘拷贝

最简单的就是使用 U 盘拷贝程序或其他文件到目标机中，可以在图形用户界面操作，也可以通过终端命令 cp 拷贝。

4.3.2 Samba 服务器

Samba 服务是 Linux 提供文件共享的常用服务，测试仪预装了些服务，其配置文件是 /etc/samba/smb.conf，可以根据需要修改或使用默认配置。服务启动成功后，在 Windows 下直接输入测试仪的 IP 即可以发现共享的文件夹进行数据交换。

4.3.3 FTP

FTP 也是一种常用的数据传输服务，如果需要，用户可以自行配置。

5 项目开发

在测试仪上运行程序所支持的开发语言与 Ubuntu Linux 一致。这里仅以 C++为例。通常一个项目会按模块由很多源代码来构成，这一般需要以项目的方式来管理源代码。Linux 下 C++开发程序，有多种方式建立项目工程组织代码结构，无论哪种方式，最终都会生成 Makefile 文件，再调用 gcc 编译器来构建二进制目标代码。

5.1 Makefile 方式

最手工最纯粹的方式。手工组织目录结构将代码分模块存储，然后写一个 Makefile 文件来管理这些代码文件，并在其中调用 gcc 编译器来生成目标代码。

以下是两个 Makefile 文件模板：

5.1.1 动态库生成模板

5.1.2 可执行程序生成模板

5.2 CMake

CMake 可以用来管理项目工程，同时能自动生成 Makefile 文件，从而编译生成目标代码。

5.3 使用 QT 的软件工具包

这是一种较为简单且推荐的方法。

QT 是一种跨平台的 C++开发框架，含有丰富的类库，能写各种常见的应用程序、动态库、静态库、插件等。

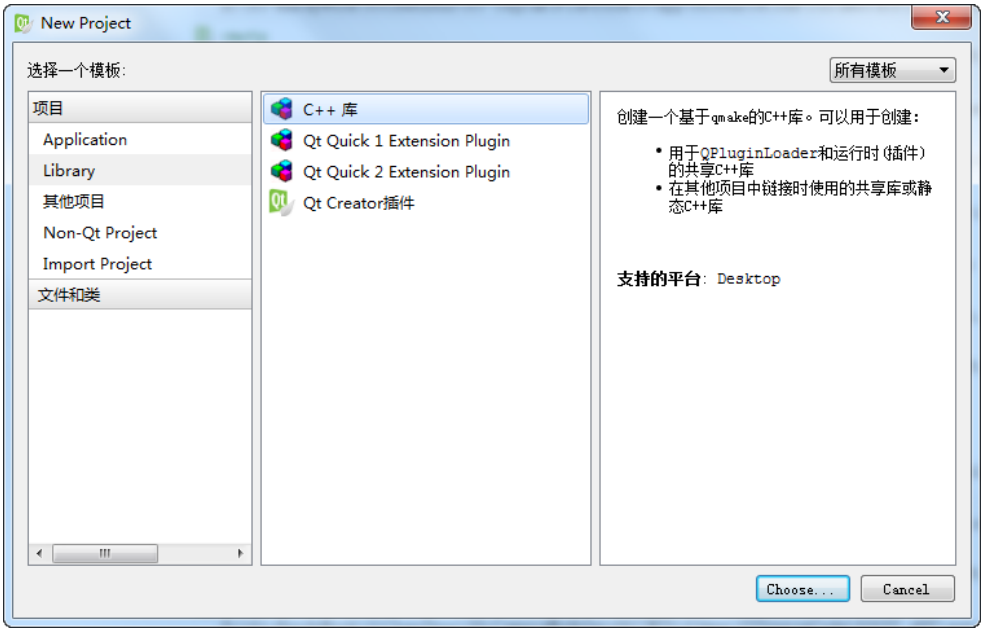
我们在开发过程中，可以使用 QT 自身的类库，也可以选择不使用 QT 的类库而用标准 C++库来完成编程

由于 QT 的跨平台性，其集成开发环境能正确识别 Windows 上创建的工程，我们完全可以在 Windows 上完成整个程序开发，然后将代码在 Linux 下重新编译即能在 Linux 平台使用。

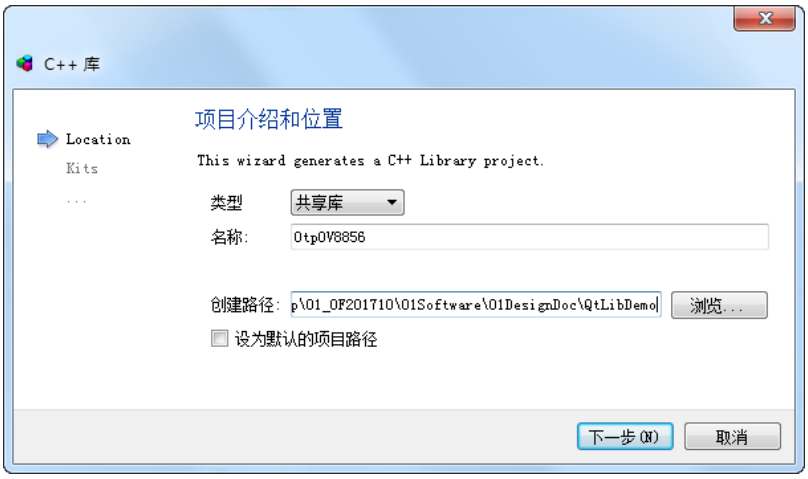
下面以开发一个算法动态库为例，讲解在 Windows 和 Linux 下，QT 工程的使用方法。

5.3.1 工程建立

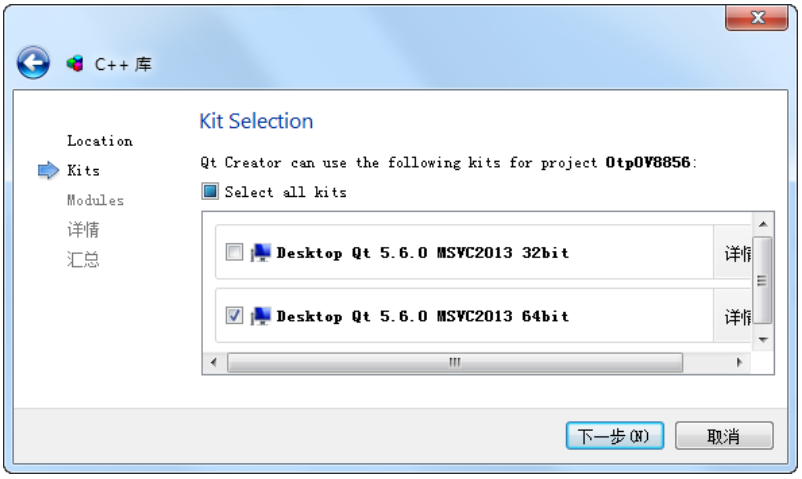
新建一个 C++库工程。



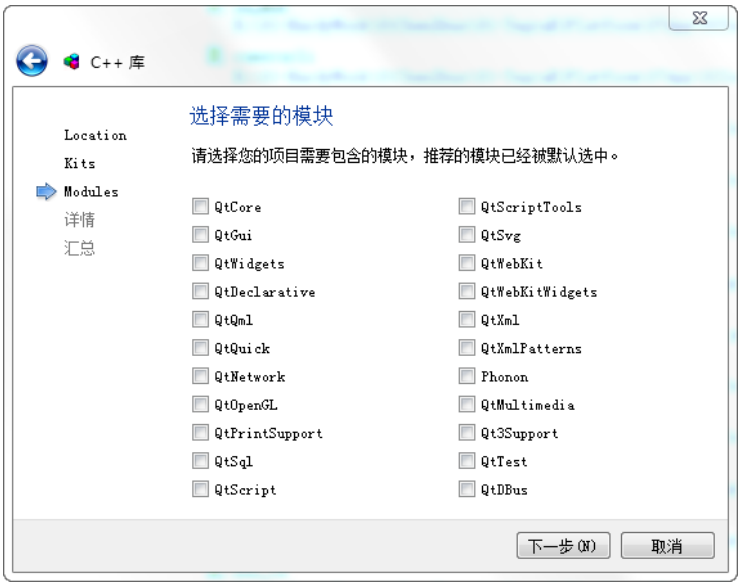
创建动态库。



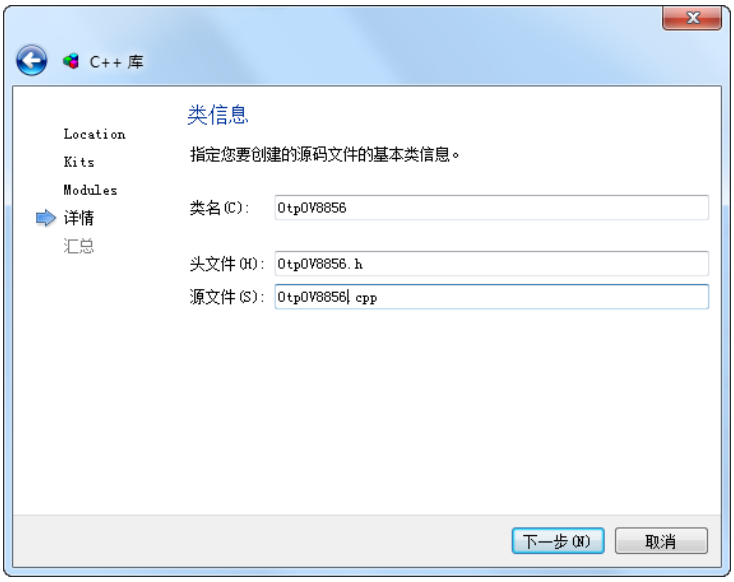
选择版本和编译模型。



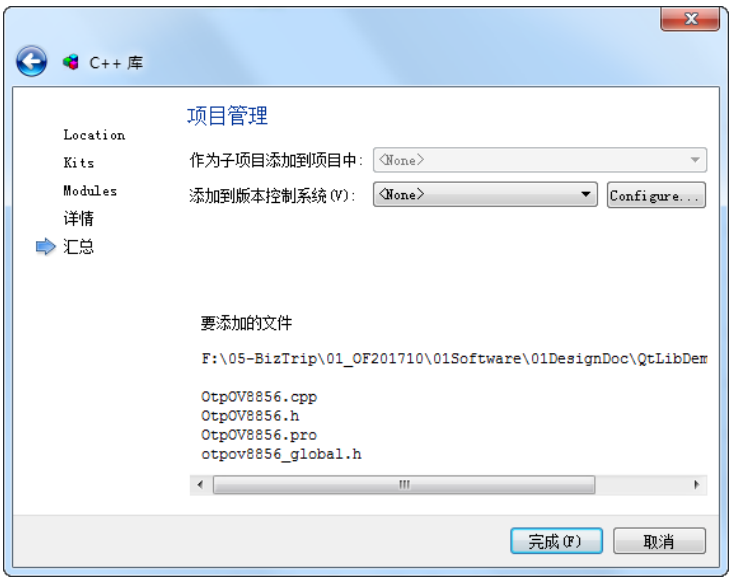
选择需要的模块。所有模块均不选中，即不使用 QT 类库。



类信息。保存文件时，QT 默认全小写，建议改为和类名一致。



完成工程创建。



对自动生成的.pro 工程文件稍作修改，具体修改方法可参考 QT 帮助文档。修改后的主要内容大致如下：

```
QT      -= core gui

CONFIG += c++11 plugin

TARGET = OtpOV8856
TEMPLATE = lib

CONFIG += debug_and_release
CONFIG(debug, debug|release) {
    DIST_DIR = Debug
} else {
    DIST_DIR = Release
    QMAKE_POST_LINK = $(STRIP) $(DESTDIR)$(TARGET)
}

contains(QT_ARCH, i386) {
    message("32-bit")
    CPU_ARCH = x86
} else {
    message("64-bit")
    CPU_ARCH = x64
}
DESTDIR = $$PWD/../lib/$${CPU_ARCH}/$${DIST_DIR}

DEFINES += OTP_EXPORTS

SOURCES += OtpOV8856.cpp

HEADERS += OtpOV8856.h



INCLUDEPATH += $$PWD/../includes/

unix {
    target.path = /usr/lib
    INSTALLS += target
```

```
}
```

主要修改了输出目录及引用头文件搜索目录。CONFIG 增加了 c++11 和 plugin 特性。

删除默认的 otpov8856_global.h 文件，此文件主要是导出符号的定义。

5.3.2 Windows 平台

上节中自动生成的代码可直接编译。在 Windows 上可以直接在 IDE 上选择菜单“构建->执行 qmake”，此步骤会生成 Makefile 文件。再选择菜单“构建->构建项目 xxx”，即可编译生成目标代码。

5.3.3 测试仪 Linux 平台

测试仪 Linux 平台上如果使用 QCreator 图形用户界面，操作方式与 Windows 一样。

通常在 Windows 下编辑好的代码，不需要在 Linux 下再编辑，这时使用终端用户界面就够了，将源代码拷贝到测试仪某个用户目录下。如果使用 Samba 服务建立了目录共享，还能使用 BeyondCompare 等工具进行代码实时比对。

在终端用户界面中，使用 cd 命令进行源代码所在目录。执行 qt 的 qmake 命令可以将 qt 的*.pro 工程生成 Makefile 文件：

$ qmake OtpOV8856.pro

再根据生成的 Makefile 文件执行相应的 make 命令：

$ make -j4

然后根据编译器给出的编译提示信息修改错误或调试代码。

5.4 Linux 平台常用调试手段

5.4.1 日志系统

在程序中建立功能完善的日志系统，有利于分析代码的执行流程，当程序执行异常时，可通过日志文件来分析代码是否有报错来判断 BUG。

5.4.2 gdb 调试工具

gdb 是 Linux 系统下功能非常完善的在线调试工具。其支持的命令很多，具体可以查看帮助文件。例如使用 backtrace 命令来调出程序崩溃时的调用栈。

6 测试仪平台软件性能优化

6.1 CUDA 编程

测试仪平台支持 CUDA 编程，利用 GPU 的并行计算，大大优化算法性能。

6.2 NEON 指令优化

7 参考资料