目录

[1. Abstract 2](#_Toc515628319)

[2. 代码编译命令 2](#_Toc515628320)

[2.1. 代码获取 2](#_Toc515628321)

[2.2. 各项目编译命令 2](#_Toc515628322)

[3. 代码修改规范 2](#_Toc515628323)

[3.1. 注释的添加 2](#_Toc515628324)

[3.1.1. 代码的修改 2](#_Toc515628325)

[3.1.2. 注释的添加 3](#_Toc515628326)

[3.2. 代码提交流程 4](#_Toc515628327)

[3.2.1. 获得修改文件列表 4](#_Toc515628328)

[3.2.2. 添加需要上传的文件 5](#_Toc515628329)

[3.2.3. 提交需要上传的文件 6](#_Toc515628330)

[3.2.4. 上传代码到远端服务器 9](#_Toc515628331)

[3.3. 代码合入服务器 9](#_Toc515628332)

[3.3.1. 添加reviewer 9](#_Toc515628333)

[3.3.2. 代码重新上传 10](#_Toc515628334)

[3.3.3. 回退代码节点 10](#_Toc515628335)

# Abstract

此文档介绍P100F代码获取，软件工程师修改代码规范以及提交流程。

# 代码编译命令

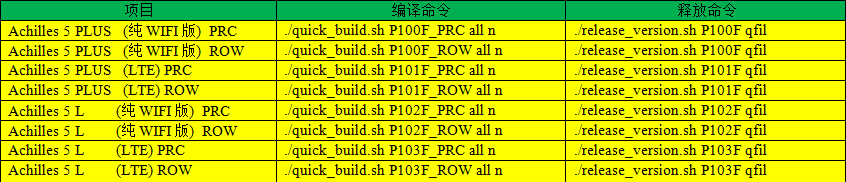
## 代码获取

repo init -u [git@10.0.30.8:LENX\_SDM450\_P100F\_SW1/tools/manifest.git -b PDU1](mailto:git@10.0.30.8:LENX_SDM450_P100F_SW1/tools/manifest.git%20-b%20PDU1)

repo sync -j4

repo start PDU1 --all

## 各项目编译命令



# 代码修改规范

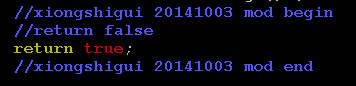
## 注释的添加

因为同一套代码，包含的项目很多，对应的需求很多，所以为了便于代码维护以及管理，新添加的代码以及修改的代码，均需要添加自己相应的注释。

### 代码的修改

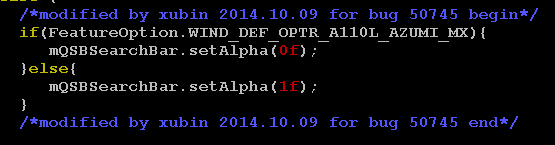
因新增需求或着平台bug，经常会有代码的修改，代码的修改除了要保持代码的可读性以及代码的整洁性外，还需要符合以下几点要求：

1、修改原有代码时，需要保留原有内容，以便后续检测。如图一



图一

2、对于项目定制需求，需要添加宏控，来管控新增代码，以免影响其他项目的代码流程。如图二



图二

### 注释的添加

对于修改的代码，新增的代码以及新增的文件，均需要添加自己的注释，以便查询和阅读。

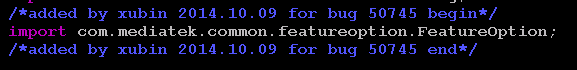
1、修改代码后，需要使用自己的注释把修改的代码上下包裹起来，格式可参照图一，图二。

注：上下注释中的内容，可根据自己习惯编写。

2、新添加的代码同样需要使用自己的注释把新增的代码上下包裹起来，格式可参照图一，图二。

注：上下注释中的内容，可根据自己习惯编写。

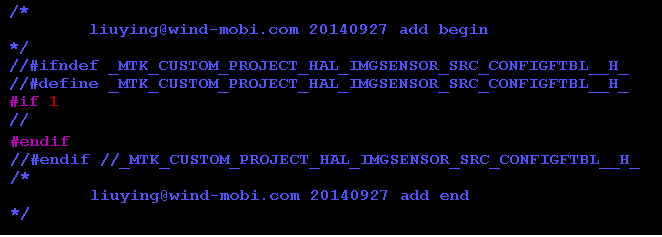
3、因宏控需要或调用函数需要，经常需要import其他的类，新添加的import同样需要使用自己的注释把新增的import上下包裹起来，格式可参照图三。



图三

注：上下注释中的内容，可根据自己习惯编写。

4、因为新调试设备或者新添加功能，经常会伴随着新的文件的引入，但是因为大版本升级的时候，会不时的对文件系统做出调整，有很多文件的新增和删除，这样就要求我们在自己新增文件的时候，也是需要在文件开头与文件结尾处添加自己的注释，以便文件不会被删除掉，格式如图四。



图四

注：上下注释中的内容，可根据自己习惯编写。

## 代码提交流程

代码修改并验证完成后，需要提交远端服务器。

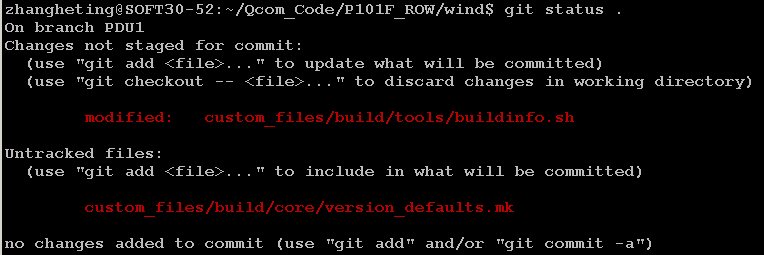
**注意：我们所有的修改提交均需要把修改的文件拷贝到wind/custom\_files相对应的路径下再进行提交，除wind目录之外的其他所有目录不允许提交。**

### 获得修改文件列表

在wind目录下，执行命令：

$git status .

会列出发生变动的文件列表，如图五：



图五

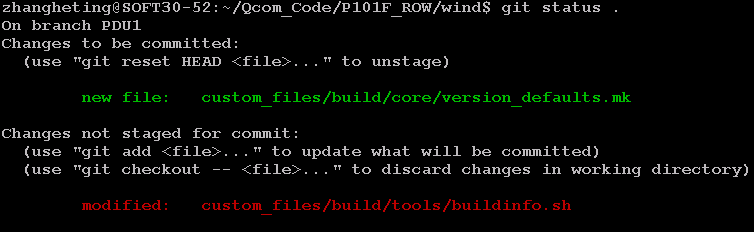
其中Untracked files:一栏为新添加的文件。

### 添加需要上传的文件

对于新添加的文件必须执行以下命令：

$git add custom\_files/build/core/version\_defaults.mk

执行好后，新增加的文件的字体会变为绿色，如图六：

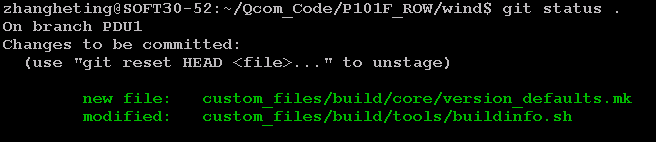


图六

如果当前目录下有改动的文件均需要提交的话，可使用以下命令，一次提交。

$git add .

add后，所以改动的文件均会变为绿色，如图七：



图七

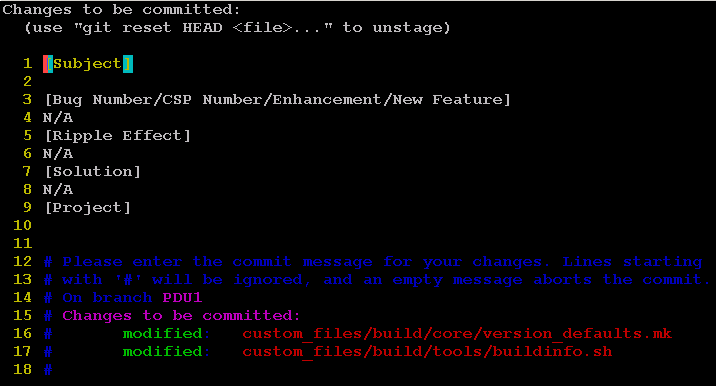
### 提交需要上传的文件

1、可执行如下命令来上传当前目录下的所有改动：

在wind目录下执行提交脚本git\_ci.sh(这个脚本的作用是：如果wind下有新加文件，则先提交一遍原始文件，再去提交对该文件的修改，方便gerrit上查看修改)

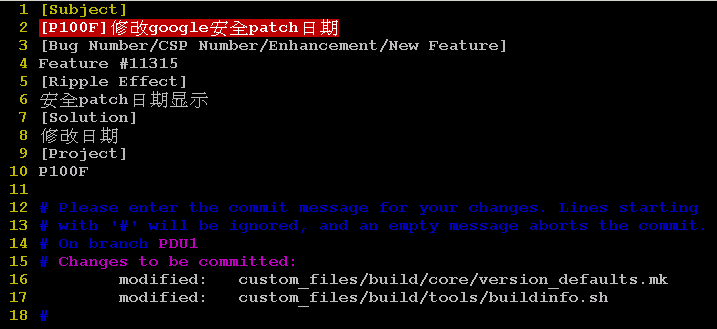
$ ./git\_ci.sh

执行命令后，会打开i界面，来填写改动的相应信息，如图八：



图八

填写完对应信息后，如图九，然后保存退出。



图九

图九中每一栏需对应添加的内容：

[Subject]-----项目名称以及修改内容

[Bug Number/CSP Number/Enhancement/New Feature] ----问题对应的bug号或feature号

[Ripple Effect]-----修改代码对应的波及分析

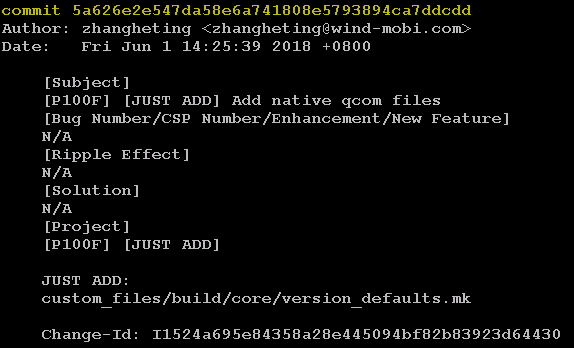
[Solution]--------问题的解决办法

[Project]--------项目名

以上各项填写完成后，需要把图中修改文件名称前面的#号去掉。

保存退出后，这时可以通过以下命令查看新提交代码的log，如图十和图十一，图十是提交脚本自动提交的一笔原始文件，图十一是对应文件的修改：

$git log .



图十



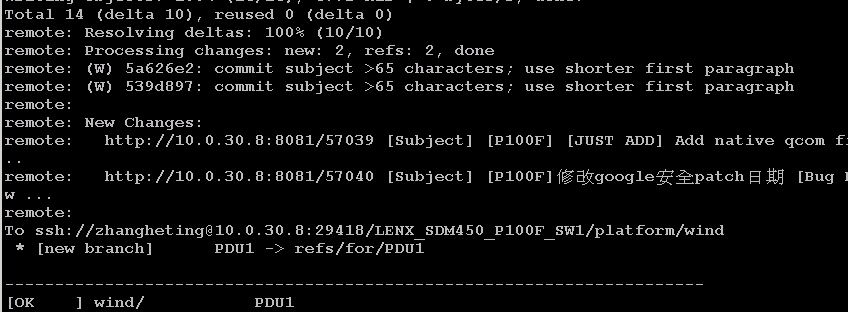
图十一

### 上传代码到远端服务器

在代码目录下执行以下命令，即可把之前步骤中提交的修改，上传到远端服务器，

$repo upload .

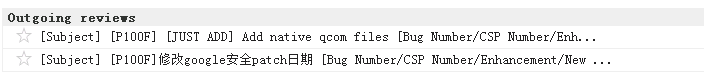
代码提交后，待会有图十二中所示的提示后，才代表提交成功。



图十二

## 代码合入服务器

上述步骤完成后，在网页端[http://10.0.30.8:8080](http://10.0.30.7:8080)中即可看到刚才的两笔提交记录，如图十三



图十三

### 添加reviewer

因并非所有人都有权限上传代码，所以需要添加reviewer来检测代码以及上传代码，如图十四，



图十四

检查一遍自己的提交，如果没有问题则自己+1，代码上传步骤结束，后续工作为merge上传代码。

### 代码重新上传

Review时发现错误，或者格式有问题，或者代码上传有冲突，就需要abandon掉提交，重新上传。

点击Abandon Change按钮，忽略掉提交，如图十五中所示按钮：



图十五

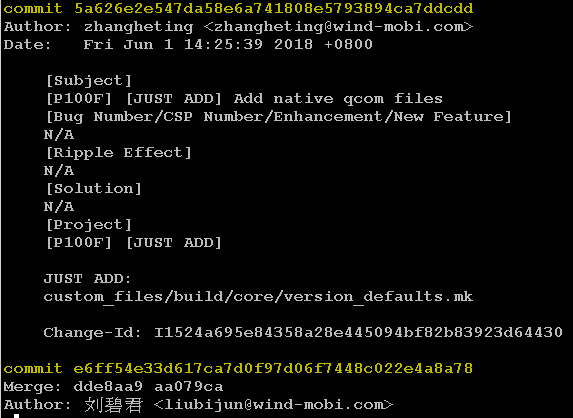
### 回退代码节点

在网页中abandon掉代码提交后，还需要在本地代码中回退节点，来重新提交。

在代码目录下，执行以下命令来查询log信息：

$git log .

查询出有问题的提交的前一次提交，如图十六：



图十六

执行以下命令，把代码回退到之前的节点：

$git reset --hard e6ff54e33d617ca7d0f97d06f7448c022e4a8a78

--hard后添加的参数为需要回到的节点的commit ID

然后重新合入修改，待修改好后，重新按照之前的步骤，提交修改的代码。