<https://blog.csdn.net/jzh1988903/article/details/41578045>

TEMPLATE：这个变量是用来定义你的工程将被编译成什么模式。如果没有这个设置，系统将默认编译为application。

                   TEMPLATE=app表示这个project将被编译成一个应用程序（application）

                                      lib(生成库的Makefile)

                                      subdirs(生成有多级目录管理的Makefile)，

                                      vcapp，vclib，vcsubdirs（对应Windows 下面VC）

TARGET：生成最后目标的名字

DESTDIR：指定生成目标的路径

DEPENDPATH：工程的依赖路径

INCLUDEPATH：这个用来指定工程要用到的头文件路径

SOURCES：工程需要的源文件

HEADERS：工程所需要的头文件

FORMS：工程要用到的ui文件（ui文件时用QT设计器生成的）

LIBS：加载动态库。LIBS+=./mitab/libmitab.so

TRASHLATIONS：加载要用到的语言翻译\*.ts文件

RESOURCES：加载要用到的资源\*.qrc文件。

CONFIG：告诉qmake应用程序的配置信息。

                   这个变量可以用来指定是生成debug模式还是release模式，也可以都生成。

                   也可以用来打开编译器警告(warn\_on输出尽可能多的警告信息)或者

                   关闭(warn\_off -编译器会输出尽可能少的警告信息)。还可以用来配置要Qt加载库。

                   想要qt+多线程：CONFIG+=qt thread

                   Eg：

                   CONFIG+=debug\_and\_release

                   CONFIG(debug,debug|release){

                   TARGET= hello

                   DESTDIR= ./debug

                   }else{

                   TARGET= hello

                   DESTDIR= ./release

                   }

UI\_DIR：UIC将ui转化为头文件所存放的目录

RCC\_DIR：RCC将qrc文件转化为头文件所存放的目录

MOC\_DIR：MOC命令将含Q\_OBJECT的头文件转换为标准的头文件存放的目录

OBJECTS\_DIR：生成的目标文件存放的目录

DEFINES：应用程序所需的额外的预处理程序定义的列表#在.h文件中就可以使用：#ifdefinexx\_xx\_xxx

LIBS += -L folderPath  //引入的lib文件的路径  -L：引入路径

                   Release:LIBS+= -L folderPath // release 版引入的lib文件路径

                   Debug:LIBS+= -L folderPath // Debug 版引入的lib 文件路径

RC\_FILE = xxx.icns   //程序图标

QT是跨平台的，所以我们在不同的平台上用同一个pro文件，这要加入有关平台的信息。

windows是win32，Linux平台是unix。

                   Eg：

                   在Windows和Linux的动态库文件格式是不一样的。一个是lib文件一个是so文件

                   win32:LIBS+= ./mitab/mitab\_i.lib

                   unix:LiBS+= ./mitab/libmitab.so

                   eg：

                   win32{

                   SOURCES+= hello\_win.cpp //win平台

                   }

                   unix{

                   SOURCES+= hello\_win.cpp //unix/linux平台

                   }

以通过在其它任何一个变量的变量名前加$$来把这个变量的值分配给当前的变量。例如：

                   MY\_DEFINES= $$DEFINE  #将DEFINE的值分配给MY\_DEFINES

/\*\*\*\*\*\*\*\*\*\*\*本人实例\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

QT      += core gui network xml sql

PATH +=$$(PATH)  #这个PATH变量在QT creator的 projects选项中Build environment可查看到，属于全局变量

INCLUDEPATH +=$$PATH /root/Desktop/TOU/API/header/#如果不加上用户自定义的有用文件绝对路径，在用户自定义cpp中将提示找不到对应的头文件

   message($$INCLUDEPATH)

DEPENDPATH +=$$(PATH)

   message($$DEPENDPATH)

OBJECTS\_DIR =tmp

UI\_DIR =inc

MOC\_DIR =tmp

RCC\_DIR = tmp

RC\_FILE = ICON.rc

RESOURCES +=

HEADERS += \

   inc/widget.h \

   inc/Shareheader.h \

   API/header/qextserialbase.h \

   API/header/posix\_qextserialport.h \

   API/header/comport.h \

   API/header/Encrypt.h \

   API/header/mylineedit.h \

   API/header/keyboardwidget.h

SOURCES += \

   src/widget.cpp \

   src/main.cpp \

   API/source/qextserialbase.cpp \

   API/source/posix\_qextserialport.cpp \

   API/source/mylineedit.cpp \

   API/source/keyboardwidget.cpp \

   API/source/comport.cpp

FORMS += \

   UI/widget.ui \

   UI/keyboardwidget.ui

contains(CONFIG,Release){   #全局函数 contains

       message(Release build!)

#如果是下载到开发板中的，请放开Q\_WS\_EMBEDED

       DEFINES =QT\_NODEBUG\_SUPPORT Q\_WS\_EMBEDED

       CONFIG    +=warn\_off

       TARGET = TOU\_release

       DESTDIR = release

 }else{

       message(Debug build!) #全局函数message

       DEFINES = DQT\_DEBUG\_SUPPORT

       CONFIG    +=warn\_on

       TARGET = TOU\_debug

       DESTDIR = debug

   }

win32: {#win

       SOURCES += ###

}

unix :{

contains(DEFINES,Q\_WS\_EMBEDED){

       INCLUDEPATH +=/usr/local/Trolltech/QtEmbedded-4.8.5-arm/include/Qwt/

       LIBS += -L /usr/local/Trolltech/QtEmbedded-4.8.5-arm/include/Qwt/lib-lqwt

}else{

       INCLUDEPATH +=/opt/qtsdk-2010.05/qt/include/Qwt/

       LIBS += -L /opt/qtsdk-2010.05/qt/include/Qwt/lib -lqwt

    }

}