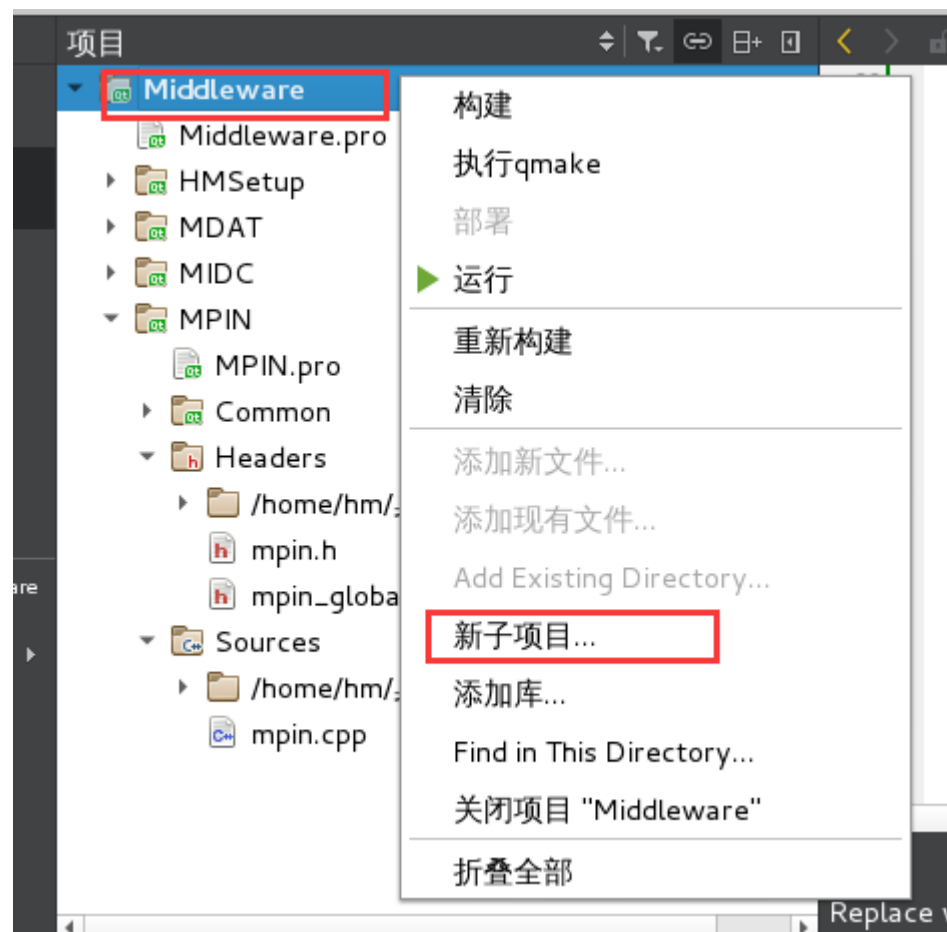
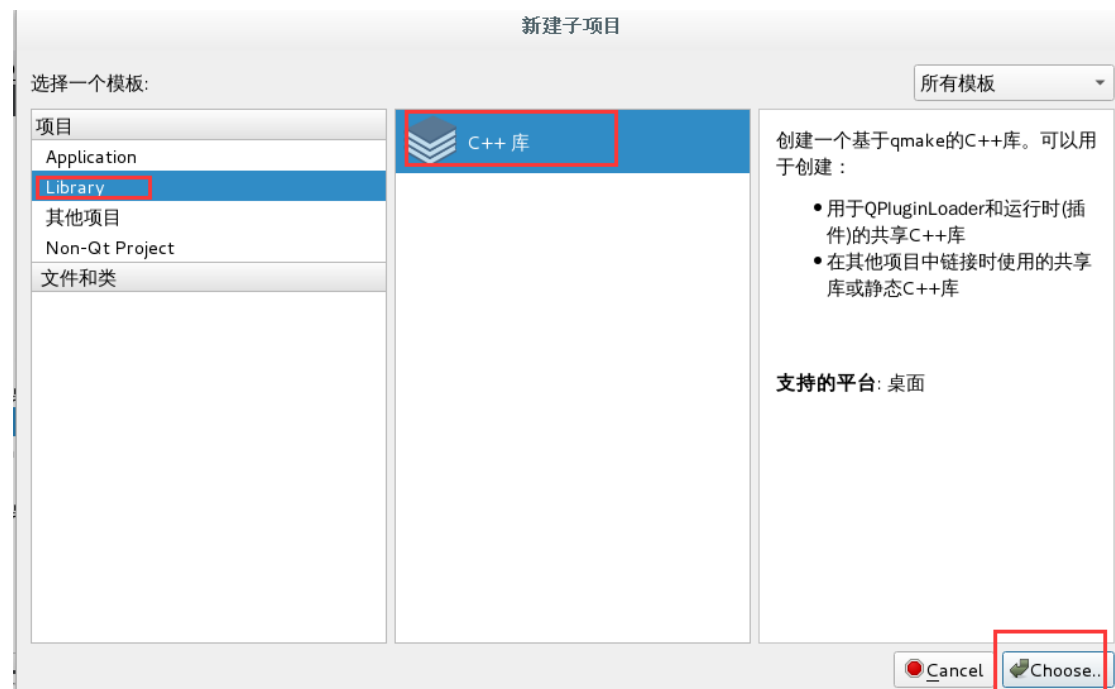


1、在主工程下新建子工程：



弹出如下界面：



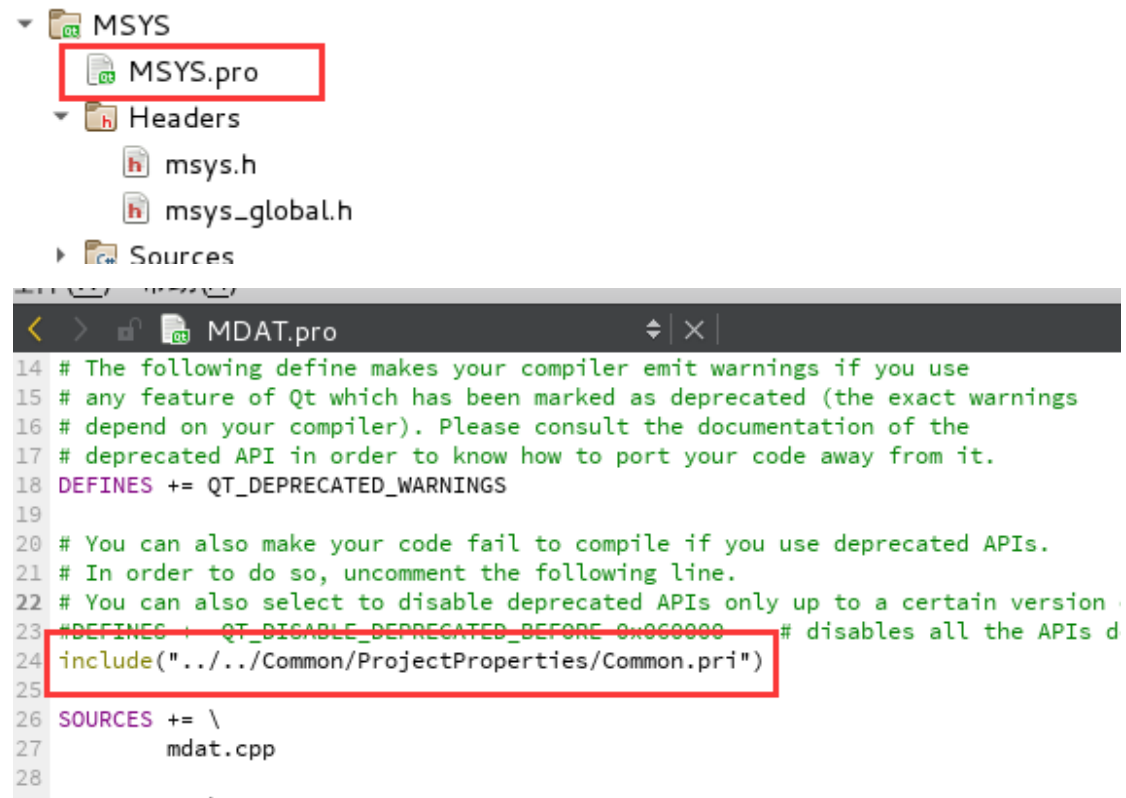


直接一直默认操作到完成即可。

2、修改子工程的项目属性：

非硬件模块在如下.pro 文件中加入：

`include("../Common/ProjectProperties/Common.pri")`

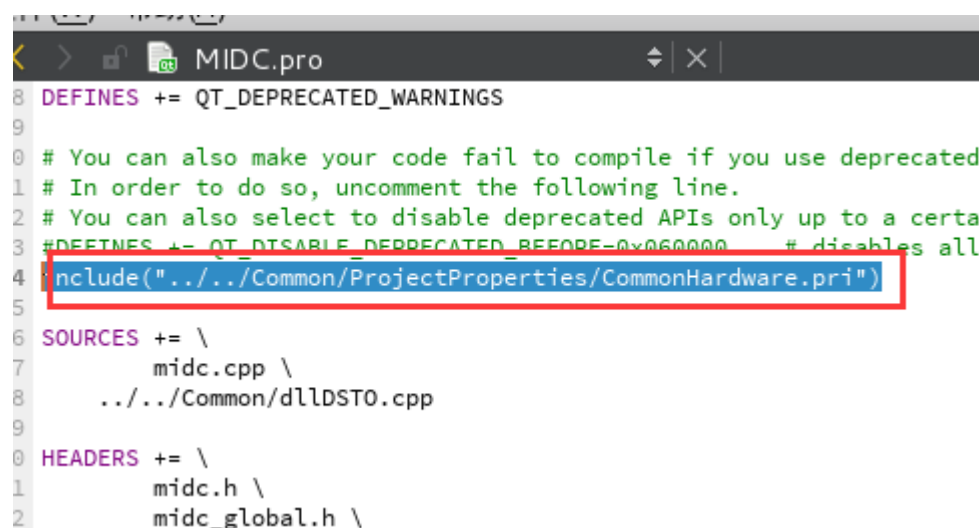


The image shows a Qt project structure and the content of the MDAT.pro file. In the project tree, the MSYS.pro file is highlighted with a red box. Below it, the Headers directory contains msys.h and msys_global.h. The Sources directory is also visible. The MDAT.pro file content is shown in a code editor, with the line `include("../Common/ProjectProperties/Common.pri")` highlighted by a red box.

```
14 # The following define makes your compiler emit warnings if you use
15 # any feature of Qt which has been marked as deprecated (the exact warnings
16 # depend on your compiler). Please consult the documentation of the
17 # deprecated API in order to know how to port your code away from it.
18 DEFINES += QT_DEPRECATED_WARNINGS
19
20 # You can also make your code fail to compile if you use deprecated APIs.
21 # In order to do so, uncomment the following line.
22 # You can also select to disable deprecated APIs only up to a certain version :
23 #DEFINES += QT_DISABLE_DEPRECATED_BEFORE 0x060000 # disables all the APIs d
24 include("../Common/ProjectProperties/Common.pri")
25
26 SOURCES += \
27     mdat.cpp
28
```

硬件模块在如下.pro 文件中加入：

`include("../Common/ProjectProperties/CommonHardware.pri")`

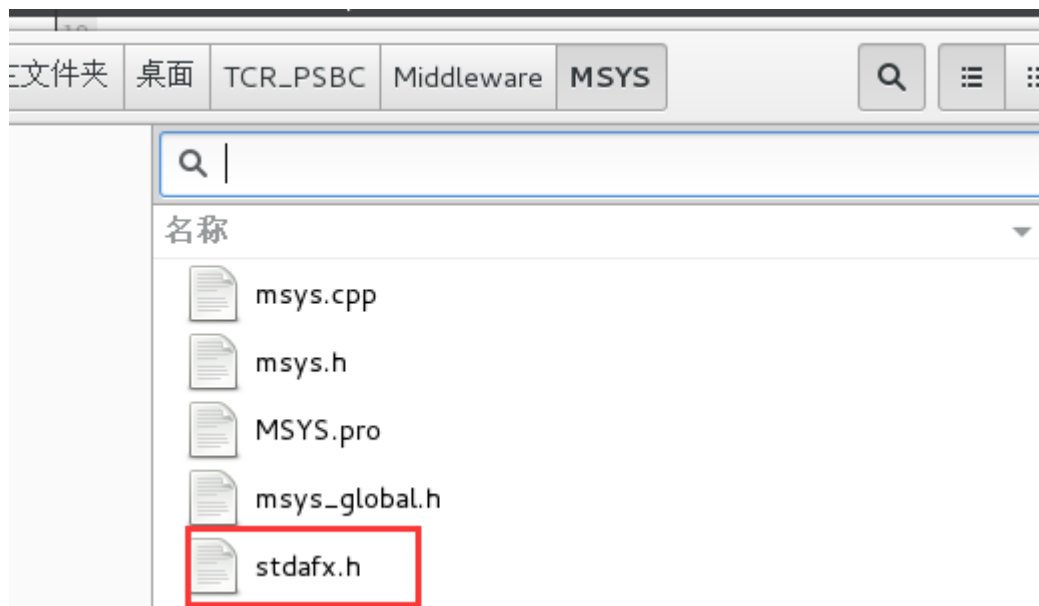


The image shows the content of the MIDC.pro file in a code editor. The line `include("../Common/ProjectProperties/CommonHardware.pri")` is highlighted by a red box.

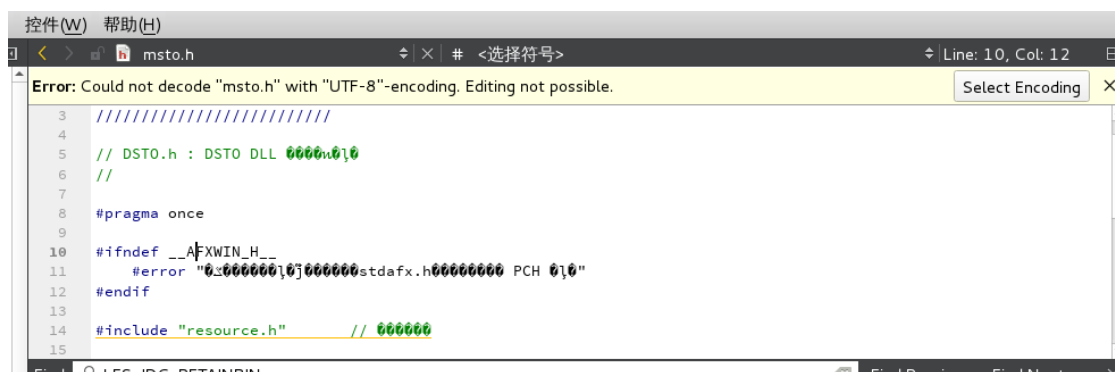
```
8 DEFINES += QT_DEPRECATED_WARNINGS
9
10 # You can also make your code fail to compile if you use deprecated
11 # In order to do so, uncomment the following line.
12 # You can also select to disable deprecated APIs only up to a certa
13 #DEFINES += QT_DISABLE_DEPRECATED_BEFORE 0x060000 # disables all
14 include("../Common/ProjectProperties/CommonHardware.pri")
15
16 SOURCES += \
17     midc.cpp \
18     ../Common/dllDST0.cpp
19
20 HEADERS += \
21     midc.h \
22     midc_global.h \

```

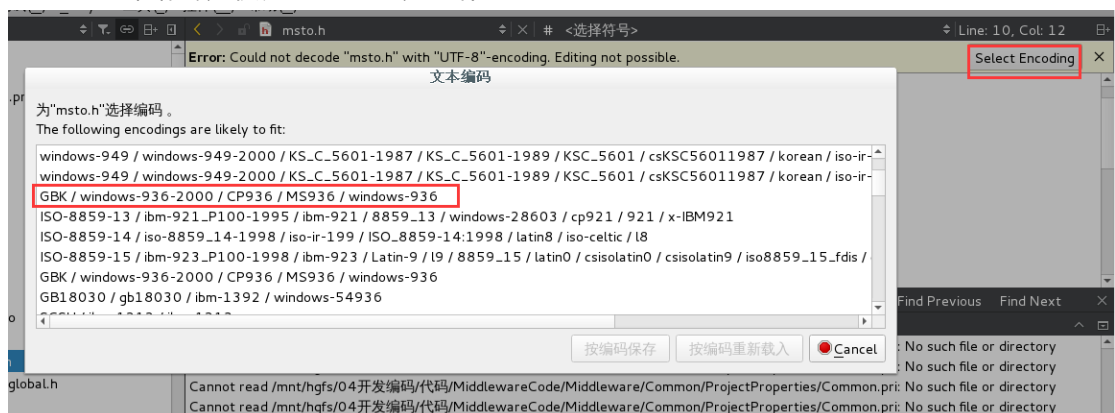
- 3、最后在当前工程目录下新增预编译头文件，可以从别的工程目录拷贝过来，然后修改成自己工程需要包含的公共头文件或定义，预编译头的使用是为了加快编译速度，如图：



- 4、将需要移植的模块文件直接拷贝到该工程下，替换同名文件（如：msys.h, msys.cpp，一般就这两个文件需要替换），打开拷贝过来的文件，会出现如下乱码的情况：



需要先选择编码，使其正常显示，操作如下：



再重新选择 utf-8 编码保存文件，操作如下：



拷贝文件移植完成，再按接下来的步骤修改相应代码

- 5、项目建好后，模块类需要继承自 QObject（硬件模块需继承自 CWebInteractObj，并实现其空函数，在上面已经将 CWebInteractObj 类文件加入到工程中），并在类的私有区域新增 Q_OBJECT 宏，该做法是为了能将该类对象在 QtWebEngine 中注册，供 JS 脚本调用，该类的供 JS 调用的公共属性、方法、及发送给 JS 的事件都需要按照如下方法编写，也可参考现有工程：

```

//属性
Q_PROPERTY(QString ServiceName READ GetServiceName WRITE SetServiceName)
Q_PROPERTY(QString StDetailedDeviceStatus READ GetStDetailedDeviceStatus)

public:
    CWebInteractObj(QObject *parent = nullptr);

    ~CWebInteractObj();

    //设置逻辑名
    void SetServiceName(const QString& strServiceName);|
    QString GetServiceName();

    virtual QString GetStDetailedDeviceStatus() = 0;

signals:
    //信号函数，类似之前的事件回调函数
    void ConnectionOpened(void);
    void OpenFailed(void);
    void ConnectionClosed(void);
    void CloseFailed(void);
    void DeviceError(void);
    void Timeout(void);
    void StatusChanged(const QString& PropertyName, const QString& OldValue, const QString& NewValue);

public slots:
    //方法，JS可直接调用
    //打开连接

public slots:
    //方法，JS可直接调用
    virtual short OpenConnection(void);

    virtual short CloseConnection(void);

```

最后注意不要忘记添加创建外部对象的函数，如：

```

extern "C" {
    MIDCSHARED_EXPORT QObject* CreateInstance(void);
}

```

具体实现请参考现有工程