API Example Usage Guide

API Version 0.55.61

2025-08-20

HAROGIC

Contents

1. C/C++ 8

1.1 Configure Development Environment 8

1.2 Usage Process for C++ Examples 13

1.2.1 Usage of General C++ Examples 13

1.2.2 Use of the AM/FM demodulation paradigm 14

1.2.3 Usage of the recording and playback example 15

1.3 Device-related 18

1.3.1 Get device information 18

1.3.2 Device standby 18

1.3.3 GNSS-related 18

1.3.4 Get and modify the IP address of the NX device 18

1.3.5 Mode switching time consumption 18

1.4 SWP mode 19

1.4.1 Standard spectrum acquisition 19

1.4.2 Simplified configuration mode 19

1.4.3 Maximum and minimum hold 19

1.4.4 Average Trace 19

1.4.5 Automatic Configuration Measurement 19

1.4.6 Frequency Compensation 19

1.4.7 Function execution time, scanning speed, and throughput 20

1.4.8 Obtain spectrum peak values 20

1.4.9 Signals and Spurious 20

1.4.10 Simultaneous Acquisition of Spectrum and IQ 20

1.4.11 Reading SWP Stream Disk Data from Application Software 20

1.4.12 Using GNSS 10MHz Reference Clock 20

1.4.13 External Trigger Mode 20

1.4.14 Trace Alignment Method 21
1.4.15 Number of Spectrum Frames Obtainable Within a Certain Time 21
1.4.16 External Trigger Calibration of Internal 10MHz Reference Clock 21
1.4.17 Phase noise measurement 21
1.4.18 Channel power measurement 21
1.4.19 Neighbourhood power ratio measurement 21
1.4.20 Percentage occupied bandwidth measurement 21
1.4.21 XdB Occupied Bandwidth Measurement 22
1.4.22 IM3 measurements 22
1.4.23 Frequency interval matching RBW 22
1.4.24 Using an External 10MHz Reference Clock 22
1.5 IQS Mode 22
1.5.1 Obtain Fixed Number or Continuous Stream of IQ Data 22
1.5.2 Simplified configuration mode 22
1.5.3 IQ data converted to voltage V units 22
1.5.4 Time taken to issue configuration and acquire IQ 23
1.5.5 IQ to Spectrum Data 23
1.5.6 IQ to Spectrum (using liquid library version) 23
1.5.7 FM Demodulation 23
1.5.8 AM Demodulation 23
1.5.9 Digital Downconversion 23
1.5.10 Digital Low-Pass Filter 23
1.5.11 Audio Analysis 23
1.5.12 Read the IQS stream disk data from Application Software 24
1.5.13 Record IQ data in .wav format 24
1.5.14 .wav changed to .csv 24
1.5.15 Streaming and reading IQ data 24
1.5.16 Multithreaded acquisition, processing, and streaming of IQ data 24

1.5.17 GNSS 1PPS trigger 24
1.5.18 IQS multi-device synchronization 24
1.5.19 External Trigger 25
1.5.20 Timer Trigger 25
1.5.21 Level Trigger(pre-trigger)........ 25
1.5.22 Level Trigger (Trigger delay) 25
1.5.23 Using GNSS 10MHz Reference Clock 25
1.5.24 Fixed Step Multi-Frequency IQ Data Acquisition........ 25
1.6 DET Mode 25
1.6.1 Obtain detection data for fixed points or continuous streams. 25
1.6.2 Simplified configuration mode 26
1.6.3 Read the DET stream disk data of Application Software 26
1.6.4 Pulse detection (to be opened later) 26
1.7 RTA mode 26
1.7.1 Obtain real-time spectrum data for fixed points or continuous stream 26
1.7.2 Simplified configuration mode 26
1.7.3 Read the RTA stream disk data of Application Software 26
1.7.4 Time consumption of each frame of data in RTA mode 26
1.8 ASG Signal Source (optional) 27
1.8.1 Output single tone/sweep/power scan signals 27

2. Digital Demodulation (optional) 28

3. Qt 29

3.1 Configure Development Environment 29
3.2 Qt Example Usage Process........ 39
3.3 Qt Example Description 40

4. Python 42

4.1 Configure Development Environment 42

4.2 Python Example Usage Process 45

4.3 Python Example Description 47

4.3.1 Get device information 47

4.3.2 Obtain Standard Spectrum Data 47

4.3.3 Obtain IQ Data for a Fixed Number of Points or Duration 47

4.3.4 Obtain Power Detection Data for a Fixed Number of Points or Duration 47

4.3.5 Obtain real-time spectrum data for a fixed number of points or duration 47

4.3.6 IQ to Spectrum Data 48

5. Matlab 49

5.1 Configure Development Environment 49

5.1.1 Install MSYS2 49

5.1.2 Configure Matlab 52

5.1.3 Call htra_api.dll Description 56

5.2 Matlab example usage process 58

5.3 Introduction to Accompanying Examples 61

5.3.1 Get device information 61

5.3.2 Obtain Standard Spectrum Data 61

5.3.3 Create multiple cursors to display the frequency and power of the cursors. 61

5.3.4 Collect the peak spectrum every five minutes. 61

5.3.5 Obtain continuous stream or fixed number of IQ data. 61

5.3.6 The acquired IQ data is converted into spectrum data. 61

5.3.7 Acquire continuous stream or fixed number of power detection data. 62

5.3.8 Acquire continuous stream or fixed duration real-time spectrum data. 62

5.3.9 Internal signal source output signal. 62

5.3.10 Lock GNSS antenna and DOCXO oscillator. 62

5.3.11 Multi-machine synchronization 62

6. C# 63

6.1 Configure Development Environment 63

6.1.1 Development Environment Confirmation 63

6.1.2 Project Setup 63

6.2 C# Example Usage Process 72

6.3 C# Example Descriptions 74

6.3.1 Get device information 74

6.3.2 Obtain Standard Spectrum Data 74

6.3.3 Obtain IQ Data for a Fixed Number of Points or Duration 74

6.3.4 Obtain Power Detection Data for a Fixed Number of Points or Duration 74

6.3.5 Obtain real-time spectrum data for a fixed number of points or duration 74

6.3.6 Output single-tone signal 74

6.3.7 AM/FM Demodulation 75

6.3.8 IQ to Spectrum Data 75

6.3.9 Low-pass filtering 75

6.3.10 Digital Downconversion 75

6.3.11 Phase noise test 75

7. Java (to be supplemented) 76

8. Labview 77

8.1 Configure Development Environment 77

8.1.1 Export library functions from htra_api.dll using LabVIEW 77

8.1.2 Using the API in the LabVIEW environment 84

8.1.3 Use the newly exported library functions in an existing project. 88

8.1.4 Generate an EXE from the vi in LabVIEW. 91

8.2 The usage process of Labview examples 95

8.3 Labview Example Description 98

8.3.1 Get device information 98

8.3.2 Standard spectrum acquisition 98

8.3.3 Obtain IQ Data for a Fixed Number of Points or Duration 98

8.3.4 Streaming and reading IQ data 98

8.3.5 IQ to Spectrum Data 98

8.3.6 Digital Downconversion 98

8.3.7 Audio Analysis 99

8.3.8 Obtain Power Detection Data for a Fixed Number of Points or Duration 99

8.3.9 Obtain real-time spectrum data for a fixed number of points or duration 99

8.3.10 ASG Signal Source Output Signal 99

9. Linux 100

9.1 Environment Version Compatibility Self-Check 100

9.2 Accompanying documentation 102

9.2.1 HTRA_C++_Examples 102

9.2.2 HTRA_Qt_Examples 102

9.2.3 HTRA_Python_Examples 103

9.2.4 Install_HTRA_SDK 103

9.3 Driver file configuration 105

9.4 C++ example usage and project creation 106

9.4.1 C++ example usage 106

9.4.2 C++ Project Creation and Compilation 107

9.4.3 C++ Project Cross-Compilation 112

9.5 Using Qt Examples and Project Creation 115

9.5.1 Using Qt Examples 115

9.5.2 Qt Project Creation and Compilation 119

9.5.3 Cross-compiling Qt projects 131

9.6 Usage of Python examples and project creation 136

9.6.1 Usage of Python examples 136

9.6.2 Python Project Creation 138

9.7 Gnuradio Module Construction and Use 138

9.7.1 HTRA OOT build use 138

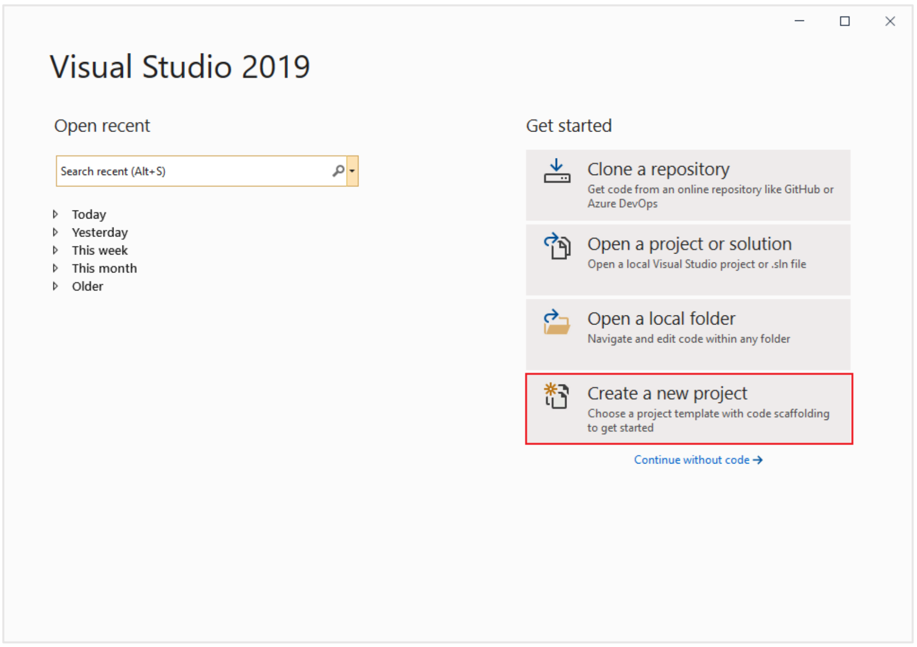
9.7.2 SoapySDR Build Usage........140

9.8 Java (to be supplemented) 143

1. C/C++

1.1 Configure Development Environment

1. Open VS Studio 2019 and create a new project.



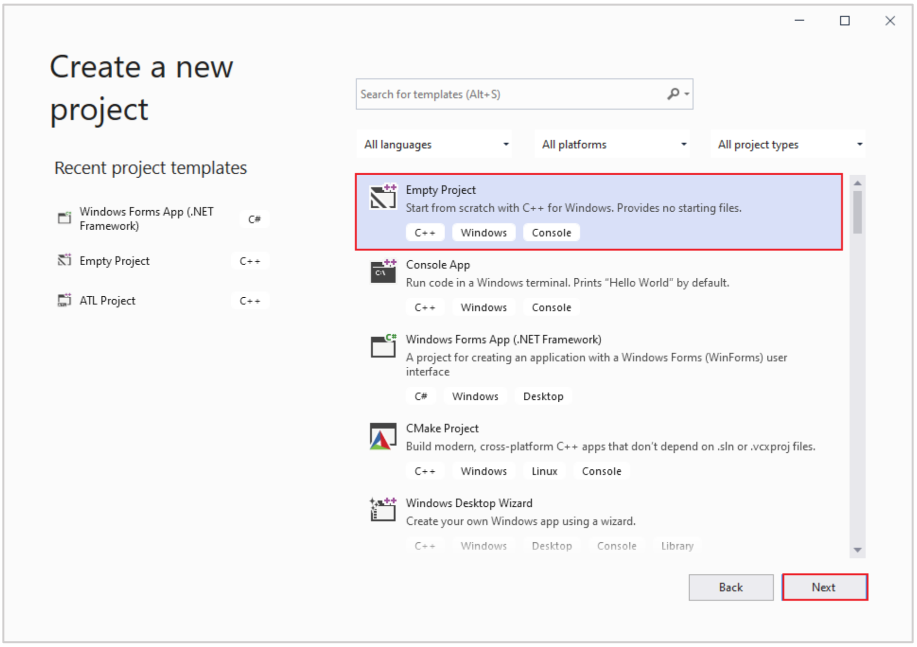
Get code from an online repository like GitHub or Azure DevOps

Open a local Visual Studio project or .sln file

Navigate and edit code within any folder

Choose a project template with code scaffolding to get started

2. Select Empty Project and click Next.



Start from scratch with C++ for Windows. Provides no starting files.

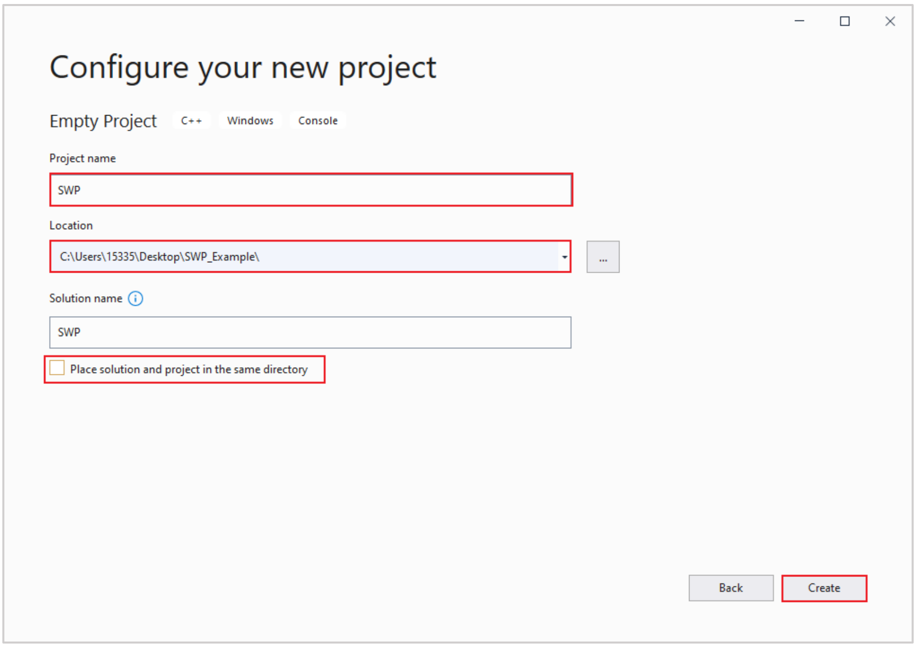
Run code in a Windows terminal. Prints "Hello World" by default.

A project for creating an application with a Windows Forms (WinForms) user interface

Build modern, cross-platform C++ apps that don't depend on .sln or .vcxproj files.

Create your own Windows app using a wizard.

3. Fill in the project name and storage location, uncheck "Place solution and project in the same directory," and then click Create.



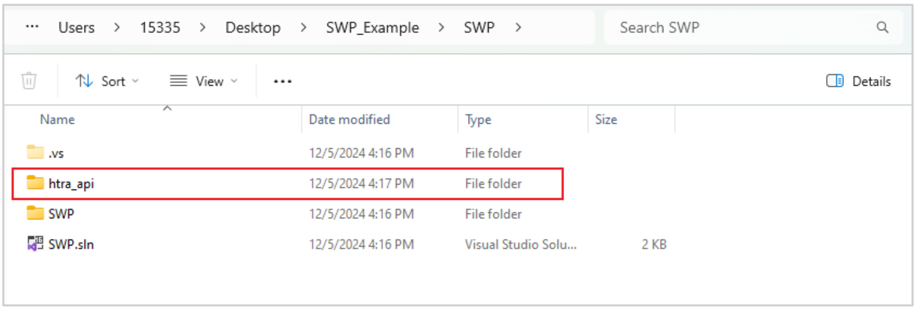
Project name: SWP

Location: C:\Users\15335\Desktop\SWP_Example\

Solution name: SWP

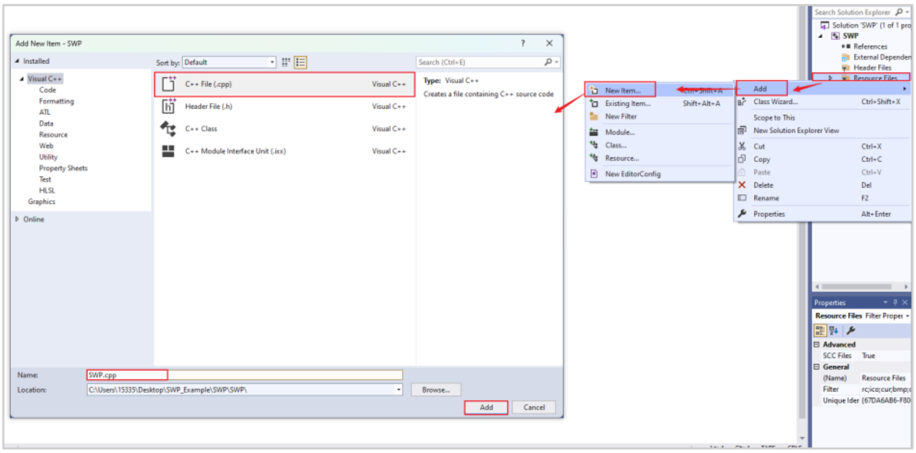
☐ Place solution and project in the same directory

4. Once created, copy the htra_api folder from the delivery USB drive Windows\HTRA_API\x86 to the same level directory of the project (this example is for configuring an x86 architecture project; if you want to configure an x64 architecture project, copy the Windows\HTRA_API\x64\htra_api folder).



Name	Date modified	Type	Size
.vs	12/5/2024 4:16 PM	File folder	
htra_api	12/5/2024 4:17 PM	File folder	
SWP	12/5/2024 4:16 PM	File folder	
SWP.sln	12/5/2024 4:16 PM	Visual Studio Solu...	2 KB

5. Double-click to open SWP.sln, and create a new SWP.cpp file in the source files.

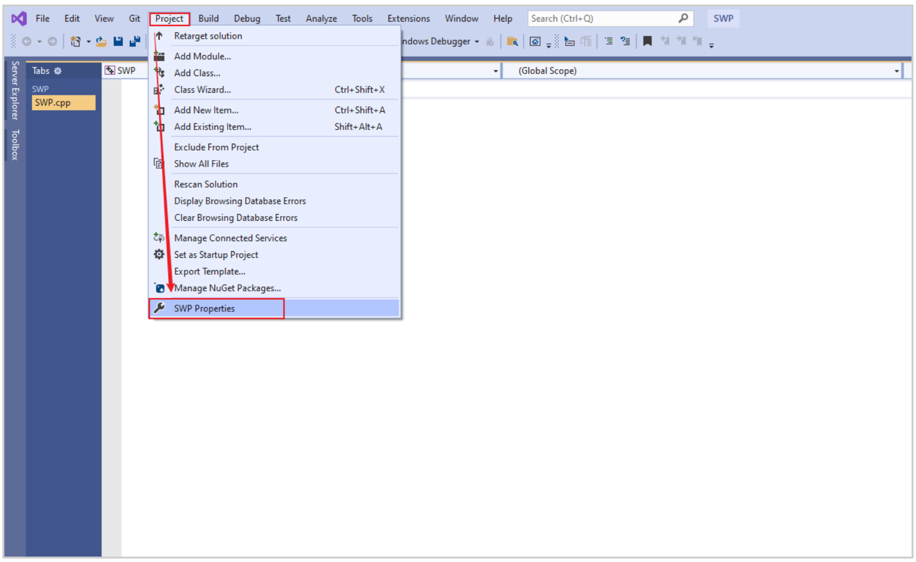


Name: SWP.cpp

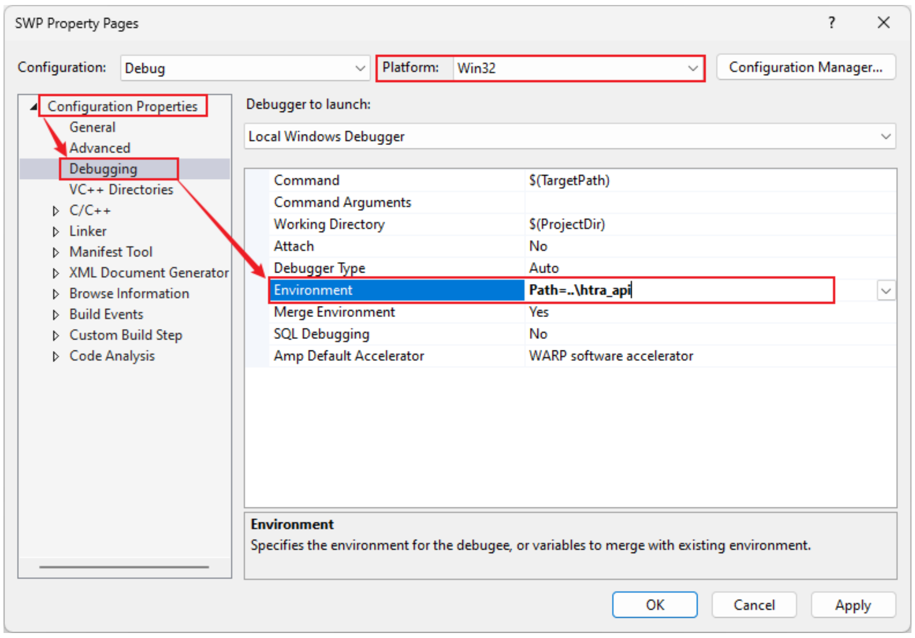
Location: C:\Users\15335\Desktop\SWP_Example\SWP\SWP\

Advanced
SCC Files: True
General
(Name): Resource Files
Filter: rc;ico;cur;bmp;
Unique Ider: {67DA6AB6-F80

6. Click on "Project" in the menu bar and select "Properties."



7. Select "Win32" for the configuration platform, and set the environment variable in Configuration Properties -> Debug to Path=..\htra_api (when configuring for the x64 architecture, select "x64" for the configuration platform; otherwise, steps 7-10 of the configuration process are the same as for the x86 architecture (Win32)).


SWP Property Pages

Configuration: Debug Platform: Win32 Configuration Manager...

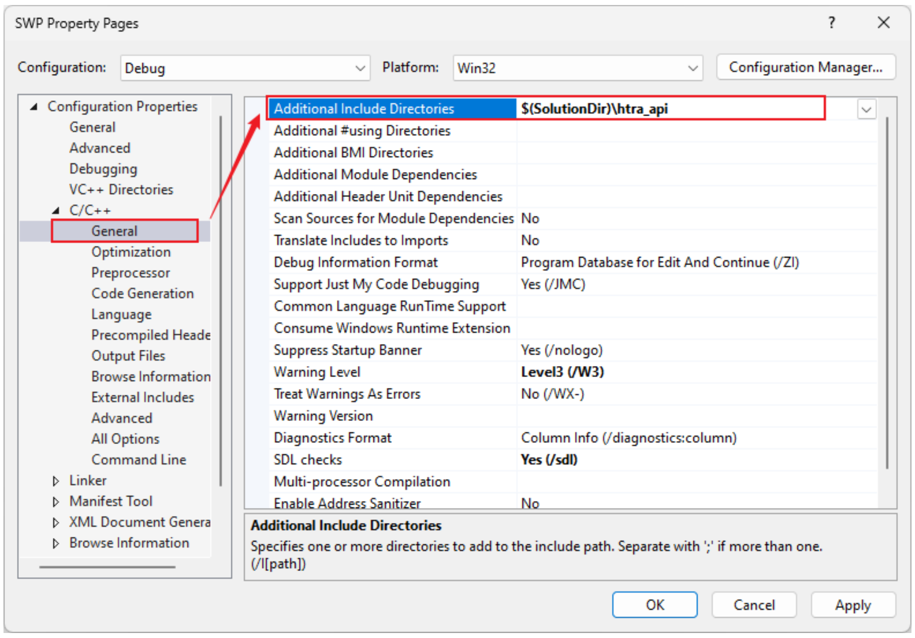
Debugger to launch:

Local Windows Debugger

Command	$(TargetPath)
Command Arguments	
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	Path=..\htra_api
Merge Environment	Yes
SQL Debugging	No
Amp Default Accelerator	WARP software accelerator

Environment
Specifies the environment for the debugee, or variables to merge with existing environment.

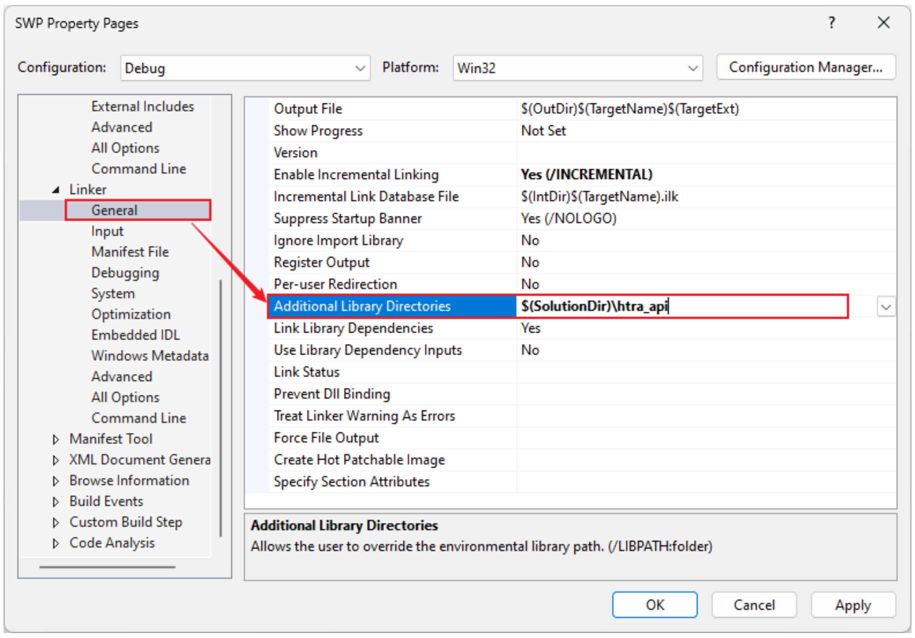
8. Set the Additional Include Directories in Configuration Properties -> C/C++ -> General to $(SolutionDir)\htra_api.



Additional Include Directories	$(SolutionDir)\htra_api
Additional #using Directories	
Additional BMI Directories	
Additional Module Dependencies	
Additional Header Unit Dependencies	
Scan Sources for Module Dependencies	No
Translate Includes to Imports	No
Debug Information Format	Program Database for Edit And Continue (/ZI)
Support Just My Code Debugging	Yes (/JMC)
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Warning Version	
Diagnostics Format	Column Info (/diagnostics:column)
SDL checks	**Yes (/sdl)**
Multi-processor Compilation	
Enable Address Sanitizer	No

Additional Include Directories
Specifies one or more directories to add to the include path. Separate with ';' if more than one. (/I[path])

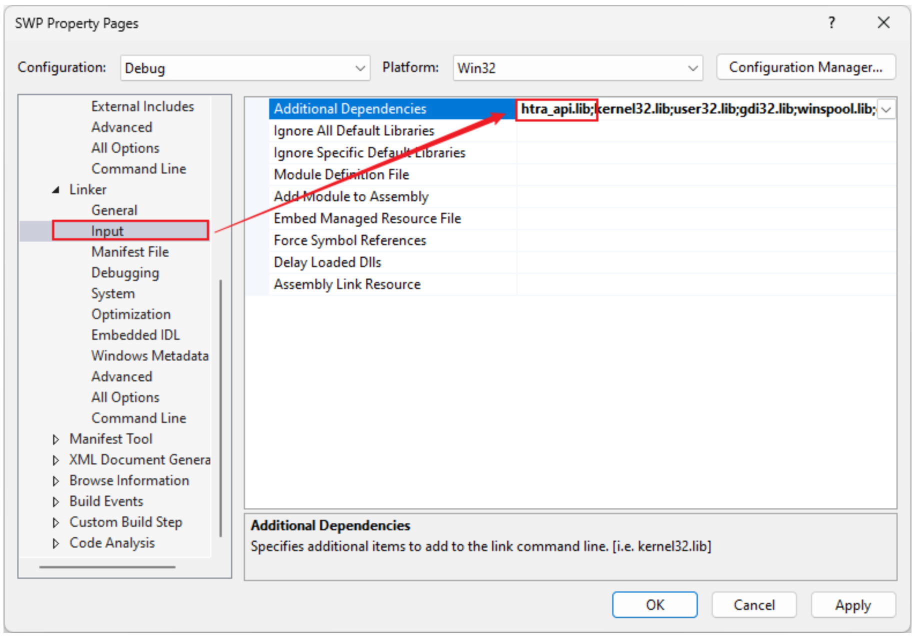
9. Set the Additional Library Directories in Configuration Properties -> Linker -> General to $(SolutionDir)\htra_api.



Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**Yes (/INCREMENTAL)**
Incremental Link Database File	$(IntDir)$(TargetName).ilk
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	$(SolutionDir)\htra_api
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

Additional Library Directories
Allows the user to override the environmental library path. (/LIBPATH:folder)

10. Add htra_api.lib to the Additional Dependencies in Configuration Properties -> Linker -> Input.


SWP Property Pages

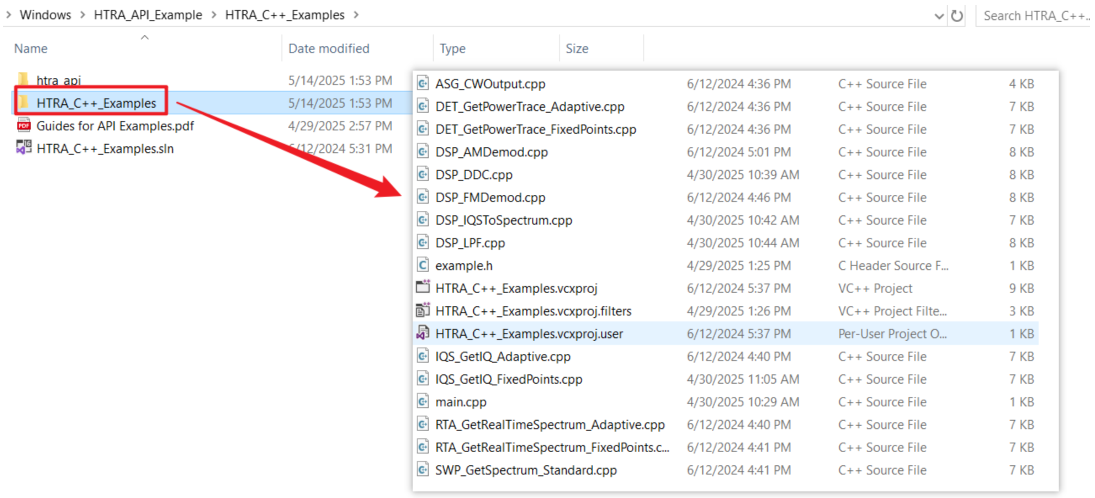
Configuration: Debug   Platform: Win32   Configuration Manager...

- External Includes
- Advanced
- All Options
- Command Line
- Linker
  - General
  - Input
  - Manifest File
  - Debugging
  - System
  - Optimization
  - Embedded IDL
  - Windows Metadata
  - Advanced
  - All Options
  - Command Line
- Manifest Tool
- XML Document Genera
- Browse Information
- Build Events
- Custom Build Step
- Code Analysis

Additional Dependencies	htra_api.lib;kernel32.lib;user32.lib;gdi32.lib;winspool.lib;
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]

11. At this point, the environment configuration is complete, and programming development can begin. You can refer to the C/C++ examples included on the USB drive, specifically in Windows\HTRA_API_Example\HTRA_C++_Examples\HTRA_C++_Examples.


Windows > HTRA_API_Example > HTRA_C++_Examples >   Search HTRA_C++..

Name	Date modified	Type	Size
htra_api	5/14/2025 1:53 PM		
HTRA_C++_Examples	5/14/2025 1:53 PM		
Guides for API Examples.pdf	4/29/2025 2:57 PM		
HTRA_C++_Examples.sln	6/12/2024 5:31 PM		

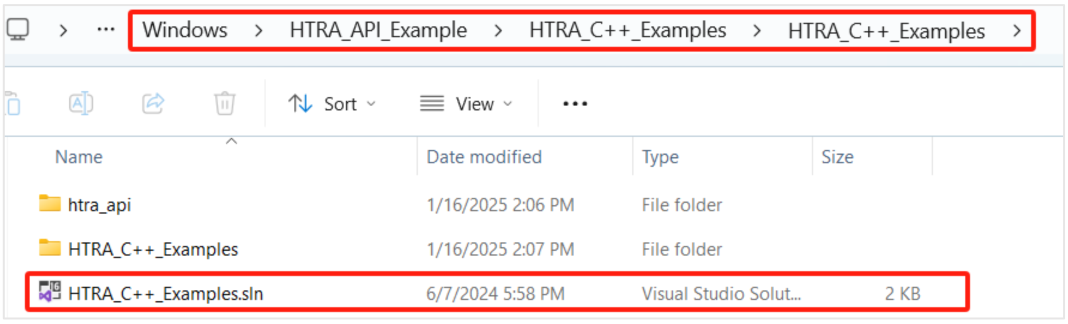
ASG_CWOutput.cpp	6/12/2024 4:36 PM	C++ Source File	4 KB
DET_GetPowerTrace_Adaptive.cpp	6/12/2024 4:36 PM	C++ Source File	7 KB
DET_GetPowerTrace_FixedPoints.cpp	6/12/2024 4:36 PM	C++ Source File	7 KB
DSP_AMDemod.cpp	6/12/2024 5:01 PM	C++ Source File	8 KB
DSP_DDC.cpp	4/30/2025 10:39 AM	C++ Source File	8 KB
DSP_FMDemod.cpp	6/12/2024 4:46 PM	C++ Source File	8 KB
DSP_IQSToSpectrum.cpp	4/30/2025 10:42 AM	C++ Source File	7 KB
DSP_LPF.cpp	4/30/2025 10:44 AM	C++ Source File	8 KB
example.h	4/29/2025 1:25 PM	C Header Source F...	1 KB
HTRA_C++_Examples.vcxproj	6/12/2024 5:37 PM	VC++ Project	9 KB
HTRA_C++_Examples.vcxproj.filters	4/29/2025 1:26 PM	VC++ Project Filte...	3 KB
HTRA_C++_Examples.vcxproj.user	6/12/2024 5:37 PM	Per-User Project O...	1 KB
IQS_GetIQ_Adaptive.cpp	6/12/2024 4:40 PM	C++ Source File	7 KB
IQS_GetIQ_FixedPoints.cpp	4/30/2025 11:05 AM	C++ Source File	7 KB
main.cpp	4/30/2025 10:29 AM	C++ Source File	1 KB
RTA_GetRealTimeSpectrum_Adaptive.cpp	6/12/2024 4:40 PM	C++ Source File	7 KB
RTA_GetRealTimeSpectrum_FixedPoints.c...	6/12/2024 4:41 PM	C++ Source File	7 KB
SWP_GetSpectrum_Standard.cpp	6/12/2024 4:41 PM	C++ Source File	7 KB

1.2 Usage Process for C++ Examples

1.2.1 Usage of General C++ Examples

The usage process for general C++ examples included on the USB drive is as follows:

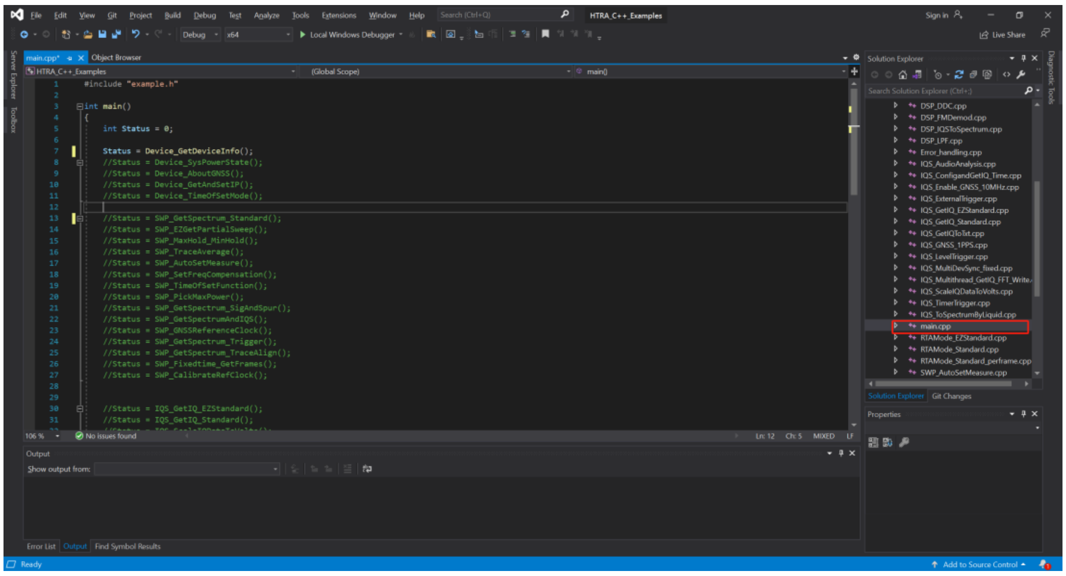
1. Use Visual Studio to open the solution HTRA_C++_Examples.sln located in the folder Windows\HTRA_API_Example\HTRA_C++_Examples on the provided USB drive.



Windows > HTRA_API_Example > HTRA_C++_Examples > HTRA_C++_Examples >

Name	Date modified	Type	Size
htra_api	1/16/2025 2:06 PM	File folder	
HTRA_C++_Examples	1/16/2025 2:07 PM	File folder	
HTRA_C++_Examples.sln	6/7/2024 5:58 PM	Visual Studio Solut...	2 KB

2. Click on the right side to access the HTRA_C++_Examples project and click on the main.cpp file within it.



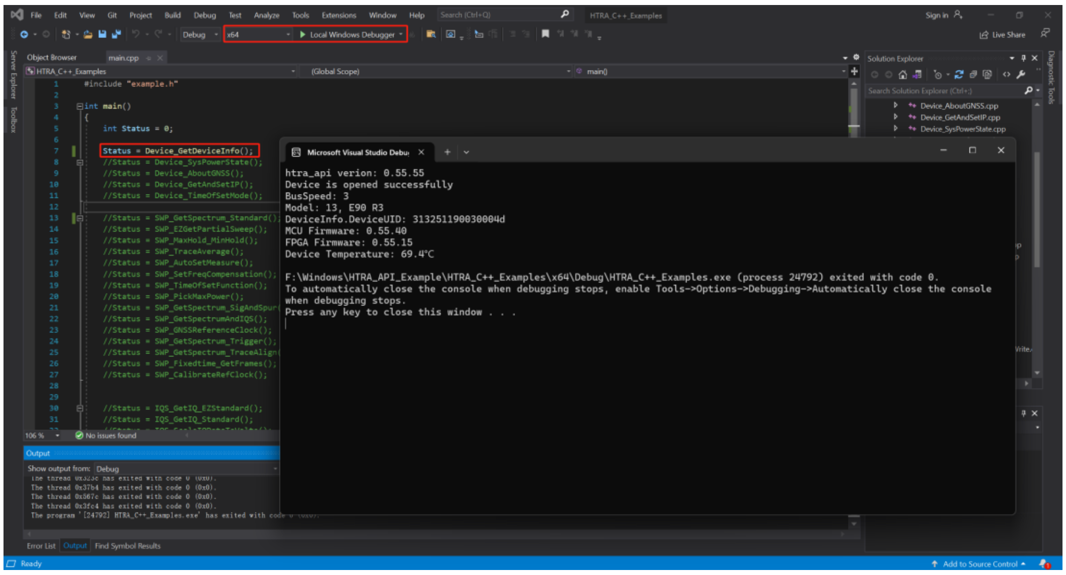
```
#include "example.h"

int main()
{
    int Status = 0;

    Status = Device_GetDeviceInfo();
    //Status = Device_SysPowerState();
    //Status = Device_AboutGNSS();
    //Status = Device_GetAndSetIP();
    //Status = Device_TimeOfSetMode();

    //Status = SWP_GetSpectrum_Standard();
    //Status = SWP_EZGetPartialSweep();
    //Status = SWP_MaxHold_MinHold();
    //Status = SWP_TraceAverage();
    //Status = SWP_AutoSetMeasure();
    //Status = SWP_SetFreqCompensation();
    //Status = SWP_TimeOfSetFunction();
    //Status = SWP_PickMaxPower();
    //Status = SWP_GetSpectrum_SigAndSpur();
    //Status = SWP_GetSpectrumAndIQS();
    //Status = SWP_GNSSReferenceClock();
    //Status = SWP_GetSpectrum_Trigger();
    //Status = SWP_GetSpectrum_TraceAlign();
    //Status = SWP_Fixedtime_GetFrames();
    //Status = SWP_CalibrateRefClock();


    //Status = IQS_GetIQ_EZStandard();
    //Status = IQS_GetIQ_Standard();
```

Solution Explorer

- DSP_DDC.cpp
- DSP_FMDemod.cpp
- DSP_IQSToSpectrum.cpp
- DSP_LPF.cpp
- Error_handling.cpp
- IQS_AudioAnalysis.cpp
- IQS_ConfigandGetIQ_Time.cpp
- IQS_Enable_GNSS_10MHz.cpp
- IQS_ExternalTrigger.cpp
- IQS_GetIQ_EZStandard.cpp
- IQS_GetIQ_Standard.cpp
- IQS_GetIQToTxt.cpp
- IQS_GNSS_1PPS.cpp
- IQS_LevelTrigger.cpp
- IQS_MultiDevSync_fixed.cpp
- IQS_Multithread_GetIQ_FFT_Write
- IQS_ScaleIQDataToVolts.cpp
- IQS_TimerTrigger.cpp
- IQS_ToSpectrumByLiquid.cpp
- main.cpp
- RTAMode_EZStandard.cpp
- RTAMode_Standard.cpp
- RTAMode_Standard_perframe.cpp
- SWP_AutoSetMeasure.cpp

3. Each routine in the C++ example is encapsulated in a separate function. To use the example, simply uncomment it (multiple examples cannot be used simultaneously). For instance, when testing the Device_GetDeviceInfo routine, uncomment it, save, select the expected compilation architecture (both x86 and

x64 are acceptable), and click run. The image shown indicates that the device is running normally.



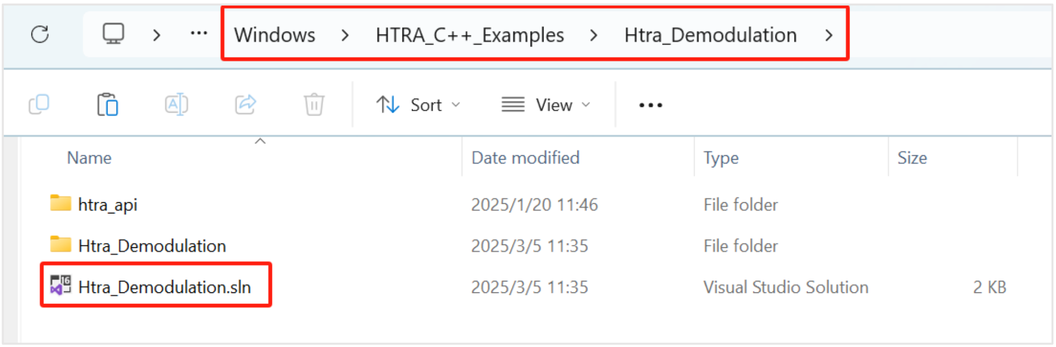
```
#include "example.h"

int main()
{
    int Status = 0;

    Status = Device_GetDeviceInfo();
    //Status = Device_SysPowerState();
    //Status = Device_AboutGNSS();
    //Status = Device_GetAndSetIP();
    //Status = Device_TimeOfSetMode();

    //Status = SWP_GetSpectrum_Standard();
    //Status = SWP_EZGetPartialSweep();
    //Status = SWP_MaxHold_MinHold();
    //Status = SWP_TraceAverage();
    //Status = SWP_AutoSetMeasure();
    //Status = SWP_SetFreqCompensation();
    //Status = SWP_TimeOfSetFunction();
    //Status = SWP_PickMaxPower();
    //Status = SWP_GetSpectrum_SigAndSpur
    //Status = SWP_GetSpectrumAndIQS();
    //Status = SWP_GNSSReferenceClock();
    //Status = SWP_GetSpectrum_Trigger();
    //Status = SWP_GetSpectrum_TraceAlign
    //Status = SWP_Fixedtime_GetFrames();
    //Status = SWP_CalibrateRefClock();


    //Status = IQS_GetIQ_EZStandard();
    //Status = IQS_GetIQ_Standard();
```

```
htra_api verion: 0.55.55
Device is opened successfully
BusSpeed: 3
Model: 13, E90 R3
DeviceInfo.DeviceUID: 313251190030004d
MCU Firmware: 0.55.40
FPGA Firmware: 0.55.15
Device Temperature: 69.4°C

F:\Windows\HTRA_API_Example\HTRA_C++_Examples\x64\Debug\HTRA_C++_Examples.exe (process 24792) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .
```

```
The thread 0x37b4 has exited with code 0 (0x0).
The thread 0x567c has exited with code 0 (0x0).
The thread 0x3fc4 has exited with code 0 (0x0).
The program '[24792] HTRA_C++_Examples.exe' has exited with code
```

1.2.2 Use of the AM/FM demodulation paradigm

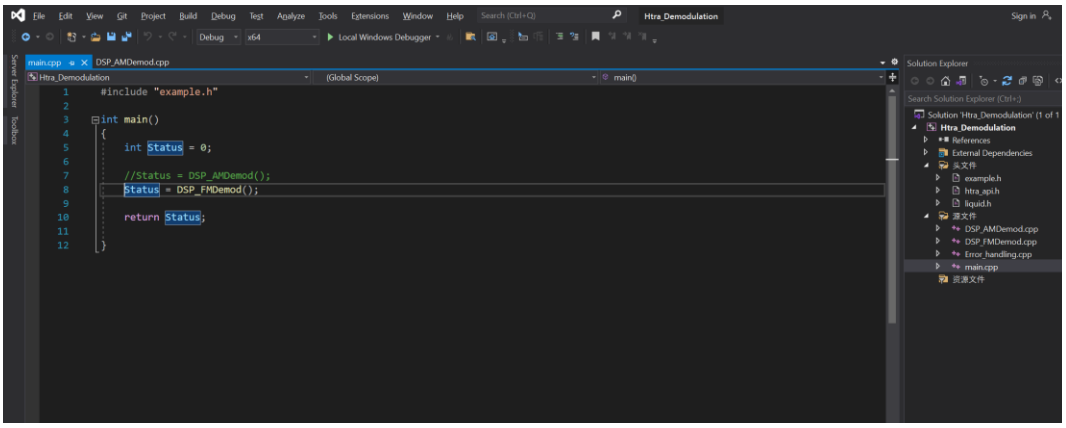
1. Open the solution Htra_Demodulation.sln in the folder Windows\HTRA_C++_Examples\Htra_Demodulation on the supplied USB stick using Visual Studio.



Windows > HTRA_C++_Examples > Htra_Demodulation >

Name	Date modified	Type	Size
htra_api	2025/1/20 11:46	File folder	
Htra_Demodulation	2025/3/5 11:35	File folder	
Htra_Demodulation.sln	2025/3/5 11:35	Visual Studio Solution	2 KB

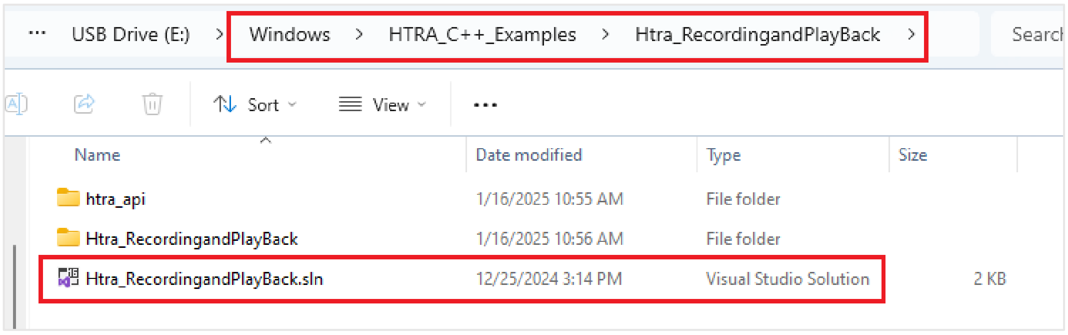
2. Click on the HTRA_C++_Examples project on the right and click on the main.cpp file.
3. C++ send with the example of each routine is encapsulated in a separate function, the use of the example can be uncommented (not at the same time to use more than one example). For example, to test the DSP_FMDemod routine, uncomment it and save it, select the compilation architecture (both x86 and x64), and click run.



```
#include "example.h"

int main()
{
    int Status = 0;

    //Status = DSP_AMDemod();
    Status = DSP_FMDemod();

    return Status;
}
```

1.2.3 Usage of the recording and playback example

1. Open the solution Htra_RecordingandPlayBack.sln located in the folder Windows\HTRA_API_Example\Htra_RecordingandPlayBack on the provided USB drive using Visual Studio.

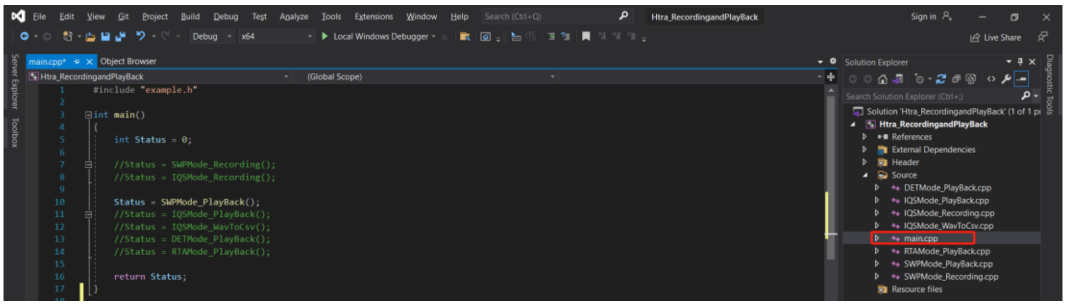


… USB Drive (E:) > Windows > HTRA_C++_Examples > Htra_RecordingandPlayBack >

Name	Date modified	Type	Size
htra_api	1/16/2025 10:55 AM	File folder	
Htra_RecordingandPlayBack	1/16/2025 10:56 AM	File folder	
Htra_RecordingandPlayBack.sln	12/25/2024 3:14 PM	Visual Studio Solution	2 KB

2. Click on the right side to access the Htra_RecordingandPlayBack project and click on the main.cpp file within it.

3. Each routine in the recording and playback example is encapsulated in a separate function. To use the example, simply uncomment it (multiple examples cannot be used simultaneously).

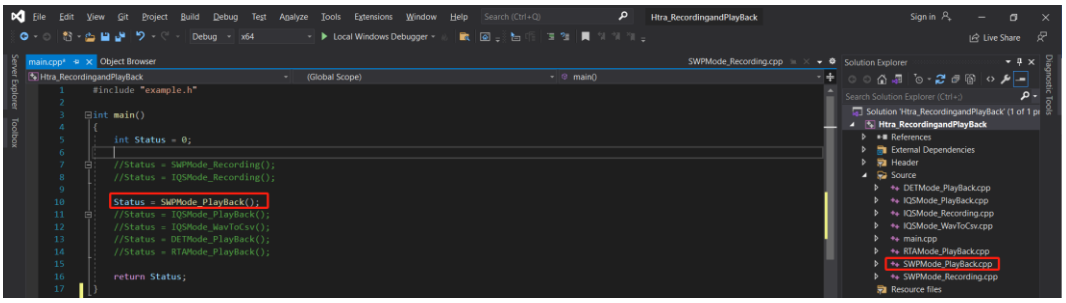


```
#include "example.h"

int main()
{
    int Status = 0;

    //Status = SWPMode_Recording();
    //Status = IQSMode_Recording();

    Status = SWPMode_PlayBack();
    //Status = IQSMode_PlayBack();
    //Status = IQSMode_WavToCsv();
    //Status = DETMode_PlayBack();
    //Status = RTAMode_PlayBack();

    return Status;
}
```

1. Usage of the reading example:

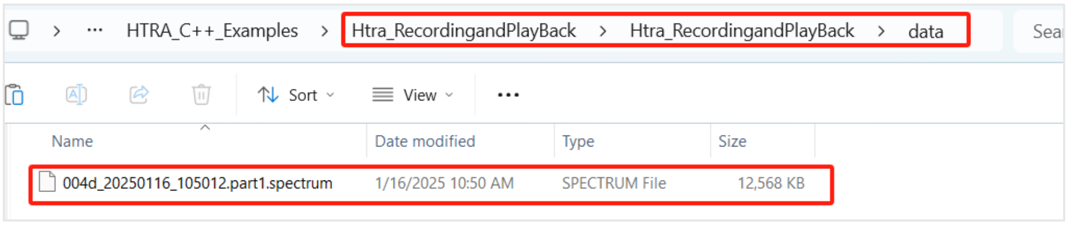
1) For example, when reading SWP mode stream disk data, uncomment the

SWPMode_PlayBack function and save.



```
#include "example.h"

int main()
{
    int Status = 0;

    //Status = SWPMode_Recording();
    //Status = IQSMode_Recording();

    Status = SWPMode_PlayBack();
    //Status = IQSMode_PlayBack();
    //Status = IQSMode_WavToCsv();
    //Status = DETMode_PlayBack();
    //Status = RTAMode_PlayBack();

    return Status;
}
```

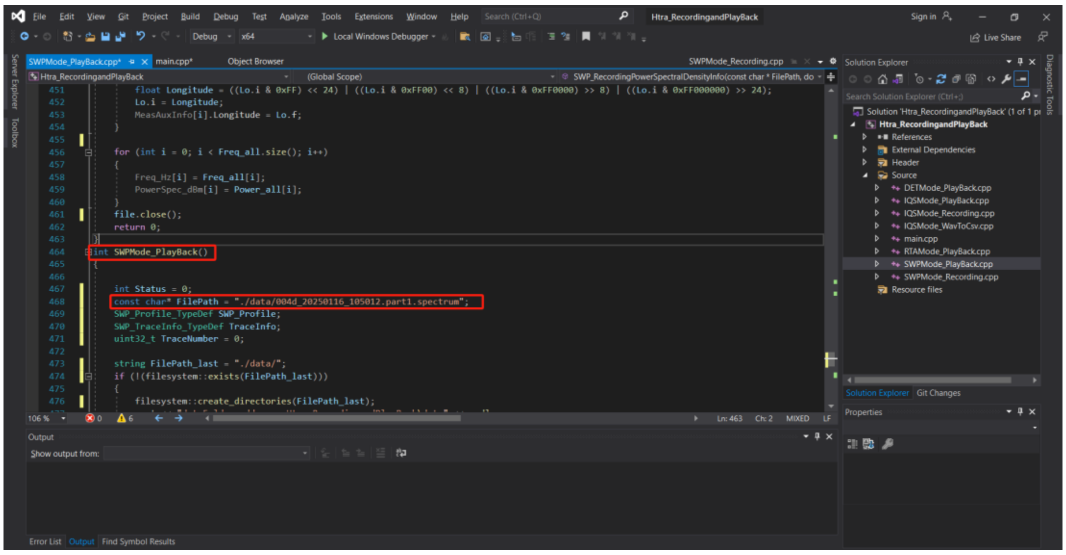
2) Place the recorded file data from SWP mode into the folder "Windows\HTRA_C++_Examples\Htra_RecordingandPlayBack\Htra_RecordingandPlayBack\data".



··· › HTRA_C++_Examples › Htra_RecordingandPlayBack › Htra_RecordingandPlayBack › data

Name	Date modified	Type	Size
004d_20250116_105012.part1.spectrum	1/16/2025 10:50 AM	SPECTRUM File	12,568 KB

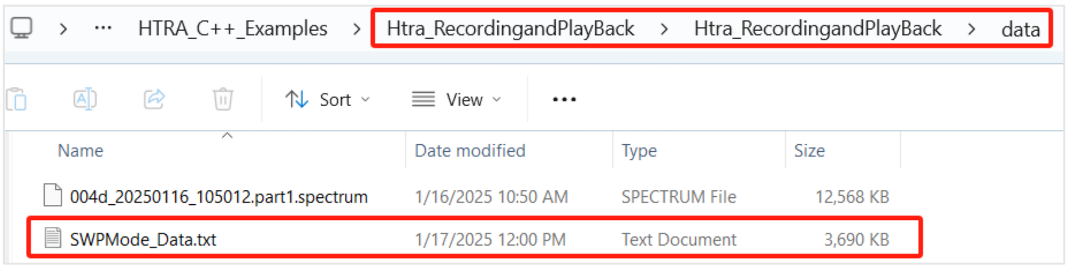
3) Click to enter SWPMode_PlayBack.cpp and modify the name of the recorded file in the SWPMode_PlayBack() function.



```
        float Longitude = ((Lo.i & 0xFF) << 24) | ((Lo.i & 0xFF00) << 8) | ((Lo.i & 0xFF0000) >> 8) | ((Lo.i & 0xFF000000) >> 24);
        Lo.i = Longitude;
        MeasAuxInfo[i].Longitude = Lo.f;
    }

    for (int i = 0; i < Freq_all.size(); i++)
    {
        Freq_Hz[i] = Freq_all[i];
        PowerSpec_dBm[i] = Power_all[i];
    }
    file.close();
    return 0;
}
int SWPMode_PlayBack()
{

    int Status = 0;
    const char* FilePath = "./data/004d_20250116_105012.part1.spectrum";
    SWP_Profile_TypeDef SWP_Profile;
    SWP_TraceInfo_TypeDef TraceInfo;
    uint32_t TraceNumber = 0;

    string FilePath_last = "./data/";
    if (!(filesystem::exists(FilePath_last)))
    {
        filesystem::create_directories(FilePath_last);
```

SWPMode_PlayBack.cpp*
main.cpp*
Object Browser
SWPMode_Recording.cpp
SWP_RecordingPowerSpectralDensityInfo(const char * FilePath, do
Ln: 463 Ch: 2 MIXED LF
Output
Show output from:
Error List Output Find Symbol Results
Solution Explorer Git Changes
Properties

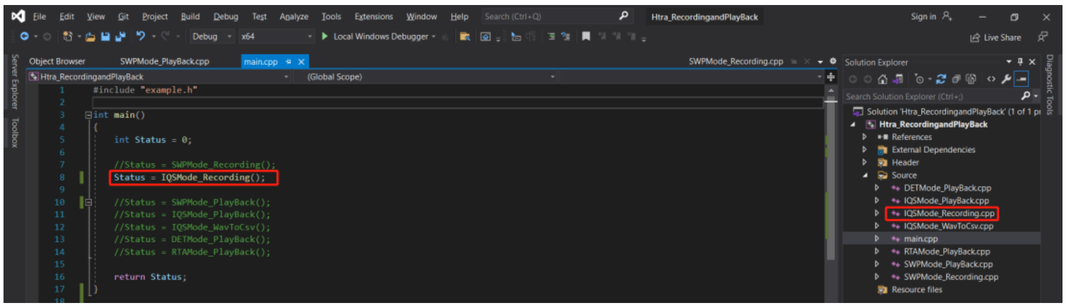
4) Running the program will generate the parsed data file "SWPMode_Data.txt" in the data folder under the SWP mode.


HTRA_C++_Examples > Htra_RecordingandPlayBack > Htra_RecordingandPlayBack > data
Sort
View

Name	Date modified	Type	Size
004d_20250116_105012.part1.spectrum	1/16/2025 10:50 AM	SPECTRUM File	12,568 KB
SWPMode_Data.txt	1/17/2025 12:00 PM	Text Document	3,690 KB

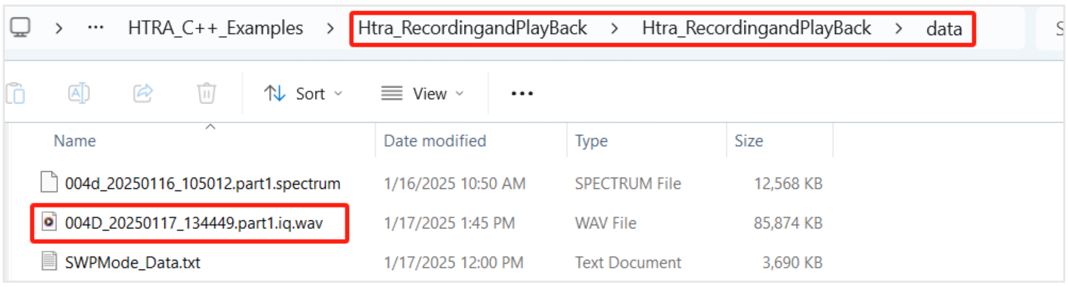
2. Usage of the recording example:

1) For example, when testing the IQSMode_Recording routine, uncomment and save.



```
#include "example.h"

int main()
{
    int Status = 0;

    //Status = SWPMode_Recording();
    Status = IQSMode_Recording();

    //Status = SWPMode_PlayBack();
    //Status = IQSMode_PlayBack();
    //Status = IQSMode_WavToCsv();
    //Status = DETMode_PlayBack();
    //Status = RTAMode_PlayBack();

    return Status;
}
```

2) Click to enter the IQSMode_Recording() function, configure the parameters, and run the program. You can find the recorded file data in the "Windows\HTRA_C++_Examples\Htra_RecordingandPlayBack\Htra_RecordingandPlayBack\data" folder under the IQS mode.


HTRA_C++_Examples > Htra_RecordingandPlayBack > Htra_RecordingandPlayBack > data
Sort
View

Name	Date modified	Type	Size
004d_20250116_105012.part1.spectrum	1/16/2025 10:50 AM	SPECTRUM File	12,568 KB
004D_20250117_134449.part1.iq.wav	1/17/2025 1:45 PM	WAV File	85,874 KB
SWPMode_Data.txt	1/17/2025 12:00 PM	Text Document	3,690 KB

1.3 Device-related

1.3.1 Get device information

Device_GetDeviceInfo.cpp: Retrieves device information, including API version, USB version, device model, device UID, MCU version, FPGA version, and device temperature.

1.3.2 Device standby

Device_SysPowerState.cpp: An example of setting the device's standby state, which can be configured to normal operating state or RF powered down state (low power).

1.3.3 GNSS-related

Device_AboutGNSS.cpp: Retrieves information such as latitude, longitude, altitude, and time obtained from the GNSS module, acquires GNSS-related latitude and time information from MeasAuxInfo in SWP mode, and obtains latitude and time information from IQStream.DeviceState in IQS mode.

1.3.4 Get and modify the IP address of the NX device

Device_GetAndSetIP.cpp: Retrieve the device's IP address and modify it using the device UID or the device's current IP address.

1.3.5 Mode switching time consumption

Device_MeasureModeSwitchTime.cpp: Get the time required for the current host computer to switch between different modes.

1.4 SWP mode

1.4.1 Standard spectrum acquisition

SWP_GetSpectrum_Standard.cpp: Obtain spectrum data by calling the function interface.

1.4.2 Simplified configuration mode

SWP_EZGetPartialSweep.cpp: Quickly acquire spectrum data using a simplified configuration.

1.4.3 Maximum and minimum hold

SWP_MaxHold_MinHold.cpp: Set the trace mode to MaxHold or MinHold, and use SWP_ResetTraceHold to reset the hold.

1.4.4 Average Trace

SWP_TraceAverage.cpp: Average processing of the acquired trace.

1.4.5 Automatic Configuration Measurement

SWP_AutoSetMeasure.cpp: Automatically configures relevant parameters based on specific SWP applications, completing measurements by issuing automatic configuration parameters.

1.4.6 Frequency Compensation

SWP_SetFreqCompensation.cpp : When an external attenuator is present, compensation can be applied to the corresponding frequency band to ensure that the test results remain accurate.

1.4.7 Function execution time, scanning speed, and throughput

SWP_TimeOfSetFunction.cpp:Obtains the call duration of the SWP_Configuration, SWP_GetPartialSweep, and SWP_GetFullSweep functions, along with the scanning speed and throughput under the current configuration.

1.4.8 Obtain spectrum peak values

SWP_PickMaxPower.cpp: Obtain the maximum power point of the current spectrum and its corresponding frequency point.

1.4.9 Signals and Spurious

SWP_GetSpectrum_SigAndSpur.cpp: This can distinguish between signals and spurious after obtaining spectrum data.

1.4.10 Simultaneous Acquisition of Spectrum and IQ

SWP_GetSpectrumAndIQS.cpp: This allows for the simultaneous acquisition of spectrum data and IQ data.

1.4.11 Reading SWP Stream Disk Data from Application Software

SWPMode_PlayBack.cpp: This can read the recorded file data in SWP mode from Application Software and write the read spectrum data into SWP Mode_Data.txt.

1.4.12 Using GNSS 10MHz Reference Clock

SWP_GNSSReferenceClock.cpp: This uses a high-quality GNSS module's 10MHz reference clock in SWP mode.

1.4.13 External Trigger Mode

SWP_GetSpectrum_Trigger.cpp: This obtains spectrum data when the trigger source is set to external trigger.

1.4.14 Trace Alignment Method

SWP_GetSpectrum_TraceAlign.cpp: Obtain spectrum data when the trace alignment method is set to align to the starting frequency or align to the center frequency.

1.4.15 Number of Spectrum Frames Obtainable Within a Certain Time

SWP_Fixedtime_GetFrames.cpp: Loop 50 times to obtain 10 seconds of spectrum data, resulting in the average number of spectrum frames that can be obtained within 10 seconds.

1.4.16 External Trigger Calibration of Internal 10MHz Reference Clock

SWP_GetSpectrum_Calibration.cpp: An example of the device calibrating the clock via GNSS-1PPS or through external trigger.

1.4.17 Phase noise measurement

SWP_Meas_PhaseNoise.cpp: Measuring the single-sideband phase noise (unit: dBc/Hz) of the received signal at the user-specified frequency deviation, supports multi-frequency point configuration.

1.4.18 Channel power measurement

SWP_Meas_ChannelPower.cpp: measures the channel power of the received signal.

1.4.19 Neighbourhood power ratio measurement

SWP_Meas_ACPR.cpp: measures the neighbouring channel power ratio of the received signal.

1.4.20 Percentage occupied bandwidth measurement

SWP_Meas_OBW.cpp: measures the occupied bandwidth of the received signal in percentage.

1.4.21 XdB Occupied Bandwidth Measurement

SWP_Meas_XdBBW.cpp: measures the occupied bandwidth of the received signal in XdB.

1.4.22 IM3 measurements

SWP_Meas_IM3.cpp: IM3 measurement of the received signal.

1.4.23 Frequency interval matching RBW

SWP_RBW_Spaced_Trace.cpp: makes the frequency interval between two points of the trace close to the set RBW value.

1.4.24 Using an External 10MHz Reference Clock

SWP_RefCLKSource_External.cpp: use external 10MHz reference clock. (Take the use of external 10MHz reference clock in SWP mode as an example, the same method is used in other modes).

1.5 IQS Mode

1.5.1 Obtain Fixed Number or Continuous Stream of IQ Data

IQS_GetIQ_Standard.cpp: Obtain a fixed number or continuous stream of IQ data under professional configuration.

1.5.2 Simplified configuration mode

IQS_GetIQ_EZStandard.cpp: Quickly obtain IQ data using a simplified configuration.

1.5.3 IQ data converted to voltage V units

IQS_ScaleIQDataToVolts.cpp: Converts the acquired IQ data into data measured in volts (V).

1.5.4 Time taken to issue configuration and acquire IQ

IQS_ConfigandGetIQ_Time.cpp: Obtains the call duration of the IQS_Configuration and IQS_GetIQStream_PM1 functions.

1.5.5 IQ to Spectrum Data

DSP_IQSToSpectrum.cpp: Converts the acquired time-domain IQ data into spectrum data using spectral analysis methods.

1.5.6 IQ to Spectrum (using liquid library version)

IQS_ToSpectrumByLiquid.cpp: Uses the liquid library to convert the acquired time-domain IQ data into spectrum data through spectral analysis methods.

1.5.7 FM Demodulation

DSP_FMDemod.cpp: Performs FM demodulation on the IQ data and plays the demodulated audio.

1.5.8 AM Demodulation

DSP_AM_Demod.cpp: Performs AM demodulation on IQ data and plays the demodulated audio.

1.5.9 Digital Downconversion

DSP_DDC.cpp: Resamples the obtained IQ data.

1.5.10 Digital Low-Pass Filter

DSP_LPF.cpp: Perform low-pass filtering on the obtained IQ data.

1.5.11 Audio Analysis

IQS_AudioAnalysis.cpp: Performs audio analysis on the demodulated IQ data to obtain audio voltage, audio frequency, signal-to-noise ratio, and total harmonic distortion.

1.5.12 Read the IQS stream disk data from Application Software

IQSMode_PlayBack.cpp: Parses the recorded file data in IQS mode from Application Software and writes the read spectrum data into the IQS Mode_Data.txt file.

1.5.13 Record IQ data in .wav format

IQSMode_Recording.cpp: Stores the acquired IQ data in .wav format.

1.5.14 .wav changed to .csv

IQSMode_WavToCsv.cpp: Parses and extracts I and Q channel data from the .wav recording file data in IQS mode, converting it into a .CSV format file for saving.

1.5.15 Streaming and reading IQ data

IQS_GetIQToTxt.cpp: Writes the obtained IQ data into a .txt file.

1.5.16 Multithreaded acquisition, processing, and streaming of IQ data

IQS_Multithread_GetIQ_FFT_Write: Simultaneously acquires IQ data, performs FFT, and writes IQ data into a .txt file.

1.5.17 GNSS 1PPS trigger

IQS_GNSS_1PPS.cpp: Configures the trigger source to be the 1PPS signal provided by the internal GNSS system.

1.5.18 IQS multi-device synchronization

IQS_MultiDevSync_fixed.cpp: Multiple devices simultaneously collect the same signal at the same time.

1.5.19 External Trigger

IQS_ExternalTrigger.cpp: Configure the trigger source as external trigger.

1.5.20 Timer Trigger

IQS_TimerTrigger.cpp: Configure the trigger source as timer trigger.

1.5.21 Level Trigger(pre-trigger)

IQS_LevelTrigger_PreTriggerr.cpp: configure the trigger source as level trigger and set the pre-trigger time.

1.5.22 Level Trigger (Trigger delay)

IQS_LevelTrigger_TriggerDelay.cpp: configure the trigger source to be level trigger and set the trigger delay time.

1.5.23 Using GNSS 10MHz Reference Clock

IQS_Enable_GNSS_10MHz.cpp: Use a high-quality GNSS module's 10MHz reference clock in IQS mode.

1.5.24 Fixed Step Multi-Frequency IQ Data Acquisition

IQS_SetFreqScan: Configure the start and end frequencies and frequency points of IQ in advance. After configuring once, the data will be collected sequentially according to the set frequency points.

1.6 DET Mode

1.6.1 Obtain detection data for fixed points or continuous streams.

DETMode_Standard.cpp: Obtain detection data for fixed points or continuous streams.

1.6.2 Simplified configuration mode

DETMode_EZStandard.cpp: Quickly obtain detection data through a simplified configuration.

1.6.3 Read the DET stream disk data of Application Software

DETMode_PlayBack.cpp: Read the recorded files in DET mode from Application Software and output the detection data to the DETMode_Data.txt file.

1.6.4 Pulse detection (to be opened later)

1.7 RTA mode

1.7.1 Obtain real-time spectrum data for fixed points or continuous stream

RTAMode_Standard.cpp: Obtain real-time spectrum data for a fixed number of points (duration) or continuous stream.

1.7.2 Simplified configuration mode

RTAMode_EZStandard.cpp: Quickly obtain real-time spectrum data through simple configuration.

1.7.3 Read the RTA stream disk data of Application Software

RTA_Mode_PlayBack.cpp: Read the recorded file data in RTA mode from Application Software, while being able to specify reading a certain packet of data, and write the read spectrum data to the RTAMode_Data.txt file.

1.7.4 Time consumption of each frame of data in RTA mode

RTAMode_Standard_perframe.cpp: Acquire 100 frames of data and calculate the average processing time for each frame.

1.8 ASG Signal Source (optional)

1.8.1 Output single tone/sweep/power scan signals

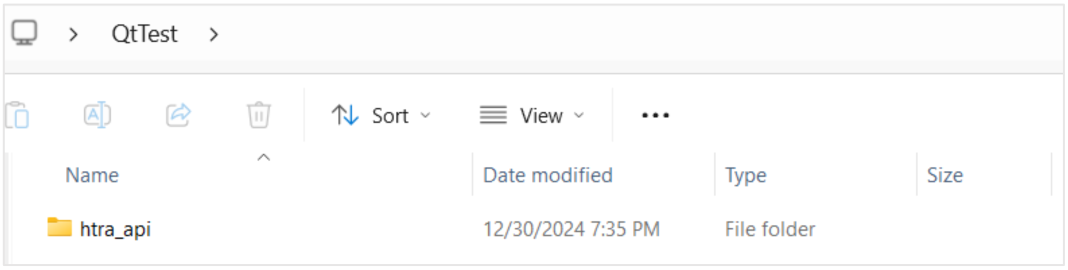
ASG_SignalOutput.cpp: Output single tone/sweep/power scan signals as needed.

2. Digital Demodulation (optional)

3. Qt

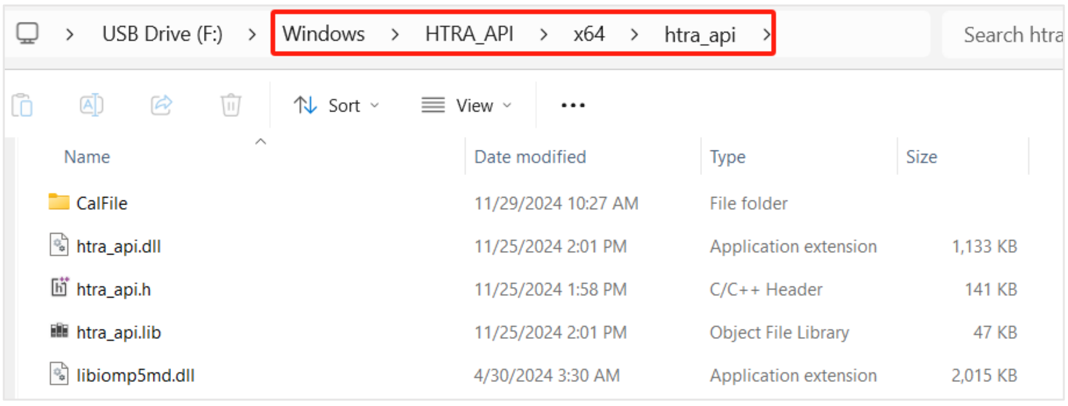
3.1 Configure Development Environment

1. As shown in the figure, first create a new folder to store the entire project (taking QtTest as an example, it is recommended not to use a Chinese path), and then create a htra_api folder within the folder to store the dynamic link libraries and calibration files.

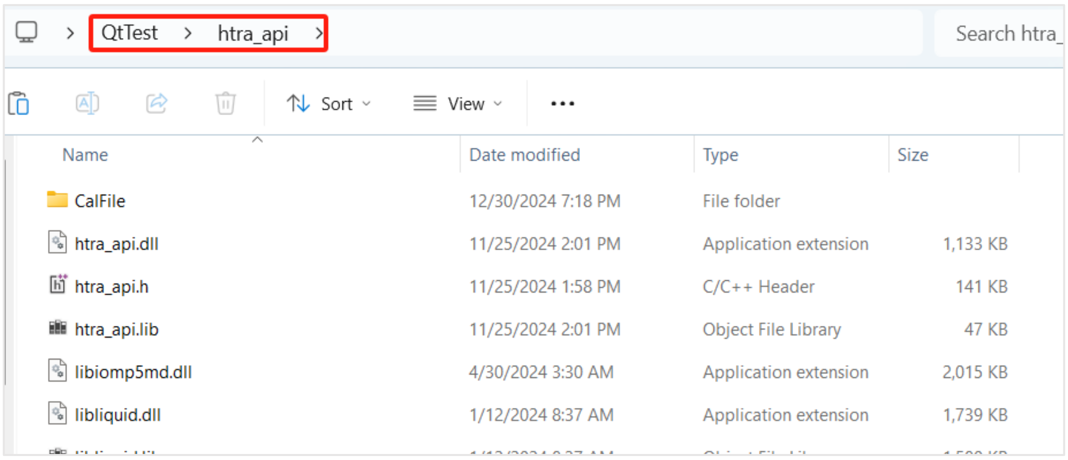


Name	Date modified	Type	Size
htra_api	12/30/2024 7:35 PM	File folder	

2. Copy all files from the Windows\HTRA_API\x64\htra_api folder on the USB drive to the newly created QtTest\htra_api folder (taking the x64 architecture program as an example; for the x86 architecture program, simply copy the corresponding architecture's libraries).

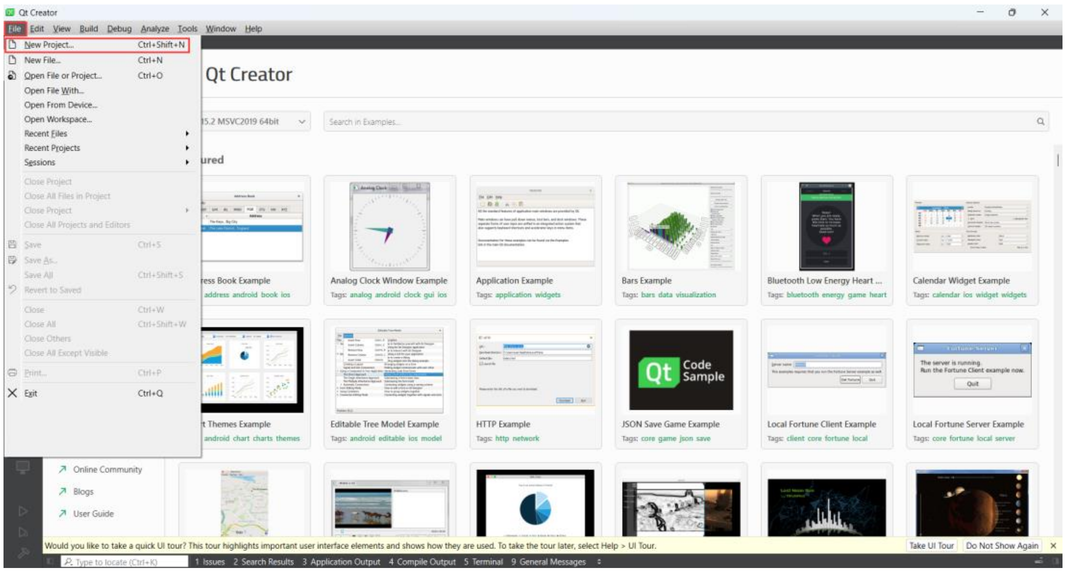


Name	Date modified	Type	Size
CalFile	11/29/2024 10:27 AM	File folder	
htra_api.dll	11/25/2024 2:01 PM	Application extension	1,133 KB
htra_api.h	11/25/2024 1:58 PM	C/C++ Header	141 KB
htra_api.lib	11/25/2024 2:01 PM	Object File Library	47 KB
libiomp5md.dll	4/30/2024 3:30 AM	Application extension	2,015 KB

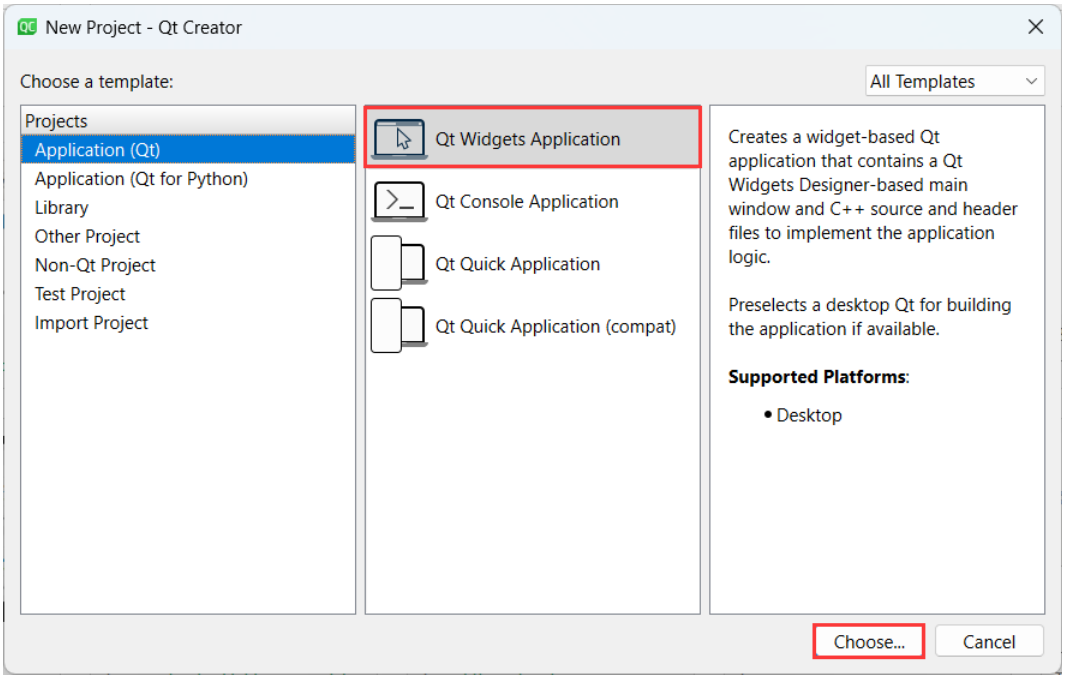


Name	Date modified	Type	Size
CalFile	12/30/2024 7:18 PM	File folder	
htra_api.dll	11/25/2024 2:01 PM	Application extension	1,133 KB
htra_api.h	11/25/2024 1:58 PM	C/C++ Header	141 KB
htra_api.lib	11/25/2024 2:01 PM	Object File Library	47 KB
libiomp5md.dll	4/30/2024 3:30 AM	Application extension	2,015 KB
libliquid.dll	1/12/2024 8:37 AM	Application extension	1,739 KB

3. Open Qt Creator, click on File, and select New File or Project.



4. Select Create Form Application.



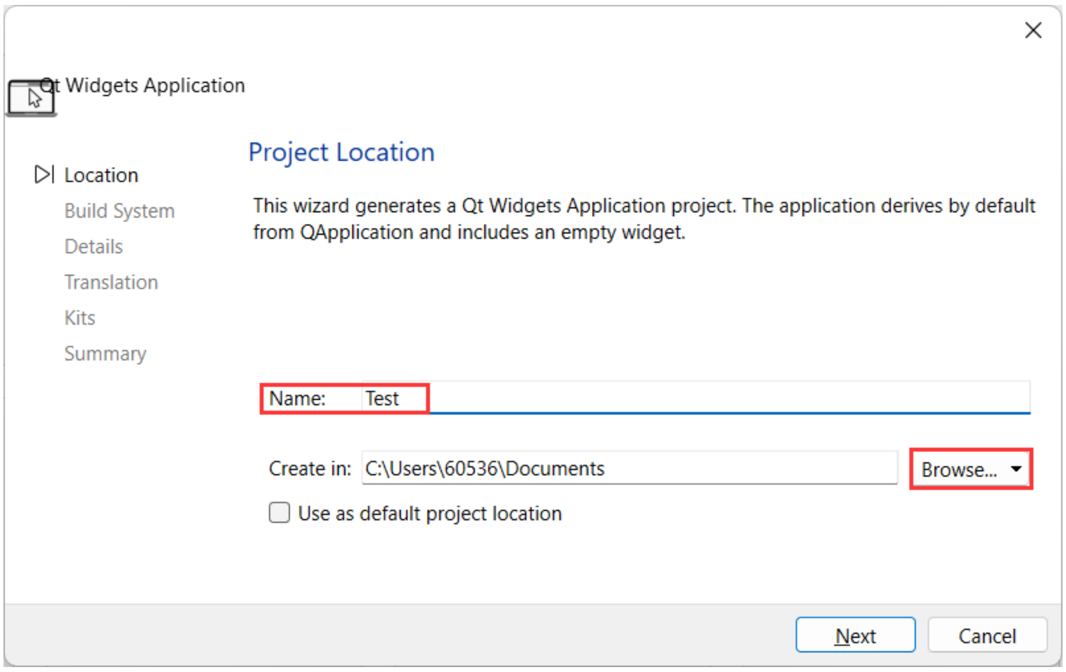
Creates a widget-based Qt application that contains a Qt Widgets Designer-based main window and C++ source and header files to implement the application logic.

Preselects a desktop Qt for building the application if available.

Supported Platforms:

- Desktop

5. After filling in the project name, click Browse to change the project path.



Project Location

This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

Name: Test

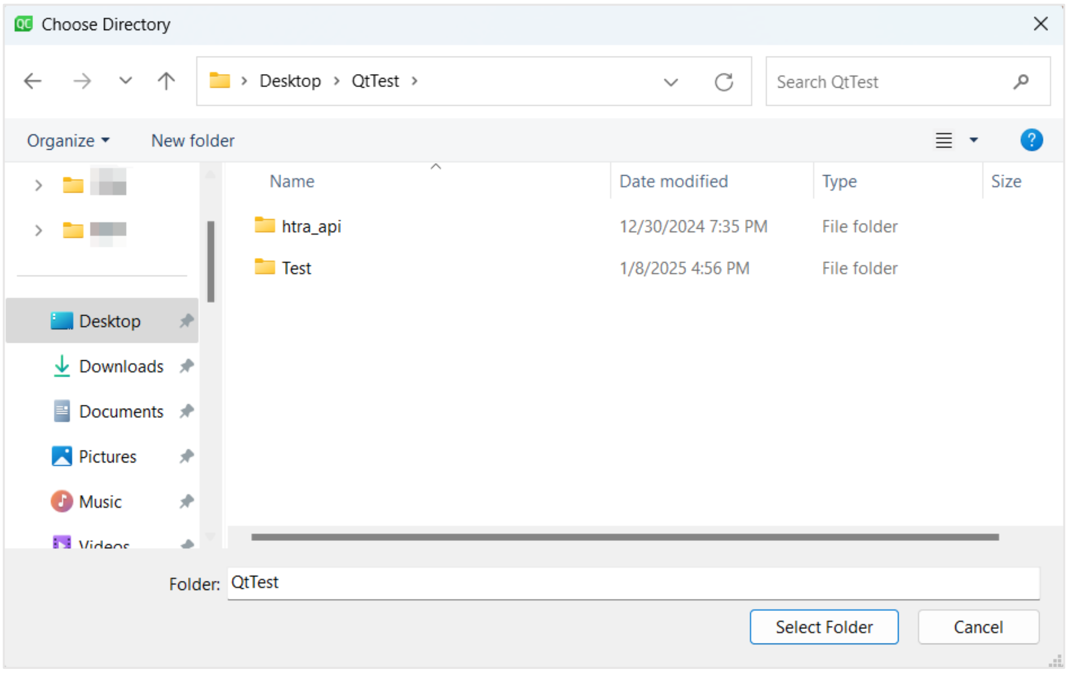
Create in: C:\Users\60536\Documents

Browse...

☐ Use as default project location

Next Cancel

6. Select the directory as the QtTest address created in the first step and click Open.

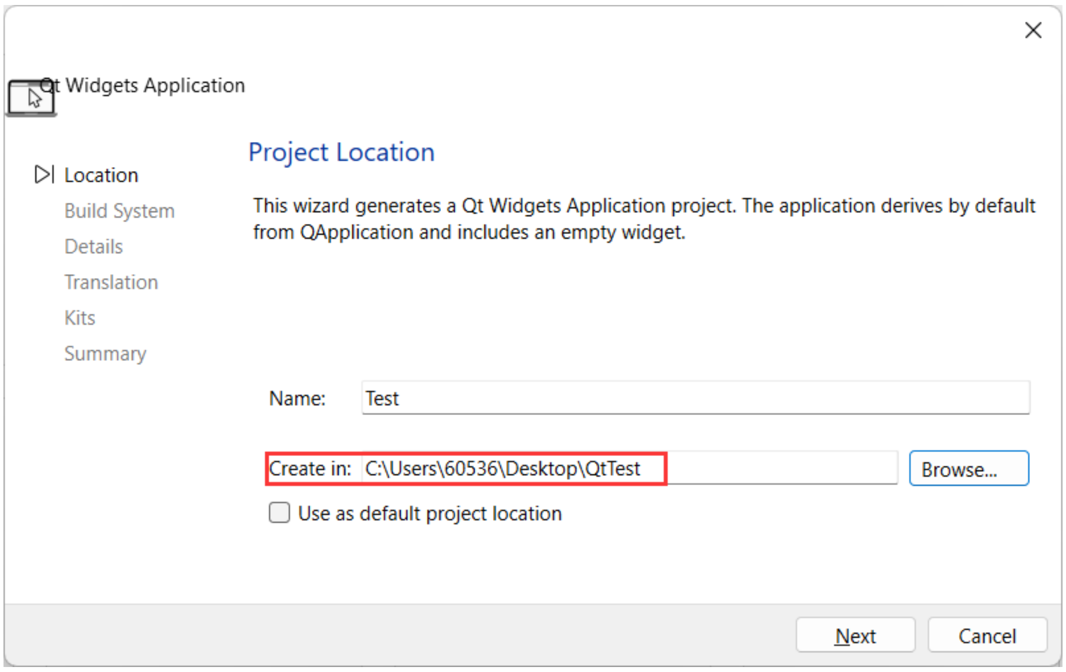


Name	Date modified	Type	Size
htra_api	12/30/2024 7:35 PM	File folder	
Test	1/8/2025 4:56 PM	File folder	

Folder: QtTest

Select Folder Cancel

7. After selecting the path, click Next.


Qt Widgets Application

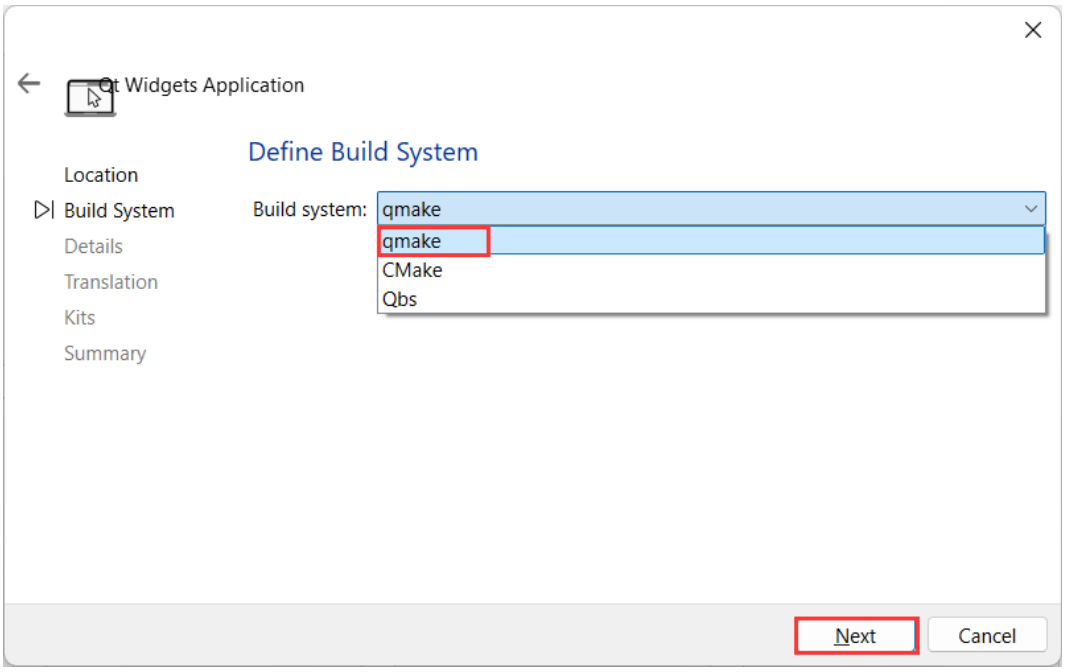
Project Location

This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

Name: Test

Create in: C:\Users\60536\Desktop\QtTest

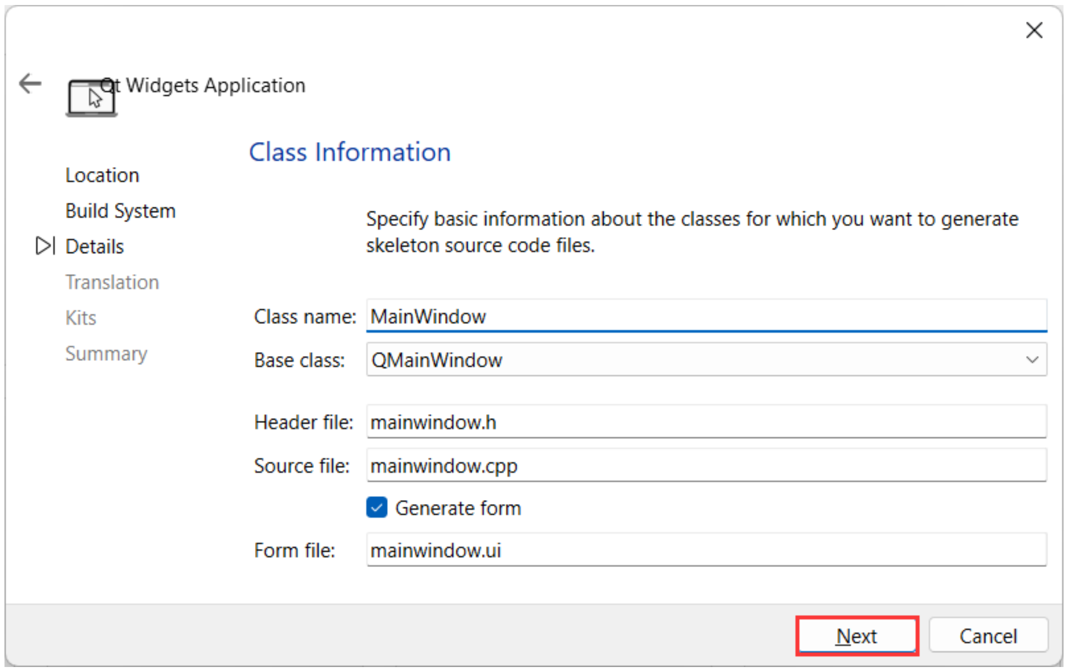
8. Select qmake and click Next to continue.


Qt Widgets Application

Define Build System

Build system: qmake

9. Click Next to continue.


Qt Widgets Application

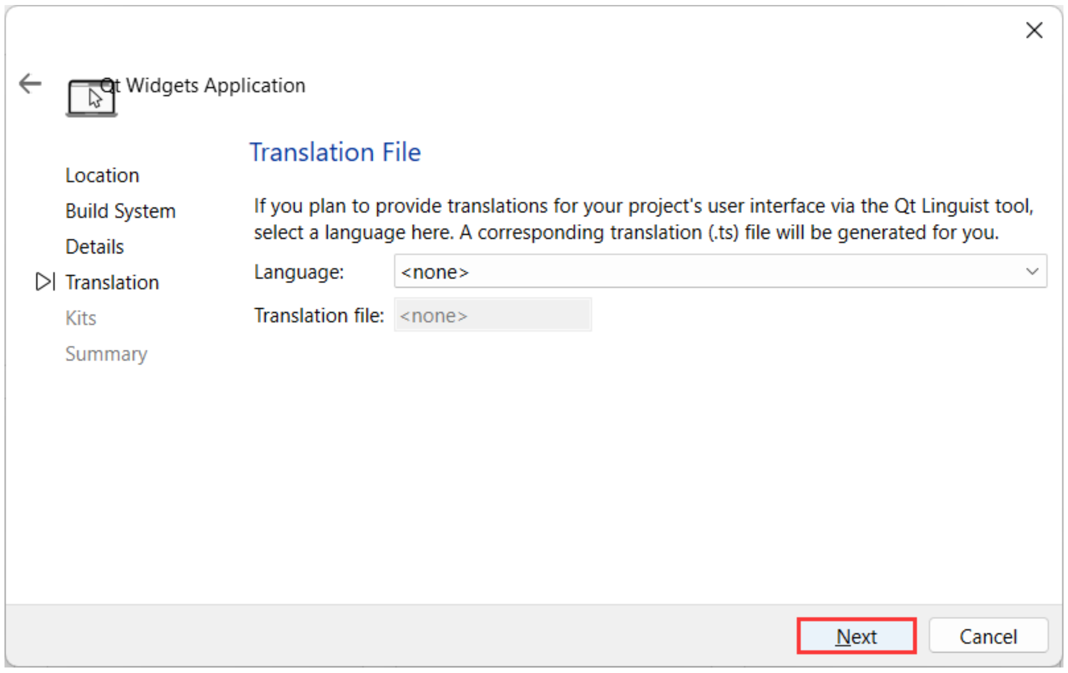
Location
Build System
Details
Translation
Kits
Summary

Class Information

Specify basic information about the classes for which you want to generate skeleton source code files.

Class name: MainWindow
Base class: QMainWindow
Header file: mainwindow.h
Source file: mainwindow.cpp
☑ Generate form
Form file: mainwindow.ui

10. Click Next to continue.


Qt Widgets Application

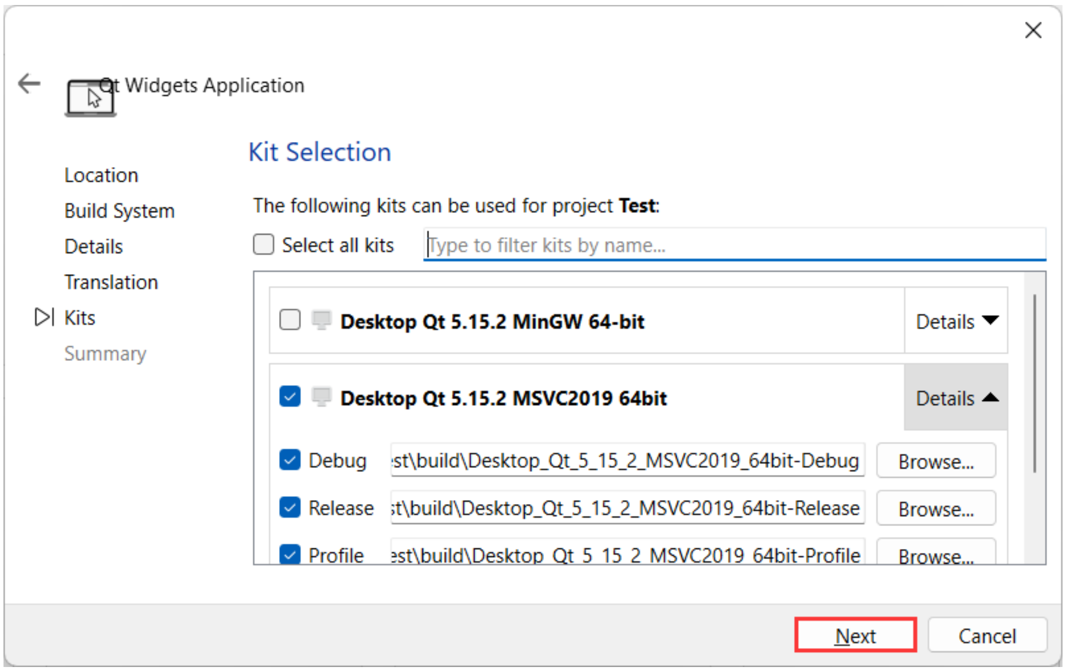
Location
Build System
Details
Translation
Kits
Summary

Translation File

If you plan to provide translations for your project's user interface via the Qt Linguist tool, select a language here. A corresponding translation (.ts) file will be generated for you.

Language: <none>
Translation file: <none>

11. Select a build environment for the project and click Next to continue.


Qt Widgets Application

Location
Build System
Details
Translation
Kits
Summary

Kit Selection

The following kits can be used for project **Test**:

☐ Select all kits Type to filter kits by name...

☐ **Desktop Qt 5.15.2 MinGW 64-bit** Details

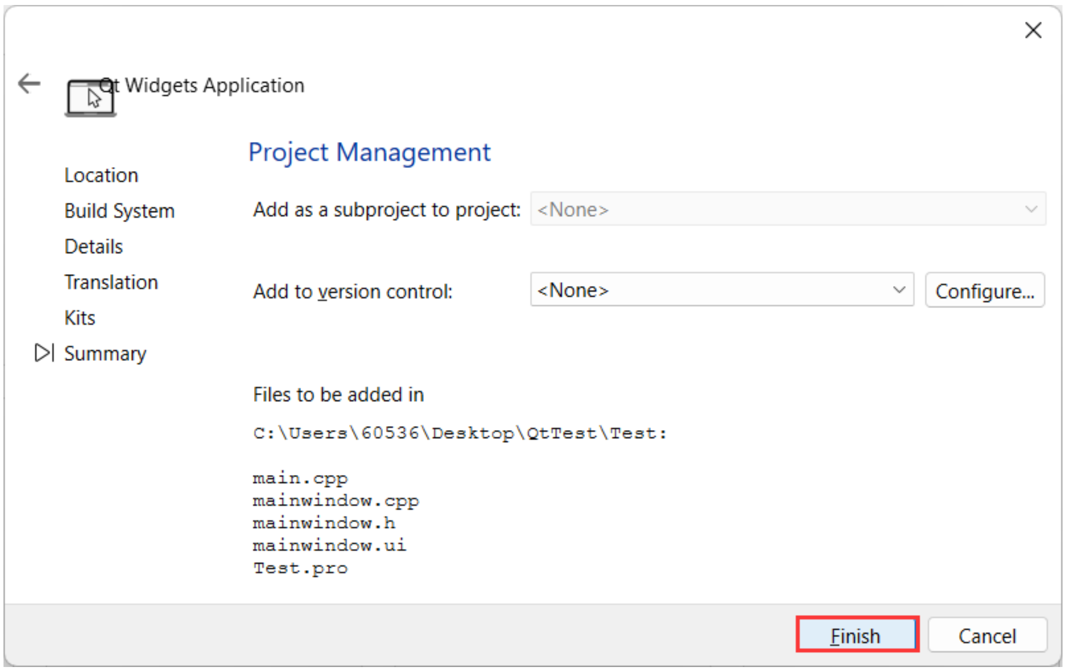
☑ **Desktop Qt 5.15.2 MSVC2019 64bit** Details

☑ Debug st\build\Desktop_Qt_5_15_2_MSVC2019_64bit-Debug Browse...

☑ Release st\build\Desktop_Qt_5_15_2_MSVC2019_64bit-Release Browse...

☑ Profile est\build\Desktop_Qt_5_15_2_MSVC2019_64bit-Profile Browse...

12. Click Finish to create the project.


Qt Widgets Application

Location
Build System
Details
Translation
Kits
Summary

Project Management

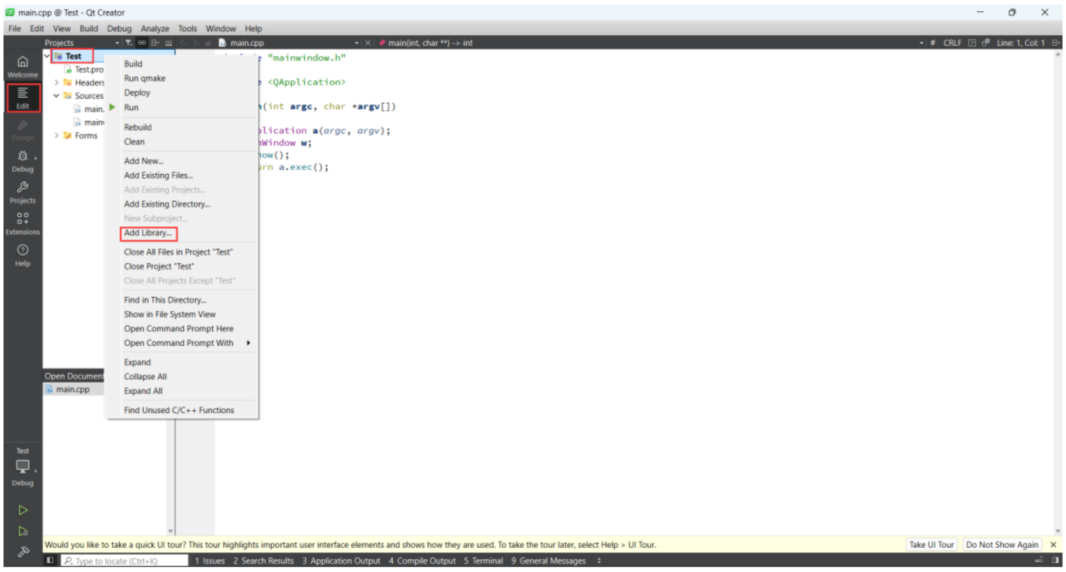
Add as a subproject to project: <None>

Add to version control: <None> Configure...

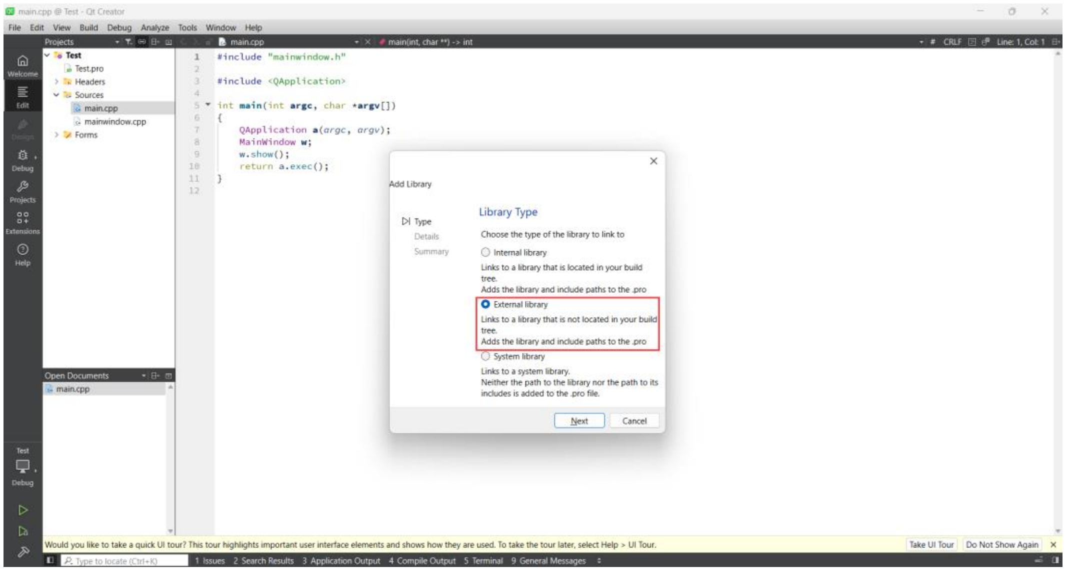
Files to be added in

```
C:\Users\60536\Desktop\QtTest\Test:

main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
Test.pro
```

13. Click Edit, right-click the Test project, and click Add Library.



14. Select External Library and click Next.



```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Add Library

Library Type

Choose the type of the library to link to

Internal library

Links to a library that is located in your build tree.
Adds the library and include paths to the .pro

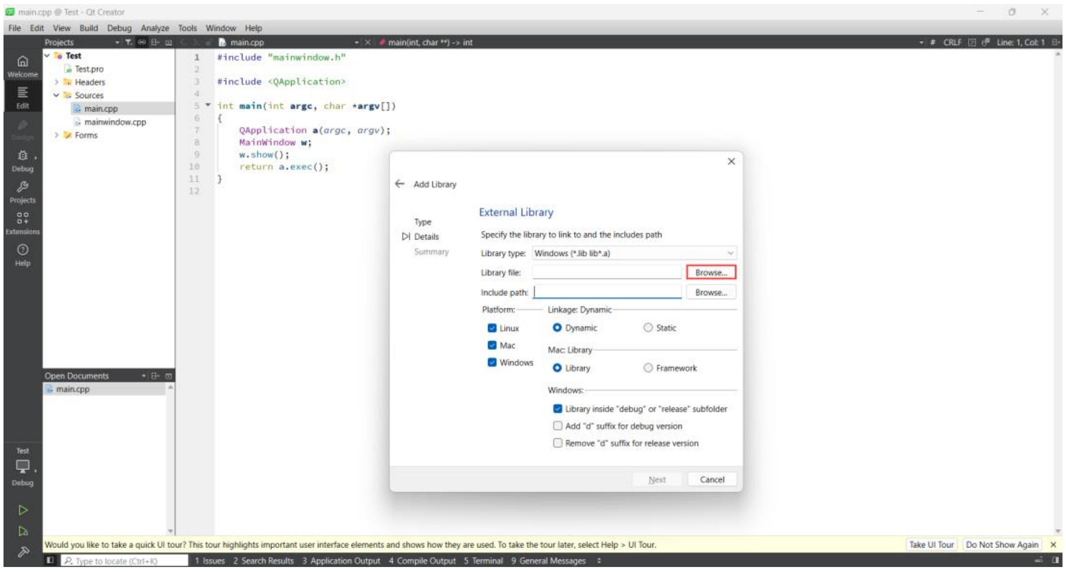
External library

Links to a library that is not located in your build tree.
Adds the library and include paths to the .pro

System library

Links to a system library.
Neither the path to the library nor the path to its includes is added to the .pro file.

15. Click Browse Library File.



Add Library

External Library

Specify the library to link to and the includes path

Library type: Windows (*.lib lib*.a)

Library file:

Include path:

Platform: Linux, Mac, Windows

Linkage: Dynamic — [✓ Dynamic] Static

Mac: Library — [✓ Library] Framework

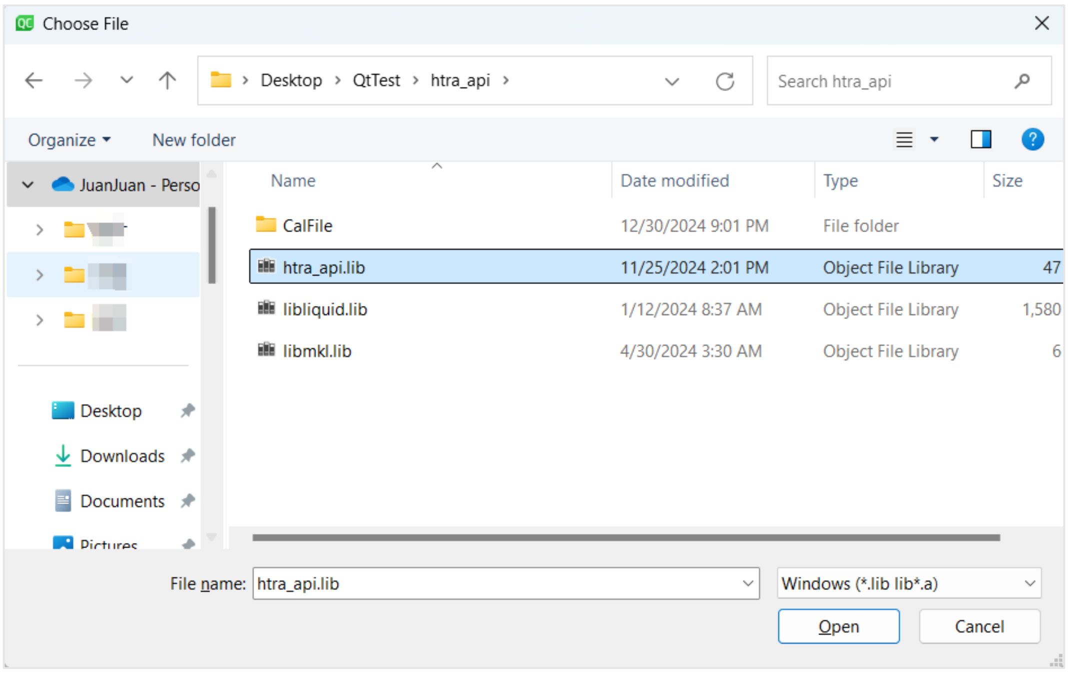
Windows:

[✓] Library inside "debug" or "release" subfolder

[ ] Add "d" suffix for debug version

[ ] Remove "d" suffix for release version

16. Select the htra_api.lib library in QTest\htra_api and click Open.



Choose File

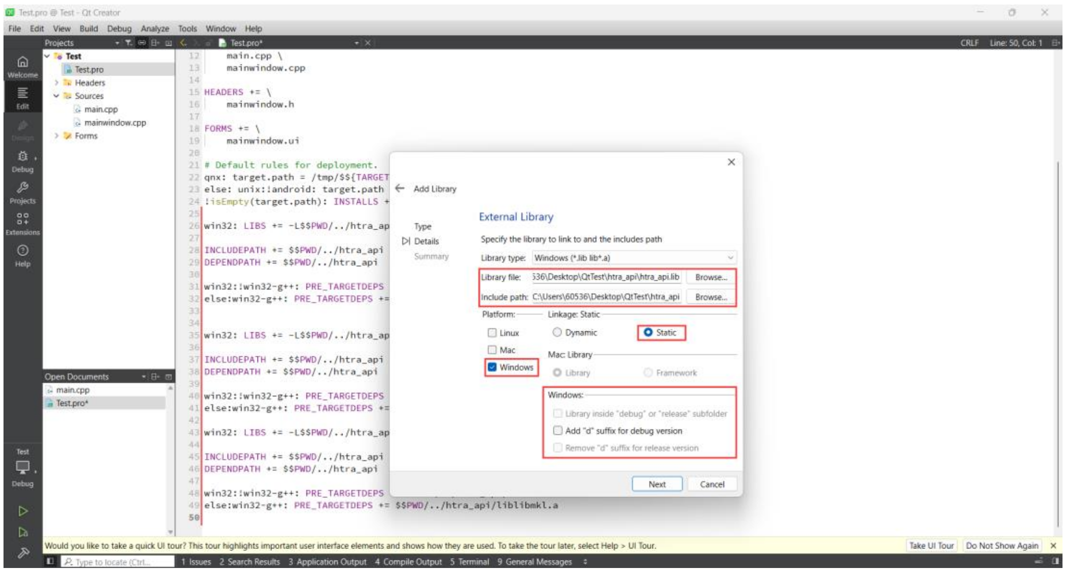
Desktop › QtTest › htra_api ›

Name	Date modified	Type	Size
CalFile	12/30/2024 9:01 PM	File folder	
htra_api.lib	11/25/2024 2:01 PM	Object File Library	47
libliquid.lib	1/12/2024 8:37 AM	Object File Library	1,580
libmkl.lib	4/30/2024 3:30 AM	Object File Library	6

File name: htra_api.lib

Windows (*.lib lib*.a)

17. Uncheck all options in Windows, click on Static Library, select the Windows platform, and click Next.



```
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET
else: unix:!android: target.path
!isEmpty(target.path): INSTALLS +

win32: LIBS += -L$$PWD/../htra_ap

INCLUDEPATH += $$PWD/../htra_api
DEPENDPATH += $$PWD/../htra_api

win32:!win32-g++: PRE_TARGETDEPS
else:win32-g++: PRE_TARGETDEPS +=


win32: LIBS += -L$$PWD/../htra_ap

INCLUDEPATH += $$PWD/../htra_api
DEPENDPATH += $$PWD/../htra_api

win32:!win32-g++: PRE_TARGETDEPS
else:win32-g++: PRE_TARGETDEPS +=

win32: LIBS += -L$$PWD/../htra_ap

INCLUDEPATH += $$PWD/../htra_api
DEPENDPATH += $$PWD/../htra_api

win32:!win32-g++: PRE_TARGETDEPS
else:win32-g++: PRE_TARGETDEPS += $$PWD/../htra_api/liblibmkl.a
```

Add Library

External Library

Specify the library to link to and the includes path

Library type: Windows (*.lib lib*.a)

Library file: 536\Desktop\QtTest\htra_api\htra_api.lib

Include path: C:\Users\60536\Desktop\QtTest\htra_api

Platform: Linux [✓ Windows] Mac

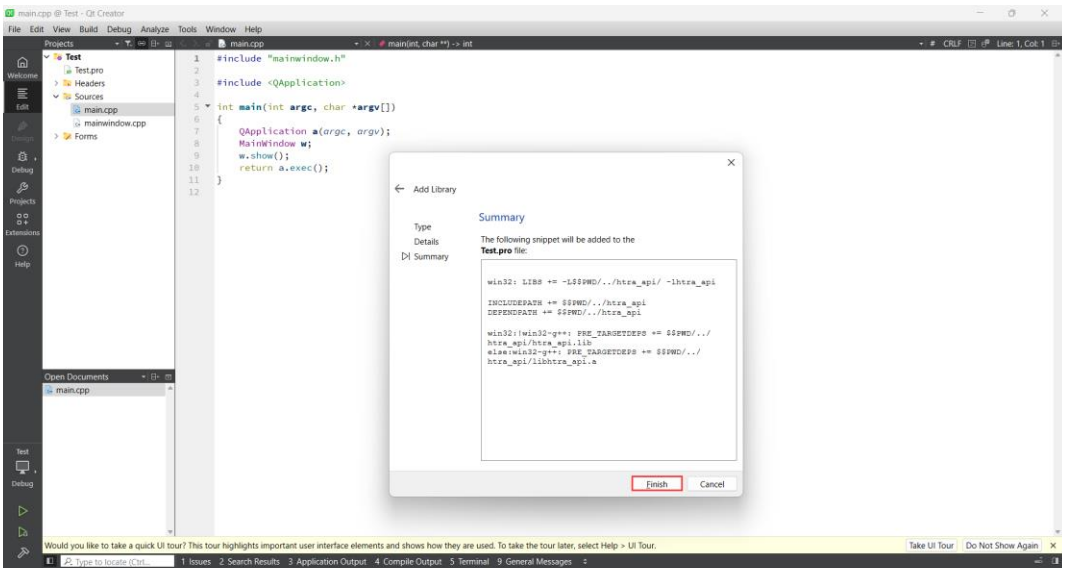
Linkage: Static — Dynamic [✓ Static]

Mac: Library — Library Framework

Windows:
- Library inside "debug" or "release" subfolder
- Add "d" suffix for debug version
- Remove "d" suffix for release version

Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.

18. Click Finish to add the external library.



```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

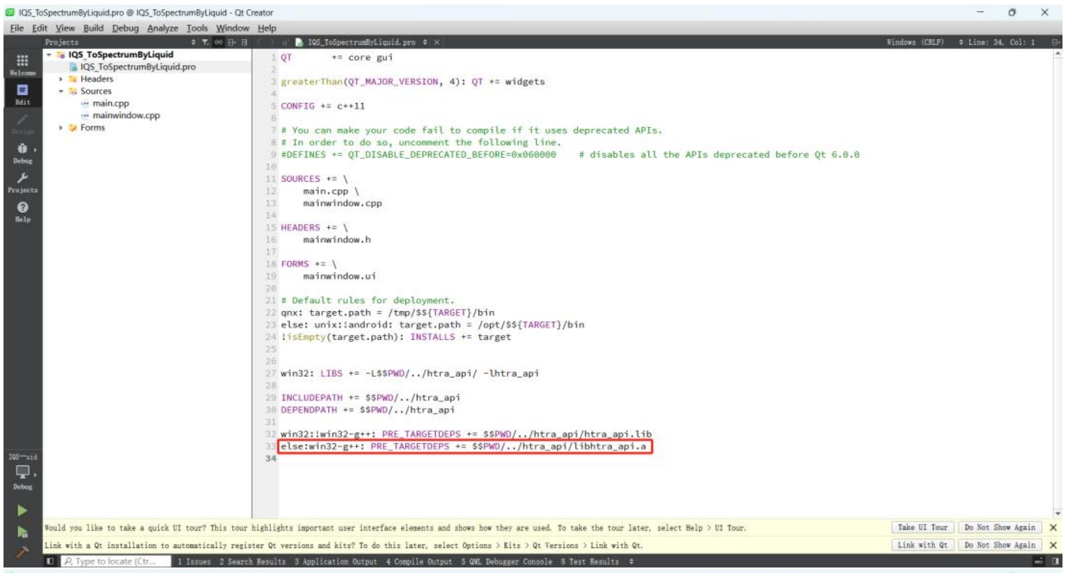
Add Library

Summary

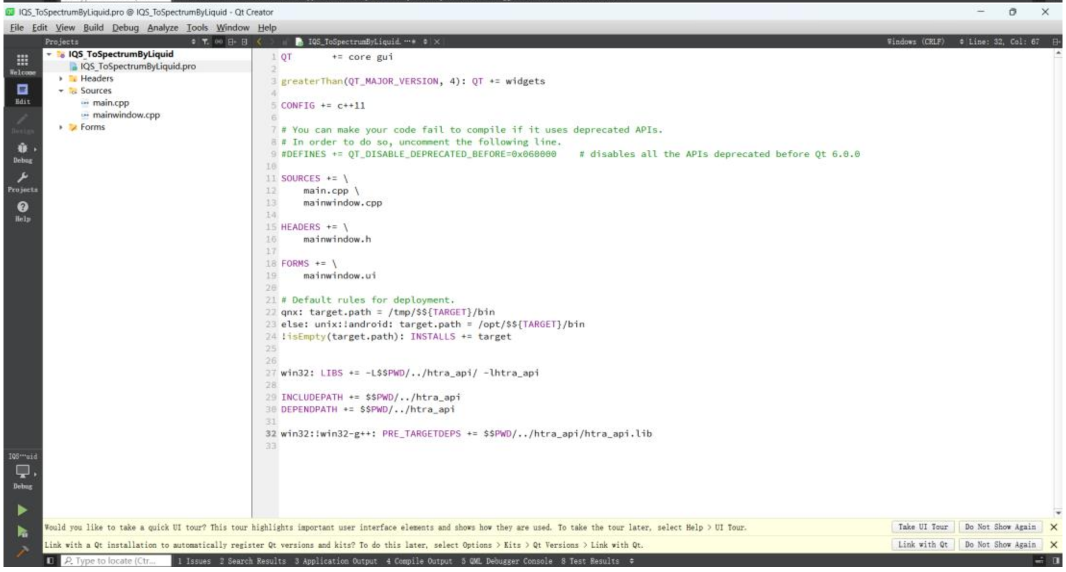
The following snippet will be added to the **Test.pro** file:

```
win32: LIBS += -L$$PWD/../htra_api/ -lhtra_api

INCLUDEPATH += $$PWD/../htra_api
DEPENDPATH += $$PWD/../htra_api

win32:!win32-g++: PRE_TARGETDEPS += $$PWD/../
htra_api/htra_api.lib
else:win32-g++: PRE_TARGETDEPS += $$PWD/../
htra_api/libhtra_api.a
```

19. Delete the last line "else: win32-g++: PRE_TARGTDEPS+=$$PWD/...". /htra_api/libhtra_api.a".

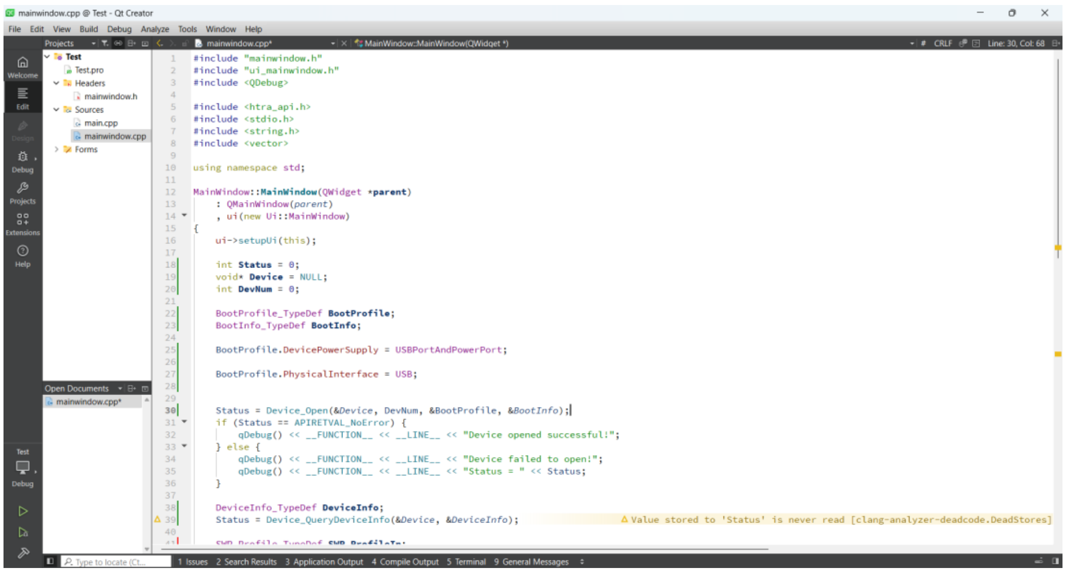


```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target


win32: LIBS += -L$$PWD/../htra_api/ -lhtra_api

INCLUDEPATH += $$PWD/../htra_api
DEPENDPATH += $$PWD/../htra_api

win32:!win32-g++: PRE_TARGETDEPS += $$PWD/../htra_api/htra_api.lib
else:win32-g++: PRE_TARGETDEPS += $$PWD/../htra_api/libhtra_api.a
```



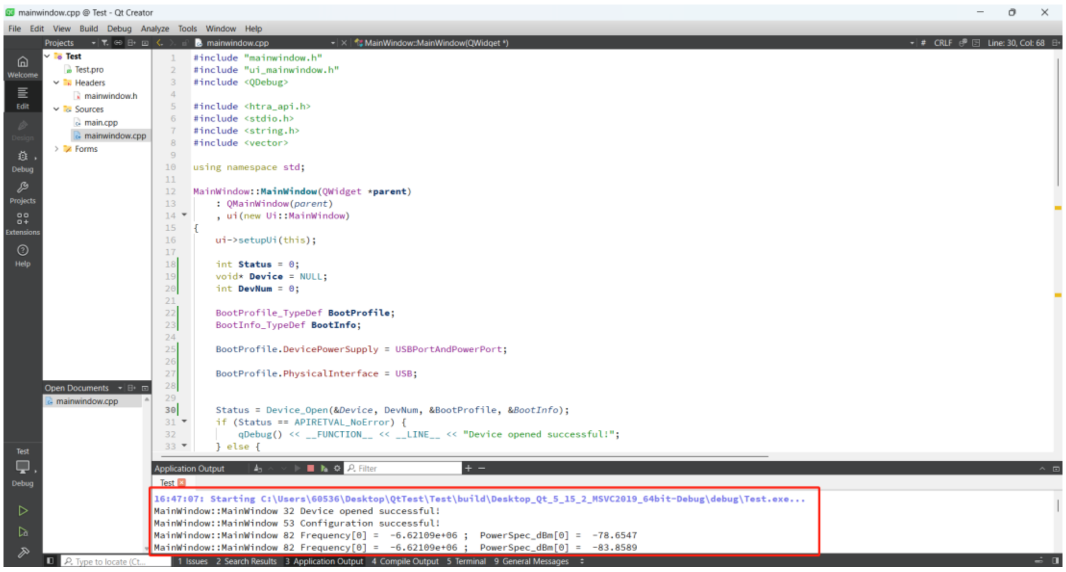
```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

win32: LIBS += -L$$PWD/../htra_api/ -lhtra_api

INCLUDEPATH += $$PWD/../htra_api
DEPENDPATH += $$PWD/../htra_api

win32:!win32-g++: PRE_TARGETDEPS += $$PWD/../htra_api/htra_api.lib
```

20. Save the Test.pro file, and then you can write the code normally.



```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>

#include <htra_api.h>
#include <stdio.h>
#include <string.h>
#include <vector>

using namespace std;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    int Status = 0;
    void* Device = NULL;
    int DevNum = 0;

    BootProfile_TypeDef BootProfile;
    BootInfo_TypeDef BootInfo;

    BootProfile.DevicePowerSupply = USBPortAndPowerPort;

    BootProfile.PhysicalInterface = USB;


    Status = Device_Open(&Device, DevNum, &BootProfile, &BootInfo);
    if (Status == APIRETVAL_NoError) {
        qDebug() << __FUNCTION__ << __LINE__ << "Device opened successful!";
    } else {
        qDebug() << __FUNCTION__ << __LINE__ << "Device failed to open!";
        qDebug() << __FUNCTION__ << __LINE__ << "Status = " << Status;
    }

    DeviceInfo_TypeDef DeviceInfo;
    Status = Device_QueryDeviceInfo(&Device, &DeviceInfo);          ⚠ Value stored to 'Status' is never read [clang-analyzer-deadcode.DeadStores]
```

21. After writing the code, click Run. The device should function normally as shown in the figure.



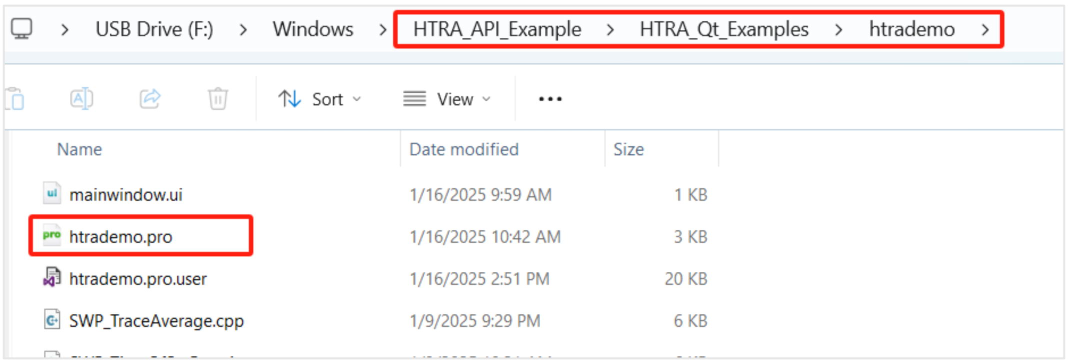
```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>

#include <htra_api.h>
#include <stdio.h>
#include <string.h>
#include <vector>

using namespace std;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    int Status = 0;
    void* Device = NULL;
    int DevNum = 0;

    BootProfile_TypeDef BootProfile;
    BootInfo_TypeDef BootInfo;

    BootProfile.DevicePowerSupply = USBPortAndPowerPort;

    BootProfile.PhysicalInterface = USB;


    Status = Device_Open(&Device, DevNum, &BootProfile, &BootInfo);
    if (Status == APIRETVAL_NoError) {
        qDebug() << __FUNCTION__ << __LINE__ << "Device opened successful!";
    } else {
```

```
16:47:07: Starting C:\Users\60536\Desktop\QtTest\Test\build\Desktop_Qt_5_15_2_MSVC2019_64bit-Debug\debug\Test.exe...
MainWindow::MainWindow 32 Device opened successful!
MainWindow::MainWindow 53 Configuration successful!
MainWindow::MainWindow 82 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -78.6547
MainWindow::MainWindow 82 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -83.8589
```

3.2 Qt Example Usage Process

The usage process for the Qt examples included in the USB drive is as follows:

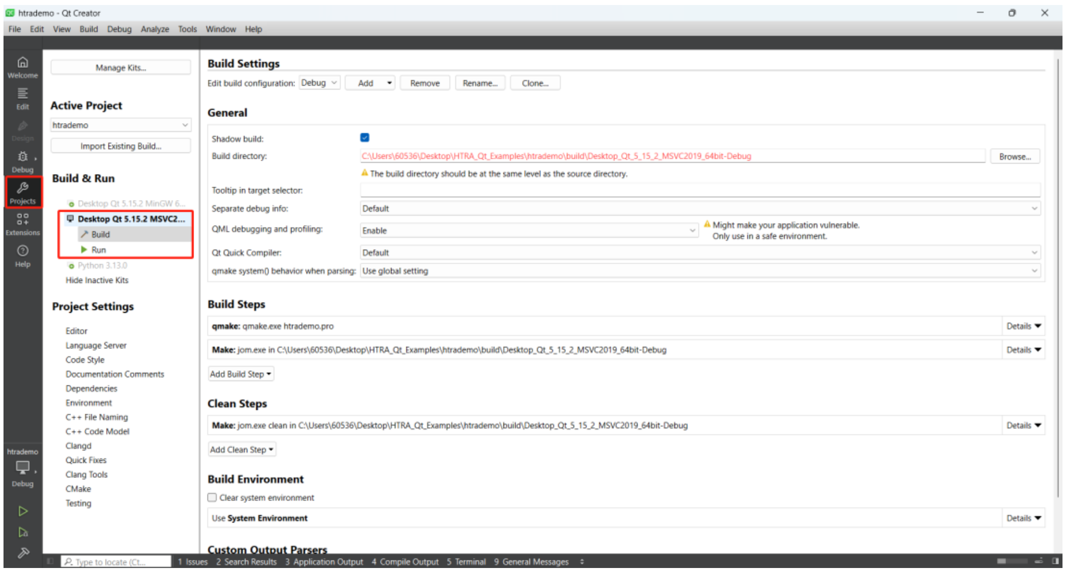
1. As shown in the figure, use Qt Creator to open the htrademo.pro file located in the Windows\HTRA_API_Example\HTRA_Qt_Examples\htrademo folder on the USB drive (please ensure the project path does not contain Chinese characters).



USB Drive (F:) > Windows > HTRA_API_Example > HTRA_Qt_Examples > htrademo >
Sort
View

Name	Date modified	Size
mainwindow.ui	1/16/2025 9:59 AM	1 KB
htrademo.pro	1/16/2025 10:42 AM	3 KB
htrademo.pro.user	1/16/2025 2:51 PM	20 KB
SWP_TraceAverage.cpp	1/9/2025 9:29 PM	6 KB

2. Click on the project to configure a build environment for it.



Build Settings

Edit build configuration: Debug

General

Shadow build: ☑

Build directory: C:\Users\60536\Desktop\HTRA_Qt_Examples\htrademo\build\Desktop_Qt_5_15_2_MSVC2019_64bit-Debug

⚠ The build directory should be at the same level as the source directory.

Tooltip in target selector:

Separate debug info: Default

QML debugging and profiling: Enable ⚠ Might make your application vulnerable. Only use in a safe environment.

Qt Quick Compiler: Default

qmake system() behavior when parsing: Use global setting

Build Steps

qmake: qmake.exe htrademo.pro

Make: jom.exe in C:\Users\60536\Desktop\HTRA_Qt_Examples\htrademo\build\Desktop_Qt_5_15_2_MSVC2019_64bit-Debug

Clean Steps

Make: jom.exe clean in C:\Users\60536\Desktop\HTRA_Qt_Examples\htrademo\build\Desktop_Qt_5_15_2_MSVC2019_64bit-Debug

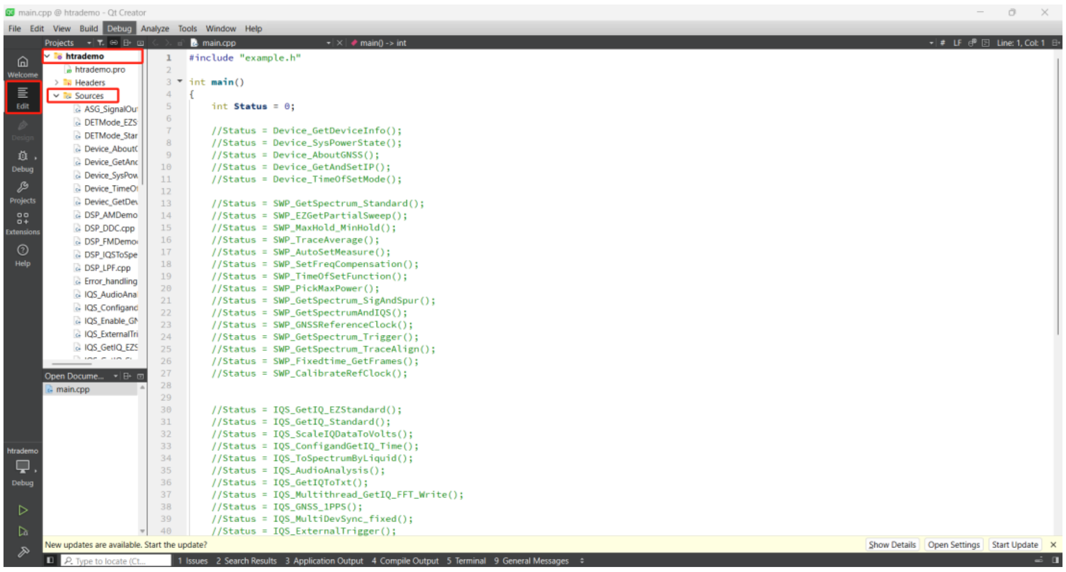
Build Environment

☐ Clear system environment

Use **System Environment**

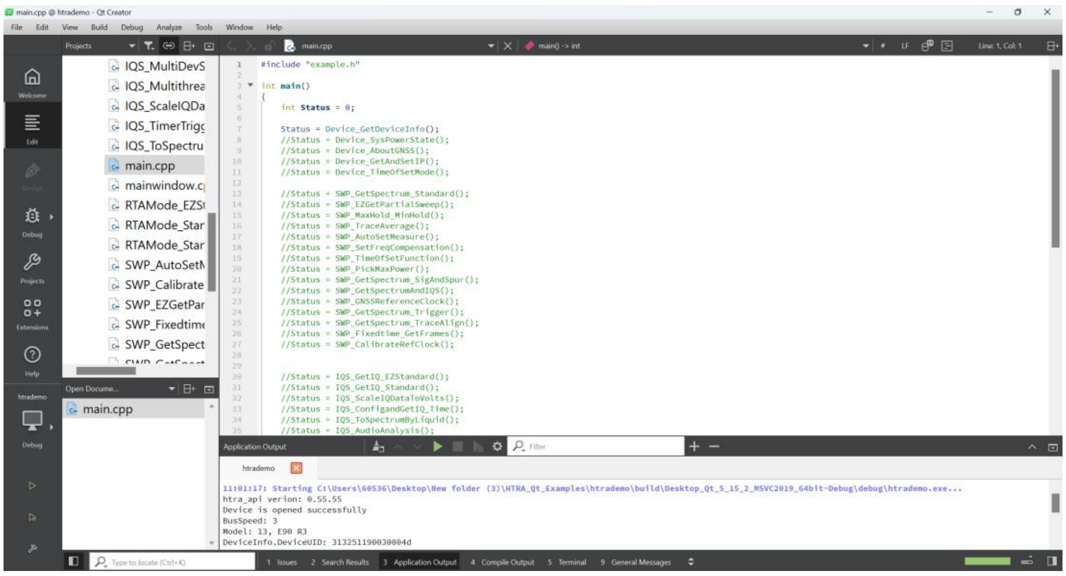
Custom Output Parsers

3. After configuring the build environment, click on Edit, and then click on main.cpp in the Sources folder of the htrademo project.



```
#include "example.h"

int main()
{
    int Status = 0;

    //Status = Device_GetDeviceInfo();
    //Status = Device_SysPowerState();
    //Status = Device_AboutGNSS();
    //Status = Device_GetAndSetIP();
    //Status = Device_TimeOfSetMode();

    //Status = SWP_GetSpectrum_Standard();
    //Status = SWP_EZGetPartialSweep();
    //Status = SWP_MaxHold_MinHold();
    //Status = SWP_TraceAverage();
    //Status = SWP_AutoSetMeasure();
    //Status = SWP_SetFreqCompensation();
    //Status = SWP_TimeOfSetFunction();
    //Status = SWP_PickMaxPower();
    //Status = SWP_GetSpectrum_SigAndSpur();
    //Status = SWP_GetSpectrumAndIQS();
    //Status = SWP_GNSSReferenceClock();
    //Status = SWP_GetSpectrum_Trigger();
    //Status = SWP_GetSpectrum_TraceAlign();
    //Status = SWP_Fixedtime_GetFrames();
    //Status = SWP_CalibrateRefClock();


    //Status = IQS_GetIQ_EZStandard();
    //Status = IQS_GetIQ_Standard();
    //Status = IQS_ScaleIQDataToVolts();
    //Status = IQS_ConfigandGetIQ_Time();
    //Status = IQS_ToSpectrumByLiquid();
    //Status = IQS_AudioAnalysis();
    //Status = IQS_GetIQToTxt();
    //Status = IQS_Multithread_GetIQ_FFT_Write();
    //Status = IQS_GNSS_1PPS();
    //Status = IQS_MultiDevSync_fixed();
    //Status = IQS_ExternalTrigger();
```

4. Since each example in the Qt provided examples is encapsulated in a separate function, you can simply uncomment the desired example when using it (multiple examples cannot be used simultaneously). For instance, when testing the Device_GetDeviceInfo example, uncomment it, save, and click run. The image shown indicates that the device is operating normally.



```
#include "example.h"

int main()
{
    int Status = 0;

    Status = Device_GetDeviceInfo();
    //Status = Device_SysPowerState();
    //Status = Device_AboutGNSS();
    //Status = Device_GetAndSetIP();
    //Status = Device_TimeOfSetMode();

    //Status = SWP_GetSpectrum_Standard();
    //Status = SWP_EZGetPartialSweep();
    //Status = SWP_MaxHold_MinHold();
    //Status = SWP_TraceAverage();
    //Status = SWP_AutoSetMeasure();
    //Status = SWP_SetFreqCompensation();
    //Status = SWP_TimeOfSetFunction();
    //Status = SWP_PickMaxPower();
    //Status = SWP_GetSpectrum_SigAndSpur();
    //Status = SWP_GetSpectrumAndIQS();
    //Status = SWP_GNSSReferenceClock();
    //Status = SWP_GetSpectrum_Trigger();
    //Status = SWP_GetSpectrum_TraceAlign();
    //Status = SWP_Fixedtime_GetFrames();
    //Status = SWP_CalibrateRefClock();


    //Status = IQS_GetIQ_EZStandard();
    //Status = IQS_GetIQ_Standard();
    //Status = IQS_ScaleIQDataToVolts();
    //Status = IQS_ConfigandGetIQ_Time();
    //Status = IQS_ToSpectrumByLiquid();
    //Status = IQS_AudioAnalysis();
```

```
11:01:17: Starting C:\Users\60536\Desktop\New folder (3)\HTRA_Qt_Examples\htrademo\build\Desktop_Qt_5_15_2_MSVC2019_64bit-Debug\debug\htrademo.exe...
htra_api verion: 0.55.55
Device is opened successfully
BusSpeed: 3
Model: 13, E90 R3
DeviceInfo.DeviceUID: 313251190030004d
```

3.3 Qt Example Description

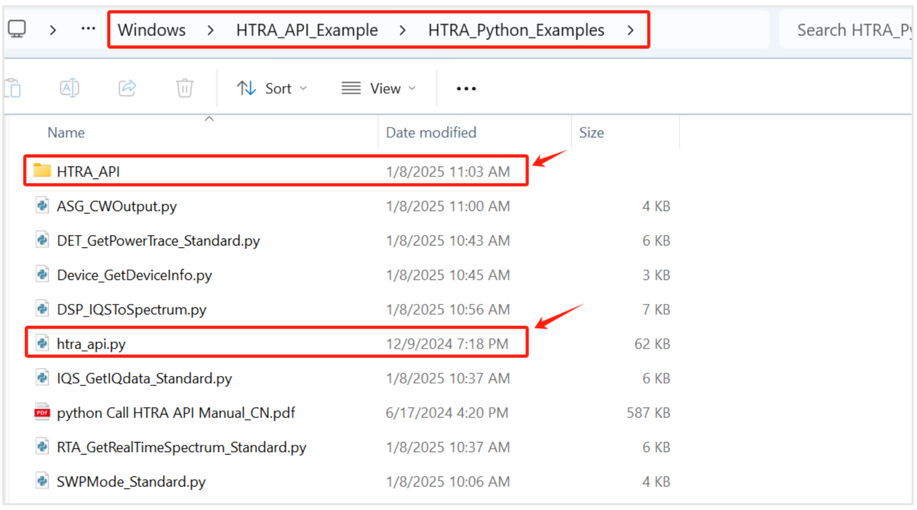
The Qt examples on the USB stick include a general example (HTRA_Qt_Example), an FM and AM demodulation example (HTRA_Demodulation), and an IQ to Spectrum example (IQS_ToSpectrumBuLiquid) using the Liquid library, refer to the C/C++

examples chapter for specific examples.

4. Python

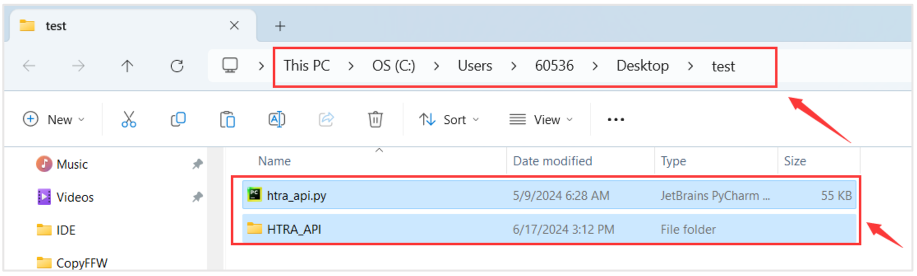
4.1 Configure Development Environment

1. Create a folder on the desktop and name it, for example, test. Open the USB drive and copy the "HTRA_API" folder and "htra_api.py" file from "\Windows\HTRA_API_Example\HTRA_Python_Examples" on the USB drive to the newly created folder.



Windows > HTRA_API_Example > HTRA_Python_Examples >

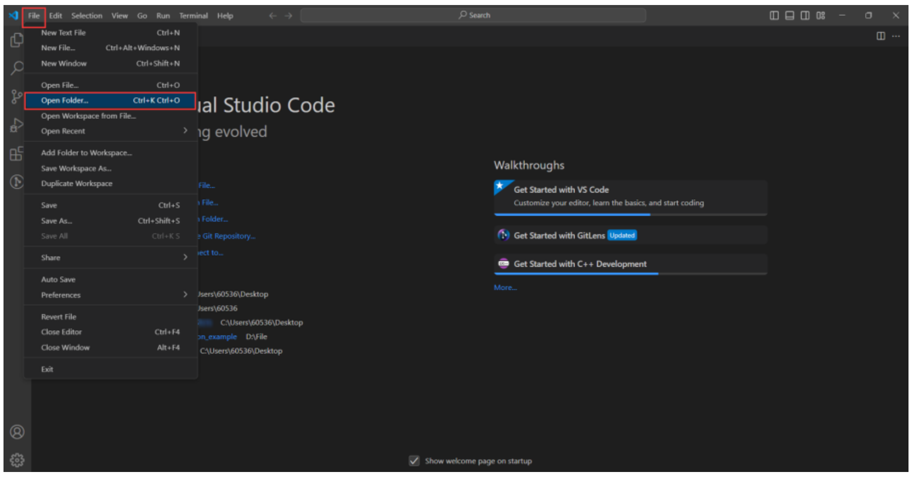
Name	Date modified	Size
HTRA_API	1/8/2025 11:03 AM	
ASG_CWOutput.py	1/8/2025 11:00 AM	4 KB
DET_GetPowerTrace_Standard.py	1/8/2025 10:43 AM	6 KB
Device_GetDeviceInfo.py	1/8/2025 10:45 AM	3 KB
DSP_IQSToSpectrum.py	1/8/2025 10:56 AM	7 KB
htra_api.py	12/9/2024 7:18 PM	62 KB
IQS_GetIQdata_Standard.py	1/8/2025 10:37 AM	6 KB
python Call HTRA API Manual_CN.pdf	6/17/2024 4:20 PM	587 KB
RTA_GetRealTimeSpectrum_Standard.py	1/8/2025 10:37 AM	6 KB
SWPMode_Standard.py	1/8/2025 10:06 AM	4 KB

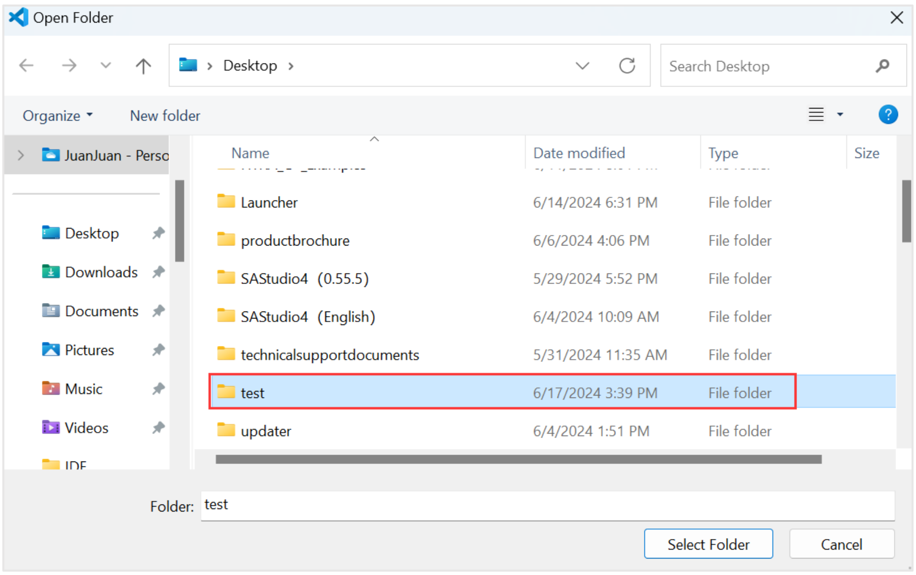


This PC > OS (C:) > Users > 60536 > Desktop > test

Name	Date modified	Type	Size
htra_api.py	5/9/2024 6:28 AM	JetBrains PyCharm ...	55 KB
HTRA_API	6/17/2024 3:12 PM	File folder	

2. Open Visual Studio Code, click on File and then Open Folder, and open the folder you just created.

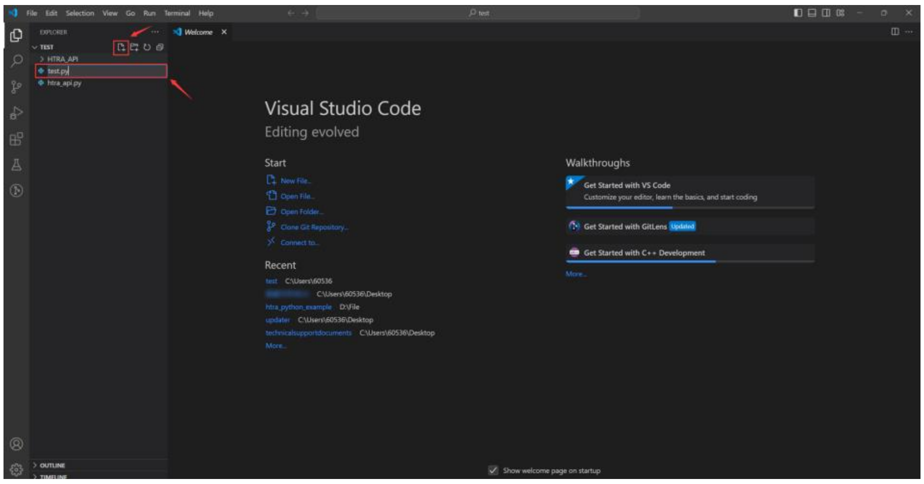




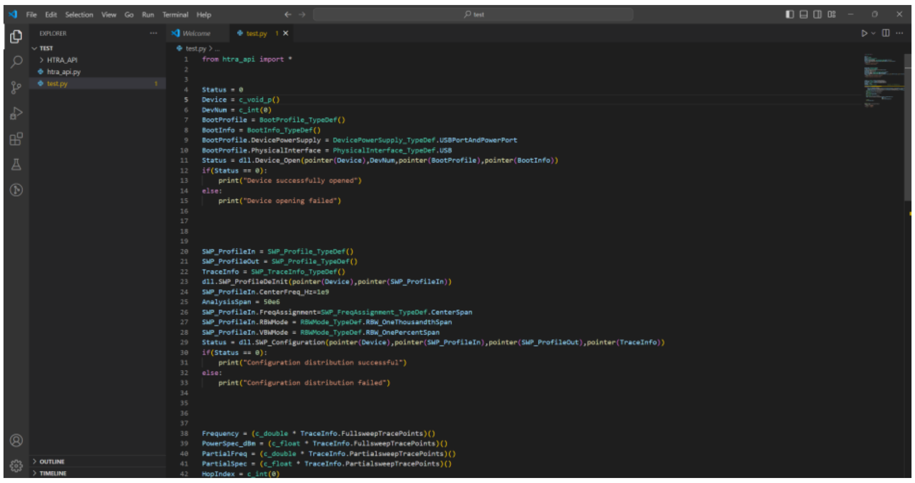
Name	Date modified	Type	Size
Launcher	6/14/2024 6:31 PM	File folder	
productbrochure	6/6/2024 4:06 PM	File folder	
SAStudio4 (0.55.5)	5/29/2024 5:52 PM	File folder	
SAStudio4 (English)	6/4/2024 10:09 AM	File folder	
technicalsupportdocuments	5/31/2024 11:35 AM	File folder	
test	6/17/2024 3:39 PM	File folder	
updater	6/4/2024 1:51 PM	File folder	

Folder: test

3. Create a new Python file.



4. Write code normally. You can refer to the Python examples included in the USB drive, specifically in the folder "\Windows\HTRA_API_Example\HTRA_Python_Examples" for the project.

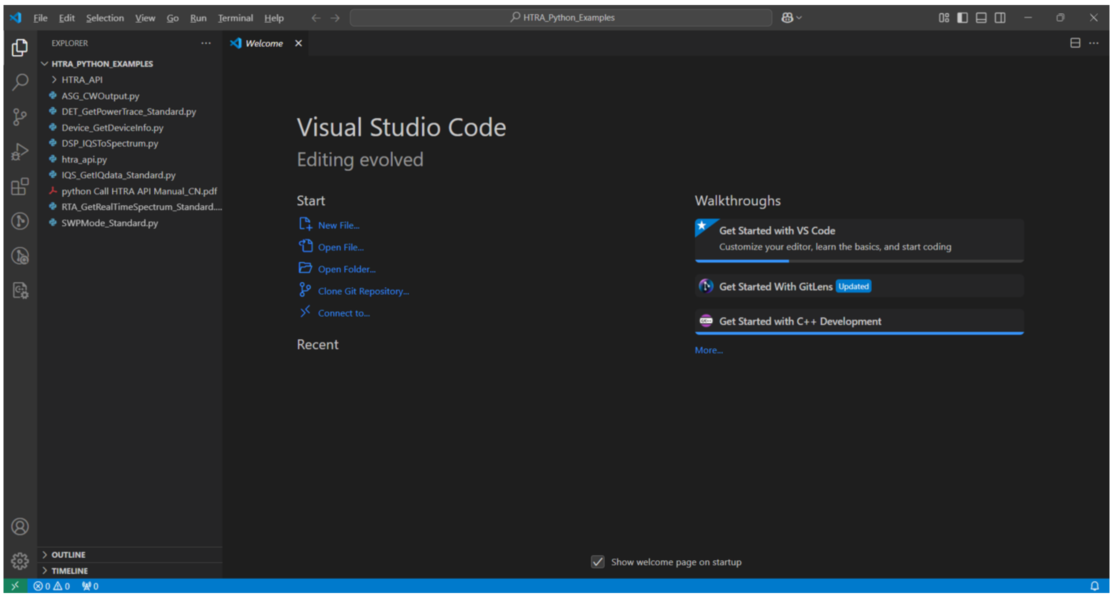


```python
from htra_api import *


Status = 0
Device = c_void_p()
DevNum = c_int(0)
BootProfile = BootProfile_TypeDef()
BootInfo = BootInfo_TypeDef()
BootProfile.DevicePowerSupply = DevicePowerSupply_TypeDef.USBPortAndPowerPort
BootProfile.PhysicalInterface = PhysicalInterface_TypeDef.USB
Status = dll.Device_Open(pointer(Device),DevNum,pointer(BootProfile),pointer(BootInfo))
if(Status == 0):
    print("Device successfully opened")
else:
    print("Device opening failed")




SWP_ProfileIn = SWP_Profile_TypeDef()
SWP_ProfileOut = SWP_Profile_TypeDef()
TraceInfo = SWP_TraceInfo_TypeDef()
dll.SWP_ProfileDeInit(pointer(Device),pointer(SWP_ProfileIn))
SWP_ProfileIn.CenterFreq_Hz=1e9
AnalysisSpan = 50e6
SWP_ProfileIn.FreqAssignment=SWP_FreqAssignment_TypeDef.CenterSpan
SWP_ProfileIn.RBWMode = RBWMode_TypeDef.RBW_OneThousandthSpan
SWP_ProfileIn.VBWMode = RBWMode_TypeDef.RBW_OnePercentSpan
Status = dll.SWP_Configuration(pointer(Device),pointer(SWP_ProfileIn),pointer(SWP_ProfileOut),pointer(TraceInfo))
if(Status == 0):
    print("Configuration distribution successful")
else:
    print("Configuration distribution failed")




Frequency = (c_double * TraceInfo.FullsweepTracePoints)()
PowerSpec_dBm = (c_float * TraceInfo.FullsweepTracePoints)()
PartialFreq = (c_double * TraceInfo.PartialsweepTracePoints)()
PartialSpec = (c_float * TraceInfo.PartialsweepTracePoints)()
HopIndex = c_int(0)
```

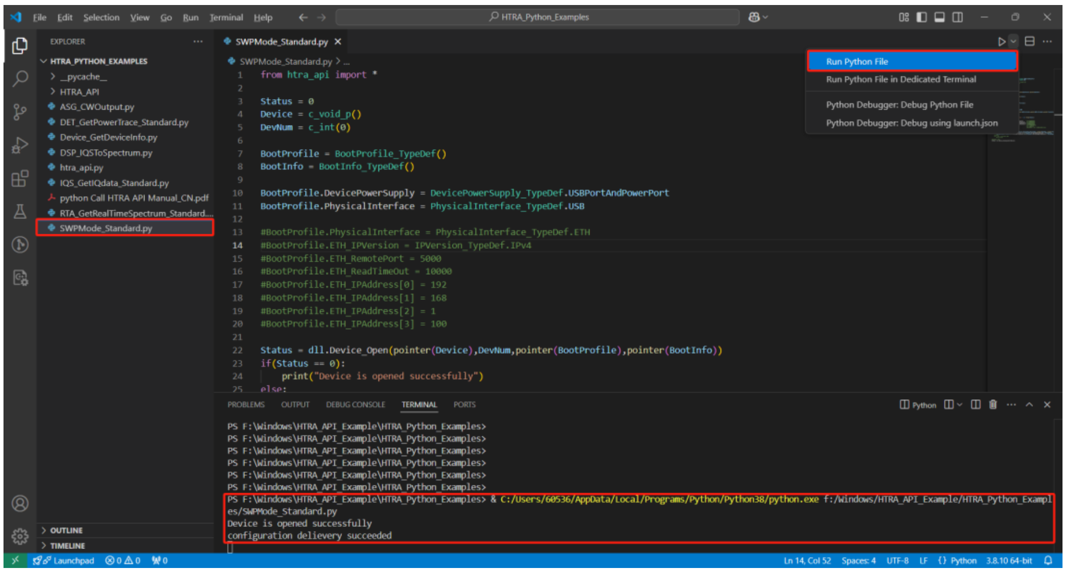
4.2 Python Example Usage Process

The usage process of the Python examples included in the USB drive is as follows:

1. As shown in the figure, open the entire project using vscode or another compiler from the USB drive provided: Windows\HTRA_API_Example\HTRA_Python_Examples. The htra_api.py file in the project is the mapping file for the dynamic link library in Python, while the other files are example programs (the role of the examples will be described in subsequent chapters).



2. Select any example program, configure the Python environment for it, and run the program directly. For instance, when using the SWPMode_Standard.py example, the device is shown to be operating normally as illustrated.



SWPMode_Standard.py
Run Python File
Run Python File in Dedicated Terminal
Python Debugger: Debug Python File
Python Debugger: Debug using launch.json

```python
from htra_api import *

Status = 0
Device = c_void_p()
DevNum = c_int(0)

BootProfile = BootProfile_TypeDef()
BootInfo = BootInfo_TypeDef()

BootProfile.DevicePowerSupply = DevicePowerSupply_TypeDef.USBPortAndPowerPort
BootProfile.PhysicalInterface = PhysicalInterface_TypeDef.USB

#BootProfile.PhysicalInterface = PhysicalInterface_TypeDef.ETH
#BootProfile.ETH_IPVersion = IPVersion_TypeDef.IPv4
#BootProfile.ETH_RemotePort = 5000
#BootProfile.ETH_ReadTimeOut = 10000
#BootProfile.ETH_IPAddress[0] = 192
#BootProfile.ETH_IPAddress[1] = 168
#BootProfile.ETH_IPAddress[2] = 1
#BootProfile.ETH_IPAddress[3] = 100

Status = dll.Device_Open(pointer(Device),DevNum,pointer(BootProfile),pointer(BootInfo))
if(Status == 0):
    print("Device is opened successfully")
else:
```

```
PS F:\Windows\HTRA_API_Example\HTRA_Python_Examples>
PS F:\Windows\HTRA_API_Example\HTRA_Python_Examples>
PS F:\Windows\HTRA_API_Example\HTRA_Python_Examples>
PS F:\Windows\HTRA_API_Example\HTRA_Python_Examples>
PS F:\Windows\HTRA_API_Example\HTRA_Python_Examples>
PS F:\Windows\HTRA_API_Example\HTRA_Python_Examples>
PS F:\Windows\HTRA_API_Example\HTRA_Python_Examples> & C:/Users/60536/AppData/Local/Programs/Python/Python38/python.exe f:/Windows/HTRA_API_Example/HTRA_Python_Exampl
es/SWPMode_Standard.py
Device is opened successfully
configuration delievery succeeded
```

4.3 Python Example Description

4.3.1 Get device information

Device_GetDeviceInfo.py: Retrieves various device information, including API version, USB version, device model, device UID, MCU version, FPGA version, and device temperature.

4.3.2 Obtain Standard Spectrum Data

SWP_GetSpectrum_Standard.py: Obtains complete spectrum data within a specified frequency band(Plotting the spectrum if the host computer includes the matplotlib library).

4.3.3 Obtain IQ Data for a Fixed Number of Points or Duration

IQS_GetIQdata_Standard.py: Obtains IQ data under different trigger modes in IQS mode.

4.3.4 Obtain Power Detection Data for a Fixed Number of Points or Duration

DET_GetPowerTrace_Standard.py: Obtains power detection data under different trigger modes in DET mode(Plotting the Time-power graph if the host computer includes the matplotlib library).

4.3.5 Obtain real-time spectrum data for a fixed number of points or duration

RTA_GetRealTimeSpectrum_Standard.py: Obtain real-time spectrum data under different trigger modes in RTA mode.

4.3.6 IQ to Spectrum Data

DSP_IQSToSpectrum.py: Convert the IQ data obtained in IQS mode into spectrum data(If the host computer contains the matplotlib library then plot the spectrum and IQ time-domain graphs).

4.3.7 GNSS Related

Device_AboutGNSS.py: get the latitude, longitude, elevation and time information obtained by the GNSS module, and get the latitude, longitude and time information in the MeasAuxInfo structure in IQS mode.

5. Matlab

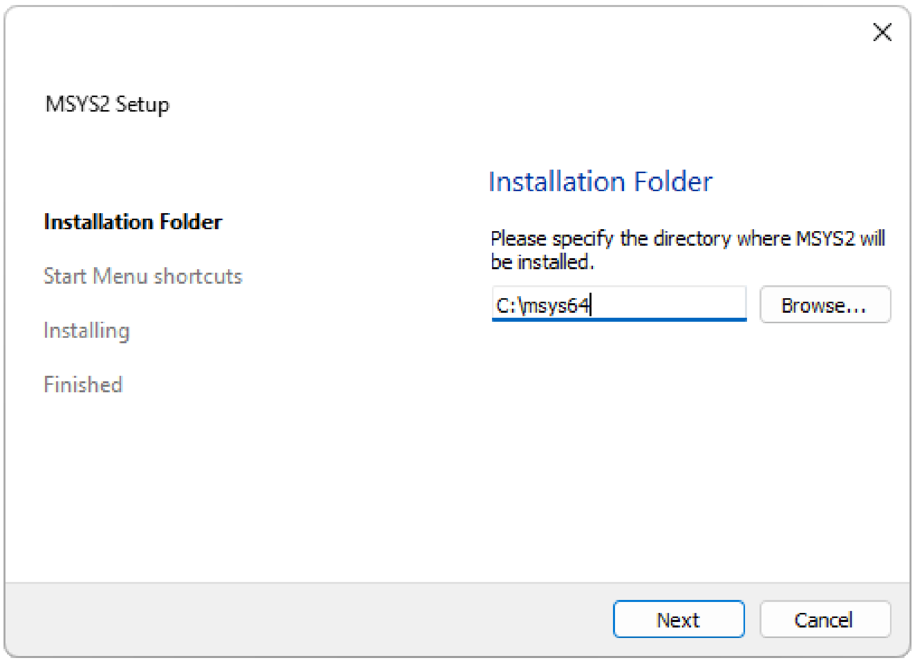
5.1 Configure Development Environment

The method of calling htra_api in 32-bit is basically the same as in 64-bit, so the following tutorial uses Matlab 2016a as an example to illustrate how to call the 64-bit htra_api.

5.1.1 Install MSYS2

Download and installation link: https://www.msys2.org/

1. Download the installer MSYS2-x86_64-20231026.exe
2. Run the installer. MSYS2 requires 64-bit Windows 8.1 or higher.
3. The default installation path is C:\msys64, but you can choose a different path as needed.

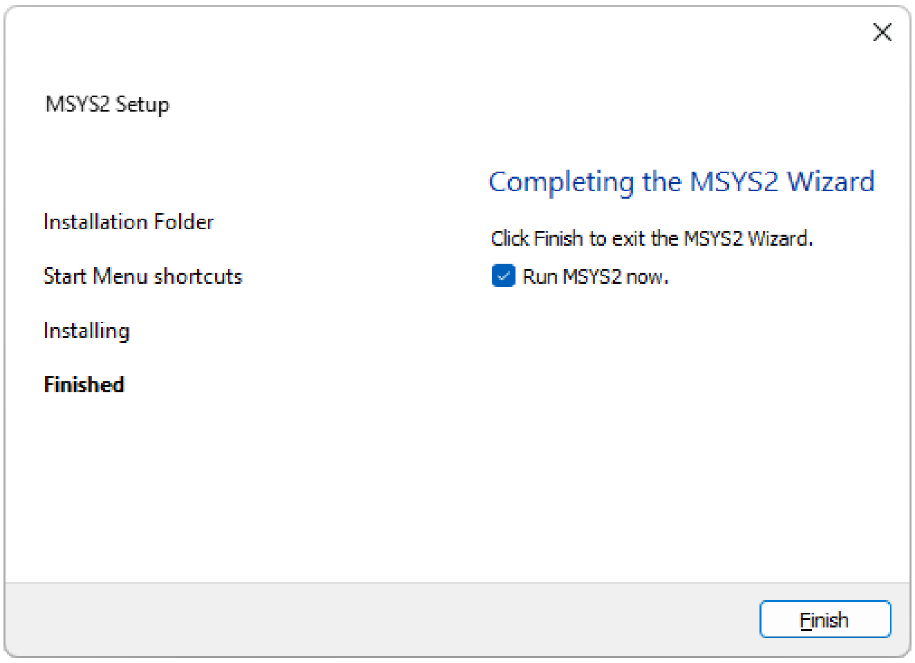

MSYS2 Setup

Installation Folder

Please specify the directory where MSYS2 will be installed.

C:\msys64

4. Once completed, click Finish.


MSYS2 Setup

Installation Folder

Start Menu shortcuts

Installing

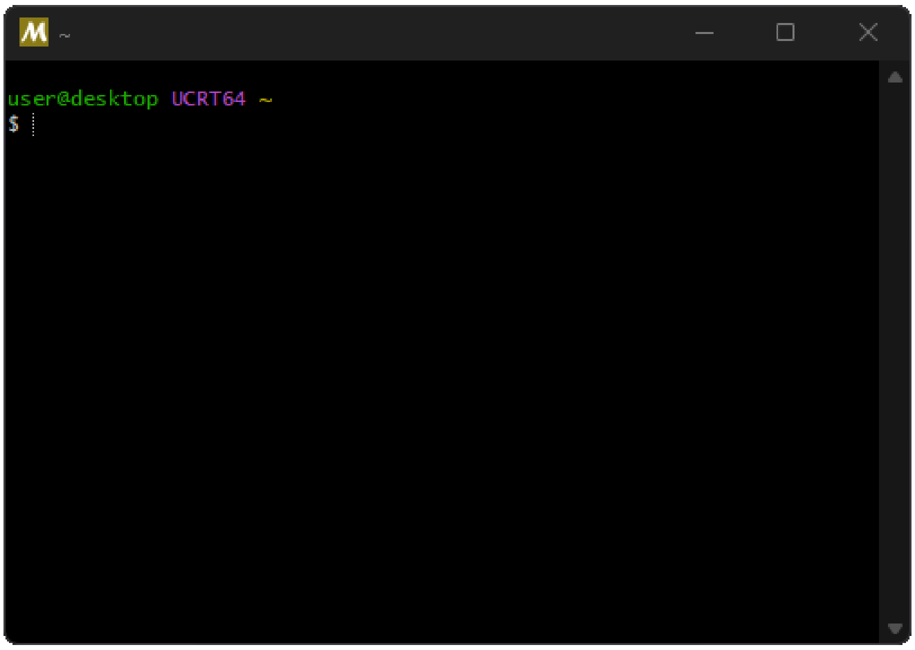
Finished

Completing the MSYS2 Wizard

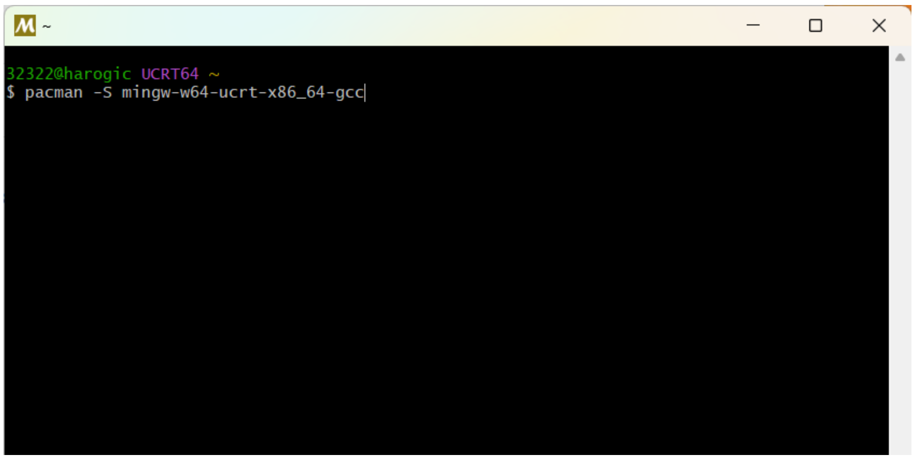
Click Finish to exit the MSYS2 Wizard.

[✓] Run MSYS2 now.

5. Now, MSYS2 is ready, and the terminal for the UCRT64 environment has started.

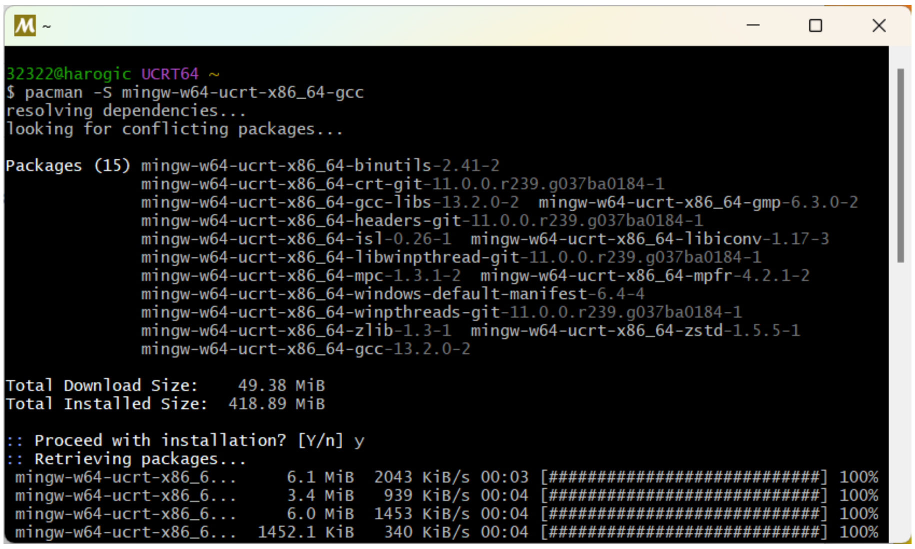

```
user@desktop UCRT64 ~
$
```

6. To install the GCC tools, enter the command: pacman -S mingw-w64-ucrt-x86_64-gcc



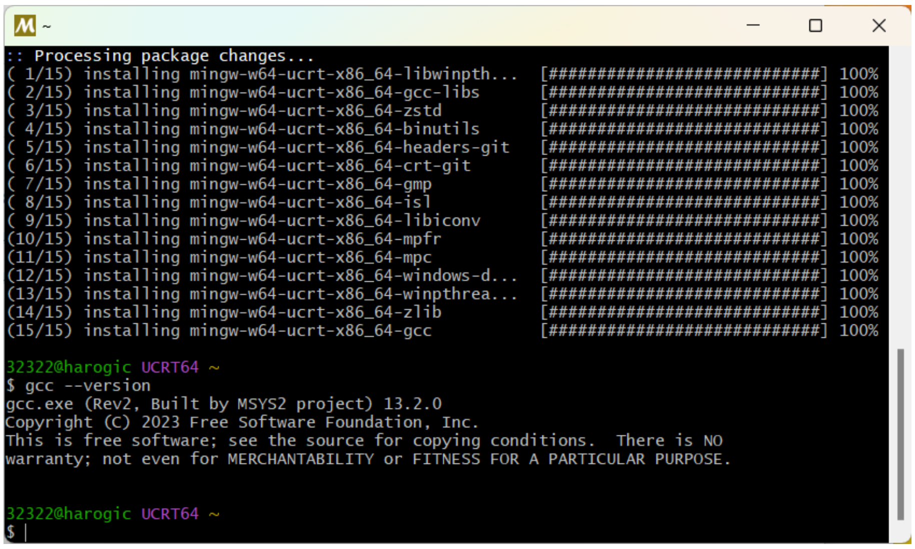
```
32322@harogic UCRT64 ~
$ pacman -S mingw-w64-ucrt-x86_64-gcc
```

7. The terminal window will display the following output. Press "Enter" to continue.



```
32322@harogic UCRT64 ~
$ pacman -S mingw-w64-ucrt-x86_64-gcc
resolving dependencies...
looking for conflicting packages...

Packages (15) mingw-w64-ucrt-x86_64-binutils-2.41-2
              mingw-w64-ucrt-x86_64-crt-git-11.0.0.r239.g037ba0184-1
              mingw-w64-ucrt-x86_64-gcc-libs-13.2.0-2  mingw-w64-ucrt-x86_64-gmp-6.3.0-2
              mingw-w64-ucrt-x86_64-headers-git-11.0.0.r239.g037ba0184-1
              mingw-w64-ucrt-x86_64-isl-0.26-1  mingw-w64-ucrt-x86_64-libiconv-1.17-3
              mingw-w64-ucrt-x86_64-libwinpthread-git-11.0.0.r239.g037ba0184-1
              mingw-w64-ucrt-x86_64-mpc-1.3.1-2  mingw-w64-ucrt-x86_64-mpfr-4.2.1-2
              mingw-w64-ucrt-x86_64-windows-default-manifest-6.4-4
              mingw-w64-ucrt-x86_64-winpthreads-git-11.0.0.r239.g037ba0184-1
              mingw-w64-ucrt-x86_64-zlib-1.3-1  mingw-w64-ucrt-x86_64-zstd-1.5.5-1
              mingw-w64-ucrt-x86_64-gcc-13.2.0-2

Total Download Size:    49.38 MiB
Total Installed Size:  418.89 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
 mingw-w64-ucrt-x86_6...      6.1 MiB  2043 KiB/s 00:03 [############################] 100%
 mingw-w64-ucrt-x86_6...      3.4 MiB   939 KiB/s 00:04 [############################] 100%
 mingw-w64-ucrt-x86_6...      6.0 MiB  1453 KiB/s 00:04 [############################] 100%
 mingw-w64-ucrt-x86_6...   1452.1 KiB   340 KiB/s 00:04 [############################] 100%
```

8. Enter the command gcc --version to check the version information of GCC.



```
:: Processing package changes...
( 1/15) installing mingw-w64-ucrt-x86_64-libwinpth...   [############################] 100%
( 2/15) installing mingw-w64-ucrt-x86_64-gcc-libs       [############################] 100%
( 3/15) installing mingw-w64-ucrt-x86_64-zstd           [############################] 100%
( 4/15) installing mingw-w64-ucrt-x86_64-binutils       [############################] 100%
( 5/15) installing mingw-w64-ucrt-x86_64-headers-git    [############################] 100%
( 6/15) installing mingw-w64-ucrt-x86_64-crt-git        [############################] 100%
( 7/15) installing mingw-w64-ucrt-x86_64-gmp            [############################] 100%
( 8/15) installing mingw-w64-ucrt-x86_64-isl            [############################] 100%
( 9/15) installing mingw-w64-ucrt-x86_64-libiconv       [############################] 100%
(10/15) installing mingw-w64-ucrt-x86_64-mpfr           [############################] 100%
(11/15) installing mingw-w64-ucrt-x86_64-mpc            [############################] 100%
(12/15) installing mingw-w64-ucrt-x86_64-windows-d...   [############################] 100%
(13/15) installing mingw-w64-ucrt-x86_64-winpthrea...   [############################] 100%
(14/15) installing mingw-w64-ucrt-x86_64-zlib           [############################] 100%
(15/15) installing mingw-w64-ucrt-x86_64-gcc            [############################] 100%

32322@harogic UCRT64 ~
$ gcc --version
gcc.exe (Rev2, Built by MSYS2 project) 13.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

32322@harogic UCRT64 ~
$ |
```

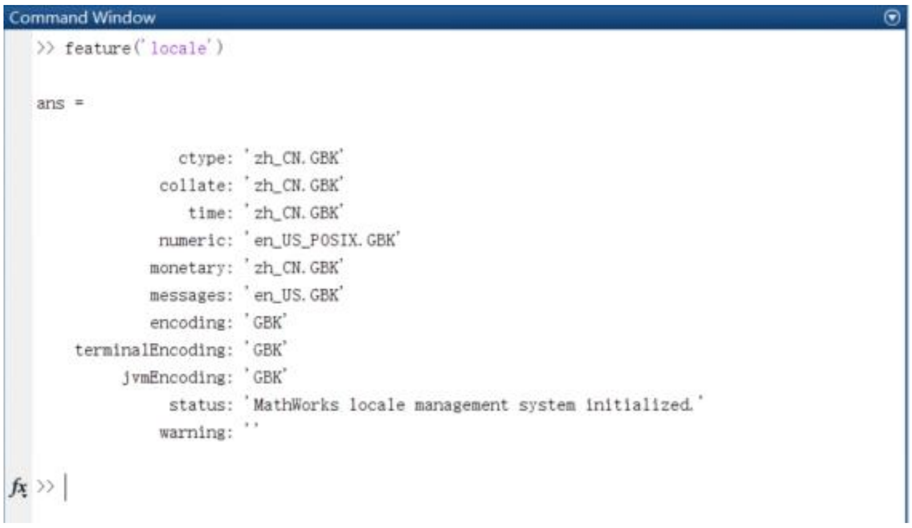
5.1.2 Configure Matlab

1. 1. Solve the case of Matlab 2016a opening .m file with Chinese garbled code.

Note: If you are using a version of Matlab higher than 2019a, ignore this step.

1) View the current coding format:

At the Matlab command line type: feature('locale')



```
Command Window
>> feature('locale')

ans =

              ctype: 'zh_CN.GBK'
            collate: 'zh_CN.GBK'
               time: 'zh_CN.GBK'
            numeric: 'en_US_POSIX.GBK'
           monetary: 'zh_CN.GBK'
           messages: 'en_US.GBK'
           encoding: 'GBK'
   terminalEncoding: 'GBK'
        jvmEncoding: 'GBK'
             status: 'MathWorks locale management system initialized.'
            warning: ''

fx >>
```

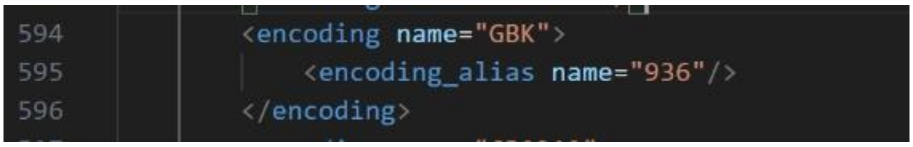
As you can see from the figure, the encoding format is GBK.

2) Right-click on the Matlab2016a shortcut and select "Open File Location" to open the folder where Matlab.exe is located.

3) In the folder shown in step (2), find lcdata.xml and lcdata_utf8.xml, rename lcdata.xml to lcdata_old.xml, i.e., backup the original lcdata.xml.

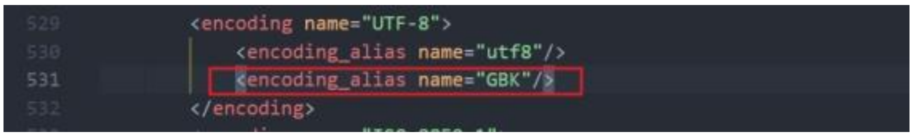
4) Make a copy of lcdata_utf8.xml and put it in the same level folder, and rename the newly copied file lcdata_utf8.xml to lcdata.xml.

5) Open lcdata.xml and delete the GBK related code shown below.



```
<encoding name="GBK">
    <encoding_alias name="936"/>
</encoding>
```

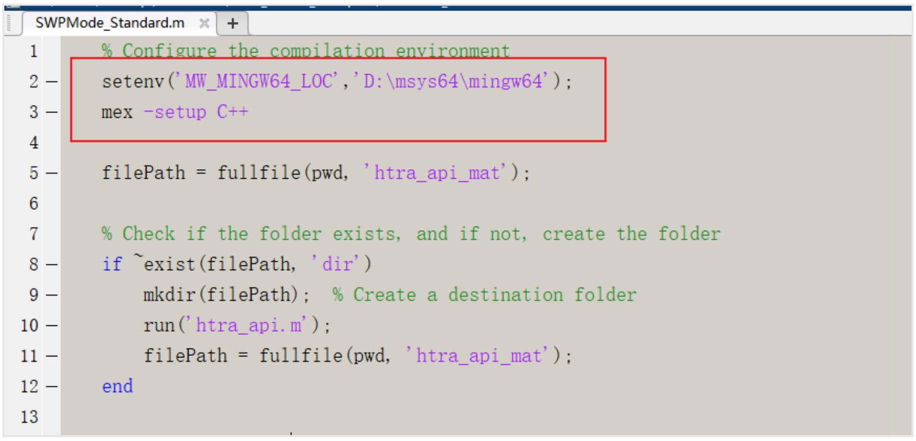
6) Find the "UTF-8" part, add the code in the marked line to the corresponding position in the figure, save lcdata.xml and close the file.



```
<encoding name="UTF-8">
    <encoding_alias name="utf8"/>
    <encoding_alias name="GBK"/>
</encoding>
```

7) After restarting Matlab, the garbled Chinese is back to normal.

2. Configuration of the compilation environment Method 1: Configuration in scripts: At script runtime, run setenv('MW_MINGW64_LOC', ' D:\msys64\ucrt64') and the mex -setup C++ command to configure the compilation environment for C++.



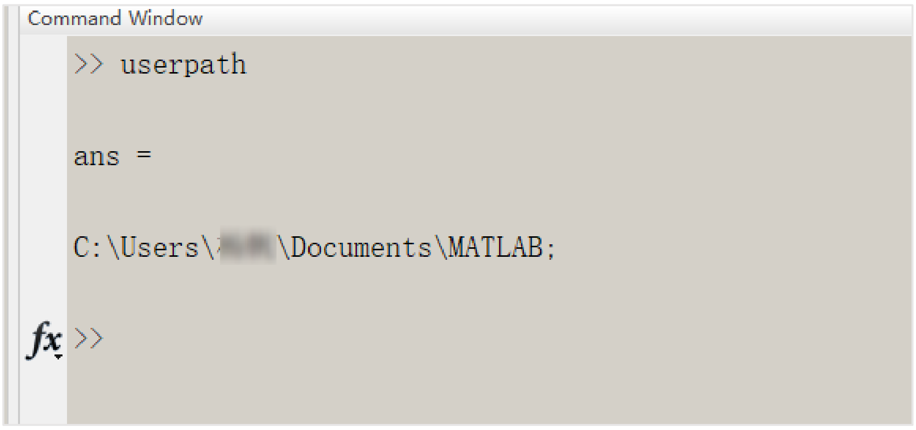
SWPMode_Standard.m

```
% Configure the compilation environment
setenv('MW_MINGW64_LOC','D:\msys64\mingw64');
mex -setup C++

filePath = fullfile(pwd, 'htra_api_mat');

% Check if the folder exists, and if not, create the folder
if ~exist(filePath, 'dir')
    mkdir(filePath);  % Create a destination folder
    run('htra_api.m');
    filePath = fullfile(pwd, 'htra_api_mat');
end
```

Note: D:\msys64\ucrt64 is the folder where the compilation environment is located. Please check if there are files such as c++.exe, g++.exe, and gcc.exe in the bin folder at this address. If they exist, this address is the compilation environment address; if not, please find the correct address of the compilation environment.

3. Configuration of the compilation environment method II: startup.m file changes.

1) In the Matlab terminal, input: userpath, and the command line window will output a result similar to: C:\Users\YourUsername\Documents\MATLAB

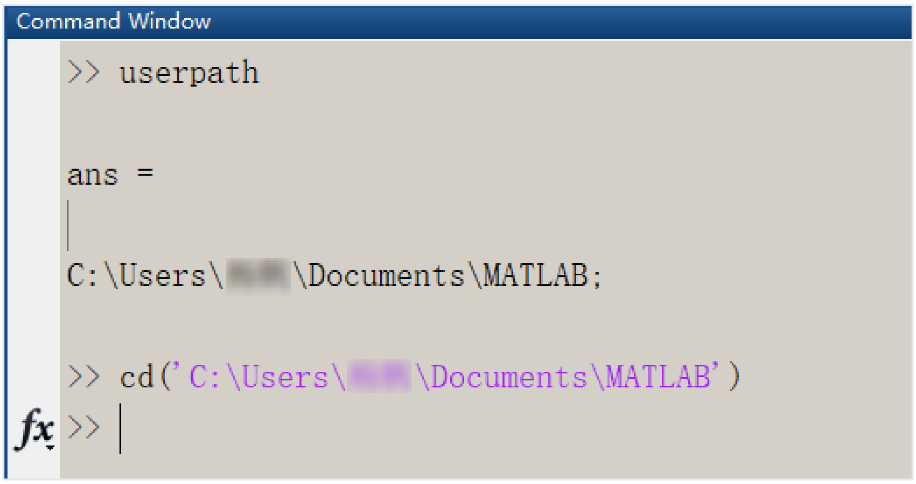


Command Window

```
>> userpath

ans =

C:\Users\[redacted]\Documents\MATLAB;

>>
```

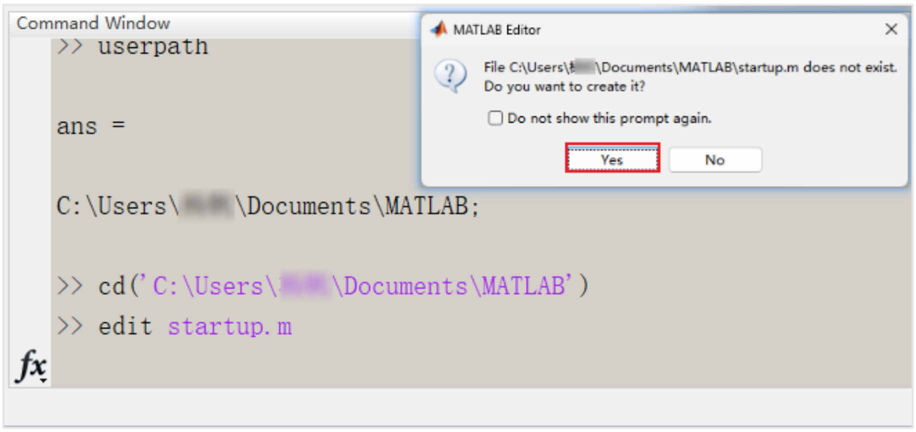
2) Check if the startup.m file exists at C:\Users\YourUsername\Documents\MATLAB. If it does not exist, create a new startup.m file at this location. The steps for creation are as follows:

i. Matlab terminal input: cd('C:\Users\YourUsername\Documents\MATLAB'), switch the working directory to C:\Users\YourUsername\Documents\MATLAB.



```
>> userpath

ans =

C:\Users\[redacted]\Documents\MATLAB;

>> cd('C:\Users\[redacted]\Documents\MATLAB')
>>
```

ii. Matlab terminal input: edit startup.m, select "Yes" in the pop-up window to create the startup.m file.



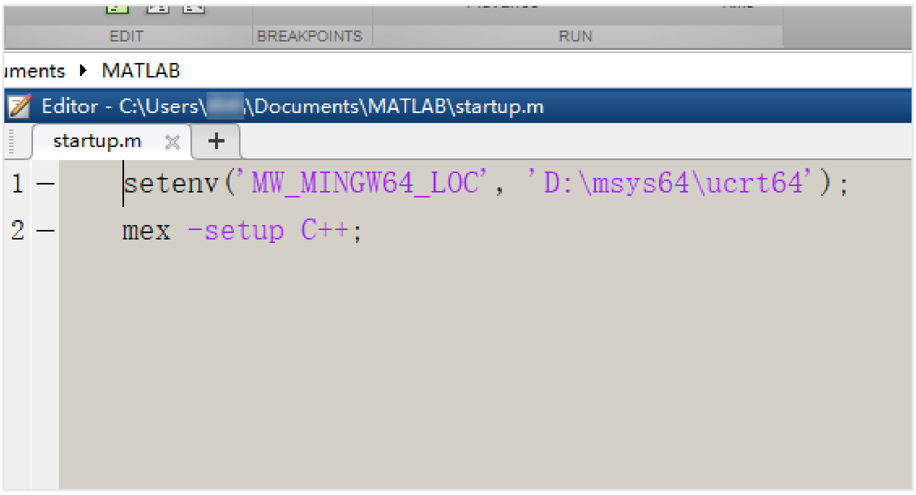
```
>> userpath

ans =

C:\Users\[redacted]\Documents\MATLAB;

>> cd('C:\Users\[redacted]\Documents\MATLAB')
>> edit startup.m
fx
```

File C:\Users\[redacted]\Documents\MATLAB\startup.m does not exist.
Do you want to create it?

☐ Do not show this prompt again.

iii. Add commands in the startup.m file:

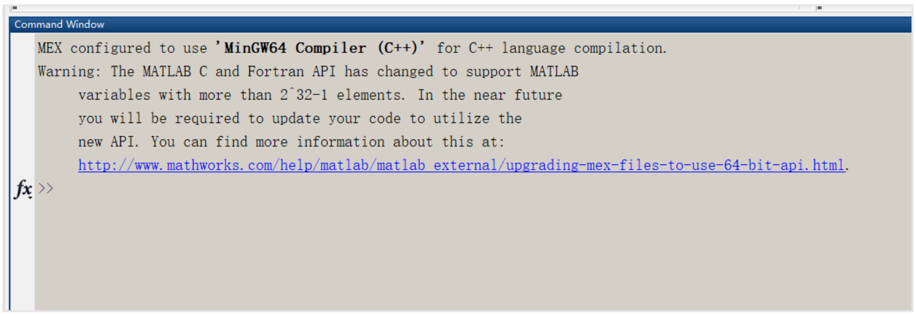
setenv('MW_MINGW64_LOC', 'D:\msys64\ucrt64'); andmex -setup C++.



```
setenv('MW_MINGW64_LOC', 'D:\msys64\ucrt64');
mex -setup C++;
```

iv. startup.m After editing the file, save and close it.

v. Restart Matlab, and observe that the command line window appears as shown in the figure below, indicating that the configuration is complete.



```
MEX configured to use 'MinGW64 Compiler (C++)' for C++ language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
     variables with more than 2^32-1 elements. In the near future
     you will be required to update your code to utilize the
     new API. You can find more information about this at:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-use-64-bit-api.html.
fx >>
```

5.1.3 Call htra_api.dll Description

1. loadlibrary

The loadlibrary function can load dynamic link libraries.

loadlibrary('.\htra_api\htra_api.dll','.\htra_api\htra_api.h'); Ensure that the file paths for .dll and .h are correct.

```
%Loadhtra_api.dll
if not(libisloaded('htra_api.dll'))
    %The file paths for .dll and .h files should be carefully noted
    loadlibrary('.\htra_api\htra_api.dll','.\htra_api\htra_api.h');
end
```

2. libfunctions

libfunctions('htra_api'); This is used to view all available functions in htra_api.dll.

```
%View all functions in the API
libfunctions('htra_api');
```

3. libpointer

libpointer allows the creation of data type pointers in Matlab and passes them to external library functions.

```
%Open the device
%Create a Device pointer
Device = libpointer;
DevNum = 0;
Status = 0;
```

4. libstruct

libstruct is used to define structure types in Matlab and pass them to external library functions.

```
%Create a BootProfile structure.
BootProfile = libstruct('BootProfile_TypeDef');
save(fullfile(folderPath,'BootProfile.mat'), 'BootProfile');

%Create the DeviceInfo structure
DeviceInfo = libstruct('DeviceInfo_TypeDef');
save(fullfile(folderPath,'DeviceInfo.mat'), 'DeviceInfo');
```

5. get

The get function is used to retrieve the property values of a structure.

```
%Call the Device_Open function
Status = calllib('htra_api', 'Device_Open', Device, DevNum, BootProfile_p, BootInfo_p);
get(BootInfo_p); %Print the value of BootInfo_p
```

6. calllib

calllib is the command in Matlab used to call functions in htra_api.dll.

```
%Call the Device_Open function
Status = calllib('htra_api', 'Device_Open', Device, DevNum, BootProfile, BootInfo);
get(BootProfile);%Prints the value of BootInfo_p
```

7. load

load is used to load the .mat structure files generated in htra_api.m.

```
% Load the BootProfile_TypeDef structure directly.
load(fullfile(filePath, 'BootProfile.mat'));
% Load the BootInfo_TypeDef structure directly
load(fullfile(filePath, 'BootInfo.mat'));
```

8. fullfile

fullfile is a function in Matlab used to generate complete file paths. It loads the BootProfile.mat and BootInfo.mat files from the filePath.

```
% Load the BootProfile_TypeDef structure directly.
load(fullfile(filePath, 'BootProfile.mat'));
% Load the BootInfo_TypeDef structure directly
load(fullfile(filePath, 'BootInfo.mat'));
```

9. unloadlibrary

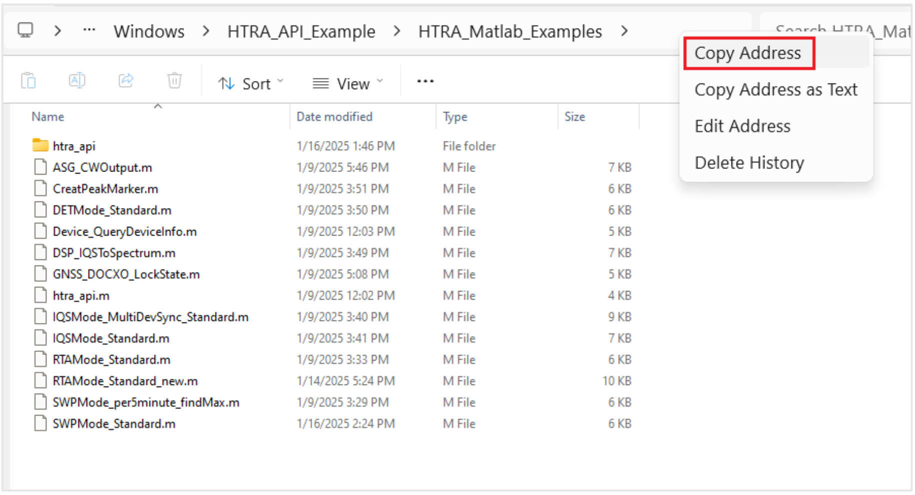
unloadlibrary is used to unload a previously loaded htra_api library, appearing in pairs with loadlibrary.

```
%Unload library file
unloadlibrary('htra_api');
disp('Uninstall complete')
```

5.2 Matlab example usage process

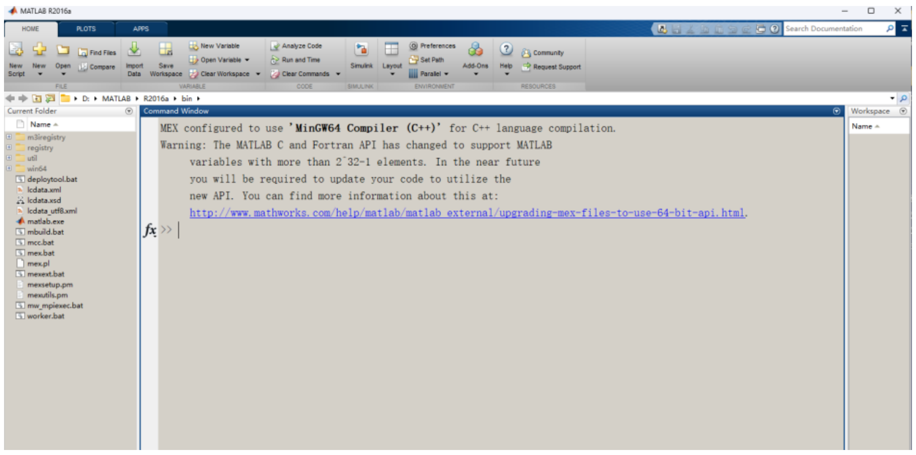
The usage process for the Matlab examples included on the USB drive is as follows:

1. Open the Windows\HTRA_API_Example\HTRA_Matlab_Examples folder on the USB drive, and double-click any .m file to open the example. For instructions on how to run the example, please refer directly to step 4.
2. If you cannot open the example in step 1, please continue to this step and copy the address Windows\HTRA_API_Example\HTRA_Matlab_Examples.



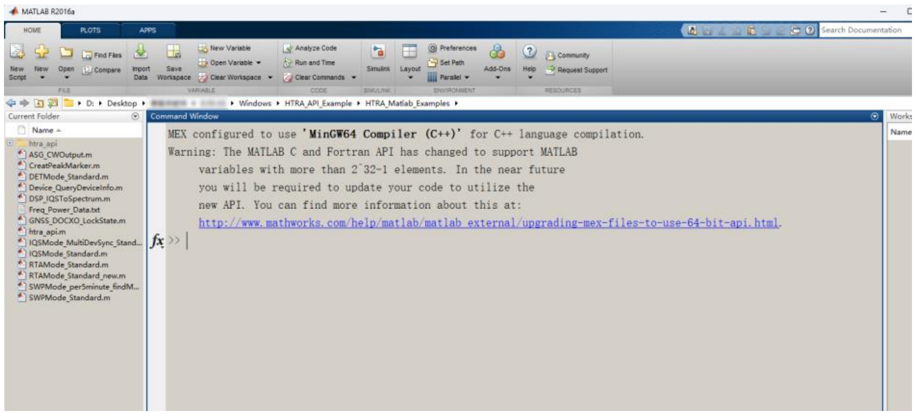
Name	Date modified	Type	Size
htra_api	1/16/2025 1:46 PM	File folder	
ASG_CWOutput.m	1/9/2025 5:46 PM	M File	7 KB
CreatPeakMarker.m	1/9/2025 3:51 PM	M File	6 KB
DETMode_Standard.m	1/9/2025 3:50 PM	M File	6 KB
Device_QueryDeviceInfo.m	1/9/2025 12:03 PM	M File	5 KB
DSP_IQSToSpectrum.m	1/9/2025 3:49 PM	M File	7 KB
GNSS_DOCXO_LockState.m	1/9/2025 5:08 PM	M File	5 KB
htra_api.m	1/9/2025 12:02 PM	M File	4 KB
IQSMode_MultiDevSync_Standard.m	1/9/2025 3:40 PM	M File	9 KB
IQSMode_Standard.m	1/9/2025 3:41 PM	M File	7 KB
RTAMode_Standard.m	1/9/2025 3:33 PM	M File	6 KB
RTAMode_Standard_new.m	1/14/2025 5:24 PM	M File	10 KB
SWPMode_per5minute_findMax.m	1/9/2025 3:29 PM	M File	6 KB
SWPMode_Standard.m	1/16/2025 2:24 PM	M File	6 KB

3. Open the Matlab software installed on your system.



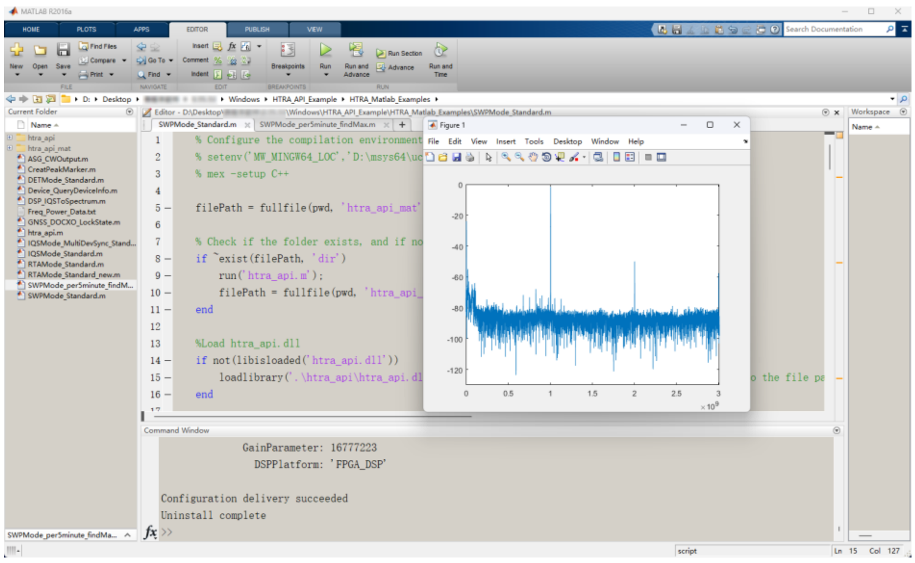
```
MEX configured to use 'MinGW64 Compiler (C++)' for C++ language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
     variables with more than 2^32-1 elements.  In the near future
     you will be required to update your code to utilize the
     new API. You can find more information about this at:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-use-64-bit-api.html.
fx >>
```

4. After pasting the copied address into the file address box, press Enter to navigate to the \Windows\HTRA_API_Example\HTRA_Matlab_Examples folder included with the materials.



```
MEX configured to use 'MinGW64 Compiler (C++)' for C++ language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
     variables with more than 2^32-1 elements.  In the near future
     you will be required to update your code to utilize the
     new API. You can find more information about this at:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-use-64-bit-api.html.
fx >>
```

5. Click on the .m file on the left as needed, click "Run," and wait for the Figure 1 window to appear, indicating that the example has run successfully. For the functional descriptions of each example, please refer to section 5.2 Matlab Example Description.



```
% Configure the compilation environment
% setenv('MW_MINGW64_LOC','D:\msys64\uc
% mex -setup C++

filePath = fullfile(pwd, 'htra_api_mat'

% Check if the folder exists, and if no
if ~exist(filePath, 'dir')
    run('htra_api.m');
    filePath = fullfile(pwd, 'htra_api_
end

%Load htra_api.dll
if not(libisloaded('htra_api.dll'))
    loadlibrary('.\htra_api\htra_api.dl
end
```

```
              GainParameter: 16777223
                DSPPlatform: 'FPGA_DSP'

Configuration delivery succeeded
Uninstall complete
>>
```

5.3 Introduction to Accompanying Examples

5.3.1 Get device information

Device_QueryDeviceInfo.m: Retrieve device information, including API version, USB version, device model, device UID, MCU version, FPGA version, and device temperature.

5.3.2 Obtain Standard Spectrum Data

SWPMode_Standard.m: Obtain complete spectrum data within the specified frequency band.

5.3.3 Create multiple cursors to display the frequency and power of the cursors.

CreatPeakMarker.m: Obtain spectrum data within the specified frequency band, create cursors, and perform peak searching.

5.3.4 Collect the peak spectrum every five minutes.

SWPMode_per5minute_findMax.m: Obtain spectrum data within the specified frequency band and search for the peak globally every five minutes.

5.3.5 Obtain continuous stream or fixed number of IQ data.

IQSMode_Standard.m: Acquire IQ data under different trigger modes in IQS mode.

5.3.6 The acquired IQ data is converted into spectrum data.

DSP_IQSToSpectrum.m: After acquiring IQ data, the obtained IQ data is converted into spectrum data.

5.3.7 Acquire continuous stream or fixed number of power detection data.

DET_GetPowerTrace_FixedPoints.m: Acquire power detection data under different triggering modes in DET mode.

5.3.8 Acquire continuous stream or fixed duration real-time spectrum data.

RTAMode_FixedPoints.m: Acquire real-time spectrum data under different triggering modes in RTA mode.

5.3.9 Internal signal source output signal.

ASG_CWOutput.m: Output single-tone signals, frequency sweep signals, and power sweep signals. Applicable only to devices with signal source options.

5.3.10 Lock GNSS antenna and DOCXO oscillator.

GNSS_DOCXO_LockState.m: Call the API interface to lock the GNSS antenna and DOCXO oscillator, applicable only to devices with IO expansion board options.

5.3.11 Multi-machine synchronization

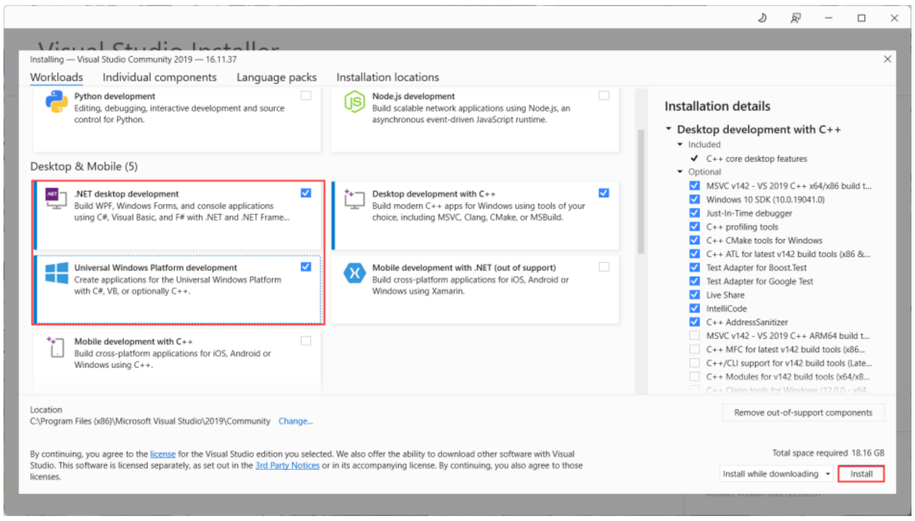
IQSMode_MultiDevSync_Standard.m: When using the same reference clock source input and the same trigger source input, two devices simultaneously acquire IQ data, allowing for the observation of the synchronization of their collected data.

6. C#

6.1 Configure Development Environment

6.1.1 Development Environment Confirmation

Open Visual Studio Installer, check the .NET desktop development components and Universal Windows Platform development components, and click Modify to ensure that Visual Studio 2019 has the C# development environment.



Installing — Visual Studio Community 2019 — 16.11.37

Workloads | Individual components | Language packs | Installation locations

Python development
Editing, debugging, interactive development and source control for Python.

Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

[✓] .NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

[✓] Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

[✓] Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Mobile development with .NET (out of support)
Build cross-platform applications for iOS, Android or Windows using Xamarin.

Mobile development with C++
Build cross-platform applications for iOS, Android or Windows using C++.

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community Change...

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

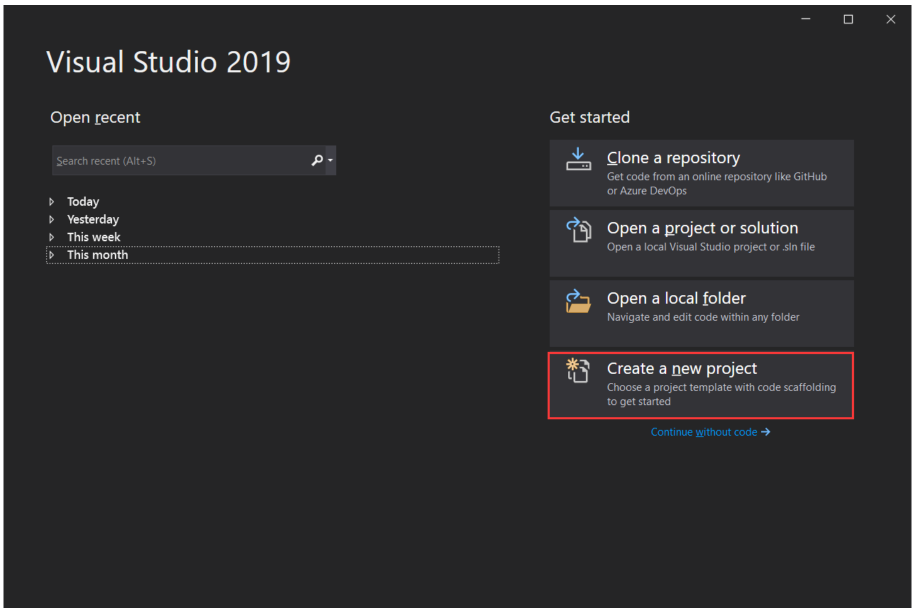
Installation details

Desktop development with C++
- Included
  - ✓ C++ core desktop features
- Optional
  - ✓ MSVC v142 - VS 2019 C++ x64/x86 build t...
  - ✓ Windows 10 SDK (10.0.19041.0)
  - ✓ Just-In-Time debugger
  - ✓ C++ profiling tools
  - ✓ C++ CMake tools for Windows
  - ✓ C++ ATL for latest v142 build tools (x86 &...
  - ✓ Test Adapter for Boost.Test
  - ✓ Test Adapter for Google Test
  - ✓ Live Share
  - ✓ IntelliCode
  - ✓ C++ AddressSanitizer
  - ☐ MSVC v142 - VS 2019 C++ ARM64 build t...
  - ☐ C++ MFC for latest v142 build tools (x86...
  - ☐ C++/CLI support for v142 build tools (Late...
  - ☐ C++ Modules for v142 build tools (x64/x8...

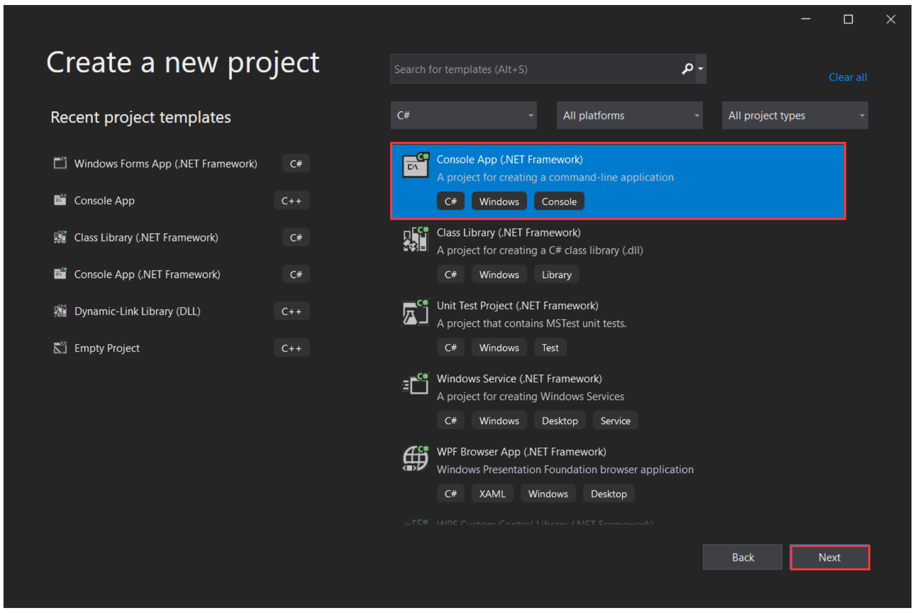
Total space required 18.16 GB

6.1.2 Project Setup

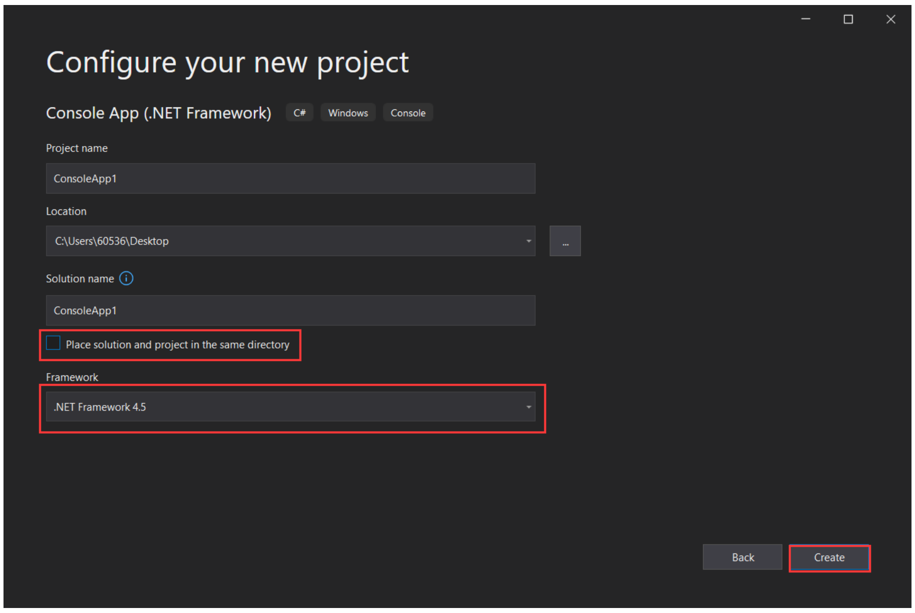
1. Open Visual Studio 2019 and click on Create a New Project.


Visual Studio 2019
Open recent
Search recent (Alt+S)
Today
Yesterday
This week
This month
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

2. Select C# Console Application and click Next.


Create a new project
Search for templates (Alt+S)
Clear all
Recent project templates
Windows Forms App (.NET Framework) C#
Console App C++
Class Library (.NET Framework) C#
Console App (.NET Framework) C#
Dynamic-Link Library (DLL) C++
Empty Project C++
C#
All platforms
All project types
Console App (.NET Framework)
A project for creating a command-line application
C# Windows Console
Class Library (.NET Framework)
A project for creating a C# class library (.dll)
C# Windows Library
Unit Test Project (.NET Framework)
A project that contains MSTest unit tests.
C# Windows Test
Windows Service (.NET Framework)
A project for creating Windows Services
C# Windows Desktop Service
WPF Browser App (.NET Framework)
Windows Presentation Foundation browser application
C# XAML Windows Desktop
Back
Next

3. Enter the project name and storage location, uncheck the option to place the solution and project in the same directory. Select .NET Framework 4.5 as the framework, and finally click Create.


Configure your new project

Console App (.NET Framework)   C#   Windows   Console

Project name: ConsoleApp1

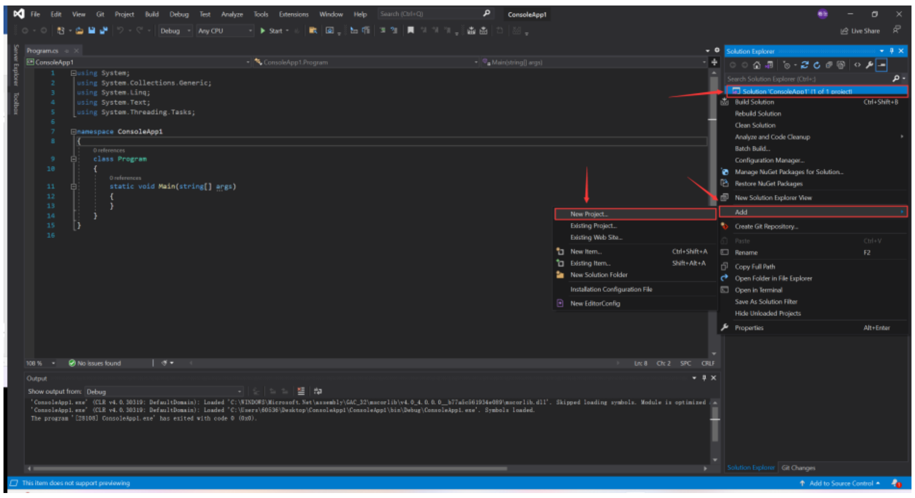
Location: C:\Users\60536\Desktop

Solution name: ConsoleApp1

[ ] Place solution and project in the same directory

Framework: .NET Framework 4.5

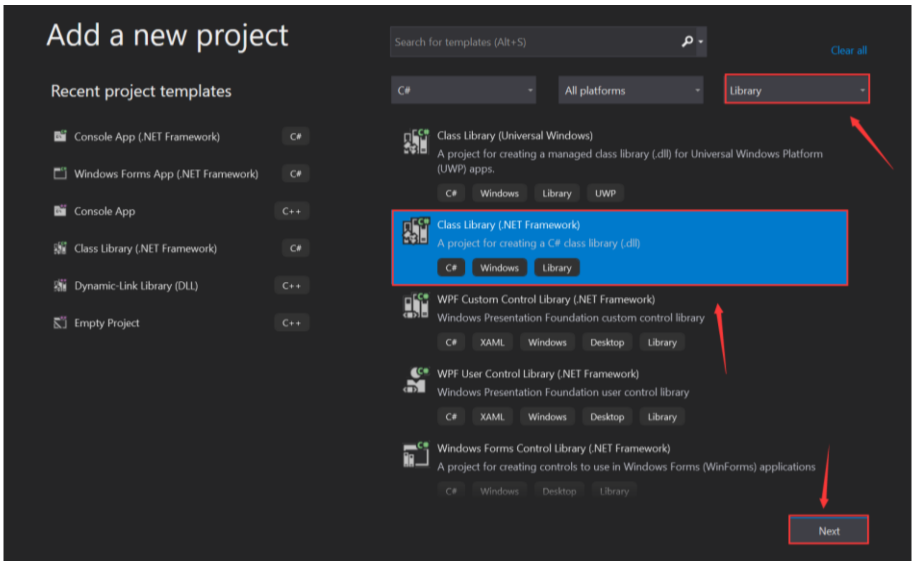
4. Once the creation is complete, open the project, right-click on the solution, and select Add New Project.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
        }
    }
}
```

```
'ConsoleApp1.exe' (CLR v4.0.30319: DefaultDomain): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_32\mscorlib\v4.0_4.0.0.0__b77a5c561934e089\mscorlib.dll'. Skipped loading symbols. Module is optimized
'ConsoleApp1.exe' (CLR v4.0.30319: DefaultDomain): Loaded 'C:\Users\60536\Desktop\ConsoleApp1\ConsoleApp1\bin\Debug\ConsoleApp1.exe'. Symbols loaded.
The program '[28108] ConsoleApp1.exe' has exited with code 0 (0x0).
```

5. Select Class Library (.NET Framework) under Library type for the project type, and click Next.


Add a new project

Recent project templates

Console App (.NET Framework)	C#
Windows Forms App (.NET Framework)	C#
Console App	C++
Class Library (.NET Framework)	C#
Dynamic-Link Library (DLL)	C++
Empty Project	C++

Search for templates (Alt+S)

C#   All platforms   Library

Class Library (Universal Windows)
A project for creating a managed class library (.dll) for Universal Windows Platform (UWP) apps.
C# Windows Library UWP

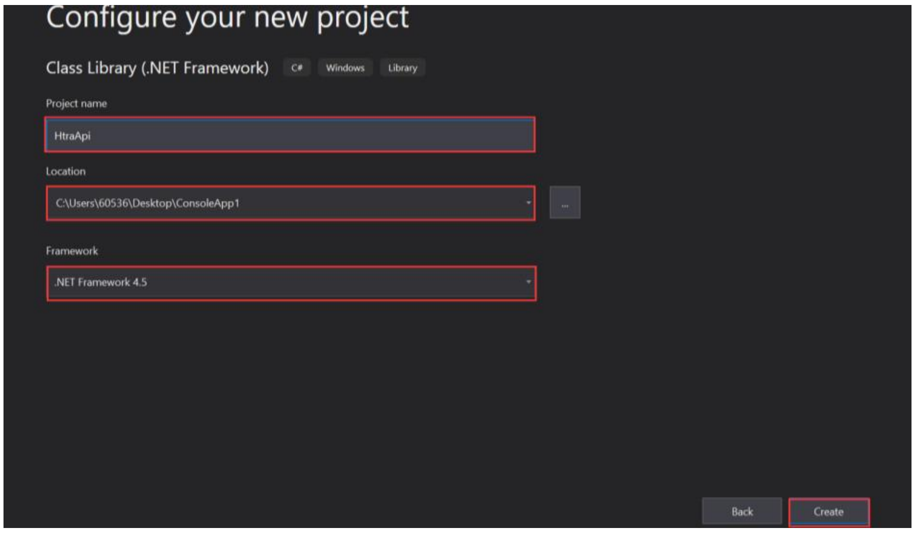
Class Library (.NET Framework)
A project for creating a C# class library (.dll)
C# Windows Library

WPF Custom Control Library (.NET Framework)
Windows Presentation Foundation custom control library
C# XAML Windows Desktop Library

WPF User Control Library (.NET Framework)
Windows Presentation Foundation user control library
C# XAML Windows Desktop Library

Windows Forms Control Library (.NET Framework)
A project for creating controls to use in Windows Forms (WinForms) applications
C# Windows Desktop Library

6. The library name can be modified as needed, such as HtraApi, and the location should not be changed, remaining at the same directory level as the solution. The result is shown in the figure, select the .NET Framework 4.5 framework, and click Create.

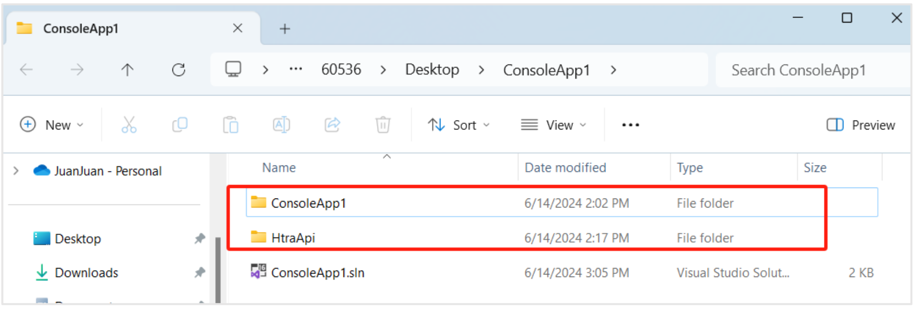

Configure your new project

Class Library (.NET Framework) C# Windows Library

Project name
HtraApi

Location
C:\Users\60536\Desktop\ConsoleApp1

Framework
.NET Framework 4.5

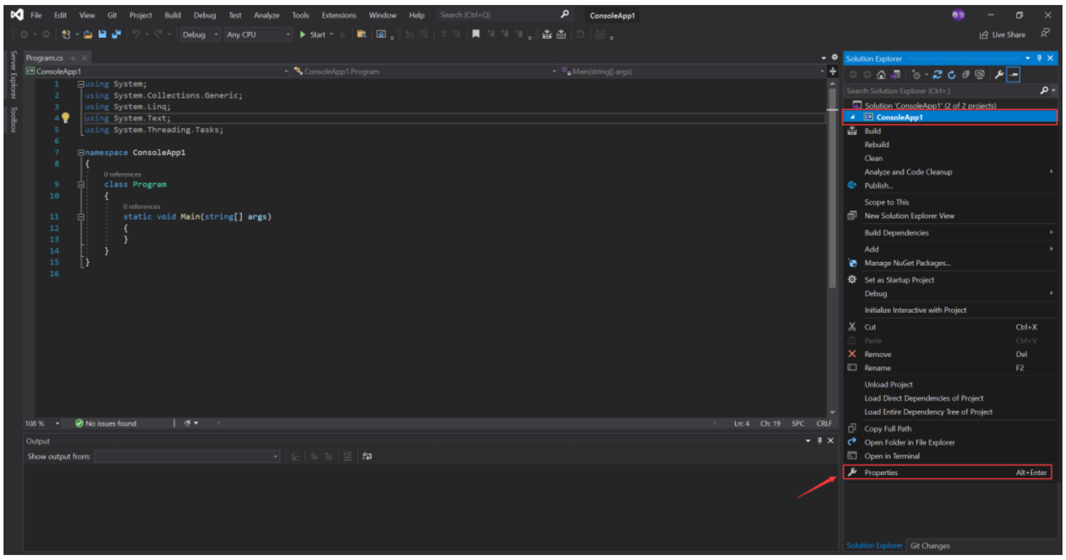

ConsoleApp1

60536 › Desktop › ConsoleApp1 ›

Search ConsoleApp1

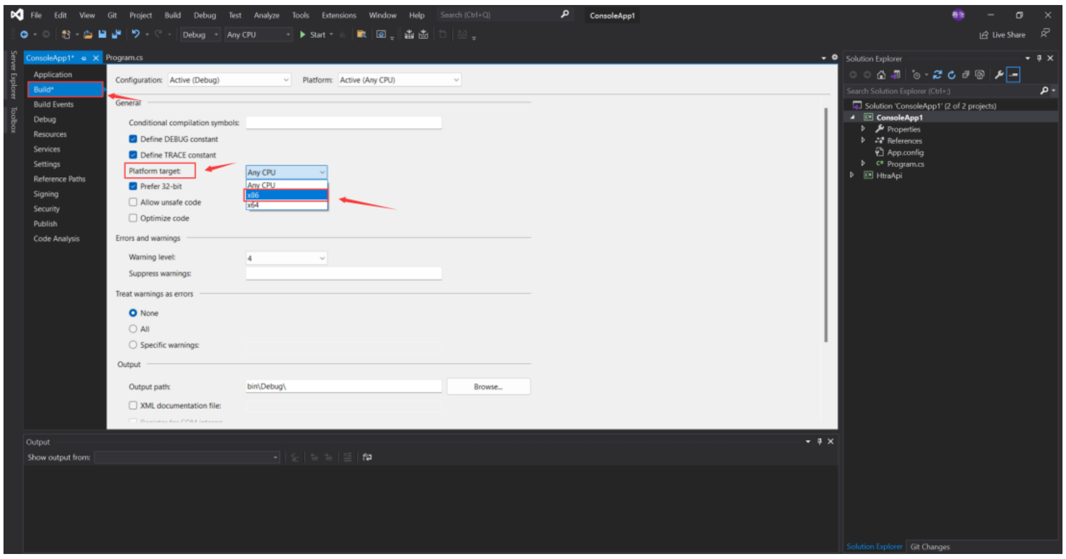
Name	Date modified	Type	Size
ConsoleApp1	6/14/2024 2:02 PM	File folder	
HtraApi	6/14/2024 2:17 PM	File folder	
ConsoleApp1.sln	6/14/2024 3:05 PM	Visual Studio Solut...	2 KB

7. Right-click on ConsoleApp1 and select Properties.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

8. View the project's build properties, change the target platform to x86, click Debug, enter htra_api\ in the working directory, and save. If the situation in Figure 12 occurs, click OK and save again.



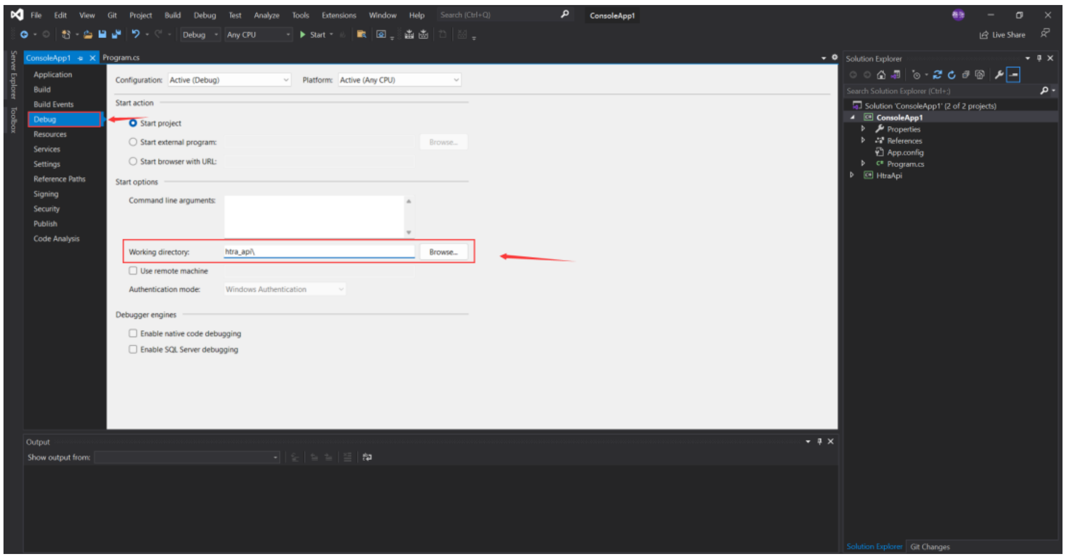
Configuration: Active (Debug) Platform: Active (Any CPU)

General
Conditional compilation symbols:
☑ Define DEBUG constant
☑ Define TRACE constant
Platform target: Any CPU (Any CPU / x86 / x64)
☑ Prefer 32-bit
☐ Allow unsafe code
☐ Optimize code

Errors and warnings
Warning level: 4
Suppress warnings:

Treat warnings as errors
● None
○ All
○ Specific warnings:

Output
Output path: bin\Debug\
☐ XML documentation file:



Configuration: Active (Debug)

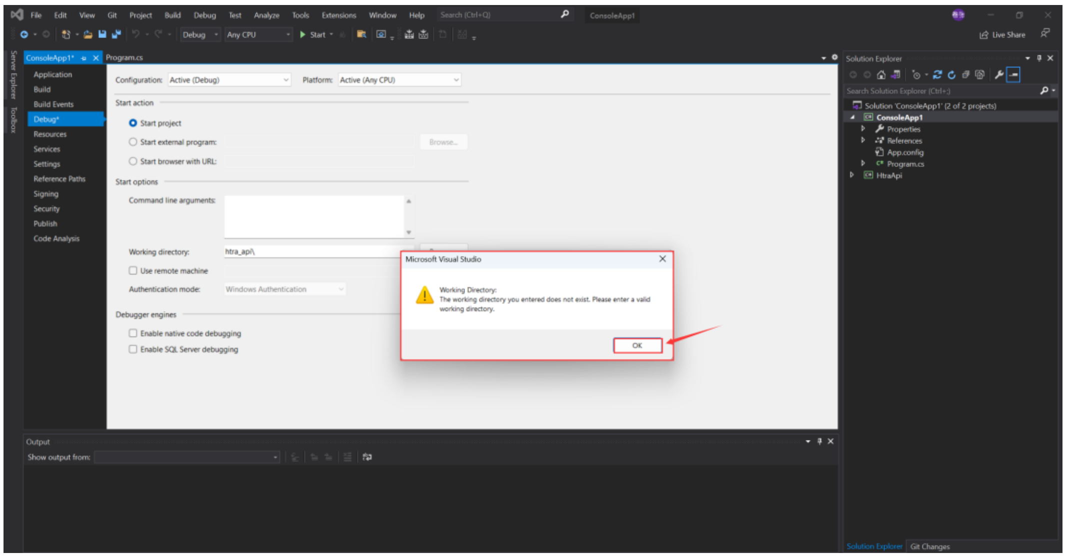
Platform: Active (Any CPU)

Start action

Start project

Working directory: htra_api\

Authentication mode: Windows Authentication

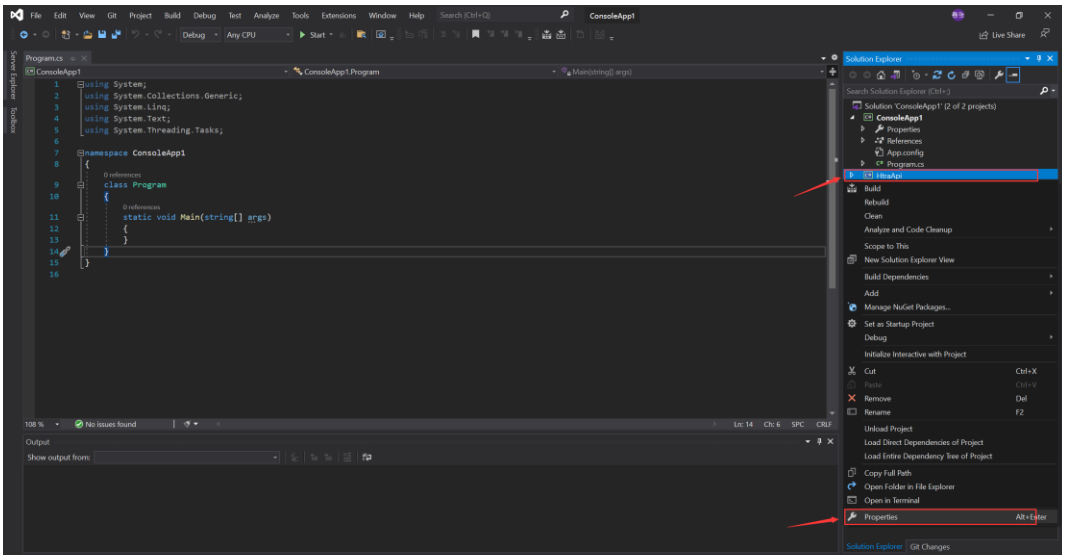


Microsoft Visual Studio

Working Directory:
The working directory you entered does not exist. Please enter a valid working directory.

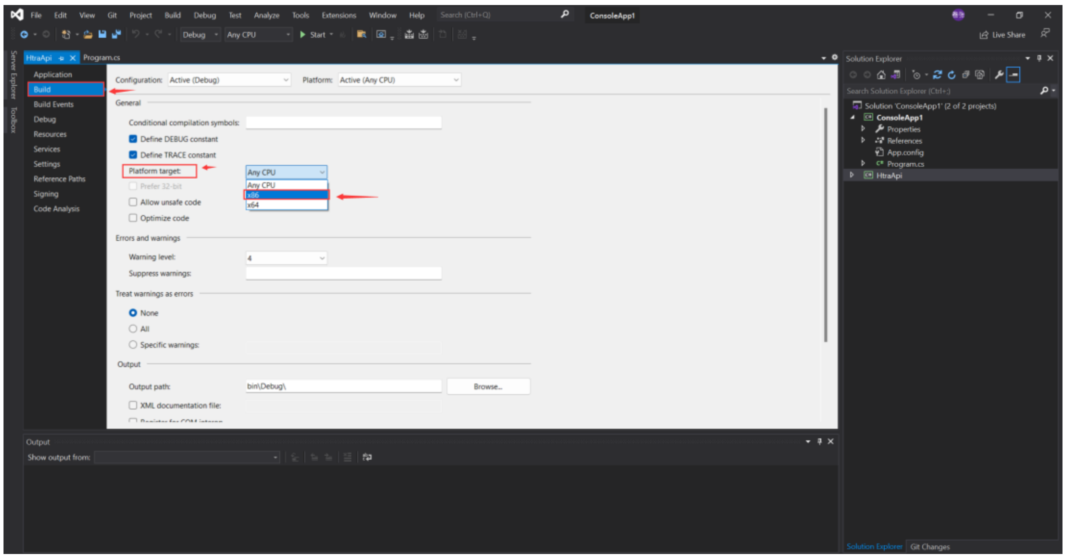
OK

9. Right-click on the library HtraApi and select Properties.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
        }
    }
}
```

10. View the library's build properties, change the target platform to x86, and save.



Configuration: Active (Debug) Platform: Active (Any CPU)

General

Conditional compilation symbols:

☑ Define DEBUG constant

☑ Define TRACE constant

Platform target: Any CPU

- Any CPU
- x86
- x64

☐ Prefer 32-bit

☐ Allow unsafe code

☐ Optimize code

Errors and warnings

Warning level: 4

Suppress warnings:

Treat warnings as errors

● None

○ All

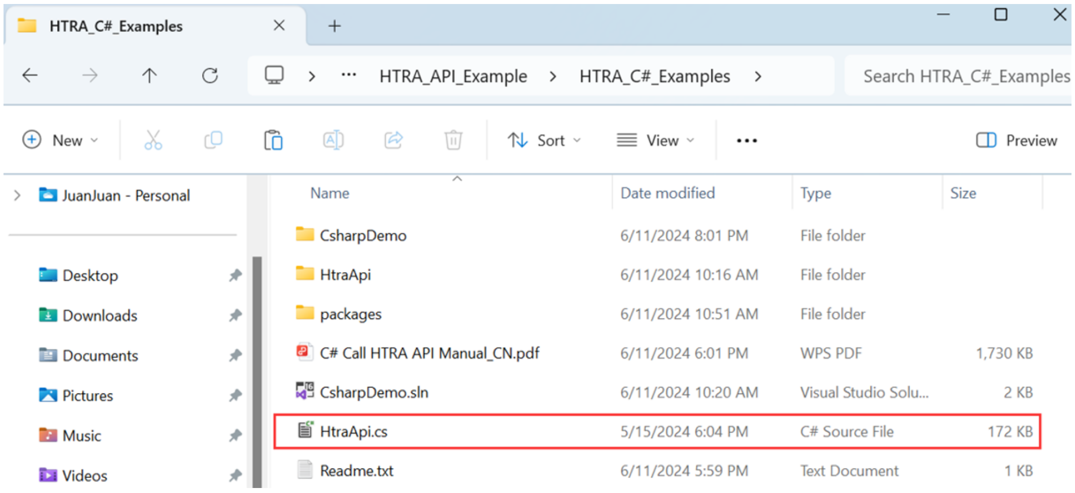
○ Specific warnings:

Output

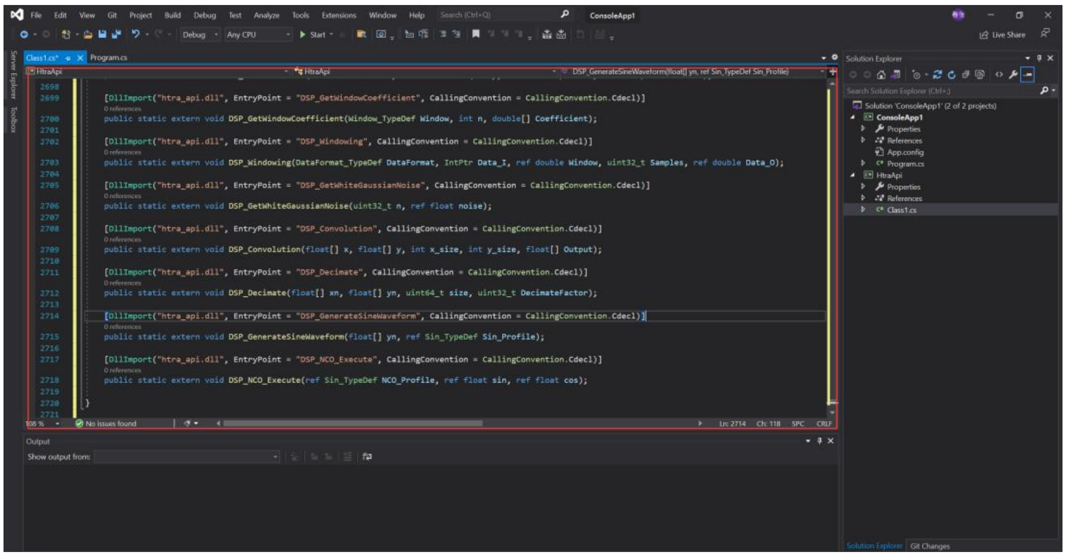
Output path: bin\Debug\ Browse...

☐ XML documentation file:

11. Copy the contents of the HtraApi.cs file from the folder \Windows\HTRA_API_Example\HTRA_C#_Examples included in the accompanying materials to the Class1.cs file in the project library and save.

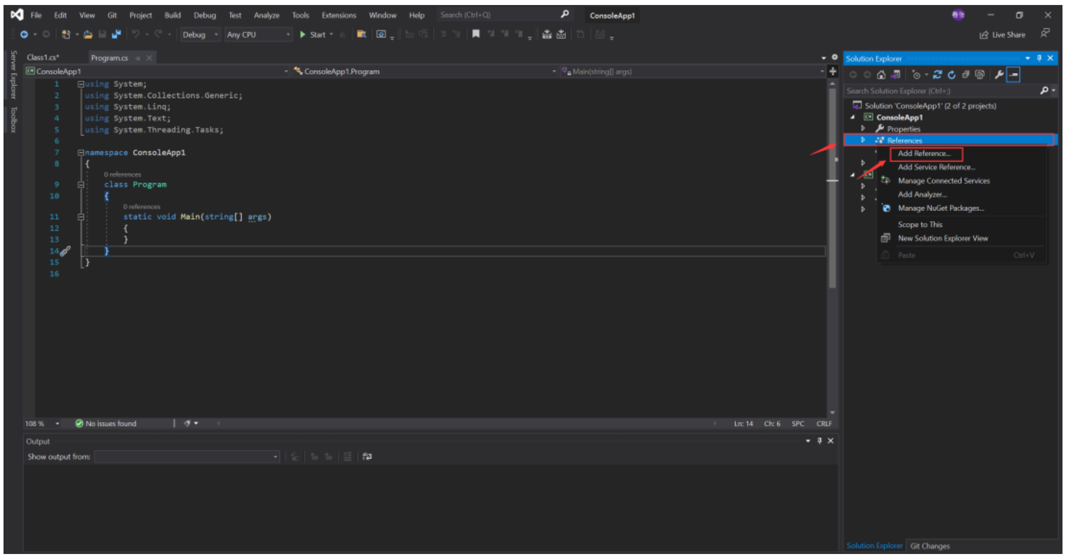


Name	Date modified	Type	Size
CsharpDemo	6/11/2024 8:01 PM	File folder	
HtraApi	6/11/2024 10:16 AM	File folder	
packages	6/11/2024 10:51 AM	File folder	
C# Call HTRA API Manual_CN.pdf	6/11/2024 6:01 PM	WPS PDF	1,730 KB
CsharpDemo.sln	6/11/2024 10:20 AM	Visual Studio Solu...	2 KB
HtraApi.cs	5/15/2024 6:04 PM	C# Source File	172 KB
Readme.txt	6/11/2024 5:59 PM	Text Document	1 KB



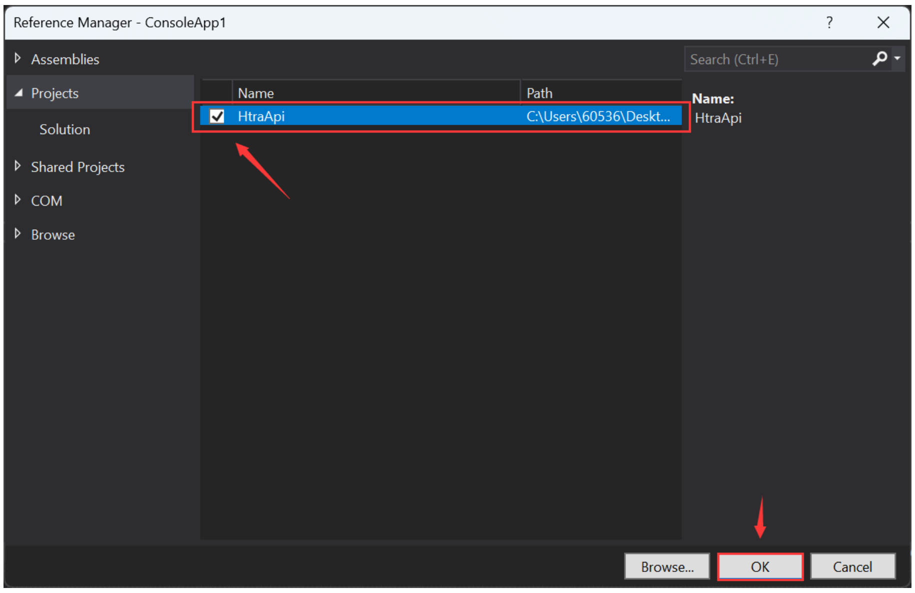
```
        [DllImport("htra_api.dll", EntryPoint = "DSP_GetWindowCoefficient", CallingConvention = CallingConvention.Cdecl)]
        0 references
        public static extern void DSP_GetWindowCoefficient(Window_TypeDef Window, int n, double[] Coefficient);

        [DllImport("htra_api.dll", EntryPoint = "DSP_Windowing", CallingConvention = CallingConvention.Cdecl)]
        0 references
        public static extern void DSP_Windowing(DataFormat_TypeDef DataFormat, IntPtr Data_I, ref double Window, uint32_t Samples, ref double Data_O);

        [DllImport("htra_api.dll", EntryPoint = "DSP_GetWhiteGaussianNoise", CallingConvention = CallingConvention.Cdecl)]
        0 references
        public static extern void DSP_GetWhiteGaussianNoise(uint32_t n, ref float noise);

        [DllImport("htra_api.dll", EntryPoint = "DSP_Convolution", CallingConvention = CallingConvention.Cdecl)]
        0 references
        public static extern void DSP_Convolution(float[] x, float[] y, int x_size, int y_size, float[] Output);

        [DllImport("htra_api.dll", EntryPoint = "DSP_Decimate", CallingConvention = CallingConvention.Cdecl)]
        0 references
        public static extern void DSP_Decimate(float[] xn, float[] yn, uint64_t size, uint32_t DecimateFactor);

        [DllImport("htra_api.dll", EntryPoint = "DSP_GenerateSineWaveform", CallingConvention = CallingConvention.Cdecl)]
        0 references
        public static extern void DSP_GenerateSineWaveform(float[] yn, ref Sin_TypeDef Sin_Profile);

        [DllImport("htra_api.dll", EntryPoint = "DSP_NCO_Execute", CallingConvention = CallingConvention.Cdecl)]
        0 references
        public static extern void DSP_NCO_Execute(ref Sin_TypeDef NCO_Profile, ref float sin, ref float cos);

    }
```

12. Select the ConsoleApp1 project, right-click on References, and choose Add Reference.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
        }
    }
}
```

13. Select the HtraApi library and confirm the addition of the library reference.

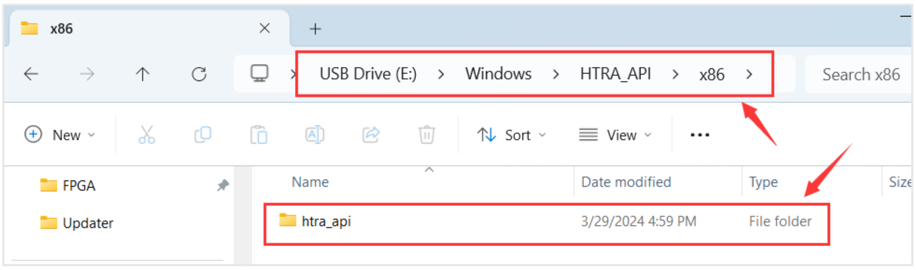


Reference Manager - ConsoleApp1

Name	Path
☑ HtraApi	C:\Users\60536\Deskt...

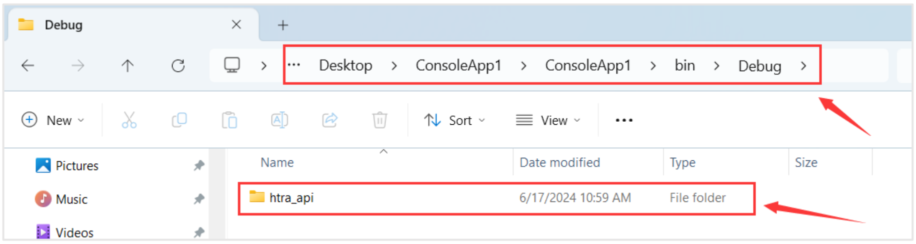
Name: HtraApi

14. Copy the htra_api folder from \Windows\HTRA_API\x86 on the accompanying USB drive to the Debug folder under the project's bin folder, and ensure that the CalFile folder within the htra_api folder contains the calibration files.

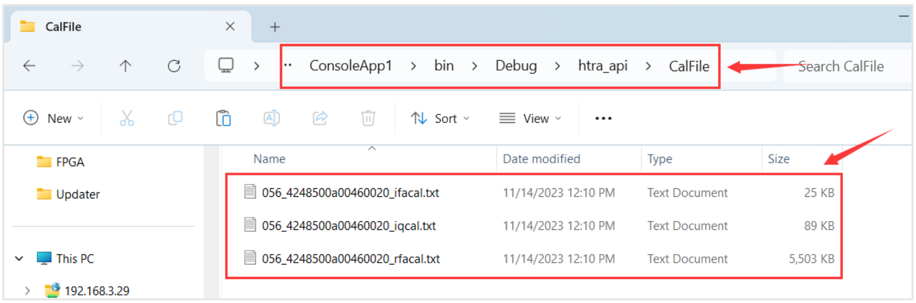


USB Drive (E:) › Windows › HTRA_API › x86 ›

Name	Date modified	Type	Size
htra_api	3/29/2024 4:59 PM	File folder	

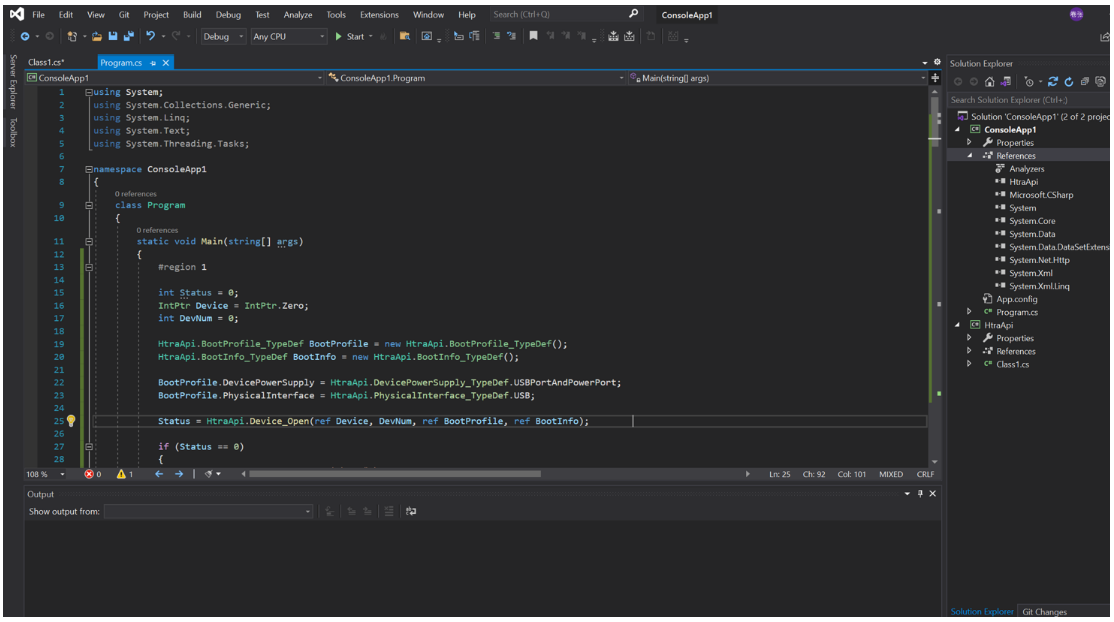

Desktop › ConsoleApp1 › ConsoleApp1 › bin › Debug ›

Name	Date modified	Type	Size
htra_api	6/17/2024 10:59 AM	File folder	


ConsoleApp1 › bin › Debug › htra_api › CalFile

Name	Date modified	Type	Size
056_4248500a00460020_ifacal.txt	11/14/2023 12:10 PM	Text Document	25 KB
056_4248500a00460020_iqcal.txt	11/14/2023 12:10 PM	Text Document	89 KB
056_4248500a00460020_rfacal.txt	11/14/2023 12:10 PM	Text Document	5,503 KB

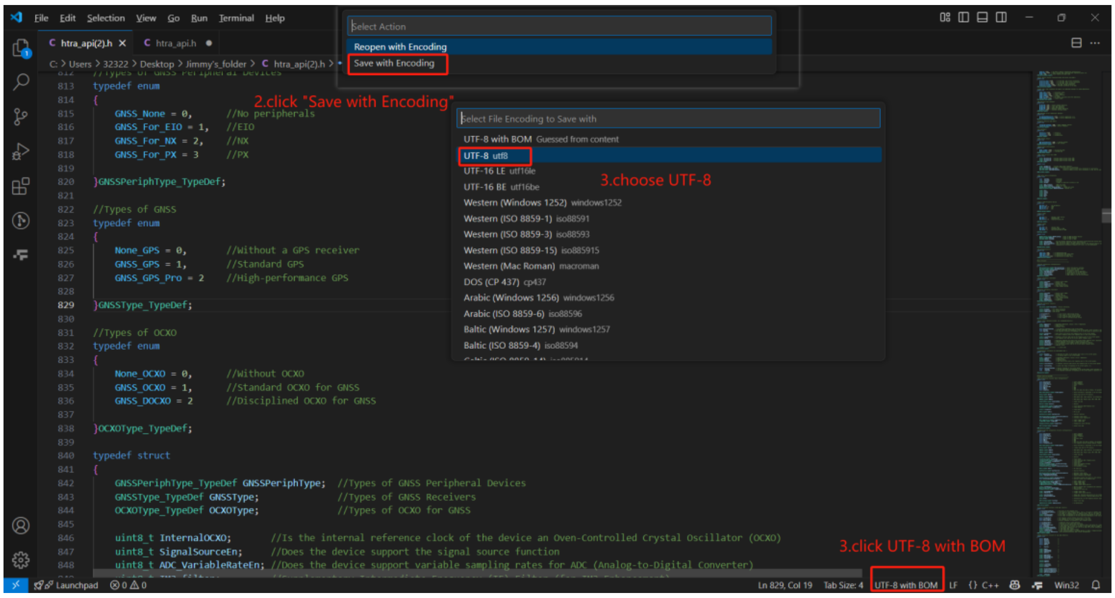
15. Write code normally. You can refer to the C# examples included in the accompanying USB drive, specifically the projects in \Windows\HTRA_API_Example\HTRA_C#_Examples.


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            #region 1

            int Status = 0;
            IntPtr Device = IntPtr.Zero;
            int DevNum = 0;

            HtraApi.BootProfile_TypeDef BootProfile = new HtraApi.BootProfile_TypeDef();
            HtraApi.BootInfo_TypeDef BootInfo = new HtraApi.BootInfo_TypeDef();

            BootProfile.DevicePowerSupply = HtraApi.DevicePowerSupply_TypeDef.USBPortAndPowerPort;
            BootProfile.PhysicalInterface = HtraApi.PhysicalInterface_TypeDef.USB;

            Status = HtraApi.Device_Open(ref Device, DevNum, ref BootProfile, ref BootInfo);

            if (Status == 0)
            {
```

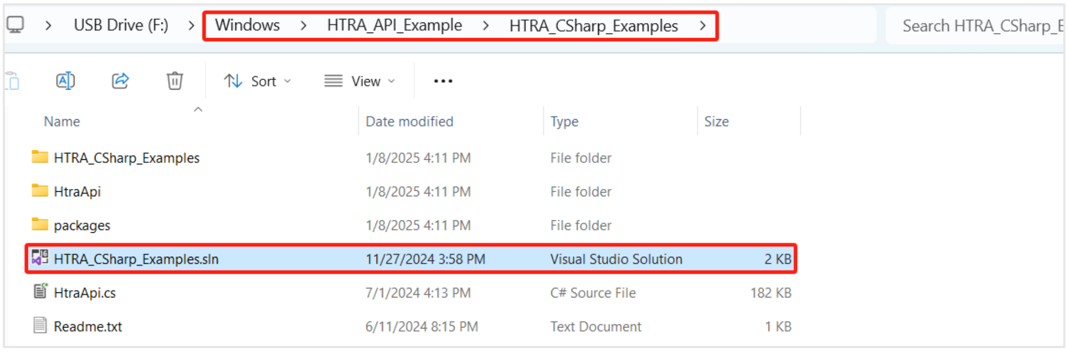
6.2 C# Example Usage Process

The usage process of the C# examples included in the USB drive is as follows:

1. Open the USB drive using Visual Studio and navigate to the folder Windows\HTRA_API_Example\HTRA_CSharp_Examples, then open the solution file HTRA_CSharp_Examples.sln.

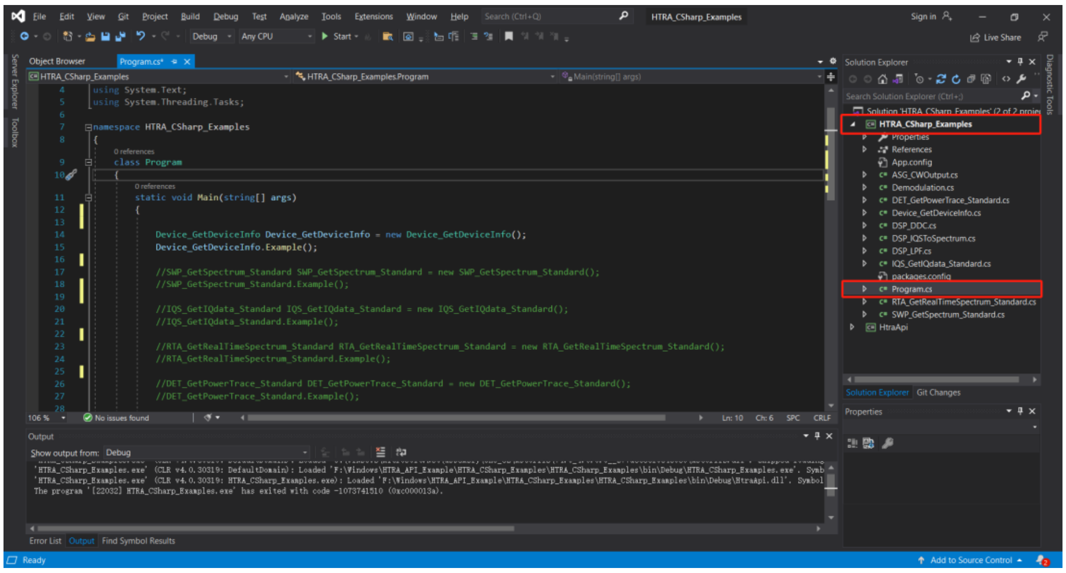

File Edit Selection View Go Run Terminal Help
htra_api(2).h
htra_api.h
C: > Users > 32322 > Desktop > Jimmy's_folder > htra_api(2).h
Select Action
Reopen with Encoding
Save with Encoding
2.click "Save with Encoding"
Select File Encoding to Save with
UTF-8 with BOM Guessed from content
UTF-8 utf8
UTF-16 LE utf16le
UTF-16 BE utf16be
Western (Windows 1252) windows1252
Western (ISO 8859-1) iso88591
Western (ISO 8859-3) iso88593
Western (ISO 8859-15) iso885915
Western (Mac Roman) macroman
DOS (CP 437) cp437
Arabic (Windows 1256) windows1256
Arabic (ISO 8859-6) iso88596
Baltic (Windows 1257) windows1257
Baltic (ISO 8859-4) iso88594
3.choose UTF-8
3.click UTF-8 with BOM
Launchpad
Ln 829, Col 19
Tab Size: 4
UTF-8 with BOM
LF
C++
Win32

```
typedef enum
{
    GNSS_None = 0,      //No peripherals
    GNSS_For_EIO = 1,   //EIO
    GNSS_For_NX = 2,    //NX
    GNSS_For_PX = 3     //PX

}GNSSPeriphType_TypeDef;

//Types of GNSS
typedef enum
{
    None_GPS = 0,       //Without a GPS receiver
    GNSS_GPS = 1,       //Standard GPS
    GNSS_GPS_Pro = 2    //High-performance GPS

}GNSSType_TypeDef;

//Types of OCXO
typedef enum
{
    None_OCXO = 0,      //Without OCXO
    GNSS_OCXO = 1,      //Standard OCXO for GNSS
    GNSS_DOCXO = 2      //Disciplined OCXO for GNSS

}OCXOType_TypeDef;

typedef struct
{
    GNSSPeriphType_TypeDef GNSSPeriphType;  //Types of GNSS Peripheral Devices
    GNSSType_TypeDef GNSSType;              //Types of GNSS Receivers
    OCXOType_TypeDef OCXOType;              //Types of OCXO for GNSS

    uint8_t InternalOCXO;      //Is the internal reference clock of the device an Oven-Controlled Crystal Oscillator (OCXO)
    uint8_t SignalSourceEn;    //Does the device support the signal source function
    uint8_t ADC_VariableRateEn; //Does the device support variable sampling rates for ADC (Analog-to-Digital Converter)
```


USB Drive (F:) > Windows > HTRA_API_Example > HTRA_CSharp_Examples >
Search HTRA_CSharp_E
Sort
View

Name	Date modified	Type	Size
HTRA_CSharp_Examples	1/8/2025 4:11 PM	File folder	
HtraApi	1/8/2025 4:11 PM	File folder	
packages	1/8/2025 4:11 PM	File folder	
HTRA_CSharp_Examples.sln	11/27/2024 3:58 PM	Visual Studio Solution	2 KB
HtraApi.cs	7/1/2024 4:13 PM	C# Source File	182 KB
Readme.txt	6/11/2024 8:15 PM	Text Document	1 KB

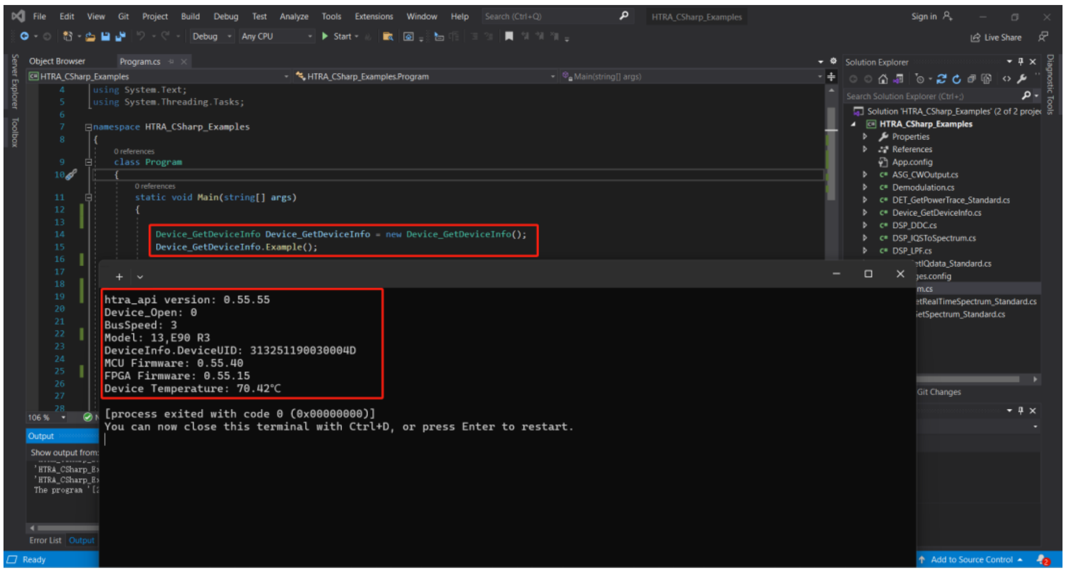
2. Click on the HTRA_CSharp_Examples project on the right side, and then click on the Programe.cs file.



```
using System.Text;
using System.Threading.Tasks;

namespace HTRA_CSharp_Examples
{
    class Program
    {
        static void Main(string[] args)
        {

            Device_GetDeviceInfo Device_GetDeviceInfo = new Device_GetDeviceInfo();
            Device_GetDeviceInfo.Example();

            //SWP_GetSpectrum_Standard SWP_GetSpectrum_Standard = new SWP_GetSpectrum_Standard();
            //SWP_GetSpectrum_Standard.Example();

            //IQS_GetIQdata_Standard IQS_GetIQdata_Standard = new IQS_GetIQdata_Standard();
            //IQS_GetIQdata_Standard.Example();

            //RTA_GetRealTimeSpectrum_Standard RTA_GetRealTimeSpectrum_Standard = new RTA_GetRealTimeSpectrum_Standard();
            //RTA_GetRealTimeSpectrum_Standard.Example();

            //DET_GetPowerTrace_Standard DET_GetPowerTrace_Standard = new DET_GetPowerTrace_Standard();
            //DET_GetPowerTrace_Standard.Example();
```

```
'HTRA_CSharp_Examples.exe' (CLR v4.0.30319: DefaultDomain): Loaded 'F:\Windows\HTRA_API_Example\HTRA_CSharp_Examples\HTRA_CSharp_Examples\bin\Debug\HTRA_CSharp_Examples.exe'. Symb
'HTRA_CSharp_Examples.exe' (CLR v4.0.30319: HTRA_CSharp_Examples.exe): Loaded 'F:\Windows\HTRA_API_Example\HTRA_CSharp_Examples\HTRA_CSharp_Examples\bin\Debug\HtraApi.dll'. Symbol
The program '[22032] HTRA_CSharp_Examples.exe' has exited with code -1073741510 (0xc000013a).
```

3. Since each example in the C# included examples is encapsulated in a separate class, you can run the examples by uncommenting them (multiple examples cannot be used simultaneously). For instance, when testing the Device_GetDeviceInfo example, uncomment it and click run; as shown in the figure, the device is running normally.



```
            Device_GetDeviceInfo Device_GetDeviceInfo = new Device_GetDeviceInfo();
            Device_GetDeviceInfo.Example();
```

```
htra_api version: 0.55.55
Device_Open: 0
BusSpeed: 3
Model: 13,E90 R3
DeviceInfo.DeviceUID: 313251190030004D
MCU Firmware: 0.55.40
FPGA Firmware: 0.55.15
Device Temperature: 70.42℃

[process exited with code 0 (0x00000000)]
You can now close this terminal with Ctrl+D, or press Enter to restart.
```

6.3 C# Example Descriptions

6.3.1 Get device information

Device_GetDeviceInfo.cs: Retrieves device information including API version, USB version, device model, device UID, MCU version, FPGA version, and device temperature.

6.3.2 Obtain Standard Spectrum Data

SWP_GetSpectrum_Standard.cs: Obtains complete spectrum data within a specified frequency band.

6.3.3 Obtain IQ Data for a Fixed Number of Points or Duration

IQS_GetIQdata_Standard.cs: Acquires IQ data under different trigger modes in IQS mode.

6.3.4 Obtain Power Detection Data for a Fixed Number of Points or Duration

DET_GetPowerTrace_Standard.cs: Obtains power detection data under different trigger modes in DET mode.

6.3.5 Obtain real-time spectrum data for a fixed number of points or duration

RTA_GetRealTimeSpectrum_Standard.cs: Retrieves real-time spectrum data under different trigger modes in RTA mode.

6.3.6 Output single-tone signal

ASG_CWOutput.cs: Devices with signal source functionality options output single-tone signals, frequency sweep signals, or power sweep signals through ASG functionality.

6.3.7 AM/FM Demodulation

Demodulation.cs: DSP_FMDemod performs FM demodulation and playback of the acquired IQ data. DSP_AMDemod performs AM demodulation and playback of the acquired IQ data.

6.3.8 IQ to Spectrum Data

DSP_IQSToSpectrum.cs: Converts the IQ data obtained in IQS mode into spectrum data.

6.3.9 Low-pass filtering

DSP_LPF.cs: Applies low-pass filtering to the acquired IQ data and converts it to spectrum.

6.3.10 Digital Downconversion

DSP_DDC.cs: Performs digital down-conversion on the acquired IQ data and converts it to spectrum.

6.3.11 Phase noise test

DSP_TraceAnalysis_PhaseNoise.cs: Demonstration of Phase Noise Test Function.

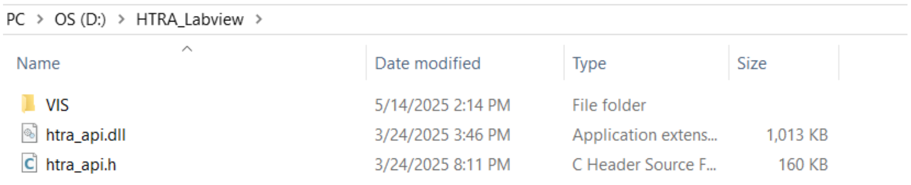
7. Java (to be supplemented)

8. Labview

8.1 Configure Development Environment

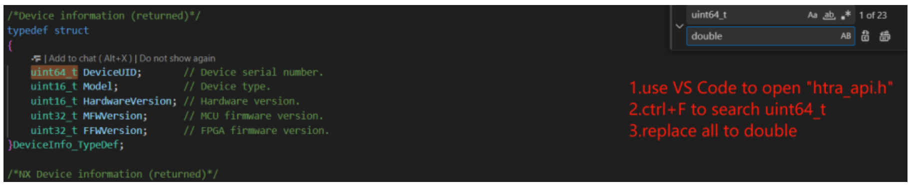
8.1.1 Export library functions from htra_api.dll using LabVIEW

1. Create a folder (e.g., HTRA_Labview) and copy the htra_api folder from the USB drive located at Windows\HTRA_API\x86 into this folder. Then create another folder (e.g., VIS) to place the exported vi.


PC > OS (D:) > HTRA_Labview >

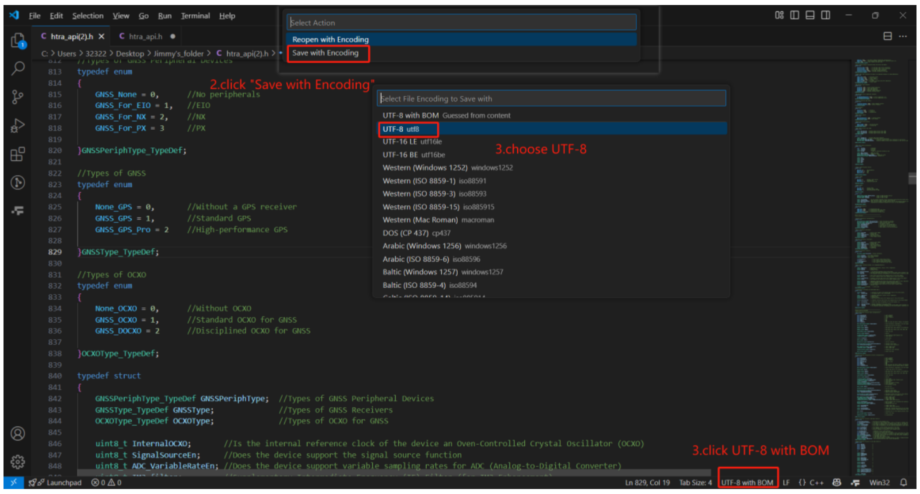
Name	Date modified	Type	Size
VIS	5/14/2025 2:14 PM	File folder	
htra_api.dll	3/24/2025 3:46 PM	Application extens...	1,013 KB
htra_api.h	3/24/2025 8:11 PM	C Header Source F...	160 KB

2. LabVIEW does not recognize the uint64_t and int64_t data types during import, so before importing, all parameters of type uint64_t and int64_t need to be changed to double. Note that after exporting the functions, these modified parameter types should be changed back to uint64_t or int64_t in the VIs.


```
/*Device information (returned)*/
typedef struct
{
    uint64_t DeviceUID;        // Device serial number.
    uint16_t Model;            // Device type.
    uint16_t HardwareVersion;  // Hardware version.
    uint32_t MFWVersion;       // MCU firmware version.
    uint32_t FFWVersion;       // FPGA firmware version.
}DeviceInfo_TypeDef;

/*NX Device information (returned)*/
```

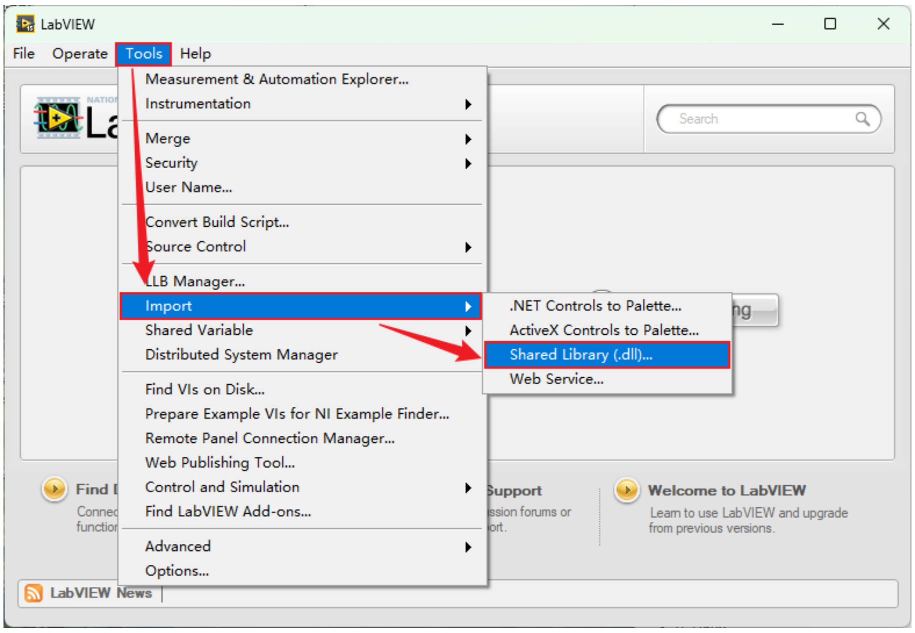
| Add to chat (Alt+X) | Do not show again
uint64_t
1 of 23
double
1.use VS Code to open "htra_api.h"
2.ctrl+F to search uint64_t
3.replace all to double

3. Change the encoding format of htra_api.h to UTF-8.

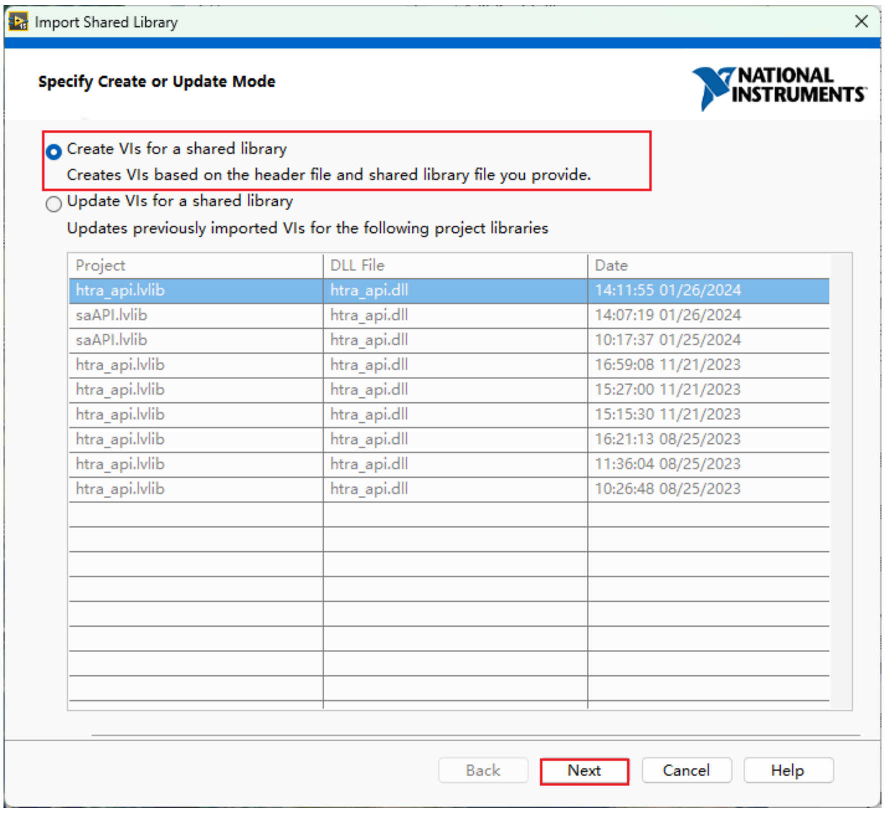

```
typedef enum
{
    GNSS_None = 0,      //No peripherals
    GNSS_For_EIO = 1,   //EIO
    GNSS_For_NX = 2,    //NX
    GNSS_For_PX = 3     //PX

}GNSSPeriphType_TypeDef;

//Types of GNSS
typedef enum
{
    None_GPS = 0,       //Without a GPS receiver
    GNSS_GPS = 1,       //Standard GPS
    GNSS_GPS_Pro = 2    //High-performance GPS

}GNSSType_TypeDef;

//Types of OCXO
typedef enum
{
    None_OCXO = 0,      //Without OCXO
    GNSS_OCXO = 1,      //Standard OCXO for GNSS
    GNSS_DOCXO = 2      //Disciplined OCXO for GNSS

}OCXOType_TypeDef;

typedef struct
{
    GNSSPeriphType_TypeDef GNSSPeriphType;  //Types of GNSS Peripheral Devices
    GNSSType_TypeDef GNSSType;              //Types of GNSS Receivers
    OCXOType_TypeDef OCXOType;              //Types of OCXO for GNSS

    uint8_t InternalOCXO;       //Is the internal reference clock of the device an Oven-Controlled Crystal Oscillator (OCXO)
    uint8_t SignalSourceEn;     //Does the device support the signal source function
    uint8_t ADC_VariableRateEn; //Does the device support variable sampling rates for ADC (Analog-to-Digital Converter)
```

File Edit Selection View Go Run Terminal Help
htra_api(2).h
htra_api.h
C: > Users > 32322 > Desktop > Jimmy's_folder > htra_api(2).h
Select Action
Reopen with Encoding
Save with Encoding
2.click "Save with Encoding"
Select File Encoding to Save with
UTF-8 with BOM Guessed from content
UTF-8 utf8
UTF-16 LE utf16le
UTF-16 BE utf16be
3.choose UTF-8
Western (Windows 1252) windows1252
Western (ISO 8859-1) iso88591
Western (ISO 8859-3) iso88593
Western (ISO 8859-15) iso885915
Western (Mac Roman) macroman
DOS (CP 437) cp437
Arabic (Windows 1256) windows1256
Arabic (ISO 8859-6) iso88596
Baltic (Windows 1257) windows1257
Baltic (ISO 8859-4) iso88594
3.click UTF-8 with BOM
Launchpad
Ln 829, Col 19
Tab Size: 4
UTF-8 with BOM
LF
C++
Win32

4. Open LabVIEW, and select “Tools--->Import--->Shared Library.”



5. Select “Create VIs for a shared library” and click “Next.”


Import Shared Library

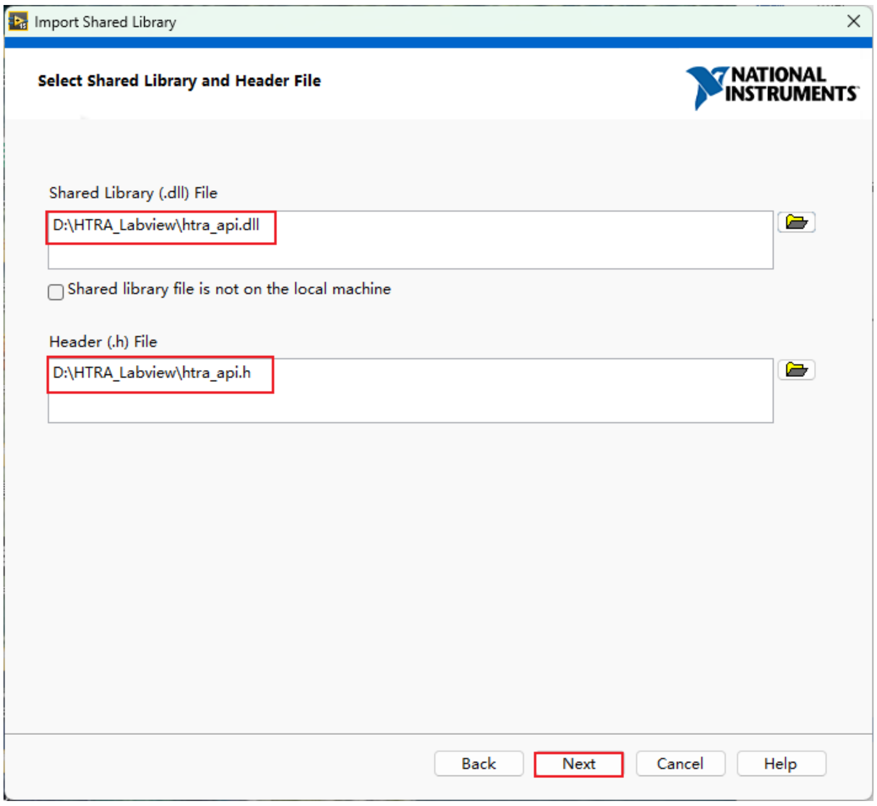
Specify Create or Update Mode

[✓ Create VIs for a shared library]
Creates VIs based on the header file and shared library file you provide.

Update VIs for a shared library
Updates previously imported VIs for the following project libraries

Project	DLL File	Date
htra_api.lvlib	htra_api.dll	14:11:55 01/26/2024
saAPI.lvlib	htra_api.dll	14:07:19 01/26/2024
saAPI.lvlib	htra_api.dll	10:17:37 01/25/2024
htra_api.lvlib	htra_api.dll	16:59:08 11/21/2023
htra_api.lvlib	htra_api.dll	15:27:00 11/21/2023
htra_api.lvlib	htra_api.dll	15:15:30 11/21/2023
htra_api.lvlib	htra_api.dll	16:21:13 08/25/2023
htra_api.lvlib	htra_api.dll	11:36:04 08/25/2023
htra_api.lvlib	htra_api.dll	10:26:48 08/25/2023

6. In the “Shared Library (.dll) File” and “Header (.h) File” sections, select the corresponding library files from the previously created folder. After selecting the shared library file path, the header file path can be automatically recognized and does not need to be selected again. Then click “Next.”


Import Shared Library

Select Shared Library and Header File

Shared Library (.dll) File

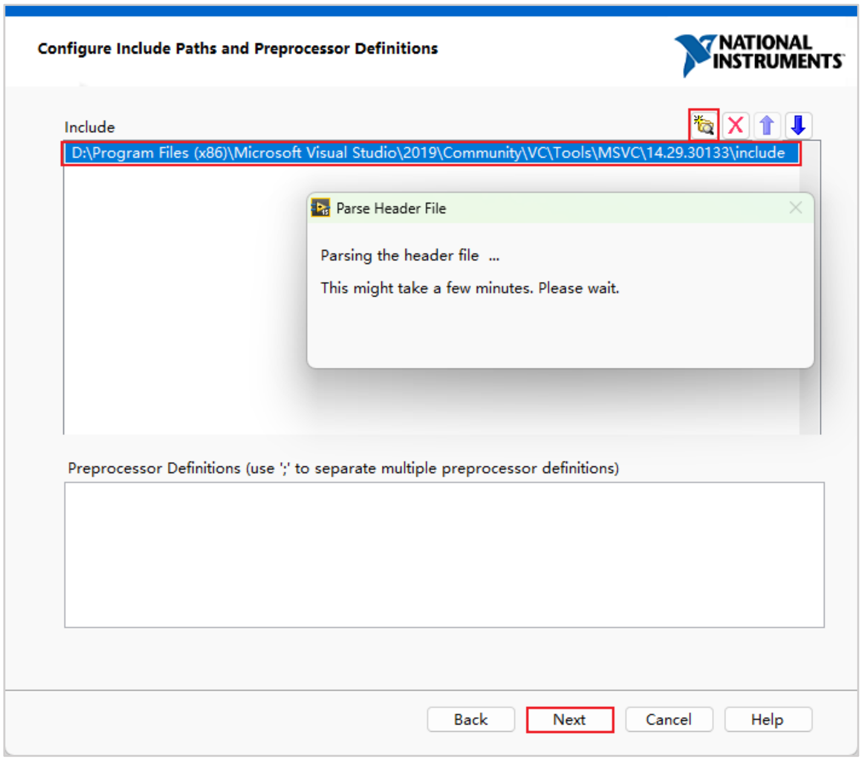
D:\HTRA_Labview\htra_api.dll

☐ Shared library file is not on the local machine

Header (.h) File

D:\HTRA_Labview\htra_api.h

7. In the “Include” section, configure the paths for other dependency files, which are generally located in the folder where Visual Studio is installed, as shown in the image below, and then click “Next” to wait for parsing.


Configure Include Paths and Preprocessor Definitions

Include

D:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30133\include

Parse Header File

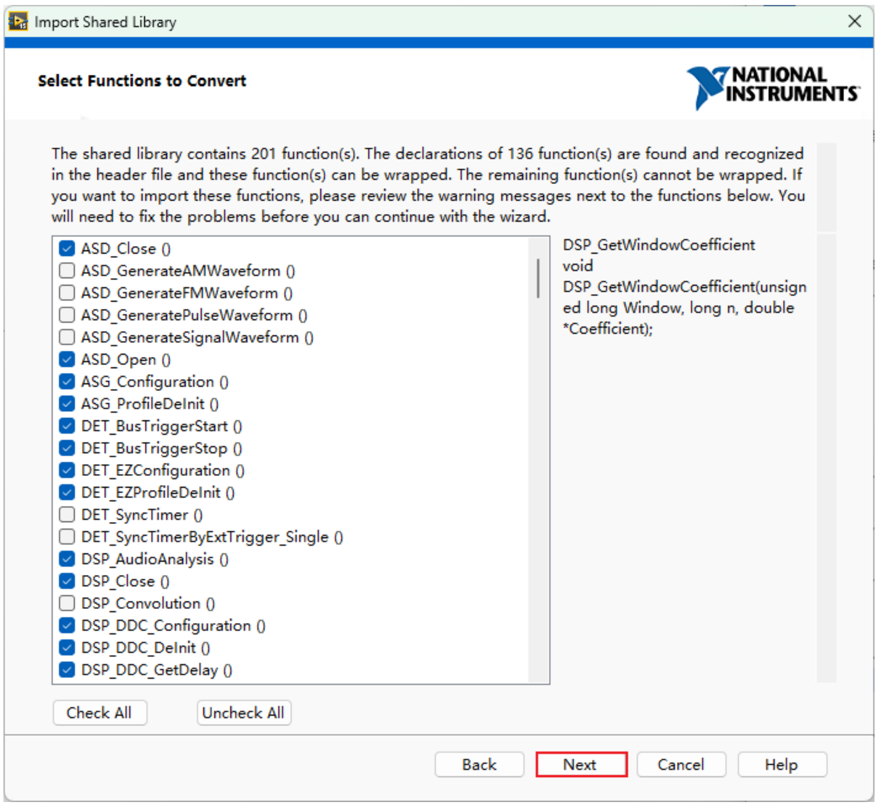
Parsing the header file ...

This might take a few minutes. Please wait.

Preprocessor Definitions (use ';' to separate multiple preprocessor definitions)

8. Select the functions to be exported; some functions have been deprecated or are

not yet available. You can refer to htra_api.h for selection, and after making your selections, click “Next.”


Import Shared Library

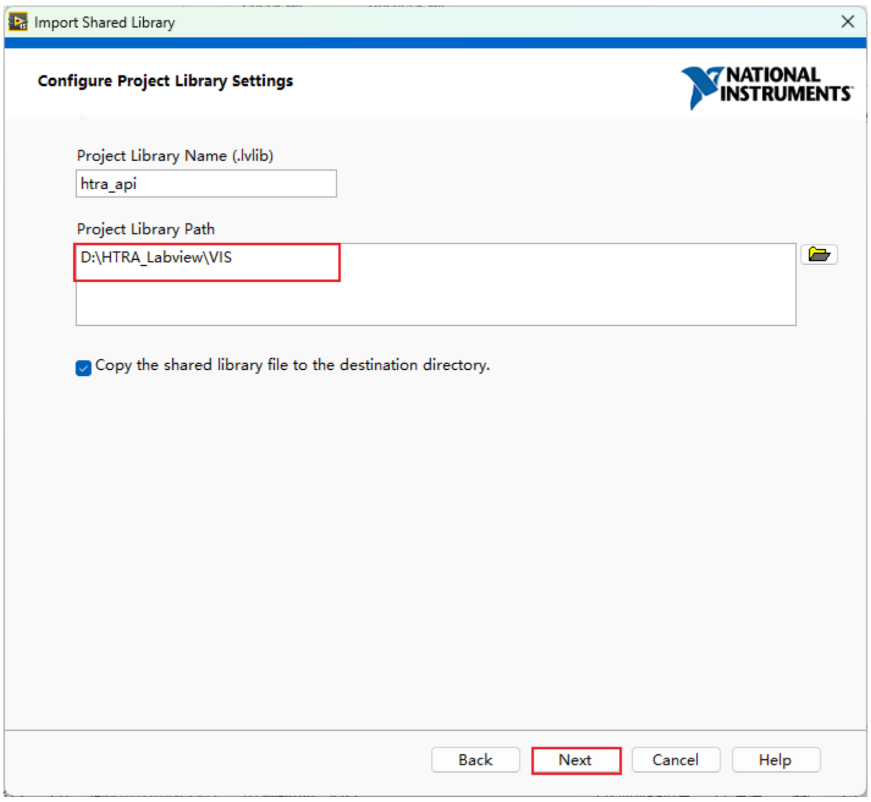
Select Functions to Convert

The shared library contains 201 function(s). The declarations of 136 function(s) are found and recognized in the header file and these function(s) can be wrapped. The remaining function(s) cannot be wrapped. If you want to import these functions, please review the warning messages next to the functions below. You will need to fix the problems before you can continue with the wizard.

- [x] ASD_Close ()
- [ ] ASD_GenerateAMWaveform ()
- [ ] ASD_GenerateFMWaveform ()
- [ ] ASD_GeneratePulseWaveform ()
- [ ] ASD_GenerateSignalWaveform ()
- [x] ASD_Open ()
- [x] ASG_Configuration ()
- [x] ASG_ProfileDeInit ()
- [x] DET_BusTriggerStart ()
- [x] DET_BusTriggerStop ()
- [x] DET_EZConfiguration ()
- [x] DET_EZProfileDeInit ()
- [ ] DET_SyncTimer ()
- [ ] DET_SyncTimerByExtTrigger_Single ()
- [x] DSP_AudioAnalysis ()
- [x] DSP_Close ()
- [ ] DSP_Convolution ()
- [x] DSP_DDC_Configuration ()
- [x] DSP_DDC_DeInit ()
- [x] DSP_DDC_GetDelay ()

DSP_GetWindowCoefficient
void DSP_GetWindowCoefficient(unsigned long Window, long n, double *Coefficient);

9. For “Project Library Path,” select the VIS folder in the library and header file paths, and then click “Next.”

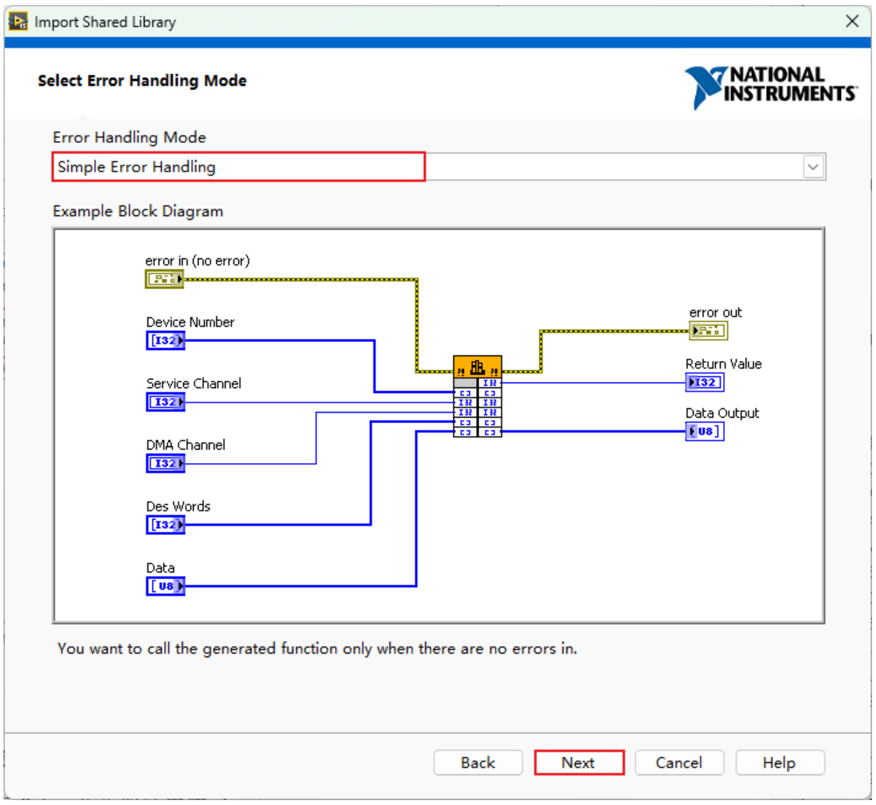


Project Library Name (.lvlib)
htra_api

Project Library Path
D:\HTRA_Labview\VIS

☑ Copy the shared library file to the destination directory.

10. For “Error Handling Mode,” it is recommended to select “Simple Error Handling,” and then click “Next.”

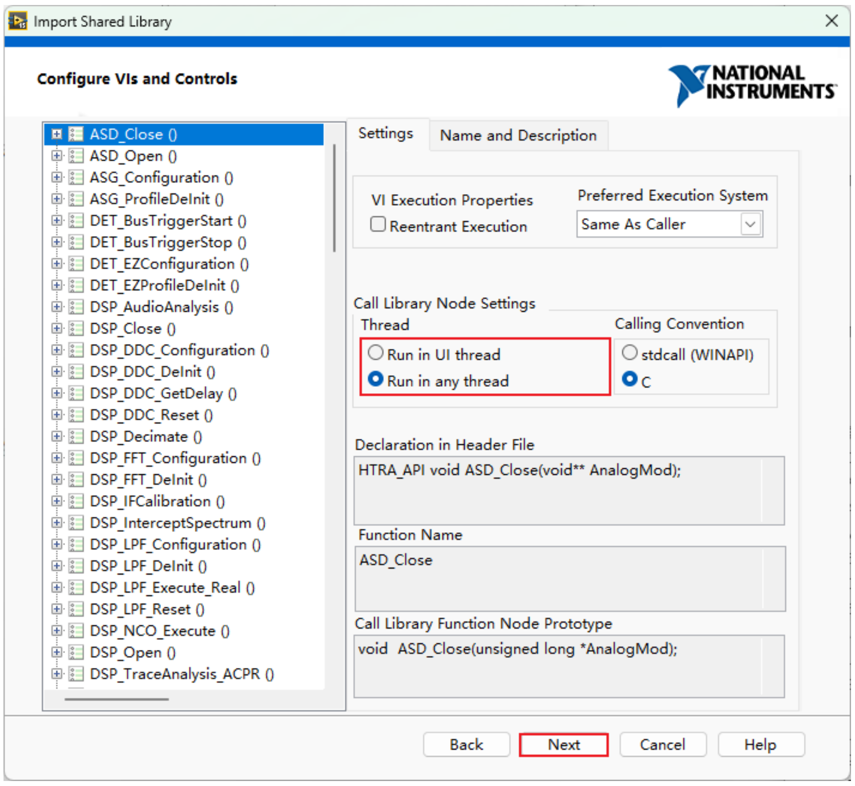


Error Handling Mode
Simple Error Handling

You want to call the generated function only when there are no errors in.

11. Set the call library node for each function to "Run in any thread." After all settings

are complete, click "Next" and wait for the VI to be generated.


Import Shared Library

Configure VIs and Controls

ASD_Close ()
ASD_Open ()
ASG_Configuration ()
ASG_ProfileDeInit ()
DET_BusTriggerStart ()
DET_BusTriggerStop ()
DET_EZConfiguration ()
DET_EZProfileDeInit ()
DSP_AudioAnalysis ()
DSP_Close ()
DSP_DDC_Configuration ()
DSP_DDC_DeInit ()
DSP_DDC_GetDelay ()
DSP_DDC_Reset ()
DSP_Decimate ()
DSP_FFT_Configuration ()
DSP_FFT_DeInit ()
DSP_IFCalibration ()
DSP_InterceptSpectrum ()
DSP_LPF_Configuration ()
DSP_LPF_DeInit ()
DSP_LPF_Execute_Real ()
DSP_LPF_Reset ()
DSP_NCO_Execute ()
DSP_Open ()
DSP_TraceAnalysis_ACPR ()

Settings | Name and Description

VI Execution Properties: Reentrant Execution [ ]
Preferred Execution System: Same As Caller

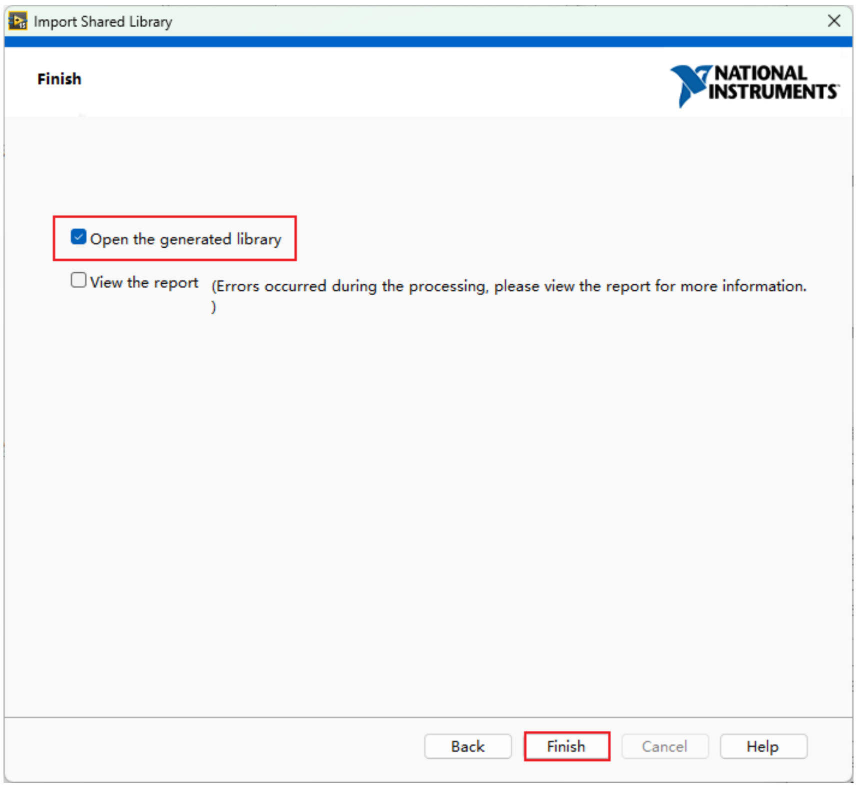
Call Library Node Settings
Thread: Run in UI thread [✓ Run in any thread]
Calling Convention: stdcall (WINAPI) [✓ C]

Declaration in Header File
HTRA_API void ASD_Close(void** AnalogMod);

Function Name
ASD_Close

Call Library Function Node Prototype
void ASD_Close(unsigned long *AnalogMod);

12. Check "Open the generated library," and "View the report" is optional; then click "Finish."

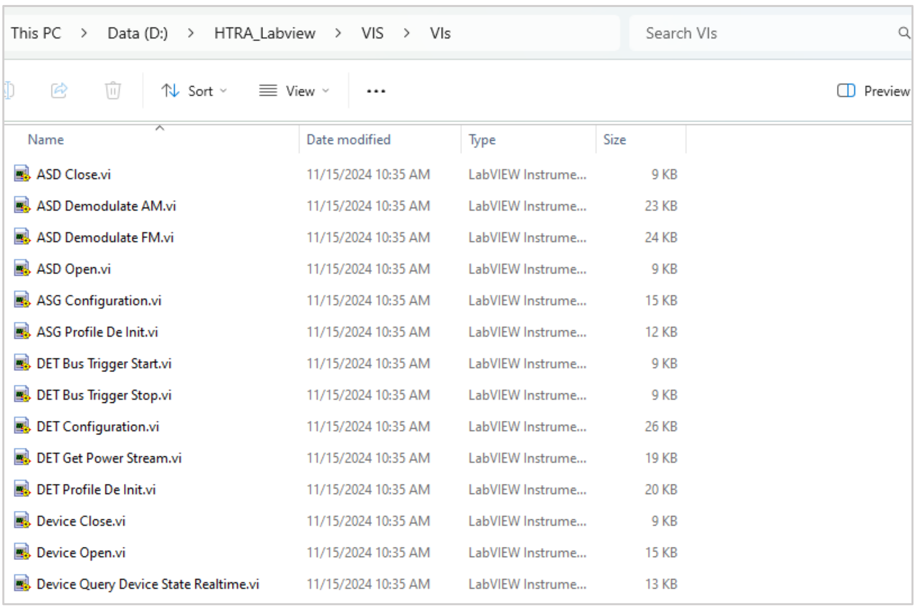

Import Shared Library

Finish

[✓] Open the generated library
[ ] View the report (Errors occurred during the processing, please view the report for more information.
)

13. The vi in the VIS folder is the exported API function. Thus, the export of the API

function is complete.


This PC > Data (D:) > HTRA_Labview > VIS > VIs

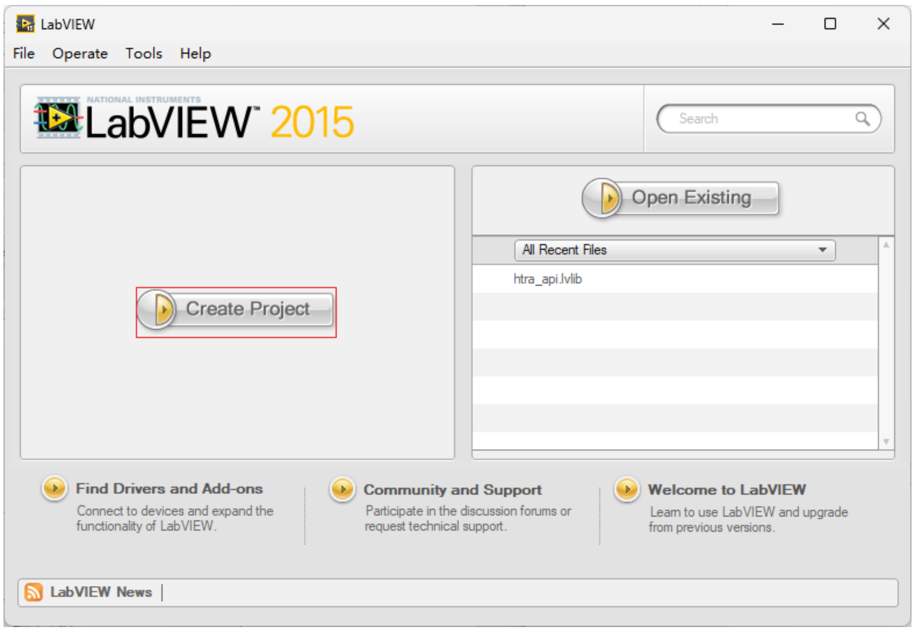
Search VIs

Sort View ··· Preview

Name	Date modified	Type	Size
ASD Close.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	9 KB
ASD Demodulate AM.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	23 KB
ASD Demodulate FM.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	24 KB
ASD Open.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	9 KB
ASG Configuration.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	15 KB
ASG Profile De Init.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	12 KB
DET Bus Trigger Start.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	9 KB
DET Bus Trigger Stop.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	9 KB
DET Configuration.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	26 KB
DET Get Power Stream.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	19 KB
DET Profile De Init.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	20 KB
Device Close.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	9 KB
Device Open.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	15 KB
Device Query Device State Realtime.vi	11/15/2024 10:35 AM	LabVIEW Instrume...	13 KB

8.1.2 Using the API in the LabVIEW environment

1. Open LabVIEW and click "Create Project."

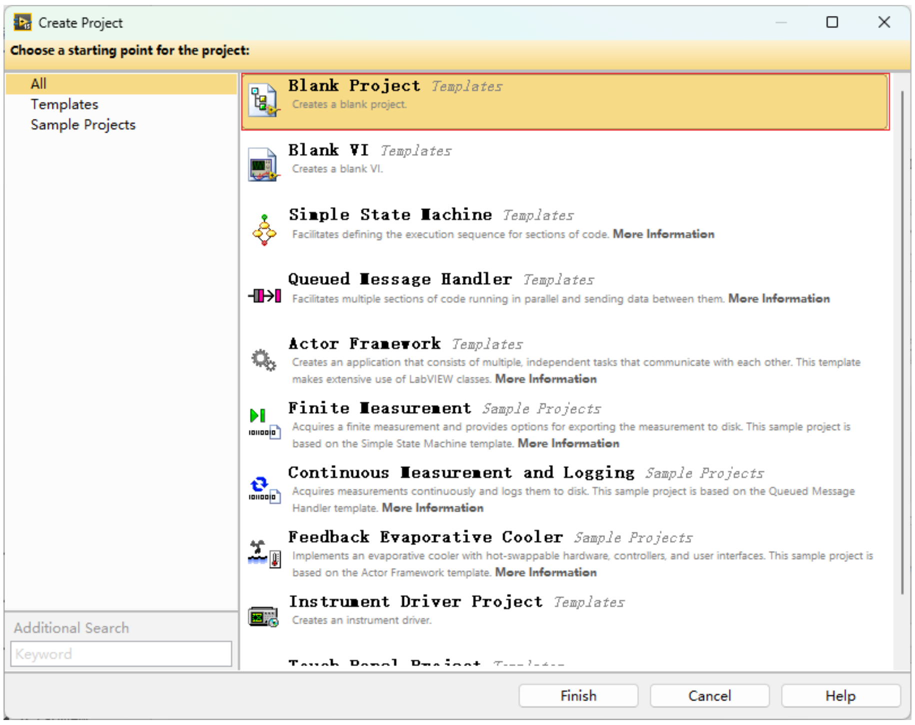


Find Drivers and Add-ons
Connect to devices and expand the functionality of LabVIEW.

Community and Support
Participate in the discussion forums or request technical support.

Welcome to LabVIEW
Learn to use LabVIEW and upgrade from previous versions.

2. Select "Blank Project" and click "Finish."



Choose a starting point for the project:

Blank Project *Templates*
Creates a blank project.

Blank VI *Templates*
Creates a blank VI.

Simple State Machine *Templates*
Facilitates defining the execution sequence for sections of code. **More Information**

Queued Message Handler *Templates*
Facilitates multiple sections of code running in parallel and sending data between them. **More Information**

Actor Framework *Templates*
Creates an application that consists of multiple, independent tasks that communicate with each other. This template makes extensive use of LabVIEW classes. **More Information**

Finite Measurement *Sample Projects*
Acquires a finite measurement and provides options for exporting the measurement to disk. This sample project is based on the Simple State Machine template. **More Information**

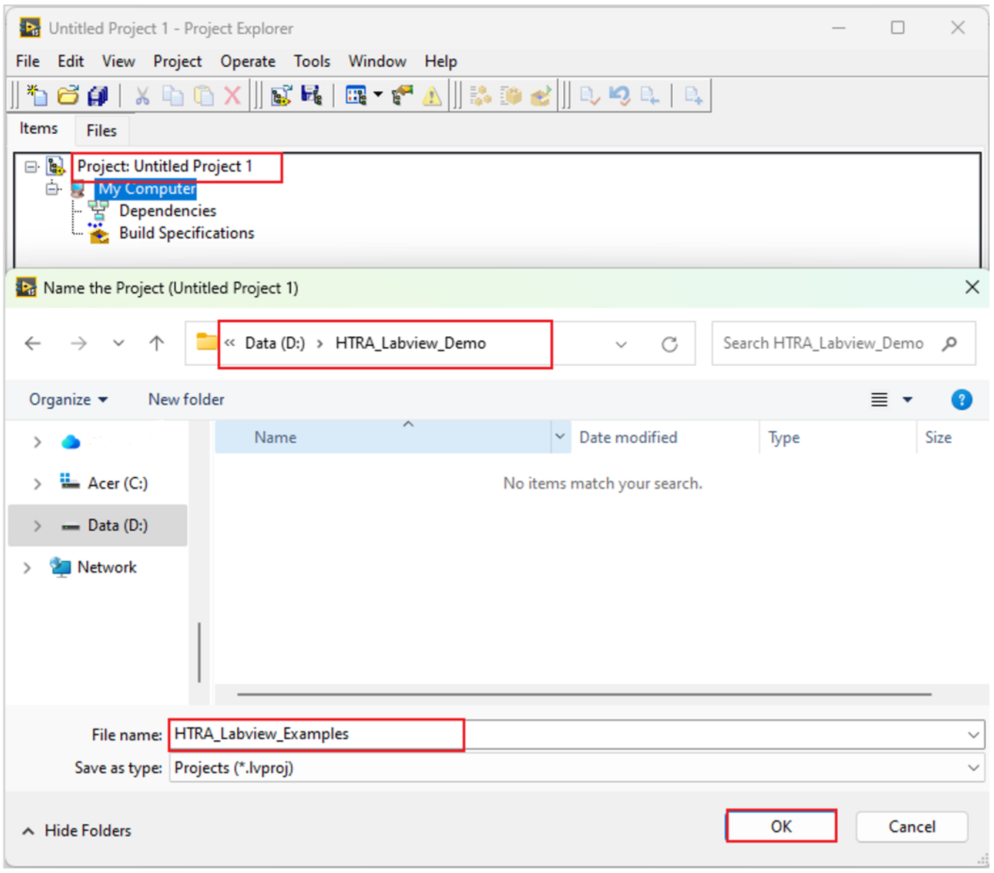
Continuous Measurement and Logging *Sample Projects*
Acquires measurements continuously and logs them to disk. This sample project is based on the Queued Message Handler template. **More Information**

Feedback Evaporative Cooler *Sample Projects*
Implements an evaporative cooler with hot-swappable hardware, controllers, and user interfaces. This sample project is based on the Actor Framework template. **More Information**

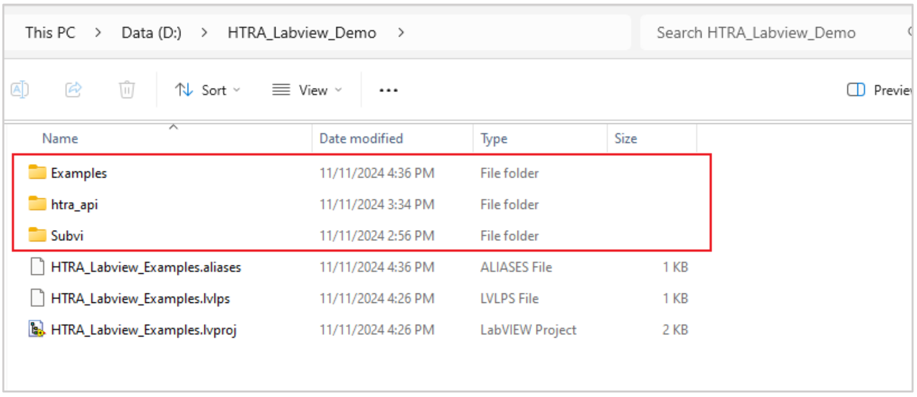
Instrument Driver Project *Templates*
Creates an instrument driver.

Touch Panel Project *Templates*

3. A blank unnamed project will appear; press "Ctrl+S" to save the project, select the project save path, name the project, and then click "OK."

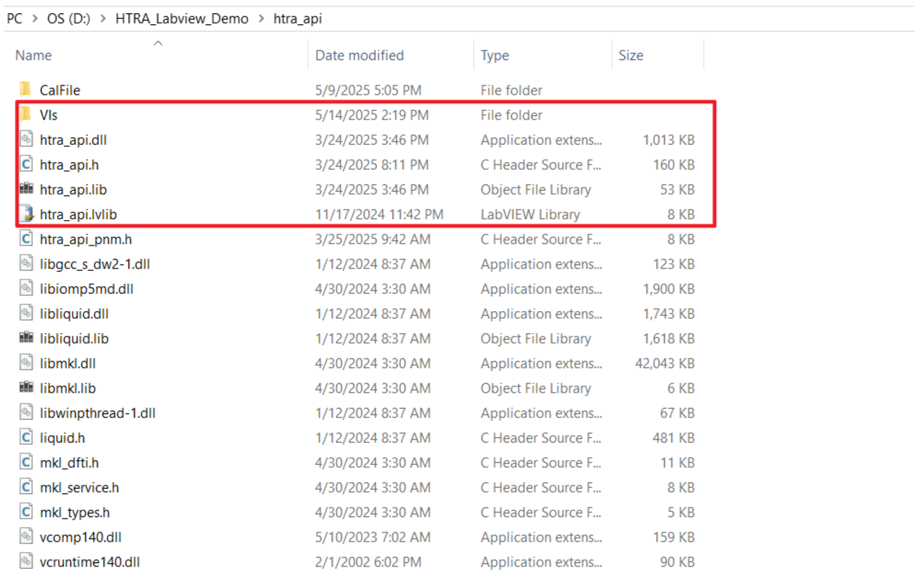

Untitled Project 1 - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Project: Untitled Project 1
My Computer
Dependencies
Build Specifications
Name the Project (Untitled Project 1)
« Data (D:) › HTRA_Labview_Demo
Search HTRA_Labview_Demo
Organize New folder
Name Date modified Type Size
No items match your search.
Acer (C:)
Data (D:)
Network
File name: HTRA_Labview_Examples
Save as type: Projects (*.lvproj)
Hide Folders
OK
Cancel

4. Copy the htra_api folder from the \Windows\HTRA_API\x86 folder on the USB drive to the same directory as the project. Additionally, you can create an Examples folder to store examples. If needed, you can also create a folder named Subvi to store sub vi.



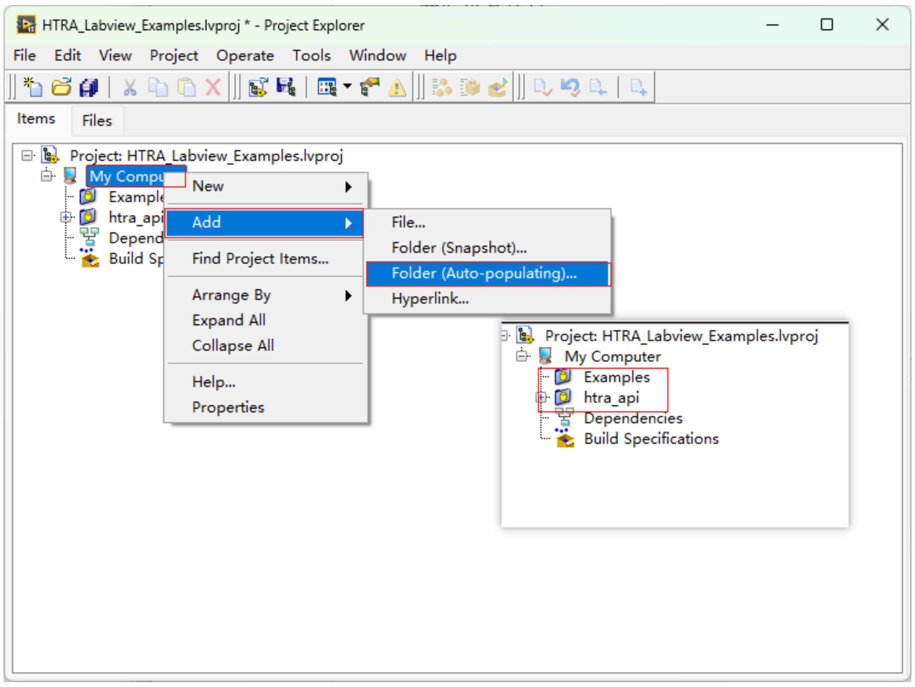
Name	Date modified	Type	Size
Examples	11/11/2024 4:36 PM	File folder	
htra_api	11/11/2024 3:34 PM	File folder	
Subvi	11/11/2024 2:56 PM	File folder	
HTRA_Labview_Examples.aliases	11/11/2024 4:36 PM	ALIASES File	1 KB
HTRA_Labview_Examples.lvlps	11/11/2024 4:26 PM	LVLPS File	1 KB
HTRA_Labview_Examples.lvproj	11/11/2024 4:26 PM	LabVIEW Project	2 KB

5. Copy the previously exported library functions, specifically the contents of the VIS folder created in section 8.1.1, to the htra_api folder.

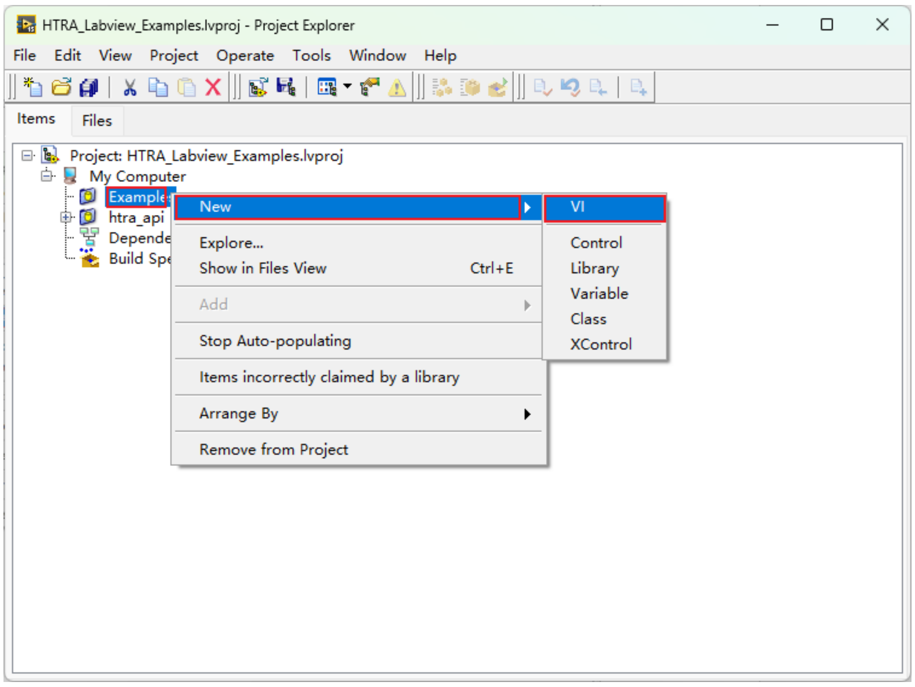

PC > OS (D:) > HTRA_Labview_Demo > htra_api

Name	Date modified	Type	Size
CalFile	5/9/2025 5:05 PM	File folder	
VIs	5/14/2025 2:19 PM	File folder	
htra_api.dll	3/24/2025 3:46 PM	Application extens...	1,013 KB
htra_api.h	3/24/2025 8:11 PM	C Header Source F...	160 KB
htra_api.lib	3/24/2025 3:46 PM	Object File Library	53 KB
htra_api.lvlib	11/17/2024 11:42 PM	LabVIEW Library	8 KB
htra_api_pnm.h	3/25/2025 9:42 AM	C Header Source F...	8 KB
libgcc_s_dw2-1.dll	1/12/2024 8:37 AM	Application extens...	123 KB
libiomp5md.dll	4/30/2024 3:30 AM	Application extens...	1,900 KB
libliquid.dll	1/12/2024 8:37 AM	Application extens...	1,743 KB
libliquid.lib	1/12/2024 8:37 AM	Object File Library	1,618 KB
libmkl.dll	4/30/2024 3:30 AM	Application extens...	42,043 KB
libmkl.lib	4/30/2024 3:30 AM	Object File Library	6 KB
libwinpthread-1.dll	1/12/2024 8:37 AM	Application extens...	67 KB
liquid.h	1/12/2024 8:37 AM	C Header Source F...	481 KB
mkl_dfti.h	4/30/2024 3:30 AM	C Header Source F...	11 KB
mkl_service.h	4/30/2024 3:30 AM	C Header Source F...	8 KB
mkl_types.h	4/30/2024 3:30 AM	C Header Source F...	5 KB
vcomp140.dll	5/10/2023 7:02 AM	Application extens...	159 KB
vcruntime140.dll	2/1/2002 6:02 PM	Application extens...	90 KB

6. In the LabVIEW project, add the newly created Examples folder and the htra_api folder to the project, and then save.

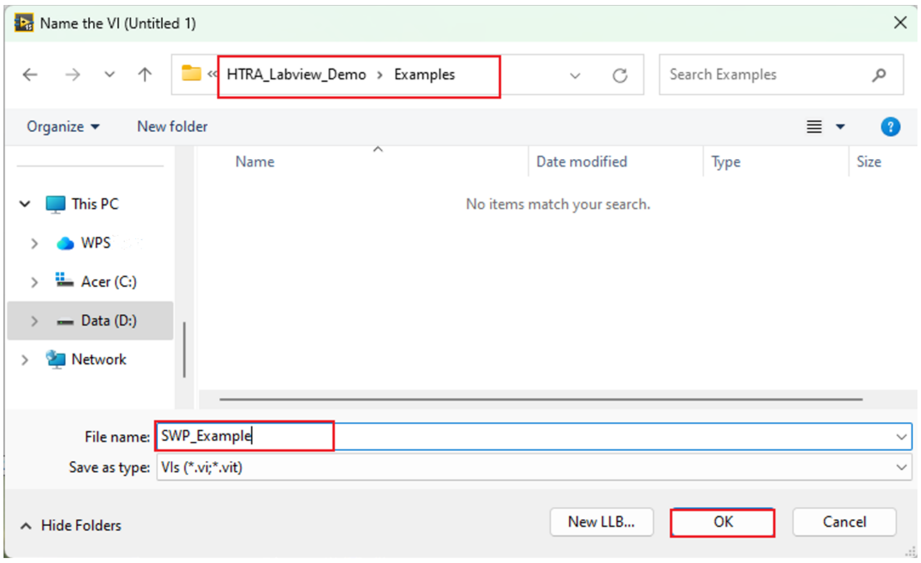


7. A new VI is created in the Examples folder, where the exported Labview API functions can be called within the block diagram page, and the calling process is consistent with that in the C environment.



HTRA_Labview_Examples.lvproj - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Project: HTRA_Labview_Examples.lvproj
My Computer
Example
htra_api
Depende
Build Spe
New
Explore...
Show in Files View Ctrl+E
Add
Stop Auto-populating
Items incorrectly claimed by a library
Arrange By
Remove from Project
VI
Control
Library
Variable
Class
XControl

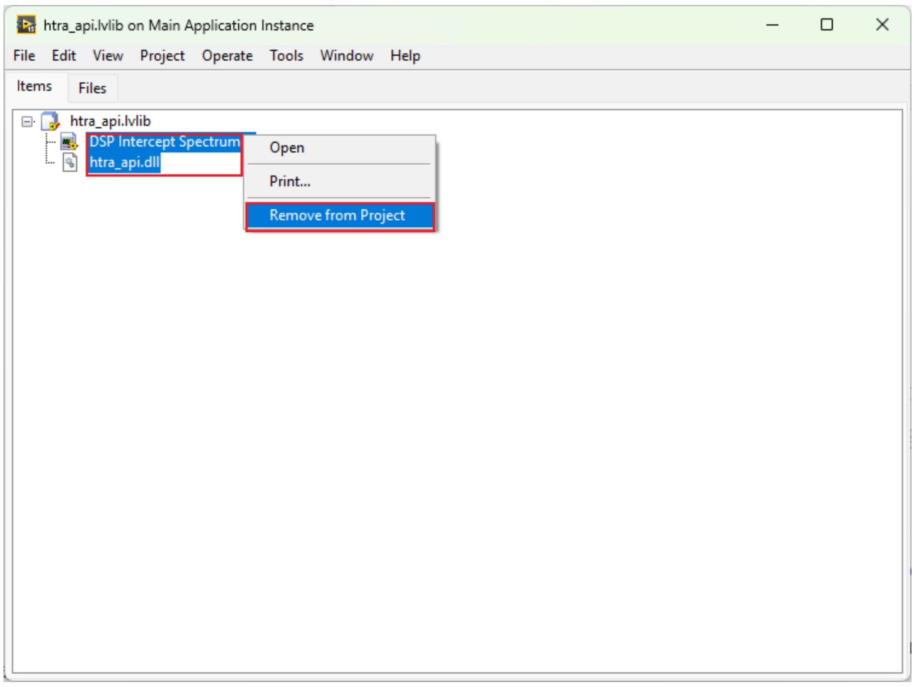
8. Finally, save this program to the Examples folder and rename it.



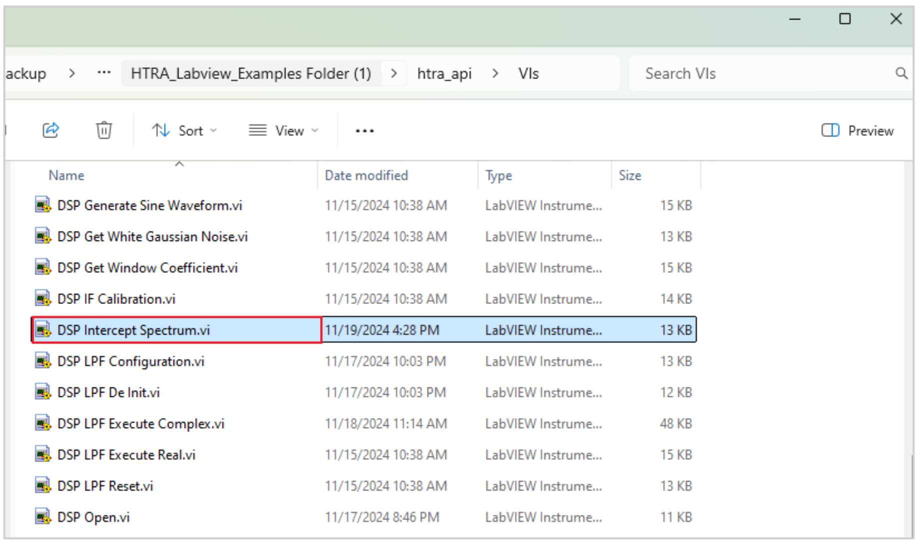
8.1.3 Use the newly exported library functions in an existing project.

Below, we will take the library function DSP_InterceptSpectrum() as an example to explain how to use the newly imported function in an existing project.

1. For the method of exporting library functions, please refer to section 8.1.1. Note that in the popped-up htra_api.lvlib, you should delete the exported functions and DLL files from the project, and then close htra_api.lvlib, as shown in the figure below.

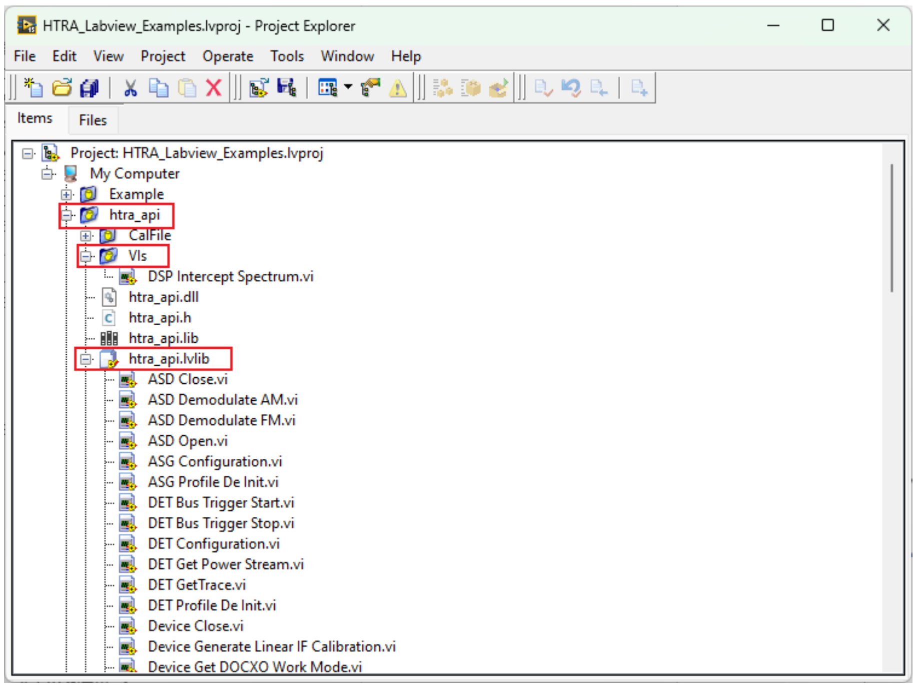

htra_api.lvlib on Main Application Instance
File Edit View Project Operate Tools Window Help
Items Files
htra_api.lvlib
DSP Intercept Spectrum
htra_api.dll
Open
Print...
Remove from Project

2. Copy the exported DSP_InterceptSpectrum function VI to the htra_api\V Is folder of the existing project.



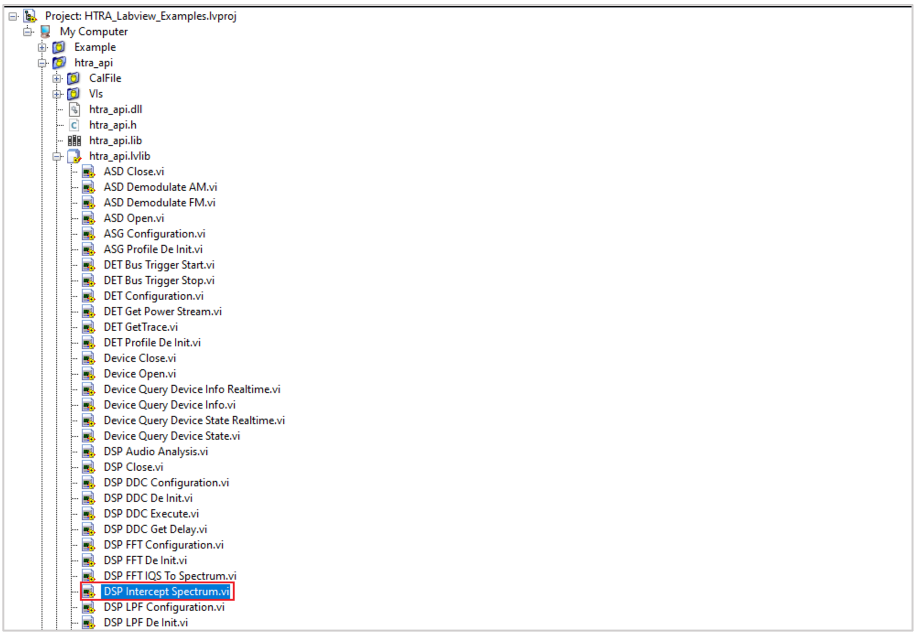
Name	Date modified	Type	Size
DSP Generate Sine Waveform.vi	11/15/2024 10:38 AM	LabVIEW Instrume...	15 KB
DSP Get White Gaussian Noise.vi	11/15/2024 10:38 AM	LabVIEW Instrume...	13 KB
DSP Get Window Coefficient.vi	11/15/2024 10:38 AM	LabVIEW Instrume...	15 KB
DSP IF Calibration.vi	11/15/2024 10:38 AM	LabVIEW Instrume...	14 KB
DSP Intercept Spectrum.vi	11/19/2024 4:28 PM	LabVIEW Instrume...	13 KB
DSP LPF Configuration.vi	11/17/2024 10:03 PM	LabVIEW Instrume...	13 KB
DSP LPF De Init.vi	11/17/2024 10:03 PM	LabVIEW Instrume...	12 KB
DSP LPF Execute Complex.vi	11/18/2024 11:14 AM	LabVIEW Instrume...	48 KB
DSP LPF Execute Real.vi	11/15/2024 10:38 AM	LabVIEW Instrume...	15 KB
DSP LPF Reset.vi	11/15/2024 10:38 AM	LabVIEW Instrume...	13 KB
DSP Open.vi	11/17/2024 8:46 PM	LabVIEW Instrume...	11 KB

3. Open the project with LabVIEW, then open the htra_api\V Is folder and htra_api.lvlib within the project.



```
Project: HTRA_Labview_Examples.lvproj
  My Computer
    Example
    htra_api
      CalFile
      VIs
        DSP Intercept Spectrum.vi
      htra_api.dll
      htra_api.h
      htra_api.lib
      htra_api.lvlib
        ASD Close.vi
        ASD Demodulate AM.vi
        ASD Demodulate FM.vi
        ASD Open.vi
        ASG Configuration.vi
        ASG Profile De Init.vi
        DET Bus Trigger Start.vi
        DET Bus Trigger Stop.vi
        DET Configuration.vi
        DET Get Power Stream.vi
        DET GetTrace.vi
        DET Profile De Init.vi
        Device Close.vi
        Device Generate Linear IF Calibration.vi
        Device Get DOCXO Work Mode.vi
```

4. Drag the DSP_InterceptSpectrum function VI from the V Is folder into htra_api.lvlib. At this point, you can use the newly added VI function normally.

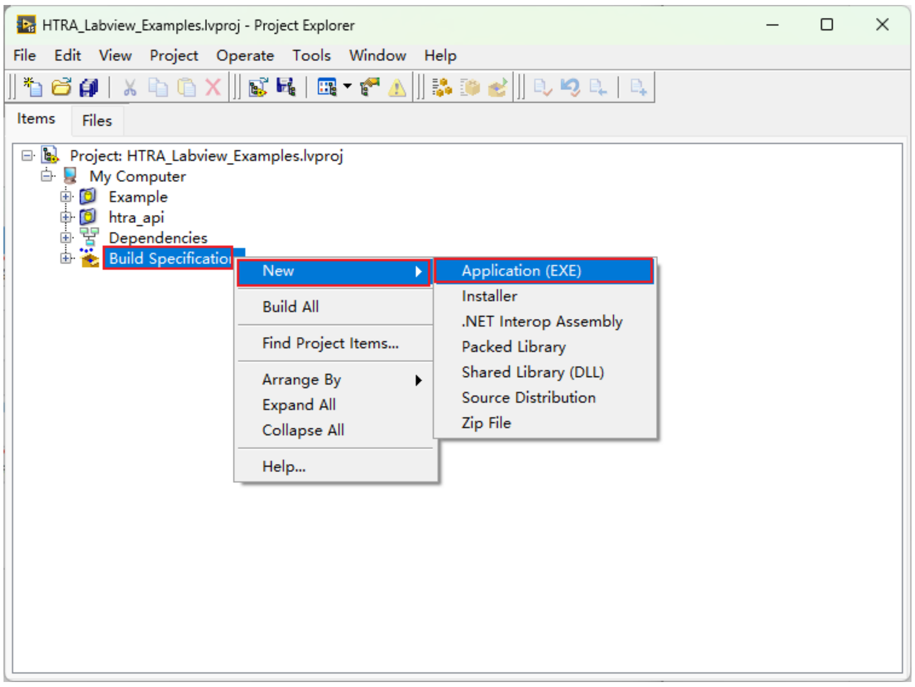


```
Project: HTRA_Labview_Examples.lvproj
  My Computer
    Example
    htra_api
      CalFile
      VIs
      htra_api.dll
      htra_api.h
      htra_api.lib
      htra_api.lvlib
        ASD Close.vi
        ASD Demodulate AM.vi
        ASD Demodulate FM.vi
        ASD Open.vi
        ASG Configuration.vi
        ASG Profile De Init.vi
        DET Bus Trigger Start.vi
        DET Bus Trigger Stop.vi
        DET Configuration.vi
        DET Get Power Stream.vi
        DET GetTrace.vi
        DET Profile De Init.vi
        Device Close.vi
        Device Open.vi
        Device Query Device Info Realtime.vi
        Device Query Device Info.vi
        Device Query Device State Realtime.vi
        Device Query Device State.vi
        DSP Audio Analysis.vi
        DSP Close.vi
        DSP DDC Configuration.vi
        DSP DDC De Init.vi
        DSP DDC Execute.vi
        DSP DDC Get Delay.vi
        DSP FFT Configuration.vi
        DSP FFT De Init.vi
        DSP FFT IQS To Spectrum.vi
        DSP Intercept Spectrum.vi
        DSP LPF Configuration.vi
        DSP LPF De Init.vi
```

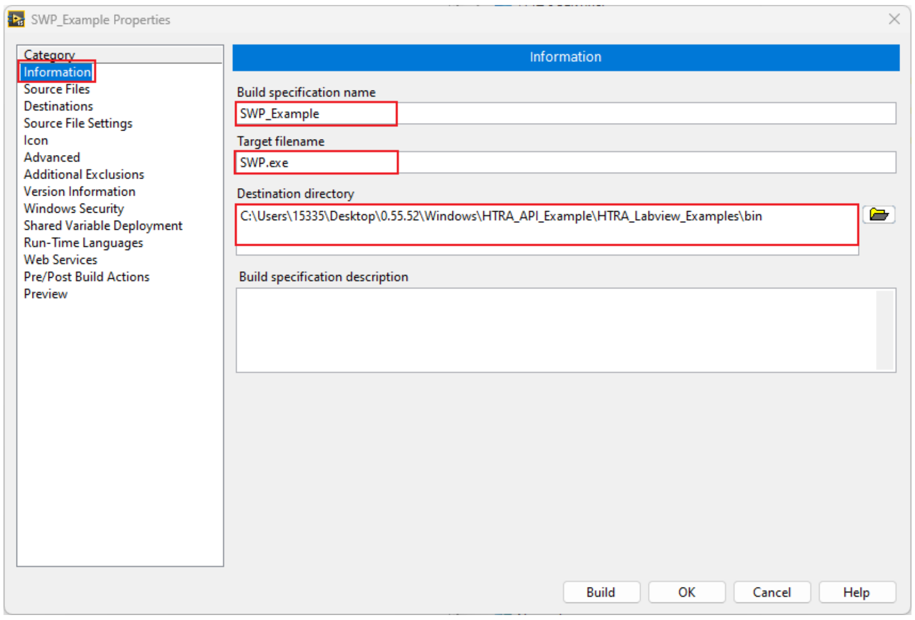
8.1.4 Generate an EXE from the vi in LabVIEW.

Below, we will take the LabVIEW project in the Windows\HTRA_API_Example folder on the USB drive as an example to explain how to generate an EXE from the program VI.

1. Open the LabVIEW project, then select "Build Specifications --> New --> Application (EXE)".



2. Fill in the standard name of the program file generated, the name of the exe, and the target directory path in "Information".



Information

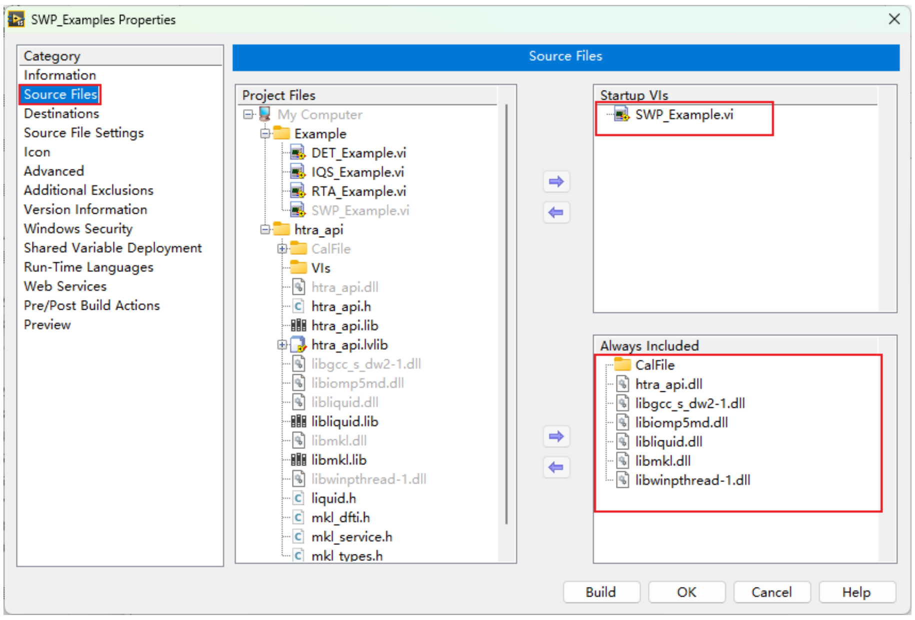
Build specification name: SWP_Example

Target filename: SWP.exe

Destination directory: C:\Users\15335\Desktop\0.55.52\Windows\HTRA_API_Example\HTRA_Labview_Examples\bin

Build specification description

3. Select the vi programs, CalFile folder, and all dependent library files to be exported as exe in "Source Files".



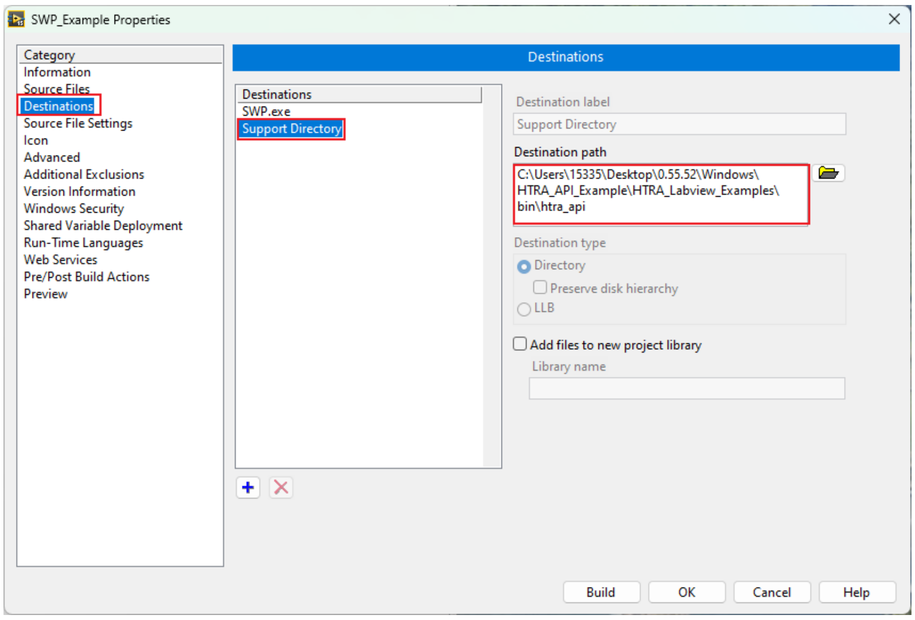
Source Files

Project Files
- My Computer
  - Example
    - DET_Example.vi
    - IQS_Example.vi
    - RTA_Example.vi
    - SWP_Example.vi
  - htra_api
    - CalFile
    - VIs
    - htra_api.dll
    - htra_api.h
    - htra_api.lib
    - htra_api.lvlib
    - libgcc_s_dw2-1.dll
    - libiomp5md.dll
    - libliquid.dll
    - libliquid.lib
    - libmkl.dll
    - libmkl.lib
    - libwinpthread-1.dll
    - liquid.h
    - mkl_dfti.h
    - mkl_service.h
    - mkl_types.h

Startup VIs
- SWP_Example.vi

Always Included
- CalFile
- htra_api.dll
- libgcc_s_dw2-1.dll
- libiomp5md.dll
- libliquid.dll
- libmkl.dll
- libwinpthread-1.dll

4. Add a layer of htra_api to the target path of "Support Directory" in "Destinations" to store the libraries and calibration folder.


SWP_Example Properties

Category: Information, Source Files, Destinations, Source File Settings, Icon, Advanced, Additional Exclusions, Version Information, Windows Security, Shared Variable Deployment, Run-Time Languages, Web Services, Pre/Post Build Actions, Preview

Destinations: SWP.exe, Support Directory

Destination label: Support Directory

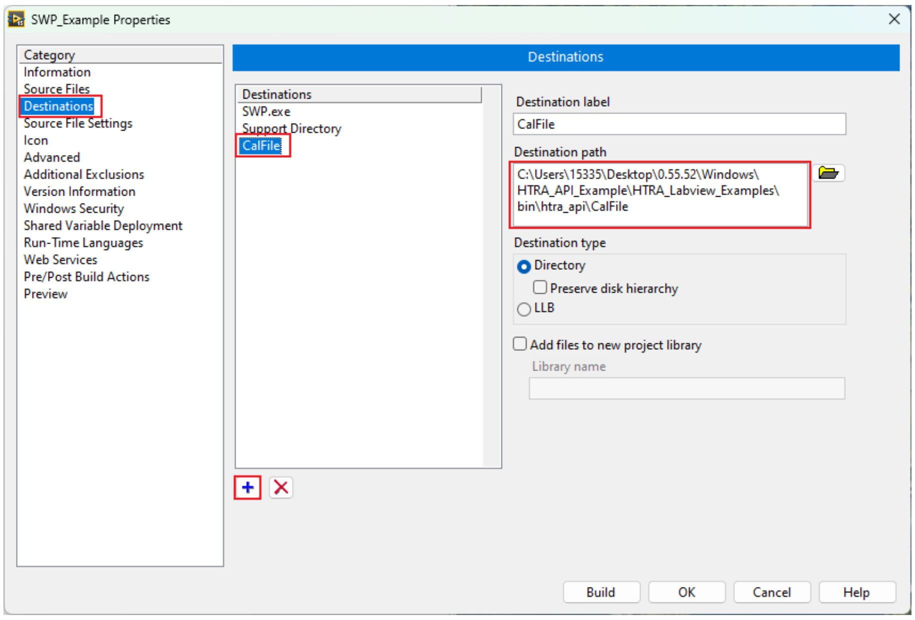
Destination path: C:\Users\15335\Desktop\0.55.52\Windows\HTRA_API_Example\HTRA_Labview_Examples\bin\htra_api

Destination type: [✓ Directory] Preserve disk hierarchy, LLB

Add files to new project library

Library name

5. Add a CalFile folder to store the device's calibration files, and add the target path of CalFile to the directory under "Support Directory".


SWP_Example Properties

Category: Information, Source Files, Destinations, Source File Settings, Icon, Advanced, Additional Exclusions, Version Information, Windows Security, Shared Variable Deployment, Run-Time Languages, Web Services, Pre/Post Build Actions, Preview

Destinations: SWP.exe, Support Directory, CalFile

Destination label: CalFile

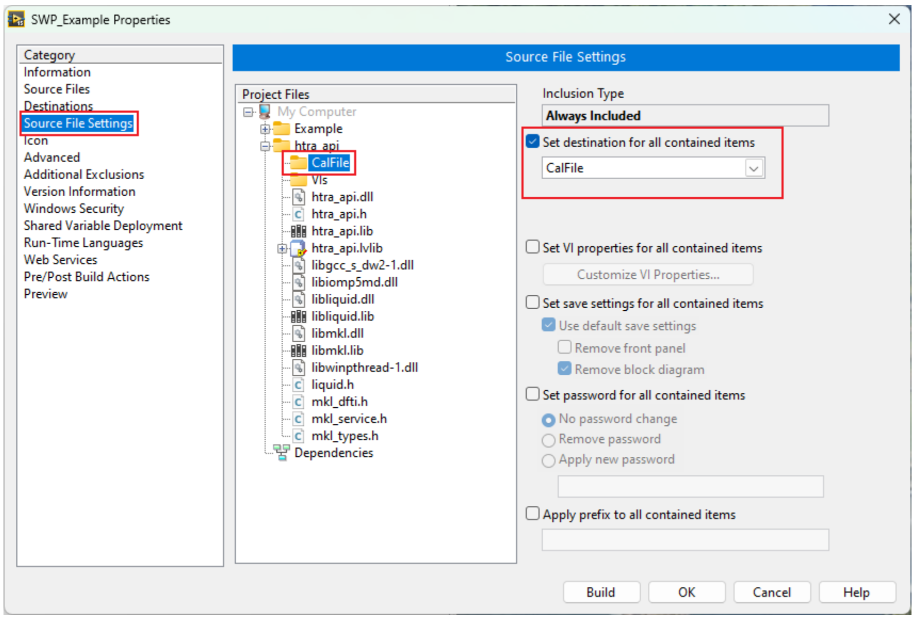
Destination path: C:\Users\15335\Desktop\0.55.52\Windows\HTRA_API_Example\HTRA_Labview_Examples\bin\htra_api\CalFile

Destination type: [✓ Directory] Preserve disk hierarchy, LLB

Add files to new project library

Library name

6. In "Source Files Settings", select the target path of the calibration files to "CalFile".



Inclusion Type: Always Included

[✓] Set destination for all contained items: CalFile

[ ] Set VI properties for all contained items

Customize VI Properties...

[ ] Set save settings for all contained items

[✓] Use default save settings

[ ] Remove front panel

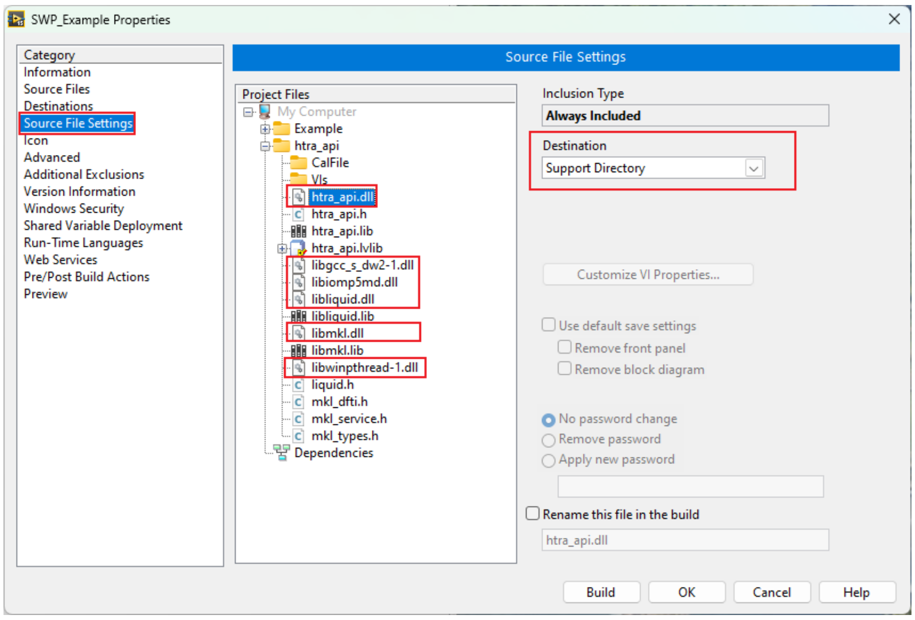
[✓] Remove block diagram

[ ] Set password for all contained items

[✓ No password change] Remove password Apply new password

[ ] Apply prefix to all contained items

7. Select the target paths of all dependent libraries to the "Support Directory".



Inclusion Type: Always Included

Destination: Support Directory

Customize VI Properties...

[ ] Use default save settings

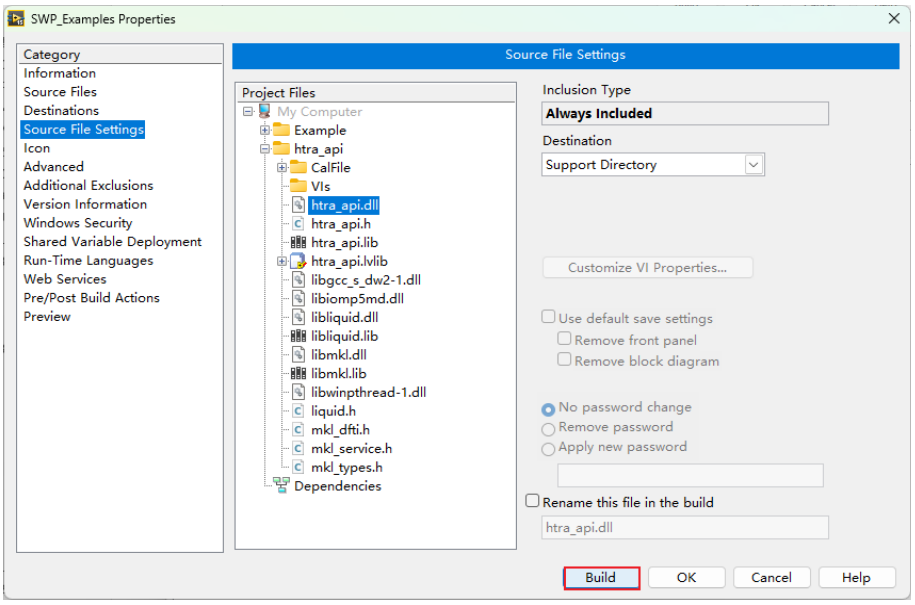
[ ] Remove front panel

[ ] Remove block diagram

[✓ No password change] Remove password Apply new password

[ ] Rename this file in the build: htra_api.dll

8. Click "Build".


SWP_Examples Properties

Category: Information, Source Files, Destinations, Source File Settings, Icon, Advanced, Additional Exclusions, Version Information, Windows Security, Shared Variable Deployment, Run-Time Languages, Web Services, Pre/Post Build Actions, Preview

Source File Settings

Project Files: My Computer, Example, htra_api, CalFile, VIs, htra_api.dll, htra_api.h, htra_api.lib, htra_api.lvlib, libgcc_s_dw2-1.dll, libiomp5md.dll, libliquid.dll, libliquid.lib, libmkl.dll, libmkl.lib, libwinpthread-1.dll, liquid.h, mkl_dfti.h, mkl_service.h, mkl_types.h, Dependencies

Inclusion Type: Always Included

Destination: Support Directory

Customize VI Properties...

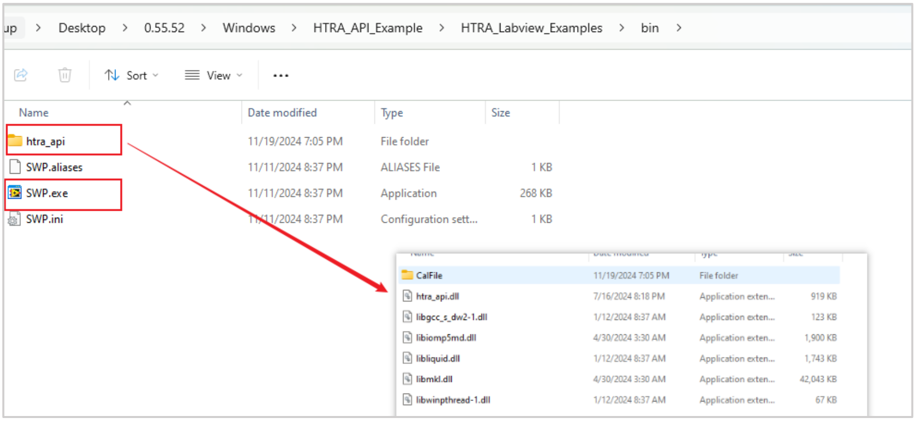
Use default save settings
Remove front panel
Remove block diagram

No password change
Remove password
Apply new password

Rename this file in the build: htra_api.dll

Build OK Cancel Help

9. At this point, the generation of the exe program from the vi program in Labview is complete.


up > Desktop > 0.55.52 > Windows > HTRA_API_Example > HTRA_Labview_Examples > bin >

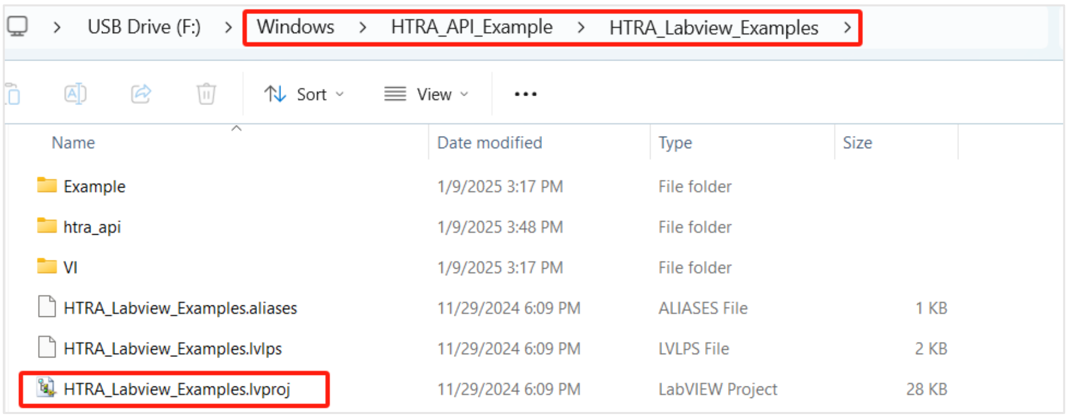
Name	Date modified	Type	Size
htra_api	11/19/2024 7:05 PM	File folder	
SWP.aliases	11/11/2024 8:37 PM	ALIASES File	1 KB
SWP.exe	11/11/2024 8:37 PM	Application	268 KB
SWP.ini	11/11/2024 8:37 PM	Configuration sett...	1 KB

Name	Date modified	Type	Size
CalFile	11/19/2024 7:05 PM	File folder	
htra_api.dll	7/16/2024 8:18 PM	Application exten...	919 KB
libgcc_s_dw2-1.dll	1/12/2024 8:37 AM	Application exten...	123 KB
libiomp5md.dll	4/30/2024 3:30 AM	Application exten...	1,900 KB
libliquid.dll	1/12/2024 8:37 AM	Application exten...	1,743 KB
libmkl.dll	4/30/2024 3:30 AM	Application exten...	42,043 KB
libwinpthread-1.dll	1/12/2024 8:37 AM	Application exten...	67 KB

8.2 The usage process of Labview examples

The usage process of Labview examples included in the USB drive is as follows:

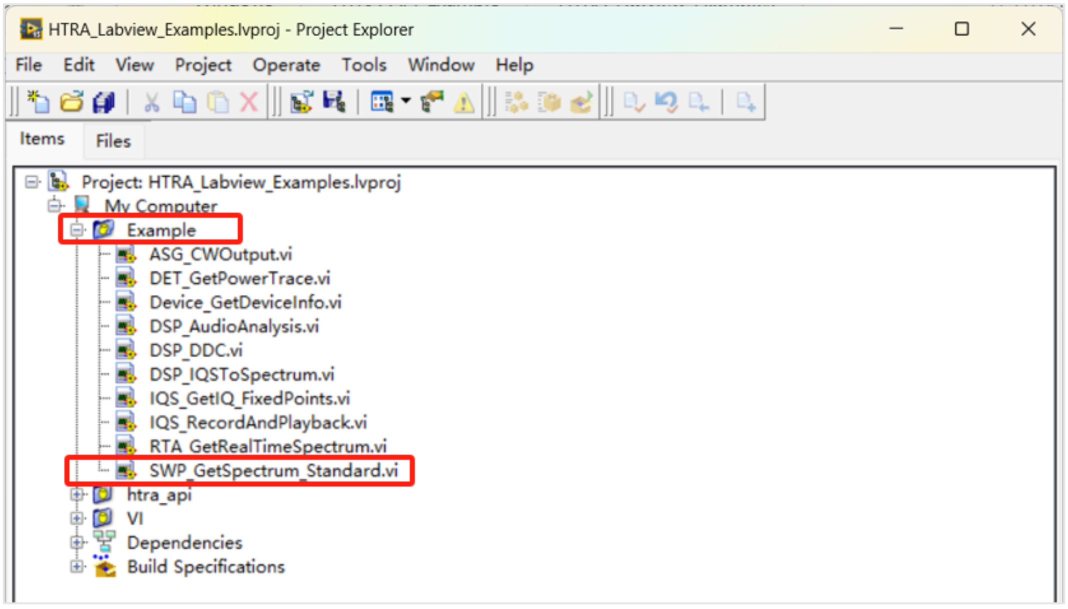
1. Open the accompanying USB drive using LabVIEW in the folder Windows\HTRA_API_Example\HTRA_Labview_Examples, and open the project HTRA_Labview_Examples.lvproj.



USB Drive (F:) > Windows > HTRA_API_Example > HTRA_Labview_Examples >

Name	Date modified	Type	Size
Example	1/9/2025 3:17 PM	File folder	
htra_api	1/9/2025 3:48 PM	File folder	
VI	1/9/2025 3:17 PM	File folder	
HTRA_Labview_Examples.aliases	11/29/2024 6:09 PM	ALIASES File	1 KB
HTRA_Labview_Examples.lvlps	11/29/2024 6:09 PM	LVLPS File	2 KB
HTRA_Labview_Examples.lvproj	11/29/2024 6:09 PM	LabVIEW Project	28 KB

2. After opening the project, double-click to open any routine vi in the Example folder. For example, here we open SWP_GetSpectrum_Standard.vi to use the SWP mode routine.



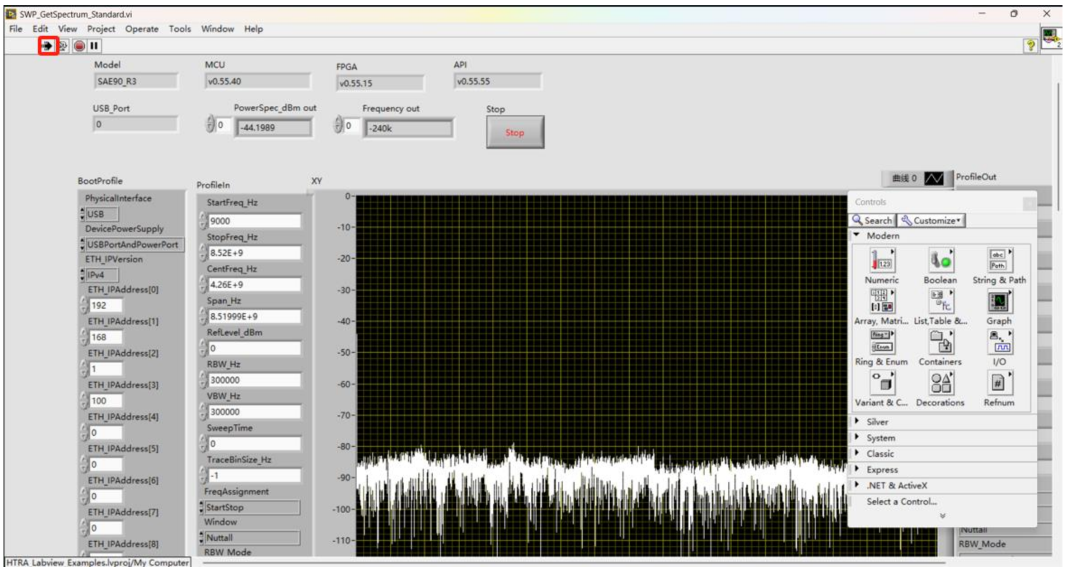
HTRA_Labview_Examples.lvproj - Project Explorer

File Edit View Project Operate Tools Window Help

Items Files

```
Project: HTRA_Labview_Examples.lvproj
  My Computer
    Example
      ASG_CWOutput.vi
      DET_GetPowerTrace.vi
      Device_GetDeviceInfo.vi
      DSP_AudioAnalysis.vi
      DSP_DDC.vi
      DSP_IQSToSpectrum.vi
      IQS_GetIQ_FixedPoints.vi
      IQS_RecordAndPlayback.vi
      RTA_GetRealTimeSpectrum.vi
      SWP_GetSpectrum_Standard.vi
    htra_api
    VI
    Dependencies
    Build Specifications
```

3. Once the program is opened, simply click the run button in the upper left corner, as shown in the image where the program is running normally.



Model: SAE90_R3
MCU: v0.55.40
FPGA: v0.55.15
API: v0.55.55
USB_Port: 0
PowerSpec_dBm out: 0 -44.1989
Frequency out: 0 -240k

BootProfile
PhysicalInterface: USB
DevicePowerSupply: USBPortAndPowerPort
ETH_IPVersion: IPv4
ETH_IPAddress[0]: 192
ETH_IPAddress[1]: 168
ETH_IPAddress[2]: 1
ETH_IPAddress[3]: 100
ETH_IPAddress[4]: 0
ETH_IPAddress[5]: 0
ETH_IPAddress[6]: 0
ETH_IPAddress[7]: 0
ETH_IPAddress[8]

ProfileIn
StartFreq_Hz: 9000
StopFreq_Hz: 8.52E+9
CentFreq_Hz: 4.26E+9
Span_Hz: 8.51999E+9
RefLevel_dBm: 0
RBW_Hz: 300000
VBW_Hz: 300000
SweepTime: 0
TraceBinSize_Hz: -1
FreqAssignment: StartStop
Window: Nuttall
RBW_Mode

8.3 Labview Example Description

8.3.1 Get device information

Device_GetDeviceInfo.vi: An example for obtaining various device information, including: API version, device model, device UID, MCU version, FPGA version, and device temperature.

8.3.2 Standard spectrum acquisition

SWP_GetSpectrum_Standard.vi: Obtains standard spectrum data within a specified frequency band and displays the image.

8.3.3 Obtain IQ Data for a Fixed Number of Points or Duration

IQS_GetIQ_FixedPoints.vi: Obtains IQ data with a fixed number of points. When the device receives a Bus trigger signal, it returns IQ data with a fixed number of points.

8.3.4 Streaming and reading IQ data

IQS_RecordAndPlayback.vi: Obtains IQ data, records the IQ data as a txt file, and plays back the recorded IQ data.

8.3.5 IQ to Spectrum Data

DSP_IQSToSpectrum.vi: Obtains IQ data and converts the acquired IQ data into spectrum data.

8.3.6 Digital Downconversion

DSP_DDC.vi: Performs digital downconversion on the IQ data stream and resamples to generate a sub-IQ stream for further spectrum analysis.

8.3.7 Audio Analysis

DSP_AudioAnalysis.vi: Analyzes audio voltage (V), audio frequency (Hz), signal-to-noise ratio (dB), and total harmonic distortion (%).

8.3.8 Obtain Power Detection Data for a Fixed Number of Points or Duration

DET_GetPowerTrace.vi: Obtain a fixed number of DET data points. When the device receives a Bus trigger signal, it returns a fixed number of DET data points.

8.3.9 Obtain real-time spectrum data for a fixed number of points or duration

RTA_GetRealTimeSpectrum.vi: Obtain a fixed number of RTA data points. When the device receives a Bus trigger signal, it returns a fixed number of RTA data points.

8.3.10 ASG Signal Source Output Signal

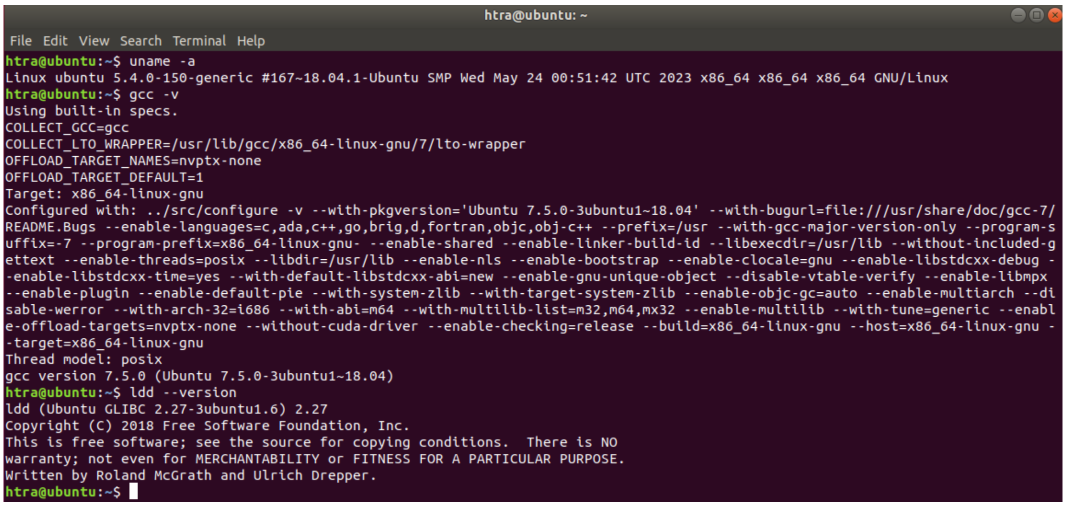
ASG_CWOutput.vi: Control the internal signal generator of the device to output single-tone signals, sweep signals, and power scan signals.

9. Linux

9.1 Environment Version Compatibility Self-Check

When using the device in a Linux system, you first need to confirm whether the current Linux environment's system architecture, gcc version, and GLIBC version are supported according to the following process:

1. Open the terminal and enter "uname -a" to check the Linux system architecture, for example, here the Linux system architecture is x86_64.
2. In the terminal, enter "gcc -v" to check the system gcc version, for example, here the gcc version is 7.5.0.
3. In the terminal, enter "ldd --version" to check the system GLIBC version, for example, here the GLIBC version is 2.27.



```
htra@ubuntu:~$ uname -a
Linux ubuntu 5.4.0-150-generic #167~18.04.1-Ubuntu SMP Wed May 24 00:51:42 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
htra@ubuntu:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1~18.04' --with-bugurl=file:///usr/share/doc/gcc-7/
README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++ --prefix=/usr --with-gcc-major-version-only --program-s
uffix=-7 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-g
ettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-bootstrap --enable-clocale=gnu --enable-libstdcxx-debug -
-enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-libmpx
--enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib --enable-objc-gc=auto --enable-multiarch --di
sable-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enabl
e-offload-targets=nvptx-none --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu -
-target=x86_64-linux-gnu
Thread model: posix
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
htra@ubuntu:~$ ldd --version
ldd (Ubuntu GLIBC 2.27-3ubuntu1.6) 2.27
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
htra@ubuntu:~$ 
```

4. Confirm whether the current environment is supported according to the terminal information comparison table. If it is not yet supported, please contact technical support personnel.

X86 processor	Support Intel and AMD processors
ARM processor	aarch64 (armv8), armv7 processors, such as: Raspberry Pi 4b, RK3399, RK3568, RK3588, T507, NVIDIA Jetson TX2
Compilation	gcc4.8, glib2.17 and above

environment	
Distribution	Customized system for Raspberry Pi 4b, Ubuntu 18.04, etc.

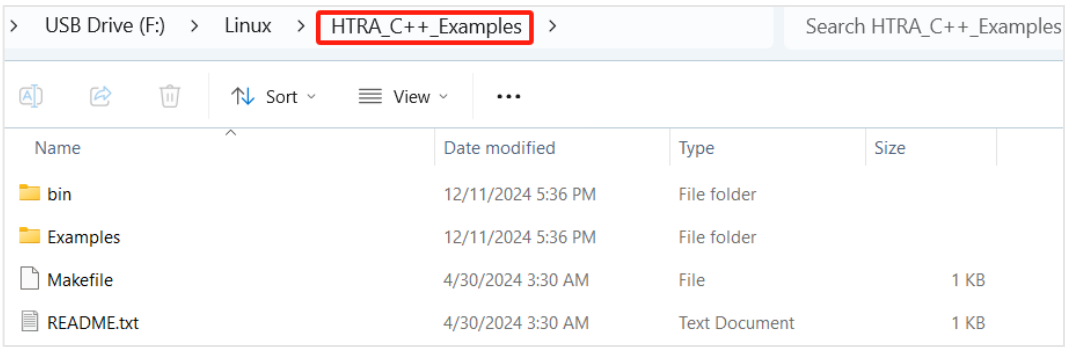
9.2 Accompanying documentation

Currently, the Linux section of the accompanying USB drive contains the following materials:

9.2.1 HTRA_C++_Examples

The HTRA_C++_Examples folder contains:

1. Examples folder: C++ example programs (see Chapter9.4 for usage).
2. Makefile: A build script used to compile the example programs into executable files.
3. bin folder: Used to store device calibration files and executable files generated from the example programs.

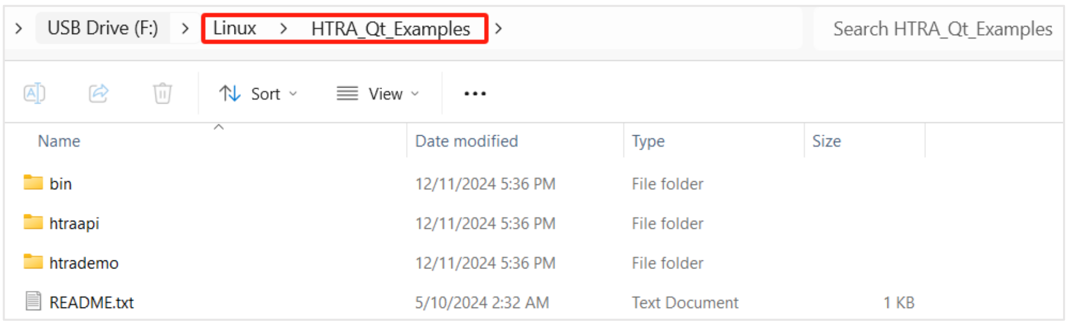


Name	Date modified	Type	Size
bin	12/11/2024 5:36 PM	File folder	
Examples	12/11/2024 5:36 PM	File folder	
Makefile	4/30/2024 3:30 AM	File	1 KB
README.txt	4/30/2024 3:30 AM	Text Document	1 KB

9.2.2 HTRA_Qt_Examples

The HTRA_Qt_Examples folder contains:

1. htrademo folder: Qt examples and pro files (see Chapter9.5 for usage).
2. bin folder: Used to store device calibration files and executable files compiled from the example programs.
3. htraapi folder: Used to store dynamic link libraries.

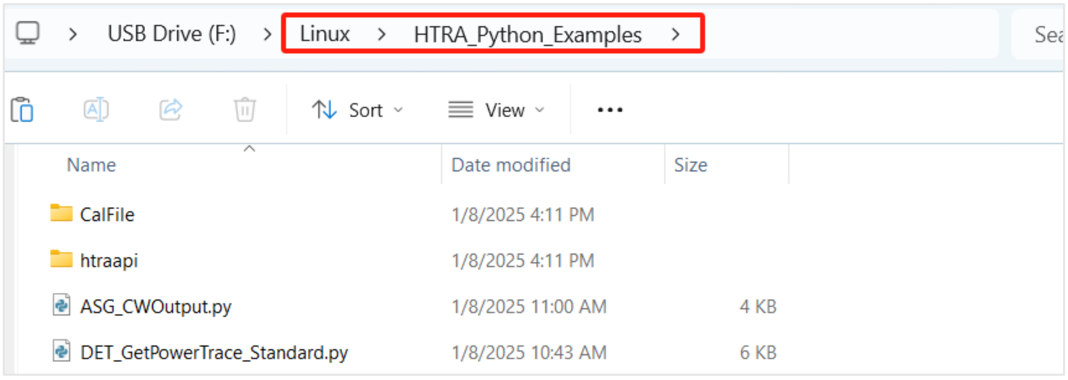


Name	Date modified	Type	Size
bin	12/11/2024 5:36 PM	File folder	
htraapi	12/11/2024 5:36 PM	File folder	
htrademo	12/11/2024 5:36 PM	File folder	
README.txt	5/10/2024 2:32 AM	Text Document	1 KB

9.2.3 HTRA_Python_Examples

HTRA_Python_Examples folder specifically contains:

1. Python example programs (see Chapter9.6 for usage).
2. CalFile folder: stores device calibration files.
3. Htraapi folder: used to store dynamic link libraries.



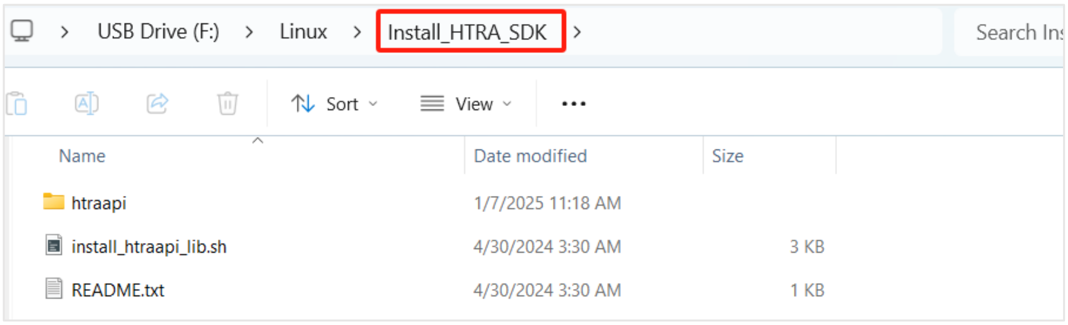
Name	Date modified	Size
CalFile	1/8/2025 4:11 PM	
htraapi	1/8/2025 4:11 PM	
ASG_CWOutput.py	1/8/2025 11:00 AM	4 KB
DET_GetPowerTrace_Standard.py	1/8/2025 10:43 AM	6 KB

9.2.4 Install_HTRA_SDK

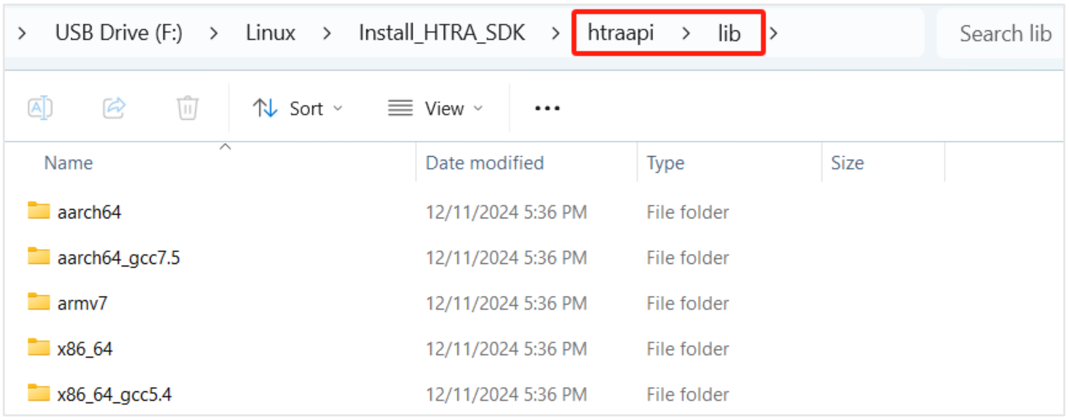
Install_HTRA_SDK folder contains:

1. install_htraapi_lib.sh: driver configuration script.
2. Install_HTRA_SDK\htraapi\configs folder: driver configuration files.
3. Install_HTRA_SDK\htraapi\inc folder: header files.
4. Install_HTRA_SDK\htraapi\lib\arrch64 folder: arrch64 architecture dynamic link libraries.
5. Install_HTRA_SDK\htraapi\lib\arrch64_gcc7.5 folder: Dynamic link library for arrch64 architecture with more efficient FFT (requires system gcc version higher than 7.5).

6. Install_HTRA_SDK\htraapi\lib\x86_64 folder: Dynamic link library for x86_64 architecture.
7. Install_HTRA_SDK\htraapi\lib\x86_64_gcc5.4 folder: Dynamic link library for x86_64 architecture with more efficient FFT (requires system gcc version higher than 5.4).
8. Install_HTRA_SDK\htraapi\lib\armv7 folder: Dynamic link library for armv7 architecture.



Name	Date modified	Size
htraapi	1/7/2025 11:18 AM	
install_htraapi_lib.sh	4/30/2024 3:30 AM	3 KB
README.txt	4/30/2024 3:30 AM	1 KB

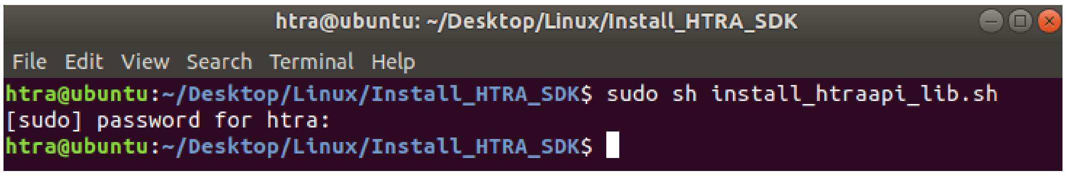


Name	Date modified	Type	Size
aarch64	12/11/2024 5:36 PM	File folder	
aarch64_gcc7.5	12/11/2024 5:36 PM	File folder	
armv7	12/11/2024 5:36 PM	File folder	
x86_64	12/11/2024 5:36 PM	File folder	
x86_64_gcc5.4	12/11/2024 5:36 PM	File folder	

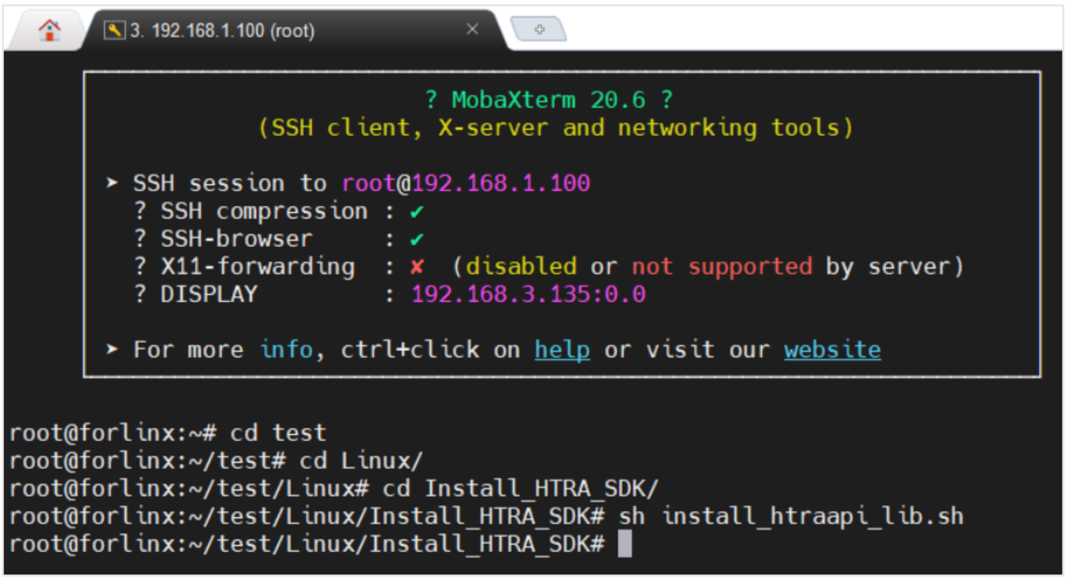
9.3 Driver file configuration

To use the device in Linux, the driver file must be configured first. The specific process is as follows:

1. Driver file configuration: First, drag the Install_HTRA_SDK folder into the Linux host computer, then open a terminal in the Install_HTRA_SDK folder and enter "sudo sh install_htraapi_lib.sh" to configure the driver file. If the special development board does not have the sudo command, simply enter "sh install_htraapi_lib.sh".

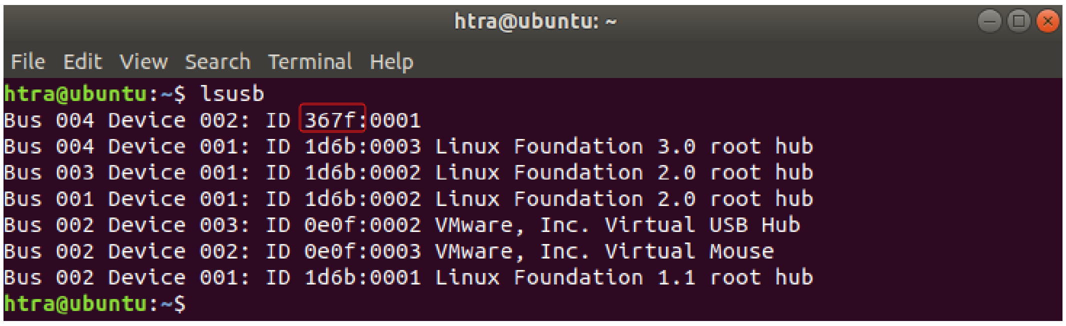

htra@ubuntu: ~/Desktop/Linux/Install_HTRA_SDK
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/Linux/Install_HTRA_SDK$ sudo sh install_htraapi_lib.sh
[sudo] password for htra:
htra@ubuntu:~/Desktop/Linux/Install_HTRA_SDK$ 
```



```
                    ? MobaXterm 20.6 ?
        (SSH client, X-server and networking tools)

 ➤ SSH session to root@192.168.1.100
   ? SSH compression : ✔
   ? SSH-browser     : ✔
   ? X11-forwarding  : ✘  (disabled or not supported by server)
   ? DISPLAY         : 192.168.3.135:0.0

 ➤ For more info, ctrl+click on help or visit our website

root@forlinx:~# cd test
root@forlinx:~/test# cd Linux/
root@forlinx:~/test/Linux# cd Install_HTRA_SDK/
root@forlinx:~/test/Linux/Install_HTRA_SDK# sh install_htraapi_lib.sh
root@forlinx:~/test/Linux/Install_HTRA_SDK# 
```

2. Configuration check: Ensure the device is correctly connected to the host computer (if the host computer is a virtual machine, ensure the device is connected to the virtual machine and that USB compatibility is 3.1) and provide normal power to the device. At this point, as shown in the figure, enter "lsusb" in the terminal to view the list of USB devices on the machine, where "ID: 6430 (or ID: 3675 or ID: 04b5)" indicates successful device connection.



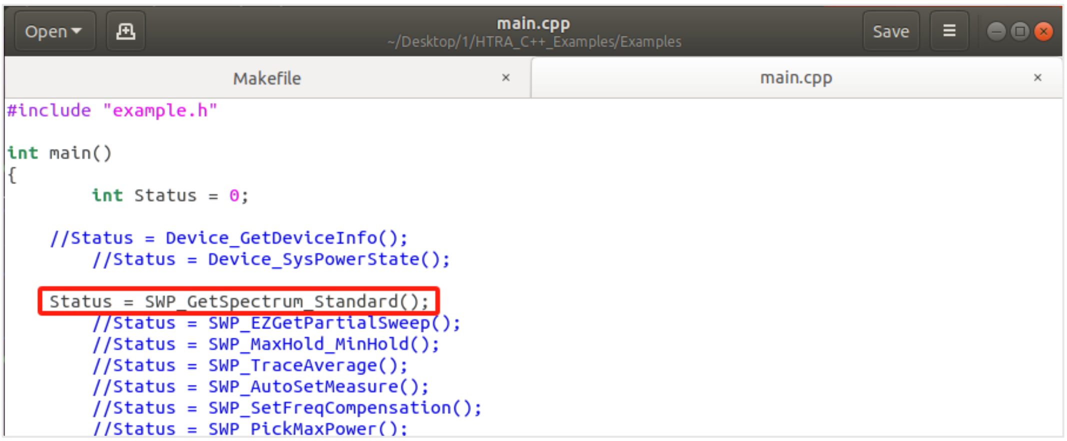
```
htra@ubuntu:~$ lsusb
Bus 004 Device 002: ID 367f:0001
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
htra@ubuntu:~$
```

9.4 C++ example usage and project creation

9.4.1 C++ example usage

Under the premise that the device is properly connected and the driver files have been correctly configured as per Chapter9.3, if you wish to use the C++ examples included on the USB drive, you can refer to the following process (the specific functions of the examples can be directly viewed in the description in Chapter 1):

1. Select the program to compile: First, copy the Linux\ HTRA_C++_Examples folder from the USB drive to the host computer. Double-click to open main.cpp in the Examples folder, and uncomment the example you need to test (the following steps will take the SWP_GetSpectrum_Standard example as an example), as shown in the figure to uncomment.



```
#include "example.h"

int main()
{
	int Status = 0;

    //Status = Device_GetDeviceInfo();
	//Status = Device_SysPowerState();

    Status = SWP_GetSpectrum_Standard();
	//Status = SWP_EZGetPartialSweep();
	//Status = SWP_MaxHold_MinHold();
	//Status = SWP_TraceAverage();
	//Status = SWP_AutoSetMeasure();
	//Status = SWP_SetFreqCompensation();
	//Status = SWP_PickMaxPower();
```

2. Refer to the system architecture in Chapter9.1, and based on the selected test example, open the terminal in the HTRA_C++_Examples folder, following the host computer's system architecture.

1) For x86_86 system, input:

(the example routine here is SWP_GetSpectrum_Standard)

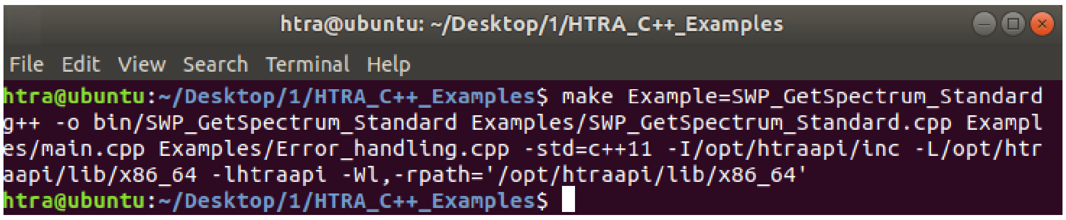
make Example=SWP_GetSpectrum_Standard

2) For aarch64 system, input:

make TARG=aarch64 Example=SWP_GetSpectrum_Standard

3) For armv7 system, input:

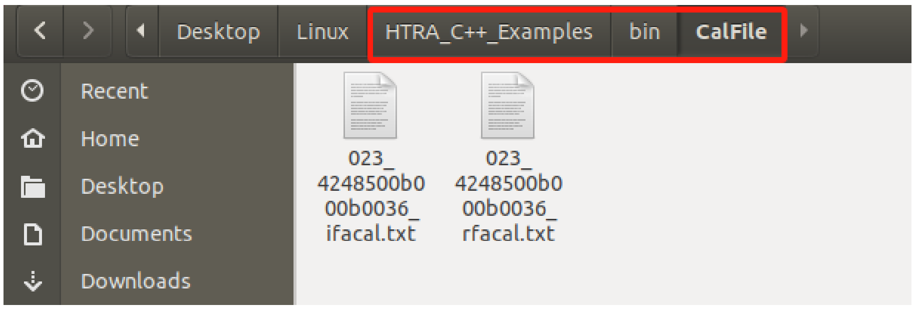
make TARG=armv7 Example=SWP_GetSpectrum_Standard



htra@ubuntu: ~/Desktop/1/HTRA_C++_Examples
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/1/HTRA_C++_Examples$ make Example=SWP_GetSpectrum_Standard
g++ -o bin/SWP_GetSpectrum_Standard Examples/SWP_GetSpectrum_Standard.cpp Exampl
es/main.cpp Examples/Error_handling.cpp -std=c++11 -I/opt/htraapi/inc -L/opt/htr
aapi/lib/x86_64 -lhtraapi -Wl,-rpath='/opt/htraapi/lib/x86_64'
htra@ubuntu:~/Desktop/1/HTRA_C++_Examples$ 
```

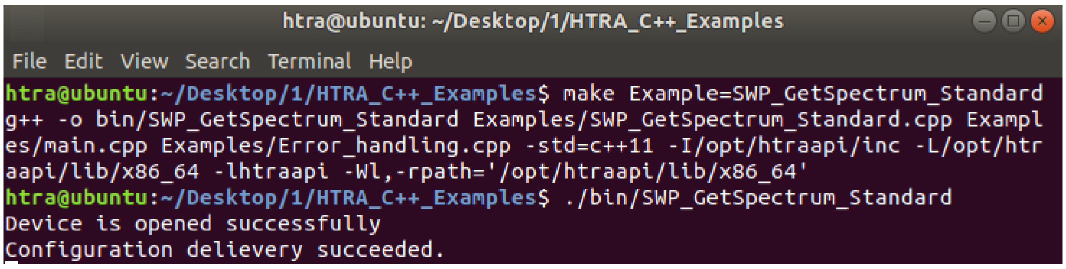
3. Verify the calibration file: Open the HTRA_C++_Examples\bin\CalFile folder, and ensure that the folder contains the device calibration files, as shown in the figure.



4. Run the program: After the program compiles successfully and the calibration is confirmed to be correct, open the terminal and input:

./bin/SWP_GetSpectrum_Standard

The example shown in the figure is a normal operation example.



htra@ubuntu: ~/Desktop/1/HTRA_C++_Examples
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/1/HTRA_C++_Examples$ make Example=SWP_GetSpectrum_Standard
g++ -o bin/SWP_GetSpectrum_Standard Examples/SWP_GetSpectrum_Standard.cpp Exampl
es/main.cpp Examples/Error_handling.cpp -std=c++11 -I/opt/htraapi/inc -L/opt/htr
aapi/lib/x86_64 -lhtraapi -Wl,-rpath='/opt/htraapi/lib/x86_64'
htra@ubuntu:~/Desktop/1/HTRA_C++_Examples$ ./bin/SWP_GetSpectrum_Standard
Device is opened successfully
Configuration delievery succeeded.
```

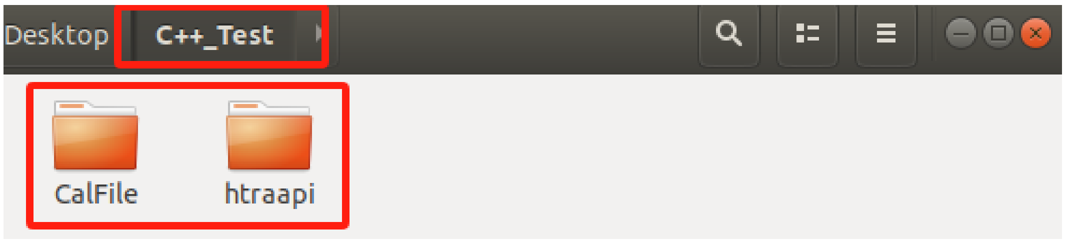
9.4.2 C++ Project Creation and Compilation

Assuming that the driver files have been correctly configured as per Chapter9.3, if you want to create a C++ project for compilation, please refer to the following process:

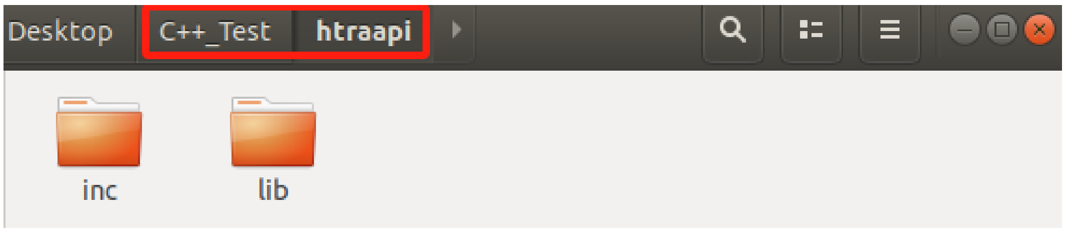
Write Code: Since the Linux dynamic link library provided with the USB drive is identical to that in Windows, the code only needs to comply with the API programming guidelines.

Compile and Run:

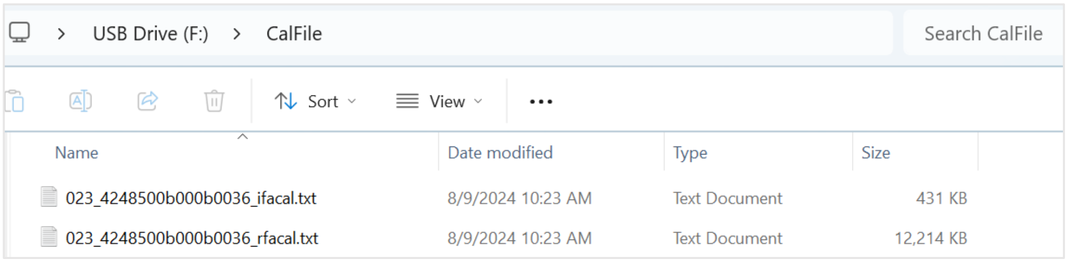
1. As shown in the figure, first create a new folder to store the entire project (taking C++_Test as an example), then create a CalFile folder within the folder to store calibration files, and create an htraapi folder to store header files and dynamic link libraries.



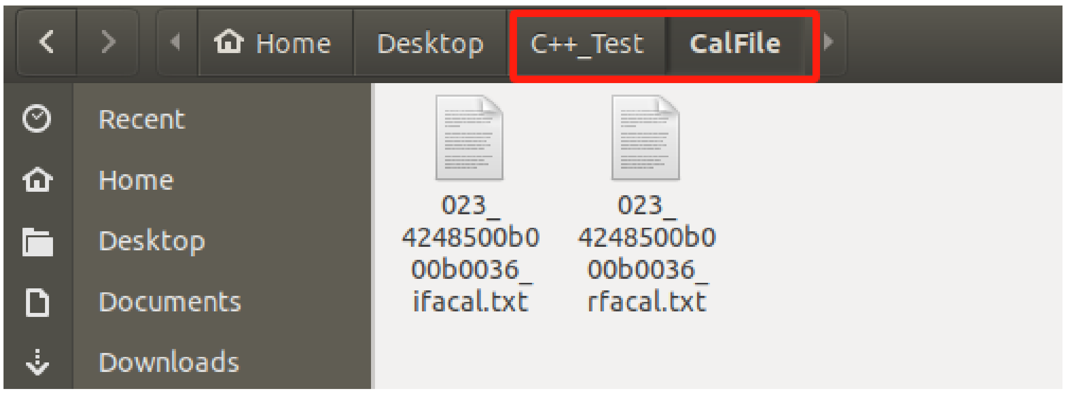
2. Create an inc folder under the htraapi folder to store header files, and a lib folder to store dynamic link libraries.



3. Copy the files from the CalFile folder on the provided USB drive to the newly created C++_Test\CalFile folder.

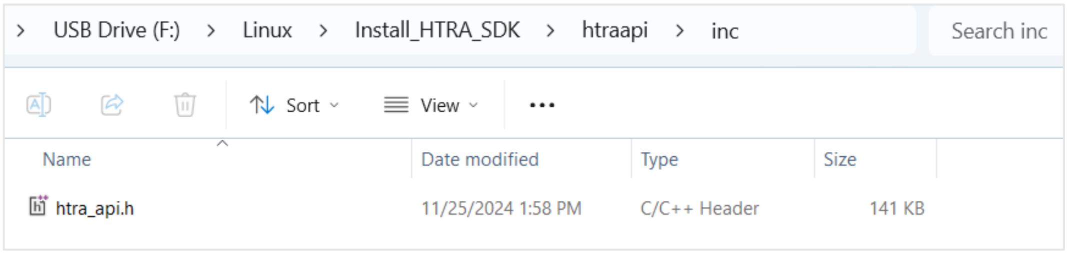


Name	Date modified	Type	Size
023_4248500b000b0036_ifacal.txt	8/9/2024 10:23 AM	Text Document	431 KB
023_4248500b000b0036_rfacal.txt	8/9/2024 10:23 AM	Text Document	12,214 KB

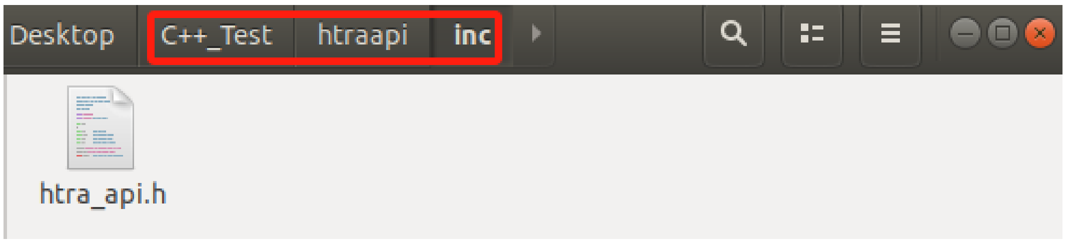


023_4248500b000b0036_ifacal.txt
023_4248500b000b0036_rfacal.txt

4. Copy the header files from the Linux\Install_HTRA_SDK\htraapi\inc folder on the provided USB drive to the newly created C++_Test\htraapi\inc folder.

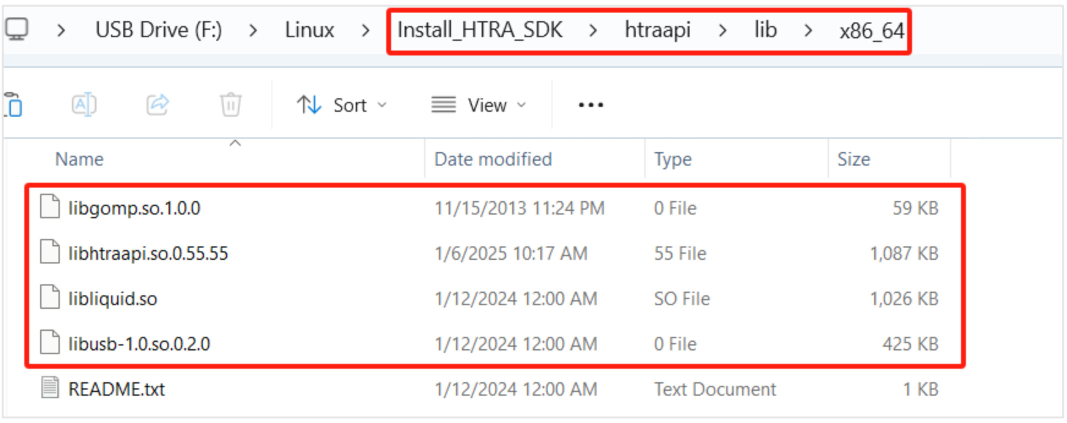


Name	Date modified	Type	Size
htra_api.h	11/25/2024 1:58 PM	C/C++ Header	141 KB

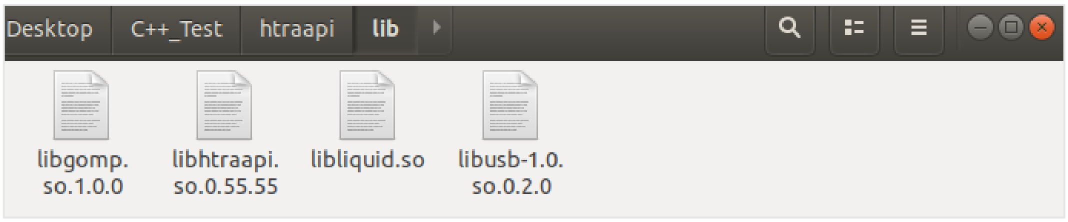


htra_api.h

5. Refer to the system architecture in Chapter9.1, and then according to the instructions in Chapter9.2.4, copy the dynamic link libraries corresponding to the architecture from the Linux\Install_HTRA_SDK\htraapi\lib folder to the newly created C++_Test\htraapi\lib folder (taking the x86_64 architecture host computer as an example).


USB Drive (F:) > Linux > Install_HTRA_SDK > htraapi > lib > x86_64
Sort
View

Name	Date modified	Type	Size
libgomp.so.1.0.0	11/15/2013 11:24 PM	0 File	59 KB
libhtraapi.so.0.55.55	1/6/2025 10:17 AM	55 File	1,087 KB
libliquid.so	1/12/2024 12:00 AM	SO File	1,026 KB
libusb-1.0.so.0.2.0	1/12/2024 12:00 AM	0 File	425 KB
README.txt	1/12/2024 12:00 AM	Text Document	1 KB

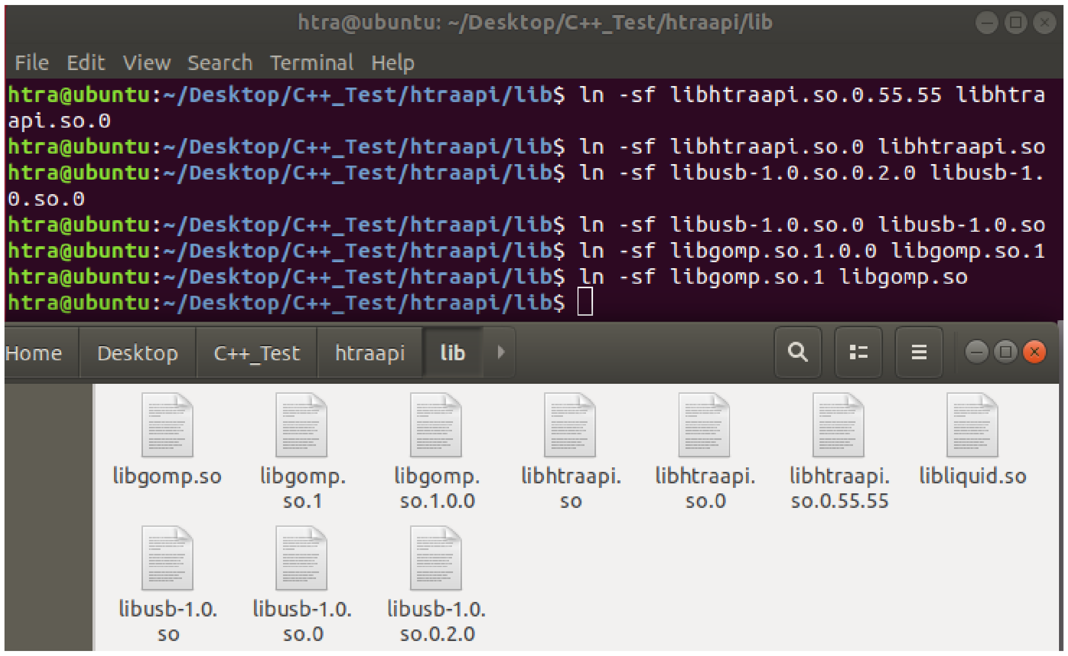


6. Open the terminal in the lib folder location and enter the following commands to create soft links for the copied dynamic link libraries (the commands for the dynamic link libraries of the three architectures are the same):

- ln -sf libhtraapi.so.0.55.5 libhtraapi.so.0

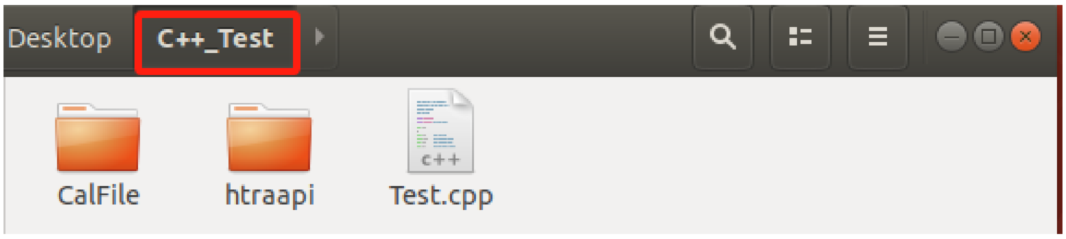
(Here, the libhtraapi.so library is taken as an example with version 0.55.55; modify the version number for other versions.)

- ln -sf libhtraapi.so.0 libhtraapi.so
- ln -sf libusb-1.0.so.0.2.0 libusb-1.0.so.0
- ln -sf libusb-1.0.so.0 libusb-1.0.so
- ln -sf libgomp.so.1.0.0 libgomp.so.1
- ln -sf libgomp.so.1 libgomp.so


htra@ubuntu: ~/Desktop/C++_Test/htraapi/lib
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/C++_Test/htraapi/lib$ ln -sf libhtraapi.so.0.55.55 libhtra
api.so.0
htra@ubuntu:~/Desktop/C++_Test/htraapi/lib$ ln -sf libhtraapi.so.0 libhtraapi.so
htra@ubuntu:~/Desktop/C++_Test/htraapi/lib$ ln -sf libusb-1.0.so.0.2.0 libusb-1.
0.so.0
htra@ubuntu:~/Desktop/C++_Test/htraapi/lib$ ln -sf libusb-1.0.so.0 libusb-1.0.so
htra@ubuntu:~/Desktop/C++_Test/htraapi/lib$ ln -sf libgomp.so.1.0.0 libgomp.so.1
htra@ubuntu:~/Desktop/C++_Test/htraapi/lib$ ln -sf libgomp.so.1 libgomp.so
htra@ubuntu:~/Desktop/C++_Test/htraapi/lib$ 
```

7. Store the written code files in the outermost folder of C++_Test.



8. Compile to generate the executable file: First, check the system architecture according to the process in Chapter 9.1, then open the terminal in the C++_Test folder (the image below takes the x86_64 system as an example), and input according to the host system architecture.

1) For x86_64 system input (example test.cpp here):

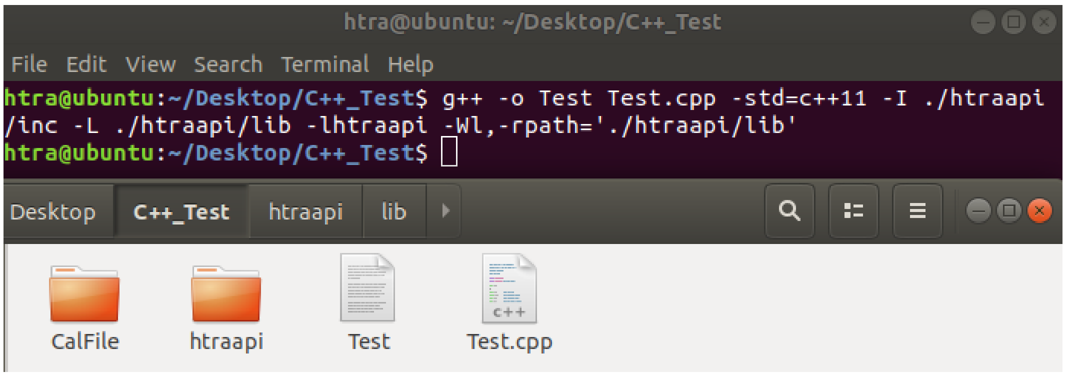
g++ -o Test Test.cpp -std=c++11 -I ./htraapi/inc -L ./htraapi/lib -lhtraapi -Wl,-rpath='./htraapi/lib'

2) Input for arrch64 system:

aarch64-linux-gnu-g++-o Test Test.cpp -std=c++11 -I ./htraapi/inc -L ./htraapi/lib -lhtraapi -Wl,-rpath='./htraapi/lib'

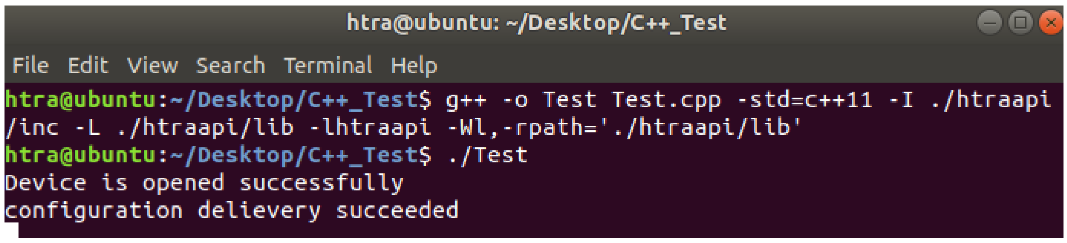
3) For armv7 system, input:

arm-linux-gnueabihf-g++-o Test Test.cpp -std=c++11 -I ./htraapi/inc -L ./htraapi/lib -lhtraapi -Wl,-rpath='./htraapi/lib'


htra@ubuntu: ~/Desktop/C++_Test
File Edit View Search Terminal Help
Desktop
C++_Test
htraapi
lib
CalFile
htraapi
Test
Test.cpp

```
htra@ubuntu:~/Desktop/C++_Test$ g++ -o Test Test.cpp -std=c++11 -I ./htraapi
/inc -L ./htraapi/lib -lhtraapi -Wl,-rpath='./htraapi/lib'
htra@ubuntu:~/Desktop/C++_Test$ 
```

9. Run the program: Enter ./Test to start the executable file.


htra@ubuntu: ~/Desktop/C++_Test
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/C++_Test$ g++ -o Test Test.cpp -std=c++11 -I ./htraapi
/inc -L ./htraapi/lib -lhtraapi -Wl,-rpath='./htraapi/lib'
htra@ubuntu:~/Desktop/C++_Test$ ./Test
Device is opened successfully
configuration delievery succeeded
```

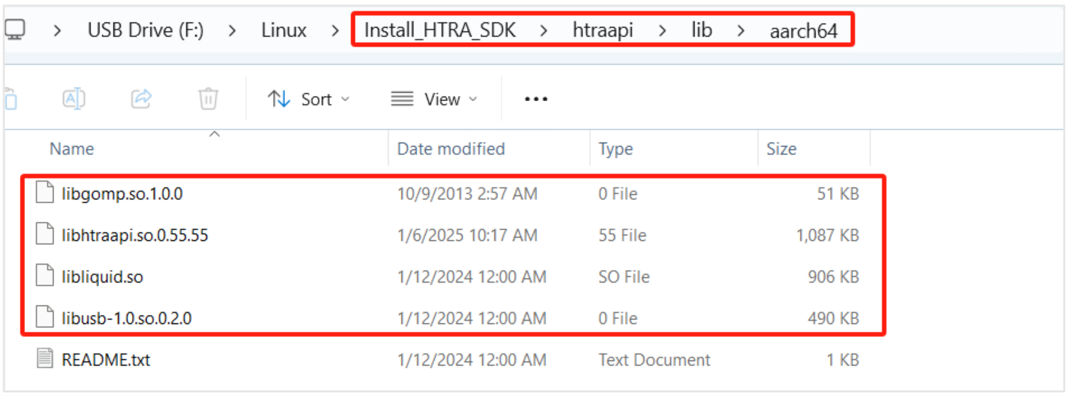
9.4.3 C++ Project Cross-Compilation

Under the premise that the host computer has a cross-compilation toolchain, if you want to cross-compile to use the device, please refer to the following process (taking the cross-compilation of an aarch64 executable program on an x86_64 host computer as an example):

1. First, generate the executable file for the target architecture:

1) Create a project and place the calibration files and header files according to steps 1-4 in the compilation and running method in Chapter 9.4.2.

2) Place the cross-compiled target architecture library files as per step 5. For example, when cross-compiling an executable for the arrch64 architecture, please place the library files for the arrch64 architecture.

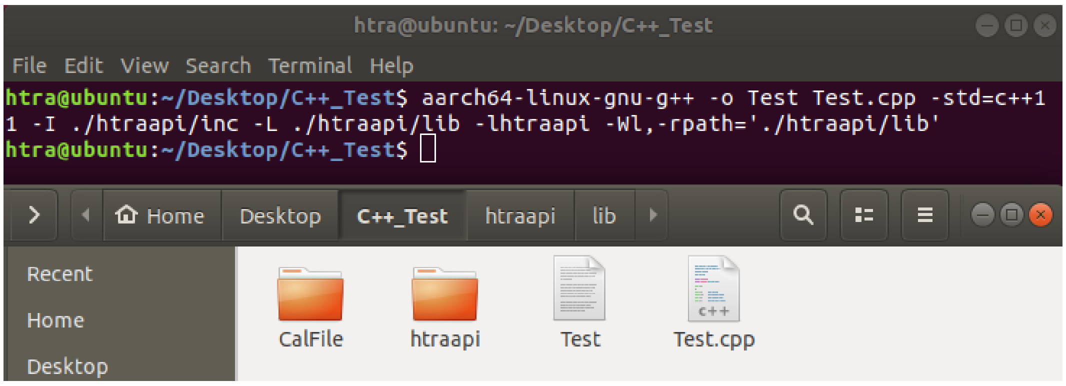


USB Drive (F:) > Linux > Install_HTRA_SDK > htraapi > lib > aarch64
Sort
View

Name	Date modified	Type	Size
libgomp.so.1.0.0	10/9/2013 2:57 AM	0 File	51 KB
libhtraapi.so.0.55.55	1/6/2025 10:17 AM	55 File	1,087 KB
libliquid.so	1/12/2024 12:00 AM	SO File	906 KB
libusb-1.0.so.0.2.0	1/12/2024 12:00 AM	0 File	490 KB
README.txt	1/12/2024 12:00 AM	Text Document	1 KB

3) Perform soft linking and program writing storage according to steps 6-7.

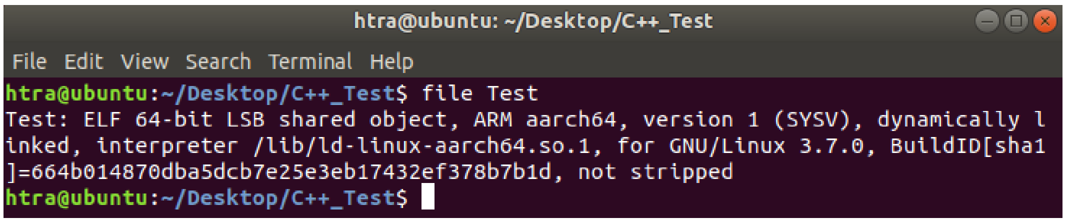
4) Compile the executable file using the target architecture compilation command as described in step 8. The compilation command for the aarch64 architecture is expected to be used when compiling the executable file for the aarch64 system, as shown in the figure below.



```
htra@ubuntu:~/Desktop/C++_Test$ aarch64-linux-gnu-g++ -o Test Test.cpp -std=c++1
1 -I ./htraapi/inc -L ./htraapi/lib -lhtraapi -Wl,-rpath='./htraapi/lib'
htra@ubuntu:~/Desktop/C++_Test$ 
```

htra@ubuntu: ~/Desktop/C++_Test
File Edit View Search Terminal Help
Home Desktop C++_Test htraapi lib
Recent
Home
Desktop
CalFile
htraapi
Test
Test.cpp

5) After generating the executable file, input 'file Test' to check the architecture of the executable program, and you will see that the current executable program architecture is aarch64.

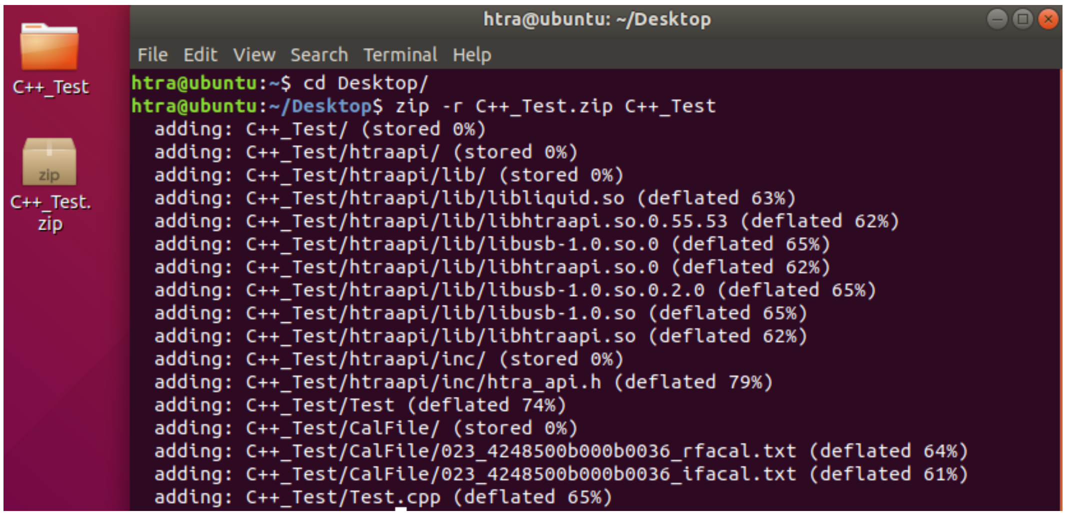


```
htra@ubuntu:~/Desktop/C++_Test$ file Test
Test: ELF 64-bit LSB shared object, ARM aarch64, version 1 (SYSV), dynamically l
inked, interpreter /lib/ld-linux-aarch64.so.1, for GNU/Linux 3.7.0, BuildID[sha1
]=664b014870dba5dcb7e25e3eb17432ef378b7b1d, not stripped
htra@ubuntu:~/Desktop/C++_Test$ 
```

htra@ubuntu: ~/Desktop/C++_Test
File Edit View Search Terminal Help

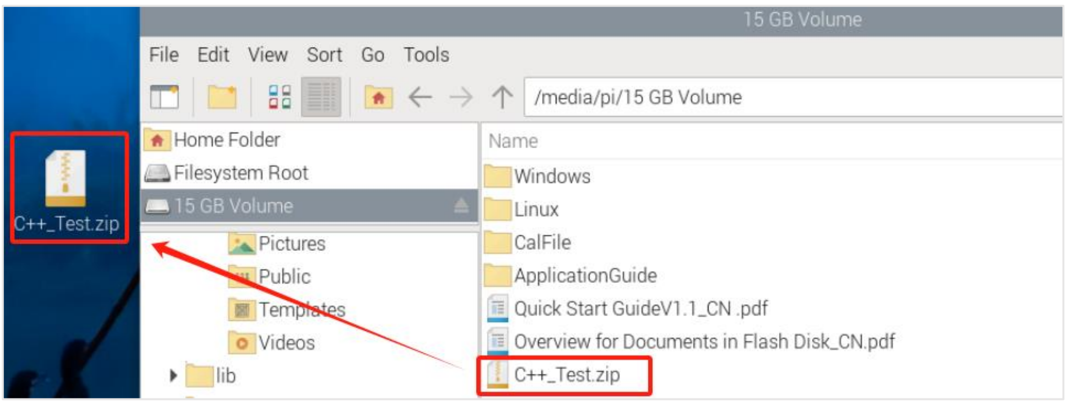
2. At this point, the executable program has been successfully generated, and you can now run the program on the aarch64 host machine using the device:

1) Navigate to the project directory (the example project is located on the desktop, so input 'cd Desktop/'), and compress the entire folder into a zip file, for example, input 'zip -r C++_Test.zip C++_Test' to create the zip archive.

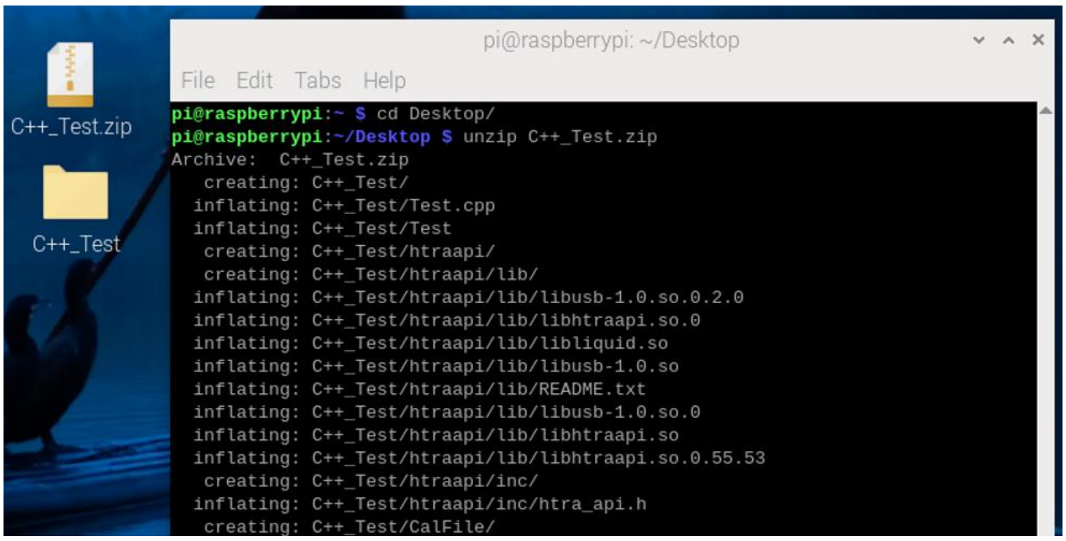


```
htra@ubuntu:~$ cd Desktop/
htra@ubuntu:~/Desktop$ zip -r C++_Test.zip C++_Test
  adding: C++_Test/ (stored 0%)
  adding: C++_Test/htraapi/ (stored 0%)
  adding: C++_Test/htraapi/lib/ (stored 0%)
  adding: C++_Test/htraapi/lib/libliquid.so (deflated 63%)
  adding: C++_Test/htraapi/lib/libhtraapi.so.0.55.53 (deflated 62%)
  adding: C++_Test/htraapi/lib/libusb-1.0.so.0 (deflated 65%)
  adding: C++_Test/htraapi/lib/libhtraapi.so.0 (deflated 62%)
  adding: C++_Test/htraapi/lib/libusb-1.0.so.0.2.0 (deflated 65%)
  adding: C++_Test/htraapi/lib/libusb-1.0.so (deflated 65%)
  adding: C++_Test/htraapi/lib/libhtraapi.so (deflated 62%)
  adding: C++_Test/htraapi/inc/ (stored 0%)
  adding: C++_Test/htraapi/inc/htra_api.h (deflated 79%)
  adding: C++_Test/Test (deflated 74%)
  adding: C++_Test/CalFile/ (stored 0%)
  adding: C++_Test/CalFile/023_4248500b000b0036_rfacal.txt (deflated 64%)
  adding: C++_Test/CalFile/023_4248500b000b0036_ifacal.txt (deflated 61%)
  adding: C++_Test/Test.cpp (deflated 65%)
```

2) Copy the zip file to the aarch64 host machine.



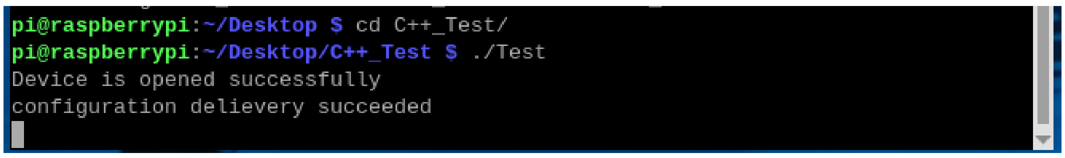
3) Navigate to the location where the zip file is stored (in this example, the zip file is on the desktop, so input 'cd Desktop/'), and extract the project (input 'unzip C++_Test').



```
pi@raspberrypi:~ $ cd Desktop/
pi@raspberrypi:~/Desktop $ unzip C++_Test.zip
Archive:  C++_Test.zip
   creating: C++_Test/
  inflating: C++_Test/Test.cpp
  inflating: C++_Test/Test
   creating: C++_Test/htraapi/
   creating: C++_Test/htraapi/lib/
  inflating: C++_Test/htraapi/lib/libusb-1.0.so.0.2.0
  inflating: C++_Test/htraapi/lib/libhtraapi.so.0
  inflating: C++_Test/htraapi/lib/libliquid.so
  inflating: C++_Test/htraapi/lib/libusb-1.0.so
  inflating: C++_Test/htraapi/lib/README.txt
  inflating: C++_Test/htraapi/lib/libusb-1.0.so.0
  inflating: C++_Test/htraapi/lib/libhtraapi.so
  inflating: C++_Test/htraapi/lib/libhtraapi.so.0.55.53
   creating: C++_Test/htraapi/inc/
  inflating: C++_Test/htraapi/inc/htra_api.h
   creating: C++_Test/CalFile/
```

4) Configure the driver files on the aarch64 host machine as per the steps in Chapter9.3.

5) After configuring the driver files, input 'cd C++_Test/' to enter the folder, and then simply input './Test' to run the program.



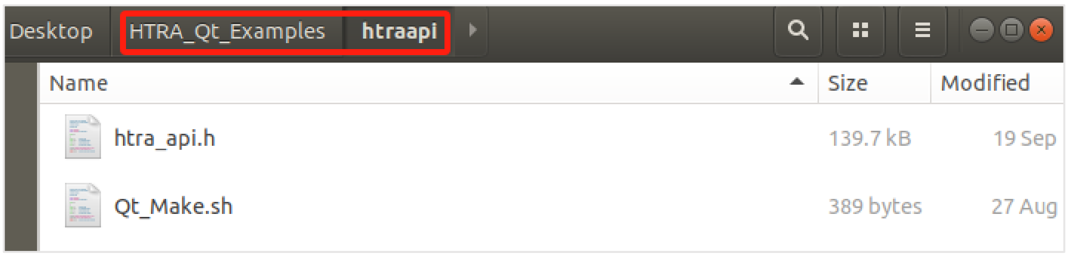
```
pi@raspberrypi:~/Desktop $ cd C++_Test/
pi@raspberrypi:~/Desktop/C++_Test $ ./Test
Device is opened successfully
configuration delievery succeeded
```

9.5 Using Qt Examples and Project Creation

9.5.1 Using Qt Examples

To use the Qt examples included on the USB drive, provided that the device is properly connected and the driver files have been configured correctly as per Chapter9.3, please refer to the following process (the purpose of this Qt example is to obtain complete spectrum data within a specified frequency band):

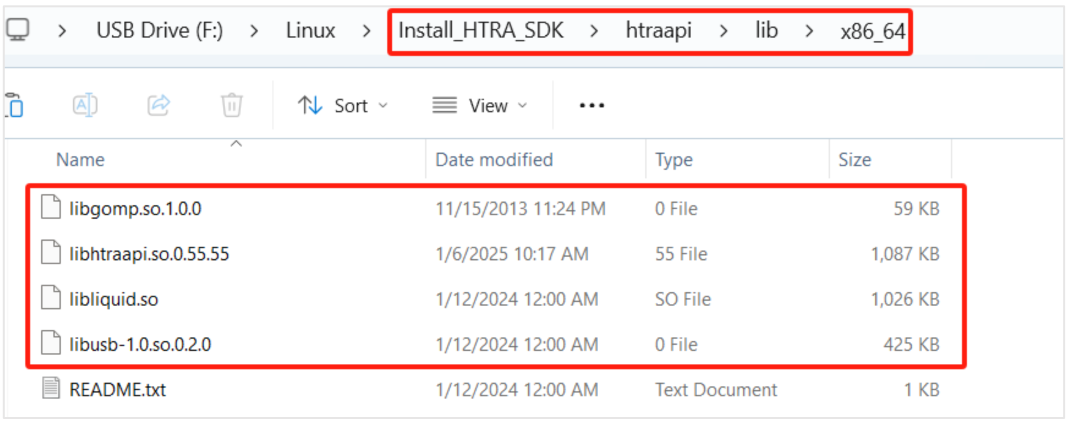
1. Copy the Linux\HTRA_Qt_Examples folder from the included USB drive to the host computer, and then navigate to the HTRA_Qt_Examples\htraapi subfolder as shown in the figure.



Desktop > HTRA_Qt_Examples > htraapi

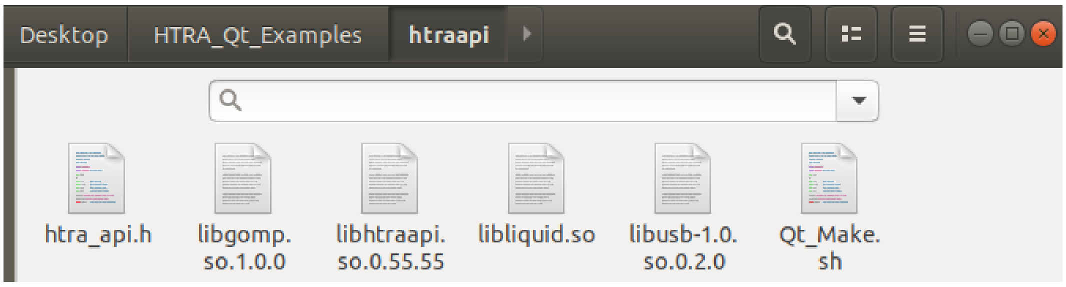
Name	Size	Modified
htra_api.h	139.7 kB	19 Sep
Qt_Make.sh	389 bytes	27 Aug

2. Refer to the system architecture in the process outlined in Chapter 9.1, and then follow the instructions in Chapter 9.2.4 to copy the dynamic link libraries corresponding to the system architecture from the Linux\Install_HTRA_SDK\htraapi\lib folder on the USB drive to the HTRA_Qt_Examples\htraapi folder (this example uses an x86_64 architecture host computer).

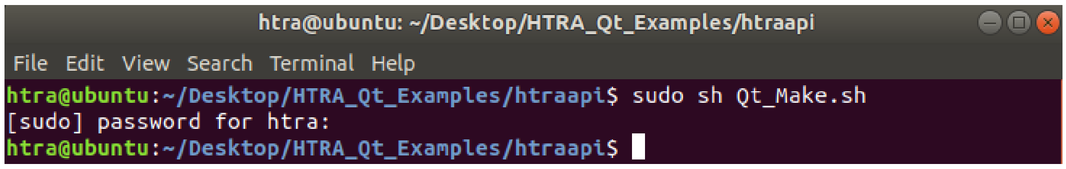


USB Drive (F:) > Linux > Install_HTRA_SDK > htraapi > lib > x86_64

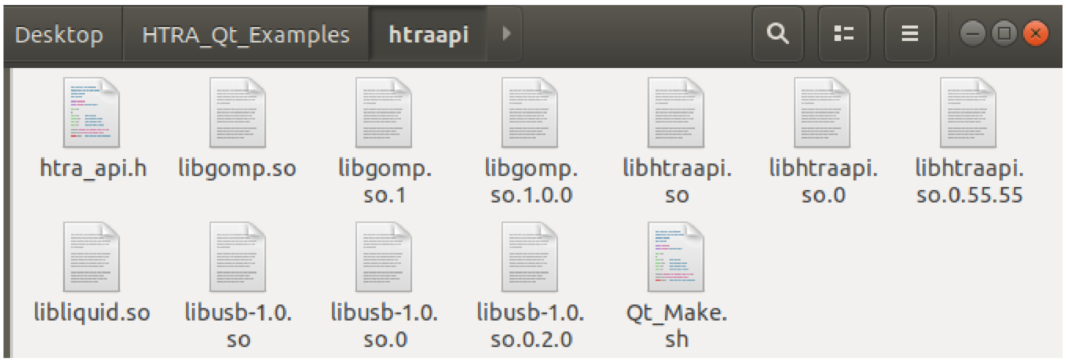
Name	Date modified	Type	Size
libgomp.so.1.0.0	11/15/2013 11:24 PM	0 File	59 KB
libhtraapi.so.0.55.55	1/6/2025 10:17 AM	55 File	1,087 KB
libliquid.so	1/12/2024 12:00 AM	SO File	1,026 KB
libusb-1.0.so.0.2.0	1/12/2024 12:00 AM	0 File	425 KB
README.txt	1/12/2024 12:00 AM	Text Document	1 KB



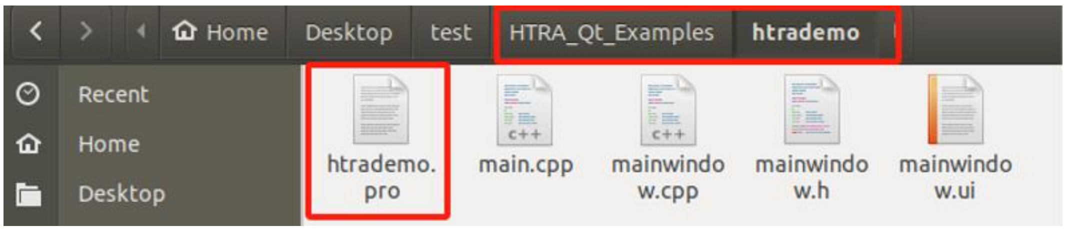
3. Open a terminal in the current folder, enter "sudo sh Qt_Make.sh", and then follow the prompts to enter the sudo password to grant permission for creating soft links to the libraries.


htra@ubuntu: ~/Desktop/HTRA_Qt_Examples/htraapi
File Edit View Search Terminal Help

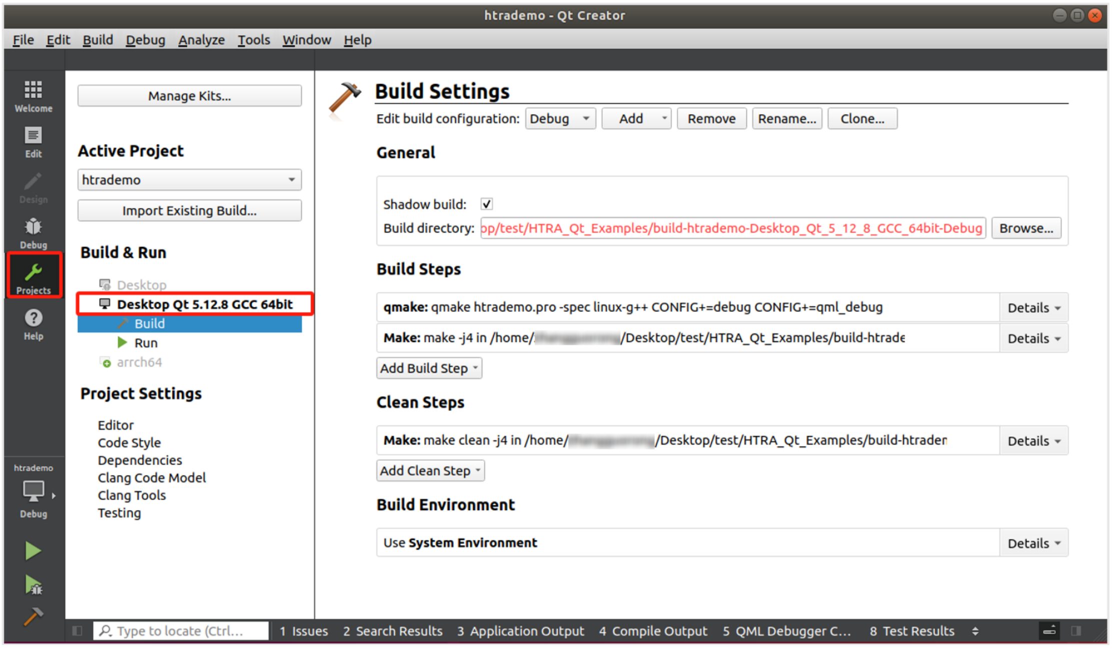
```
htra@ubuntu:~/Desktop/HTRA_Qt_Examples/htraapi$ sudo sh Qt_Make.sh
[sudo] password for htra:
htra@ubuntu:~/Desktop/HTRA_Qt_Examples/htraapi$ 
```



4. After running the script, use Qt Creator to open the htrademo.pro file in the htrademo folder.



5. First, select the build environment for the project.



Build Settings

Edit build configuration: Debug

General

Shadow build: ✓

Build directory: op/test/HTRA_Qt_Examples/build-htrademo-Desktop_Qt_5_12_8_GCC_64bit-Debug

Build Steps

qmake: qmake htrademo.pro -spec linux-g++ CONFIG+=debug CONFIG+=qml_debug

Make: make -j4 in /home/[redacted]/Desktop/test/HTRA_Qt_Examples/build-htrade

Clean Steps

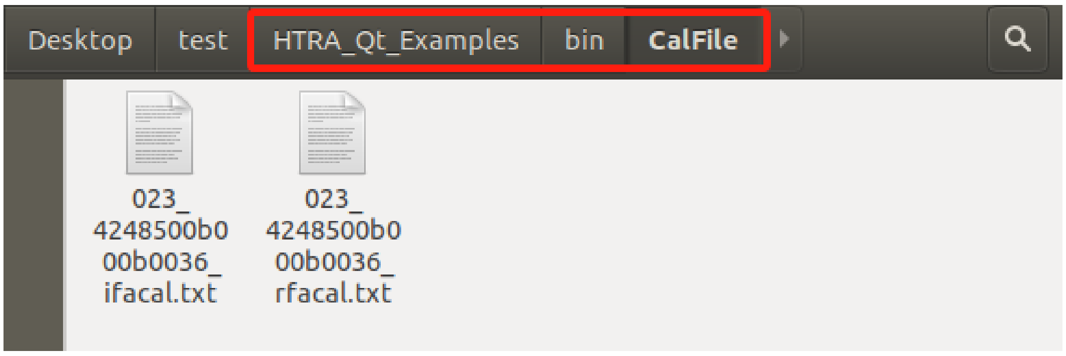
Make: make clean -j4 in /home/[redacted]/Desktop/test/HTRA_Qt_Examples/build-htrader

Build Environment

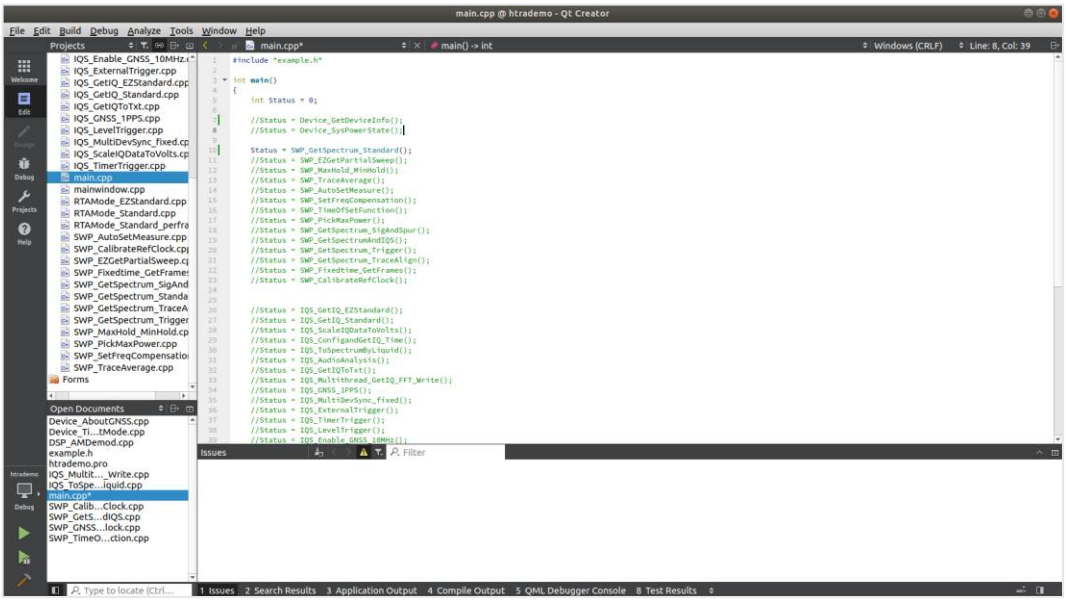
Use **System Environment**

htrademo - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Manage Kits...
Active Project
htrademo
Import Existing Build...
Build & Run
Desktop
Desktop Qt 5.12.8 GCC 64bit
Build
Run
arrch64
Project Settings
Editor
Code Style
Dependencies
Clang Code Model
Clang Tools
Testing
Add Remove Rename... Clone... Browse... Details Add Build Step Add Clean Step
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger C... 8 Test Results

6. Open the HTRA_Qt_Examples\bin\CalFile folder and ensure that there is a device calibration file in the folder.

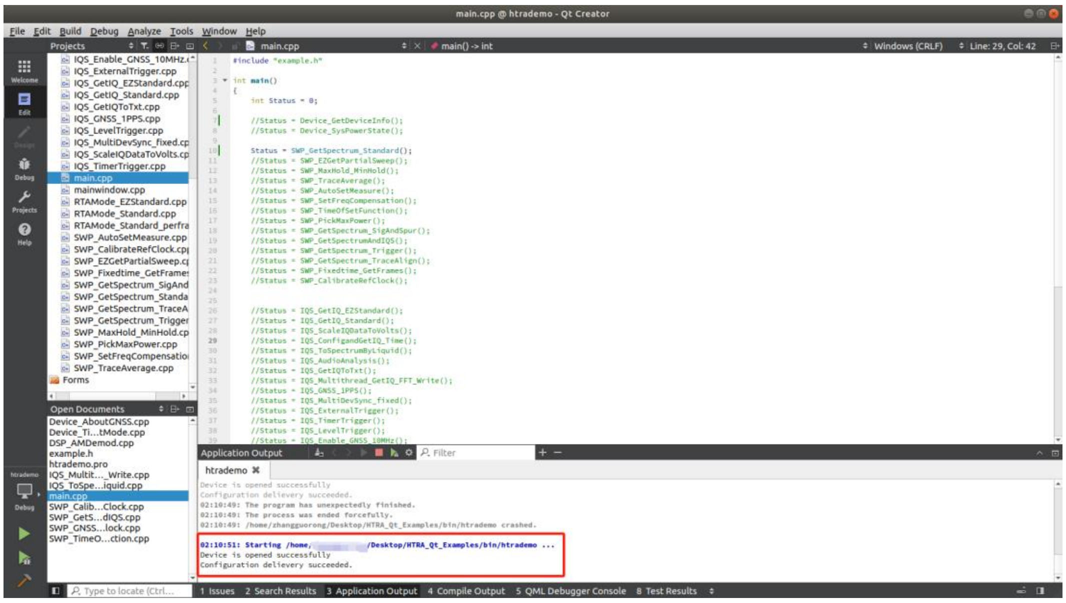

Desktop
test
HTRA_Qt_Examples
bin
CalFile
023_4248500b000b0036_ifacal.txt
023_4248500b000b0036_rfacal.txt

7. After confirming that there is a calibration file, select any example in main.cpp.



```
#include "example.h"

int main()
{
    int Status = 0;

    //Status = Device_GetDeviceInfo();
    //Status = Device_SysPowerState();

    Status = SWP_GetSpectrum_Standard();
    //Status = SWP_EZGetPartialSweep();
    //Status = SWP_MaxHold_MinHold();
    //Status = SWP_TraceAverage();
    //Status = SWP_AutoSetMeasure();
    //Status = SWP_SetFreqCompensation();
    //Status = SWP_TimeOfSetFunction();
    //Status = SWP_PickMaxPower();
    //Status = SWP_GetSpectrum_SigAndSpur();
    //Status = SWP_GetSpectrumAndIQS();
    //Status = SWP_GetSpectrum_Trigger();
    //Status = SWP_GetSpectrum_TraceAlign();
    //Status = SWP_Fixedtime_GetFrames();
    //Status = SWP_CalibrateRefClock();


    //Status = IQS_GetIQ_EZStandard();
    //Status = IQS_GetIQ_Standard();
    //Status = IQS_ScaleIQDataToVolts();
    //Status = IQS_ConfigandGetIQ_Time();
    //Status = IQS_ToSpectrumByLiquid();
    //Status = IQS_AudioAnalysis();
    //Status = IQS_GetIQToTxt();
    //Status = IQS_Multithread_GetIQ_FFT_Write();
    //Status = IQS_GNSS_1PPS();
    //Status = IQS_MultiDevSync_fixed();
    //Status = IQS_ExternalTrigger();
    //Status = IQS_TimerTrigger();
    //Status = IQS_LevelTrigger();
    //Status = IQS_Enable_GNSS_10MHz();
```

8. Click Run, and as shown in the figure, the device is functioning normally.



```
#include "example.h"

int main()
{
    int Status = 0;

    //Status = Device_GetDeviceInfo();
    //Status = Device_SysPowerState();

    Status = SWP_GetSpectrum_Standard();
    //Status = SWP_EZGetPartialSweep();
    //Status = SWP_MaxHold_MinHold();
    //Status = SWP_TraceAverage();
    //Status = SWP_AutoSetMeasure();
    //Status = SWP_SetFreqCompensation();
    //Status = SWP_TimeOfSetFunction();
    //Status = SWP_PickMaxPower();
    //Status = SWP_GetSpectrum_SigAndSpur();
    //Status = SWP_GetSpectrumAndIQS();
    //Status = SWP_GetSpectrum_Trigger();
    //Status = SWP_GetSpectrum_TraceAlign();
    //Status = SWP_Fixedtime_GetFrames();
    //Status = SWP_CalibrateRefClock();


    //Status = IQS_GetIQ_EZStandard();
    //Status = IQS_GetIQ_Standard();
    //Status = IQS_ScaleIQDataToVolts();
    //Status = IQS_ConfigandGetIQ_Time();
    //Status = IQS_ToSpectrumByLiquid();
    //Status = IQS_AudioAnalysis();
    //Status = IQS_GetIQToTxt();
    //Status = IQS_Multithread_GetIQ_FFT_Write();
    //Status = IQS_GNSS_1PPS();
    //Status = IQS_MultiDevSync_fixed();
    //Status = IQS_ExternalTrigger();
    //Status = IQS_TimerTrigger();
    //Status = IQS_LevelTrigger();
    //Status = IQS_Enable_GNSS_10MHz();
```

```
Device is opened successfully
Configuration delievery succeeded.
02:10:49: The program has unexpectedly finished.
02:10:49: The process was ended forcefully.
02:10:49: /home/zhangguorong/Desktop/HTRA_Qt_Examples/bin/htrademo crashed.

02:10:51: Starting /home/[redacted]/Desktop/HTRA_Qt_Examples/bin/htrademo ...
Device is opened successfully
Configuration delievery succeeded.
```

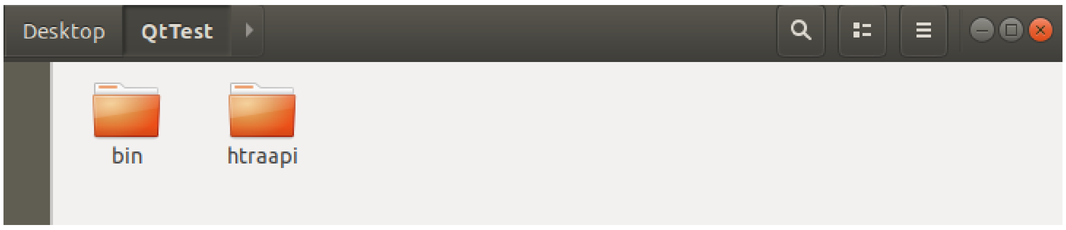
9.5.2 Qt Project Creation and Compilation

Under the premise that the driver files have been correctly configured according to Chapter 9.3, if you want to create a Qt project for compilation, please refer to the following process:

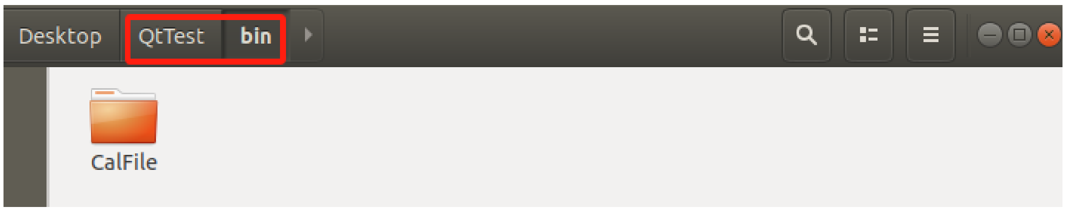
First, when writing code, since the Linux dynamic link libraries provided with the USB drive are identical to those for Windows, the code only needs to comply with the API programming guidelines.

Next, the process for creating the project is as follows:

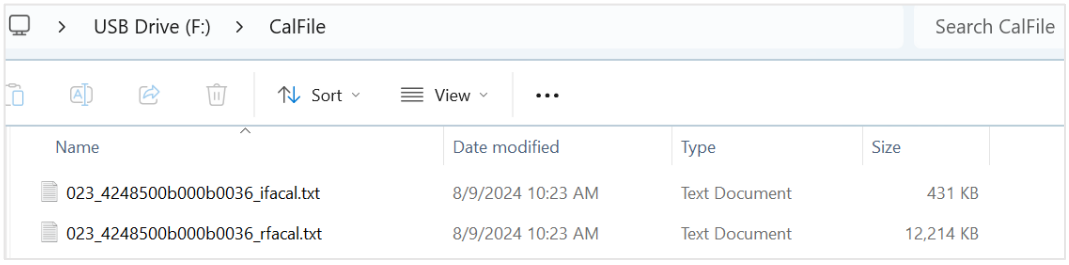
1. As shown in the figure, first create a new folder to store the entire project (taking QtTest as an example), then create a bin folder within this folder to store calibration files and the generated executable files, and create an htraapi folder to store header files and dynamic link libraries.



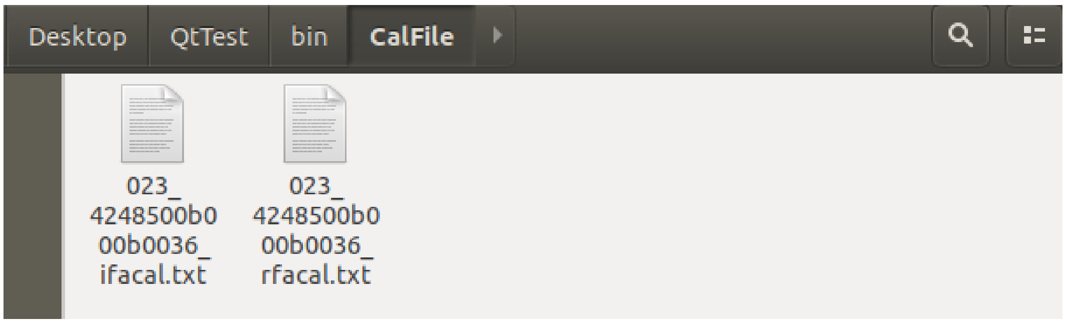
2. Create a CalFile folder under the bin folder to store the device calibration files.



3. Copy the files from the CalFile folder on the provided USB drive to the newly created QtTest\bin\CalFile folder.



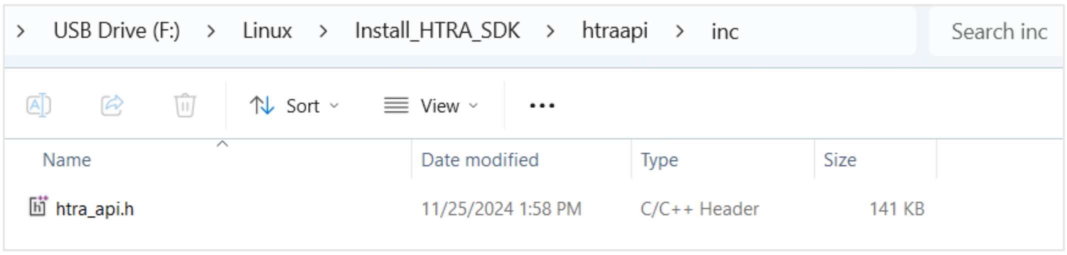
Name	Date modified	Type	Size
023_4248500b000b0036_ifacal.txt	8/9/2024 10:23 AM	Text Document	431 KB
023_4248500b000b0036_rfacal.txt	8/9/2024 10:23 AM	Text Document	12,214 KB



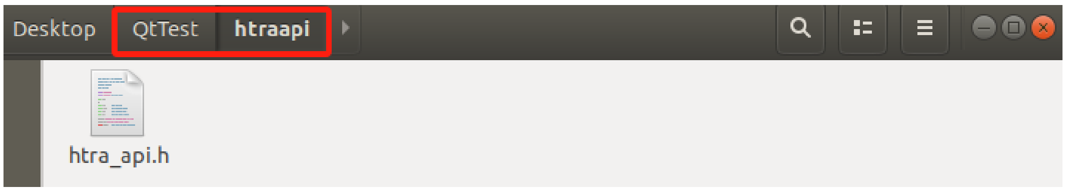
023_4248500b000b0036_ifacal.txt

023_4248500b000b0036_rfacal.txt

4. Copy the header files from the Linux\Install_HTRA_SDK\htraapi\inc folder on the provided USB drive to the newly created QtTest\htraapi folder.

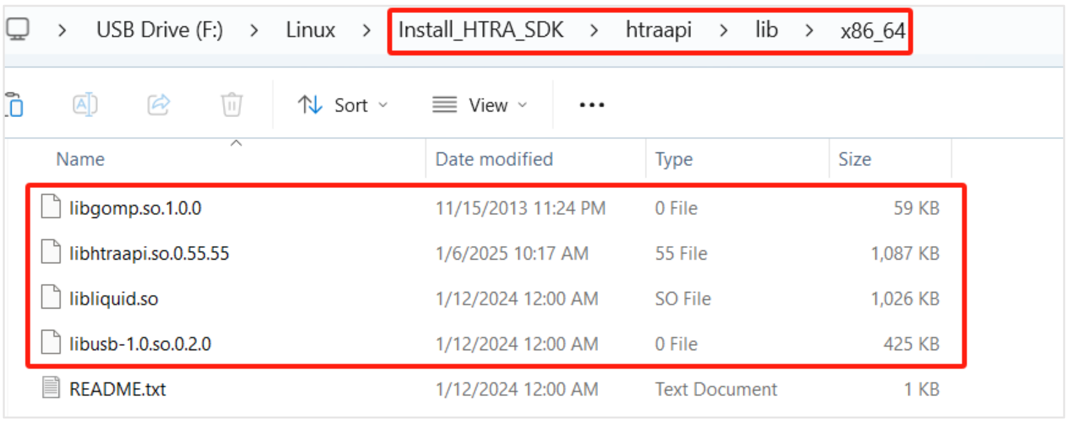


Name	Date modified	Type	Size
htra_api.h	11/25/2024 1:58 PM	C/C++ Header	141 KB

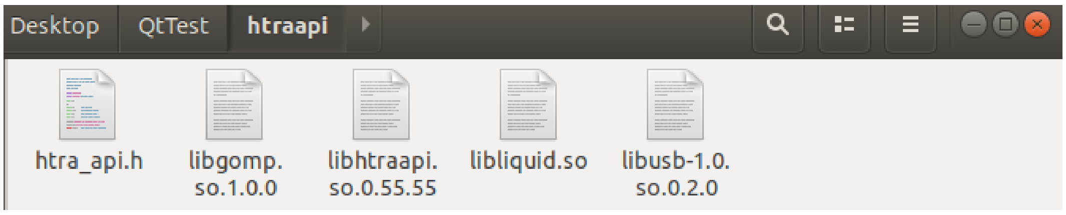


htra_api.h

5. Refer to the system architecture in Chapter 9.1, and then according to Chapter 9.2.4, copy the dynamic link libraries corresponding to the system architecture from the Linux\Install_HTRA_SDK\htraapi\lib folder on the provided USB drive to the newly created QtTest\htraapi folder (taking the x86_64 architecture for the host computer as an example).


USB Drive (F:) > Linux > Install_HTRA_SDK > htraapi > lib > x86_64
Sort
View

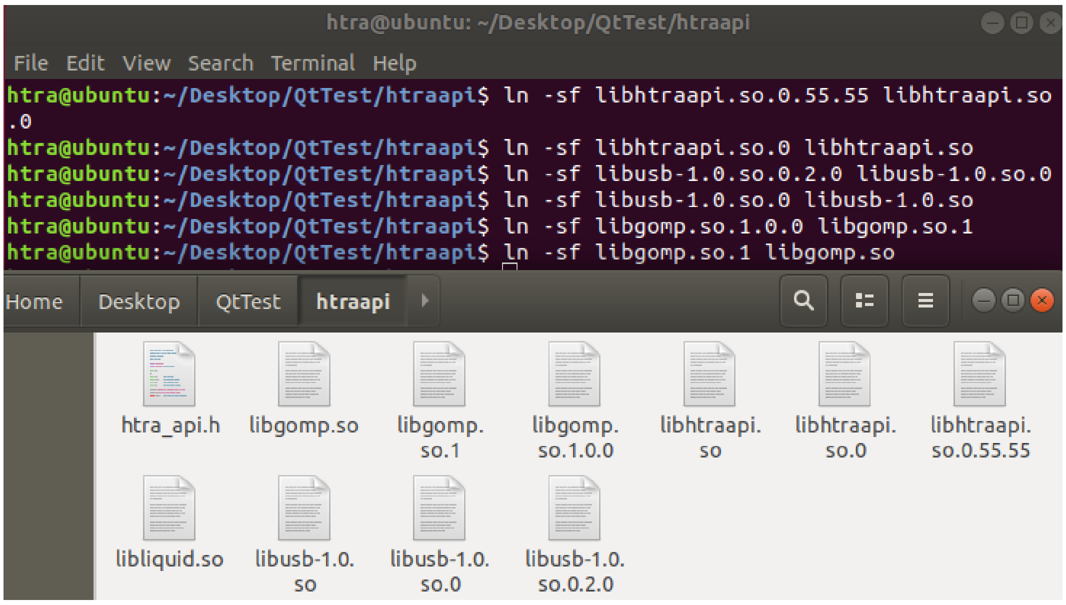
Name	Date modified	Type	Size
libgomp.so.1.0.0	11/15/2013 11:24 PM	0 File	59 KB
libhtraapi.so.0.55.55	1/6/2025 10:17 AM	55 File	1,087 KB
libliquid.so	1/12/2024 12:00 AM	SO File	1,026 KB
libusb-1.0.so.0.2.0	1/12/2024 12:00 AM	0 File	425 KB
README.txt	1/12/2024 12:00 AM	Text Document	1 KB



6. Open a terminal at the location of the htraapi folder and enter the following command to create a soft link for the copied dynamic link libraries (the soft link command for different architectures is the same):

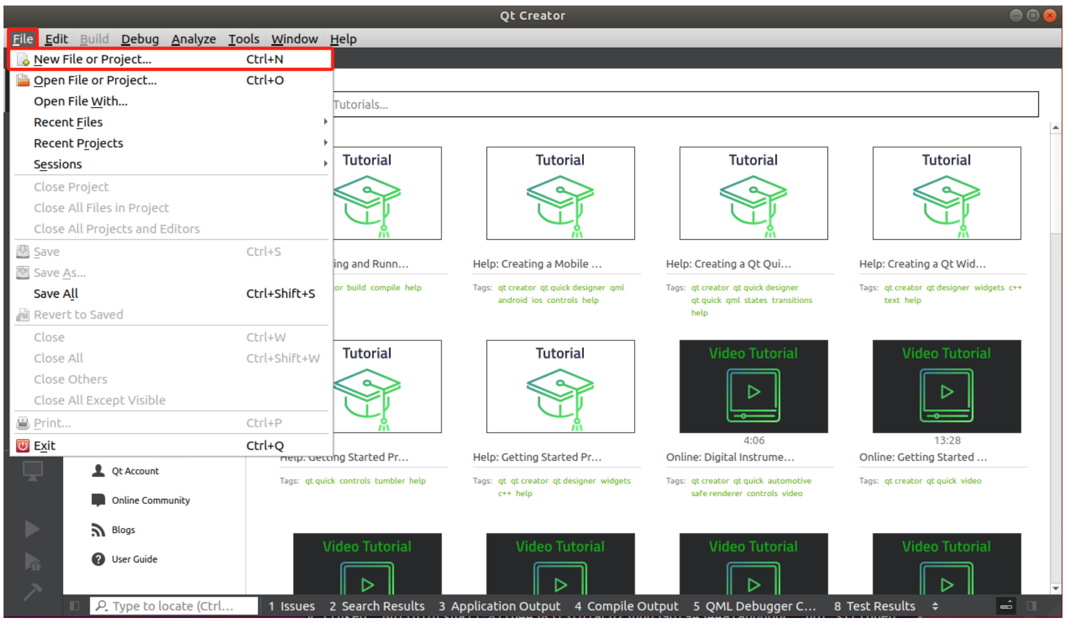
- ln -sf libhtraapi.so.0.55.5 libhtraapi.so.0

  (Here, the libhtraapi.so library is taken as an example with version 0.55.55; modify the version number for other versions.)
- ln -sf libhtraapi.so.0 libhtraapi.so
- ln -sf libusb-1.0.so.0.2.0 libusb-1.0.so.0
- ln -sf libusb-1.0.so.0 libusb-1.0.so
- ln -sf libgomp.so.1.0.0 libgomp.so.1
- ln -sf libgomp.so.1 libgomp.so

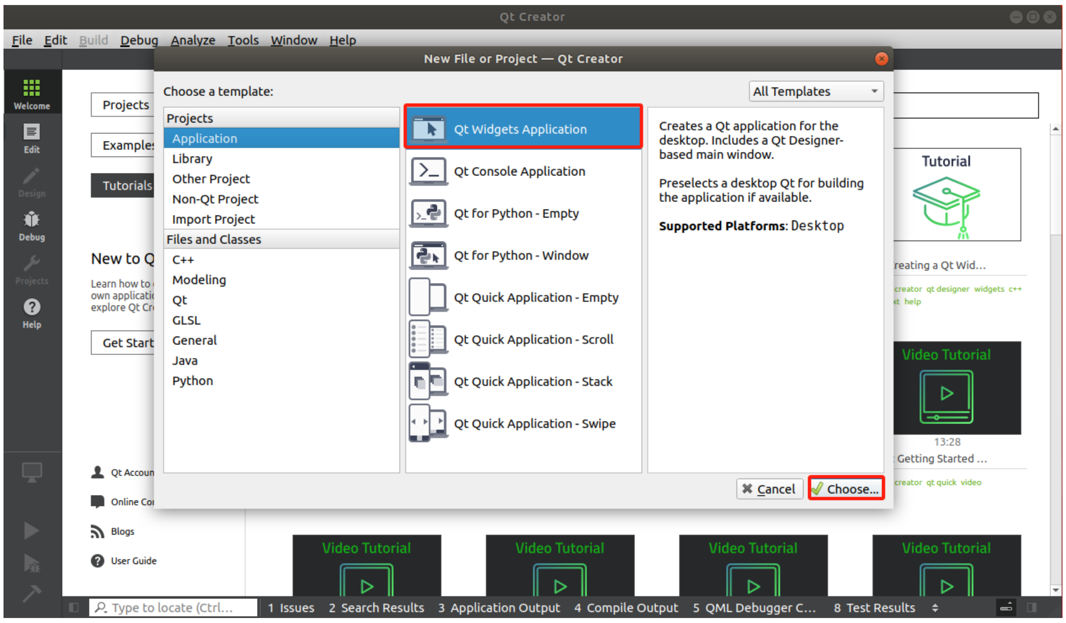

htra@ubuntu: ~/Desktop/QtTest/htraapi
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/QtTest/htraapi$ ln -sf libhtraapi.so.0.55.55 libhtraapi.so
.0
htra@ubuntu:~/Desktop/QtTest/htraapi$ ln -sf libhtraapi.so.0 libhtraapi.so
htra@ubuntu:~/Desktop/QtTest/htraapi$ ln -sf libusb-1.0.so.0.2.0 libusb-1.0.so.0
htra@ubuntu:~/Desktop/QtTest/htraapi$ ln -sf libusb-1.0.so.0 libusb-1.0.so
htra@ubuntu:~/Desktop/QtTest/htraapi$ ln -sf libgomp.so.1.0.0 libgomp.so.1
htra@ubuntu:~/Desktop/QtTest/htraapi$ ln -sf libgomp.so.1 libgomp.so
```

7. Open Qt Creator, click on File, and select New File or Project.



8. Select Create Form Application.

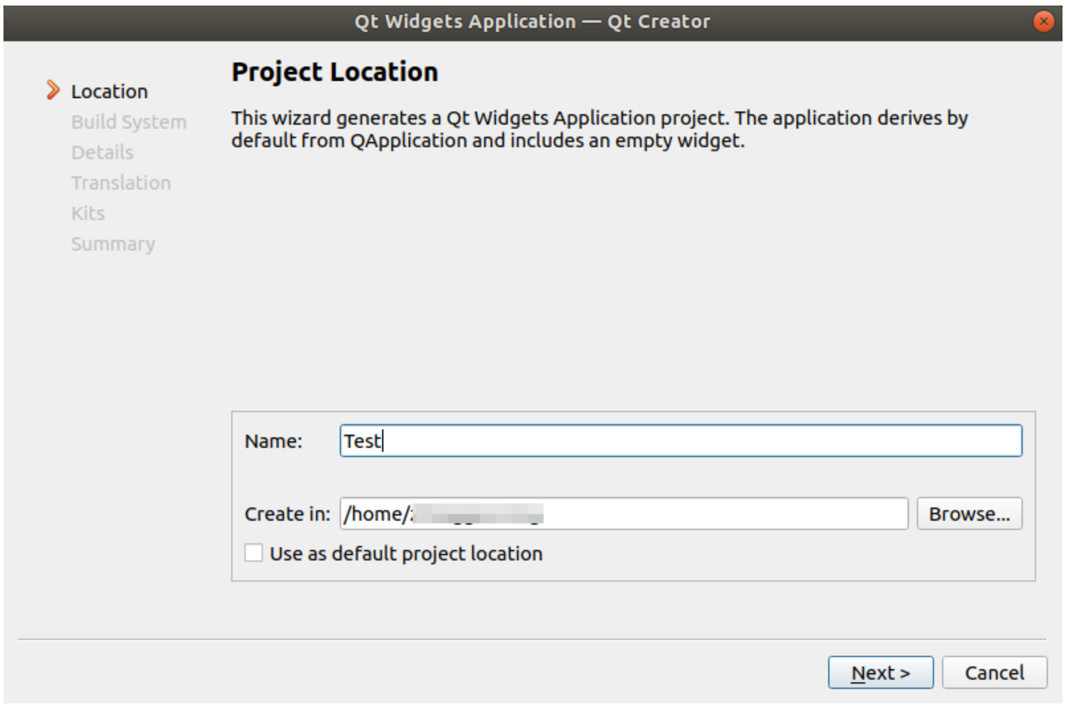


Creates a Qt application for the desktop. Includes a Qt Designer-based main window.

Preselects a desktop Qt for building the application if available.

Supported Platforms: Desktop

9. After filling in the project name, click Browse to change the project path.


Qt Widgets Application — Qt Creator

Location
Build System
Details
Translation
Kits
Summary

Project Location

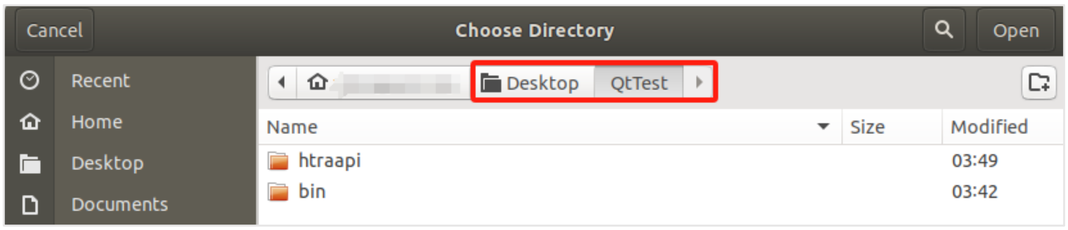
This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

Name: Test

Create in: /home/[redacted]

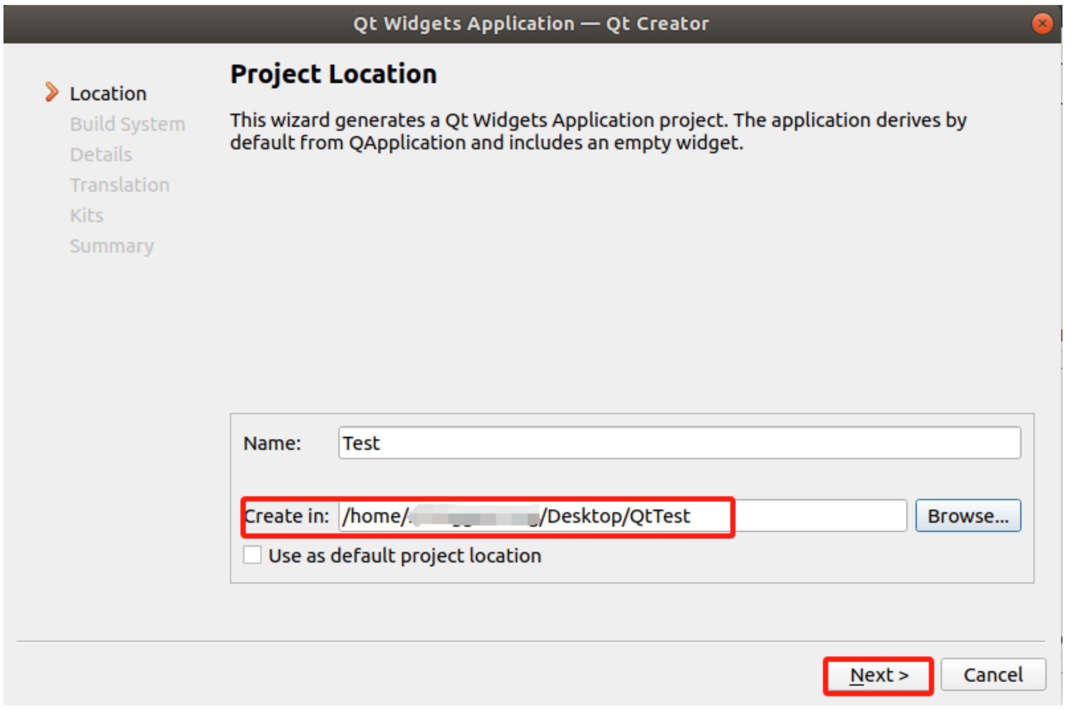
Use as default project location

10. Select the directory as the QtTest address created in the first step and click Open.


Choose Directory

Name	Size	Modified
htraapi		03:49
bin		03:42

11. After selecting the path, click Next.


Qt Widgets Application — Qt Creator

Location
Build System
Details
Translation
Kits
Summary

Project Location

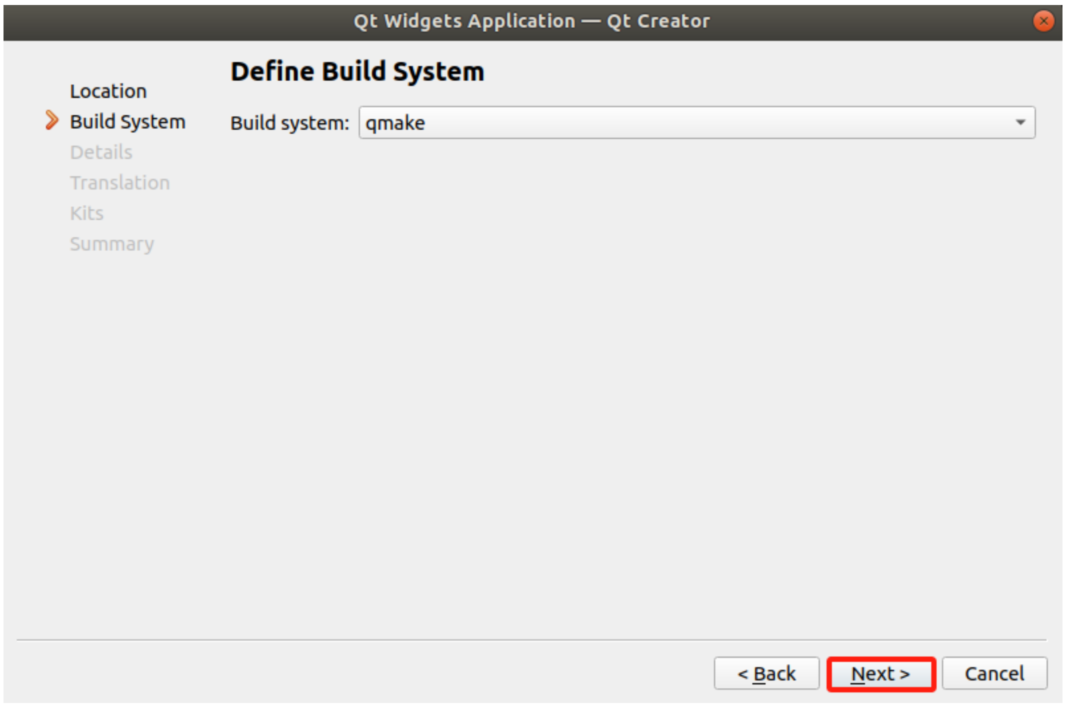
This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

Name: Test

Create in: /home/[redacted]/Desktop/QtTest

Use as default project location

12. Click Next to continue.

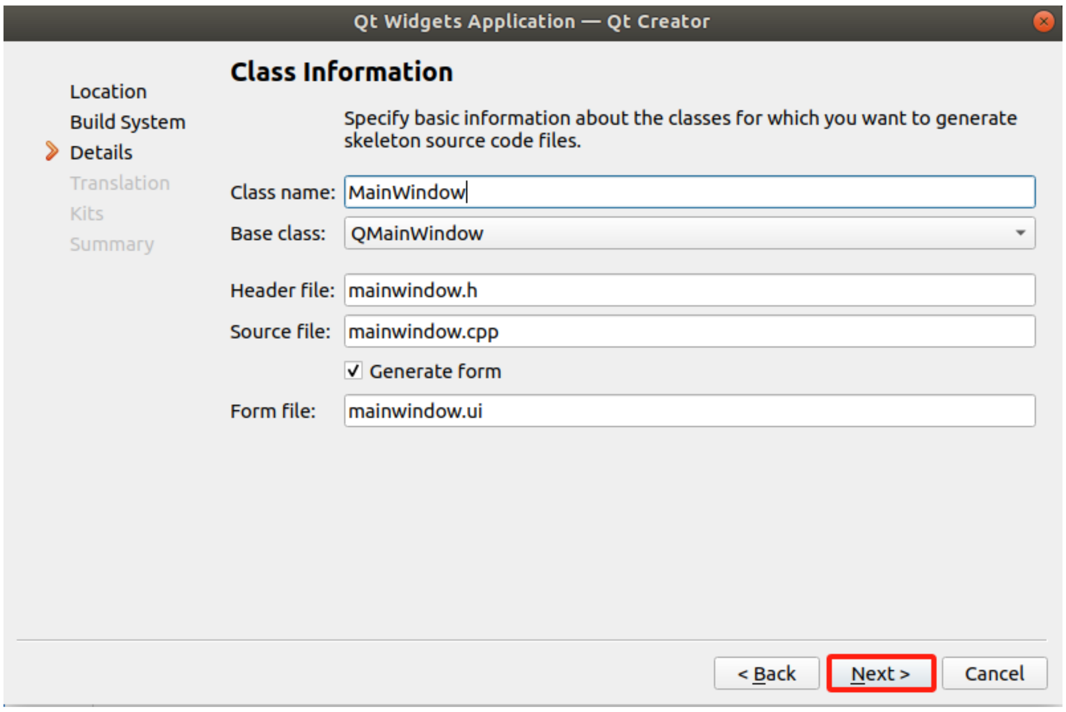

Qt Widgets Application — Qt Creator

Location
Build System
Details
Translation
Kits
Summary

Define Build System

Build system: qmake

13. Click Next to continue.


Qt Widgets Application — Qt Creator

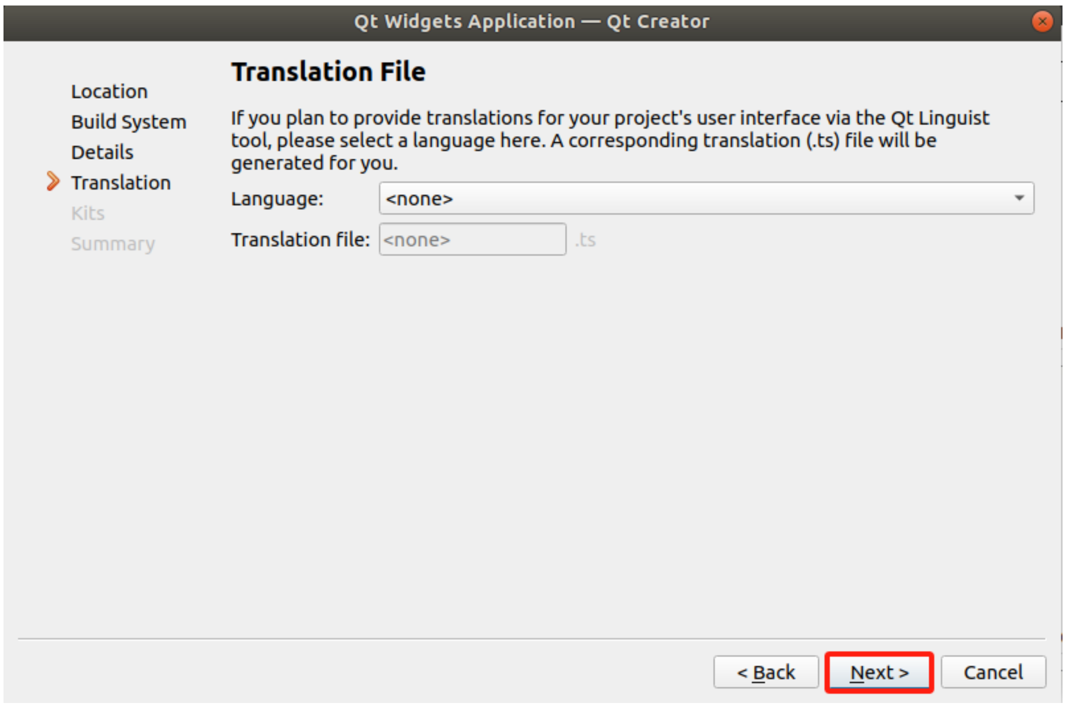
Location
Build System
Details
Translation
Kits
Summary

Class Information

Specify basic information about the classes for which you want to generate skeleton source code files.

Class name: MainWindow
Base class: QMainWindow
Header file: mainwindow.h
Source file: mainwindow.cpp
✓ Generate form
Form file: mainwindow.ui

14. Click Next to continue.


Qt Widgets Application — Qt Creator

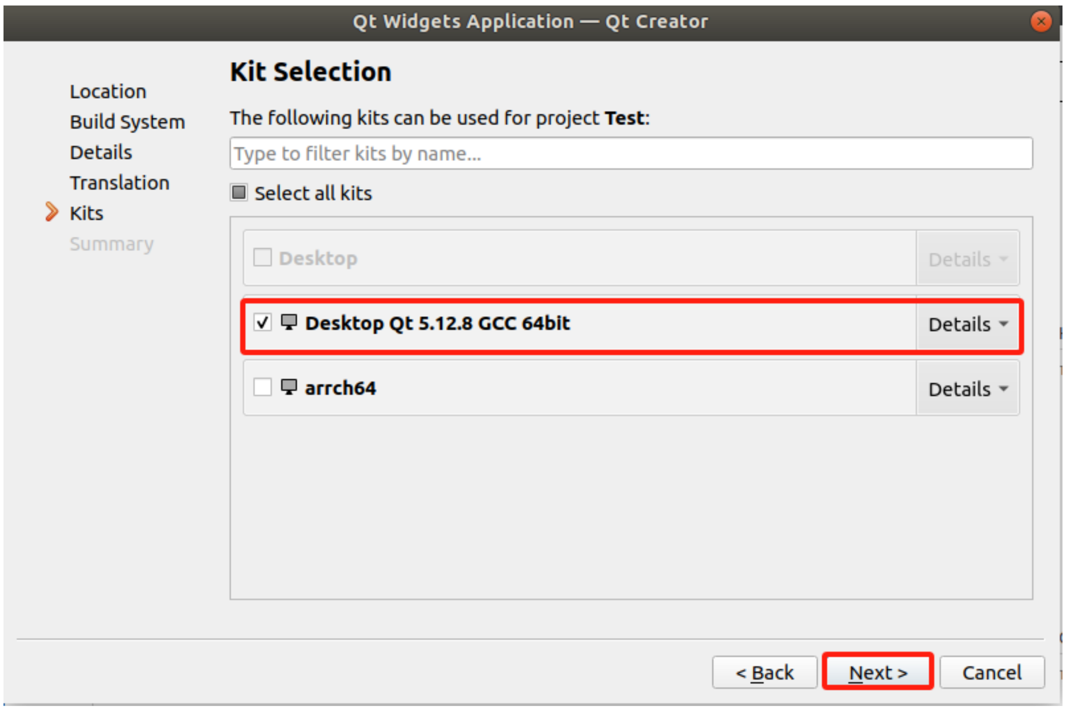
Translation File

If you plan to provide translations for your project's user interface via the Qt Linguist tool, please select a language here. A corresponding translation (.ts) file will be generated for you.

Language: <none>

Translation file: <none> .ts

15. Select the x86_64 build environment for the project and then continue by clicking next.


Qt Widgets Application — Qt Creator

Kit Selection

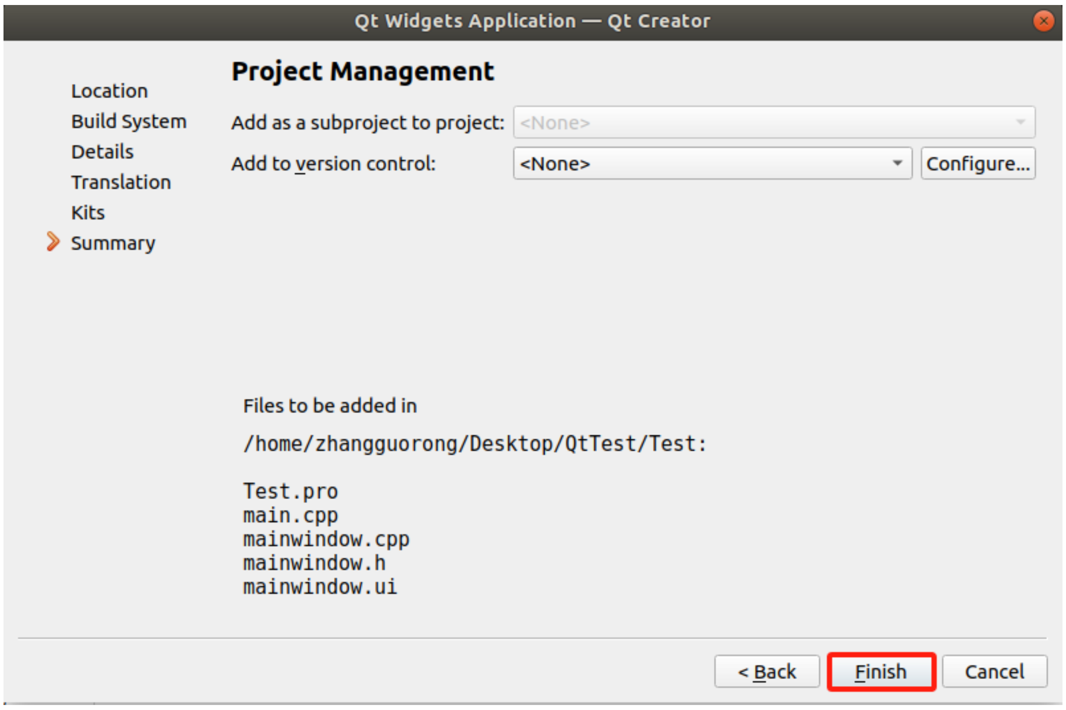
The following kits can be used for project **Test**:

Type to filter kits by name...

Select all kits

- [ ] Desktop — Details
- [✓] Desktop Qt 5.12.8 GCC 64bit — Details
- [ ] arrch64 — Details

16. Click Finish to create the project.


Qt Widgets Application — Qt Creator

Location
Build System
Details
Translation
Kits
Summary

Project Management

Add as a subproject to project: <None>

Add to version control: <None> Configure...

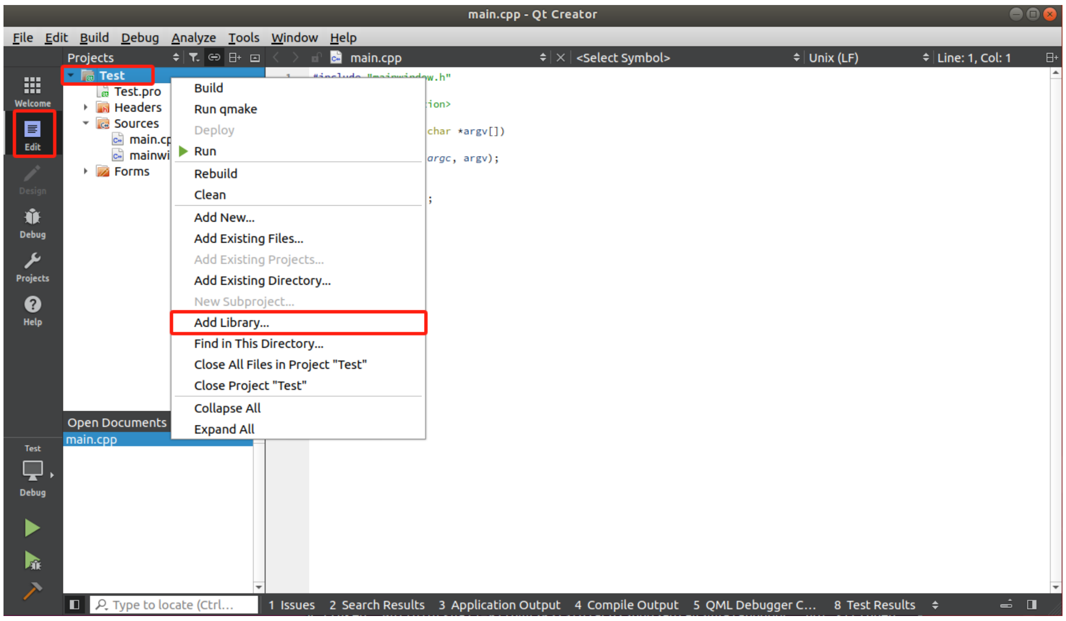
Files to be added in

/home/zhangguorong/Desktop/QtTest/Test:

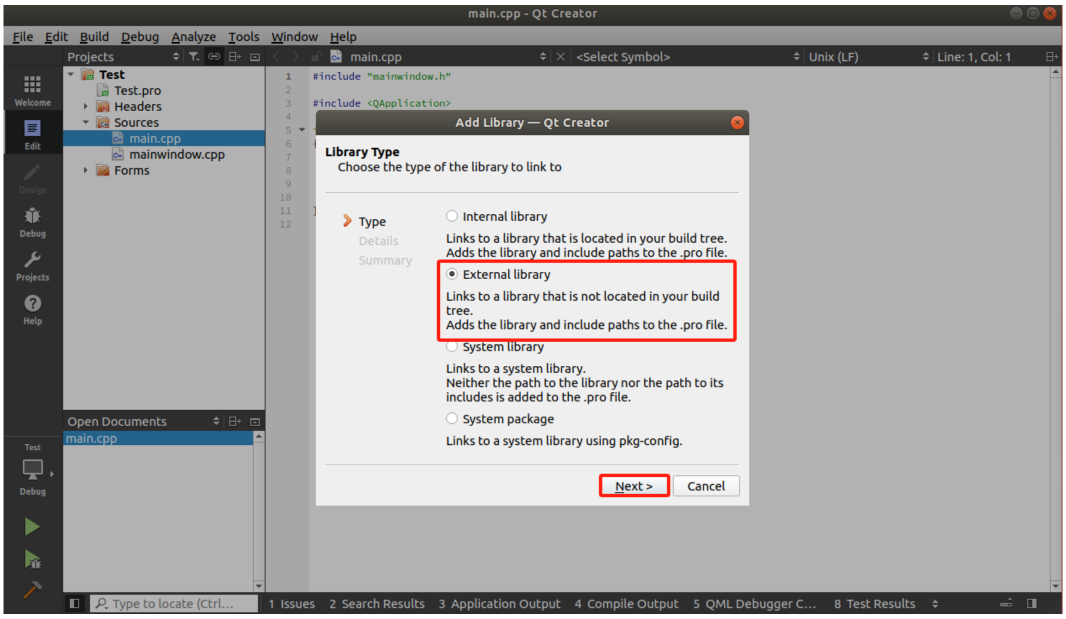
```
Test.pro
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
```

< Back Finish Cancel

17. Click Edit, right-click the Test project, and click Add Library.



18. Select External Library and click Next.



```
#include "mainwindow.h"

#include <QApplication>
```

Add Library — Qt Creator

Library Type
Choose the type of the library to link to

Type
Details
Summary

( ) Internal library
Links to a library that is located in your build tree.
Adds the library and include paths to the .pro file.

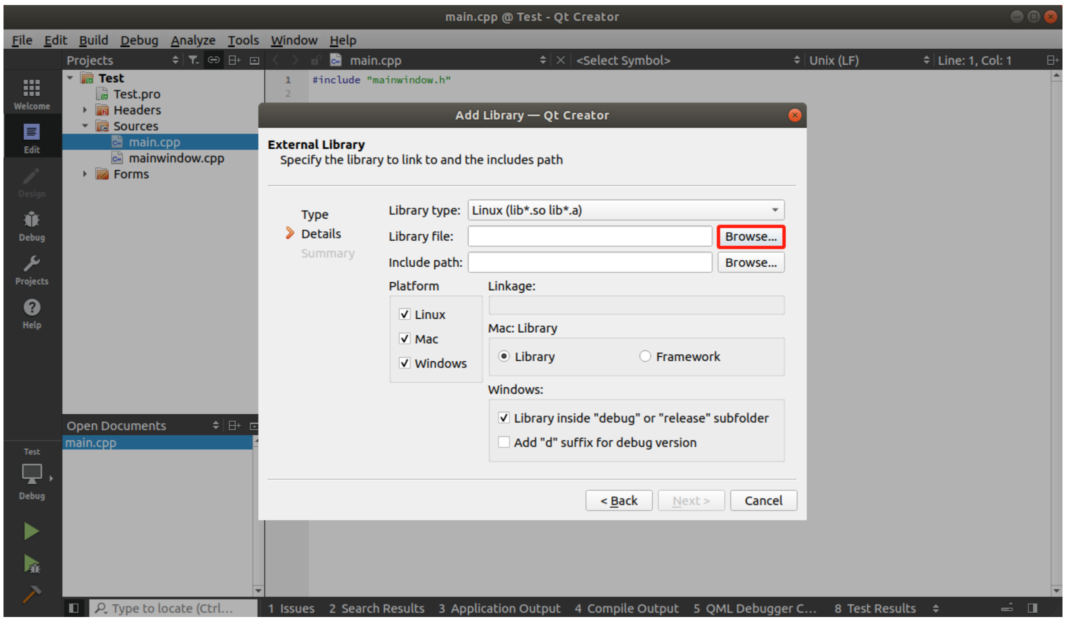
(●) External library
Links to a library that is not located in your build tree.
Adds the library and include paths to the .pro file.

( ) System library
Links to a system library.
Neither the path to the library nor the path to its includes is added to the .pro file.

( ) System package
Links to a system library using pkg-config.

Next > | Cancel

19. Click Browse Library File.


main.cpp @ Test - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
Test
Test.pro
Headers
Sources
main.cpp
mainwindow.cpp
Forms
Open Documents
main.cpp
Type to locate (Ctrl...
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger C... 8 Test Results

```
#include "mainwindow.h"
```

Add Library — Qt Creator

External Library
Specify the library to link to and the includes path

Type
Details
Summary

Library type: Linux (lib*.so lib*.a)
Library file: Browse...
Include path: Browse...

Platform
[✓] Linux
[✓] Mac
[✓] Windows

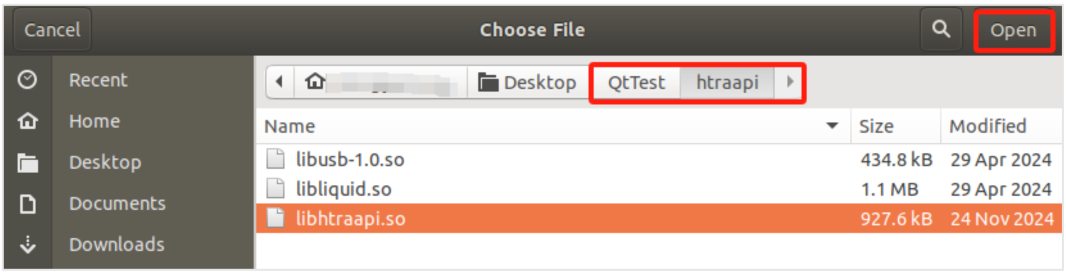
Linkage:

Mac: Library
(●) Library ( ) Framework

Windows:
[✓] Library inside "debug" or "release" subfolder
[ ] Add "d" suffix for debug version

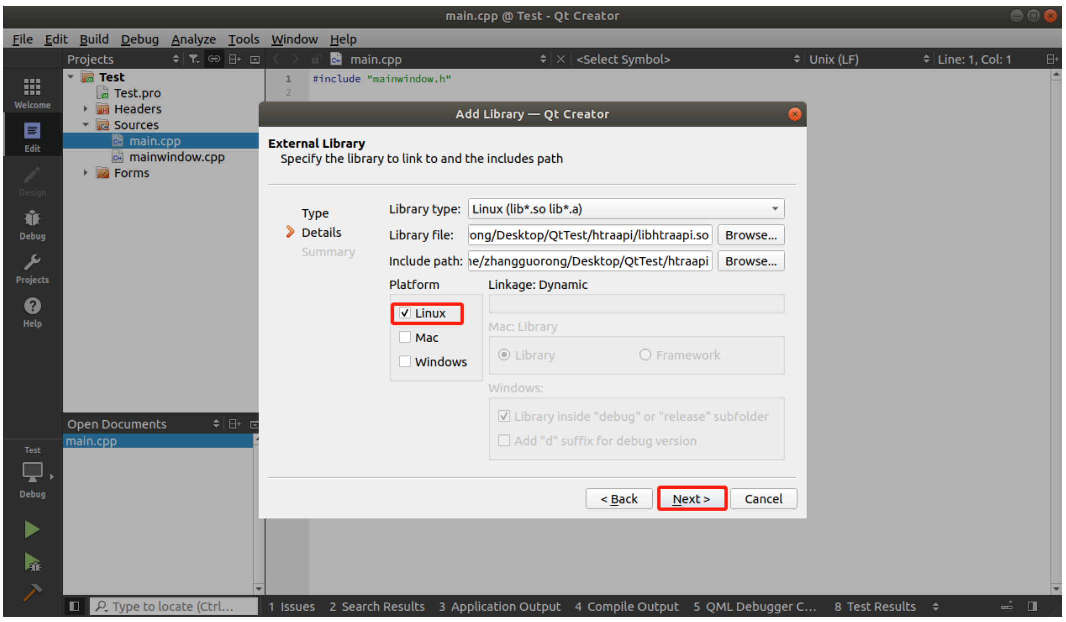
< Back | Next > | Cancel

20. Select Q Test\htraapi, the library previously copied and soft-linked, and click Open.



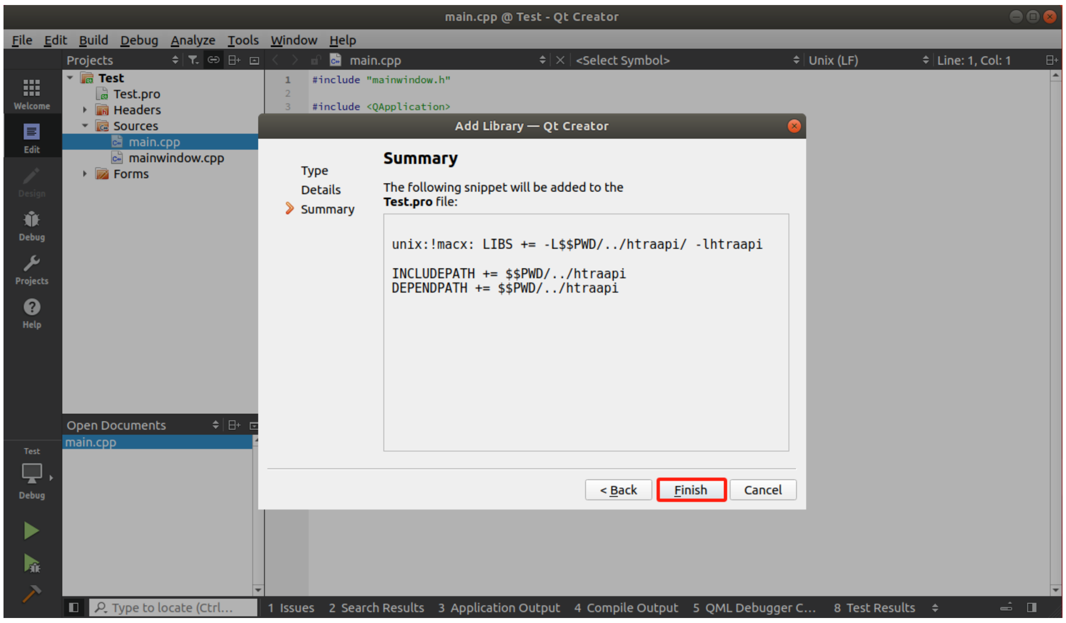
Name	Size	Modified
libusb-1.0.so	434.8 kB	29 Apr 2024
libliquid.so	1.1 MB	29 Apr 2024
libhtraapi.so	927.6 kB	24 Nov 2024

21. Select the Linux platform and click Next.



Library type: Linux (lib*.so lib*.a)
Library file: ong/Desktop/QtTest/htraapi/libhtraapi.so
Include path: ıe/zhangguorong/Desktop/QtTest/htraapi
Platform: [✓ Linux] Mac Windows
Linkage: Dynamic

22. Click Finish to add the external library.

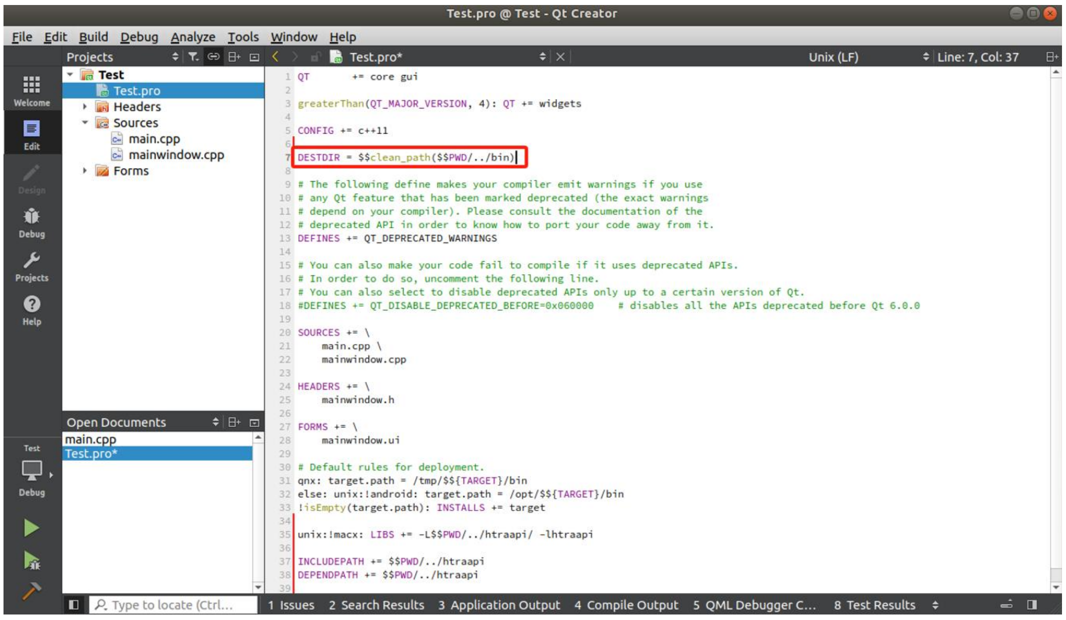


Summary

The following snippet will be added to the **Test.pro** file:

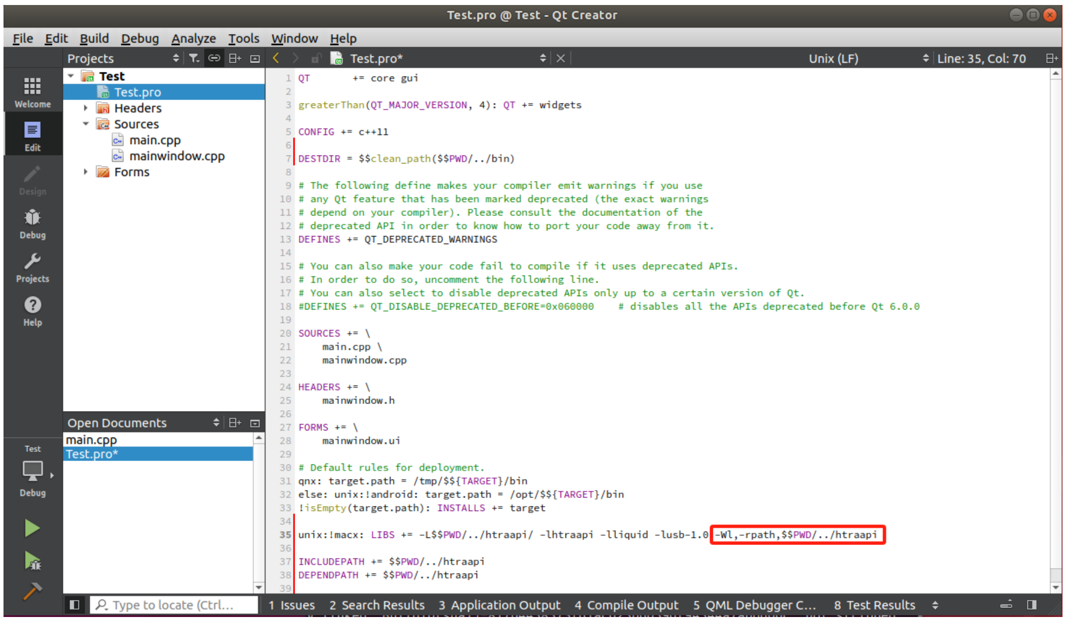
```
unix:!macx: LIBS += -L$$PWD/../htraapi/ -lhtraapi

INCLUDEPATH += $$PWD/../htraapi
DEPENDPATH += $$PWD/../htraapi
```

23. As shown below, add DESTDIR = $$clean_path($$PWD/...) after CONFIG += c++11. /bin) (this step specifies where the executable is generated)



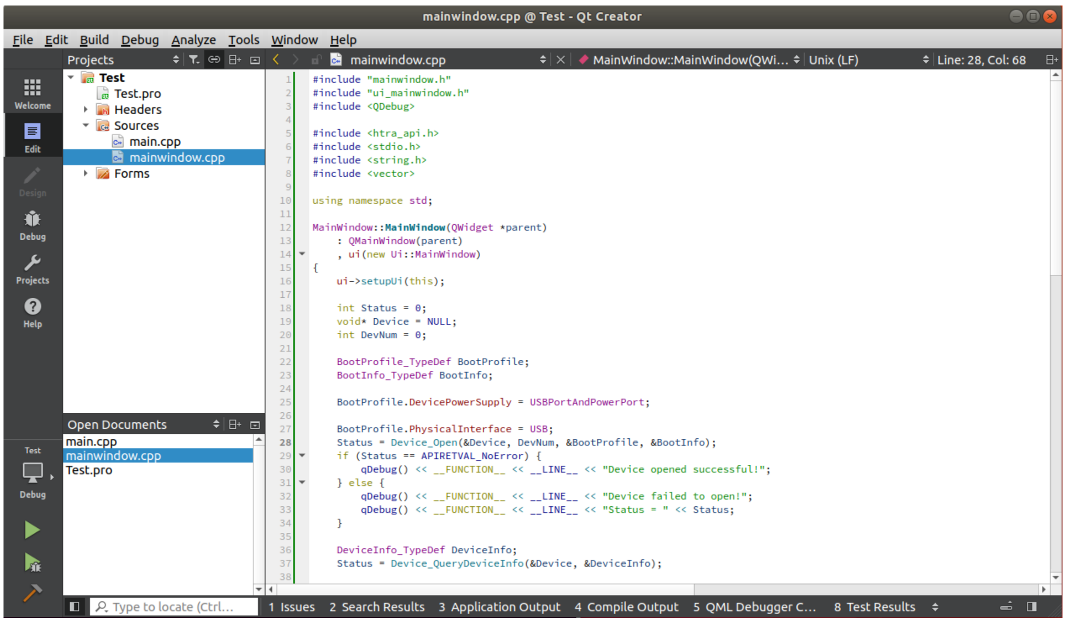
```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

DESTDIR = $$clean_path($$PWD/../bin)

# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

unix:!macx: LIBS += -L$$PWD/../htraapi/ -lhtraapi

INCLUDEPATH += $$PWD/../htraapi
DEPENDPATH += $$PWD/../htraapi
```

24. 24. As shown in the figure below, after the library file, add -Wl,-rpath,$$PWD/... /htraapi (this step specifies the library path for the executable).



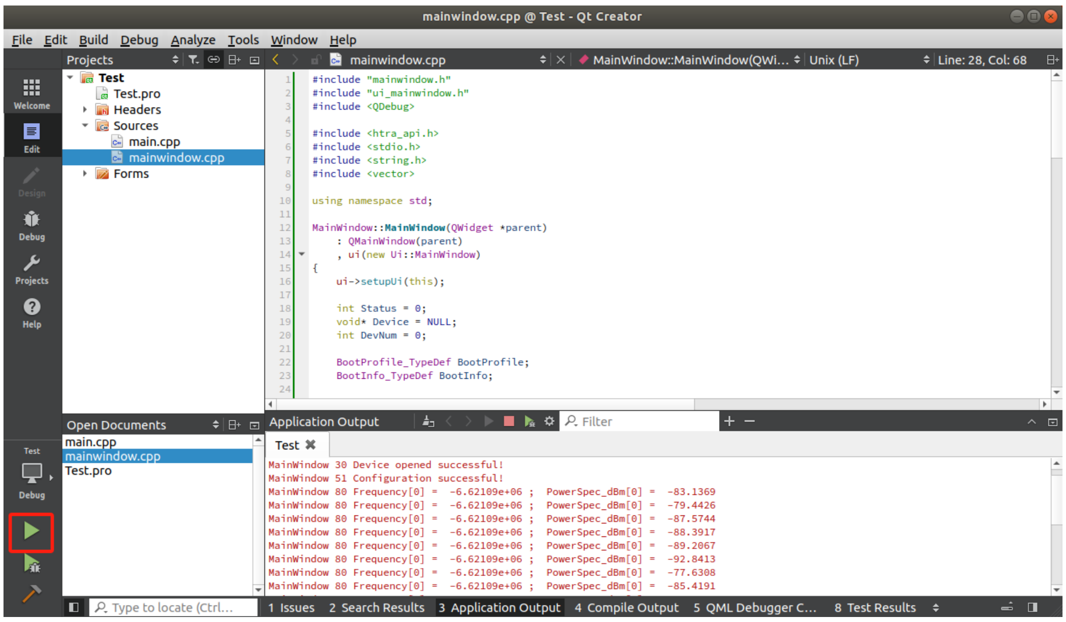
```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

DESTDIR = $$clean_path($$PWD/../bin)

# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

unix:!macx: LIBS += -L$$PWD/../htraapi/ -lhtraapi -lliquid -lusb-1.0 -Wl,-rpath,$$PWD/../htraapi

INCLUDEPATH += $$PWD/../htraapi
DEPENDPATH += $$PWD/../htraapi
```

25. Save the Test.pro file and write the code in mainwindow.cpp afterwards.



```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>

#include <htra_api.h>
#include <stdio.h>
#include <string.h>
#include <vector>

using namespace std;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    int Status = 0;
    void* Device = NULL;
    int DevNum = 0;

    BootProfile_TypeDef BootProfile;
    BootInfo_TypeDef BootInfo;

    BootProfile.DevicePowerSupply = USBPortAndPowerPort;

    BootProfile.PhysicalInterface = USB;
    Status = Device_Open(&Device, DevNum, &BootProfile, &BootInfo);
    if (Status == APIRETVAL_NoError) {
        qDebug() << __FUNCTION__ << __LINE__ << "Device opened successful!";
    } else {
        qDebug() << __FUNCTION__ << __LINE__ << "Device failed to open!";
        qDebug() << __FUNCTION__ << __LINE__ << "Status = " << Status;
    }

    DeviceInfo_TypeDef DeviceInfo;
    Status = Device_QueryDeviceInfo(&Device, &DeviceInfo);

```

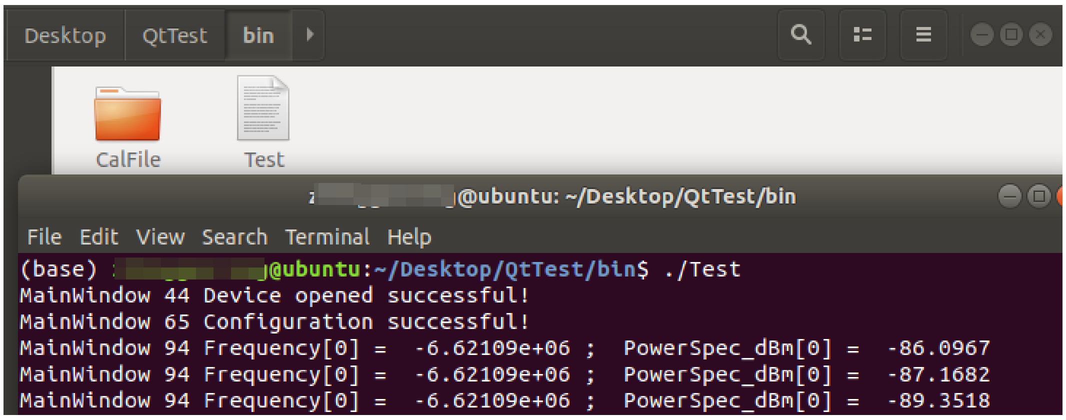
26. After writing the code correctly, click run. As shown in the figure, you can see that the program is running normally in the application output.



```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>

#include <htra_api.h>
#include <stdio.h>
#include <string.h>
#include <vector>

using namespace std;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    int Status = 0;
    void* Device = NULL;
    int DevNum = 0;

    BootProfile_TypeDef BootProfile;
    BootInfo_TypeDef BootInfo;

```

```
MainWindow 30 Device opened successful!
MainWindow 51 Configuration successful!
MainWindow 80 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -83.1369
MainWindow 80 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -79.4426
MainWindow 80 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -87.5744
MainWindow 80 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -88.3917
MainWindow 80 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -89.2067
MainWindow 80 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -92.8413
MainWindow 80 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -77.6308
MainWindow 80 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -85.4191
```

27. Close Qt Creator, enter the QtTest\bin folder, open the terminal, and type ./Test to run the executable program.


Desktop
QtTest
bin
CalFile
Test
[redacted]@ubuntu: ~/Desktop/QtTest/bin
File Edit View Search Terminal Help

```
(base) [redacted]@ubuntu:~/Desktop/QtTest/bin$ ./Test
MainWindow 44 Device opened successful!
MainWindow 65 Configuration successful!
MainWindow 94 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -86.0967
MainWindow 94 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -87.1682
MainWindow 94 Frequency[0] =  -6.62109e+06 ;  PowerSpec_dBm[0] =  -89.3518
```

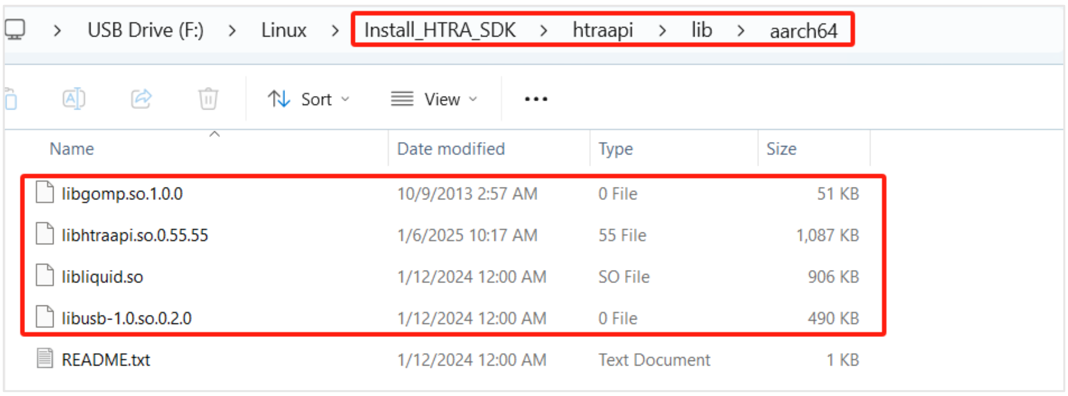
9.5.3 Cross-compiling Qt projects

Under the premise that the host computer has a cross-compilation toolchain, if you want to cross-compile to use the device, please refer to the following process (taking the cross-compilation of an aarch64 executable program on an x86_64 host computer as an example):

1. First, generate the executable file for the target architecture:

1) Create the project and place the calibration files and header files according to steps 1 - 4 of the QtProject Creation and Compilation.

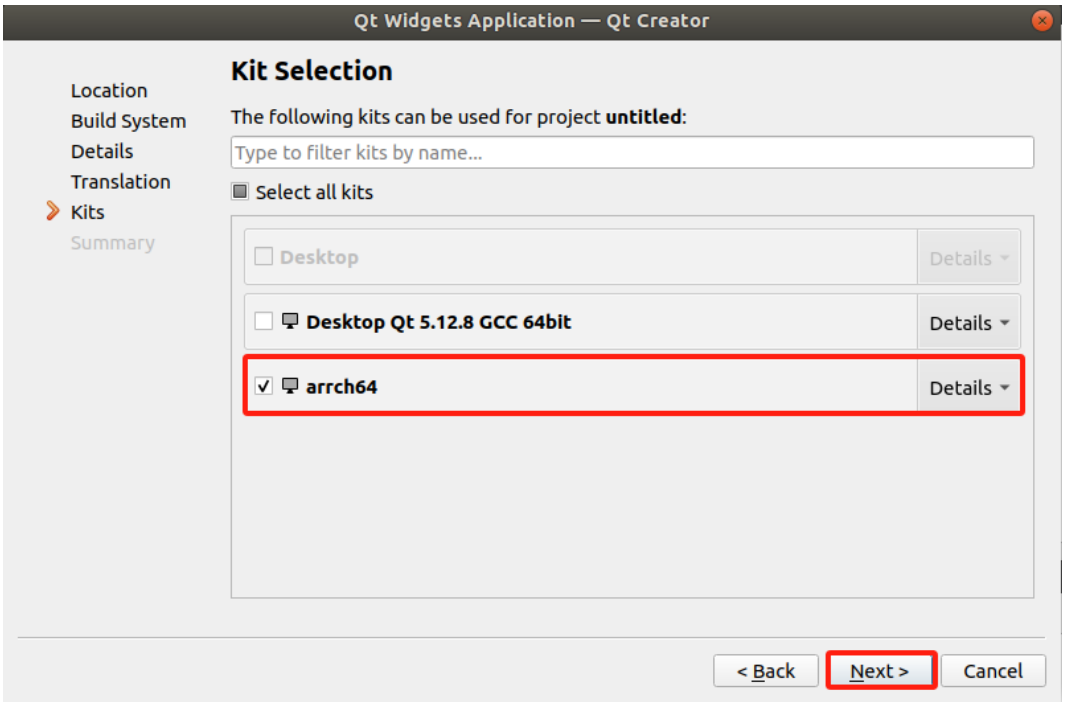
2) Place the cross-compilation target architecture library files according to step 5 of the form program creation process. For example, if you expect to cross-compile an executable for the arrch64 architecture, please place the library files for the arrch64 architecture.


USB Drive (F:) > Linux > Install_HTRA_SDK > htraapi > lib > aarch64
Sort
View

Name	Date modified	Type	Size
libgomp.so.1.0.0	10/9/2013 2:57 AM	0 File	51 KB
libhtraapi.so.0.55.55	1/6/2025 10:17 AM	55 File	1,087 KB
libliquid.so	1/12/2024 12:00 AM	SO File	906 KB
libusb-1.0.so.0.2.0	1/12/2024 12:00 AM	0 File	490 KB
README.txt	1/12/2024 12:00 AM	Text Document	1 KB

3) Create a soft link for the dynamic link library according to step 6 of the form program creation process.

4) Create the program according to steps 7-13 of the form program creation process, and in step 14, select the build kit for the cross-compilation target architecture (for example, in this example, select the build kit for the arrch64 architecture).



Kit Selection

The following kits can be used for project **untitled**:

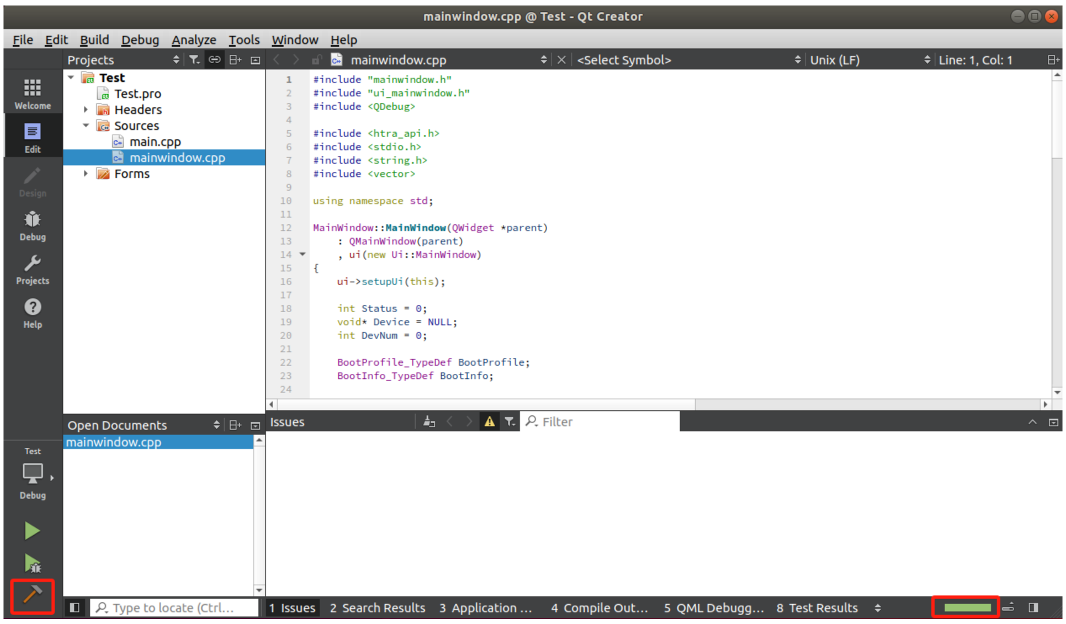
Type to filter kits by name...

Select all kits

- [ ] Desktop — Details
- [ ] Desktop Qt 5.12.8 GCC 64bit — Details
- [✓] arrch64 — Details

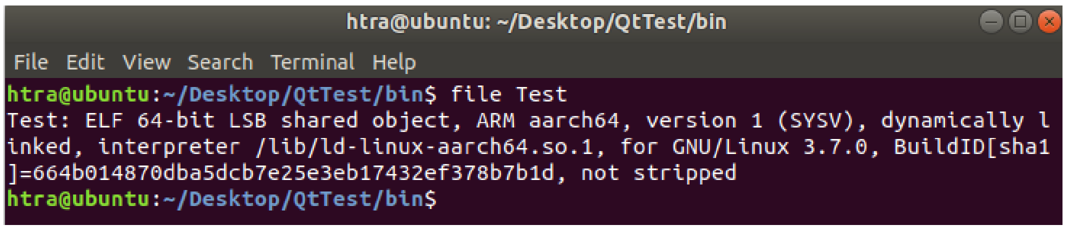
5) Follow steps 16-26 of the form program creation process to create the project, reference the libraries, modify the location for generating the executable program, and write the project.

6) Click the build button in the lower left corner to build the executable program.



```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>

#include <htra_api.h>
#include <stdio.h>
#include <string.h>
#include <vector>

using namespace std;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    int Status = 0;
    void* Device = NULL;
    int DevNum = 0;

    BootProfile_TypeDef BootProfile;
    BootInfo_TypeDef BootInfo;

```

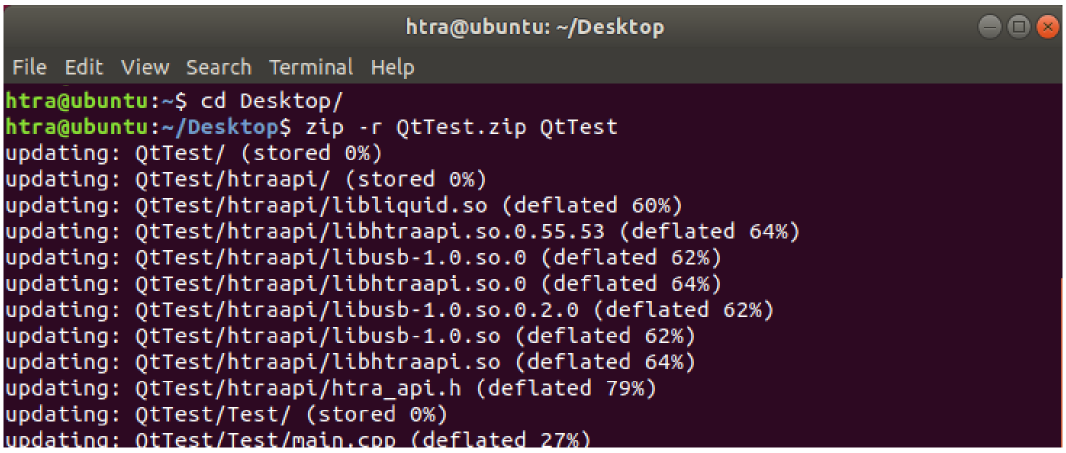
7) After building the executable program, open the terminal in the QtTest\bin folder and type file Test to check the architecture of the executable program (the name of the executable program here is Test, so type file Test).



```
htra@ubuntu:~/Desktop/QtTest/bin$ file Test
Test: ELF 64-bit LSB shared object, ARM aarch64, version 1 (SYSV), dynamically l
inked, interpreter /lib/ld-linux-aarch64.so.1, for GNU/Linux 3.7.0, BuildID[sha1
]=664b014870dba5dcb7e25e3eb17432ef378b7b1d, not stripped
htra@ubuntu:~/Desktop/QtTest/bin$
```

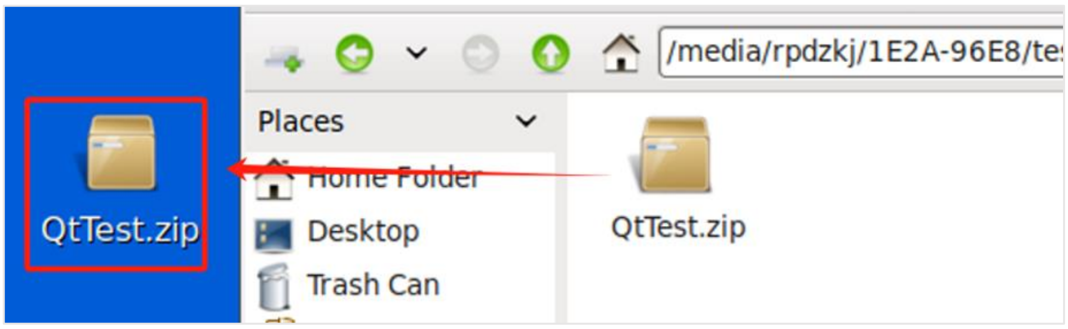
2. At this point, the executable program has been successfully generated, and you can now run the program on the aarch64 host machine using the device:

1) Navigate to the project location (in this example, the project is located on the desktop, so type cd Desktop/), and compress the entire folder into a zip file, for example, type zip -r Qt Test.zip Qt Test to create a zip archive.

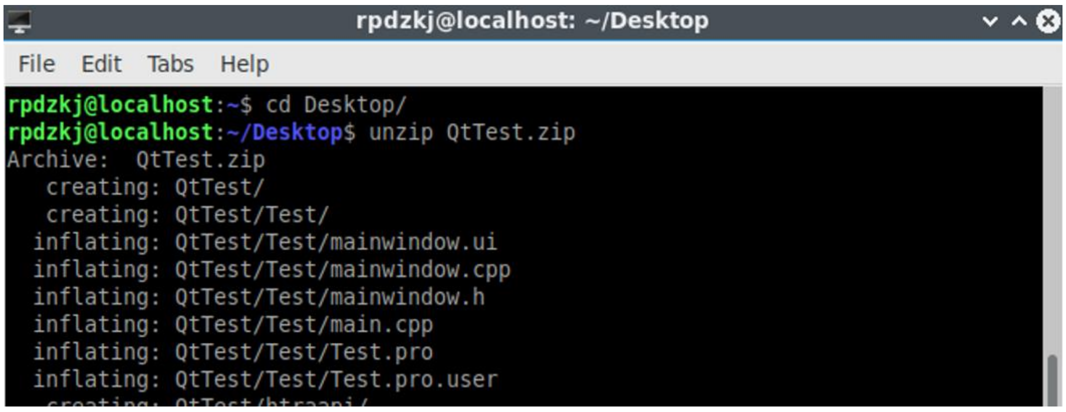


```
htra@ubuntu:~$ cd Desktop/
htra@ubuntu:~/Desktop$ zip -r QtTest.zip QtTest
updating: QtTest/ (stored 0%)
updating: QtTest/htraapi/ (stored 0%)
updating: QtTest/htraapi/libliquid.so (deflated 60%)
updating: QtTest/htraapi/libhtraapi.so.0.55.53 (deflated 64%)
updating: QtTest/htraapi/libusb-1.0.so.0 (deflated 62%)
updating: QtTest/htraapi/libhtraapi.so.0 (deflated 64%)
updating: QtTest/htraapi/libusb-1.0.so.0.2.0 (deflated 62%)
updating: QtTest/htraapi/libusb-1.0.so (deflated 62%)
updating: QtTest/htraapi/libhtraapi.so (deflated 64%)
updating: QtTest/htraapi/htra_api.h (deflated 79%)
updating: QtTest/Test/ (stored 0%)
updating: QtTest/Test/main.cpp (deflated 27%)
```

2) Copy the zip file to the aarch64 host machine.



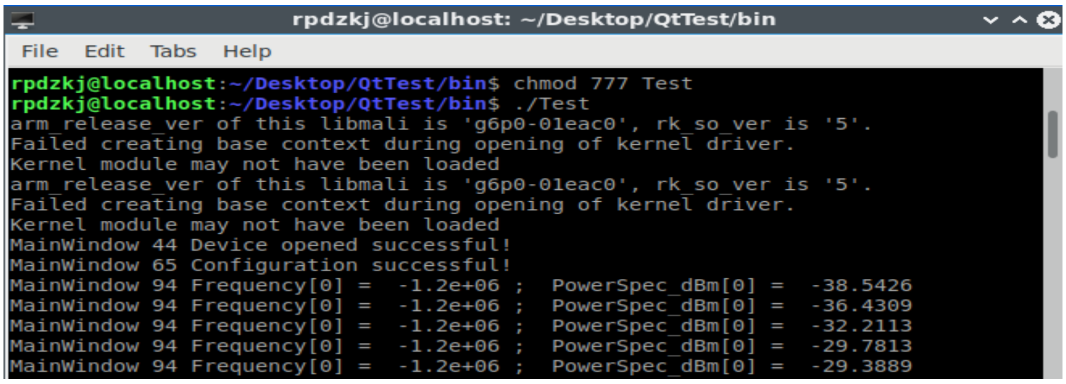
3) Navigate to the location of the compressed package (in this example, the compressed package is located on the desktop, so input cd Desktop/), and unzip the project (input unzip QtTest here).



```
rpdzkj@localhost:~$ cd Desktop/
rpdzkj@localhost:~/Desktop$ unzip QtTest.zip
Archive:  QtTest.zip
   creating: QtTest/
   creating: QtTest/Test/
  inflating: QtTest/Test/mainwindow.ui
  inflating: QtTest/Test/mainwindow.cpp
  inflating: QtTest/Test/mainwindow.h
  inflating: QtTest/Test/main.cpp
  inflating: QtTest/Test/Test.pro
  inflating: QtTest/Test/Test.pro.user
```

4) Configure the driver files on the aarch64 host machine as per the steps in Chapter9.3.

5) After configuring the driver files, input cd QtTest/ to enter the folder, then input chmod 777 Test to provide execution permissions for the program, and finally input ./Test to run the program.



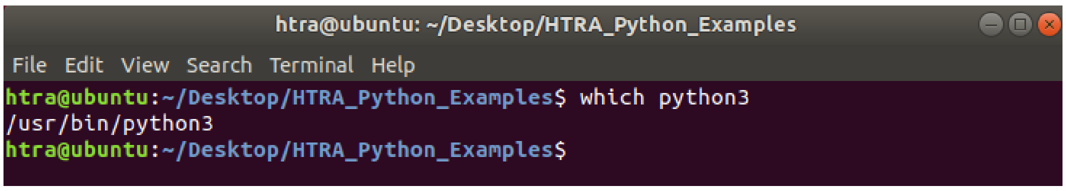
```
rpdzkj@localhost:~/Desktop/QtTest/bin$ chmod 777 Test
rpdzkj@localhost:~/Desktop/QtTest/bin$ ./Test
arm_release_ver of this libmali is 'g6p0-01eac0', rk_so_ver is '5'.
Failed creating base context during opening of kernel driver.
Kernel module may not have been loaded
arm_release_ver of this libmali is 'g6p0-01eac0', rk_so_ver is '5'.
Failed creating base context during opening of kernel driver.
Kernel module may not have been loaded
MainWindow 44 Device opened successful!
MainWindow 65 Configuration successful!
MainWindow 94 Frequency[0] =  -1.2e+06 ;  PowerSpec_dBm[0] =  -38.5426
MainWindow 94 Frequency[0] =  -1.2e+06 ;  PowerSpec_dBm[0] =  -36.4309
MainWindow 94 Frequency[0] =  -1.2e+06 ;  PowerSpec_dBm[0] =  -32.2113
MainWindow 94 Frequency[0] =  -1.2e+06 ;  PowerSpec_dBm[0] =  -29.7813
MainWindow 94 Frequency[0] =  -1.2e+06 ;  PowerSpec_dBm[0] =  -29.3889
```

9.6 Usage of Python examples and project creation

9.6.1 Usage of Python examples

Assuming the device is properly connected and the driver files have been configured correctly as per Chapter 9.3, if you want to use the Python examples included on the USB drive, you can refer to the following process (the specific functions of the Python examples can be directly viewed in Chapter 4.2):

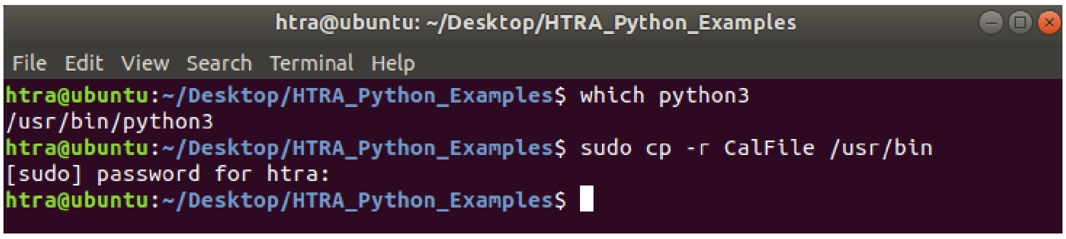
1. Copy the Linux\ HTRA_ Python_ Examples folder from the included USB drive to the host computer, then open a terminal in the HTRA_ Python_ Examples folder and input which python3 to check the location of the Python interpreter (for example, it may be located at /usr/bin).



htra@ubuntu: ~/Desktop/HTRA_Python_Examples
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/HTRA_Python_Examples$ which python3
/usr/bin/python3
htra@ubuntu:~/Desktop/HTRA_Python_Examples$
```

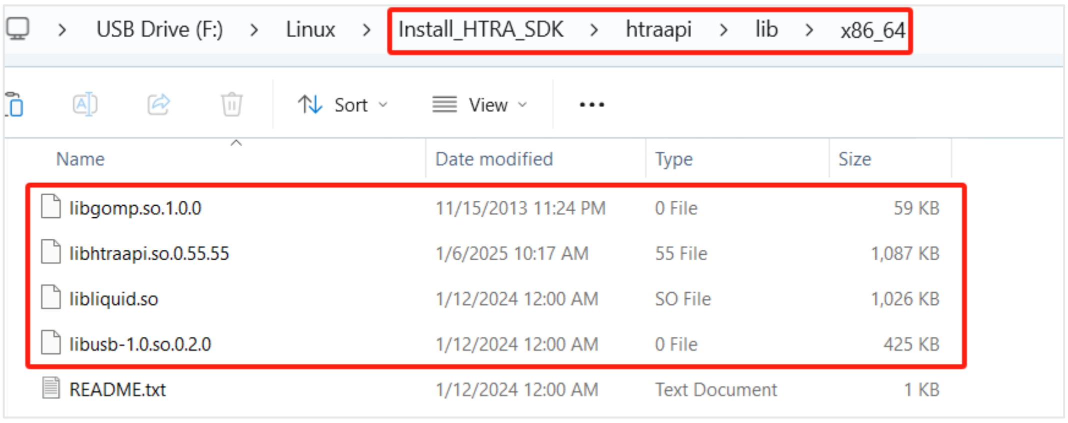
2. Based on the interpreter address obtained earlier, input sudo cp -r CalFile /usr/bin to copy the device calibration file to the same directory as the interpreter (Python3) (for example, in this case, the same directory is /usr/bin; if the actual interpreter location is different, modify the address accordingly).



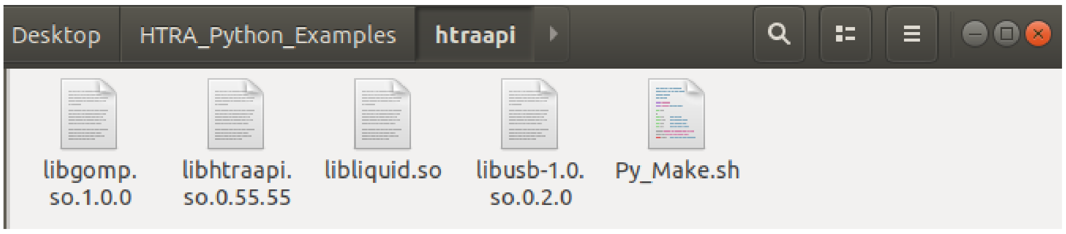
htra@ubuntu: ~/Desktop/HTRA_Python_Examples
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/HTRA_Python_Examples$ which python3
/usr/bin/python3
htra@ubuntu:~/Desktop/HTRA_Python_Examples$ sudo cp -r CalFile /usr/bin
[sudo] password for htra:
htra@ubuntu:~/Desktop/HTRA_Python_Examples$ 
```

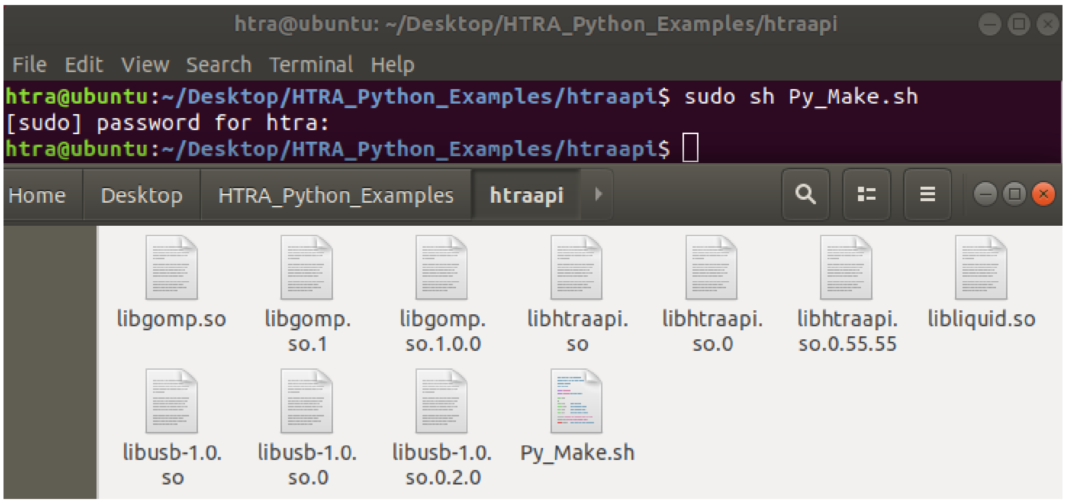
3. First, refer to the system architecture in Chapter9.1, then according to Chapter9.2.4, copy the dynamic link libraries from the corresponding system architecture folder under the included USB drive Linux\Install_HTRA_SDK\htraapi\lib to the HTRA_Python_Examples\htraapi folder (this example uses the x86_64 architecture host computer).


USB Drive (F:) > Linux > Install_HTRA_SDK > htraapi > lib > x86_64
Sort
View

Name	Date modified	Type	Size
libgomp.so.1.0.0	11/15/2013 11:24 PM	0 File	59 KB
libhtraapi.so.0.55.55	1/6/2025 10:17 AM	55 File	1,087 KB
libliquid.so	1/12/2024 12:00 AM	SO File	1,026 KB
libusb-1.0.so.0.2.0	1/12/2024 12:00 AM	0 File	425 KB
README.txt	1/12/2024 12:00 AM	Text Document	1 KB

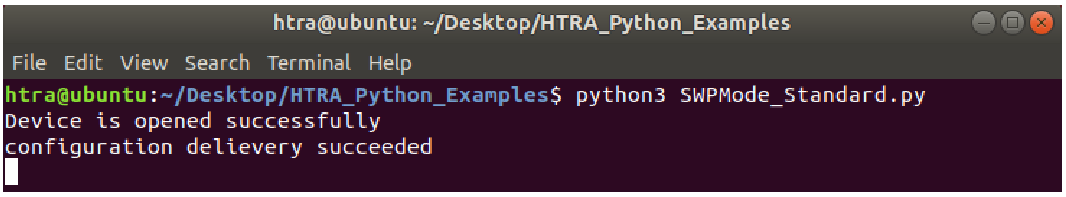


4. Open a terminal in the current folder and input “sudo sh Py_Make.sh”, then follow the prompts to enter the sudo password to provide permissions for creating soft links to the libraries.


htra@ubuntu: ~/Desktop/HTRA_Python_Examples/htraapi
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/HTRA_Python_Examples/htraapi$ sudo sh Py_Make.sh
[sudo] password for htra:
htra@ubuntu:~/Desktop/HTRA_Python_Examples/htraapi$
```

5. In the HTRA_ Python_ Examples location, open a terminal and input python3 SWPMode_Standard.py to run the example. (This example uses SWPMode_Standard.py as an example)


htra@ubuntu: ~/Desktop/HTRA_Python_Examples
File Edit View Search Terminal Help

```
htra@ubuntu:~/Desktop/HTRA_Python_Examples$ python3 SWPMode_Standard.py
Device is opened successfully
configuration delievery succeeded
```

9.6.2 Python Project Creation

Under the premise that the driver files have been correctly configured according to Chapter9.3, if you want to create and write a Python project, please refer to the following process:

First, when writing code, since the Linux dynamic link library provided with the USB drive is identical to that in Windows, the code only needs to comply with the API programming guidelines.

Secondly, when running the program, simply place the program in the example folder and follow the process outlined in Python Example Usage.

9.7 Gnuradio Module Construction and Use

After the device is properly connected and the driver files are configured as described in Section 9.3, you can use the device with GNU Radio in either of the following two ways.

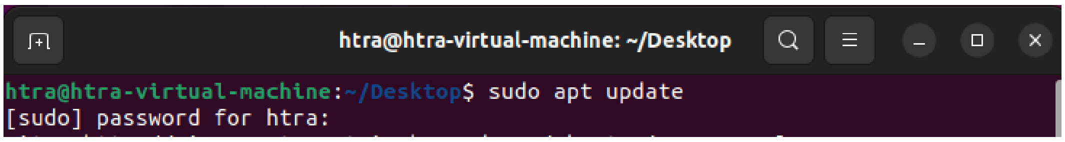
9.7.1 HTRA OOT build use

There are two ways to obtain the HTRA OOT module:

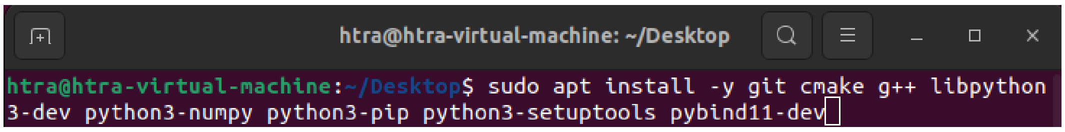
1) Retrieve it directly from the Linux\HTRA_Gnuradio_Examples folder included with the accompanying materials;
2) Pull it from GitHub using the following repository URL: https://github.com/HAROGIC-Technologies/gr-htra

1. Configure the dependency environment: Enter the following command to download the dependency environment required for the OOT module build:

- sudo apt update
- sudo apt install -y git cmake g++ libpython3-dev python3-numpy python3-pip python3-setuptools pybind11-dev



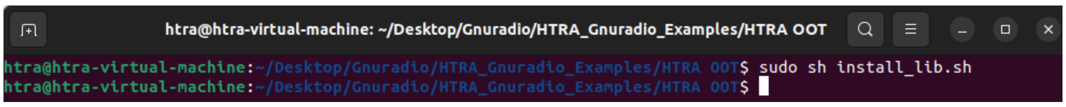
```
htra@htra-virtual-machine:~/Desktop$ sudo apt update
[sudo] password for htra:
```



```
htra@htra-virtual-machine:~/Desktop$ sudo apt install -y git cmake g++ libpython
3-dev python3-numpy python3-pip python3-setuptools pybind11-dev
```

2. Configure the libraries and headers required for the build: open a terminal in the HTRA OOT folder and enter:

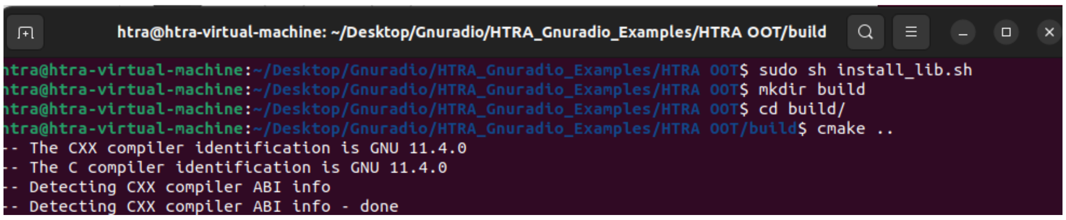
- sudo sh install_lib.sh


htra@htra-virtual-machine: ~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT

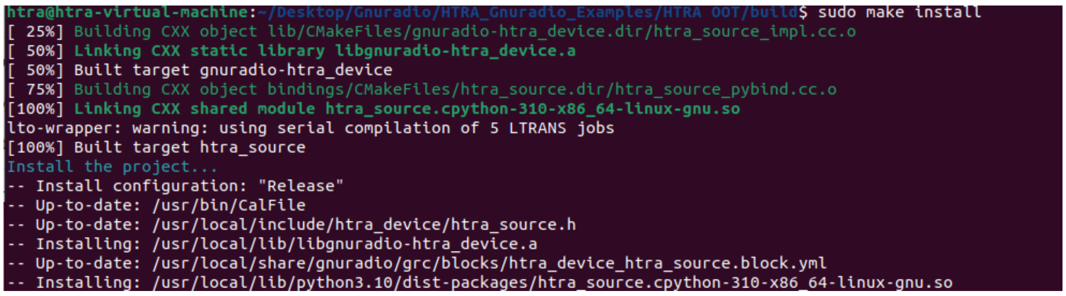
```
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT$ sudo sh install_lib.sh
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT$
```

3. OOT module construction: enter the following commands in sequence:

- mkdir build
- cd build
- cmake ..
- sudo make install

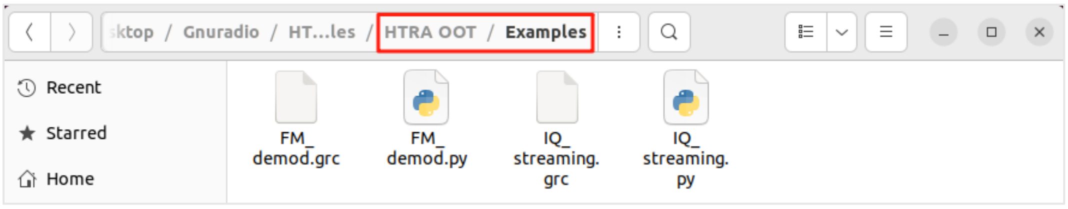

htra@htra-virtual-machine: ~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT/build

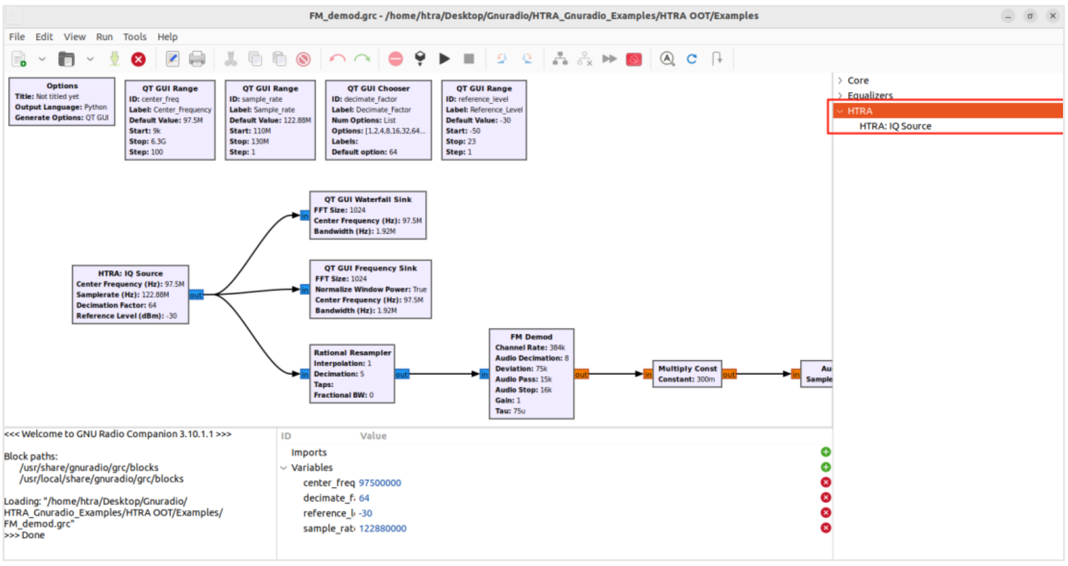
```
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT$ sudo sh install_lib.sh
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT$ mkdir build
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT$ cd build/
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT/build$ cmake ..
-- The CXX compiler identification is GNU 11.4.0
-- The C compiler identification is GNU 11.4.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
```



```
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT/build$ sudo make install
[ 25%] Building CXX object lib/CMakeFiles/gnuradio-htra_device.dir/htra_source_impl.cc.o
[ 50%] Linking CXX static library libgnuradio-htra_device.a
[ 50%] Built target gnuradio-htra_device
[ 75%] Building CXX object bindings/CMakeFiles/htra_source.dir/htra_source_pybind.cc.o
[100%] Linking CXX shared module htra_source.cpython-310-x86_64-linux-gnu.so
lto-wrapper: warning: using serial compilation of 5 LTRANS jobs
[100%] Built target htra_source
Install the project...
-- Install configuration: "Release"
-- Up-to-date: /usr/bin/CalFile
-- Up-to-date: /usr/local/include/htra_device/htra_source.h
-- Installing: /usr/local/lib/libgnuradio-htra_device.a
-- Up-to-date: /usr/local/share/gnuradio/grc/blocks/htra_device_htra_source.block.yml
-- Installing: /usr/local/lib/python3.10/dist-packages/htra_source.cpython-310-x86_64-linux-gnu.so
```

4. Using the device: Once built, you can use the device in Gnuradio by referring to the examples in the Examples folder (here FM demodulation is used as an example):




FM_demod.grc - /home/htra/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA OOT/Examples
File Edit View Run Tools Help
Core
Equalizers
HTRA
HTRA: IQ Source

Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI

QT GUI Range
ID: center_freq
Label: Center_Frequency
Default Value: 97.5M
Start: 9k
Stop: 6.3G
Step: 100

QT GUI Range
ID: sample_rate
Label: Sample_rate
Default Value: 122.88M
Start: 110M
Stop: 130M
Step: 1

QT GUI Chooser
ID: decimate_factor
Label: Decimate_Factor
Num Options: List
Options: [1,2,4,8,16,32,64...
Labels:
Default option: 64

QT GUI Range
ID: reference_level
Label: Reference_Level
Default Value: -30
Start: -50
Stop: 23
Step: 1

HTRA: IQ Source
Center Frequency (Hz): 97.5M
Samplerate (Hz): 122.88M
Decimation Factor: 64
Reference Level (dBm): -30

QT GUI Waterfall Sink
FFT Size: 1024
Center Frequency (Hz): 97.5M
Bandwidth (Hz): 1.92M

QT GUI Frequency Sink
FFT Size: 1024
Normalize Window Power: True
Center Frequency (Hz): 97.5M
Bandwidth (Hz): 1.92M

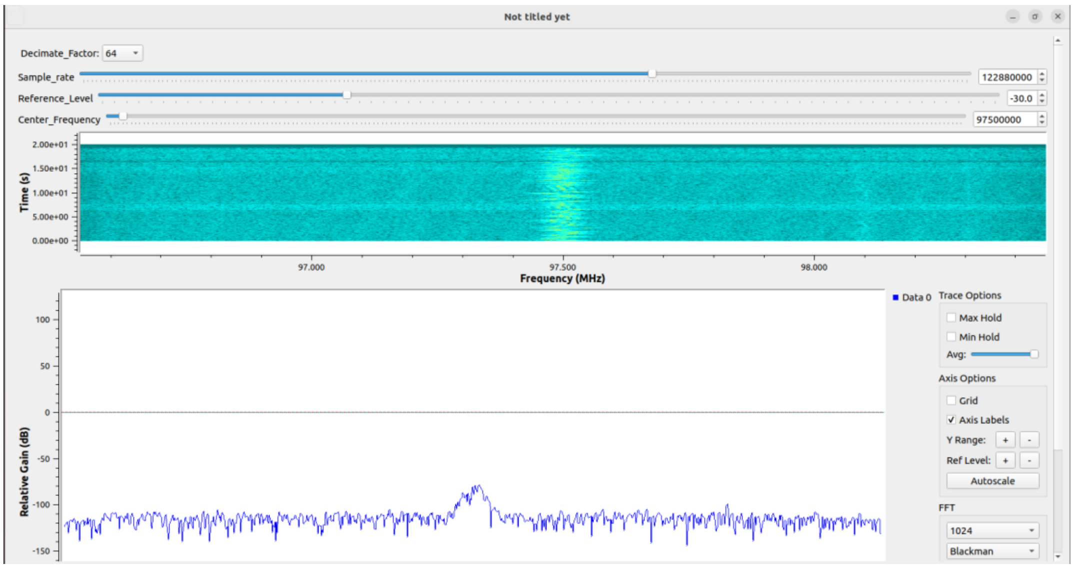
Rational Resampler
Interpolation: 1
Decimation: 5
Taps:
Fractional BW: 0

FM Demod
Channel Rate: 384k
Audio Decimation: 8
Deviation: 75k
Audio Pass: 15k
Audio Stop: 16k
Gain: 1
Tau: 75u

Multiply Const
Constant: 300m

```
<<< Welcome to GNU Radio Companion 3.10.1.1 >>>

Block paths:
    /usr/share/gnuradio/grc/blocks
    /usr/local/share/gnuradio/grc/blocks

Loading: "/home/htra/Desktop/Gnuradio/
HTRA_Gnuradio_Examples/HTRA OOT/Examples/
FM_demod.grc"
>>> Done
```

ID	Value
Imports	
Variables	
center_freq	97500000
decimate_f	64
reference_l	-30
sample_rat	122880000



Decimate_Factor: 64
Sample_rate: 122880000
Reference_Level: -30.0
Center_Frequency: 97500000
FFT: 1024, Blackman

9.7.2 SoapySDR Build Usage

SoapySDR adaptations are obtained in two ways:

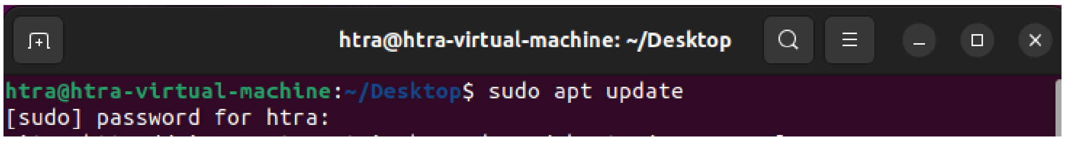
1. Directly from the Linux\HTRA_Gnuradio_Examples folder of the accompanying material;
2. Pull it through github (recommended), the git address is as follows:

https://github.com/HAROGIC-Technologies/soapy-htra

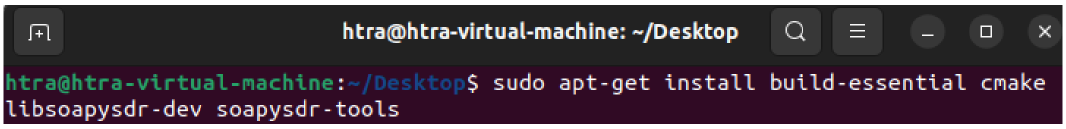
If you are obtaining the information by sending it along, you can refer to the following process to use it:

1. Configure dependencies: Enter the following commands to download the dependencies required for building the SoapySDR module:

- sudo apt-get update
- sudo apt-get install build-essential cmake libsoapysdr-dev soapysdr-tools



```
htra@htra-virtual-machine:~/Desktop$ sudo apt update
[sudo] password for htra:
```



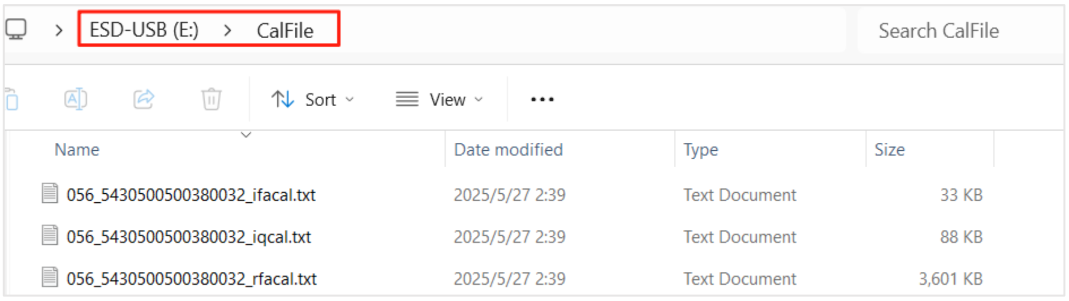
```
htra@htra-virtual-machine:~/Desktop$ sudo apt-get install build-essential cmake
libsoapysdr-dev soapysdr-tools
```

2. Copy the calibration files: Copy the files in the CalFile folder of the USB stick included in the shipment to the usr/bin/CalFile folder:

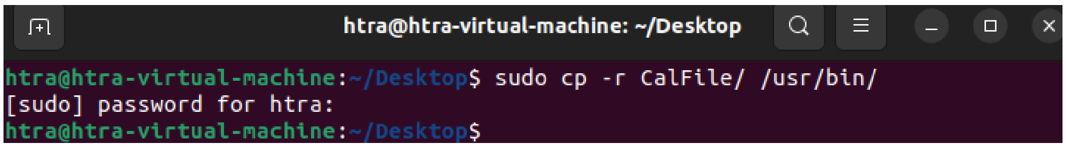
- sudo cp -r CalFile/ /usr/bin/

3. After copying, you can view the copy via:

- ls /usr/bin/CalFile/



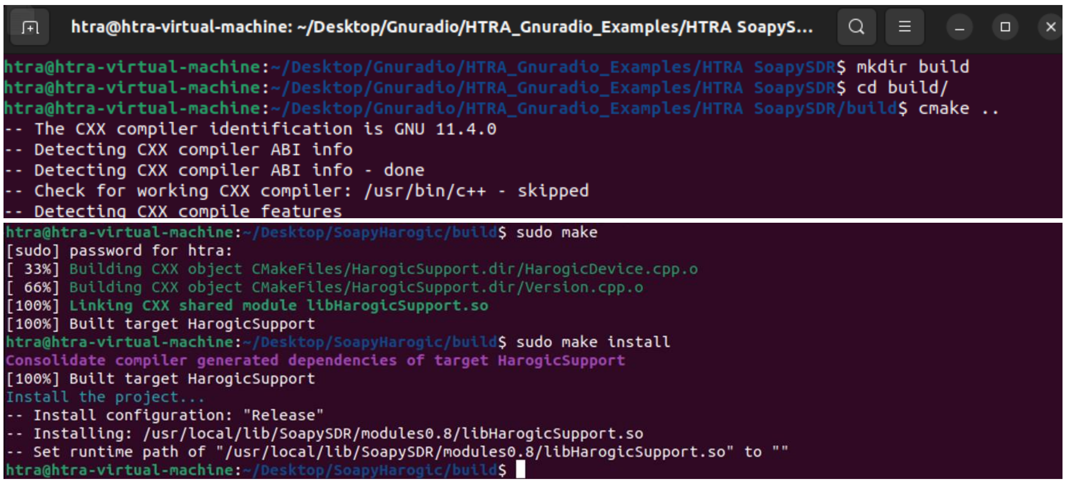
Name	Date modified	Type	Size
056_5430500500380032_ifacal.txt	2025/5/27 2:39	Text Document	33 KB
056_5430500500380032_iqcal.txt	2025/5/27 2:39	Text Document	88 KB
056_5430500500380032_rfacal.txt	2025/5/27 2:39	Text Document	3,601 KB



```
htra@htra-virtual-machine:~/Desktop$ sudo cp -r CalFile/ /usr/bin/
[sudo] password for htra:
htra@htra-virtual-machine:~/Desktop$
```

4. SoapySDR module construction: enter the following commands in sequence:

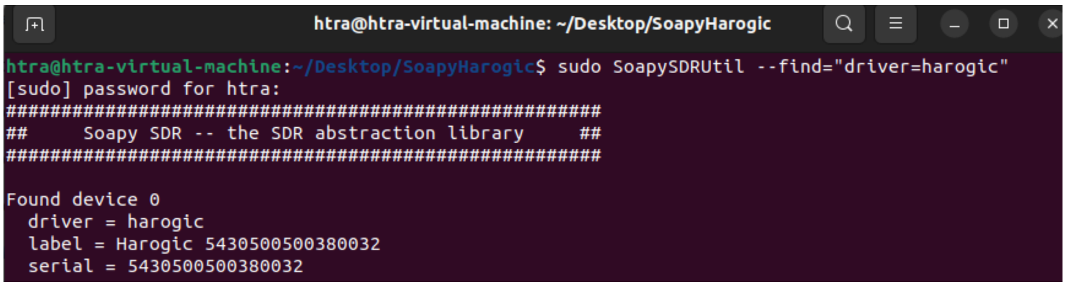
- mkdir build:
- cd build
- cmake ..
- sudo make
- sudo make install



```
htra@htra-virtual-machine: ~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA SoapyS...
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA SoapySDR$ mkdir build
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA SoapySDR$ cd build/
htra@htra-virtual-machine:~/Desktop/Gnuradio/HTRA_Gnuradio_Examples/HTRA SoapySDR/build$ cmake ..
-- The CXX compiler identification is GNU 11.4.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
```

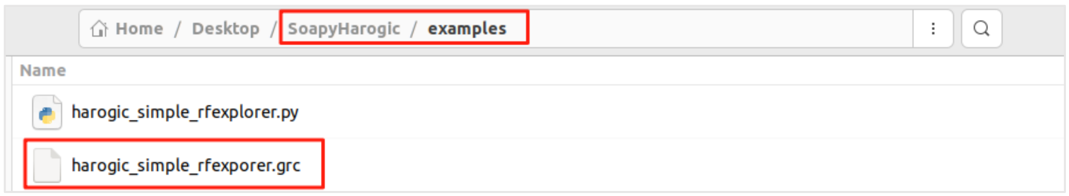
```
htra@htra-virtual-machine:~/Desktop/SoapyHarogic/build$ sudo make
[sudo] password for htra:
[ 33%] Building CXX object CMakeFiles/HarogicSupport.dir/HarogicDevice.cpp.o
[ 66%] Building CXX object CMakeFiles/HarogicSupport.dir/Version.cpp.o
[100%] Linking CXX shared module libHarogicSupport.so
[100%] Built target HarogicSupport
htra@htra-virtual-machine:~/Desktop/SoapyHarogic/build$ sudo make install
Consolidate compiler generated dependencies of target HarogicSupport
[100%] Built target HarogicSupport
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/lib/SoapySDR/modules0.8/libHarogicSupport.so
-- Set runtime path of "/usr/local/lib/SoapySDR/modules0.8/libHarogicSupport.so" to ""
htra@htra-virtual-machine:~/Desktop/SoapyHarogic/build$
```

5. After building, type sudo SoapySDRUtil --find="driver=harogic" to find the device:

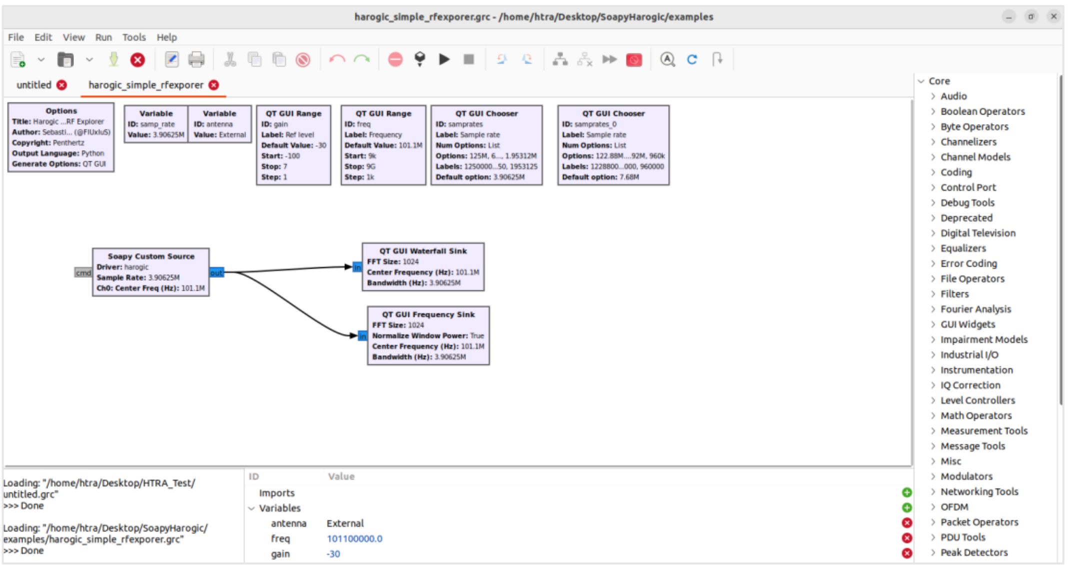


```
htra@htra-virtual-machine: ~/Desktop/SoapyHarogic
htra@htra-virtual-machine:~/Desktop/SoapyHarogic$ sudo SoapySDRUtil --find="driver=harogic"
[sudo] password for htra:
######################################################
##     Soapy SDR -- the SDR abstraction library     ##
######################################################

Found device 0
  driver = harogic
  label = Harogic 5430500500380032
  serial = 5430500500380032
```

6. You can then refer to the documentation in the Examples folder to use the equipment:



harogic_simple_rfexplorer.py
harogic_simple_rfexporer.grc



harogic_simple_rfexporer.grc - /home/htra/Desktop/SoapyHarogic/examples

Options
Title: Harogic ...RF Explorer
Author: Sebasti... (@FIUxluS)
Copyright: Penthertz
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 3.90625M

Variable
ID: antenna
Value: External

QT GUI Range
ID: gain
Label: Ref level
Default Value: -30
Start: -100
Stop: 7
Step: 1

QT GUI Range
ID: freq
Label: Frequency
Default Value: 101.1M
Start: 9k
Stop: 9G
Step: 1k

QT GUI Chooser
ID: samprates
Label: Sample rate
Num Options: List
Options: 125M, 6..., 1.95312M
Labels: 1250000...50, 1953125
Default option: 3.90625M

QT GUI Chooser
ID: samprates_0
Label: Sample rate
Num Options: List
Options: 122.88M....92M, 960k
Labels: 1228800....000, 960000
Default option: 7.68M

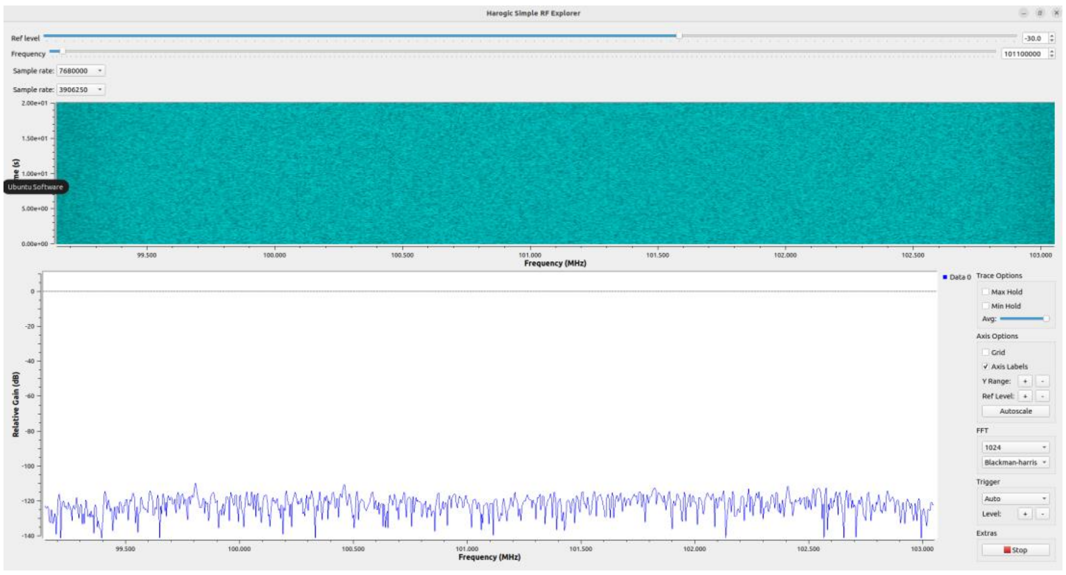
Soapy Custom Source
Driver: harogic
Sample Rate: 3.90625M
Ch0: Center Freq (Hz): 101.1M

QT GUI Waterfall Sink
FFT Size: 1024
Center Frequency (Hz): 101.1M
Bandwidth (Hz): 3.90625M

QT GUI Frequency Sink
FFT Size: 1024
Normalize Window Power: True
Center Frequency (Hz): 101.1M
Bandwidth (Hz): 3.90625M

```
Loading: "/home/htra/Desktop/HTRA_Test/
untitled.grc"
>>> Done

Loading: "/home/htra/Desktop/SoapyHarogic/
examples/harogic_simple_rfexporer.grc"
>>> Done
```

ID	Value
Imports	
Variables	
antenna	External
freq	101100000.0
gain	-30


Harogic Simple RF Explorer

Ref level: -30.0

Frequency: 101100000

Sample rate: 7680000

Sample rate: 3906250

Trace Options: [ ] Max Hold, [ ] Min Hold

Axis Options: [ ] Grid, [✓ Axis Labels]

FFT: 1024, Blackman-harris

Trigger: Auto

9.8 Java (to be supplemented)

It is important to note that when writing programs in Java to control devices on a Linux system, the CalFile folder needs to be placed in the same directory as the Java interpreter.