Telink IoT Studio User guide

AN-22063002-E1

Ver1.0.10

2025.02.21

Keyword

Telink, IoT Studio

Brief

This document is to guide how to use Telink IoT Studio.

Published by
Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,
Zhangjiang Hi-Tech Park, Shanghai, China

Legal Disclaimer

This document is provided as-is. Telink Semiconductor reserves the right to make improvements without further notice to this document or any products herein. This document may contain technical inaccuracies or typographical errors. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein.

Information

For further information on the technology, product and business term, please contact Telink Semiconductor Company www.telink-semi.com

For sales or technical support, please send email to the address of:

telinksales@telink-semi.com

telinksupport@telink-semi.com

Version

Revision	Date	Author	Change log
V1.0.0	2022-06	Xiongjun.He	First release
V1.0.1	2022-07	Xiongjun.He	Add more known issues Add IDE setting chapter
V1.0.2	2022-09	Xiongjun.He	Select version with chip type
V1.0.3	2022-11	H.X.J,W.F	Add Jtag debugging section
V1.0.4	2023-03	H.X.J,W.F	Telink IDE is renamed to Telink IoT Studio Add usage of some tools
V1.0.5	2023-08	H.X.J,W.F	Update toolchain to V5.1.2 Add description about building V3.2.3 and V5.1.2 projects/ configurations simultaneously Add installation and uninstallation in linux
V1.0.6	2024-03	H.X.J,W.F	Add usage of Smart Build Button and Telink ICEman GDB Debugging Add explanation for andes doc and different versions of ICEman
V1.0.7	2024-04	H.X.J	Add usage of Telink Formatter
V1.0.8	2024-06	H.X.J,W.F	Add usage of Telink IoTStudio to CMake and update Telink menu pictures Delete all content related to build tool
V1.0.9	2024-09	H.X.J,W.F	Modify the description of toolchains and ICEman Add instructions on how to compile in the command line Add methods to enable Windows OS case sensitivity
V1.0.10	2025-02	H.X.J,W.F	Modify the usage of TelinkFormatter Add the usage of SVD Support

Contents

List of Figures

1 Desktop shortcuts

The Telink IoT studio installer will create 2 shortcuts on the desktop:

- `Telink IoT studio` : used to launch the IoT Studio.
- `Telink TC32 console` : used to launch the development console for TLSR8, users can input commands like `tc32-elf-gcc` and `make` in it.

2 Different IDEs

The following 3 different IDEs will be referred to in this user guide:

- `Telink IoT studio` : the IDE contains this document.
- `Telink old IDE` : the IDE list on this wiki page (IDE for TLSR8 Chips) for TLSR8 chips.
- `Telink RDS IDE` : the IDE list on this wiki page (IDE for TLSR9 Chips) for TLSR9 chips.

The above 3 IDEs have the following different desktop shortcut icons:

Telink IoT studio:

Figure 2.1: Telink IoT studio Icon

Telink old IDE:

Figure 2.2: Telink Old IDE for TLSR8 Icon

Telink RDS IDE:

Figure 2.3: Telink RDS IDE for TLSR9 Icon

3 IoT Studio or integrated tools

- Build TLSR8 Projects and SDK
- Import and build TLSR9 Telink RDS Projects and SDK
- Telink Windows BDT used to flash image for both TLSR8 and TLSR9
- Telink libUsb version BDT used to flash image for both TLSR8 and TLSR9, still in beta phase.
- Jtag_Burn used to flash image for TLSR9 with JTAG

4 Project property difference between Telink RDS and Telink IoT studio

4.1 Ext HW DSP option

This option is used to configure whether to use the DSP and its libraries:

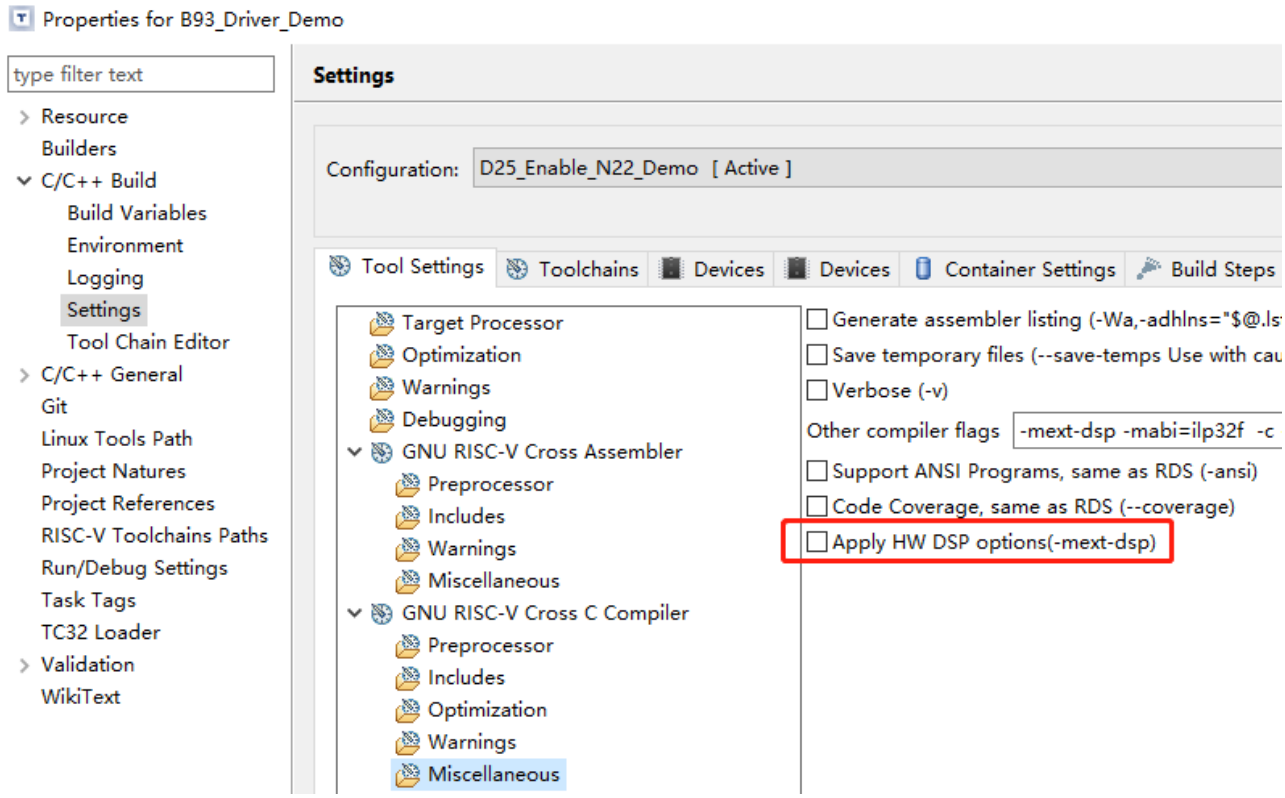

Figure 4.1: DSP option in Telink IoT studio

This option would add `-mext-dsp` flag to `cc` and `ld` , and also it would add `-lm` and `-ldsp` flags to linker in Telink RDS IDE. While the Telink IoT studio would only add `-mext-dsp` flag to `cc` and `ld` without the `-lm` and `-ldsp` extra libraries flags. This makes the configuration more flexible.

Users can add the `math` and `dsp` libraries to linker specifically on the linker library setting page:

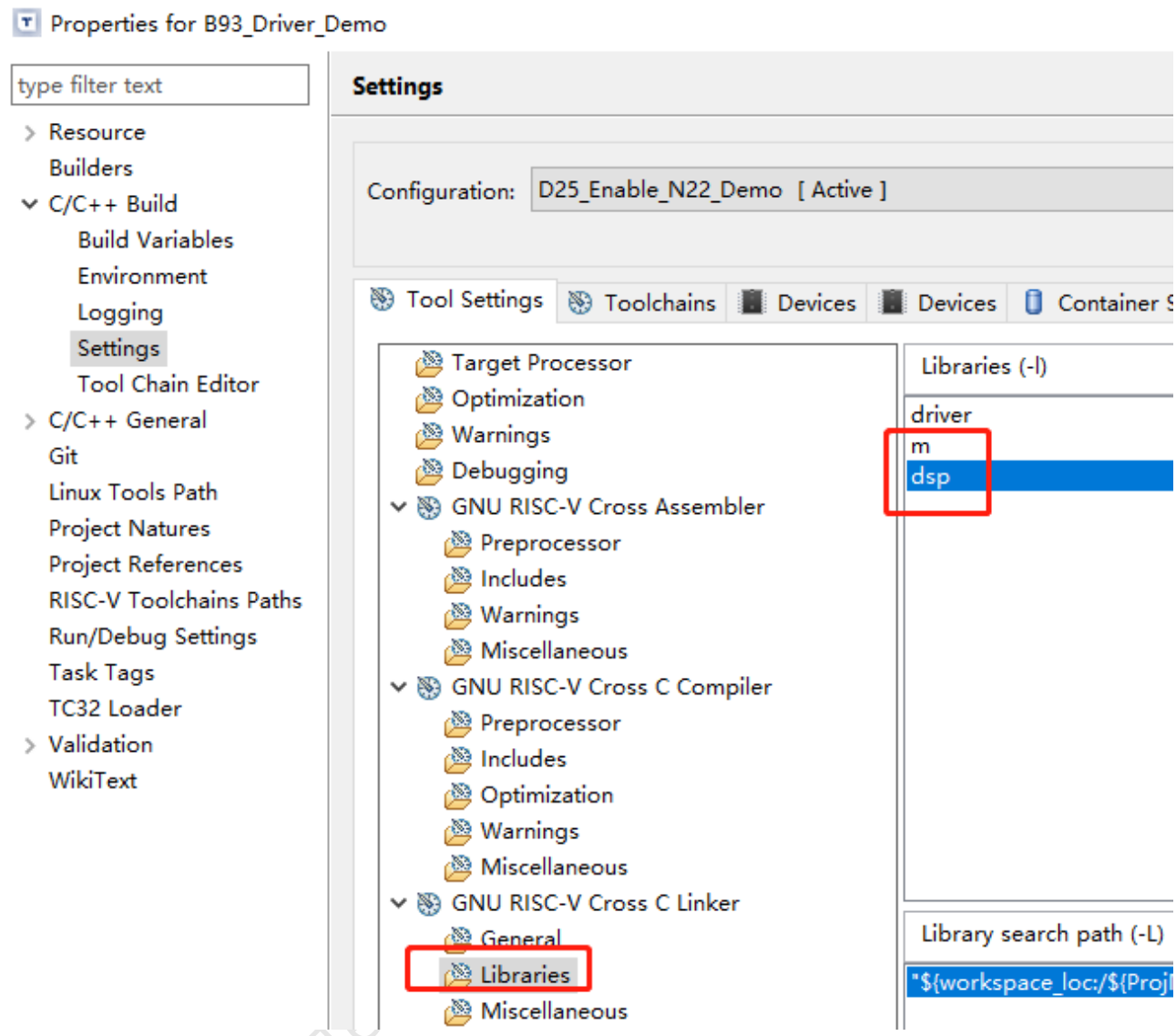

Figure 4.2: Add DSP libraries in Telink IoT studio

4.2 `-msave-restore` and `-msmall-data-limit`

The `Telink RDS IDE` uses the IoT Studio built-in (default) value for both `-msave-restore` and `-msmall-data-limit` without options on configuration page, while the `Telink IoT studio` has a more flexible configuration for these options.

Right now, users should use the `Use toolchain default value` option for TLSR9 chips.

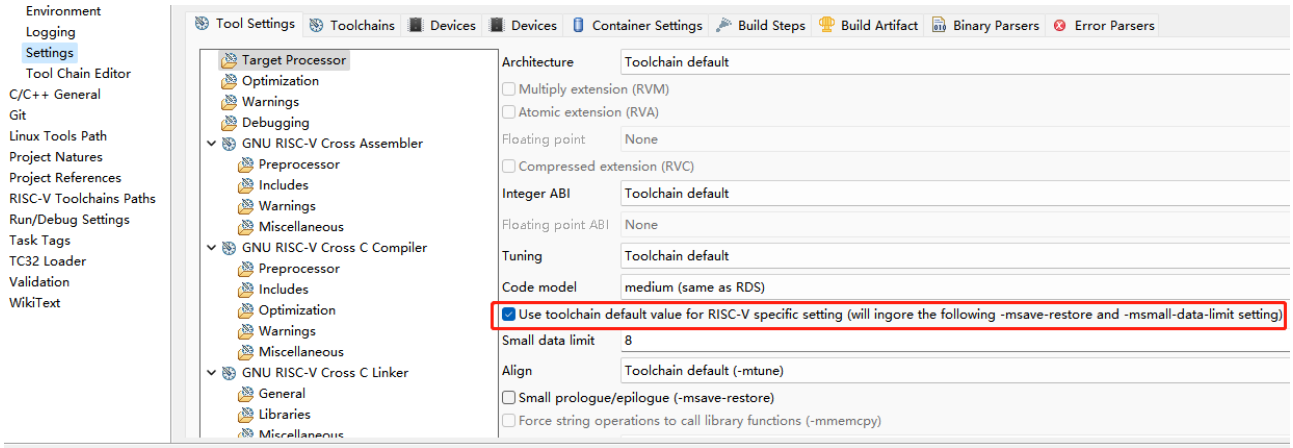

Figure 4.3: RISC-V specific option on Telink IoT studio

5 Toolchain

5.1 Toolchain information

The Telink IoT studio contains several toolchains for TLSR8 and TLSR9:

- Telink TC32 Cross Compiler with GCC version 4.5.1 (tc32-elf) for TLSR8 which is the same as the Telink old IDE.
- Cross Compiler with GCC version 7.4 from Telink RDS IDE V3.2.3 for TLSR9 (D25F and N22 architecture) chips
- Cross Compiler with GCC version 10.3 from Telink RDS IDE V5.1.2 for TLSR9 (D25F and N22 architecture) chips
- Cross Compiler with GCC version 12.2 from Telink RDS IDE V5.3.x for TLSR9 (D25F and N22 architecture) chips

5.1.1 Toolchain location

We assume that the Telink IoT studio is installed at location `$IoTStudio` :

- Telink TC32 Cross Compiler with GCC version 4.5.1 (tc32-elf-1.5) for TLSR8: `$IoTStudio/opt/tc32/bin`
- Cross Compiler with GCC version 7.4 from Telink RDS IDE V3.2.3 for TLSR9 (D25F and N22 architecture) chips
- N22: `$IoTStudio/RDS/V3.2.3/toolchains/nds32le-elf-mculib-v5`
- D25F: `$IoTStudio/RDS/V3.2.3/toolchains/nds32le-elf-mculib-v5f`
- Cross Compiler with GCC version 10.3 from Telink RDS IDE V5.1.2 for TLSR9 (D25F and N22 architecture) chips
- N22: `$IoTStudio/RDS/V5.1.2/toolchains/nds32le-elf-mculib-v5`
- D25F: `$IoTStudio/RDS/V5.1.2/toolchains/nds32le-elf-mculib-v5f`
- Cross Compiler with GCC version 12.2 from Telink RDS IDE V5.3.x for TLSR9 (D25F and N22 architecture) chips

- N22: `$IoTStudio/RDS/V5.3.x/toolchains/nds32le-elf-mculib-v5`
- D25F: `$IoTStudio/RDS/V5.3.x/toolchains/nds32le-elf-mculib-v5f`

5.2 Select a different toolchain for TLSR9 projects

As only one toolchain is included for TLSR8, this section is only applied to TLSR9 chips.

To change the toolchain of a project, open the project properties, navigate to `C/C++ Build` -> `Settings` -> `Toolchains` tab, and select the different one from the drop-down list:

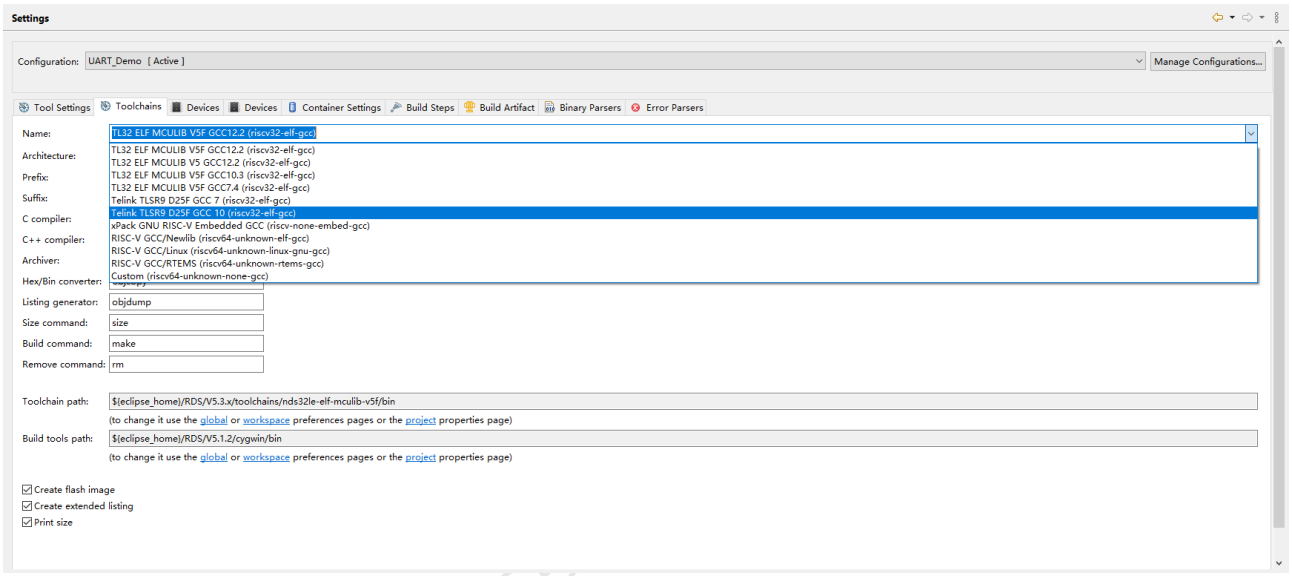

Figure 5.1: Select Toolchain

The listed toolchains are:

- `TL32 ELF MCULIB V5F GCC12.2` : for D25F architecture chips, from RDS V5.3.x
- `TL32 ELF MCULIB V5 GCC12.2` : for N22 architecture chips, from RDS V5.3.x
- `TL32 ELF MCULIB V5F GCC10.3` : for D25F architecture chips, from RDS V5.1.2
- `Telink TLSR9 D25F GCC 10` : same as `TL32 ELF MCULIB V5F GCC10.3`
- `TL32 ELF MCULIB V5F GCC7.4` : for D25F architecture chips, from RDS V3.2.3
- `Telink TLSR9 D25F GCC 7` : same as `TL32 ELF MCULIB V5F GCC7.4`

5.2.1 Advanced usage: toolchain setting according to path

Normally, users shouldn't set the toolchain by changing toolchain path. Instead, a toolchain should be set with the method mentioned in the previous section: `Select a different toolchain for TLSR9 projects` .

This section is for advanced users with special requirements.

Open the project property dialog by clicking the Property menu item on right and click the pop-up menu on the project:

Switch to the `C/C++ Build` -> `Settings` -> `Toolchains` tab:

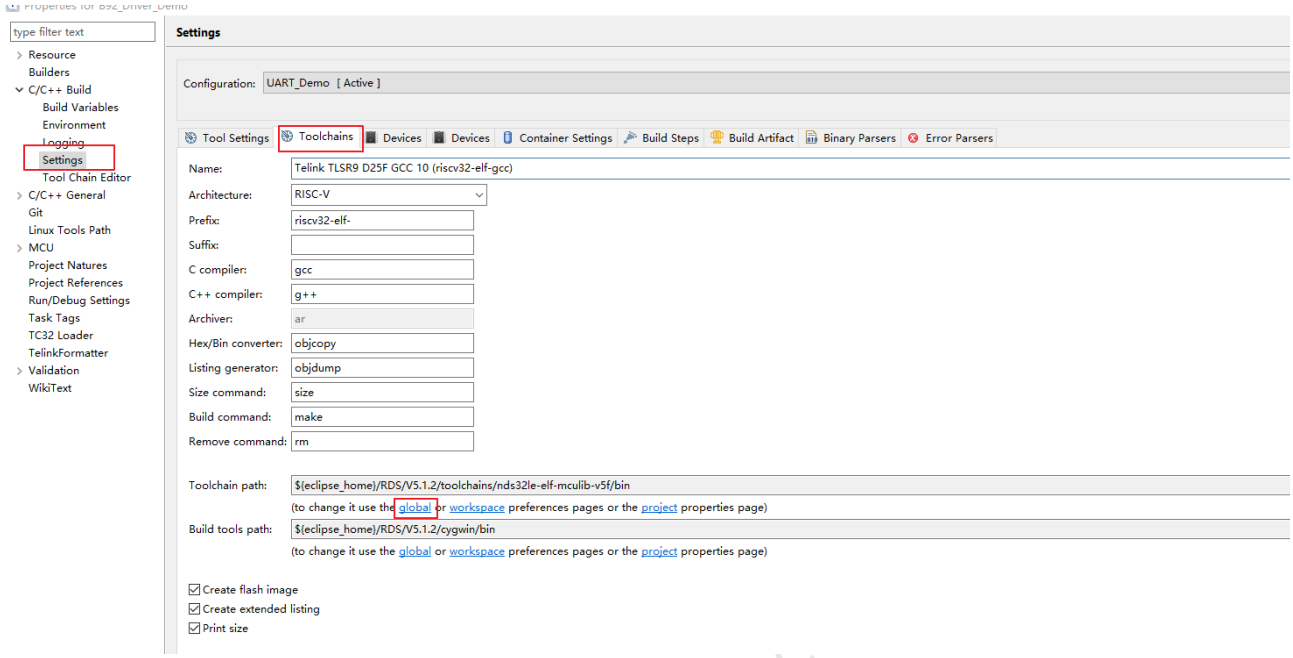

Figure 5.2: Toolchain setting

Click the `global` link text which is surrounded in red square block.

Set the path string in the line edit.

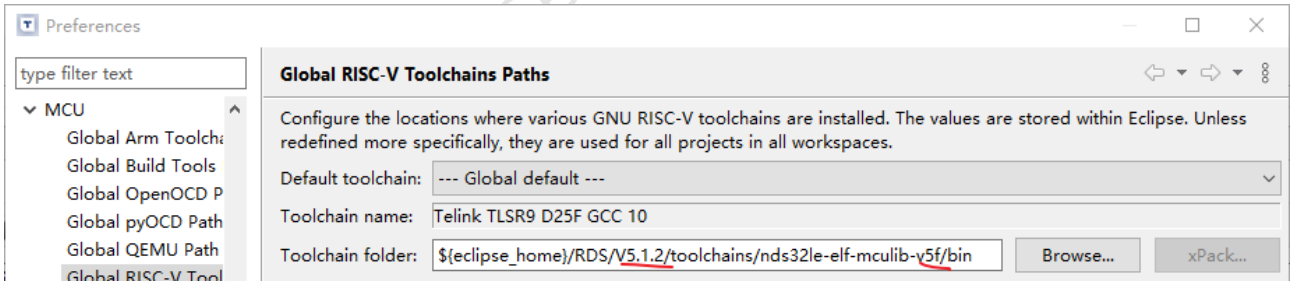

Figure 5.3: Set Toolchain path

Normally, users would only be required to change the version and the suffix which are marked with red lines.

The first red line marks the toolchain base version string (in the image it is `V5.1.2`) of the toolchain, users can change it to one of the following items:

- `V3.2.3` : from Telink RDS IDE V3.2.3
- `V5.1.2` : from Telink RDS IDE V5.1.2
- `V5.3.x` : from Telink RDS IDE V5.3.x

The second red line indicates the version of the toolchain variants string (in the image it is `nds32le-elf-mculib-v5f`) of the toolchain, users can change it to one of the following items:

- `nds32le-elf-mculib-v5f` : for Telink TLSR9 D25F chips
- `nds32le-elf-mculib-v5` : for Telink TLSR9 N22 chips

Once the toolchain path variable has been changed, click the `apply and close` button to save it.

6 Import and build projects

6.1 TLSR8

Telink IoT studio supports SDK or projects import function like the Telink old IDE.

Users can use the `File` -> `Import...` to import SDK or projects:

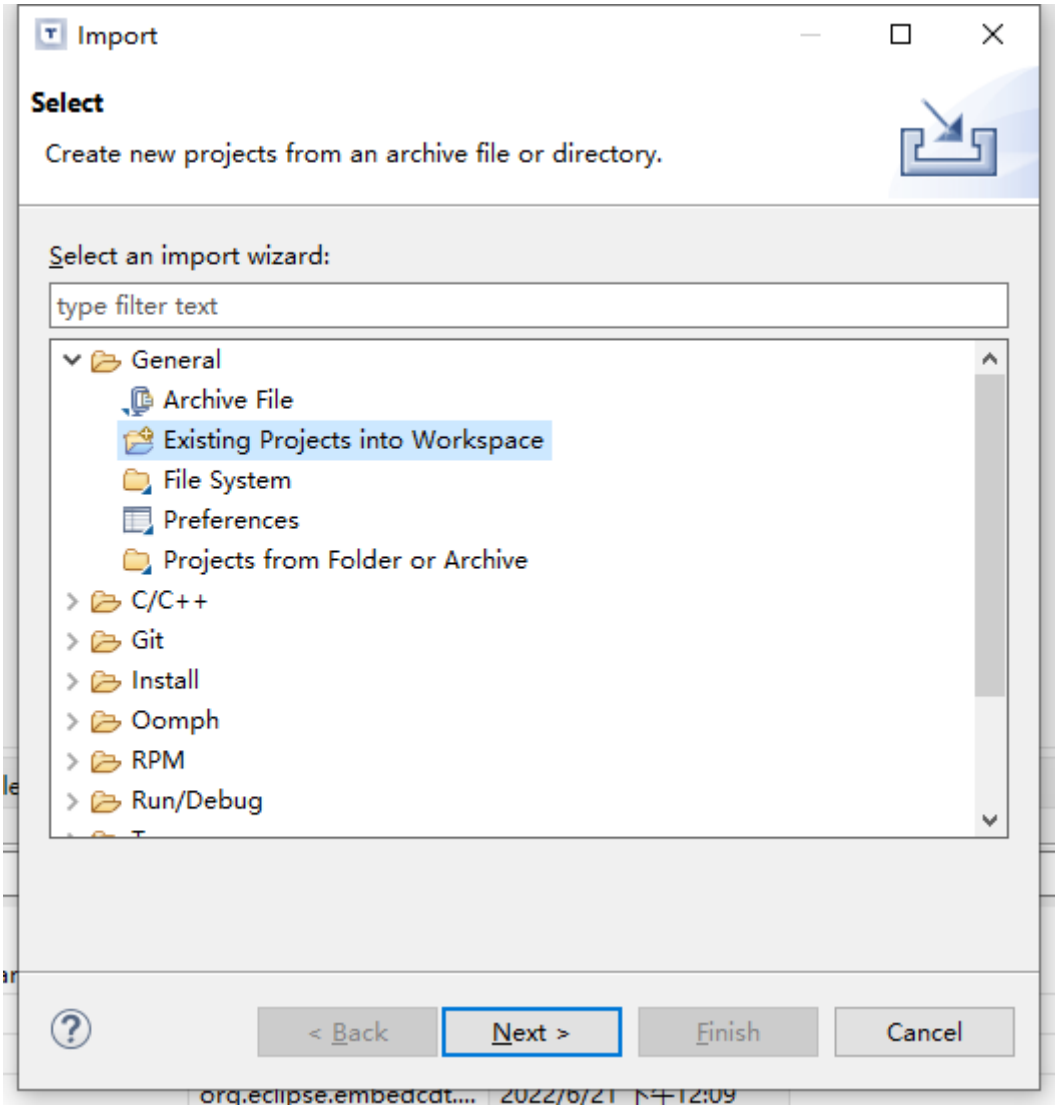

Figure 6.1: Import

Users can refer to this Telink wiki page.

6.2 TLSR9

For `Telink RDS IDE` format projects(for example, the SDKs for TLSR9 chips at Telink official wiki page), it must be converted to `Telink IoT studio` format before importing it into the workspace.

Users can refer to the following `RDS to IDE converter` section on `Telink menu` chapter to view how to do it.

7 Telink menu and toolbar entries

7.1 RDS to IoT Studio converter

If you get the SDK or projects for TLSR9 chip `B91` from Telink official wiki page, they are in `Telink RDS IDE` format, you can use this function to convert it to `Telink IoT studio` format before importing it into Telink IoT studio, and SDK or projects of `B92` do not require conversion.

Click the `Telink RDS to IDE Converter` in `Telink` menu:

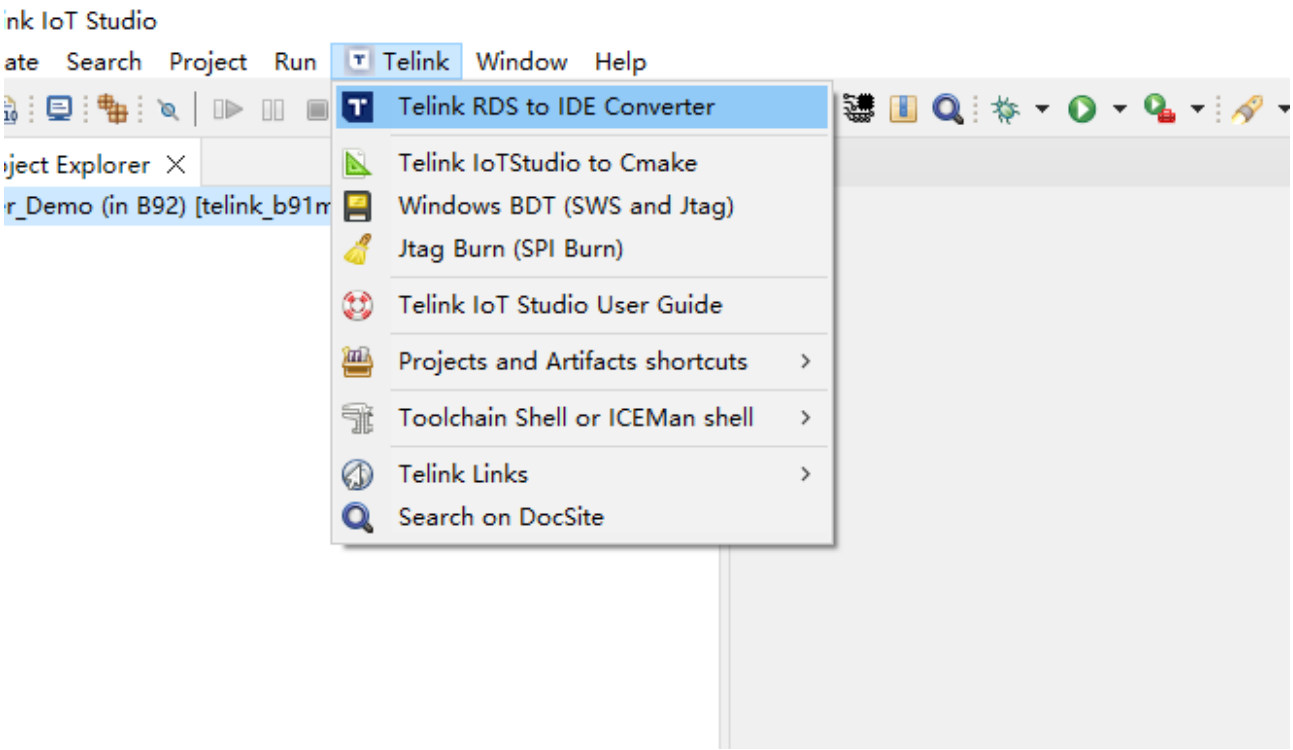

Figure 7.1: Telink Menu

Click the `Select .cproject file` button in the dialog to choose the Telink RDS IDE format `.cproject` file.

In Linux OS, users can use `Ctrl + h` or other shortcut keys to show the hide files (`.cproject`), and the filter should be changed to `*` :

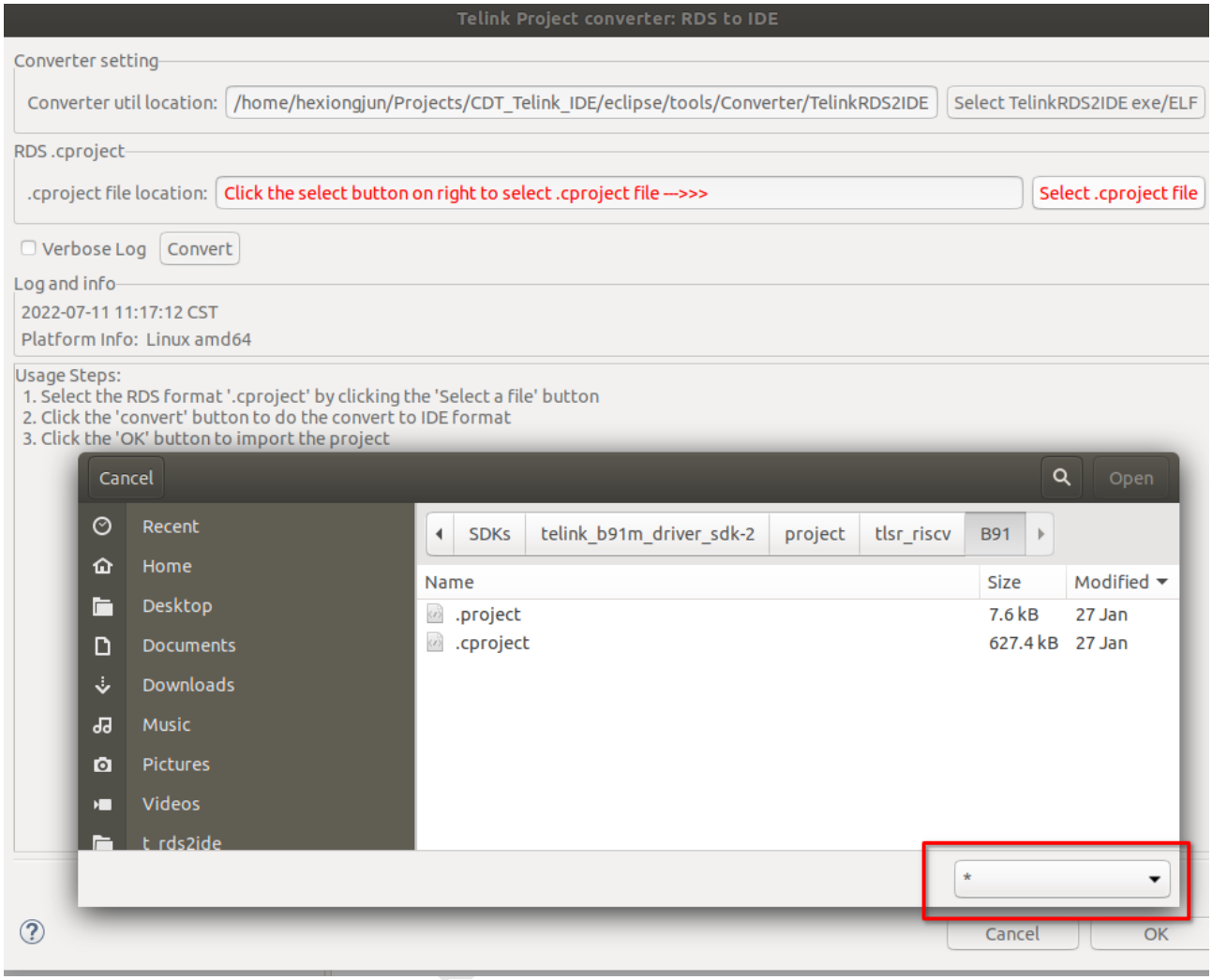

Figure 7.2: Linux Select cproject

If a `.cproject` file has been selected, it will display the file path:

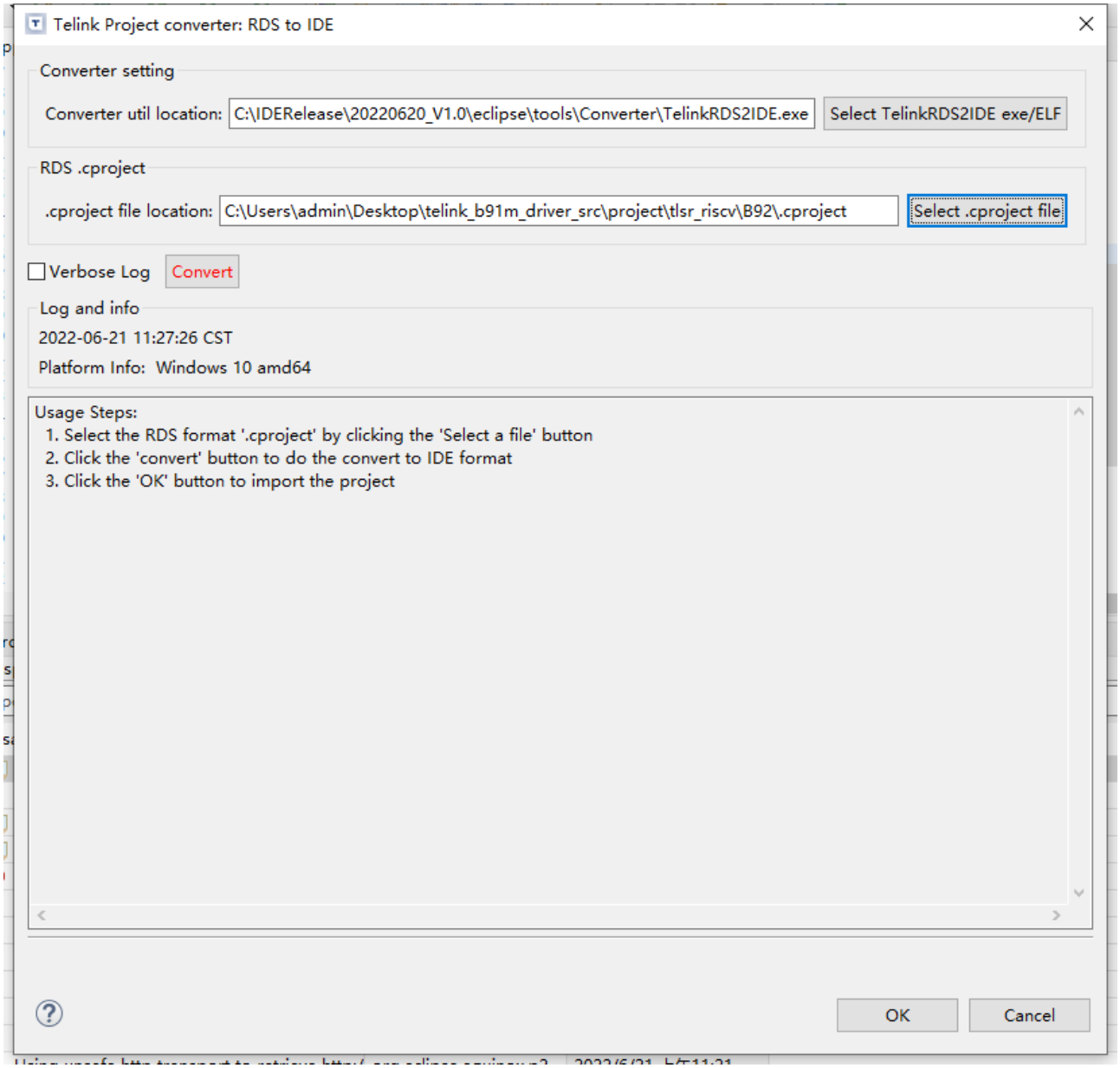

Figure 7.3: Telink project converter

Then click the `convert` button, the result would show on the large text block:

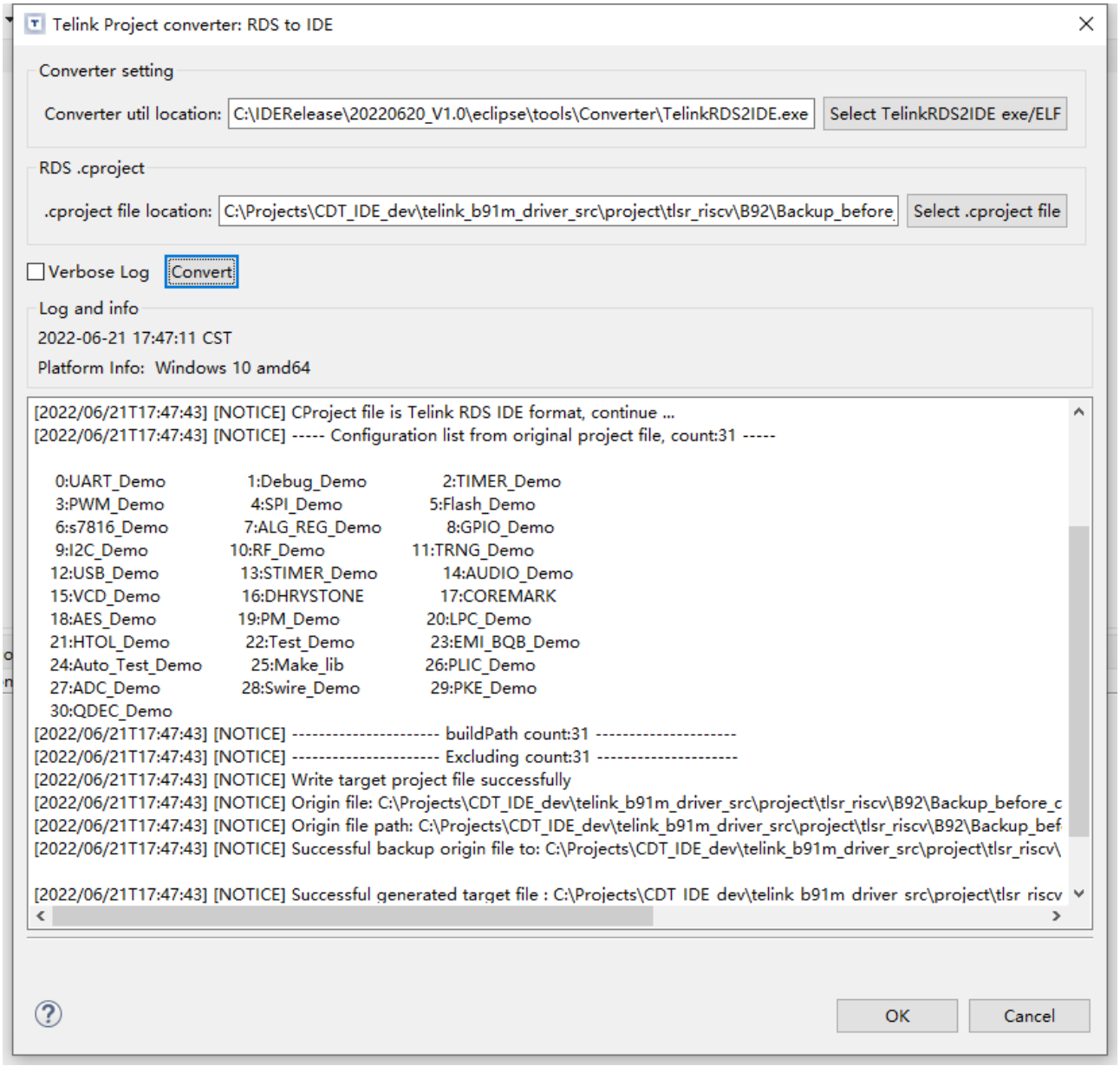

Figure 7.4: Converter result

If it is fails, the background of log text block would be set to red. Users can copy and send the log to `Telink IoT studio` developer/FAE to get support.

7.1.1 Using converter as a standalone utility

The converter can be used as a standalone utility, users can use it in command line.

The converter is installed at:

- Windows: $IoTStudio_PATH/tools/Converter/TelinkRDS2IDE.exe
- Linux: $IoTStudio_PATH/tools/Converter/TelinkRDS2IDE

To convert a `.cproject` file from command line in Windows OS, users can use following the command:

```
$IoTStudio_PATH/tools/Converter/TelinkRDS2IDE.exe Path/To/.cproject
```

In Linux OS:

```
$IoTStudio_PATH/tools/Converter/TelinkRDS2IDE Path/To/.cproject
```

7.1.2 Notice

The converted project (`.cproject`) would set the toolchain to `Telink TLSR9 D25F GCC` by default, which will use the `Telink RDS IDE` 7.4 GCC Cross compiler for D25F.

7.2 Telink IoTStudio to Cmake converter

If you need to convert the Telink IoTStudio project to a cmake project, you can use the tool shown in the following figure.

First, you need to double-click to select the project you want to convert, and then click `Telink IoTStudio to Cmake` in `Telink` menu.

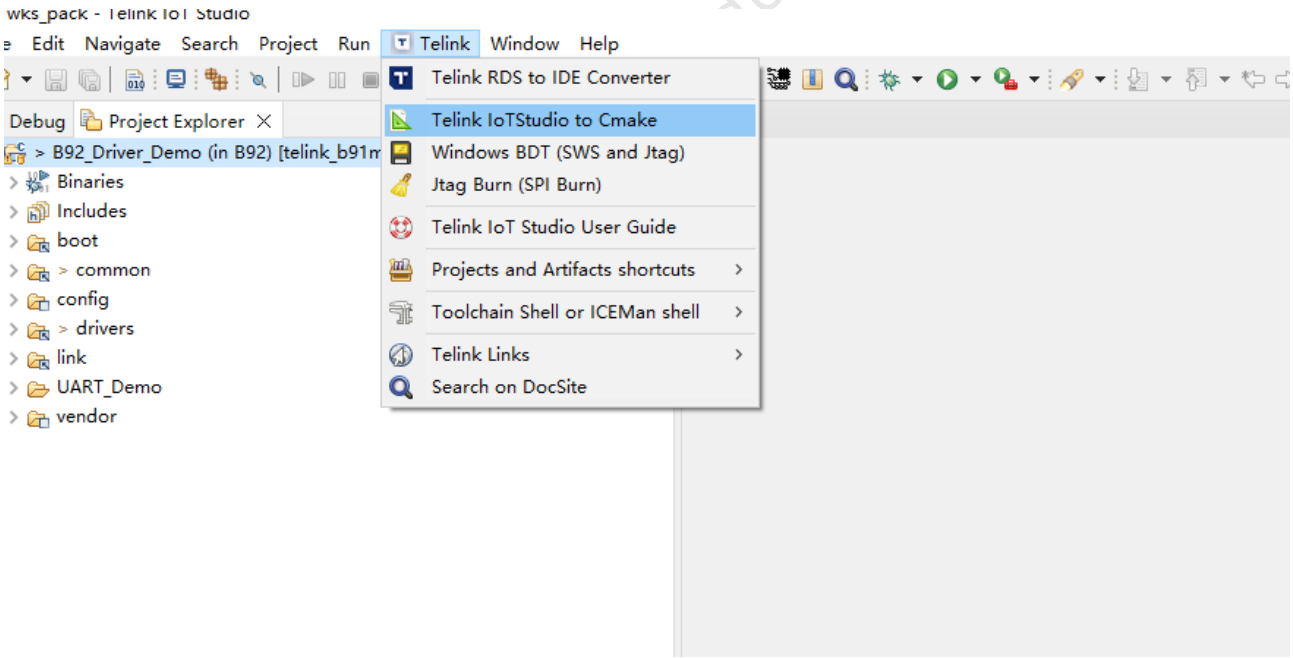

Figure 7.5: Telink IoTStuio to Cmake

You can choose whether to generate the `settings.json` file required for the VS Code project. Click Convert to obtain the `CMakelists.txt` and `settings.json` files, and the log will indicate their location.

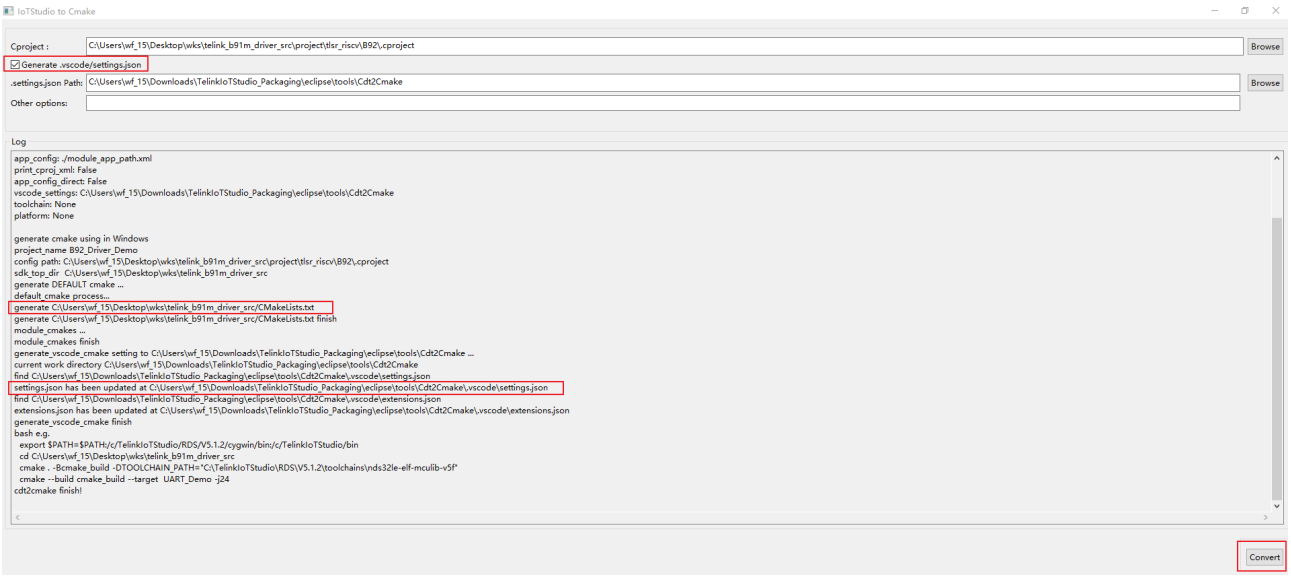

Figure 7.6: Telink IoStudio to Cmake UI

7.3 Jtag burn menu

The `Jtag_Burn` function is similar to the `Telink RDS IDE` Jtag_Burn function, both use the JTAG to burn the image to TLSR9X chips.

Except for the `Jtag_Burn` , TLSR9 can be burned with `Windows BDT` software through `SWS` protocol with `Burning EVK` , which is the recommended option.

The `SWS` protocol requires only one GPIO to flash the binary image to TLSR8 and TLSR9 chips, while the JTAG uses 2 or 4 GPIO pins for TLSR9 chips.

7.3.1 Parameters

Once the Jtag_Burn dialog has been opened, users can fill or set parameters for Jtag_Burn and ICEMan:

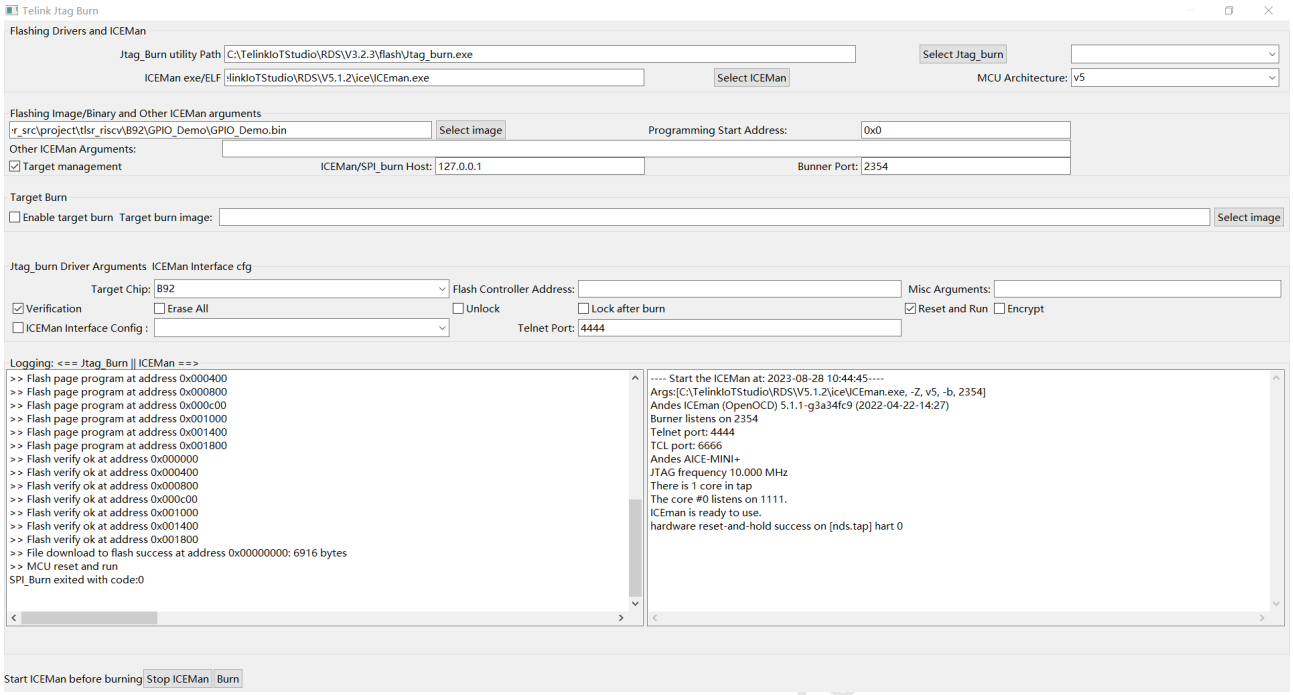

Figure 7.7: Jtag burn dialog

Parameters for `Jtag_Burn` can be set in `Jtag_Burn driver Arguments` .

7.3.2 Use ICEMan in Jtag_Burn dialog

ICEMan is a tool to manage the TLSR9 JTAG ICE (support 4-wire and 2-wire mode).

Users can refer to the document files in the `doc` directory to get more information.

7.3.3 ICEMan in Jtag_Burn ELF setting

Users can set the other ICEMan and Jtag_Burn ELF from `Flashing Drivers and ICEMan` section, the set/selected value will be saved and reused in next re-opening the dialog.

The normal ELF files' names are listed for reference:

- ICEMan: `ICEMan.exe` in Windows, and `ICEMan` on Linux OS
- Jtag_Burn: `Jtag_Burn.exe` in Windows, and `Jtag_Burn` on Linux OS

7.3.4 Burn Program

When burning the program, pay attention to burner's path and chip type, and then click Start ICEman. After ICEman is ready, the telnet_port and burner_port should be consistent with ICEman's, then click Burn, as shown below:

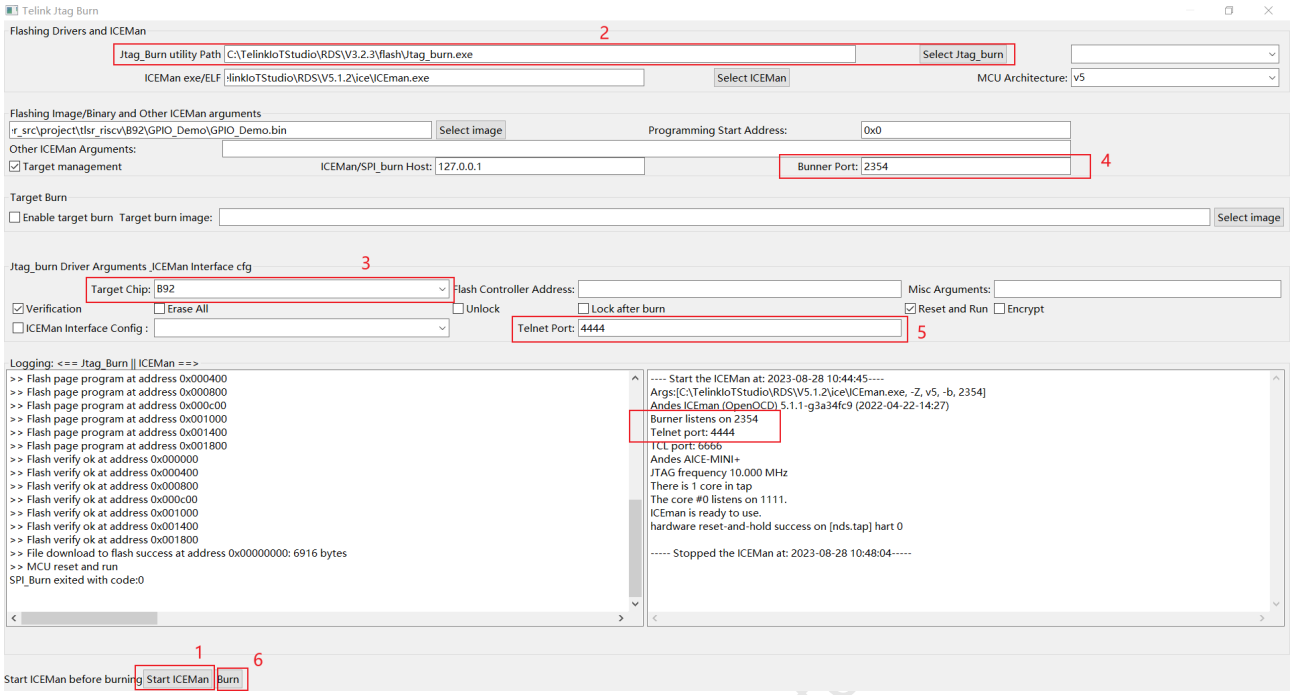

Figure 7.8: Jtag burn

Jtag_ Burn is located in the $IoTStudio_PATH/RDS/V3.2.3/flash/directory, and its help documents are also in the same directory.

7.3.5 Security Download

If you need to use the security download function of Jtag_Burn, check the encrypt checkbox

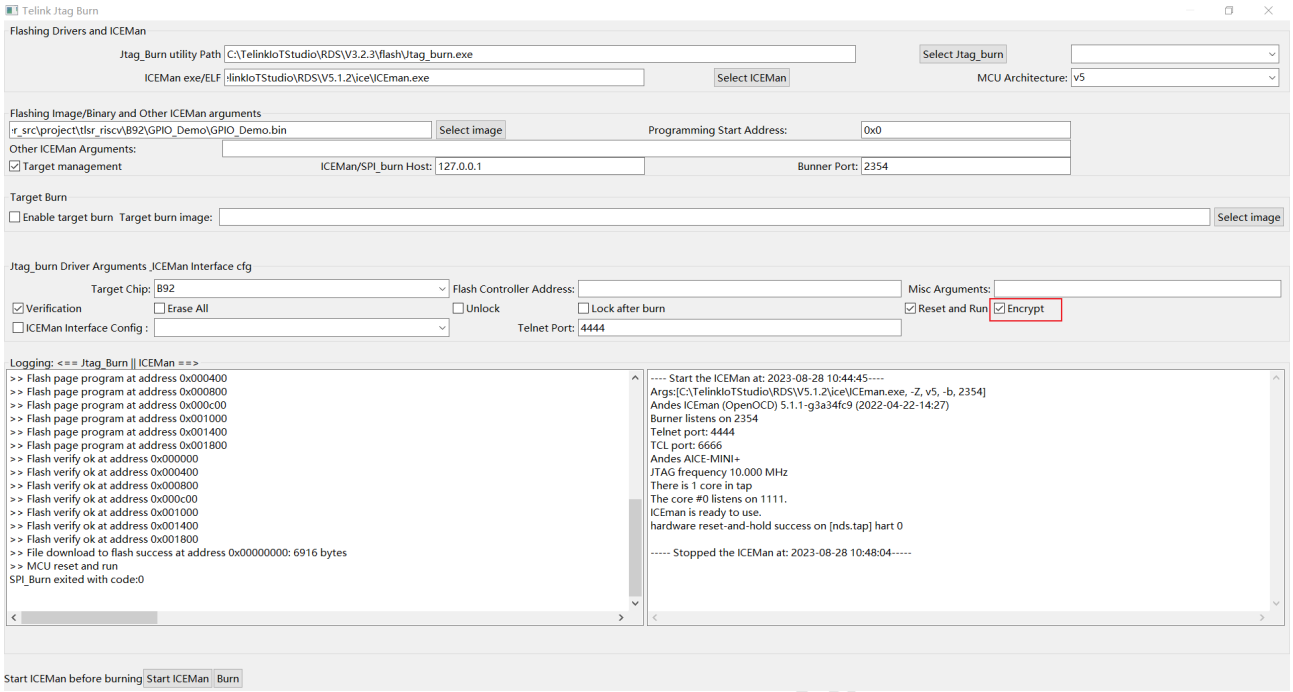

Figure 7.9: Jtag Burn Encrypt

7.4 Telink links

In this sub-menu, users can quickly visit Telink websites for development resources or information:

- Telink forum
- Telink wiki
- Telink official webpage

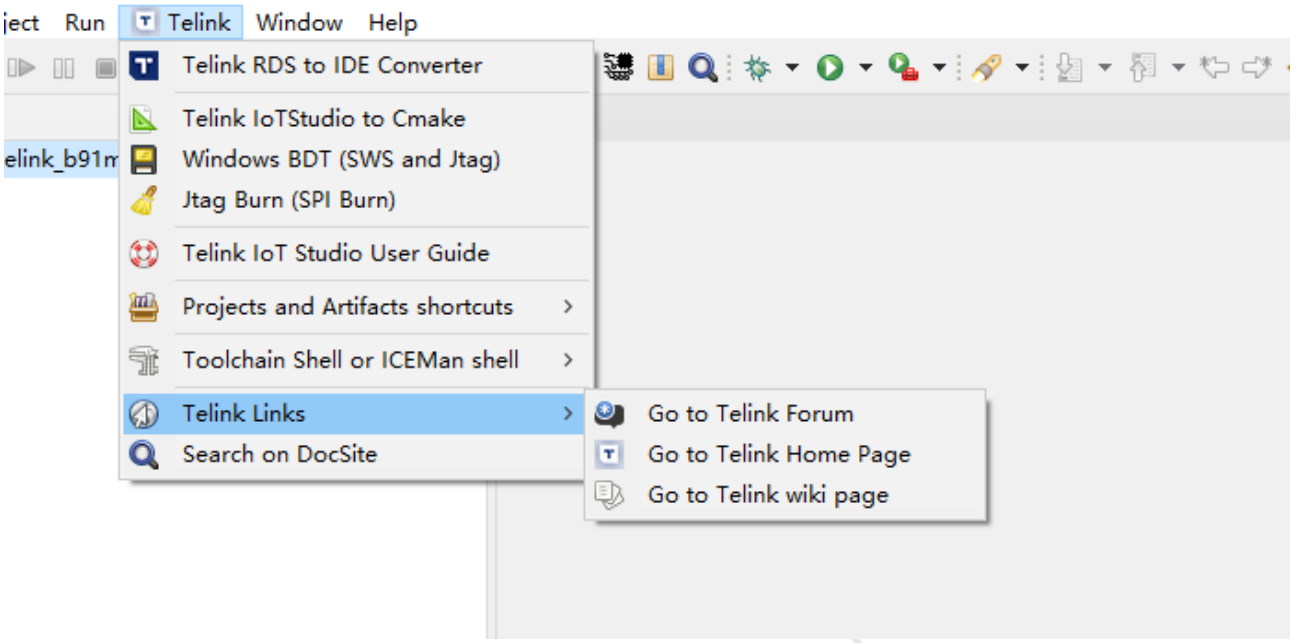

Figure 7.10: Telink Links

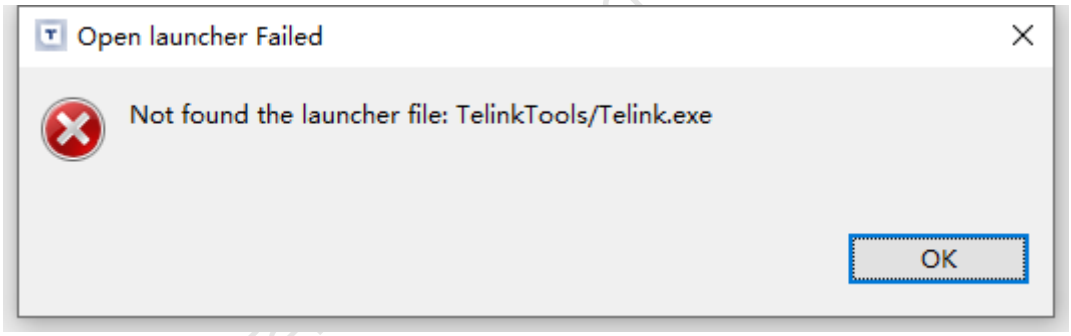

Figure 7.11: Telink tools launcher

7.5 libusb version BDT

The `libusb version BDT` is a new burning tool which is similar to `Windows BDT` for Windows OS and Linux OS, but this tool requires Burning EVK with new firmware. This tool can be used to flash the image to TLSR8 and TLSR9 chips through SWS. This tool is still in the beta phase. If this tool is not working on some Windows OS, use the `Windows BDT` (mentioned in the following `Windows BDT` section) instead.

Users can launch this tool and click the help on libUSB BDT menu to learn how to use it.

Clicking this menu entry would open the libusb version of BDT and pass the artifact bin path to it if a project has been selected and the artifact has been found, so users do not require to copy the artifact path manually:

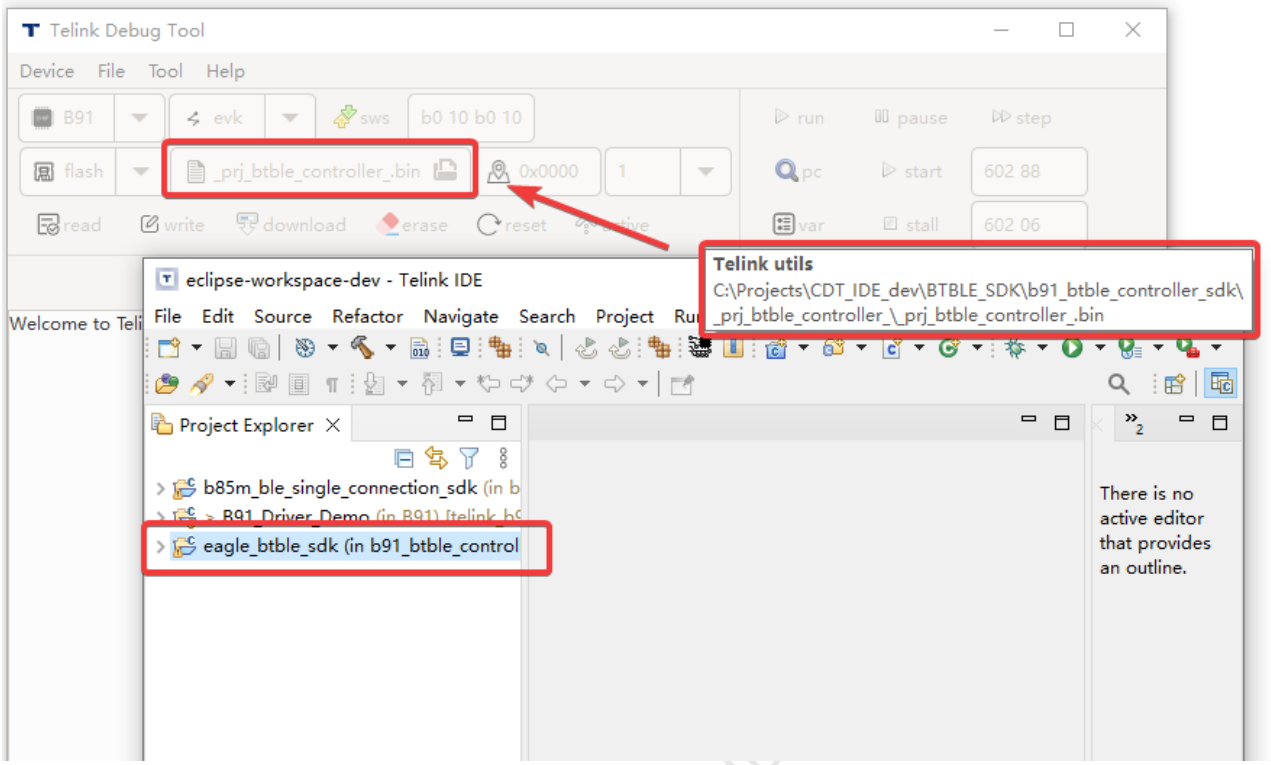

Figure 7.12: Open libUSB BDT with artifact Path

If no project is selected, a error dialog would show, users can close it, the libusb BDT program would be started normally as well:

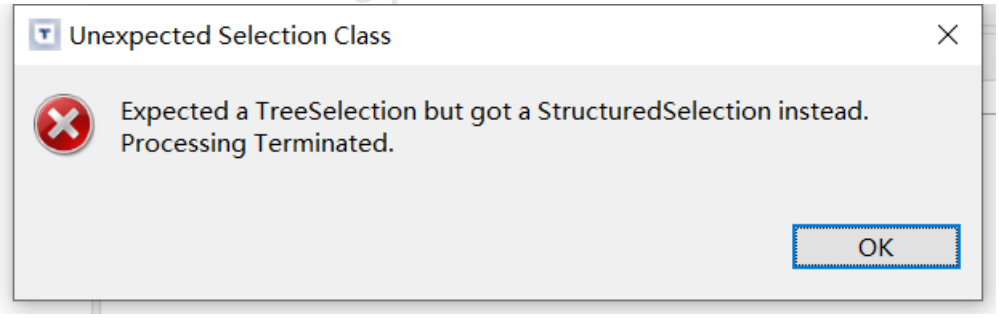

Figure 7.13: Open libUSB BDT without artifact Path

7.6 Toolchain shell or ICEMan shell menu entry

This is a submenu, it has several menu entries to start the toolchain console or ICEMan console, users can input commands (for example: `riscv-elf-gcc` or `ICEMan`) in these shells.

It is worth noting that we provide two versions of ICEman. Under `${IoTStudio_Path}/tools/ice_cygwin_from_andes/` , there is the original version of ICEman, while under `${IoTStudio_Path}/tools/ice/` , there is a new version of ICEman used by IoTStudio by default. It has made some modifications to match the functionality of IoTStudio and currently does not support IPv6.

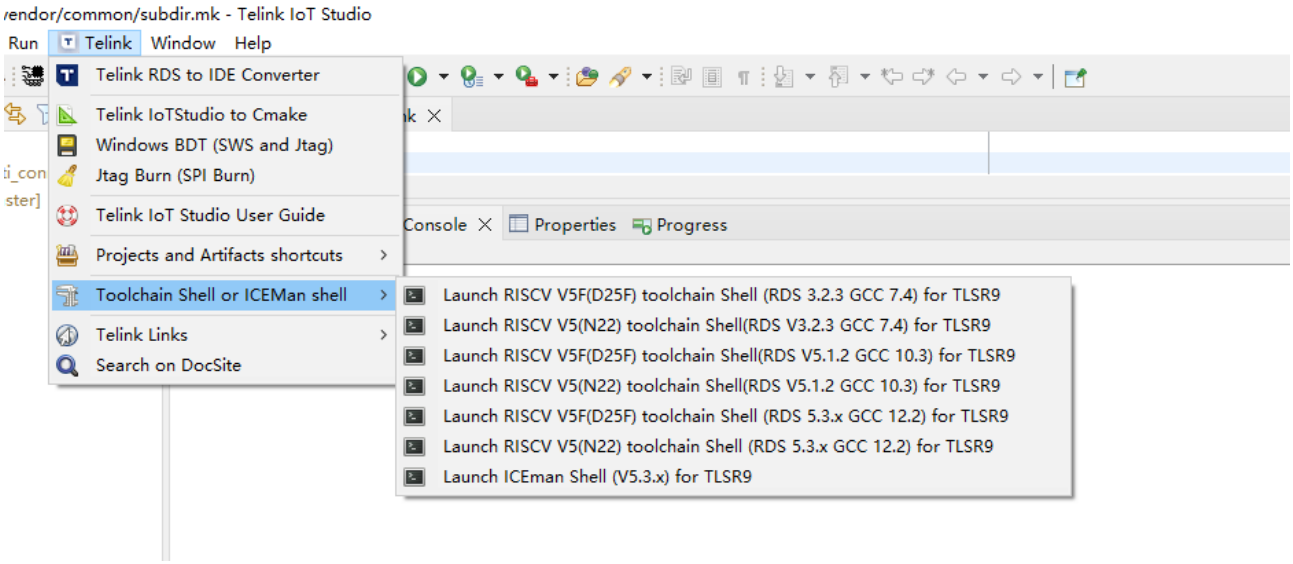

Figure 7.14: Shells

7.7 Windows BDT

The `Windows BDT (SWS and JTAG)` menu item would open the standalone program `Windows BDT`. Users can use this tool to flash or debug the TLSR8 and TLSR9 chips.

Refer to this wiki page on how to use it.

7.8 Open artifact path

There's an icon on the toolbar used to open the artifact directory:

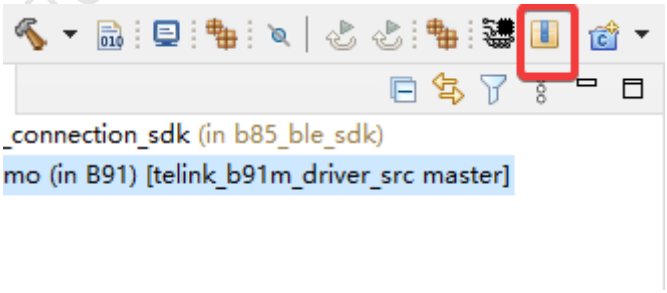

Figure 7.15: Open artifact path

It wouldn't work if the project was not properly configured (the `.cproject` file), this occurs in many old TLSR8 SDKs.

While most TLSR9 SDK projects work as expected.

7.8.1 Notice

The project or project file must be selected before clicking the icon, otherwise it will not work.

7.9 Copy artifact path

The Telink IoT studio added a handy menu entry to help users copy the artifact path:

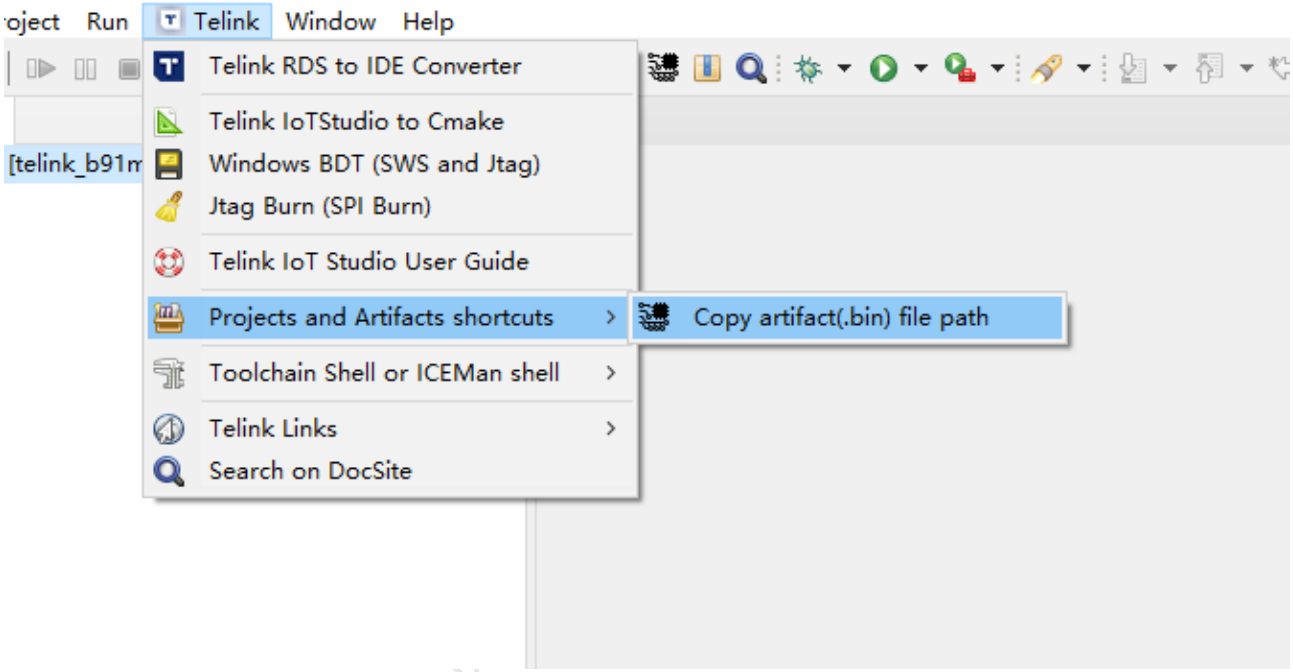

Figure 7.16: Copy artifact path

Users can click this icon to copy the artifact path to the system clipboard.

7.9.1 Notice

The project or project file must be selected before clicking the icon, otherwise it will not work.

7.9.2 In other places

The `Copy artifact path` menu entry also shows at toolbar and right-click the pop-up menu:

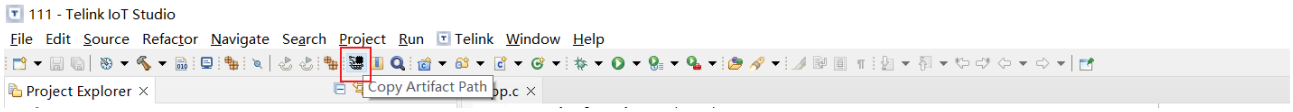

Figure 7.17: Copy artifact path on Toolbar

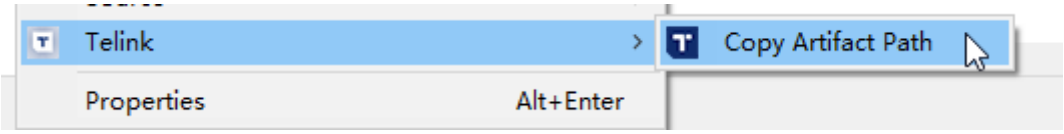

Figure 7.18: Copy artifact path on pop-up menu

7.10 Search on DocSite

This tool can be used to search for specific keywords in the Telink Document Center website. Double click a keyword in the project file and click "Search on DocSite" to jump to the Document Center search page.

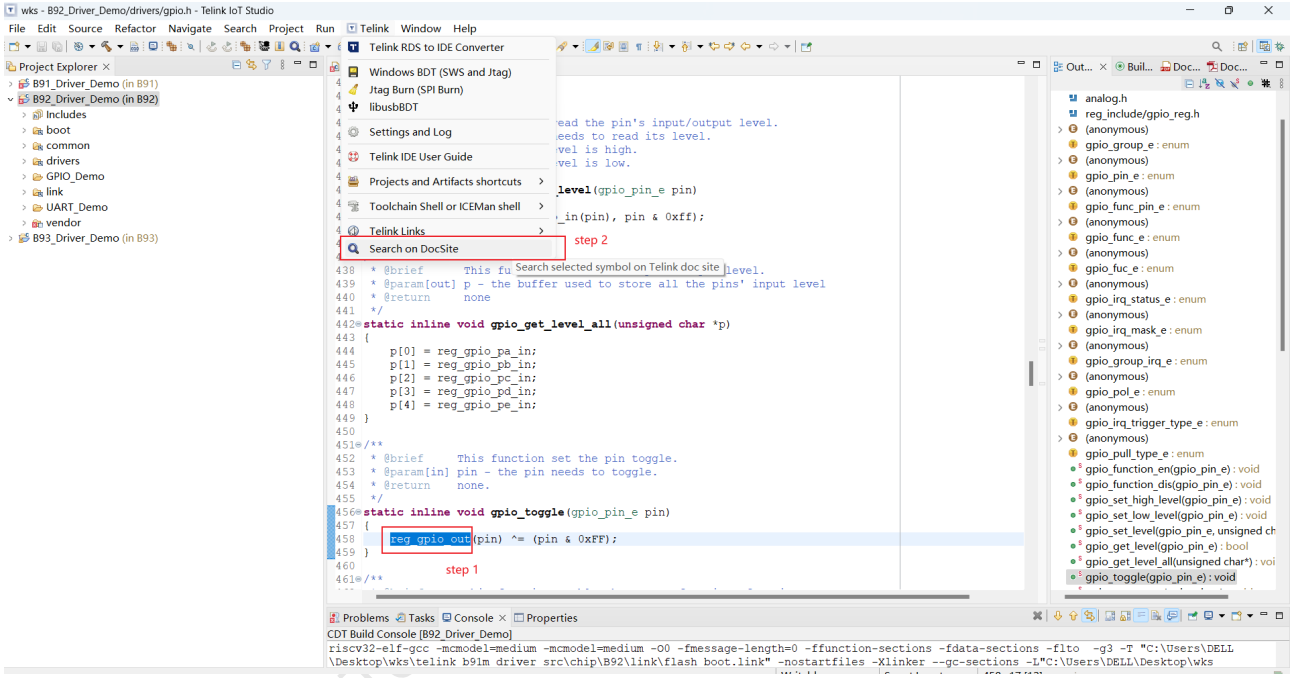

Figure 7.19: keyword

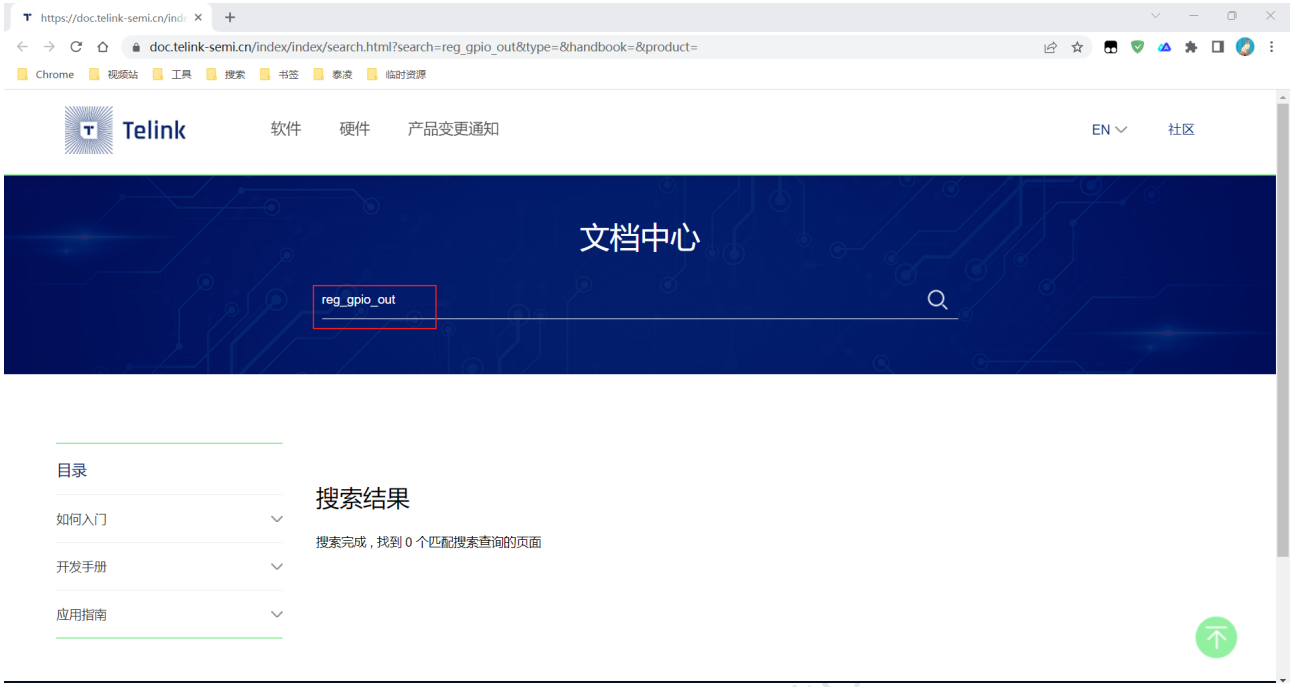

Figure 7.20: doc site

8 Common compiling error

8.1 Can't find the link script file

This type of error would occur when link script file path is in relative format. This wouldn't be an issue in `Telink RDS IDE` , but there will be an error in `Telink IoT studio` .

The error log would hint like the following:

```
Building target: AES_Demo.elf
Invoking: GNU RISC-V Cross C Linker
riscv32-elf-gcc -mcmodel=medium -mcmodel=medium -O2 -fmessage-length=0 -ffunction-sections -fdata-sections -flto  -g3 -T ../link/flash_boot.link -nostartfiles -Xlinker --gc-sections -L"C:\Projects
\CDT_IDE_dev\telink_b91m_driver_src\chip\B91\drivers\lib" -Wl,-Map,"AES_Demo.map" -O2  -fpack-struct -fshort-enums -g3 -mcpu=d25f -ffunction-sections -fdata-sections -mext-dsp -mabi=ilp32f -
fmessage-length=0 -fomit-frame-pointer -fno-strict-aliasing -fshort-wchar -fuse-ld=bfd -Wno-gnu-zero-variadic-macro-arguments -o "AES_Demo.elf"  ./vendor/common/flash/puya_common.o
./vendor/common/flash/puya_flash_scratch.o ./vendor/common/flash/puya_flash_trim.o  ./vendor/common/calibration/calibration.o  ./vendor/common/plic_isr.o ./vendor/common/printf.o
./vendor/AES_Demo/app.o ./vendor/AES_Demo/main.o  ./drivers/lib/src/pke/eccp_curve.o ./drivers/lib/src/pke/ecdh.o ./drivers/lib/src/pke/ecdsa.o ./drivers/lib/src/pke/pke.o
./drivers/lib/src/pke/pke_common.o ./drivers/lib/src/pke/x25519.o  ./drivers/lib/src/emi.o ./drivers/lib/src/plic.o ./drivers/lib/src/pm.o ./drivers/lib/src/rf.o ./drivers/lib/src/swire.o
./drivers/lib/src/sys.o ./drivers/lib/src/trng.o  ./drivers/flash/flash_mid146085.o ./drivers/flash/flash_mid156085.o ./drivers/flash/flash_mid166085.o  ./drivers/adc.o ./drivers/aes.o
./drivers/analog.o ./drivers/audio.o ./drivers/clock.o ./drivers/flash.o ./drivers/gpio.o ./drivers/i2c.o ./drivers/lpc.o ./drivers/mdec.o ./drivers/npe.o ./drivers/pwm.o ./drivers/s7816.o
./drivers/spi.o ./drivers/stimer.o ./drivers/timer.o ./drivers/uart.o ./drivers/usbhw.o  ./common/usb_dbg/myudb_usb.o ./common/usb_dbg/myudb_usbdesc.o  ./common/bt_debug/dbgport.o
./boot/cstartup_flash.o ./boot/cstartup_ram.o  -ldriver -laxon_driver_lib
/cygdrive/c/IDERelease/20220620_V1.0/eclipse/RDS/V3.2.3/toolchains/nds32le-elf-mculib-v5f/bin/../lib/gcc/riscv32-elf/7.4.0/../../../../riscv32-elf/bin/ld.bfd: cannot open linker script file
../link/flash_boot.link: No such file or directory
collect2: error: ld returned 1 exit status
make: *** [makefile:71: AES_Demo.elf] Error 1
"make -j4 all" terminated with exit code 2. Build might be incomplete.

10:54:59 Build Failed. 1 errors, 0 warnings. (took 31s.427ms)
```

Figure 8.1: Link error

To resolve this problem, users can open the project property, then navigate to linker's `General` page of `Tool Settings` tab as following image, then double click the link script file:

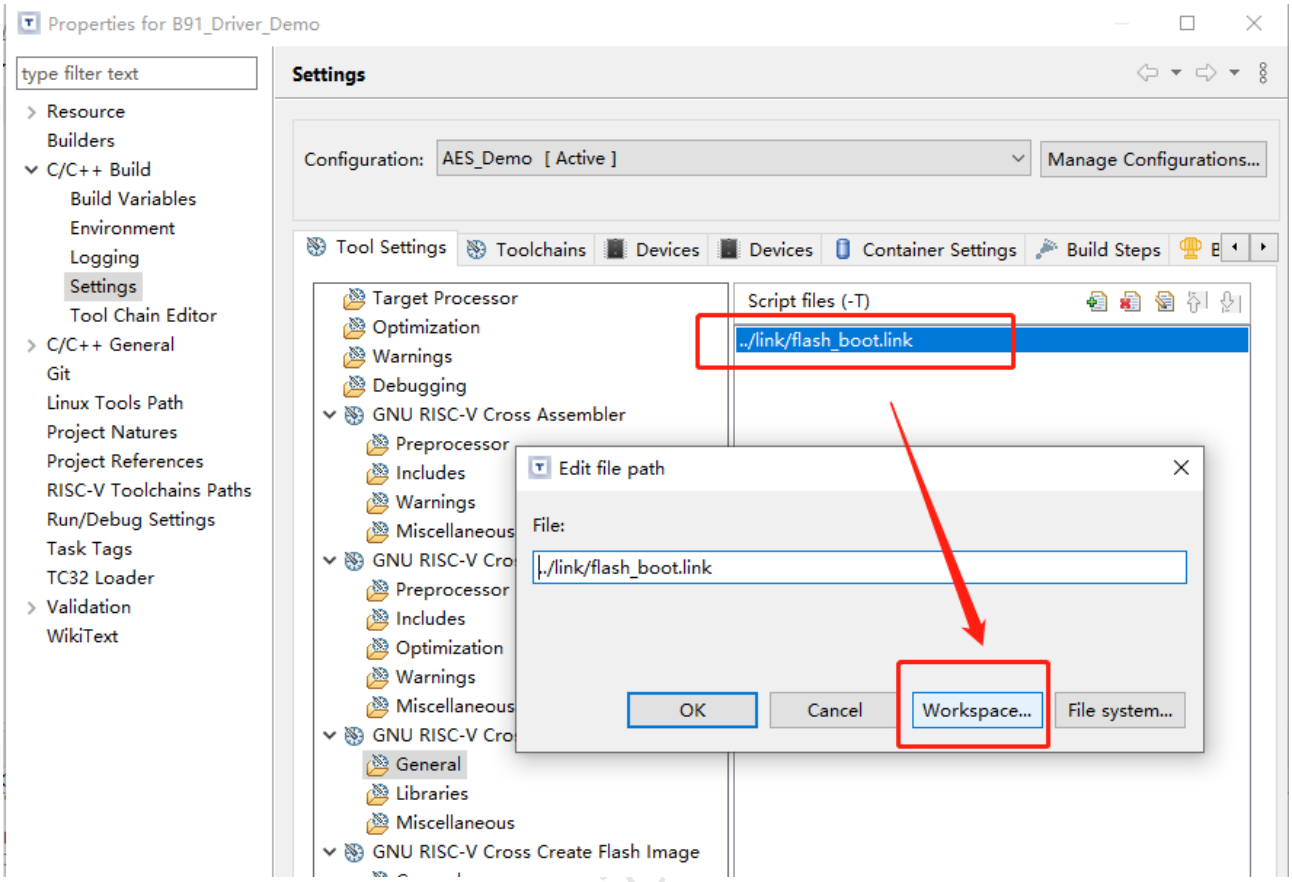

Figure 8.2: Link file path

On opened dialog, select it from `workspace` :

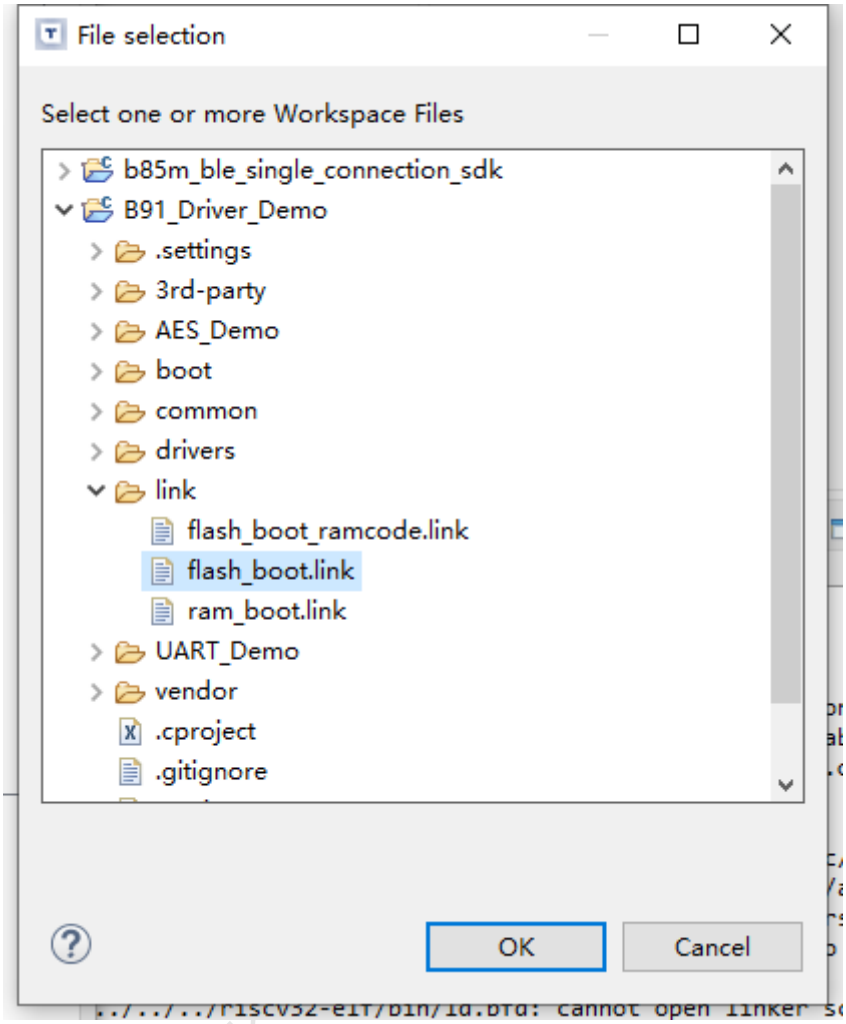

Figure 8.3: Link file select

Click `OK` and `apply and close` button, and rebuild the project to check the results.

9 Other plugins or functions

9.1 Telink Formatter

This plugin is used to format the code/file with format shortcut or saving a file with clang-format.

9.1.1 Use as formatter

To use it in formatter, users should set the `Telink Formatter` as the default formatter. Open the setting from the menu `Windows` -> `Preference` -> `C/C++` -> `Code style` -> `Formatter` , select the `TelinkFormatter` in `Code Formatter:` list, then click the `Apply and Close` button:

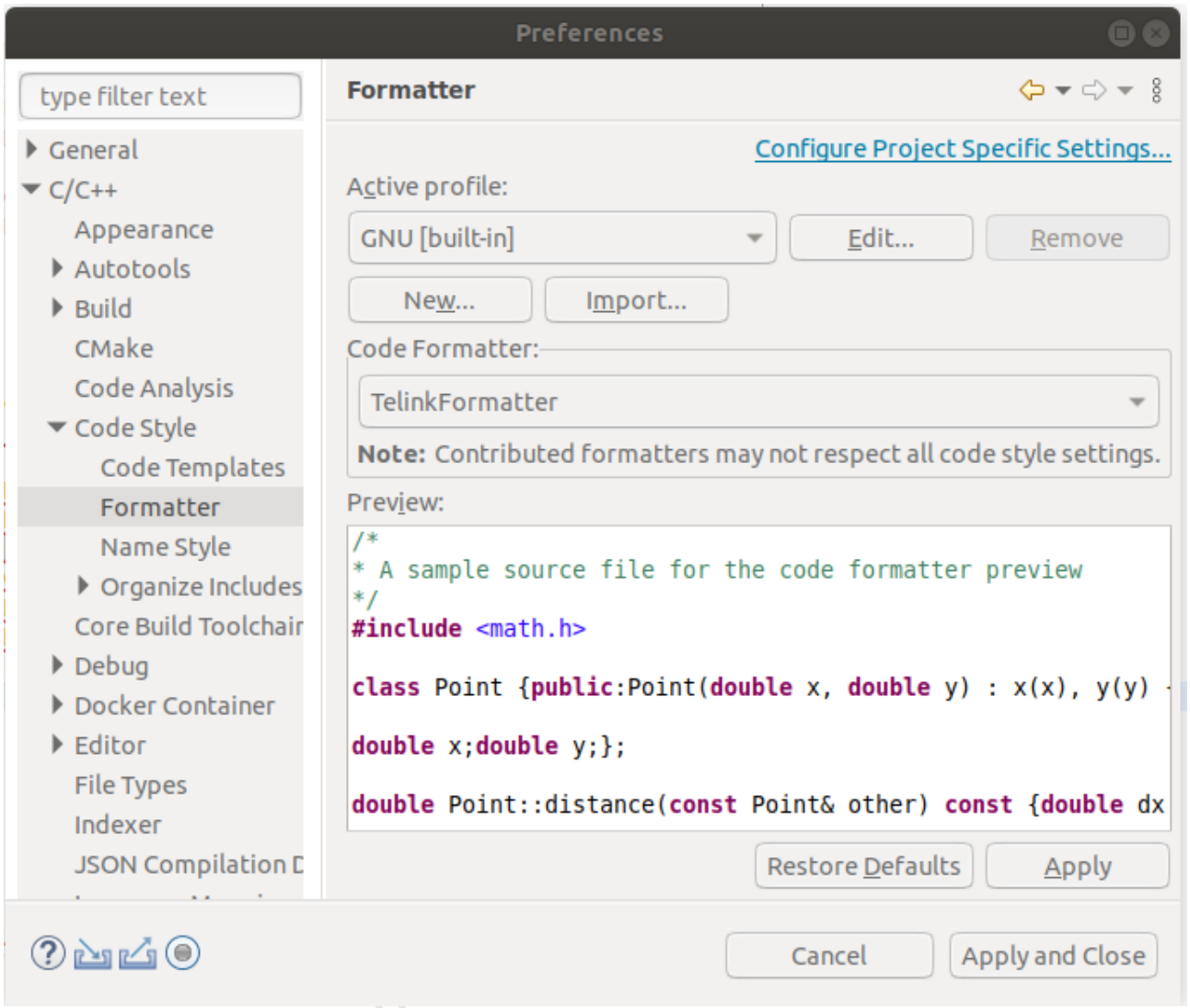

Figure 9.1: Formatter setting

Then select Telink style in the settings of `TelinkFormatter` . Open the setting from the menu `Windows -> Preference -> C/C++ -> TelinkFormatter` , select the `Telink` in `Clang-ormat fallback style:` list, then click the `Apply and Close` button:

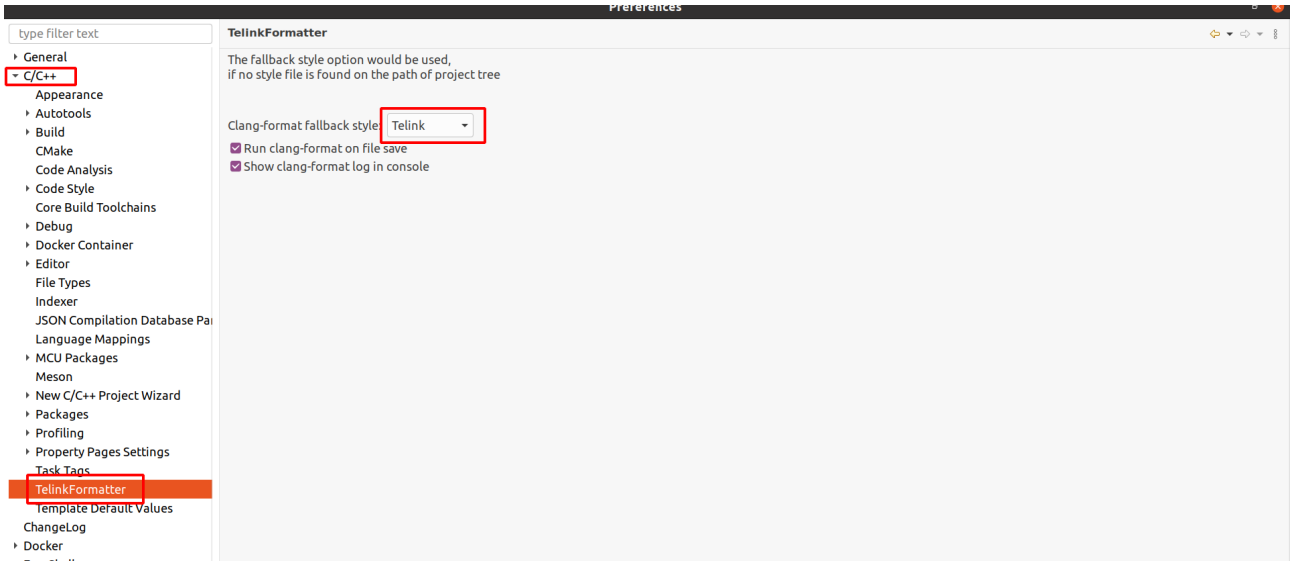

Figure 9.2: TelinkFormatter setting

Afterwards users can press the shortcut keys (by default, it's `Ctrl+Shift+F`) on any openned source code file editor, a dialog would show, uses click the corronsponding button and option to confirm:

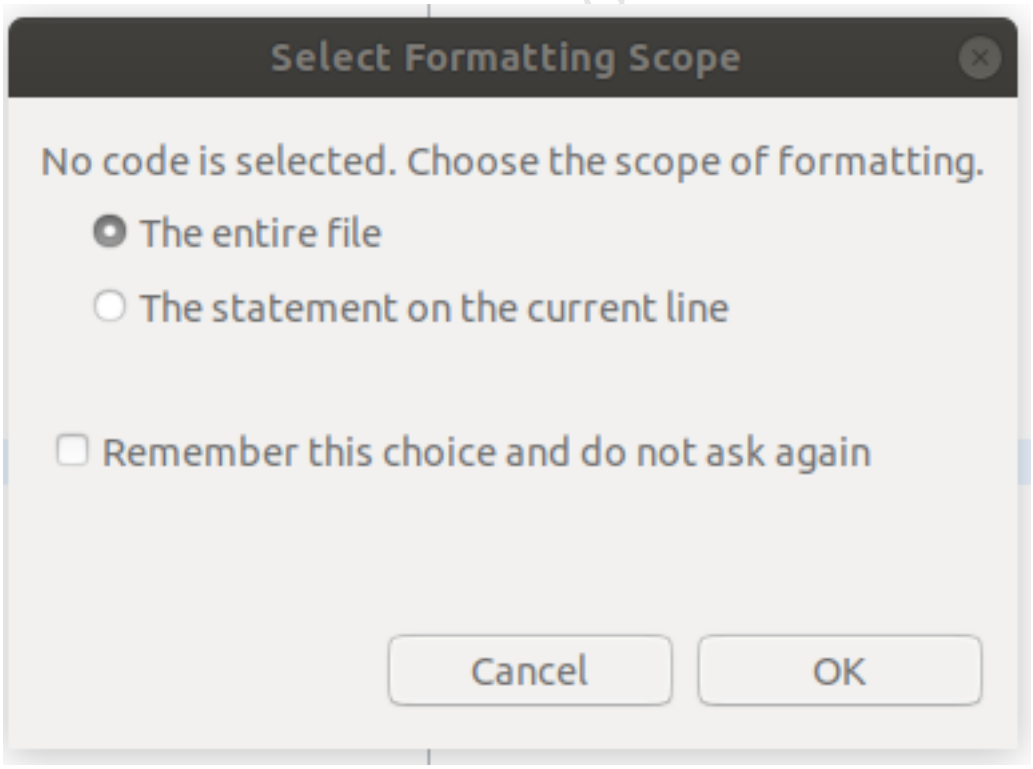

Figure 9.3: Formatter hint

After the `Telink Formatter` has formatted the code/file, users can view the logs on the `Telink Formatter` output console for details (notice that the log function should be enabled in the `TelinkFormatter` Preference dialog, check the following guide):

```
Problems  Console ×
Telink Formatter
------------ 2024-04-18 17:48:31 Format with options ------------
        -assume-filename=/home/hexiongjun/Projects/CI/telink_b91m_driver_src/chip/B91/drivers/pwm.c
        -style=file
        -fallback-style=Google
        -offset=0
        -length=6357
===> File has been formatted in Formatter mode, you can save it now
```

Figure 9.4: Formatter console output after formmtting

9.1.2 Trigger with saving event(recommanded method)

The formatter can be triggered when a file or multiple files are saving, this requires the selection of the function in `TelinkFormat` setting(through the menu `Windows` -> `Preference` -> `C/C++` -> `TelinkFormatter`):

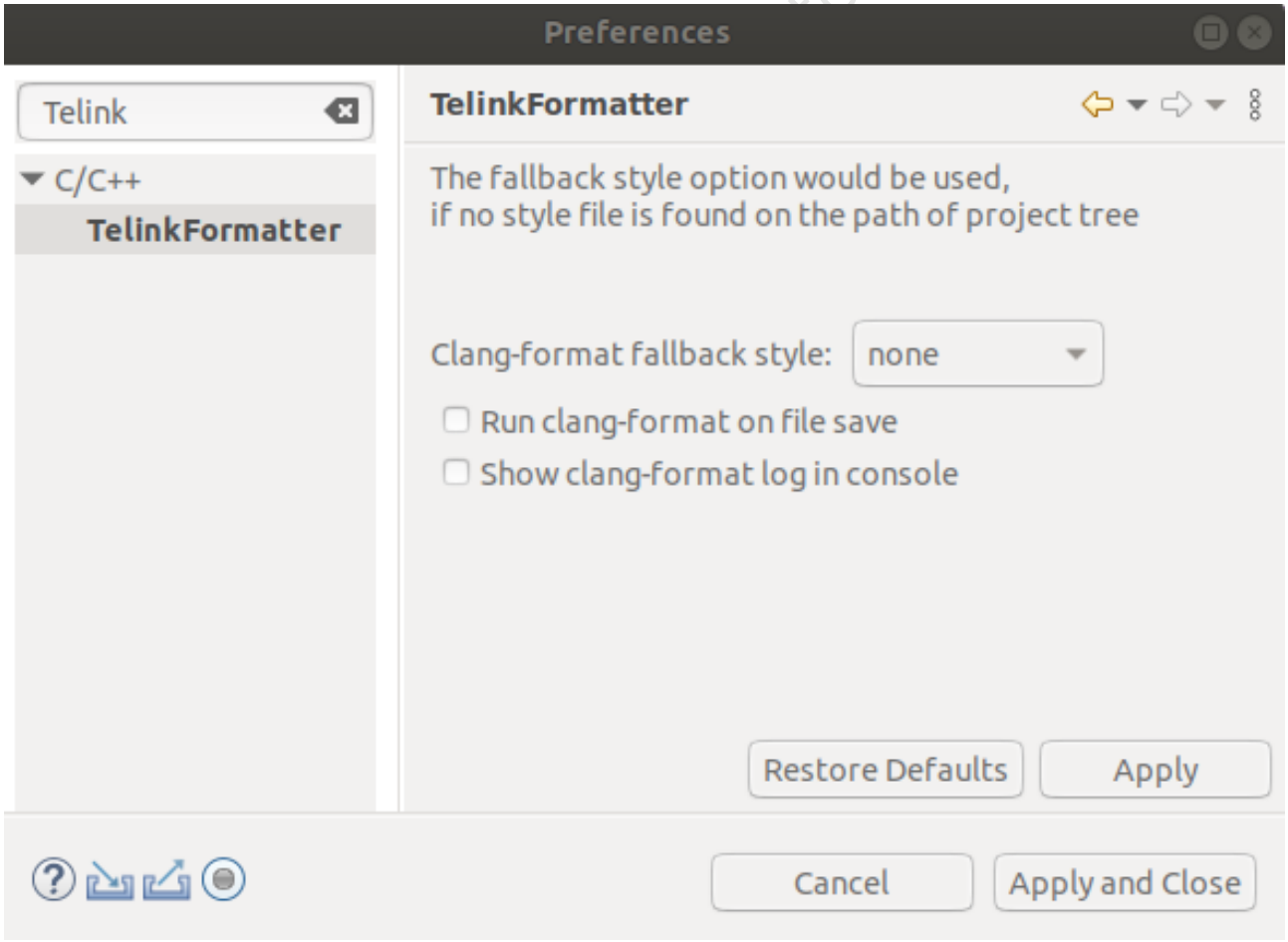

Figure 9.5: Telink Formatter Preference

After the `Telink Formatter` has formatted the code/file, users can view the logs on the `Telink Formatter`

output console for details if the log has been enabled in setting:

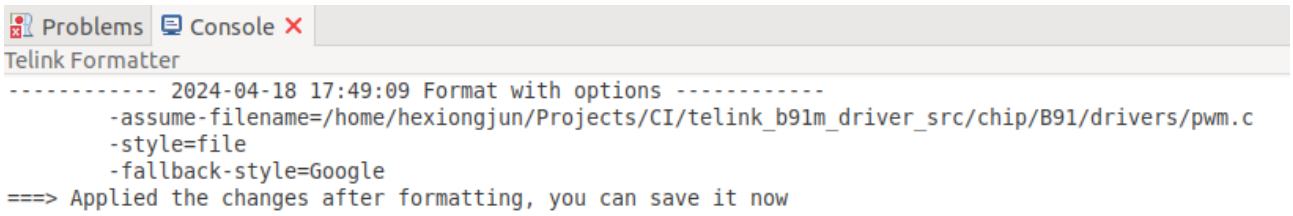

Figure 9.6: Formatter console output after saving

9.1.3 Known issue

9.1.3.1 `Save` button is disabled at the first time

The "Save" button is int disabled state when a file is modified at the first time. After save it using "CTRL+S" (or `File` -> `Save` , or with `Save all` menu-item/button), modify this file again, it will become enabled state.

9.2 Easy Shell

This plugin allows to open a shell window or file manager from the pop-up menu in the navigation tree or editor view. Additionally, it is possible to run selected files in the shell, copy file or directory path or run user defined external tools.

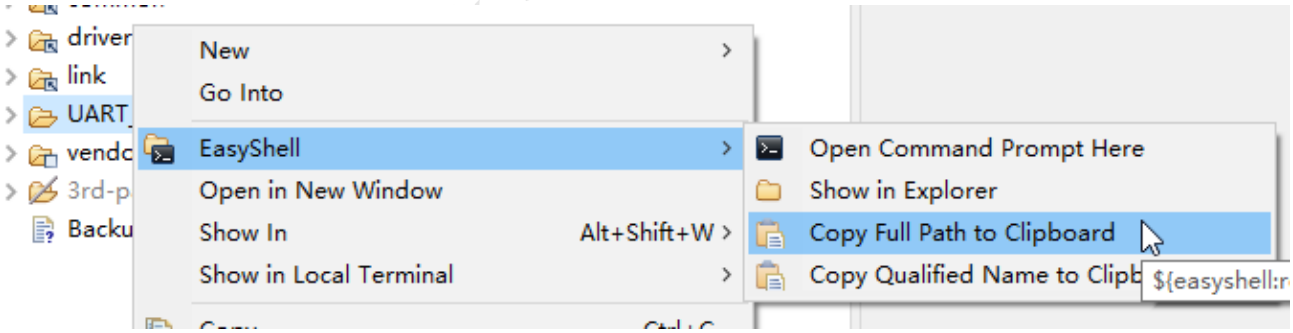

Figure 9.7: Easy Shell

9.3 ECalculator

A calculator with lots of functions, including:

- Display of the calculation in base-n numbers
- Standard, scientific and trigonometric calculation
- Enter arithmetic expressions naturally

Users can open it from the menu: `Window` -> `Show view` -> `other...` , then input `Ecalculator` to search and open it:

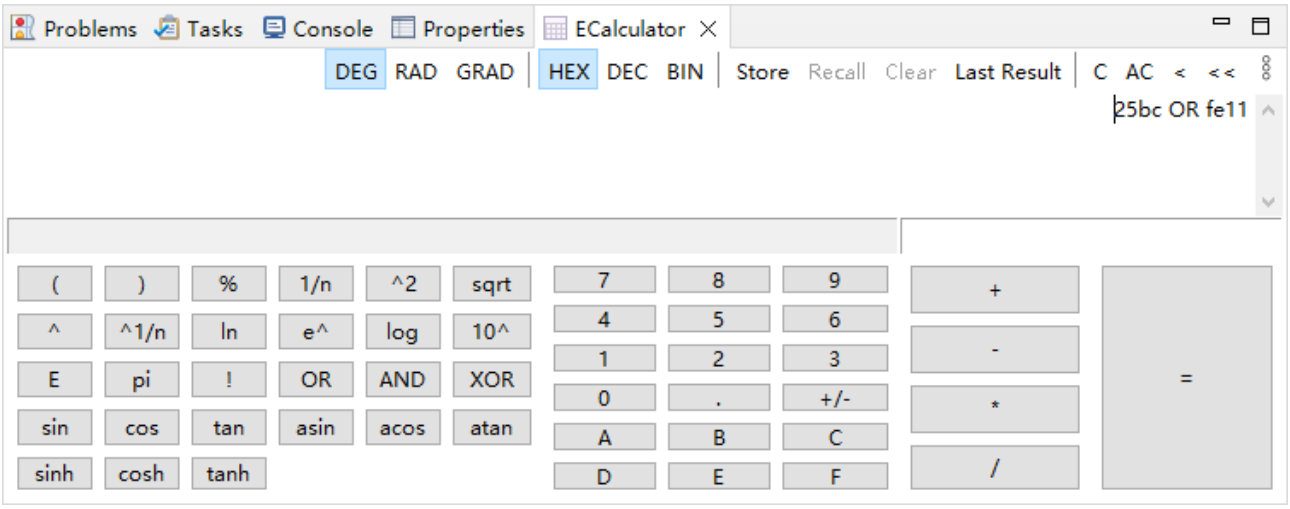

Figure 9.8: ECalculator

9.4 Terminal in Eclipse

The terminal can be used for inputting git commands and serial debugging.

Users can open it from the menu `Window` -> `Show View` -> `Terminal` :

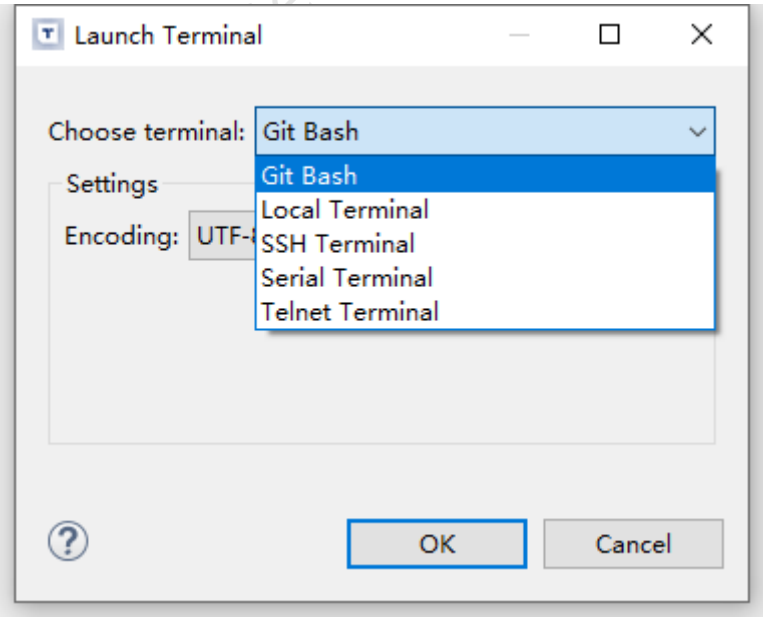

Figure 9.9: Terminal Selection

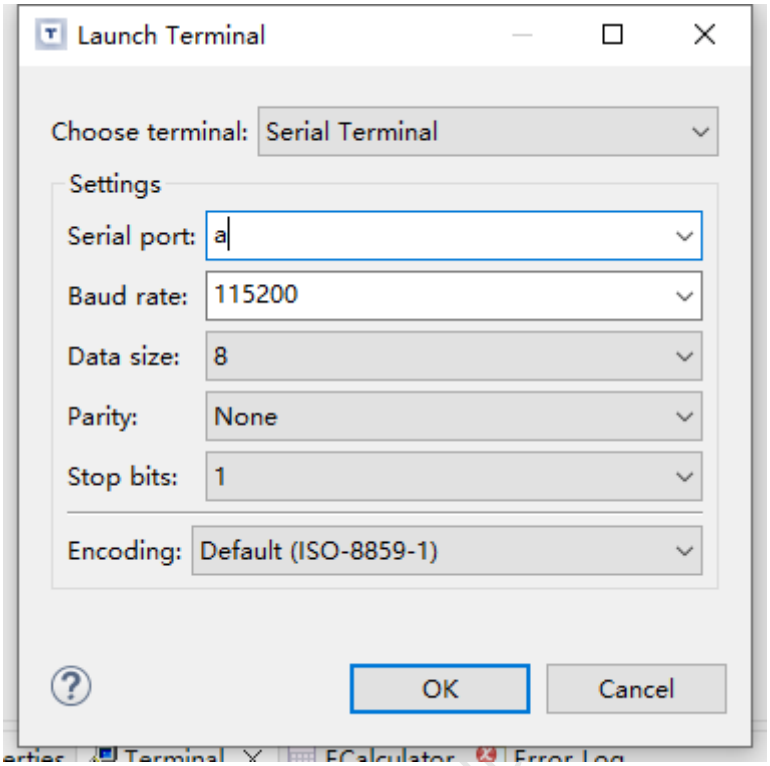

Figure 9.10: Terminal for Serial

If your serial debugging log output frequency is very fast, use other dedicated tools instead.

9.5 Binary Viewer

This plugin is used to open the binary files:

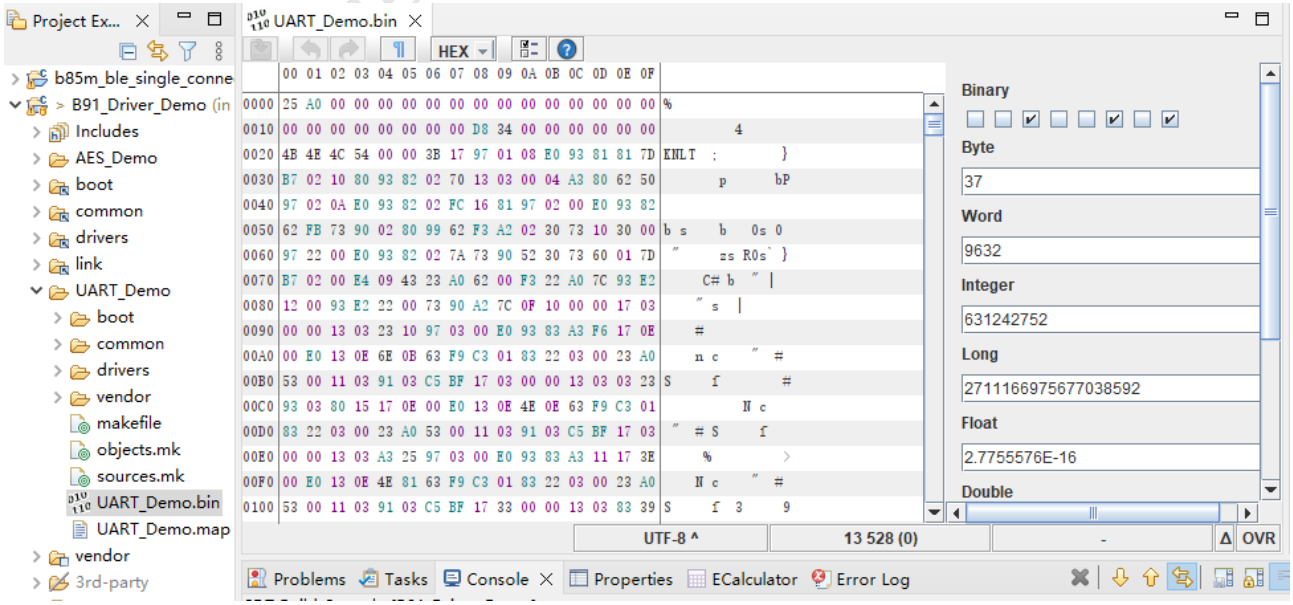

Figure 9.11: Binary Viewer

Users can double-click a binary file to open and view it in IoT Studio.

10 FAQ

10.1 `Error 127` when compiling the code

This error would give the `Error 127` at console window:

Figure 10.1: Error127

This issue is caused by the incorrect setting of the build tool version, setting the correct build tool version would fix this problem.

10.2 Not found make in PATH or nothing to build when building

These problems are normally caused by importing the Telink RDS IDE format projects to Telink IoT studio directly without converting.

To resolve this problem, remove the project and convert it to Telink IoT studio before importing.

10.3 Orphaned configuration and no options on Settings tab

When this occurs, the properties of the project would be like the following:

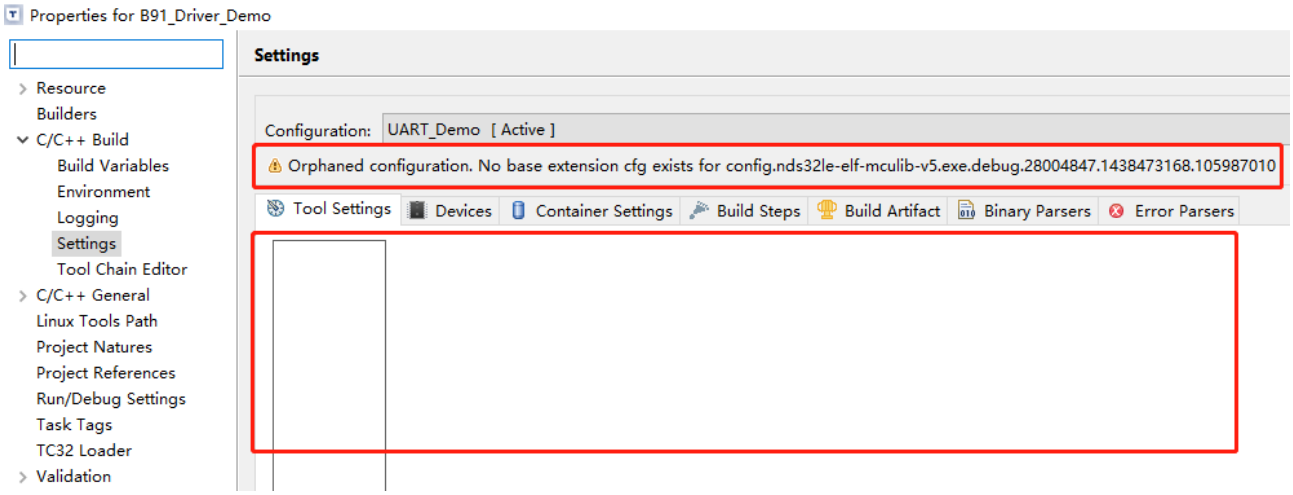

Figure 10.2: Orphaned configuration

This problem is caused by the same reason as previous FAQ, converting Telink RDS format project before importing it.

10.4 How to verify the current used toolchain gcc version for TLSR9

Users can set the following command to post command in project properties:

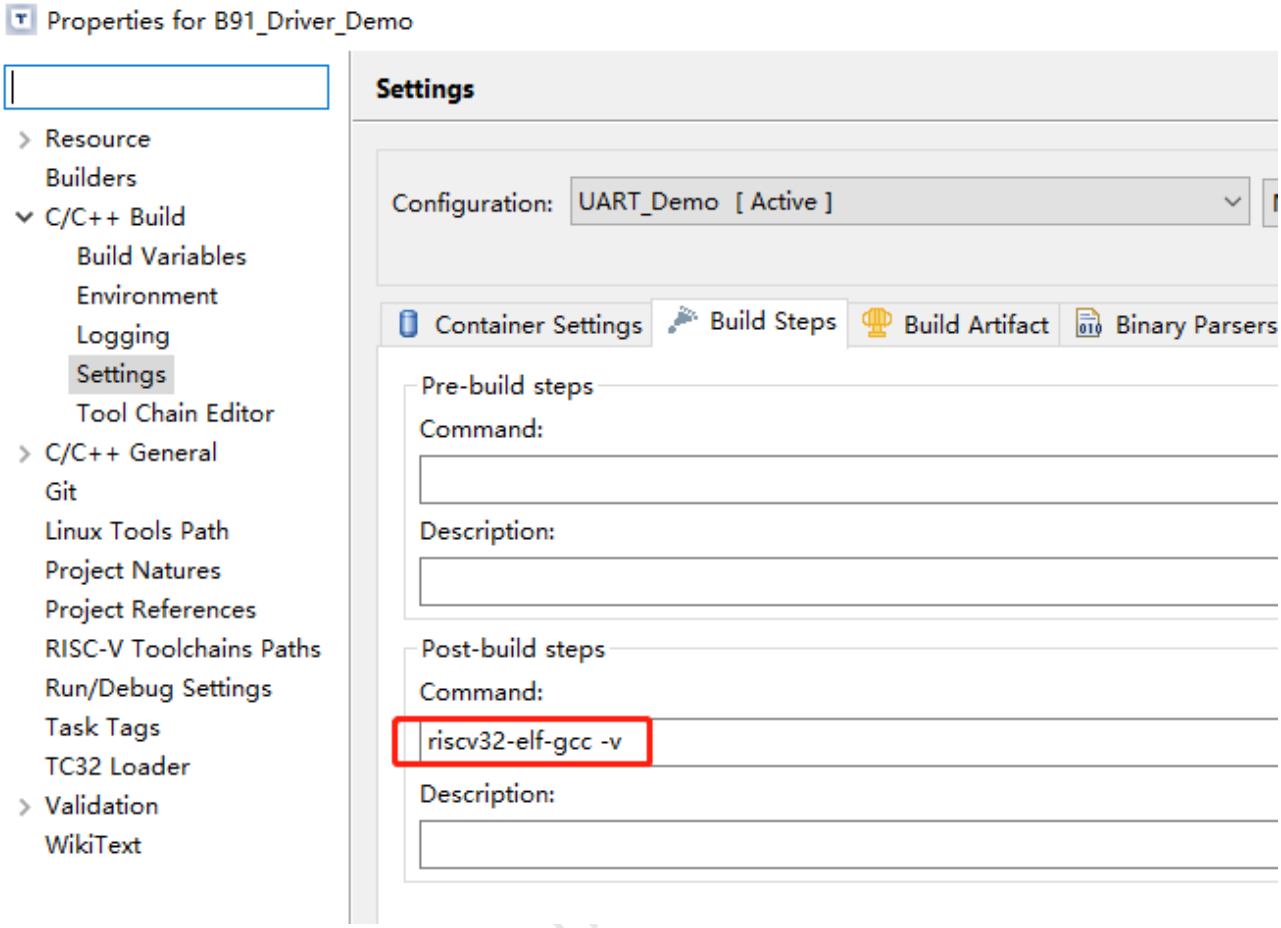

Figure 10.3: GCC version setting in Post command

Then build the project, the console will output the GCC version:

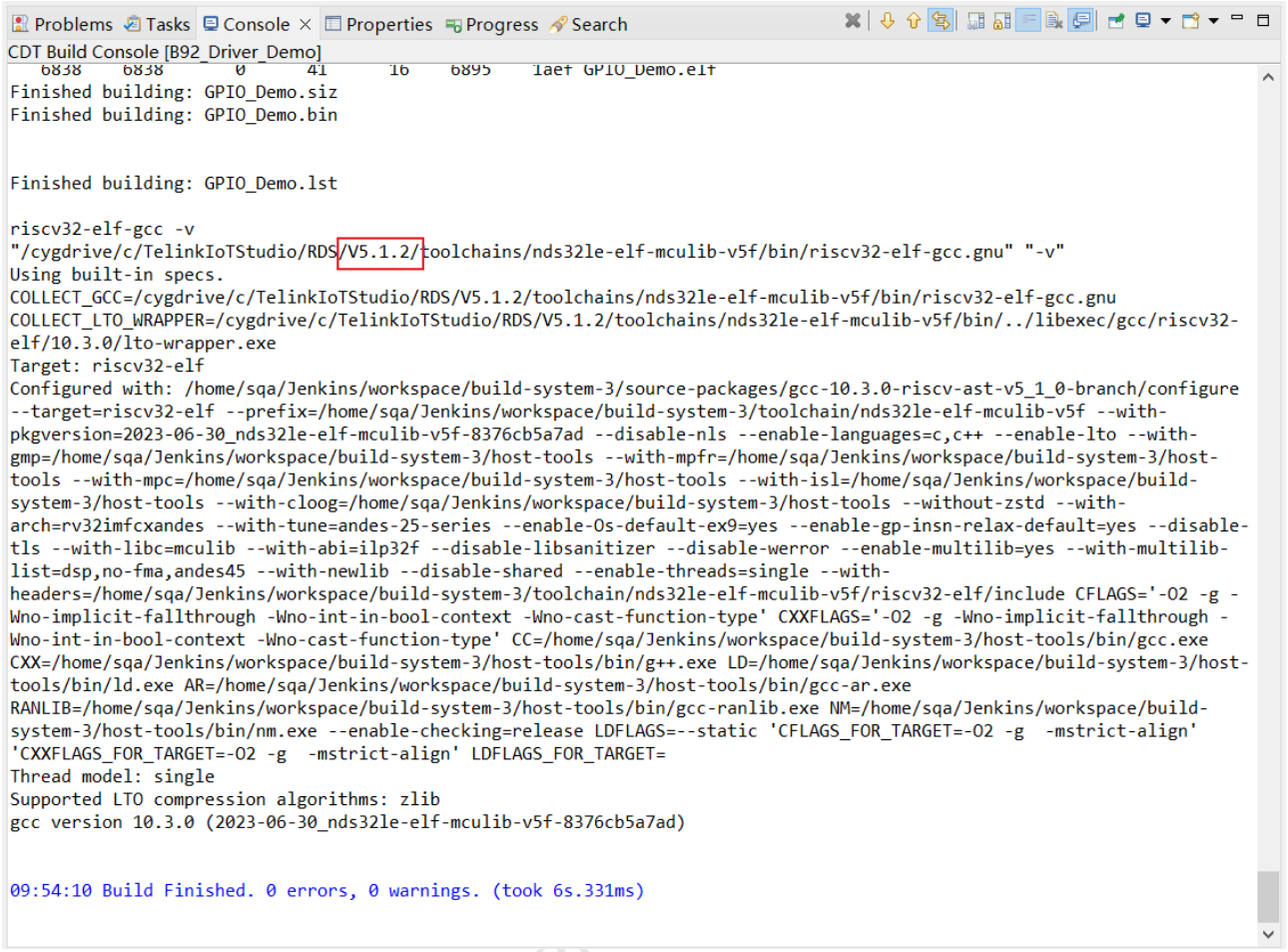

Figure 10.4: Verify the GCC version

10.5 How to verify current used `make tools`

Users can set following command (`echo "${PATH}"`) to post command in project properties:

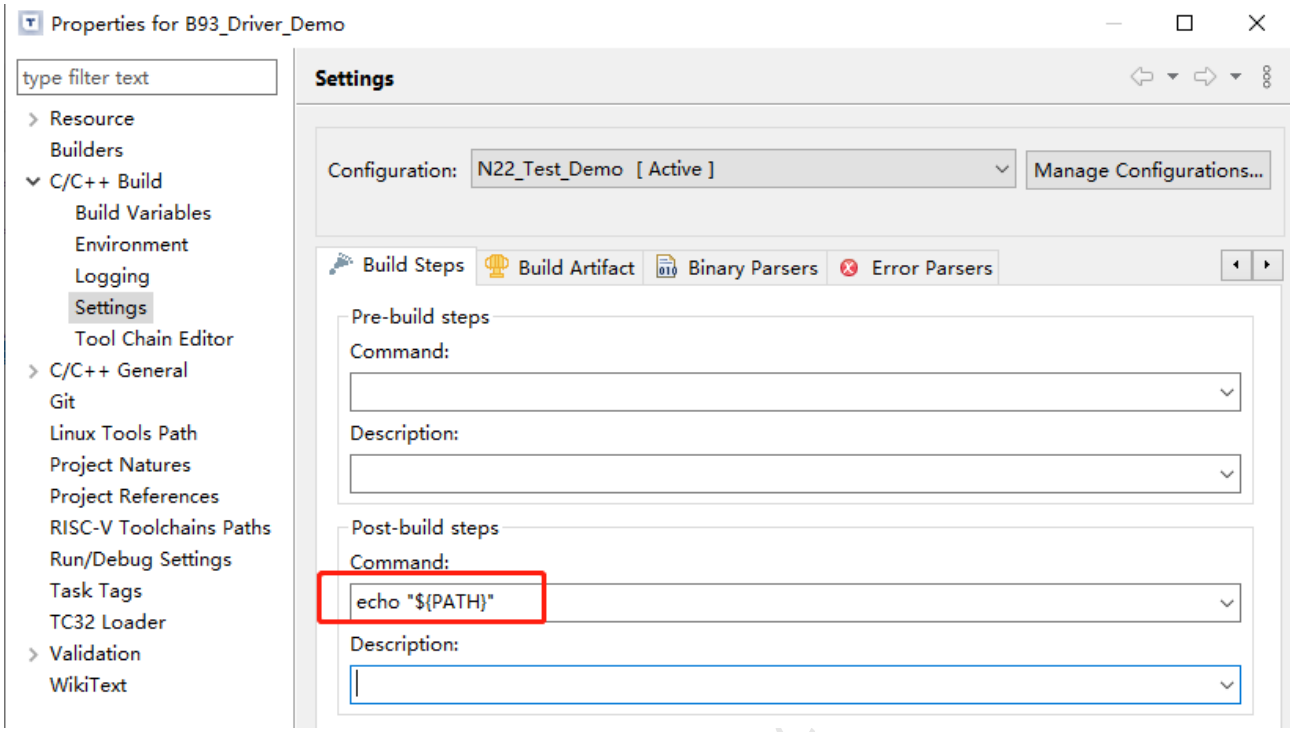

Figure 10.5: Make tool setting in Post command

Then build the project, the console will output the path that the first one is the make tool path:

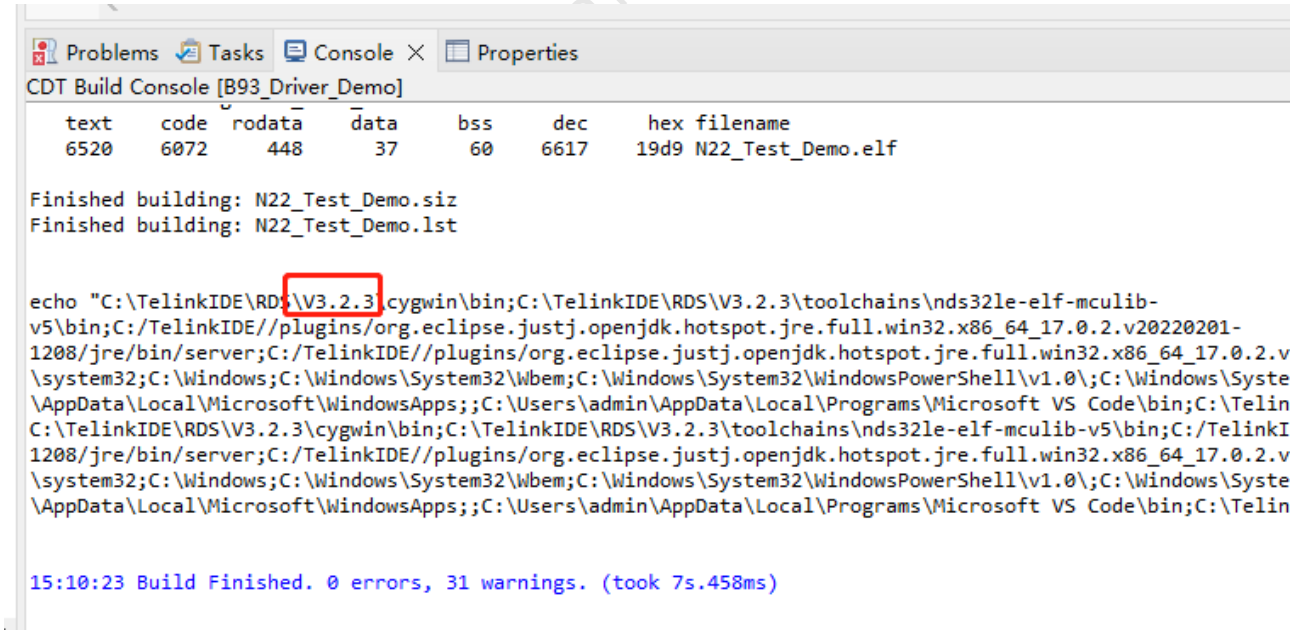

Figure 10.6: Verify the make tool path

In the above example result, the `make tool` path is `C:\TelinkIDE\RDS\V3.2.3\cygwin\bin` with base version `V3.2.3` .

10.6 The permission error occurred while opening the Telink TC32 console

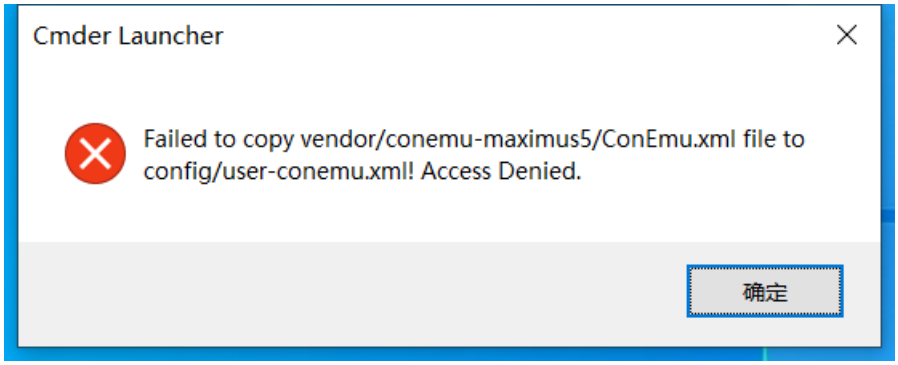

Figure 10.7: Permission problem

The user should open `Telink TC32 console` with the privileges of `administrator` .

10.7 How to enable case sensitivity feature on Windows OS

On the Windows OS, by default, file names are not case sensitive. For example, when you have a header file name of `app.h` and another file name of `App.h` , by default, Windows cannot distinguish them correctly. If you need to enable case sensitivity, you can use the following methods.

Firstly, you need to enable WSL:

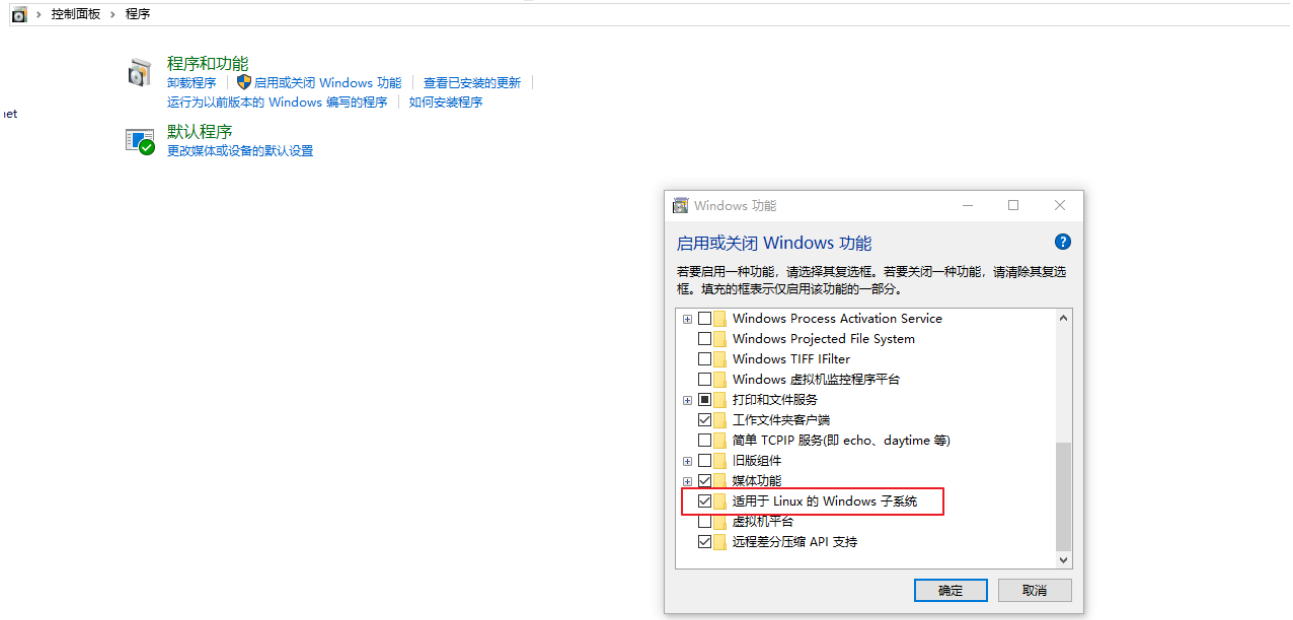

Figure 10.8: Enable WSL

Then, open Windows PowerShell as an administrator and navigate to the directory you need to use, and execute the following command:

```
(Get-ChildItem -Recurse -Directory).FullName |
ForEach-Object {fsutil.exe file setCaseSensitiveInfo $_ enable}
```

11 Known issues

11.1 `Save` button is disabled at the first time

Use the `Ctrl+S` or menu item `File` -> `Save` , or click the `Save All` button/menu-item or use `Ctrl + Shift +S` . The save button would become enabled state in the following modifications.

This is caused by `Telink Formatter` plugin.

11.2 IoT Studio exits if users click the `Open artifact path` multiple times

This occurs in that the IoT Studio cannot find the artifact path. Users should avoid clicking the toolbar icon `Open artifact path` multiple times in a short time.

11.3 IoT Studio will block for a while after importing project and changing toolchain

This is a normal case as the IoT Studio would take a while to do the `code indexing` and scan compiler files (for example the toolchain system header files and directories).

This procedure would take a few seconds. Users should wait for this job done before other operations (eg: building). This is a one-shot action, it will happen on toolchain changing settings or importing projects.

11.4 The size of generated artifacts are zero

This issue will occur if users start to build a configuration when the project is just imported and the makefile hasn't been generated fully.

When this happens, a log similar to the following example would display on the console log window:

```
riscv32-elf-gcc -mcmodel=medium -mcmodel=medium -O2 -fmessage-length=0 -ffunction-sections
-fdata-sections -flto  -g3 -T ../boot.link -nostartfiles -Xlinker --gc-sections
-L"E:\TelinkIDE\[Src]telink_eagle_ble_multi_connection_src\B91_ble_multi_conn_src\algorithm\lc3"
-Wl,-Map,"B91_demo.map" -g3 -mcpu=d25f -ffunction-sections -fdata-sections -fmessage-length=0
-fno-builtin -fomit-frame-pointer -fno-strict-aliasing -fshort-wchar -fuse-ld=bfd -fpack-struct
-O2 -o "B91_demo.elf"  ./vendor/common/hci_transport/hci_dfu.o
./vendor/common/hci_transport/hci_dfu_port.o ./vendor/common/hci_transport/hci_h5.o
./vendor/common/hci_transport/hci_slip.o ./vendor/common/hci_transport/hci_tr.o
./vendor/common/hci_transport/hci_tr_h4.o ./vendor/common/hci_transport/hci_tr_h5.o
./vendor/common/blt_common.o ./vendor/common/blt_fw_sign.o ./vendor/common/blt_led.o
./vendor/common/blt_soft_timer.o ./vendor/common/common_dbg.o ./vendor/common/custom_pair.o
./vendor/common/device_manage.o ./vendor/common/flash_fw_check.o ./vendor/common/simple_sdp.o
./common/tl_queue.o ./common/utility.o   -lm -ldsp
/cygdrive/d/TelinkIDE/RDS/V3.2.3/toolchains/nds32le-elf-mculib-v5f/bin/../lib/gcc/riscv32-elf/7.4.0/
../../../../riscv32-elf/bin/ld.bfd: warning: cannot find entry symbol _RESET_ENTRY; defaulting to
0000000020000000
Finished building target: B91_demo.elf

Invoking: GNU RISC-V Cross Create Flash Image
Invoking: GNU RISC-V Cross Create Listing
Invoking: GNU RISC-V Cross Print Size
riscv32-elf-objcopy -O binary "B91_demo.elf"  "B91_demo.bin"
riscv32-elf-size "B91_demo.elf"
riscv32-elf-objdump --source --all-headers --demangle --line-numbers --wide "B91_demo.elf" >
"B91_demo.lst"
   text    code  rodata    data     bss     dec     hex filename
      0       0       0       0       0       0       0 B91_demo.elf
```

Figure 11.1: Artifact size is 0

To resolve this problem, users should wait a few seconds then build the configuration again.

To avoid this issue, users should wait for the makefiles to be generated and the indexing to be finished. The process can be viewed at the right bottom of the IoT Studio:

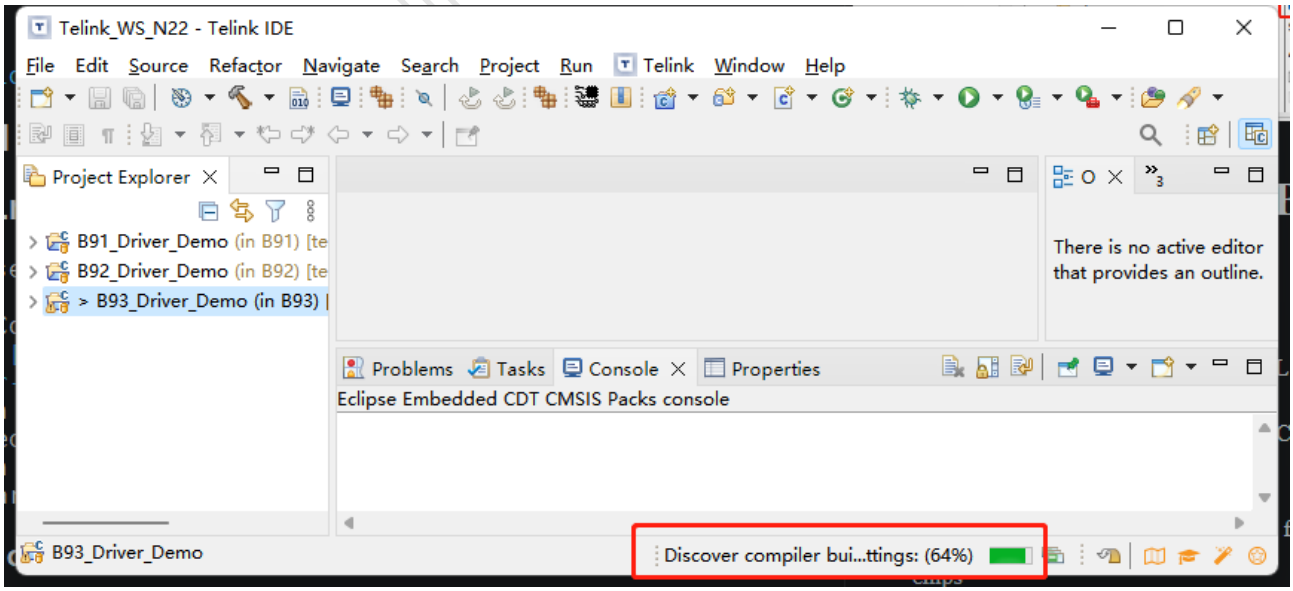

Figure 11.2: IoT Studio is busy

11.5 Clicking the telink menu is invalid

This issue will occur if users do not select any project. To avoid this issue, select a project before clicking the telink drop-down menu.

11.6 The problem that the source file cannot be found in debugging

In the process of debugging, there may be an error that the source file cannot be found. At this time, click Edit Source Path... , as shown below:

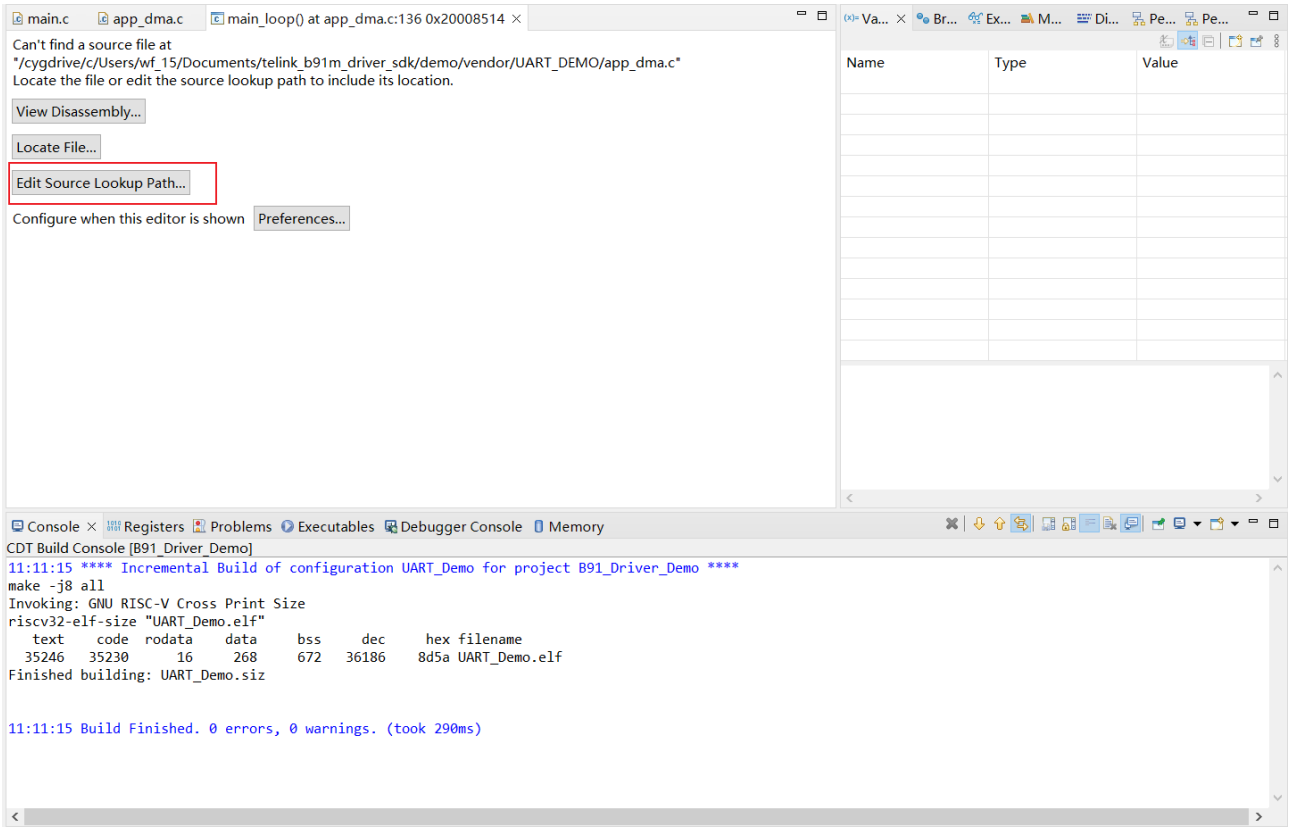

Figure 11.3: locate

Then follow the instructions in Chapter Edit Path Mapping to configure path mapping

12 IoT Studio settings

12.1 Hide print margin

The print margin will show in default, which is a vertical line in the code editor:

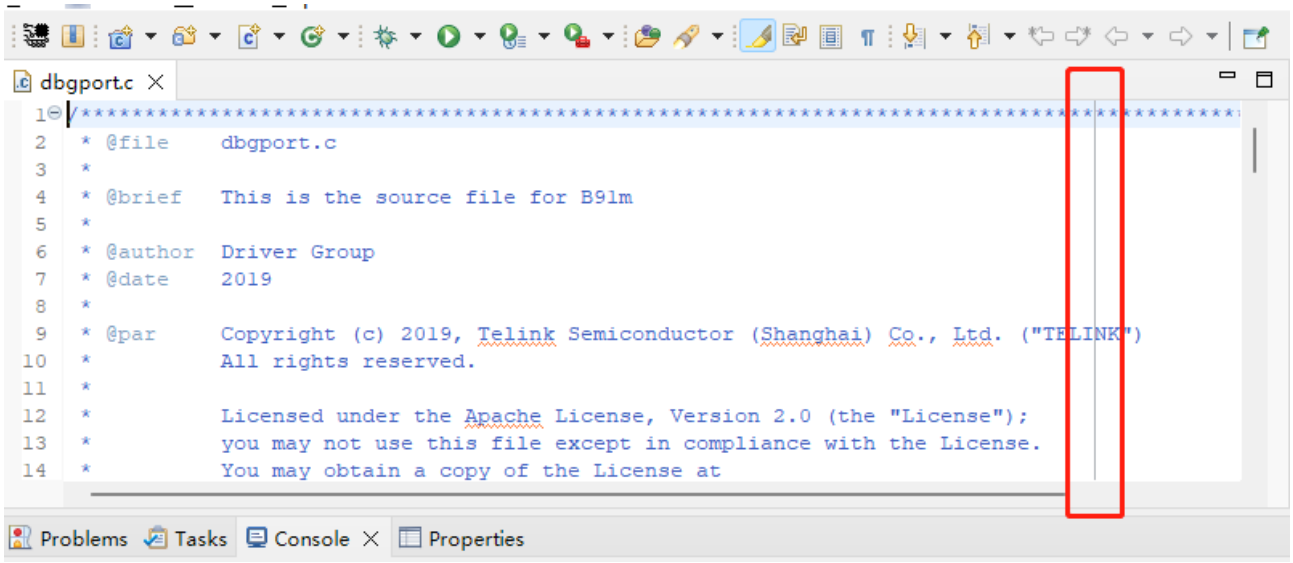

Figure 12.1: Print margin in text editor

If you want to hide it, you can uncheck the `Show print margin` option on the `Text Editors` page of preferences.

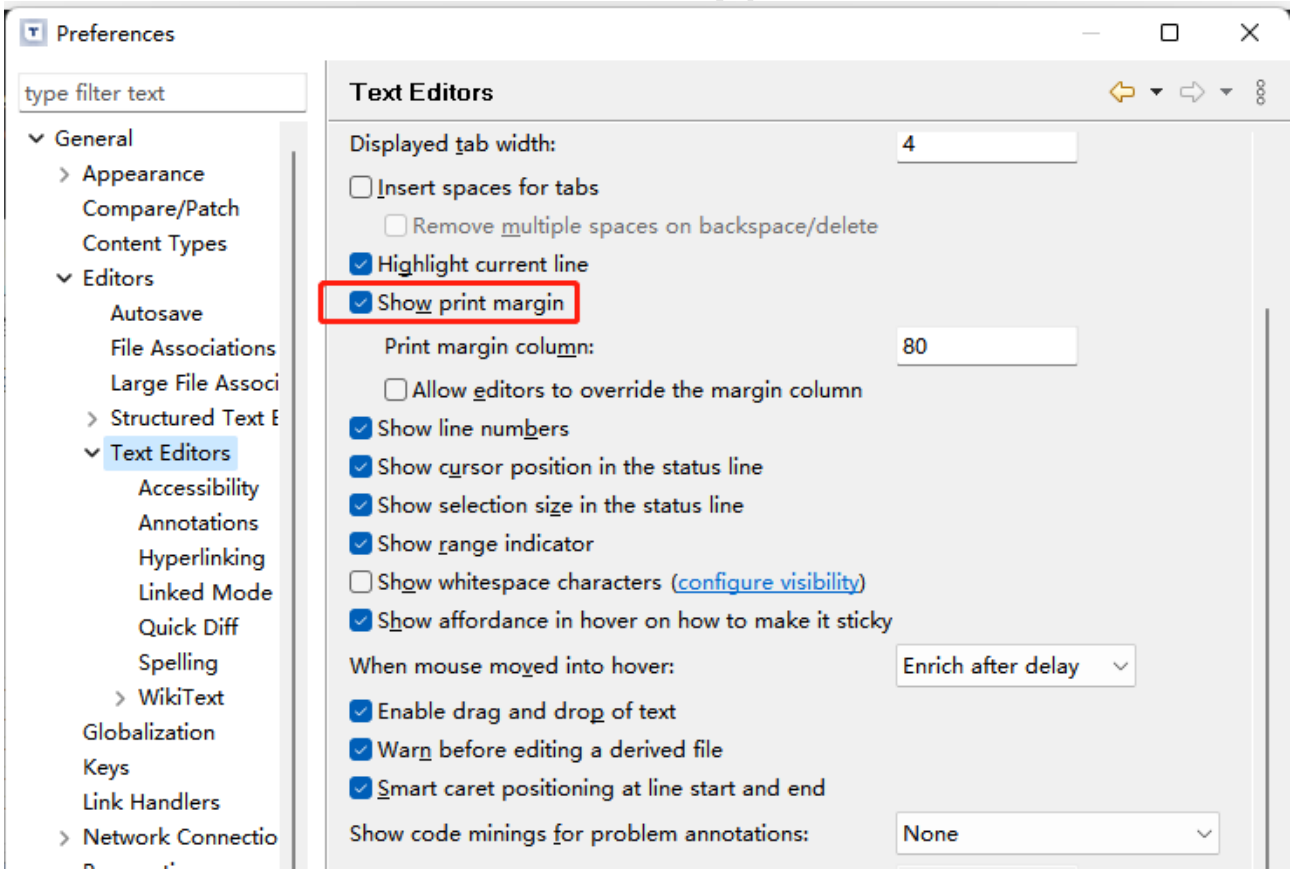

Figure 12.2: Configure the print margin

13 IoT Studio Debug Tool For TLSR9

Note: This jtag debug tool is only used for TLSR9 series chips

13.1 Build debuggable programs

To use the debugging function, when building an executable file, add the -g option to the compilation options, and pay attention to check the compiler and linker parameters, and delete all the -O options.

And the optimization option should be selected as -Og, and cannot be set to -O0, because in the test, it is found that the program compiled with the -O0 option cannot run normally after downloading, and the following error will occur when trying to download again:

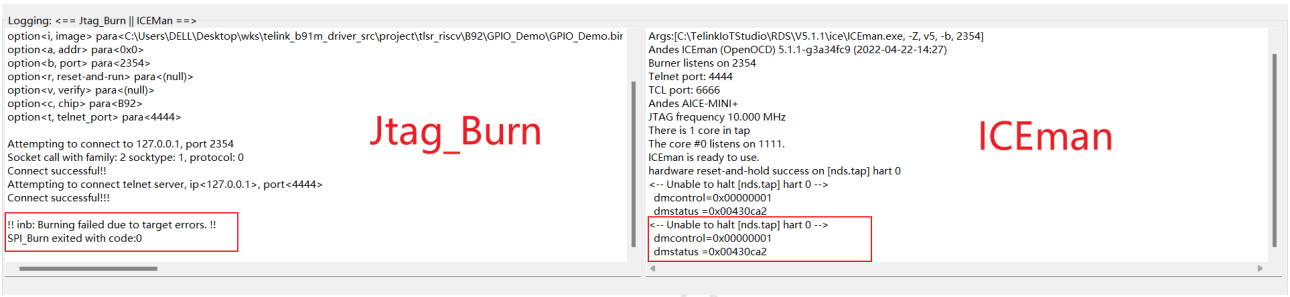

Figure 13.1: Jtag Burn err

In the process of building the program, you may encounter syntax errors like the one shown in the figure below. The solution is to delete options such as -Wall -Werror of the compiler (also pay attention to checking the parameters of assembler and linker):

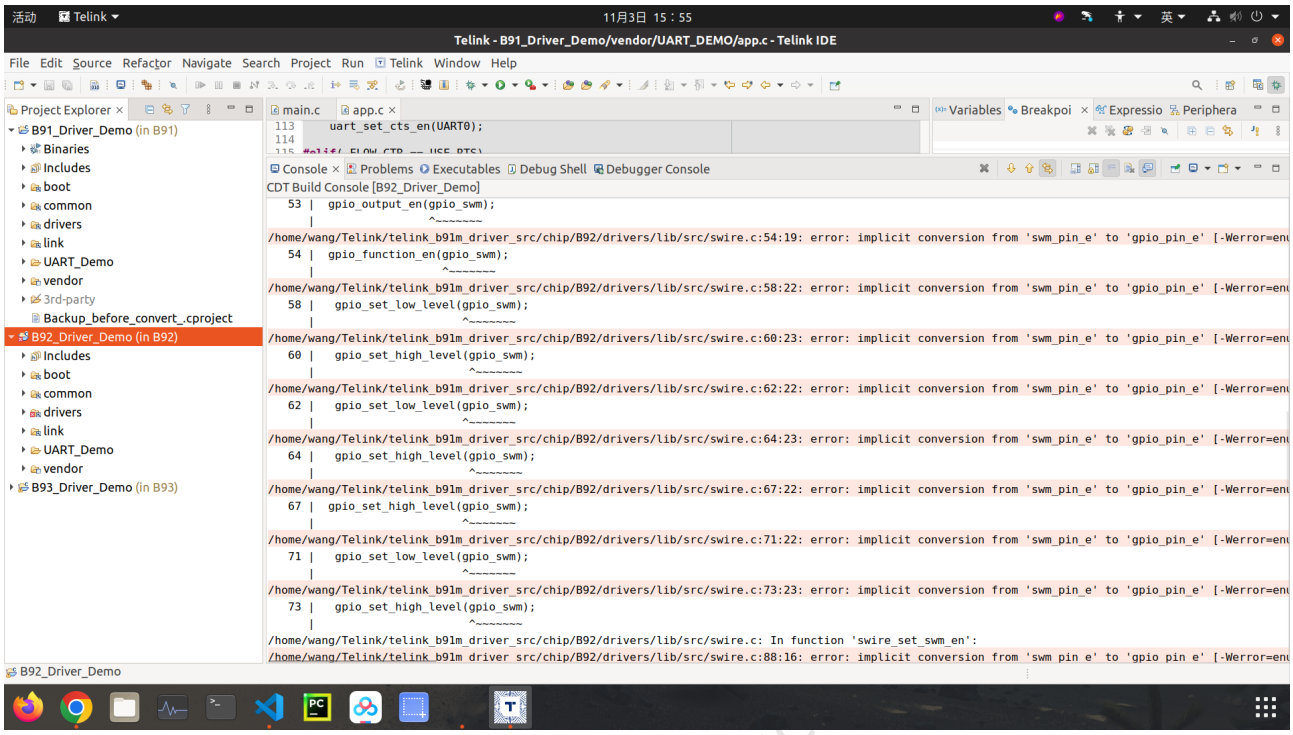

Figure 13.2: syntax err

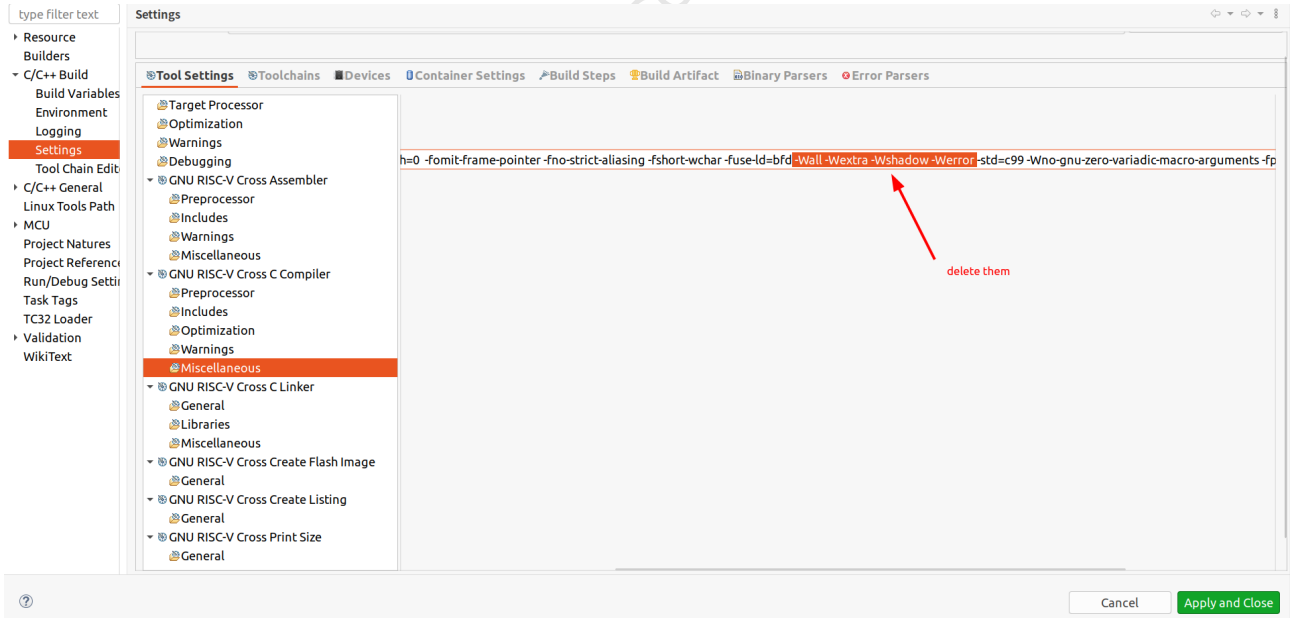

Figure 13.3: delete option

One thing to note is that when building the V5.1.2 N22 program, because of the tool chain, when using the GCC 10 version of the N22 tool chain, delete the -mabi=ilp32f option (also pay attention to check the parameters of assembler and linker).

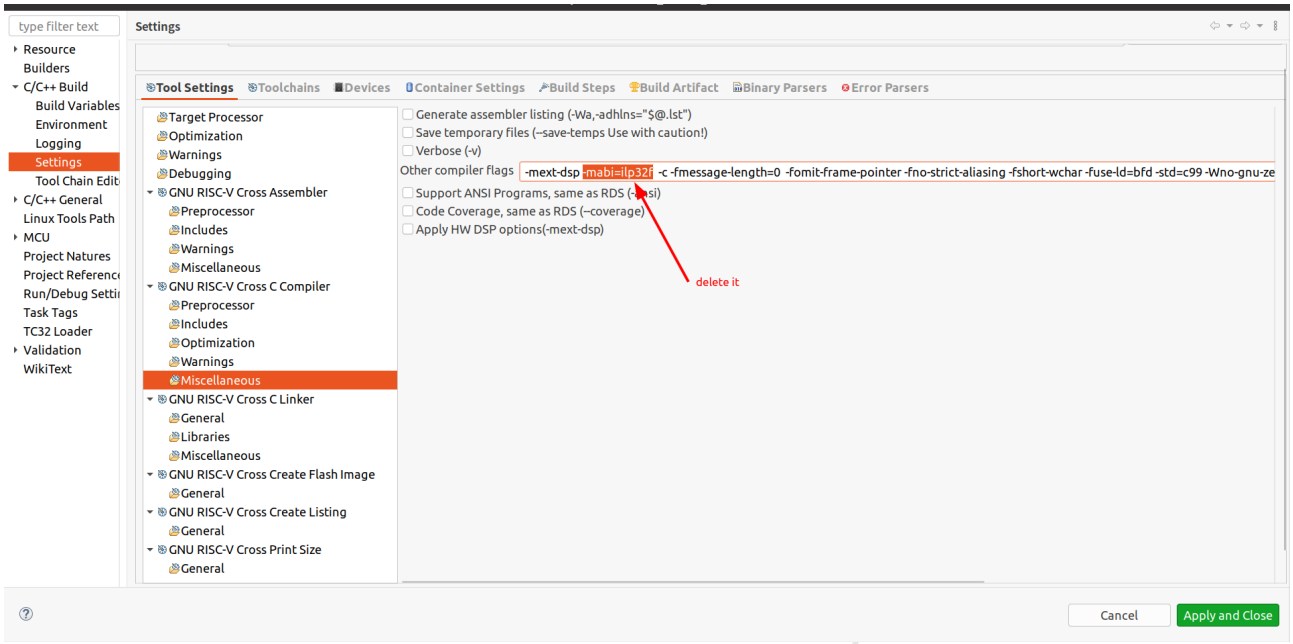

Figure 13.4: delete_mabi

13.2 Burn program

When burning the program, pay attention to burner's path and chip type, and then click Start ICEman. After ICEman is ready, the telnet_port and burner_port should be consistent with ICEman's, then click Burn, as shown below:

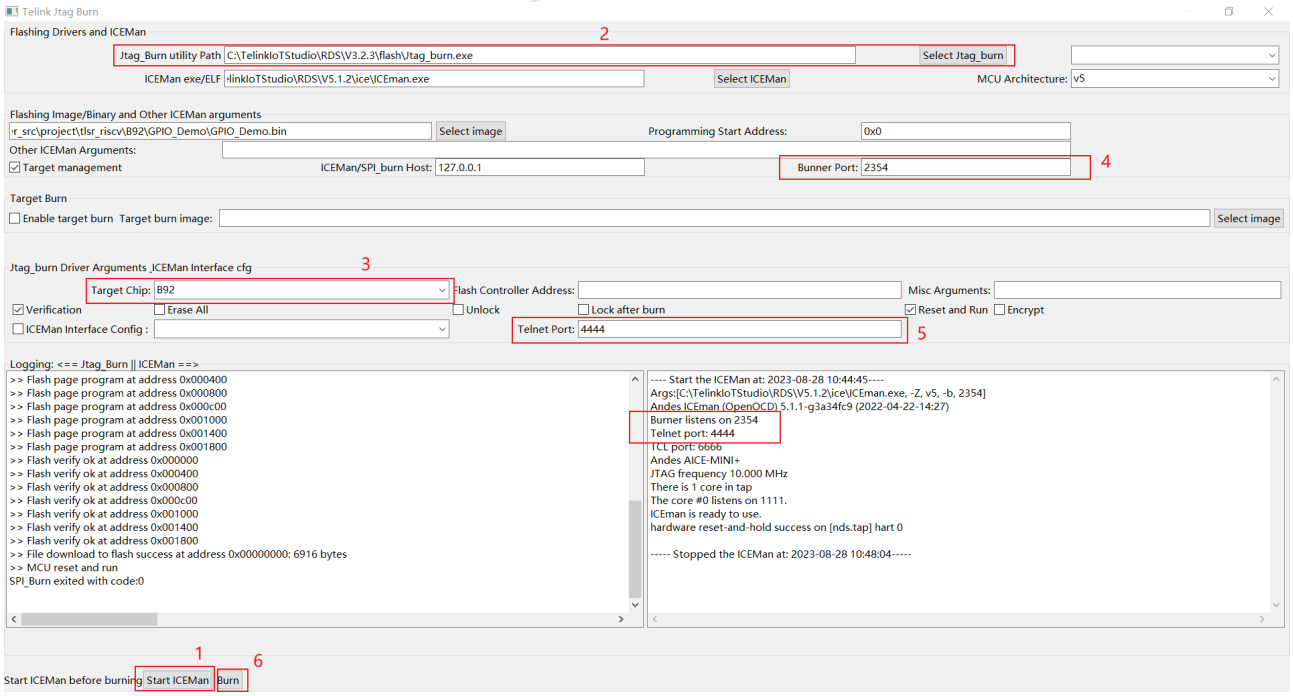

Figure 13.5: Jtag burn

Jtag_ Burn is located in the $IoTStudio_PATH/RDS/V3.2.3/flash/directory, and its help documents are also in the same directory.

13.3 Telink ICEman GDB Debugging (Recommended)

Before starting debugging, you must ensure that the elf file you want to use is selected, as shown in the figure.

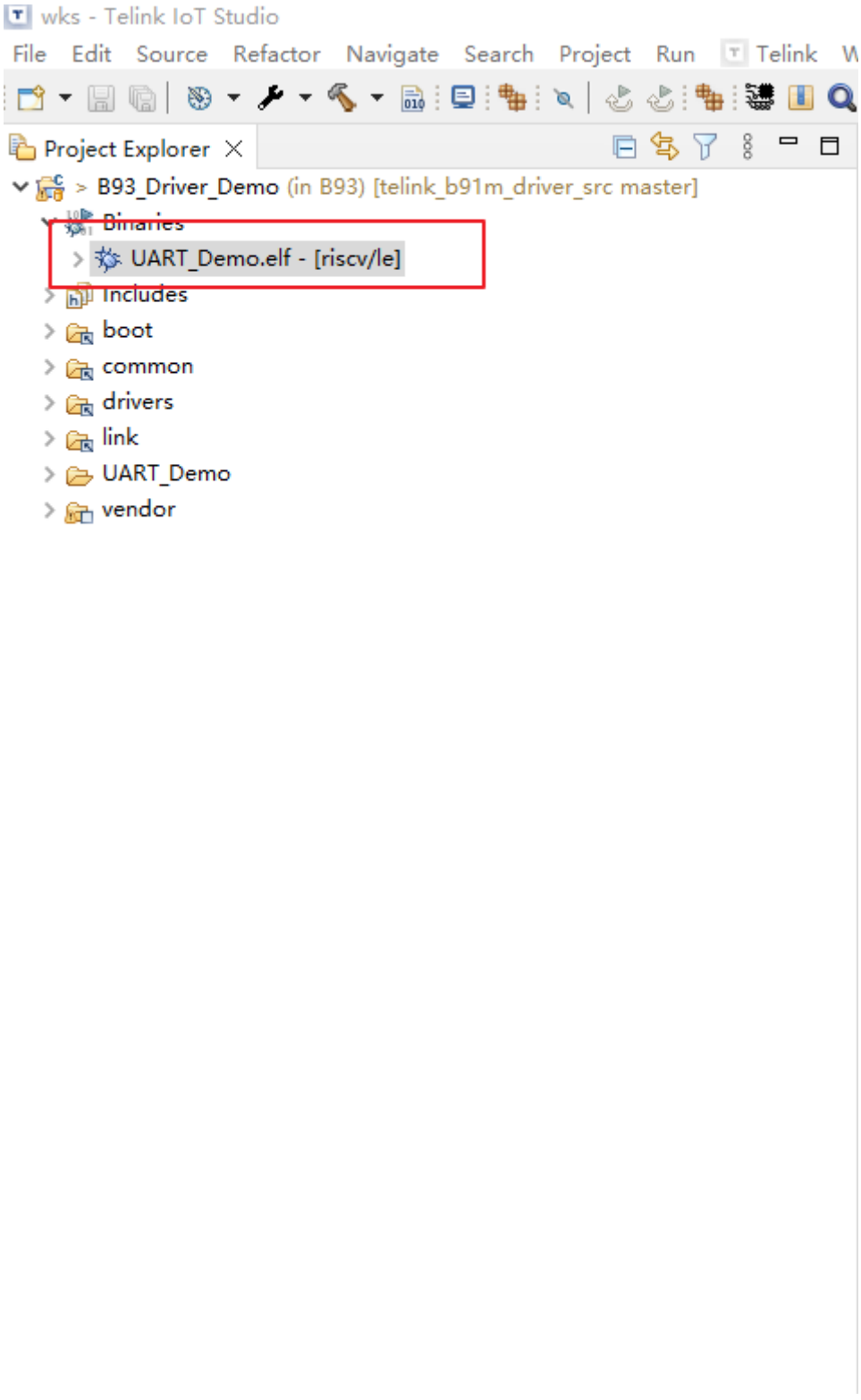

Figure 13.6: Select ELF

Then click the drop-down arrow of the Debug icon on the toolbar and select Debug Configuration.

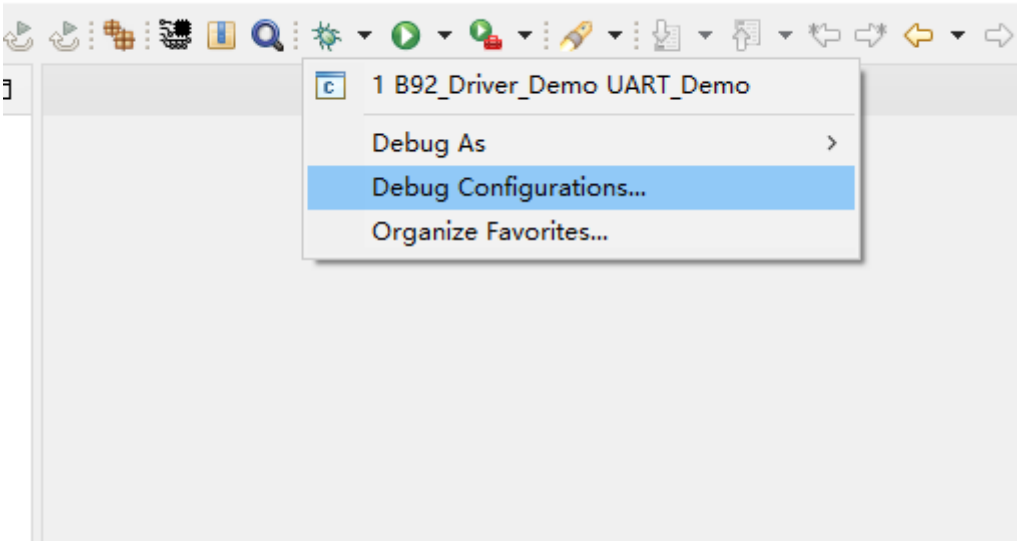

Figure 13.7: debug

Double clicking on Telink ICEMan GDB Debugging, and a default debug configuration will be generated. You can choose whether you need an ICEman interface in the Debugger tab, as shown in the figure.

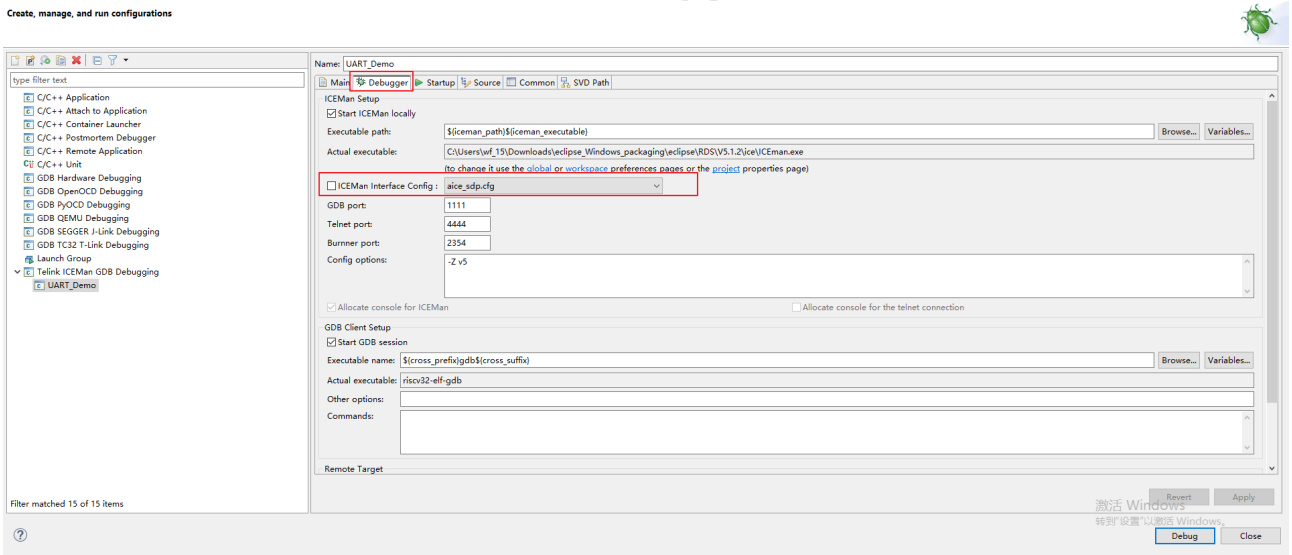

Figure 13.8: ICEman interface

13.4 View peripheral registers through SVD file

Users can select the correct chip type in the SVD tab, and the IDE will find the required SVD file on its own. Users can also customize the path of SVD files, but it is not recommended.

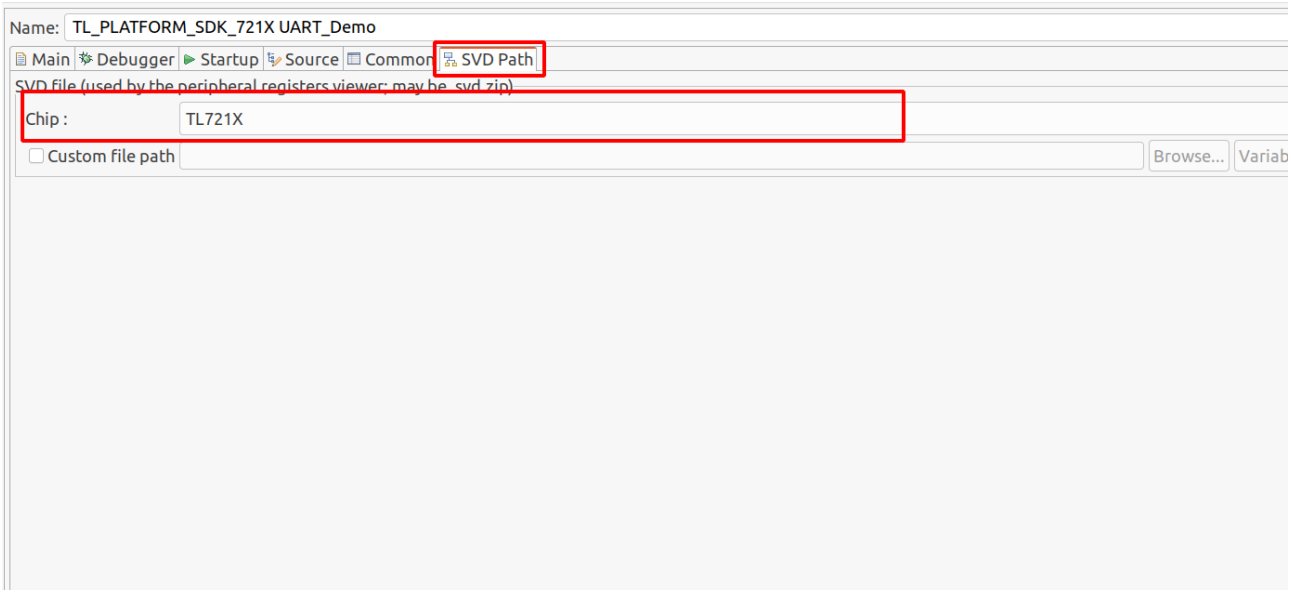

Figure 13.9: SVD Tab

In the Debug view, users can select the peripherals they want to view through the Peripheral tab.

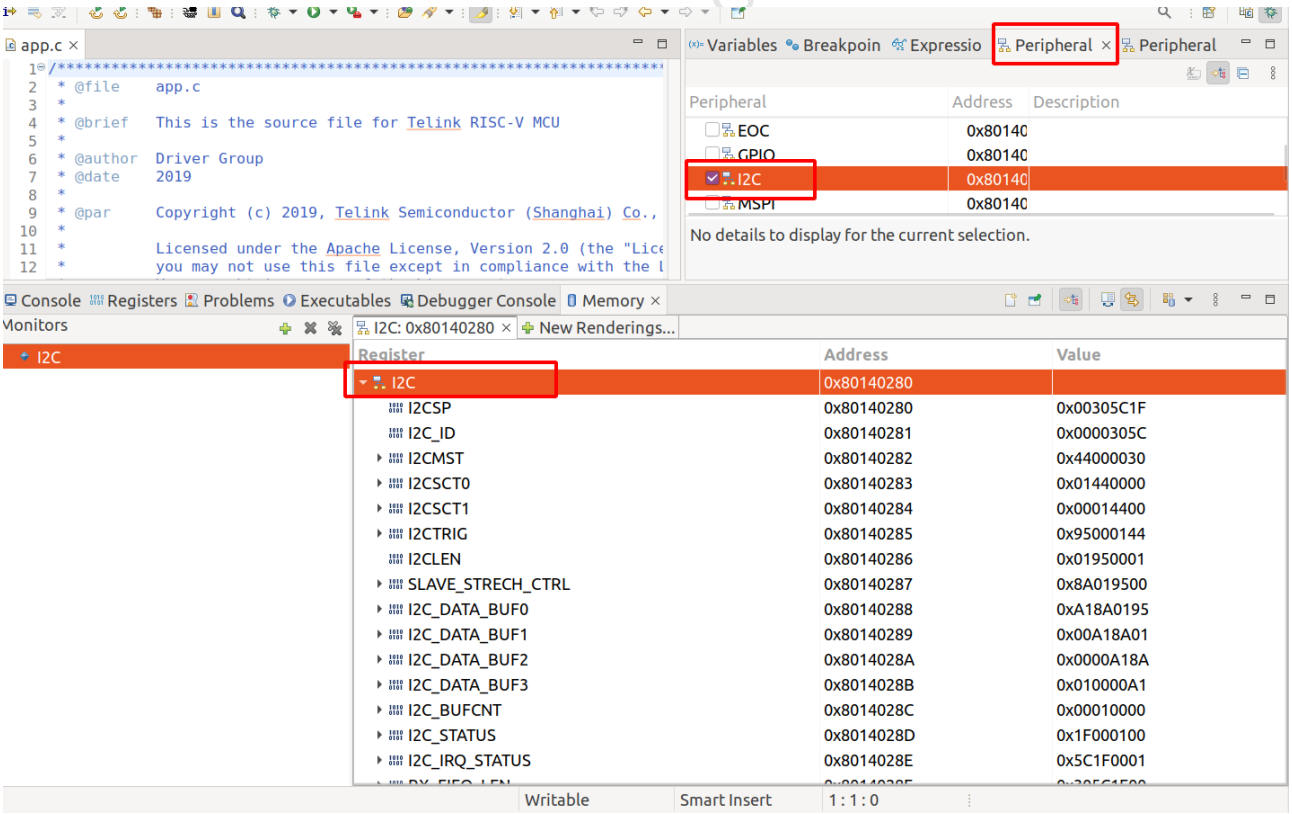

Figure 13.10: View Peripheral

13.5 Breakpoint

At present, TLSR9 series SoC supports up to two hardware breakpoints. When you find that the starting address of your program is 0x20000000, it means that it is running in flash and needs to use hardware breakpoints. Commands such as `step in` or `step over` will use a breakpoint, so users can only customize one breakpoint when debugging, otherwise an exception will occur.

When a `cannot access memory at address xx exception` occurs during debugging, you can use the `info br` command to check the number of breakpoints.

14 Independent use of some tools in the IoT Studio

14.1 Converter

The purpose of this tool is to convert the project file format of the original Telink RDS IDE to the current Telink IoT Studio project format.

To use it on the command line, you should switch to the directory of the `.cproject` file that needs to be converted, and executing $(converter_path)/TelinkRDS2IDE in this directory.

The directory where the converter is located is the $IoTStudio_PATH/tools/Converter/.

For example:

```
DELL@G15-5520 /cygdrive/c/TelinkIoTStudio/RDS/V5.1.1/ice
$ cd /cygdrive/c/Users/DELL/Desktop/wks/telink_b91m_driver_src/project/tlsr_riscv/B91

DELL@G15-5520 /cygdrive/c/Users/DELL/Desktop/wks/telink_b91m_driver_src/project/tlsr_riscv/B91
$ ls -a
.  ..  .cproject  .project  .settings  UART_Demo

DELL@G15-5520 /cygdrive/c/Users/DELL/Desktop/wks/telink_b91m_driver_src/project/tlsr_riscv/B91
$ /cygdrive/c/TelinkIoTStudio/tools/Converter/TelinkRDS2IDE.exe
Telink RDS to Telink IDE converter
  Version:     8b157697f88194fe3428ca92abf6e756d410d04
  BuildTime:   2022-06-28T10:42:53Z
  BuildInfo:   HeXiongjun@hexiongjun-5590

Set log to default level
[2023/03/17T16:19:40] [NOTICE] No input file in parameter, use the default .cproject file
[2023/03/17T16:19:40] [NOTICE] CProject file is Telink RDS IDE format, continue ...
[2023/03/17T16:19:40] [NOTICE] ----- Configuration list from original project file, count:40 -----

    0:UART_Demo               1:Debug_Demo              2:TIMER_Demo
    3:PWM_Demo                4:SPI_Demo                5:Flash_Demo
    6:s7816_Demo              7:ALG_REG_Demo            8:GPIO_Demo
    9:I2C_Demo               10:RF_Demo                11:TRNG_Demo
   12:NPE_Demo               13:USB_Demo               14:STIMER_Demo
   15:AUDIO_Demo             16:VCD_Demo               17:DHRYSTONE
   18:COREMARK               19:AES_Demo               20:PM_Demo
   21:LPC_Demo               22:MDEC_Demo              23:HTOL_Demo
   24:Test_Demo              25:EMI_BQB_Demo           26:Auto_Test_Demo
   27:Make_lib               28:PLIC_Demo              29:BT_AutoTest_Demo
   30:ADC_Demo               31:Swire_Demo             32:PKE_Demo
   33:Freertos_Demo          34:BT_EMI_Demo            35:AXON_Demo
   36:DUT_Demo               37:DUT_Test               38:Charger_Demo
   39:BQB_HCI_Demo
[2023/03/17T16:19:40] [NOTICE] ---------------------- buildPath count:40 ----------------------
[2023/03/17T16:19:40] [NOTICE] ---------------------- Excluding count:40 ----------------------
[2023/03/17T16:19:40] [NOTICE] Write target project file successfully
[2023/03/17T16:19:40] [NOTICE] Origin file: .cproject
[2023/03/17T16:19:40] [NOTICE] Origin file path: .cproject
[2023/03/17T16:19:40] [NOTICE] Successful backup origin file to: Backup_before_convert_.cproject

[2023/03/17T16:19:40] [NOTICE] Successful generated target file : .cproject


DELL@G15-5520 /cygdrive/c/Users/DELL/Desktop/wks/telink_b91m_driver_src/project/tlsr_riscv/B91
$
```

Figure 14.1: converter cmd

14.2 Jtag Burn

The function of this tool is to burn and write binary files into the TLSR9 chip. Before using it, you need to first use ICEman. The Jtag Burn tool is located in the $IoTStudio_PATH/RDS/V3.2.3/flash/, and the user manual is also in this directory. The following is a example of usage:

```
./Jtag_Burn --chip B92 --reset-and-run --verify --addr 0 --image /home/wang/UART_Demo.bin --port 2354 --telnet_port 4444 --unlock
```

The parameter of --port and --telnet_port should be same with the parameter of ICEman's Burner port and telnet port. If you see the running results of ICEman indicating its Burner Port is 2354 and its telnet port is 4444 (as shown in the following figure):

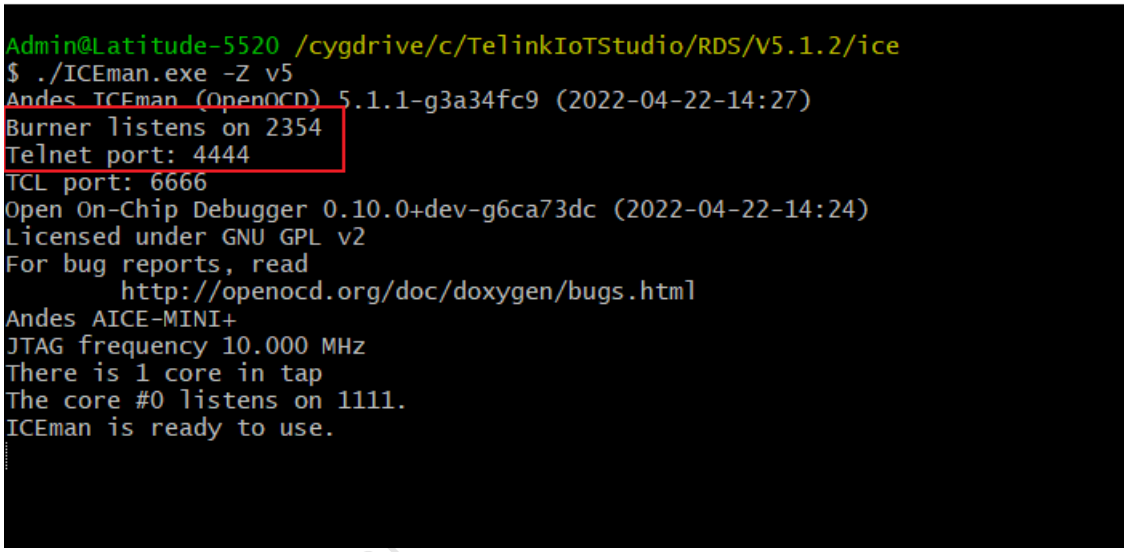

Figure 14.2: ICEman defalut port

In this case, you can run the Jtag_Burn without --port and --telnet_port.

15 Installation and uninstallation of IoTStudio in linux

15.1 Installation

In linux, run the installation program " `Telink_IoT_Studio_xxxx_Installer.run` " to install the IoTStudio. You can give it executable permissions and run it. During the execution, the user needs to enter the installation path. Be sure to enter the absolute path, as shown in the figure below.

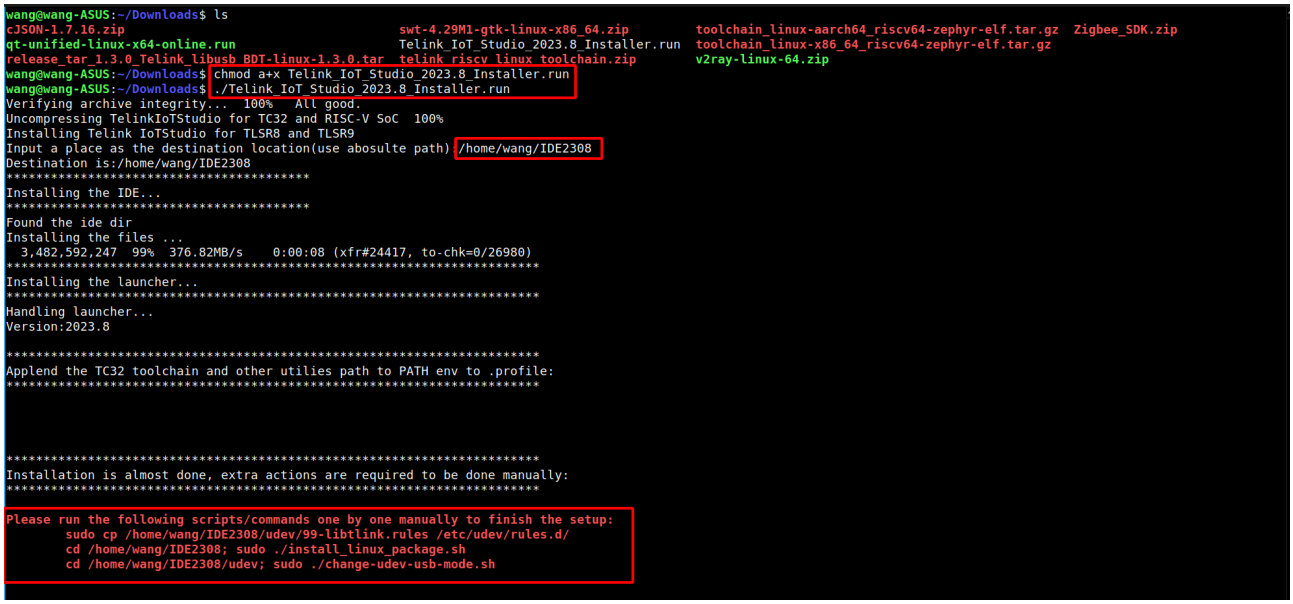

Figure 15.1: Installation

After the program execution ends, you need to execute the command given at the end of the program to complete the installation.

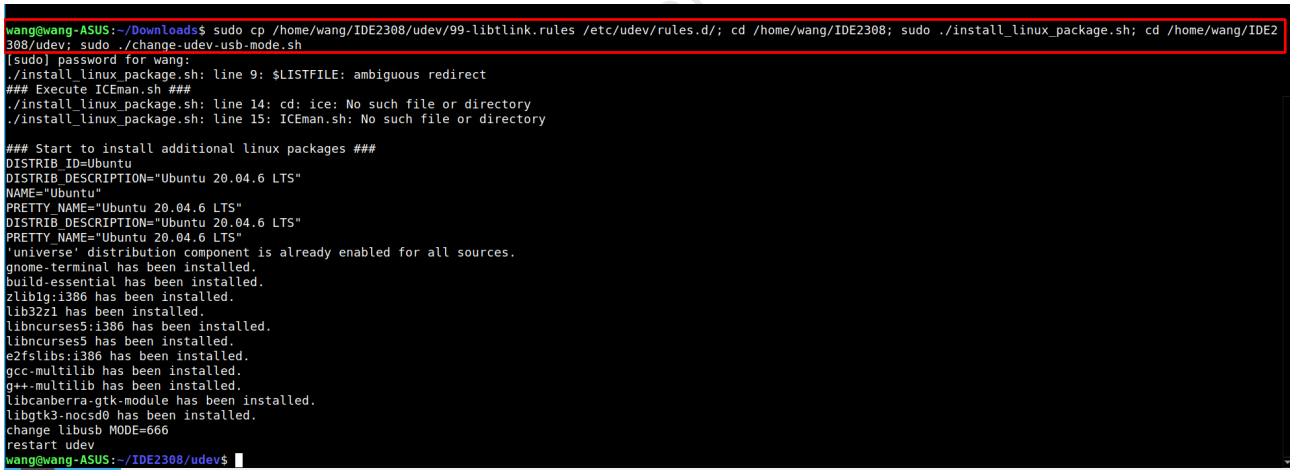

Figure 15.2: Post of installation

15.2 Uninstallation

In the IoTStudio installation directory, there is an `uninstall.sh` file that can be used to uninstall IoTStudio. You can give it executable permissions and run it. Users need to pass the path of IoTStudio's .desktop file as a parameter to uninstall.sh. Note that there may be a TelinkIoTStudio.desktop file in the installation directory, do not use it as a parameter. You should use the .desktop file with the version number and you can refer to the following example.

```
./uninstall.sh /home/wang/IDE2308/TelinkIoTStudio_2023.8.desktop
```

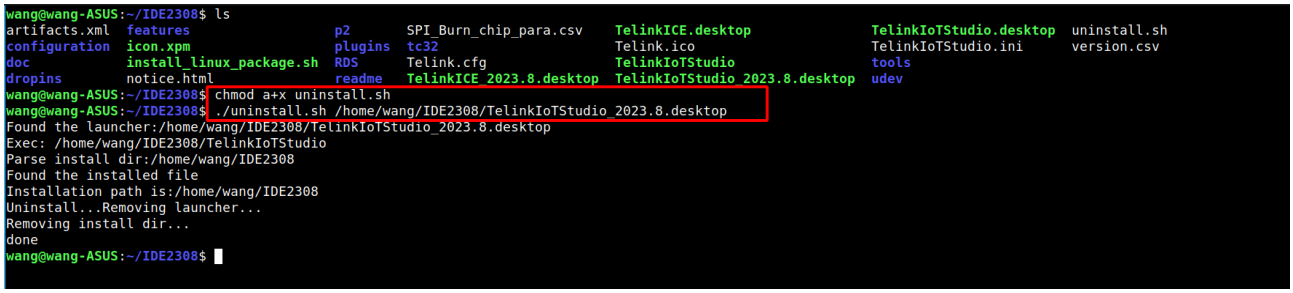

Figure 15.3: Uninstall

16 Docs for AndeSight

We have provided the relevant documents for AndeSight, which include the usage documents for Andes Riscv toolchain and other tools. You can find them in the path below according to your needs.

- `${IoTStudio_Path}/doc/Andes_V323_doc/` : Docs for AndeSight V3.2.3
- `${IoTStudio_Path}/doc/Andes_V512_doc/` : Docs for AndeSight V5.1.2
- `${IoTStudio_Path}/doc/Andes_V532_doc/` : Docs for AndeSight V5.3.2

17 How to compile IoTStudio projects in the command line

If you want to compile IoTStudio projects in the command line, you can refer to the following content.

Firstly, you need to build this project in IoTStudio, and then a folder containing makefiles and compilation results will be generated in the project directory, which is usually in the same directory as .cproject

Then open the shell of the toolchain that matches the project in Telink ->Toolchain Shell or ICEman Shell, The example toolchain in the figure is `V5.1.2 GCC10.3 V5F toolchain Shell`

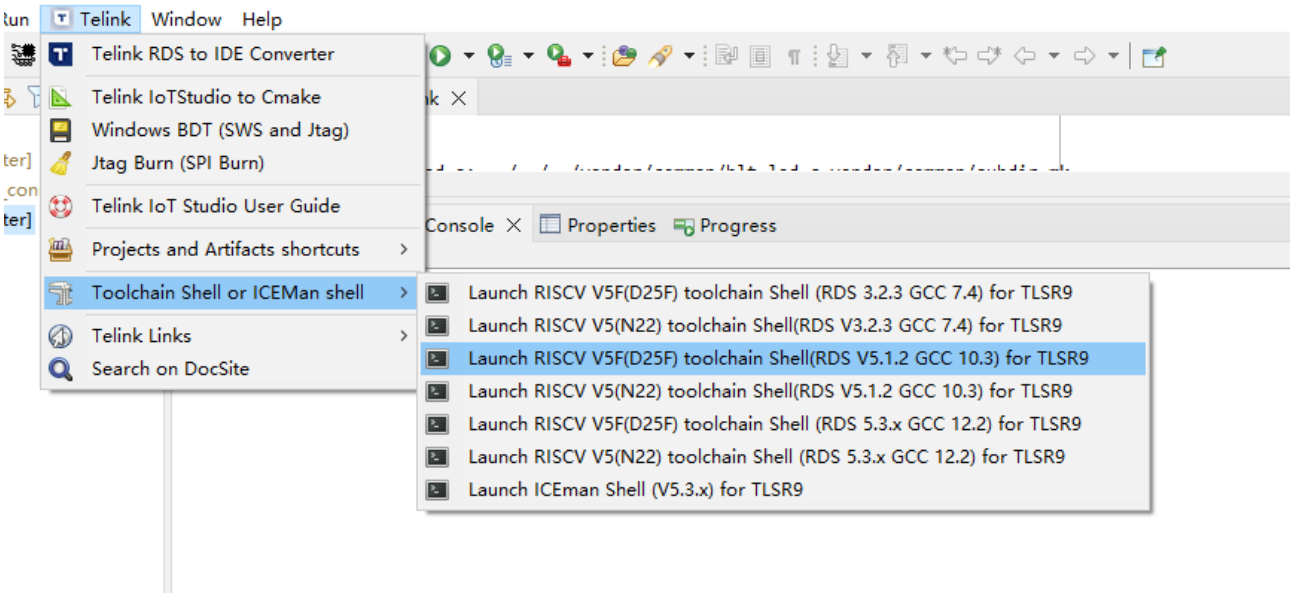

Figure 17.1: select toolchain shell

Then jump the directory to the folder containing the makefile and execute `make clean` `make all`

```
telink@DESKTOP-GOULGO6 ~
$ cd /cygdrive/c/Users/wf_15/Desktop/wks/telink_b91m_driver_src/project/tlsr_riscv/B92/UART_Demo/

telink@DESKTOP-GOULGO6 /cygdrive/c/Users/wf_15/Desktop/wks/telink_b91m_driver_src/project/tlsr_riscv/B92/UART_Demo
$ make clean
rm -rf ./vendor/common/calibration/calibration.o  ./vendor/common/auto_test/pc_interface.o  ./vendor/common/common.o ./vendor/common/excep
_DEMO/main.o  ./drivers/lib/src/pke/eccp_curve.o ./drivers/lib/src/pke/ecdh.o ./drivers/lib/src/pke/ecdsa.o ./drivers/lib/src/pke/pke.o ./
vers/lib/src/aoa.o ./drivers/lib/src/flash_base.o ./drivers/lib/src/hadm_drv.o ./drivers/lib/src/plic.o ./drivers/lib/src/pm.o ./drivers/l
h/flash_mid146085.o ./drivers/flash/flash_mid156085.o ./drivers/flash/flash_mid1560c8.o ./drivers/flash/flash_mid166085.o ./drivers/flash/
ivers/charger.o ./drivers/charger_bin.o ./drivers/clock.o ./drivers/core.o ./drivers/ctb.o ./drivers/emi.o ./drivers/flash.o ./drivers/gpi
16.o ./drivers/spi.o ./drivers/stimer.o ./drivers/timer.o ./drivers/uart.o ./drivers/usbhw.o ./drivers/watchdog.o  ./common/usb_dbg/myudb_
ram.o UART_Demo.bin UART_Demo.lst UART_Demo.siz ./boot/cstartup_flash.d ./boot/cstartup_ram.d ./vendor/common/calibration/calibration.d  .
_DEMO/app.d ./vendor/UART_DEMO/app_dma.d ./vendor/UART_DEMO/app_dma_llp.d ./vendor/UART_DEMO/main.d  ./drivers/lib/src/pke/eccp_curve.d ./
src/pke/pke_utility.d ./drivers/lib/src/pke/rsa.d ./drivers/lib/src/pke/x25519.d  ./drivers/lib/src/aoa.d ./drivers/lib/src/flash_base.d .
s/lib/src/sys.d ./drivers/lib/src/trng.d  ./drivers/flash/flash_common.d ./drivers/flash/flash_mid146085.d ./drivers/flash/flash_mid156085
or_handler.d  ./drivers/adc.d ./drivers/aes.d ./drivers/analog.d ./drivers/audio.d ./drivers/charger.d ./drivers/charger_bin.d ./drivers/c
mdec.d ./drivers/mspi.d ./drivers/plmt.d ./drivers/pwm.d ./drivers/qdec.d ./drivers/s7816.d ./drivers/spi.d ./drivers/stimer.d ./drivers/t
on/bt_debug/dbgport.d  ./common/sdk_version.d  UART_Demo.elf


telink@DESKTOP-GOULGO6 /cygdrive/c/Users/wf_15/Desktop/wks/telink_b91m_driver_src/project/tlsr_riscv/B92/UART_Demo
$ make all
Building file: ../../../../demo/vendor/common/B92/calibration/calibration.c
Invoking: GNU RISC-V Cross C Compiler
riscv32-elf-gcc -mcmodel=medium -O2 -fmessage-length=0 -ffunction-sections -fdata-sections -flto -Werror -Wall -Wextra -Wshadow -Wimplicit
rict-prototypes -Wmissing-field-initializers -Wpointer-arith -Wdeprecated-declarations -Wredundant-decls -Werror=cpp -Wenum-conversion -Wp
DMCU_CORE_B930=0 -DMCU_CORE_B95=0 -DMCU_CORE_B96=0 -I"C:\Users\wf_15\Desktop\wks\telink_b91m_driver_src\chip\B92\drivers" -I"C:\Users\wf_1
 -I"C:\Users\wf_15\Desktop\wks\telink_b91m_driver_src\demo\vendor\common\common" -Wall -Wextra -Wshadow -Wimplicit-fallthrough -Wpointer-a
g-field-initializers -Wpointer-arith -Wdeprecated-declarations -Wredundant-decls -Werror=cpp -Wenum-conversion -Wpacked-not-aligned -Waddr
td=c99 -fpack-struct -fshort-enums -fno-jump-tables -fno-fat-lto-objects -mcmodel=medium -Wno-nonnull-compare -MMD -MP -MF"vendor/common/c
emo/vendor/common/B92/calibration/calibration.c"
Finished building: ../../../../demo/vendor/common/B92/calibration/calibration.c
```

Figure 17.2: make in shell