

AK100 Emulator for ARM User Manual

AN05220022 V1.00 Date: 2011/04/22

Application Note

Categories	Content
Keywords	AK100 Emulator ARM User Manual
Summary	AK100 Emulator for ARM User Manual





The revision of history

Version	Date	Reason
V1.00	2011/04/22	Create the document

Table of Contents

Chapter 1 TKSscope Emulator Introduction	1
1.1 Supported ARM Cores	1
1.2 Simulation Performance for ARM Core.....	2
Chapter 2 Use TKSscope Emulator.....	4
2.1 Install Driver	4
2.2 Hardware Connection.....	6
Chapter 3 Simulation ARM In Keil RealView MDK	8
3.1 Simulation Environment Setting	8
3.2 Emulator Settings.....	9
3.2.1 Device&hardwire type	9
3.2.2 Main Options.....	10
3.2.3 TAP config	11
3.2.4 Flash Writer.....	12
3.2.5 Init File.....	15
3.2.6 TKSscope Doctor.....	17
3.3 Debugging.....	18
3.3.1 Start Debugging.....	18
3.3.2 Debugging Tools	19
3.3.3 Debugging Results	20
Chapter 4 Simulation ARM In ADS	21
4.1 Add Driver File	21
4.2 Debugging	24
4.2.1 Debugging Tools	24
4.2.2 Debugging Results	25
Chapter 5 Simulation ARM In IAR	26
5.1 Add Driver File	26
5.2 Debugging.....	30
5.2.1 Start Debugging.....	30
5.2.2 Debugging Tools	31
5.2.3 Debugging Results	32

Chapter 1 AK100 Emulator Introduction

1.1 Supported ARM Cores

TKScope embedded intelligent emulator designed by Guangzhou Zhiyuan Electronics Co., Ltd. in 2008, is a high-performance general-purpose integrated emulator. TKScope supports a full range of ARM / DSP / AVR / 8051 / C166 / C251 / C8051F / MX Core etc, supports all the mainstream IDE, such as TKStudio / Keil / ADS / IAR / CCS / RealView / AVRStudio / SDT and so on, to ensure consistency of your development platform, and with its advanced debugging features. At the same time, TKScope built-in 64-way professional logic analyzer, zlgLogic advanced software is fully supported.

TKScope emulator able to support ARM core simulation models are summarized as follows:

- K Series: K8 / K9;
- DK Series: DK9 / DK10;
- AK Series: AK100.

At present, TKScope emulator supports ARM core categories are as follows:

- ARM7: ARM7TDMI / ARM7TDMI-S / ARM7EJ-S / ARM720T;
- ARM9: ARM9TDMI / ARM920T / ARM922T / ARM926EJ-S / ARM946E-S / ARM966E-S / ARM968E-S / ARM966HS;
- ARM11: ARM1136 / ARM1156 / ARM1176;
- Cortex: Cortex-M0 / Cortex-M1 / Cortex-M3;
- XSCALE: PXA255 / PXA270.

At present, TKScope simulation environment for ARM core supports IDE are as follows:

- TKStudio, Zhiyuan company, Chinese/english environment, multi-core compiling/debugging, powerful built-in editor;
- Keil, Keil company, english environment, 8051/251/C166/ARM compiling/debugging;
- ADS, ARM company, english environment, full ARM cores compiling/debugging;
- IAR, IAR company, english environment, multi-core compiling/debugging;
- RealView, ARM company, english environment, full ARM cores compiling/debugging.



Figure 1.1 Mainstream IDE Interface

1.2 Simulation Performance for ARM Core

At present, TKScope supports a full range of ARM7 / ARM9 / Cortex-M0 / Cortex-M1 / Cortex-M3 / XSCLAE / ARM11 simulation, including Thumb mode. Follow, TKScope will support a full range of Cortex-R4 / Cortex-A8 simulation by software update.

TKScope emulator hardware Index:

- USB2.0 High-Speed Communication Interface;
- Standard 20-pin JTAG interface target board connection, support hot-swappable;
- Detection of all JTAG signals and target board voltage;
- Adaptive target board voltage, to support a wide voltage range of 1.8V~5V;
- JTAG maximum clock 25MHz, can reach the speed limit debugging;
- Automatic speed recognition;
- Supports real-time RTCK synchronous clock (adaptive clock);
- With the TKScope Doctor, facilitate the detection of hardware failure to exclude.

TKScope emulator features for ARM simulation:

- Support the full range of ARM core simulation, such as ARM7 / ARM9 / Cortex-M0 / Cortex-M1 / Cortex-M3 / XSCLAE / ARM11 and so on, including Thumb mode;
- Support the Cortex-M0 / Cortex-M1 / Cortex-M3 core serial debug (SWD) mode;
- Support all the mainstream IDE, such as TKStudio / Keil / ADS / IAR / RealView / SDT and so on;
- Support the on-chip Flash in-circuit programming / debugging, providing each chip corresponding Flash programming algorithm file;
- Support the off-chip Flash in-circuit programming / debugging, providing hundreds of commonly used Flash device programming algorithm file;
- Support multiple interface types of external Flash programming / debugging, such as NOR / NAND / SPI Flash and so on;
- Allowing users to add their own Flash programming algorithm file;
- Provide a separate programming-Flash independent software to increase productivity;
- Support for unlimited RAM breakpoint debugging;
- Support for unlimited Flash breakpoint debugging, breaking limit the number of hardware breakpoints;
- Synchronous Flash technology, fast refresh Flash breakpoints, the speed as fast as RAM debugging;
- Support dynamic breakpoints can set / cancel any breakpoint in running;
- Support program breakpoints and data breakpoints, user-friendly and accurately track complex programs to run;
- Rapid single-step, the fastest 150 steps / sec;
- Ensure the fastest and most stable frequency changes in the target system to debug;
- Built-in special debugging algorithms, reliable debugging of the ARM core in an irregular situation;
- Support for daisy-chain connection of multi-device simulation;
- Chip-based design concepts for the hundreds of kinds of chips to provide a sound initialization file.



- Built-in comprehensive interpretation of the actuator initialization files, can be flexible system settings before and after reset / run around / Flash download around, including register settings / ARM initialization / clock Settings / delay / information and so on.



Figure 1.2 TKScope K8

Chapter 2 Use AK100 Emulator

2.1 Install Driver

AK100 emulator driver must be installed before use, otherwise, does not work!

Double-click TKScoSetup_ARM.EXE, the system pop-up dialog box as shown in Figure 2.1, click[Next]to continue.

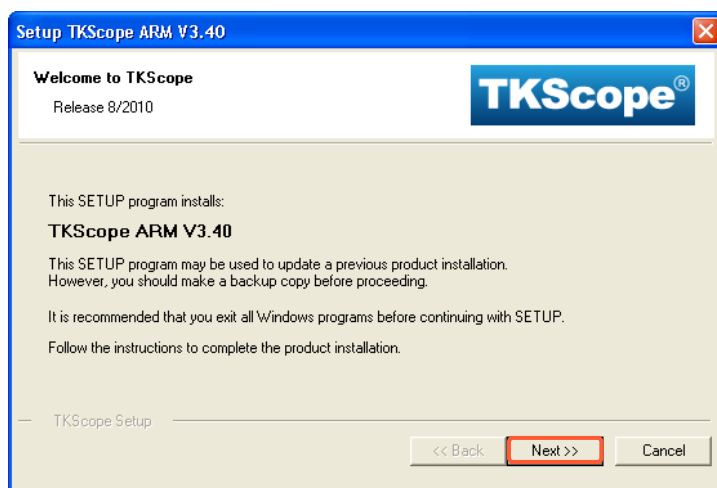


Figure 2.1 Install AK100 Driver

Drivers must be installed to the Keil RealView MDK directory, if you are using Keil RealView MDK development environment; else arbitrary directory.

For example, Keil RealView MDK development environment installed on the D drive, therefore, AK100 emulator driver installation path to D:\Keil, as shown in Figure 2.2. Click [Next]to continue until the installation is completed.

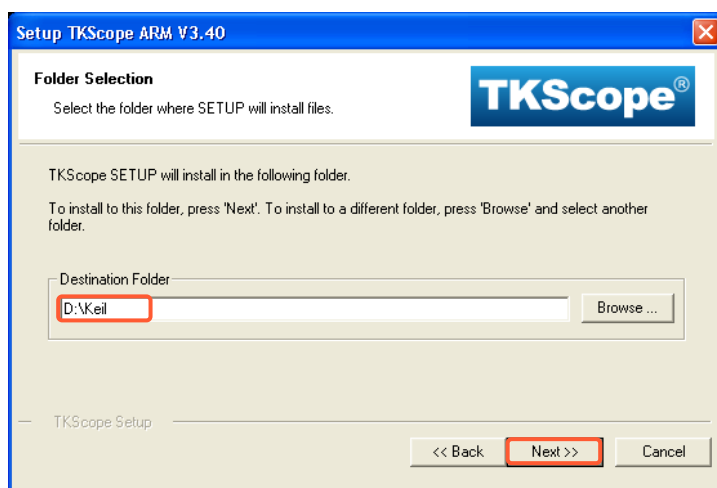


Figure 2.2 Driver Installation Path

AK100 emulator driver installed, it is recommended users install vc8 from the Microsoft. Double-click vc8dist_x86_en.exe, the system pop-up dialog box as shown in Figure 2.3, click[Yes]to continue.

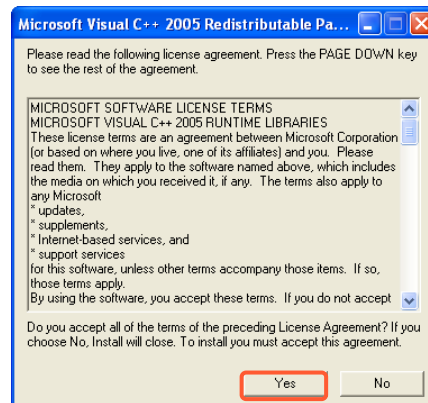


Figure 2.3 Install Vc8

Until now, AK100 emulator all the necessary drivers installed.

In the installation directory (example is D:\Keil\TKScope), you can see driver files for various environmental, as shown in Figure 2.4.

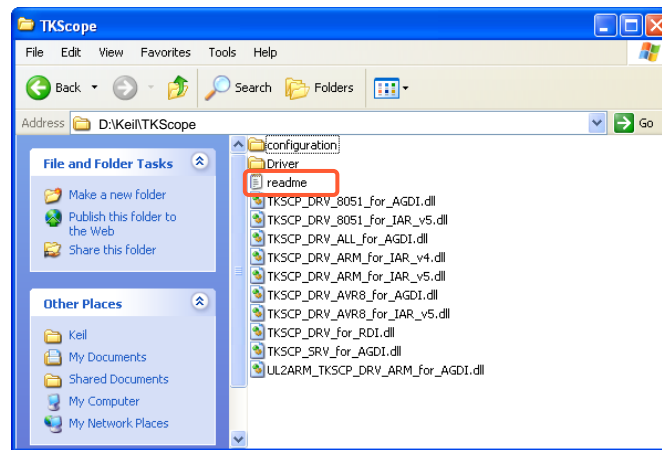


Figure 2.4 Driver Files

Various driver files belong to the type and application development environment, users should see the readme file under the installation directory (example Figure 2.4).

Table 2.1 lists all the driver files for ARM at present.

Table 2.1 Driver File List

Driver File Name	Drive Type	Application Environment
UL2ARM_TKSCP_DRV_ARM_for_AGDI.dll	ARM	Keil Uvsnion4/Uvsnion3/Uvsnion2
TKSCP_DRV_ARM_for_IAR_v4.dll	ARM	IAR V4
TKSCP_DRV_ARM_for_IAR_v5.dll	ARM	IAR V5
TKSCP_DRV_for_RDI.dll	ARM	AXD(ADS), and other RDI agreement

2.2 Hardware Connection

For the first time, AK100 emulator power use, the system will pop-up dialog box as shown in Figure 2.5.

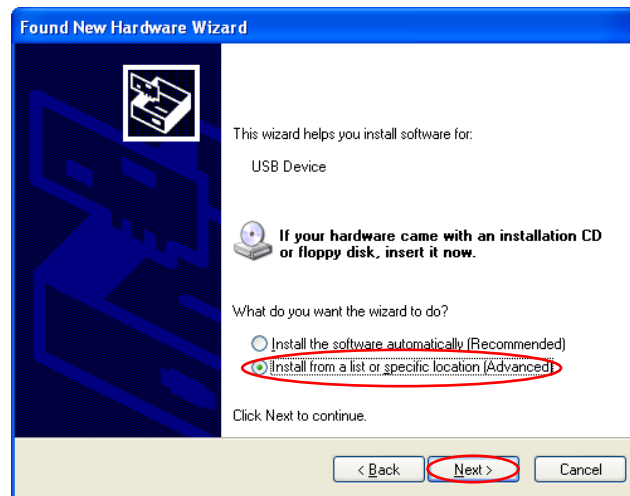


Figure 2.5 New Hardware Installation Wizard

Select [Install from a list or specific location(Advanced)]options in Figure 2.5, click[Next], the system will pop-up dialog box as shown in Figure 2.6.

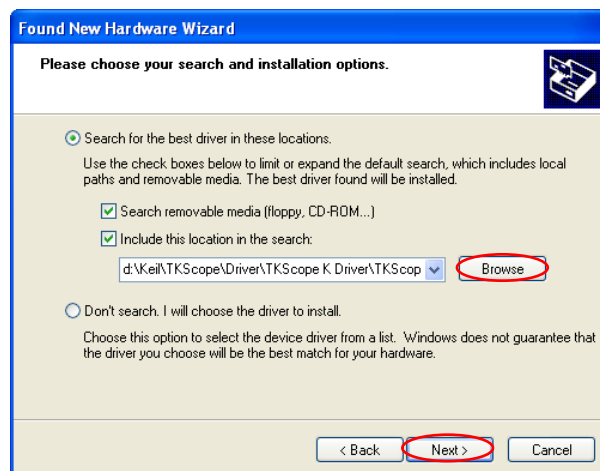


Figure 2.6 Select Drive Box

In Figure 2.6, click[Browse], open the dialog box as shown in Figure 2.7. Find the driver files in AK100 emulator installation directory (example is D:\Keil\TKScope\Driver\AK100 Driver), click[OK].

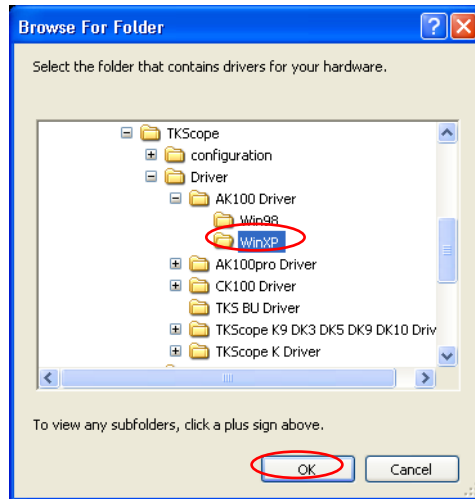


Figure 2.7 Designated Driver

Drive installed, the system will pop-up dialog box as shown in Figure 2.8, click [Finish] to complete.

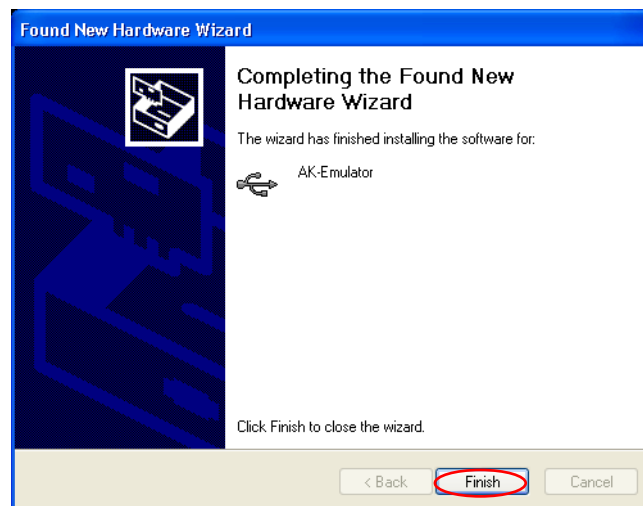


Figure 2.8 New Hardware Installation Completed

Chapter 3 Simulation ARM In Keil RealView MDK

3.1 Simulation Environment Setting

In Keil RealView MDK environment, open the project to compile ok, as shown in Figure 3.1.

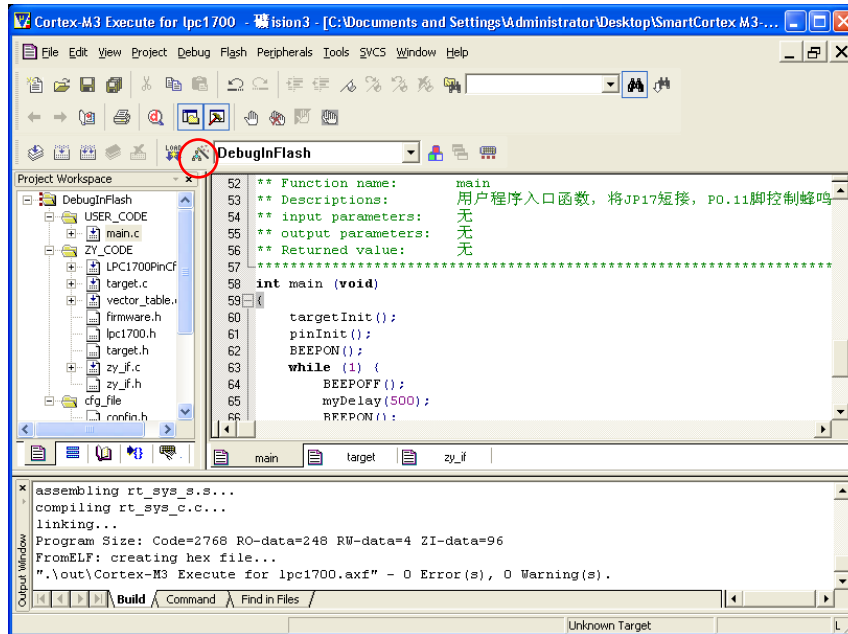


Figure 3.1 MDK Main Interface

Click icon  in Figure 3.1, open the project settings interface as shown in Figure 3.2.

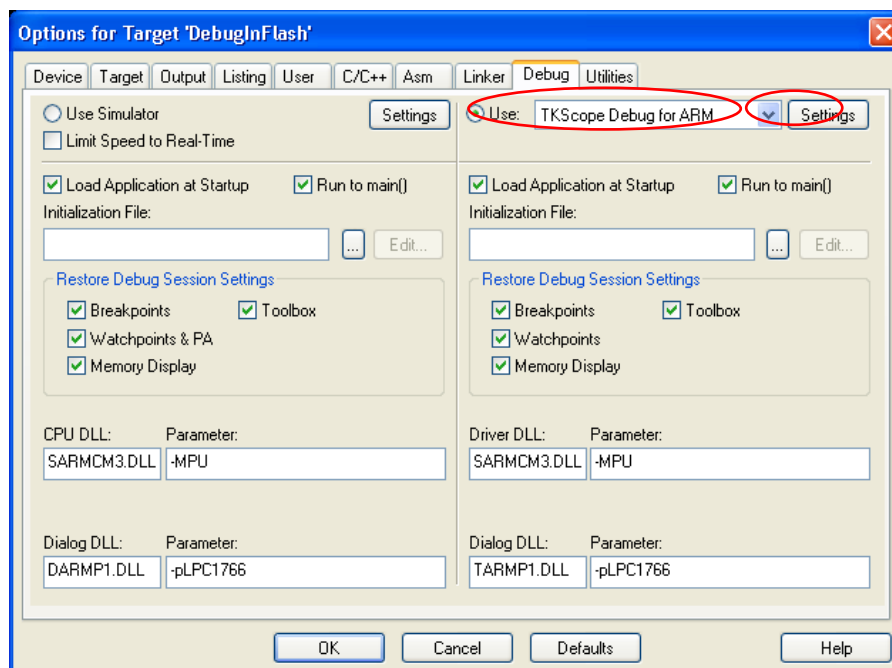


Figure 3.2 Project Settings Interface

In Figure 3.2, select the hardware emulation, the corresponding driver select[TKScope Debug for ARM]. Then, click [Settings] into the AK100 emulator settings interface, as shown in Figure 3.3.

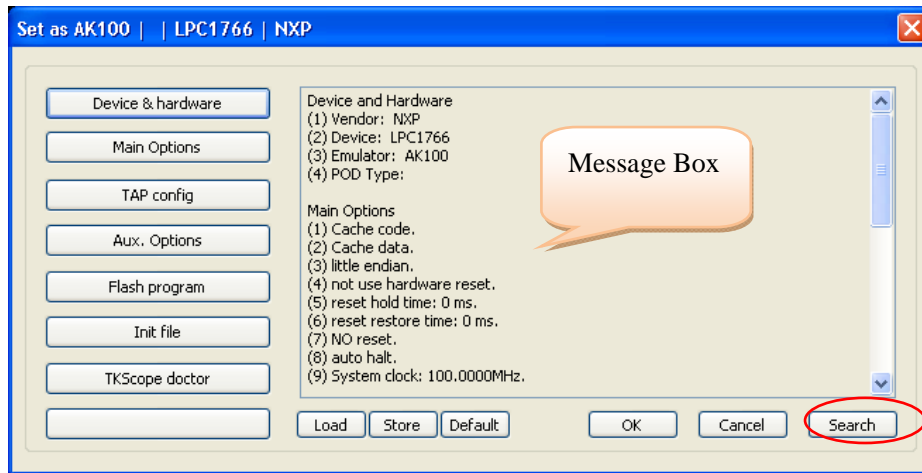


Figure 3.3 AK100 emulator settings interface

In Figure 3.3, click the left side of the various options, the system will pop-up the appropriate settings interface, while the right side of the information prompt box will appear the specific meaning of the various settings.

3.2 Emulator Settings

3.2.1 Device&hardwire type

In Figure 3.3,click[Device&hardwire type], into the interface as shown in Figure 3.4.

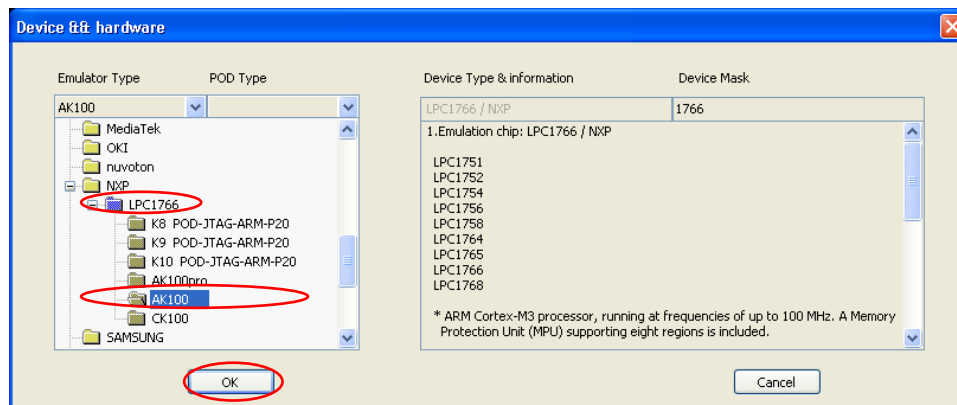


Figure 3.4 Device&hardwire type Interface

In Figure 3.4, users must select the correct chip type and emulator type. Then click[OK] to return to the interface as shown in Figure 3.3.

3.2.2 Main Options

In Figure 3.3, click[Main Options], into the interface as shown in Figure 3.5.

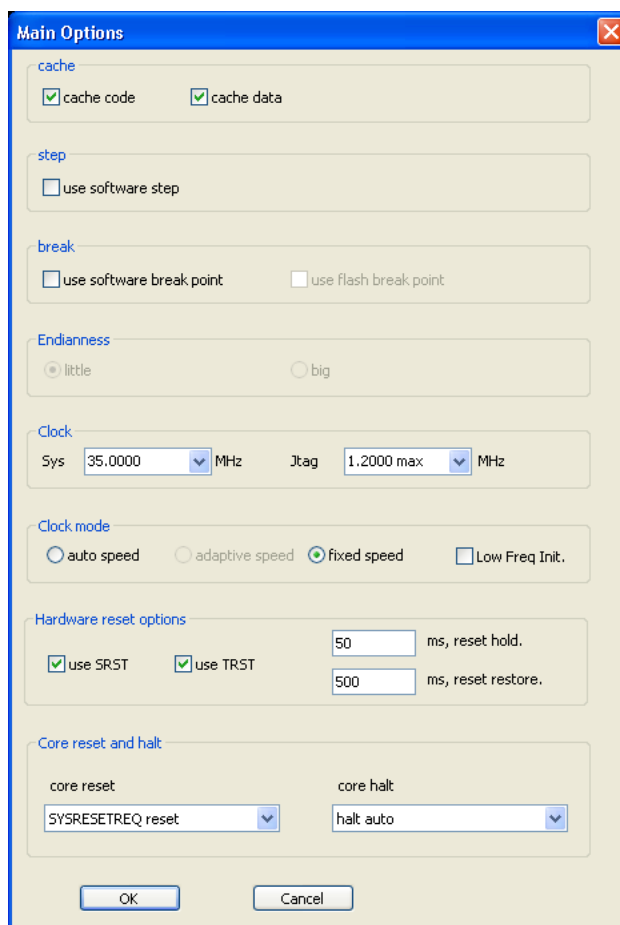


Figure 3.5 Main Options Interface

1. Cache

[cache code]: Don't read code from hardware if it be readed.

[cache data]: Don't read data from hardware if it be readed, except meet a run.

2. Step

[use software step]: Don't repeat programming Flash breakpoints.

3. Break

[use software break point]: To achieve unlimited breakpoints to debug in RAM.

[use flash break pint]: To achieve unlimited breakpoints to debug in flash.

4. Endianness

Choose endian for the device, big endian or little endian.

If the endian of the device is fixed, no endian choice here.

5. Clock

[Sys clock]: Your system clock frequency(MHz).

[JTAG clock]: Your JTAG clock frequency(MHz).

6. Clock mode

[Auto speed]: Auto select the best JTAG speed.

[Adaptive speed]: Select max. speed based on RTCK if the device has.

[Fixed speed]: Select the speed you want.

7. Hardware reset options

[Use SRST]: Use SRST.

[Use TRST]: Use TRST.

[Reset hold]: Reset hold delay time(ms).

[Reset restore]: Reset release delay time(ms).

8. Core reset and halt

[Core reset]: no reset, soft reset.

[Core halt]: auto, use DBRQ, use break point, use special method.

3.2.3 TAP config

In Figure 3.3, click [TAP config], into the interface as shown in Figure 3.6.

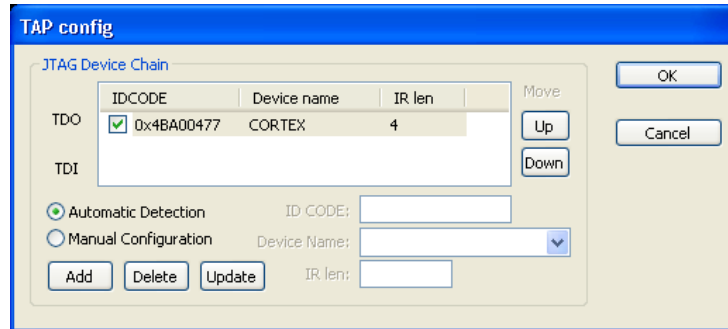


Figure 3.6 TAP config Interface

TAP configurations to setup the JTAG chain, include number of devices IR length, active device.

It's important to setup when there is unknown device in the JTAG chain. Normally, [Automatic Detection] option will be ok.

[devices list]: List all devices in scan chain, include name, IR length, idcode.

[IDCODE]: The idcode of the device, for add, update operations.

[name]: The name of the device.

[IR length]: The IR length of the JTAG device.

[Automatic Detection]: Setup scan chain automatically.

[Manual Configuration]: Setup manually.

[Add]: Add a new device in the devices list.

[Delete]: Delete a device in the device list.

[Update]: Update the device in the device list.

[Up]: Move the selected device upward.

[Down]: Move the selected device downward.

3.2.4 Flash Writer

In Figure 3.3, click [Flash Writer], into the interface as shown in Figure 3.7.

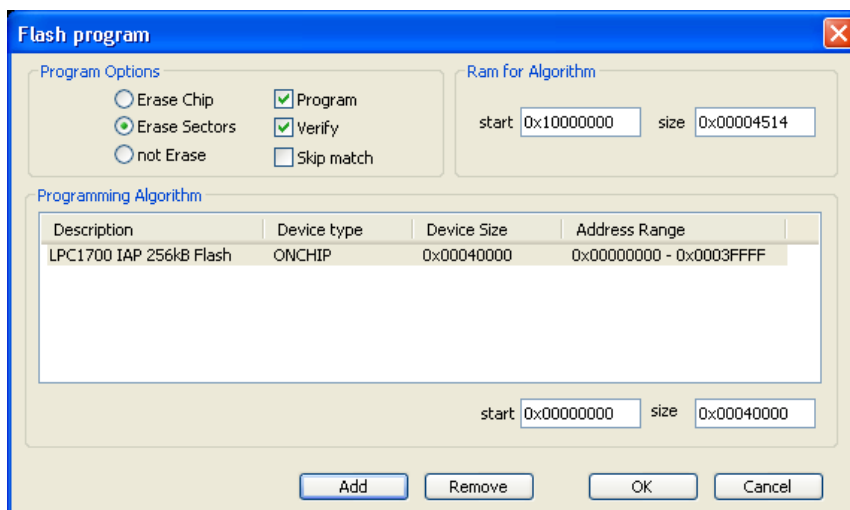


Figure 3.7 Flash Writer Interface

1. Program Options

[Erase Chip]: Erase all Flash onchip before download.

[Erase Sectors]: Erase the sector that download used.

[not Erase]: Don't erase the flash.

2. Ram for Algorithm

[Start]: Start address of ram for run flash program algorithm.

[Size]: The ram size.

3. Programming Algorithm

Flash Programming algorithm, up to 4 algorithm can be loaded. Yourself algorithm can be written, and loaded here.

[Start]: Flash Algorithm start re_address.

[Size]: Flash Algorithm start re_size.

4. Flash Access

[Add]: Add a new programming Algorithm.

[Remove]: Remove a programming Algorithm selected.

Users must select the [program] [verify] options,if you debug in Flash,and select [Erase Chip] or [Erase Sectors].

In addition, it is recommended users select the [skip match] option, so the repetition of Flash programming can greatly enhance the programming speed.

The following will explain a few commonly used methods of Flash programming.

1. Chip Flash

Chip with Flash, the system will automatically load the Flash algorithm file.

For example, NXP LPC1766 chip with 256K Flash, debug this chip in Flash, setting interface as shown in Figure 3.8.

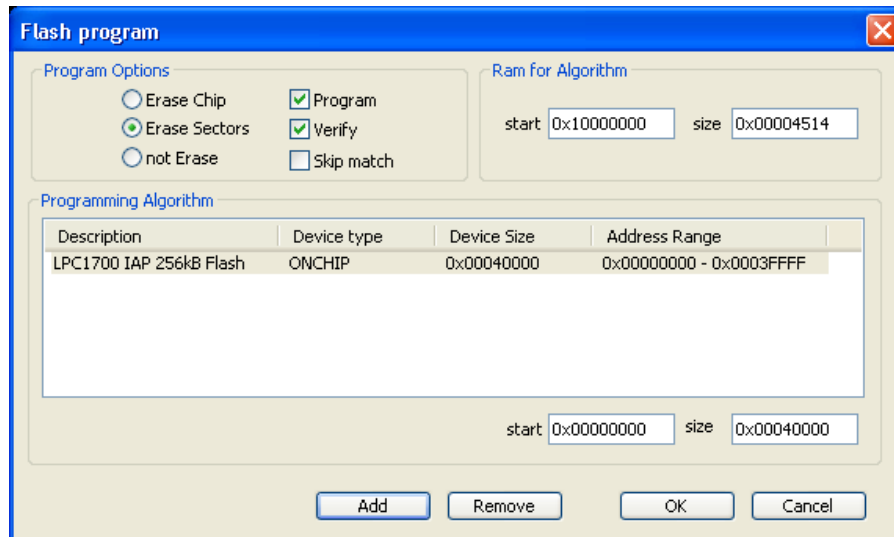


Figure 3.8 Flash Writer Interface 1

2. External Falsh

Chip without Flash, users need to add external Flash algorithm file, and properly set start address and size.

For example, NXP LPC2220 chip without Flash, debug in external Flash SST39VF160. Click[Add] option, load SST39VF160 Flash algorithm file. Then, start address is set to 0x80000000 according to chip characteristics. The setting interface is as shown in Figure 3.9.

AK100 emulator offers lots of external Flash algorithm files, sotred in the driver installation directory(TKScope\configuration\ExtFlash).

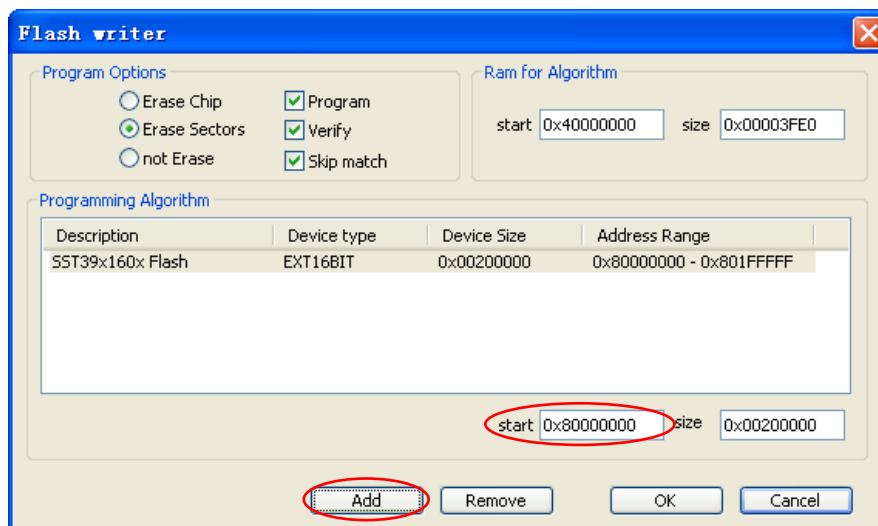


Figure 3.9 Flash Writer Interface 2

3. RAM Debug

Debug in RAM, users don't need to select[program] option, the setting interface is as shown in Figure 3.10.

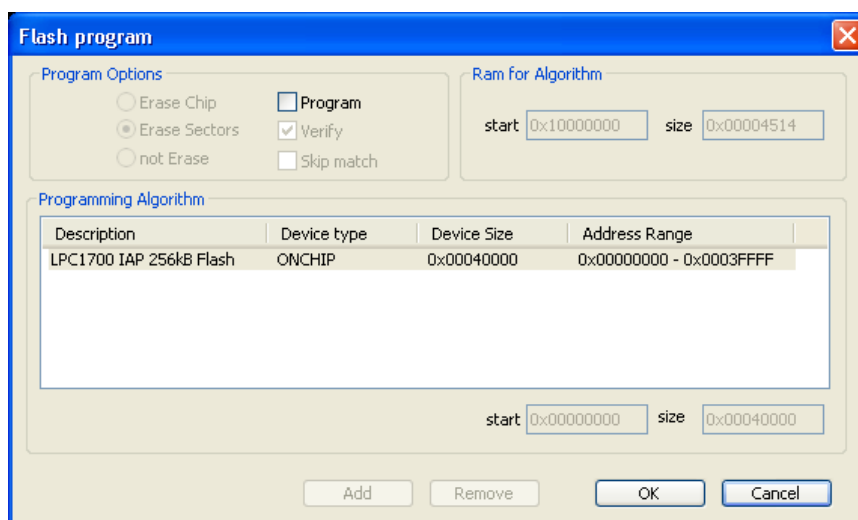


Figure 3.10 Flash Writer Interface 3

3.2.5 Init File

In Figure 3.3,click[Init File],into the interface as shown in Figure 3.11.

Users can click  coin to add the init file.

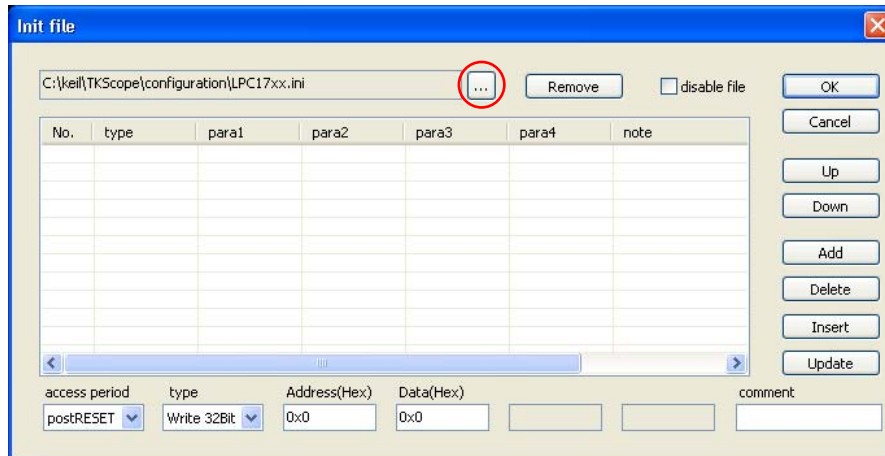


Figure 3.11 Init File Interface

AK100 emulator offers lots of init files, sorted in the driver installation directory (TKScope\configuration\ExtFlash).

Init macro is loaded as a file with postfix .ini. Period and access can be defined in a ini file.

Period supported:

[preRESET]:	execute before reset.
[postRESET]:	execute after reset.
[preRUN]:	execute before run.
[postRUN]:	execute after reset.
[preFlash]:	execute before Flash write.
[postFlash]:	execute after Flash write.
[preDownload]:	execute before program download.
[postDownload]:	execute after program download.

Access supported:

"Read 32bit":	read 32bit data.
"Read 16bit":	read 16bit data.
"Read 8bit":	read 8bit data.
"Write 32bit":	write 32bit data.
"Write 16bit":	write 16bit data.
"Write 8bit":	write 8bit data.

"SetJtagClock":	set jtag mode and frequency(decimal).
"Delay":	delay, unit ms(decimal).
"SetPC":	set program counter(hex).
"Message":	output a message.

Users can write their own init files according to their own applications. The following is an example.

```
[postRESET]
InitStep0_Action = "Write 32bit"
InitStep0_Comment = "MEMMAP internal flash"
InitStep0_Value0 = 0xE01FC040
InitStep0_Value1 = 0x1
```

```
InitStep1_Action = "SetJtagClock"
InitStep1_Comment = "1MHz "
InitStep1_Value0 = "FixedJtagClock"
InitStep1_Value1 = 10000000
```

```
InitStep2_Action = "Read 32bit"
InitStep2_Comment = " Read 32bit data"
InitStep2_Value0 = 0xE01FC080
InitStep2_Value1 = 0
```

```
InitStep3_Action = "Delay"
InitStep3_Comment = "delay 1000ms"
InitStep3_Value0 = 1000
```

```
InitStep4_Action = "Message"
InitStep4_Value0 = "Hi! This is a message!"
```

```
InitStep5_Action = "SetPC"
InitStep5_Comment = "Set PC"
InitStep5_Value0 = 0x12345678
```

```
[preRun]
InitStep0_Action = "SetJtagClock"
InitStep0_Comment = "Set Auto Clock"
InitStep0_Value0 = "AutoJtagClock"
```

```
[postRun]
InitStep0_Action = "SetJtagClock"
InitStep0_Comment = "Set adaptive Clock"
InitStep0_Value0 = "SycJtagClock"
```

3.2.6 TKScope Doctor

In Figure 3.3, click[TKScope doctor], into the interface as shown in Figure 3.12.

TKScope doctor's function is very powerful, detecting own hardware init, USB communication, hardware reset, and reading ARM core ID 100000 times.

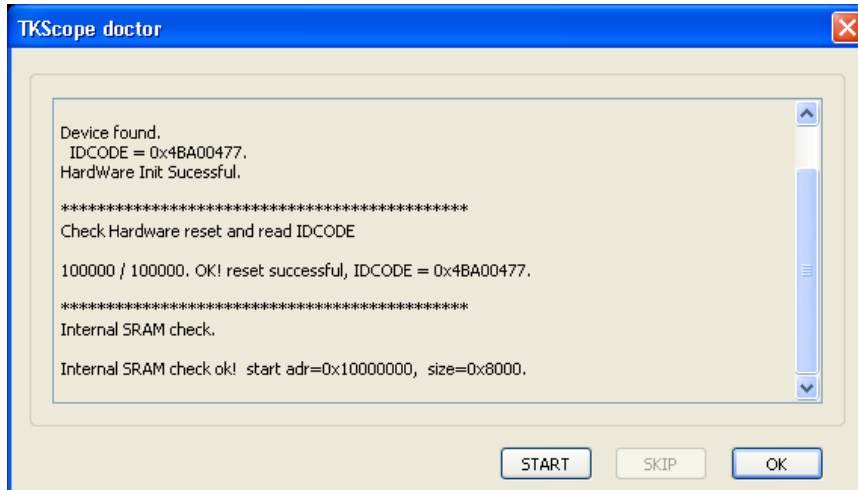


Figure 3.12 TKScope Doctor Interface

3.3 Debugging

3.3.1 Start Debugging

Users can debug after the completion of the emulator settings.

Click [Debug] menu [Start/Stop Debug Session] option, or click  icon, into debugging status, as shown in Figure 3.13.

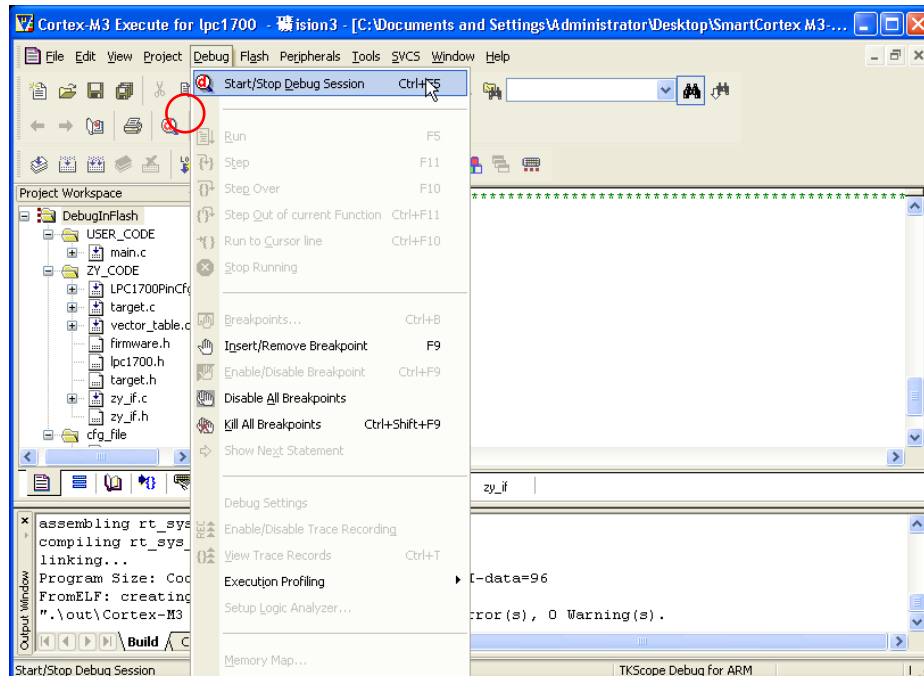


Figure 3.13 Debugging Interface

3.3.2 Debugging Tools

Keil RealView MDK debugging environment provides the following debugging tools, as shown in Figure 3.14.

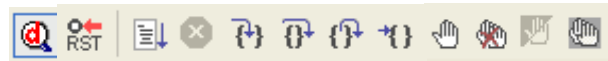





Figure 3.14 Debugging Tools

-  Start/Stop Debug Session
-  Reset CPU
-  Go
-  Stop
-  Step In
-  Step
-  Step Out
-  Run To Cursor
-  Insert/Remove Breakpoint
-  Kill All Breakpoints
-  Enable/Disable Breakpoint
-  Disable All Breakpoints

Keil RealView MDK debugging environment provides the following watching tools, as shown in Figure 3.15.



Figure 3.15 Watching Tools

-  Disassembly Window
-  Watch and Call Stack Window
-  Memory Window

3.3.3 Debugging Results

After debugging ended, users can click[Debug] menu [Start/Stop Debug Session] option, or click  icon, exit debugging status.


If users debug in RAM, programs written in the chip RAM, will lost after power down. Programs will not run in target board if power up again.

If users debug in Flash, programs written in the Flash, will save after power down. Programs will run in target board if power up again.

If users debug in Flash and encryption, can not debug again. Unless, global erase chip, will be re- debugging.

Chapter 4 Simulation ARM In ADS

4.1 Add Driver File

In ADS environment, open the project to compile ok, as shown in Figure 4.1. Click  icon, into AXD debugging environment, as shown in Figure 4.2

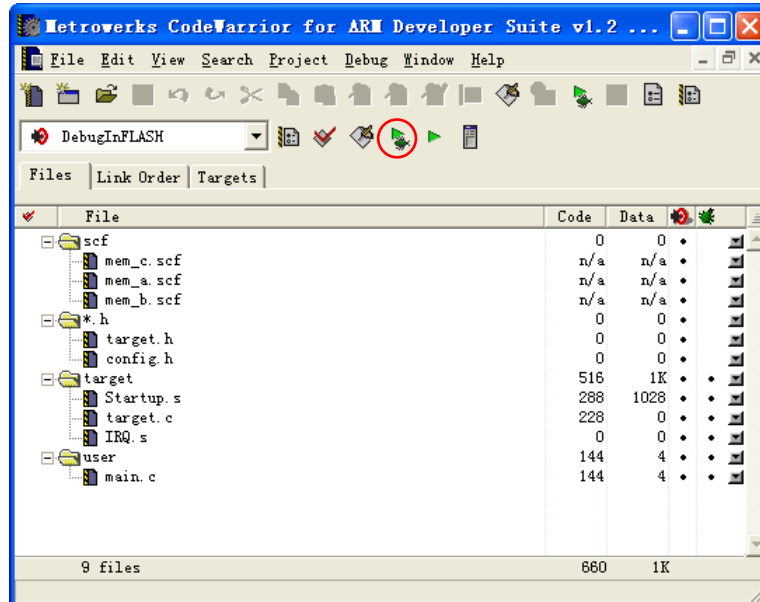


Figure 4.1 ADS main interface

Select [Options] menu [Configure Target] option as shown in Figure 4.2.

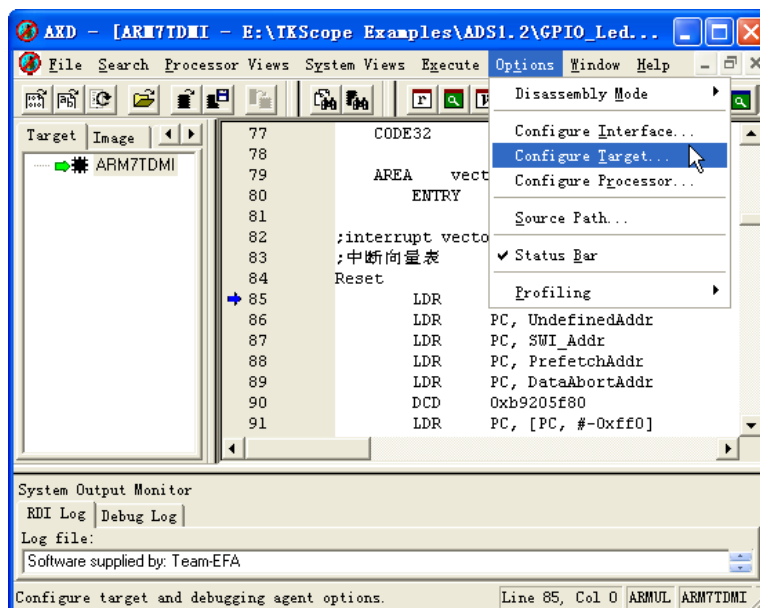


Figure 4.2 AXD main interface

The system will pop-up [Choose Target] window, as shown in Figure 4.3. Click [Add] to add drive file.

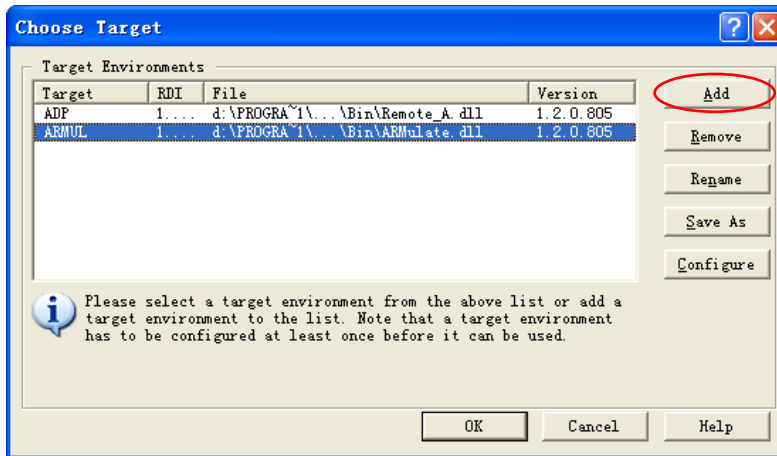


Figure 4.3 Choose Target window

The system will pop-up dialog box, open TKScope emulator driver installation directory (Example is D:\Keil\TKScope), select TKSCP_DRV_for_RDI.dll, as shown in Figure 4.4.

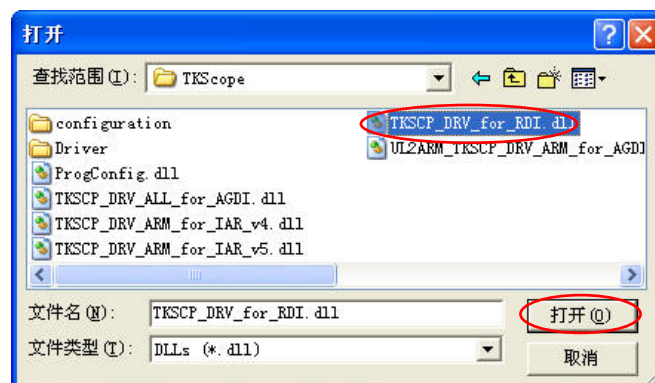


Figure 4.4 Select driver file

[Choose Target] window displays the currently installed driver options, as shown in Figure 4.5. Select TKScope emulator driver, click [Configure] to into the emulator settings interface, as shown in Figure 4.6.

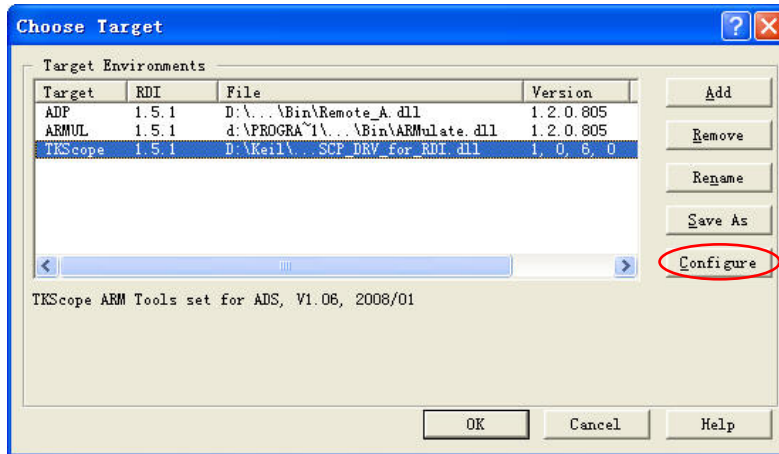


Figure 4.5 Driver file installed

The TKScape emulator settings interface is the same in any IDE environment (For example, Figure 4.6 in ADS and Figure 3.3 in Keil RealView MDK), setting methods are the same.

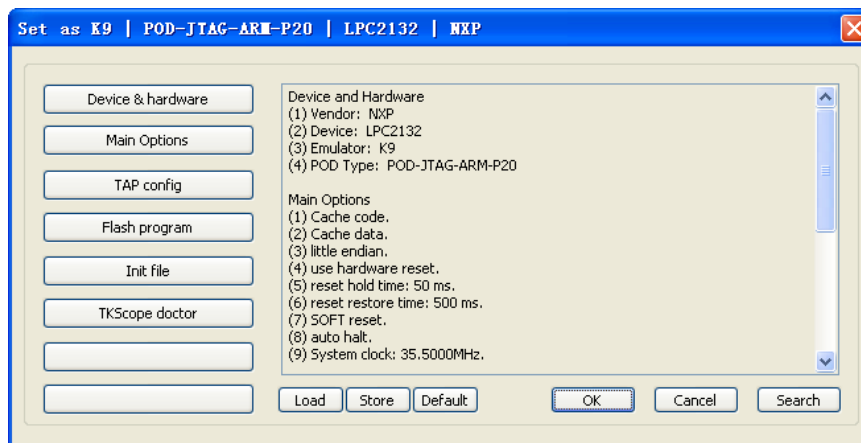


Figure 4.6 TKScape emulator settings interface

TKScope emulator must be set correctly, otherwise, may result in simulation errors or failure! The setting methods are not repeated described here, users please refer to the 3.2 section《Emulator Settings》.








4.2 Debugging

4.2.1 Debugging Tools

AXD debugging environment provides the following debugging tools, as shown in Figure 4.7.





Figure 4.7 Debugging Tools

-  Go
-  Stop
-  Step In
-  Step
-  Step Out
-  Run To Cursor
-  Toggle BreakPoint

AXD debugging environment provides the following watching tools, as shown in Figure 4.8.



Figure 4.8 Watching Tools

-  Processor Registers
-  Processor Watch
-  Context Variable
-  Memory
-  Disassembly

AXD debugging environment provides the following file operation tools, as shown in Figure 4.9.



Figure 4.9 File OperationTools



Load Image



Reload Current Image

4.2.2 Debugging Results

After debugging ended, users close AXD environment directly to exit debugging status.

If users debug in RAM, programs written in the chip RAM, will lost after power down. Programs will not run in target board if power up again.

If users debug in Flash, programs written in the Flash, will save after power down. Programs will run in target board if power up again.

If users debug in Flash and encryption, can not debug again. Unless, global erase chip, will be re- debugging.

Chapter 5 Simulation ARM In IAR

5.1 Add Driver File

In IAR environment, open the project to compile ok. Select the project, click the right mouse button, then select [Options], as shown in Figure 5.1.

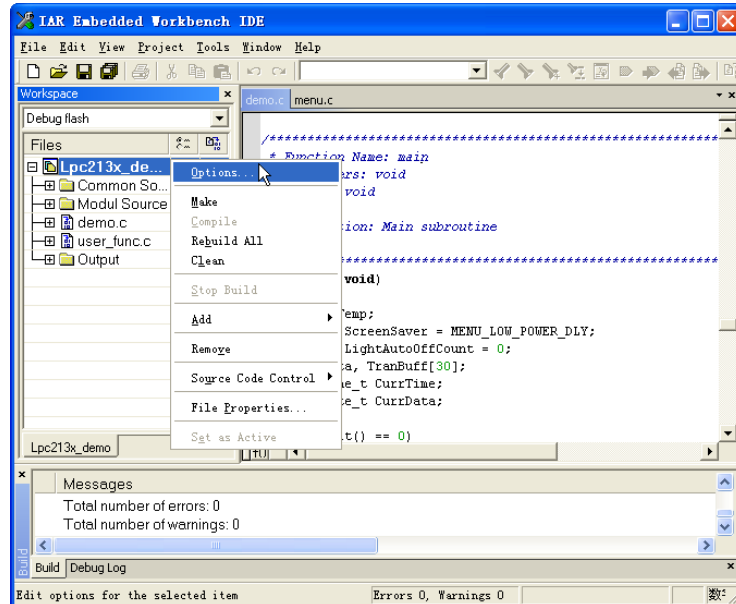


Figure 5.1 IAR Main Interface

Select [Debugger] option, [Setup] window in right side is set as shown in Figure 5.2. [Driver] select [Third-Party Driver], and select [Run to main].

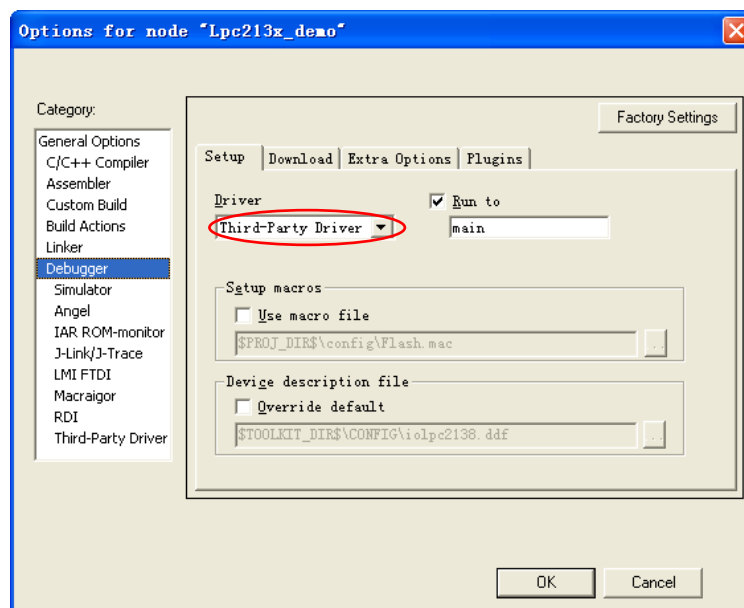


Figure 5.2 Debugger Setup Interface

In Figure 5.2, click [Download] option, settings interface is as shown in Figure 5.3, all the options are not selected.

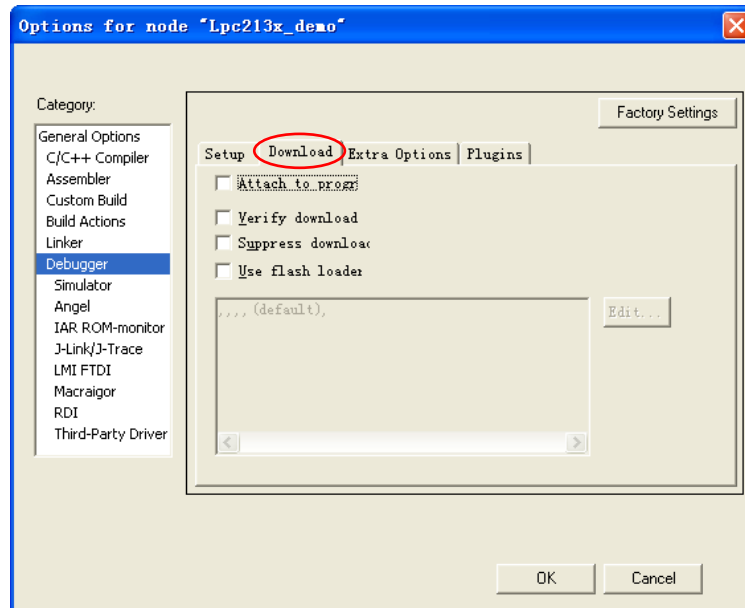



Figure 5.3 Debugger Download Interface

Select [Third-Party Driver] option, the interface is as shown in Figure 5.4. Click  icon to add TKScope emulator driver file.

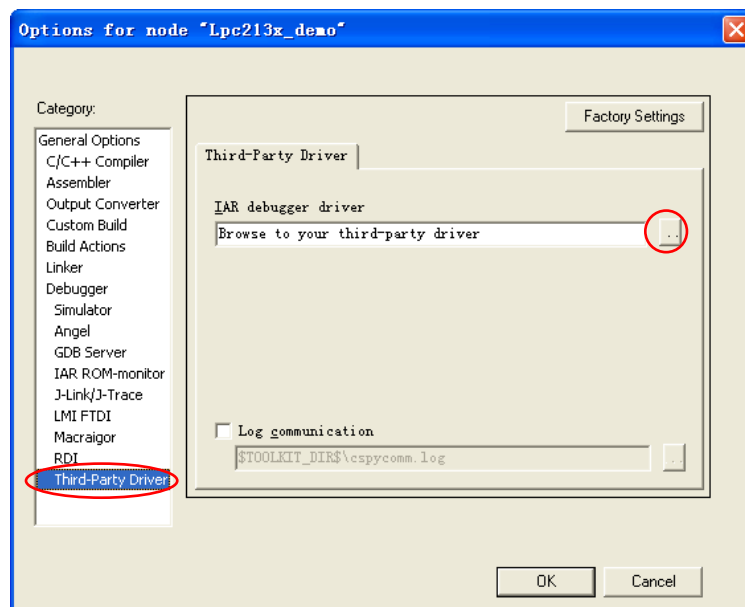


Figure 5.4 Third-Party Driver Interface

The system will pop-up dialog box, open TKScope emulator driver installation directory (Example is D:\Keil\TKScope).

If you used IAR V4, select TKSCP_DRV_ARM_for_IAR_v4.dll.

If you used IAR V5, select TKSCP_DRV_ARM_for_IAR_v5.dll.

For example, used IAR V4.42, driver file selected result is as shown in Figure 5.5.

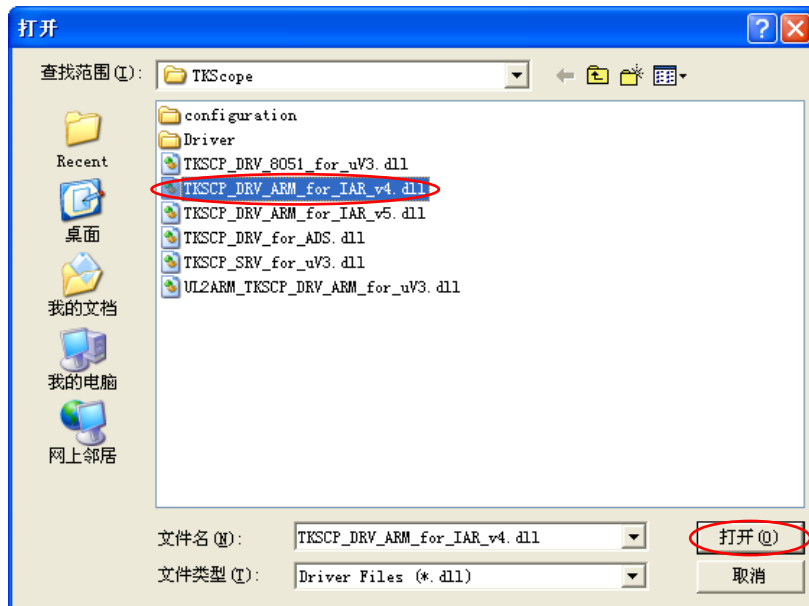


Figure 5.5 Select driver file

[Third-Party Driver] window displays the currently installed driver options, as shown in Figure 5.6.

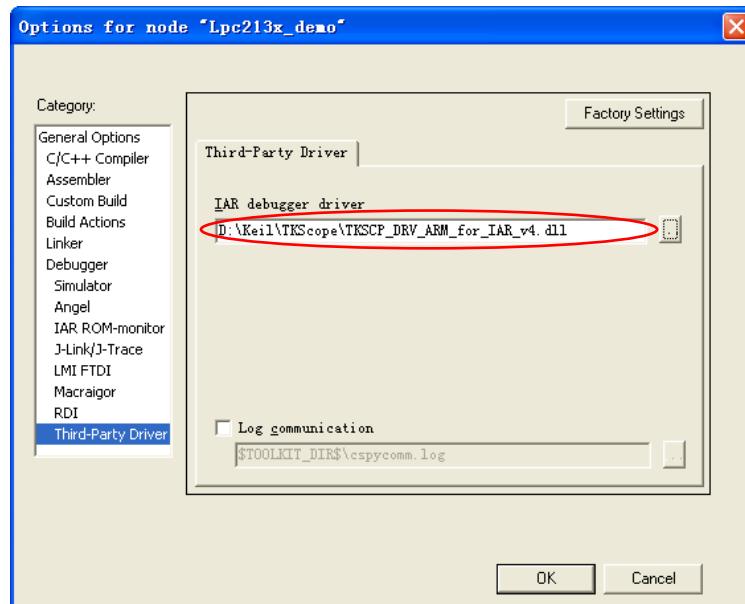


Figure 5.6 Driver file installed

IAR main menu will display [TKScope] option after driver installed, as shown in Figure 5.7.

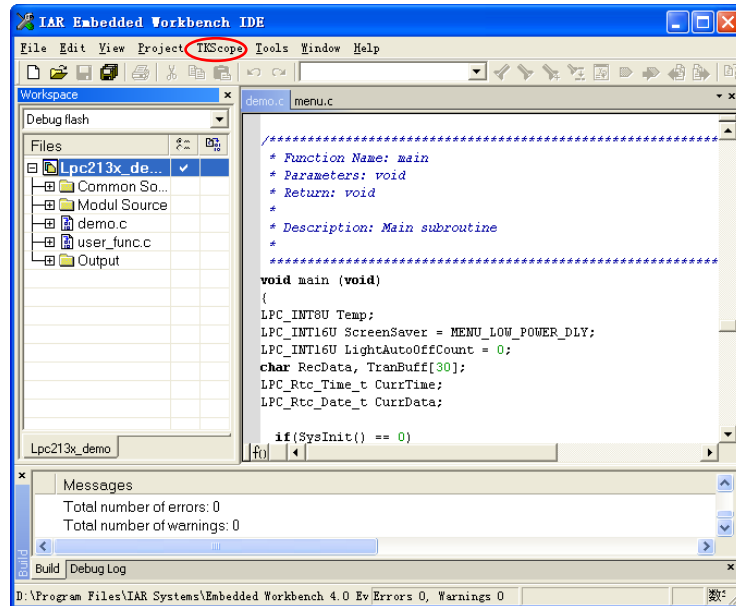


Figure 5.7 Main Interface Installed Driver

Select [TKScope] menu [Setup] option, as shown in Figure 5.8, into the emulator settings interface.

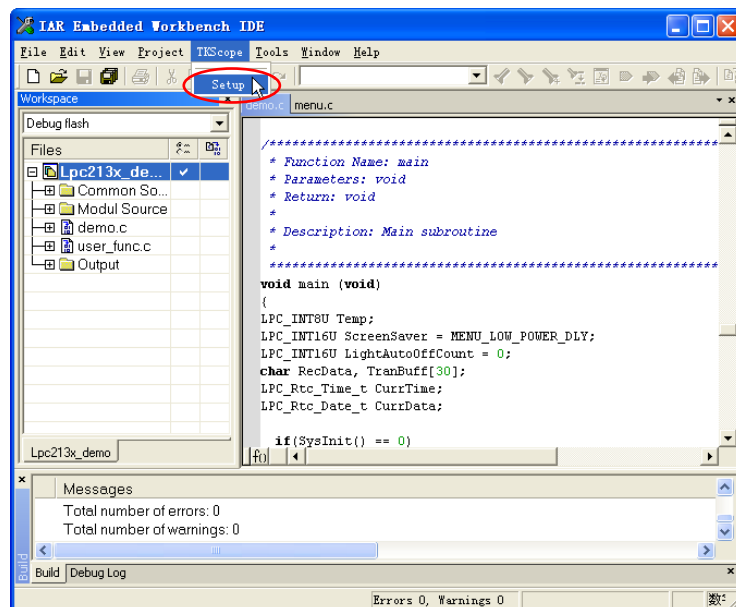


Figure 5.8 Select TKScope Setting

The TKScope emulator settings interface is the same in any IDE environment (For example, Figure 5.9 in IAR and Figure 3.3 in Keil RealView MDK), setting methods are the same.

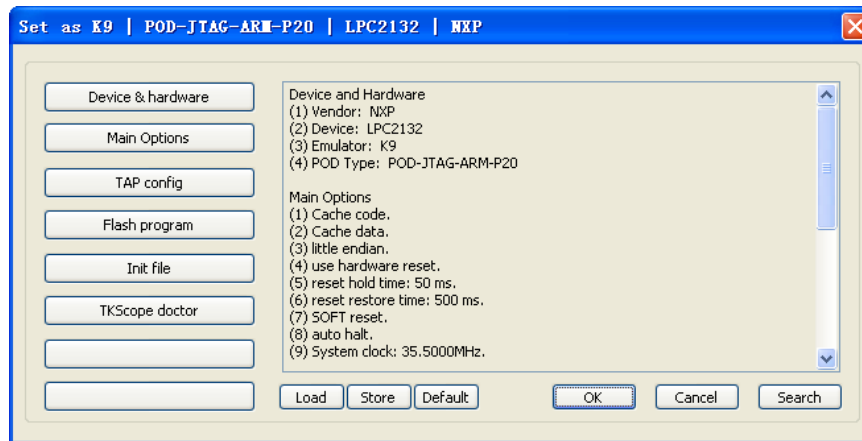


Figure 5.9 TKScope emulator settings interface

TKScope emulator must be set correctly, otherwise, may result in simulation errors or failure! The setting methods are not repeated described here, users please refer to the 3.2 section《Emulator Settings》.

5.2 Debugging

5.2.1 Start Debugging

Users can debug after the completion of the emulator settings.

Click [Project] menu [Debug] option, or click  icon, into debugging status, as shown in Figure 5.10.

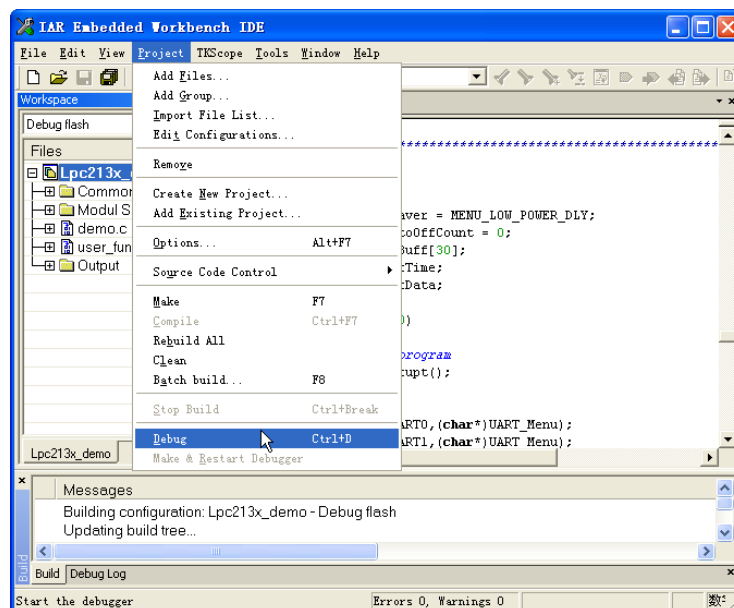


Figure 5.10 Start Debugging

5.2.2 Debugging Tools

IAR debugging environment provides the following debugging tools, as shown in Figure 5.11.

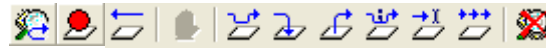













Figure 5.11 Debugging Tools

-  Make and Debug
-  Toggle Breakpoint
-  Reset
-  Break
-  Step Over
-  Step Into
-  Step Out
-  Next Statement
-  Run to Cursor
-  Go
-  Stop Debugging

Users can open windows under [View] menu to watch simulation results in debugging, as shown in Figure 5.12.

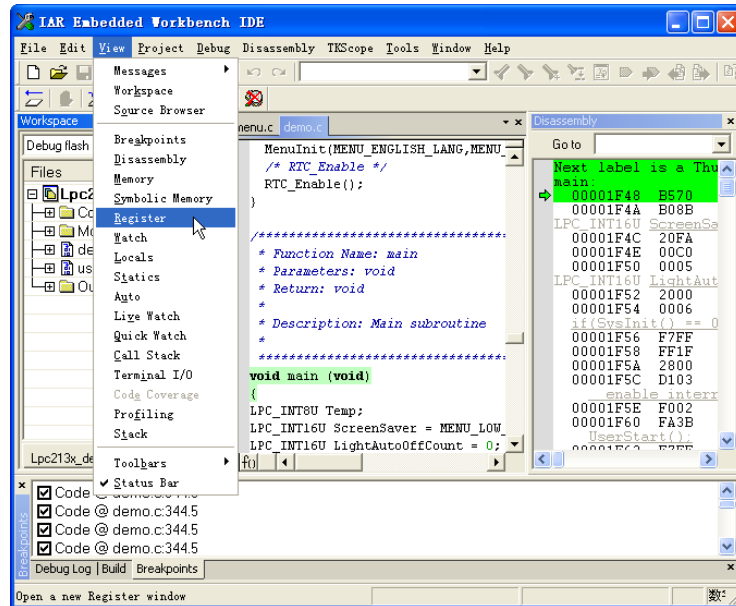


Figure 5.12 Open View Windows

5.2.3 Debugging Results

After debugging ended, users click  icon to exit debugging status.

If users debug in RAM, programs written in the chip RAM, will lost after power down. Programs will not run in target board if power up again.

If users debug in Flash, programs written in the Flash, will save after power down. Programs will run in target board if power up again.

If users debug in Flash and encryption, can not debug again. Unless, global erase chip, will be re- debugging.