



# **H-JTAG**

# **USER MANUAL**

Doc Edition      L

Release Date:    2014-02-28

**[WWW.HJTAG.COM](http://WWW.HJTAG.COM)**

---

## **H-JTAG USER MANUAL**

Copyright © 2014 H-JTAG All Rights Reserved

### **Release Information**

Date	Issue	Change
2007-10-01	A	Release first edition
2007-11-30	B	Revised edition
2008-03-03	C	Corrected the illustration of TAP configuration
2009-01-08	D	Updated for new software version
2009-04-24	E	Updated for new software version
2009-10-08	F	Updated for new software version
2010-01-20	G	Updated for new software version
2010-06-01	H	Updated for new software version
2010-08-20	I	Updated for new software version
2011-06-01	J	Updated for new software version
2012-02-15	K	Updated for new software version
2014-02-28	L	Updated for new software version

### **Property Notice**

1. JTAG is a standard (IEEE-1149) proposed by IEEE. The copyright belongs to IEEE.
2. All the registered trademarks and logos mentioned in this manual belong to their respective owners.
3. All the products and services described in this manual also belong to their respective owners.
4. If this manual harms your copyright, please contact us and we will make the correction accordingly.
5. This manual is an open document. User can redistribute it freely if only the integrity can be guaranteed.

### **Official Website**

[HTTP://WWW.HJTAG.COM](http://WWW.HJTAG.COM)

### **Support**

[HTTP://FORUM.HJTAG.COM](http://FORUM.HJTAG.COM)

[SUPPORT@HJTAG.COM](mailto:SUPPORT@HJTAG.COM)

---

# Contents

## Preface

A	About this Manual .....	IV
B	Using this Manual .....	IV
C	Feedback .....	IV

## Chapter 1 H-Jtag Introduction

1.1	About H-Jtag .....	1-1
1.2	Communication Structure .....	1-1
1.3	Supported ARM Cores.....	1-2
1.4	Supported IDEs.....	1-2

## Chapter 2 Installation and GUI

2.1	Installation .....	2-1
2.2	Uninstallation .....	2-3
2.3	GUI of H-Jtag .....	2-3
2.4	GUI of H-Flasher .....	2-8

## Chapter 3 H-Jtag USB Emulator

3.1	Hardware Feature and Interface .....	3-1
3.2	Different Emulator Editions .....	3-3
3.3	Driver Installation .....	3-4

## Chapter 4 Configure H-Jtag

4.1	Detect Target .....	4-1
4.2	Reset Target .....	4-1
4.3	Auto Flash Download .....	4-1
4.4	Initialization Script .....	4-2
4.5	USB/LPT Interface Selection .....	4-3
4.6	JTAG Configuration .....	4-3
4.7	LPT Port Setting .....	4-6
4.8	Target Setting .....	4-6
4.9	Target Manager .....	4-7
4.10	TAP Configuration .....	4-7
4.11	H-Jtag Options .....	4-8
4.12	H-Jtag Tools .....	4-10
4.13	Check for Updates .....	4-10

## Chapter 5 Configure H-Flasher

5.1	H-Flasher Configuration File .....	5-1
5.2	H-Flasher Production Mode.....	5-1
5.3	Workflow of H-Flasher .....	5-2

---

5.4	H-Flasher Program Wizard .....	5-3
5.5	Useful Tips .....	5-9
5.6	Example 1 - AT91SAM7X256 .....	5-10
5.7	Example 2 - LPC2210 + SST39VF1601.....	5-14
<b>Chapter 6</b>	<b>Initialization Script</b>	
6.1	Definition of Script Commands .....	6-1
6.2	Edit Initialization Script .....	6-4
<b>Chapter 7</b>	<b>Configure Debuggers</b>	
7.1	Configure AXD .....	7-1
7.2	Configure RVDS .....	7-4
7.3	Configure IAR .....	7-8
7.4	Configure KEIL/MDK .....	7-12
<b>Chapter 8</b>	<b>Auto Flash Download</b>	
8.1	Example: Auto Flash Download For LPC1766 .....	8-2
<b>Chapter 9</b>	<b>H-Flasher Production Mode</b>	
<b>Appendix A H-JTAG Q&amp;A</b>		
<b>Appendix B List of Chips Supported by H-JTAG</b>		

---

## Preface

### A. About this manual

H-Jtag user manual introduces how to configure and use H-Jtag and H-Flasher. Some illustrative examples are also given in this manual for reference. For more information, please visit [www.hjtag.com](http://www.hjtag.com) or [forum.hjtag.com](http://forum.hjtag.com).

### B. Using this manual

This manual is intended to assist user in the use of H-Jtag and H-Flasher. If you are a beginner, this manual is a good quick start guide. If you are a advanced user, you can use this manual as a reference and read it selectively.

### C. Feedback

If you find any error or omission in this document, please contact us. Any suggestions and comments are welcome. The contact email address is [twentyone@hjtag.com](mailto:twentyone@hjtag.com).

---

# Chapter 1 H-Jtag Introduction

## 1.1 About H-Jtag

H-Jtag is a debug agent, like the popular Multi-ICE. H-Jtag includes three tools, H-Jtag server, H-Flasher and H-Converter (Fig 1-1). H-Jtag server is a debug agent, H-Flasher is a flash programmer, and H-Converter is a conversion tool, which supports different file formats like BIN, HEX and ELF.



Fig 1-1 H-JTAG Structure

H-Jtag supports the debug of all the CORTEX-M0, CORTEX-M3, CORTEX-M4, ARM7, ARM9, ARM11 and XSCALE based chips and can be used with most of the popular debuggers, including ADS, RVDS, IAR and KEIL/MDK. H-Jtag provides flexible configuration, with which H-Jtag can work with H-JTAG USB emulator, Wiggler, SDT-Jtag and other user-defined JTAG interface boards. The integrated H-Flasher supports the programming of most flash chips. With H-Jtag, it is easy to build up a debug platform. Summed up, H-Jtag has following features.

1. Support wide range of ARM processors: ARM7, ARM9, ARM11, XSCALE, PXA3XX, CORTEX-M0, CORTEX-M3 and CORTEX-M4.
2. Support all the popular IDEs: ADS1.2, IAR, KEIL/MDK and RVDS2.2.
3. Support flexible target initialization.
4. Support both THUMB and ARM mode.
5. Support Little-Endian and Big-Endian.
6. Support high speed H-JTAG USB emulator, provide reliable and high-performance solution.
7. Support LPT port, provide a low cost and reliable solution.
8. Support different Windows platforms: NT, 2000, XP, VISTA and WINDOWS7, WINDOWS8.
9. Support the programming of ON-CHIP Flash, NOR Flash, NAND Flash and SPI Flash.
10. Support auto flash download during debugging.
11. Support production mode for maximum efficiency.

## 1.2 Communication Structure

H-Jtag supports RDI interface from ARM Limited. Through the RDI interface, H-Jtag can support most of the popular debuggers. The connection structure for debugging is shown in Fig 1-2.

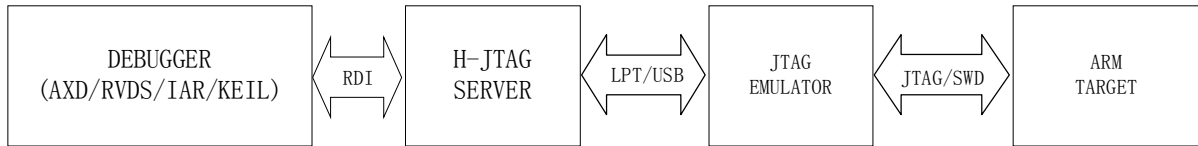


Fig 1-2 Connection for Debugging

Debuggers communicate with H-Jtag server via RDI. H-Jtag server accesses the JTAG/SWD port of target system through the JTAG controller connected to LPT/USB. With proper configuration, H-Jtag can work with H-JTAG USB emulator and different LPT emulators.

Besides debugging, H-Flasher can download data/application into flash chips. H-Flasher supports most on-chip flashes and external flashes. For flash programming, the connection between H-Flasher and H-Jtag server is shown in Fig 1-3.

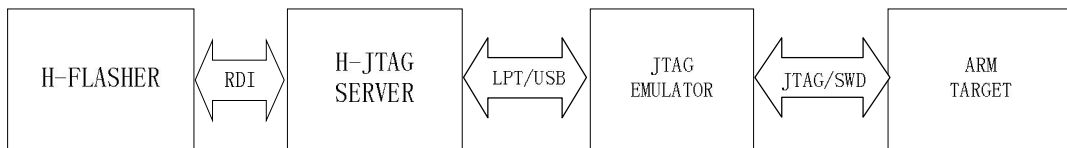


Fig 1-3 Connection between H-Flasher and H-Jtag Server

As shown in the above figure, H-Flasher also talks to H-Jtag server via RDI, which is similar to that for debugging.

### 1.3 Supported ARM Cores

H-JTAG supports the debugging and programming of all the popular ARM cores.


CORTEX-M0, CORTEX-M3, CORTEX-M4

ARM7TDMI, ARM7TDMI-S, ARM720T, ARM740T

ARM9TDMI, ARM920T, ARM922T, ARM940T, ARM926EJ-S, ARM946E-S, ARM966E-S

ARM1136, ARM1176

PXA21X, PXA25X, PXA27X, PXA3XX, IXP4XX, IXP2XXX

 **Note:** Detailed list of chips supported by H-JTAG can be found in the Appendix.

### 1.4 Supported IDEs

H-JTAG supports all the popular IDEs, which include SDT2.5, ADS1.2, RVDS2.0, RVDS2.2, KEIL/MDK and IAR Embedded Workbench.

---

## Chapter 2 Installation and GUI

This chapter introduces both the installation of H-JTAG and the software GUI. For the detailed configurations, please refer to Chapter 4-7.

### 2.1 Installation

User can download the latest version installation file from [www.hjtag.com](http://www.hjtag.com). Normally, the downloaded file is zipped and need to be unzipped first. After unzipped, user can get the installation exe file `h-jtag.exe`. By double clicking the exe file, the installation can be started.

First, user will see the welcome dialog, as shown in Fig 2-1. Click “Next” and go to next step.



Fig 2-1 Installation Step-1

In the second step, user will see the license agreement, as shown in Fig 2-2. Please read the agreement carefully. To accept it, tick “I agree with the above terms and conditions.” Then click “Next”. Otherwise, click “Exit” to exit the installation.

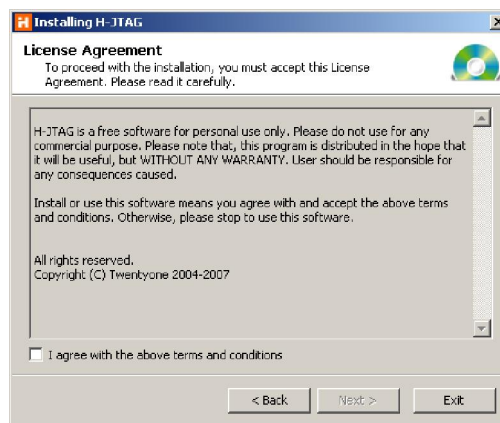


Fig 2-2 Installation Step-2

In the third step, user will see the destination folder dialog (shown in Fig 2-3). In this dialog, user can choose the destination folder. After choosing the destination folder, click “Next” and go to next step.



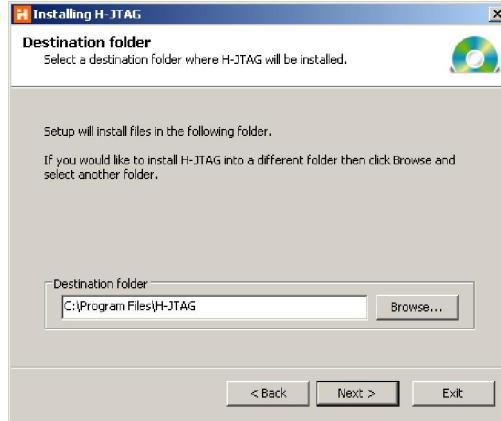


Fig 2-3 Installation Step-3

In step-4, the progress dialog shown in Fig 2-4 will be seen. This dialog shows the progress of installation. In this step, H-Jtag will install all the files automatically.

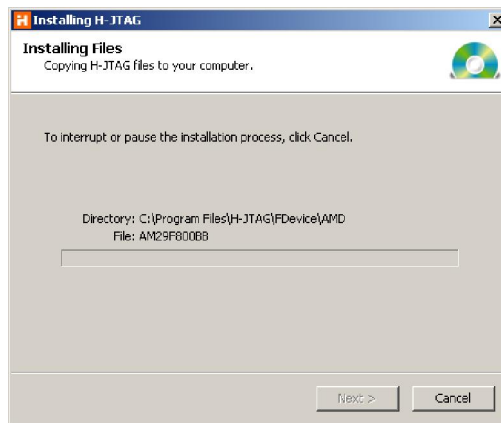


Fig 2-4 Installation Step-4

In the final step, user will see the dialog shown in Fig 2-5. The dialog indicates that H-Jtag has been installed successfully. To complete the installation, click "Finish".

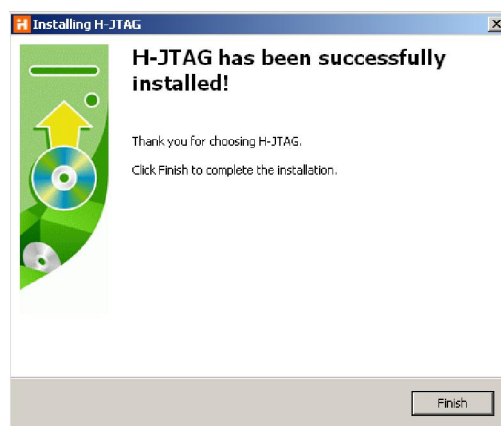


Fig 2-5 Installation Step-5

After the installation, shortcuts will be created on the desktop and the start menu respectively. The shortcuts are shown in Fig 2-6.

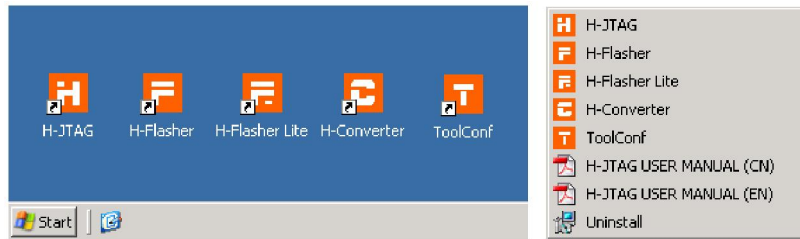


Fig 2-6 Shortcuts of H-Jtag

**Note:**

The driver for H-JTAG USB emulator needed to be installed manually. Please refer to Chapter-3 on how to install the driver.

## 2.2 Uninstallation

To uninstall H-Jtag, please run `uninstall.exe` from the start menu. This program will uninstall H-Jtag automatically. During the process, please follow the instructions to perform the uninstallation.

## 2.3 GUI of H-Jtag

The main window of H-Jtag is shown in Fig 2-7. In the figure, (1) is the menu bar, (2) is the toolbar, (3) is the detected ARM core, (4) is the device ID and (5) shows the connected debugger, RDI version, TCK speed and hardware interface.

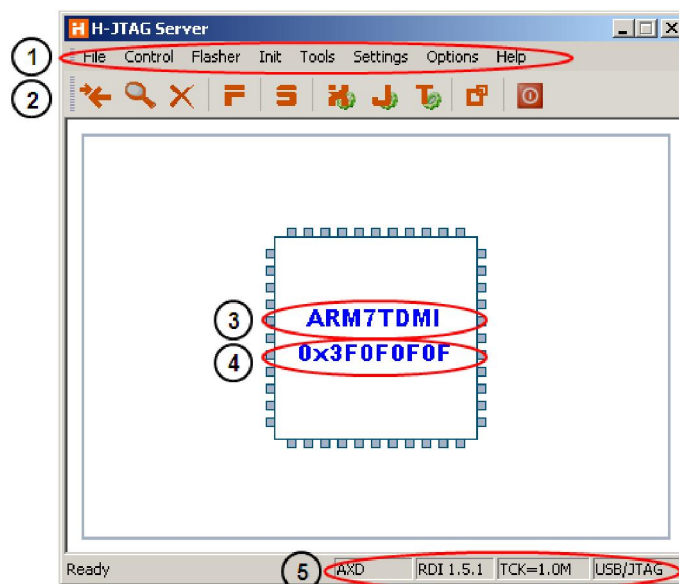


Fig 2-7 Main window of H-Jtag

The menu includes all the operations and configurations, and the toolbar includes most of the common ones. When a connected target is detected, H-Jtag displays the ARM core and 32-bit device ID in the middle of the main window. When no target is detected or the target is unrecognized, H-Jtag displays UNKNOWN.

---

### 2.3.1 Menu of H-Jtag

- File Menu, as shown in following figure.



Fig 2-8 File Menu

-  Exit – Exit H-Jtag.

- Control Menu, as shown in Fig 2-9.

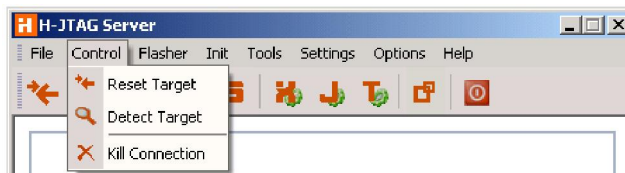





Fig 2-9 Control Menu

-  Reset Target – Reset target
-  Detect Target – Detect target
-  Kill Connection – Kill current connection

- Flasher menu, as shown in Fig 2-10.

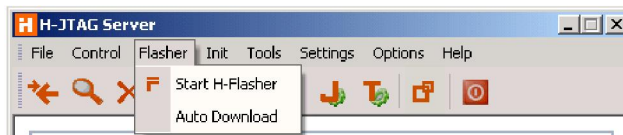




Fig 2-10 Flasher Menu

-  Start H-Flasher – Start H-Flasher
-  Auto Download – Enable/Disable auto flash download

- Init Menu, as shown in Fig 2-11.

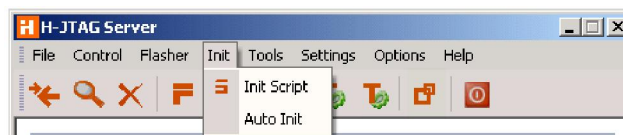




Fig 2-11 Init Menu

-  Init Script – Configure init script
-  Auto Init – Enable/Disable auto init

- Tools Menu, as shown in Fig2-12.

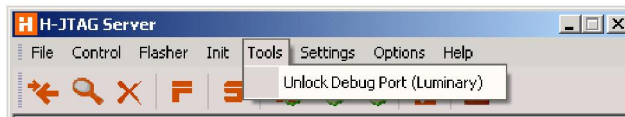


Fig 2-12 Tools Menu

- ✚ Unlock Debug Port – Unlock debug port for Luminary series

- Settings Menu, as shown in Fig2-13.

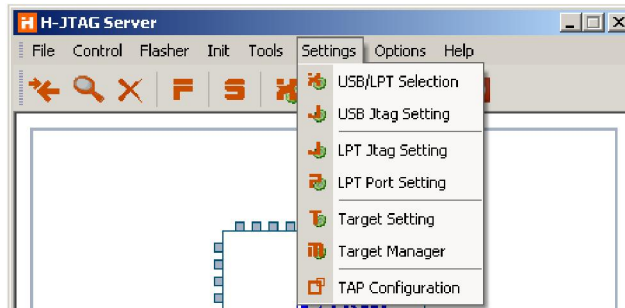


Fig 2-13 Settings Menu

- ✚ USB/LPT Selection – Selection of hardware interface
  - ✚ USB JTAG Setting – JTAG setting for USB
  - ✚ LPT JTAG Setting – JTAG setting for LPT
  - ✚ LPT PORT Setting – Port setting for LPT
  - ✚ Target Setting – Target setting
  - ✚ Target Manager – Device ID manager
  - ✚ TAP Configuration – TAP configuration

- Options menu, as shown in Fig2-14.

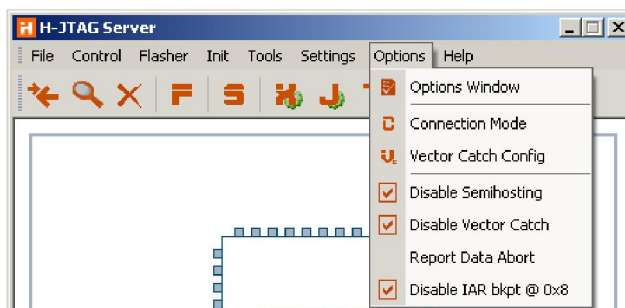


Fig 2-14 Options Menu

- ✚ Options Window – Open the main window of options
  - ✚ Connection Mode – Selection of different connection modes
  - ✚ Vector Catch Config – Configuration for vector catch
  - ✚ Disable Semihosting – Disable semihosting function
  - ✚ Disable Vector Catch – Disable vector catch function
  - ✚ Report Data Abort – Enable/Disable report of data abort

- ✚ Disable IAR bkpt @ 0x8 – Disable the breakpoint set at 0x8 by IAR

- Help menu, as shown in Fig 2-15.

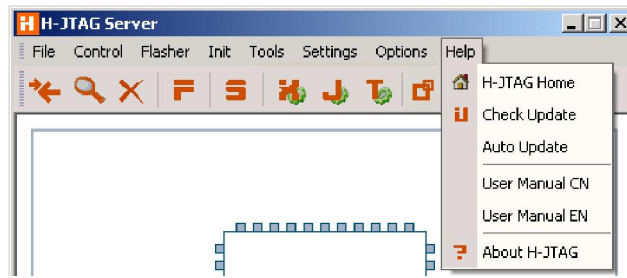


Fig 2-15 Help Menu

- ✚ H-JTAG Home – Visit H-Jtag homepage
- ✚ Check Update – Check for updates
- ✚ Auto Update – Enable/Disable automatic check for updates
- ✚ User Manual CN – Chinese user manual
- ✚ User Manual EN – English user manual
- ✚ About H-JTAG – Information about H-Jtag

### 2.3.2 H-Jtag Toolbar

This section gives a brief introduction to the toolbar. The toolbar of H-Jtag includes most of the common operations and settings. The toolbar is shown in Fig 2-16.



Fig 2-16 Toolbar of H-Jtag

The definition and function of each button is shown below.

- ✚ Reset target
- ✚ Detect target
- ✚ Kill current connection
- ✚ Start H-Flasher
- ✚ Configure init script
- ✚ Selection of USB/LPT
- ✚ JTAG setting for USB
- ✚ Configure target
- ✚ Configure TAP
- ✚ Exit H-Jtag

---











### 2.3.3 H-Jtag Tray Menu

When H-Jtag is minimized, the main window is hidden automatically and only an icon is shown in the system tray. The main window can be restored by left clicking the icon. Right clicking the icon, the system tray menu is popped up. The tray menu includes some common operations and settings.



Fig 2-17 System Tray Menu of H-Jtag

The tray menu is defined as follows.

- |   |                             |
|---|-----------------------------|
|  Restore          | – Restore the main window   |
|  H-JTAG Home     | – Visit homepage of H-Jtag  |
|  About H-JTAG    | – Information about H-Jtag. |
|  Options         | – Options menu              |
|  Script          | – Script menu               |
|  Flasher         | – H-Flasher menu            |
|  Kill Connection | – Kill current connection   |
|  Detect Target   | – Detect target             |
|  Reset Target    | – Reset target              |
|  Exit            | – Exit H-Jtag               |

---

## 2.4 GUI of H-Flasher

The main window of H-Flasher looks like Fig 2-18. In the figure, (1) is the menu bar, (2) is the program wizard and (3) is the configuration window. In the wizard, user can go to different steps. According to the selection on the wizard, the configuration window has different displays. For detailed information, please refer to Chapter 5.

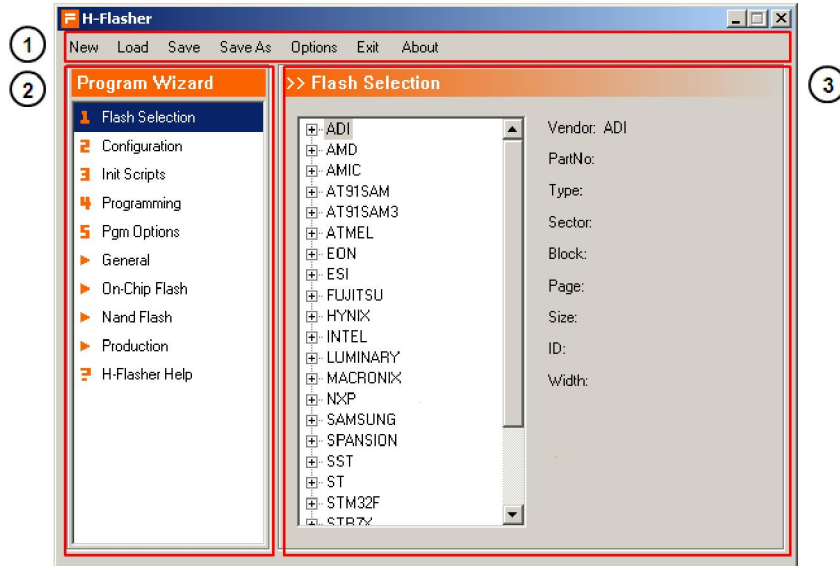


Fig 2-18 Main Window of H-Flasher

### 2.4.1 Menu of H-Flasher

The menu of H-Flasher is shown in Fig 2-19.

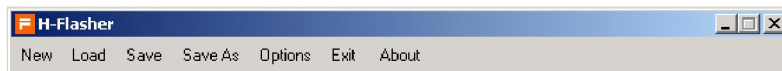








Fig 2-19 Menu of H-Flasher

The menu is defined as follows.

-  Load – Load configuration file
-  Save – Save current configuration
-  SaveAs – Save current configuration as another file
-  Options – Select different options
-  Exit – Exit H-Flasher
-  About – Information about H-Flasher.

### 2.4.2 H-Flasher Tray Menu

When H-Flasher is minimized, the main window is hidden automatically and only an icon is shown in the system tray. The main window can be restored by left clicking the icon. Right clicking the icon, the system tray menu is popped up.

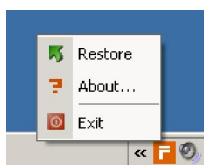





Fig 2-20 H-Flasher Tray Menu

The tray menu is defined as follows.

-  Restore – Restore the main window
-  About – Information about H-Flasher
-  Exit – Exit H-Flasher

### 2.4.3 H-Flasher Program Wizard

The program wizard includes 5 steps and 1 help section. This section gives a brief introduction to the program wizard.

#### (1) Flash Selection

Flash selection is the first step of the program wizard, as shown in Fig 2-21. In this step, all the supported flash chips are categorized by vendors. User needs to specify the target flash chip. When a chip is selected, the basic information of the chip is displayed on the right side.

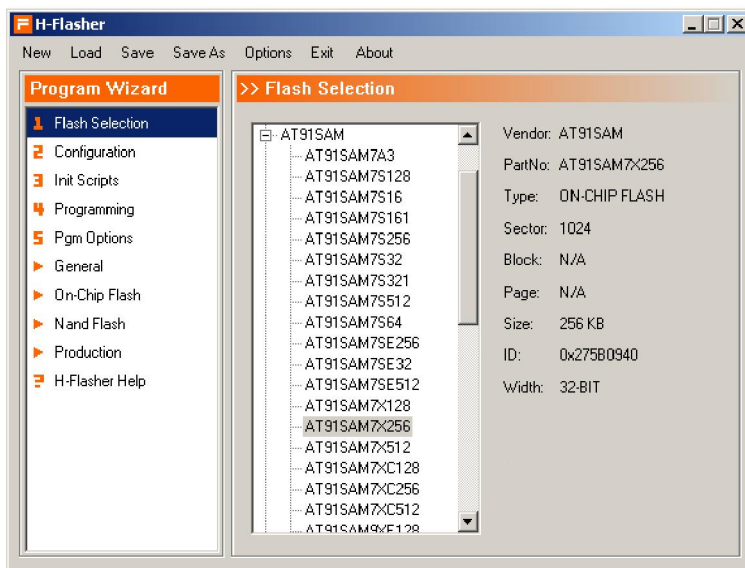


Fig 2-21 Program Wizard – Flash Selection



## (2) Configuration

Configuration is the second step of the wizard. The display is shown in Fig 2-22. In this step, user needs to provide the basic addressing information, for example, bit width, flash start address, RAM start address, XTAL, init TCK and program TCK. All this information is a must. In this step, if an input area is in gray, it means no input is required. Normally, the information is fixed for on-chip flash.

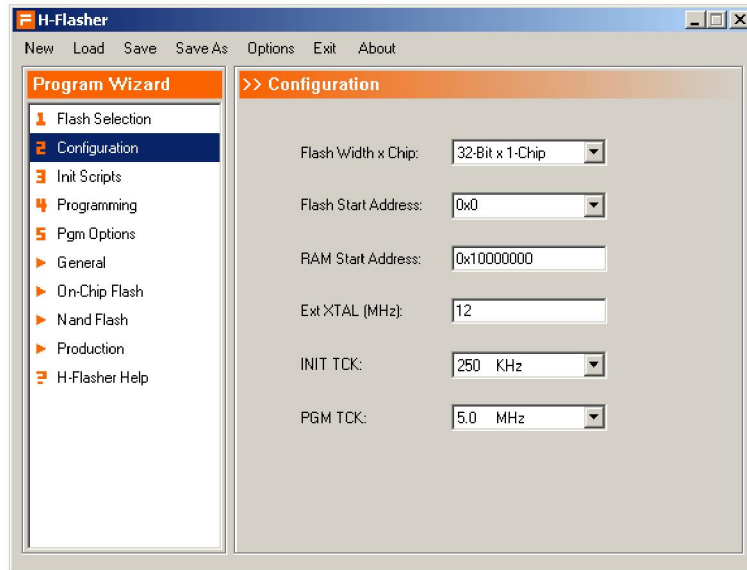


Fig 2-22 Program Wizard – Configuration

## (3) Init Script

Init script is the third step of the programming wizard. In this step, the display is shown in Fig 2-23. User can input any scripts for initialization. During the programming, H-Flasher first executes the provided init script in order to initialize the target system. For on-chip flash, the initialization has been included in the flash driver, so no additional init script is needed. In this case, the buttons for editing are all disabled.

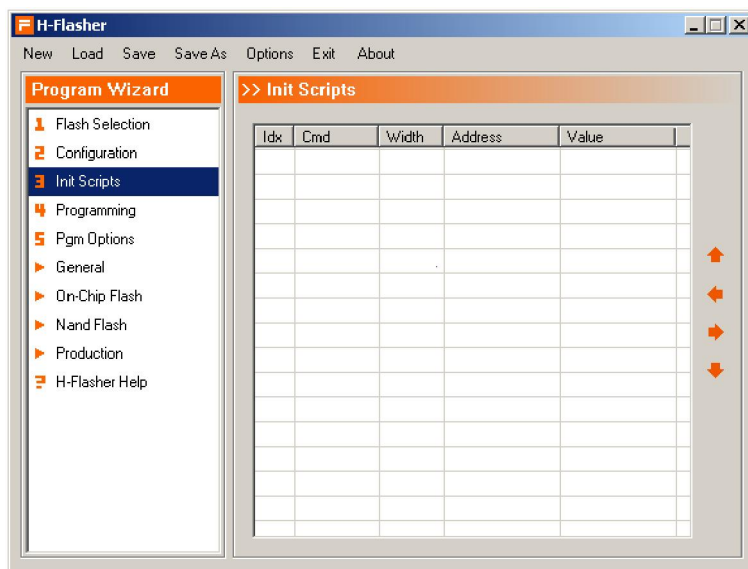


Fig 2-23 Program Wizard – Init Script

## (4) Programming

Programming is the fourth step of the wizard. In this step, user can operate on the target flash, like reset target, check flash, write/erase flash, verify and check if it is blank. Depends on the type of the selected flash, user can see different dialogs, as shown in Figs 2-24, 2-25 and 2-26.

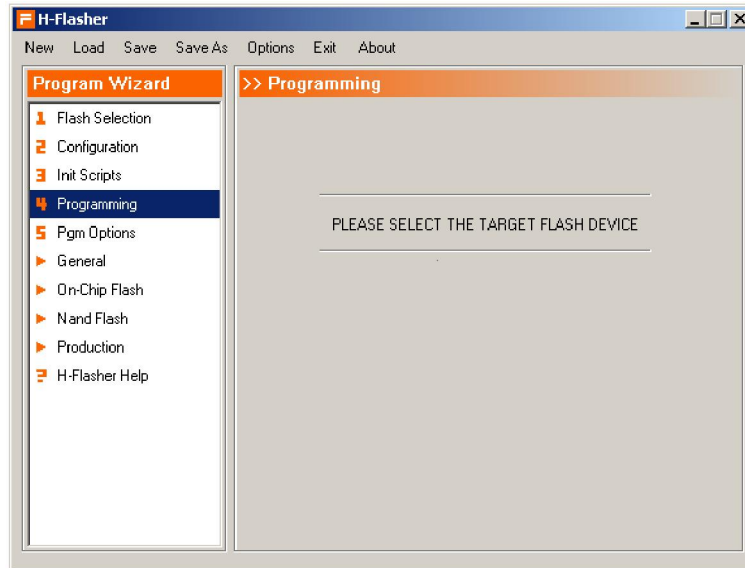


Fig 2-24 Program Wizard – Programming (No Chip Is Selected)

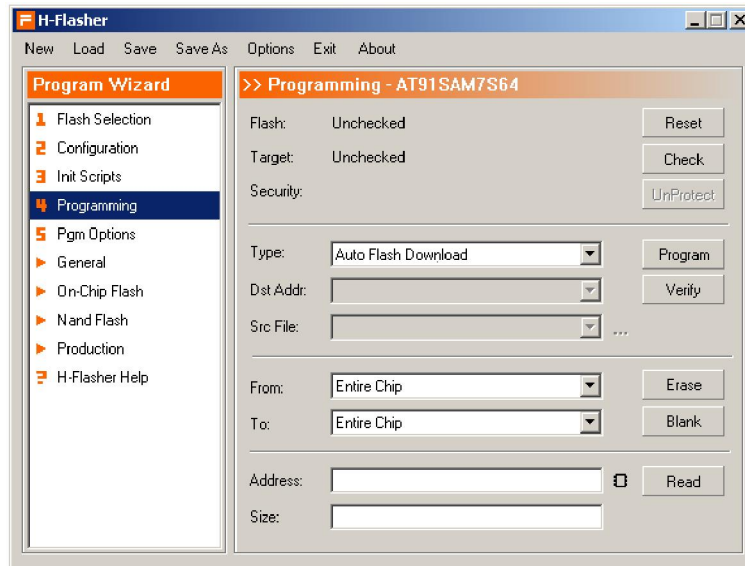


Fig 2-25 Program Wizard – Programming (On-Chip Flash, Nor Flash or Spi Flash Is Selected)

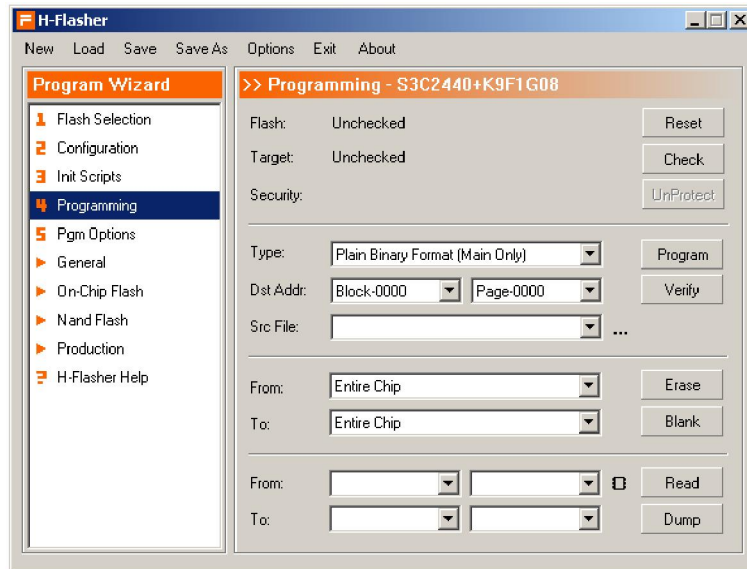


Fig 2-26 Program Wizard – Programming (Nand Flash Is Selected)

## (5) Program Options

This is the fifth step of the programming wizard. As shown in following figures, user can make different selections based on the selected FLASH chip.

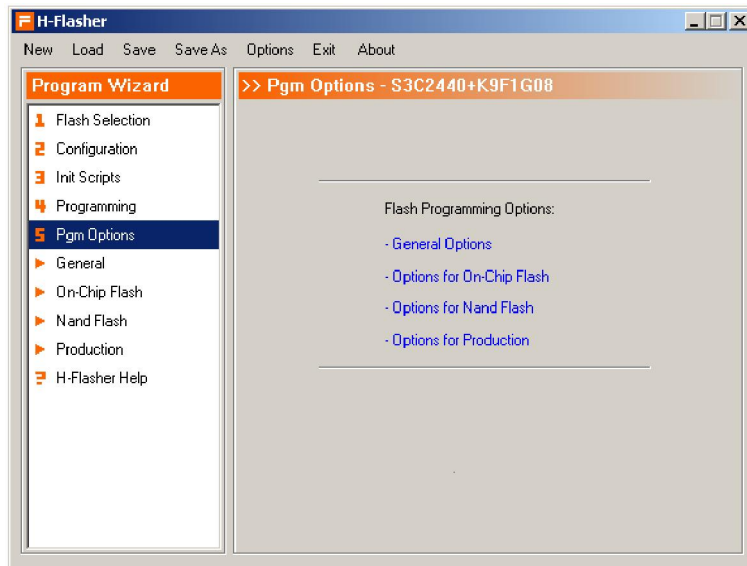


Fig 2-27 Program Wizard – Program Options

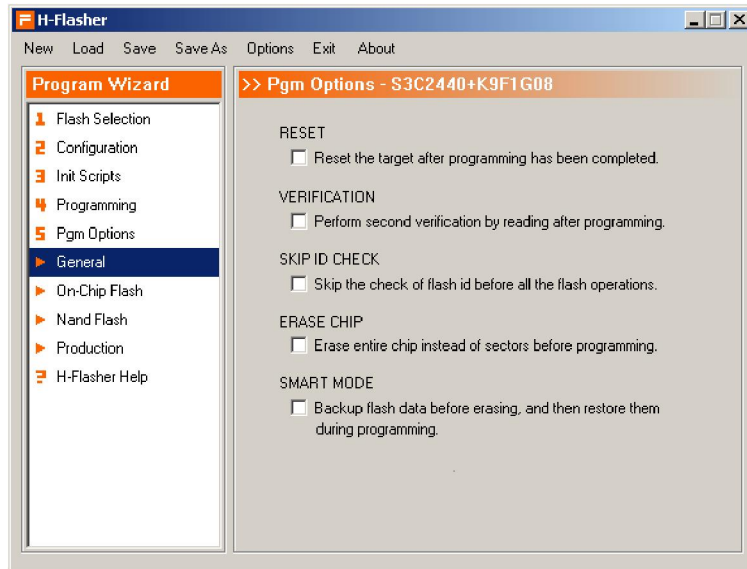


Fig 2-28 Program Wizard – General Options

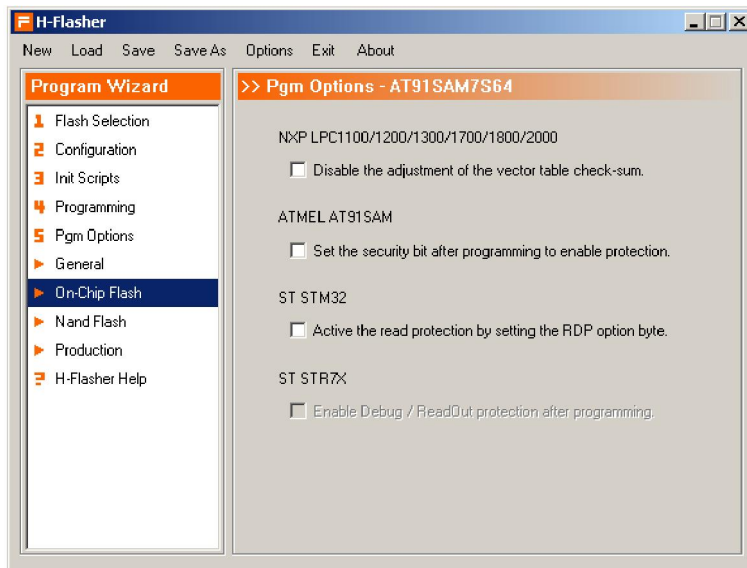


Fig 2-29 Program Wizard – On-Chip Flash Options

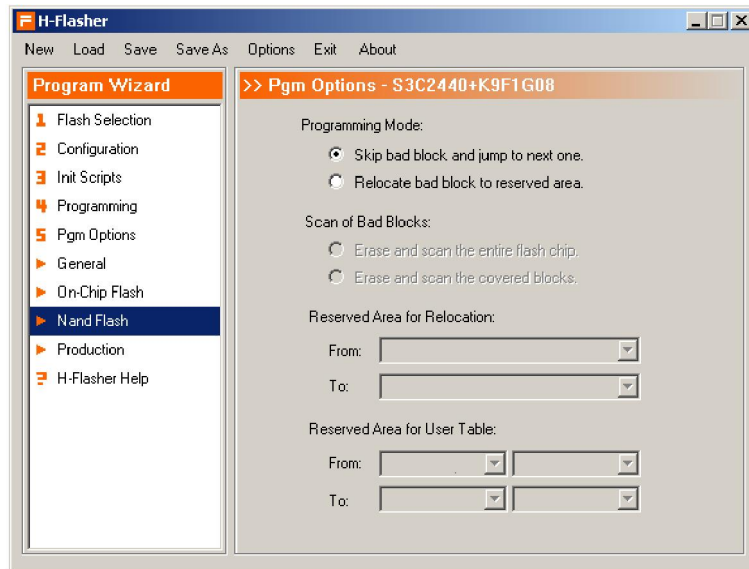


Fig 2-30 Program Wizard – Nand Flash Options

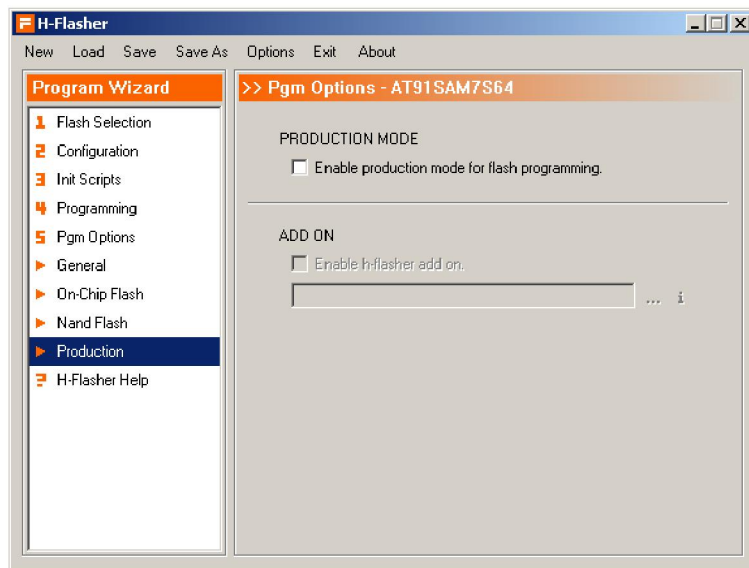


Fig 2-31 Program Wizard – Production Options

---

## (6) Help

In the wizard, a help section is also included. This help section has the basic information on how to use H-Flasher.

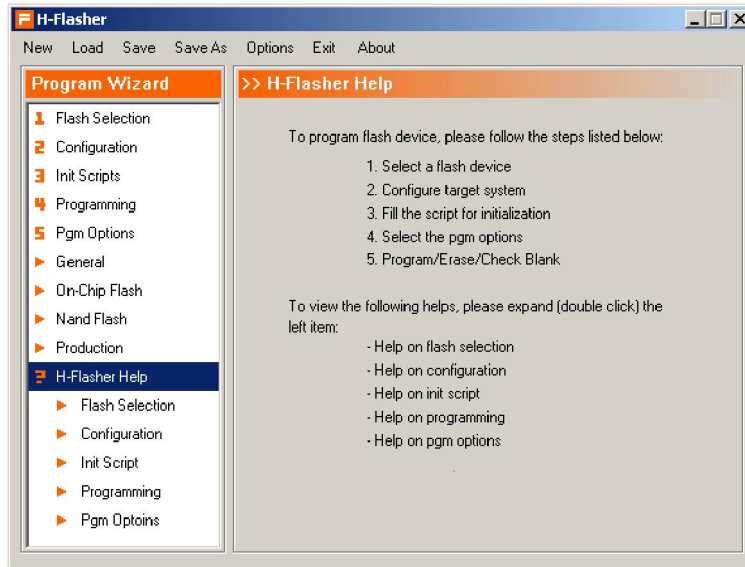


Fig 2-32 Program Wizard - Help

---

## Chapter 3 H-JTAG USB Emulator


This chapter introduces the hardware interface and some hardware features of H-JTAG USB emulator and the installation of the driver.

### 3.1 Hardware Features and Interface

H-JTAG USB emulator is a high-performance ARM in circuit emulator, which supports both JTAG and SWD debug interfaces, and provides 10K~15M Hz JTAG/SWD clock. The maximum download speed is up to 800KB/S and the maximum upload speed is up to 550KB/S.

#### Hardware Features:

- USB 2.0 Controller+ FPGA design
- High speed USB 2.0
- Powered through USB
- 10K ~ 15MHz JTAG/SWD clock
- 20-PIN standard JTAG interface
- Standard SWD interface
- Wide target voltage 1.8 ~5.0V

 **Note:** H-JTAG USB emulator only supports high-speed USB 2.0 interface.

#### Maximum RAM Read/Write Speed:

ARM7 -	Max write speed 750KB/S Max read speed 550K/S
ARM9 -	Max write speed 750KB/S Max read speed 520K/S
ARM11 -	Max write speed 800KB/S Max read speed 550K/S
XSCALE -	Max write speed 750KB/S Max read speed 520K/S
CORTEX-M3 -	Max write speed 420KB/S Max read speed 340K/S

 **Note:** The above data is for reference only. The actual speed may be different.

#### Appearance:

The appearance of the emulator is shown in Fig 3-1. The USB interface locates on the left side and the 20-pin JTAG/SWD interface locates on the right side. There are three LED indicators on the upper side, which are used to indicate the USB power, target power and JTAG activity respectively.

- A. USB: Represent the USB power.
- B. TGT: Indicate if target is connected appropriately.
- C. ACT: Indicate JTAG/SWD activity when flashing.

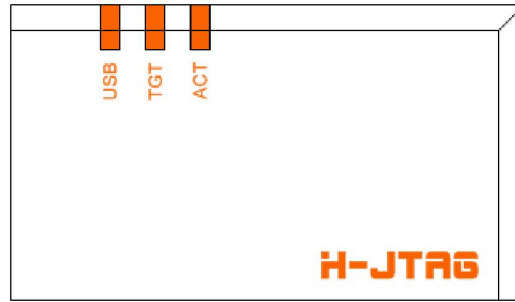


Fig 3-1 Appearance of Emulator

## Debug Interface

There are two common debug interfaces, JTAG and SWD, for ARM MCUs. The JTAG interface is used on all ARM MCUs, while the SWD interface is only seen on CORTEX-M/R based MCUs. H-JTAG USB emulator supports both JTAG and SWD. The H-JTAG emulator equipped with a standard 20-pin JTAG/SWD interface. The definitions of signals are shown in Fig 3-2.

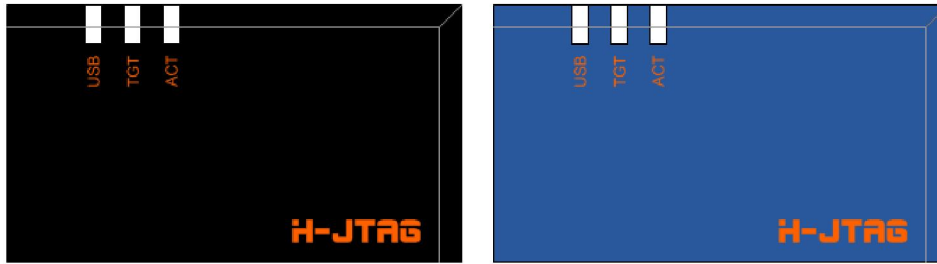
VREF	1	2	NC
nTRST	3	4	GND
TDI	5	6	GND
SWDIO/TMS	7	8	GND
SWDCLK/TCK	9	10	GND
RTCK	11	12	GND
TDO	13	14	GND
nSRST	15	16	GND
NC	17	18	GND
NC	19	20	GND

Fig 3-2 JTAG/SWD interface of H-JTAG emulator



### 3.2 Different Emulator Editions

H-JTAG USB emulator has two editions, which are the standard edition and the professional edition. These two editions can be differentiated by colors, as shown in Fig 3-3.



(a) Standard Edition

(c) Professional Edition

Fig 3-3 Different emulator editions

The differences of functionalities are listed in Table 3-1.

H-Jtag Edition		Standard	Professional
Speed	Download	750KB/S	750KB/S
	Upload	550KB/S	550KB/S
ARM CORE	ARM7	YES	YES
	ARM9	YES	YES
	CORTEX-M3	YES	YES
	XSCALE	YES	YES
	ARM11	NO	YES
	PXA3XX	NO	YES
Supported IDE		ADS1.2 RVDS KEIL/MDK IAR	
Supported OS		NT/WIN2000/WINXP/VISTA/WIN7/WIN8	

Table 3-1 Differences of functionalities

---

### 3.3 Driver Installation

After the installation of H-JTAG software, a folder named Drivers is created under the installation directory. This folder includes the drivers for different WINDOWS platforms. Next, user needs to install the driver for emulator. This section shows how to install the driver step by step.

First, connect the emulator to PC via a USB cable. The PC will show a new hardware wizard dialog soon, which looks like Fig 3-4.



Fig 3-4 New hardware wizard

In the wizard, choose “Install from a list or specific location (Advanced)”, as shown in Fig 3-5. Then, click “Next”.



Fig 3-5 Install from list

Next, a search and installation options dialog (Fig 3-6) will be shown. In this dialog, choose “Search for the best driver in these locations”. De-select Search removable media. Select “Include this location in the search”. Meanwhile, use the Browse button to locate the proper driver under the Drivers folder in the H-JTAG installation directory. User needs to ensure that the proper windows platform and architecture is selected. Then, click “Next”.

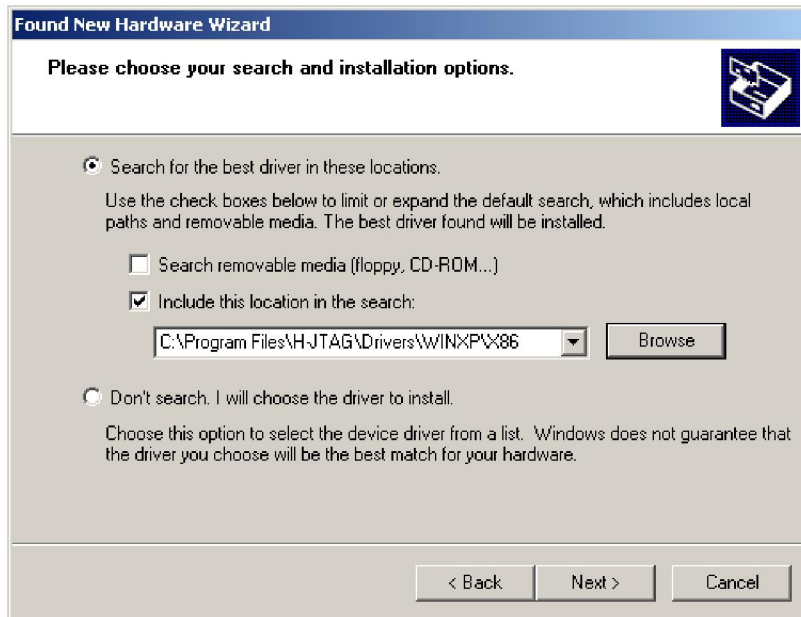


Fig 3-6 Search and installation options

Soon, a confirm dialog will be popped up after a quick search. In the following picture, click “Continue Anyway” to proceed with the installation.

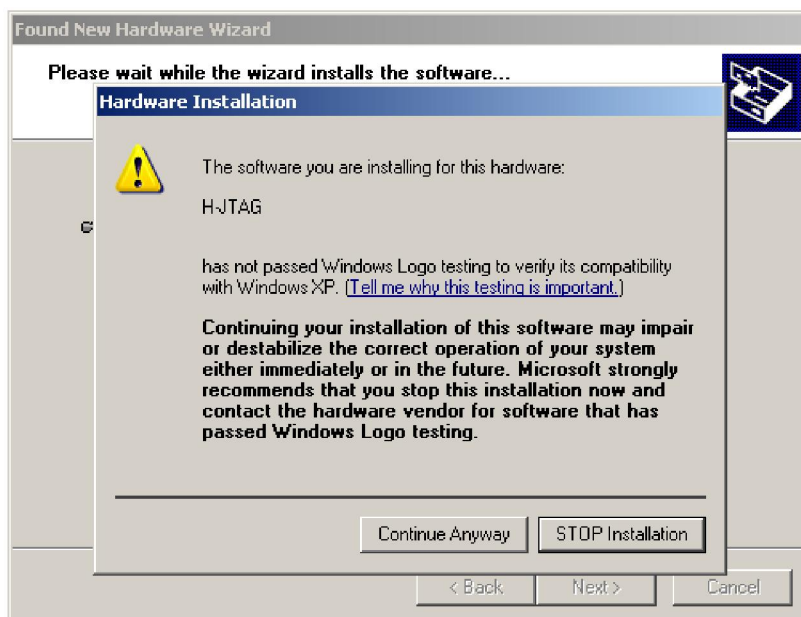


Fig 3-7 Confirm dialog

Later, a completing dialog will be seen. If the installation is completed successfully, the dialog looks like Fig 3-8.



Fig 3-8 Completing dialog

After the driver has been installed successfully, H-JTAG is listed as a device in the device manager, as shown in Fig 3-9. If you can't see "H-JTAG" in the device manager, please try to install the driver again.

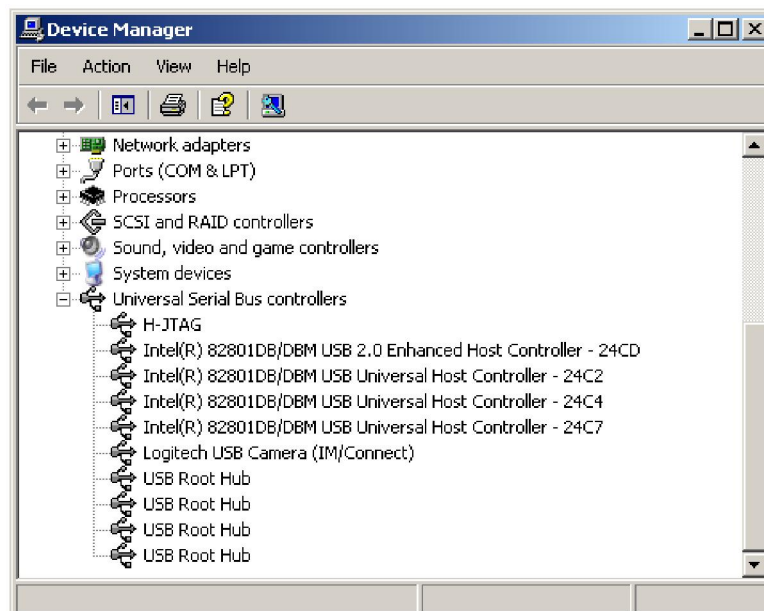


Fig 3-9 Device manager

---

## Chapter 4 Configure H-Jtag

This chapter introduces how to configure and use H-Jtag in details. Meanwhile, some simple examples are given for reference.

### 4.1 Detect Target

Before detection, please connect the JTAG emulator to the USB or parallel port and connect target to the JTAG emulator. During the start up, H-Jtag tries to detect the target automatically. After H-Jtag is started up, user can click the detect button to detect target. If the target is detected successfully, the ARM core and device ID are displayed in the middle of the main window. Otherwise, please check the configuration and hardware connection.

### 4.2 Reset Target

User can reset target via H-Jtag. For a standard JTAG interface, two independent reset signals, system reset (nSRST) and JTAG/TAP reset (nTRST), are defined. With these two reset signals, H-Jtag can perform system reset and JTAG/TAP reset.

#### **Note:**

Some LPT based JTAG emulator doesn't provide system reset signal, so H-Jtag can't perform system reset on target. The target can only be reset manually.

### 4.3 Auto Flash Download

H-Jtag supports auto flash download for debugging. With auto flash download, program can be directly downloaded or written to flash for debugging, just like debug in RAM/SDRAM. To use auto flash download, please enable the auto download option as shown in Fig 4-1. Meanwhile, please choose appropriate target flash and provide right configuration in H-Flasher. During downloading, H-Jtag will determine where the program should be downloaded according to the information extracted from the image. For those need to be written to flash, H-Jtag will call H-Flasher to complete it automatically.

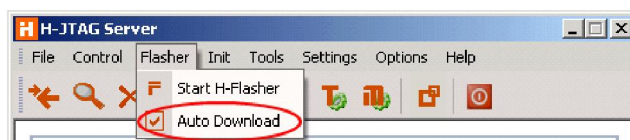


Fig 4-1 Auto Flash Download

#### **Note:**

Auto flash download can be used for both on-chip flash and external NOR flash. For chips supports complicated memory configuration, like MMU/REMAP, it is recommended to disable MMU/REMAP through init script before debugging.

#### **Note:**

H-Flasher Lite does not support Auto Flash Download. To use Auto Flash Download, please run H-Flasher.

---

## 4.4 Initialization Script

For most systems, initialization need to be performed after powered up. The initialization of memory system is one of these. Most of the time, flash and on-chip SRAM can be accessed directly after powered up, but this is not the case for external SDRAM. External SDRAM needs to be initialized before it can be accessed correctly. For beginners, one of the common problems is that the program can not be downloaded into external SDRAM correctly. The reason is that the external SDRAM is not initialized properly. There are two methods to initialize the target system. The first one is to write an initialization program into flash. This program will be executed right after the target is powered up. So, the target is initialized automatically after powered up and is ready for use. The second one is to use initialization script. For the convenience of user, H-Jtag defines some script commands and provides auto initialization. To use auto init, user needs to input or load proper initialization scripts and enables auto init. If auto init is enabled, H-Jtag will execute the provided initialization scripts whenever a connection is opened by debugger. The script editor is shown in following figure. For the details on the initialization script, please refer to Chapter 6.

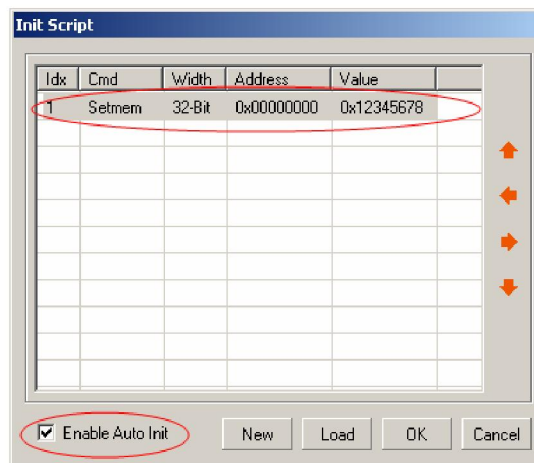


Fig 4-2 Script Editor

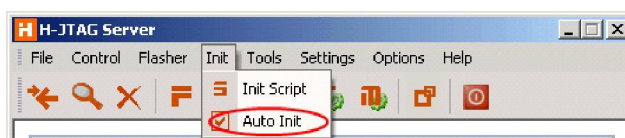


Fig 4-3 Auto Init

 **Note:**

If auto init is enabled in H-Jtag server, initialization scripts must be provided. Otherwise, H-Jtag would notify user with error message “Can’t open specified init script.”

 **Note:**

For the detailed definition of script commands, please refer to Chapter 6.

---

## 4.5 USB/LPT Interface Selection

H-Jtag supports both USB based H-JTAG high-speed emulator and LPT based JTAG emulator. User should select the right hardware interface accordingly.

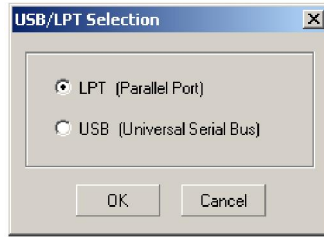


Fig 4-4 USB/LPT Interface Selection

## 4.6 JTAG Configuration

This section introduces the definition of JTAG interface, connection to USB/LPT port and the configuration of JTAG.

### 4.6.1 JTAG Signals

JTAG is a test standard proposed by IEEE. For ARM debugging, JTAG is used as the interface. The ARM JTAG interface defines 7 signals, TMS, TCK, TDI, TDO, RTCK, nSRST and nTRST. For the debugging of ARM7 and ARM9, TMS, TCK, TDI and TDO are indispensable, while RTCK, nSRST and nTRST are optional.

 **Note:**

For XScale, separate nSRST and nTRST signals are indispensable. Otherwise, debugging can not be proceeded.

### 4.6.2 JTAG Connection

The typical JTAG connection looks like Fig 4-5. The JTAG emulator connects to both the USB/LPT and the target. H-JTAG generates JTAG signals via the JTAG emulator to control the target. The JTAG interface between JTAG emulator and ARM target normally adopts the standard 20-pin interface. The H-Jtag server and the JTAG emulator can communicate to each other via USB or LPT. When LPT is used, user needs to provide the accurate JTAG configuration and tell H-JTAG exactly how the JTAG emulator is connected to LPT. For the details on how to configure JTAG, please refer to the following sections.

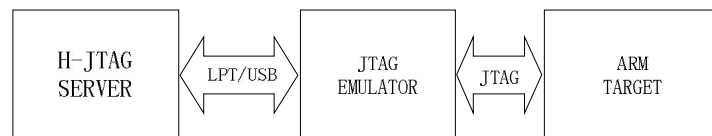


Fig 4-5 JTAG Connection

---

### 4.6.3 USB JTAG Setting

H-Jtag USB emulator supports two different modes, JTAG mode and SWD mode. The SWD mode is a new debug interface proposed for the debugging of CORTEX-M/R cores. H-Jtag USB emulator also supports different TCK speeds (25K – 15M Hz). In the USB JTAG setting dialog (Fig 4-6), user can specify the TCK speed manually, let H-JTAG select the TCK speed automatically, or use adaptive TCK (RTCK). When AUTO TCK is chosen, H-Jtag will determine the appropriate TCK speed automatically through testing. When adaptive TCK is used, the real TCK speed is determined by the RTCK signal from MCU.

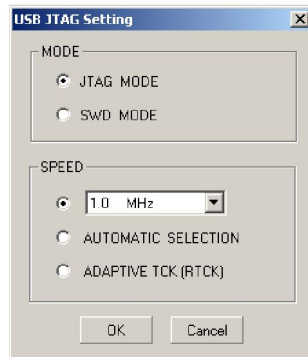


Fig 4-6 USB JTAG Setting

#### **Note-1:**

The TCK speed has direct affect on the debugging performance. Different target system has different highest supported TCK speed. The highest supported TCK speed also depends on the system clock configuration of target system. Appropriate TCK speed should be designated. Over-high TCK speed leads to unpredictable debugging behavior. AUTO TCK may not work under some situation. In this case, please designate a lower TCK speed manually.

#### **Note-2:**

The adaptive TCK is only for JTAG mode and it is not useable for SWD mode. To use adaptive TCK, the target under debug needs to support RTCK and the RTCK signal also need to be connected to the JTAG interface properly.

### 4.6.4 LPT JTAG Setting

The LPT based JTAG controller does not have a fixed schematic, even for WIGGLER and SDT-JTAG. Some JTAG emulator comes with nSRST, while others don't have. Some JTAG emulator provides separate nSRST and nTRST, while others connected them together. To support different LPT based JTAG emulators, H-JTAG provides a flexible configuration interface. What user needs to do is tell H-Jtag exactly how the JTAG emulator is connected to LPT.

The LPT provides 8 data bits, D0-D7, as output and several status bits as input. The data bits can be used as JTAG output signals, TMS, TCK, TDI, nSRST and nTRST. Any one of the status bits can be used as input to sample TDO. The JTAG configuration is to specify how the JTAG signals are connected with the data bits and status bits. On some JTAG emulator, the nSRST and nTRST are inverted. These also need to be specified in the JTAG configuration.



Next, let's look at an illustrative example. The schematic of the JTAG emulator used in the example is shown in Fig 4-7. The connection between LPT and JTAG can be obtained from the schematic and is listed in the following table. Please note that the nTRST signal is inverted and no nSRST signal is provided.

TMS	➔	LPT D1 (PIN3)
TCK	➔	LPT D2 (PIN4)
TDI	➔	LPT D3 (PIN5)
TDO	➔	LPT BUSY (PIN11)
nTRST	➔	LPT D0 (PIN2) <b>INVERTED</b>
nSRST	✗	NOT AVAILABLE

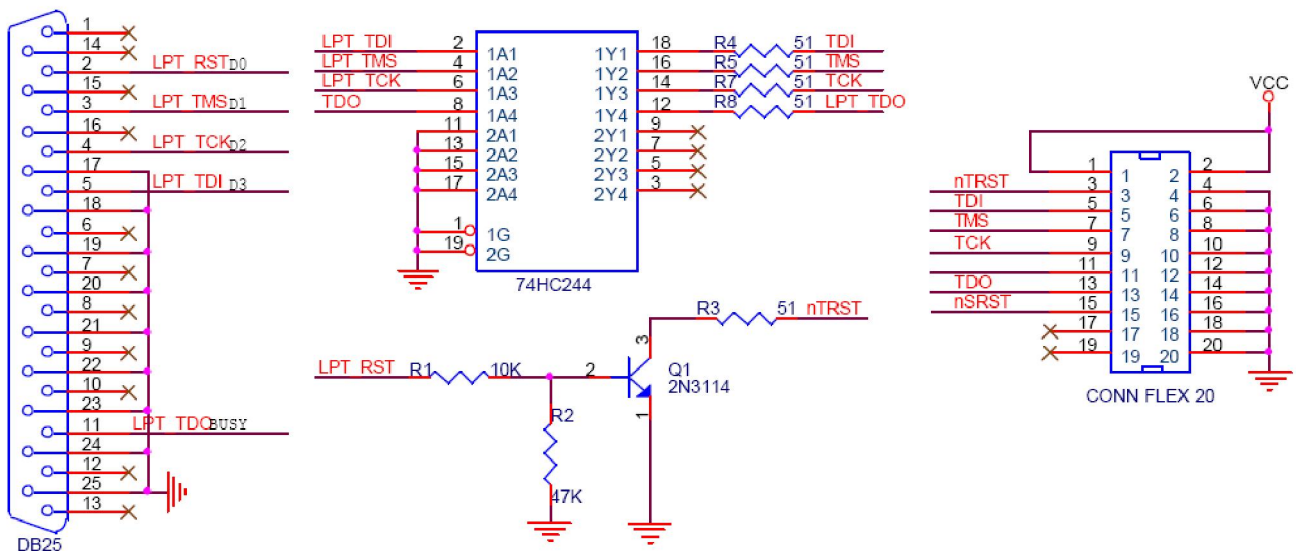


Fig 4-7 Example Schematic of JTAG Emulator

Based on the schematic given in Fig 4-7 and the above analysis, any of the following settings can be used. Both the settings given in Fig 4-8 tell exactly how LPT is connected to the JTAG interface via the JTAG emulator. The given example is only for reference. In practice, please configure according to the schematic of user's own JTAG emulator.

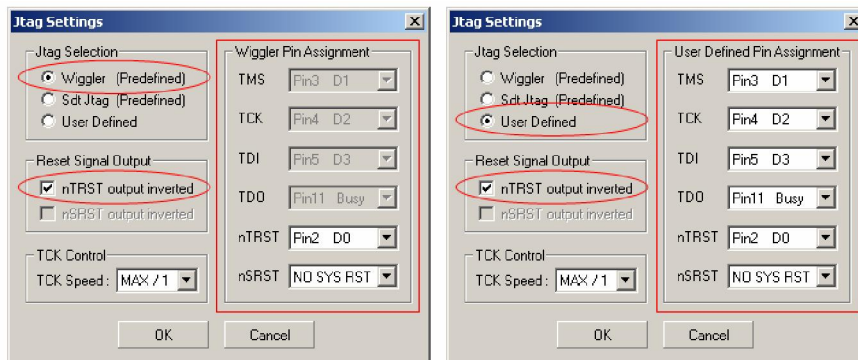


Fig 4-8 Example JTAG Settings

In the dialog of LPT JTAG settings, user can also choose different TCK speed. The selectable TCK speed ranges from MAX to MAX/8. Actually, the parallel port is a low speed interface. In practice, user is suggested to use MAX/1 as the default speed to achieve maximum performance.

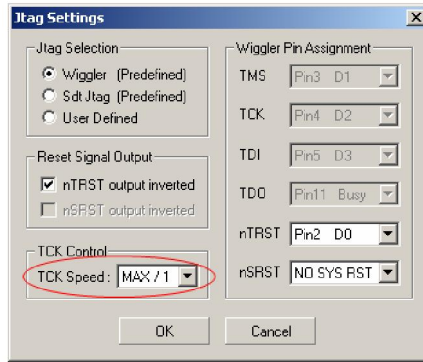


Fig 4-9 Selection of LPT TCK Speed

## 4.7 LPT Port Setting

For most PCs, the default LPT address is 0x378, but there are some exceptions. In H-Jtag, different port addresses can be specified. The dialog for port setting is shown in Fig 4-10. In the dialog, there also has a test button, which can be used for some simple port read/write test.

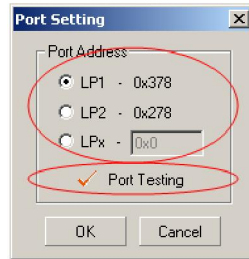


Fig 4-10 LPT Port Setting

## 4.8 Target Setting

H-Jtag reads device ID via JTAG and determines what the ARM core is. H-Jtag can recognize most of the common chips. For target that can't be recognized, user can designate the ARM core in target settings. The dialog for target settings is shown in Fig 4-11.

Most of the ARM chips support both little endianness and big endianness. With different endianness, the storage of data and instructions are totally different. If the endianness is not specified correctly, the debug definitely goes wrong. User can specify the endianness in target settings (see below).

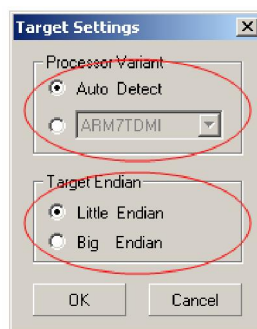


Fig 4-11 Target Setting

---

## 4.9 Target Manager

In above section, it is mentioned that H-Jtag determines the ARM core based on the device ID. For chips can't be recognized, user can specify the ARM core manually. User can also add the new device ID into the chip list. After the chip list is updated, H-Jtag can detect and recognize the new chip automatically. All this can be done in the target manager. To add a new device ID, user needs to input the ID and specify the according ARM core. In target manager, user also can delete existing device IDs. The target manager is shown in Fig 4-12.

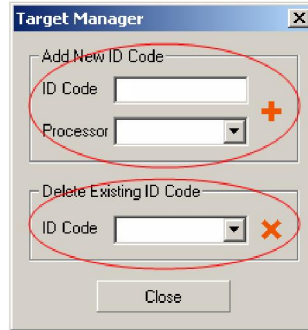


Fig 4-12 Target Manager

### Tip:

According to the IEEE-1149 standard, device ID is 32-bit and the lowest bit should be 1. User can tell whether an ID is valid or not based on this. If you have any chip that H-Jtag can't recognize, please email us the device ID and the ARM core. We will update the chip list in next version.

## 4.10 TAP Configuration

For most ARM chips, the JTAG scan chains are separated. For these chips, the default TAP configuration should be adopted, as shown in Fig 4-13. The figure indicates that no other scan chain is concatenated before and after that of the ARM core.

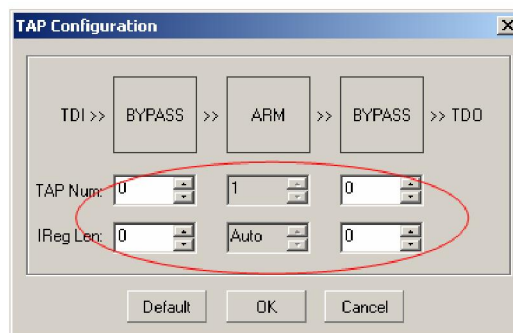


Fig 4-13 Default TAP Configuration

Some chip integrates other scan chains besides that for ARM core. For this kind of chips, user needs to configure TAP accordingly. STR91xF is a chip from ST. This chip integrates several scan chains inside, as shown in Fig 4-14. STR91xF has 3 TAPs, TAP#1, TAP#2 and TAP#3. Among all these TAPs, only TAP#2 is for ARM debugging. TAP#1 and TAP#3 are concatenated before and after TAP#2. The IR length of TAP#1 and TAP#3 is 5-Bit and 8-Bit respectively. For this chip, the TAP configuration should look like Fig 4-15. The TAP

configuration shown in Fig 4-15 tells that there is one scan chain before that of the ARM core and its IR length is 5-Bit. In addition, there is another scan chain after that of the ARM core and its IR length is 8-Bit. Based on the TAP configuration, H-Jtag knows how to access the scan chain of the ARM core for debugging.

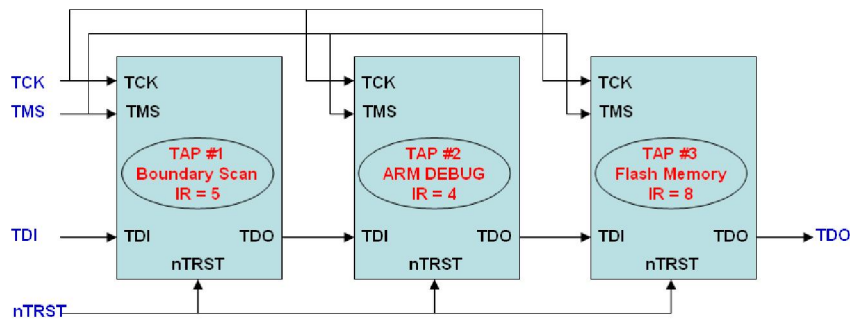


Fig 4-14 STR91xF Scan Chains

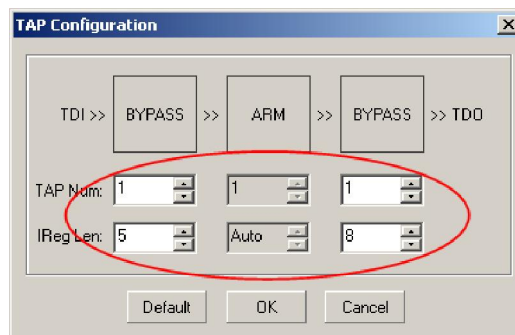


Fig 4-15 TAP Configuration (STR91xF)

## 4.11 H-Jtag Options

H-Jtag provides some common options. User can make the selection using the main options window or the option menu. The main window of options is shown in Fig 4-16.

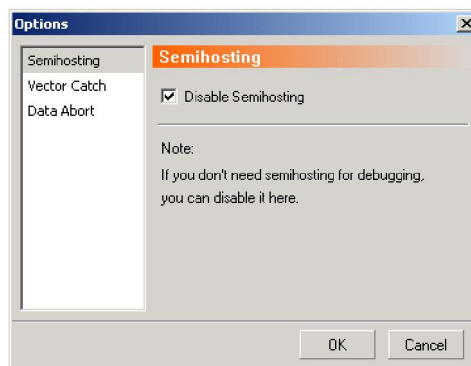


Fig 4-16 Main Window of Options

---

## ■ CONNECTION MODE

Different connection modes are defined in H-Jtag. From the connection mode list, user can select different modes as required. User can also check the description of each connection mode in the configuration dialog.

## ■ VECTOR CATCH CONFIG

H-Jtag server manages the vector catch configuration. In the configuration dialog (Fig 4-17), user can decide which exception(s)/interrupt(s) to be caught during the debugging. When all the flags are cleared, vector catch is disabled automatically. In another case, where the global disable vector catch option is enabled, the configuration on the following dialog is ignored automatically.

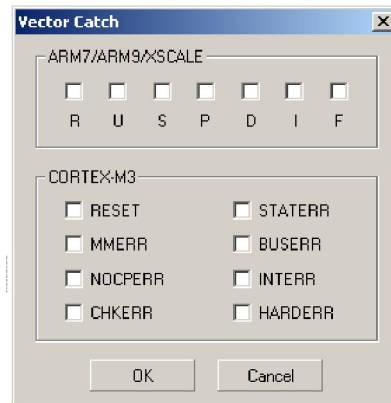


Fig 4-17 Vector Catch Configuration

## ■ DISABLE SEMIHOSTING

Semihosting is a mechanism for debugging, which can be used for the communication between host and target. Semihosting needs the support from both the emulator and the program running inside the target. In addition, semihosting can only be used for debugging and can't be use in real product. Semihosting also consumes breakpoint resources. So, user is suggested to disable semihosting.

## ■ DISABLE VECTOR CATCH

Vector catch is used to capture exceptions. When vector catch is enabled in both H-Jtag and debugger, exceptions are notified when they occur. Please note that vector catch also consumes breakpoint resources. So, user is suggested to disable vector catch.

## ■ REPORT DATA ABORT

During debugging, even when the processor is halted, debugger needs to access the memory of target. Data abort may occur when undefined space or access protected area is accessed. If report data abort is enabled, H-Jtag would notify user when data abort occurs. Otherwise, H-Jtag would only handle the data abort internally without notifying user.

## ■ DISABLE IAR BKPT @ 0x8

Under IAR, a breakpoint is set at 0x8 during debugging. This breakpoint is invisible in the breakpoint list. When debugging inside flash, user can disable this invisible breakpoint to free a breakpoint unit.

---

## 4.12 H-jtag Tools

H-Jtag provides some tool under the Tools menu. In the future, we are going to add more tools to this menu as required by user.

## 4.13 Check for Updates

User can check for updates by using the check menu. If there is any newer version available, H-Jtag would notify user. User also can visit the homepage ([www.hjtag.com](http://www.hjtag.com)) of H-Jtag for more information.

---

## Chapter 5 Configure H-Flasher

H-Flasher is a powerful flash programming tool, which supports the programming of On-Chip flash, Nor Flash, Spi Flash and Nand Flash. This chapter introduces how to configure and use H-Flasher in details. At the end of this chapter, two examples are given for reference.

### 5.1 H-Flasher Configuration File

H-Flasher supports the save/load of HFC (H-Flasher Configuration File). HFC contains all the configuration information, includes flash type, flash width, flash starting address, RAM starting address and initialization scripts. User can save current configuration as a HFC for later use or load existing HFC into H-Flasher.

Under the installation directory of H-JTAG, there is a folder called HFC EXAMPLES. In this folder, user can find lots of HFC configuration files, which can be used as reference.

### 5.2 H-Flasher Production Mode

H-Flasher supports production mode. In production mode, the programming process is controlled automatically through the detection of connecting/disconnecting of targets, which can improve the efficiency significantly. To enable production mode, please tick the option as shown in following figure.

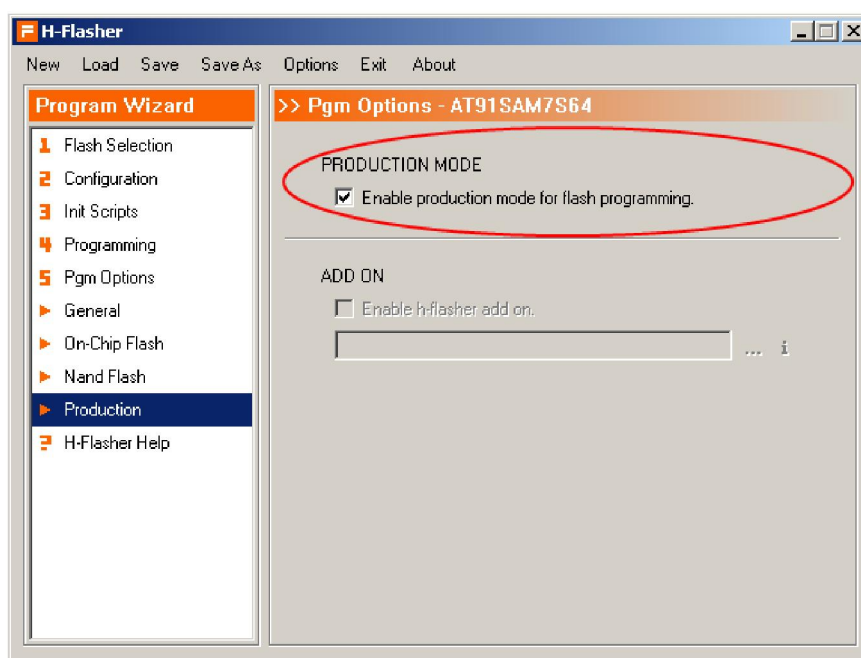


Fig 5-1 Enable Production Mode

---

## 5.3 H-Flasher Workflow

The workflow of H-Flasher is very simple. As shown in Fig 5-2, the workflow includes four steps: execute init script, download flash driver, check flash ID and operate on flash. These four steps are executed in sequence. If any step goes wrong, the operation is interrupted immediately.

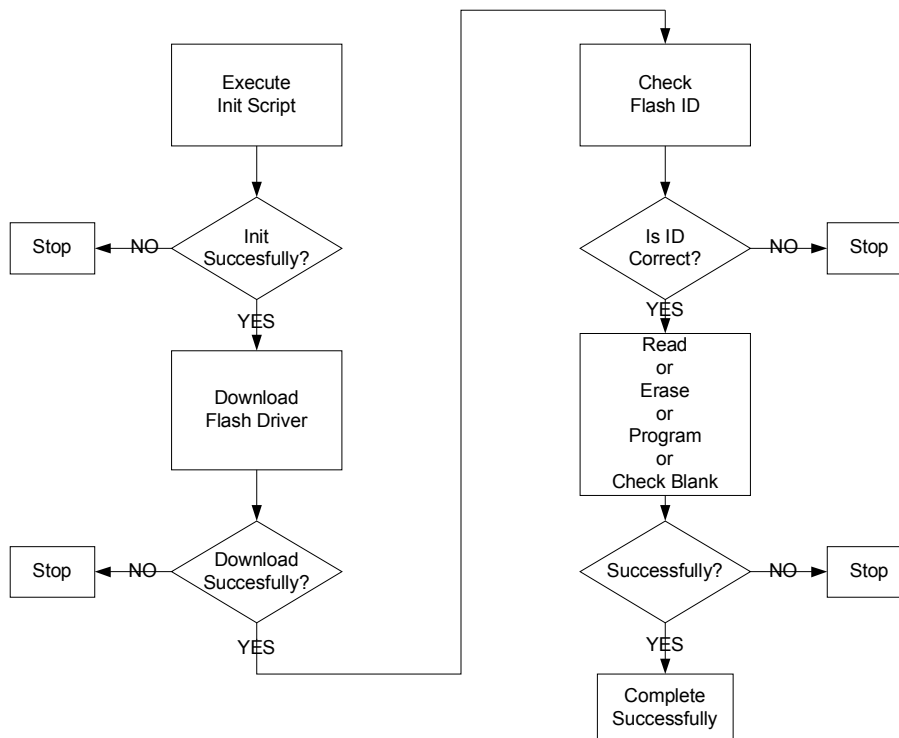


Fig 5-2 Workflow of H-Flasher

### 5.3.1 Execute Init Script

If init scripts are specified, H-Flasher first executes them to initialize the target. If no init script is specified, or no script is needed, H-Flasher skips this step. If something goes wrong during the initialization, H-Flasher stops immediately and notifies user.

### 5.3.2 Download Flash Driver

After successful completion of initialization, H-Flasher searches appropriate flash driver according to the selected flash and specified bit width. Then, H-Flasher downloads the flash driver into SRAM/SDRAM based on the designated RAM start address. If the download is successful, go to next step. Otherwise, error out and notify user with message: Can't download driver to specified address.

### 5.3.3 Check Flash ID

After the download of driver, H-Flasher checks the flash ID. The purpose is to ensure that the selected flash is the right one and check whether the flash can be accessed by the specified address.



---

### 5.3.4 Read/ Erase/ Program/Verify/Check Blank

After the previous three steps have been executed successfully, H-Flasher can perform any operations on target flash as required. The supported operations include read, erase, program, verify and check whether the flash is blank.

## 5.4 H-Flasher Program Wizard

H-Flasher comes with a program wizard to easy the configuration. User can follow the wizard to complete the configuration.

### 5.4.1 Flash Selection

In this first step, user can select the target flash from the list. User needs to check the information and ensure that the selection is correct. The correct selection is very important, because different flash chips define different command sets.

### 5.4.2 Configuration

In this step, user needs to provide necessary information about the target system, which includes flash bit width, flash start address, RAM start address, XTAL and TCK.

#### **Bit-Width and Chip Number**

Some external flash chips can operate at different bit-widths, for example 8-Bit, 16-Bit and 32-Bit. Normally, drivers for different bit-widths are different. Hence, user needs to specify the bit-width for this kind of chips. For chips support only one bit-width, the default one is used and user doesn't need to specify it. In some design, multiple chips are used. In this case, user also needs to specify the right chip number.

#### **Flash Start Address**

To operate on the target flash, H-Flasher needs to know the flash start address. So, user needs to specify the start address of the flash. To H-Flasher, the valid address space is from (Flash Start Address) to (Flash Start Address + Size - 1). Any address out of this range is treated as illegal. Generally, flash can be accessed by address 0x0 without initialization. But, some system supports remap and can map flash to different addresses. For this case, user needs to ensure that the specified flash start address is consistent to the provided init scripts. In a word, please provide the init scripts when necessary and make sure that H-Flasher can access flash on the specified address.

#### **RAM Start Address**

User needs to designate a RAM space, which should be  $\geq$  4K Bytes, because H-Flasher requires a 4K Bytes RAM space for driver use. The valid address range is from (RAM Start Address) to (RAM Start Address + 4K - 1). H-Flasher downloads the flash driver into this area. The flash driver can be downloaded into both SRAM and SDRAM. If the target system has on-chip SRAM, it is suggested to use on-chip SRAM instead of external SDRAM. The reason is that the access of SRAM is much faster than that of SDRAM. Meanwhile, please provide necessary init scripts for the initialization of memory system and ensure that the designated

---

RAM space is accessible.

#### **XTAL**

For some chips, H-Flasher needs to know the frequency of the external crystal oscillator. H-Flasher uses this for the configuration of system clock. When the flash chip is specified, the input for XTAL may be disabled or enabled accordingly. When it is enabled, please specify the XTAL. Otherwise, ignore it.

#### **INIT TCK & PGM TCK**

INIT TCK specifies the TCK speed used during the initialization stage and the PGM TCK specifies the TCK speed used during the actual programming stage. Normally, the target system can support higher TCK speed after appropriate initialization. Hence, user can specify a lower TCK speed for successful initialization and a higher TCK speed for better programming speed. Please note that INIT TCK and PGM TCK are active only when the following conditions are satisfied.

- ✧ USB based H-JTAG emulator is in use;
- ✧ Both INIT TCK and PGM TCK are set;
- ✧  $(\text{INIT TCK}) < (\text{TCK SET IN H-JTAG SERVER}) < (\text{PGM TCK})$ ;

### **5.4.3 Init Script**

In this step, user needs to provide the init scripts for the initialization of target system. User can edit scripts in the editor comes with H-Flasher. For more information on the definition of scripts, please refer to Chapter 6.

For on-chip flash, no init script is needed, because the driver already includes it. For external flash, init scripts are necessary. The purpose of init scripts is to configure the system clock and memory system. The later one is more important because correct initialization of memory system is the prerequisite. Otherwise, H-Flasher can't access flash and RAM/SDRAM.

If H-Flasher can't download driver to RAM, it notifies user with error message: Can't download driver to specified address. Most of the time, the cause is no init script is provided or the init script is not correct. To provide right scripts, user needs to have good understanding on the target system. Hence, user is suggested to read the datasheet carefully, especially the configuration of memory and clock.

#### **Tip:**

When USB based H-JTAG emulator is used, user is suggested to configure the system clock through init scripts to achieve better performance.

### **5.4.4 Programming**

In this step, user can perform operations on flash. The supported operations include check flash and target information, program flash, verify flash, erase flash and check whether the flash is blank.

#### **Reset**

The reset operation can be used to perform system reset on target.

---

## **Check**

The check operation reads the flash ID and other basic target information. User can use this operation to test whether the configuration is correct or not.

## **UnProtect**

This operation can be used to un-protect a protected flash chip. When the selected flash is not supported by this operation, the button will be disabled automatically. Currently, the STM32F series are supported.

## **Program**

For different flash types, H-Flasher provides different types of programming.

### ■ **On-Chip Flash / Nor Flash / Spi Flash**

#### **A - Auto Flash Download**

For Auto Flash Download, no source file and destination address are needed to be specified. All the information is from H-Jtag server.

#### **B - Intel HEX Format**

HEX file includes both the data/program and address information. Hence, user only needs to specify the source file when Intel HEX format is selected. H-Flasher extracts the address from HEX file automatically and uses it as the destination address.

#### **C - Plain Binary Format**

Plain binary file includes only the program/data. For plain binary format, user needs to specify both the source file and the destination address

### ■ **Nand Flash**

#### **A- Plain Binary Format (Main Only)**

When this type is selected, it means that the specified binary file only contains data for the main area of each page.

#### **B- Plain Binary Format (Main + Spare/Oob)**

When this type is selected, it means that the specified binary file contains data for both the main area and spare/oob area of each page. As a result, the length of the file should align to the size of (main area + spare/oob area).

## **Verify**

This operation can be used to verify the programming by reading the content of target flash and comparing it with the source file.

## **Erase & Check Blank**

The erase and check blank operations can be used to erase the flash or check whether the flash is blank. For both operations, user can specify the range using the list boxes.

## **Read/Dump**

The read operation is provided to obtain the content of memory on specified address. To read the content of memory, please specify both the start address and size in terms of bytes for Nor Flash and On-Chip Flash, or specify the starting sector, starting page, ending sector and ending page for Nand Flash. Besides READ

---

operation, DUMP operation is also provided for Nand Flash. For Nand Flash, the read operation only read data from the main area of each page, while the dump operation read data from both the main area and spare/oob area of each page.

### 5.4.5 Program Options

H-Flasher provides some useful options, for example, reset target after programming, additional verification and encryptions. User can choose these options as required.

 **RESET**

When enabled, H-Flasher will reset the target when the programming is done.

 **VERIFICATION**

When enabled, H-Flasher will read the data from flash and compare them with the source file for the second time, when the programming is done.

 **SKIP ID CHECK**

When enabled, H-Flasher will skip the check of flash ID during the operations.

 **ERASE CHIP**

When enabled, H-Flasher will erase the entire flash chip instead of those covered sectors before perform the flash programming.

 **SMART MODE**

When enabled, H-Flasher will backup the content of the flash before programming and then restore them during the programming. This option can ensure the data which are not covered by the programming keep unchanged.

 **NXP LPCLPC1100/LPC1200/LPC1300/1700/LPC1800/2000**

For LPC series, the checksum of the vector table is used by the internal bootloader to determine if user's application is valid or not. By default, H-Flasher adjusts the checksum automatically before the programming. If user doesn't want to adjust the checksum, please tick the box.

 **ATMEL AT91SAM**

When enabled, H-Flasher will set the security bit to enable flash protection when the programming is done.

 **ST STM32F**

When enabled, H-Flasher will set the RDP option byte to enable flash protection when the programming is done.

---

## NAND FLASH PGM OPTIONS

To program NAND flash, user needs to provide the following information to H-Flasher.

- ✓ Programming Mode:  
Specify how to program the NAND flash, skip bad blocks or relocate bad blocks.
- ✓ Verification Mode  
Specified how to perform the verification, verify the main area only or verify both the main and spare area. Please note that this option is active only when the file type is specified as “Plain Binary Format (Main + Spare/Oob)”.
- ✓ Scan of Bad Blocks:  
Specify how to scan the bad blocks for relocation mode, erase the entire chip or only erase the used blocks, reserved area for relocation and reserved blocks for relocation.
- ✓ Reserved Area for Relocation:  
Specify the reserved blocks for relocation purpose.
- ✓ Reserved Area for User Table:  
Specify the reserved area for bad block table, relocation table or user data. During programming, H-Flasher skips this area. User can write this area at the end of the programming through function `nand_info_table()`.

H-Flasher supports two NAND flash programming modes, skip mode and relocation mode. For the skip mode, H-Flasher will skip a bad block and jump to the next good one. For the relocate mode, H-Flasher will relocate a bad block to a good block in the specified reserved area. When skip mode is selected, H-Flasher will simply skip a bad block and program the data to the next good one in sequence. When the relocate mode is selected, H-Flasher will first scan the flash to collect the bad block information by erasing the entire chip or only erasing the affected blocks. During the programming, if a bad block is encountered, H-Flasher will program the data to the first free good block in the reserved area for relocation. At the same, H-Flasher will record the relocation information. When the programming is done, H-Flasher will send the bad block table and relocation table information to the NAND flash driver. In the flash driver, a function named `nand_info_table()` will receive all the information sent by H-Flasher. User can modify this function to build the bad block table and relocation table in the reserved area for user table as required.

### Note-1:

In skip mode, H-Flasher will not send the bad block table and relocation table information to the NAND flash driver when the programming is done.

### Note-2:

Due to the characteristic of NAND flash, it is not possible to provide a common flash driver which fits all hardware platforms and software architecture. When the bad block table or relocation table are needed, user should modify the `nand_info_table()` function in the flash driver based on user's own requirements.

---

 **PRODUCTION MODE**

When this option is enabled, H-Flasher enters production mode. In production mode, the software operations are simplified to improve the efficiency. After entering production mode, H-Flasher keeps detecting target. When a target is detected, H-Flasher starts the flash programming process automatically. When the programming is done, user is notified to disconnect the target and connect the next one. So, the only thing user needs to do is to connect a target for programming and disconnect it when the programming is done, then connect the next one.

---

## 5.5 Useful Tips

 **Tip-1:**

H-Flasher and Flasher Lite are same, except that H-Flasher Lite does not support Auto Flash Download. To use Auto Flash Download, please run and configure H-Flasher instead of H-Flasher Lite.

 **Tip-2:**

During the configuration, if any edit box or list box is in gray, it means that only one option is available. User doesn't need to specify.

 **Tip-3:**

Before starting the programming, H-Flasher will erase certain part of the flash automatically. Hence, user doesn't need to erase the flash manually.

 **Tip-4:**

The erase operation is sector based. To avoid the loss of data, H-Flasher provides auto backup and restore mechanism. H-Flasher will backup some data before erase and restore them during the programming. With this mechanism, the contents of the flash are kept unchanged except those covered by current programming.

 **Tip-5:**

When user sees the error message "Destination flash address is out of range", it means that the specified destination address or the destination address extracted from HEX file is not inside the valid flash address range. Please check the destination address and ensure it is valid.

 **Tip-6:**

For some chip, the ID might be changed after some new version is released. In this case, please contact us. We will provide updated flash driver.

 **Tip-7:**

If something goes wrong during the operation, please check the configuration and ensure that correct configuration is provided. If the error still occurs when correct configuration is provided, please contact us. We will analyze where the problem is and provide updated flash driver when necessary.

## 5.6 Example 1 – AT91SAM7X256

AT91SAM7X256 is an ARM7 based chip from ATMEL. This chip comes with 256K Bytes internal flash and 64K Bytes internal SRAM. This section shows how to configure and program it with H-Flasher.

### 5.6.1 Flash Selection

In this step, select AT91SAM7X256 as the target flash, as shown in Fig 5-3.

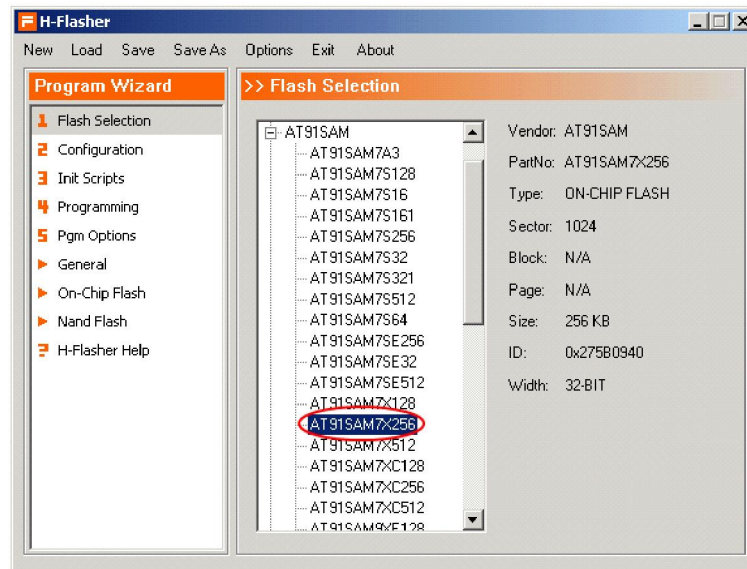


Fig 5-3 Flash Selection: AT91SAM7X256

### 5.6.2 Configuration

In this step, user needs to provide some basic memory information. For AT91SAM7X256, the bit-width, flash start address and RAM start address are all fixed. Default configuration is used. In this example, no XTAL needs to be specified.

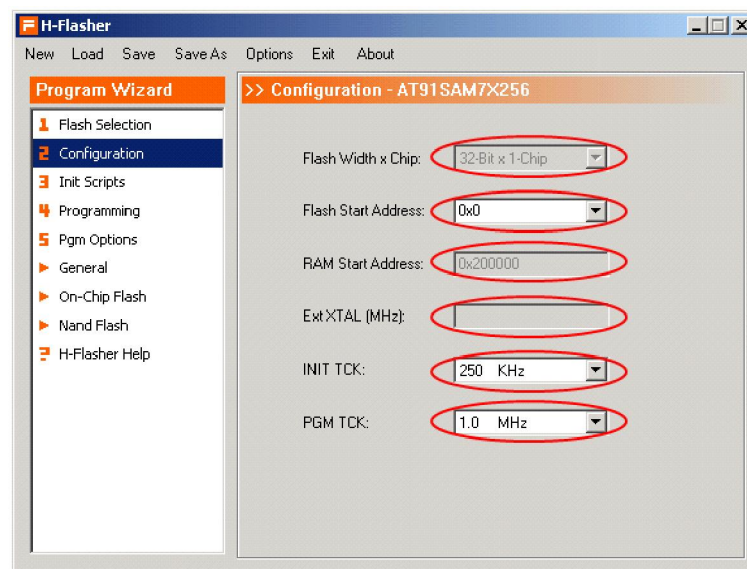


Fig 5-4 Configuration: AT91SAM7X256



### 5.6.3 Init Script

In this third step, user needs to provide the init script. For AT91SAM7X256, the driver already includes the initialization. Hence, user doesn't need to provide init script for AT91SAM7X256. In this case, the edit buttons are all disabled automatically, as shown in Fig 5-5.

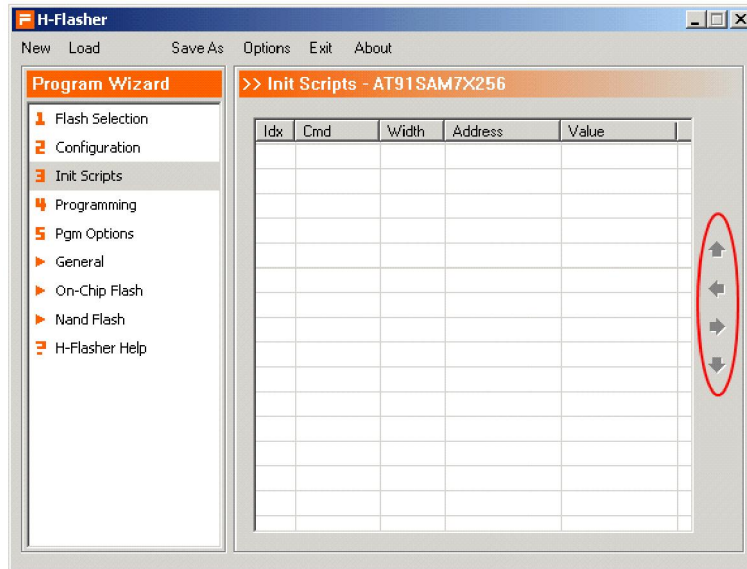


Fig 5-5 Init Script: AT91SAM7X256

### 5.6.4 Program Options

Before starting the operation, user can choose different program options as required. In this example, we choose to perform the additional verification and reset target when the programming is done.

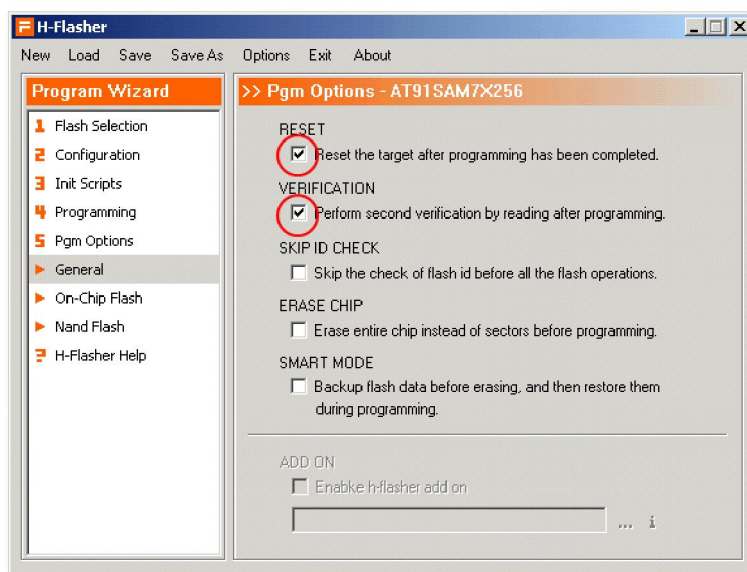


Fig 5-6 Program Options

## 5.6.5 Programming

After the configuration is completed in the first 4 steps, user can operate on flash in this step. First, let's try to check the target information by click "Check". In this example, the check result is shown in Fig 5-7. The result indicates that the configuration works well.

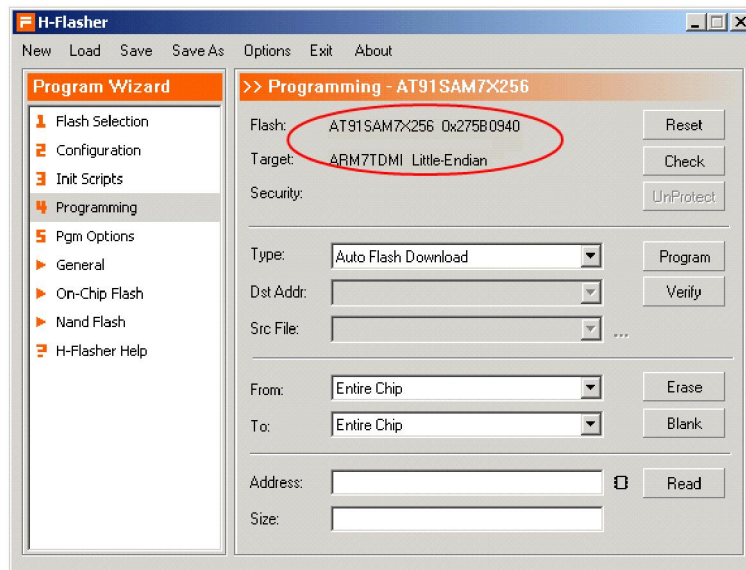


Fig 5-7 Check Result: AT91SAM7X256

Next, let's try to program a binary file. In this example, the settings are shown in Fig 5-8. Plain binary format is selected, the source file is TEST.bin located under C:\ and the destination address is 0x0 (Flash base address).

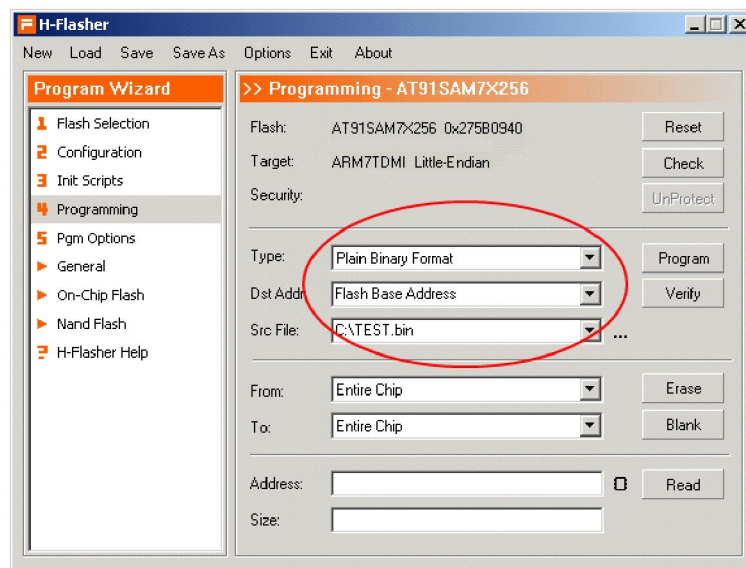


Fig 5-8 Programming Settings

Then, click "Program" to start the programming. During the process, user can see the progress, average speed and used time. When the programming is completed successfully, H-Flasher notifies user with a message, as shown in Fig 5-9.



Fig 5-9 Programming Is Completed

### 5.6.6 Save Configuration

User can save the above configuration as a HFC file. Later, user can load the HFC file in H-Flasher directly and need not to configure again.

---

## 5.7 Example 2 – LPC2210 + SST39VF1601

LPC2210 is an ARM7 based chip from NXP (Former PHILIPS Semiconduct). This chip is equipped with 16K Bytes internal SRAM, but without internal flash. This chip has four external memory banks, which can be used to expand external flash and SDRAM. In this example, it is assumed that BANK0 is used for external flash (SST39VF1601). The address range is 0x80000000 to 0x80FFFFFF. BANK1 is used for external SDRAM and the address range is 0x81000000 to 0x81FFFFFF. Next, we are going to introduce how to program H-Flasher + SST39VF1601 with H-Flasher.

### 5.7.1 Flash Selection

In the first step, as indicated, select target flash. In this example, SST39VF1601 should be selected. The selection is shown in Fig 5-10.

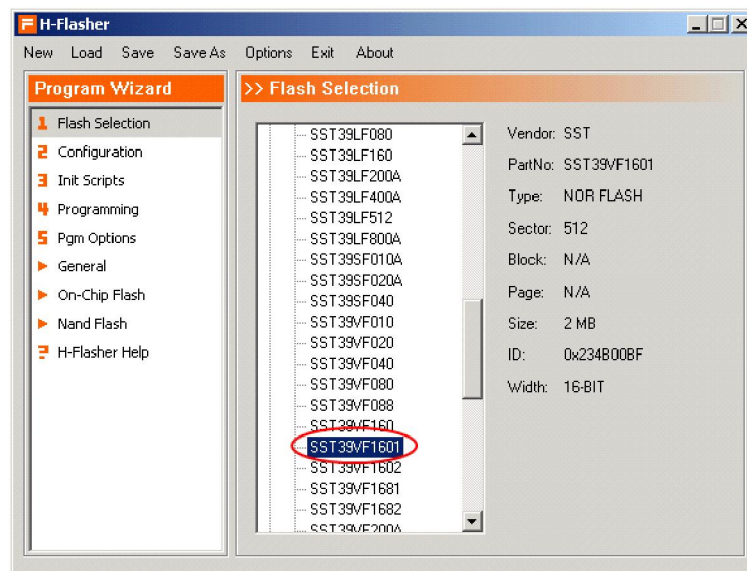


Fig 5-10 Flash Selection: SST39VF1601

### 5.7.2 Configuration

In this step, we need to provide the information on memory system. SST39VF1601 supports only 16-Bit mode. So, we can use the default bit-width. SST39VF1601 is expanded via BANK0, so that the flash start address is 0x80000000. Although LPC2210 has internal SRAM, we use external SDRAM in this example for illustration. The external SDRAM is expanded via BANK1. Hence, the RAM start address is 0x81000000. Actually, we can specify any 4K space within the SDRAM range. The XTAL also doesn't need to be specified. In this example, the final settings are shown in Fig 5-11.

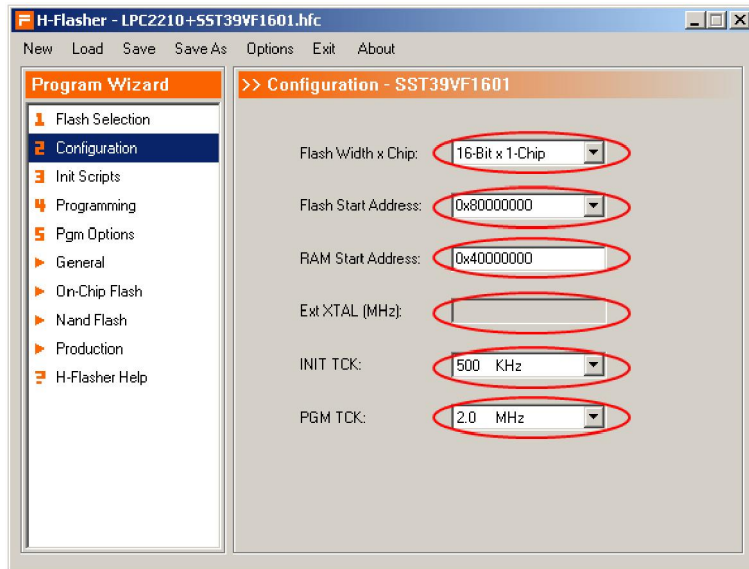


Fig 5-11 Configuration: LPC2210 + SST39VF1601

### 5.7.3 Init Script

According to the datasheet of LPC2210, we need to configure three registers: PINSEL2@0xE002C014, BCFG0@0xFFE00000, and BCFG1@0xFFE00004. PINSEL2 is a pin selection register, which is used to configure some multiplexed pins. BCFGx is bank configuration register, which is used to set the bus width and read/write wait cycles. For details, please refer the datasheet. In this example we need three scripts listed in Fig 4-12 to make sure that the flash and SDRAM are accessible.

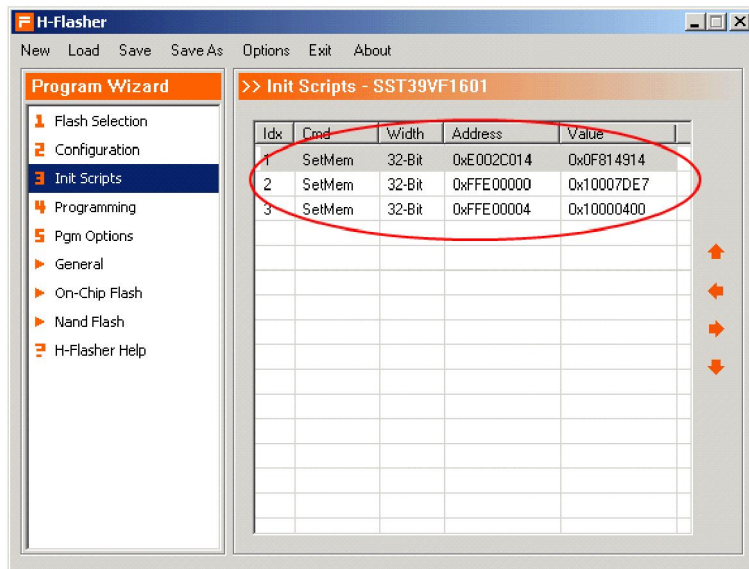


Fig 5-12 Init Script: LPC2210 + SST39VF1601

## 5.7.4 Program Options

Before starting the operation, user can choose different program options as required. In this example, we choose to perform the additional verification and reset target when the programming is done.

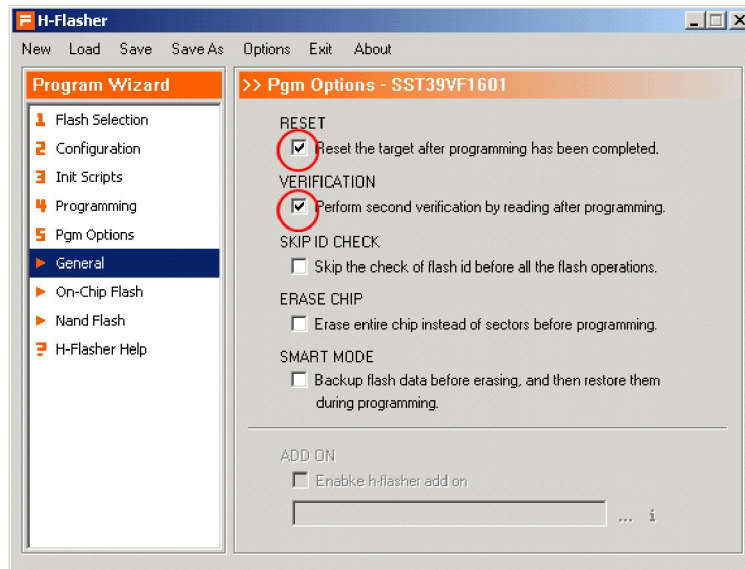


Fig 5-13 Program Options

## 5.7.5 Programming

After the configuration is done, we can operate on target flash in the fourth step of the wizard. First, let's try to check the target information by clicking "Check". In this example, the check result is shown in Fig 5-14. The result indicates that the configuration is correct.

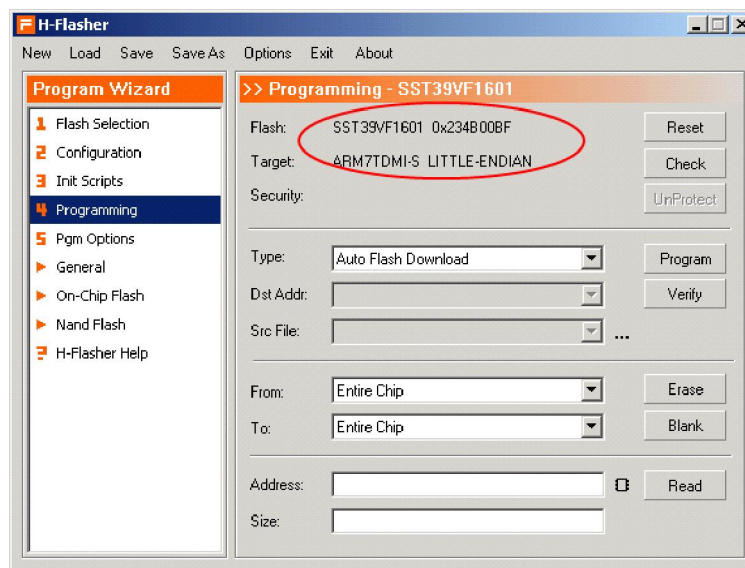


Fig 5-14 Check Result: LPC2210 + SST39VF1601

Next, let's try to program a binary file. In this example, the settings are shown in Fig 5-15. Plain binary format is selected, the source file is TEST.bin located under C:\ and the destination address is 0x80000000.

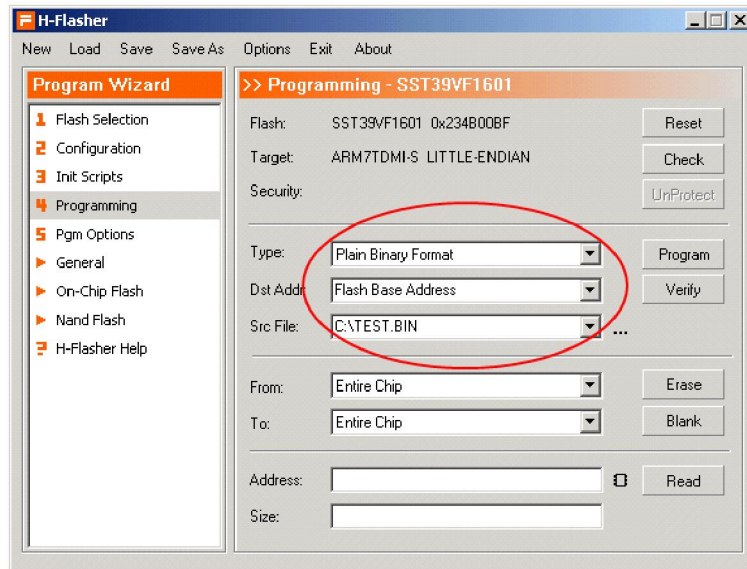


Fig 5-15 Programming Settings

Then, click “Program” to start the programming. During the process, user can see the progress, average speed and used time. When the programming is completed successfully, H-Flasher notifies user with a message, as shown in Fig 5-16.

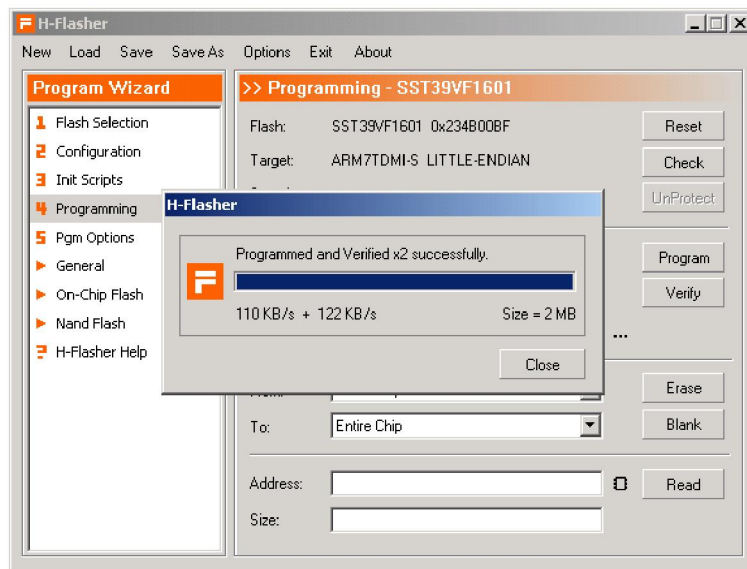


Fig 5-16 Programming Is Completed

### 5.7.6 Save Configuration

As shown in previous example, user can also save the above configuration as a HFC file for future use.

---

## Chapter 6 Initialization Script

This chapter introduces the definition of initialization scripts and how to edit scripts in H-Jtag and H-Flasher.

### 6.1 Definition of Script Commands

H-Jtag defines 6 script commands, *Setmem*, *Getmem*, *Delay*, *SysReset*, *SoftReset* and *SetPPMR*. The purpose of each script command is listed in Table 6-1. These 5 script commands can be combined to initialize different systems.

Table 6-1 Script Commands

Command	Function
<i>Setmem</i>	Set memory value
<i>Getmem</i>	Read memory value
<i>Delay</i>	Add delay
<i>SysReset</i>	Perform system reset
<i>SoftReset</i>	Perform soft reset
<i>SetPPMR</i>	Set CP15 Peripheral Port Memory Remap Register for ARM11

#### Note:

Currently, only 6 script commands are defined. But these commands can satisfy most situations. In the future, the command set may be extended to accommodate new requirements.

#### 6.1.1 Setmem

*Setmem* is the most important script command. This command can be used to set the value of memory, which includes memory mapped registers. The format of this script command is

***Setmem bit-width dest-address value***

- *Setmem* – Command;
- *Bit-Width* – Bit width of the write operation, which could be 8-bit, 16-bit or 32-bit;
- *Dest-Address* – Destination address of the write operation. Please ensure that the address is aligned;
- *Value* – The value written to the destination address;

Examples:

*Setmem 08-Bit 0x0 0x12* – Write 0x12 to 0x0, bit width is 8-bit

*Setmem 16-Bit 0x0 0x1234* – Write 0x1234 to 0x0, bit width is 16-bit

*Setmem 32-Bit 0x0 0x12345678* – Write 0x12345678 to 0x0, bit width is 32-bit



---

### 6.1.2 Setmem

This command can be used to read the value of a specific memory address. The format of this script command is

***Getmem bit-width dest-address***

- *Getmem* – Command;
- *Bit-Width* – Bit width of the read operation, which could be 8-bit, 16-bit or 32-bit;
- *Dest-Address* – Destination address of the read operation. Please ensure that the address is aligned;

Examples:

*Getmem 08-Bit 0x0* – Read the value at address 0x0, bit width is 8-bit

*Getmem 16-Bit 0x0* –Read the value at address 0x0, bit width is 16-bit

*Getmem 32-Bit 0x0* –Read the value at address 0x0, bit width is 32-bit

### 6.1.3 Delay

*Delay* command is used to add certain delay between two scripts. Some operation needs some time before it takes affect. This command can be used as null operation to wait previous script gets completed. The format of this script command is

***Delay time (millisecond)***

- *Delay* – Command;
- *Time* – Delay time in millisecond;

Examples:

*Delay 100* – *Delay 100 milliseconds;*

*Delay 5000* – *Delay 5000 milliseconds;*

### 6.1.4 SysReset

*SysReset* command is used to perform system reset. This command can only be used as the first command in user's scripts. If *SysReset* appears in other position, it will be ignored.

**Note:** *SysReset* command can only be used as the first command in user's scripts.

The format of this script command is

***SysReset (No parameter)***

- *SysReset* – Command;

---

### 6.1.5 SoftReset

*SoftReset* command is used to perform software reset. The main purpose is to reset CP15 control register. For target with CACHE and MMU, this command can be used to disable both the MMU and CACHE. For this kind of target, the program or OS inside the flash may configure the MMU and perform complicated remap operation. To re-program the target, it is suggested to disable the MMU and CACHE with *SoftReset*. By disabling the MMU and CACHE, it is easy to manage the memory map and make it same as what we expected. The format of this command is

***SoftReset*** (No parameter)

- *SoftReset* – Command;

### 6.1.6 SetPPMR

*SetPPMR* command is used to set the CP15 Peripheral Port Memory Remap Register for ARM11. The format of this command is

***SetPPMR*** *Value*

- *SetPPMR* – Command;
- *Value* – New value for PPMR register;

Example:

*SetPPMR 0x70000013* – Set PPMR register to 0x70000013;

---

## 6.2 Edit Init Script

Both H-Jtag and H-Flasher provide script editor. The editors are shown in Fig 6-1 and Fig 6-2 respectively. Actually, the editors of H-Jtag and H-Flasher are same. The introduction in this section covers both.

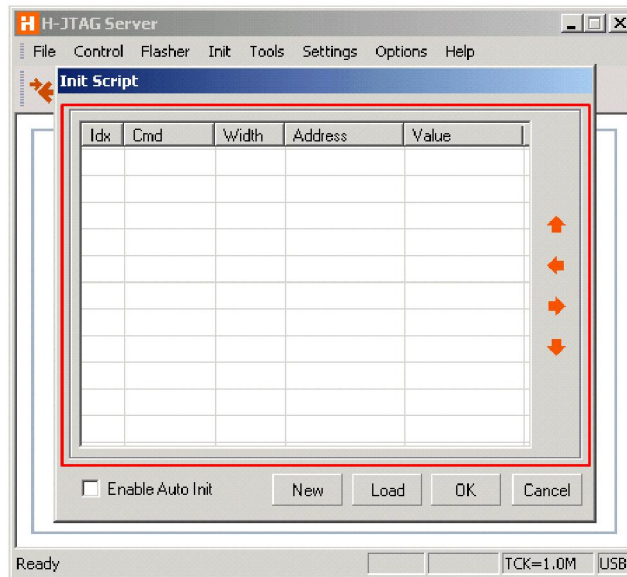


Fig 6-1 H-Jtag Script Editor

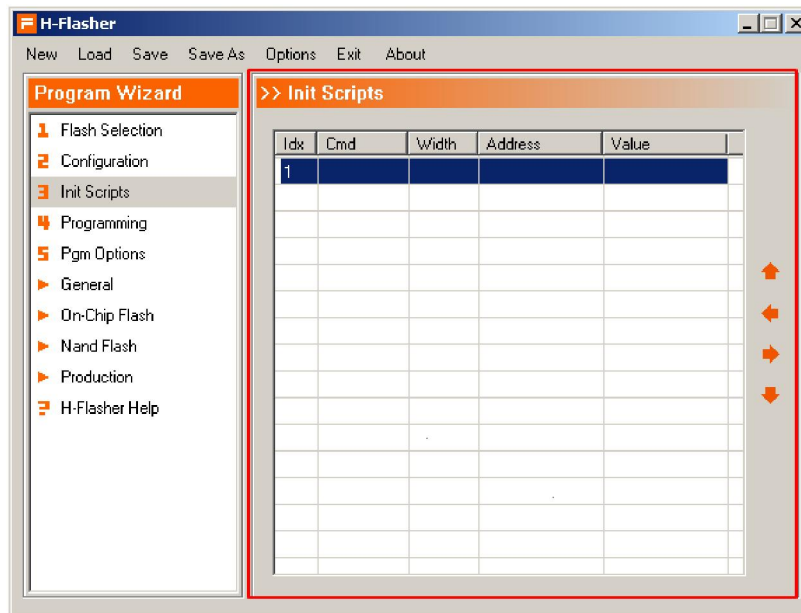










Fig 6-2 H-Flasher Script Editor

---

## 6.2.1 Edit Buttons

In the script editor, there are four buttons for editing. These four buttons are used to add, delete, move up or move down a script. The detailed definitions are listed below.

-   Move up the selected script
-   Add a new script
-   Delete the selected script
-   Move down the selected script

## 6.2.2 Edit Script

For each new script, user first needs to select a command and then specify the parameters as defined. To add a new script, click “Add” on the right side. After a new script has been added, the editor looks like Fig 6-3.

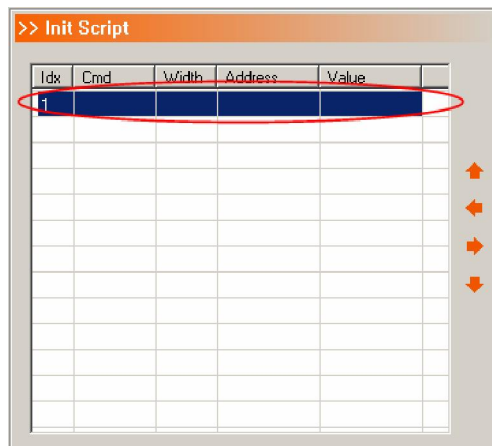


Fig 6-3 Add a New Script

A command list can be shown up by double clicking the Cmd column. The command list is shown in Fig 6-4. User can select a command from the list as needed.

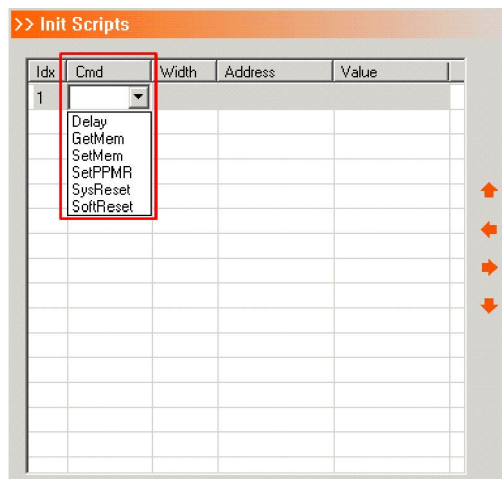


Fig 6-4 Command List

---

If *SysReset* is selected, no other parameter is needed. Fig 6-5 shows how the editor looks like after the completion of the *SysReset* script.

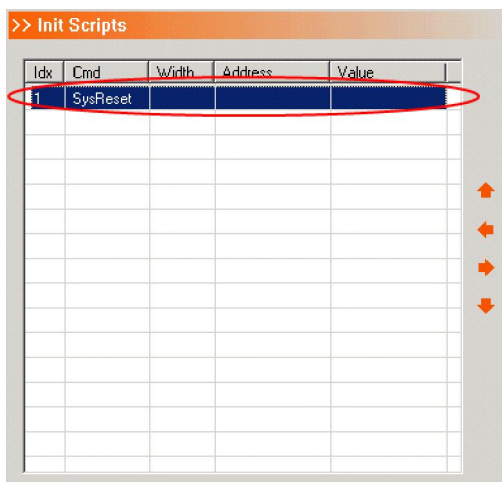


Fig 6-5 SysReset Script

If *SoftReset* is selected, no other parameter is needed. Fig 6-6 shows how the editor looks like after the completion of the *SoftReset* script.

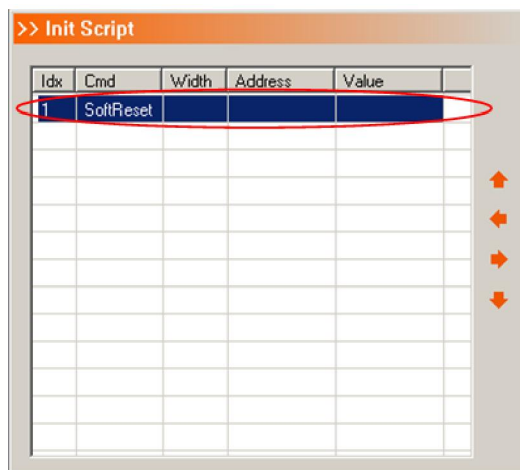
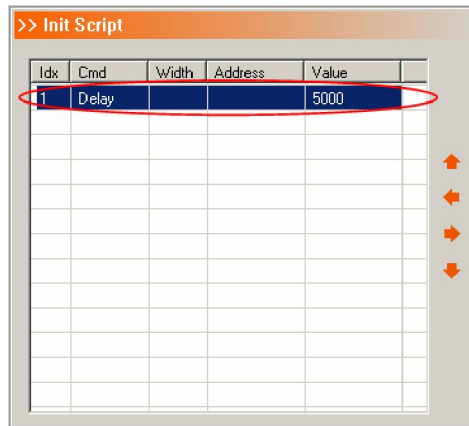


Fig 6-6 SoftReset Script

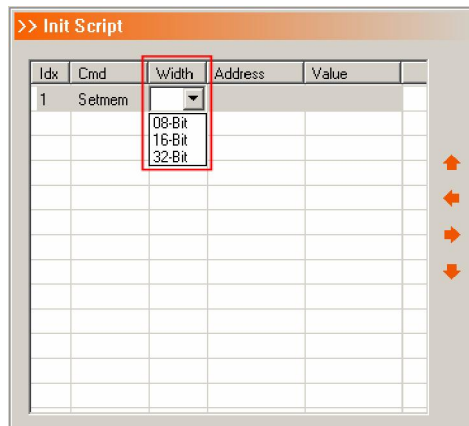
If *Delay* command is selected, the delay value also needs to be specified. Fig 6-7 shows a script which delays 5000 milliseconds.



Idx	Cmd	Width	Address	Value
1	Delay			5000

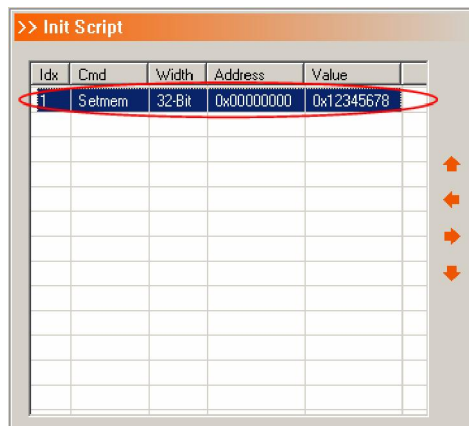
Fig 6-7 Delay Script

If *Setmem* command is selected, all the bit-width, destination address and target value need to be specified. When *Setmem* command is selected, a bit-width list will be shown up by double clicking the Width column (Fig 6-8). User can select the right bit width from the list. Next, user needs to input the destination address and value in the Address column and Value column respectively. Fig 6-9 shows script *Setmem 32-Bit 0x0 0x12345678*, which is to write 0x12345678 to 0x0.



Idx	Cmd	Width	Address	Value
1	Setmem	08-Bit 16-Bit 32-Bit		

Fig 6-8 List of Bit-width



Idx	Cmd	Width	Address	Value
1	Setmem	32-Bit	0x00000000	0x12345678

Fig 6-9 Setmen Script

If *Getmem* command is selected, both the bit-width and destination address need to be specified. When *Getmem* command is selected, a bit-width list will be shown up by double clicking the Width column (Fig 6-10). User can select the right bit width from the list. Next, user needs to input the destination address in the Address column. Fig 6-11 shows script *Getmem 8-Bit 0x10000000*, which is to read the 8-bit value at address 0x10000000.

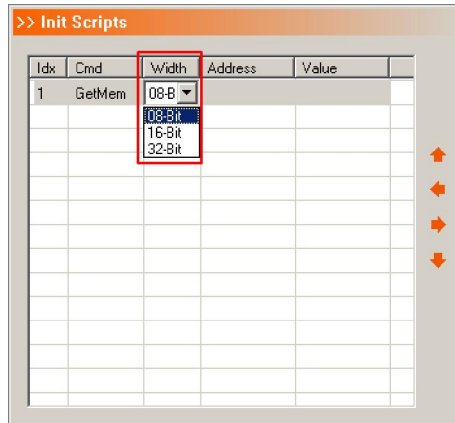


Fig 6-10 List of Bit-width

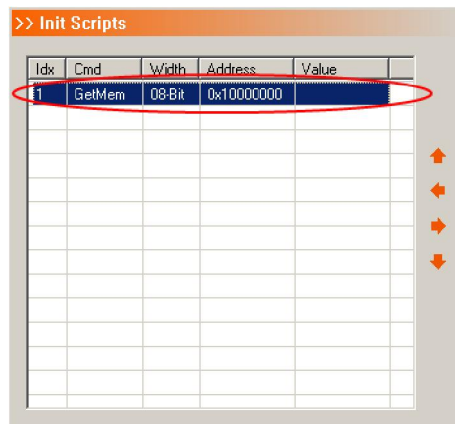
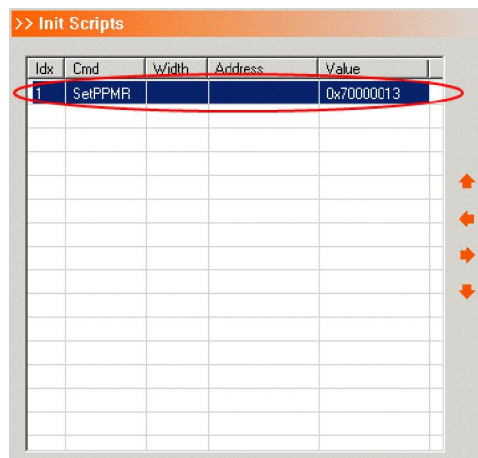


Fig 6-11 Getmen Script

If *SetPPMR* command is selected, the value for PPMR register also needs to be specified. Fig 6-12 shows a script which set PPMR register to 0x70000013.



6-12 SetPPMR Script

---

### 6.2.3 Add Script Comment

For each script command, user can add a comment. In the script editor, double click the index number before a command, a dialog will be popped up, as shown in Fig 6-13. User can review the existing comment or add a new one in the dialog.

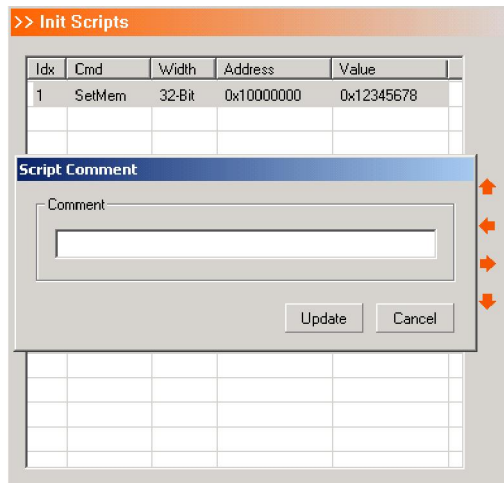


Fig 6-13 Script Comment



---

## Chapter 7 Configure Debuggers

This chapter illustrates how to configure common debugger software, which include AXD, RVDS, IAR and KEIL. For details on how to use respective debuggers, please refer to their own user manuals.

### 7.1 Configure AXD

ADS (Arm Developer Suit) is the most widely used IDE from ARM Corp. AXD is the debugger comes with ADS. This section introduces how to configure AXD and use it with H-Jtag.

First, start AXD and click Options -> Configure Target, as shown in Fig 7-1.

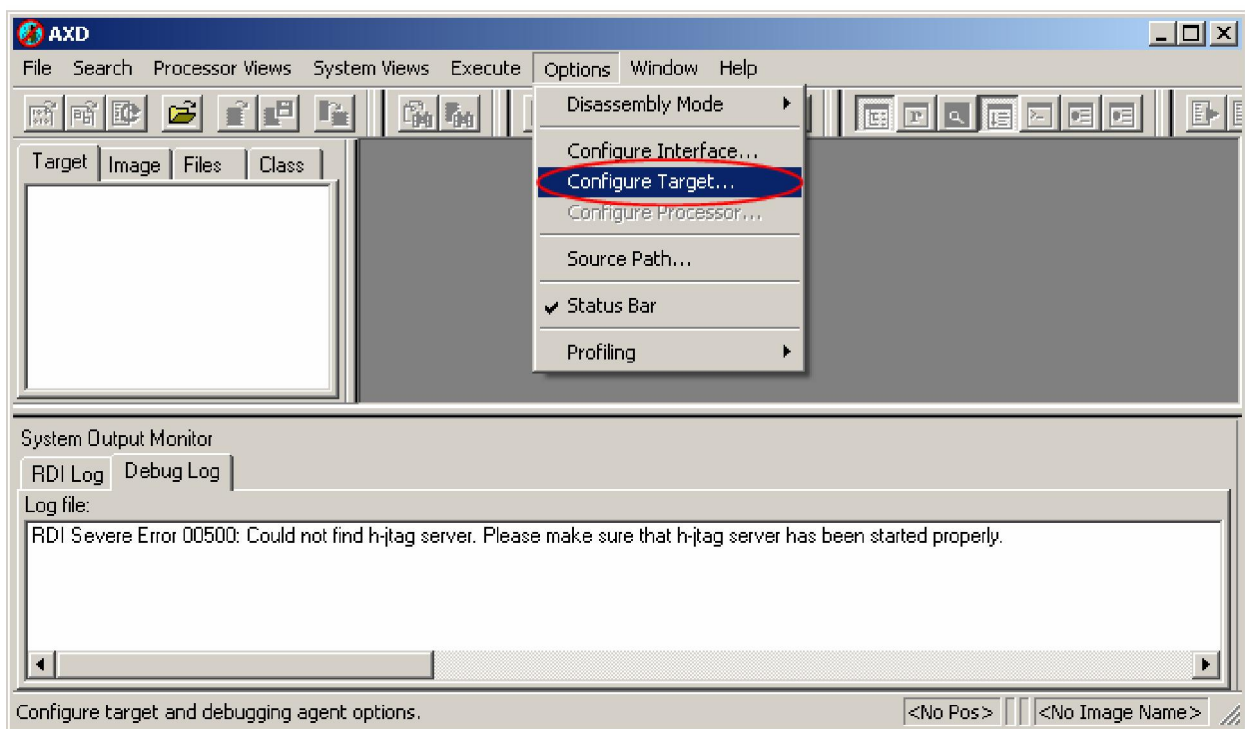


Fig 7-1 Configure Target Menu

Next, the choose target dialog (Fig 7-2) is popped up.

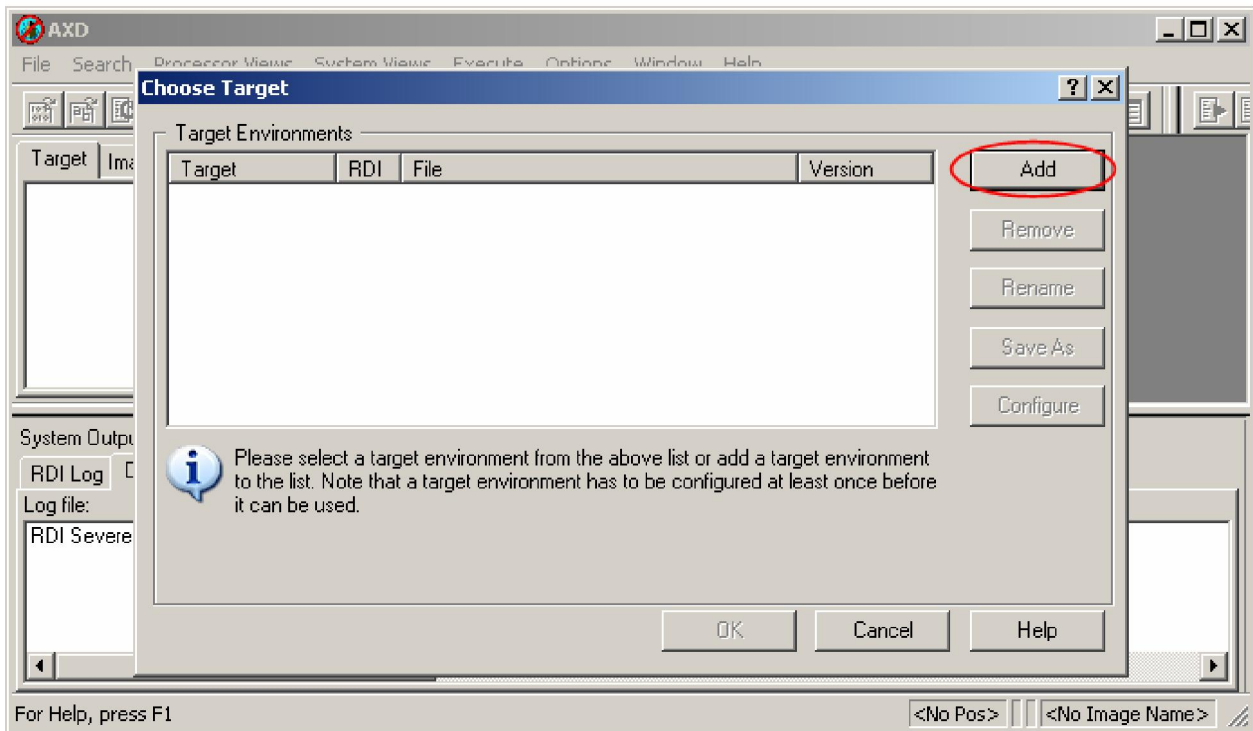


Fig 7-2 Choose Target Dialog

In the above figure, click “Add”, an open dialog (Fig 7-3) is popped up. In this dialog, please choose H-JTAG.DLL, which is located under the folder where H-Jtag is installed. Then click “Open”.

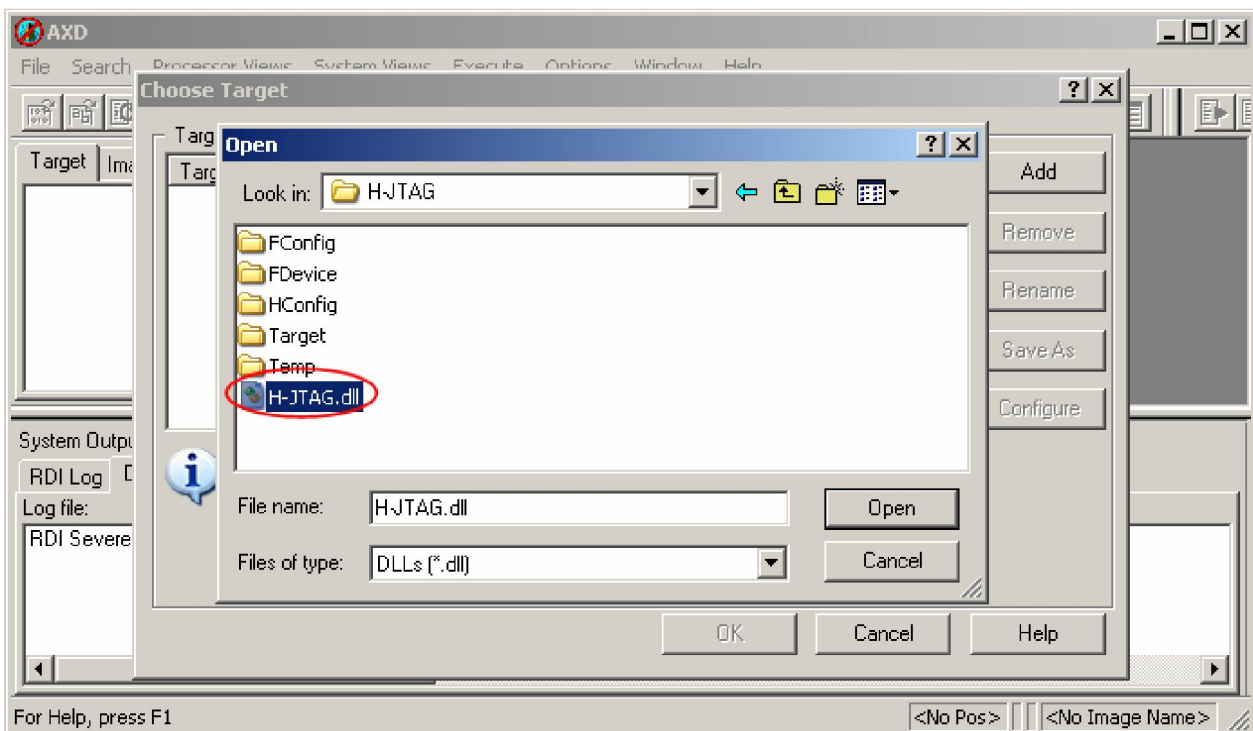


Fig 7-3 Choose H-JTAG.DLL

After H-JTAG.DLL is selected, user can find that H-JTAG.DLL has been added, as shown in Fig 7-4. In this step, user can check the basic information about H-Jtag by clicking “Configure” or double clicking H-JTAG. The basic information is shown in Fig 7-5. To complete the configuration, please click “OK” in Fig 7-4.

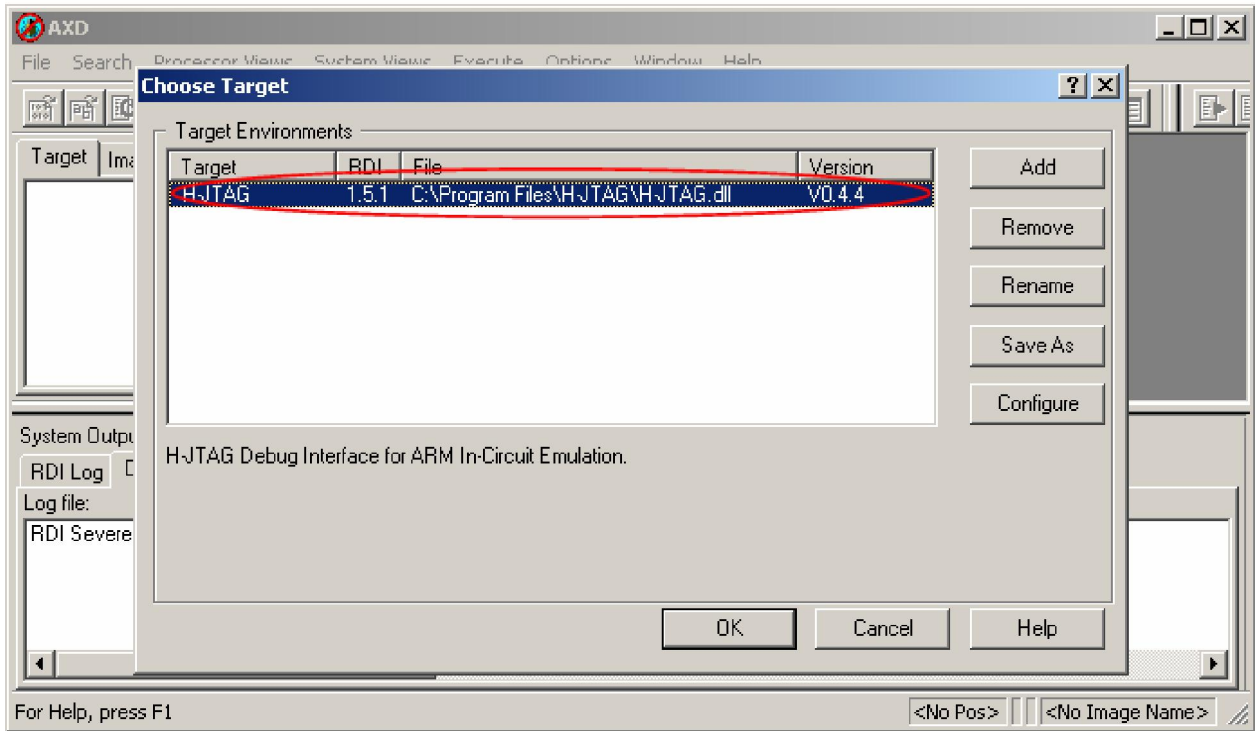


Fig 7-4 AXD Configuration



Fig 7-5 H-Jtag Information

## 7.2 Configure RVDS

RVDS stands for Realview Developer Suit, which is from ARM Corp. This section introduces how to configure RVDS to use with H-Jtag. The introduction is based on RVDS2.0, but the configuration for other versions is similar.

First start RVDS and then click “Click to Connect to a Target”, as shown in Fig 7-6.

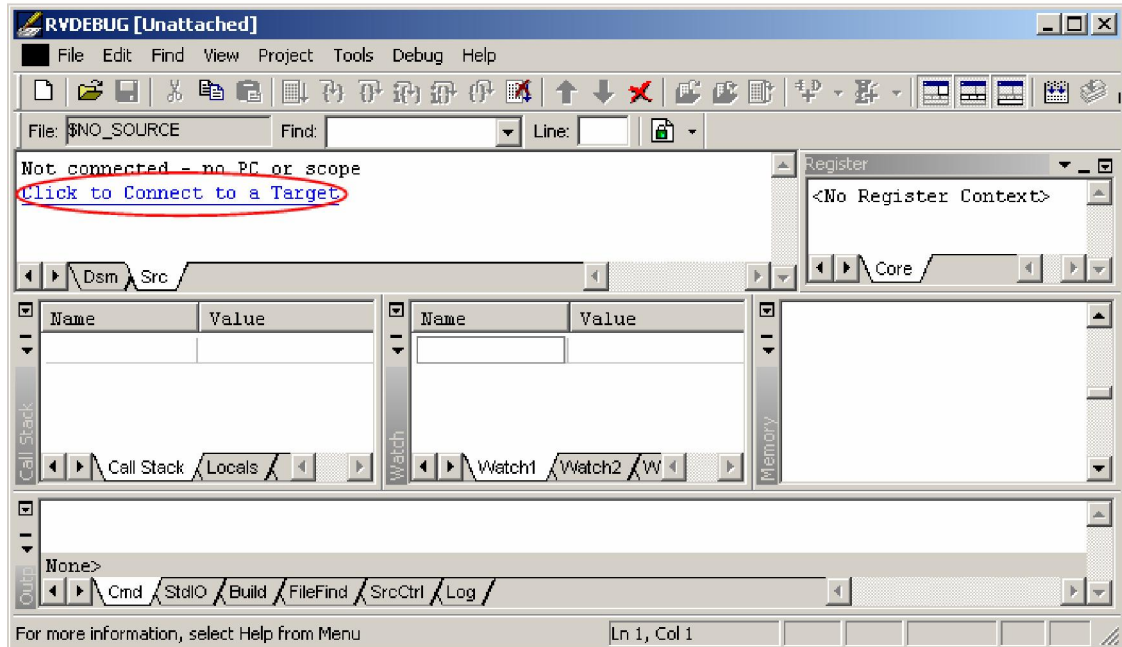


Fig 7-6 RVDS Main Window

Next, the connection control dialog is popped up (Fig 7-7).

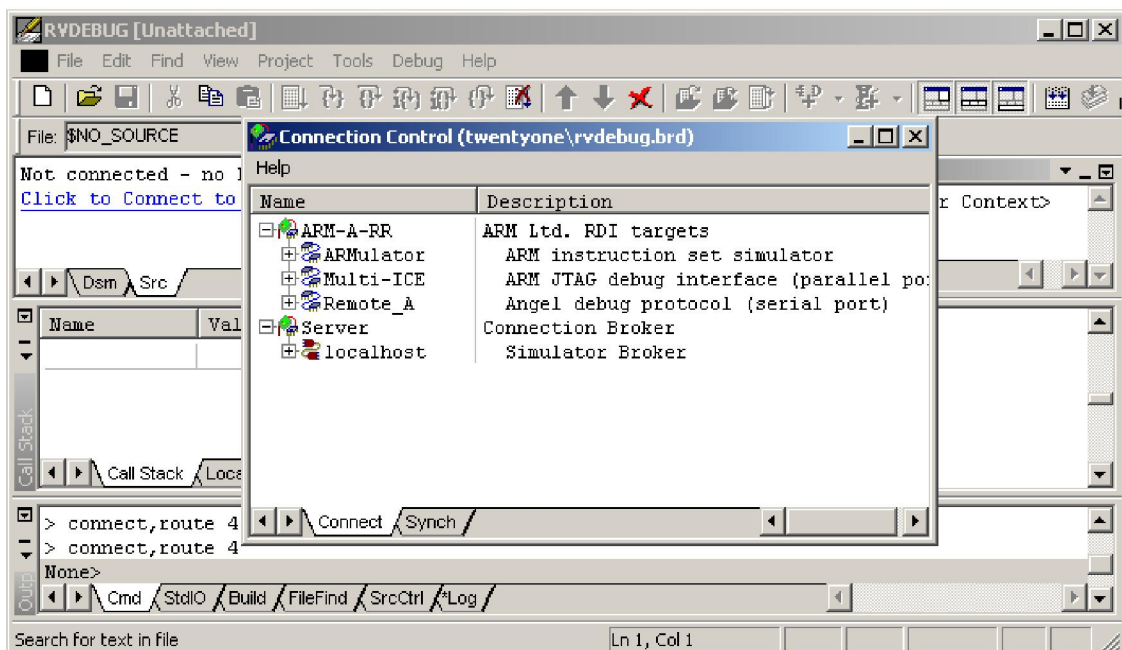


Fig 7-7 Connection Control Dialog

In Fig 7-7, right click inside the dialog to pop up the context menu (shown in Fig 7-8).

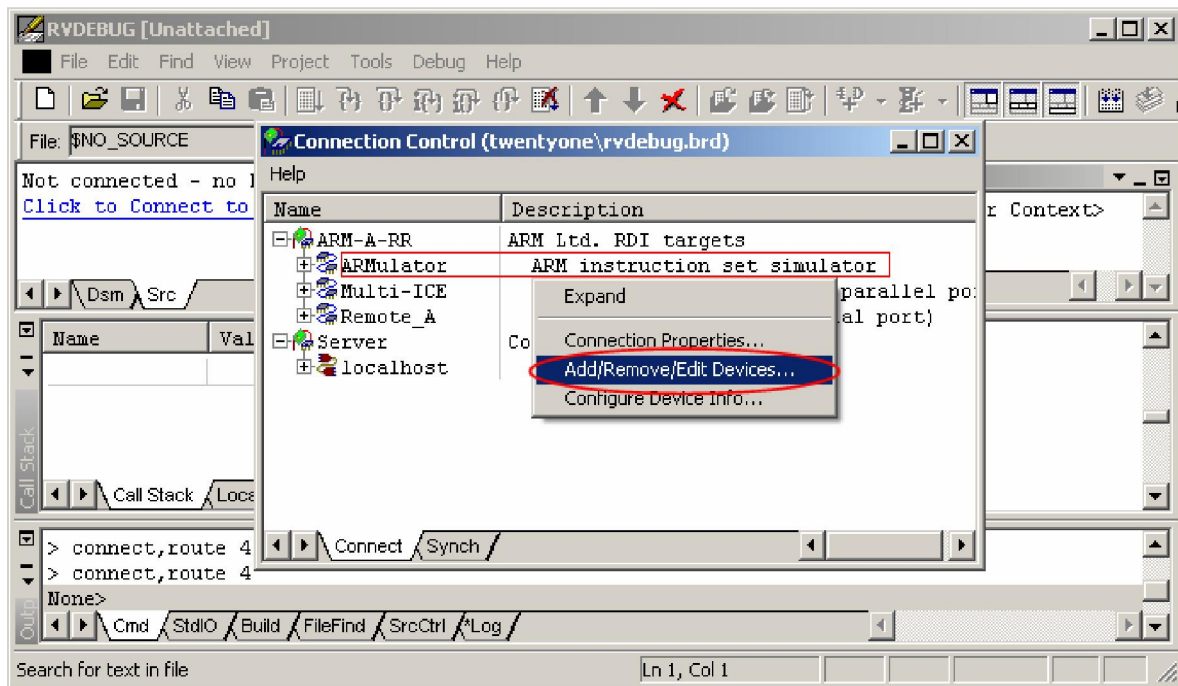


Fig 7-8 Context Menu

In the menu shown in Fig 7-8, click “Add/Revmove/Edit Devices”, the RDI Target List dialog is popped up. The dialog is shown in following figure.

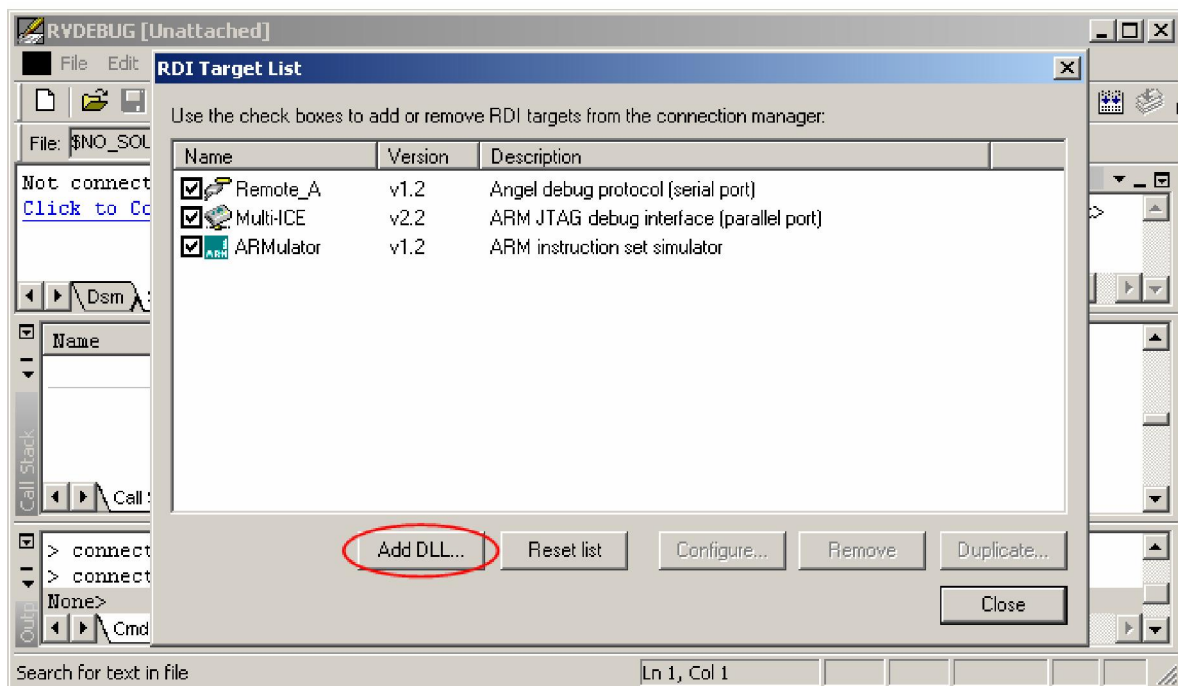


Fig 7-9 RDI Target List Dialog

In Fig 7-9, click “Add DLL”, the Select RDI DLL dialog is popped up. The Select RDI DLL dialog is shown below. In this dialog, choose H-JTAG.DLL located under the installation folder of H-Jtag. Then click “Open”.

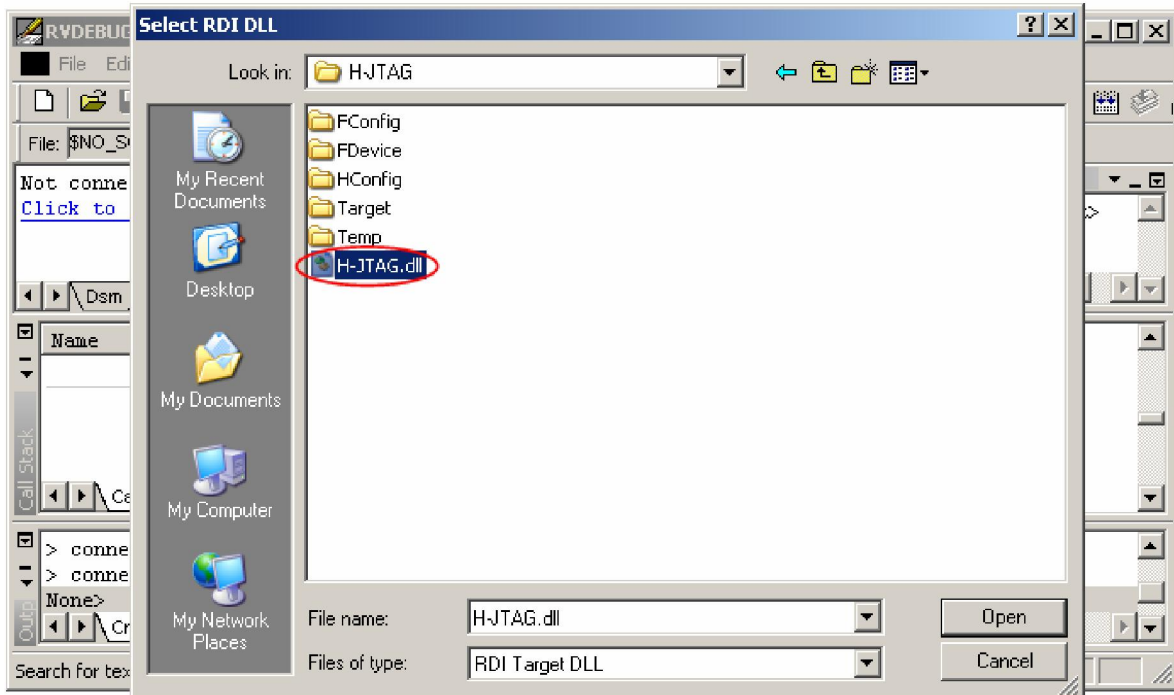


图 7-10 选择 H-JTAG.DLL

After H-JTAG.DLL is selected, a new dialog (Fig 7-11) is popped up. The new dialog is used to create a new RDI target. In this dialog, user can input a short name and a brief description or use the default ones.

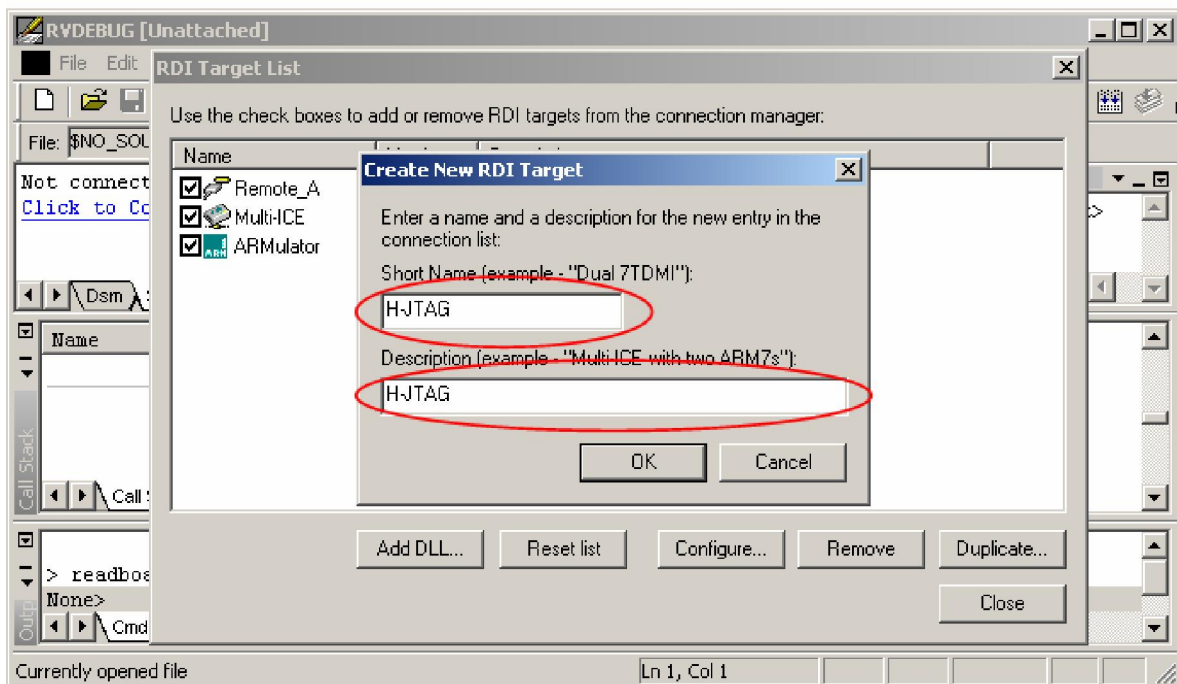


Fig 7-11 Create New RDI Target Dialog

After that, user can find that H-Jtag has been added into RVDS as shown in Fig 7-12. In this step, user can check the basic information about H-Jtag by clicking “Configure” or double clicking H-JTAG. The basic information is shown in Fig 7-13. To complete the configuration, please click “Close” in Fig 7-12.

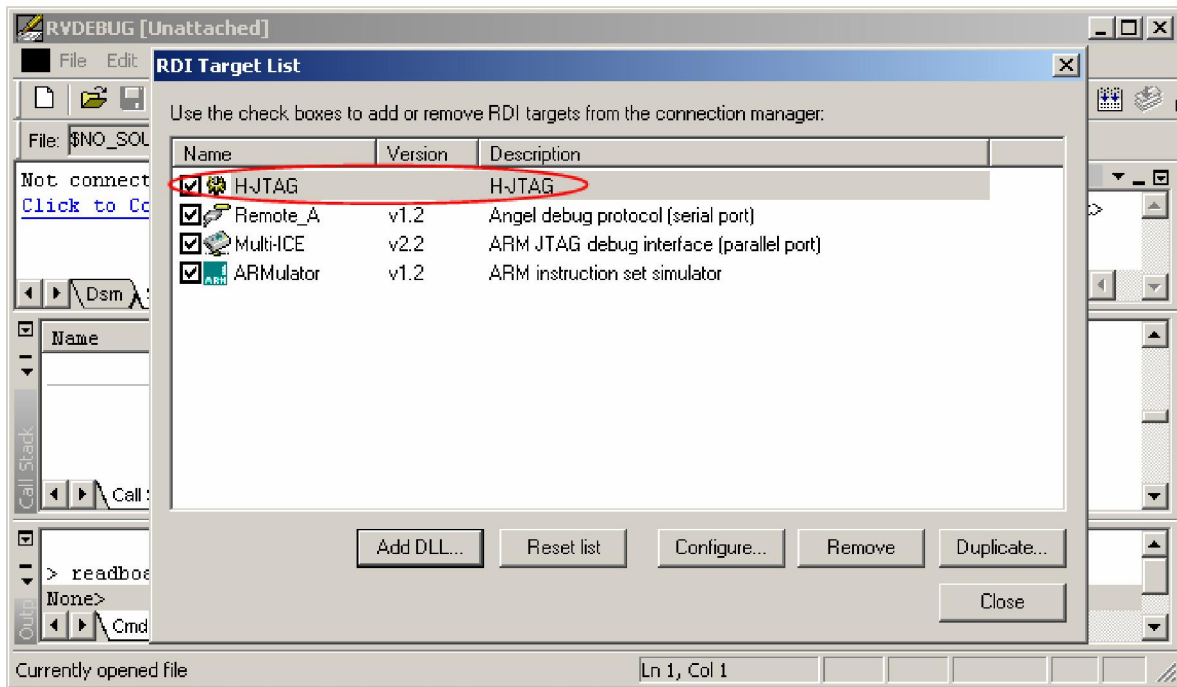


Fig 7-12 RVDS Configuration



Fig 7-13 H-Jtag Information

## 7.3 Configure IAR

IAR stands for IAR Embedded Workbench, which is from IAR Corp. This section introduces how to configure IAR to work with H-Jtag.

First, start IAR and click Project -> Options.

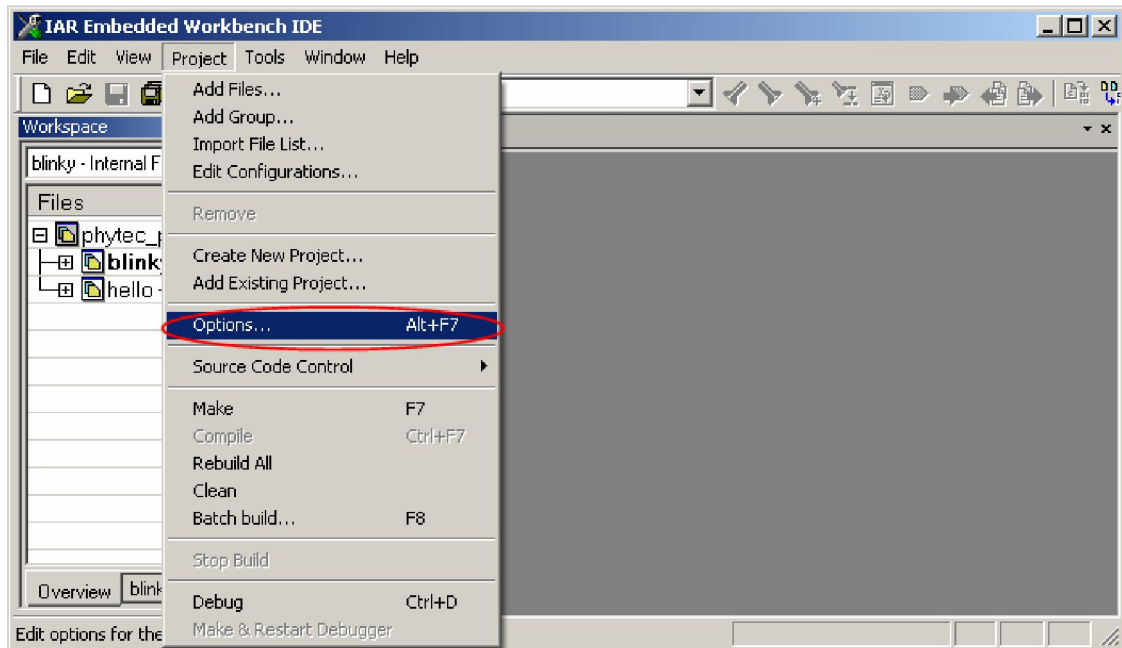


Fig 7-14 IAR Options Menu

Then, following dialog of options is shown up.

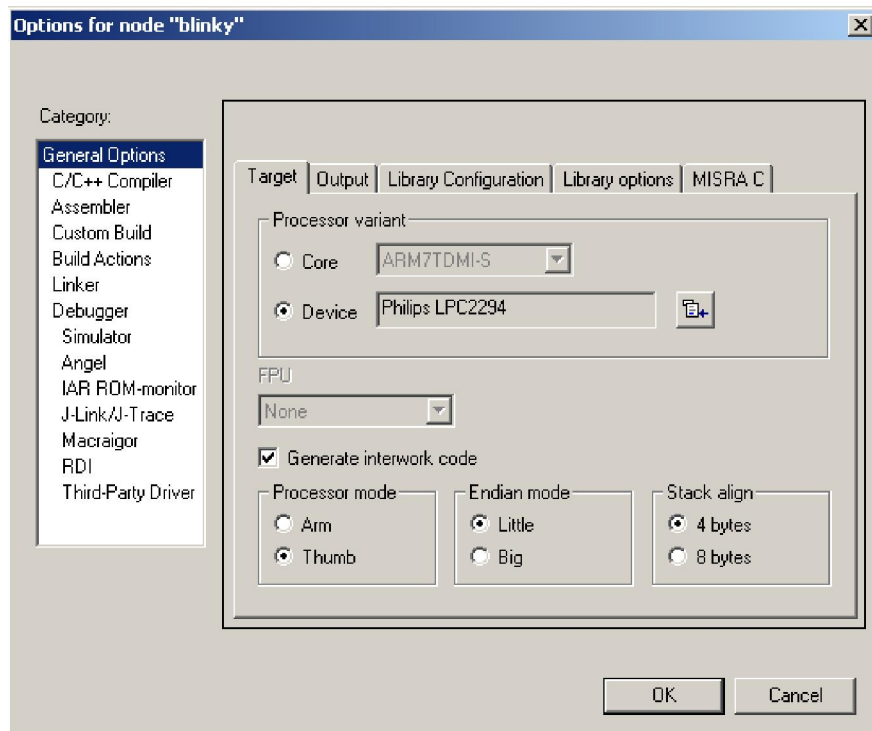


Fig 7-15 Dialog of Options



In Fig 7-15, select Debugger category and then active Setup page. In Setup page, please select RDI as the driver. After the selection, the page looks like Fig 7-16.

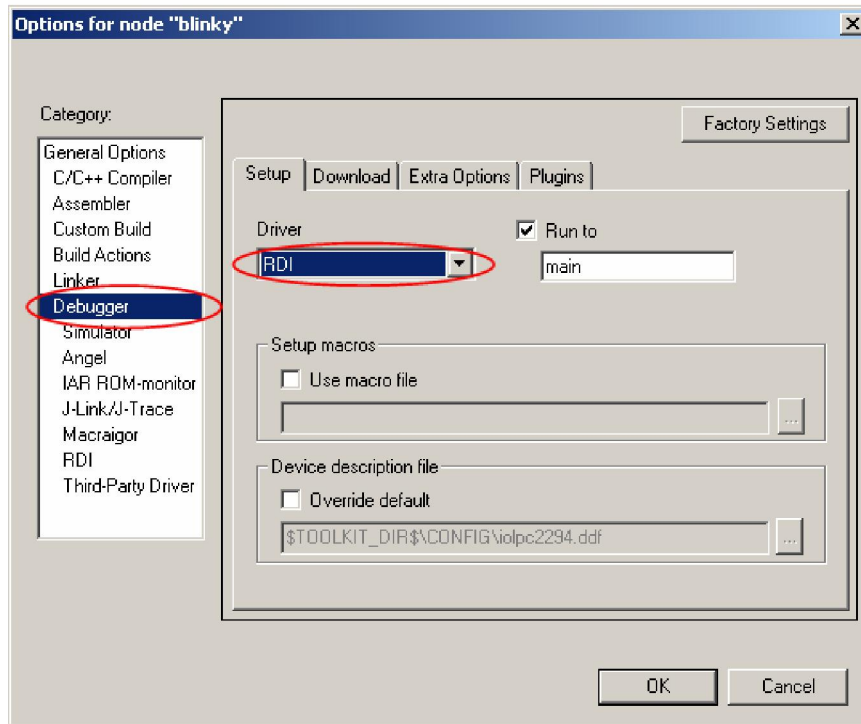


Fig 7-16 Debugger Configuration

Then select RDI category, as shown in following figure. In this page, user needs to specify the path for H-JTAG.DLL.

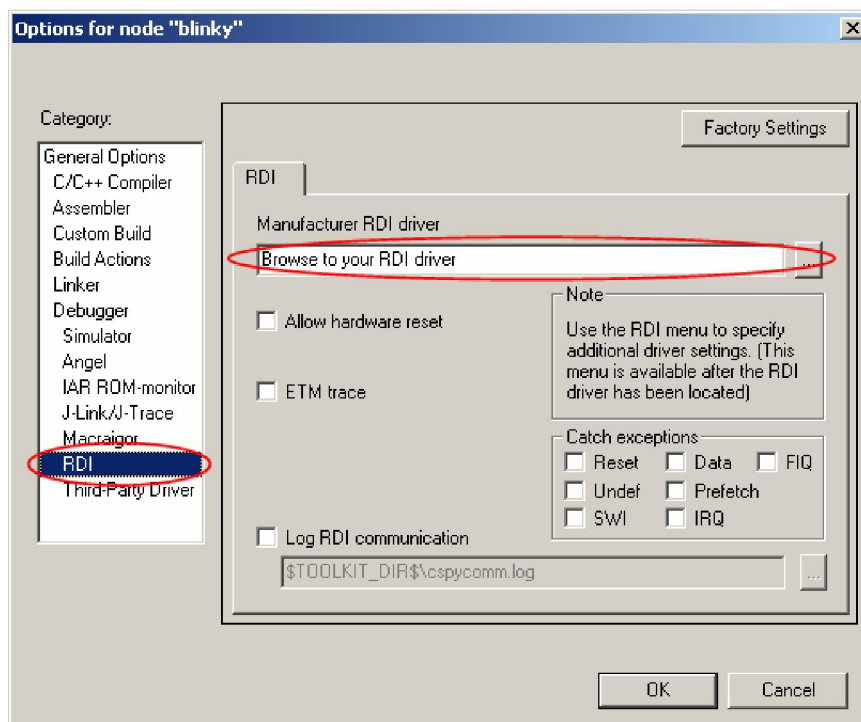


Fig 7-17 RDI Configuration

In Fig 7-17, click “Browse” and select H-JTAG.DLL located under the installation folder of H-JTAG. After this, the dialog looks like Fig 7-18.

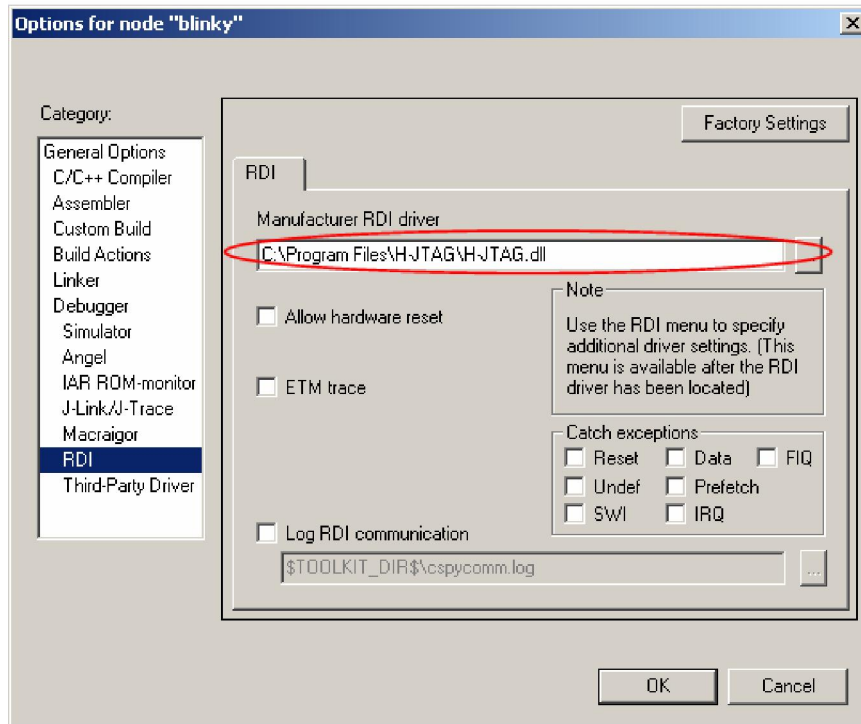


Fig 7-18 Choose H-JTAG.DLL

In above figure, click “OK” to complete the configuration. After the configuration is completed, a new menu named RDI is added in the main window, as shown in Fig 7-19. To check the basic information about H-Jtag, click RDI -> Configure. The basic information about H-Jtag is shown in Fig 7-20.

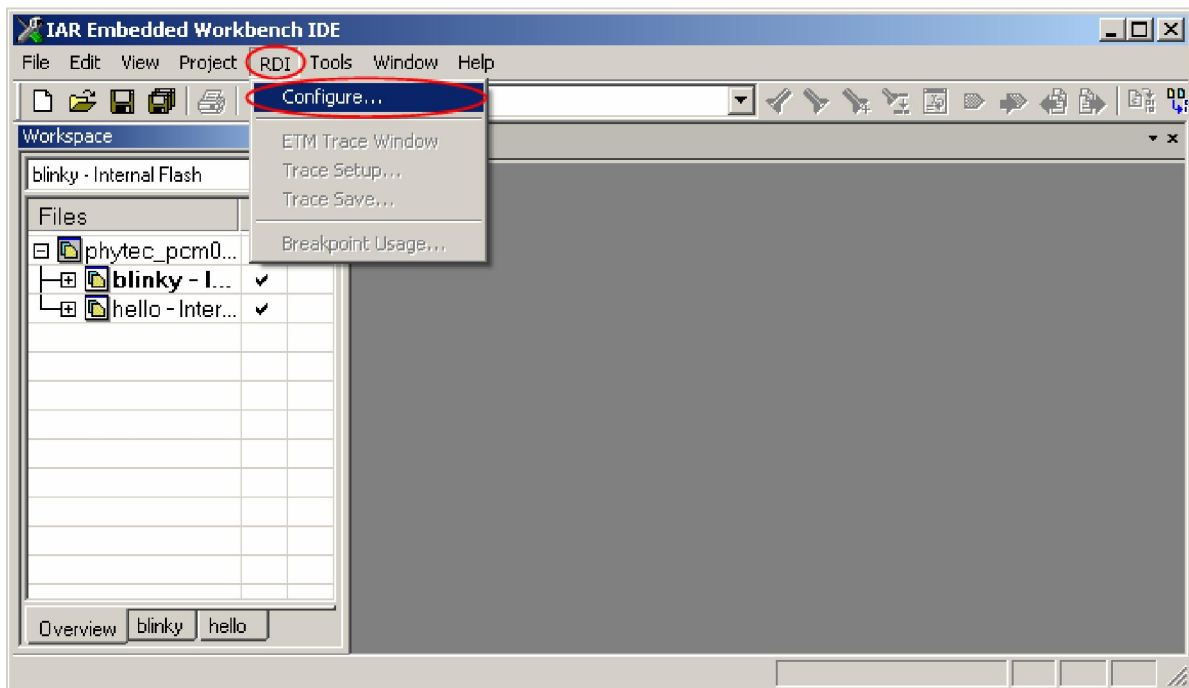


Fig 7-19 New RDI Menu



Fig 7-20 H-Jtag Information



**Note:**

If user wants to use auto flash download in IAR, please disable “Verify Download” and “Use Flash Loader(s)” in IAR. Fig 7-21 shows how to disable these options.

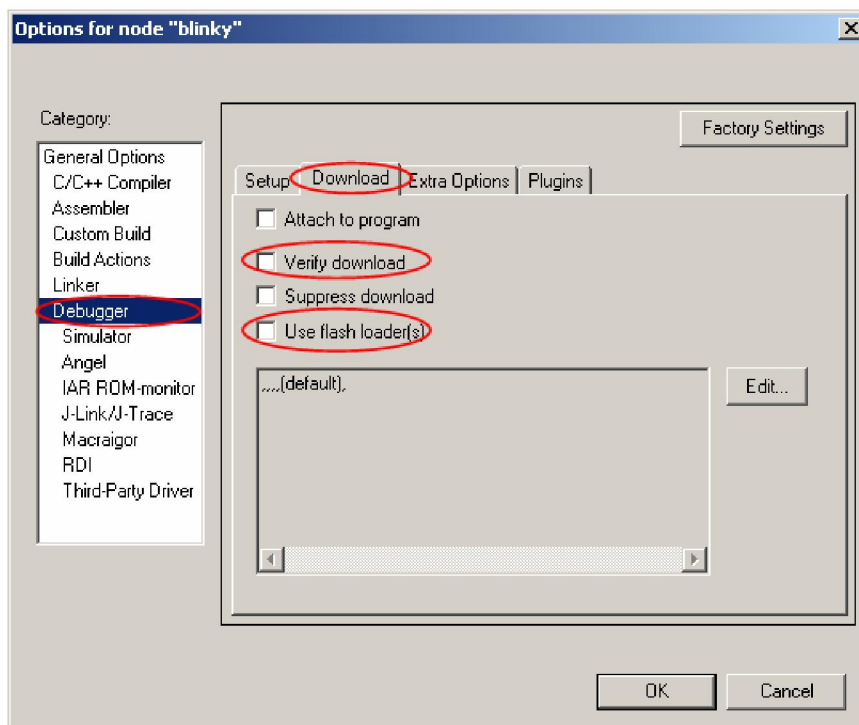


Fig 7-21 Disable Verify Download Option

## 7.4 Configure KEIL/MDK

KEIL for ARM is an IDE from KEIL Corp. This section introduces how to configure KEIL to work with H-Jtag

From H-JTAG V0.9.2, H-JTAG uses the AGDI interface under KEIL. The RDI interface has been disabled by H-JTAG under KEIL

First, run TOOLCONF.EXE located under the installation direction of H-JTAG. The purpose is to update the configuration file (TOOS.INI) of KEIL. In TOOLCONF.EXE, use the browse button to locate the configuration file TOOLS.INI, which can be found under the installation direction of KEIL, as shown in Fig 7-22. Then click the Config button to perform the update. After the update is completed successfully, click Exit to exit TOOLCONF.EXE.

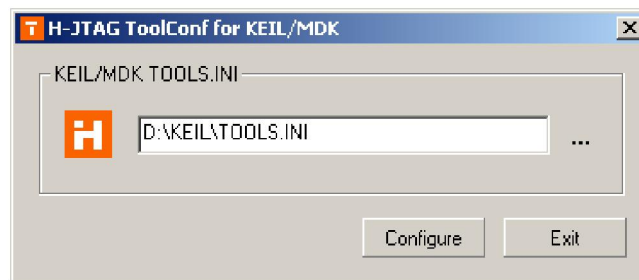


Fig 7-22 ToolConf for KEIL/MDK

Next, start KEIL and open a project. Then click Project -> Options for Target, as shown in Fig 7-23.

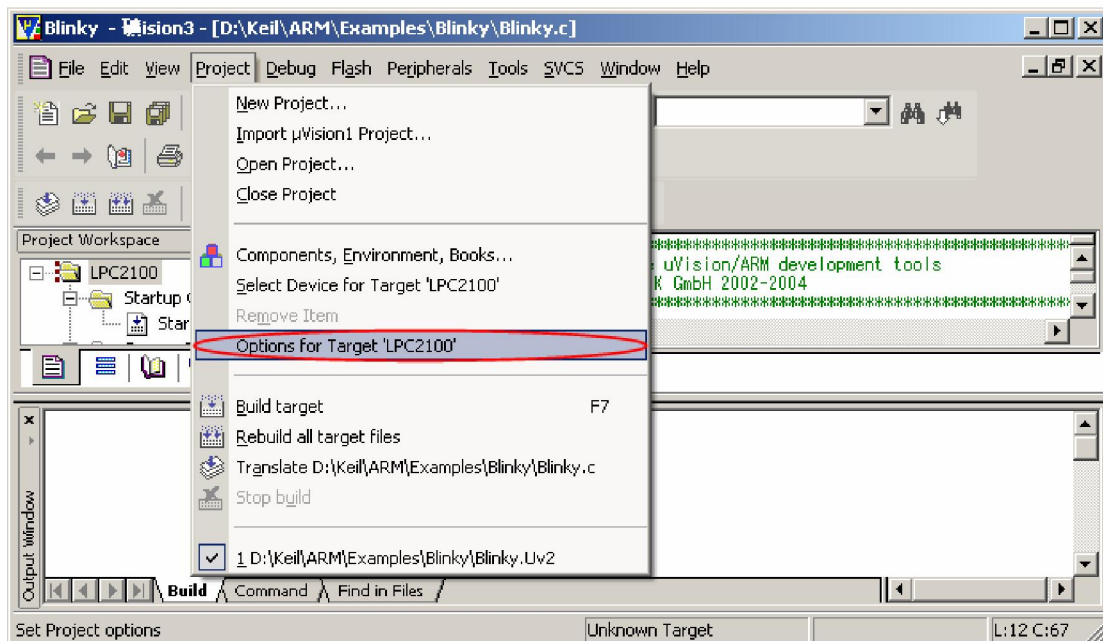


Fig 7-23 Menu of Options

Then, a dialog of options is shown up, which looks like Fig 7-24.

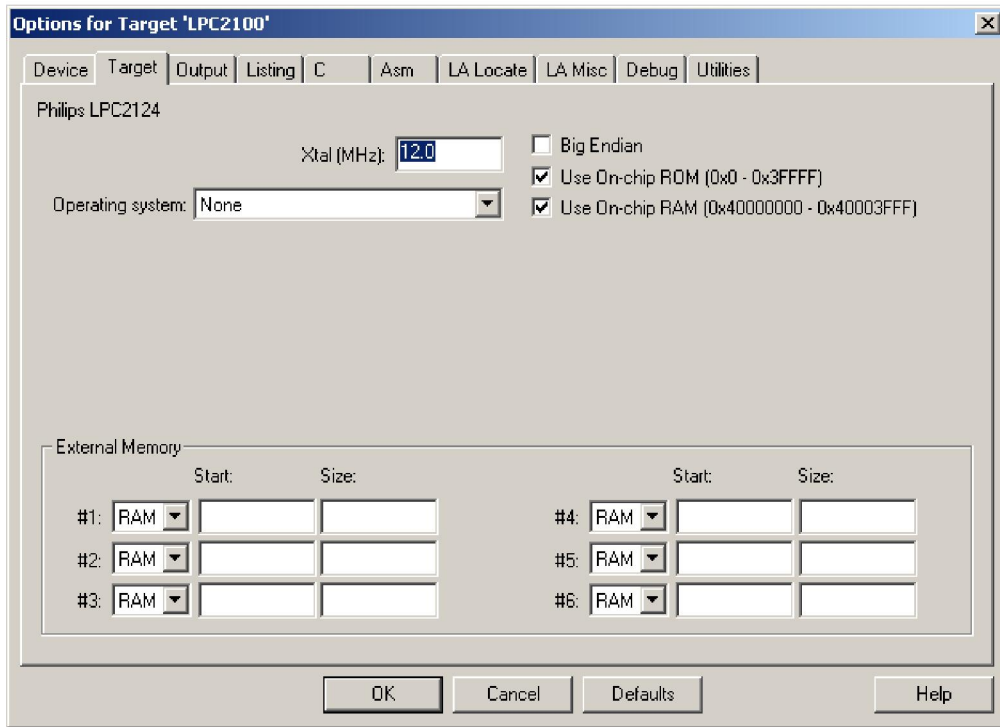


Fig 7-24 Dialog of Options

In the dialog shown in Fig 7-24, active the Debug page.

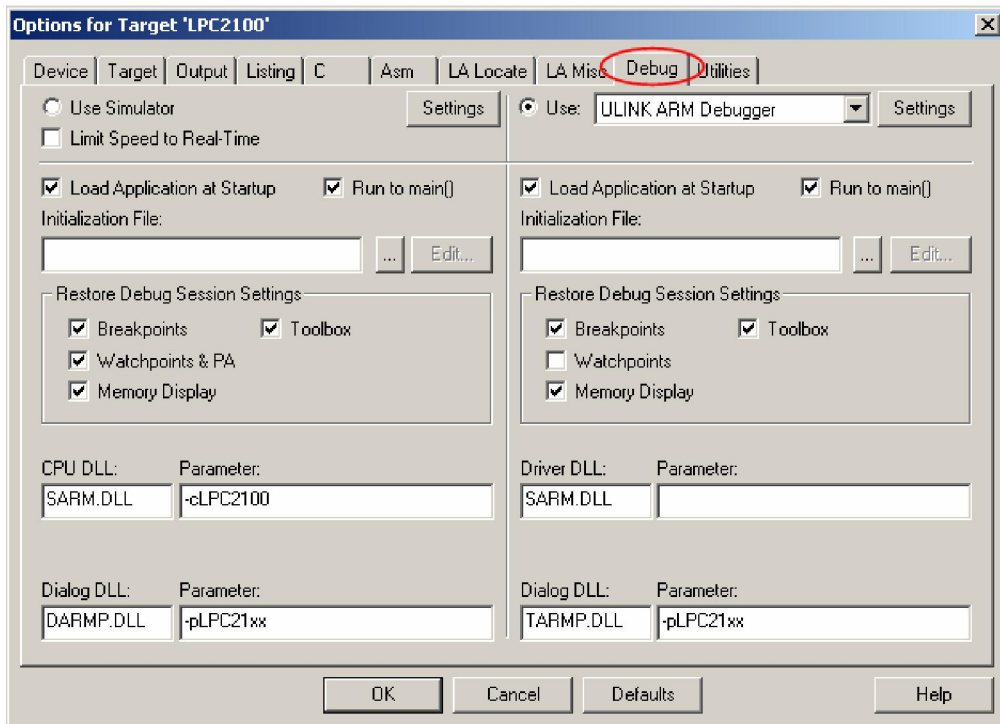


Fig 7-25 Debug Page

In the Debug page, H-JTAG drivers, H-JTAG ARM, H-JTAG CORTEX-M0, H-JTAG CORTEX-M3 and H-JTAG CORTEX-M4, can be found on the list, as shown in Fig 7-26 and 7-27. User should select one of the drivers accordingly.

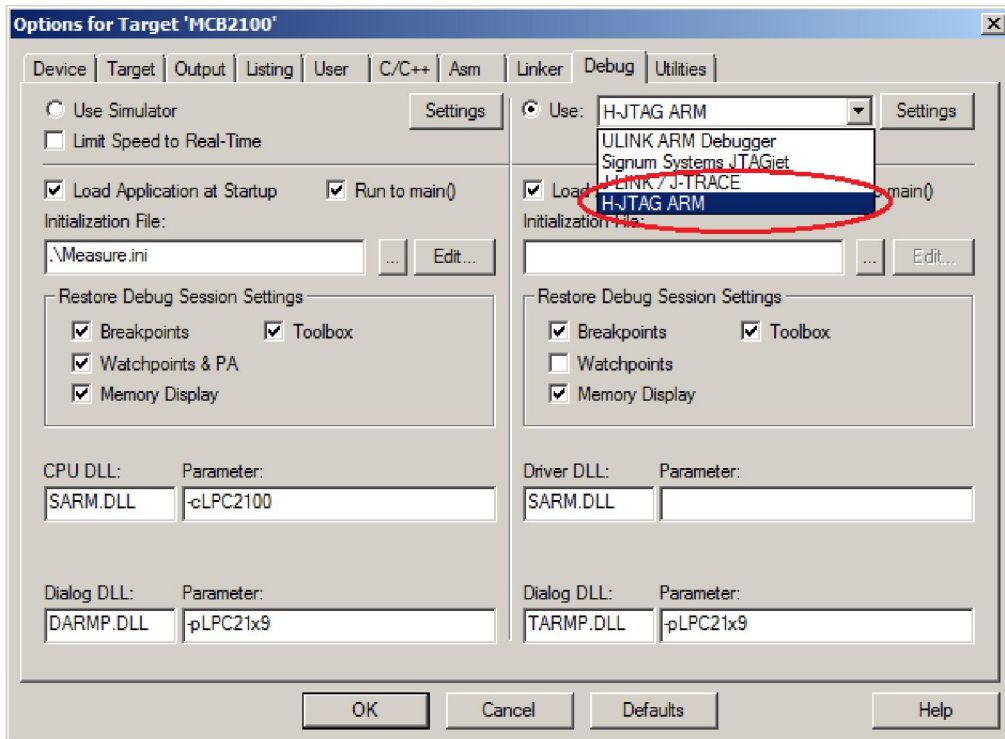


Fig 7-26 H-JTAG Debug Drivers

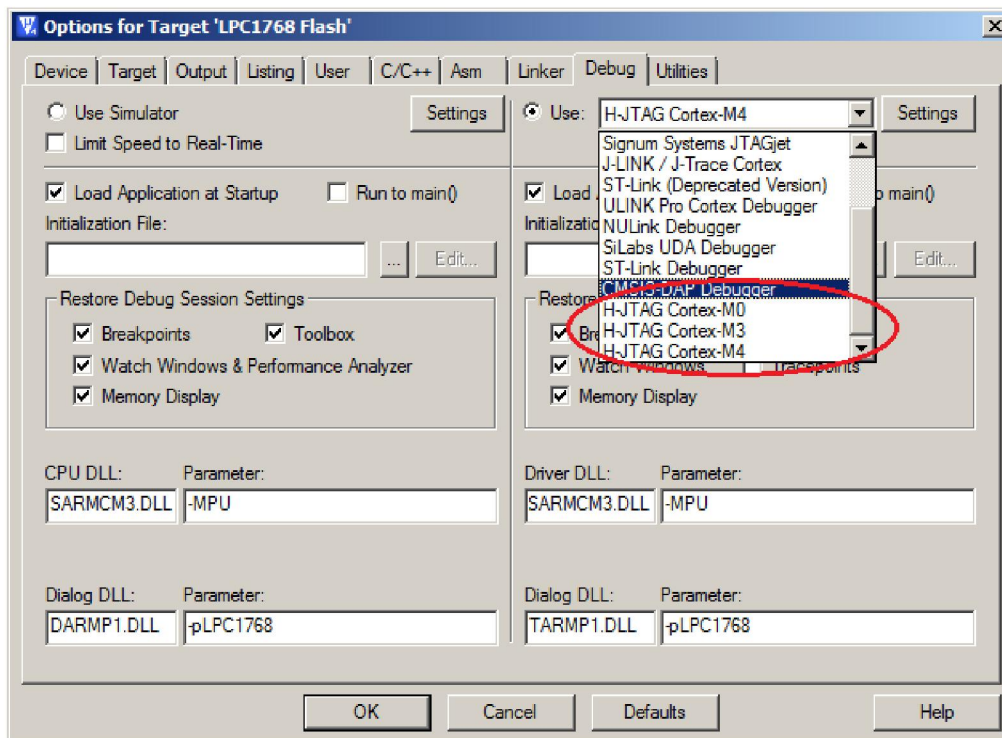


Fig 7-27 H-JTAG Debug Drivers

Then, active the Utilities page, as shown in Fig 7-28.

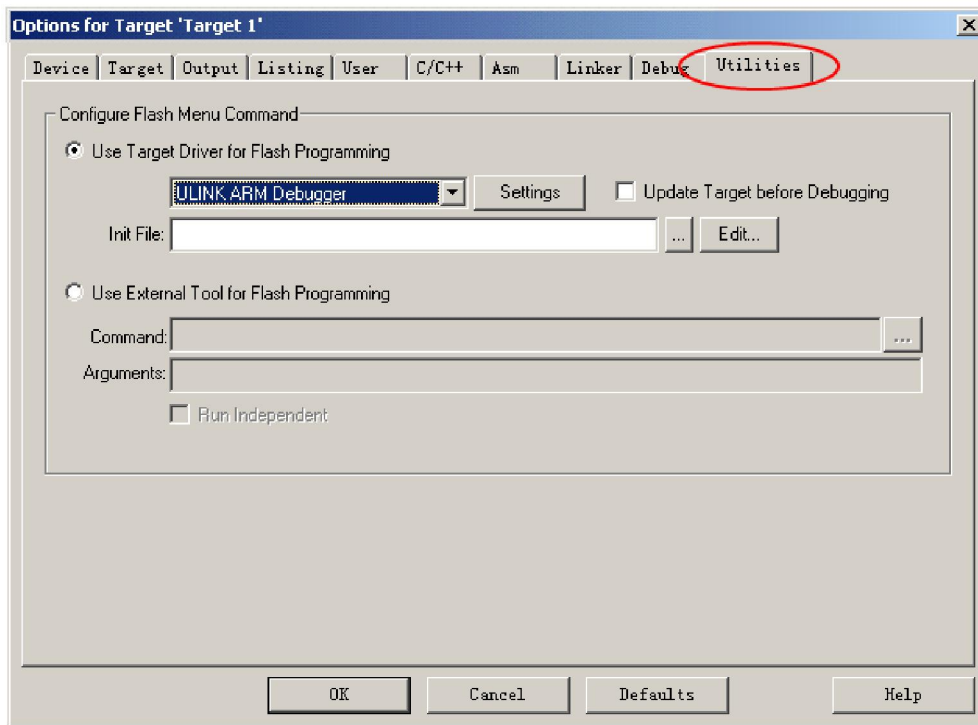


Fig 7-28 Utilities Page

In the Utilities page, H-JTAG drivers, H-JTAG ARM, H-JTAG CORTEX-M0, H-JTAG CORTEX-M3 and H-JTAG CORTEX-M4, can be found on the list, as shown in Fig 7-29 and 7-30. User should select one of the drivers accordingly.

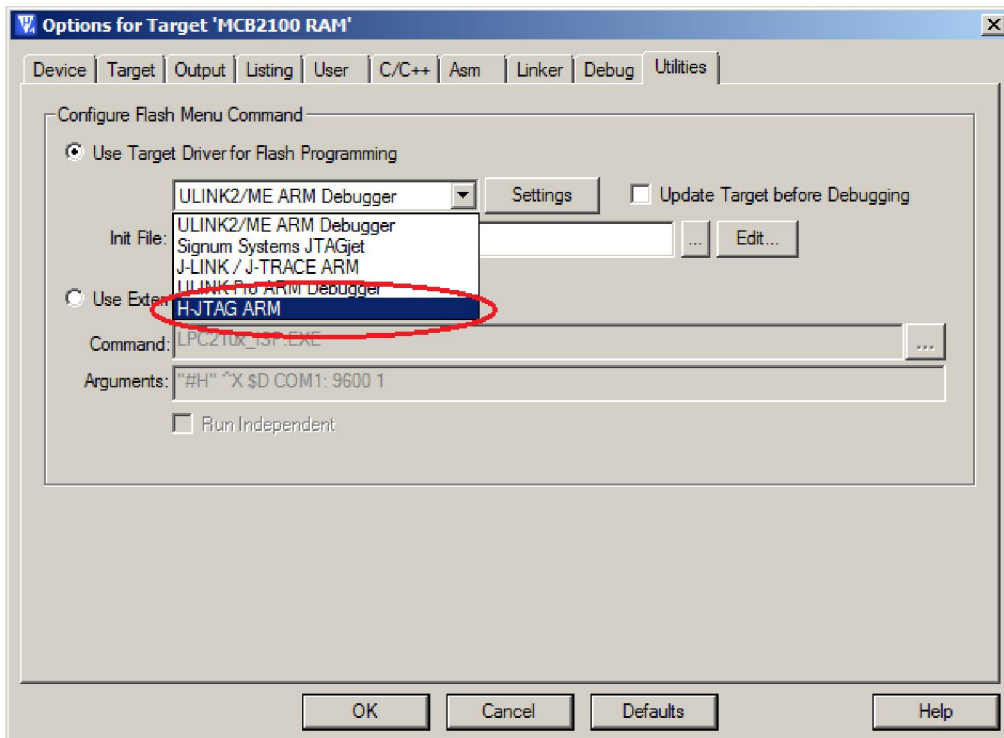


Fig 7-29 H-JTAG Utilities Drivers

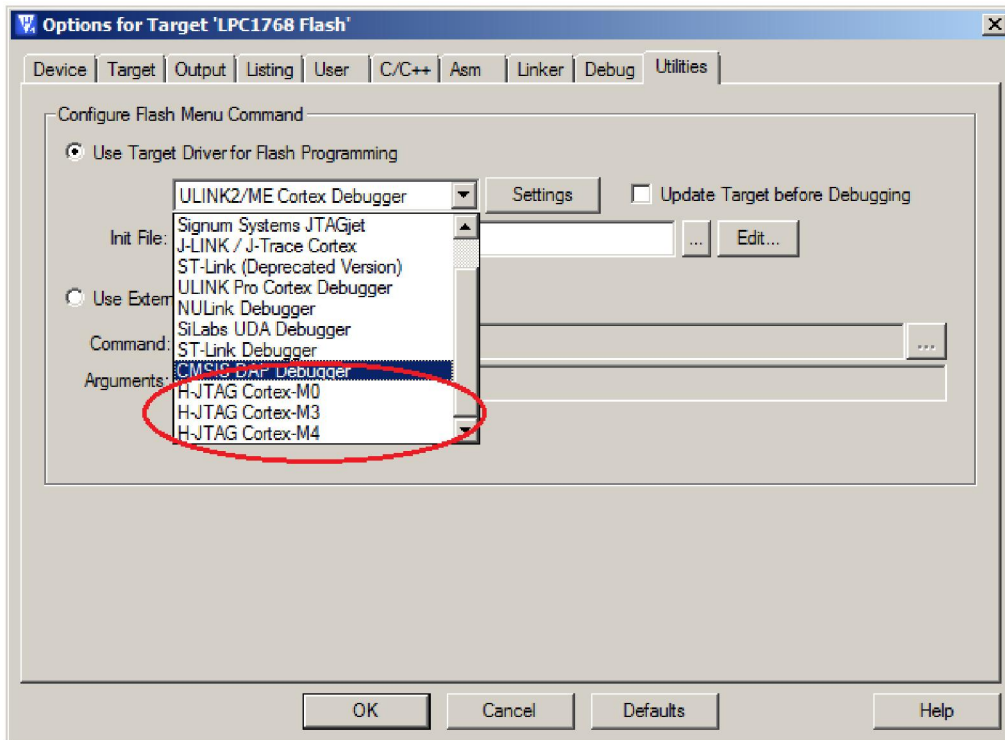


Fig 7-30 H-JTAG Utilities Drivers



---

## Chapter 8 Auto Flash Download

Sometimes, user needs to debug the program in Flash. Flash and RAM are different. To write data into Flash, certain instructions must be executed. So, to debug in Flash, the program need to be written into it first. H-Jtag supports auto flash download for debugging in flash. With auto flash download, program can be directly downloaded or written into flash before debugging, just like debug in RAM/SDRAM.

To use auto flash download, please enable the auto download option in H-Jtag as shown in Fig 8-1. Meanwhile, please select right target flash and provide appropriate configuration in H-Flasher.

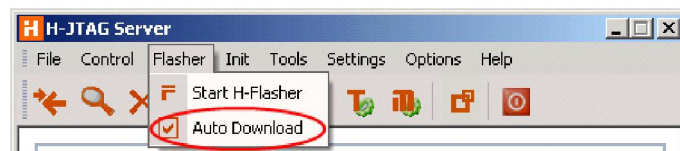


Fig 8-1 Enable Auto Flash Download

While starting debugging, H-Jtag will determine where and which part of the program should be downloaded according to the information extracted from the image. For those need to be written into flash, H-Jtag will call H-Flasher to complete it automatically. Meanwhile, a dialog will be shown to indicate the progress.

 **Note:**

Auto flash download can be used for both on-chip flash and external NOR flash. For chips supports complicated memory configuration, like MMU/REMAP, it is recommended to disable MMU/REMAP through init script before debugging.

 **Note:**

H-Flasher Lite does not support Auto Flash Download. To use Auto Flash Download, please run H-Flasher.

Next, an example based on NXP LPC1766 is given to illustrate how to use Auto Flash Download for flash debugging. For other hardware platforms, the basic procedure is the same.

## 8.1 Example: Auto Flash Download For LPC1766

In this example, LPC1766 is used to illustrate how to use Auto Flash Download for debugging in Flash. First, enable Auto Flash Download in H-Jtag, as shown in Fig 8-2.

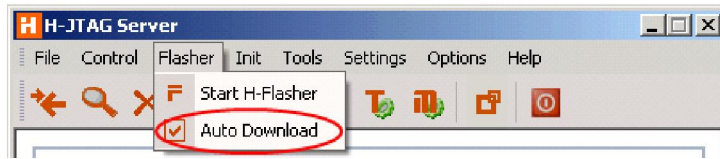


Fig. 8-2 Enable Auto Download

After enabling Auto Flash Download, a dialog will be shown to tell that H-Flasher may change the clock configuration of the MCU during flash programming. This change might have potential affect on the debugging of user's program.

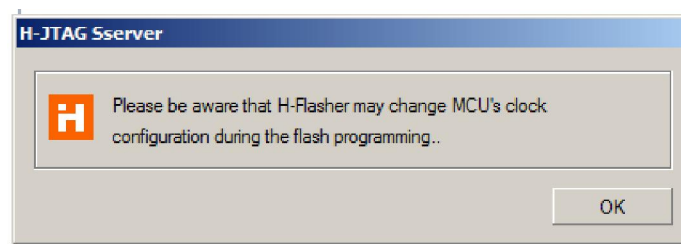


Fig. 8-3 Auto Download Note

Next, start H-Flasher and select LPC1766 as the target flash, as shown in Fig. 8-4.

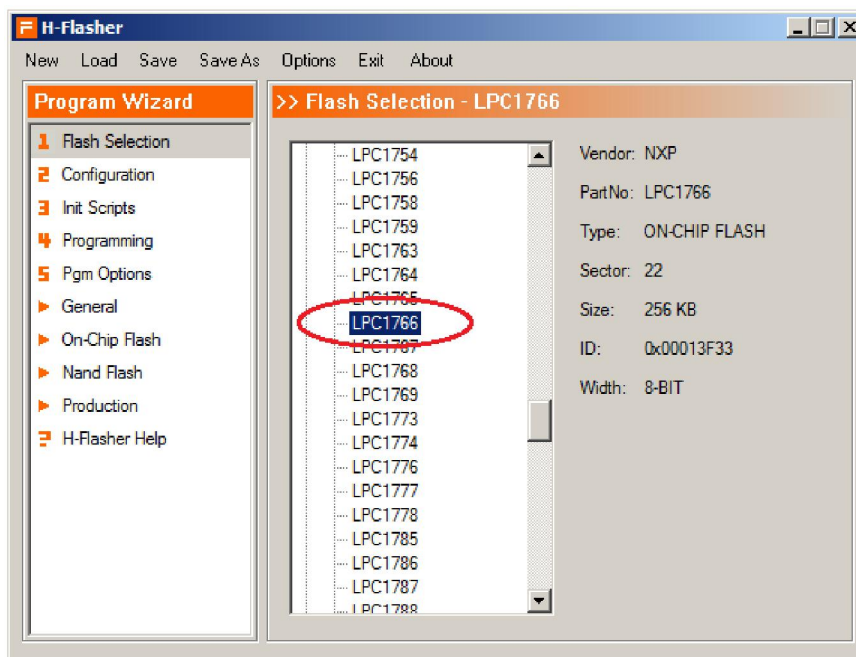


Fig. 8-4 Select LPC1766

Because H-Flasher contains all the essential information for LPC1766, user can skip other configurations and go directly to the Programming page. In the programming page, click “CHECK” to see if the target flash could be checked successfully. This operation can tell user if the configuration works or not. As shown Fig. 8-5, LPC1766 can be checked successfully.

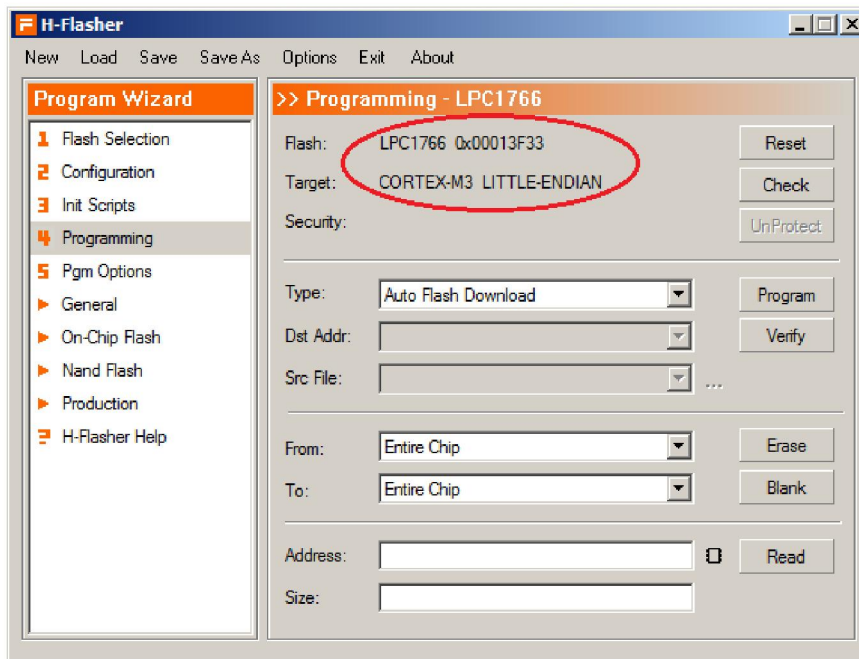


Fig. 8-5 Check Flash Successfully

Next, user can launch debugger from the IDE. First, user will see the programming progress dialog of H-Flasher, as shown in Fig. 8-6. When it is done, user can start debugging.

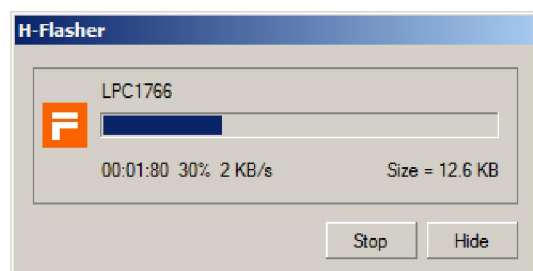


Fig 8-6 Progress Dialog

---

## Chapter 9 H-Flasher Production Mode

H-Flasher supports production mode. In production mode, the programming process is controlled automatically through the detection of connecting/disconnecting of targets, which can improve the efficiency significantly.

To enable production mode, please tick the option as shown in following figure.

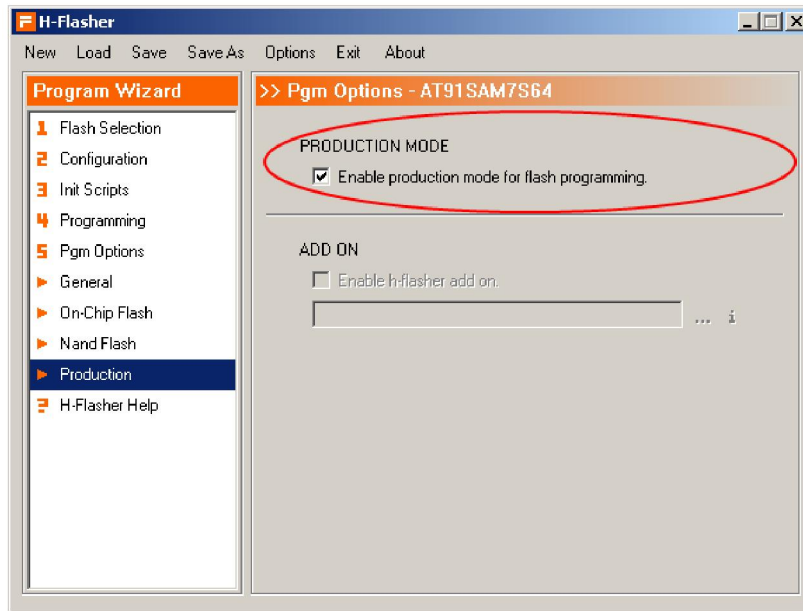


Fig 9-1 Enable Production Mode

When production mode is enabled, H-Flasher will enter this mode by clicking Program in the Programming page. As shown in Fig. 9-2, H-Flasher is in waiting status and user is required to connect target to the emulator.

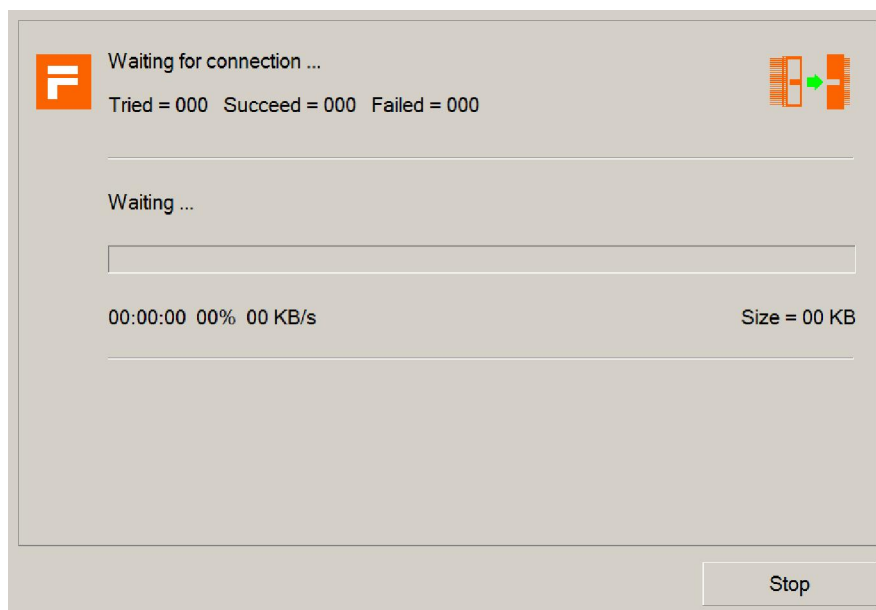


Fig. 9-2 Production Mode: Waiting Status, Please Connect Target

When a target is connected, H-Flasher will detect it and then start the programming automatically, as shown in Fig. 9-3.

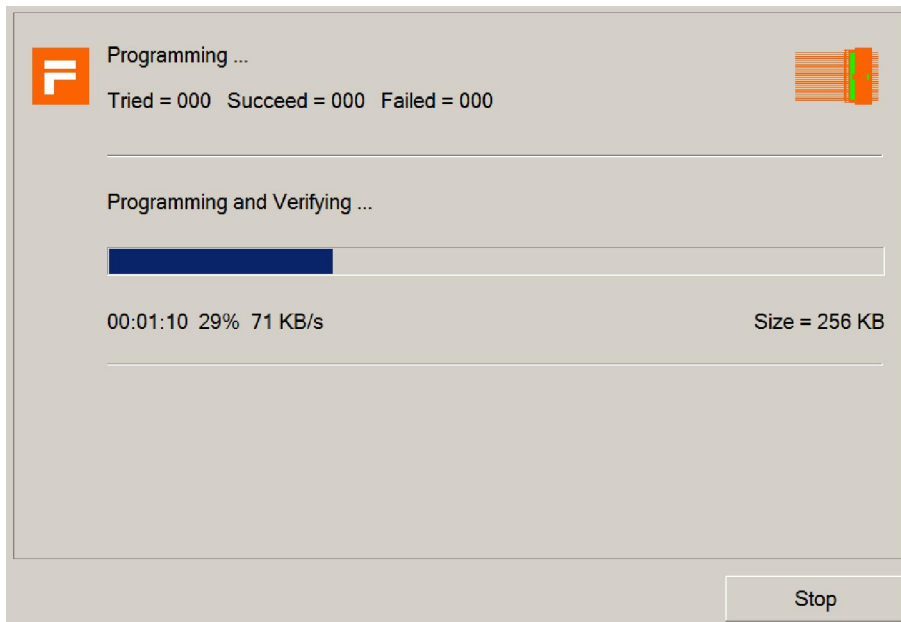


Fig. 9-3 Production Mode: Programming

When the programming is completed, H-Flasher will show the result and tell user to disconnect the target from emulator.

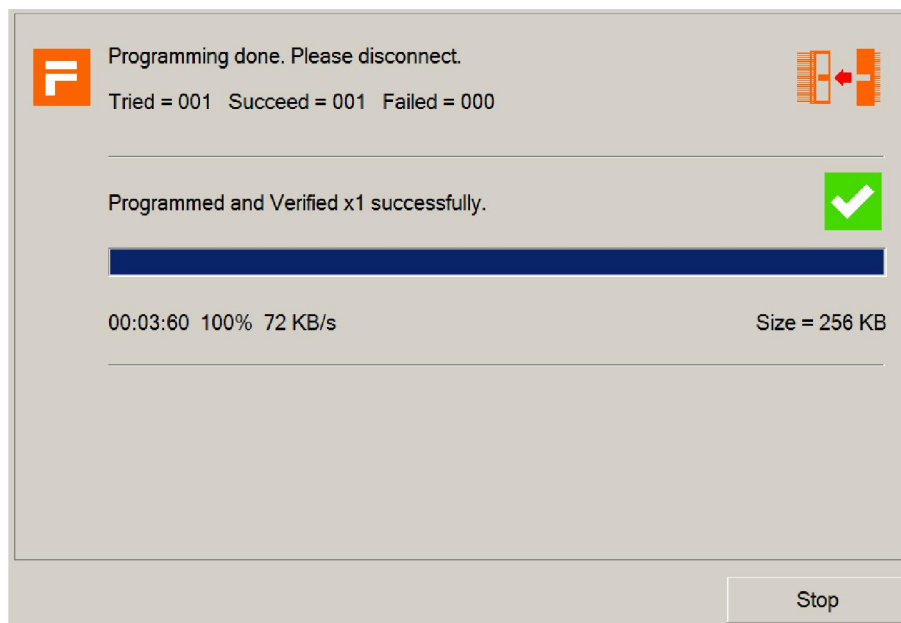


Fig. 9-4 Production Mode: Programming Completed, Please Disconnect Target

---

When the target is disconnected, H-Flasher will enter waiting status again. User can connect next target for programming.

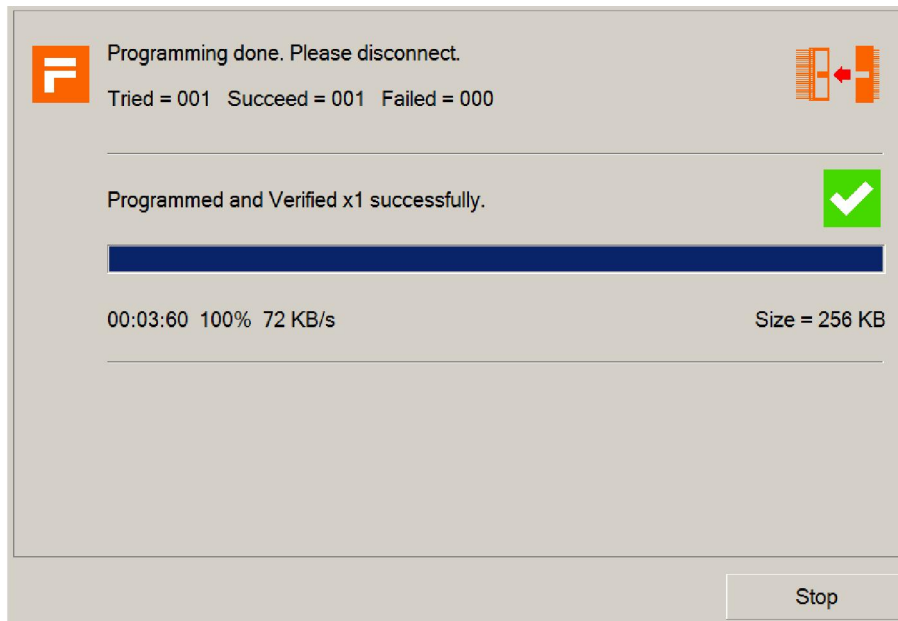


Fig. 9-5 Production Mode: Waiting Status, Please Connect Next Target

---

## Appendix A: H-JTAG Q & A

This is the Q&A section, which collects some of the common questions and answers. User can refer to this section for trouble shooting. If you need any further help, please contact our technical supports.

### **Q. Can't halt target and make it enter debug state.**

**A.** Normally, this is caused by hardware issue. It is also possible that the problem is caused by other issues.

To locate the cause, please do the following checks.

1. Check if the TAP configuration is correct. Some chip requires special TAP configuration.
2. Check if the target is encrypted, which makes the JTAG inaccessible.
3. Check if the JTAG cable is connected well. If a non-standard cable is used, double check the connection.

### **Q. Unable to find target. Please make sure that the hardware is properly connected and powered up.**

**A.** Please do following checks to locate the cause.

1. Check if the target is connected to the emulator properly?
2. Check if the JTAG cable is connected well. If a non-standard cable is used, double check the connection.
3. Check if the TAP configuration is correct.
4. Check if the TGT light of the H-JTAG USB emulator is on.

### **Q. Can't open the H-JTAG USB hardware/emulator.**

**A.** It means that the H-JTAG SERVER can't access the H-JTAG USB emulator. Please check follows.

1. Check if the emulator is connected to PC properly.
2. Check if the driver of the emulator is installed correctly.

### **Q. H-jtag server is not ready.**

**A.** Generally, it means that no target is detected by H-JTAG SERVER.

### **Q. Can't find h-jtag server.**

**A.** It means that H-JTAG SERVER is not running. Please start H-JTAG SERVER.

### **Q. Auto init has been enabled, but no init script is found.**

**A.** It indicates that no init script is found, but the Auto Init option is enabled. Please do one of the follows.

1. Disable Auto Init (H-JTAG SERVER -> Init -> Disable Auto Init).
2. Provide the init script (H-JTAG SERVER -> Init -> Init Script).

### **Q. Unknown target.**

**A.** It means that H-JTAG SERVER can't recognize the target. Possible reasons are

1. The JTAG ID of the target is not in the list of H-JTAG SERVER. So, it can't be recognized. User can designate the ARM core manually.
2. H-JTAG SERVER read the wrong JTAG ID. Please check the software settings and the hardware connection.

---

**Q. Can't receive response from target's SWD interface.**

**A.** This error indicates that the SWD mode is selected, but H-JTAG SERVER can't receive response from target.

The possible reasons are

1. The target doesn't support SWD.
2. The connection has some problem.

**Q. No RTCK signal is detected. Please ensure that the target supports RTCK and the signal is connected.**

**A.** This error indicates that ADAPTIVE TCK is selected, but H-JTAG SERVER can't detect the RTCK signal.

The possible reasons are

1. The target doesn't support RTCK;
2. The connection of the RTCK signal has some problem.

**Q. The TCK speed is too high. Please reduce the TCK speed manually.**

**A.** This indicates that the TCK speed is too fast for the target. The possible reasons are

1. The TCK is too high.
2. The TAP configuration is not correct.

**Q. Can't find h-flasher. Please make sure that h-flasher has been started properly.**

**A.** This message indicates that H-FLASHER is not running. Please start H-FLASHER.

**Q. Can't download driver into the specified address.**

**A.** This error indicates that H-FLASHER can't download the driver into the specified RAM space.

Possible reasons are

1. The address is not correct.
2. The specified RAM space isn't initialized properly.

**Q. Flash ID doesn't match. Please make sure that the right flash device is selected.**

**A.** This means that the read FLASH ID is different from the expected one. Possible reasons are

1. The configuration is not correct, which makes H-FLASHER can't read or read the wrong FLASH ID.
2. The selected FLASH device is not correct.
3. The FLASH ID is modified by manufacturer. In this case, please enable SKIP ID CHECK in H-FLASHER.

**Q. Application isn't successfully downloaded into RAM during debugging.**

**A.** Normally, this is because the RAM isn't initialized, which make the program can't be downloaded successfully.

**Q. Application isn't successfully downloaded into FLASH during debugging.**

**A.** This problem happens when user wants to debug the application in FLASH. To solve it, please

1. Enable Auto Flash Download in H-JTAG SERVER (H-JTAG SERVER -> Flasher -> Auto Download).
2. Start H-FLASHER and configure accordingly.



---

## Appendix B: List of Chips Supported by H-JTAG

H-JTAG supports all the MCUs based on CORTEX-M0, CORTEX-M3, ARM7, ARM9, ARM11 and the XSCALE series from INTEL/MARVELL. In addition, H-JTAG supports the programming of large amount of FLASH devices, which include ON-CHIP FLASH, NOR FLASH, NAND FLASH and SIP FLASH. We will upgrade our software frequently to keep it up to date with new MCUs and FLASH chips. The following list show part of the MCUs and FLASH chips supported by H-JTAG. If you can't find your chip in our lists, please feel free to contact us and we will be pleased to provide supports.

### AMD

#### FLASH DEVICES

AM29DL640G AM29F002B AM29F002T AM29F004BB AM29F004BT AM29F010 AM29F010B  
AM29F016 AM29F016B AM29F016D AM29F017D AM29F032 AM29F032B AM29F040  
AM29F040B AM29F080 AM29F080B AM29F100B AM29F100T AM29F160DB AM29F160DT  
AM29F200BB AM29F200BT AM29F400BB AM29F400BT AM29F800BB AM29F800BT  
AM29LV001BB AM29LV001BT AM29LV002B AM29LV002BB AM29LV002BT AM29LV002T  
AM29LV004B AM29LV004BB AM29LV004BT AM29LV004T AM29LV008B AM29LV008BB  
AM29LV008BT AM29LV008T AM29LV010B AM29LV017B AM29LV017D AM29LV017M  
AM29LV033C AM29LV033MU AM29LV040B AM29LV065D AM29LV065M AM29LV081  
AM29LV081B AM29LV116BB AM29LV116BT AM29LV116DB AM29LV116DT AM29LV116MB  
AM29LV116MT AM29LV128MH AM29LV128ML AM29LV160BB AM29LV160BT AM29LV160DB  
AM29LV160DT AM29LV160MB AM29LV160MT AM29LV200B AM29LV200T AM29LV256MH  
AM29LV256ML AM29LV320DB AM29LV320DT AM29LV320MB AM29LV320MH AM29LV320ML  
AM29LV320MT AM29LV400B AM29LV400T AM29LV640D AM29LV640MB AM29LV640MH  
AM29LV640ML AM29LV640MT AM29LV640MU AM29LV641D AM29LV641MH AM29LV641ML  
AM29LV652D AM29LV800B AM29LV800BB AM29LV800BT AM29LV800DB AM29LV800DT  
AM29LV800T

### ANALOG DEVICE

#### ARM MCUs

ADuC7019 ADuC7020 ADuC7021 ADuC7022 ADuC7023 ADuC7024 ADuC7025 ADuC7026  
ADuC7027 ADuC7028 ADuC7029 ADuC7060 ADuC7061

### AMIC

#### FLASH DEVICES

A29L160AT A29L160AU A29L160T A29L160U A29L320AT A29L320AU A29L320T A29L320U  
A29L400AT A29L400AU A29L400T A29L400U A29

### ATMEL

#### ARM MCUs

AT91SAM7A3 AT91SAM7S128 AT91SAM7S16 AT91SAM7S161 AT91SAM7S256 AT91SAM7S32  
AT91SAM7S321 AT91SAM7S512 AT91SAM7S64 AT91SAM7SE256 AT91SAM7SE32  
AT91SAM7SE512 AT91SAM7X128 AT91SAM7X256 AT91SAM7X512 AT91SAM7XC128

---

AT91SAM7XC256 AT91SAM7XC512

AT91SAM9XE128 AT91SAM9XE256 AT91SAM9XE512

AT91SAM9200 AT91SAM9260 AT91SAM9261 AT91SAM9263 AT91SAM9G45

ATSAM3S1A ATSAM3S1B ATSAM3S1C ATSAM3S2A ATSAM3S2B ATSAM3S2C ATSAM3S4A

ATSAM3S4B ATSAM3S4C ATSAM3U1C ATSAM3U1E ATSAM3U2C ATSAM3U2E ATSAM3U4C

ATSAM3U4E

#### **FLASH DEVICES**

AT49BV040B AT49BV160 AT49BV1604 AT49BV1604A AT49BV1604AT AT49BV1604T

AT49BV160T AT49BV161 AT49BV1614 AT49BV1614A AT49BV1614AT AT49BV1614T

AT49BV161T AT49BV162A AT49BV162AT AT49BV163A AT49BV163AT AT49BV163D

AT49BV163DT AT49BV322A AT49BV322AT AT49BV322D AT49BV322DT AT49BV6416

AT49BV6416T AT49BV642D AT49BV642DT AT49BV802A AT49BV802AT AT49F1024A

AT49LV160 AT49LV161 AT49LV1614A AT49LV1614AT AT49LV161T AT49SV322D AT49SV322DT

#### **CIRRUS LOGIC**

##### **ARM MCUs**

EP7309 EP7311 EP7312 EP9301 EP9302 EP9307 EP9312 EP9315

#### **EON**

##### **FLASH DEVICES**

EN29F010 EN29F512 EN29LV010 EN29LV040A EN29LV160AB EN29LV160AT EN29LV160BB

EN29LV160BT EN29LV320AB EN29LV320AT EN29LV320BB EN29LV320BT EN29LV400AB

EN29LV400AT EN29LV512 EN29LV640B EN29LV640H EN29LV640L EN29LV640T

EN29LV800BB EN29LV800BT EN29LV800CB EN29LV800CT EN29SL160B EN29SL160T

EN29SL400B EN29SL400T EN29SL800B EN29SL800T

#### **ESI**

##### **FLASH DEVICES**

ES29LV160XB ES29LV160XT ES29LV320XB ES29LV320XT ES29LV400XB ES29LV400XT

ES29LV640XB ES29LV640XT ES29LV800XB ES29LV800XT

#### **FREESCALE**

##### **ARM MCUs**

MAC7101 MAC7106 MAC7111 MAC7112 MAC7116 MAC7121 MAC7122 MAC7126 MAC7131

MAC7136 MAC7141 MAC7142

I.MX21 I.MX23 I.MX25 I.MX27 I.MX35

#### **FUJITSU**

##### **FLASH DEVICES**

MBM29DL640E MBM29LV160BE MBM29LV160TE MBM29LV400BC MBM29LV400TC

MBM29LV800BE MBM29LV800TE

---

 **HISILICON**

**ARM MCUs**

HI3510 HI3511 HI3512 HI3515 HI3516 HI3520

 **HYNIX**

**FLASH DEVICES**

HY29F040 HY29F040A HY29LV160B HY29LV160T HY29LV320B HY29LV320T HY29LV400B  
HY29LV400T HY29LV800B HY29LV800T  
HY27US(08/16)121A HY27SS(08/16)121A H8BCS0SI0MBR H8ACS0EH0ACR

 **INTEL**

**ARM MCUs**

PXA210 PXA250 PXA255 PXA260 PXA270 PXA300 PXA310 PXA320 IXP4XX IXP2XXX

**FLASH DEVICES**

28F004B3B 28F004B3T 28F008B3B 28F008B3T 28F016B3B 28F016B3T 28F128J3 28F128K18  
28F128K3 28F128L18B 28F128L18T 28F128L30B 28F128L30T 28F128P30B 28F128P30T  
28F128P33B 28F128P33T 28F128W18B 28F128W18T 28F128W30B 28F128W30T 28F160B3B  
28F160B3T 28F160C3B 28F160C3T 28F256J3 28F256K18 28F256K3 28F256L18B 28F256L18T  
28F256L30B 28F256L30T 28F256P30B 28F256P30T 28F256P33B 28F256P33T 28F320B3B  
28F320B3T 28F320C3B 28F320C3T 28F320J3 28F320W18B 28F320W18T 28F320W30B  
28F320W30T 28F400B3B 28F400B3T 28F640B3B 28F640B3T 28F640C3B 28F640C3T 28F640J3  
28F640K18 28F640K3 28F640L18B 28F640L18T 28F640L30B 28F640L30T 28F640P30B  
28F640P30T 28F640P33B 28F640P33T 28F640W18B 28F640W18T 28F640W30B 28F640W30T  
28F800B3B 28F800B3T 28F800C3B 28F800C3T

 **LUMINARY**

**ARM MCUs**

LM3S101 LM3S102 LM3S1110 LM3S1133 LM3S1138 LM3S1150 LM3S1162 LM3S1165 LM3S1332  
LM3S1435 LM3S1439 LM3S1512 LM3S1538 LM3S1601 LM3S1607 LM3S1608 LM3S1620  
LM3S1625 LM3S1626 LM3S1627 LM3S1635 LM3S1637 LM3S1751 LM3S1776 LM3S1850  
LM3S1911 LM3S1918 LM3S1937 LM3S1958 LM3S1960 LM3S1968 LM3S2016 LM3S2110  
LM3S2139 LM3S2276 LM3S2410 LM3S2412 LM3S2432 LM3S2533 LM3S2601 LM3S2608  
LM3S2616 LM3S2620 LM3S2637 LM3S2651 LM3S2671 LM3S2678 LM3S2730 LM3S2739  
LM3S2776 LM3S2911 LM3S2918 LM3S2939 LM3S2948 LM3S2950 LM3S2965 LM3S2B93  
LM3S300 LM3S301 LM3S308 LM3S310 LM3S315 LM3S316 LM3S317 LM3S328 LM3S3651  
LM3S3739 LM3S3748 LM3S3749 LM3S3759 LM3S3768 LM3S3N26 LM3S5632 LM3S5652  
LM3S5662 LM3S5732 LM3S5737 LM3S5739 LM3S5747 LM3S5749 LM3S5752 LM3S5757  
LM3S5762 LM3S5767 LM3S5768 LM3S5769 LM3S600 LM3S601 LM3S608 LM3S610 LM3S6100  
LM3S611 LM3S6110 LM3S612 LM3S613 LM3S615 LM3S617 LM3S618 LM3S628 LM3S6420  
LM3S6422 LM3S6432 LM3S6537 LM3S6610 LM3S6611 LM3S6618 LM3S6633 LM3S6637  
LM3S6730 LM3S6753 LM3S6911 LM3S6916 LM3S6918 LM3S6938 LM3S6950 LM3S6952  
LM3S6965 LM3S800 LM3S801 LM3S808 LM3S811 LM3S812 LM3S815 LM3S817 LM3S818  
LM3S828 LM3S8530 LM3S8538 LM3S8630 LM3S8730 LM3S8733 LM3S8738 LM3S8930  
LM3S8933 LM3S8938 LM3S8962 LM3S8970 LM3S8971 LM3S9790 LM3S9792 LM3S9B90

---

LM3S9B92 LM3S9B95 LM3S9B96

 **MACRONIX**

**FLASH DEVICES**

MX29F040X MX29F200XB MX29F200XT MX29F400XB MX29F400XT MX29GL128E  
MX29GL256 MX29LV002XB MX29LV002XT MX29LV004XB MX29LV004XT MX29LV008XB  
MX29LV008XT MX29LV040X MX29LV128XB MX29LV128XT MX29LV160XB MX29LV160XT  
MX29LV161XB MX29LV161XT MX29LV320XB MX29LV320XT MX29LV321XB MX29LV321XT  
MX29LV400XB MX29LV400XT MX29LV640XB MX29LV640XT MX29LV800XB MX29LV800XT

 **MARVELL**

**ARM MCUs**

PXA210 PXA250 PXA255 PXA260 PXA270 PXA300 PXA310 PXA320 IXP4XX IXP2XXX

 **MICREL**

**ARM MCUs**

KSZ8695 KSZ8695PX KSZ8695X KSZ9692PB KSZ9692XPB KSZ8692PB KSZ8692XPB

 **MICRON**

**FLASH 芯片**

MT29F1GxxABB MT29F1GxxABB MT29F2G08 MT29F4G08 MT29F8G08 MT29F2G16  
MT29F4G16

 **NUMONYX**

**FLASH 芯片**

NAND01G-B2B NAND02G-B2C

 **NXP**

**ARM MCUs**

LPC1101LV LPC1102 LPC1102LV LPC1104 LPC1110 LPC1111-002 LPC1111-101 LPC1111-102  
LPC1111-103 LPC1111-201 LPC1111-202 LPC1111-203 LPC1112-101 LPC1112-102 LPC1112-103  
LPC1112-201 LPC1112-202 LPC1112-203 LPC1112LV-003 LPC1112LV-103 LPC1113-201  
LPC1113-202 LPC1113-203 LPC1113-301 LPC1113-302 LPC1113-303 LPC1114-102 LPC1114-201  
LPC1114-202 LPC1114-203 LPC1114-301 LPC1114-302 LPC1114-303 LPC1114-323 LPC1114-333  
LPC1114LV-103 LPC1114LV-303 LPC1115-303 LPC11A02 LPC11A04 LPC11A11-001  
LPC11A12-101 LPC11A13-201 LPC11A14-301 LPC11C12-301 LPC11C14-301 LPC11C22-301  
LPC11C24-301 LPC11D14-302 LPC11E11-101 LPC11E12-201 LPC11E13-301 LPC11E14-401  
LPC11E36-501 LPC11E37-501 LPC11U12-201 LPC11U13-201 LPC11U14-201 LPC11U23-301  
LPC11U24-301 LPC11U24-401 LPC11U34-311 LPC11U34-421 LPC11U35-401 LPC11U35-501  
LPC11U36-401 LPC11U37-401 LPC11U37-501 LPC1224-101 LPC1224-121 LPC1225-301  
LPC1225-321 LPC1226-301 LPC1227-301 LPC12D27-301 LPC1311 LPC1313 LPC1315 LPC1316  
LPC1317 LPC1342 LPC1343 LPC1345 LPC1346 LPC1347 LPC1751 LPC1752 LPC1754 LPC1756  
LPC1758 LPC1759 LPC1763 LPC1764 LPC1765 LPC1766 LPC1767 LPC1768 LPC1769 LPC1773  
LPC1774 LPC1776 LPC1777 LPC1778 LPC1785 LPC1786 LPC1787 LPC1788 LPC2101 LPC2102

---

LPC2103 LPC2104 LPC2105 LPC2106 LPC2109 LPC2114 LPC2119 LPC2124 LPC2129 LPC2131  
LPC2132 LPC2134 LPC2136 LPC2138 LPC2141 LPC2142 LPC2144 LPC2146 LPC2148 LPC2194  
LPC2212 LPC2214 LPC2292 LPC2294 LPC2361 LPC2361-IRC LPC2362 LPC2362-IRC LPC2364  
LPC2364-IRC LPC2365 LPC2365-IRC LPC2366 LPC2366-IRC LPC2367 LPC2367-IRC LPC2368  
LPC2368-IRC LPC2377 LPC2377-IRC LPC2378 LPC2378-IRC LPC2387 LPC2387-IRC LPC2388  
LPC2388-IRC LPC2458 LPC2458-IRC LPC2468 LPC2468-IRC LPC2478 LPC2478-IRC  
LPC3250

 **SAMSUNG**

**ARM MCUs**

S3C44B0 SEC4510 S3C24A0 S3C2410 S3C2416 S3C2440 S3C2442 S3C2443 S3C2450 S3C2500  
S3C6400 S3C6410 S5L2010

**FLASH DEVICES**

K5L3316CAM K8A2815EBB K8A2815ETB K8A3215EBE K8A3215ETE K8A5615EBA  
K8A5615ETA K8A6415EBB K8A6415ETB K8C1215EBM K8C1215ETM K8C1315EBM  
K8C1315ETM K8C5615EBM K8C5615ETM K8C5715EBM K8C5715ETM K8D1716UBC  
K8D1716UTC K8D3216UBC K8D3216UTC K8D6316UBM K8D6316UTM K8F1215EBM  
K8F1215ETM K8F1315EBM K8F1315ETM K8F5615EBM K8F5615ETM K8F5715EBM  
K8F5715ETM K8P2915UQB K8P5615UQA K8S2815EBB K8S2815ETB K8S3215EBE  
K8S3215ETD K8S3215ETE K8S5615EBA K8S5615ETA K8S6415EBB K8S6415ETB  
K9F5608 K9F1208 K9F1G08 K9F2G08 K9F4G08 K9F8G08 K9G8G08 K9K8G08

 **SPANSION**

**FLASH DEVICES**

S29AL008D-M01 S29AL008D-M02 S29AL008D-MR1 S29AL008D-MR2 S29AL008J-M01  
S29AL008J-M02 S29AL008J-M03 S29AL008J-M04 S29AL008J-MR1 S29AL008J-MR2  
S29AL016D-M01 S29AL016D-M02 S29AL016JB S29AL016JT S29AL016MB S29AL016MT  
S29AL032D-M00 S29AL032D-M03 S29AL032D-M04 S29GL016A-M01 S29GL016A-M02  
S29GL016A-MR1 S29GL016A-MR2 S29GL01GP S29GL032A-MR1 S29GL032A-MR2  
S29GL032A-MR3 S29GL032A-MR4 S29GL032A-MW3 S29GL032A-MW4 S29GL032M-MR0  
S29GL032M-MR1 S29GL032M-MR2 S29GL032M-MR3 S29GL032M-MR4 S29GL032N-M01  
S29GL032N-M02 S29GL032N-M03 S29GL032N-M04 S29GL032N-MV1 S29GL032N-MV2  
S29GL064A-MR1 S29GL064A-MR2 S29GL064A-MR3 S29GL064A-MR4 S29GL064A-MR5  
S29GL064A-MR6 S29GL064A-MR7 S29GL064A-MR8 S29GL064A-MR9 S29GL064M-MR0  
S29GL064M-MR1 S29GL064M-MR2 S29GL064M-MR3 S29GL064M-MR4 S29GL064M-MR5  
S29GL064M-MR6 S29GL064M-MR7 S29GL064M-MR8 S29GL064M-MR9 S29GL064N-M01  
S29GL064N-M02 S29GL064N-M03 S29GL064N-M04 S29GL064N-M06 S29GL064N-M07  
S29GL064N-MV1 S29GL064N-MV2 S29GL064N-MV6 S29GL064N-MV7 S29GL128M S29GL128N  
S29GL128P S29GL256M S29GL256N S29GL256P S29GL512N S29GL512P S29JL032H-M01  
S29JL032H-M02 S29JL032H-M21 S29JL032H-M22 S29JL032H-M31 S29JL032H-M32  
S29JL032H-M41 S29JL032H-M42 S29JL064H S29NS064N S29NS128N S29NS128P S29NS256N  
S29NS256P S29NS512P S29PL032J S29PL064J S29PL127J S29PL127N S29PL129N S29PL256N  
S29WS128N S29WS128P S29WS256N S29WS256P S29WS512P

---

## SST

### FLASH DEVICES

SST29SF020 SST29SF040 SST29VF020 SST29VF040 SST34HF1681 SST36VF1601C  
SST36VF1601E SST36VF1601G SST36VF1602C SST36VF1602E SST36VF1602G SST36VF3203  
SST36VF3204 SST39LF010 SST39LF020 SST39LF040 SST39LF080 SST39LF160 SST39LF200A  
SST39LF400A SST39LF512 SST39LF800A SST39SF010A SST39SF020A SST39SF040 SST39VF010  
SST39VF020 SST39VF040 SST39VF080 SST39VF088 SST39VF160 SST39VF1601 SST39VF1601C  
SST39VF1602 SST39VF1602C SST39VF1681 SST39VF1682 SST39VF200A SST39VF320  
SST39VF3201 SST39VF3201B SST39VF3202 SST39VF3202B SST39VF400A SST39VF512  
SST39VF6401 SST39VF6401B SST39VF6402 SST39VF6402B SST39VF800A SST39WF400A  
SST39WF800A

## ST Microelectronics

### ARM MCUs

STM32F030X4 STM32F030X6 STM32F030X8 STM32F050X4 STM32F050X6 STM32F051X4  
STM32F051X6 STM32F051X8 STM32F100X4 STM32F100X6 STM32F100X8 STM32F100XB  
STM32F100XC STM32F100XD STM32F100XE STM32F101X4 STM32F101X6 STM32F101X8  
STM32F101XB STM32F101XC STM32F101XD STM32F101XE STM32F101XF STM32F101XG  
STM32F102X4 STM32F102X6 STM32F102X8 STM32F102XB STM32F103X4 STM32F103X6  
STM32F103X8 STM32F103XB STM32F103XC STM32F103XD STM32F103XE STM32F103XF  
STM32F103XG STM32F105X8 STM32F105XB STM32F105XC STM32F107XB STM32F107XC  
STM32F205XB STM32F205XC STM32F205XE STM32F205XF STM32F205XG STM32F207XB  
STM32F207XC STM32F207XE STM32F207XF STM32F207XG STM32F215XE STM32F215XG  
STM32F217XE STM32F217XG STM32F302XB STM32F302XC STM32F303XB STM32F303XC  
STM32F313XB STM32F313XC STM32F373X8 STM32F373XB STM32F373XC STM32F383X8  
STM32F383XB STM32F383XC STM32F405XC STM32F405XE STM32F405XF STM32F405XG  
STM32F407XC STM32F407XE STM32F407XF STM32F407XG STM32F415XC STM32F415XE  
STM32F415XF STM32F415XG STM32F417XC STM32F417XE STM32F417XF STM32F417XG  
STM32F427XG STM32F427XI STM32F429XG STM32F429XI STM32F437XG STM32F437XI  
STM32F439XG STM32F439XI STM32L100X6 STM32L100X8 STM32L100XB STM32L100XC  
STM32L151X6 STM32L151X8 STM32L151XB STM32L151XC STM32L151XD STM32L152X6  
STM32L152X8 STM32L152XB STM32L152XC STM32L152XD STM32L162XD  
STR71XFX0 STR71XFX1 STR71XFX2 STR73XFX0 STR73XFX1 STR73XFX2 STR75XFX0  
STR75XFX1 STR75XFX2  
STR91XFAX32 STR91XFAX42 STR91XFAX44 STR91XFAX46 STR91XFAX47 STR91XFX32  
STR91XFX42 STR91XFX44 STR91XFX46 STR91XFX47

### FLASH DEVICES

M29DW128F M29DW128G M29DW323DB M29DW323DT M29DW324DB M29DW324DT  
M29DW640F M29DW641F M29W008AB M29W008AT M29W064FB M29W064FT M29W128FH  
M29W128FL M29W128GH M29W128GL M29W160EB M29W160ET M29W160FB M29W160FT  
M29W320DB M29W320DT M29W320EB M29W320ET M29W320FB M29W320FT M29W400DB  
M29W400DT M29W640FB M29W640FT M29W640GB M29W640GH M29W640GL M29W640GT  
M29W800DB M29W800DT M29W800FB M29W800FT  
NAND01G-B2B NAND02G-B2C

---

 **TOSHIBA**

**FLASH DEVICES**

TC58FVM7BDD TC58FVM7TDD TV00570002CDGB TV00570003CDGB

 **WINBOND/NUVOTON**

**ARM MCUs**

NUC100LC1BN NUC100LD1BN NUC100LD2BN NUC100RC1BN NUC100RD1BN  
NUC100RD2BN NUC100LD3AN NUC100LE3AN NUC100RD3AN NUC100RE3AN  
NUC100VD2AN NUC100VD3AN NUC100VE3AN  
NUC120LC1BN NUC120LD1BN NUC120LD2BN NUC120RC1BN NUC120RD1BN  
NUC120RD2BN NUC120LD3AN NUC120LE3AN NUC120RD3AN NUC120RE3AN  
NUC120VD2AN NUC120VD3AN NUC120VE3AN  
NUC122ZC1AN NUC122ZD2AN NUC122LC1AN NUC1LD2AN NUC122RC1AN NUC122RD2AN  
NUC130LC1BN NUC130LD2BN NUC130RC1BN NUC130RD2BN NUC130LD3AN  
NUC130LE3AN NUC130RD3AN NUC130RE3AN NUC130VD2AN NUC130VD3AN  
NUC130VE3AN  
NUC140LC1BN NUC140LD2BN NUC140RC1BN NUC140RD2BN NUC140LD3AN  
NUC140LE3AN NUC140RD3AN NUC140RE3AN NUC140VD2AN NUC140VD3AN  
NUC140VE3AN  
NUC501A NUC501B NUC710A NUC740A NUC745A NUC910A NUC920A NUC945A NUC950A  
NUC960A

**FLASH DEVICES**

W19B320AB W19B320AT W19B320BB W19B320BT W39F010 W39L010 W39L020 W39L040  
W39L040A W39L512 W39V040A W39V040B W39V040C W39V040FA W39V040FB W39V040FC  
W39V080A W39V080FA W49F002U W49F020

 **NAND FLASH (MCU + FLASH COMBINATION)**

Hi3515+HY27UG088G Hi3515+K9F1G08 Hi3515+K9F2G08 Hi3515+K9F4G08 i.MX27+K9F1208  
i.MX27+K9F1G08 i.MX27+K9F2G08 i.MX27+MT29F2G08 LPC2478+K9F1G08  
LPC2478+K9F5608 LPC32X0+K9F1G08-MLC LPC32X0+K9F1G08-SLC  
LPC32X0+K9F2G08+MLC LPC32X0+K9F2G08+SLC PXA300+H8BCS0SI0MBR  
PXA300+K9F1G08 PXA300+K9F2G08 PXA300+K9F4G08 PXA300+K9K8G08  
PXA300+MT29F1G16 PXA300+NAND01GR3B2B PXA300+NAND01GR4B2B  
PXA300+NAND01GW3B2B PXA300+NAND01GW4B2B PXA300+NAND02GR3B2C  
PXA300+NAND02GR4B2C PXA300+NAND02GW3B2C PXA300+NAND02GW4B2C  
PXA300+TYA000B000ALKF40 PXA30X+H8BCS0SI0MBR PXA30X+K9F1G08  
PXA30X+K9F2G08 PXA30X+K9F4G08 PXA30X+K9K8G08 PXA30X+MT29F1G16  
PXA30X+NAND01GR3B2B PXA30X+NAND01GR4B2B PXA30X+NAND01GW3B2B  
PXA30X+NAND01GW4B2B PXA30X+NAND02GR3B2C PXA30X+NAND02GR4B2C  
PXA30X+NAND02GW3B2C PXA30X+NAND02GW4B2C PXA30X+TYA000B000ALKF40  
PXA31X+H8BCS0SI0MBR PXA31X+K9F1G08 PXA31X+K9F2G08 PXA31X+K9F4G08  
PXA31X+K9K8G08 PXA31X+MT29F1G16 PXA31X+NAND01GR3B2B  
PXA31X+NAND01GR4B2B PXA31X+NAND01GW3B2B PXA31X+NAND01GW4B2B  
PXA31X+NAND02GR3B2C PXA31X+NAND02GR4B2C PXA31X+NAND02GW3B2C

---

PXA31X+NAND02GW4B2C PXA31X+TYA000B000ALKF40 PXA32X+H8BCS0SI0MBR  
PXA32X+K9F1G08 PXA32X+K9F2G08 PXA32X+K9F4G08 PXA32X+K9K8G08  
PXA32X+MT29F1G16 PXA32X+NAND01GR3B2B PXA32X+NAND01GR4B2B  
PXA32X+NAND01GW3B2B PXA32X+NAND01GW4B2B PXA32X+NAND02GR3B2C  
PXA32X+NAND02GR4B2C PXA32X+NAND02GW3B2C PXA32X+NAND02GW4B2C  
PXA32X+TYA000B000ALKF40 S3C2410+K9F1208 S3C2410+K9F1G08 S3C2410+K9F2G08  
S3C2410+K9F4G08 S3C2410+K9F5608 S3C2410+K9G8G08 S3C2410+K9K8G08  
S3C2416+K9F1208 S3C2416+K9F1G08 S3C2416+K9F2G08 S3C2416+K9F4G08  
S3C2416+K9F5608 S3C2416+K9G8G08 S3C2416+K9K8G08 S3C2440+H8ACS0EH0ACR  
S3C2440+HY27US08121A S3C2440+K9F1208 S3C2440+K9F1G08 S3C2440+K9F2G08  
S3C2440+K9F4G08 S3C2440+K9F5608 S3C2440+K9G8G08 S3C2440+K9K8G08  
S3C2443+K9F1208 S3C2443+K9F1G08 S3C2443+K9F2G08 S3C2443+K9F4G08  
S3C2443+K9F5608 S3C2443+K9G8G08 S3C2443+K9K8G08 S3C2450+K9F1208  
S3C2450+K9F1G08 S3C2450+K9F2G08 S3C2450+K9F4G08 S3C2450+K9F5608  
S3C2450+K9G8G08 S3C2450+K9K8G08 S3C6410+K9F1208 S3C6410+K9F1G08  
S3C6410+K9F2G08 S3C6410+K9F4G08 S3C6410+K9F5608 S3C6410+K9G8G08  
S3C6410+K9K8G08