User Manual for

USBJTAG NT

September 2010
(0.42)

Preface

Disclaimer

The information in this document is subject to change without notice. The manufacture makes no representations or warranties with respect to contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. The manufacture reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of the manufacturer to notify any person of such revision or changes.
Use USBJTAG NT at your own risk. Nothing is implied outside this document. The manufacture is not responsive for any damage using the USBJTAG NTsoftware and hardware.

USB JTAG NT is not freeware. It is not based on any freeware. It must only be used on products purchased on www.usbjtag.com or its dealer.
If you are not sure your product is from this site, please send email to usbbdm@usbjtag.com
If you use clone hardware do not use this software. Do not use the software/hardware if you do not agree the terms.

Terms

In this document, **target** or **target board** refers to any device that has MIPS, ST20 (ARM in the future) core CPU. Eg.Broadcom, LSI, etc. USB JTAG and USB EJTAG are both used for USB JTAG device. EJTAG is the JTAG name for MIPS core. DCU is the JTAG name for ST20 core.
For USBBDM NT hardware, **target** or **target board** refers to any device that has Motorola CPU like MC6833x

Only power on the JTAG (or BDM) and the target **AFTER** the JTAG (or BDM) and the target are connected. Connecting the JTAG (or BDM) and the target with either of their power on could burn either the JTAG (or BDM) or the target.

Warranty

For normal usage the devices can last very long. But since this is testing device and you are making contact to another powered device, there is a risk that hardware might get burnt out. The devices offer three months free repair. After three months, a small fee needs to repair the USB JTAG NT or USB BDM NT. Warranty voided if covered is removed.

TABLE OF CONTENTS

Chapter 1 Basic concept

1.1 What is JTAG

Here is the quote from <<MIPS EJTAG Specification>>

EJTAG is a hardware/software subsystem that provides comprehensive debugging and performance tuning capabilities to MIPS® microprocessors and to system-on-a-chip components having MIPS processor cores. It exploits the infrastructure provided by the IEEE 1149.1 JTAG Test Access Port (TAP) standard to provide an external interface, and extends the MIPS instruction set and privileged resource architectures to provide a standard software architecture for integrated system debugging.

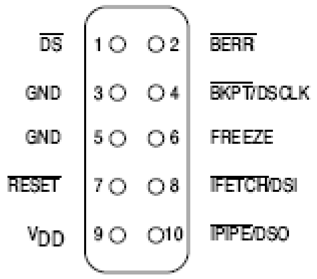
USB JTAG now support ST20 core from STMicroelectronics.
Arm7, Arm 9 support is on the plan. SPI programming are also supported.

USB JTAG NT software runs on Microsoft Windows system. The tested OS are Windows XP, Windows 2000, Windows Vista 32 bit, Vista 64 bit. Windows 98 should be supported but have not been tested. In Aug 2010, all code are ported to new platform and soon Linux and Mac will be supported. This document will also be updated.

USBJTAG NT is the second generation of USBJTAG developed by www.usbjtag.com. It uses USB 2.0 technology and it supports a wider range of CPU and provides even faster speed than earlier version of USBJTAG which runs on USB 1.1.

1.2 What is BDM

BDM stands for Background Debugging Mode. Here is the pin layout of 10 pin BDM.



For detailed BDM information, please refer to chapter 5 of MC68331Ref.pdf.

USBJBDM NT is the second generation of USBBDM developed by www.usbjtag.com. It uses USB 2.0 technology and it supports a wider range of CPU and provides even faster speed than earlier version of USBJTAG which runs on USB 1.1.

USB JTAG NT 0.40 now also support old USB BDM. There is no activation needed for old USB BDM.

1.3 What can be done with USBJTAG NT

USB JTAG provides an affordable yet powerful enough tool to have most functions that a professional JTAG tool can provide. It allows you to

1. Read memory from target board.
2. Write memory to target board.
3. Program flash using target CPU and memory.
4. Do simple debugging. Break and single step the target in its execution. (MIPS32 only for now)

In the future, it is possible to attach to GNU debugger. More debug functions will be added in the future.

1.4 What can be done with USBBDM NT

USB BDM provides an affordable yet powerful enough tool to have most functions that a professional BDM tool can provide. It allow you to

1. View the registers of the target CPU.
2. Single step the target CPU.
3. Get memory from target board.
4. Set memory to target board.
5. Program flash using target CPU and memory.

Chapter 2 Install USBJTAG NT (USBBDM NT)

2.1 Setup software

First time installation:

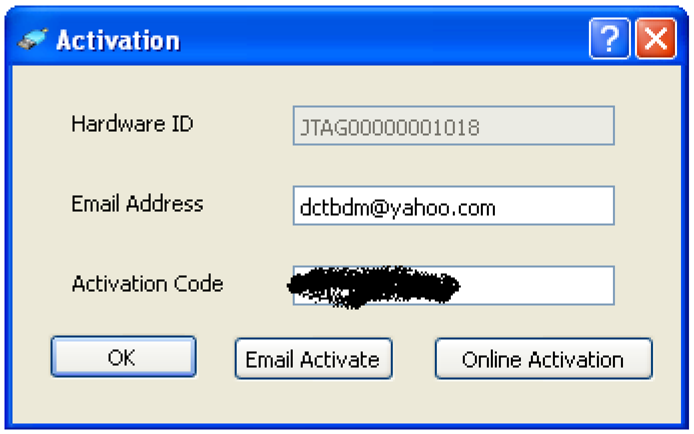
Download the setup.rar file from www.usbjtag.com, extract the setup.exe file and run it.
Plug in the USB, when it ask for driver, select automatic and click OK. (For Vista 64 bit download the 64 bit driver from www.usbjtag.com and browse and select the .inf file).
If you are asked for missing files download from Microsoft site and install.
http://www.microsoft.com/downloads/en/details.aspx?FamilyID=200b2fd9-ae1a-4a14-984d-389c36f85647&DisplayLang=en
If it still not working a small trick is to download and install yahoo messenger (You can uninstall latter but the needed dlls will allow you to run USB JTAG NT software.
http://messenger.yahoo.com/download/

Update:

USB JTAG NT software updates quite frequently. If there is only minor changes to the software you do not need to download full setup.rar. Simply download the software an unzip to the directory where the software was installed. Default for Windows is C:\Program Files\usbjtag\usbjtagnt

2.2 Activation

Your activation code is a unique number working with your email address (case sensitive and your hardware ID.



Hardware ID: JTAG00000001018

Email Address: dctbdm@yahoo.com

Activation Code: [redacted]

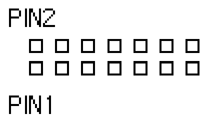
1. When you do not have the code, enter email address and click "Email Activation" and it will popup your email software and send out the email request.
2. If you get the code then enter the code manually (Make sure the email is right, email is case sensitive). **Click OK when done.**
3. If you have activate before and you lost your activation code, you can enter email address and click "Online Activation". It will try to get the code from the online database.
4. The software does not limit the number of computer you run on. You can install as many computers as you like.

When email to activation@usbjtag.com to request activation, please specify the hardware ID in the email.

2.3 Install hardware

USBJTAG NT

The EJTAG connector on the board matches the standard Broadcom 14 pin EJTAG connector.



Here are the definitions of EJTAG.

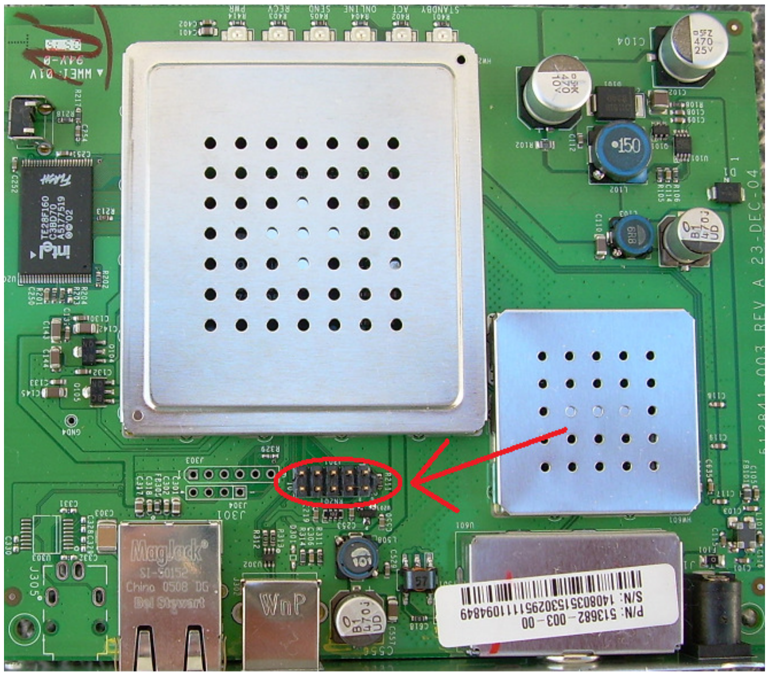
Pin1 – TRST
Pin3 – TDI
Pin5 - TDO
Pin7 – TMS
Pin9 – TCK
Pin11 – RESET
Pin13 – DEBUG (Not used)
Pin 2, 4,6,8,10 Ground. Pin 14 NC.

On back of the USB JTAG NT module, printed lableis used to indicate pin 1. The preconnected ribbon cable has a red pin indicate pin 1,too.

If your target does not have the same pinout, you need to make a connector to match your target.

Some board might not have JTAG connector and you need to solder a connector for it.

Here is an example for sb5101 cable modem with a JTAG connector soldered on.



1. Power off the target board.
2. Plug the JTAG connector to the JTAG connector on target board
   <u>Improper connecting to the target will not work and might kill your target or USB JTAG.</u>
3. Connect the USB JTAG into one of the PC's USB slot.
4. Power on the target board.

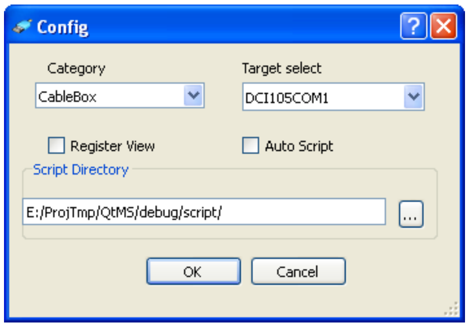
USBBDM NT

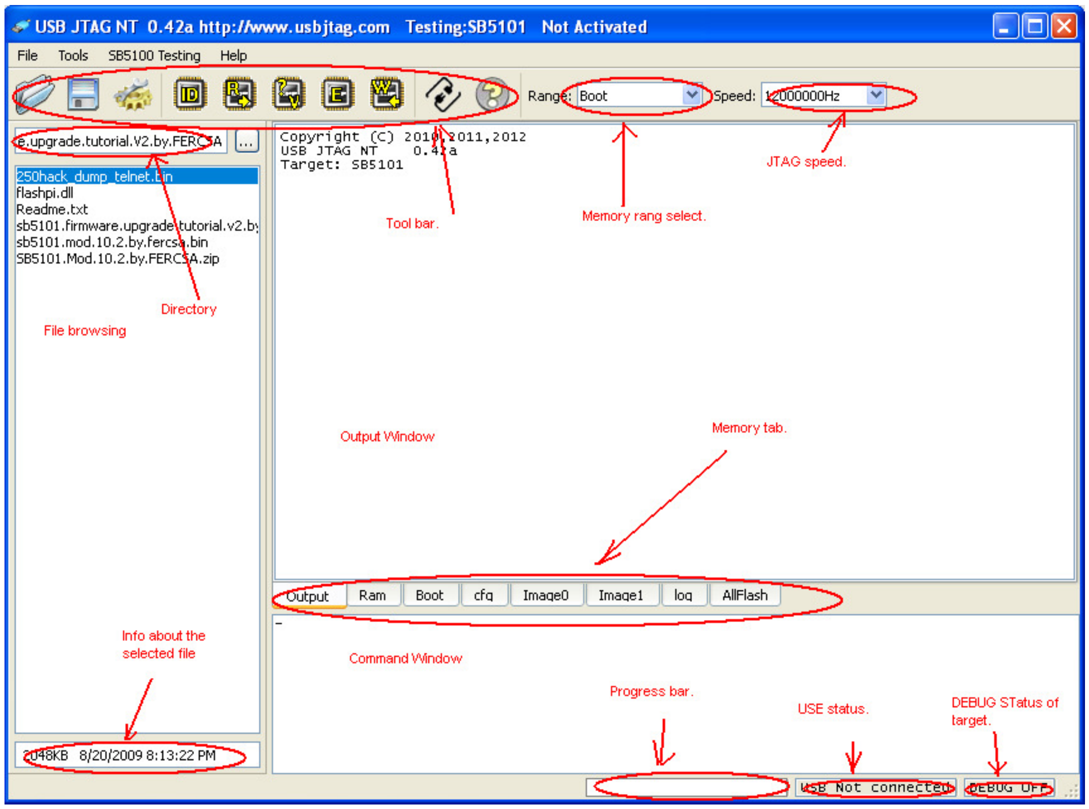
1. Power off the target board.
2. Plug the BDM connector to the BDM connector on target board. Make sure pin 1 matches the pin 1 on the target BDM connector. Reversing the BDM will cause the BDM to not function properly. If this happened, power off the target and plug back in.
3. Connect the USB BDM one of the PC's USB slot.
4. Power on the target board.

Chapter 3. Using USBJTAG NT software

3.1 Software layout.

There are two types of layout, Enable register view when configuring. Register view provides the information of the registers during debugging. (Register view only support EJTAG32 , EJTAG64 and BDM)





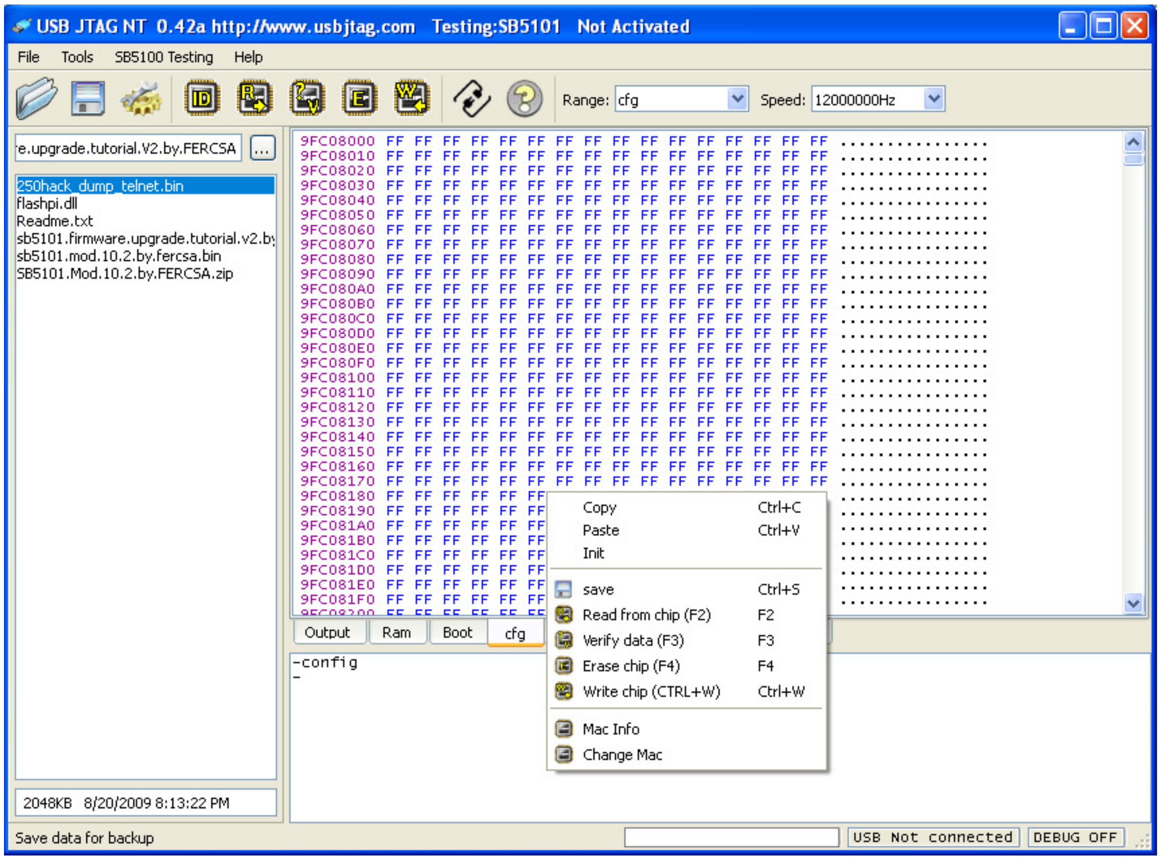
```
Copyright (C) 2010,2011,2012
USB JTAG NT    0.42a
Target: SB5101
```

Normal view.

1. Title bar show USBJTAG NT and the test module used. If you plug USB JTAG NT in you must see it is activated. Otherwise it will not work.
2. Command window (bottom window) accept your commands.
3. Output Window (middle window) prints the result of the execution of commands.
4. Memory tabs can be configured in target xml file in config directory.
5. Connect status (in the bottom bar) indicates the USB JTAG to PC connect state, during the running of the software, you can unplug/plug the USB connector to PC. When JTAG is not connected, no command can be send to the target CPU.
6. DEBUG state should be monitored carefully. During memory access, DEBUG should be off. DEBUG ON is only used for programming flash. (It is very

important to make sure that any watchdog is disabled so flash can be programmed properly. If you are not sure about this, DO NOT PORGRAM the flash).

7. TRAP ON is used to tell if we can access target memory and program the flash.
8. Progress bar is used for some lengthy operation like read/write memory or program flash.
9. File browser can be used to drag the file to memory tab.


USB JTAG NT 0.42a http://www.usbjtag.com Testing:SB5101 Not Activated

File Tools SB5100 Testing Help

Range: cfg Speed: 12000000Hz

e.upgrade.tutorial.V2.by.FERCSA

250hack_dump_telnet.bin
flashpi.dll
Readme.txt
sb5101.firmware.upgrade.tutorial.v2.by
sb5101.mod.10.2.by.fercsa.bin
SB5101.Mod.10.2.by.FERCSA.zip

2048KB 8/20/2009 8:13:22 PM

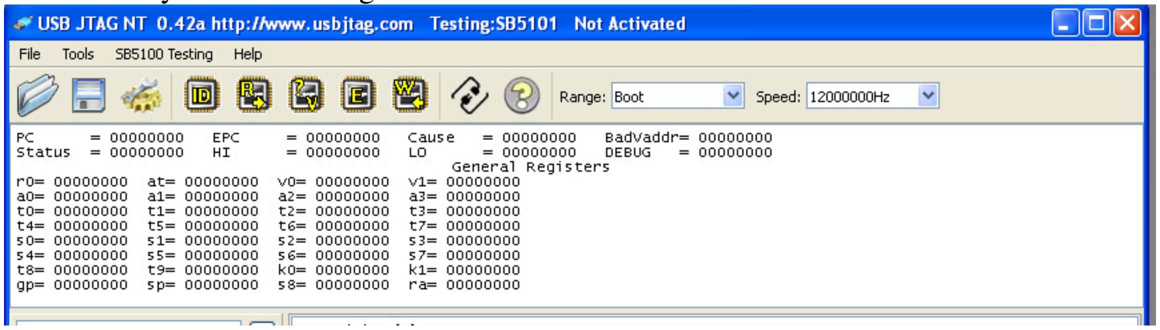
```
9FC08000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC080A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC080B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC080C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC080D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC080E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC080F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08100 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08110 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08120 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08130 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08140 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08150 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08160 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08170 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
9FC08180 FF FF FF FF FF FF FF                            ................
9FC08190 FF FF FF FF FF FF FF                            ................
9FC081A0 FF FF FF FF FF FF FF                            ................
9FC081B0 FF FF FF FF FF FF FF                            ................
9FC081C0 FF FF FF FF FF FF FF                            ................
9FC081D0 FF FF FF FF FF FF FF                            ................
9FC081E0 FF FF FF FF FF FF FF                            ................
9FC081F0 FF FF FF FF FF FF FF                            ................
```

```
-config
-
```

Save data for backup

USB Not connected DEBUG OFF

Memory view.
You can view the memory and also execute commands with the right click the memory tab to trigger pull down menu.
To read file you can use drag from the left file browser or external file browser.


USB JTAG NT 0.42a http://www.usbjtag.com Testing:SB5101 Not Activated

File Tools SB5100 Testing Help

Range: Boot Speed: 12000000Hz

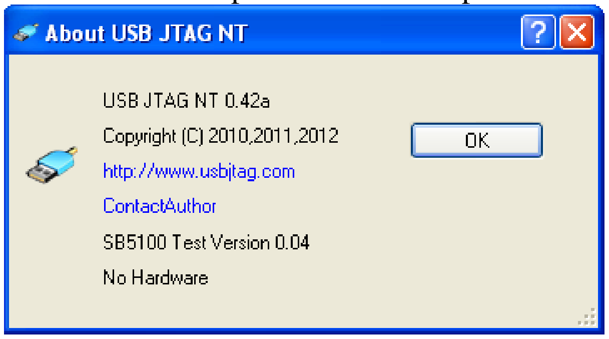
```
PC      = 00000000  EPC     = 00000000  Cause    = 00000000  BadVaddr= 00000000
Status  = 00000000  HI      = 00000000  LO       = 00000000  DEBUG   = 00000000
                                    General Registers
r0= 00000000  at= 00000000  v0= 00000000  v1= 00000000
a0= 00000000  a1= 00000000  a2= 00000000  a3= 00000000
t0= 00000000  t1= 00000000  t2= 00000000  t3= 00000000
t4= 00000000  t5= 00000000  t6= 00000000  t7= 00000000
s0= 00000000  s1= 00000000  s2= 00000000  s3= 00000000
s4= 00000000  s5= 00000000  s6= 00000000  s7= 00000000
t8= 00000000  t9= 00000000  k0= 00000000  k1= 00000000
gp= 00000000  sp= 00000000  s8= 00000000  ra= 00000000
```

Register view, display MIPS (Or MC6833X) register values when in debug mode.

Note: For some board when power on and you see DEBUG ON, type "r" command. If PC is 0xBFC00000 then type "g" or press F5 to run the target.

Check the version of software

Help->about, version of the software printed out. Example



USB JTAG NT 0.42a
Copyright (C) 2010,2011,2012
http://www.usbjtag.com
ContactAuthor
SB5100 Test Version 0.04
No Hardware

USBJTAG NT is the main application. Test module is the plug in DLL. If present, a new pull down menu exists after the “Tools” menu. ***<u>The plug in DLL must exist in the same folder as the main USB JTAG application.</u>***

Configure the software

Before the software can be used, configuration must be done based on your target. Xml configuration files exists under the config directory.

3.2 GUI Usage

Even though the command line interface provides more power, many of its functions can be simplified by the included GUI interface.

Range selection

Memory access needs the start address and the length of the target memory. In the tool bar there is a drop down box that contains predefined start addresses and length with sudo-names. One can also click the tabs under the middle window of the picture shown above to determine what range selection is used.

Speed selection

This is to select the JTAG (BDM) clock speed. Even though we normally need high speed to program the target. It is needed sometime (especially when target is bricked) to use slower speed. The current hardware allow you t select three JTAG (BDM) speed.

Toolbar

Going from left to right

- Open file (command line equivalent: ldram)
- Save file (command line equivalent: save)
- Tools → configuration (command line equivalent: config)
- Target id detection (command line equivalent: detect)
- Read (command line equivalent: getram)
- Verify (command line equivalent: cmpram)
- Erase flash/EEPROM (command line equivalent: erase)
- Program flash (command line equivalent: program/sprogram)
- Connection. Show how to connect the target.
- About/Copyright info

3.3 Commands

USB EJTAG NT software is command driven. In command window typing "help" will give you all the command available. Type "help" and then a command will give you details of that command. Up, Down arrow keys can be used to go through the command history in the command window.

Common commands

1. **d:** Display the address.
   Syntax: d address (in hexadecimal)
   Example: `d 9fc80000`
2. **exit:** Exit the whole application.
   Syntax: exit
3. **help:** prints command help.
   Syntax: help — This will print all the command names.
   Syntax: help (cmd) — This will print the usage of the cmd.
   Example: `help flshdct`
4. **reset:** Reset the target.
   Syntax: reset
   Note: Not all the target board has the reset pin connected to JTAG port and even if the JTAG pin is connected to the JTAG port, the reset command might not reset the target for some other reasons. If this happened a power off/on will do the same work as reset.
5. **detect:** Detect the target CPU and possible flash types. If there are memory tabs defined as flash then a flash detect command is also issued.
   Syntax: detect
   Shortcut: F1
   Example: (sb5101)

```
-detect
 IDCODE 334917F
 Broadcom BCM3349
 IMPCODE 800904
 DMA supported
 Found Address= 9fc00000 Intel 28F160C3B
```

6. **search:** Search the memory block. This is ONLY used for an unknown target and you want to find the memory map. Most important to find where the firmware starts. For most users this **command** is not used.
   Syntax: search start end step.
   Example:
```
-SEARCH 90000000 a0000000 200000
Address 90000000 data=FFFFFFFF
Address 90200000 data=FFFFFFFF
. . . . . .
Address 9E000000 data=0BF00004
Address 9E200000 data=0BF00004
Address 9E400000 data=0BF00004
Address 9E600000 data=0BF00004
Address 9E800000 data=0BF00004
Address 9EA00000 data=0BF00004
Address 9EC00000 data=0BF00004
Address 9EE00000 data=0BF00004
Address 9F000000 data=0BF00004
Address 9F200000 data=0BF00004
Address 9F400000 data=0BF00004
Address 9F600000 data=0BF00004
Address 9F800000 data=0BF00004
Address 9FA00000 data=0BF00004
Address 9FC00000 data=0BF00004
Address 9FE00000 data=0BF00004
```
   By further analyzing the firmware we can then figure out that the firmware starts at 9fc00000. For MIPS CPU they also maps as 1fc00000 or Bfc00000.
7. **flshdct:** Detect the flash type.
   Syntax: flshdct tabname
   flshdct address
   Example:
```
-flshdct boot
Found Address= 9fc00000 Intel 28F160C3B
-flshdct 9fc00000
Found Address= 9fc00000 Intel 28F160C3B
```
8. **flshset:** Set a flash type to the tab. This is used when the target is complete dead and normal rescue method cannot be used.
   Syntax: flshdct tabname value1 value2. (Value1 and value2 will be the same value otherwise flshdct is used. Manufacture ID and chip ID).
   Example:
```
-FLSHSET boot 89 8891
Found Address= 9fc00000 Intel 28F160C3B
-configshow
Test name:     SB5100
Test DLL:  SB5100.dll
IRLength:  5
Endian:    Big
Boot       Flash=Intel 28F160C3B
cfg        Flash=Intel 28F160C3B
Image0          Flash=Intel 28F160C3B
```

```
Image1          Flash=Intel 28F160C3B
log        Flash=Intel 28F160C3B
```

9. **init**. This is to initialize the target (in dll or in xml).
10. **initusb:** Initialize the USB PORT. This will trigger USB PORT to reinitialize the USB JTAG. It might take several seconds to get back "JTAG connected" state.
    Syntax: initusb
11. **getram:** Read memory from target to PC. This is lengthy operation and the progress bar will show roughly where you are. After completion of the memory read, the memory in the tabs will be updated. You can view and edit the memory in the memory tabs. Be careful when editing the memory map, since most flash firmware have complicated checksum to avoid data corruption, simply editing the firmware and programming it back might not work.
    Syntax: getram tab
    getram start length
    Example: `getram boot`
    `getram 9fc00000 200000`
12. **save:** Save the PC memory to a file. The default file extension is ".bin"
    Syntax: save tabname
    save start length
    Example: `save boot`
    `save 9fc00000 200000`
13. **ldram:** Load binary file to PC memory. This is opposite to "save" command.
    Syntax: ldram tabname (filename)
    ldram address
    Example: `ldram boot`
    `ldram 9fc00000`
14. **cmpram:** Compare the PC memory with target memory. This is very useful especially for programming flash. If you use EJTAG you cannot do cmpram right after the programming if non-DMA is used. The OK means the memory are identical between PC and the target. Otherwise the failed address will be displayed.
    Syntax: cmpram tabname
    cmpram address length
    Example: `cmpram boot`
    `cmpram 9fc00000 200000`
15. **setram:** Opposite to getram, this set the target memory from PC. *This can only be used for ram not for flash. For flash you can only use "program" or "sprogram" to alter the target memory.*
    Syntax: setram tabname
    setram address length
    Example: `setram 80000000 200`
16. **peek:** Get one word from target.
    Syntax: peek address
    Example: `peek 80000000`
17. **pokeh:** Poke two bytes.
    Syntax: pokeh address value
18. **pokeb:** Poke one byte.
    Syntax:pokeb address value

19. **peekh:** Peek two bytes.
    Syntax:peekh address
20. **peekb:** Peek one byte
    Syntax: peekb address
21. **poke:** Set one word to target.
    Syntax: poke address value
22. **flshlist:** List all the flash types that are defined in flash.def
    Syntax: flshlist
23. **config:** Start configuration dialog to configure the target and view.
24. **about:** Display the about dialog box.
    Syntax: about
25. **cls:** Clear the screen
    Syntax: cls
26. **e:** Edit data in PC memory. To update to the target ram or flash you need to use "setram" or "program" commands.
    Syntax: e address data1 data 2 ….
    Example: `-e 9fc08000 11 22 33 44`
27. **f:** Fill data in PC memory. To update the target ram or flash you need to use "setram" or "program" commands.
    Syntax: f tabname value
    f start length value
    Example:
    ```
    f bootff
     f 9fc00000 200000 ff
    ```
28. **s:** Search patterns in PC memory.
    Syntax: s tabname string
    s tabname data1 data2 …
    s start length string
    s start length data1 data2 …
    Example:
    ```
    -s image1 "SB5100"
    -s image1 40 08 80
    -s 9fd00000 100000 "SB5100"
    -s 9fd00000 100000  40 08 80
    ```
29. **configshow:** Show all the configuration.
    Syntax:configshow
    Example:
    ```
    -CONFIGSHOW
    Test name:      SB5100
    Test DLL:  SB5100.dll
    IRLength:  5
    Endian:    Big
    Boot       Flash=Intel 28F160C3B
    cfg        Flash=Intel 28F160C3B
    Image0          Flash=Intel 28F160C3B
    Image1          Flash=Intel 28F160C3B
    log        Flash=Intel 28F160C3B
    ```
30. **erase:** Erase the flash. The erase command is used with sprogram. Normal program command auto erase the flash. This command is only used when normal

program command does not work. ST20 target must use erase/sprogram to program the flash. Please note that the erase command does not have feedback while erasing. And normally erase take quite a long time. A 2M flash's erase normally will take up to 20-40 seconds. If after a long time the program does not return something has gone wrong and you need to stop the program and start again.
Syntax: erase tabname
            erase address length
Example:
```
-ERASE image0
Erase starts
Erase time  00:00:08 .021
```
31. **sprogram:** Slow program. This is slow program compared to normal program. In EJTAG this method is not used to target ram. In EJTAG when the boot is not in setup and the initialization sequence to access ram is unknown, sprogram is normally used for program a boot block. Make sure the target flash is erased.
Syntax: sprogram tabname
           sprogram start length
Example:
```
-ERASE boot
Erase starts
Erase time  00:00:00 .031
-SPROGRAM boot
Program Starts...
Program time 00:00:08 .084
-CMPRAM boot
Compair data OK
```

32. **program:** Programs the flash or eeprom. If you program the flash make sure you have executed "flshdct" or "detect" command. The right flash type must be set to the memory.
Syntax: program tabname
           program address length
Example:
```
program boot
       program 9fc00000 200000
       program eeprom
-PROGRAM image0
Erase starts...
Erase time 00:00:08 .071
Program speed 138.26 KB/s
Program time 00:00:07 .081
Program pass, if no further programming needed, power
off/on the target
```
33. **bk:** Breakpoint in the target. Normally use this with register view enabled.
Syntax: bk
Shortcut: F6
34. **r:** Read registers or set register value to the target
Syntax: r
           r register value
Example: `r r1 8000200`

35. **t:** Single step. (Not for EJTAG64)
    Syntax: t
    Shortcut: F10
36. **g:** Execute in full speed.
    Syntax:g
    Shortcut: F5
37. **tap:** Send tap command to the device (Used for de-bricking MIPS device).
    Syntax: tap x (y)
    Example: tap c
    tap a fffffff
38. **blkchk:** Check if the flash is blanked. If the result is not blanked do not try to use sprogram.
    Syntax: blkchk tabname
    blkchk start length
    Example:
    ```
    -blkchk flash
    Flash blanked
    ```
39. **flshdcth:** This happens on IRD6000 dish receiver, while there were two flash chip and one hold the high word of data and another flash hold lower word of data. flshcdth uses different routine to detect the flash. In the usbjtag.def the "Protocol=DCU" and "HiLo=1". If not the flshdcth will not get the right result.
    Syntax:flshdcth tabname
40. **hdramh:** This is the same reason as for flshdcth. When file saved in high word or lower word. You can use ldramh command to only load to high word or lower word of the memory in pc.
    Syntax: ldram address
    Example:
    `-ldramh 7fc00000` (lower word file)
    `-ldramh 7fc00002` (high word file.)
41. **speed:** Select JTAG speed.
    -speed 2 (select 3MHz JTAG or 1.5MHz BDM)

3.4 JTAG Usage

1. Read memory
   Use "getram" command. Eg. "getram u22"
2. Write memory
   Use "setram" command. Eg. "setram nvram"

Program flash

The flash contains all the essential programs for your target to work properly. Therefore it is very important to understand the process of programming flash. In most case, you can burn flash in fast mode. But if the flash is accidentally erased and no initialization is known, then a slow mode is used
First you need to make sure that watchdog is disabled. For known board with proper initialization a "detect" command will both detect the CPU and initialize the target. IT

IS VERY IMPORTANT EACH TIME TARGET IS PLUGGED IN, A "DETECT" needs to be executed. F1 is the short cut key.

EJTAG only.
If it is the first time you program the flash, type the command "bk", this command puts the CPU in "DEBUG ON" mode. Please do nothing for at least 30 seconds. If "debug on" stays, then the watch dog is disabled and you can program the flash safely. You only need to do this once for each type of target. If DEBUG goes off, then the watchdog is not disabled and program the flash is unsafe. Before you do some real programming, make sure the data is valid. ("bk" is break command and "g" is run to normal command).

You can then program the flash by typing the command "program tabname or program start length". It is highly recommend the first time find a non-used sector to program and compare to make sure you have can program properly.

3.4 Bug report.

Please report bug to usbbdm@usbjtag.com . Since there are so many targets out there and people use them in different way, there will be bugs in the software. (Especially when using scripts). So if you find a bug or want to make an enhancement request, do not hesitate to write to me. The support forum can also be used.
www.usbjtag.com/vbforum

Chapter 4 Script

USB JTAG NT support very complicated script so it can automate a lot of the jobs.

4.1 Basic script.

Basic script does not use controls and only have commands. Here is an example of basic script. (5101a.usp)

```
// ***************************
// sb5100 restore script
// ***************************
detect
ldram 9fc00000 %1
echo Press enter to program, any other character with return exit the script.
pause
program 9fc00000 200000
```

In the command window you type in

```
Sb5101a backup1.bin
```

Which the script will take one parameter as backup1.bin. Here is the explanation of how the script works.

```
// ***************************
// sb5100 restore script
// ***************************
```

These lines only displays the output to screen. (Echo does the same thing but will not display “Echo”)

```
detect
```

Execute command “detect”

```
ldram 9fc00000 %1
```

Equivalent to `ldram 9fc00000 backup.bin`

```
echo Press enter to program, any other character with return exit the script.
```

Display a prompt.

```
pause
```

Pause for you to hit “Enter”. You can enter any character and Enter to exit the script.

```
program 9fc00000 200000
```

Do the programming.