Using the FTDI IC-based USB adapter to debug MIPSfpga system software

The *MIPSfpga Getting Started Guide* describes only one way of onboard software debug: Bus Bluster ($50). This document describes how to get the same results with the USB adapter ($10), based on FTDI IC with Multi-Protocol Synchronous Serial Engine (MPSSE) support: FT232H, FT2232H, FT4232H, FT2232D.
It also contains some info about the integration of Codescape GNU gdb debugger with Visual Studio Code (VSCode) GUI in remote debug mode.

Main features

- this solution was successfully tested on Terasic DE10-Lite board with Pinboard II board FTDI-module (based on FT2232D);
- the command line debug (gdb console) is fully supported;
- the debug with GUI tools is limited by VSCode vscode-cpptools plugin capabilities. As it is oriented on work with high-level language (C/C++) it does not have such things as: assembler view, memory view, register view.
- This document, all source and additional docs are available on github: https://github.com/zhelnio/memos

Files description

mipsfpga_ftdi.cfg

- OpenOCD config file;
- contains interface and targets parts (all in one), can be divided on 2 parts;
- all configuration parameters have comments inside file;
- used to run OpenOCD and connect it to the FPGA board;
- the only configuration file that is required to run command line debug.

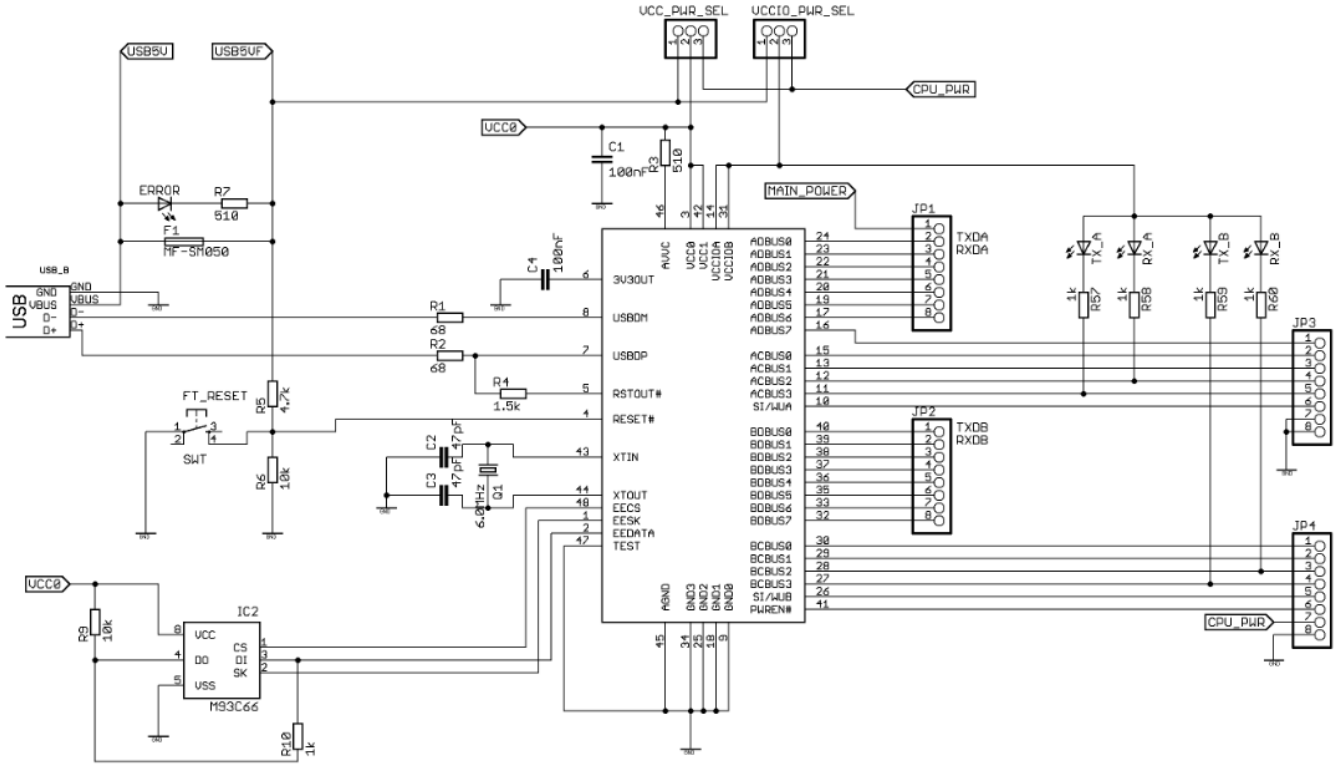
c_cpp_properties.json

- vscode-cpptools plugin config;
- contains information about header files placement to simplify navigation and code navigation and auto-completion;

launch.json

- contains Debug profile settings: everything that describes the behavior of VSCode managed gdb running process and its connection to OpenOCD process;
- contains two profiles: "MIPS Load" - upload and debug; "MIPS Attach" - attach and debug loaded software;
- see comments inside this file;

USB module schematic



This image contains the fragment of Pinboard II schematic. Image source: easyelectronics.ru

USB module to FPGA board connection table

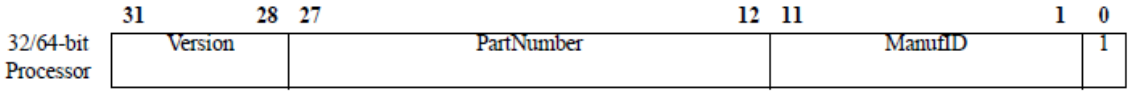
#	FT2232D						Goal	DE10-Lite	MIPSfpga	
	Generic Pin name	232 UART Mode	MPSSE	Bit Num (x16)	Init value	Direction		GPIO	Signal	Info
1	2	3	4	5	6	7	8	9	10	11
1	Channel **A** (MPSSE present)									
2	ADBUS0	TXD	TCK	0	0	1 (out)	EJTAG TCK	17	EJ_TCK	
3	ADBUS1	RXD	TDI	1	0	1 (out)	EJTAG TDI	21	EJ_TDI	
4	ADBUS2		TDO	2	0	0 (in)	EJTAG TDO	19	EJ_TDO	
5	ADBUS3		TMS	3	1	1 (out)	EJTAG TMS	23	EJ_TMS	
6	ADBUS4		GPIOL0	4	1	1 (out)	EJTAG SRSTn	20	~SI_ColdReset	
7	ACBUS3		GPIOH3	11	0	1 (out)	LED INDICATOR			
8	Channel **B** (MPSSE **not** present)									
9	BDBUS0	TXD					UART TX	31 (33)	UART_RX (UART_SRX)	
10	BDBUS1	RXD					UART RX	32 (35)	UART_TX (UART_STX)	
11									EJ_TRST_N	always 1 after Power On
12									EJ_DINT	always 0

Installation

- check the USB module schematic: the voltage on its output pins should be less than the maximum allowable voltage on fpga port;
- connect the USB module to the PC;
- change the USB module drivers to WinUSB using Zadig (for ftdi channel with MPSSE support);
- place the mipsfpga_ftdi.cfg file somewhere not far from the openocd-0.9.2.exe binary;
- connect the USB module to the FPGA board according to the connection table;
- if your connection schema is different from connection table, then you need to update the mipsfpga_ftdi.cfg parameters. The magic numbers in it are based on columns 5-7 of connection table:
  ftdi_layout_init 0x0018 0x081b
  0x0018 - init value (bits 3 and 4 are up: ADBUS3 and ADBUS4);
  0x081b - direction (ADBUS0,1,3,4, ACBUS3 are output);

RTL settings

- change the IDCODE settings in mfp_system.v (EJ_ManufID, EJ_PartNumber). This will simplify the connection test;
- The IDCODE register format is shown below:



- Check EJ_TRST_N and EJ_DINT wires. They are not used in this configuration and should be:
  assign EJ_TRST_N = 1'b1;
  assign EJ_DINT = 1'b0;
  Set this values in code or by setting jumpers;

Compile and memory settings

- check that memory settings in mipsfpga_ftdi.cfg tap configuration command parameters relate with gcc compile settings;
- add debug symbols to output file with -g -gdwarf-2 gcc options;
- set the optimization level to -O0 or -O1. It will also work with -O2, but you can see some "jumping" current operation cursor in interface in this case.

Connection test

- read all the comments in mipsfpga_ftdi.cfg
- uncomment the shutdown command and comment all the commands bellow;
- run openocd-0.9.2.exe -f mipsfpga_ftdi.cfg
- you should see something like this:

```
Open On-Chip Debugger 0.9.1-dev-microAptiv-dirty (2015-05-08-15:32)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
adapter speed: 10000 kHz
adapter_nsrst_delay: 100
jtag_ntrst_delay: 100
srst_only separate srst_nogate srst_push_pull connect_deassert_srst
shutdown command invoked
```

```
Info : clock speed 10000 kHz
Warn : There are no enabled taps.  AUTO PROBING MIGHT NOT WORK!!
Warn : AUTO auto0.tap - use "jtag newtap auto0 tap -expected-id 0x000f1005 ..."
Warn : AUTO auto0.tap - use "... -irlen 5"
Warn : gdb services need one or more targets defined
```

Where 0x000f1005 is your IDCODE.

- check that you see the same IDCODE value as it was set in RTL (mfp_system.v);
- check the connection or change the speed parameter if your IDCODE is broken;
- after successful connection test (RTL IDCODE is identical to received) comment the shutdown command and uncomment others;

Command line mode debug

- run openocd-0.9.2.exe -f mipsfpga_ftdi.cfg I prefer to run it in the separate terminal window because openocd process can sometimes hangs after connection loosing;
- you should see something like this

```
Open On-Chip Debugger 0.9.1-dev-microAptiv-dirty (2015-05-08-15:32)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
adapter speed: 10000 kHz
adapter_nsrst_delay: 100
jtag_ntrst_delay: 100
srst_only separate srst_nogate srst_push_pull connect_deassert_srst
scan delay: 20000 nsec
running in fast queued mode
Info : clock speed 10000 kHz
Info : JTAG tap: auto0.tap tap/device found: 0x000f1005 (mfg: 0x002, part: 0x00f1, ver: 0x0)
```

- open new terminal window and run the gdb with some commands to check its work: connected to the system under debug, stopping it, loading the program into its memory, setting the breakpoint on the main function enter, continuing, getting registers values after breakpoint is achieved.

```
> mips-mti-elf-gdb -q program.elf
Reading symbols from program.elf...done.
(gdb) target remote localhost:3333
Remote debugging using localhost:3333
0xbfc00000 in ?? ()
(gdb) set endian little
The target is assumed to be little endian
(gdb) monitor reset halt
JTAG tap: auto0.tap tap/device found: 0x000f1005 (mfg: 0x002, part: 0x00f1, ver: 0x0)
target state: reset
entered debug state at PC 0xbfc00000, target->state: halted
target state: halted
target halted in MIPS32 mode due to debug-request, pc: 0xbfc00000
(gdb) load
Loading section .text_ram, size 0x260 lma 0x80001000
Loading section .init, size 0x24 lma 0x80001260
Loading section .fini, size 0x1c lma 0x80001284
Loading section .eh_frame, size 0x4 lma 0x800012a0
Loading section .data, size 0xc lma 0x800012a4
Loading section .ctors, size 0x8 lma 0x800012b0
Loading section .dtors, size 0x8 lma 0x800012b8
Loading section .jcr, size 0x4 lma 0x800012c0
Loading section .reset, size 0x280 lma 0x9fc00000
Start address 0xbfc00000, load size 1348
Transfer rate: 12 KB/sec, 149 bytes/write.
(gdb) b main
Breakpoint 1 at 0x800011e0: file main.c, line 14.
(gdb) c
Continuing.
entered debug state at PC 0x800011e0, target->state: halted

[Remote target] #1 stopped.
main () at main.c:14
warning: Source file is more recent than executable.
14      {
(gdb) i r
          zero       at       v0       v1       a0       a1       a2       a3
 R0   00000000 deadbeef 800011e0 00000010 00000000 00000002 80001000 00000000
            t0       t1       t2       t3       t4       t5       t6       t7
 R8   deadbeef deadbeef deadbeef deadbeef deadbeef deadbeef deadbeef deadbeef
            s0       s1       s2       s3       s4       s5       s6       s7
 R16  deadbeef deadbeef deadbeef deadbeef deadbeef deadbeef deadbeef deadbeef
            t8       t9       k0       k1       gp       sp       s8       ra
 R24  deadbeef deadbeef deadbeef deadbeef 800092a8 80040000 00000000 9fc00274
       status       lo       hi badvaddr    cause       pc
     00400000 00000100 00000000 c0000000 00000008 800011e0
(gdb)
```

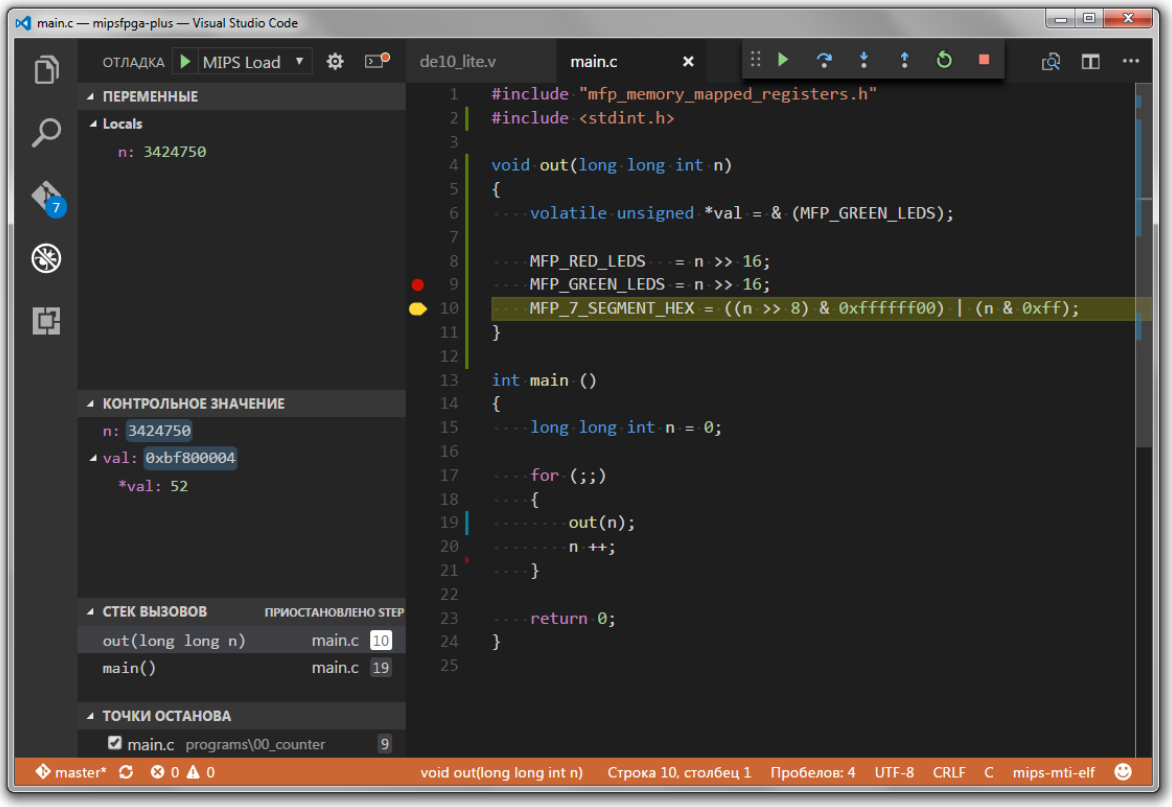
- also the **compare-sections** command can be used to check the sections in device memory after **load**.

GUI mode debug settings

- install the Visual Studio Code and its vscode-cpptools plugin;
- run it and open in some workspace directory that contains source code files;
- copy c_cpp_properties.json and launch.json to the .vscode folder in the top of the workspace directory;
- open these files, update toolchain path and compiled program elf file path;
- there is a small bug and in my case the full file path should be specified in gdb file command;

GUI mode debug

- run openocd-0.9.2.exe -f mipsfpga_ftdi.cfg in the similar way as it was described above;
- open the Debug action panel of VSCode and select MIPS Load profile;
- run the debug process;
- if something goes wrong uncomment the "logging" settings in launch.json and read error messages in debug console (**ctrl + `**);
- the successfully running VSCode debug session is looking something like this:

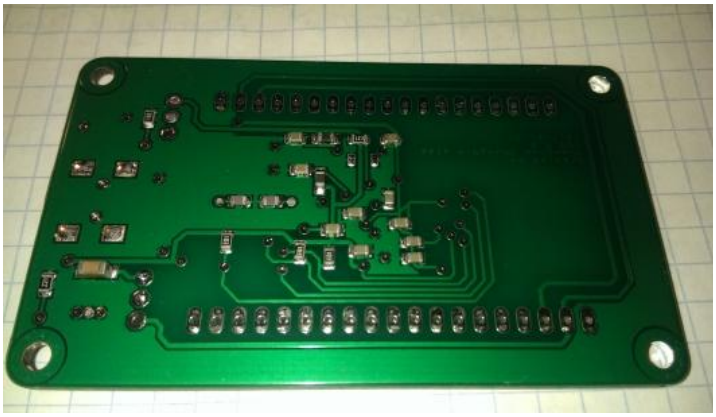


UPDATE 2017.06.05

I have successfully checked this EJTAG debug configuration with this FT2232**H**L based module from Aliexpress (price $15). All the debug procedures works without any settings changes as it was described above:

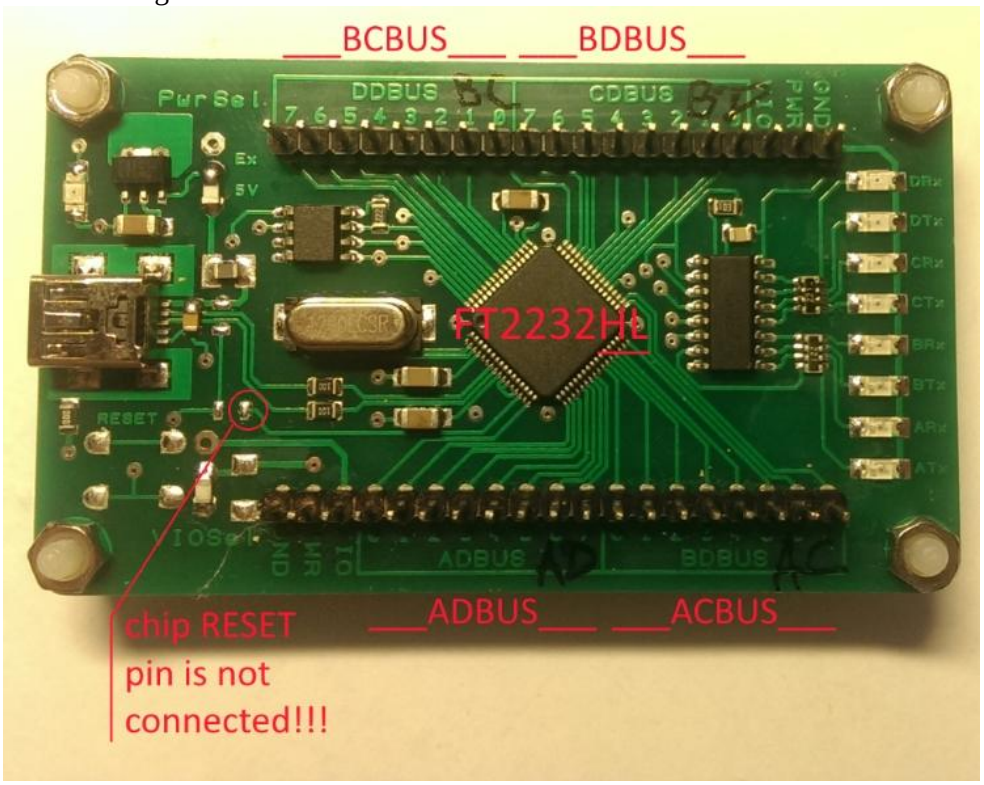
Front View

Back View



There are some details that one should to take into consideration when working with it (see the photo below for detailes):

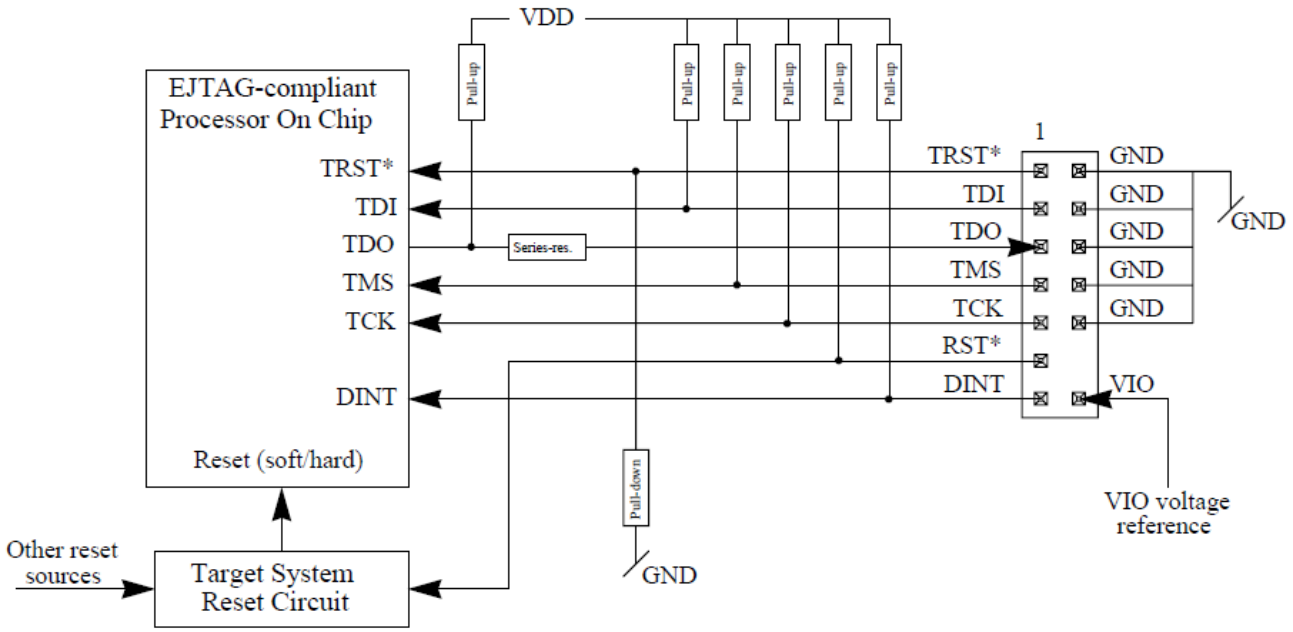
- The reset circuit elements (and the reset button) are not present;
- The chip reset input is not pulled-up. One can reset the chip when just measuring voltage level on this pin;
- The bus names marking on the board is incorrect.



Electrical connection

According to the EJTAG Specification the debugger should be connected to the SoC with pull-up/down and serial connected resistors:

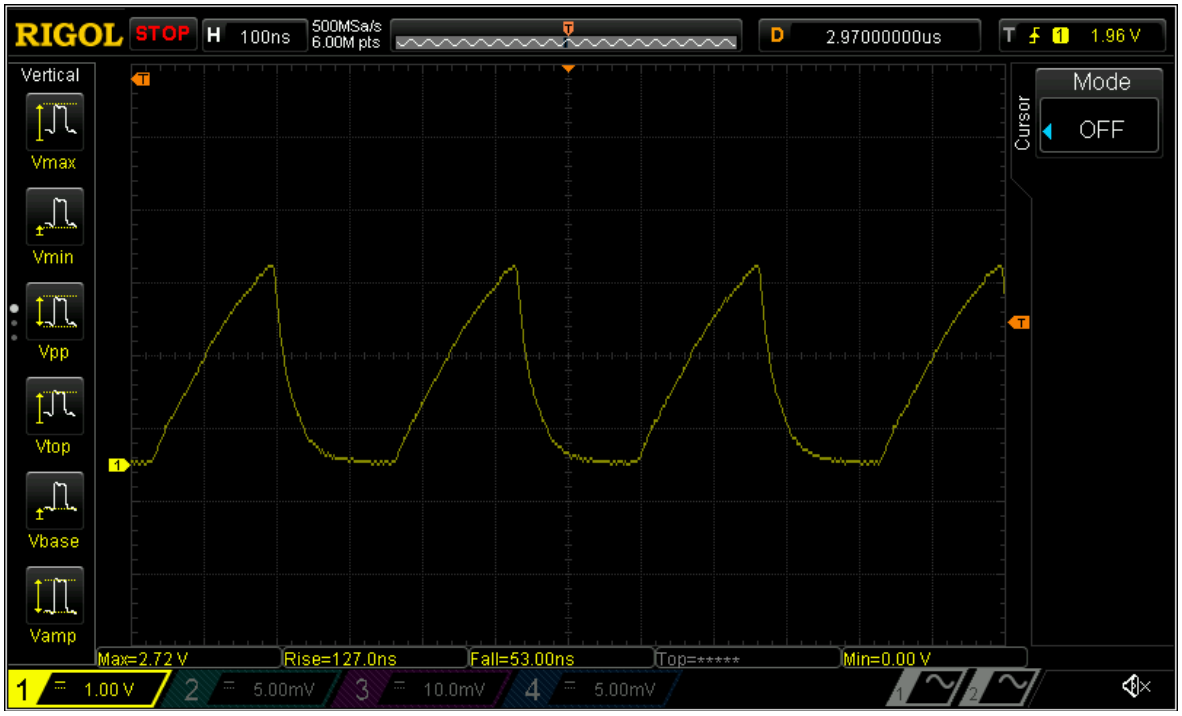
Figure 8-7 Target System Electrical EJTAG Connection



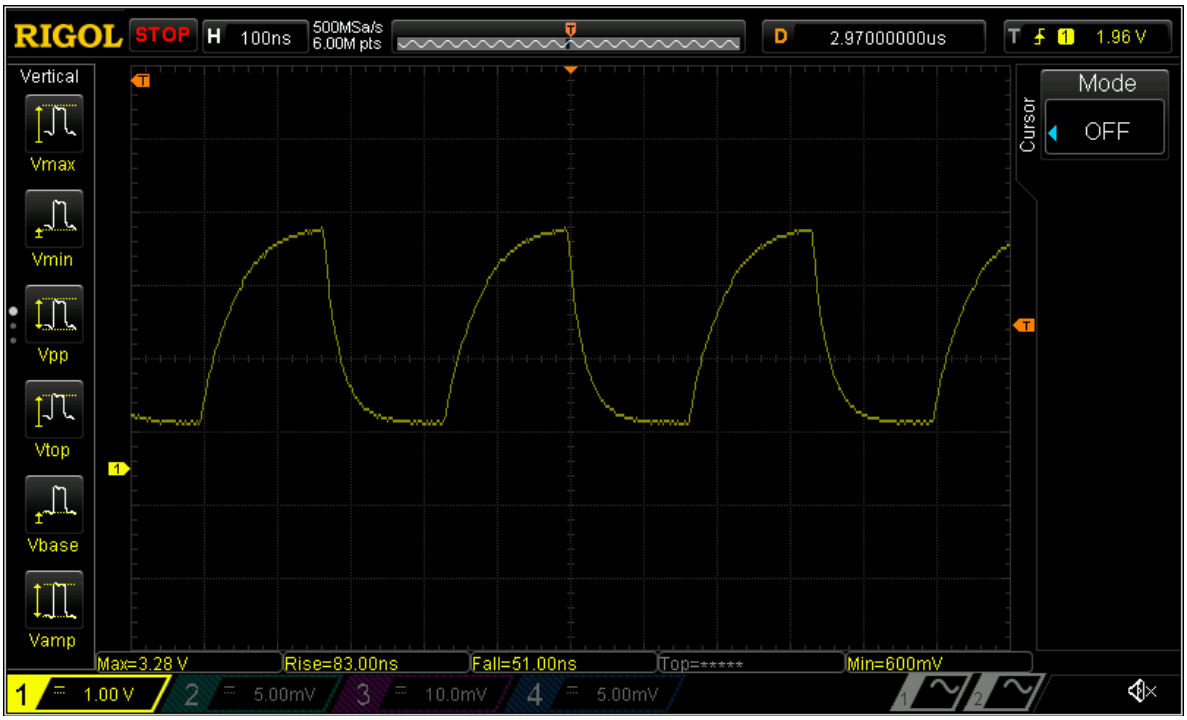
In Figure 8-7, the pull-up resistors for TCK, TMS, TDI, DINT, and RST*, the pull-down resistor for TRST*, and the series resistor for TDO must be adjusted to the specific design. However, the recommended pull-up/down resistor is 1.0 kΩ because a low value reduces crosstalk on the cable to the connector, allowing higher TCK frequencies. A typical value for the series resistor is 33 Ω. Recommended resistor values have 5% tolerance.

In my case the USB board was connected directly to the FPGA pins and this configuration is working fine. But for the better solution stability one have to use those resistors. Some example oscillograms are presented below:

The EJTAG TCK signal (no pull-up)



The EJTAG TCK signal (1K pull-up)



Document sources

1. MIPSfpga Getting Started Guide
2. FTDI Application Note AN 108. Command Processor for MPSSE and MCU Host Bus Emulation Modes.
3. FTDI Application Note AN 135. FTDI MPSSE Basics.
4. FTDI Software Application Development. D2XX Programmer's Guid.
5. FTDI FT2232D Datasheet.
6. Using the GNU Compiler Collection. Codescape GNU Tools 2016.05-03 for MIPS MTI Bare Metal.
7. Debugging with gdb. Codescape GNU Tools 2016.05-03 for MIPS MTI Bare Metal.
8. EJTAG Specification. Document Number: MD00047.
9. OpenOCD User’s Guide for release 0.8.0.
10. The schematic of Pinboard II devboard.