OPEN JTAG

www.openjtag.org

Communication Protocol Manual

Release 1.1
January 9, 2011

General considerations

The OPENJTAG PROJECT protocol is simple and easy to understand. Only one or two bytes are needed to make a complex operation, like move the target TAP state machine, shift a byte, etc. It is not as others FTDI based JTAG, that a single byte moves the JTAG pins high or low. The OPENJTAG PROJECT uses commands as macro-instructions, and the on-board CPLD interpret and execute these macro-instructions. For example, to shift 8 bits inside the target TAP, the others FTDI based JTAG use a sequence of bytes to switch high/low pin states. One byte to switch TCK high, the next byte to switch TCK low, and so on. OPENJTAG PROJECT uses only two bytes: the first byte is the command, telling to the CPLD how many bits must be shifted (1 to 8), and the second byte is the byte to shift out, reading the TDO results in the same operation. This method allows the hardware to shift TCK to a maximum frequency of 24 MHZ.

For example:

We send only 3 bytes to the JTAG USB bus:

0x41 - Sets Target TAP state to Shift-DR
0xF6 - Shifts out 8 bits and sets TMS high at last bit (to change state to Exit1-DR)
0x99 - Byte to shift

This is a timing output sequence from Quartus II simulation. We supposed the current target TAP state is Test-Logic-Reset:

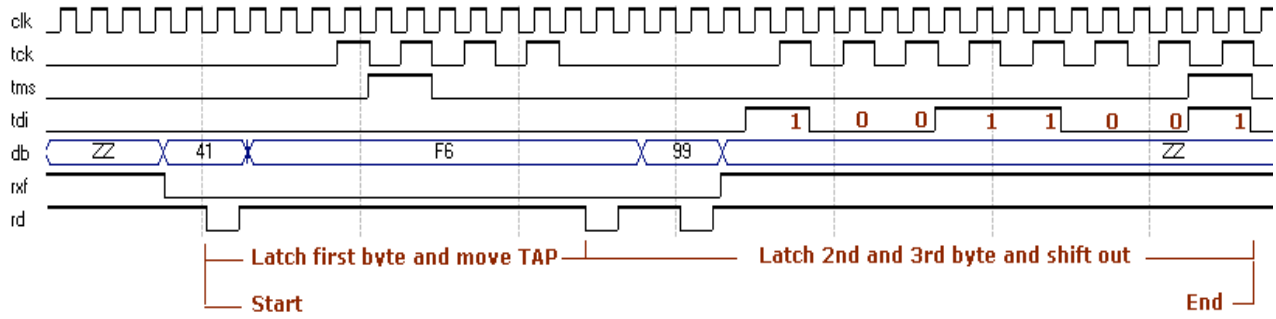

Figure 1

As the principal bottleneck in the FT245 is the USB communication speed, about 300 KB/sec, OPENJTAG PROJECT has found a method to minimize the USB traffic. The CPLD make the shift operations speedy and secure.
In the Figure 1, the db bus is the FT245 data bus that communicates directly with the CPLD. The first four TCK pulses moves the target TAP to the Capture-DR state, the next eight TCK pulses shift out the TDI signal, passing to the Exit1-DR state at the last TCK pulse (TMS=1).

Implementing this protocol is very simple. You must use the D2XX libraries from FTDIChip. These libraries are free and can be used without royalties fees. Please read the FTDIChip documentation at:

http://www.ftdichip.com/Documents/ProgramGuides.htm

In the official OPENJTAG PROJECT web site, www.openjtag.org, you can find C++ examples using Microsoft Visual Studio 6.

Protocol description

The protocol is only one byte. The first 4 bits (Bit 0, 1, 2 and 3) are the command part, and the bits 4, 5, 6 and 7 the parameters. If necessary, a second byte could be sent containing the data part. The JTAG does not send an answer and does not implement any error return. Whenever the JTAG should send an answer, it is because it has data to send to the computer side.
With the implementation of these commands (see Command summary), the PC software is capable to work around the target's TAP: the application by using these commands may or may not know in which state of the state machine is currently working or the command needed to move to another state. By sending the command, the JTAG will automatically do all the required signaling.

The instruction byte format is:

7	6	5	4	3	2	1	0
P	P	P	P	C	C	C	C

Where:

- **CCCC** is the command part
- **PPPP** is the parameter part (if any)

Command summary

0000	Set clock divisor
0001	Set target TAP state
0010	Get target TAP state
0011	Software reset target TAP
0100	Hardware reset target TAP
0101	Set LSB/MSB mode
0110	Shift out and read n bits
0111	RTI Loop
1000	Shift out and read n bytes

The commands 1001 to 1111 are reserved for future software expansion, like implementing the new ARM SW debug protocol, read/write the user input/output, read the target VCC and other issues

Command description

0000 – Set clock divisor

This command sets the clock JTAG divisor. The main clock is 48MHZ. There is not data byte to be sent for this command.

Where byte is: DDDTCCCC

- **CCCC** => 0000 Set clock divisor
- **T** => Use ARM Adaptive Clock (to be implemented)
- **DDD** => Clock divisor:
    - 000 => 48 MHZ (no divide)
    - 001 => 24 MHZ
    - 010 => 12 MHZ
    - 011 => 6 MHZ
    - 100 => 3 MHZ
    - 101 => 1.5 MHZ
    - 110 => 750 KHZ
    - 111 => 375 KHZ

Note: The TCK frequency will be clock/2

0001 – Set target TAP state

This command moves TCK and TMS signals to set the target TAP to the desiderate IEEE Std 1149.1 state. There is not data byte to be sent for this command. Please see the TAP state machine at end of this manual.

Where byte is: SSSSCCCC

- **CCCC** => 0001 Set target TAP state
- **SSSS** => New state
    - 0000 => Test-Logic-Reset
    - 0001 => Run-Test/Idle
    - 0010 => Select-DR-Scan
    - 0011 => Capture-DR
    - 0100 => Shift-DR
    - 0101 => Exit1-DR
    - 0110 => Pause-DR
    - 0111 => Exit2-DR
    - 1000 => Update-DR
    - 1001 => Select-IR-Scan
    - 1010 => Capture-IR
    - 1011 => Shift-IR
    - 1100 => Exit1-IR
    - 1101 => Pause-IR
    - 1110 => Exit2-IR
    - 1111 => Update-IR

Note: If *new state* is equal to *old state*, no action will be performed.

Note 1: There is not a way to know the TRUE target's TAP state, the hardware only move TCK and TMS signals to change from a known state to another known state.

0010 – Get target TAP state

This command return the current TAP state. There is not data byte to be sent for this command.

Where byte is: XXXXCCCC

- **CCCC** => 0010 - Get target TAP state
- **XXXX** => Don't care

The JTAG sends a byte containing:

XXTBCCCC

Where byte is:

- **CCCC** => The current TAP state
- **B** => 1 = MSB bit ON, 0 = MSB bit OFF
- **T** => 1 = Target power ON, 0 = Target power OFF
- **XX** => Don't care, garbage

Note: There is not a way to know the TRUE target's TAP state. The JTAG only returns the current state.

0011 – Software reset target TAP

This command sends 5 TCK pulses with TMS high to move the target's TAP to the Test-Logic-Reset state. See the **Hardware reset TAP** command to a hardware alternative. There is not data byte to be sent for this command.

Where byte is: XXXXCCCC

- **CCCC** => 0011 - Software reset target's TAP
- **XXXX** => Don't care

Note: There is not a way to know the TRUE target's TAP state. The JTAG assume that the target's TAP is in Test-Logic-Reset state.

0100 – Hardware reset target TAP

This command asserts the TRST signal low during PPPP TCK pulses. This is the safest mode to reset the target's TAP, but not all the devices support this mode. See **Software reset TAP** command to a software alternative. There is not data byte to be sent for this command.

Where byte is: PPPPCCCC

- **CCCC** => 0100 – Hardware reset target TAP
- **PPPP** => TCK pulses (0000 = 1 pulse, 1111 = 16 TCK pulses)

Note: There is not a way to know the TRUE target's TAP state. The JTAG assumes that the target's TAP is in Test-Logic-Reset state.

0101 – Set LSB/MSB mode

This command sets the shift mode (read and write). When set, remains in this state unless another **Set LSB/MSB mode** command is sent. The default starting mode is MSB first. There is not data byte to be sent for this command.

Where byte is: XRBMCCCC

- **CCCC** => 0101 – Set LSB/MSB mode
- **M** => Mode (0 = MSB first, 1 = LSB first)
- **B** => 1 = Blue LED ON, 0 = Blue LED OFF
- **R** => 1 = Red LED ON, 0 = Red LED OFF
- **X** => Don't care

0110 – Shift out and Read n bits

This command sends the next byte to the target TAP, and read the TDO input, returning the read byte. The shift direction is the default. See **Set LSB/MSB mode** command.

Where the first byte is: NNNTCCCC
Where the second byte is: DDDDDDDD

- **CCCC** => 0110 - Shift out and Read n bits
- **T** => Change state bit. (1 = TMS high at last bit, 0 = TMS low at last bit)
- **NNN** => Bits to transfer (000 = 1, 111 = 8)
- **DDDDDDDD** => Data to shift out to TDI

The JTAG sends back the NNN bits read from the TDO input.

Please see the Figure 1 timing diagram.

Note: This command only works when the TAP is in the Shift-DR and Shift-IR states. If it is sent when the state machine is not in the proper state, the byte command and the data byte will be ignored.

0111 – RTI Loop

Performs 1 to 16 clock cycles with TMS=0. The current TAP state **MUST** be RTI (Run-Test-Idle) otherwise this command will be ignored.

Where byte is: NNNNCCCC

- **CCCC** => 0111 – RTI Loop
- **NNNN** => Clock cycles (0000 = 1, 1111 = 16)

*****TO BE IMPLEMENTED*****

1000 – Shift out and read n bytes

This command sends the next n byte to the target TAP, and read the TDO input, returning the read byte. The shift direction is the default. See **Set LSB/MSB mode** command.

Where the first byte is: NNNTCCCC
Where the other bytes are: DDDDDDDn

- **CCCC** => 1000 - Shift out and Read n bits
- **T** => Change state bit. (1 = TMS high at last bit of last byte, 0 = TMS low at last bit)
- **NNN** => bytes to transfer (000 = 1, 111 = 8)
- **DDDDDDD1** => Byte 1 data shift out to TDI
- **DDDDDDD2** => Byte 2 data shift out to TDI
- **.**
- **.**
- **DDDDDDDn** => Byte n data shift out to TDI

The JTAG sends back the NNN bytes read from the TDO input.

IEEE Std 1149.1 Test Access Port state machine

The TAP is controlled by the test clock (TCK) and test mode select (TMS) inputs. These two inputs determine whether an instruction register scan or data register scan is performed. The TAP consist of a small controller design, driven by the TCK input, which respond to the TMS input as shown in the state diagram in Figure 2. The IEEE Std 1149.1 test bus uses both clock edges of TCK. TMS and TDI are sampled on the rising edge of TCK, while TDO changes on the falling edge of TCK. Note that state numbers are under OPENJTAG PROJECT numeration, not the IEEE numeration

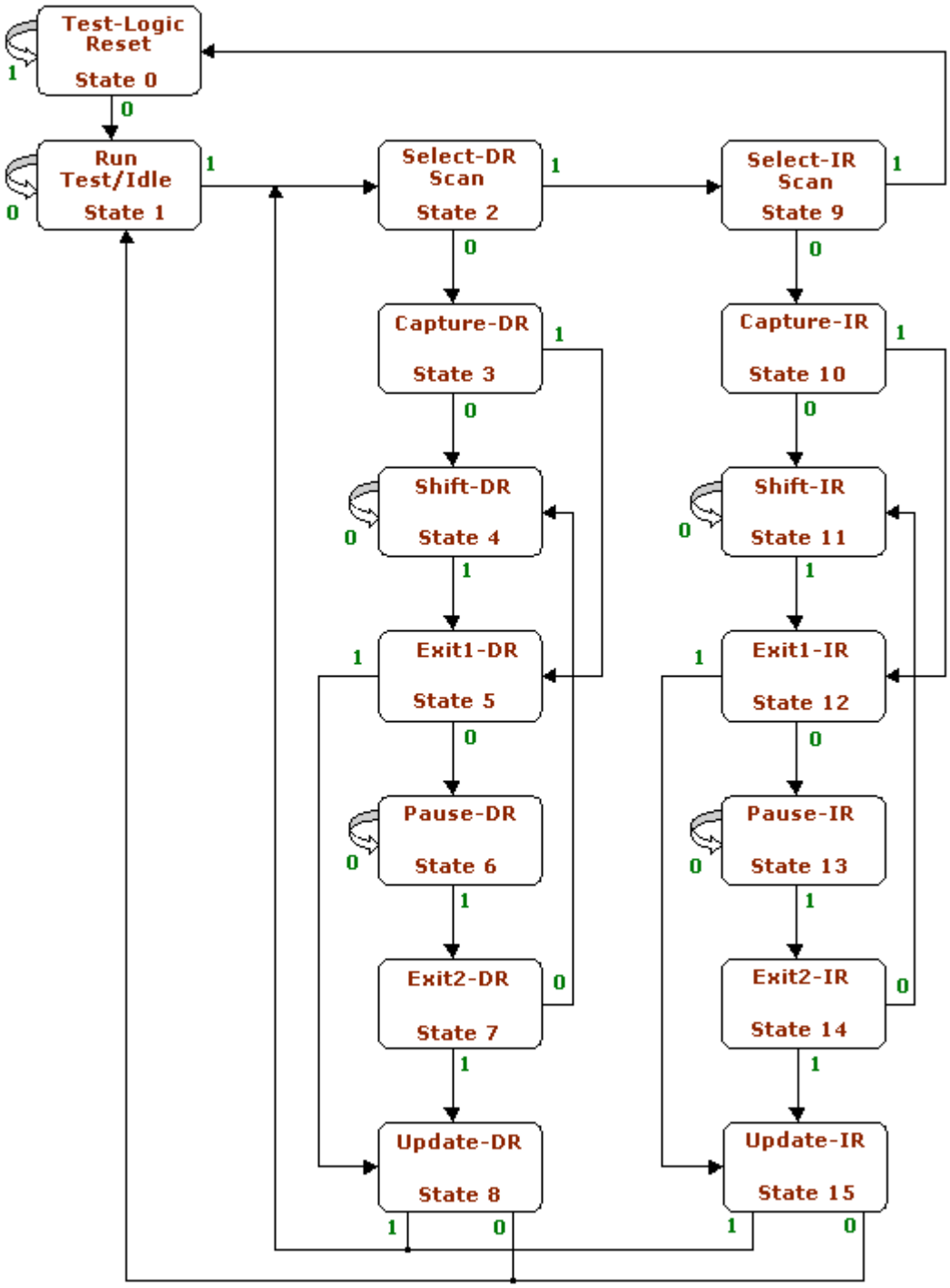

Figure 2