USBprog JTAG Interface

Benedikt Sauter

sauter@ixbat.de

1. Operating Mode

To get the best performance for a JTAG connection over USB its necessary to create a special protocol. The problem is USB transfer packets only every millisecond. But in a millisecond max. 1023 Bytes can be transfered (with USB 1.1). On the other hand to this packet oriented bus, JTAG implementations are often work with bit slices for a JTAG TAP. To get a optimal connection for a JTAG device over USB a special buffer for command packets are created. In a buffer many small operations can be collected , transfered to USBprog, handled from USBprog to the JTAG interfaces and send back a batch of responses back.

Execute JTAG commands means now, create a JTAG command buffer, fill buffers width commands, if buffer is full enough or its necessary the packets transfered to USBprog. All scan results are send back in a grouped packet.

2. USB Interface

- Endpoint Number 2 for outgoing packets (from pc to USBprog)
- Endpoint Number 0x82 for incomung packets (from USBprog to pc)
- Vendor Requests at Endpoint Number 0 for configuration

Send- or receive packets should be max. 320 Bytes large. With this limit USBprog reaches the best performance.

3. Hardware Interface

Connect TDI of USBprog to TDI of the first JTAG TAP. If you use USBprog in a JTAG chain TDO of the first device after USBprog must connected to the next device's TDI signal (and so on). Last TDO of the JTAG chain must be connect to TDO of USBprog.

4. JTAG transfer commands

Following commands can be used to transfer JTAG chains from OpenOCD or another JTAG chain generator with USBprog into the target.

Packets for transfer buffer are shown in the next table. The typical packet structure is

```
[CMD][LEN (opt.)][DATAX...0 (opt.)] -> [RESPONSE_DATAX...0 (opt.)]
```

- CMD is a command from table commands
- LEN is Byte or Bit length for a data out, in or in and out packet
- DATAX….0 data out, in or in and out Bytes

Table 1. Commands

Command	b7	b6	b5	b4	b3	b2	b1	b0
SCAN	0	0	1	TDI(1)	Byte(1)	Read(1)	Write(1)	Value
SCAN	0	0	1	TMS(0)	Bit(0)	No Read(0)	No Write(0)	Value
GPIO	0	1	0	set(1)	Number2	Number1	Number0	Value
GPIO	0	1	0	get(0)	Number2	Number1	Number0	Value

Example Buffer from pc to USBprog:

```
[SCAN (only write)][LEN]DATA3][DATA2][DATA1][DATA0][SCAN (only read)][LEN]
[SCAN_RESPONSE][DATA3][DATA2][DATA1][DATA0]
```

Result Buffer

Only TD0 outputs and GPIO get operations are send back from USBprog to pc in Result Buffer. The order of the packets are the same as in the Send Buffer. This means the order of the Send Buffer must be saved by own application logic. Only on this way the answer bytes can be decoded correctly.

5. Commands

5.1. SCAN: For TDI/TDO and TMS scans

These commands are create from table Commands. All important combinations are here described. Some commands has got [DATA] in send packet and some at result packet. It depends on the Read/no Read and Write/ no Write Flags. But for all packets (in send buffer) [CMD] and [LEN] must be exists.

Table 2. JTAG Commands

CMD	Function	Description	RESPONSE
0x3A	Clock Data Bytes Out	Falling Clock Edge, LSB First (no Read)	-
0x32	Clock Data Bits Out	Falling Clock Edge, LSB First (no Read)	-
0x3C	Clock Data Bytes In	Rising Clock Edge, LSB First (no Write)	Scan
0x34	Clock Data Bits In	Rising Clock Edge, LSB First (no Write)	Scan
0x3E	Clock Data Bytes Out In	Out at Falling, In at Rising, LSB First	Scan
0x36	Clock Data Bits Out In	Out at Falling, In at Rising, LSB First	Scan
0x23	Clock Data to TMS (TDI 1)	Falling Clock, LSB First (no Read)	-
0x22	Clock Data to TMS (TDI 0)	Falling Clock, LSB First (no Read)	-
0x27	Clock Data to TMS (TDI 1)	Out at Falling, In at Rising LSB First	Scan
0x26	Clock Data to TMS (TDI 0)	Out at Falling, In at Rising LSB First	Scan

5.2. GPIO: Control JTAG pins manually

Control TDI,TMS,TCK,TRST and SRST or get actual value

Command	b7	b6	b5	b4	b3	b2	b1	b0
GPIO	0	1	0	set(1)	Number2	Number1	Number0	Value
GPIO	0	1	0	get(0)	Number2	Number1	Number0	Reserved

- Number is the pin number from table XXX written in binary format.

```
Example: 4 (TRST) = Number2 : 1, Number1 : 0, Number0 : 0
```

- Value is for set operation to define a new level (high or low) and at get operations it shows the actual level on the given pin.

Note: Only at pin TDO *GPIO get* operation is meaningful! Its the only input pin.

Table 3. JTAG Output Pins

Number	Pin	Dir
1	TDI	out
2	TMS	out
3	TCK	out
4	TRST	out
5	SRST	out
6	TDO	in

Example:

- Set TDI [0x53] (no LEN, no DATA and no RESPONSE_DATA)
- Get TDO [0x4C] (LEN=1, no DATA but RESPONSE_DATA -> if high: [0x01] else low: [0x00]

6. Configuration (Vendor Requests at Endpoint 0)

Every USB device can be configured with vendor requests. Therefore USBprogs offers vendor requests for JTAG speed configuration and operate the signal led and configuration's jumper position.

6.1. FIRMWARE UPDATE: Change Firmware of USBprog

- bmRequestType
- bRequest 0x01 (FIRMWARE UPDATE)
- wValue
- wIndex
- wLength

6.2. SPEED: Setup JTAG speed

Configure JTAG speed. (6 kHz - 5 MHz)

- bmRequestType
- bRequest 0x02 (set SPEED), 0x03 (get SPEED)
- wValue 0x0000 - 0xffff
- wIndex
- wLength

Every speed between 6 kHz and 5 MHz can be used.

Table 4. JTAG Example Speeds

Speed	Value
6 kHz	0x0006
100 kHz	0x0064
250 kHz	0x00FA
500 kHz	0x01F4
1 MHz	0x03eb
5 MHz	0x1388

At *bRequest 0x01* a zero packet is send back as result. At *bRequest 0x02* actual speed is send back in 2 bytes.

6.3. USER: User Interface (LED and Jumper as switch)

- Change status led of USBprog
- Get Jumper position
- bmRequestType
- bRequest 0x04 (USER)
- wValue 0x01 (set LED), 0x02 (clear LED), 0x03 (get Jumper position)
- wIndex
- wLength

At *bValue 0x01 and 0x02* a zero packet is send back as result. At *bValue 0x03* the answer 0x00 means Jumper is open and 0x01 closed.

6.4. VERSION: Get actual Firmware Version

- bmRequestType
- bRequest 0x05 (VERSION)
- wValue
- wIndex
- wLength

As result a 4 Byte version number is send back.

7. Software API

7.1. Open and close connection

Open and close connection.

```
usb_dev_handle *usbprog_locate(void);
```

This functions return a *struct usb_dev_handle *usbprog_handle* as result.

```
int usbprog_close(usb_dev_handle * usbprog_handle);
```

To close the connection usbprog_close must be called.

7.2. Chain transfer (TDI/TDO and TMS)

Send JTAG commands (described in this datasheet) and receive them results. In *buffer* all commands are in an array. With *lenght* the lenght of all commands in *buffer* is mean.

```
int usbprog_command_buffer(usb_dev_handle * usbprog_handle,
    char * write_buffer, int write_length,
    char * read_buffer, int read_length);
```

7.3. Control Reset Signals

With these both functions the JTAG reset signals can be managed. Use value 1 as high (3,3 or 5 V) and 0 as low (0 V, GND).

```
int usbprog_trst(usb_dev_handle * usbprog_handle, int value);
int usbprog_srst(usb_dev_handle * usbprog_handle, int value);
```

7.4. Speed configuations

Setup JTAG speed. As *value* parameter give the number of kHz (6 kHz = 6 or 5 MHz = 5000).

```
int usbprog_speed(usb_dev_handle * usbprog_handle, int value);
```

7.5. Status LED

There is a status/control LED on USBprog. With this function the led can be turned on or off. (on=1, off=0)

```
int usbprog_led(usb_dev_handle * usbprog_handle, int value)
```