Digilent Adept JTAG Interface (DJTG) Programmer's Reference Manual

Revision: September 11, 2013

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

This document describes the programming interface to the Digilent Adept JTAG Interface (DJTG) subsystem for version 2 of the Digilent Adept software system. It describes the capabilities of the DJTG subsystem and the API functions used to access its features.

JTAG (Joint Test Access Group) is a synchronous serial interface primarily intended for in-system test, debug, and programming of integrated circuits. It is defined and described as I.E.E.E specification 1149.1. Familiarity with the I.E.E.E. 1149 specification is assumed in this document.

The JTAG specification defines four mandatory signals: TMS (Test Mode Select), TDI (Test Data In), TDO (Test Data Out), and TCK (Test Clock). JTAG compatible devices are connected in a serial arrangement, with the TDO of one device being connected to the TDI of the next device, forming a large shift register. This serial arrangement is called a JTAG scan chain.

The JTAG specification also defines a TAP (Test Access Port) controller. The TAP controller is a state machine that is used as the primary interface to the JTAG functionality of the device or devices on the JTAG scan chain. The TMS pin is used to move the TAP controller through its state diagram to various TAP states. The TMS pin is sampled on the rising edge of TCK and the next TAP state is determined by the current state and the signal level on TMS. The TAP controller contains an instruction register and a data register, and data can be shifted into and/or out of these registers when in the appropriate TAP controller states. Data shifted into the registers is presented on the TDI input and data shifted out is presented at the TDO output of the scan chain.

The DJTG interface provides a set of API functions that can be used to manipulate the TAP controller state as well as to shift data into and out of the JTAG scan chain.

All DJTG API calls return a Boolean value: TRUE if the call is successful, FALSE if not successful.

DJTG Port Properties

The port property bits are used to indicate which optional parts of the DJTG interface are supported by a given port. A DJTG port corresponds to a single set of the four JTAG signals and connects to a single JTAG scan chain.

The following port properties bits are defined for DJTG ports: These values are defined in the header file *djtg.h*.

dprpJtgSetSpeed	This bit indicates that the DjtgSetSpeed function is supported by the port.
dprpJtgSetPinState	This bit indicates that the DjtgSetTmsTdiTck function is supported by the port.
dprpJtgWait	This bit indicates that the DjtgWait function is supported by the port.
dprpJtgBatch	This bit indicates that the DjtgBatch and DjtgGetBatchProperties functions are supported by the port.
dprpJtgSetAuxReset	This bit indicates that the DjtgSetAuxReset function is supported by the port.
dprpJtgSetGetGpio	This bit indicates that the DjtgSetGpioState, DjtgGetGpioState, DjtgSetGpioDir, DjtgGetGpioDir, and DjtgGetGpioMask functions are supported by the port.

DJTG Batch Operation

Master batch mode operation allows scripting a sequence of DJTG transactions. A script of batch commands specifying the transactions to be performed is placed in a buffer. The batch buffer is sent via the DjtgBatch function call for execution. Support for batch mode operation is indicated by the *dprpJtgBatch* port property bit. Ports without this property bit do not support batch mode operation.

The batch command buffer is filled with a sequence of batch commands and their parameters. Data to be sent to slave devices is specified as parameter data to commands and is therefore part of the command buffer. Any data read from slave devices on the bus is returned in a separate data buffer. If multiple reads are specified by the batch command buffer, the data returned by each read is packed in the return data buffer.

The batch commands and their parameters are byte oriented binary data. The commands and their parameters are placed in the command buffer as a packed sequence of bytes. The symbols related to batch commands are defined in the header file *djtg.h*. Ten batch commands are defined.

1. jcbSetTmsTdiTck – set the state of the TMS, TDI, and TCK pins.

   Parameters bytes:
   a. fsTmsTdiTck

Bit Number	7	6	5	4	3	2	1	0
	X	X	X	X	X	TMS State	TDI State	TCK State

2. jcbGetTmsTdiTdoTck – get the state of the TMS, TDI, TDO, and TCK pins.

   Parameter bytes: None

   Return Data Format:

Bit Number	7	6	5	4	3	2	1	0
	0	0	0	0	TMS State	TDI State	TDO State	TCK State

3. jcbPutTms – shift TMS bits into the JTAG scan chain.

   Parameter bytes:
   a. cbit[7:0]
   b. cbit[15:8]
   c. cbit[23:16]
   d. cbit[31:24]
   e. TMS data byte 0
   f. ….
   g. TMS data byte N

4. jcbPutTmsGetTdo – shift TMS bits into the JTAG scan chain and receive TDO bits.

   Parameter bytes:
   a. cbit[7:0]
   b. cbit[15:8]
   c. cbit[23:16]
   d. cbit[31:24]
   e. TMS data byte 0
   f. ….
   g. TMS data byte N

   Return Data Format:
   a. TDO data byte 0
   b. ….
   c. TDO data byte N

5. jcbPutTdi – shift TDI bits into the JTAG scan chain. The TMS pin remains in the state that was set by a previous operation. Prior to the last TDI bit being shifted the TMS pin is set to the state specified by *fsLastTms*. If a bit count of 1 is specified then the TMS pin is set prior to the bit being shifted.

   Parameter bytes:
   a. cbit[7:0]
   b. cbit[15:8]
   c. cbit[23:16]
   d. cbit[31:24]
   e. fsLastTms

Bit Number	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	X	TMS State

   f. TDI data byte 0
   g. ….
   h. TDI data byte N

6. jcbPutTdiGetTdo – shift TDI bits into the JTAG scan chain and receive TDO bits. The TMS pin remains in the state that was set by a previous operation. Prior to the last TDI/TDO bit being shifted the TMS pin is set to the state specified by *fsLastTms*. If a bit count of 1 is specified then the TMS pin is set prior to the bit being shifted.

   Parameter bytes:
   a. cbit[7:0]
   b. cbit[15:8]
   c. cbit[23:16]
   d. cbit[31:24]
   e. fsLastTms

Bit Number	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	X	TMS State

   f. TDI data byte 0
   g. ….
   h. TDI data byte N

   Return Data Format:
   a. TDO data byte 0
   b. ….
   c. TDO data byte N

7. jcbGetTdo – shift TDO bits out of the JTAG scan chain. The TMS and TDI pins remain in the states that were set by a previous operation. Prior to the last TDO bit being shifted the TMS pin is set to the state specified by *fsLastTms*. If a bit count of 1 is specified then the TMS pin is set prior to the bit being shifted.

   Parameter bytes:
   a. cbit[7:0]
   b. cbit[15:8]
   c. cbit[23:16]
   d. cbit[31:24]
   e. fsLastTms

Bit Number	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	X	TMS State

8. jcbClockTck - cycle TCK with no data transfer for the specified number of times.

   Parameter bytes:
   a. cclk[7:0]
   b. cclk[15:8]
   c. cclk[23:16]
   d. cclk[31:24]

9. jcbWaitUs -delay for the specified number of microseconds.

   Parameter bytes:
   a. ctus[7:0]
   b. ctus[15:8]
   c. ctus [23:16]
   d. ctus [31:24]

10. jcbSetAuxReset - set the state of the RESET pin.

    Parameter bytes:
    a. fsAuxReset

Bit Number	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	Enable Output on Aux Reset Pin	AUX RESET State

11. jcbGetGpioMask - get 32-bit masks that indicate which GPIO pins are available as outputs (*mskOutput*) and inputs (*mskInput*). A '1' in any given bit position indicates that the corresponding GPIO pin supports output or input. A '0' indicates that output or input is not supported.

    Parameter bytes: None.

    Return Data Format:
    a. mskOutput[7:0]
    b. mskOutputk[15:8]
    c. mskOutput[23:16]
    d. mskOutput[31:24]
    e. mskInput[7:0]
    f. mskInput[15:8]
    g. mskInput[23:16]
    h. mskInput[31:24]

12. jcbSetGpioDir - set the GPIO pin direction. The bit mask in *fsDirReq* specifies a '1' for each pin to be configured as an output and '0' for pins to be configured as inputs. Bits corresponding to unsupported pins should be specified as '0'. The value of *fsDirSet* indicates the resulting state of the pins.

    Parameter bytes:
    a. fsDirReq[7:0]
    b. fsDirReq[15:8]
    c. fsDirReq[23:16]
    d. fsDirReq[31:24]

    Return Data Format:
    a. fsDirSet[7:0]
    b. fsDirSet[15:8]
    c. fsDirSet[23:16]
    d. fsDirSet[31:24]

13. jcbGetGpioDir - get a bit mask containing the current setting of the GPIO pin direction. The value returned for *fsDir* will have a '1' for each pin that is currently configured as an output. The bits of *fsDir* will be set to '0' for pins that are either configured as inputs or are not supported by the current port.

    Parameter bytes: None

    Return Data Format:
    a. fsDir[7:0]
    b. fsDir[15:8]
    c. fsDir[23:16]
    d. fsDir[31:24]

14. jcbSetGpioState - set the state of GPIO pins that are presently configured as outputs. Each pin configured as an output will be set to a '1' or a '0' depending on the value of the corresponding bit in fsState. Bits corresponding to input pins or unsupported pins will be ignored and should be set to '0'.

    Parameter bytes:
    a. fsState[7:0]
    b. fsState[15:8]
    c. fsState[23:16]
    d. fsState[31:24]

15. jcbGetGpioState - get the state of the GPIO pins associated with the DJTG port. Pins that are configured as inputs will have the corresponding bit in *fsState* set to the state of the input condition at the pin. Pins that are configured as outputs will have the corresponding bit in *fsState* set to the last value written to the pin. Any bits corresponding to unsupported pins will be set to '0'.

    Parameter bytes: None
    Return Data Format:
    a. fsState[7:0]
    b. fsState[15:8]
    c. fsState[23:16]
    d. fsState[31:24]

DJTG Batch Properties

All DJTG ports that support batch operation are required to implement support for the following commands: jcbSetTmsTdiTck, jcbGetTmsTdiTdoTck, jcbPutTms, jcbPutTmsGetTdo, jcbPutTdi, jcbPutTdiGetTdo, jcbGetTdo, and jcbClockTck.

Some DJTG ports may implement support for additional commands.

The following batch properties bits are defined for DJTG ports that support batch operation: These values are defined in the header file *djtg.h*.

djbpWaitUs	This bit indicates that the jcbWaitUs command is supported by the port.
djbpSetAuxReset	This bit indicates that the jcbSetAuxReset command is supported by the port.
djbpSetGetGpio	This bit indicates that the jcbGetGpioMask, jcbSetGpioDir, jcbGetGpioDir, jcbSetGpioState, and jcbGetGpioState commands are supported by the port.

The DjtgGetBatchProperties function may be called to determine which commands are supported by a DJTG port that supports batch operation.

DJTG API Functions

The following API functions make up the DJTG interface.

DjtgGetVersion(char * szVersion)

Parameters

szVersion	- pointer to buffer to receive version string

This function returns a version number string identifying the version number of the DJTG DLL. The symbol cchVersionMax declared in dpcdecl.h defines the longest string that can be returned in *szVersion*.

DjtgGetPortCount(HIF hif, INT32 * pcprt)

Parameters

hif	- open interface handle on the device
pcprt	- pointer to variable to receive count of ports

This function returns the number of DJTG ports supported by the device specified by interface handle *hif*.

DjtgGetPortProperties(HIF hif, INT32 prtReq, DWORD * pdprp)

Parameters

hif	- open interface handle on the device
prtReq	- port number to query
pdprp	- pointer to variable to return port property bits

This function returns the port properties bits for the specified DJTG port. The port properties bits indicate the specific features of the DJTG specification implemented by the specified port.

DjtgGetBatchProperties(HIF hif, INT32 prtReq, UINT32 * pdjbp)

Parameters

hif	- open interface handle on the device
prtReq	- port number to query
pdjbp	- pointer to variable to return port batch property bits

This function returns the batch property bits for the specified JTG port. The batch properties bits indicate which of the optional batch commands are implemented by the specified port. This function is not supported unless the dprpJtgBatch bit is set in the port properties for the port.

DjtgEnable(HIF hif)

Parameters

hif	- open interface handle on the device

This function is used to enable the default DJTG port (port 0) on the specified device. This function must be called before any functions that operate on the DJTG port may be called for the specified device.

DjtgEnableEx(HIF hif, INT32 prtReq)

Parameters

hif	- open interface handle on the device
prtReq	- DJTG port number

This function is used to enable a specific port on devices that support multiple DJTG ports. This function must be called before any functions that operate on the DJTG port may be called. The *prtReq* parameter specifies the port number of the DJTG port to enable.

DjtgDisable(HIF hif)

Parameters

hif	- open interface handle on the device

This function is used to disable and end access to the currently enabled DJTG port on the specified interface handle.

DjtgGetSpeed(HIF hif, DWORD * pfrqCur)

Parameters

hif	- open interface handle on the device
pfrqCur	- pointer to return current TCK frequency

This function is used to get the current speed of the DJTG port. The value returned in pfrqCur is in HZ.

DjtgSetSpeed(HIF hif, DWORD frqReq, DWORD * pfrqSet)

Parameters

hif	- open interface handle on the device
frqReq	- requested TCK clock frequency (in Hz)
pfrqSet	- pointer to return actual TCK frequency obtained

This function is used to set the clock frequency at which TCK will be clocked when shifting data into the JTAG scan chain. The TCK clock frequency will be set to the highest supported value that doesn't exceed frqReq, or the lowest frequency supported if frqReq is lower than the lowest supported value. The actual clock frequency obtained is returned in frqSet. This function is not supported unless the dprpJtgSetSpeed bit is set in the port properties for the port.

The clock frequency set will be the highest frequency that will appear on the TCK pin of the port. Depending on the device and the communications protocol used to communicate with the device, the average clock frequency (and therefore the average data rate in to or out of the device) may be lower.

DjtgSetTmsTdiTck(HIF hif, BOOL fTms, BOOL fTdi, BOOL fTck)

Parameters

hif	- open interface handle on the device
fTms	- sets the state of the TMS pin
fTdi	- sets the state of the TDI pin
fTck	- sets the state of the TCK pin

This function is used to set the JTAG interface pins to the specified states. A boolean TRUE value causes a pin to be set to the logic 1 state and a FALSE value causes the pin to be set to the logic 0 state. This function is not supported unless the dprpJtgSetPinState bit is set in the port properties for the port.

DjtgGetTmsTdiTdoTck(HIF hif, BOOL* pfTms, BOOL* pfTdi, BOOL* pfTdo, BOOL* pfTck)

Parameters

hif	- open interface handle on the device
pfTms	- pointer to return current state of TMS pin (true = 1, false = 0)
pfTdi	- pointer to return current state of TDI pin (true = 1, false = 0)
pfTdo	- pointer to return current state of TDO pin (true = 1, false = 0)
pfTck	- pointer to return current state of TCK pin (true = 1, false = 0)

This function is used to read the current state of the TMS, TDI, TDO, and TCK pins.

DjtgSetAuxReset(HIF hif, BOOL fReset, BOOL fEnOutput)

Parameters

hif	- open interface handle on the device
fReset	- sets the state of the RESET pin
fEnOutput	- enable output on the RESET pin

This function is used to set the AUX RESET pin to the specified state. A boolean TRUE value causes the pin to be set to the logic 1 state and a FALSE value causes the pin to be set to the logic 0 state. Specifying a boolean FALSE for fEnOutput disables output on the AUX RESET pin. JTAG interface pins to the specified states. This function is not supported unless the dprpJtgSetAuxReset bit is set in the port properties for the port.

DjtgGetGpioMask (HIF hif, DWORD * pfsMsakOut, DWORD * pfsMaskIn)

Parameters

hif	- open interface handle on the device
pfsMaskOut	- variable to receive output mask
pfsMaskIn	- variable to receive input mask

This function is used to determine which GPIO pins are available on the port currently enabled on the specified interface handle. It returns two 32 bit masks indicating which bits in the port are supported for input, output, or both. The mask returned in *pfsMaskOut* will have the bit set for each bit position that can be used for output. The mask returned in *pfsMaskIn* will have the bit set for each bit position that can be used for input. Bi-directional pins will have the corresponding bit set in each mask.

DjtgSetGpioDir (HIF hif, DWORD fsDirReq DWORD * pfsDirSet)

Parameters

hif	- open interface handle on the device
fsDirReq	- requested pin direction mak
pfsDirSet	- variable to receive pin directions actually set

This function is used to set the GPIO pin direction (input/output) for the DJTG port currently enabled on the specified interface handle. The bit mask in fsDirReq has the bit set to 1 for each pin to be configured as an output, and 0 for pins to be configured as inputs. Bits corresponding to unsupported pins should be set to 0. The value returned in pfsDirSet indicates the resulting state of the pin directions.

DjtgGetGpioDir (HIF hif, DWORD * pfsDir)

Parameters

hif	- open interface handle on the device
pfsDir	- variable specifying pin directions

This function will return the current setting of the pin directions for the GPIO pins on the DJTG port currently enabled on the specified interface handle. The value returned in *pfsDir* will have the bit set to 1 for each pin currently configured as an output. The bit will be 0 for pins that are either configured as inputs or not supported by the port.

DjtgSetGpioState (HIF hif, DWORD fsState)

Parameters

hif	- open interface handle on the device
fsState	- variable specifying output pin states to set

This function is sets the state of GPIO pins configured as outputs on the DJTG port currently enabled on the specified interface handle. Each GPIO pin configured as an output will be set to 1 or 0 depending on the value of the corresponding bit in *fsState*. Bits corresponding to input pins or unsupported pins will be ignored and should be set to 0.

DjtgGetGpioState (HIF hif, DWORD * pfsState)

Parameters

hif	- open interface handle on the device
pfsState	- variable to receive current input pin state

This function returns the current state of the GPIO pins associated with DJTG port currently enabled on the specified interface handle. The current pin state is returned in *pfsState*. Pins that are configured as inputs will have the corresponding bit set to the state of the input condition at the pin. Pins that are configured as outputs will have the corresponding bit set to the value last written to the pin. Any bits corresponding to unsupported pins will be set to 0.

DjtgPutTdiBits(HIF hif, BOOL fTms, BYTE * rgbSnd, BYTE * rgbRcv, DWORD cbits, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fTms	- value theTMS pin will be held at during shifting (true = 1, false = 0)
rgbSnd	- buffer containing TDI bits. Bits are shifted in sequentially starting at the first element in the array, from LSB to MSB.
rgbRcv	-buffer to hold TDO bits. If not used, set to NULL.
cbits	- number of TDI bits to clock into the TAP controller
fOverlap	- TRUE if operation should be overlapped

This function shifts the specified number of bits into TDI and (optionally) returns bits shifted out of TDO. The Buffer rgbSnd supplies the bits to be shifted into TDI. Each bit is shifted in sequentially, starting at the first element in the array, from least significant bit to most significant bit (i.e. shifted to the right). Below is an example of how the TDI bits are placed in each byte of rgbSnd.

rgbSnd[0]

TDI 8	TDI 7	TDI 6	TDI 5	TDI 4	TDI 3	TDI 2	TDI 1

rgbSnd[1]

TDI 16	TDI 15	TDI 14	TDI 13	TDI 12	TDI 11	TDI 10	TDI 9

If rgbRcv is not NULL, all bits shifted out of TDO are returned and stored in the buffer specified by rgbRcv. Each bit is shifted out sequentially starting at the first element in the rgbRcv, from lowest significant bit to most significant bit. TMS is held at the value specified by fTms while bits are being shifted into TDI.

DjtgPutTmsBits(HIF hif, BOOL fTdi, BYTE * rgbSnd, BYTE * rgbRcv, DWORD cbits, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fTdi	- value the TDI pin will be held at during shifting (true = 1, false = 0)
rgbSnd	- buffer containing TMS bits. Bits are shifted in sequentially starting at the first element in the array, from LSB to MSB.
rgbRcv	-buffer to hold TDO bits. If not used, set to NULL.
cbits	- number of TDI bits to clock into the TAP controller
fOverlap	- TRUE if operation should be overlapped

This function shifts a specified number of bits into TMS and (optionally) returns bits shifted out of TDO. The buffer rgbSnd specifies the bits to be shifted into TMS. Each bit is shifted in sequentially, starting at the first element in the array, from least significant bit to most significant bit. Below is an example of how the TMS bits are placed in each byte of rgbSnd.

rgbSnd[0]

TMS 8	TMS 7	TMS 6	TMS 5	TMS 4	TMS 3	TMS 2	TMS 1

rgbSnd[1]

TMS 16	TMS 15	TMS 14	TMS 13	TMS 12	TMS 11	TMS 10	TMS 9

If fReturnTdo is set to true, all bits shifted out of TDO are stored in rgbRcv. Each bit is shifted out sequentially starting at the first element in the rgbRcv, from lowest significant bit to most significant bit. TDI is held at the value specified by fTdi while bits are being shifted into TMS.

DjtgPutTmsTdiBits(HIF hif, BYTE * rgbSnd, BYTE * rgbRcv, DWORD cbitpairs, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
rgbSnd	- buffer containing TMS bits. Bits are shifted in sequentially starting at the first element in the array, from LSB to MSB.
rgbRcv	-buffer to hold TDO bits. If not used, set to NULL.
cbitpairs	- number ofTMS and TDI bit pairs
fOverlap	- TRUE if operation should be overlapped

This function shifts a specified number of bit pairs into TMS and TDI and (optionally) returns bits shifted out of TDO. The buffer rgbSnd specifies the bit pairs to be shifted intoTMS and TDI. Each bit pair is shifted in sequentially; starting at the first element in the array, from least significant bit pair to most significant bit pair. In each pair, the TMS value is the MSB and the TDI value is the LSB. Below is an example of how the TMS/TDI bit pairs are placed in each byte of rgbSnd.

rgbSnd[0]

TMS 4	TDI 4	TMS 3	TDI 3	TMS 2	TDI 2	TMS 1	TDI 1

rgbSnd[1]

TMS 8	TDI 8	TMS 7	TDI 7	TMS 6	TDI 6	TMS 5	TDI 5

If fReturnTdo is set to true, all bits shifted out of TDO are stored in rgbRcv. Each bit is shifted out sequentially starting at the first element in the rgbRcv, from lowest significant bit to most significant bit.

DjtgClockTck(HIF hif, BOOL fTms, BOOL fTdi, DWORD cclk, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fTdi	- value the TDI pin will be held at during shifting (true = 1, false = 0)
fTms	- value the TMS pin will be held at during shifting (true = 1, false = 0)
cclk	- number of cycles to clock TCK.
fOverlap	- TRUE if operation should be overlapped

This function clocks TCK a specified number of cycles. During this time, TDI and TMS are held at the specified values. TDO is not sampled.

DjtgWait(HIF hif, DWORD tusWait, DWORD * ptusWaited)

Parameters

hif	- open interface handle on the device
tusWait	- number of microseconds to wait (delay)
ptusWaited	- pointer to variable to return the number of microseconds actually waited

This function uses the underlying hardware of the DJTG port to wait for a specified number of microseconds. Some DJTG ports may place a restriction on the maximum amount of time that they are willing to delay. The maximum amount of delay supported by the port may be determined by passing in a value of 0 for the tusWait parameter. This function is not supported unless the dprpJtgWait bit is set in the port properties for the port.

DjtgBatch(HIF hif, DWORD cbSnd, BYTE * rgbSnd, DWORD cbRcv, BYTE * rgbRcv, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
cbSnd	- number of data bytes to send
rgbSnd	- pointer to data to send
cbRcv	- number of data bytes to be returned
rgbRcb	- buffer to receive returned data
fOverlap	- TRUE if operation should be overlapped

This function is used to initiate batch transactions on the JTAG scan chain associated with the JTG port currently enabled on the specified interface handle. The *rgbSnd* buffer contains a batch command script specifying a series of transactions to occur. The device will interpret the contents of the buffer, performing the requested transactions.

The *cbSnd* and *rgbSnd* parameters specify the batch command buffer to be sent for execution. The *cbRcv* and *rgbRcv* parameters specify the expected number of bytes and the buffer to receive the bytes to be returned from the device as a result of executing the commands in the batch script.

See the DJTG Batch section of this document for a more complete description of JTG batch operations and the format of the batch command buffer. This function is not supported unless the dprpJtgBatch bit is set in the port properties for the port.

IEEE 1149.7-2009 Addendum

The sections that follow describe a number of functions and definitions that have been added to the DJTG interface to facilitate communication with devices that conform to the IEEE 1149.7-2009 specification.

DJTG Port Properties Additions

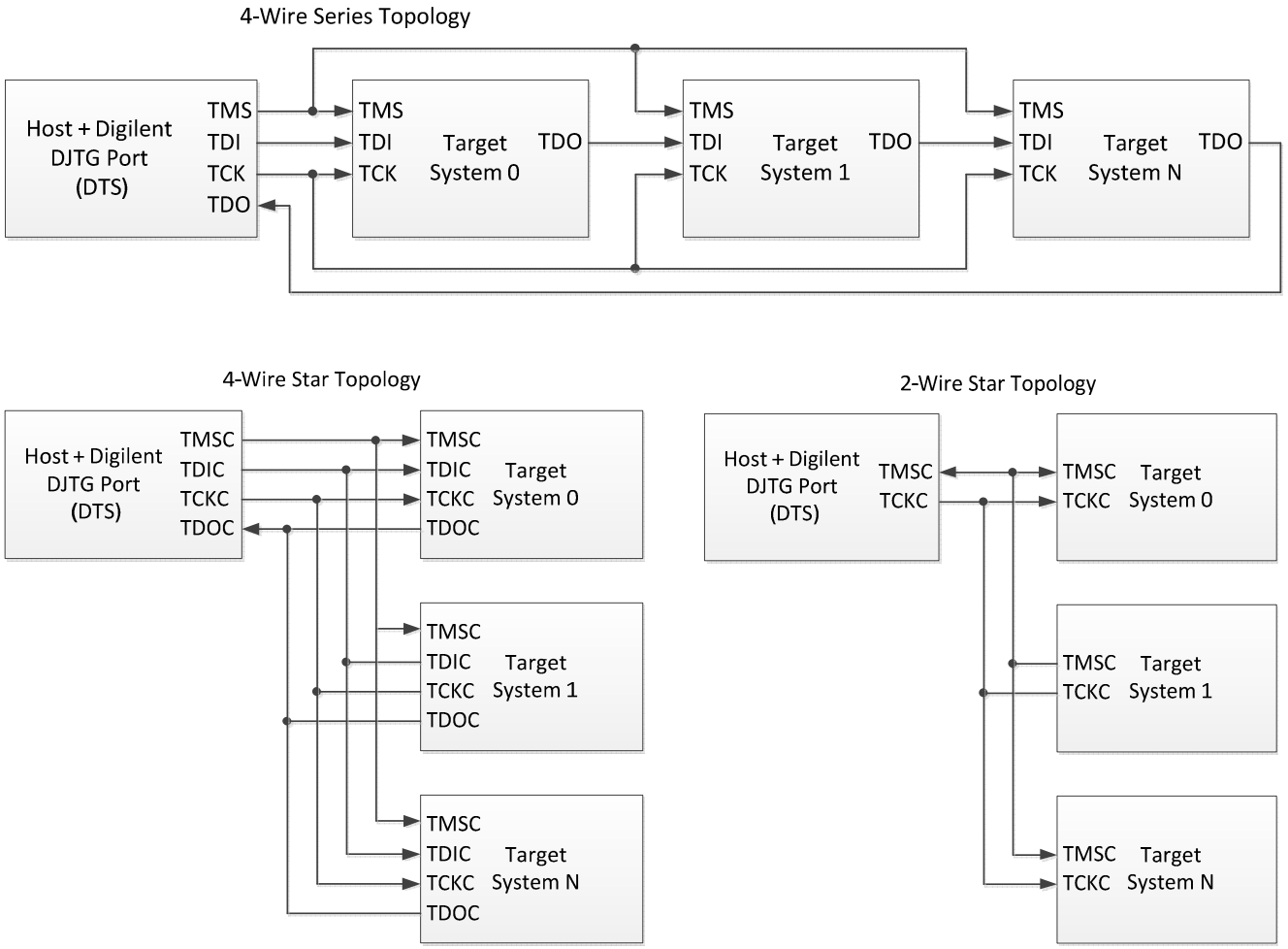
The following port properties bits have been added to djtg.h:

dprpJtgMScan	This bit indicates that the MScan format is supported by the port.
dprpJtgOScan0	This bit indicates that the OScan0 format is supported by the port.
dprpJtgOScan1	This bit indicates that the OScan1 format is supported by the port.
dprpJtgOScan2	This bit indicates that the OScan2 format is supported by the port.
dprpJtgOScan3	This bit indicates that the OScan3 format is supported by the port.
dprpJtgOScan4	This bit indicates that the OScan4 format is supported by the port.
dprpJtgOScan5	This bit indicates that the OScan5 format is supported by the port.
dprpJtgOScan6	This bit indicates that the OScan6 format is supported by the port.
dprpJtgOScan7	This bit indicates that the OScan7 format is supported by the port.
dprpJtgDelayCnt	This bit indicates that the *DjtgSetDelayCnt* and *DjtgGetDelayCnt* functions are supported by the port.
dprpJtgReadyCnt	This bit indicates that the *DjtgSetReadyCnt* and *DjtgGetReadyCnt* functions are supported by the port.
dprpJtgEscape	This bit indicates that the *DjtgEscape* function is supported by the port.
dprpJtgCheckPacket	This bit indicates that the DjtgCheckPacket function is supported by the port.

Scan Topologies

The IEEE 1149.7-2009 specification defines three different topologies that may be used to construct a scan chain: 4-Wire Series, 4-Wire Star, and 2-Wire Star. All Digilent devices that expose a DJTG port are capable of interfacing with a 4-Wire Series Scan Topology. The DJTG port(s) exposed by some Digilent devices are also capable of interfacing with 2-Wire and 4-Wire Star Scan Topologies.

A device that supports *DjtgEscape* function is capable of interfacing with a 4-Wire Star Scan Topology. A device that supports any of the advanced scan formats (MScan, OScan, SScan) is capable of interfacing with a 2-Wire Star Scan Topology. Use the *DjtgGetPortProperties* function to determine if a particular port is capable of interfacing the desired scan topology.
The diagrams below depict the use of a Digilent DJTG port in each of the different scan topologies. Please note that the TMSC, TDIC, TDOC, and TCKC signals share the same physical pins as the TMS, TDI, TDO, and TCK signals.



Scan Format

Scan format is used to describe how data is transmitted between the Data Test System (DTS) and the Target System (TS). Some scan formats can use all four of the primary JTAG signals (TMSC, TCKC, TDIC, and TDOC) to transmit information between the DTS and TS while others only use the TMSC and TCKC signals.

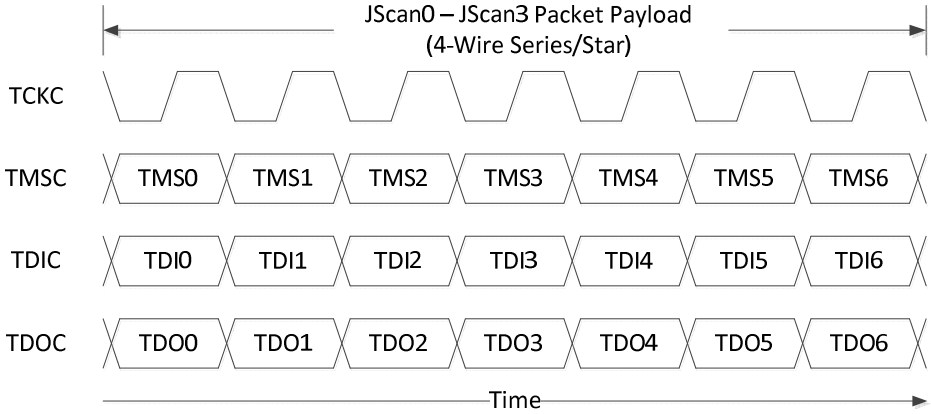
All devices that are compatible with Digilent's JTAG interface support the JScan formats. Some devices also support the MScan and OScan0 – OScan7 formats. Devices that support the MScan and/or OScan formats default to the JScan0 format and may be instructed to transmit data (scan packets) using a different format by calling the *DjtgSetScanFormat* function.

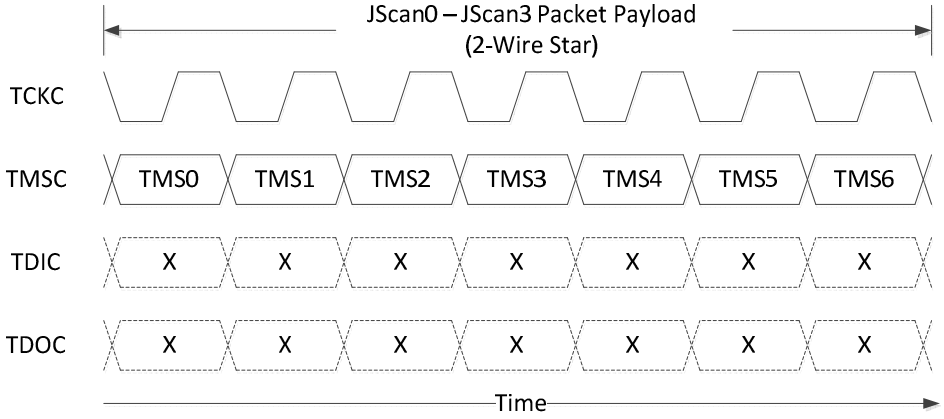
The following symbols defined in djtg.h are used to specify the scan format when calling *DjtgSetScanFormat*:

jtgsfJScan0	JScan0
jtgsfJScan1	JScan1. This is the same as setting JScan0.
jtgsfJScan2	JScan2. This is the same as setting JScan0.
jtgsfJScan3	JScan3. This is the same as setting JScan0.

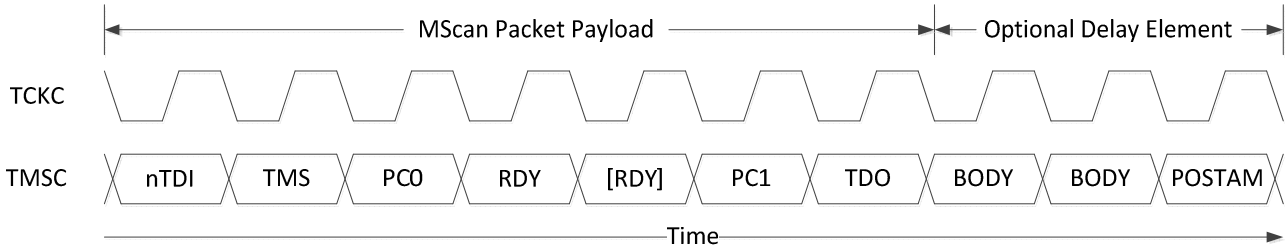
jtgsfMScan	MScan
jtgsfOScan0	OSCan0
jtgsfOScan1	OSCan1
jtgsfOScan2	OSCan2
jtgsfOScan3	OSCan3
jtgsfOScan4	OSCan4
jtgsfOScan5	OSCan5
jtgsfOScan6	OSCan6
jtgsfOScan7	OSCan7

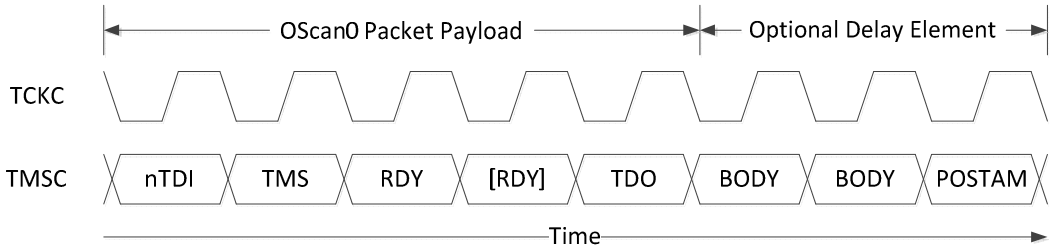
The diagrams that follow depict how data is transmitted between the DTS and TS for each of the different scan format settings. Please note that the packet payload of some scan formats change based on the state of the tap controller.

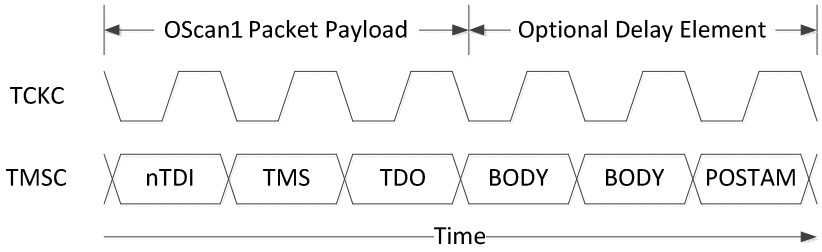


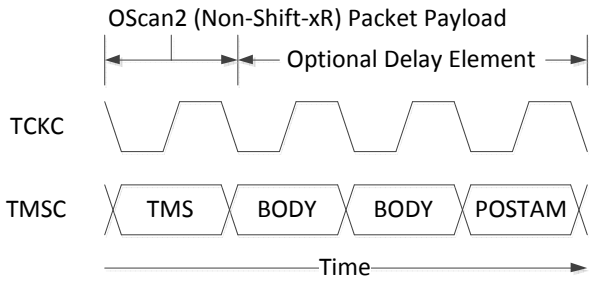


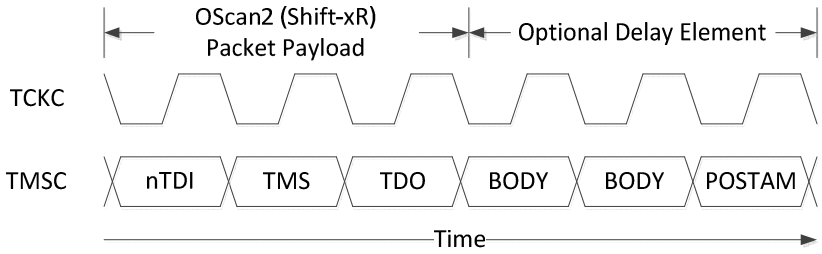
Note: The TDIC and TDOC signals are shown because they are still physically present on the DJTG port. However, when operating in a 2-Wire Star Topology their levels don't matter because these pins aren't connected to the TS.

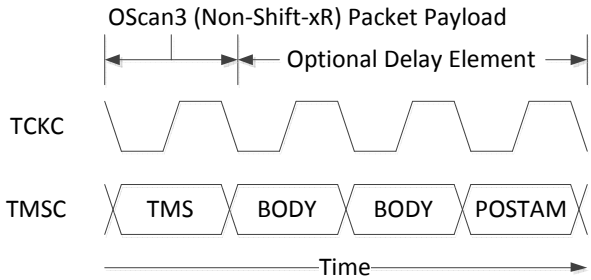


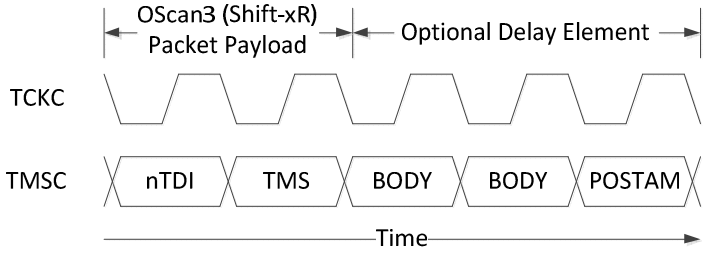


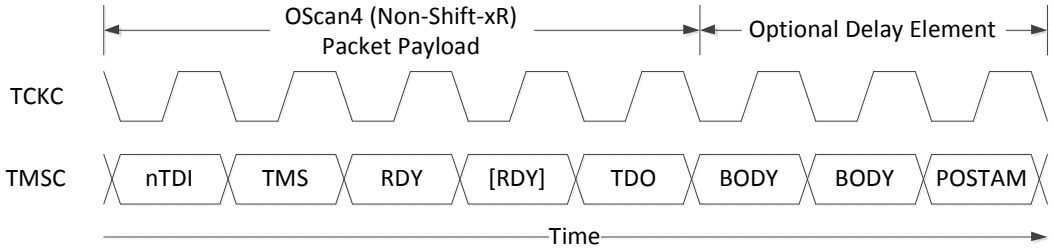


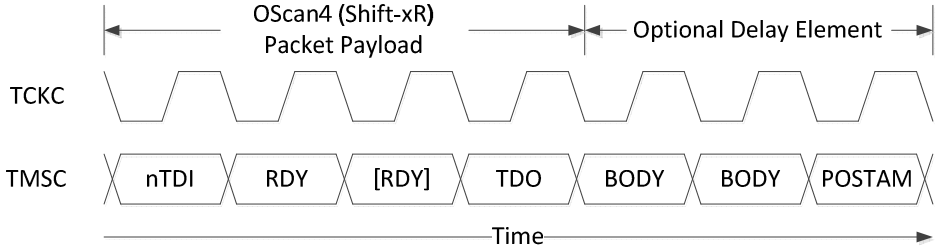


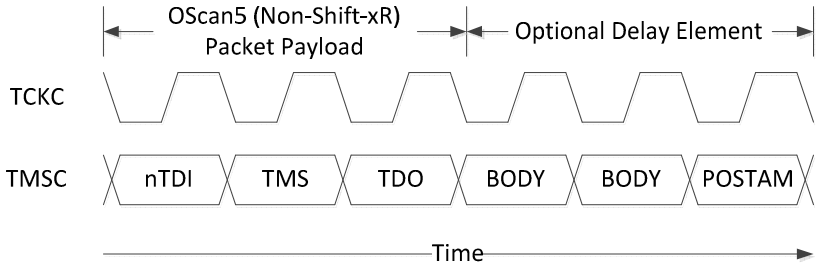


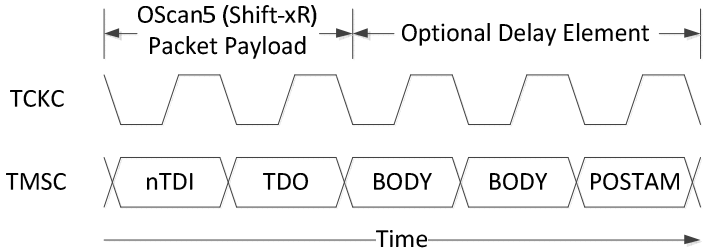


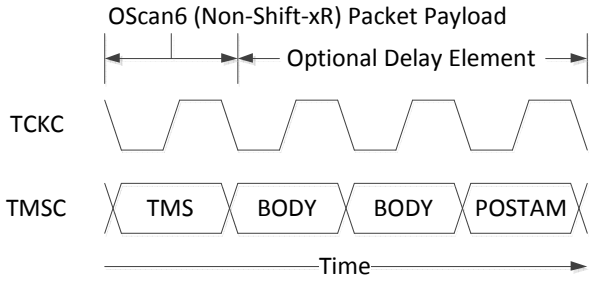


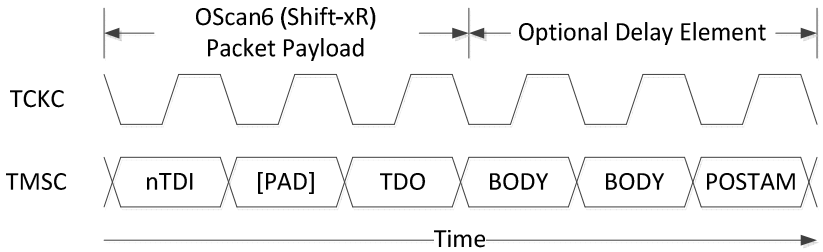


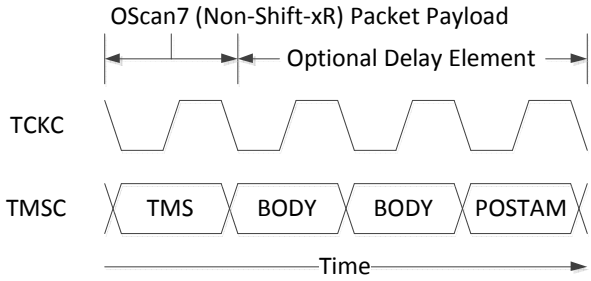


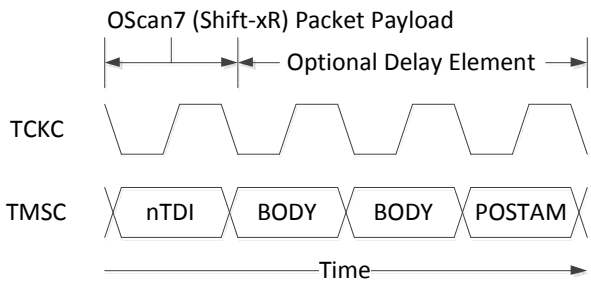












DJTG API Function Additions

DjtgSetScanFormat(HIF hif, BYTE jtgsfFmt, BOOL fShiftXR)

Parameters

hif	- open interface handle on the device
jtgsfFmt	- scan format to set
fShiftXR	- current state of the tap controller, TRUE for shift-XR states, FALSE otherwise

This function specifies the scan format used by the DJTG port (DTS) while communicating with the Target System (TS). Some scan formats (Oscan3-OScan7) have a different payload when the tap controller is in a Shift-XR state. For these scan formats, the fShiftXR parameter is used to specify the state of the tap controller. The fShiftXR parameter is ignored for all other scan formats. This function is only supported by devices that advertise support for the MScan, OScan, or SScan formats. Call DjtgGetPortProperties to determine which scan formats the device supports.

Please note that calls made to this function do not cause any changes to be made to the SCNFMT register of the TS. Prior to calling this function DjtgPutTmsBits should be used to generate the bit sequence required to configure the SCNFMT register of the TS with a value between 0 and 16. For more information about scan formats see sections 23.4.1.4.5, 23.4.2, 24, 25, and 26 of the IEEE 1149.7-2009 specification.

DjtgGetScanFormat(HIF hif, BYTE* pjtgsfFmt, BOOL* pfShiftXR)

Parameters

hif	- open interface handle on the device
pjtgsfFmt	- pointer to a variable to receive the current scan format
pfShiftXR	- pointer to a variable to receive the current shift-XR state

This function is used to get the scan format that's currently associated with the DJTG port.

DjtgSetReadyCnt(HIF hif, BYTE cbitRdy, DWORD* pcretOutReq, DWORD* pcretOutSet)

Parameters

hif	- open interface handle on the device
cbitRdy	- number of expected RDY bits (1-4, RDYC + 1)
pcretOutReq	- pointer to a variable specifying the requested number of times to retry the output portion of a bit frame
pcretOutSet	- pointer to a variable to receive the actual number of times that the output portion of a bit frame will be retried

This function is used to set the number of ready (RDY) bits that the DJTG port (DTS) should expect to be present in the output portion of a scan packet's payload. The meaning of cbitRdy depends on the scan format that is currently select. When the MScan format is selected cbitRdy specifies the number of RDY bits that are expected to follow each precharge (PC0/PC1) bit. When an OScan or SScan format that contains RDY bits is selected, cbitRdy sepecifies the number of RDY='1' bits that are expected to preced the TDO bit.

The TS may stall the completion of the output portion of a bit frame by returning '0' for any RDY bit that's shifted as part of the bit frame. In the event that the TS returns a RDY='0' bit the DJTG port will re-attempt to complete the output portion of the bit frame up to the number of times specified by the

variable pointed to by pcrefOutReq before aborting the transaction. If NULL is specified for pcrefOutReq then the retry count will be set to the default value, which is device specific.

Please note that calls made to this function do not cause any changes to be made the RDYC register of the TS. Prior to calling this function DjtgPutTmsBits should be used to generate the bit sequence required to configure the RDYC register of the TS with a value between 0 and 3. This function should then be called with cbitRdy specified as RDYC + 1.For more information see sections 22.2.4, 23.4.1.4.3, 23.4.2, 24.4.1.3, 24.4.2, 25.4.1.6, and 25.4.2 of the IEEE 1149.7-2009 specification.

Scan formats that do not require ready bit detection will not be affected by calls made to this function.

DjtgGetReadyCnt(HIF hif, BYTE* pcbitRdy, DWORD* pcretOut)

Parameters

hif	- open interface handle on the device
pcbitRdy	- pointer to a variable to receive the count of expected ready bits
pcretOut	- pointer to return the number of times that the device will re-attempt to complete the output portion of a bit frame after receiving a RDY='0' bit

This function is used to get the number of ready bits that the DJTG port (DTS) expects to be present in the output portion of a scan packet's payload as well as the number of times that the port should re-attempt to complete the bit frame after receiving a RDY='0' bit.

DjtgSetDelayCnt(HIF hif, DWORD cbitDlyReq , DWORD* pcbitDlySet, BOOL fReset)

Parameters

hif	- open interface handle on the device
cbitDlyReq	- number of delay bits (clocks) to insert
pcbitDlySet	- pointer to return the actual number of delay bits
fReset	- terminate variable length delay element with DLY_RSO or DLY_END directive (fTrue/fFalse)

This function is used to set the number of delay bits (clocks) inserted between scan packets. When cbitDly is specified as non-zero a delay element that consists of up to cbitDly clocks will be output immediately following each scan packet's payload. The default delay count is zero. Specifying a value greater than two for cbitDly results in a variable length delay being performed. Variable length delays are terminated with one of two delay directives: DLY_RSO or DLY_END. When a varible length delay is terminated with DLY_RSO the TS performs a type 3 reset.

The number of delay bits that can be inserted between each scan packet depends on the current configuration of the DLYC register. When the DLYC register is configured with a value of 0 the TS does not expect any delay clocks to follow the last bit of a scan packets payload. When the DLYC register is configured with a value of 1 or 2 the TS expects one or two delay clocks between each scan packet. When the DLYC register is configured with a value of 3 the TS expects a variable length delay that consists of a minimum of three clocks.

Please note that calls made to this function do not cause any changes to be made the DLYC register of the TS. Prior to calling this function DjtgPutTmsBits should be used to generate the bit sequence required to configure the DLYC register of the TS with a value between 0 and 3. Also note that the delay elements can only be inserted between scan packets associated with the advanced protocol. As a result, a device that has been configured with a non-zero delay count will not insert any delay clocks

between scan packets whose format are JScan0, JScan1, JScan2, or JScan3. For more information see section 22.2.3.3 of the IEEE 1149.7-2009 specification.

DjtgGetDelayCnt(HIF hif, DWORD* pcbitDly, BOOL* pfReset)

Parameters

hif	- open interface handle on the device
pcbitDly	- pointer to return the count of delay bits (clocks)
pfReset	- pointer to return the termination directive associated with a variable length delay (fTrue = DLY_RSO, fFalse = DLY_END)

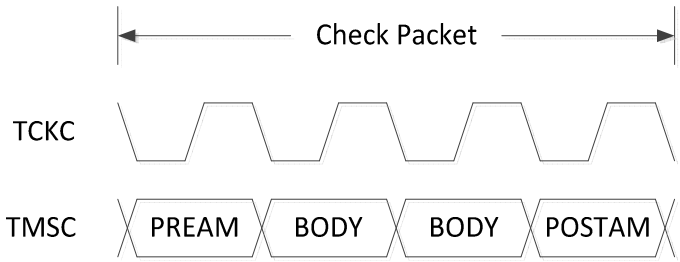
This function is used to get the count of delay bits (clocks) that make up the delay element that is appended to the output of each scan packet.

DjtgCheckPacket(HIF hif, BYTE cedgeNop, BOOL fReset, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
cedgeNop	- number of additional body bits to extend the check packet using the CP_NOP directive
fReset	- terminate the check packet with CP_RSO or CP_END (fTrue/fFalse)
fOverlap	- TRUE if operation should be overlapped

This function is used to send a Check Packet to the target system (TS). A Check Packet is sent to the TS by outputting a preamble bit, two or more body bits, and a postamble bit on the TMSC pin.

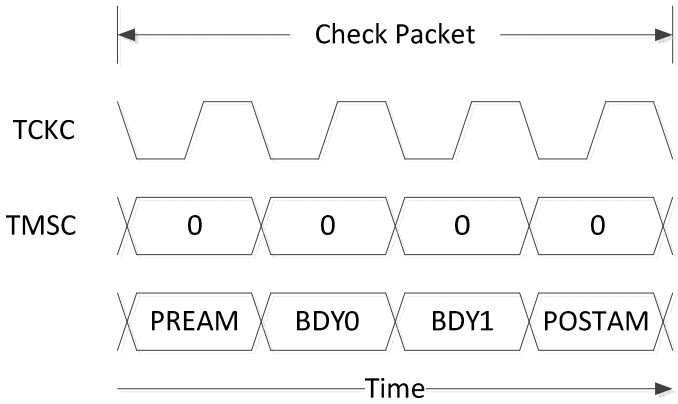


The TS processes the two most recently received body bits as one of three directives: CP_END ("00"), CP_RSO ("11"), and CP_NOP("01" or "10"). Each CP_NOP directive extends the body of the check packet by one bit period. The cedgeNop parameter is used to specify the number of CP_NOP directives that are sent to the TS before a CP_END or CP_RSO directive is used to terminate the Check Packet.

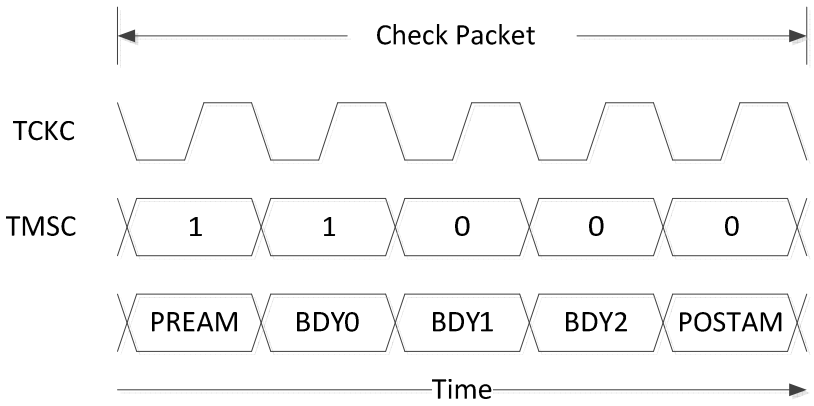
The fReset parameter determines whether a Check Packet is terminated using the CP_RSO (fTrue) directive or the CP_END (fFalse) directive. When a Check Packet is terminated with a CP_RSO directive, a Type-3 reset occurs following the postamble bit.

The examples on the pages that follow demonstrate the bit sequences output on the TMSC pin for several different calls to DjtgCheckPacket.

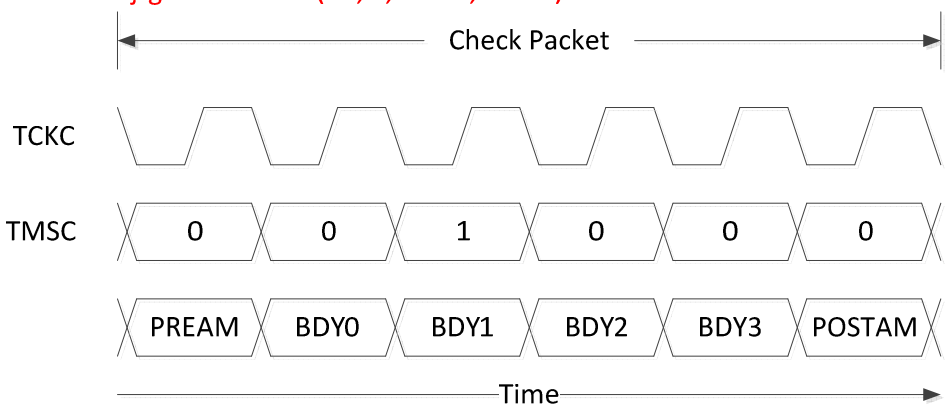
DjtgCheckPacket(hif, 0, fFalse, fFalse)



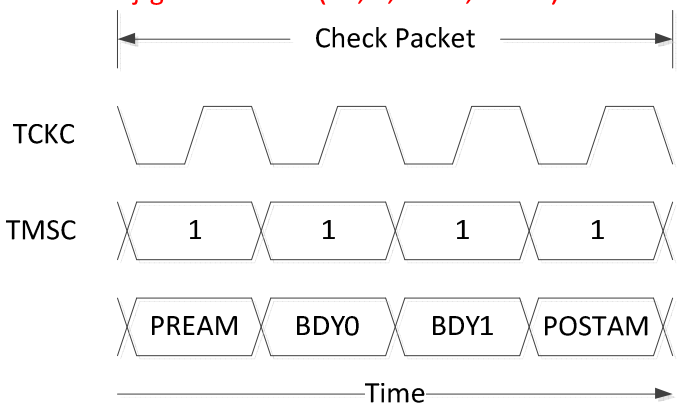
DjtgCheckPacket(hif, 1, fFalse, fFalse)



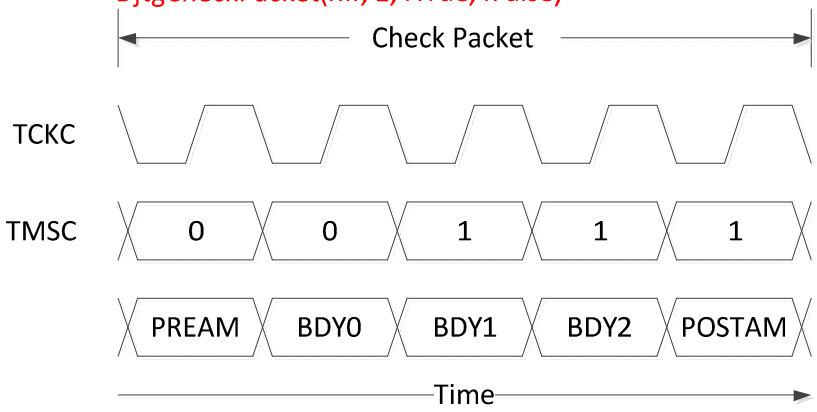
DjtgCheckPacket(hif, 2, fFalse, fFalse)



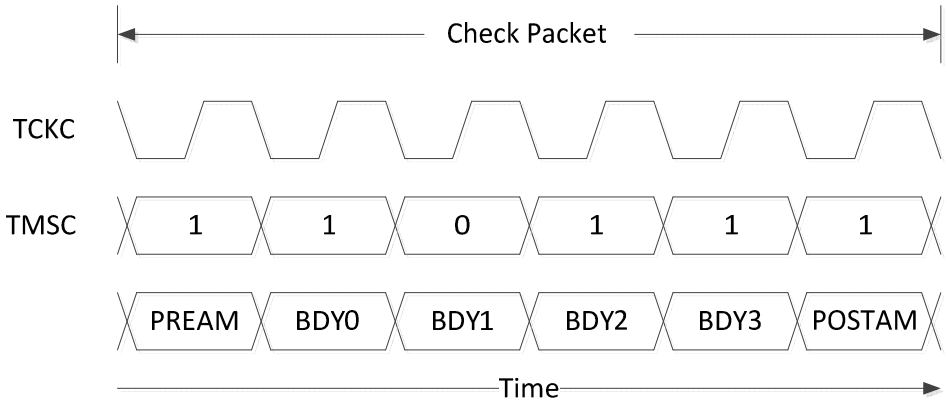
DjtgCheckPacket(hif, 0, fTrue, fFalse)



DjtgCheckPacket(hif, 1, fTrue, fFalse)



DjtgCheckPacket(hif, 2, fTrue, fFalse)



DjtgEscape(HIF hif, BYTE cedgeEsc, BOOL fOverlap)

Parameters

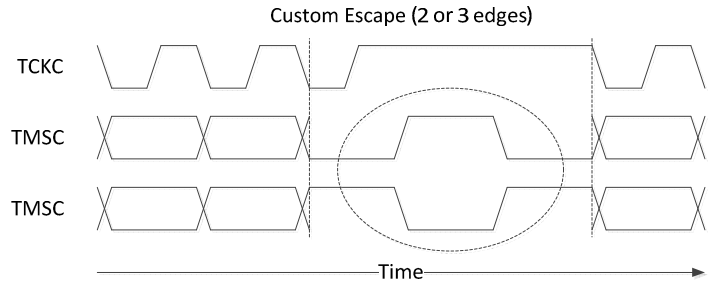
hif	- open interface handle on the device
cedgeEsc	- number of edges required to perform the escape
fOverlap	- TRUE if operation should be overlapped

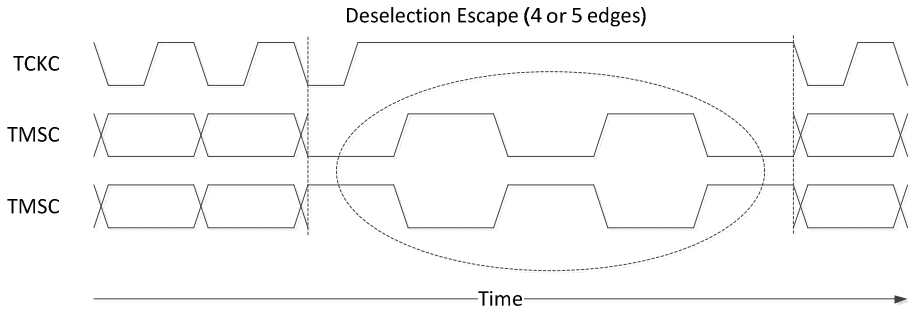
This function performs a special signal sequence know as an escape. An escape is primarily used in two wire (compact) JTAG mode to relay control information to the tap controller. Escapes are performed by toggling the TMSC signal while the TCKC signal is a '1'. This function sets TCKC to a '1', toggles the TMSC signal cedgeEsc times, and then sets TCKC to a '0'. Specifying a non-even number for cedgeEsc will result in an error being returned.

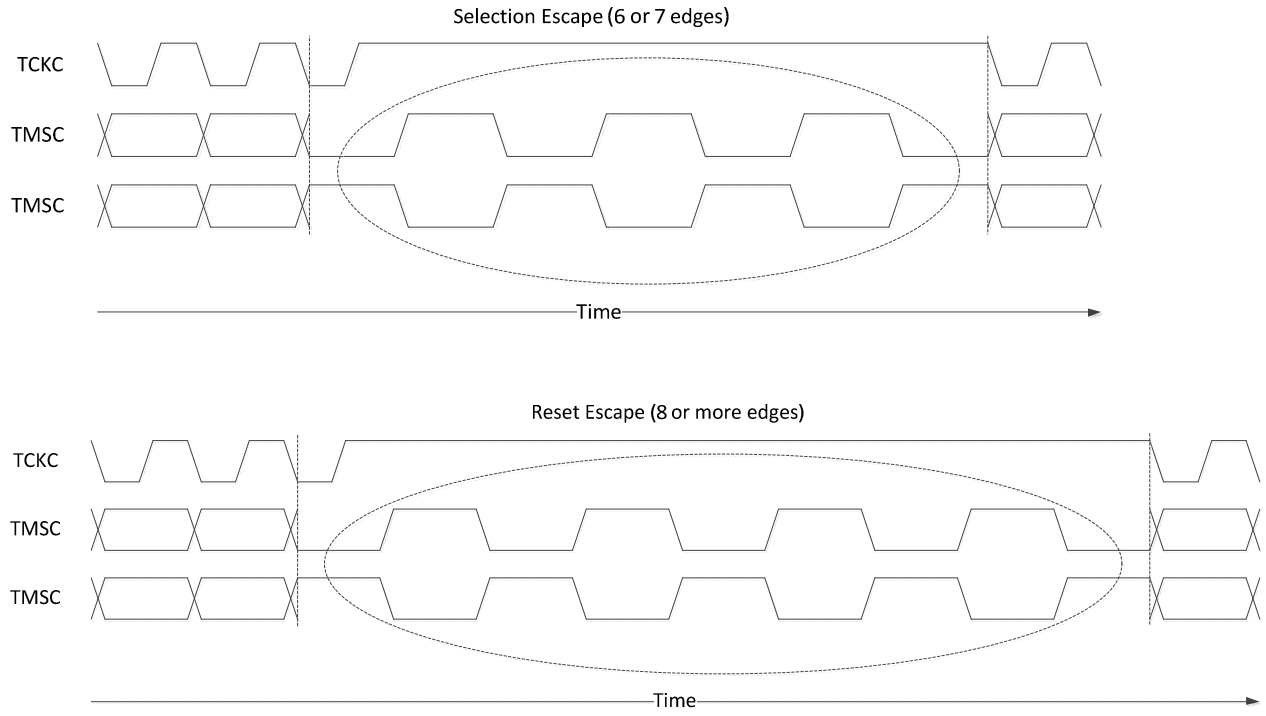
The following symbols defined in djtg.h may be used to specify the number of edges required to perform an escape sequence:

cedgeJtgCustom	Custom Escape
cedgeJtgDeselect	Deselection Escape
cedgeJtgSelect	Selection Escape
cedgeJtgReset	Reset Escape

The diagrams that follow depict the signaling sequences generated by *DjtgEscape* when cedgeJtgCustom, cedgeJtgDeselect, cedgeJtgSelect, or cedgeJtgReset are specified for cedgeEsc. For more information on escapes see sections 4.3.1, 10.4.1.1, 10.4.1.2, 10.4.1.3, 10.4.1.4, 10.4.1.5, and 10.4.2 of the IEEE 1149.7-2009 specification.







Use of Legacy API Functions with 1149.7-2009

DjtgSetTmsTdiTck(HIF hif, BOOL fTms, BOOL fTdi, BOOL fTck)

Parameters

hif	- open interface handle on the device
fTms	- sets the state of the TMS pin
fTdi	- sets the state of the TDI pin
fTck	- sets the state of the TCK pin

This function is not supported while the device is configured for any scan format other than JScan0, JScan1, JScan2, or JScan3. When the DJTG port is placed in a Star-2 scan topology the TDI and TDO pins aren't connected to the TS. As a result, changes to the state of the TDI pin should not be visible to the TS.

DjtgGetTmsTdiTdoTck(HIF hif, BOOL* pfTms, BOOL* pfTdi, BOOL* pfTdo, BOOL* pfTck)

Parameters

hif	- open interface handle on the device
pfTms	- pointer to return current state of TMS pin (true = 1, false = 0)
pfTdi	- pointer to return current state of TDI pin (true = 1, false = 0)
pfTdo	- pointer to return current state of TDO pin (true = 1, false = 0)
pfTck	- pointer to return current state of TCK pin (true = 1, false = 0)

This function is not supported while the device is configured for any scan format other than JScan0, JScan1, JScan2, or JScan3. When the DJTG port is placed in a Star-2 scan topology the TDI and

TDO pins aren't connected to the TS. As a result, the values returned for these pins provide no meaningful information to the caller.

DjtgPutTdiBits(HIF hif, BOOL fTms, BYTE * rgbSnd, BYTE * rgbRcv, DWORD cbits, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fTms	- value theTMS pin will be held at during shifting (true = 1, false = 0)
rgbSnd	- buffer containing TDI bits. Bits are shifted in sequentially starting at the first element in the array, from LSB to MSB.
rgbRcv	-buffer to hold TDO bits. If not used, set to NULL.
cbits	- number of TDI bits to clock into the TAP controller
fOverlap	- TRUE if operation should be overlapped

When the DJTG port is configured for JScan0, JScan1, JScan2, or JScan3 the fTms parameter specifies the state at which the TMSC pin is held for the duration of the data transfer. When the DJTG port is configured for any other scan format the fTms parameter specifies whether a '1' or a '0' is output on the TMSC pin for each TMS bit in a scan packet. If the scan format that the port is currently configured for doesn't contain any TMS bits then the fTms parameter is ignored.

Regardless of the scan format configuration, the rgbSnd parameter specifies a buffer containing the sequence of the TDI bits that should be shifted. The TDI bits contained in rgbSnd are automatically complemented as necessary. For example, if the current scan format is OScan1 and rgbSnd[0] = 0x91 then 0x6E will be shifted for the nTDI bits. For more information regarding the format of the data shifted by this function, see the section titled "Scan Formats."

The table below describes the circumstances under which DjtgPutTdiBits may be called. Calling DjtgPutTdiBits under any other circumstance may result in an error being returned.

Scan Format	TAP Controller State	Option to return TDO Bits
JScan0 – JScan3	All	Yes
MScan	All	Yes
OScan0	All	Yes
OScan1	All	Yes
OScan2	Shift-xR only	Yes
OScan3	Shift-xR only	No
OScan4	All	Yes
OScan5	All	Yes
OScan6	Shift-xR only	Yes
OScan7	Shift-xR only	No

DjtgPutTmsBits(HIF hif, BOOL fTdi, BYTE * rgbSnd, BYTE * rgbRcv, DWORD cbits, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fTdi	- value the TDI pin will be held at during shifting (true = 1, false = 0)
rgbSnd	- buffer containing TMS bits. Bits are shifted in sequentially starting at the first element in the array, from LSB to MSB.
rgbRcv	-buffer to hold TDO bits. If not used, set to NULL.
cbits	- number of TDI bits to clock into the TAP controller
fOverlap	- TRUE if operation should be overlapped

When the DJTG port is configured for JScan0, JScan1, JScan2, or JScan3 the fTdi parameter specifies the state at which the TDIC pin is held for the duration of the data transfer. When the DJTG port is configured for any other scan format the fTdi parameter specifies whether a '1' or a '0' is output on the TMSC pin for each TDI bit in a scan packet. If the scan format that the port is currently configured for doesn't contain any TDI bits then the fTdi parameter is ignored.

The state of fTdi is automatically complemented for scan packets that require nTDI bits to be output on the TMSC pin. For example, if the current scan format is OScan1 and fTdi is specified as true then a '0' will be output for each nTDI bit that's shifted. For more information regarding the format of the data shifted by this function, see the section titled "Scan Formats."

The table below describes the circumstances under which DjtgPutTmsBits may be called. Calling DjtgPutTmsBits under any other circumstance may result in an error being returned.

Scan Format	TAP Controller State	Option to return TDO Bits
JScan0 – JScan3	All	Yes
MScan	All	Yes
OScan0	All	Yes
OScan1	All	Yes
OScan2	All	Yes, in Shift-xR state only
OScan3	All	No
OScan4	Non-Shift-xR only	Yes
OScan5	Non-Shift-xR only	Yes
OScan6	Non-Shift-xR only	No
OScan7	Non-Shift-xR only	No

DjtgPutTmsTdiBits(HIF hif, BYTE * rgbSnd, BYTE * rgbRcv, DWORD cbitpairs, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
rgbSnd	- buffer containing TMS bits. Bits are shifted in sequentially starting at the first element in the array, from LSB to MSB.
rgbRcv	-buffer to hold TDO bits. If not used, set to NULL.
cbitpairs	- number ofTMS and TDI bit pairs
fOverlap	- TRUE if operation should be overlapped

Regardless of the scan format configuration, the rgbSnd parameter specifies a buffer containing the sequence of the TDI and TMS bits that should be shifted. The TDI bits contained in rgbSnd are automatically complemented as necessary. For example, if the current scan format is OScan1 and rgbSnd[0] = 0xAB and rgbSnd[1] = 0xEB then first byte of TDI bits is 0x91 and 0x6E will be shifted for the nTDI bits. For more information regarding the format of the data shifted by this function, see the section titled "Scan Formats."

The table below describes the circumstances under DjtgPutTmsTdiBits may be called. Calling DjtgPutTmsTdiBits under any other circumstance may result in an error being returned.

Scan Format	TAP Controller State	Option to return TDO Bits
JScan0 – JScan3	All	Yes
MScan	All	Yes
OScan0	All	Yes
OScan1	All	Yes
OScan2	Shift-xR only	Yes
OScan3	Shift-xR only	No
OScan4	Non- Shift-xR only	Yes
OScan5	Non- Shift-xR only	Yes
OScan6	None	No
OScan7	None	No

DjtgGetTdoBits(HIF hif, BOOL fTdi, BOOL fTms, BYTE * rgbRcv, DWORD cbits, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fTdi	- value the TDI pin will be held at during shifting (true = 1, false = 0)
fTms	- value the TMS pin will be held at during shifting (true = 1, false = 0)
rgbRcv	- buffer to hold TDO bits
cbits	- number of TMS and TDI bit pairs
fOverlap	- TRUE if operation should be overlapped

When the DJTG port is configured for JScan0, JScan1, JScan2, or JScan3 the fTdi and fTms parameters specify the state at which the TDIC and TMSC pins are held for the duration of the data transfer. When the DJTG port is configured for any other scan format the fTdi and fTms parameters specify whether a '1' or a '0' is output on the TMSC pin for each TDI and TMS bit in the scan packet. If the scan format that the port is currently configured for doesn't contain any TDI bits then the fTdi parameter is ignored. Similarly, if the scan format doesn't contain any TMS bits then the fTms parameter is ignored.

The state of fTdi is automatically complemented for scan packets that require nTDI bits to be output on the TMSC pin. For example, if the current scan format is OScan1 and fTdi is specified as TRUE then a '0' will be output for each nTDI bit that's shifted. For more information regarding the format of the data shifted by this function, see the section titled "Scan Formats."

The table on the following page describes circumstances under which DjtgGetTdoBits may be called. Calling DjtgGetTdoBits under any other circumstance may result in an error being returned.

Scan Format	TAP Controller State	Option to return TDO Bits
JScan0 – JScan3	All	Required
MScan	All	Required
OScan0	All	Required
OScan1	All	Required
OScan2	Shift-xR only	Required
OScan3	None	No
OScan4	All	Required
OScan5	All	Required
OScan6	Shift-xR only	Required
OScan7	None	No

DjtgClockTck(HIF hif, BOOL fTms, BOOL fTdi, DWORD cclk, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fTdi	- value the TDI pin will be held at during shifting (true = 1, false = 0)
fTms	- value the TMS pin will be held at during shifting (true = 1, false = 0)
cclk	- number of cycles to clock TCK.
fOverlap	- TRUE if operation should be overlapped

This function is not supported while the device is configured for any scan format other than JScan0, JScan1, JScan2, or JScan3.