XJTAG®

XJIO v2 Hardware

User Guide

Version 3

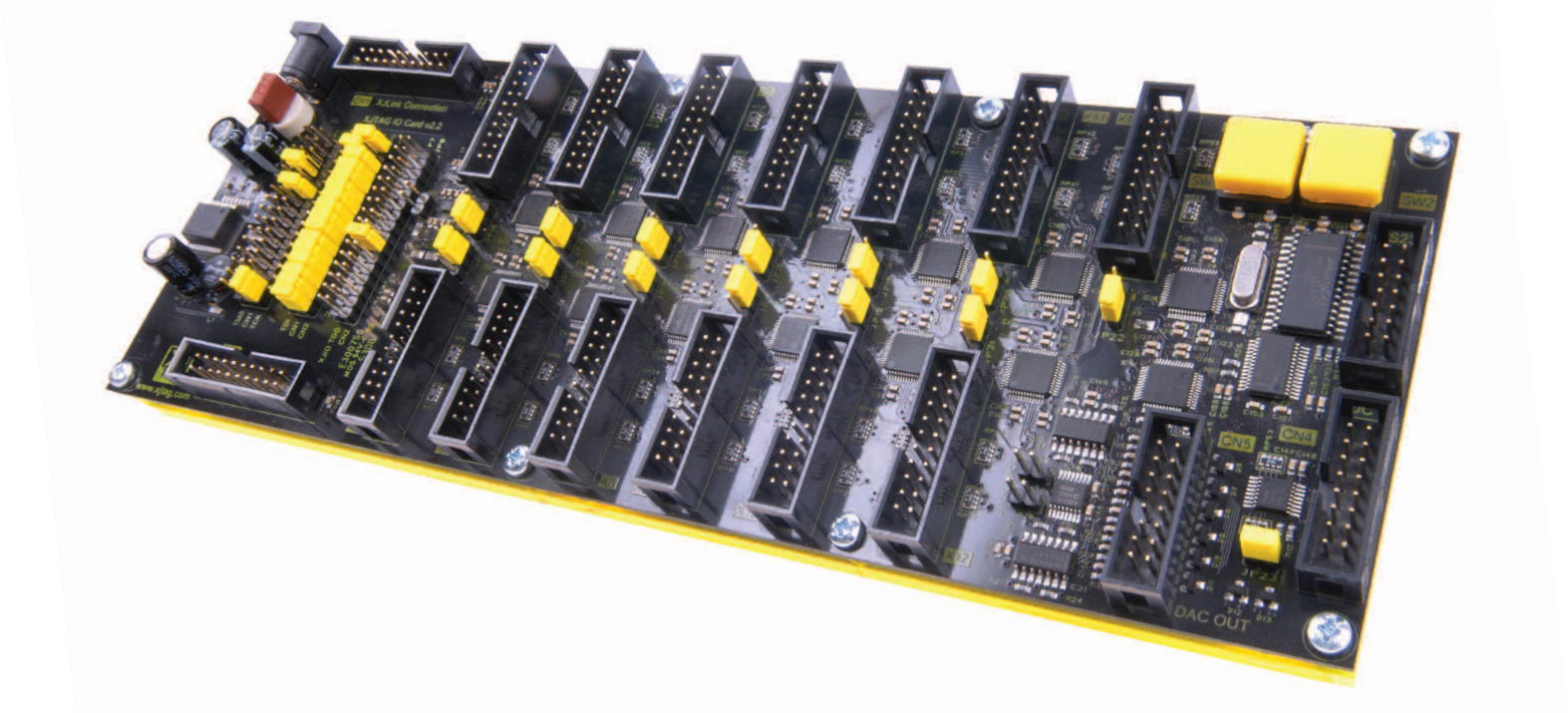


Table of Contents

Introduction

The v2 XJIO board provides an easy way to test a UUT's external I/O as part of a boundary scan test.

The JTAG connection is configurable for different pin mappings. All connectors have a standard 2.54 mm pitch for economical and efficient cable assemblies.

The XJIO board provides 208 digital I/O, eight analogue inputs, eight analogue outputs, an RS232 transceiver, and a UART. The digital I/O are 5 V compatible and can be configured between 1.8 V and 3.3 V (in banks of 16). If additional I/O are required, multiple XJIO boards can be daisy chained using their external JTAG connectors.

The on-board ADC is controlled via the JTAG interface and allows basic analogue measurements such as checking power rail voltages are within defined limits. The DAC provides stimuli for a UUT's analogue inputs, allowing improved test coverage.

The RS232 interfaces allow testing at RS232 voltage levels, and the UART allows testing at speed.

The board also includes two switches and three bi-colour LEDs that can be used during a test for simple user-interaction.

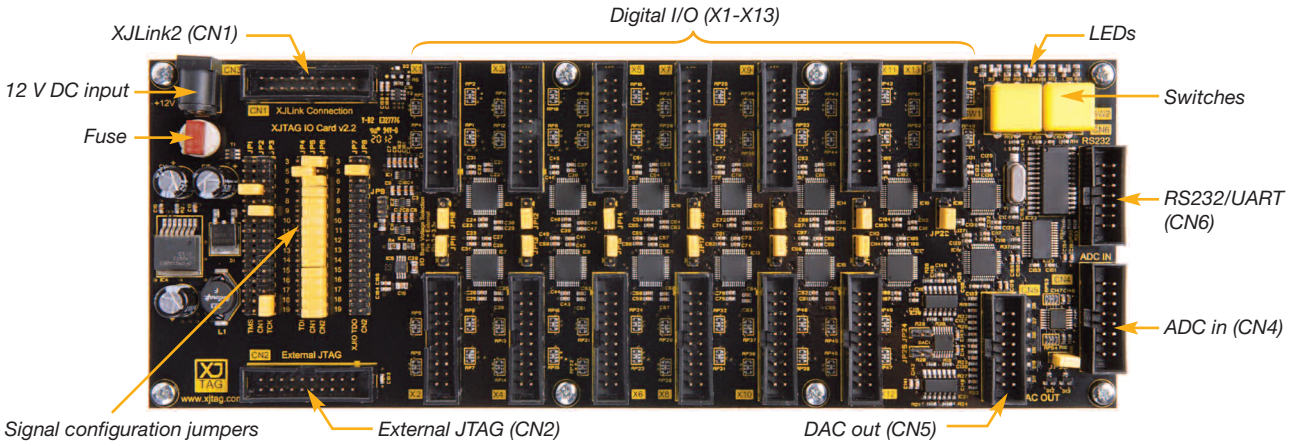


Figure 1 – XJIO board

Powering The Board

The board is powered at 12 V from a DC input socket. Connector type: 2.1 mm central pin, 5.5 mm outer diameter, 9.5 mm barrel length.

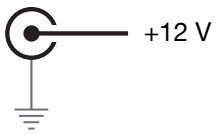


Figure 2 – Powering the XJIO board

Suitable mating plug: Farnell 224923, Digi-Key EP501A-ND, RS 487-858.

The current requirement is dependent on the number of I/O being switched, the rate of switching, and the loading on them. The board is protected by a replaceable 1 A time-delay TE5 fuse (e.g. Farnell 9516280 and Digi-Key WK6277CT-ND).

Although pin 1 of the XJLink2 is a 5 V power source, it should not be used to power the XJIO board because the XJLink2 only has low output current capability. Note that this XJLink output is always connected to pin 1 of the External JTAG connector (CN2).

XJLink2 Signal Routing

Because the XJLink2's connector pinout is defined by the user, the XJIO must be configured to route the four JTAG signals to the XJIO board's chain and, if applicable, to the External JTAG connector (CN2) for passing to another board. In addition, any spare PIO pins on the XJLink2 can be routed to the external connector for use during testing. This routing is configured by a series of jumpers, JP1 – JP8 as shown in Figure 3.

Note that pins 1, 2, 4, and 20 of the XJLink2 are not available for mapping.
Pin 2 is permanently connected to pin 2 the External JTAG connector.

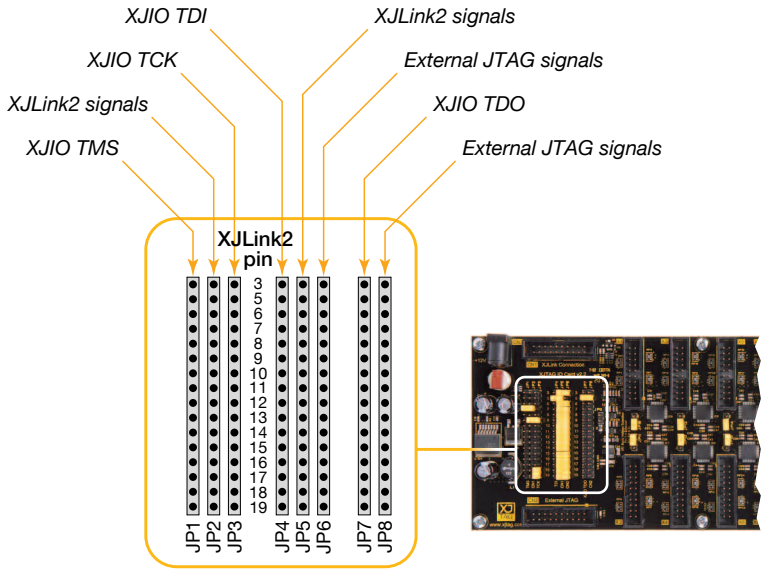


Figure 3 – Mapping the XJLink2 signals.

Routing the JTAG Signals

The XJIO's TMS input is connected to all the pins of JP1 in parallel, while the XJLink2 signals are connected to the individual pins of JP2 as shown in Figure 4. By placing a jumper between the relevant pin of JP2 and a JP1 pin, the XJLink2's TMS output can be routed to the XJIO's chain. In the example routing of Figure 4, the TMS signal is on pin 7 of CN1.

If the external JTAG connector is being used, a jumper should also be placed between JP5 and JP6 to route the TMS signal to CN2.

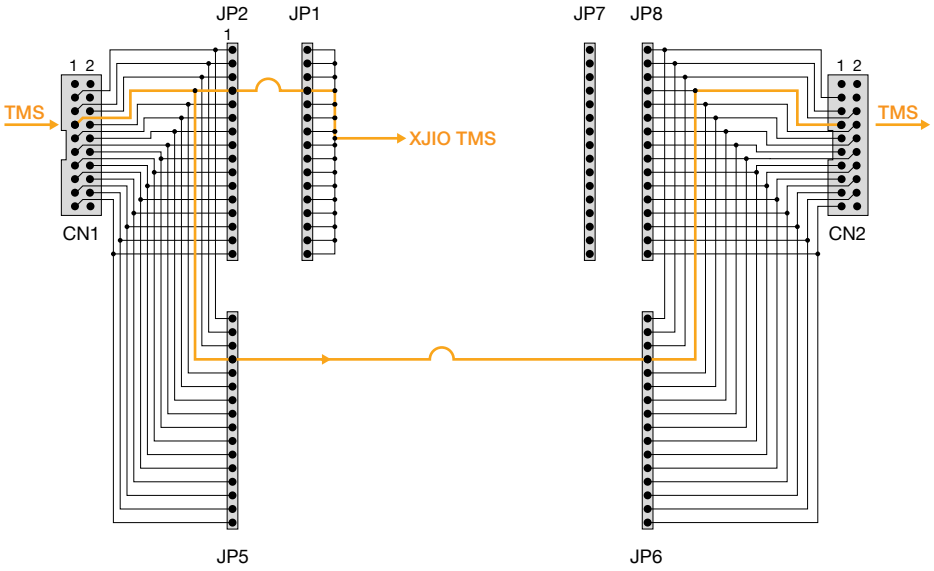


Figure 4 – Routing the TMS signal

The XJIO's TCK input is connected to all the pins of JP3 in parallel. Placing a jumper between the relevant signal of JP2 and a pin of JP3 will therefore route the XJLink2's TCK output to the XJIO's chain. Figure 5 shows an example routing when the TCK signal is on pin 9 of CN1.

If the external JTAG connector is being used, a jumper should also be placed between JP5 and JP6 to route the TCK signal to CN2.

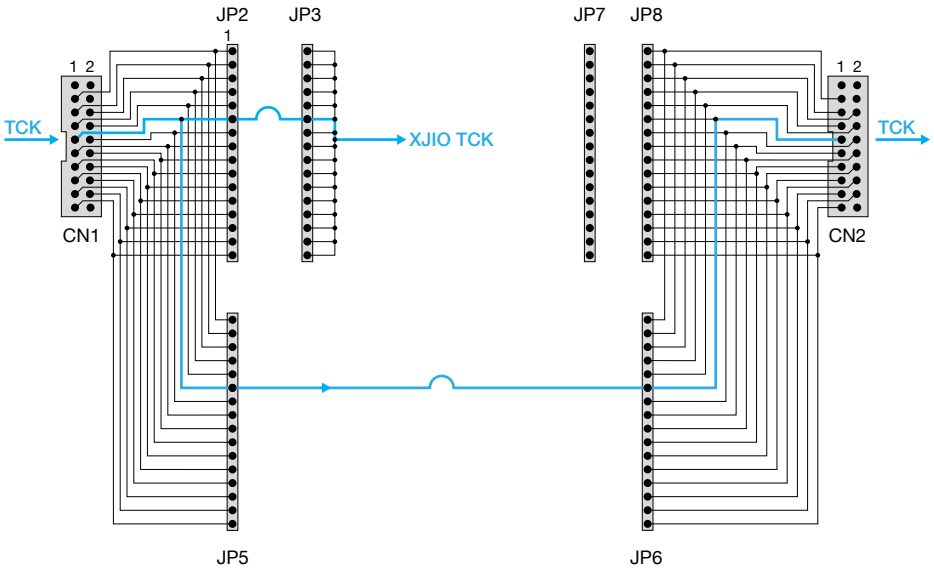


Figure 5 – Routing the TCK signal

Similarly, the XJIO's TDI signal is connected to all pins of JP4 in parallel, and the XJLink2 signals are connected to the individual pins of JP5. Placing a jumper between the TDI signal of JP5 and a pin on JP4 will therefore route the XJLink2's TDI signal to the start of the chain on the XJIO as shown in Figure 6.

To route TDO back to the XJLink2, two links are required to allow the option of routing the signal through another board first via connector CN2. Examples for these two situations are shown in Figure 6 and Figure 7, where TDI is on pin 5 of the main connectors CN1 and CN2, and TDO is on pin 13.

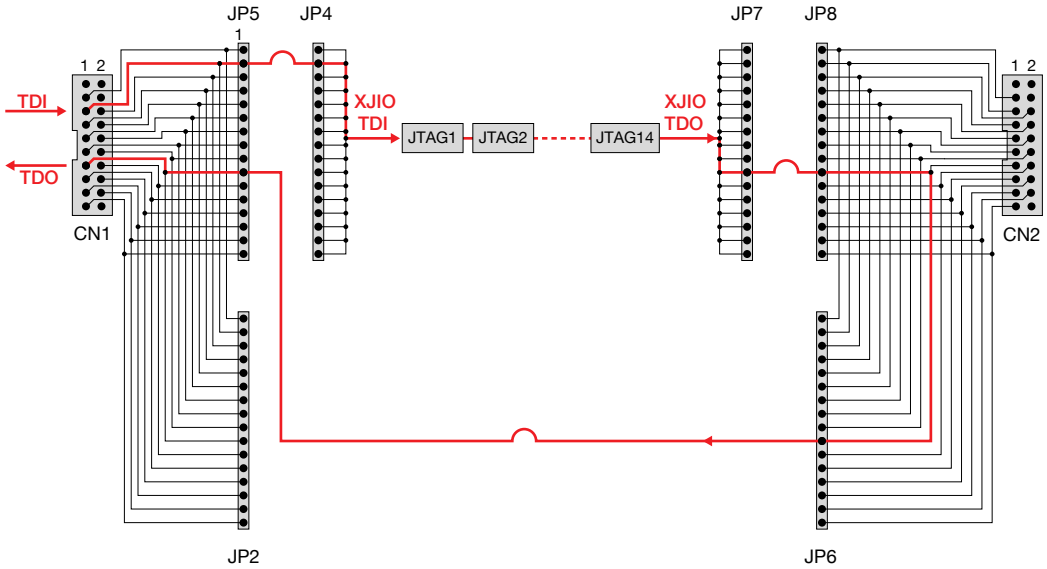


Figure 6 – Routing TDI and TDO without external JTAG

When external JTAG is not being used (Figure 6), connecting a jumper between JP7 and the TDO position on JP8 will route TDO to the required pin on JP6, from where a jumper to JP5 will connect the signal back to the XJLink2.

If the External JTAG connector is being used, the jumper positions need to be between different connectors as shown in Figure 7.

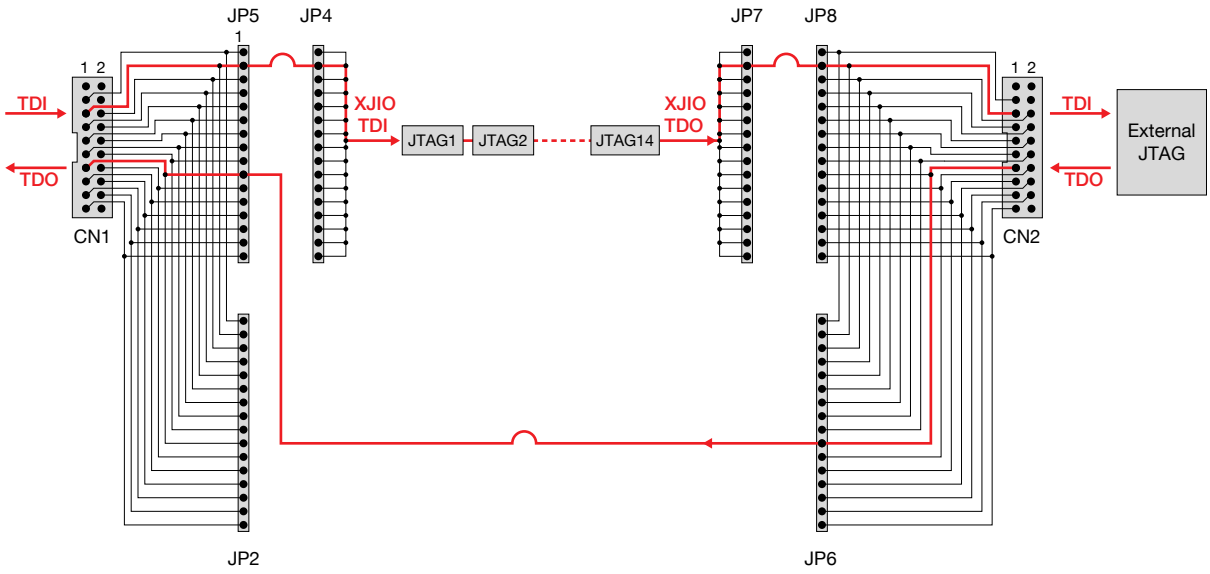


Figure 7 – Routing TDI and TDO with external JTAG

In that instance, the jumper from JP7 should instead connect to JP8's TDI position to route the signal to the TDI input of the next board. As before, a jumper between JP6's TDO position and JP5 will route the TDO from the other board back to the XJLink2.

TCK Termination Resistor

When the JTAG chain has been extended across several boards, it may be necessary to terminate the TCK signal to prevent glitches. Placing a link on jumper JP9 will add a 120 Ω termination to ground. This should be done only on the final board in the chain.

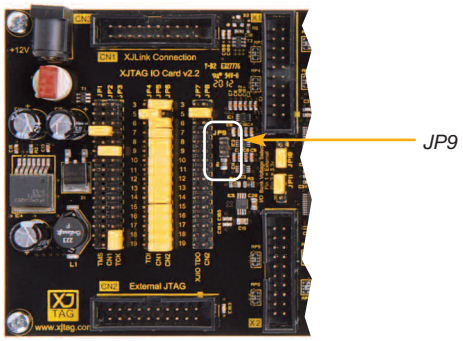


Figure 8 – Jumper to add a TCK termination if necessary

Passthrough for XJLink2's PIO

Signals from the XJLink2 can be passed directly to the External JTAG connector by adding jumpers between JP5 and JP6. This allows another JTAG chain and pins configured as PIO to be passed externally.

Note that XJLink2 pins 1, 4, and 20 are not available for passthrough in this way. Pin 2 is always connected directly to pin 2 of the External JTAG connector and cannot be routed elsewhere.

Default Jumper Positions

The XJIO board is supplied with jumpers placed in the default positions shown in Figure 9. In this configuration, it is assumed that only one XJIO is being used, and the TDO signal fed back to the XJLink2 is therefore taken from board's internal JTAG chain. The available XJLink2 PIO signals are passed to the External JTAG connector (CN2) for use during a test.

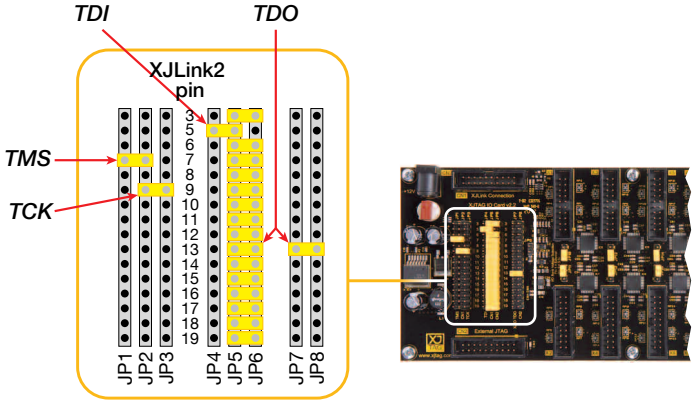


Figure 9 – Default jumper positions for signal routing

These default jumper settings assume the following XJLink2 pin configuration:

- TDI: CN1 pin 5
- TMS: CN1 pin 7
- TCK: CN1 pin 9
- TDO: CN1 pin 13

If several XJIO boards are to be chained together, the default TDO settings are suitable for the final XJIO board in the chain (assuming pin 13 is being used for TDO) but not for the other boards. The jumper between JP7 and JP8 on the other boards needs to be moved so that TDO is taken from the External JTAG connector: move the JP7-JP8 jumper to match the location of TDI on CN2.

Digital I/O

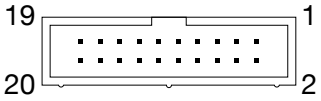
Connectors X1 to X13 provide up to 208 external digital I/O. Each connector is a 20-way 0.1” (2.54 mm) pitch 2 x 10 pin header and has sixteen I/O. By default, each connector is set to operate with LVCMOS 3.3 logic levels (5 V tolerant), but each connector can be configured independently to work at a different voltage domain by applying an external reference voltage (from 1.8 V to 3.3 V).

Example mating connectors:

TE Connectivity 1658623-4	Digi-Key: ASC20B-ND Farnell: 1164297 RS: 782-8805
3M 89120-0101	Digi-Key: MKC20A-ND Farnell: 2216911 RS: 192-7473
Amphenol ICC 71600-020LF	Digi-Key: 609-1741-ND Farnell: 1103920 RS: 771-0085

Pinout

Each connector has 16 digital I/O, three ground pins, and an input for an optional external reference voltage reference. The pinout is listed in Table 1.



Pin	Signal	Pin	Signal
1	External reference voltage input	2	GND
3	GND	4	IO1
5	IO2	6	IO3
7	IO4	8	IO5
9	IO6	10	IO7
11	IO8	12	IO9
13	IO10	14	IO11
15	IO12	16	IO13
17	IO14	18	IO15
19	IO16	20	GND

Table 1 – Pinout for each digital I/O Connector

NOTE: it is very important to ensure a good ground connection between the XJIO board and the UUT. It is recommended that you connect as many of the XJIO's grounds as possible to ground of the UUT.

Interface Voltages

By default, each connector is set to work with LVCMOS 3.3 voltages (5 V tolerant). Alternatively, each connector can be independently configured for a different voltage domain by changing a jumper position and applying the external reference voltage.

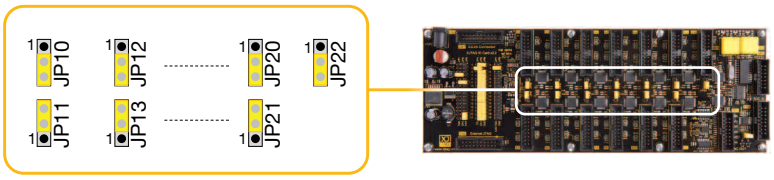


Figure 10 – Bank voltage selection jumpers

Logic Levels	Jumper Position
3.3 V	Pins 2 – 3
Set by external reference voltage	Pins 1 – 2

Table 2 – Selecting method for setting logic levels

I/O Connector	Jumper	I/O Connector	Jumper
X1	JP10	X2	JP11
X3	JP12	X4	JP13
X5	JP14	X6	JP15
X7	JP16	X8	JP17
X9	JP18	X10	JP19
X11	JP20	X12	JP21
X13	JP22		

Table 3 – Locating the jumpers to set logic levels for each I/O connector

When a connector is configured so that its logic voltages are set by an input reference voltage, the user-generated voltage should be applied to that connector's pin 1. The applied voltage should be between 1.8 V and 3.3 V (typical).

The bank supply voltages required for various standard logic families are given in the following table:

Standard	Input Bank Voltage (V)		
	Min	Typ	Max
LVTTL	3.0	3.3	3.6
LVCMOS 3.3	3.0	3.3	3.6
Extended LVCMOS 3.3	2.7	3.15	3.6
LVCMOS 2.5	2.3	2.5	2.7
LVCMOS 1.8	1.65	1.8	1.95
PCI 3.3	3.0	3.3	3.6

Table 4 – Input bank voltages for standard logic families

Electrical Characteristics

Parameter	Condition	Min	Typ	Max	Units
Input leakage current	$0 \leq V_{IN} <$ Vbank		0.5	1	µA
Input High Leakage current	$V_{bank} < V_{IN} < 5.5$ V			10	µA

Table 5 – Digital I/O leakage current

Standard	V_{IL} (V)		V_{IH} (V)	
	Min	Max	Min	Max
LVTTL	-0.3	0.8	2.0	5.5
LVCMOS 3.3	-0.3	0.8	2.0	5.5
LVCMOS 2.5	-0.3	0.7	1.7	3.6
LVCMOS 1.8	-0.3	$0.35*V_{bank}$	$0.65*V_{bank}$	3.6
PCI 3.3	-0.3	$0.3*3.3*(V_{bank}/1.8)$	$0.5*3.3*(V_{bank}/1.8)$	5.5

Table 6 – Digital I/O input voltage limits

Standard	V_{OL} Max (V)	V_{OH} Max (V)	I_{OL} (mA)	I_{OH} (mA)
LVTTL	0.4	V_{bank} - 0.4	8.0	-4.0
	0.2	V_{bank} - 0.2	0.1	-0.1
LVCMOS 3.3	0.4	V_{bank} - 0.4	8.0	-4.0
	0.2	V_{bank} - 0.2	0.1	-0.1
LVCMOS 2.5	0.4	V_{bank} - 0.4	8.0	-4.0
	0.2	V_{bank} - 0.2	0.1	-0.1
LVCMOS 1.8	0.4	V_{bank} - 0.45	2.0	-2.0
	0.2	V_{bank} - 0.2	0.1	-0.1
PCI 3.3	$0.1 * V_{bank}$	$0.9 * V_{bank}$	1.5	-1.5

Table 7 – Digital I/O output voltages at low and maximum output currents

DAC Outputs

The XJIO board uses an AD5318 DAC (DAC1) to provide simple voltage references to improve test coverage of analogue circuitry. There are eight 10-bit buffered voltage outputs, controlled over a serial interface that is accessed via a JTAG device on the board. An XJEase test device file, *AD5318.xje* can be found in the XJTAG Shared Files directory that is installed with XJTAG (in the *XJIO Board* folder).

The parts incorporate a power-on reset circuit, which ensures the DAC outputs power up at 0 V and remain there until a valid write to the device.

The DAC outputs may be updated collectively or individually, and can be powered down individually. The outputs are buffered and available on CN5, a 16-way 0.1" (2.54 mm) box header.

Setting the DAC Output Voltage Range in Hardware

By default, the board is configured not to use external voltage references for the DAC, and the device's internal register must therefore be set using XJEase so that V_{DD} is used instead (see *Controlling and Configuring* the DAC below); this gives a nominal output range of 0 – 3.3 V. To achieve other ranges, external reference voltages should be used, one for DAC channels VOUTA to VOUTD, and another for VOUTE to VOUTH. Each group of channels has a 2-pin jumper as shown in Figure 11 and Table 8. The reference voltage should be applied to pin 1 of the jumper, and a ground to pin 2.

The reference voltage must be in range 0.25 – 3.3 V if the DAC is being used in unbuffered mode or 1.0 – 3.3 V when it is in buffered mode (see *Setting Gain, Buffering, and the VREF Input* below).

The output range is 0.001 V to V_{REF} when unity gain has been selected in software. If the gain is set to 2 in software, the output range is 0.001 V to 2*V_{REF}, limited to 1 mV below the DAC's supply (which is nominally 3.3 V).

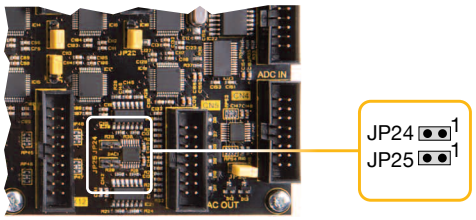
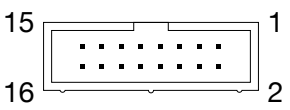


Figure 11 – Headers for applying external DAC reference voltages

Pinout

Pin	Signal	V_{REF} jumper
1	V_{OUT}A	JP24
3	V_{OUT}B	
5	V_{OUT}C	
7	V_{OUT}D	
9	V_{OUT}E	JP25
11	V_{OUT}F	
13	V_{OUT}G	
15	V_{OUT}H	

Pin	Signal
2	GND
4	GND
6	GND
8	GND
10	GND
12	GND
14	GND
16	GND



Suitable mating connectors: Digi-Key ASC16B-ND, Farnell 1174661, and RS 782-8861.

Table 8 – DAC connector pinout

Controlling and Configuring the DAC in Software

The *ADS5318.xje* test device file contains functions that can be used to control the DAC (see Table 9). As well as providing a mechanism for setting the DAC's output voltage, they can configure settings such as output gain and the source of V_{REF}, and can power down individual outputs.

The default XJTAG installation includes a demo project for the XJIO board that configures the DAC outputs to provide voltages that can be fed back to its ADC inputs for measurement via a loopback cable. It can be found in the XJIO folder, which is in the XJTAG directory (normally located in the Public Documents user directory). In addition, a circuit code file that uses the DAC's XJEase functions is shown in the XJIO Board Example in the Hardware section of the XJTAG Help system.

The XJEase functions in the DAC's test device file are as follows:

XJEase Function	Operation
`Init()()`	For each group of channels (A-D and E-H), this sets the source of V_{REF}, sets the input as buffered or unbuffered, and sets the DAC output gain. It leaves the physical $\overline{LDAC}$ pin high.
`SetDAC()()`	Writes to a particular channel.
`SetLDAC()()`	Writes a 2-bit word to the LDAC mode register to determine how values are latched into the DAC output registers.
`ToggleLDAC()()`	Uses boundary scan to toggle the physical $\overline{LDAC}$ pin to load a value from the input register to the DAC output (if the relevant LDAC mode has been selected).
`PowerDown()()`	Powers down selected channels.
`ResetData()()`	Clears all the DAC input registers
`ResetDAC()()`	Clears all the DAC input registers and resets all control bits to their power-up defaults.

Table 9 – XJEase functions in the DAC test device file

Setting Gain, Buffering, and the V_{REF} Input

The V_{REF} inputs can be buffered to draw virtually no current from the external source. In this setting, V_{REF} must be in the range 1 – 3.3 V. In the unbuffered state, the input range for V_{REF} is 0.25 – 3.3 V.

The DAC gain can be set to 1x or 2x. At unity gain, the output range is 0 to V_{REF}. At 2x gain, the output range is 0 to 2*V_{REF} but is capped at 1 mV below V_{DD} (which is nominally 3.3 V).

NOTE: When V_{DD} is used as the reference, it is always unbuffered and the output gain is set to unity, regardless of the gain and buffer settings. Unless an external reference is applied to the relevant jumper, V_{DD} must be selected as the source.

The control word to configure these settings is shown in Figure 12. Note that channels cannot be controlled independently, but only in two groups – channels A to D and channels E to H.

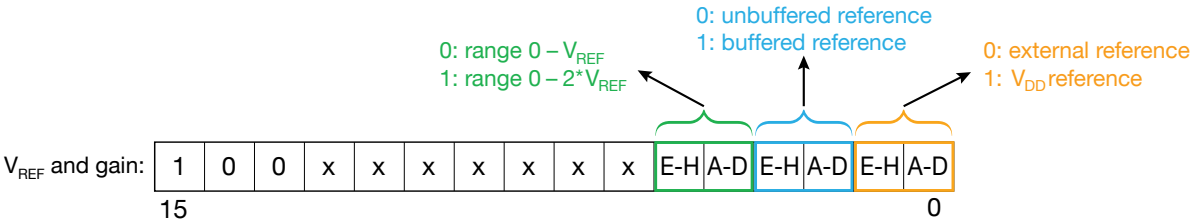


Figure 12 – The control word to configure V_{REF} and gain

The XJEase function `Init()()` sets the DAC to these default values:

- V_{REF} source: internal V_{DD} (bits 0 – 1 high)
- Unbuffered reference (bits 2 –3 low)
- Unity gain. Output range: 0 – V_{REF} (bits 4 – 5 low)

The DAC settings can be modified as part of the setup.

Setting the Output Voltage

The function `SetDAC()()` loads a value into the selected DAC's input register, ready to be driven out. It takes two arguments: the first (a 3-bit binary code) selects the channel, and the second is the value to be written (a 10-bit binary code).

To select channel A, pass in 000 to the function; to select channel B, pass in 001, etc.

The nominal output voltage is determined as follows:

$$V_{OUT} = \frac{(V_{REF} * D)}{1024}$$

Where D is the value passed to the DAC in decimal.
V_{REF} is either nominally 3.3 V or the applied external voltage.

NOTE: the DAC outputs do not have guaranteed accuracy and should not be used as calibrated sources.

Setting DAC Outputs to Zero

Individual DAC outputs can be set to 0 V at the connector by powering down the relevant channel using the function `PowerDown()()`. An 8-bit binary word is passed into the function, constructed as shown in Figure 13, where A – H is the channel reference:

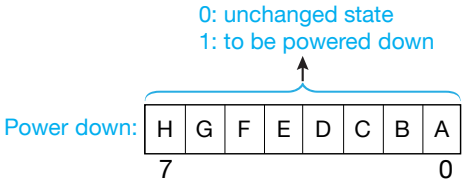


Figure 13 – Control word to power down individual channels

`PowerDown()()` constructs the required 16-bit control word and writes it to the control register.

Latching the Output Value

The DAC has a double-buffered interface consisting of two banks of registers: input registers (that hold the value to be written) and DAC output registers (that determine what appears on the output pin). This allows multiple input register values to be set before any output voltages are updated, thus allowing a set of DAC outputs to be changed simultaneously (note: values are only written to the DAC outputs if the value in the input register has been changed). Alternatively, the output registers can be made transparent so that the outputs change automatically as soon as the input register has been written.

The manner in which the values from the input registers are latched into the DAC outputs is determined by the value of two LDAC bits, which can be set using the XJEase function `SetLDAC()()`. The three valid settings are shown in Table 10.

LDAC Bits	Function	Comments
00	The DAC registers will be updated continuously.	
01	The input registers can be changed without affecting the contents of the DAC output registers. A subsequent operation is required to load values into the DAC output registers.	This is the default mode.
10	This causes the DAC output register to be updated without needing to use the physical pin (writing this mode causes a single pulse on the internal $\overline{\text{LDAC}}$ signal).	

Table 10 – Function of the LDAC control bits

To set the LDAC bits, use the `SetLDAC()()` function and pass in the required two-bit binary value.

Access to the DAC output registers is controlled by both the $\overline{\text{LDAC}}$ pin and the LDAC mode bits. There are three ways to load the value from the input register to the DAC output:

A) Controlling the physical Load pin over boundary scan

The physical pin is controllable over boundary scan using the `ToggleLDAC()()` function. To use the pin to load the DAC outputs, the LDAC control bits should be left at their default (01) values, and the `ToggleLDAC()()` function called to load the values to the outputs.

B) Using the LDAC register

As an alternative to using the physical pin to load the value into the output, the LDAC register can be used instead. To do this, the $\overline{\text{LDAC}}$ pin must be held high (note: the `Init()()` function puts the pin into this state). The LDAC control bits can then be used to apply a pulse to the internal $\overline{\text{LDAC}}$ signal (see Table 10).

C) Continuous output updates

The DAC output registers can be updated continuously by setting the LDAC bits as shown in Table 10. In this mode, the relevant outputs are changed as soon as the write is complete.

Reset Modes

Two reset functions are available – `ResetData()()` and `ResetDAC()()`.

`ResetData()()` resets all the DAC input and output registers to zero; all outputs will the driven to 0 V.

`ResetDAC()()` also resets all the DAC input and output registers to zero, but additionally reinitialises the DAC into its power-up modes. After this reset, the outputs will be driven to 0 V, and the DAC settings will be as follows:

- LDAC bits set to "01" – i.e., the DAC input registers can be set up without the output values changing.
- Voltage reference from unbuffered V_{REF} inputs.
- Output range is 0 to V_{REF}.

Electrical Characteristics

Parameter	Min	Typ	Max	Units	Conditions
Resolution	8			bits	
Relative accuracy		±0.15	±1	LSB	
V_{REF} input range	1.0		3.3	V	Buffered reference mode
	0.25		3.3	V	Unbuffered reference mode
V_{REF} input impedance		>10		MΩ	Power-down mode & buffered reference mode
	37	45		kΩ	Unbuffered reference mode, unity V_{REF} gain
	18	22		kΩ	Unbuffered reference mode, x2 V_{REF} gain

Table 11 – DAC electrical characteristics

Analogue Inputs

The XJIO board uses an ADS7830 ADC (ADC1) to provide eight single-ended 8-bit analogue inputs. They are intended for general voltage measurements such as checking the UUT's power rails are within specification.

NOTE: the accuracy of the ADC measurement is not guaranteed, and the ADC should not be considered as calibrated equipment.

A test device file *ADS7830.xje* is supplied that has a test function `Test()()` that can be included in your test list. It can be configured to read input voltages, compare them with target values, and return a pass/fail result.

The test device file also includes a function called `ReadADC()()` that can be used for simple single-channel measurements.

Pinout

The ADC inputs are accessible on connector CN4, a 16-way 0.1” box header. Ground connections are available on adjacent pins to support the use of twisted-pairs.

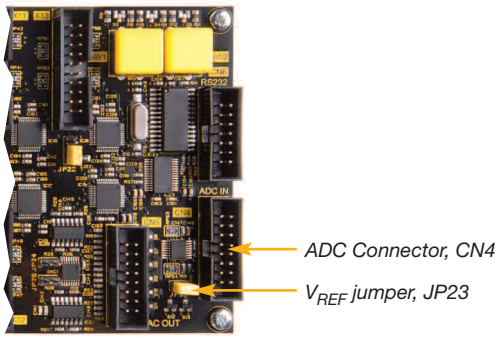


Figure 14 – ADC inputs and V_{REF} jumper

Pin	Signal	Pin	Signal
1	CH0	2	GND
3	CH1	4	GND
5	CH2	6	GND
7	CH3	8	GND
9	CH4	10	GND
11	CH5	12	GND
13	CH6	14	GND
15	CH7	16	GND

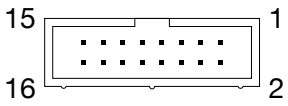


Table 12 – ADC connector CN4’s pinout

Input Voltage Range and Reference Voltage

The input voltage range for the ADC inputs is determined by the reference voltage being used. The ADC can be used with the device’s internal +2.5 V voltage reference (giving an input scan range of 0 – 2.5 V) or with the 3.3 V reference on the XJIO board (giving an input scan range of 0 – 3.3 V). It is recommended that the 2.5 V internal reference is used because it is more accurate, but the alternative 3.3 V source may be required for a wider input range. The test device file for the ADC uses configuration variables to set which reference is being used.

To use the board’s 3.3 V voltage reference, ensure jumper JP23 is fitted (see Figure 14). Jumper JP23 is fitted by default so **it is important to remove the jumper if the ADC is configured to use its internal reference** (to avoid back-driving the voltage reference pin, which becomes an output when the internal reference is used).

Each ADC input channel is protected by a pair of diodes and a 1 kΩ resistor that will attempt to prevent voltages above 3.3 V from damaging the inputs. If it is required to measure voltages greater than 3.3 V, a simple potential divider should be implemented externally to guarantee the input voltage is always below the maximum.

Configuring the Supplied Test Device File

The test device file *ADS7830.xje* uses the following configuration variables to tell the software how the ADC is being used in your test setup:

- Details of V_{REF} being used
- Which channels to test
- Expected voltages to be read
- Values used in any external potential divider (if present)

These variables can be set from XJDeveloper's *Categorise Device* screen: select the ADC and click **Configure…** To use the board's V_{DD} as a 3.3 V external V_{REF}, ensure there is no tick in the *Use Internal Reference* box and specify an external reference of 3300 mV.

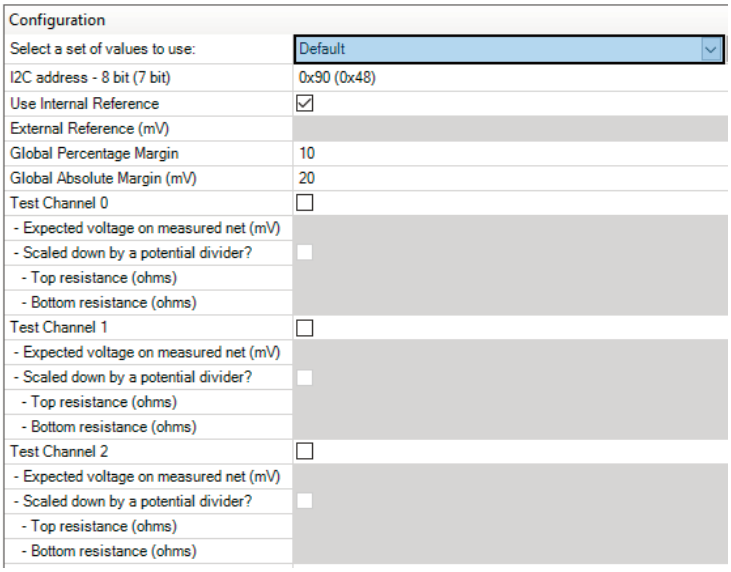


Figure 15 – Setting the ADC's configuration variables when using the supplied test device file

For each channel being used, place a tick in the box for it to be included in the test, and enter the expected reading in mV. The pass/fail limits are common to all channels and are set at the top of the list as a percentage or absolute margin (see Figure 15).

If no channel is selected, just a simple connectivity test will be performed.

If a channel's input is scaled down by an external potential divider, the upper and lower resistor values should be provided in Ohms so that the reported voltage reading will take them into account.

The ADC uses an I²C serial bus with a slave address set in hardware to 0x90.

Reading the ADC

To perform a simple ADC reading instead of running the full test in the supplied test device file, the function `ReadADC()()` that the file includes can be used instead. Three parameters need to be passed into the function:

Input Parameter	Description	Size	Comment
channel	Channel to read	3 bits	Channel 0: 000, channel 1: 001, etc.
useInternalRef	V_{REF} source	1 bit	1: use DAC's internal 2.5 V reference
			0: use a reference external to the DAC
refVoltage	Value of V_{REF}		Value in mV

Table 13 – Input parameters for the ReadADC()() function

The function returns the reading in mV and the integer result `RESULT_PASS` if the I^2C transfers completed successfully.

The default XJTAG installation includes a demo project for the XJIO board that configures the ADC and reads input voltages fed to it from the DAC's outputs via a loopback cable. The project can be found in the XJIO folder, which is in the XJTAG directory (normally located in the Public Documents user directory). In addition, an example of a circuit code file that uses the `ReadADC()()` function is shown in the Hardware section of the XJTAG Help system.

Electrical Characteristics

Typical leakage current on analogue inputs: ±1 µA

RS232 Transceiver and UART

There are two RS232 devices on the XJIO Board: a MAX3241 RS232 transceiver (IC23) for simple connectivity testing at RS232 signal levels, and a MAX3111 UART device (IC22) to enable full RS232 communication testing. Both devices connect to CN6, a 14-way 0.1" box header.

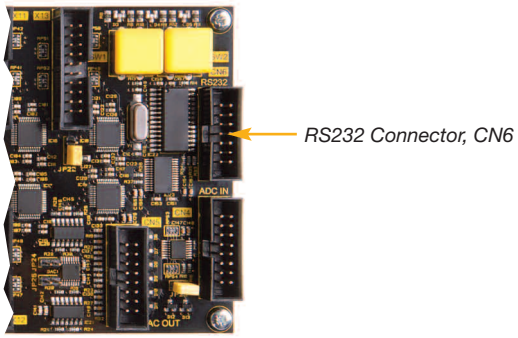


Figure 16 – RS232 connections are on CN6

Pinout

Pin	Signal
1	RS232_R1IN
3	RS232_T1OUT
5	GND
7	RS232_T3OUT
9	RS232_R5IN
11	UART_RTSn
13	UART_TX

Pin	Signal
2	RS232_R2IN
4	RS232_T2OUT
6	RS232_R3IN
8	RS232_R4IN
10	GND
12	UART_CTSn
14	UART_RX

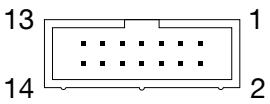


Table 14 – Pinout of RS232 connector, CN6

Using the RS232 Transceiver

The RS232 transceiver has three driver outputs and five receiver inputs. The XJTAG XJEase library includes a test device file, *MAX3241.xje*, that contains functions to control the RS232 drivers and receivers. Figure 17 shows a typical setup using the XJIO board to test an RS232 device on the UUT by performing an interactive communications test.

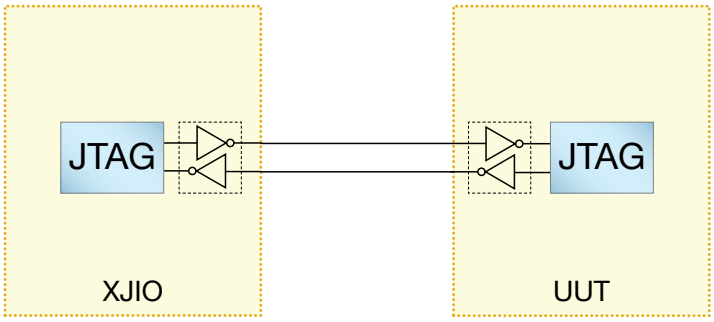


Figure 17 – Connecting to the UUT to perform an RS232 test

To create this test, a circuit code file is required that calls the supplied XJEase functions and similar functions for the UUT device to perform the device-to-device communication.

The *MAX3241.xje* test device file contains the following XJEase functions:

- `EnableRS232()()`: enables the RS232 driver. No input arguments are required.
- `DisableRS232()()`: disables the RS232 driver. No input arguments are required.
- `SetTX()()`: writes to the driver. A binary code should be passed into the function.
- `ReadRX()()`: reads from the receiver and returns the received binary code.

Using the RS232 UART

The RS232 UART can operate at up to 230 kBaud.

The XJTAG library contains the XJEase test device file *MAX3111E.xje*, which includes control functions to perform the following:

- Initialise the UART
- Read the state of the UART's interrupt request pin
- Transmit data or change the state of the RTS pin
- Receive data
- Set and read the UART's configuration

Initialisation

`Initialise()()` brings the UART out of shut down and applies the following settings:

- FIFO: enabled
- Software shutdown: disabled
- Transmit buffer empty interrupt: disabled
- Receive FIFO not empty interrupt: enabled
- Parity bit received interrupt: disabled
- Receiver activity/framing error interrupt: disabled
- IrDA mode: disabled
- Parity: disabled
- One stop bit
- 8-bit words

If the `BAUD_RATE` variable is not set, the baud rate will be 230.4 kBaud.

Read the Interrupt Request Pin

`ReadIRQn()()` returns the state of the UART's IRQn pin.

Transmit Data

`TxData()()` allows data to be sent or the state of the RTS pin to be modified.
The function has four input parameters and six output parameters:

Input Parameter	Description	Size	Comment
txDataEn	Enable Transmit	1 bit	0: transmit all bits
			1: update RTS only
rts	Set RTS signal	1 bit	0: set the $\overline{\text{RTS}}$ pin to logic high
txParity	TX parity bit	1 bit	Ignored if parity has been disabled
txData	Transmit data	8 bits	Data to send (if txDataEn = 0)

Table 15 – Input parameters for the `TxData()()` function

Output Parameter	Description	Size	Comment
rxDataValid	Receive buffer status	1 bit	1: receive data is available
			0: receive buffer is empty
txBufferEmpty	Transmit buffer status	1 bit	1: transmit buffer is empty
framingError	Framing error flag	1 bit	1: a framing error occurred
cts	State of the $\overline{\text{CTS}}$ pin	1 bit	0: $\overline{\text{CTS}}$ pin is logic high
rxParity	Received parity bit	1 bit	Only valid if parity is enabled
rxData	Received data	8 bits	

Table 16 – Output parameters for the `TxData()()` and `RxData()()` functions

Receive Data

The `RxData()()` function allows data to be read from the UART. The function has no input parameters, and its output parameters are the same as for `TxData()()` (see Table 16).

Set the UART's Configuration

`WriteConfig()()` allows the UART's configuration to be changed. The function has eleven input parameters and two output parameters.

Input Parameter	Description	Size	Comment
fifoEn	Enable FIFO	1 bit	0: enable receive FIFO
shutdown	Software shutdown	1 bit	1: enter software shutdown
txBufferEmptyIrqEn	IRQ mask for empty Tx buffer	1 bit	1: enable interrupt for empty transmit buffer
rxDataValidIrqEn	IRQ mask for valid data received	1 bit	1: enable interrupt for data available in receive register
rxParityIrqEn	IRQ mask for received parity bit	1 bit	1: enable interrupt for high parity bit received
framingErrorIrqEn	IRQ mask for framing error	1 bit	1: enable interrupt for receive framing error
irTimingEn	Enable IrDA mode	1 bit	1: enable IrDA timing mode
twoStopBits	Set the number of stop bits	1 bit	1: transmit 2 stop bits
			0: transmit 1 stop bit
parityEn	Enable Tx & Rx parity	1 bit	1: enable parity for Rx and Tx
sevenBitData	Word length setting	1 bit	1: 7-bit words (8 with parity)
			0: 8-bit words (9 with parity)
baud	Baud rate	4 bits	Baud rate in bits per second

Table 17 – Input parameters for the `WriteConfig()()` function

Output Parameter	Description	Size	Comment
rxDataValid	Receive buffer state	1 bit	1: data has been received
txBufferEmpty	Transmit buffer state	1 bit	1: transmit buffer is empty

Table 18 – Output parameters from the `WriteConfig()()` function

Read the UART's Configuration

`ReadConfig()()` allows the UART's current configuration to be read and the device to be put into test mode. In test mode, when is low, the signal switches at 16 times the baud rate.

The function has one input and thirteen output parameters. The input parameter enables the test mode when it is set to 1. The output parameters are the same as the output and input parameters for the `WriteConfig()()` function.

Example Code

In the *Board Test* directory for the XJIO Board (in the XJTAG Shared Files folder, which is usually in c:\users\public\public documents\XJTAGx.y), there is an example code file, *RS232 UART.xje*, that tests two linked UARTs. The example is for two XJIO boards connected together.

User Interaction (Switches & LEDs)

There are three tri-colour LEDs and two buttons on the XJIO board. These are included to enable simple test control and user feedback. The switches and LEDs are interfaced directly to the JTAG chain to allow them to be part of an XJTAG test.