PIO-1149.1/E
Parallel Port
Boundary-Scan Controller

User's Manual

PIO-1149.1/E
Parallel Port
Boundary-Scan Controller

User's Manual

PIO-1149.1/E User's Manual

12607 Hiddencreek Way
Cerritos, CA 90703-2146
Tel.: (562) 926-6727 • Fax.: (562) 404-6196

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition is published.

A software code may be printed before the date; this indicates the version of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

Edition 1, September 2001

Edition 2, December 2002

Edition 3, March 2003

GENERAL NOTICE

Information contained in this document is subject to change without notice. CORELIS shall not be liable for errors contained herein for incidental or consequential damages in connection with the furnishing, performance, or use of material contained in this manual. This document contains proprietary information, which is protected by copyright. All rights reserved. No part of this document may be reproduced or translated to other languages without the prior written consent of CORELIS.

CORELIS assumes no responsibility for the use of or reliability of its software on equipment that is not furnished by CORELIS.

PRODUCT WARRANTY

This CORELIS product has a warranty against defects in material and workmanship for a period of 90 days from date of shipment. During the warranty period, CORELIS will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by CORELIS. Outside CORELIS service travel areas, warranty service will be performed at the Buyer's facility only upon CORELIS' prior agreement and Buyer shall pay CORELIS' round trip travel expenses.

For products returned to CORELIS for warranty service, the Buyer shall prepay shipping charges to CORELIS and CORELIS shall pay shipping charges to return the product to the Buyer. However, the Buyer shall pay all shipping charges, duties, and taxes for products returned to CORELIS from another country.

CORELIS warrants that its software and firmware designated by CORELIS for use with an instrument will execute its programming instructions when properly installed on that instrument. CORELIS does not warrant that the operation of the instrument, software, or firmware will be uninterrupted or error-free.

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. CORELIS SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES CONTAINED HEREIN ARE THE CUSTOMER'S SOLE AND EXCLUSIVE REMEDIES. CORELIS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

Product maintenance agreements and other customer assistance agreements are available for Corelis products. For assistance, contact your nearest Corelis Sales and Service Office.

RETURN POLICY

No items returned to CORELIS for warranty, service, or any other reason shall be accepted unless first authorized by CORELIS, either direct or through its authorized sales representatives. All returned items must be shipped pre-paid and clearly display a Returned Merchandise Authorization (RMA) number on the shipping carton. Freight collect items will NOT be accepted. Customers or authorized sales representatives must first contact CORELIS with notice of request for return of merchandise. RMA's can only originate from CORELIS. If authorization is granted, an RMA number will be forwarded to the customer either directly or through its authorized sales representative.

Table Of Contents

Table of Figures

Table of Tables

Chapter 1
Product Overview

Introduction

The PIO-1149.1/E Parallel Port Controller provides the simplest possible solution to interfacing a PC with any IEEE Standard 1149.1 compatible target. The PIO-1149.1/E controls the operation of an IEEE Standard 1149.1 (JTAG) scan test path by generating the proper signals under software control to interface with the target devices. With the PIO-1149.1/E, you can control boundary-scan operations per IEEE Standard 1149.1 through software. You can command target circuitry Built-In-Self-Test (BIST), verify PCB interconnects, perform functional testing and debug without manual probing. Furthermore, you can access device internal functions that are not accessible to external probing through the JTAG interface, to isolate faults within the device itself.

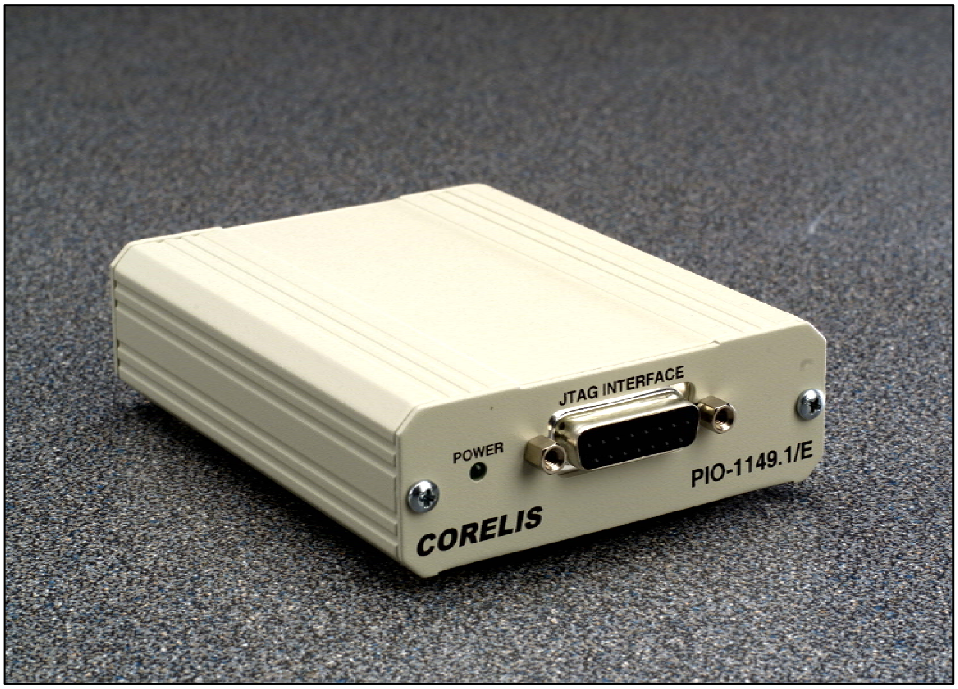
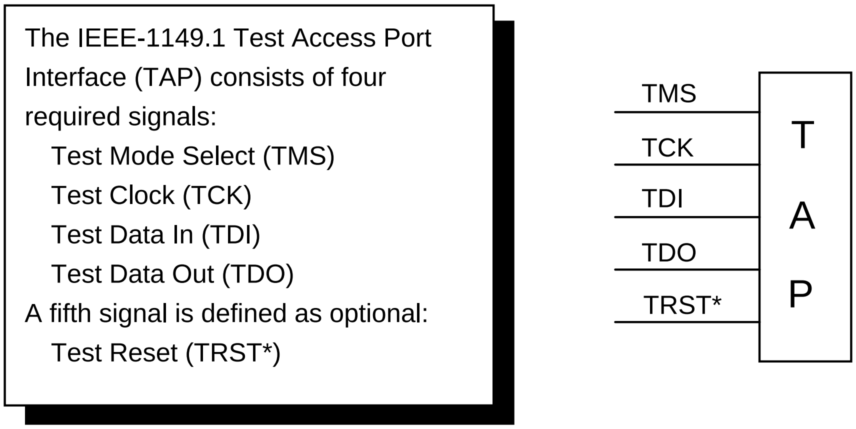


Figure 1-1. The Corelis PIO-1149.1/E Boundary-Scan Controller

What Is IEEE Standard 1149.1?

The IEEE Standard 1149.1 test bus and boundary-scan architecture allows you to control an IC, and similarly a board or system, via a standard four-signal interface. Each IEEE Standard 1149.1 compliant IC incorporates a feature known as *boundary-scan* that allows you to control and observe each functional pin of the IC via the four-wire interface. You can load test, debug, or initialization patterns serially into the appropriate IC(s) via the IEEE Standard 1149.1 test bus. This allows you to observe or control IC, board, or system functions with limited physical access.

The IEEE Standard 1149.1 test bus contains two main elements: a Test Access Port (TAP), which interfaces internal IC logic with the external world via a four-signal (optionally five-signal) bus as shown in Figure 1-2**;** and a boundary-scan architecture, which defines standard boundary cells that drive and receive data at the IC pins. IEEE Standard 1149.1 also defines both mandatory and optional opcodes and test features. The test bus signals are: Test Clock (TCK), Test Mode Select (TMS), Test Data In (TDI), Test Data Out (TDO), and the optional Test Logic Reset (TRST).



The IEEE-1149.1 Test Access Port Interface (TAP) consists of four required signals:

- Test Mode Select (TMS)
- Test Clock (TCK)
- Test Data In (TDI)
- Test Data Out (TDO)

A fifth signal is defined as optional:

- Test Reset (TRST*)

Figure 1-2. Test Access Port (TAP)

Features of the PIO-1149.1/E

The Corelis PIO-1149.1/E is a sophisticated test controller that can test devices, boards or systems that comply with IEEE Standard 1149.1. The PIO-1149.1/E tester supports two independent boundary-scan chains. You can configure the four discrete I/O pins through software as standard inputs, outputs, or open collector drivers to test or control non-boundary-scan areas of the unit under test (UUT). Software-controlled voltage-translating logic enables the PIO-1149.1/E to test low voltage systems.

A set of software drivers written in C ship with the PIO-1149.1/E device to allow you to create powerful test programs that are tailored to your needs.

Programmable Clocks

The PIO-1149.1/E TCK output to the IEEE Standard 1149.1 compatible target system is programmable under software control. Allowable frequencies range from 1 MHz to 40 MHz.

Adjustable Voltage Interfaces

The voltage level of the parallel I/O and JTAG TAP interfaces is software-programmable and you can set it to any voltage between 1.25 V and 3.40 V in increments of 0.05 V. These interfaces are TTL compatible and 5 V tolerant.

Discrete Input/Output Pins

You can configure the four discrete I/O pins through software as standard inputs, outputs, or open collector drivers to test or control non-boundary-scan areas of the UUT. Direct software control to drive, sense, and tri-state each of the four pins provides this configurability.

The input and output ports can control and sense various functions in the target system that you cannot control or observe through boundary-scan operations. These ports are useful for testing of target systems that incorporate components that do not comply with IEEE Standard 1149.1.

Self-Test Capabilities

The PIO-1149.1/E has a built-in self-test capability. Internal loop-back logic validates shift integrity for the JTAG TAP. The unit can also read back programmable voltage levels as well as discrete I/O pins.

Parallel Port Interface

The PIO-1149.1/E is a full-speed device compliant with IEEE-1284 ECP Mode Phase. Refer to IEEE Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers Specification (IEEE Std 1284-1994).

PIO-1149.1/E Specifications

IEEE-1149.1

Number of TAP Controllers	2
Maximum TCK frequency	40 MHz
Maximum scanning data length	2^{32} bits

Parallel I/O and JTAG TAP signal levels

Voltage levels	programmable from 1.25 to 3.40 V
Increment step	0.05 V

Parallel I/O and TAP Signals DC Characteristics

			MIN	MAX	UNIT
V_{IH} High Level Input Voltage		V_{CC} = 2.7-3.6V	1.5		V
V_{IL} Low Level Input Voltage		V_{CC} = 2.7-3.6V		0.8	V
I_{OH} High Level Output Current		V_{CC} = 2.7V		-12	mA
		V_{CC} = 3V		-24	mA
I_{OL} Low Level Output Current		V_{CC} = 2.7V		12	mA
		V_{CC} = 3 V		24	mA
V_{OH}	I_{OH} = -100µA	V_{CC} =1.65-3.6V	V_{CC}-0.2		V
	I_{OH} = -4mA	V_{CC} = 1.65V	1.2		V
	I_{OH} = -8mA	V_{CC} = 2.3V	1.7		V
	I_{OH} = -12mA	V_{CC} = 2.7V	2.2		V
		V_{CC} = 3V	2.4		V
	I_{OH} = -24mA	V_{CC} = 3V	2.2		V
V_{OL}	I_{OL} = 100µA	V_{CC} =1.65-3.6V		0.2	V
	I_{OL} = 4 mA	V_{CC} =1.65V		0.45	V
	I_{OL} = 8 mA	V_{CC} =2.3V		0.7	V
	I_{OL} = 12 mA	V_{CC} =2.7V		0.4	V
	I_{OL} = 24 mA	V_{CC} = 3V		0.55	V

Notes:

1. PIO-1149.1/E I/O pins and TAP signal outputs are pulled up to the user programmable voltage level through 1K pull-up resistors.

Physical

Unit Dimensions	3.0 in. × 4.0 in. × 1.0 in. (W×L×H)

JTAG and I/O Connector

JTAG and I/O Connector	15-pin D-SUB (receptacle) (AMP part no. 747845-4 or equivalent)
Mating Connector	15-pin D-SUB (pin) (AMP part no. 747306-3 or equivalent)

Power Requirements

5 V	500 mA maximum

Operating Environment

Temperature	0°C to 55°C
Relative Humidity	10% to 90%, non condensing

Storage Environment

Temperature	-40°C to 85°C

Chapter 2
PIO-1149.1/E Installation

The PIO-1149.1/E product consists of the user's manual, which you are currently reading, and the following other components:

- PIO-1149.1/E Boundary-Scan Controller Device
- PIO-1149.1/E Software Disk
- Host computer to PIO-1149.1/E device cable
- PIO-1149.1/E device to Target JTAG TAP cable
- 5 VDC Power Supply

Ensure all materials listed are present and free from visible damage or defects before proceeding. If anything appears to be missing or damaged, contact Corelis at the number listed on the front cover immediately.

System Requirements

The host PC requires Windows 95/98/NT/2000 Operating System.

The host PC or Laptop must have a parallel port interface that can operate in ECP mode (you configure ECP mode through your BIOS interface). Consult your PC documentation for directions on how to configure the port for ECP operation.

What's on the Disk

The Setup.exe program on the Installation disk will create the following directory structure and copy the files listed below:

Readme.txt

\Library\PIOE_SFL.h	Header File for PIO-1149.1/E Function Prototypes.
\Library\PIOE_SFL.lib	Microsoft Compiled Import Library
\Library\Win95_98\PIOE_SFL.dll	Microsoft Compiled DLL for Windows 95/98
\Library\WinNT_2K\PIOE_SFL.dll	Microsoft Compiled DLL for Windows NT/2000
\Library\Readme.txt	Description of SFL files.
\Drivers\Win95_98\PIO.vxd	Windows 95/98 Virtual Device Driver.
\Drivers\WinNT\PIO_1149.sys	Windows NT Kernel Mode Device Driver.
\Drivers\Win2K\PIO_1149.sys	Windows 2000 WDM Device Driver.
\Drivers\Win2K\PIO_1149.inf	Windows 2000 Device Information File.
\Example\Example.cpp	Example program to demonstrate how to use the PIO-1149.1/E's SFL.
\Example\Example.exe	Example program executable.
\Example\74bct8374.bsd	BSDL for the boundary-scan components in the example.
\Example\Readme.txt	Directions on how to compile the Example program.
\Self-Test\Test.exe	Self-test application program

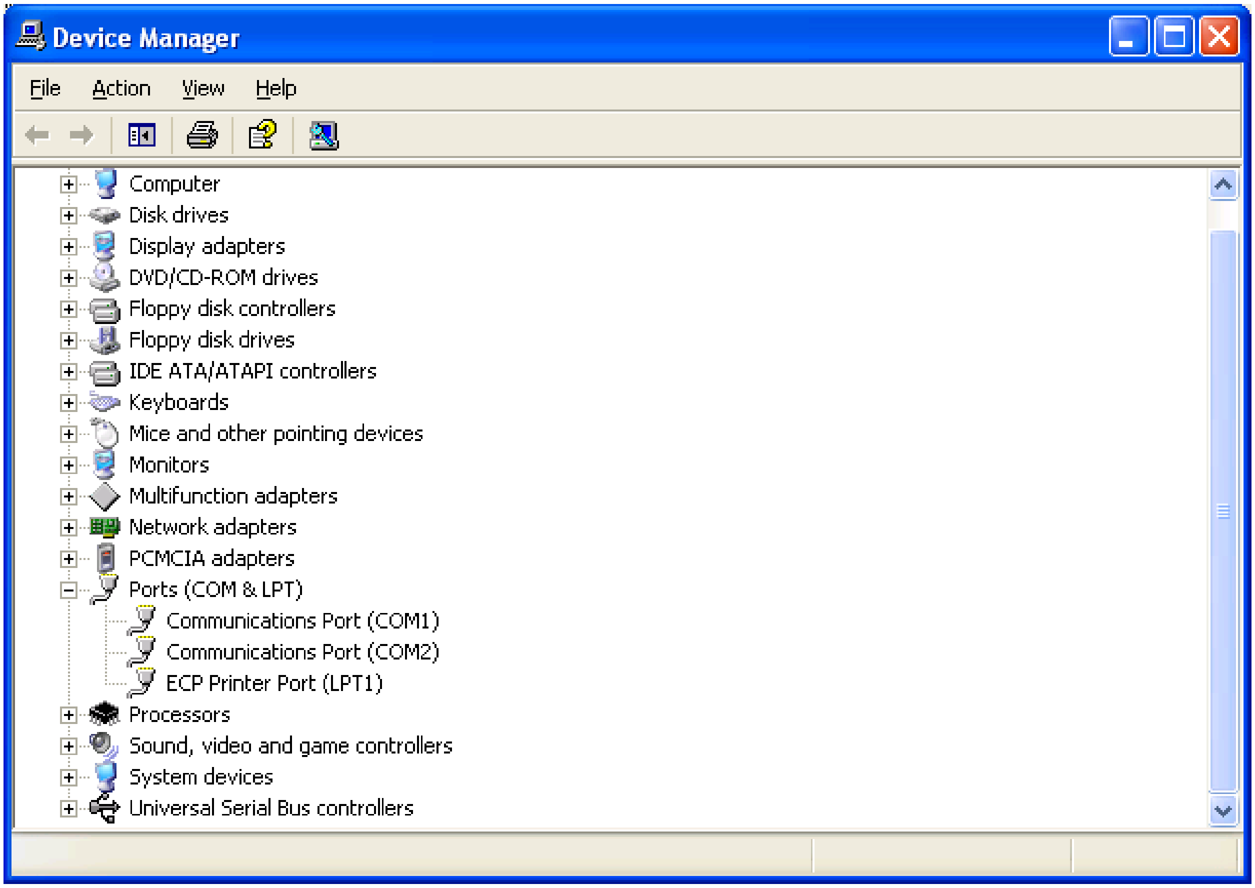
PIO-1149.1/E Software Installation

Run the Setup.exe utility on the PIO-1149.1 distribution diskette. Read the release notes that the utility installs with the software installation. The release notes provide the latest updates available and also provide additional information concerning how to verify proper host configuration.

PIO-1149.1/E Device Installation

To install the PIO-1149.1/E on your computer, perform the following steps:

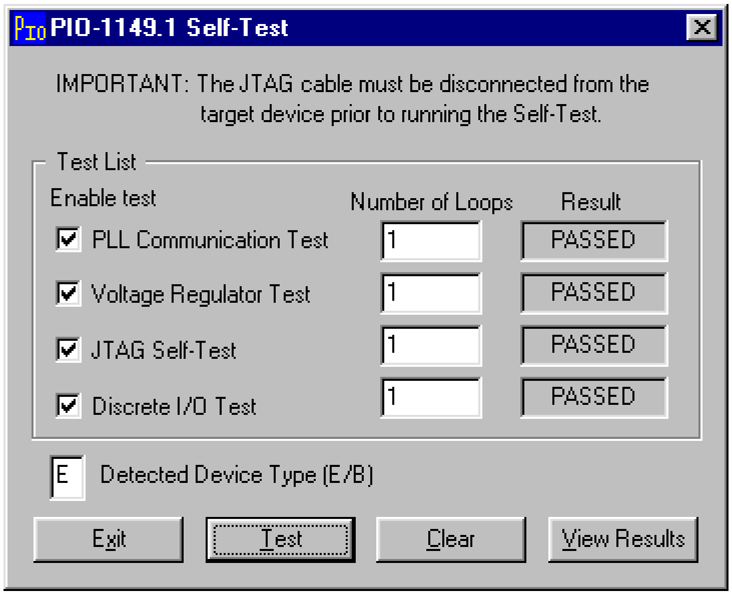
- Make sure that the parallel port on your PC is configured to operate in ECP mode. On most modern computers, parallel ports can be configured to work in ECP mode in the system BIOS.
- Check in Window device manger that the parallel port is configured to operate in ECP mode. Below is a picture of Windows XP device manger showing that parallel port (printer port in the picture) is configured to operate in ECP mode.



- Install one end of the host-computer-to-PIO-1149.1/E parallel port cable into the PIO-1149.1/E device and the other end into the parallel port connector on the computer.
- Apply power to the PIO-1149.1/E device by attaching the provided power supply.

Self-Test

After you have installed the software, run the Self-Test on the PIO-1149.1/E Device. To run the Self-Test, select and run \self-test\test.exe. A small pop-up should appear. Click on **Test** to run the self-test, and the results should look like Figure 2-1.



PIO-1149.1 Self-Test

IMPORTANT: The JTAG cable must be disconnected from the target device prior to running the Self-Test.

Test List

Enable test	Number of Loops	Result
☑ PLL Communication Test	1	PASSED
☑ Voltage Regulator Test	1	PASSED
☑ JTAG Self-Test	1	PASSED
☑ Discrete I/O Test	1	PASSED

Detected Device Type (E/B): E

Figure 2-1. PIO-1149.1/E Self-Test Results

ScanPlus Applications

When used with ScanPlus Runner, ScanPlus Debugger or ScanPlus Flash Programmer applications the PIO-1149.1/E can be used on the same parallel port as the security key (dongle). From the SETUP menu select CONTROLLER and then select the PIO-1149.1/E icon. Refer to the relevant ScanPlus tool user's manual for further details.

If the ScanPlus program is unable to locate either the dongle or the PIO-1149.1/E then:

- Turn the PIO-1149.1/E power OFF
- Disconnect the PIO-1149.1/E from the parallel port cable
- Remove the dongle from the parallel port connector of your computer
- Check the computer BIOS and verify that the parallel port is set to ECP mode
- Connect the dongle to the parallel port connector
- Connect the PIO-1149.1/E via the parallel port cable to the computer parallel port
- Turn the PIO-1149.1/E power ON
- Try the ScanPlus software again and see if the problem is fixed

Chapter 3
Connecting to the Target

Connectors

The 15-pin D-SUB JTAG interface connector includes all the signals for the boundary-scan TAP. This connector is located on the front panel of the PIO-1149.1/E device. Table 3-1 summarizes the pin-out of the JTAG connector.

Pin	Signal Name	I/O	Description
8	TRST*	Out	Test Reset Output
7	TDO	Out	Test Data Out - Channel 1 and 2
6	TDI	In	Test Data In- Channel 1 and 2
5	TMS1	Out	Test Mode Select - Channel 1
4	TCK	Out	Test Clock – Channel 1 and 2
3	PP0	Out/In	General Purpose Discrete I/O pin
10	PP1	Out/In	General Purpose Discrete I/O pin
2	PP2	Out/In	General Purpose Discrete I/O pin
9	PP3	Out/In	General Purpose Discrete I/O pin
1	TMS2	Out	Test Mode Select - Channel 2

Table 3-1. JTAG Connector Pin Assignment

Note: All other pins of the connector (11–15) are connected to ground (GND).

Connecting to the Target

The PIO-1149.1/E device can control the boundary-scan operations of two separate targets. On the PIO-1149.1/E's JTAG interface connector, each single-ended test signal terminates with an adjacent ground connection. Thus, the mating ribbon cable presents a fixed impedance to signals leaving the PIO-1149.1/E device or the user's target. Maintain this signal/ground configuration when connecting to the target(s).

The PIO-1149.1/E device ships with a 10-pin target interface cable for connection to a single target JTAG TAP. Table 3-2 summarizes the Netlist for this cable.

10-Pin Target Connector 3M part # 3473-6610	Signal Name	15-Pin JTAG Connector AMP part # 747306-3
1	TRST*	8
3	TDO	7
5	TDI	6
7	TMS1	5
9	TCK	4
NC	PP0	3
NC	PP1	10
NC	PP2	2
NC	PPI3	9
NC	TMS2	1
2	GND	15
4	GND	14
6	GND	13
8	GND	12
10	GND	11

Table 3-2. Supplied JTAG Cable Netlist

Designing a Target System

The PIO-1149.1/E connects to a target system through a 10-pin IEEE-1149.1 (JTAG) Port connector. The following section will describe the pin-out of this connector in detail.

Target Board Boundary Scan TAP Connector

The Boundary Scan Test Access Port (TAP) is a well-defined, IEEE-1149.1-compatible electrical interface between boundary-scan test equipment and the boundary-scan-compatible devices in the user's target board. Boundary-scan-based test equipment, such as the Corelis ScanPlus™ family of products, utilize a single TAP to interface to the UUT. This section explains how to implement a TAP connector that is compatible with most standard test equipment.

The boundary-scan TAP contains 5 signals: TCK, TMS, TDO, TDI and optionally TRST*. It also contains ground signal(s). Corelis recommends the industry-standard TAP connector shown in Figure 3-1. Note that each signal is terminated with a resistor to minimize signal cross talk in the interface cable and maximize noise immunity.

The connector on the user's target should have the standard flat cable compatible pinout. Figure 3-1 shows the top view of the target's 10 pin connector header (0.100" x 0.100" spacing).

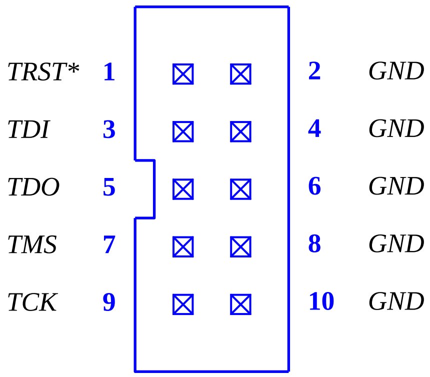


Figure 3-1. Standard TAP connector (top view)

Table 3-3 lists two 3M brand part numbers for the connector. Both are 0.100" x 0.100" headers, one with and one without latch/ejector. Equal connectors from other manufacturers are also acceptable.

Reference	Description	Manufacturer	Part Number
Target TAP	Straight header, 10 pin, 4 wall, with center notch	3M	30310-6002HB
	Latch/Ejector Straight header, 10 pin, 4 wall, with notch	3M	3448-3040

Table 3-3. Standard TAP Connectors

The following table shows the direction of the signals and terminating resistor values that Corelis recommends:

Pin	Signal	Direction	Termination
1	TRST*	Input to the UUT	1K pull-up (or 1.5K pull-down)
2	GND		
3	TDI	Input to the UUT	1K pull-up
4	GND		
5	TDO	Output of the UUT	33 ohm series
6	GND		
7	TMS	Input to the UUT	1K pull-up
8	GND		
9	TCK	Input to the UUT	1K pull-up
10	GND		

Note: Some target boards may require a pull-down resistor on the TRST* signal to assure normal device operations when not in boundary scan test mode.

Table 3-4. Signal Description and Termination

Figure 3-2 shows typical schematics of the target TAP connector and the recommended termination resistors. The 1K pull-up resistors can connect to any Vcc supply with nominal voltage between 2.7V to 5.0V. Recommended resistor values are +/- 5%.

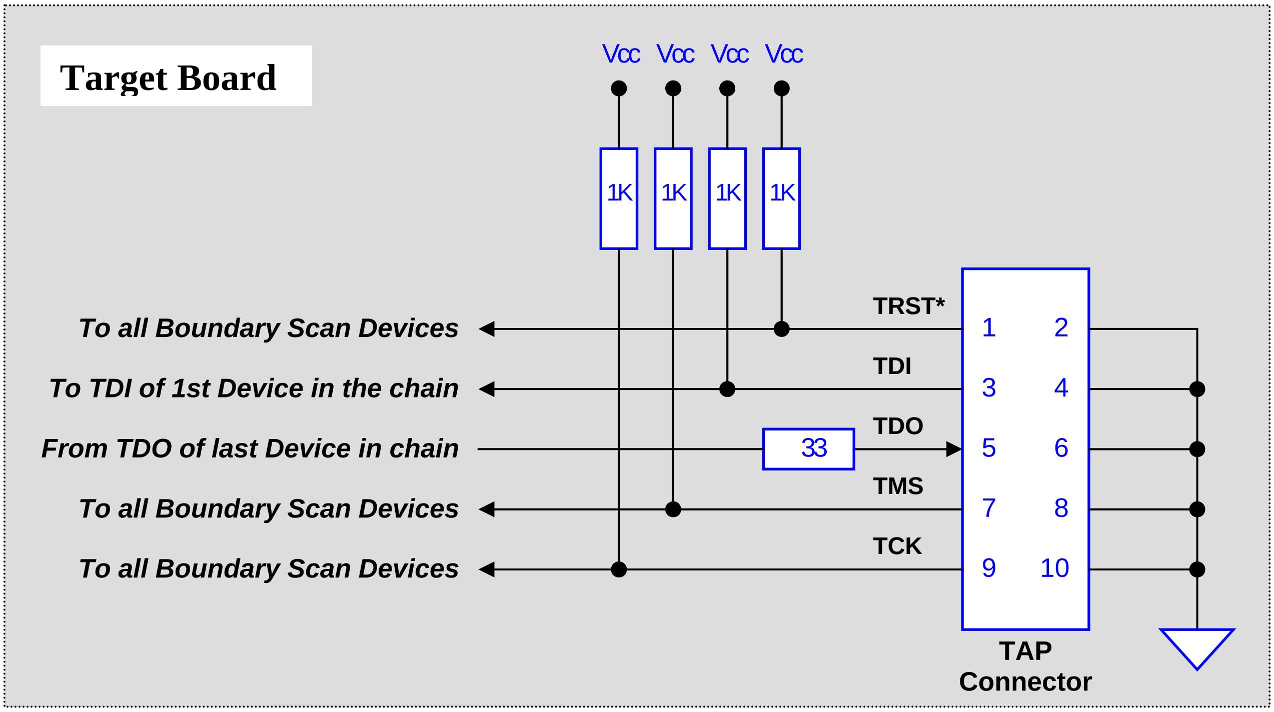

Target Board
Vcc
Vcc
Vcc
Vcc
1K
1K
1K
1K
TRST*
To all Boundary Scan Devices
TDI
To TDI of 1st Device in the chain
TDO
33
From TDO of last Device in chain
TMS
To all Boundary Scan Devices
TCK
To all Boundary Scan Devices
1
2
3
4
5
6
7
8
9
10
TAP Connector

Figure 3-2. TAP Connector Schematics

Chapter 4
Software Development

Notes on Usage

When designing programs to interface with the PIO-1149.1/E you dynamically link to the 32-bit Windows DLL.

Writing Applications for the PIO-1149.1/E Device

Before performing any testing operations, the application must first call hard_reset(Port). This will initialize the PIO-1149.1/E device for testing.

Example Test Code

The PIO-1149.1/E Software Disk contains an example test program \Example\Example.cpp for the Corelis JTAG Demo Board (a small boundary-scan board that contains four 74BCT8374 boundary-scan devices). The program first calls a test function to insure proper linkage to the DLL, and then it performs a hard reset of the PIO-1149.1/E Controller Device. After reset, it performs the following test:

- Two of the components, U3 and U4, are put into bypass mode
- Boundary-scan cells drive a walking-ones pattern onto the outputs of U1
- Boundary-scan cells receive the pattern on the inputs of U2.

The Corelis JTAG Demo Board is available for purchase. Please contact the Corelis sales department regarding the JTAG Demo Board (Corelis part number AS01210001-A0).

Enumerated Data Types

To make it more intuitive to pass information to the card, and to add a method of type checking, we have converted four classes of parameters to enumerated types:

- **TAP Selection.** When selecting a TAP, use the enumeration {`TAP1, TAP2`} as your selection parameter.
- **JTAG State Machine.** You may choose any of the stable states {`STATE_TLR, STATE_RTI, STATE_PDR, STATE_PIR, STATE_SDR, STATE_SIR`}.
- **JTAG TCK Frequency Selection.** The frequencies you may choose from range from 1 MHz to 40 MHz. Note the "_" character that must proceed the frequencies because of naming conventions in C/C++.
- **Output Voltage Selection.** There are 42 possible choices within the range of 1.25 V to 3.30 V in increments of 0.05 V; i.e. {`_1_25, _2_05, _2_10, …, …, _3_25, _3_30`}. You can supply additional entries to meet your requirements.

Before You Begin

Make sure you read the **readme.txt** on the PIO-1149.1/E Software disk before you begin, to find out about any last minute update information for the installation software and documentation.

Chapter 5
Scan Function Libraries

Summary

This section summarizes each function in the Scan Function Libraries (SFL). Detailed descriptions for each function follow the summary.

circulate_dr()

This function flushes out data from the selected target's Data Register (DR) by scanning in the selected bit length number of zeros. Then the function scans the flushed-out data back into the target's Data Register. In this way, you can read the target's Data Register without modification.

get_driver_info()

This function returns a pointer to a string that indicates the version number of the scan function library and the revision of the firmware and hardware in the PIO-1149.1/E device.

hard_reset()

This function performs a hard reset of all internal functions of the PIO-1149.1/E device and causes a transition of the target TAPs into the Test-Logic-Reset state. Call this function before any other scan function library function calls.

move_to_any_state()

This function causes a transition of the target JTAG device's state machine to the desired final state using the user specified transition path.

move_to_state()

This function causes a transition of the target JTAG device's state machine to the desired final stable state.

read_io()

This function reads the logical values that are sensed from the Discrete I/O pins.

scan_dr()

This function scans data from a specified array, out of the PIO-1149.1/E device and into the target JTAG device's Data Register (DR). It stores the data from the target JTAG device's Data Register (DR) in a specified array of the PIO-1149.1/E. The first bit scanned out is the LSB of the output array's first member. The first bit scanned in is stored in the LSB of the input array's first member. Following the scan operation, the device's JTAG state machine is in the Run-Test/Idle state.

scan_ir()

This function scans data from a specified array, out of the PIO-1149.1/E device and into the target JTAG device's Instruction Register (IR). It stores the data from the target JTAG device's Instruction Register (IR) in a specified array of the PIO-1149.1/E. The first bit scanned out is the LSB of the output array's first member. The first bit scanned in is stored in the LSB of the input array's first member. Following the scan operation, the device's JTAG state machine is in the Run-Test/Idle state.

scan_to_pause_dr()

This function scans data from a specified array, out of the PIO-1149.1/E device and into the target JTAG device's Data Register (DR). It stores the data from the target JTAG device's Data Register (DR) in a specified array of the PIO-1149.1/E. The first bit scanned out is the LSB of the output array's first member. The first bit scanned in is stored in the LSB of the input array's first member. Following the scan operation, the device's JTAG state machine is in the Pause-DR state.

scan_to_pause_ir()

This function scans data from a specified array, out of the PIO-1149.1/E device and into the target JTAG device's Instruction Register (IR). It stores the data from the target JTAG device's Instruction Register (IR) in a specified array of the PIO-1149.1/E. The first bit scanned out is the LSB of the output array's first member. The first bit scanned in is stored in the LSB of the input array's first member. Following the scan operation, the device's JTAG state machine is in the Pause-IR state.

set_io()

This function sets the output levels of the selected Discrete I/O pins.

set_scan_clk()

This function sets the TCK clock speed for JTAG operations. Note that the TCK is only present during tms_reset(), move_to_state(), move_to_any_state() and scan operations.

set_tri()

This function sets the values applied to the selected tri-state pins to control the operation of the Discrete I/O and JTAG TAP pins.

set_trst()

This function sets the target's TRST* pin to the specified level.

set_voltage()

This function sets the output voltage for the Discrete I/O and JTAG TAP pins.

test()

This function is a simple test of the application program's ability to execute library function calls. It returns the unsigned character passed into the function.

tms_reset()

This function holds the TMS signal high for 5 TCKS to put the target's JTAG state machine into Test-Logic-Reset state.

trst_reset()

This function sets the TRST* signal low for 1 millisecond, then sets it back high.

Detailed Descriptions

Function: circulate_dr()

Description:

This function flushes out data from the selected target's Data Register (DR) by scanning in the selected bit length number of zeros. Then the function scans the flushed-out data back into the target's Data Register. In this way, you can read the target's Data Register without modification.

Prototype:

```
unsigned char circulate_dr(enum TAPS test_bus, unsigned long
                           bit_length, unsigned short *in_data);
```

Input Parameters:

`test_bus`: Which TAP to scan.
`{TAP1, TAP2}`

`bit_length`: Unsigned long (32-bits) holds the number of bits to be shifted out the PIO-1149.1/E device.

`in_data`: Unsigned short integer pointer to an array of data words to hold the data shifted into the PIO-1149.1/E device.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
enum TAPS test_bus;         // Which TAP to scan
unsigned long bit_length;   // Number of bits to be shifted
unsigned short in_data[10];// Array to hold data shifted to card
unsigned char result;       // Return code

result = circulate_dr(test_bus, bit_length, in_data);
```

Function: get_driver_info()

Description:

This function returns a pointer to a string that indicates the version number of the scan function library and the revision level of the firmware and hardware.

Prototype:

```
unsigned char get_driver_info(char **info_string);
```

Input Parameters:

`info_string:`	A pointer to a string that indicates the version number of the scan function library and the revision level of the firmware and hardware.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
char *info_string;          // buffer to hold return data
unsigned char result;       // Return code

result = get_driver_info(&info_string);

printf("%s \n",info_string);
```

Function: hard_reset()

Description:

This function performs a hard reset of all internal functions of the PIO-1149.1/E device and causes a transition of the target TAPs into the Test-Logic-Reset state. Call this function before any other scan function library function calls. The hard_reset() function will tri-state all discrete I/O pins and enable all JTAG TAP pins, set the JTAG TCK frequency to 1MHz, set the voltage at the TAP pins, and perform a TMS reset on JTAG TAPs 1 and 2.

Prototype:

```
usigned char hard_reset(unsigned char Port, unsigned char Voltage);
```

Input Parameters:

`Port:`

Windows 95/98 - A Port value of one (1) will specify to use the port configured as LPT1 (this Port usually has input/output range 0x378-0x37F). A Port value of two (2) will specify to use the port configured as LPT2.

Windows NT/2K – A Port value of one (1) will communicate with the PIO-1149.1 device through input/output range 0x378–0x37F. A Port value of two (2) will communicate with the PIO-1149.1 through the input/output range 0x278–0x27F.The user must verify that the LPT1 port or LPT2 port is configured for the proper input/output ranges listed above.

`Voltage:`

`Output voltage applied to the JTAG TAP pins.`

Return Values:

0x00	Success
0xFF	Failure occurred during function cal.

Example Call:

```
if(hard_reset())
exit(1);
```

Function: move_to_any_state()

Description:

This function causes a transition of the target JTAG device's state machine to the desired final state using the user supplied transition path. States in the transition path must form a valid state transition path and must start from the current state.

NOTES:

1. Maximum number of states in the user specified path is 16.
2. In order for following scan functions to work correctly the move_to_any_state() function (or consecutive move_to_any_state functions) must end the TAP parked in one of the following stable state:
    - TEST-LOGIC-RESET
    - RUN-TEST-IDLE
    - PAUSE-DR
    - PAUSE-IR.

Prototype:

```
unsigned char move_to_any_state(enum TAPS test_bus,
                                enum STATES final_state,
                                int path_count,
                                enum STATES transition_path[])
```

Input Parameters:

`test_bus:` Which TAP to scan.

```
{TAP1, TAP2}
```

`final_state:` Enumeration that specifies the final JTAG state:

```
STATE_TLR  // TEST-LOGIC RESET
STATE_RTI  // RUN-TEST/IDLE
STATE_PDR  // PAUSE-DR
STATE_PIR  // PAUSE-IR
STATE_SDR  // SHIFT-DR
STATE_SIR  // SHIFT-IR
STATE_E1DR // EXIT1-DR
STATE_E1IR // EXIT1-IR
STATE_SDRS // SELECT-DR-SCAN
STATE_SIRS // SELECT-IR-SCAN
STATE_CDR  // CAPTURE-DR
STATE_CIR  // CAPTURE-IR
STATE_E2DR // EXIT2-DR
```

```
STATE_E2IR // EXIT2-IR
STATE_UDR  // UPDATE-DR
STATE_UIR  // UPDATE-IR
```

`path_count:` Number of states in the following parameter.

`transition_path:` An array of state enumerations which forms a valid state transition.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
enum TAPS test_bus = TAP1;       // Which TAP to transition
enum STATES final_state = STATE_RTI;// Final transition state
int  path_count;                 // number states in the path
enum STATES transition_path[7] =
            {STATE_TLR /* current state */,
             STATE_RTI, STATE_SDRS, STATE_CDR, STATE_E1DR, STATE_UDR,
             final_state}; // states in the path

tms_reset (test_bus); // current state must be Test-Logic-Reset
move_to_any_state(test_bus, final_state, 7, transition_path);
```

Function: move_to_state()

Description:

This function causes a transition of the target JTAG device's state machine to the desired final stable state.

Prototype:

```
unsigned char move_to_state(enum TAPS test_bus, enum STATES
                            final_state)
```

Input Parameters:

`test_bus:`	Which TAP to scan. `{TAP1, TAP2}`

`final_state:`	Enumeration that specifies the final JTAG state:
	`STATE_TLR  // TEST-LOGIC RESET`
	`STATE_RTI  // RUN-TEST/IDLE`
	`STATE_PDR  // PAUSE-DR`
	`STATE_PIR  // PAUSE-IR`
	`STATE_SDR  // SHIFT-DR`
	`STATE_SIR  // SHIFT-IR`

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
enum TAPS test_bus;           // Which TAP to transition
enum STATES final_state;      // Final transition state
unsigned char result;         // Return code

result = move_to_state(test_bus, final_state);
```

Function: read_io()

Description:

This function reads the logical values that are sensed from the Discrete I/O pins.

Prototype:

```
unsigned char read_io(unsigned char *input_data)
```

Input Parameters:

`input_data:` An unsigned character pointer to the value sensed from the Discrete I/O pins.

The format of the returned value is:

Bit 0: Value sensed from PP0 (JTAG connector pin 3)

Bit 1: Value sensed from PP1 (JTAG connector pin 10)

Bit 2: Value sensed from PP2 (JTAG connector pin 2)

Bit 3: Value sensed from PP3 (JTAG connector pin 9)

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
unsigned char *input_data; // Value sensed from Discrete I/O pins
unsigned char result;      // Return Code

result = read_io(input_data);
```

Function: scan_dr()

Description:

This function scans data from a specified array, out of the PIO-1149.1/E device and into the target JTAG device's Data Register (DR). It stores the data from the target JTAG device's Data Register (DR) in a specified array of the PIO-1149.1/E. The first bit scanned out is the LSB of the output array's first member. The first bit scanned in is stored in the LSB of the input array's first member. Following the scan operation, the device's JTAG state machine is in the Run-Test/Idle state.

Prototype:

```
unsigned char scan_dr(enum TAPS test_bus, unsigned short *out_data,
                      unsigned long bit_length, unsigned short
                      *in_data);
```

Input Parameters:

`test_bus:`	Which TAP to scan. `{TAP1, TAP2}`
`out_data:`	Unsigned short pointer to an array of data words to be shifted out the PIO-1149.1/E device.
`bit_length:`	Unsigned long (32-bits) holds the number of bits to be shifted out the PIO-1149.1/E device.
`in_data:`	Unsigned short pointer to an array of data words to hold the data shifted into the PIO-1149.1/E device.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
enum TAPS test_bus;              // Which TAP to Scan
unsigned short out_data[10];     // Data shifted out of card
unsigned long bit_length;        // Number of bits to be shifted
unsigned short in_data[10];      // Data to be shifted into card
unsigned char result;            // Return Code

result = scan_dr(test_bus, out_data, bit_length, in_data);
```

Description:

This function scans data from a specified array, out of the PIO-1149.1/E device and into the target JTAG device's Instruction Register (IR). It stores the data from the target JTAG device's Instruction Register (IR) in a specified array of the PIO-1149.1/E. The first bit scanned out is the LSB of the output array's first member. The first bit scanned in is stored in the LSB of the input array's first member. Following the scan operation, the device's JTAG state machine is in the Run-Test/Idle state.

Prototype:

```
unsigned char scan_ir(enum TAPS test_bus, unsigned short *out_data,
                      unsigned long bit_length, unsigned short
                      *in_data)
```

Input Parameters:

`test_bus:` Which TAP to scan.
`{TAP1, TAP2}`

`out_data:` Unsigned short pointer to an array of data words to be shifted out the PIO-1149.1/E device.

`bit_length:` Unsigned long (32-bits) holds the number of bits to be shifted out the PIO-1149.1/E device.

`in_data:` Unsigned short pointer to an array of data words to hold the data shifted into the PIO-1149.1/E device.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
enum TAPS test_bus;              // Which TAP to Scan
unsigned short out_data[10];     // Data shifted out of card
unsigned long bit_length;        // Number of bits to be shifted
unsigned short in_data[10];      // Data to be shifted into card
unsigned char result;            // Return Code

result = scan_ir(test_bus, out_data, bit_length, in_data);
```

Function: scan_to_pause_dr()

Description:

This function scans data from a specified array, out of the PIO-1149.1/E device and into the target JTAG device's Data Register (DR). It stores the data from the target JTAG device's Data Register (DR) in a specified array of the PIO-1149.1/E. The first bit scanned out is the LSB of the output array's first member. The first bit scanned in is stored in the LSB of the input array's first member. Following the scan operation, the device's JTAG state machine is in the Pause-DR state.

Prototype:

```
unsigned char scan_to_pause_dr(enum TAPS test_bus, const unsigned
                               short *out_data, unsigned long bit_length,
                               unsigned short *in_data);
```

Input Parameters:

`test_bus:`	Which TAP to scan. `{TAP1, TAP2}`
`out_data:`	Unsigned short pointer to an array of data words to be shifted out the PIO-1149.1/E device.
`bit_length:`	Unsigned long (32-bits) holds the number of bits to be shifted out the PIO-1149.1/E device.
`in_data:`	Unsigned short pointer to an array of data words to hold the data shifted into the PIO-1149.1/E device.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
enum TAPS test_bus;              // Which TAP to Scan
unsigned short out_data[10];     // Holds data shifted out of card
unsigned long bit_length;        // Number of bits to be shifted
unsigned short in_data[10];      // Data to be shifted into card
unsigned char result;            // Return Code

result = scan_to_pause_dr(test_bus, out_data, bit_length, in_data);
```

Function: scan_to_pause_ir()

Description:

This function scans data from a specified array, out of the PIO-1149.1/E device and into the target JTAG device's Instruction Register (IR). It stores the data from the target JTAG device's Instruction Register (IR) in a specified array of the PIO-1149.1/E. The first bit scanned out is the LSB of the output array's first member. The first bit scanned in is stored in the LSB of the input array's first member. Following the scan operation, the device's JTAG state machine is in the Pause-IR state.

Prototype:

```
unsigned char scan_to_pause_ir(enum TAPS test_bus, const unsigned
                               short *out_data, unsigned long bit_length,
                               unsigned short *in_data);
```

Input Parameters:

`test_bus:`	Which TAP to scan. {TAP1, TAP2}
`out_data:`	Unsigned short pointer to an array of data words to be shifted out the PIO-1149.1/E device.
`bit_length:`	Unsigned long (32-bits) holds the number of bits to be shifted out the PIO-1149.1/E device.
`in_data:`	Unsigned short pointer to an array of data words to hold the data shifted into the PIO-1149.1/E device.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
enum TAPS test_bus;             // Which TAP to Scan
unsigned short out_data[10];    // Holds data shifted out of card
unsigned long bit_length;       // Number of bits to be shifted
unsigned short in_data[10];     // Data to be shifted into card
unsigned char result;           // Return Code

result = scan_to_pause_ir(test_bus, out_data, bit_length, in_data)
```

Function: set_io()

Description:

This function sets the output levels of the selected Discrete I/O pins.

Prototype:

```
unsigned char set_io(unsigned char output_data);
```

Input Parameters:

`output_data:` The value driven onto the Discrete I/O pins.

The format of the value is:

Bit 0: Value driven on PP0 (JTAG connector pin 3)

Bit 1: Value driven on PP1 (JTAG connector pin 10)

Bit 2: Value driven on PP2 (JTAG connector pin 2)

Bit 3: Value driven on PP3 (JTAG connector pin 9)

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
unsigned char output_data;       // Value driven on Discrete I/O pins
unsigned char result;            // Return Code

result = set_io(output_data);
```

Function: set_scan_clk()

Description:

This function sets the TCK clock speed for the JTAG TAP. Note that the TCK is only present during tms_reset(), move_to_state(), move_to_any_state() and scan operations.

Prototype:

```
unsigned char set_scan_clk(enum FREQUENCEY freq);
```

Input Parameters:

`clk_select:` Enumerator that specifies which base clock oscillator to use:

`_1_0MHZ`

`_1_5MHZ`

…

`_40_0MHZ`

Return Values:

0x00	Success
0xFF	Failure occurred during function call.

Example Call:

```
enum OSCILLATORS clk_select;      // Which clock oscillator to use
unsigned char clk_divider;        // Used as clock divisor
unsigned char result;             // Return Code

clk_select = _40MHZ;              // Defined in PIO_SFL.h
clk_divider = CLK_DIV_BY_2;       // Defined in PIO_SFL.h

set_scan_clk(clk_select, clk_divider);
```

Function: set_tri()

Description:

This function sets the values applied to the selected tri-state pins to control the operation of the Discrete I/O and JTAG TAP pins.

Prototype:

```
unsigned char set_tri(unsigned char output_data);
```

Input Parameters:

`output_data:`

The value applied to the tri-state pins. A 1 in a selected bit position will tri-state the corresponding pin and a 0 in a bit position will enable the corresponding pin for output.

The format of the output_data value is:

Bit 0: Tri-State the PP0 pin (JTAG connector pin 3)

Bit 1: Tri-State the PP1 pin (JTAG connector pin 10)

Bit 2: Tri-State the PP2 pin (JTAG connector pin 2)

Bit 3: Tri-State the PP3 pin (JTAG connector pin 9)

Bit 4: Tri-State the JTAG TAP pins (JTAG connector pins 8,7,5,4,1)

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
unsigned char output_data;        // Value applied on tri-state pins
unsigned char result;             // Return Code

result = set_tri(output_data);
```

Description:

This function sets the target's TRST* pin to the specified level.

Prototype:

```
unsigned char set_trst(unsigned char output_data);
```

Input Parameters:

The value applied to the Target's TRST* pin.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
unsigned char result;                   // Return Code

/* Set TRST* pin Low */
result = set_trst(0);

/* Set TRST* pin High */
result = set_trst(1);
```

Function: set_voltage()

Description:

This function sets the output voltage for the Discrete I/O and JTAG TAP pins. Voltages can be set within the ranges of 1.25 volts to 3.30 volts. An enumerated type is provided in the header file to simplify the selection value.

Prototype:

```
unsigned char set_voltage(unsigned char output_data);
```

Input Parameters:

`output_data:`	The desired output voltage to be applied to the Discrete I/O and JTAG TAP pins.

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
unsigned char output_data=_3_40;      // New voltage selection
unsigned char result;                 // Return Code

result = set_tri(output_data);
```

Function: test()

Description:

This function is a simple test of the application program's ability to execute library function calls. It simply returns the unsigned character passed into the function.

Prototype:

```
unsigned char test(unsigned char dummy)
```

Input Parameters:

`dummy:`	Any valid unsigned character value.

Return Values:

If the function returns the input parameter correctly, the DLL is present and accessible. If it returns anything else, then the program is unable to access the DLL correctly.

Example Call:

```
unsigned char dummy;            // Value to return

if(test(dummy) != dummy)
return 0;                               // error
```

Function: tms_reset()

Description:

This function holds the TMS signal high for 5 TCKS to put the target's JTAG state machine into Test-Logic-Reset state.

Prototype:

```
unsigned char tms_reset(enum TAPS test_bus);
```

Input Parameters:

`test_bus:`	Which TAP to reset. `{TAP1, TAP2}`

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
enum TAPS test_bus = TAP1;             // Which TAP to reset
unsigned char result;                  // Return Code


result = tms_reset(test_bus);
```

Function: trst_reset()

Description:

This function sets the TRST* signal low for 1 millisecond, then sets it back high.

Prototype:

```
unsigned char trst_reset(void);
```

Input Parameters:

NONE

Return Values:

0x00	Success
0xFF	Failure occurred during function call

Example Call:

```
unsigned char result;                    // Return Code

result = trst_reset();
```