SERIAL CABLES

serialcables-atlas3

Python API

User's Manual

Atlas3 PCIe Gen6 Host Adapter Card

PCI6-AD-x16HI-BG6-144

Version 0.1.5
December 2025

Serial Cables, LLC
8811 American Way, Suite 110
Englewood, CO 80107

www.serialcables.com

Table of Contents

1. Introduction

1.1 Overview

The serialcables-atlas3 Python API provides programmatic control of the Serial Cables Atlas3 PCIe Gen6 Host Adapter Card (PCI6-AD-x16HI-BG6-144). This library enables test engineers to automate hardware validation workflows, monitor system health, and integrate PCIe Gen6 test infrastructure into automated test frameworks.

The Atlas3 Host Adapter Card features the Broadcom PEX90144 PCIe Gen6 Switch, supporting up to 144 lanes and 72 ports with full PCIe Gen6 (64 GT/s) capability including FLIT mode and Forward Error Correction (FEC).

1.2 Key Features

- Full Python API wrapping all MCU CLI commands
- Type-safe implementation using dataclasses and enums
- Context manager support for automatic connection handling
- Comprehensive error handling with descriptive exceptions
- Built-in command-line interface (atlas3-cli)
- Cross-platform support (Linux, Windows, macOS)
- Python 3.8+ compatibility

1.3 Intended Audience

This manual is intended for PCIe and NVMe test/validation engineers who need to:

- Automate PCIe Gen6 device testing and validation
- Build custom test frameworks and dashboards
- Integrate Atlas3 hardware into CI/CD pipelines
- Monitor and diagnose PCIe link issues programmatically
- Develop automated compliance test suites

1.4 System Requirements

- Python 3.8 or higher
- pyserial >= 3.5
- USB-C connection to Atlas3 CN7 (MCU CLI) port
- Operating System: Linux, Windows, or macOS

2. Hardware Overview

2.1 Atlas3 Host Adapter Card

The Atlas3 Host Adapter Card is a PCIe Gen6 switch-based adapter featuring the Broadcom PEX90144 silicon. It provides multiple downstream MCIO connectors for connecting NVMe devices and other PCIe endpoints for testing and validation.

2.1.1 Switch Capabilities

- Up to 144 lanes and 72 ports
- PCIe Gen6 (64 GT/s) with FLIT mode support
- Forward Error Correction (FEC)
- Ultra-low latency operation
- Dynamic Port Reconfiguration (DPR)
- Hot-plug support

2.2 Connector Reference

The following table describes the physical connectors on the Atlas3 card:

Connector	Description	Ports	API Connector #
CN1	PCIe Straddle mount (x16)	Port 80	4 (CON4)
CN2	MCIO x8 (EXT upper)	Ports 120-127	1 (CON1)
CN3	MCIO x8 (EXT lower)	Ports 112-119	0 (CON0)
CN4	MCIO x8 (INT upper)	Ports 136-143	3 (CON3)
CN5	MCIO x8 (INT lower)	Ports 128-135	2 (CON2)
CN6	USB-C (SDB/SMART)	—	—
CN7	USB-C (MCU CLI)	—	—

Note: CN7 is the USB-C port used by the serialcables-atlas3 API for communication.

2.3 Station and Port Mapping

The PEX90144 switch organizes ports into stations. Understanding this mapping is essential for interpreting port status and configuring the card:

Station	Connector	Ports	Description
STN2	—	32	Golden Finger (Upstream)
STN5	CON4	80	PCIe Straddle
STN7	CON0	112-118	EXT MCIO upper
STN7	CON1	120-126	EXT MCIO lower
STN8	CON2	128-134	INT MCIO upper
STN8	CON3	136-142	INT MCIO lower

2.4 LED Indicators

2.4.1 Status LEDs

LED	Color	Description
SLED1	Red	System error indicator
SLED2	Green	Heartbeat (blinking = managed FW running)
SLED3	Green	MCU status (blinking = MCU running)

2.4.2 Link Speed LEDs (Blue)

Blink Rate	PCIe Speed
0.25 Hz	Gen1 (2.5 GT/s)

Blink Rate	PCIe Speed
0.5 Hz	Gen2 (5.0 GT/s)
1.0 Hz	Gen3 (8.0 GT/s)
2.0 Hz	Gen4 (16.0 GT/s)
4.0 Hz	Gen5 (32.0 GT/s)
Solid ON	Gen6 (64.0 GT/s)

3. Installation

3.1 Installing from PyPI

The recommended installation method is via pip from the Python Package Index:

```
pip install serialcables-atlas3
```

3.2 Installing from Source

To install from source for development or to get the latest changes:

```
git clone https://github.com/serialcables/serialcables-atlas3.git
cd serialcables-atlas3
pip install -e .
```

3.3 Development Installation

To install with development dependencies for testing and code quality tools:

```
pip install -e ".[dev]"
```

This includes pytest, black, isort, mypy, and ruff for testing and code formatting.

3.4 Verifying Installation

Verify the installation by checking the version:

```
python -c "import serialcables_atlas3;
print(serialcables_atlas3.__version__)"
```

Or use the CLI to check available devices:

```
atlas3-cli --version
```

3.5 Serial Port Permissions (Linux)

On Linux systems, you may need to add your user to the dialout group to access serial ports:

```
sudo usermod -a -G dialout $USER
# Log out and back in for changes to take effect
```

4. Quick Start Guide

4.1 Hardware Setup

1. Install the Atlas3 card in a PCIe slot with adequate power.
2. Connect the 12V power cable to CN9.
3. Connect a USB-C cable from CN7 (MCU CLI) to your host computer.
4. Power on the host system.
5. Identify the serial port (e.g., /dev/ttyUSB0 on Linux, COM3 on Windows).

4.2 Basic Python Usage

The following example demonstrates basic API usage with the context manager pattern:

```
from serialcables_atlas3 import Atlas3

# Connect using context manager (recommended)
with Atlas3("/dev/ttyUSB0") as card:
    # Get version information
    info = card.get_version()
    print(f"Model: {info.model}")
    print(f"MCU Version: {info.mcu_version}")

    # Get host card status
    status = card.get_host_card_info()
    print(f"Temperature:
{status.thermal.switch_temperature_celsius}°C")
    print(f"Power: {status.power.load_power}W")

    # Get port status
    ports = card.get_port_status()
    for port in ports.ext_mcio_ports:
        if port.is_linked:
            print(f"Port {port.port_number}: {port.speed}
x{port.width}")

    # Run built-in self-test
    bist = card.run_bist()
    print(f"BIST: {'PASS' if bist.all_passed else 'FAIL'}")
```

4.3 Using the CLI

The atlas3-cli command provides quick access to common operations:

```
# Show version information
atlas3-cli -p /dev/ttyUSB0 version

# Show host card status
atlas3-cli -p /dev/ttyUSB0 status

# Show port status
atlas3-cli -p /dev/ttyUSB0 ports

# Run BIST
atlas3-cli -p /dev/ttyUSB0 bist

# Output as JSON for scripting
atlas3-cli -p /dev/ttyUSB0 --json status
```

5. API Reference

5.1 Atlas3 Class

The Atlas3 class is the main interface for communicating with the host adapter card.

5.1.1 Constructor

```
Atlas3(port, baudrate=115200, timeout=5.0, auto_connect=True)
```

Parameters:

- port (str): Serial port path (e.g., "/dev/ttyUSB0" or "COM3")
- baudrate (int): Serial baudrate (default: 115200)
- timeout (float): Command timeout in seconds (default: 5.0)
- auto_connect (bool): Automatically connect on initialization (default: True)

5.1.2 Connection Methods

Method	Description
`connect()`	Establish connection to the Atlas3 device
`disconnect()`	Close the connection to the device
`is_connected`	Property: Returns True if connected
`find_devices()`	Static method: Find available Atlas3 devices

5.1.3 System Information Methods

Method	Description
`get_version()`	Returns VersionInfo with product and firmware details
`get_host_card_info()`	Returns HostCardInfo (thermal, fan, voltages, power)
`get_system_info()`	Returns complete system information string

5.1.4 Port Status Methods

Method	Description
`get_port_status()`	Returns PortStatus for all ports
`get_error_counters()`	Returns AllErrorCounters with per-port statistics
`clear_error_counters()`	Clears all error counters, returns bool

5.1.5 Configuration Methods

Method	Description
`get_mode()`	Returns current OperationMode (1-4)
`set_mode(mode)`	Set operation mode (requires reset)
`get_spread_status()`	Returns SpreadStatus (SSC configuration)
`set_spread(mode)`	Set clock spread (off, 2500PPM, 5000PPM)
`get_clock_status()`	Returns ClockStatus for all connectors
`set_clock_output(enable)`	Enable/disable clock output
`get_flit_status()`	Returns FlitStatus for all stations
`set_flit_mode(station, disable)`	Configure FLIT mode per station

5.1.6 Reset Methods

Method	Description
`reset_connector(con)`	Send PERST# to connector (0-4 or "all")
`reset_mcu()`	Reset the on-board MCU
`run_bist()`	Run built-in self-test, returns BistResult

5.2 Data Models

The API uses dataclasses to provide structured, type-safe responses. Key models include:

- VersionInfo: Product and firmware version information
- HostCardInfo: Thermal, fan, voltage, and power readings
- PortStatus: Link status for all ports
- PortInfo: Individual port details (speed, width, status)
- AllErrorCounters: Per-port error statistics
- BistResult: Built-in self-test results
- ClockStatus, SpreadStatus, FlitStatus: Configuration states

5.3 Enumerations

5.3.1 LinkSpeed

PCIe link speed generations:

- GEN1: 2.5 GT/s
- GEN2: 5.0 GT/s
- GEN3: 8.0 GT/s
- GEN4: 16.0 GT/s
- GEN5: 32.0 GT/s
- GEN6: 64.0 GT/s

5.3.2 LinkStatus

Port link status values:

- ACTIVE: Link operating at desired speed and width
- DEGRADED: Link up but not at maximum capability
- IDLE: No physical link detected

5.3.3 OperationMode

Host card operation modes:

- MODE_1: Common clock, precoding enabled
- MODE_2: Common clock, precoding disabled
- MODE_3: SRNS, precoding enabled
- MODE_4: SRNS, precoding disabled

6. Usage Examples

6.1 Monitoring Port Health

This example demonstrates continuous monitoring of port status and error counters:

```
from serialcables_atlas3 import Atlas3
import time

with Atlas3("/dev/ttyUSB0") as card:
    while True:
        ports = card.get_port_status()
        counters = card.get_error_counters()

        # Check for degraded links
        for port in ports.ext_mcio_ports:
            if port.is_degraded:
                print(f"WARNING: Port {port.port_number} degraded")
                print(f"  Speed: {port.speed} (max:
{port.max_speed.value})")

        # Check for errors
        if counters.total_errors > 0:
            for c in counters.ports_with_errors:
                print(f"Port {c.port_number}: {c.total_errors}
errors")

        time.sleep(60)
```

6.2 Automated Device Reset Sequence

Example of resetting devices and verifying link re-establishment:

```
from serialcables_atlas3 import Atlas3
import time

def reset_and_verify(card, connector):
    """Reset a connector and verify link comes back up."""
    # Get initial state
    before = card.get_port_status()

    # Send reset
    card.reset_connector(connector)
    time.sleep(2)  # Wait for link training

    # Verify link status
    after = card.get_port_status()

    # Compare and report
    for port in after.ext_mcio_ports:
        if port.is_linked:
            print(f"Port {port.port_number}: Link UP at
{port.speed}")

with Atlas3("/dev/ttyUSB0") as card:
    reset_and_verify(card, 0)  # Reset CON0
```

6.3 Exporting Error Data

Export error counters to various formats for analysis:

```
from serialcables_atlas3 import Atlas3

with Atlas3("/dev/ttyUSB0") as card:
    counters = card.get_error_counters()

    # Export to JSON
    json_str = counters.to_json(indent=2)
    with open("errors.json", "w") as f:
        f.write(json_str)

    # Export to pandas DataFrame
    df = counters.to_dataframe()
    errors_df = df[df['has_errors'] == True]
    errors_df.to_csv("errors.csv")
```

6.4 Configuration for Gen6 Testing

Configure the card for PCIe Gen6 validation testing:

```
from serialcables_atlas3 import Atlas3, OperationMode, SpreadMode

with Atlas3("/dev/ttyUSB0") as card:
    # Set operation mode for Gen6
    card.set_mode(OperationMode.MODE_1)

    # Disable spread spectrum for cleaner signal
    card.set_spread(SpreadMode.OFF)

    # Enable FLIT mode on all stations (required for Gen6)
    card.set_flit_mode("all", disable=False)

    # Enable clock output
    card.set_clock_output(True)

    print("Gen6 configuration complete")
    print("Note: Reset controller for mode change to take effect")
```

7. Error Handling

7.1 Exception Hierarchy

The API provides a comprehensive exception hierarchy for error handling:

- Atlas3Error: Base exception for all library errors
- ConnectionError: Connection to device failed
- CommandError: CLI command failed or returned error
- TimeoutError: Command timed out waiting for response
- InvalidParameterError: Invalid parameter passed to method
- ParseError: Failed to parse device response

7.2 Error Handling Example

```
from serialcables_atlas3 import (
    Atlas3,
    Atlas3Error,
    ConnectionError,
    CommandError,
    TimeoutError,
    InvalidParameterError,
)

try:
    with Atlas3("/dev/ttyUSB0") as card:
        card.set_mode(5)  # Invalid mode
except InvalidParameterError as e:
    print(f"Invalid parameter: {e}")
except ConnectionError as e:
    print(f"Connection failed: {e}")
except TimeoutError as e:
    print(f"Command timed out: {e}")
except CommandError as e:
    print(f"Command failed: {e}")
except Atlas3Error as e:
    print(f"Atlas3 error: {e}")
```

8. Command-Line Interface Reference

8.1 Global Options

Option	Description
`-p, --port`	Serial port (required)
`-b, --baudrate`	Serial baudrate (default: 115200)
`-t, --timeout`	Command timeout in seconds (default: 5.0)
`-j, --json`	Output in JSON format

8.2 Commands

Command	Description
`find`	Find available Atlas3 devices
`version`	Show version and product information
`status`	Show host card status (thermal, voltages, power)
`ports`	Show port link status
`counters`	Show or clear error counters (--clear to reset)
`bist`	Run built-in self-test
`mode [1-4]`	Get or set operation mode
`spread [1|2|off]`	Get or set clock spread
`reset [0-4|all]`	Send PERST# reset to connector

9. Troubleshooting

9.1 Connection Issues

Problem: "Permission denied" error on Linux
Solution:

1. Add your user to the dialout group: sudo usermod -a -G dialout $USER
2. Log out and log back in for the change to take effect
3. Alternatively, run with sudo (not recommended for production)

Problem: Device not found
Solution:

- Verify USB-C cable is connected to CN7 (MCU CLI port)
- Check that the Atlas3 card is powered (12V connected to CN9)
- Try a different USB-C cable
- Run atlas3-cli find to discover available devices

9.2 Link Issues

Problem: Port shows DEGRADED status
Solution:

- Check cable quality and seating
- Verify connected device supports the expected speed
- Check for signal integrity issues (cable length, routing)
- Review error counters for specific failure modes

Problem: Cannot achieve Gen6 speeds
Solution:

- Ensure FLIT mode is enabled: card.set_flit_mode("all", False)
- Verify the endpoint device supports Gen6
- Check that the host system's BIOS supports Gen6
- Use high-quality cables rated for Gen6 operation

9.3 Common Error Counters

Counter	Possible Causes
BadTLP	Signal integrity issues, cable problems, endpoint errors
BadDLLP	Data link layer errors, flow control issues
LinkDown	Link training failures, hot-plug events, power issues
FlitError	Gen6 FLIT mode errors, FEC failures

Appendix A: Support and Resources

A.1 Contact Information

Serial Cables, LLC

8811 American Way, Suite 110

Englewood, CO 80107

Website: https://serialcables.com
Email: support@serialcables.com

A.2 Online Resources

- PyPI Package: https://pypi.org/project/serialcables-atlas3/
- GitHub Repository: https://github.com/serialcables/serialcables-atlas3
- Issue Tracker: https://github.com/serialcables/serialcables-atlas3/issues

A.3 Related Products

- HYDRA JBOF Controller - 8-bay E3 passive JBOF
- Hermes PCIe Gen6 EDSFF Redriver Adapters
- CalypsoPy+ PCIe/NVMe Testing Framework

Document Version: 1.0 | API Version: 0.1.5