



TECHNICAL INSIGHTS

An introduction to the Introspect Vector Blaster

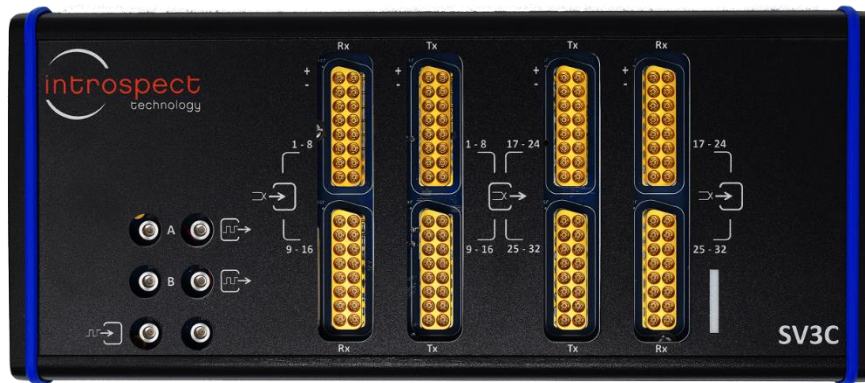


Table of Contents

Introduction.....	3
Benefits and Applications	3
Key Specifications and Features	3
Vector Blaster Input File Format	4
Python Functions for the Introspect ESP Software	6
Example: Sending a Reset Signal to a DUT	8
Connectors and Pinout Specifications	10

Introduction

Each SV3C Personalized SerDes Tester houses a powerful microcontroller with a wide bidirectional GPIO interface. Introspect Technology has developed a flexible, extremely easy-to-use interface for controlling these GPIOs and has fully integrated the control into our Introspect ESP software. This interface is called the “IESP Vector Blaster”.

Benefits and Applications

The IESP Vector Blaster provides a simple mechanism for GPIO control, achieved via an integration of low-level vector files and high-level Python programming, providing the best of two programming worlds. Rapid development and quick turn-around is possible for applications such as:

- implementation of flags and triggers for direct communication with a DUT
- implementation of 32 or 64 bit wide bidirectional buses for communicating with a DUT
- implementation of low-speed interfaces such as I2C

Key Specifications and Features

- Total GPIO pins available: 108
- Maximum Data Rate (per pin): 25 MHz
- Signal standard: 2.5 V LVCMOS
- Seamless integration of IESP Vector Blaster feature set into existing Introspect ESP Software applications

The format for vector files is provided in the next section, followed by an overview of available Python classes for full script automaton. A pinout from the SV3C family of products is also provided.

Python Functions for the Introspect ESP Software

A subset of useful Python functions for use with the IESP Vector Blaster are described below. The complete set of available commands and complete descriptions are available from the Introspect ESP Software GUI via the pull-down Help Menu (Help -> low-level IESP functions...):

sendVectorsFromFile(vectorFilePath)

This function compiles an ASCII-format vector file as described above into the binary form required by the IESP hardware and sends the vector data to the IESP hardware.

Example of use:

```
info = iesp.sendVectorsFromFile(vectorFilePath)  
if info is not None:  
    (numVectors, numComparisons) = info
```

setNumVectors(numVectors)

Set the number of vectors to be executed. This is only needed if you want to execute a subset of the vectors that were sent, since '*sendVectorsFromFile()*' automatically sets the execution length to all the vectors that were in the file.

Example of use:

```
iesp.setNumVectors(numVectors)
```

startVectorExecution()

This function starts executing the loaded vectors.

Example of use:

```
iesp.startVectorExecution()
```

stepVector()

Execute the next vector if execution paused partway through the loaded vectors.

Example of use:

```
iesp.stepVector()
```

resumeVectorExecution()

Resume executing to the end of the vector list.

Example of use:

```
iesp.resumeVectorExecution()
```

getVectorCompareResults()

This function reads the vector compare results and returns a string (of 0's and 1's) giving the results. If all the vectors in the vector file are executed, the number of bits (number of results of comparisons) should be equal to the 'numComparisons' returned from 'sendVectorsFromFile'.

Example of use:

```
bitStr = iesp.getVectorCompareResults()
```

getVectorPinState(pin)

Get the current state of a specified vector pin. A 'pin' is an integer specifying a vector pin and should be in the range 0-107. The return value will be either "0" for low or "1" for high.

Example of use:

```
pin = 0  
pinState = iesp.getVectorPinState(pin)
```

getVectorPassFail()

Return a Boolean indicating whether the vector execution passed (True) or failed (False). A full vector execution is considered to have passed if each bit of the vector comparison result is "1".

Example of use:

```
status = iesp.getVectorPassFail()
```

Example: Sending a Reset Signal to a DUT

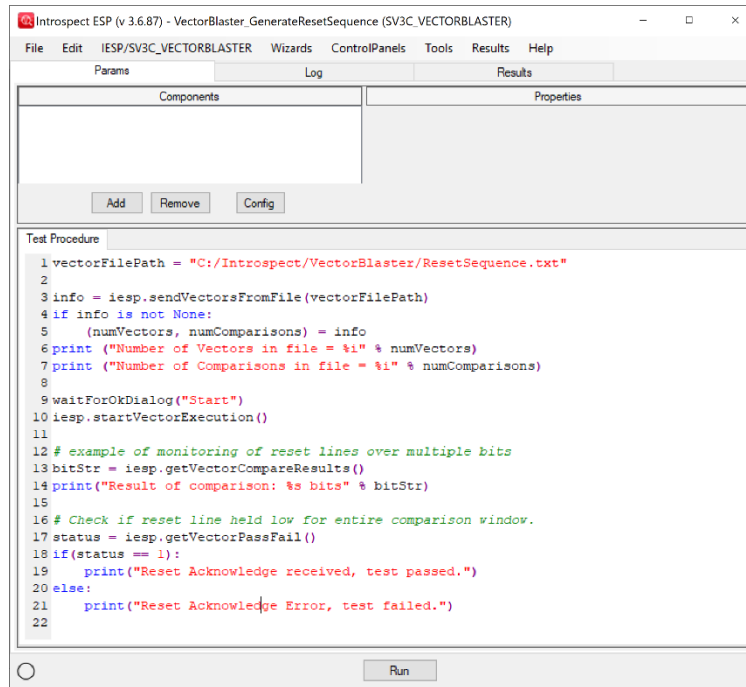
Below is an example which demonstrates a simple application of the IESP Vector Blaster. The vector file consists of a **clock signal**, a **reset** pin (output) and a **reset acknowledge** pin (input), as shown. The following page shows the python-level implementation in the Introspect GUI.

Example Vector File:

```

1 # Example of sending a reset signal to a DUT and monitoring the reset acknowledgement
2 #
3 # Vector Clock Rate = 25 Mhz
4 # RESET starts high for 10 bit cycles, sends low for 10 bit cycles, then returns high and waits 10 bit cycles
5 # Then RESET ACK monitored a "0" response for 10 bit cycles
6 #
7 # bit 0 = RESET_N
8 # bit 4 = RESET_ACK_N
9 # bit 8 = 25 MHz reference
10
11 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R10
12 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R10
13 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R10
14 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
15 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
16 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
17 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
18 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
19 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
20 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
21 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
22 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
23 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX R1
24

```

For a test case where the “reset acknowledge” remained low only for 4 bits within the desired window, the following output log messages would be generated:

Output log file:

Components used by Test Procedure: []
Number of Vectors in file = 13
Number of Comparisons in file = 10
Result of comparison: 0000111100 bits
Reset Acknowledge Error, test failed.

For the case where the Reset Acknowledge remained low for all comparison bits within the desired window, the following output log messages would be generated:

Output log file:

```
Components used by Test Procedure: []
Number of Vectors in file = 13
Number of Comparisons in file = 10
Result of comparison: 111111111 bits
Reset Acknowledge received, test passed.
```

Connectors and Pinout Specifications

The 108 GPIO pins on SV3C devices may be accessed via the high-density Samtec Searay connector on the left side of the module, as shown in the figure on the left below. The pin orientation of the Searay connector the SV3C itself is shown below on the right.

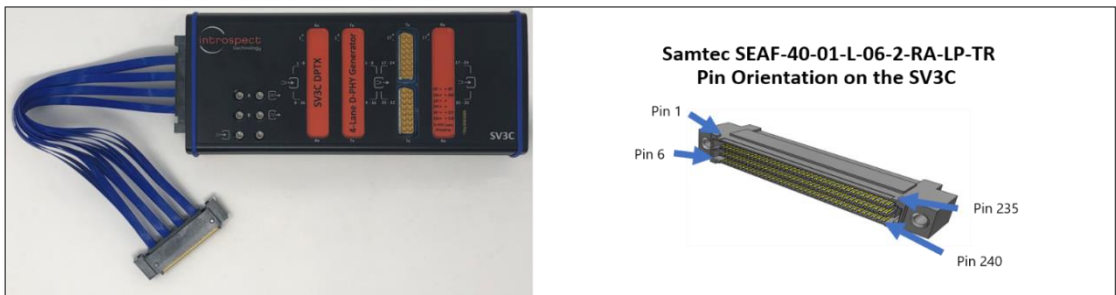
The Samtec part number numbers for connecting to the Vector Blaster interface are as follows:

SV3C Connector: Samtec SEAF-40-01-L-06-2-RA-LP-TR

Connection Cable: Samtec SEAC-040-06-12.0-TU-TU

Recommended customer-side connectors:

Samtec SEAF-40-01-L-06-2-RA-LP-TR (right angle) or SEAF-40-05.0-L-06-2-A-K-TR (straight)



For the complete Searay pinout, please contact Introspect Technology.



Revision Number	History	Date
1.0	Document Release	January 13, 2020

The information in this document is subject to change without notice and should not be construed as a commitment by Introspect Technology. While reasonable precautions have been taken, Introspect Technology assumes no responsibility for any errors that may appear in this document.