

DSP/BIOS™ LINK

TEST SUITE

LNK 015 DES

Version 1.02

Version 1.02 Page 1 of 43



This page has been intentionally left blank.

Version 1.02 Page 2 of 43



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third—party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address: Texas Instruments Post Office Box 655303 Dallas, Texas 75265

Copyright ©. 2003, Texas Instruments Incorporated

Version 1.02 Page 3 of 43



This page has been intentionally left blank.

Version 1.02 Page 4 of 43



TABLE OF CONTENTS

1	Introduction	8
	1.1 Purpose and Scope	8
	1.2 References	8
	1.3 Overview	8
2	Requirements	9
3	Assumptions	10
4	High Level Design	11
5	File: TST_Analysis.h	14
	5.1 API Definition	14
6	File: TST_FileOperation.h	19
	6.1 API Definition	19
7	File: TST_Framework.h	
	7.1 Typedefs and Data Structures	22
	7.2 API Definition	24
8	File: TST_GetArgs.h	26
	8.1 API Definition	26
9	File: TST_Helper.h	
	9.1 Typedefs and Data Structures	
	9.2 API Definition	
10	File: TST_PrintFuncs.h	
	10.1 API Definition	
11	File: TST_TestInputs.h	
	11.1 API Definition	36
12	File: testsuiteANA.h	
	12.1 API Definition	37
13	File: testsuiteAPI.h	
	13.1 API Definition	38
14	File: testsuiteBVR.h	
	14.1 API Definition	39
15	File: testsuiteSTS.h	40
	15.1 API Definition	40
16	File: testsuiteAPP.h	
	16.1 API Definition	41
17	File: TST_RunTest.h	42
	17.1 API Definition	42



Version 1.02 Page 6 of 43



TABLE OF FIGURES

Figure 1. Test Framework Components......11

Version 1.02 Page 7 of 43



1 Introduction

1.1 Purpose and Scope

This document describes the overall design and architecture of the Test Framework of DSP/BIOS™ LINK.

It lists the interfaces exposed by Test Framework and describes the overall design for implementation of these interfaces.

The return values as returned by a function in the document may not reflect all possible values returned by that function.

1.2 References

1.	LNK 002 ARC	DSP/BIOS™ LINK
		High Level Architecture
		Version 1.02, dated JUL 15, 2003
2.	LNK 007 TST	DSP/BIOS™ LINK
		Test Plan
		Version 0.10, dated APR 29, 2002

1.3 Overview

This framework provides a common shell to execute individual test cases. It is tolerant to different faults that occur during the execution of the test cases. It also ensures that testing can continue normally even after a test fails due to a major defect. Unrecoverable system failures are exceptions to this. It also provides the generic functions required for implementing various test suites.

Version 1.02 Page 8 of 43



2 Requirements

2.1.1 Generic

- 1. Each test case is easily plug-able into the existing framework.
- 2. The directory structure for the source code follows the test suite hierarchy.
- 3. Each test case returns a success or error code.

2.1.2 Test Framework

This framework provides a common shell to execute individual test cases. It partitions the argument validation etc. from the actual execution of the test.

This framework tolerates different faults that occur during the execution of the test cases. This ensures that the testing can continue normally even after a test fails due to a major defect. Irrecoverable system failures are exceptions to this.

2.1.3 API Tests

This test suite verifies the compliance to the documented API. It tests the API against valid and invalid arguments, basic data transfer (with data integrity), and operations that cause state changes.

2.1.4 Behavioral Tests

This test suite verifies the behavior through a series of typical usage scenarios. It exercises these scenarios in both single and multi-threaded environments.

2.1.5 Analysis Tests

This test suite measures the system performance through a raw single channel data transfer (with data integrity check). The time for checking data integrity is excluded from the calculations.

The test is done for data transfer in synchronous and asynchronous modes with varying size of data buffers.

2.1.6 Stress Tests

This test suite measures the limits of the system under stress conditions. These conditions can be:

- § Low resource availability
- § Clients in multiple processes
- § Data transfer from multiple concurrent threads
- § Various timing delays to simulate application processing

2.1.7 Application Test

This test contains an actual application that runs on the target platform and exercises the system beyond the earlier test suites. For example, an application that plays audio files using $\mathsf{DSP/BIOS^{TM}}$ LINK.

Version 1.02 Page 9 of 43



2.1.8 Install Tests

This test suite verifies the correct installation of DSP/BIOS $^{\text{TM}}$ LINK and its components.

3 Assumptions

- 1. If an OS supports multi-processing environment, the test suite executes as a process. Each test case executes in its own thread context within this process.
- 2. If an OS does not support multi-processing environment, the test suite executes as a thread/task. Each test case executes in its own thread/task context.

Version 1.02 Page 10 of 43



4 High Level Design

Figure 1 shows the relationship of the Test Framework components with each other

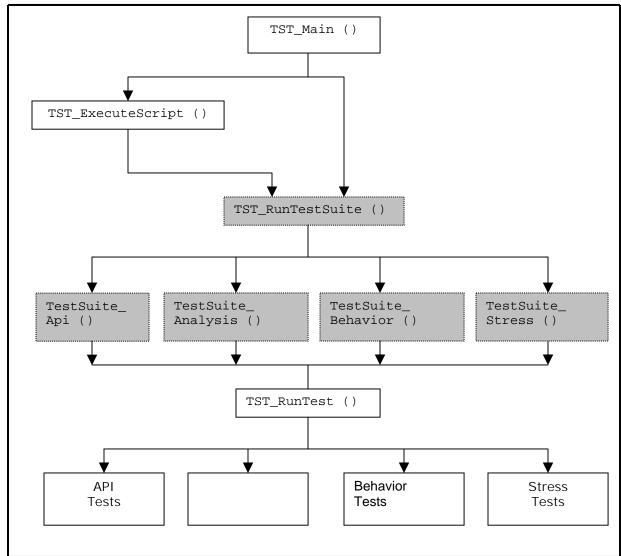




Figure 1. Test Framework Components

The TST_Main () function is the entry point to the Test Framework. It supports two modes:

- § SINGLE
- § SCRIPT

Depending on the arguments specified, the TST_ExecuteScript () or the TST RunTestSuite () function is called.

Version 1.02 Page 11 of 43



The TST_ExecuteScript () function in turn calls the TST_RunTestSuite () function repeatedly until it processes the whole script.

The TST_RunTestSuite () is a function generated during MAKE using the directory structure of the test suites. This controls the entry point for all the test suites. The TST_RunTestSuite () maintains the list of test suites currently available. Depending on the arguments passed, it invokes one of the test suites (that are also generated depending on the test cases in that test suite). These test suites then call TST RunTest () with a function pointer of the test case to invoke.

TST_RunTest () parses the data file passed to it and calls the test case repeatedly until the data file is exhausted. It also prints the status of the test after the test has completed execution. Depending on the status returned by the test, it also increments passcount or failcount, which is used to print the summary in the end.

The framework provides generic functions for implementing various test suites such as:

```
- TST_OpenFile ()
```

Opens a given file and returns a file pointer

```
- TST_CloseFile ()
```

Closes an already open file

- TST_GetArgs ()

Reads one line of arguments when the script file is given

- TST_TestInputs ()

Reads one line of arguments when the data file is given

TST_GetFileSize ()

Gives the size of the file

- TST_StrToStatus ()

Converts the status from string form to the standard status codes

- TST_StringToInt ()

Converts strings to integers

- TST_DoAnalysis ()

Does the bandwidth analysis

- TST_ToLower ()

Converts strings to lowercase

- TST_ToUpper ()

Converts strings to uppercase

The framework provides various printing functions such as:

- TST PrnError ()

Prints error strings if an error occurs

- TST_PrnDebug ()

Used for debugging purposes

Version 1.02 Page 12 of 43



- TST_PrnInfo ()

Prints information

- TST_PrnStatus ()

Prints the status of the test executed

- TST_PrnTestCase ()

Prints the names of testsuite and test case

- TST_PrnSummary ()

Prints the summary of all the tests

Other functions apart from TST_RunTestSuite () that are generated are:

- TestSuite_Analysis ()
- TestSuite_Api ()
- TestSuite_Behavior ()
- TestSuite_Stress ()

If the user wants to add another test suite to the existing test suites, the user creates a directory in the format "\$(DSPLINK)\gpp\src\test\Nucleus\" directory. All the test cases pertaining to the test suite reside in that test suite directory. Then the user must run MAKE to make the entry point to the test suite and to add that entry point of the test suite to the function TST_RunTestSuite (). This ensures that even when the new tests are added or some tests are removed, no change needs to be done in Test Framework.

Version 1.02 Page 13 of 43



5 File: TST_Analysis.h

5.1 API Definition

5.1.1 TestSuite_ANA

This function decides which test case to run.

Syntax

Arguments

IN Int32 argc

Count of the no. of arguments passed

IN Char8 ** argv

List of arguments

IN Result * frameworkResult

Pointer to result structure passed by framework

Return Values

DSP_SOK Operation completed successfully

DSP_EFAIL Operation failed.

DSP_EMEMORY Operation failed due to insufficient memory.

DSP_EPOINTER Invalid pointer passed.

Pre Condition(s

argc must not be less than one argv must not be NULL.

Post Condition(s)

None.

See Also

None.

Version 1.02 Page 14 of 43



5.1.2 TST_ZeroTime

This function resets the system timer.

Syntax

```
Void TST_ZeroTime () ;
```

Arguments

None.

Return Values

None.

Pre Condition(s

None.

Post Condition(s)

None

See Also

None.

Modifies

None.

Version 1.02 Page 15 of 43



5.1.3 TST_GetTime

This function gives the time elapsed since the last reset.

Syntax

```
Void TST_GetTime () ;
```

Arguments

None.

Return Values

Time elapsed since the last reset.

Pre Condition(s

None.

Post Condition(s)

None

See Also

None.

Modifies

None.

Version 1.02 Page 16 of 43



5.1.4 TST_DoAnalysis

This function calculates the throughput of the test under consideration

Syntax

DSP_STATUS TST_DoAnalysis (Uint32 startTime, Uint32 stopTime, Uint32 iterations, Uint32 bufSize);

Arguments

IN	Uint32	startTime
TIN	UIIIL3Z	Startime

Starting tickcount value

IN Uint32 stopTime

Ending tickcount value

IN Uint32 iterations

It is the number of iterations for which the transfer was made

IN Uint32 bufSize

Size of buffer based on which throughput is to be calculated

Return Values

DSP_SOK Operation completed successfully.

Pre Condition(s

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 17 of 43



5.1.5 TST_GetFileSize

This function gives the size of the file in bytes.

Syntax

DSP_STATUS TST_GetFileSize (Char8 * fileName, Uint32 * size) ;

Arguments

IN Char8 * fileName

The name of the file whose size is to be determined

OUT Uint32 * size

The size of the file

Return Values

DSP_SOK Operation completed successfully

DSP_EINVALIDARG Invalid arguments

DSP_EFILE File error

DSP_EPOINTER Invalid file object

Pre Condition(s

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 18 of 43



6 File: TST_FileOperation.h

6.1 API Definition

6.1.1 TST_ReadLine

Parses the data file to read one line.

Syntax

Arguments

IN Void * filePtr

File pointer of the file to be parsed.

OUT Uint32 argo

Placeholder to return the number of arguments obtained from a line in

the data file.

OUT Char8 argv

Placeholder to return the arguments obtained from the line in the data

file.

Return Values

DSP_SOK Operation completed successfully

DSP_EINVALIDARG Invalid argument given

DSP_EFILE File error

Pre Condition(s

filePtr must not be NULL.

argc must not be NULL.

argv must not be NULL.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 19 of 43



6.1.2 TST_OpenFile

This function opens a given file and returns a file pointer.

Syntax

TST_OpenFile (Char8 * fileName, Void ** filePtr) ;

Arguments

IN Char8* filename

The name of the file to be opened.

IN Void ** filePtr

The address of the file pointer of the file which is being opened

Return Values

DSP_SOK Operation completed successfully

DSP_EINVALIDARG Invalid arguments.

DSP_EFILE File not found

DSP_EMEMORY Memory allocation error

Pre Condition(s

fileName must not be NULL.

filePtr must not be NULL.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 20 of 43



6.1.3 TST_CloseFile

This function accepts a file pointer and closes that file.

Syntax

TST_CloseFile (Void ** filePtr) ;

Arguments

IN Void ** filePtr

The address of the file pointer of the file which is being opened

Return Values

DSP_SOK Operation completed successfully

DSP_EFILE File error

Pre Condition(s

filePtr must be valid.

Post Condition(s)

None.

See Also

None.

Modifies

None

Version 1.02 Page 21 of 43



7 File: TST_Framework.h

7.1 Typedefs and Data Structures

7.1.1 PtrToTestCase

Its Function signature to Test cases.

Syntax

```
typedef DSP_STATUS (*PtrToTestCase) (Uint32 argc, Char8 ** argv);
```

Field(s)

argc Number of arguments to testcase

argv The arguments to be passed to the testcase

Post Condition(s)

None.

See Also

None.

7.1.2 Result

The structure stores the data pertaining to the test.

Syntax

```
typedef struct Result_tag {
    Uint32 testCount ;
    Uint32 testRun ;
    Uint32 passCount ;
    Uint32 failCount ;
}
```

Field(s)

testCount Count of tests to be run

testRun Count of actual tests run

passCount Count of tests which passed

failCount Count of tests which failed

See Also

None.

7.1.3 TestInfo

The structure stores the names of the test suite and the test case.

Syntax

```
typedef struct TestInfo_tag {
```

Version 1.02 Page 22 of 43



```
Char8 * testSuite ;
  Char8 * testCase ;
} TestInfo ;
```

Field(s)

testSuite Name of the test suite

testCase Name of the test case

See Also

None.

Version 1.02 Page 23 of 43



7.2 API Definition

7.2.1 TST_Main

It is the entry point to the Main Framework function.

Syntax

```
DSP_STATUS TST_Main (Uint32 argc, Char8 ** argv);
```

Arguments

IN Uint32 argc

Number of arguments

IN Char8 ** argv

List of arguments

Return Values

DSP_SOK Operation completed successfully

DSP_INVALIDARG Number of wrong arguments passed

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

7.2.2 TST_ExecuteScript

This function is invoked with the script argument and it also invokes the desired testcase.

Syntax

Arguments

IN Uint32 argc

Number of arguments

IN Char8 ** **argv

Version 1.02 Page 24 of 43



List of arguments

OUT Result * frameworkResult

Pointer to result structure

Return Values

DSP_SOK Operation completed successfully

DSP_INVALIDARG Number of wrong arguments passed

DSP_EMEMORY Operation failed due to insufficient memory

DSP_EPOINTER Invalid pointer passed

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 25 of 43



8 File: TST_GetArgs.h

8.1 API Definition

8.1.1 TST_GetArgs

It parses the Script file and fetches the arguments to execute a test case.

Syntax

Arguments

IN Void * filePtr

File pointer of the file to be parsed

OUT Uint32 * argc

Number of arguments in the line parsed

OUT Char8 argv

The pointer to the list of arguments

Return Values

DSP_SOK Operation successfully completed.

DSP_INVALIDARG Invalid argument given

DSP_EFILE File error

Pre Condition(s)

filePtr must not be NULL.

argc must not be NULL.

argv must not be NULL.

Post Condition(s)

None.

See Also

None.

Version 1.02 Page 26 of 43



9 File: TST_Helper.h

9.1 Typedefs and Data Structures

9.1.1 ISSPACE

Description

Tests for the white space characters.

9.1.2 ISDIGIT

Description

Tests for the digit characters

9.2 API Definition

9.2.1 TST_StrToStatus

This function returns an appropriate status when provided with the status in string format.

Syntax

Arguments

IN CHAIO " STATUSSELLI	IN Char8	*	statusString
------------------------	----------	---	--------------

The status in string format

OUT DSP_STATUS * statusFromFile

Placeholder for DSP_STATUS corresponding to the string passed

Return Values

DSP_SOK Operation successfully completed.

DSP_EINVALIDARG Operation failed due to Invalid arguments.

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

9.2.2 TST_ToLower

This function converts an uppercase string to lowercase.

Version 1.02 Page 27 of 43



Syntax

DSP_STATUS TST_ToLower (Char8 * source, Char8 * destination);

Arguments

IN Char8 * source

This argument is the string in the lowercase which is to be converted . . .

into uppercase.

OUT Char8 * destination

This is the buffer which is provided by the user.

Return Values

DSP_SOK Operation completed successfully

DSP_EINVALIDARG Operation failed due to invalid arguments

Pre Condition(s)

source must not be NULL.

destination must not be NULL.

Post Condition(s)

None.

See Also

None.

Modifies

None.

9.2.3 TST_ToUpper

This function converts a lowercase string to uppercase.

Syntax

DSP_STATUS TST_ToUpper (Char8 * source, Char8 * destination);

Arguments

IN Char8 * source

This argument is the string in the lowercase which is to be converted

into uppercase.

OUT Char8 * destination

This is the buffer which is provided by the user.

Return Values

DSP_SOK Operation completed successfully

Version 1.02 Page 28 of 43



DSP_EINVALIDARG

Operation failed due to invalid arguments

Pre Condition(s)

source must not be NULL.

destination must not be NULL.

Post Condition(s)

None

See Also

None.

Modifies

None.

9.2.4 TST_StringToInt

It converts a string to an integer, supports positive as well as negative numbers.

Syntax

```
Int32 TST_StringToInt (Char8 * string) ;
```

Arguments

IN Char8 *

string

The string, which is to be converted into a number.

Return Values

The integer value of the string.

Pre Condition(s)

string must not be NULL.

Post Condition(s)

None

See Also

None.

Modifies

None

9.2.5 TST_Strcmp

Compares 2 ASCII strings. Works the same way as stdio's strcmp.

Syntax

```
DSP_STATUS TST_Strcmp (IN CONST Char8 * string1, IN CONST Char8 * string2,
```

Version 1.02 Page 29 of 43



OUT Int32 * cmpResult);

Arguments

IN Const Char8 * string1

First string for comparison

IN Const Char8 * string2

Second string for comparison

OUT Int32 * cmpResult

Result of comparison (zero if equal, non-zero otherwise).

Return Values

DSP_SOK Soperation successfully completed

DSP_EFAIL General Failure.

DSP_EINVALIDARG Failure due to invalid argument.

Pre Condition(s)

Subcomponent must be initialized.

The buffer to store first string must be valid.

The buffer to store second string must be valid.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 30 of 43



10 File: TST_PrintFuncs.h

10.1 API Definition

10.1.1 TST_PrnError

This function prints error strings.

Syntax

Void TST_PrnError (Char8 * fmt) ;

Arguments

IN Char8 * fmt

The string, which is to be printed.

Return Values

None.

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

10.1.2 TST_PrnDebug

This function prints debug strings.

Syntax

Void TST_PrnDebug (Char8 * fmt) ;

Arguments

IN Char8 * fmt

The string, which is to be printed.

Return Values

None.

Pre Condition(s)

None.

Version 1.02 Page 31 of 43



Post Condition(s)

None

See Also

None.

Modifies

None.

10.1.3 TST_PrnArgs

This function prints arguments used by testcase.

Syntax

```
Void TST_PrnArgs (Uint32 argc, Char8 ** argv) ;
```

Arguments

IN Uint32 argc

Number of arguments

IN Char8 ** argv

Pointer to array of arguments to be printed.

Return Values

None.

Pre Condition(s)

None.

Post Condition(s)

None

See Also

None.

10.1.4 TST_PrnInfo

This function prints the information strings.

Syntax

```
Void TST_PrnInfo (Char8 * fmt, ...) ;
```

Arguments

IN Char8 *fmt

The string which is to be printed

Return Values

None.

Version 1.02 Page 32 of 43



Pre Condition(s)

None.

Post Condition(s)

None

See Also

None.

Modifies

None.

10.1.5 TST PrnStatus

This function prints the status of the tests.

Syntax

Void TST_PrnStatus (DSP_STATUS statusToPrint) ;

Arguments

IN DSP_STATUS statusToPrint

The status of the test executed

Return Values

None.

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

10.1.6 TST_PrnTestCase

This function prints the test suite and the test case name.

Syntax

Void TST_PrnTestCase (TestInfo * testInfo) ;

Arguments

IN TestInfo * testInfo

Version 1.02 Page 33 of 43



The pointer to the structure containing names of the test suite and the test case to execute.

Return Values

None.

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

10.1.7 TST_PrnSummary

This function prints the summary of the tests run.

Syntax

```
Void TST_PrnSummary (Result * finalResult) ;
```

Arguments

IN Result * finalResult

The pointer to the structure containing the results

Return Values

None.

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

10.1.8 TST_Print

This function provides the standard printf functionality abstraction.

Version 1.02 Page 34 of 43



Syntax

Void TST_Print (Char8 * format, ...) ;

Arguments

IN Char8 format

The string to print.

Return Values

None.

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 35 of 43



11 File: TST_TestInputs.h

11.1 API Definition

11.1.1 TST_TestInputs

Parses the data file to read one line.

Syntax

Arguments

IN Void * filePtr

File pointer of the file to be parsed

OUT Uint32 argo

Number of arguments in the line parsed

OUT Char8 [][] argv

The pointer to the list of arguments

Return Values

DSP_SOK Operation completed successfully

DSP_EINVALIDARG Invalid argument

DSP_EFILE File error

Pre Condition(s)

filePtr must not be NULL.

argc must not be NULL.

argv must not be NULL.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 36 of 43



12 File: testsuiteANA.h

12.1 API Definition

12.1.1 TestSuite_ANA

This function decides which testcase to run.

Syntax

Arguments

IN Int32 argc

Number of arguments in the line parsed

IN Char8 ** argv

The pointer to the list of arguments

OUT Result * frameworkResult

Pointer to result structure passed by framework

Return Values

DSP_SOK Operation completed successfully

DSP_EFAIL Operation failed

DSP_EMEMORY Operation failed due to insufficient memory

DSP_EPOINTER Invalid pointer passed

Pre Condition(s)

argc must not be not be less than one.

argy must not be null.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 37 of 43



13 File: testsuiteAPI.h

13.1 API Definition

13.1.1 TestSuite_API

This function decides which api testcase to run.

Syntax

Arguments

IN Int32 argc

Number of arguments in the line parsed

IN Char8 ** argv

The pointer to the list of arguments

OUT Result * frameworkResult

Pointer to result structure passed by framework

Return Values

DSP_SOK Operation completed successfully

DSP_EFAIL Operation failed

DSP_EMEMORY Operation failed due to insufficient memory

DSP_EPOINTER Invalid pointer passed

Pre Condition(s)

argc must not be less than one.

argv must not be null.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 38 of 43



14 File: testsuiteBVR.h

14.1 API Definition

14.1.1 TestSuite_Behavior

This function decides which testcase to run.

Syntax

Arguments

IN Int32 argc

Number of arguments

IN Char8 ** argv

The pointer to the list of arguments

OUT Result * frameworkResult

Pointer to result structure passed by framework

Return Values

DSP_SOK Operation completed successfully.

DSP_EFAIL Operation failed.

DSP_EMEMORY Operation failed due to insufficient memory.

DSP_EPOINTER Invalid pointer passed

Pre Condition(s)

argc must not be less than one.

argy must not be null.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 39 of 43



15 File: testsuiteSTS.h

15.1 API Definition

15.1.1 TestSuite_Stress

This function decides which testcase to run.

Syntax

Arguments

IN Int32 argc

Number of arguments

IN Char8 ** argv

The pointer to the list of arguments

OUT Result * frameworkResult

Pointer to result structure passed by framework

Return Values

DSP_SOK Operation completed successfully.

DSP_EFAIL Operation failed.

DSP_EMEMORY Operation failed due to insufficient memory.

DSP_EPOINTER Invalid pointer passed.

Pre Condition(s)

argc must not be less than one.

argy must not be null.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 40 of 43



16 File: testsuiteAPP.h

16.1 API Definition

16.1.1 TestSuite_Application

This function decides which testcase to run.

Syntax

```
DSP_STATUS TestSuite_Application (Int32 argc,
Char8 ** argv,
Result * frameworkResult);
```

Arguments

IN Int32 argc

Number of arguments

IN Char8 ** argv

The pointer to the list of arguments

OUT Result * frameworkResult

Pointer to result structure passed by framework

Return Values

DSP_SOK Operation completed successfully.

DSP_EFAIL Operation failed.

DSP_EMEMORY Operation failed due to insufficient memory.

DSP_EPOINTER Invalid pointer passed.

Pre Condition(s)

argc must not be less than one.

argy must not be null.

Post Condition(s)

None.

See Also

None.

Modifies

None.

Version 1.02 Page 41 of 43



17 File: TST_RunTest.h

17.1 API Definition

17.1.1 TST_RunTest

This function runs the testcase.

Syntax

IN Result * frameworkResult);

Arguments

IN Uint32 argc

Number of arguments

IN Char8 ** argv

The pointer to the list of arguments

OUT Result * frameworkResult

Pointer to result structure passed by framework

IN PtrToTestCase funcPtr

Pointer to the test case to be invoked

IN TestInfo * testName

Pointer to structure containing names of testsuites and testcase.

Return Values

DSP_SOK Operation completed successfully.

DSP_INVALIDARG Wrong no of arguments passed.

Pre Condition(s)

None.

Post Condition(s)

None.

See Also

None.

Version 1.02 Page 42 of 43



Version 1.02 Page 43 of 43