TESTEXEC™

Driving the Maturity and Reliability of softFEP and satTRAC Products

August 2015

INNOVATIVE • ADVANTAGE

AMERGINT

2315 Briargate Parkway, Suite 100
Colorado Springs, CO 80920
www.amergint.com

Why We Created TestExec

The importance of test is amplified with software applications. This is particularly true for software applications that replace traditional hardware functions.

Real time software applications have the added complexity of accounting for the small variations in processing delay. Compared to hardware, software requires a more sophisticated approach to instrumenting it for test. It's just not easy to stick an oscilloscope in there between two software modules! On the other hand, software testing can be highly automated and more in-depth.

Testing Our Software Applications Required New Thinking

The advantages of today's software-based applications for equipment such as modems, data recorders, and front ends easily offset the increased need for in-depth software testing. The new products offer flexibility. They can be easily tailored. They have also driven costs down an order of magnitude.

AMERGINT developed and uses our TestExec Framework to exercise and verify our products. TestExec allows this to happen at multiple levels and with a high degree of automation and repeatability. This ensures the softFEP and satTRAC products you install are thoroughly tested and have the maturity and reliability needed for satellite ground systems.

Let's see how.

TestExec Framework

TestExec is a software test framework and set of utilities designed to work specifically with our products and their product architecture. TestExec manages the full test environment from test development and organization through to test execution and reporting.

The softFEP Apps and satTRAC Apps are implemented with re-usable software devices (we refer to them as SwDs). The SwDs are linked and ordered in to processing chains, which are organized into software applications.

TestExec allows us to verify the software at all three levels of implementation.

There is a high degree of automation in the TestExec Framework. All tests are scripted, allowing them to execute with little or no manual intervention. A sequence of test scripts can be executed, one after another, and in any order.

- **Multi-Level Verification**
- **Modular Test Sequences**
- **Execution & Reporting Automation**
- **Requirements Tracking**
- **Delivered with Each Product**

TestExec eliminates many of tedious aspects of test for our engineers. And guess what? As a result, they write more tests! The as-run test procedures are automatically generated. Test results are captured, analyzed, and included in each individual test report.

There's one last item of note in this introduction. The tests that are used to verify the product live long after factory acceptance testing. We deliver them along with the products. Customers can re-run the tests after installation. Customers can also extend the TestExec tests to exercise bigger parts of the system.

Meanwhile, here at AMERGINT we continue to refine the tests as we solve issues and make updates. We compile a full suite of automated regression tests for each App that's delivered.

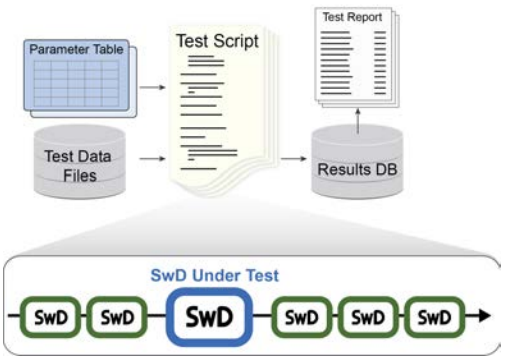
Testing SwDs, Chains, and Applications

One of the most powerful features of TestExec is that it can fully exercise and test at the SwD level.

SwD is short for software device and we have more than 700 of these SwDs that can be linked and used in our Apps. Why so many SwDs? One driver is that SwDs are designed to be "atomic," meaning they perform a singular function. As atomic SwDs, they connect to adjacent SwDs to accomplish the larger task. There are tightly controlled interfaces to and from each SwD.

Just as Apps can connect SwDs together into a processing chain, TestExec can connect SwDs together into a processing chain. Only here, the adjacent SwDs are driving the SwD under test, exercising the data interfaces, sending and receiving data. TestExec also connects to the SwDs GEMs (Ground Equipment Monitoring System) interface to control and status the SwD.

TestExec Verifies Individual SwDs



So TestExec has complete control over the configuration and data flow through the SwD. This allows us to create low-level tests for each SwD. These SwD Tests fully exercise the functions of the SwD and verify those nasty corner cases. These all-important SwD Tests are maintained along with the SwD itself, and they are run every time the SwD is built.

Our product applications are built from these SwD components, with most SwDs being used over and over again across deliveries. Many of the infrastructure SwDs have been deployed thousands of times.

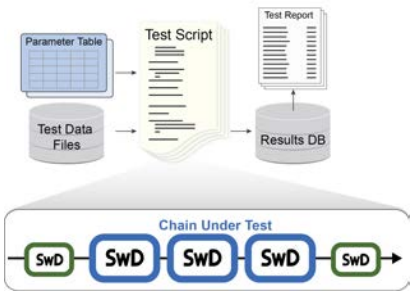
Since SwDs are extensively re-used, having them be well-tested and proven goes a long way towards having a reliable software application.

This same testing concept extends to SwD Processing Chains. Here TestExec SwDs are generating the input to the processing chain and receiving the corresponding output. Running TestExec tests against a processing chain (or a sub-chain) enables us to verify how data flows and is transformed as it moves from SwD to SwD.

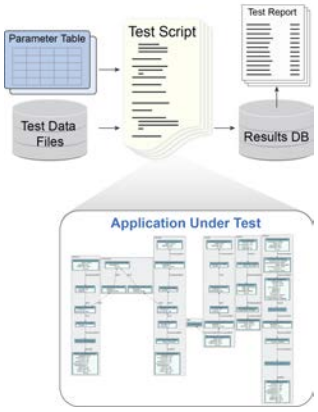
Like SwD testing, testing an individual processing chain clears away the clutter and allows for a more focused, precise test.

The top tier of testing exercises the full product Application. Tests at this level are often system performance tests (e.g. throughput, latency), data flow tests, and long-duration tests that prove out the reliability of the software. These tests start and configure/re-configure the App, while running data through all of the processing chains, exercising both the data and control/status interfaces.

Testing a Processing Chain



Testing a Full Application



Test Scripts and Test Sequences

Test Scripts – A driving principle of our product test approach is to develop atomic test scripts. Each test script performs a well-defined, self-contained test. So each test script completes the necessary set-up, performs the testing, and then "tears down" the test configuration. This simple construct enables the test scripts to be ordered and combined in any sequence.

Writing Atomic Test Scripts is Fundamental within TestExec

TestExec's Test Scripts are written in Python, a flexible programming language with widespread support. The test engineer does have to have good software development skills and an understanding of what's being tested. That said, **the TestExec Framework greatly simplifies the development of the Test Scripts.**

The underlying framework provides a set of function calls that write the test steps, perform verifications, and track the results. There are libraries that connect test scripts to commercial test and measurement equipment such as a spectrum analyzer.

Test Execution – The atomic Tests Scripts are combined and ordered into a TestExec Test Sequence. Doing so builds a series of tests that are akin to a large test procedure. TestExec allows the definition of multiple Test Sequences. It also allows any subset of the Test Scripts in a Test Sequence to be run. You can run one of the tests, a subset, or all of the tests.

Unlike monolithic test procedures that must be run and re-run from the start, a TestExec Test Sequence can jump right to the test section of interest. This allows focused testing when dealing with a new function or chasing one of those elusive problems.

The TestExec Framework also allows for Test Scripts to be parameterized and repeated with different parameter sets. A simple CSV file defines the test parameters in the columns and then each row defines a separate test run. Running Bit Error Rate tests at multiple data rates over the desired range of signal-to-noise ratios is a good example of a parameterized test.

Any Test Sequence can be loaded and executed from a command line or with the TestExec graphical user interface. While the user interface gives the test engineer the option to monitor the Test Script as it executes, most tests are written to be fully automated. Tests that require a manual operation prompt the user at the corresponding points in the script execution.

There are hundreds of SwD and product-level tests in place for our products. The list below gives a sense of the breadth and depth of tests in place for a satTRAC Modem.

Example satTRAC Modem Tests

- Transmitter Output (Flatness and Maximum Slope)
- Output Frequency (Accuracy and Resolution)
- Output Power (Accuracy, Range, Resolution, and Monotonicity)
- Output SNR (Signal-to-Noise Ratio over the Operating Bandwidth)
- Output Spurious (Carrier-Related and Non-Carrier Related)
- Two Tone Input (Measures Intermodulation Products)
- AGC Interferer
- Receiver Signal Strength Indication (Accuracy and Monotonicity)
- Carrier Detect Sensitivity (Detect Low-Level Carrier)
- Receiver Compression Test (1db Compression Point)
- Tone Frequency (CW Tone Accuracy)
- Modulation Index
- Modulator Mode Switch
- Output Stability (Frequency and Power Level Stability)
- Spectral Mask
- Acquire and Track (Input Acquisition Range, Doppler Tolerance, Carrier Offset, Input Level, and Acquisition Time)
- Output Phase Noise
- Configuration Checks
- General BER Test Class
- Commanding BERT
- Telemetry BERT
- Gain Control
- Uplink Calibration
- Downlink Calibration
- Attenuator (Accuracy)
- Demodulator Dynamic Range
- Demodulator Mode Switch
- VSWR
- Subcarrier Acquisition (Time and Range)
- Subcarrier Modulation (Accuracy)
- Doppler RMS
- Spectral Comparison

Test Reports are Auto-Generated

One of the most appreciated features of TestExec here at AMERGINT is the auto-generation of Test Reports. **The test results are fully documented just by running the tests.** This alone saves hours of test engineer time and energy. The test report also aids in fault isolation, as it identifies exactly where in the test script the failure occurred.

The TestExec Framework has utilities that define what goes into the report and the verification results that are captured. The As Run Test Report is then "written" directly from the execution, guaranteeing that the Test Report matches the test script's execution through branches, loops, and other conditional logic that controls the order of execution.

The Test Execution "Writes" the Test Report

Test Reports are generated in XML and include graphics/plots using various Python graphing functions.

Each execution produces a separate set of test results so that a parameterized test has an execution report for each row in the CSV file.

These test reports can quickly produce volumes of test data, more than an engineer would ever want to type. Hot links within the file make it easy to jump to the section of the report for a certain test and test run. Test reports are archived on the system for future reference.

Project Verification Testing

The projects that we do for our customers often require a formal verification; known as the Factory Acceptance Test or FAT.

With these projects, our customers provide AMERGINT with detail technical specifications for the product they need. In many cases, existing TestExec tests are used to verify many of the general requirements. New tests are written for the requirements that fall outside of an existing test. These requirements might include detailed performance tests or tests that validate the customer-specific functions while testing the full softFEP or satTRAC Application.

Customers need a detailed record for which requirements were tested in which test and how they fared. The TestExec Framework enables all of the requirements for a specific project to be loaded into TestExec's database and then mapped to the associated Test Scripts.

As tests execute, the requirement's Pass/Fail status is updated. Each test execution then reports the Pass/Fail status for the corresponding requirements. The Summary Test Report documents the overall Pass/Fail status for each requirement with hotlinks to the associated tests.

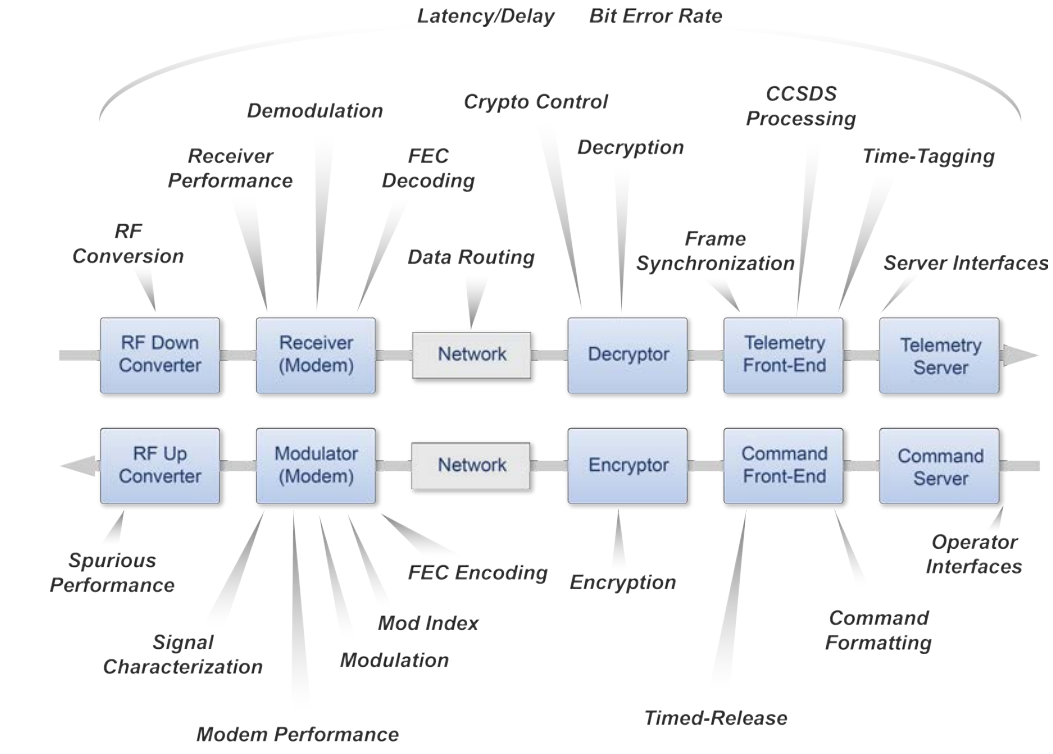
Tests That Come Along at Product Delivery

We've all seen where there's this Herculean effort to get through factory acceptance testing, the product gets shipped to the customer, and the tests that got it there are left behind and forgotten.

Not the case here! We deliver the TestExec Framework and product test scripts to our customers. This allows AMERGINT's customers to re-run specific tests as part of their integration and checkout. Our customers can even extend the TestExec test scripts so that they become part of their verification test and sell-off and part of your site trouble-shooting procedures.

softFEP and satTRAC Products include the TestExec Framework and Product Test Scripts

We have had customers leverage the TestExec Framework to develop test suites for their use. The figure below shows example system level tests for a satellite ground system.



The internal Developer's Guide and a User Manual are available to customers looking to utilize TestExec and AMERGINT Test Scripts for their integration and testing.

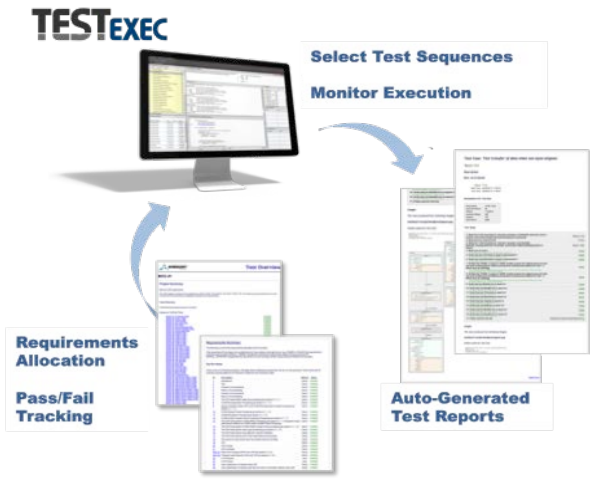
Product Maturity and Reliability

AMERGINT can honestly state that our softFEP and satTRAC Products are the most tested, mature, and reliable products in the marketplace. It's TestExec that allows us to make that claim.

Compared to a manual test, it takes longer to build a TestExec test. This is due to the level of automation and rigor induced by the framework.

But we have realized a long-term payback on our investment:

- TestExec's many features have elevated our product testing to increased levels.
- Test are continually re-used and refined, making them more effective.
- Customers report fewer issues when they put the products into use.
- We have full regression test suites to run against software updates.



Here's a contact at AMERGINT to explore this further.

Randy Culver
randy@amergint.com
719-522-2802