



**BTC vs MQ Tester**

|  |  |  |
| --- | --- | --- |
| **Compare the points** | **BTC Embedded Platform 25.2p1** | **MQ Tester 2020** |
| **Automatic generation of test cases for mass production code and statistics on structure coverage** | 1)Can automatically generate test cases with 100% coverage of MCDC in accordance with ISO26262 standard;  2)The dead code contained in the object under test can be completely parsed;  3) Use cases can be produced according to customized coverage goals. | 1) Some test cases can only be generated by combinatorial arrangement or by engineers writing scripts to judge, etc. It cannot guarantee the generation of complete test cases for mass production code, and it cannot guarantee coverage;  2) No dead code analysis function. |
| **Test case editing environment** | 1. User-friendly test case editing interface; 2. Provide boundary warning for input/output parameters and observed variables; 3. All signals support error setting directly in the test case editing interface; 4. Provide Excel-like editing interface for input/output parameters and observed variables for specific value display and real-time modification; 5. Bidirectional tracking of test cases and test requirements; 6. Provide signal generator and M script decision function and other auxiliary tools to help the editing of complex test cases. Signal generator, custom function combination and evaluation function are abundant, and it is convenient to use graphically; 7. Signal changes can be displayed   graphically;  8) Support for editing scripts or cell formats use cases, | 1) No test case editor built in the tool;   1. No boundary crossing prompt; 2. It needs to be set in the form of a script back in MATLAB; 3. The relevant signals of the actual measured object should be returned to Excel for modification and re-import; 4. The correlation between test cases and requirements is not clear; it is not convenient to track requirements;   6) Signal generator, custom function and evaluation function need to be manually written in Excel. |
| **Selection of subjects** | 1) Support direct selection of a certain sub-level for testing, and do not block other systems or modules, so as to avoid the inability to perform due to partial blocking | 1)need to manually select the coverage filter and ignore other systems or modules that do not need to be tested |
| **SIL simulation** | 1. The SIL test environment can be built quickly and highly automatically, and the SIL test environment is independent of MATLAB (MATLAB is not required for the separate SIL test);   2) Can automate the SIL test environment and  The mapping of MIL/PIL enables reuse of test cases in MIL/SIL/PIL. | 1. can't test the C code individually   ->MQTester 2020 automated build SIL test environment, simplifies the test configuration process.  2) Manual selection of MIL/SIL/PIL mapping is required |
| **Achieve 100% code coverage** | 1. Embedded, no third-party software is required;   2) Support all indicators of functional safety requirements and fully automatic update coverage. | 1) It does not support code coverage target functions, and cannot automatically generate coverage reports and update coverage. |
| **shakedown test** | 1) One-click generation of code or model debugging environment, automatically generated in Matlab/Simulink and Microsoft  The debugging framework in Visual Studio ensures that the original model is safe;  2) The debugging environment can be shared | 1) Debugging directly on the original model is risky;  2) It is difficult to share the debugging environment with other testers |
| **Test case evaluation** | 1. MIL and SIL can be executed by one click for BTC; 2. BTC can automatically execute and compare the actual output value with the expected value;   3) BTC can view the execution of test cases in real time. | 1) MIL/SIL should be manually selected and compared with reference values;  2) The evaluation function needs to be imported manually;  3) The report needs to be exported to view the specific implementation. |
| **requirement** | 1. Support the import of natural language requirements in various formats, and provide traceability among requirements, test cases and objects under test;   2) Support the synchronization of test environment and requirement management tools, such as DOORS, PTC, Excel, etc. | 1) Lack of demand traceability method, need to develop by oneself  -> Update: MQTester 2020 supports two-way traceability between requirements and use cases  2) There are specific format requirements for imported requirements |
| **Test execution and test report** | 1) Automatically generate model coverage and code coverage test report. | 1) No-code coverage report.  -> Update: MQTester 2020 automatically generates code coverage reports |
| **Test outputs management** | All test cases, calibration variables, test reports, etc. can be easily managed in one project, and the relevant relationships are clear. | In the form of files, they are distributed in different paths below, which is not convenient for organization and management. |
| **Back-to-back testing** | 1. Automatically complete the mapping between model and code; 2. Hierarchical comparison can be used to support complex models;   3) Automatically perform equivalence testing and give detailed hints about the differences between model and code test execution. | There is no readily available equivalent test environment. |
| **AutoSar support** | Seamlessly supports AutoSar code testing and can automatically generate RTE | Supports AutoSar code testing, but does not automatically generate RTE |
| **Formalization of requirements** | An intuitive form of the editor, three steps to achieve computer understanding needs. | No such feature |
| **formal testing** | Automatically run all tests and automatically cross-check whether all forms of requirements are violated based on the results. | No such feature |
| **Automatically generate test cases based on requirements** | Automatically generate test cases to meet requirements, quantify requirement coverage, and report requirement coverage. | No such feature |
| **Formal validation** | Based on the formal requirements to automatically find code violations of requirements, this function uses the Turing Award-winning Model Checking technique. | No such feature |
| **The processor is in the Loop Test (PIL)** | with Lauterbach (Trace32) and  iSYSTEM solutions (MIL SIL PIL equivalence testing). | No such feature |
| **Hardware in the Loop test (HIL)** | It can automatically generate test cases and be applied to HIL system testing. It provides demand observer to automatically monitor system-level violations of requirements in the background system of dSPACE real-time simulator.  ask 。 | Update: MQTester 2020 has enhanced integration with HIL test tools (e.g., dSPACE). |
| **Document review** | BTC tools can click the link to jump to the document when opening the interface for easy query | It does not support directly clicking links in the interface to view documents, and queries need to be made through external channels |
| **Automatically create test environment** | 1) BTC provides a highly automated test environment creation feature. Through integration with Simulink Ribbon Bar, users can easily update and convert test environments through the interface.  2) Support for automatically adjusting test environment configurations based on the type of generated code used (e.g., EmbeddedCoder® or TargetLink®). | Supports automated generation of test environments compatible with the model, most of which require manual configuration.  -> Update: For models that are difficult to import and cannot be run, MQtester can customize the test environment and test scheme. |