Automated testing of embedded systems in medical device development

MedConf 2011 – Munich
06 Oct 2011

Mechatronic AG
Thomas Jetter & Sven Rippel
Mechatronic AG

- Founded in 1987
- 70 Employees
- Industry: Medical devices
- Business activities:
  - System development
  - Hardware development
  - Software development
  - Production Engineering
  - Manufacturing

Darmstadt (Head office)

Höhn (Subsidiary)
Automated testing of embedded systems in medical device development

Agenda

1. Introduction
2. Initial considerations
3. Conception
4. Realization
5. Application example
6. Verification / Validation
7. Summary and perspective
Why use automated testing?

- Generate unambiguous test cases
- Repeat test execution with minimal efforts
- Ensure reproducible test results
- Achieve high test coverage
- Improve efficiency during development (find defects early)
- Implement and use stress testing already during design
Terms and definitions

Levels of software testing

- **Unit testing**
  Single, isolated software module

- **Integration testing**
  Sub set of integrated software components

- **System testing**
  Fully integrated software system on target platform
Terms and definitions

SiL (Software in the Loop)

Software module (under test) ---- SW interfaces ---- Simulated software (stubs, mocks)
Terms and definitions

HiL (Hardware in the Loop)

Embedded device
(under test)

Simulated hardware
(sensors, actors)

Electrical interfaces
Terms and definitions

EiL (Environment in the Loop)

Embedded device (under test)  
With sensors and actors

Physical interfaces

Simulated environment  
(forces, temperature, …)
Automated testing of embedded systems in medical device development

Agenda

1. Introduction
2. Initial considerations
3. Conception
4. Realization
5. Application example
6. Verification / Validation
7. Summary and perspective
Core requirements

- System testing on target platform (SiL, HiL, EiL)
- Cross-project automation approach
- Good integration with external hardware
- Simple, comprehensible scripts
- Flexible document generation
- Adaptable to internal processes
Core requirements

Why test on the target?

- Timing critical sequences
- Interrupt-Handling
- Compiler errors
- Memory overflows
- Integration with hardware
Demarcation

- Not for unit level testing
- Not for production testing
Market analysis

- Many tools with SiL approach
  - Relatively simple to implement
  - All-purpose
  - Often strong focus on unit level testing

- HiL/EiL approach under represented
  - Requirements are too diverse
  - External hardware is a must

- Conclusion: Proprietary development
Additional challenges with in-house development

- Budgeting (own development project)
- Time lines (delayed availability)
- Sound system concept
Automated testing of embedded systems in medical device development

Agenda

1. Introduction
2. Initial considerations
3. Conception
4. Realization
5. Application example
6. Verification / Validation
7. Summary and perspective
Basic concept

Test framework

Feedback, Measurement results

Device Under Test (DUT)

Software Under Test

Stimulation, hardware simulation

?
Solution

Test Framework

Device Under Test

- Software Interfaces
- Hardware/Phys. Interfaces
Detailed concept

- Test Project
  - Requirements
  - Test Case Scripts
  - Assets
  - Use Case Scripts

- Test Tool
  - Editor
  - Compiler / Interpreter

- Report Generator (SW Plug-In)

- DLLs

- Target Interface

- DUT

- External hardware equipment
Automated testing of embedded systems in medical device development

Agenda

1. Introduction
2. Initial considerations
3. Conception
4. Realization
5. Application example
6. Verification / Validation
7. Summary and perspective
Scripts

Requirements for scripts

- Comprehensible and simple to review
- Direct editing in the tool, no external editor (usability)
- Access to reference data (assets)
- Adding attachments to report at run time
- Mandatory pass/fail decision
- Textual test description (for protocol and report)
Outsource complexity as far as possible:

- Target Interface
- Libraries
- Use Case scripts

Abandonment of language constructs which are difficult to comprehend (e.g. lambda expressions, linq, delegates, events)
bool Run() {
    var currents_uA = new double[] {12.5, 14.8};
    foreach (var referenceCurrent in currents_uA) {
        USECASE.ApplyCurrent(referenceCurrent);
        var measuredCurrent = TARGET.MeasureCurrent();
        ATTACH(referenceCurrent, “ReferenceCurrent [µA]”);
        ATTACH(measuredCurrent, “MeasuredCurrent [µA]”);
        if (measuredCurrent != referenceCurrent) {
            return FAIL;
        }
    }
    return PASS;
}
Interface DUT ↔ Target Interface
Test execution

- Test execution from within the tool
- Definable set of test cases to be executed
- Storage of results in a file
- Report generation on bases of these data
Requirements for report generator

- Different formats, e.g. WORD and HTML
- Use of templates
- Protocol and report generation for approval
- Trace matrices
- Test summary
Requirements for report generator

- Test case single view
  - Pass/Fail information
  - Textual description
  - Traces to requirements
  - Visualization of attachments
Automated testing of embedded systems in medical device development

### Agenda

1. Introduction
2. Initial considerations
3. Conception
4. Realization
5. Application example
6. Verification / Validation
7. Summary and perspective
The product

- Mobile in-vitro diagnostic device
- Part of a device family
- Approx. 1,000 pcs. p.a.
- 32 Bit µC
- New measurement technology
Surrounding conditions

- Communication interface and protocol pre-set by customer (device family)
- Software built on basis of existing code
- Many manual test cases for device family existing and re-usable
- Testing focused on new measurement algorithms
- Implementation of measurement algorithms optimized for automated testing in final system
Characteristics of algorithms under test

- Measurement transients consist of several hundred data points
- Complex calculations using floating point math
- Timing critical, interrupt driven
- Extensive error detection mechanisms
- Temperature compensation
Integration into the framework

Test framework

Target Interface (USB interface)

Integration layer

Measurement algorithms

Measurement hardware / drivers

Signal generator (voltage source)

DUT
Software architecture

Test framework (Integration layer)

Timer (interrupt)

Algorithm core

Access layer

Analog frontend (interrupt)

Algorithm calculations

Transient buffer
Core functions of the Target Interface

- Reading and writing of measurement transients
- Injection of simulated temperature values
- Triggering of measurement runs
- Diverse implementation of algorithms in 128 Bit floating point
- Complex comparison operators
- Manipulation of transients
Gained Experience

- Permutations: approx. 7000 in 12 hours
- Proof of stability established early and repetitively
- Script creation < 1 week
- Creation of target interface: approx. 1 month
- Minimal efforts for additional devices of same product family
Automated testing of embedded systems in medical device development

Agenda

1. Introduction
2. Initial considerations
3. Conception
4. Realization
5. Application example
6. Verification / Validation
7. Summary and perspective
Test tool in general

- Focus on core functionalities
  - Editor
  - Compiler
  - Test Execution
  - Report Provider
  - Tracing
Project context

- **Target Interface**
  - Test scripts
  - Manual test plan if applicable
  - Review

- **Scripts and assets**
  - Review

- **Libraries**
  - Test scripts
  - Unit test and review
Screen shots

Context sensitive toolbar

Project explorer

Property explorer

Document section
Automated testing of embedded systems in medical device development

Agenda

1. Introduction
2. Initial considerations
3. Conception
4. Realization
5. Application example
6. Verification / Validation
7. Summary and perspective
Summary and perspective

- Enhancements test tool:
  - Descriptions as RTF
  - Multi user capabilities
  - Integrated configuration management
  - Expand plug-in approach
  - Improve usability

- Code base (embedded)
  - Generic event interfaces
Conclusion

- Expected advantages confirmed
- Flexible mixing of SiL, HiL and EiL possible
- Testing approach fully scalable
- Tool established cross projects

Good decision!