Embedded Software Development and Test in 2011 using a „mini-HIL“ approach

Kurzfassung

Abstract
This paper describes a basic workplace setup of an embedded software developer in the year 2011. It points out today’s challenges arising from low level functionality design as well as different ways of testing such software. It includes the aspect of test automation, how to test IO drivers and how to do functional testing (black box and white box, which is code coverage) within a developer's workplace. All these are in close relationship with safety critical demands that require more in depth look into embedded design to meet required standards, e.g., to achieve a desired level of certification. The next generation workplace of an embedded software developer will combine different traditional software development tools such as integrated development environments (including compilers and on-chip debuggers) with prototype and system test equipment for hardware-in-the-loop tests such as National Instruments hardware (sBRO) and software (LabVIEW). It may be called "mini-HIL" approach for embedded software developers in order to generate and record digital and analog signals and integrate them in testing at an early stage. This is done by connectivity (open and available APIs) offered from traditional software debugging and test tools today in order to integrate and link such tool environments to HIL hardware and software.
Overview

Intense testing for software bugs has taken up more and more space in the overall embedded software development process. Different standards for various markets are reissued or adjusted over and over to enhance system and software safety: IEC61508 (functional safety of electrical/electronic/programmable electronic safety-related systems), ISO CD 26262 (functional safety of road vehicles), DO-178B/C (required by FAA and EASA for the certification of software for use in aviation), ISO/TS 16949 (combines existing general quality management system requirements), DIN EN 62304 (software lifecycle processes of medical device software), IEC 60601-1 (3rd edition, safety requirements for medical electrical equipment) etc. These standards and many more, have considerable impact on the actual software development process and the setup of a developer’s/tester’s workplace already today.

Basic setup of a software developer’s workplace

We could argue what a basic embedded development setup is these days. But we won’t. In this paper it will be an integrated development environment (IDE) that provides all necessary programming tools (editor, compiler etc.), a debugger and of course a target as depicted on picture 1. At this point we can split the lacks of this kind of setup in three groups:

First group would be easiest to overcome and is the debugger functionality. Basic debugging (download, run control and memory access while stopped) is not enough in terms of embedded CPU measuring such as memory accesses and execution logging (trace), code and decision coverage, code profiling etc. As said this can be easily solved choosing the right micro controller and the appropriate debugging tools available on the market.

Picture 1: Basic Software Development Workplace Setup

Extended setup of a software developer’s workplace

Second group especially gives headaches to developers designing low level functionality such as IO drivers. In that stage it is important to have a responsive environment to the target under development. When the target is a part of a complex system, giving stimulus to the target can be a challenge, sometimes just due to physical limitations like placing the whole system on a developer’s desk. A solution is a simulator which is a new piece of hardware in the overall development setup. Time needed to construct one is critical and should be as short as possible. NI's rapid prototyping solution based on real time platforms such as PXI or RIO is suitable in
that terms. With a simulator the basic setup (picture 1) gets extended as shown in picture 2. Note, that you also gain a user interface to the simulator with prepared NI constructs.

**Picture 2: Environment Simulation**

The most challenging problem to solve hierarchically lies above all mentioned and that would be connectivity of all these components. It is important while gathering data from a target, i.e. to obtain simulation context. Even more so in automated processes such as regression tests. To address connectivity, iSYSTEM provides an open and flexible interface (called isystem.connect) which bundles all available data from development setup into a single pipe that can be used in third party applications. Picture 3 shows how LabView can be a client to iSYSTEM's development setup with its debugger hardware interfac to the target and the IDE providing the connectivity APIs.

**Picture 3: LabVIEW – isystem.connect client**

Next to a debugger iSYSTEM's solution can also connect to any simulation hardware through a predefined interface. Doing that simulation hardware turns into hardware-in-the-loop (HIL) because a distinctive loop is formed with all development blocks (picture 4) giving the user correlated context from a target and the simulated environment.
To extend the development setup combinations mentioned above, connections can be made implementing HIL as NI's real time target driven by LabVIEW (picture 5). While developing software simulation parameters can be tweaked within LabVIEW while the debugger is used to set up parameters during automated testing. It is up to the user to use them as he/she sees fit.

Last development setup lacks the target context in LabVIEW if a user decides to use a LabVIEW application as topmost application for in-system measurements on an embedded target.

Connection combination on picture 6 addresses this with usage of the previously mentioned generic API isystem.connect. This interface also exposes connectivity to a HIL through the debugger IDE which makes a direct connection to the HIL from within a LabVIEW application optional. This development setup enables HIL and target context correlation on both LabVIEW and a debugger’s IDE level.
Example using development tools from iSYSTEM linked to NI LabVIEW and sbRIO

This integration example is based on iSYSTEM's standard debug tool which is enclosed into a development loop with NI's sbRIO platform as an HIL. The implementation was done with iSYSTEM’s VI library for LabVIEW. This VI library was extended with HIL adaptation, communication and configuration functionality. Thus, access to exposed parameters is gained in a custom LabVIEW design.

![Demo Setup](image1)

**Picture 7:** Demo Setup

![LabVIEW application using iSYSTEM VIs for HIL integration](image2)

**Picture 8:** LabVIEW application using iSYSTEM VIs for HIL integration

The sbRIO is used as an adjustable output voltage source. Voltage is fed to an embedded target under development where a firmware is responsible to convert that input voltage and act accordingly on its output pins which are controlled back to sbRIO.

![FrontPanel application](image3)

**Picture 9:** FrontPanel application
As a test example a firmware function converting the input analog value is tested in a way of changing the output on sbRIO through the prepared interface and evaluating the firmware’s variable storing the conversion result through the debug interface. The test is performed using a script executed within the isystem tool environment.

```python
# Function under test

hiloOutV = 0.0
while hiloOutV <= 5:
    print "Generating on sbRIO HIL: " + str(hiloOutV) + " V",
    # Change voltage through HIL interface#
    hilo.write("VOut.W0:=" + str(hiloOutV))

    # Evaluate variable with conversion result through DEBUG interface#
    val = round(isystem.connect.IConnectDebug.fRealTime, "g_fVoltage")
    print "Measured on embedded target: " + str(val) + " V",

    # Compare both values and print the result#
    if abs(hiloOutV - val) > 0.05:
        print "FAILED"
    else:
        print "OK"
    hiloOutV = hiloOutV + 0.5
    time.sleep(0.5)
```

**output:**

- Generating on sbRIO HIL: 0.0V Measured on embedded target: 0.01V OK
- Generating on sbRIO HIL: 0.5V Measured on embedded target: 0.41V FAILED
- Generating on sbRIO HIL: 1.0V Measured on embedded target: 0.91V FAILED
- Generating on sbRIO HIL: 1.5V Measured on embedded target: 1.4V FAILED
- Generating on sbRIO HIL: 2.0V Measured on embedded target: 1.95V OK

**Summary**

Adding and integrating „mini-HIL“ equipment to a developer’s workplace provides a flexible base for professional embedded development with reusable rapid prototyping simulation hardware from National Instruments and a versatile connectivity solution of traditional software development and test software and hardware from iSYSTEM to enable even further expansion with scripting or unit testing functionality. Embedded software developers will learn how to start testing their applications while developing the software. This early stage testing is most likely not done today because it is time consuming, different and new tool environments are needed etc. On the other hand embedded software testers will learn how to work together closely with software developers and what synergies can be achieved using “system test” type of equipment already in development.