powersoft nineteen QSQA
consider it done!

Increasing Test Coverage In Embedded Applications

White paper - August 2014

Table of Contents

Introduction | 1
Algorithms and their Role in Safety-Critical Embedded Systems Testing | 1
Using Debuggers for Algorithm Testing | 4
Concluding Remarks | 6
About Powersoft 19 SQA | 7

Introduction

Overview

For safety-critical embedded systems, precision of environment variables being measured is of the highest importance. This white paper summarizes Powersoft19 SQA team's approach towards efficiently handling the environment variables in safety-critical embedded systems testing. The paper describes the details of our experience in testing these systems using algorithms and JTAG (Join Test Action Group) as debugging tool.

Background

All the information in this paper is based on the real world projects that were successfully completed by Powersoft19 SQA team. Most of the experience is drawn from testing the instruments that warn mine workers of hazardous atmospheric conditions. These devices are required to sense toxic gases in environment, record these as data, and then turn them into visible readings on the instrument screen.

Once the instruments sensed the atmosphere and converted the toxicity to electrical signals; the signals were read in the firmware as variables. These variables were used to calculate the readings on the output screens for humans. In order to test these devices, it was required to test the variables and their integrity against the underlying mathematical algorithm with absolute precision. To successfully carry out this testing of the algorithm, we needed to perform tests based on a specific set of values acquired from the algorithm
mathematically. This specific set of values included all border conditions, upper and lower limits of the algorithm in test.

At Powersoft19 SQA's testing Center of Excellence (CoE), we tested the limits and boundary conditions of the algorithm in question on the safety-critical instruments. This was carried out in real-time and real world scenarios to properly emulate the actual behavior. For this purpose we created boundary conditions, upper and lower limits, and completeness coverage of the algorithm and seeded these values into the variables in the code in a real-time environment. The best results were achieved by using JTAG (Join Test Action Group) standard which enabled us to recreate the actual scenario and a step by step execution of the firmware.

Algorithms and their Role in Safety-Critical Embedded Systems Testing

Introduction of Algorithms

Algorithms are a step by step procedure followed in order to solve a problem or accomplish a task. Similarly, on computers and embedded systems an algorithm provides step by step exact instructions for the system to follow in order to accomplish its end goal. In a system, an algorithm is fed with some input data; this data is processed by the algorithm and the result is the output.

A typical example of a mathematical algorithm in software:

```
Step 1: Start
Step 2: Declare variables age, fixedAge, result
Step 3: Read value age                                  //From Input source
Step 4: Compare age with fixedAge
      If age > fixedAge then result = "Older"
     Elseif age < fixedAge then result = "Younger"
     Else result = "Same"
Step 5: Display result                                  //At output
Step 6: Stop
```

Algorithm Testing

Based on the requirements and design of the algorithm we are able to calculate the outputs corresponding to a set of input data and likewise the input data for a set of required outputs.

The foremost testing that can be performed on the algorithm is boundary value analysis on both the input and output data.

Boundaries of input data age:

- Age cannot be below or equal to zero
- Age cannot be above 99 (let's say it is an industry requirement)

Boundaries of output data result:

- result can only be "Older", "Younger" or "Same"

Boundary value analysis is a test case design technique to test boundary values between both valid and invalid boundary partitions. It is an input or output value which includes minimum and maximum values at inside and outside boundaries.

Test scenarios at boundaries:

- If input age is -1
- If input age is 0
- If input age is +1
- If input age is 99
- If input age is 100
- If input age is 101
- If input age is fixedAge

How does the system respond?

Based on the requirements we would know the desired outputs on specific inputs, inside or outside the limits. We can create test data corresponding to the desired algorithm behaviors and likewise test the algorithm's actual behavior based on this test data.

Expected results when fixedAge is equal to 30:

- age = -1, result = Error invalid age
- age = 0, result = Error invalid age
- age = +1, result = Younger
- age = 99, result = Older
- age = 100, result = Error invalid age
- age = 101, result = Error input condition
- age = fixedAge, result = Same

In order to decrease the amount of test data and decrease the testing time we can use the Equivalence class partitioning in our test data designing. According to the Equivalence class partitioning, all the input and output data is divided by placing them into classes. All the values that are present in the same class have the same result with respect to the behavior of the algorithm; therefore, to decrease the number of tests, the test values are picked from each class for testing.

Expected results when age and fixedAge is equal to 1, 30, 70, 99:

- age = fixedAge, result = Same

Challenges In Algorithms Testing

Based on mathematical formulas of the algorithms we can calculate the output values for inputs and vice versa. Subsequently, we can perform functional Black Box testing on our system using these input values to test the algorithm. Depending on the system the input values can be of various types like digital data, analog values, or timing specific values.

The major problems, that we faced in our real world project, were related to the analog values and the timing of specific values. To perform Boundary value analysis, our input values or timings needed to be accurate in order to verify the algorithm.

Expected results when age is 5 years, 2 months, 3 days, 16 hours, 40 minutes and 3 seconds:

- age >= fixedAge, result = Same?

When performing functional Black Box testing on the system, providing specific real values at specific times was a big challenge as it resulted in hits and misses with respect to timing; resulting in variable time for testing of the algorithm.

Practical implementation of Black Box testing has certain shortcomings. Firstly, we cannot observe the behavior of the system with precise input values because these values cannot be set at Black Box level. Secondly,
even if it was possible to set precise values, the instance at which the values needed to be set to recreate the scenario is almost impossible. Coming up with a solution to these problems associated with purely Black Box testing was a challenge; the solution was uncovered in algorithm testing.

Using Debuggers for Algorithm Testing

Embedded Application Debuggers

A debugger is a programming tool used to test and debug target programs. Nowadays, it features a graphical user interface (GUI) for the interpreter, which can be used for debugging and testing of programs. Debuggers provide sophisticated functionalities such as:

1. Step by step execution of program
2. Pausing program execution to examine current state of the program
3. Halting on specific instructions via breakpoints
4. Tracking values of variables
5. Jumping to different lines of code
6. Monitoring the memories

An important task of the debugger is to relate the main memory contents (e.g., machine code or data bytes) back to the original program.

Once the appropriate modules are interpreted, breakpoints can be set at relevant locations in the source code. Breakpoints are set on a line by line basis. When the hardware reaches a point in the code where a breakpoint is placed, it stops and waits for commands (step, skip, continue, etc.) from the debugger (programmer or tester).

In-Circuit Debugging

With the help of JTAG, that provides the debugger an interface to the processor on the system, the program can be run on the actual hardware and tested for actual behavior against expected behavior.

By using this debugging technique, we can perform algorithm testing more effectively. This technique has following advantages over functional Black Box testing:

1. Ability to execute the algorithm line by line
2. Capacity to monitor and edit any values we need to change
3. Provision to provide precise values
4. Facility to manage timing perfectly in order to provide specific values at specific times
5. Capability to pause the execution at any desired point and alter the execution of the algorithm when needed by using the breakpoints

JTAG

JTAG (Joint Test Action Group) is the common name for the IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture. It is also widely used for IC debug ports. It can provide debugging capability to perform single stepping by using breakpoints.

An in-circuit emulator which uses the JTAG provides access to the debug modules present on chip inside the target CPU. Access to these modules provides the capability to debug the software directly at the machine instruction level or in terms of high level language source code.

With the JTAG connected to the pins of the microcontroller, we can access the entire memory of the microcontroller and run it in desired manner.

The basic building block of a JTAG is the Test Access Point or TAP controller. This allows access to all the custom features within a specific processor, and must support a minimum set of commands. In addition to a reset, commands exist to read and write registers beyond the TAP controller, and to bypass this device.

For embedded device developers, a JTAG connection is often the only debug channel that can be used. One of the advantages of the JTAG is that it does not interfere with the regular device operational modes.

Advantages of Using JTAG

Cost Reduction

To perform algorithm testing, we can use in-circuit debugging instead of buying new expensive tools to support the testing of algorithm functionality. Program development environments contain an editor, a compiler, a linker and a debugger. We can run the debuggers in the debugging mode and connect it via a Jlink using the JTAG protocol. This allows us to utilize the existing resource to perform algorithm testing without spending money on costly equipment.

Integration

The JTAG protocol support comes embedded in most modern processors and almost all Integrated Development Environments used to develop software.

Efficient Memory Usage

Once the JTAG is connected to the instrument, the memory of the instrument can be easily accessed. Other processes such as flashing the firmware, clearing the memory and resetting the microcontroller can also be performed.

Overall Testing Efficiency

Using algorithm testing, a lot of test scenarios can be created which are somewhat impossible or cumbersome through conventional Black Box testing such as error seeding. We can test certain errors on the device by manipulating the flow of the firmware through the debugger, which otherwise would require physical damage of the instrument rendering it useless for further testing.

Concluding Remarks

With the smart use of algorithms and JTAG as in-circuit debugging tool, Powersoft19 SQA team efficiently expanded the test coverage of safety-critical embedded systems. The real world safety-critical instruments testing project–that is the basis of this white paper–provides a trustworthy evidence in favor of using algorithm and JTAG as debugging tools.

What More can You Achieve with In-Circuit Debugging

The benefits of using in-circuit debugging for testing does not just end with testing the functionality of algorithms. Here is a list of some more amazing testing feats that you can accomplish by using in-circuit debugging:

1. Using either jumps or the step by step functionality in debuggers you can test all the control paths of the program by simply altering the values at each node.

2. You can control the working of the instrument by stopping the flow at desired locations in the code by placing breakpoints. This type of testing is commonly known as halt mode debugging.
3. By using in-circuit debugging capability, you can perform fault coverage i.e. faults resulting from tampering with the hardware can be avoided by simply setting the flags in the instrument and subse quently verifying the program behavior under fault conditions.
4. With in-circuit debuggers, you can traverse to any line of code in the program by controlling the flow through the debugger and altering the behavior on the actual hardware.

About Powersoft19 SQA

To compete in this race of bringing newer and better products to market faster than ever, Powersoft19 SQA adopts a unique approach. We revive the roots of quality assurance with a focus on increasing the efficiency rather than mere corrective actions. We believe in providing real, measurable value for the investment in Software Quality Assurance. We achieve this with a result-oriented approach of reducing the overall product cost and increasing the quality at the same time.

Being independent Quality Assurance consultants, we offer end to end quality assurance services focusing on delivering value at each step of the software development process. By introducing quality assurance early on in the development process, we help organizations meet time to market deadlines by mitigating risks early in the process. Powersoft19 SQA acts as a Testing Center of Excellence where knowledge, skills, tools and processes are shared among projects.

With this integrated approach towards quality testing, we have devised processes to get a product's quality right the first time. Our clients leverage our expertise and experience to achieve higher efficiency and better quality. Leading names in the rail, gas detection, mining, health care, material handling, and power industries engage us for a variety of services ranging from requirements analysis to agency approvals.

Contact Us

Explore ways to use our expertise in growing your business while establishing a valuable partnership with us.

Contact our consultants at:

Phone: +1.412.533.1700
E-mail: sqa@powersoft19.com
Website: www.powersoft19.com/sqa