

DISTRIBUTED BUILT-IN SELF-TEST ADMINISTRATION USING MOBILE/WAP ACCESS

L. Miclea, Enyedi Sz.

*Technical University of Cluj-Napoca, Automation Department
Barițiu str. 26-28, Cluj-Napoca, Romania
Liviu.Miclea@aut.utcluj.ro, Szilard.Enyedi@aut.utcluj.ro*

Abstract

This paper presents the idea and experimental results on distributed built-in self-test (DBIST) administration, using a combination of Web technologies, PHP scripting and WAP mobile data access. The devices and tests are selected on an electronic WAP form, in a WAP microbrowser of a mobile device. The WAP browser then sends the selections to the DBIST web server, through the WAP gateway. The server uses PHP scripts to start the selected tests on the selected devices. The scripts use TCP/IP to communicate with the BIST modules of the devices. The BIST modules run the required tests and return the results to the scripts, which generate a WML page with the test results and send it back to the requesting mobile device's microbrowser.

In the version presented here, the PHP scripts generate a human-readable "test results" WML page, but in future implementations, they can use raw TCP/IP or other protocols to communicate with a central administrative authority, other DBIST networks or mobile networks, interact with SQL databases or do other high or low level test configuration/test results management.

Keywords: BIST, DBIST, WAP, PHP, TCP/IP

1. INTRODUCTION

1.1. Generalities

This paper continues the work [1] of the authors for developing solutions for DBIST systems. This paper, however, presents a different approach. The test selection, running and displaying the results is accomplished through the aid of a Web server, PHP scripting, on-the-fly WML pages and a WAP-enabled microbrowser in a mobile device.

In the following, we will present some basics of the concepts, and then a simple example to illustrate the idea.

1.2. DBIST

Built-in self-test has been around for quite a while. One of the current trends in BIST technology is Distributed BIST, or DBIST. The distributed nature of DBIST means that each of the modules in the DUT has its own BIST routine, which runs the test more or less independently from the other modules. This way, the actual BIST of the whole device is decomposed into smaller, dedicated BISTs, which should be simpler and easier to develop and maintain.

1.3. PHP

1.3.1. Generalities

PHP (Personal Homepage Preprocessor) is a scripting language, used mostly on web servers. The idea is that the PHP commands are embedded in the HTML file itself. When a browser requests a page from the web server, the server first runs the PHP interpreter on the requested file, and then sends the results to the requesting browser.

1.3.2. Example

```
<html>
<body>
<?php
echo "This comes from the PHP script.<br>";
?>
</body>
</html>
```

When the above example is parsed by the server's PHP engine, the `<?php ... ?>` tag is replaced by the result of the contained PHP commands, i.e. by the string "Test text.
". Only then is the resulted page sent to the browser that requested it.

1.4. WAP

The mobile communications industry invented WAP (Wireless Access Protocol), a way through which mobile devices, for example mobile phones, can request and display information from the Internet.

Through the WAP protocol, these devices can access and display WML (Wireless Markup Language) pages, which are very similar to HTML pages. Still, WML pages have less detail and are optimized for the monochrome and small display of the mobile devices. Such a WAP-enabled device contains a WAP microbrowser that can parse and display WML pages.

1.5. WML

WML files are called "decks", and the decks are composed of "cards". WML is an XML language, with markups, like HTML, but the markups are different from those in HTML.

1.5.1. Example

The code below is WML code for a deck that contains a single card:

```
<wml>
<card title="Input">
<p>
Name: <input name="Name" size="15"/><br/>
Age: <input name="Age" size="15" format="*N"/><br/>
Sex: <input name="Sex" size="15"/>
</p>
</card>
</wml>
```

And the corresponding page in the microbrowser:

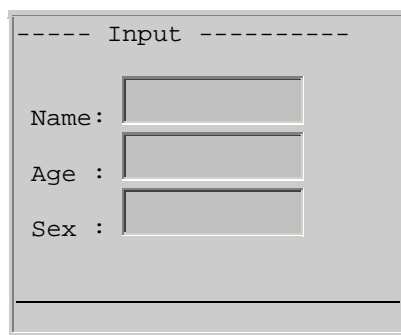


Figure 1 – A simple form as displayed in a WAP/WML microbrowser.

1.5.2. PHP and WML

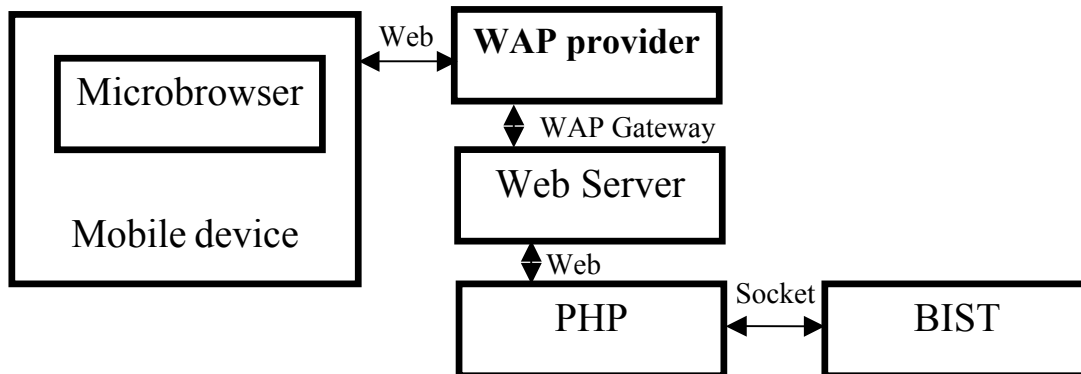


Figure 2 – The structure of the DBIST system using WML/WAP and PHP

The steps of the DBIST process:

- The microbrowser requests from the WAP provider the page containing the PHP script on the specified web server;
- The WAP provider connects to the specified web server and transmits the request as it was from a normal desktop browser;
- The web server (actually, the PHP script) creates on the fly the WML form and sends it to the WAP provider, which sends it further to the mobile device;

- The user fills the data in the form and sends it back to the WAP provider, which sends it to the web server;
- The PHP script on the server opens a TCP/IP socket, connects to the specified BIST module of the specified device and sends it the user-requested test;
- The contacted BIST module runs the requested test and returns the results to the script, through the same TCP/IP connection;
- The script generates the results WML page and sends it to the WAP provider, which forwards it to user's browser, where it is displayed.

2. IMPLEMENTATION AND EXPERIMENTS

2.1. The experimental BIST module

The BIST module [1] we used for experimenting is a software module. It accepts commands that start the BIST tests – check the available space, share name etc. of the drives.

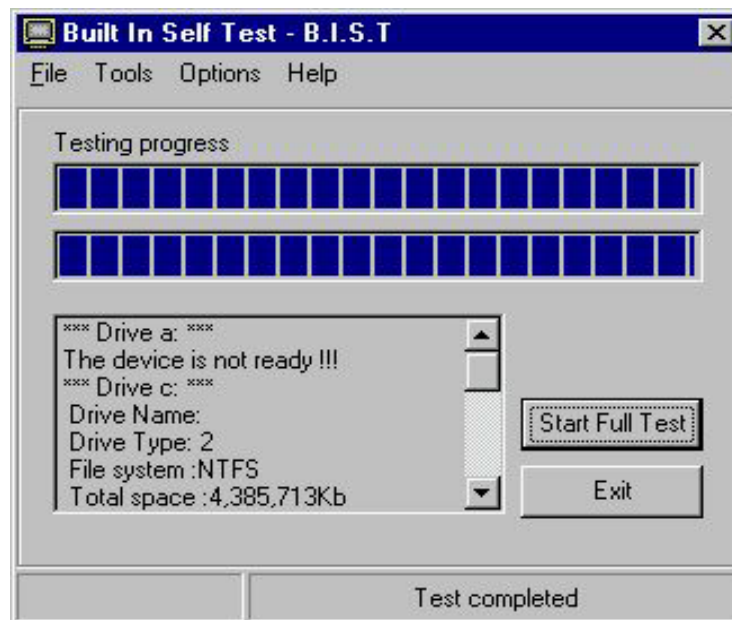


Figure 3 – The BIST module's graphical interface

The commands are expected in ASCII format, on TCP/IP socket no. 2811. Only known commands are executed, the unknown ones are sent back to the requesting agent.

The BIST module executes the requested test and sends back the results on the same port on which the command arrived.

2.2. The PHP script and the WML pages

The PHP script creates the WML form, manages the DBIST testing system and also generates the results WML page.

The code that generates the WML form:

```
echo "<anchor>";  
echo "Send";  
echo "<go href=\"\$PHP_SELF\" method=\"POST\">";  
//POST variables to send back  
echo "<postfield name=\"addr\" value=\$(addr)/>";  
echo "<postfield name=\"test\" value=\$(test)/>";  
echo "</anchor>"; //closing the "refresh" and "go" block
```

Note that we have to specify, for the microbrowser, the data which is to be sent back to the server, using the <postfield> tag. An HTML browser knows how to do this automatically.

We will focus on the BIST connectivity part below.

We try to connect to the BIST module at the address specified in the returned form, port 2811:

```
$fp = fsockopen ($addr, 2811);
```

If connection was successful, send the user-requested test name to the BIST module, all uppercase:

```
fputs ($fp, strtoupper($test).".");
```

Receive the test result from the BIST module, maximum 128 characters; append to the result string as we receive, until the transmission end character (“.”) is met; the “.” (dot) operator is string concatenation in PHP (which is, coincidentally, the same as the end-of-transmission character we chose):

```
$recvd = fgets ($fp,128);  
while (!preg_match("/(\.)/", $recvd))  
    { $recvd = $recvd . fgets ($fp,128); }
```

Put the received result into the generated web page:

```
echo "<big>$recvd</big>";
```

Close the connection to the BIST module:

```
fclose ($fp);
```

3. CONCLUSIONS

In this paper, we presented a DBIST architecture that uses PHP scripts to let the user select the tests remotely, from a WAP-enabled mobile device. The scripts generate the WML form, transport the user selection to the BIST modules through TCP/IP and give the test results back to the user's mobile device. A simple example that shows the basic idea has been discussed.

PHP offers high flexibility, and rapid development, since all that is required from the BIST modules is TCP/IP and ASCII-based communication. Furthermore, PHP is very similar to C, so the learning curve for senior developers is fast, offering a convenient way for Web-enabled DBIST management. The possibilities are wide.

WML/WAP is the standard markup language for Internet content on mobile devices, therefore it is device-independent.

On the other hand, since PHP is an interpreted language, it is slower than native, compiled/linked binaries. But the speed overhead is not in the scripts, but in the BIST modules themselves; however, even these BIST routines are rarely called, compared to the normal functioning of the device. Another limitation may be the need for a web server, although tiny embedded web servers are becoming a one- or two-chip commercial product.

Another drawback is that if the mobile device does not have a convenient keyboard, it is compulsory to use selection-based form elements.

A future – necessary – improvement is user authentication.

4. REFERENCES

1. L.Miclea, Enyedi Sz., R. Orghidan, (2001), “*On-line BIST Experiments for Distributed Systems*”, IEEE European Test Workshop ETW'2001, Stockholm, Sweden, May 29th–June 1st, , pp. 37-39.
2. L. Miclea, Enyedi Sz., H. Vălean, (2000), “*Remote Data Acquisition and Control Using Programmable Structures*”, Proceedings of International Conference on Quality, Automation and Robotics Q&A-R 2000, Cluj- Napoca, Romania, 19th–20th May, 2000, Tome 2, pp. 425-430.
3. Monica Lobetti Bodoni, A. Benso, S. Chiusano, G. di Natale, P. Prinetto, (2000), “*An Effective Distributed BIST Architecture for RAMs*”, Informal Digest of IEEE European Test Workshop ETW 2000, pp. 201-206
4. R. Pendurkar, A. Chatterjee, Y. Zorian, (1996), “*A Distributed BIST Technique for Diagnosis of MCM Interconnections*”, International Test Conference 1996 Proceedings, pp. 214-221
5. Y. Zorian, H. Bederr, (1996), “*Designing Self-Testable Multi-Chip Modules*”, European Design and Test Conference 1996 Proceedings, pp. 181-185
6. ***, “*PHP Manual*”, (2002), <http://www.php.net/docs.php>
7. ***, “*WAP Tutorial*”, (2001), <http://www.w3schools.com>