Testing the Target system

Vijayalakshmi **K Rajanikanth**
vijaya210@gmail.com **rajanikanth@msrit.edu**
Department of Information Science & Engineering
M S Ramaiah Institute of Technology, Bangalore

Abstract

An embedded system consists of heterogeneous layers including hardware, HAL (Hardware Abstraction Layer), OS kernel and application layer. Interactions between these layers are the software interfaces to be tested in an embedded system. The identified interfaces are important criterion that selects test cases and monitors the test results in order to detect faults and trace their causes. Because embedded systems commonly have slow processor and small memory, embedded software must be more efficient and compact against the poor resource. In this paper I discuss some of the tools used to test the target board (embedded system) using host system and propose my tool to write the scripts and detect the defects in the target board. Tool is called as Card Advanced Research Tool (CART) which run on host system.

Index Terms- ***Card Advanced Research Tool (CART), Target board, host system, firmware, Multimedia card (MMC).***

1. Introduction

Presence of the Ubiquitous, 'Embedded, Everywhere' from the cell phones to D-TV makes the embedded systems rapidly widespread in real life. Embedded software occupies around 10~20% of embedded systems. More than 80% of the system faults are caused by not hardware but embedded software. Considering the trend of increase in embedded software, its problems are no longer considered as a simple software fault. Therefore, the embedded software test becomes very important [1].

Furthermore, the importance of software in the embedded system's market has an impact on testing. Time-to-market and quality become major competing factors. It means that more and more complex software has to be developed at a high quality level in less time. A well- structured test process alone is not sufficient to fulfil quality demands within stipulated time. With the proper use of test tool, the faster testing with good quality is possible [1].

As an embedded system generally offers less computing resources than a general-purpose computer system does, developers make every effort to improve the quality of their embedded software and make it to always have a good performance in resource usage. To do this, developers occasionally use embedded software evaluation tools to increase development efficiency for embedded software. With a software evaluation tool, developers get to know whether the developed software is efficiently optimized for embedded system's restricted resources [2].

Recent trend shows that embedded software is tested via a debugger equipped from the simulator/emulator or a test tool to measure the performance of entire embedded system. Each of the case has some limitations to test embedded software.

Firstly, in order to detect a fault, the debugger allows software engineers to set a break point, determine a symbol to be monitored and decide 'pass' or 'fail' of a test result. If software engineers are not quite experienced in testing or they do not know the architecture of entire embedded system then they cannot execute the ad-hoc testing by trial and error. They take more time to test, and they cannot confirm whether the test was completed reliably and the fault evaluation has been performed properly.

Secondly, in order to solve the hardware dependency for embedded software testing, the test tool should support the test environment such as a simulator, an emulator, and a real target. The test tool focuses to automate the test execution based on the test environment. A good test tool should support the labour-intensive tasks such as the test execution and the technology-intensive

tasks including test item identification, test case selection and test result analysis oriented to embedded software. In order to help the software engineer, a test tool should support above mentioned tasks.

Thirdly, they started using the tool which run from the host system to detect the defects in the target board. The output they could see from the host system itself. The limitation is serial communication resulted in slower execution speed.

Many test tools have limitations to analyze and test the internal features of embedded software that is tightly coupled with OS, middleware, device driver and target hardware. To analyze the interfaces between the heterogeneous layers is very important to the embedded software testing. As it is impossible to test embedded software independently, it is to be tested on the entire embedded system. Testing the entire embedded system is difficult to detect the potential software fault, its location and cause. Therefore, the interface test of embedded software is important to identify the fault location.

In this paper, I propose an automated tool to test the embedded software (called as firmware) on the target board. This tool shows the results of test cases that run in target board on the host system and provides user, efficient and friendly environment to test and detect the defects in the MMC/SD attached to the target board.

2. System Design

The System composed of many layers which can be divided among host side and target side. [3]

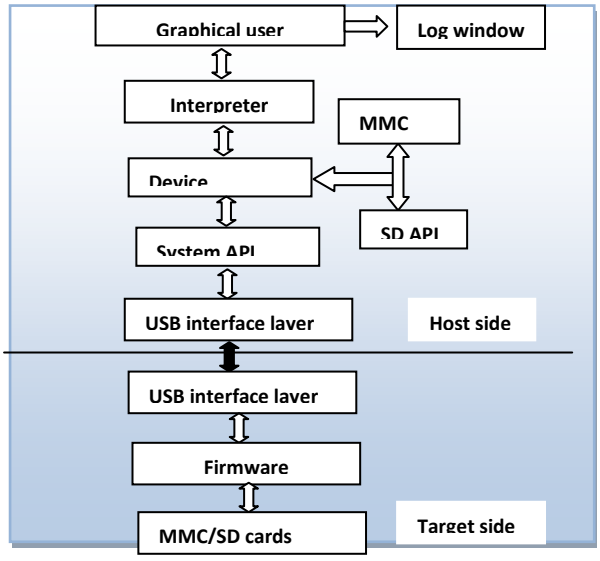

Fig 1: Block diagram

2.1 Graphical User Interface: This layer provides a user interface through which user can access the complete functionality of the tool. A GUI avail visual indicators, instead of text based commands. GUI provides user-friendly environment, where novice user can learn the features easily.

2.2 Interpreter: Interpreter supports C language grammar. All system APIs are declared in this layer. Scripts written in GUI to test the firmware will be interpreted into the form that is understandable to the next layer. Language's data type conversion complies with standard C and allows user to do type conversion. Scope of variables defined is within the script and all un-initialized variables have integer '0' as initial data.

2.3 Device Manager: The main functionality of the device manager is to select either the SD module or MMC module depending on the device selected.

2.4 System API: Script cannot define or declare functions. Instead, CART provides predefined functions called as System APIs. APIs are defined in this layer, but declaration must be there in interpreter layer. There are 2 type of system APIs:

- **CART Board control APIs:** API that send command to and get response from CART target Board. These APIs need communication with a device through USB and the Board. These APIs are MMC specific commands and cannot change the structure and functioning.

- **CART PC control APIs**: CART control API is the function that need not communicate to the target board. These API will be used by user to get PC control command. These APIs are system APIs can be defined by developer and can change the structure and function as he needs.

2.5 USB Interface layer: This layer will provide the USB interface for API to communicate to target board. Similar module

will execute on the target board which will communicate with the host PC.

2.6 Firmware: This module will control the complete board. The firmware will be modified to add the functionality of the SD, MMC and USB support. This firmware is loaded into the target board via binary file.

2.7 Call flow for command API

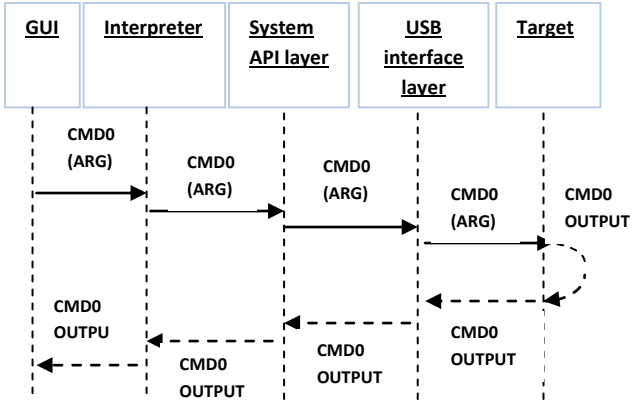

Fig 2: Cal flow for command API

Fig 2 shows the call flow of the commands, from GUI to target board. Once GUI recognizes commends it will send it to interpreter for declaration of the command. Once commands are found in interpreter layer, it will check the definition of API in system API layer. From API layer command checked in target side, ie. in USB interface layer and target (MMC).

3. Testing the target board using CART

3.1 Writing scripts

CART tool provides user-friendly interface from the host system. When application is launched, user should download the firmware into the target board using binary file. Once firmware is downloaded, host system can be connected to target using USB interface. Once USB connected status shown users can write the scripts and should save the scripts with .c extension in preferred folder.

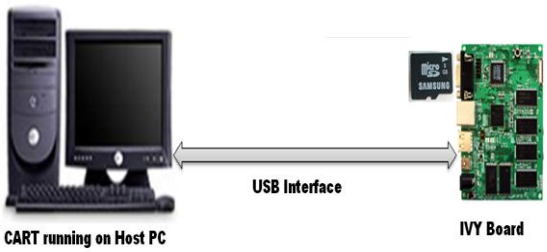

Fig 3: CART System

script which contains commands supported by MMC 4.41 specification. The target (IVY) board is connected to host PC and through the USB interface. When application run the script from GUI, commands will be executed from target firmware and the result can be seen in host system. Step by step execution of the commands can be seen in the host system, and errors can be traced.

3.2 Testing scripts

Test scripts supports C language constructs and system specific Application Program Interfaces (APIs).

Script language supports C data types, structures and arrays. Script uses cart pre-processors. Script editor has the features of syntax colouring, parenthesis matching. Scripts are saved with .c extension, and can run as single or in a group. One script can be imported in other script file using SCRIPT API with filename as argument.

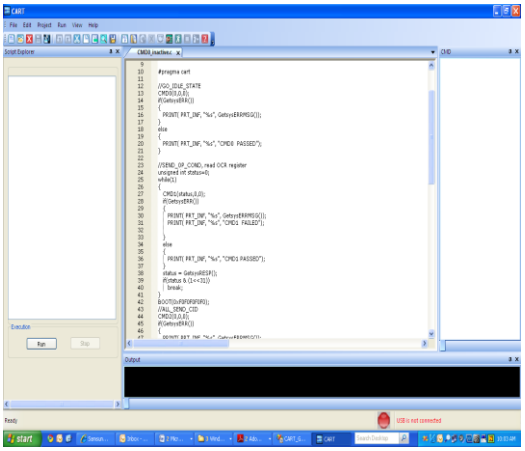

Fig 4: CART running in host system

4. Conclusion

CART provides faster communication speed with target board. Users are allowed to write

the test cases and execute the test cases using single tool. This avoids user searching for different tools and learning the user interface of tool for writing scripts and executing scripts. Tool supports both MMC and SD cards, test cases written will run in similar way in both the cards. User can load the tool with minimum hardware configuration on host system provided with target board, MMC and USB connection.

5. References

[1] Jooyoung Seo, Ahyoung Sung, Byoungju Choi, Sungbong Kang, "Automating Embedded Software Testing on an Emulated Target Board", 'Second International Workshop on Automation of Software Test', IEEE 2007

[2]Dongkyu Kwack, Yongyun Cho, Jaeyoung Choi, Chae-Woo Yoo, "A XML-based Testing Tool for Embedded Softwares", IEEE 2007international conference on multimedia and ubiquitous Engineering.

[3] Design document and programmers guide of CART.

*[4] **Link**: http://ezinearticles.com/?How-to-Evaluate-Embedded-Software-Testing-Tools&id=4920001*