Test System Architecture
A Fundamental Model For Moving Toward Common Test Platform

S.A. Chuah
Land Mobile Product Sector
Bayan Lepas Free Industrial Zone,
Phase III, 11900 Penang,
Malaysia
sachuah@comm.mot.com

Abstract - *This Paper defines a fundamental Test System Architecture Model for the test engineer to reference in order to move toward a standardized common test platform.*

INTRODUCTION

In the past, the test system engineer while working on improving the test system tries very hard to standardize the system. However, no system model had been created for the test engineer to reference.

This paper proposes a test system architecture model which is built by using seven fundamental modules. Using this new architecture model, standardizing the system is made simple, more clear cut, more structure, more manageable and well define.

BACKGROUND

There are seven fundamental modules in the system model:

1. User/Machine Interface
2. Tracking System (Database)
3. Controller
4. Radio Comm (Radio Protocol)
5. Hardware Interface (Fixture)
6. Equipment
7. DUT/UUT (Radio Design)

Before going into detail of this model, let's look at how the architecture model is formed.

FORMING THE MODEL

Initially we break down the test system set-up into two major parts, hardware and software. Hardware is the physical equipment set-up and software is the core and brain that interconnects the hardware that drives the entire system.

HARDWARE BAGROUND

Figure 1 shows a block diagram for the hardware setup in a typical test system.

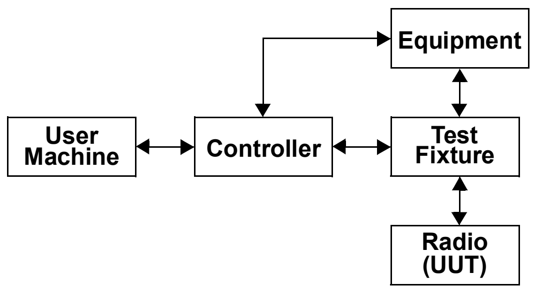

Figure 1 : A Fundamental Building Block Test System Hardware Setup

In general a group of equipment is connected to a controller as well as the test fixture. The test fixture is also connected to the controller for fixture control. A UUT/DUT (Radio) is then placed into the fixture for testing. User (can be human or machine) will then trigger the controller through some interface (can be keyboard, scanner or some kind of sensor) to activate the test system in order to execute the test. Upon completing of the test, the system will indicate the pass/fail status thru some interface (monitor, indicator light, switch. etc) to alert the user.

SOFTWARE BACKGROUND

Figure 2 shows a block diagram of a typical software test program.

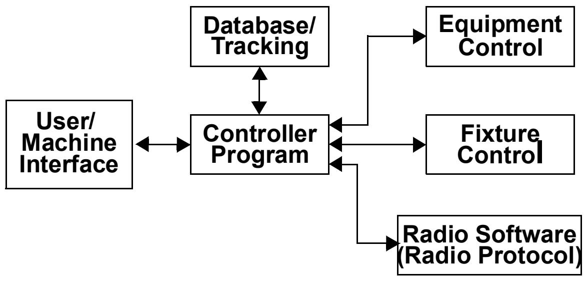

Figure 2 : A Fundamental Building Block for Test Software

A typical test software program is built in a few major functional block. We normally have a group of functions to handle user / machine request that activate the system and also obtain testing status from the system. The tracking system module is used to help in obtaining radio (UUT) information. We also have a set of functions that are used to set up the equipment and a set of functions are used to set up the fixture control. Normally, there is also a set of radio commands (Radio Protocol) that are used to set the radio into some test conditions.

FINAL MODEL

By merging the hardware and software building block, we to finalized a typical test system model with seven fundamental modules.

Figure 3 shows the forming of the seven major modules of the architecture model.

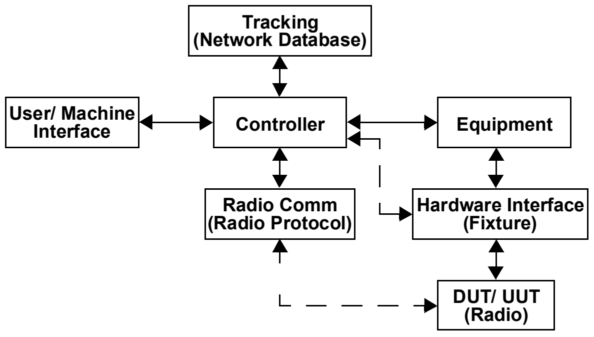
Tracking (Network Database)
User/ Machine Interface
Controller
Equipment
Radio Comm (Radio Protocol)
Hardware Interface (Fixture)
DUT/ UUT (Radio)

Figure 3 : The Complete System Model

In this model, the users/ machine interface will take the request from the users and trigger the controller. Once the controller is activated, it will trigger the tracking system for all information required by the UUT/DUT (Radio). Upon getting all information about the radio from the tracking system, the system is then ready to set up the equipment status as well as fixture control for all the tests and test conditions for the UUT. Some test programs require the system to send in commands to the radio to set the radio into certain test modes. To make this model complete, we also include the UUT/DUT (Radio) as part of the system because the radio is required to be designed for testing if it is required to be tested. If the radio did not take testing into consideration, a lot of unresolved problems may be encountered in production.

THE SYSTEM MODEL

The system architecture model from figure 3 is redrawn again as shown in figure 4.

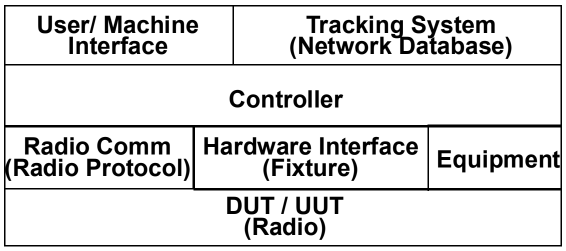
User/ Machine Interface
Tracking System (Network Database)
Controller
Radio Comm (Radio Protocol)
Hardware Interface (Fixture)
Equipment
DUT / UUT (Radio)

Figure 4 : The System Architecture Model

Let's look at each module in detail.

User/ Machine Interface: This module is defined as the interface between the user and controller. User here can be human or machine. Therefore it can be defined as the interface between the machine and the controller as well.

In general, this module will take the request from the user/machine and pass it to the controller. It also will handle the test status that is passed back from the system to the user/ machine.

Terminal display is a very important component when the operator is the user. Therefore it is very important to standardize the display information in order to reduce the learning cycle for the operator to support the system. The document "Man Machine Interface Standard" defines some of the standard requirements. [1]

As for entry, keyboard, mouse, barcode scanner and sensor are used. Barcode scanner and sensor trigger may be the direction that we are moving into as it will bring us one step closer to interface with the machine.

Currently most of the systems that we have are semi-auto. A lot of the entry points refer to keyboard or barcode scanner.

Going forward, barcode scanner and sensor may be the direction we will move into towards full automation.

Tracking System: This module is defined to handle all the tracking techniques used in tracking down the UUT/DUT (Radio) process status.

In general, we have a media that carries a track id and the track id is linked to a database that contains all information for the radio.

Currently, track id is used to track radio at board level and serial no is used to track the radio once they are programmed with serial no. Database is used to link radio information with track id and serial no. As the database plays an important role in tracking, we are currently putting up some efforts to standardize the database table. Oracle database is widely used here and by Corporate.

Barcode code (code39) and radio code-plug are used to store the track id in the radio. Going forward, we will be working together with R&D to get a standard code-plug location to store the tracking status of the radio in production.

Radio Comm: This module is defined to handle all protocols used in communicating with the radio. It also defines the types of protocol used in a radio.

Currently PConPlus is used where applicable to handle this module. In some cases where there is no support by PConPlus some direct communication techniques and standard routines are used. In this case, we try to maintain compatibility with PConPlus command.

One major problem we face today is that too many types of radio protocol are used by our radio designers. We are working with R&D to standardize the protocol used in the radio. Ideally we should have one protocol for all radios. However, there may be a cost factor prevents them from doing it. If that is true, we will work towards a maximum of 3 protocols: One for high end radio, one for medium range radio and one for low cost radio. If the standardization is achieved, it will benefit the test engineer as the test engineer spends quite a lot of engineering time due to different sets of protocal used.

Hardware Interface (Fixture) : This module is defined to handle the interface between the UUT/DUT (Radio) and the system.

In general, this module interconnect all the hardware connection among equipment, controller and UUT/DUT (Radio).

In the past, there were various types of test fixture for different types of radio. The connection between the test fixture and the test system is not compatible therefore sharing test system is almost impossible.

Now, Compro and Matris are the first steps taken to standardize the connection between the test system and the test fixture. Basically what we have here is an interface box that will connect to the test system. The test fixture will then connect to this standard test interface box. Going forward, we will try to define the standard for designing the fixture nest.

Equipment : This module is defined to handle the type of equipment used in the test system.

This is the most standardized module among the other six modules. Most of the equipment used here should have GPIB/HPIB connection. As most of the equipment bought here are from HP, standardizing does not seem to be a problem.

UUT/DUT (Radio) : This module is defined as Unit Under Test or the Radio itself. Here we are looking at the radio design and defining the requirements that allows testing feasible in a manufacturing environment. Most of the time, the test system is unable to perform reliably due to the design of the radio which did not take into account the test requirement as part of the design. In order to reduce testing problem up-front, this module plays an important role in defining the test requirement for the design group to work on.

We have draw out some guidelines for R&D as part of their requirement. These guidelines also include the requirement for board level testing. Some of the examples like the requirement for the test pad should be at one side, and also the size of the test pad should be provided by the board, etc.

Controller : This module is defined as the central control unit of the entire system. It plays a major part in linking up the rest of the module and drives the entire system. It is the most complex module and also the most un-standardized module among the rest of the six modules.

The controller plays an important part, interlinking all the rest of the modules and drives the entire system to perform the testing correctly.

Due to the complexity of this module it is sub-divided into 5 sub-modules:

1. Hardware
2. O/S
3. Development Platform
4. Core
5. Application

Figure 5 shows the sub-module for the controller module. There are 5 sub-module in this module.

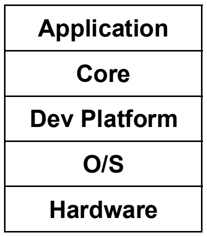

Figure 5 : Break down sub-module for controller

Let's look into each of this sub-module in detail.

Controller (Hardware) : This sub-module defines the type of computer and the hardware requirement required as a test controller.

The two major computer used here are the HP300/700 series and PC.

For HP300/700 series computer, requirements are two serial port, one parallel port, one lan card, one GPIB/HPIB card.

As for PC, requirement is a minimum of four expansion slot ISA/PCI. GPIB in one slot, 3Com Combo lan card in one slot, one optional digital I/O card and a spare slot. The PC should also have two serial port and one parallel port.

Controller (O/S) : This sub-module defines the type of operating system used in the controller.

In the past, we actually supported a lot of O/S, HP-BASIC, SRM, UNIX, DOS/Windows3.11. We are in the process of phasing out the obsolete O/S but will continue to support the major O/S that is UNIX and Window NT.

As we will moving towards using UNIX to set up complex test system and using NT to support the less complex type of system.

Controller (Development Platform) : This sub-module is defined as the development environment, compiler or platform that is used to develop the test program.

In the past, HP-BASIC, SRM PASCAL, C are used as the development platform. Going forward, C/C++ will be used as the development platform. HP-VEE may be another option for the future as it is still under evaluation.

Controller (Core) : This sub-module is the skeleton or framework of the entire test program. There are three part in this sub-module.

1. Core Program and Core Library
2. Core File Structure
3. Core Database

Core Program and Core Library provide the developer the framework and useful library function to develop the test program. *Core File Structure* provide all the necessary information to drive the system and also setting up different configurations for the system. *Core database* helps to provide tracking information for a tested radio.

In the past we have supported quite a number of cores (Big Foot, RUST, MURS, PATS etc). We are working on reducing the number of cores required to be supported. Going forward we hope to have only PATS, MATRIX and COMPROG in use.

Controller (Application) : This sub-module is the actual test program that is required by the system to test the radio. The application will be developed by making use of the core framework and also the core library. Test setting and system setting of the application will be stored in the file as define in the core file structure. Any tracking information and datalog in the database will make use of the core database routine which is developed in the core.

At present, most of the new application programs are develop in PATS, MATRIS and COMPRO.

FUTURE PLANS & RECOMMENDATION

The system model here may not be the best, however, it is a good start to break down the test system into modules to allow us to analyze our test system in a more structure way. Future recommendation is to gather engineers from different factories to enhance or to re-design and agree on the designed model. A well define system model will definitely increase the re-usability of software and hardware efforts and therefore reduce development cycle time.

CONCLUSION

Having the system model defined for the test system actually helps us to break down the system into smaller modules and allow us to analyze and make improvement to the test system. It also helps us to identify which part of the system need improvement and which part is already performing well. Engineering time can therefore be spent in a more efficient way thereby reducing cycle time and optimizing engineering effort.

REFERENCE

[1] A.T. Tan, “Man Machine Interface Standard”., Motorola Penang TPD Number TE-SPD-012. Dec 1, 1994.