AN AUTOMATED TESTING APPROACH FOR PXI CHASSIS SOFTWARE DRIVER

NG WAI SHYAN

UNIVERSITI SAINS MALAYSIA

2017

AN AUTOMATED TESTING APPROACH FOR PXI CHASSIS SOFTWARE DRIVER

by

NG WAI SHYAN

A Dissertation submitted for partial fulfillment of the requirement for the degree of Master of Science (Electronic Systems Design Engineering)

July 2017

ACKNOWLEDGMENTS

I would like to thank you to my supervisor Dr. Mohamad Khairi Ishak for providing me his guidance and helpful advice over this time.

I would like to acknowledge with gratitude, the support and love to my wife and my parents for their support and encouragement over the years, they all keep me going where I am today. Especially to my wife for taking care the family when I am busy with the thesis.

My thanks and appreciation to my Keysight colleagues for knowledge sharing and lab support to complete my work. To Keysight management thanks for giving the opportunity to further my studies.

Lastly, I would like to express my deepest gratitude to the Electrical and Electronic Engineering Department of Universiti Sains Malaysia (USM) for providing me the chance to undertake this remarkable Master in Science Course.

PENDEKATAN MENGUJI KUALITI PEMACU PERISIAN PXI CESI

ABSTRAK

Cesi PXI merupakan suatu produk yang boleh beroperasi dengan pelbagai vendor peranti dan boleh disambungkan dengan pelbagai jenis cesi, modul dan komputer. Satu siri ujian yang rumit perlu dijalankan untuk memastikan pemandu peranti dapat berfungsi dalam konfigurasi yang tertentu. Proses ujian untuk menguji perisian pemacu PXI sangat kompleks kerana ia perlu melibatkan berbilang konfigurasi untuk satu pemandu. Ujian ini menghadapi masalah kerana kebanyakan sistem ujian automatik masa kini hanya melaksanakan ujian dengan tanpa menyemak sama ada kelengkapannya telah dipenuhi ataupun tidak. Tambahan pula, trend ujian automatik sekarang hanya dapat membenarkan ujian dilaksanakan pada satu konfugursi sahaja. Justeru, tempoh masa yang panjang diperlukan untuk menyempurnakan ujian tersebut. Untuk memudahkan dan memendekkan tempoh masa ujian untuk cesi PXI pemandu IVI dalam berbilang sistem ujian, satu alat perisian automatik telah diciptakan. Dalam usaha mencipta alat ini, konsep pelayan-pelanggan diguna pakai. Konsep pelayan-pelanggan mempunyai kelebihan yang dapat memusatkan ujian apabila berbilang sistem ujian perlu dijalankan. Pelanggan-perisian akan mula melaksanakan tugasnya sebaik sahaja memasuki sistem operasi sepenuhnya, sistem ujian akan menyambung ke pelayan dan menunggu tindakan selanjutnya daripada pelayan. Apabila pelayan mengesan sambungan daripada pelanggan, ia akan memeriksa secara automatik dan menetapkan kelengkapannya. Jika pelanggan memenuhi kelengkapan ujian suite, ia akan mula melaksanakan ujian tersebut. Ringkasan semua hasil ujian , pelanggan akan memaklumkan kepada pelayan. Keputusan menunjukkan ujian yang dijalankan menggunakan perisian automatik dapat mengurangkan tempoh ujian sebanyak purata 17.1% dan keputusan ujian juga mencadangkan penurunan masa ujian juga berlaku jika komputer berprestasi tinggi digunakan.

AN AUTOMATED TESTING APPROACH FOR PXI CHASSIS SOFTWARE DRIVER

ABSTRACT

PXI chassis is a multi-vendor interoperable device, it can interconnect with many chassis, module and computer type. To make sure the device driver is able to function in specific configuration it must go through a series of testing. The complexity of PXI software testing has increased when it need to cover multiple configuration for single driver. Majority of the automated test system will execute test without have a mechanism to verify the test environment. The current trend of automated software test only execute on single test configuration, to improve the PXI chassis IVI driver test duration a test software with the capability of execute multi test configuration is developed. In order to develop this tool, a server-client concept is adopted. The advantage of server-client is to centralize the testing when multiple test system perform test at same time. The software tool client will start once completely boot in to operating system, the test system will connect to server and wait for further action from server. When server detected incoming client connection it will automatically verify and fix the testing environment, if the client fulfilled the test suite requirement it will start execute test. All clients test summary result will be feedbacked to server. The results show an average of 17.1% test duration reduction on the planned test configuration when the automated software tool applied on testing. Besides that the results suggest that execute the test on higher controller hardware performance can reduce the test duration as well.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANSI-C	American National Standard Institute C programming language
COM	Component Object Model
COTS	Commercial Off-The Shelf
CPU	Central Processing Unit
DP	Display Port
DLL	Dynamic-link library
HID	Human Interface Device
IVI	Interchangeable Virtual Instruments
LAN	Local-Area Network
MMIO	Memory-mapped I/O
OS	Operating System
PPM	Parts Per Million
PXI	Peripheral Component Interconnect extensions for instrumentation
PXIe	Peripheral Component Interconnect extensions
SUT	Software Under Test

CHAPTER 1

INTRODUCTION

1.1 Background

A hardware device operates on an operating system need a software driver to configure the device so that the operating system able to recognize the device properly. The device driver provides libraries of configuration parameter and kernel module during the operating system startup. A device driver not only limited to provide a correct configuration to operating system it also safely exposure function, support application and help reference document for end user.

To produce and make sure a device driver in high-quality state the software must go through a comprehensive testing phase. Test and measurement equipment driver provides user application programming and software application as part of driver package. This piece of software support different platform and hardware configuration because of the flexibility and complexity of the instrument device driver it is important the testing phase to have an essential multiple testing methods and tools during this phase.

Peripheral Component Interconnect extensions for instrumentation (PXI) is a PC-based architecture that profoundly utilized PCI and PCIe electrical-bus features. Most of the industrial test, manufacturing test, military, aerospace, automotive and machine monitoring is using this standard since the platform is scalable and meet most of the solution requirement. The PXI Systems Alliance promotes and maintains it PXI standard include software and hardware specification.

The device driver package for PXI is growing more complex the number of modern environment and operating system platform to support the PXI is increasing. All this changes is to improve usability and stability of the devices. With all this changes the software testing become more challenging since the area of coverage keep increasing and software quality need to be achieved. Each of the feature provided to end user have its own criteria to get the test passed. All the driver feature must go through a specific testing with different configuration and platform.

PXI interconnected uplink and downlink on each test system is not fixed, this adding more complex to the testing when hardware configurations affect the actual driver and application function. The hardware itself have upgradable firmware and to make testing more complex this firmware will directly affect the Interchangeable Virtual Instruments (IVI) driver work since the IVI driver will query information place by the FPGA on the same Memory-mapped I/O (MMIO)

1.2 Problem Statements

Most of the papers in the past discussed about the strategies and test suite generation of software testing. The previous work lacks of mentioning the hardware and software configuration affect the testing strategies. The SUT has become complex because of the built-in firmware that allow to upgraded or changed by the user. When planning a test suites those critical parameter that might affect the functionality need to include.

Software developer and end user affected if an undiscovered defect hidden. A software defect cost less to fix in the beginning of the software development stage rather than near end release stage. If a defect found after release for consumption, it will cause the organization reputation damage in term of product quality. It is important the software is tested with all supported configurations in early stage to reduce the cost and reputation damage to the organization.

The task to test the correctness of IVI driver and validate the product design based on allowed hardware configuration need to be systematic. The traditional testing method is not efficient and not able to sustain the increasing of test coverage. An automated method specific on PXI is needed to speed up the existing testing and provide a quantitative measurement based on generated test suite.

1.3 Research Objectives

The objectives of this research project are as following;

1) To reduce test time by develop an automated test execution for PXI chassis IVI driver testing.
2) To run test suite when test system meets the test requirement.
3) To investigate and compare the testing duration between different configuration.

1.4 Research Scope

The scope of this work will cover the PXI chassis driver testing. This testing activity is a part of the software qualification prior to package release. The test execution is based on Windows operating system software driver with Keysight PXI chassis with embedded and external controller. The method implement hope to improve testing time by easing the test flow.

1.5 Research Contribution

This work is important because software testing activity should start as soon as the software development started. The software tool will keep developer actively test their work and quantify the software quality to a specific configuration. This software tool can actively deploy test suite to any idle test system without much manual interaction. An automated test system is developed to facilitate the testing on PXI chassis IVI driver.

1.6 Thesis Outline

Considering the detail, this work contains overall five chapters with the following detail discussed as below. Chapter 1 consist of introduction to the background on this thesis, objective of the thesis had been discussed and research contribution in this thesis. Following is Chapter 2, software testing on unit and functional test, PXI requirement, introduction to IVI driver, software test and quality mean, test automation, and PXI controller and chassis for this work setup. Chapter 3 will explain the concept used in developing automated software application in this thesis, detail on how server and client operation flows, method to invoke a function, instrument for the testing including external and embedded controller configuration, fixture used in testing, software and other test file required for the tests to execute and final test system setup. The result and discussion are discussed in Chapter 4, covering automated test execution result, two driver tests include KtMPxiChassis and KtMTrig IVI execution, manual test execution, comparison between manual and automated, Win 10 and WES 7 result comparison and advantage of the developed software tool. The last chapter on this thesis is Chapter 5 discussion on conclusion.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter will review the past research done mainly on software testing, IVI and PXI. The highlighted subtopic includes Unit and Functional testing, PCI eXtentions for Instrumentation (PXI) requirement, Interchangeable Virtual Instruments (IVI) requirement, software test and quality, test automation, controller and chassis for PXI.

2.2 Unit and Functional Tests

Unit testing is a testing on an individual unit code, this includes method and properties in a class with all it dependencies mocked up. Each unit test performed to validate the method produces an expected output with a known input the input can be in the method bound or out of the bound. Unit test provides a fundamental health of the device driver code and gives an indication that the code is having the correct logic. Unit testing is an effective technique to improve software quality, flexibility and time-to-market. The main concept is that each piece of code needs a test case. It is preferred developer himself develop the unit test but most of the time manually written unit is not sensible as it is too costly since the unit tests need maintained. Some developers do not have enough experience to write a good unit test, the outcome of the unit test does not improve the software quality. As a result, lack of experience and to keep the effort low, developer will seek for automated test generation tools to complete the unit testing [1].

Functional test is a testing on the sub-system functionality, it will test on methods or use cases function, most of the test will set and get feedback from the device. Functional test is focusing on feature offered to end users where the intended functions are created and used at system level. It is almost impossible to create functional test that cover each of the requirements and interactions among the test cases therefore to improve the test quality an automatically generate functional test cases that cover entire requirement in a complex software. The functional test case generation needs to be effective and efficient. The automated functional software tests applied need to be maintainable to support feature improvement on the test cases requirements. The automated functional test needs to adapt software development life-cycle models so that it will have a proper development step. The automated software test development infrastructure is derived by the product software's feature specifications so that it perfectly supports the need of the SUT [2].

2.3 PXI requirement

PXI is a short form for PCI extensions for instrument where the technology inheriting and makes use the benefit of PCI architecture thus the architecture, performance, industry adoption and Commercial Off-The Shelf (COTS) are compliant with PCI. By using the new Peripheral Component Interconnect Express(PCIe) standard with a shared switch allows each device its own direct access to the bus. PCIe provides each device with dedicated data pipeline and data sent through serially in packets using pairs of transmit and receive signal lanes. Multiple lanes can be grouped together to increase bandwidth. In PXI Hardware Specification all chassis configuration contains a system controller to setup a basic system. A controller can be embedded or external controller as a system where user can define instructions, execute application and store necessary instructions or data.

The PXI specification is to allow vendor device to be interoperable. PXI instrument is widely adopted into test instrument thus instrument designer is encouraged to follow the specification. The benefit of following the specification is that vendor's device will able to integrate with other vendor device. The interoperability can be achieved by adopting the same technology and specification to reduce issues during the design and implementation. The specification also defines the frameworks and incorporates the existing test and measurement standards include Microsoft Windows and IVI Foundation as the main driver interface standard [3].

2.4 Interchangeable Virtual Instruments (IVI) driver

IVI Foundation is an open consortium that guard IVI standard and currently it has three interface technologies: Component Object Model (COM), American National Standard Institute C programming language (ANSI-C) and .NET framework. Instrument developer will need to conform to one or more of the standard technologies. Developer will choose which architectures to support based on the needs of customer.

IVI-COM instrument driver use COM technology. IVI-COM has a COM API and distributed on Windows to users in the form of a Win32-DLL. The current measurement industrial is using IVI-COM to design application by using Visual C++, LabWindows/CVI, LabVIEW and Agilent VEE support calling into COM object. IVI-COM is interchangeable with an IVI-COM class-compliant specific driver and to make use of this it need to make a call to the IVI-COM Session Factory where this software component is defined by the IVI Foundation.

IVI-C is a driver with a C API distributed on Windows in the form of a Win32-DLL. The IVI-C also shares the same user as mentioned in IVI-COM. The IVI-NET driver is a .NET API and the driver is export as .NET API by distribute on the form of a .NET assembly. Common used application development environment such as C# and VB.NET calling into .NET assembly.

Figure 2-1 shows the application specific software layer interacting with instrument modules hardware need to go through a series of software layer. Most of the modules offer more than one software combination options to choose when communicating with the devices. Is important to choose which combination is suitable when developing an application. Users may access through different layers and combinations to communicate with the hardware instrument. Software development environments is a development platform provide by the hardware vendor itself or other third party vendor when writing a test program. Besides providing a layer to accessing instrument drivers this software development application provides an abstraction to engineers to manage linking with the instrument driver. IVI driver normally is supplied by PXI module manufacturers it provides a programming interface to the specific device module. VISA is a communication protocol that talks to many interfaces including GPIB, RS-232, USB, Ethernet and PXI. Developer can access this layer to controller the device but it requires a certain set of commands that is usually not provided directly by device manufacturer. The best method to access the device module is by using the IVI driver provided by manufacturer. The lowest layer before the instruction reach the device it must go through the specialized kernel driver, this layer provides an interface between instrument and the PCIe bus [4].

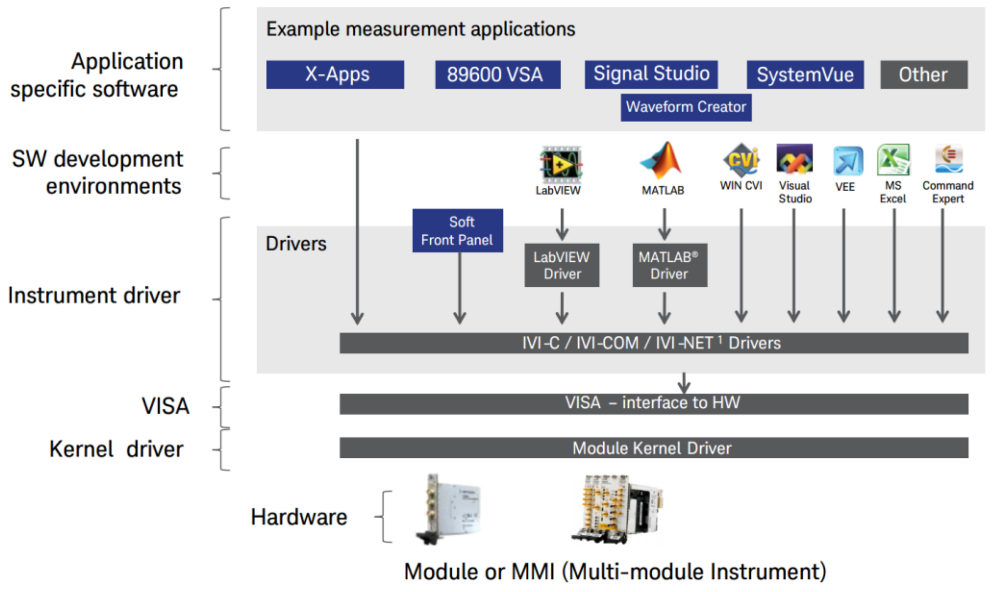


Figure 2-1 Modular system software architecture [4]

Figure 2-2 is a Venn diagram showing the common and specific driver that developers can choose the type of driver to match the applications needs. IVI driver is an instrument driver that inherent capabilities, it can directly communicate with instrument hardware or act as a pass-through layer to other IVI drivers. Developers need to choose an IVI driver as a specific driver or an IVI class driver. IVI specific driver is a driver that controls a specific instrument and directly communicates with the instrument hardware it can be class-compliant with other instrument interchangeability or not compliant with defined IVI class specification since is a custom API for specialized instrument. IVI class driver is interchange instrument since it uses IVI class-compliant specific drivers. This class driver exports an API complying with defines IVI class specifications. IVI class driver are interchangeable when IVI-C and IVI-COM class-compliant specific drivers are used [5].

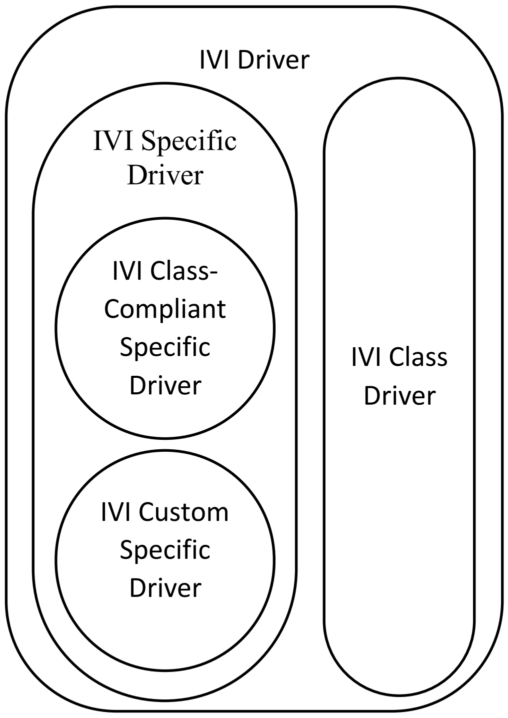

IVI Driver
IVI Specific Driver
IVI Class-Compliant Specific Driver
IVI Custom Specific Driver
IVI Class Driver

Figure 2-2 Relationship between driver types

IVI Foundation promotes the programming standard for test instrument. It is important that a vendor build on an existing industry standard that allows interchanging instruments and provides and maintains high-performance software [6].

In most tests and measurement equipment there is a hardware and software interface to control, set and query. A complex measurement needs a series of setting, feedback from equipment before the final measurement can be achieve. A large test measurement system requires a mechanism to manage and organize those data. The current trend is automating the measurement process by using programming and user interface (UI). This application is based on IVI driver since it is a widely-used interface. Currently there are 3 flavors of IVI driver available IVI-C and IVI-COM and IVI.NET is the latest interface when make use of .NET technologies. The IVI.NET provides simpler source code thus provide IVI.NET easier to develop [7].

2.5 Software Test and Quality

Nowadays software products are largely used by industry and domestic in various aspect to perform routine tasks. The software that applied today is become large and increasing in complexity. All this is because the computer processing power is continuing to increase so as other hardware component to increase software capability to handle more tasks. The increasing of software complexity gained attention how to ensure the software quality assurance. Many parties have tried to introduce a complete quality assurance to manage the whole software testing process. Thus, the quality of testing can be visualized and standardized through measurement. Figure 2-3 shows an effective flow for testing management, where the flow includes measuring the testing process and most important what is the control process as a close loop check [8].

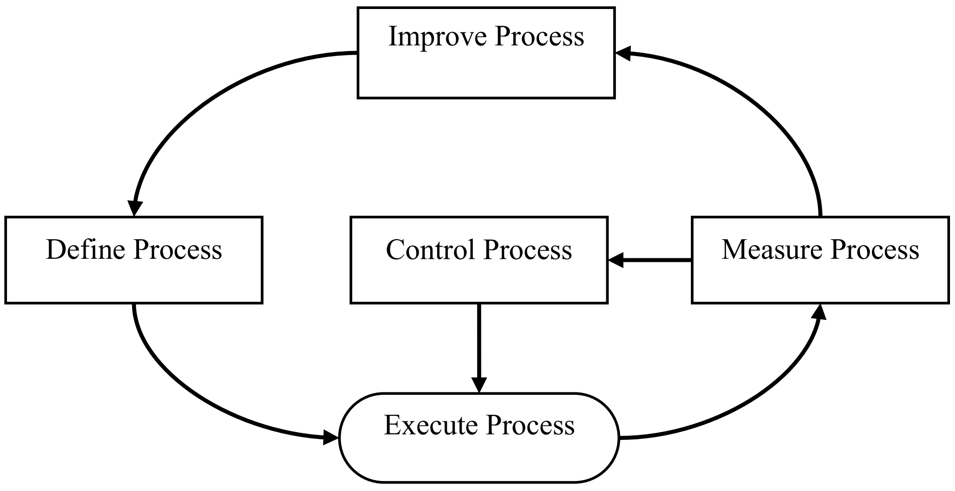


Figure 2-3 Software Process Management Flow

The software quality definition from the user is expectations is the software system able to perform useful functions as specified, this include performing right functions as specified that might fit the user's needs and performing the specified functions correctly over repeated use or over a long period i.e. the system performs its functions reliably. From the programmer's perspective, the software quality mean the system software is conformed to the product specification.

ISO-9126 is a widely-used quality framework and is define well each of the quality characteristics. There are 6 main characteristics defined in ISO-9126 which are functionality, reliability, usability, efficiency, maintainability and portability [9].

Software testing is to evaluate the software product capability whether it meets the expected results. Software testing normally consists of test case planned in test flow according to the design requirement. Test strategies give an overview guide to the testing flow on how the test activities execute more efficiently and effectively. The common test strategies practice is reviewing the testing requirement and flow to reduce error testing planning. Software testing usually starts with basic functions testing such as unit test, followed by integration, system and accepting testing. This early stage testing is conducted by the developer with each of the test case has it very technique and specific purpose. If the testing finds any abnormality, then the developer needs to initiate debugging activity to simplify the testing when there is an issue found and it need more investigation to rectify the issues. The software quality of a product will be improved after going through a planned test. Early detection of an error can save cost and time of the project. It is important the software testing should be applied to all stages in the development cycle [10].

The importance of software testing should concentrate on user-focus it is important the usability of the driver. Previous paper also stress that IVI drive need a testing and validation strategy to make sure the driver is conformant to IVI specifications and are easy to use. Some of the common and important item to check are API interfaces, properties and methods, simulation mode, usable development environment, installation, uninstallation and repair [11].

To achieve a level of reliability and confidence on software is through testing. Statistical sampling test case is a method way to assess this process in unbiased and efficient in convention way this usually requires a large number of test cases to reach statically confident. An adaptive sampling test case is much more efficient since it uses more reliable feedback from previous test results, this method is useful when testing on subsequence cycle or release. The test cases allocation depends on the software feature further critical analysis need to consider are test historical/design data or optimized based on previous test suite model [12].

The current soft panel design trend is using WPF which is a subset of .NET framework and must use a .NET Framework programming language such as C# and Visual Basic and both of these languages are classified as a managed language. .Net Framework compiled into intermediate language while later compile into machine language before running in common language runtime (CLR). Some of the low-level VISA library is written using unmanaged C language when use on soft panel design it need a wrapper to import the Dynamic Link Library (Dll) [13].

The current software trend has grown to a stage of high complexity and the quality assurance of the software increased in demand since some of the software will be used by critical and crucial applications. There are a few challenges to make sure the software delivered is in a good quality such as testing a large system with a diversity of individual components as a whole, it is hard to organize all the running tests since the software is distributed and difficult to justify when testing each component since a single component usually needs other dependencies [14].

2.6 Test Automation

The automated testing is a test execution with test tools on computer without human intervention and manual test defined as test case execution without any tool [15]. Previous paper also mentioned that a particular testing should start with low level test and moving upward followed by integration testing and finally system testing. A comprehensive set of testing will can reduce development cost and produce end user is satisfaction. Regression test is a way that a software test case is retained and reused for other test. The regression tester is important to make sure the continuously software improvement does not break or affect other existing functionalities [15].

Software tools are used to automate some of the manual tasks because manual testing is tedious, error-prone and resource consuming. Theoretically there is no fully automated testing possible since there are some activities, such as result checking and some physical configuration are needed. Nonetheless, some degree of automation is needed to sustain the test and individual activities. Commercial software tools are used to reduce the test deployment time. However, some large organizations can develop their own tools. The main objective is to reduce testing time and increase productivity without involving human on tedious and repetitive tasks that is possibly done by using software tools. In some cases, planning for test automation is not simple since every product is different and regular commercial software tools might not fully support the need on specific product test automation. Some of the points to consider when planning test automation include specific needs, potential test for automation, consideration of existing testing tools, cost to build test automation tools, user training and impact on schedule of project [9].

The automated test requirements are derived from the software feature specifications. The test cases are then written according to specific functional part of the SUT. In order to make sure every test cases are applicable each test feature should have a precise description on purpose, input processing, outputs and method to handle error. The feature offered in SUT should be testable and some of the features are dependent or aggregation relation between them.

2.7 Controller and chassis for PXI

PXI embedded controller consists of complicated interactive interface buses which has raised the concern on the software quality assurance. Embedded system is ready with network and with this feature, the testing method has changed to networked embedded software testing platform. With network connected the controller test data and a terminal server can be integrated as a framework to fulfill more complex test systems [16].

Testing and measurement systems have incorporated field programmable gate arrays (FPGAs) to instrument and allowing user to access and customize the firmware or upgrade the firmware version. The main concern of this FPGA-enabled instrument is the API such as IVI driver need to function without breaking the software API compatibility. The firmware of the instrument becomes an important component to consider when planning a test suits. Testing the IVI driver become more complex and it need a different firmware version or setting to verify the same software driver [17].

PXI system is a flexible platform that can be derived from one primary host and connected with one or more sub-systems with processing devices. For example, two PXI system can be communicated through the endpoints by a non-transparent bridge that allows PCI traffic between memory spaces [18].

Keysight has PXI product since the first chassis M9018A in 2011. M9018A is PXIe chassis with a total of 18 slots including the PXIe controller and timing module slots. M9018A has 16 PXIe hybrid slots that can accept PXIe, 32-bit CompactPCI and Hybrid Slot-Compatible Module, this chassis can deliver 867.5 W of DC power. M9018A is compatible with PCIe Gen2 speed and it provides twelve PXIe peripheral slots of x4 that provide maximum data bandwidth of 2 GB/s and four PXIe x8 links providing a maximum data bandwidth of 4 GB/s. The system slot has a PCIe switch fabric that can configure as 2-link (2x8) or 4-link (4x4) configuration. The chassis backplane speeds capable up to 4 GB/s data rate speed with external controller to PXIe module slots configuration. The latest model of M9018A is M9018B that offer the same capability with an addition front panel trigger ports [19].

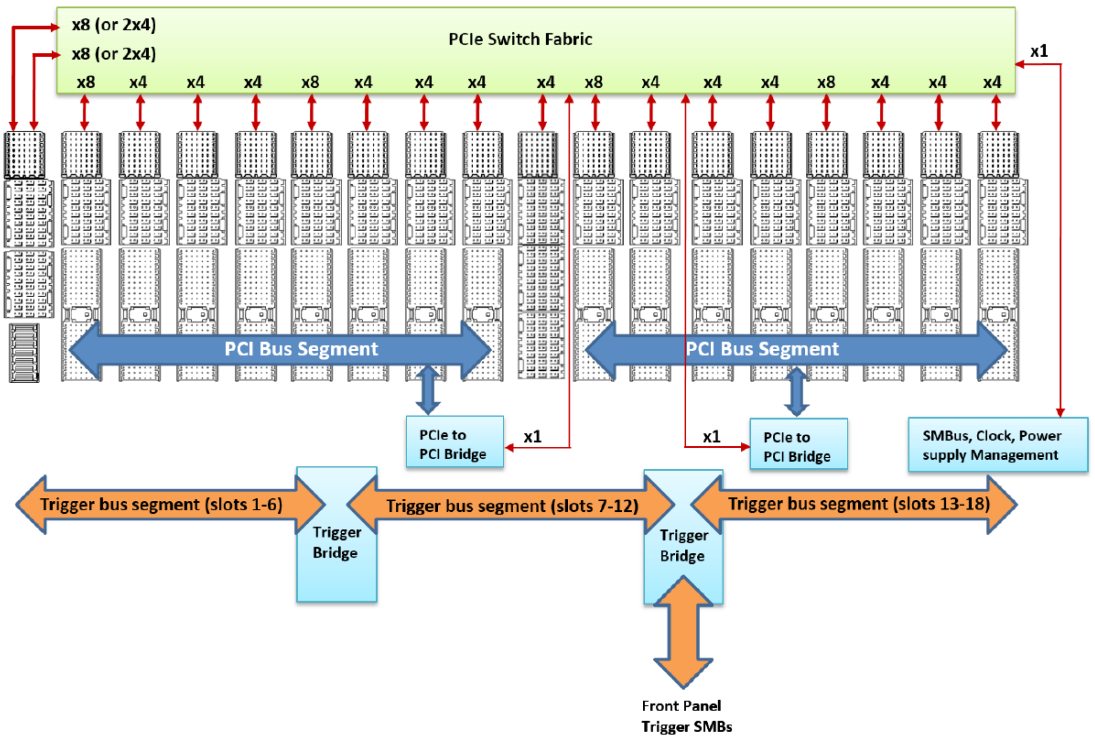

x8 (or 2x4)
x8 (or 2x4)
PCIe Switch Fabric
x1
x8 x4 x4 x4 x8 x4 x4 x4 x4 x8 x4 x4 x4 x8 x4 x4 x4
PCI Bus Segment
PCI Bus Segment
PCIe to PCI Bridge
x1
x1
PCIe to PCI Bridge
SMBus, Clock, Power supply Management
Trigger bus segment (slots 1-6)
Trigger Bridge
Trigger bus segment (slots 7-12)
Trigger Bridge
Trigger bus segment (slots 13-18)
Front Panel Trigger SMBs

Figure 2-4 M9018B backplane configuration and triggering system [19]

Keysight also provides ultra-high performance PCIe switch fabric that operates at Gen 3 speeds, 18 slots for PXIe product performance through its new chassis M9019A. It has a two-link system slot (x8, x16) that can generate maximum data bandwidth of 24 GB/s by utilizing it 24 PCIe lanes. When this chassis used with system module M9023A and M9049A PCIe adapter on external PC, it can generate data bandwidth of 16 GB/s between slots [19].

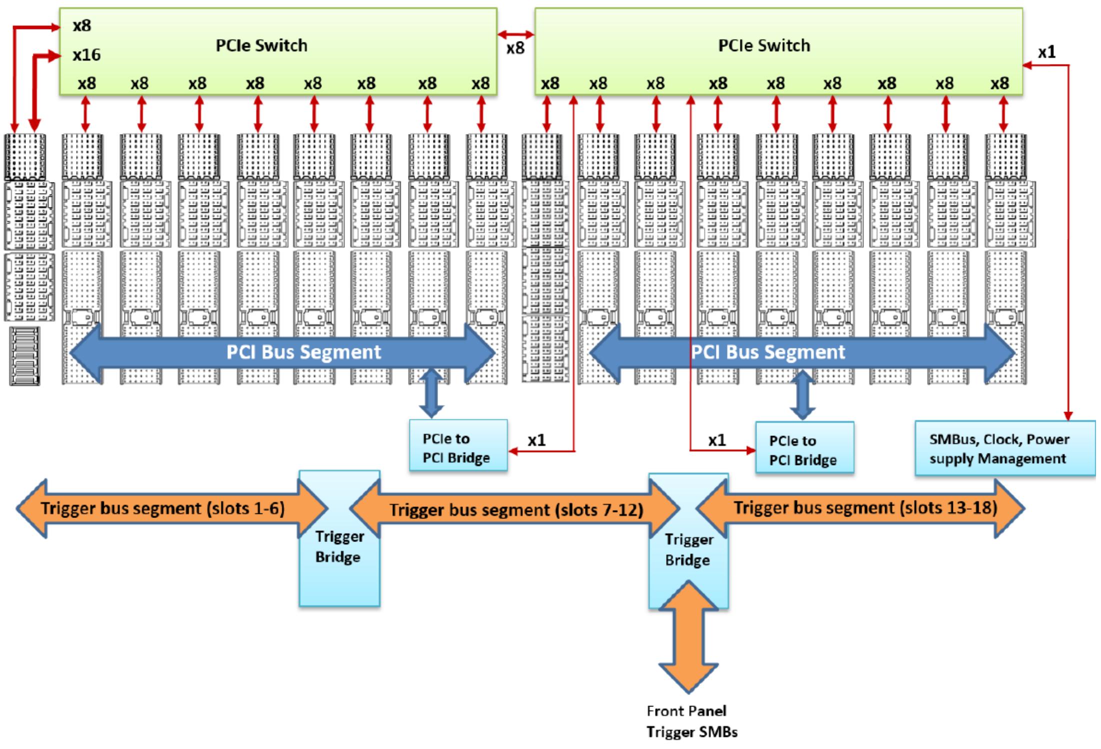


Figure 2-5 M9019A backplane configuration and triggering system [19]

Another PXIe chassis that can deliver Gen 3 speed form Keysight is M9010A that comes with 10 slots. The good thing from this model is that it has the same performance while in a smaller form-factor. The chassis has eight PXIe hybrid slots, one PXIe timing slot and one PXIe system slot that has the capability to deliver up to 24 GB/s data bandwidths. This chassis delivers up to 16GB/s data bandwidth between PC and the PXIe chassis [20].

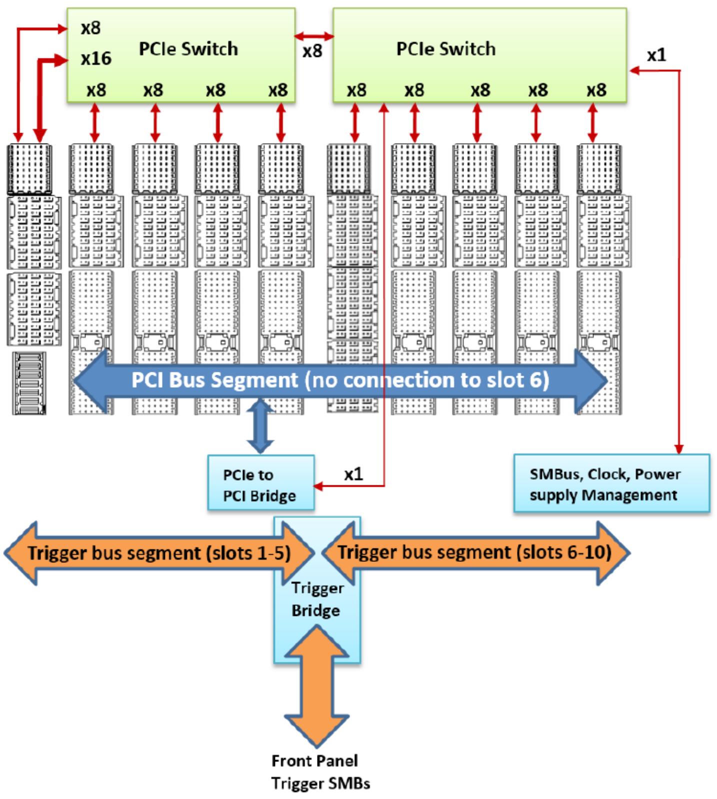

x8
x16
PCIe Switch
x8
PCIe Switch
x1
x8
x8
x8
x8
x8
x8
x8
x8
x8
PCI Bus Segment (no connection to slot 6)
PCIe to PCI Bridge
x1
SMBus, Clock, Power supply Management
Trigger bus segment (slots 1-5)
Trigger bus segment (slots 6-10)
Trigger Bridge
Front Panel Trigger SMBs

Figure 2-6 M9010A backplane configuration and triggering system [20]

Keysight publishes its tested computer this which includes the capability of personal computer and embedded PC controller with PXI/AXIe [21]. This is important so that the end user never bought PC which did not fully function on PXI. Most of the tested computers have the capability to do enumeration on many PCI devices. The computer BIOS allows and passes the number of PCI devices detected to the OS so that the following enumeration done by OS and processor are correctly mapped into the address space of the computer system. If the enumeration process failed, the OS or application will never get to communicate with the PXI module since most of the read/write instructions and device configurations need to go through a specific address space that the device is mapped.

2.8 Summary

There is a vast amount of literature on software quality and test automation it gives a clear definition on software quality, testing process and software automation but there has been lack of work done in IVI driver test on PXI instrument. This typical software and hardware need a special attention due to its complex operation environment. PXI devices have the interoperability function and IVI provides the access to the hardware by shared library or custom functions. The process to verify this software driver needs to be flexible yet it needs some degree to reduce the hardware change over test duration.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This section will reveal the software tool for automating the test for PXI chassis driver in a systematic way. The sub topics discussed below include structure of the automated software tool function and layout, server-client concept, the server and client application process flow, the idea of invoking a function, testing configuration, test fixture, file setup and the last sub section is the final hardware setup.

3.2 Overall method

The method introduces in this work start to take place when SUT hardware is done setup. Figure 3-1 shows Test Executive Server and Test Executive Client started communicate. Before the test can be start, the test system must get to know the client hardware and software setup information. After the test system done identify the client configuration and make necessary installation on the client, the server will start load test into client.

The client will start the test after it received command from the server in sequence order. In this work the PXI Family Chassis Driver consist of two categories of driver, KtMPxiChassis and KtMTrig. Each driver categories will have 12 set of tests that are combination of IVI-C, IVI-NET, x86 and x64 platform.

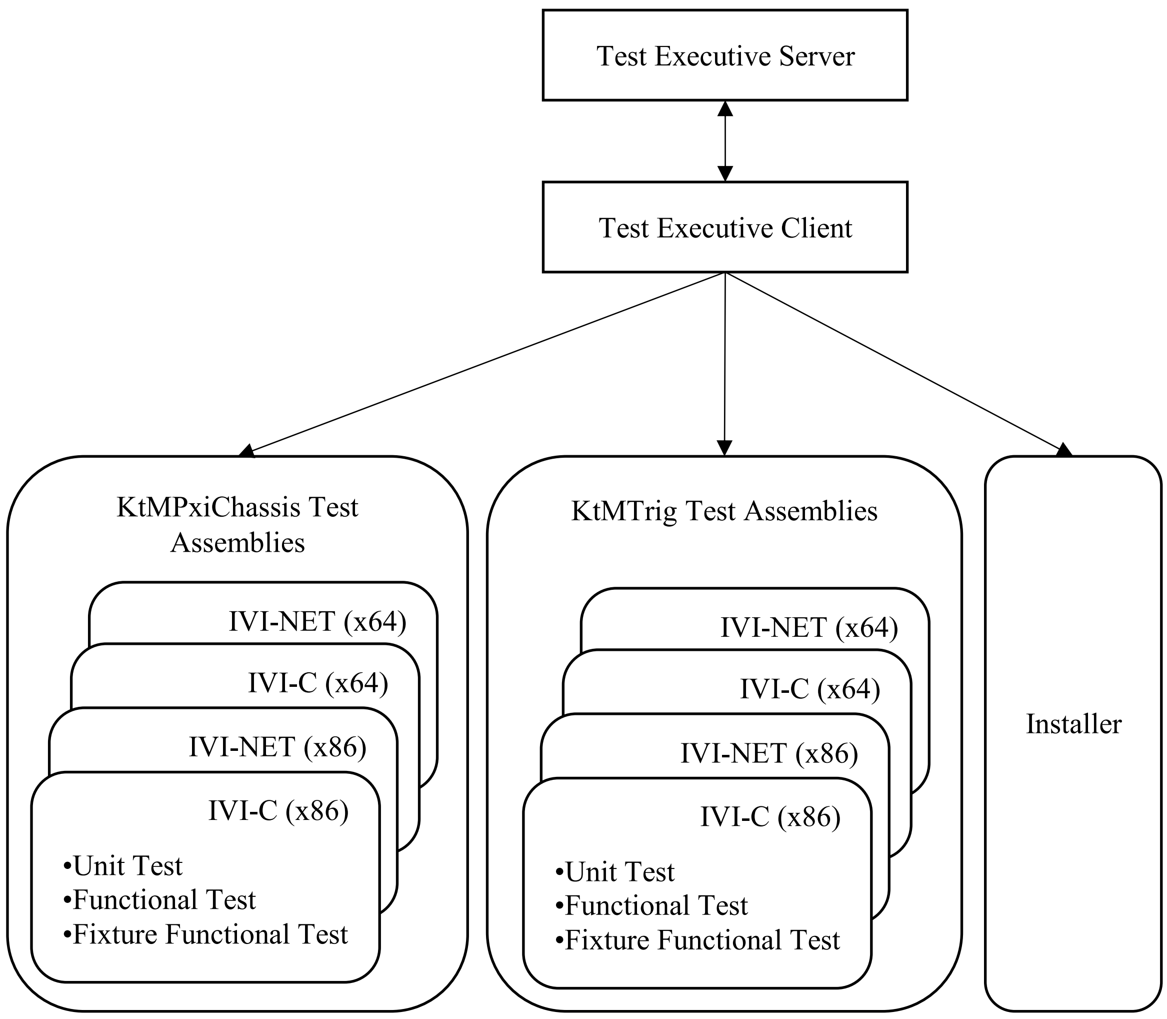


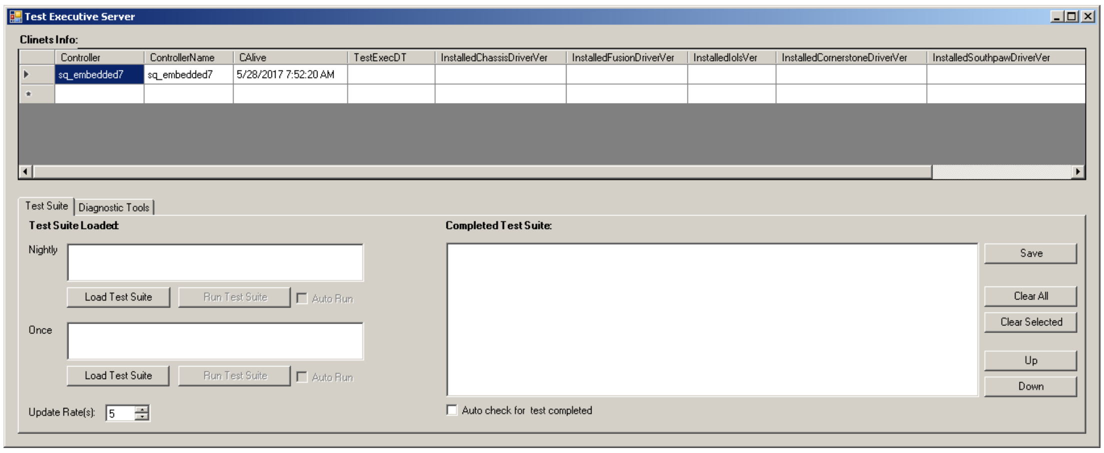
Figure 3-1 Overall test flow

3.3 Automated software tool application

The automated software tool developed in this thesis is to enhance the PXI chassis IVI driver testing. It helps reduce the total testing duration and support easy test configuration and deployment. This automated software tool is divided into two parts, namely Test Executive Server and Test Executive Client. Consider in a normal dynamic Internet Protocol (IP) environment scenario, the server application will be executing first to allow the server IP recorded in a shared file, the server application will wait for the client to come alive. In a static IP address environment, either server or client can be executing without sequence because the IP on server is recorded and it will never change.

There are two option to start the Test Executive Client. The first option is, in a dynamic IP address system the server application will be execute first before the client application get execute, while second option is, the client connects with the server by providing the server IP address as argument when executing client application. The server application has no issues if the client connection suddenly drops-out or trying to restart the client, the server will automatically reconnect the client once it detects the incoming socket connection.

Figure 3-2 presents the Test Executive Server in Test Suites tab. The server application has a data grid view on top section to shows the active clients currently connected. The data grid view shows the detail of each client parameter, this include the last record client send alive message, record executed test suite, installed chassis driver version, Fusion version, IOLS version and other fixture module driver version. The client information will be added up in the data grid view when a new active client is connected to the server. The server application will delete inactive client in the data grid view when it is not able to detect client acknowledgement message. The bottom of the UI application shows a multi tab control window. The first tab consists of test suite function where the main function of this tab is to load and start the test suite, this function can be seen on the bottom-left. The bottom-right list box will show the summary test result returned from client application when running the test.

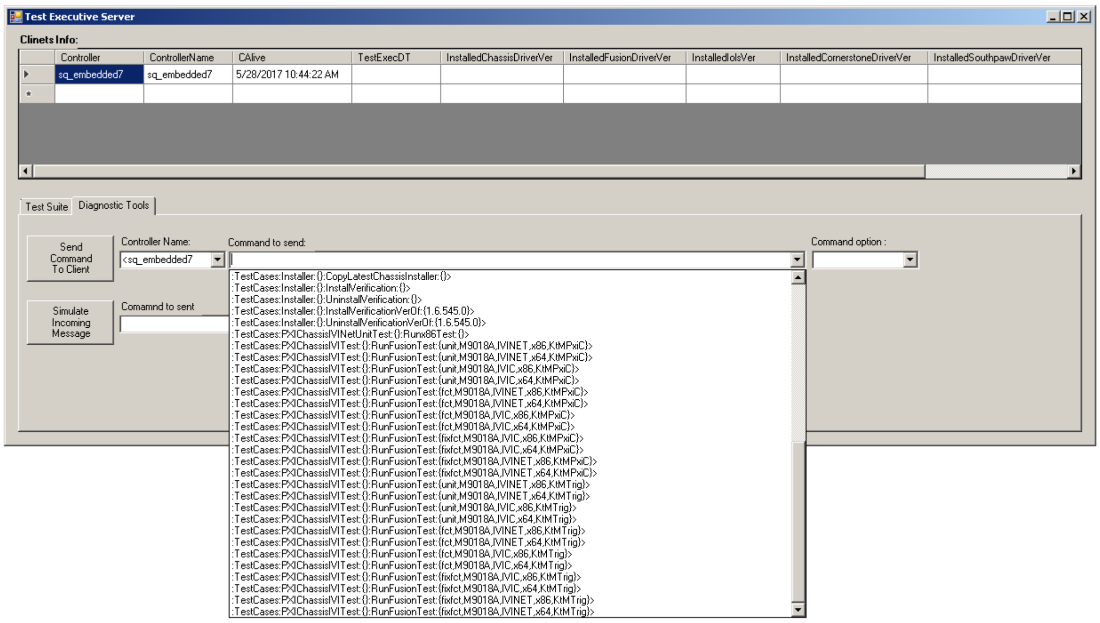


Controller	ControllerName	CAlive	TestExecDT	InstalledChassisDriverVer	InstalledFusionDriverVer	InstalledIolsVer	InstalledCornerstoneDriverVer	InstalledSouthpawDriverVer
sq_embedded7	sq_embedded7	5/28/2017 7:52:20 AM						

Update Rate(s): 5

Figure 3-2 Test Executive Server without client connected

Figure 3-3 shows the diagnostic tool on the second tab bottom of Test Executive server. The main function in this tab is to manually send individual command to dedicated client. This diagnostic function is useful when the need to run a single test group or installation or query for client detail.



Controller	ControllerName	CAlive	TestExecDT	InstalledChassisDriverVer	InstalledFusionDriverVer	InstalledIolsVer	InstalledCornerstoneDriverVer	InstalledSouthpawDriverVer
sq_embedded7	sq_embedded7	5/28/2017 10:44:22 AM						

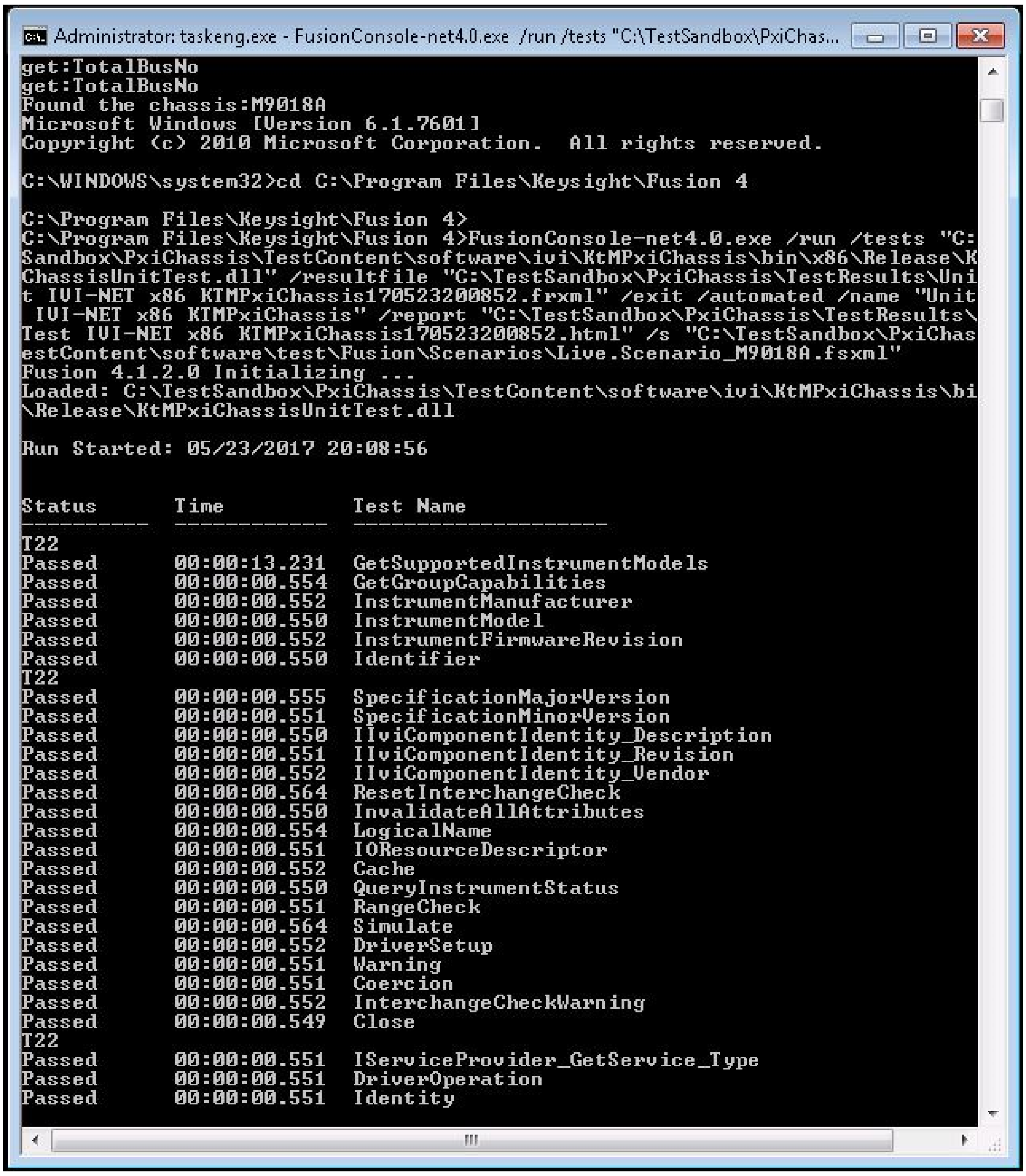
Controller Name: <sq_embedded7

Command to send:

```
:TestCases:Installer:{}:CopyLatestChassisInstaller:{}>
:TestCases:Installer:{}:InstallVerification:{}>
:TestCases:Installer:{}:UninstallVerification:{}>
:TestCases:Installer:{}:InstallVerificationVerOf:{1.6.545.0}>
:TestCases:Installer:{}:UninstallVerificationVerOf:{1.6.545.0}>
:TestCases:PXIChassisIVINetUnitTest:{}:Runx86Test:{}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVINET,x86,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVINET,x64,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVIC,x86,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVIC,x64,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVINET,x86,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVINET,x64,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVIC,x86,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVIC,x64,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVIC,x86,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVIC,x64,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVINET,x86,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVINET,x64,KtMPxiC}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVINET,x86,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVINET,x64,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVIC,x86,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVIC,x64,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVINET,x86,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVINET,x64,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVIC,x86,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVIC,x64,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVIC,x86,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVIC,x64,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVINET,x86,KtMTrig}>
:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVINET,x64,KtMTrig}>
```

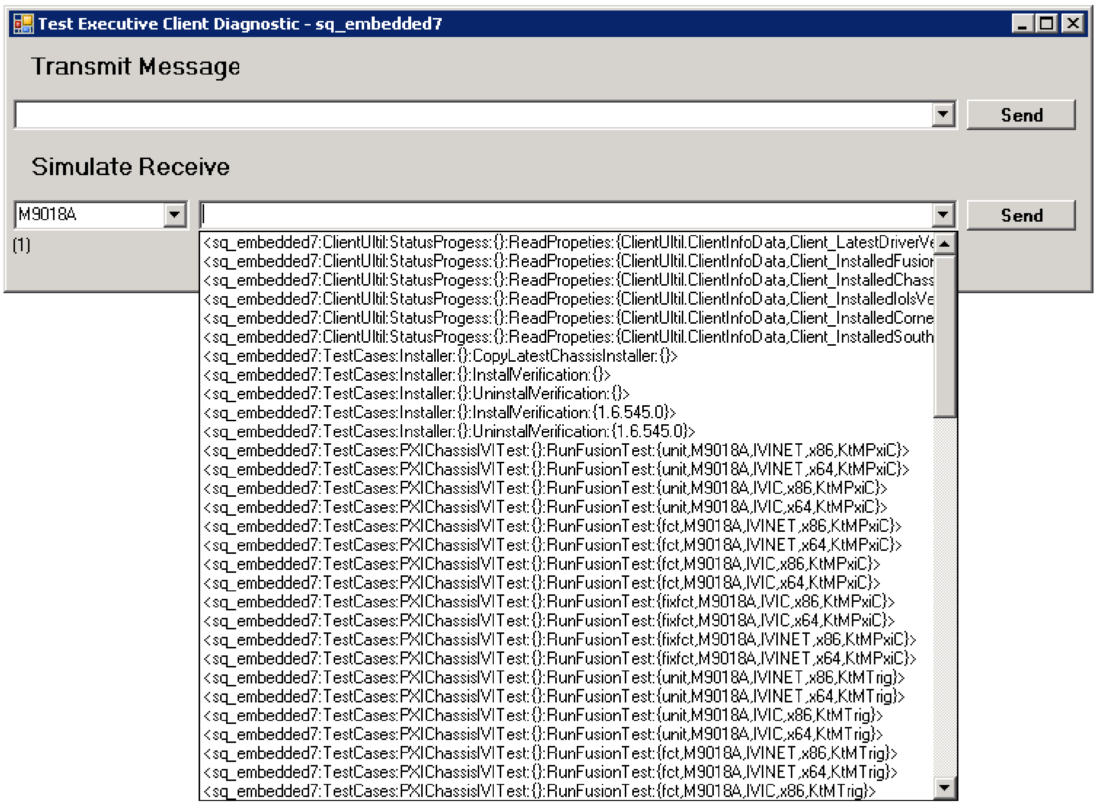
Figure 3-3 Test Executive Server Diagnostic Tools

Figure 3-4 illustrates the Test Executive Client console, this console will start all the necessary threads when it has received command from the server. This client application will automatically start as soon as the Windows operating system completes boot into desktop. Figure 3-5 shows the diagnostic interface for the client application, this UI have two function, to send a simulate command to server and execute command by injecting a command into receive queue.


Administrator: taskeng.exe - FusionConsole-net4.0.exe /run /tests "C:\TestSandbox\PxiChas...

```
get:TotalBusNo
get:TotalBusNo
Found the chassis:M9018A
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2010 Microsoft Corporation.  All rights reserved.

C:\WINDOWS\system32>cd C:\Program Files\Keysight\Fusion 4

C:\Program Files\Keysight\Fusion 4>
C:\Program Files\Keysight\Fusion 4>FusionConsole-net4.0.exe /run /tests "C:
Sandbox\PxiChassis\TestContent\software\ivi\KtMPxiChassis\bin\x86\Release\K
ChassisUnitTest.dll" /resultfile "C:\TestSandbox\PxiChassis\TestResults\Uni
t IVI-NET x86 KTMPxiChassis170523200852.frxml" /exit /automated /name "Unit
 IVI-NET x86 KTMPxiChassis" /report "C:\TestSandbox\PxiChassis\TestResults\
Test IVI-NET x86 KTMPxiChassis170523200852.html" /s "C:\TestSandbox\PxiChas
estContent\software\test\Fusion\Scenarios\Live.Scenario_M9018A.fsxml"
Fusion 4.1.2.0 Initializing ...
Loaded: C:\TestSandbox\PxiChassis\TestContent\software\ivi\KtMPxiChassis\bi
\Release\KtMPxiChassisUnitTest.dll

Run Started: 05/23/2017 20:08:56


Status      Time          Test Name
----------  ------------  --------------------
T22
Passed      00:00:13.231  GetSupportedInstrumentModels
Passed      00:00:00.554  GetGroupCapabilities
Passed      00:00:00.552  InstrumentManufacturer
Passed      00:00:00.550  InstrumentModel
Passed      00:00:00.552  InstrumentFirmwareRevision
Passed      00:00:00.550  Identifier
T22
Passed      00:00:00.555  SpecificationMajorVersion
Passed      00:00:00.551  SpecificationMinorVersion
Passed      00:00:00.550  IIviComponentIdentity_Description
Passed      00:00:00.551  IIviComponentIdentity_Revision
Passed      00:00:00.552  IIviComponentIdentity_Vendor
Passed      00:00:00.564  ResetInterchangeCheck
Passed      00:00:00.550  InvalidateAllAttributes
Passed      00:00:00.554  LogicalName
Passed      00:00:00.551  IOResourceDescriptor
Passed      00:00:00.552  Cache
Passed      00:00:00.550  QueryInstrumentStatus
Passed      00:00:00.551  RangeCheck
Passed      00:00:00.564  Simulate
Passed      00:00:00.552  DriverSetup
Passed      00:00:00.551  Warning
Passed      00:00:00.551  Coercion
Passed      00:00:00.552  InterchangeCheckWarning
Passed      00:00:00.549  Close
T22
Passed      00:00:00.551  IServiceProvider_GetService_Type
Passed      00:00:00.551  DriverOperation
Passed      00:00:00.551  Identity
```

Figure 3-4 Test Executive Client console



```
<sq_embedded7:ClientUltil:StatusProgess:{}:ReadPropeties:{ClientUltil.ClientInfoData,Client_LatestDriverVe
<sq_embedded7:ClientUltil:StatusProgess:{}:ReadPropeties:{ClientUltil.ClientInfoData,Client_InstalledFusior
<sq_embedded7:ClientUltil:StatusProgess:{}:ReadPropeties:{ClientUltil.ClientInfoData,Client_InstalledChass
<sq_embedded7:ClientUltil:StatusProgess:{}:ReadPropeties:{ClientUltil.ClientInfoData,Client_InstalledIolsVe
<sq_embedded7:ClientUltil:StatusProgess:{}:ReadPropeties:{ClientUltil.ClientInfoData,Client_InstalledCorne
<sq_embedded7:ClientUltil:StatusProgess:{}:ReadPropeties:{ClientUltil.ClientInfoData,Client_InstalledSouth
<sq_embedded7:TestCases:Installer:{}:CopyLatestChassisInstaller:{}>
<sq_embedded7:TestCases:Installer:{}:InstallVerification:{}>
<sq_embedded7:TestCases:Installer:{}:UninstallVerification:{}>
<sq_embedded7:TestCases:Installer:{}:InstallVerification:{1.6.545.0}>
<sq_embedded7:TestCases:Installer:{}:UninstallVerification:{1.6.545.0}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVINET,x86,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVINET,x64,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVIC,x86,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVIC,x64,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVINET,x86,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVINET,x64,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVIC,x86,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVIC,x64,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVIC,x86,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVIC,x64,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVINET,x86,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fixfct,M9018A,IVINET,x64,KtMPxiC}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVINET,x86,KtMTrig}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVINET,x64,KtMTrig}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVIC,x86,KtMTrig}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{unit,M9018A,IVIC,x64,KtMTrig}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVINET,x86,KtMTrig}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVINET,x64,KtMTrig}>
<sq_embedded7:TestCases:PXIChassisIVITest:{}:RunFusionTest:{fct,M9018A,IVIC,x86,KtMTrig}>
```

Figure 3-5 Test Executive Client Diagnostic UI

3.3.1 Server and client concept

The server-client concept in Figure 3-6 was chosen in this work because it is one of the most feasible ways to manage multiple controllers by using a single computer. To fulfill all the test coverage the software and hardware on the client will always change after complete a specific test. The Test Executive Server application has the capability to detect if there is a client alive and return the information for further action. The Test Executive Server will interactively send command and receive information from the client. The Test Executive Server will start with query the client status including existing hardware and software currently available on client side, once it finishes the query it will continue to check and compare the test requirement if it meet the loaded test suite. If the software requirement does not meet it will try to install the needed software. The Test Executive Server will continue verify and download necessary test files from the shared file server. After complete verify the file needed it will continue to load