Research on Development and Test of Signal-oriented Driver

Yiping Wang
Naval Aeronautical and Astronautical University, Yantai, China
Email: 13505358626@163.com

Tianzhu Wen and Yongshuang Shang
Naval Aeronautical and Astronautical University, Yantai, China
Email: wentianzhu1987@yahoo.com.cn

***Abstract*—In this paper, the development and test of signal-oriented driver are studied. Firstly, the signal and test model based on STD (Signal and Test Definition) standard are introduced. Signal models include static signal model and dynamic signal model. Test models include intrinsic measurement and generic measurement. Secondly, the developing process of signal-oriented driver is analyzed, including instruments analysis, instrument capability description file editing, driver framework determination, internal code compilation and driver debugging. At last the test programs for Ag34410A signal-oriented driver compiled by TPL (Test Procedure Language) are shown. The test results illuminate that the development and test of signal-oriented driver are reasonable, and the driver for Ag34410A is right, but there are also some other problems should be resolved for realizing signal-oriented test.**

***Index Terms*—STD, signal model, signal-oriented, TPL, ATS.**

I. Introduction

Since the function of connecting instrument with computer and intelligence device is supplied, RS232、GPIB、USB and LAN interfaces have been developed by manufacturers one after another. In that case, the instrument driver is generated to unify the method of using and controlling the instrument. The essence of instrument driver is a set of subroutines which can be invoked by users. By using it, various functions of instrument can be completed even without knowing the programming protocol and steps of each instrument. In order to ensure that the instrument control commands or language has good consistency and achieve the goal of instrument interchangeable and TPS portable, series of software technical specifications and standards are launched [1,2].

As a result of different objects, test technologies are divided into two categories: instrument-oriented test and signal-oriented test. Software technical specifications and standards of instrument-oriented test mainly include IEEE-488、SCPI (standard command for programming instrument) and VPP (VXI plug & play). The software technical specifications and standards of signal-oriented test mainly include ATLAS (Abbreviated Test language for All System), IVI-Signal and STD [3]. The development course of them is depicted in Fig. 1.

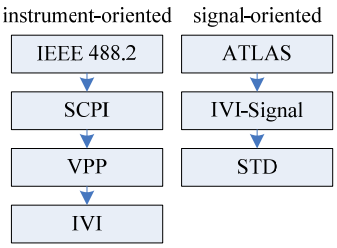

Figure 1 The development of software technical specifications and standards

IEEE-488.2 defines the encoding, syntax format, information exchange control protocol and public remote command statement of using GPIB bus, but doesn't define the instrument-related command. To a certain extent, it has achieved GPIB bus-based instrument of the same manufacturer interchangeable. SCPI which is on the basis of IEEE-488.2 and IEEE-754 is a set of common commands. It can control many types of instrument in the same way but also limit the expansion of instrument function at the same time, so it can be said that it achieves part of instrument interchangeable whose function is strictly matched. VPP specification unifies the software control interface of various instruments by defining VISA (Virtual Instrument Software Architecture) library. However, there is no strict semantics definition, so it can only achieve the same type instrument of the same manufacturer interchangeable. IVI model is a driver design standard which is defined on the basis of VPP specification by IVI foundation. It achieves the same type instrument interchangeable by defining class driver and physical driver, but it has many disadvantages, such as the useful standard is little, the openness of standard is low and only 80% function of the same type instrument can be interchangeable.

Instrument-oriented test is disadvantage to implement instrument interchangeable, so signal-oriented test on which expectations are centered. ATLAS is a special test language which has complete syntax structure and semantics definition, but there are many problems arising from lack of expansion capability and rigorous mathematical definition. IVI-Signal defines the method of how to convert the instrument control command into the test signal description and achieves a high level

doi:10.4304/jsw.7.8.1807-1815

instrument interchangeable. But the instrument driver takes the form of instrument-oriented COM components, so the really instrument interchangeable isn't achieved. STD standard provides the means to define and describe signals used in testing. It includes SML (Signal Model Language) layer, BSC (Basic Signal Component) layer, TSF (Test Signal Framework) layer and Test Requirement layer. According to it, users are allowed to choose the operating environment and even the programming languages. So it can resolve the problems which we encountered before and achieve the goal of instrument interchangeable and TPS portable.

The study described in this paper consists of the development and test of signal-oriented instrument driver which are based on STD standard. The organization of this paper is as follow. Section II introduces signal and test model based on STD standard. In section III, the model of signal-oriented driver is shown, and the developing process of signal-oriented driver is analyzed. In addition, the Ag34401A driver is made as an example. In section IV, the method of test signal-oriented driver and the TPL of test Ag34401A driver are given. Finally, future work for the further development of this system is proposed.

II. Signal And Test Model

The signal and test model should be first introduced before developing signal-oriented driver. Because it can be used to describe not only test requirement but also instrument. By citing a specific example, the way of building self-defined signal model based on BSCs and TSF signals in STD standard is analyzed, and the test block diagram based on both test models are given.

A. Signal Model

Signal includes static signal and dynamic signal in STD standard. Static signal is a signal whose definition does not change over time. All basic signal components (BSCs) and test signal framework (TSF) models are static signals. Dynamic signal is a signal whose definition changes over time, by use of the control interface. These changes must be initiated with one of the signal method calls or by changing the interconnections of a signal model [4].

1) Static signal model

BSCs, the fundamental components of STD standard, play the role of building blocks which can be used to define more complex signals but can't be decomposed into simpler components. Fig. 2 represents a generalized form of a BSC and shows all possible interfaces and properties.

In the BSC model shows in Fig. 2, Class Name is the name of the BSC subclasses, e.g., Sinusoid. Signal Name is the name of the specific signal being modeled e.g., Phase A. BSC provides the signal interface that transfers a signal, its attributes, and its values to other BSCs. In is the reference of input signal. Out is the output signal. Sync and Gate are events which decided the behavior of BSC.

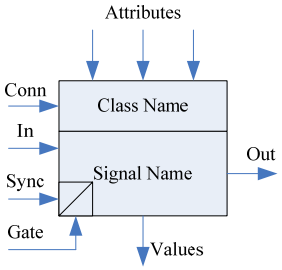

Figure 2 BSC diagram

2) Dynamic signal model

Dynamic signal model reflects the changes of signal states and the interaction between the interfaces of all the BSCs in signal model.

The states of signal defined in STD standard contain stopped, paused and running [4].

Stopped—The Stopped state indicates that the signal is in a generalized reset condition, i.e., no signal activity is present. Thus a Stopped Signal can represent either no signal at all or a signal from an allocated resource that has not been activated or triggered. All Signals initiate to the Stopped state.

Paused—The Paused signal is waiting to be triggered into the Running state by an external event. A Paused signal does not yet exist, but all the necessary resources have been acquired and prepared and are awaiting the final on or go event.

Running—The Running signal is active and exists as a signal or gated event stream. A Running Signal is measurable and available for use.

The methods defined in STD standard contain stop, run and change.

Stop ([timeout=0])—The Stop method resets, disconnects, or turns off any Paused or Running Signal and frees any associated signal resources. Following a successful Stop, the state of the Signal will become Stopped.

Run ([timeout=0])—The Run method sets up, starts, connects, or turns on a Stopped signal. Following a successful Run, the state of the Signal is Paused and subsequently becomes Running. The Run method on a Running or Paused Signal will reinitialize the Signal to its value at time t = 0.

Change ([timeout=0])—The Change method initiates the Signal to its next setting. If no further settings are pending, Change () indicates that the current Signal is finished and no longer needed. This knowledge allows the source BSCs to change the signal to the next available setup. If no further signal conditions are available, Change () resets the signal to the Stopped state.

Methods and states are interdependent. Calling a method indicates an intention for a state to change. See Fig. 3.

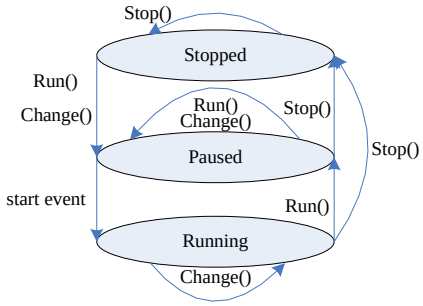

Figure 3 Signal state changes

B. *Measurenment Model*

The STD standard provide for two methods with which measurement values (and test) can be specified. One is intrinsic measurement (e.g., RMS, Instantaneous) and the other is generic measurement (Measure As) [5, 6].

1) Intrinsic Measurement

Intrinsic measurement defines methods or techniques which are signal type independent, including

Counter: counts when a measurement would be taken, but does not take any specific measurement.

Interval: measures the interval between the In/Sync event going active and the Gate event going active.

Instantaneous: measures the amplitude of the signal in the dimension “type” at specified instances in time.

RMS: measures the root-mean-square (rms) value of a signal.

Average: is the arithmetic mean of all the signal values during the gate time.

PeakToPeak: is the difference between the highest value and the lowest value during the gate time.

Peak: is the measured value that is furthest away from the mean value.

PeakPos: is the value obtained by subtracting the mean value from the maximum measurement of the signal during the gate time.

PeakNeg: is the value obtained by subtracting the mean value from the minimum measurement of the signal during the gate time.

MaxInstantaneous: is the maximum measurement of the signal during the gate time.

MinInstantaneous: is the minimum measurement of the signal during the gate time.

2) Generic Measurement

Fig. 4 is used to explain the principle of generic measurement, the red broken line indicate the AC signal to be measured, and the test parameter is frequency, in which XML static signal description is <Measure As="tsf: AC_SIGNAL" attribute="freq" nominal=" 120 V"/>. Because of the reference signal is AC_SIGNAL whose voltage is 120 V, changed the frequency of reference signal until the minimum rms value is gotten, and the value of that reference signal frequency is the test result.

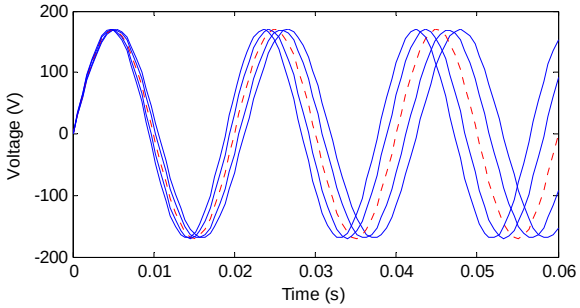

Figure 4 Generic measure best match example

C. *Example*

In this example, the UUT test requirements are defined as follow

Input a 120V, 50Hz three-phase ac signal with 1A current limit o the UUT.

Assuming the output signal is ac signal, measure the RMS voltage.

Assuming the type of output signal is unknown, measure the RMS voltage.

1) signal modeling

According to the test requirement, the three-phase ac signal which has current limit attribute is needed to build. It should have seven BSCs. The interface properties and model description of AC3PHASE_POWER are shown in Table Ⅰ and Table Ⅱ respectively.

TABLE I.
AC3PHASE_POWER INTERFACE PROPERTIES

Description	Name	Type	Default	Range
AC3PHASE_POWER Signal amplitude	ampl	Physical	—	—
AC3PHASE_POWER Signal frequency	freq	Physical	—	—
AC3PHASE_POWER Signal Current limit	curr_limit	Physical	—	—

TABLE II.
AC3PHASE_POWER MODEL DESCRIPTION

Name	Type	Terminal	Input	Output	Formula
Three Phase Wye	ThreePhaseWye	Signal[Out]	—	AC3PHASE_POWER	—
		channelWidth	—	—	3
		Signal[In]	Current Limit A	—	—
		Signal[In]	Current Limit B	—	—
		Signal[In]	Current Limit C	—	—
Current Limit A	Limit	Signal[Out]	—	Three Phase Wye	—
		limit	curr_limit	—	—
		Signal[In]	Phase A	—	—
Current Limit B	Limit	Signal[Out]	—	Three Phase Wye	—
		limit	curr_limit	—	—
		Signal[In]	Phase B	—	—
Current Limit C	Limit	Signal[Out]	—	Three Phase Wye	—
		limit	curr_limit	—	—
		Signal[In]	Phase C	—	—
Phase A	Sinusoid	Signal[Out]	—	Current Limit A	—
		amplitude	ampl	—	—
		frequency	freq	—	—
		phase	—	—	-2/3 π
Phase B	Sinusoid	Signal[Out]	—	Current Limit B	—
		amplitude	ampl	—	—
		frequency	freq	—	—
		phase	—	—	0
Phase C	Sinusoid	Signal[Out]	—	Current Limit C	—
		amplitude	ampl	—	—
		frequency	freq	—	—
		phase	—	—	2/3 π

The equation of Phase A, Phase B and Phase C in AC3PHASE are:

Phase A: $Y_a = A\cos(2\pi ft - \frac{2}{3}\pi)$

Phase B: $Y_b = A\cos(2\pi ft)$

Phase C: $Y_c = A\cos(2\pi ft + \frac{2}{3}\pi)$

Where A is the amplitude, f is the frequency.

XML static signal description of AC3PHASE_POWER is < AC3PHASE_POWER name = "AC3P1" ampl = "120 V" freq = "50 Hz" curr_limit = "2 A" / >, and the waveform is shown in Fig. 5.

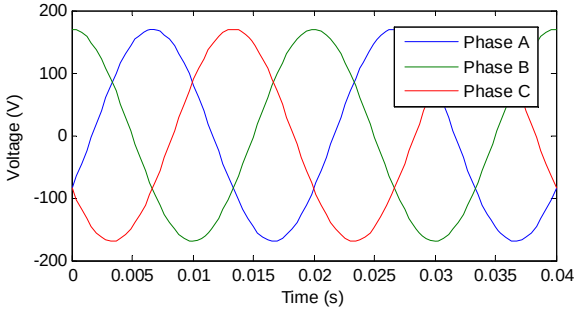

Figure 5 AC3PHASE_POWER example

The signal model of AC3PHASE_POWER is shown in Fig.6.

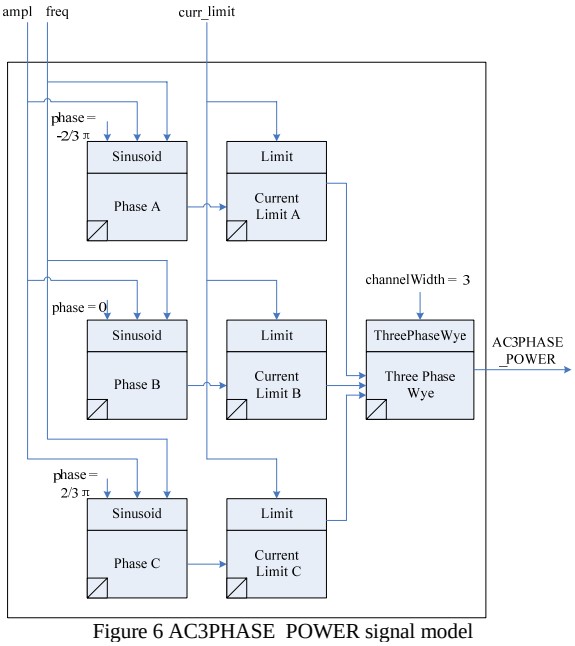

Figure 6 AC3PHASE POWER signal model

2) Signal testing

According to the test requirement, the three-phase ac signal should be supplied to the UUT. If the output signal is ac signal, the generic measurement will be taken. Else if the output signal is unknown, the intrinsic measurement will be taken. UUT test block diagram is shown in Fig. 7[7].

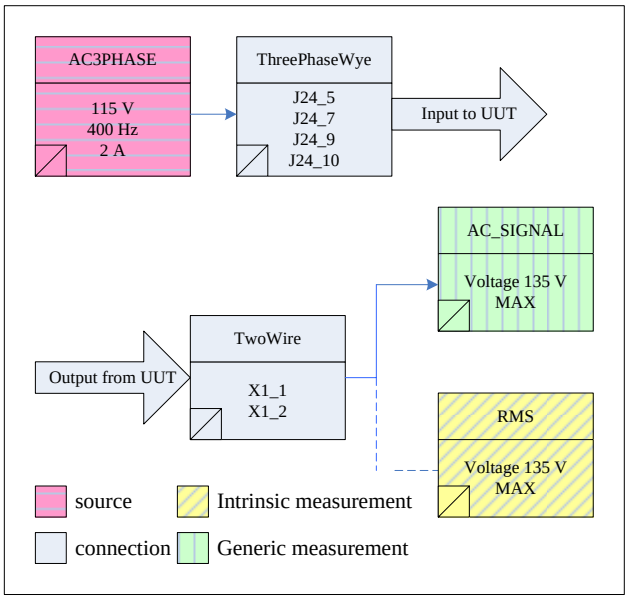

Figure 7 UUT test block diagram

In Fig. 7 red block indicates the source who supplies the tree-phase ac signal, green block indicates the generic measurement sensor, yellow block indicates the intrinsic measurement sensor and blue block indicates the connection between BSCs and UUT. Broken lines indicate that the two measurement models can't run at the same time.

III. Signal-oriented Dreiver Development

Signal-oriented test process is shown in Fig.8. When the test requirement and instrument are described in signals, virtual resources and physical resources are generated in turn. Then Run time system completes the resources matching and path searching according to test information, path information and instrument capability, and calls appropriate signal-oriented driver to perform the test. Signal-oriented model is in the

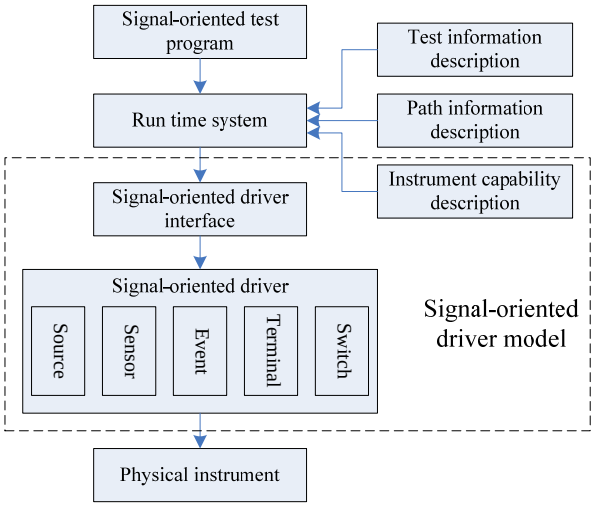

Figure 8 Signal-oriented test process

Driver development, in essence, is using SCPI command and manufactures' driver to compile instrument common functions and physical resource interface functions. The development of signal-oriented driver includes five steps, namely instrument analysis, instrument capability description file editing, driver framework determination, internal code compilation and driver debugging. Flowchart of driver development is shown in Fig. 9.

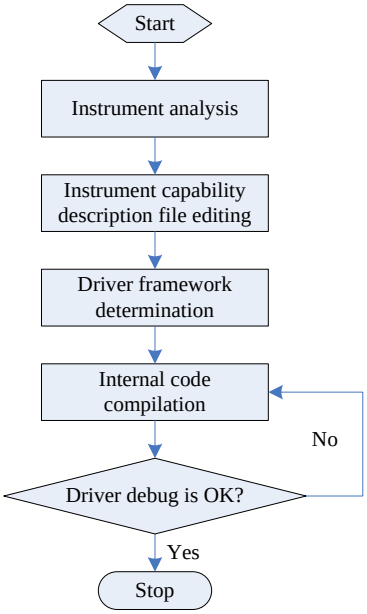

Figure 9 Flowchart of driver development

A. Insreumen Analysis

Instrument analysis is the first step in driver development. The basic information of instrument such as bus type, model and address can be got after analysis. Then manual and program control of instrument are needed. With instrument manual control, not only the results of instrument single function test can be validated, but also the logic and rigor of program will be enhanced. With instrument program control, the usage of programmed instructions and manufactures' drivers can be learned.

For example, analyze Ag34410A which is produced by Agilent. It is a digital multimeter which contains six physical resources, including DC voltage, DC current, AC voltage, AC current, frequency and resistance physical resource. The bus type of it is GPIB, and the four ports of it are INPUT_HI, INPUT_LO, CURR_HI and CURR_LO.

B. Instrument Capability Description File Editing

Driver development is based on instrument capability description file, in which the role, signal and ports of instrument physical resource are described. The role of instrument physical resource can be categorized into five groups: source, sensor, terminal, event and switch. Source is used to supply the signal to UUT. Sensor is used to measure the signal from UUT. Terminal is used to both supply and measure the signal from the UUT. Event is used to supply the sync signal and trigger signal. Switch is used to connect and disconnect the signal. The architecture of instrument capability description file is shown in Fig. 10.

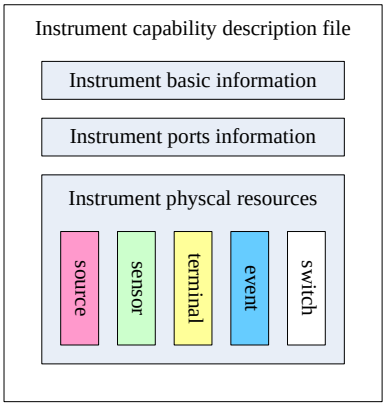

Figure 10 Architecture of instrument capability file

C. Driver Framework Determination

There are common functions and interface functions in the driver. The common functions contain setup function, close function, reset function and self test function. And the interface functions are depended on the role of instrument physical resource, which the mapping relation between interface functions and the role of instrument physical resource is shown in TableⅢ.

According to the instrument capability description file, the functions' name and type can be got, so the driver framework which contains all the source codes except programmed instructions and manufactures' drivers can be generated automatically.

TABLE III.
ROLE AND INTERFACE FUNCTIONS MAPPING

role	Interface functions	function
source	InstrmentPrefix_ResourcePrefix_Setup (SigPSource source, int TimeOutValue).	Setup source
	InstrmentPrefix_ResourcePrefix_Run (SigPSource source, int TimeOutValue).	Output source signal
	InstrmentPrefix_ResourcePrefix_Change (SigPSource source, int TimeOutValue).	Change source attributes
	InstrmentPrefix_ResourcePrefix_Stop (SigPSource source, int TimeOutValue).	Stop source signal output
sensor	InstrmentPrefix_ResourcePrefix_Setup (SigPSensor sensor, int TimeOutValue).	Setup sensor
	InstrmentPrefix_ResourcePrefix_Run (SigPSensor sensor, int TimeOutValue).	Measure input signal
event	InstrmentPrefix_ResourcePrefix_Setup (SigPEvent event, int TimeOutValue).	Setup event
	InstrmentPrefix_ResourcePrefix_Run (SigPEvent event, int TimeOutValue).	Enable event
	InstrmentPrefix_ResourcePrefix_Stop (SigPEvent event, int TimeOutValue).	Disenable event
terminal	InstrmentPrefix_ResourcePrefix_Setup (SigPTerminal terminal, int TimeOutValue).	Setup terminal
	InstrmentPrefix_ResourcePrefix_Transmit (SigPTerminal terminal, int length, int itemSize, int data [], int TimeOutValue).	Transmit data
	InstrmentPrefix_ResourcePrefix_Receive (SigPTerminal terminal, int length, int itemSize, int data [], int TimeOutValue).	Receive data
	InstrmentPrefix_ResourcePrefix_Change (SigPTerminal terminal, int TimeOutValue).	Change terminal attributes
	InstrmentPrefix_ResourcePrefix_Stop (SigPTerminal terminal, int TimeOutValue).	Stop terminal
switch	InstrmentPrefix_ResourcePrefix_Connect (SigPSwitch switch, int Pole, int TimeOutValue).	Connect switch
	InstrmentPrefix_ResourcePrefix_Connect (SigPSwitch mtx, int row, int col, int TimeOutValue).	Connect matrix
	InstrmentPrefix_ResourcePrefix_Disconnect(SigPSwitch mtx, int row, int col, int TimeOutValue).	Disconnect matrix

D. Internal Code Compile

When driver framework is established, only internal code is left to fill. The internal code is the secondary packaging of SCPI and manufacture's drivers, which are learned in instrument analysis. So the signal-oriented driver will be completed, after internal code compilation.

To take Ag34410A for an example, there are 7 parts of contents in the driver, including common, DC voltage, DC current, AC voltage, AC current, frequency and resistance. Each part has its own functions which are shown in Figure 11.

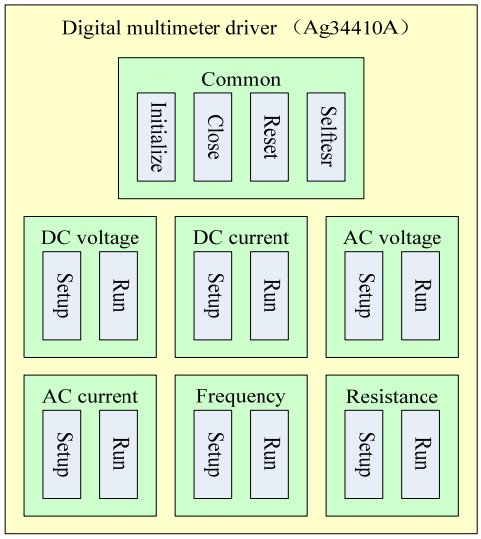

Figure 11 Architecture of instrument capability file

E. *Driver Debuging*

Driver debugging is a boring and tedious process, but very necessary. Through repeated testing an instrument, it can be make sure that whether the function of driver is complete, the operation of testing is reasonable, or the output and measurement results are correct. During the driver debugging, if any problem is found, the previous steps should be re-analyzed, and new driver will be generated.

Three instruments are needed to debug the six physical resources of Ag34410A. DC power Ag6673 is supplied for debugging DC voltage and DC current physical resource. AC power Ci801RP is supplied for debugging AC voltage, AC current and frequency physical resource. Programmable resistance Ra4072A is supplied for debugging resistance physical resource. The driver debugging model of Ag34410A is shown in Fig 12.

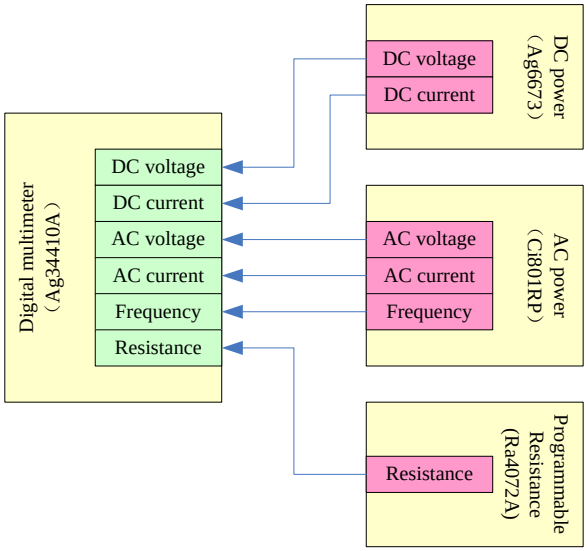

Figure 12 driver debugging model of Ag34410A

IV. SIGNAL-ORIENTED DREIVER TEST

The signal-oriented driver needs to be tested, after its development. TPL is used to compile the test program of driver, for the reason that it can be embedded into any carrier language, and has good portability.

A. *TPL Statements*

TPL statements are focused on single actions. Single action test statements describe a critical testing action that can't be further subdivided with respect to the UUT. TPL statements provided in STD are shown in Table Ⅳ.

TABLE IV.
TPL STATEMENTS

keywords	TPL
Setup	Setup source Setup sensor Setup signal-based event Setup event-based event Setup time-based event Setup clock Setup time interval measurement Setup event counter Setup sensor(for undefined signal) Setup signal
Reset	Reset
Connect	Connect source Connect sensor Connect pin to pin
Disconnect	Disconnect
Enable	Enable(general case) Enable signal-based event
Disable	Disable
Read	Read
Change	Change
Compare	Compare
Wait_For	Wait_For

B. *Test Program Interface*

The test program of driver is needed to testify whether the driver meet the requirement and is convenience to use. And it can control instrument directly. The interface of Ag34410A test program is shown in Fig 13.

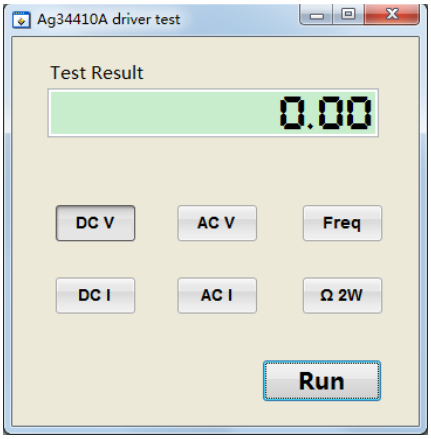

Figure 13 Driver test interface

C. Test Requirement in TPL

Test requirements can be written in a text format by TPL. And different from using ATLAS, TPL can be embedded into any carrier language, so the user can adopt their own preferred programming language [8, 9]. For example, when test the driver of Ag34410, the DC voltage signal, DC current signal, AC voltage signal, AC current signal and load signal are all needed. Those signal requirements can be written by TPL. In the following, the "snr" indicates the handle of sensor signal.

1) Require a signal

Require a DC voltage signal and measure its amplitude whose range is from -5V to 5V can be written as:

Setup *DC_SIGNAL dc_ampl range -5 V to 5 V as sensor snr;*

Require a DC current signal and measure its amplitude which range is from -0A to 3A can be written as:

Setup *DC_SIGNAL(Current) dc_ampl range 0 A to 3 A as sensor snr;*

Require an AC voltage signal and measure its amplitude whose maximum value is 300 V can be written as:

Setup *AC_SIGNAL ac_ampl range Max 300 V as sensor snr;*

Require an AC current signal and measure its amplitude whose range is from 0A to 1A can be written as:

Setup *AC_SIGNAL(Current) ac_ampl range 0 A to 1 A as sensor snr;*

Require an AC voltage signal and measure its frequency whose range is from 0 Hz to 400 Hz can be written as:

Setup *AC_SIGNAL freq range 0 Hz to 400 Hz as sensor snr;*

Require a load signal and measure its resistance whose range is from 0 Ohm to 200 M Ohm can be written as:

Setup *Load resistance range 0 Ohm to 200 M Ohm as sensor snr;*

2) Connect signal

Connect voltage signal and load signal to the ports of instrument can be written as:

Connect *hi INS_DMM::INPUT_HI lo INS_DMM::INPUT_LO to snr;*

Connect current signal to the ports of instrument can be written as:

Connect *hi INS_DMM::CURR_HI lo INS_DMM::INPUT_LO to snr;*

3) Read measurement results

Read the measurement results can be written as:

Read *snr into testResult;*

4) Disconnect signal

After measurement, disconnect signal from ports can be written as:

Disconnect *snr;*

5) Reset signal

At last the signal should be released, which can be written as;

Reset *snr;*

D. Test Results

The nominal value and measured value of DC voltage, DC current, AC voltage, AC current, frequency and resistance are shown in TABLE V

TABLE V.
TEST RESULTS

Test item	Nominal value	Measured value
DC voltage	5 V	4.99V
DC current	0.1 A	0.10 A
AC voltage	115 V	114.99V
AC current	1 A	1.00 A
frequency	400 Hz	399.98 Hz
resistance	100 M Ω	101.05 M Ω

V. CONCLUSION

In this paper the development and test of signal-oriented driver based on STD standard are studied. The signal and test model are introduced to describe instrument. The developing process and testing approach of signal-oriented driver are given. The model of tree-phase AC signal is built to illuminate how to built self-defined signal, and two different kinds of measurement methods are used to attest the validity of this model. The model of drive for Ag4410A DC voltage physical resource is shown, and the corresponding test program is compiled by TPL. However, there are also many problems needed to study such as signal matching, path searching and TPL complier.

REFERENCES

[1] Ashley Hulme, Keri Nash. Implementing IEEE 1641 – Using a complete system [C]. IEEE AUTOTESTCON 2008 PROCEEDINGS. US: IEEE, 2008:282-288

[2] Qiao Liyan, Liu Zhaoqing, Peng Yu, Peng Xiyuan. A TPS Integrated Development Environment Implementing IEEE1641 and ATML [C]. IEEE AUTOTESTCON 2009 PROCEEDINGS. US: IEEE, 2009:246-250

[3] Ashley M B Hulme. Physical signals, events and digital streams Their relationship and how they affect SignalFunctions in IEEE 1641 [C]. IEEE AUTOTESTCON 2009 PROCEEDINGS. US: IEEE, 2009:205-310

[4] IEEE Std 1641™-2010, IEEE Standard for Signal and Test Definition[S]. Institute of Electrical and Electronics Engineers, Inc.,2010

[5] IEEE Std 1641.1™-2006, IEEE Guide for the Use of IEEE Std 1641, Standard for Signal and Test Definition[S]. Institute of Electrical and Electronics Engineers, Inc, 2006.

[6] Ashley M B Hulme. Measurement in IEEE 1641 and its application in a CASS TSF Library [C]. IEEE AUTOTESTCON 2010 PROCEEDINGS. US: IEEE, 2010:174-179

[7] Ron Taylor. Test Diagram Generation: A Practical Application of the ATML Standards [C]. IEEE AUTOTESTCON 2009 PROCEEDINGS. US: IEEE, 2009:322-326

[8] Christophe Grard. A practical usage of the 1641 signal definition using the Test Procedure Language(TPL) and a Carrier Language (CL) [C]. IEEE AUTOTESTCON 2008 PROCEEDINGS. US: IEEE, 2008:295-299

[9] Ashley M B Hulme. Managing the transition to IEEE 1641, via ATLAS based test systems [C]. IEEE AUTOTESTCON 2009 PROCEEDINGS. US: IEEE, 2009:251-255

Yiping Wang was born in Taidong, China, in 1975. He received MS from Naval Aeronautical and Astronautical University in 2003. Now she is a PhD candidate in Naval Aeronautical and Astronautical University. Her research interests include parallel test and signal-oriented ATS.

Tianzhu Wen was born in Changchun, China, in 1987. He received MS from Naval Aeronautical and Astronautical University in 2011. Now he is a PhD candidate in Naval Aeronautical and Astronautical University. His research interests include parallel test and signal-oriented ATS.

Yongshuang Shang was born in Liaoyuan, China, in 1981. He received MS from Aviation University of Air Force in 2007. Now he is a PhD candidate in Naval Aeronautical and Astronautical University. His research interests include Prognostic and Health Management and ATS.