Kazam User's Manual

REVISION HISTORY

REV	CHANGE DESCRIPTION	APPROVED BY	DATE
01	Initial Version	Fei Huang	12/04/17
02	Added sub testplan description Added other language description	Fei Huang	12/16/17
03	Modify folder structure description Added next step control description	Fei Huang	03/16/18
04	Added an external program interface description	Fei Huang	05/03/18
05	Update the Labview Run-time Version	Fei Huang	05/21/18
06	Added case sensitive descriptions	Fei Huang	05/30/18
07	Added a new reserved global variable descriptions	Fei Huang	06/04/18
08	Added important parameters descriptions Corrected test plan descriptions Added the new error handle feature Modified the user manual file name	Fei Huang	07/12/18
09	Enhance "enable" feature	Fei Huang	08/07/18
10	Added "switch…case" for next step control Added some parameters description for GUI Added new methods how to use global variables Added some new parameters in bench configuration file Added auto-update kazam description	Fei Huang	11/06/18
11	Added default error handler feature Modified the log vi name Modified global variables description Added new commands for external API	Fei Huang	04/04/19
12	Add new functionality for "if fail goto", Kazam can handle more jump conditions Add new functionality for "goto", Kazam can handle more jump conditions	Fei Huang	06/04/19

	Add a plug-in to kazam, it is used to synchronize time with the log server Added a new feature for GUI, the SFC menu can be disabled by setting parameters in the shell configuration file Added a new feature, Kazam can load the xlsx file as TestPlan Subtest plan delimiter changed from ":" to "." Updated Kazam development steps		
13	Added parameters for "Enable", the developer can customize "Enable" parameters Add Transmit and Receive property Added a new "Non-blocking" setting in testplan Added the new features for debugging mode Added ResetPlan for debugging mode Added new GUI for Actuator mode, it is named the Actuator GUI Added new reserved global variable Added a new feature to disable start button for all GUI	Fei Huang	08/19/19
14	Added a new feature to call python script directly in "Test Vi Name" column Added a new feature to call bat file directly in "Test Vi Name" column Modified the exe call feature Replaced "Non-blocking" with "RunInParallel" Add "Pause Slot" into "Next Step CTL" Add "Continue Slot" into "Next Step CTL" Add "Stop Slot" into "Next Step CTL" Add "Slots_info" global variable new definition format	Fei Huang	10/25/19
15	Added sequence producer description	Fei Huang	12/05/19
16	Added new multiple single GUI	Fei Huang	01/14/20

	Modified the call description for exe, python, bat file		
17	Added new feature to call c# source code directly in the testplan A new global variable is defined, which is named HD_info, and it's called a hardware variable.	Fei Huang	09/29/20
18	Added a new format of BenchCFG file for centralized management Kazam Added amz_FileSync.vi to sync files Added amz_RetentionPolicy.vi to be used for data retention policy on the local computer Added the "OR" feature for result judgement Added a new feature, the sub testplan can be loaded in the 'Function Call' field Rename testplan header Added single log compression feature for amz_log_single.vi Added step by step logging feature for amz_log_single.vi Modified the description of "Route_check" global variable Modified and updated some descriptions of test plan keywords	Fei Huang	07/12/21
19	Added a new tool for CPK analysis Added single log browser Added the "NOT" feature to Kazam core Add a new reserved keyword "Result" to the input parameter A new global variable is defined, which is named Bench_CFG, and it's called a bench config variable Added fixed header sequence to check the fixed column of daily log Added search feature to shortcut menu on Kazam GUI Add some new reserved variables	Fei Huang	11/03/21

1. Purpose

This document will introduce how to develop a new test or automation project using the software platform, and let the developers focus on the test or automation itself.

2. Definitions & Acronyms

AME: Advanced Manufacturing Engineering

TE: Test Engineering

GUI: Graphical User Interface

DUT: Device Under Test

SFC: Shop Floor Control System

TestPlan: It is a csv file, using for manage all the test step

CM: Contract Manufacturer

Kazam: The name of the Amazon's test framework

XLSX: It is a file extension name, it created by Microsoft Excel version 2007 and later.

Slot: If a fixture can test several DUTs, each of its DUT test locations is defined as a slot. It is also a thread concept

Actuator: It can be a robot, a cylinder, a rotating motor, a laser, a camera, a 2D/3D scanner and so on, and it could be a combination of them. It is also a thread concept, so we could say that a slot is equal to an actuator

3. Platform Architecture

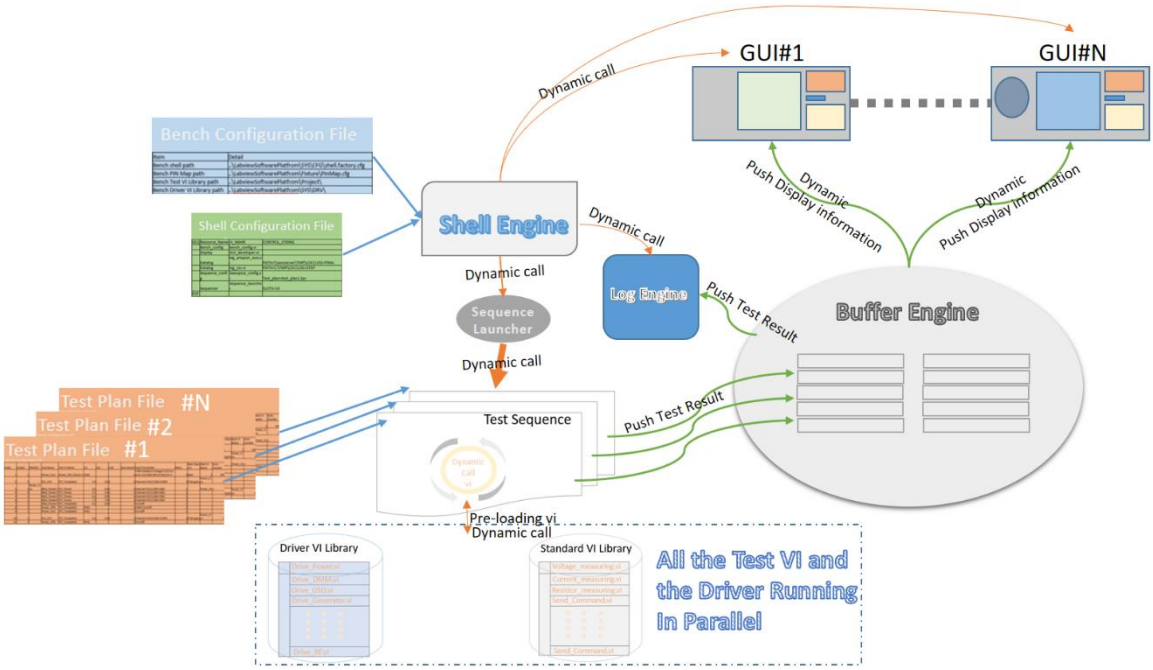


Figure 1

The Figure 1 is the software platform architecture, all the test vi and driver vi will be call by dynamic, we can use the TestPlan to set the call mode.

4. Features

4.1 Add/replace/delete function

The platform can add/replace/delete function through configure the shell configuration file.

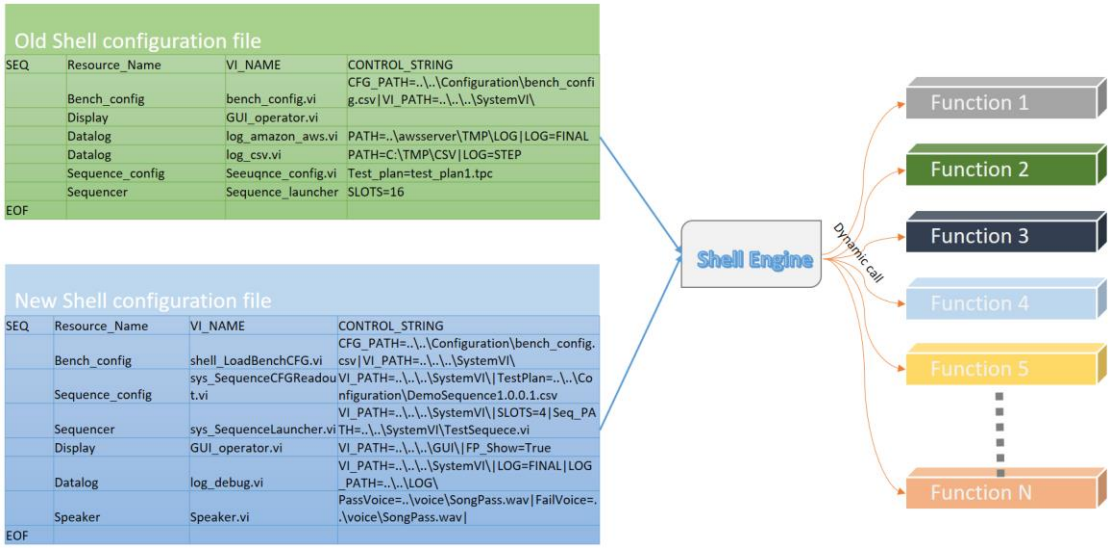


Figure 2

4.2 Running different test plan

The platform can run the different TestPlan at different slot, the Figure 3 is a simple example.

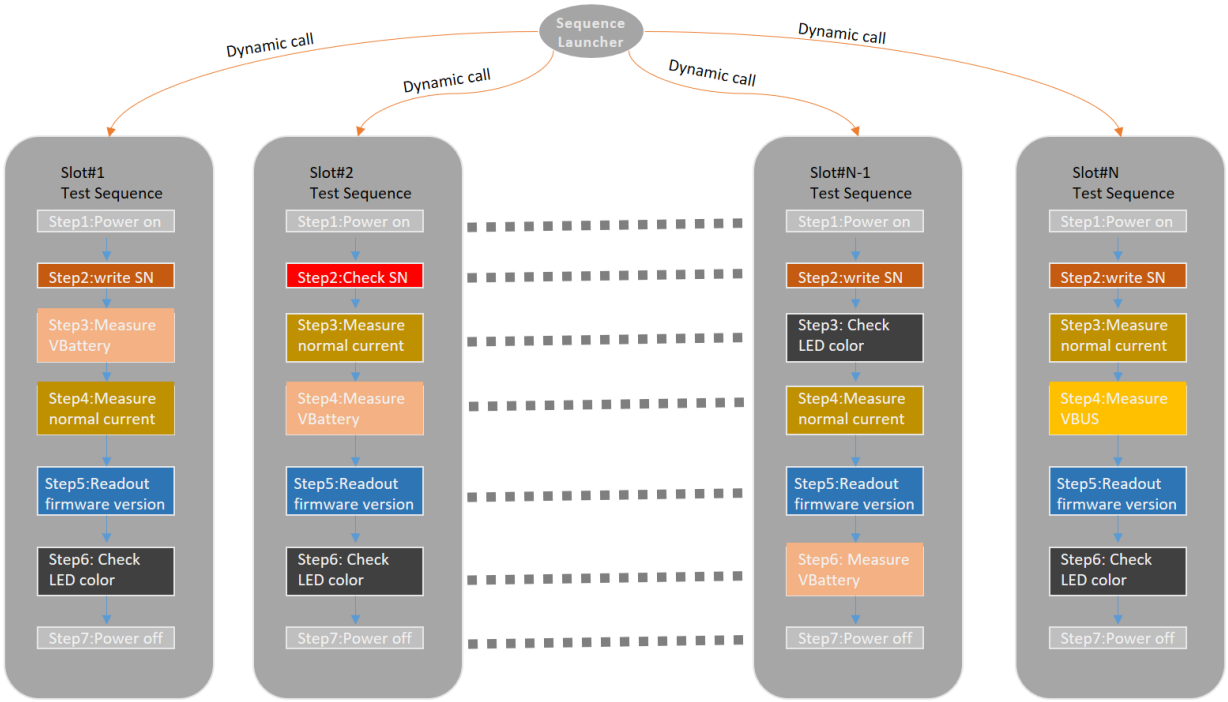


Figure 3

4.3 Running in parallel

The platform can run all the test steps in parallel as Figure 4, it is a step by step execution, from step1 to step7 at each slot. Each slot is independent of operation.

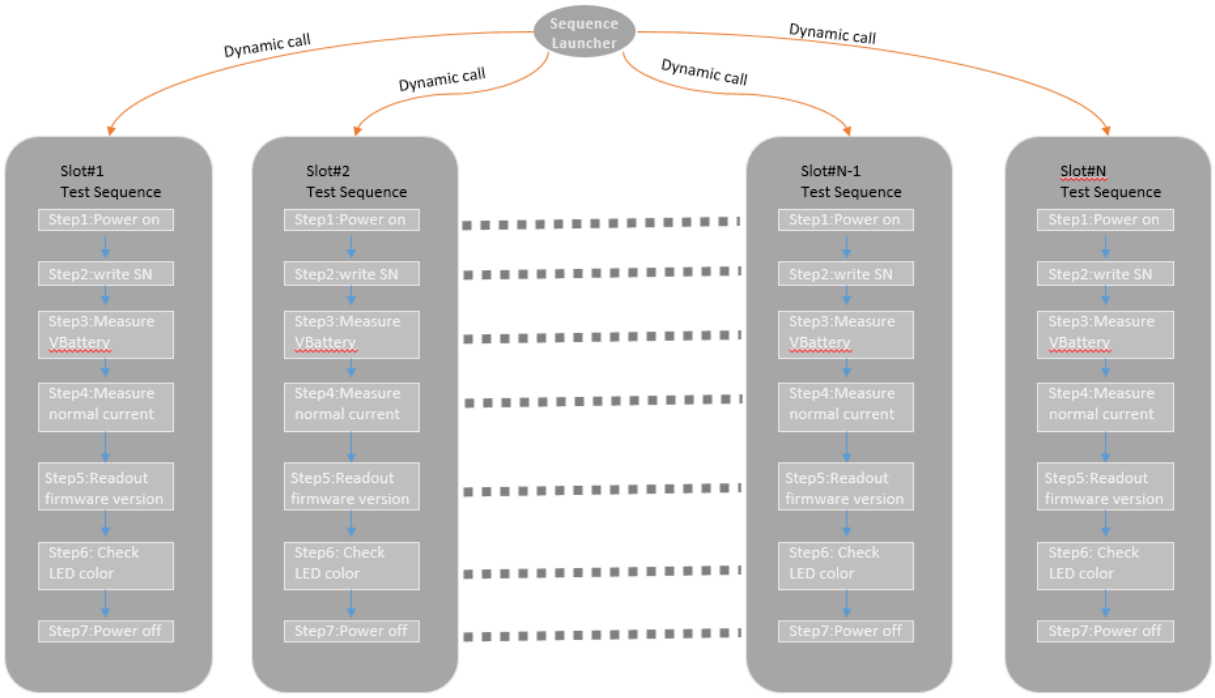


Figure 4

The platform can run parallel test steps at each slot. As Figure 4, it has seven steps, we can let the step 2 to step6 running at the same time, it as Figure 5.

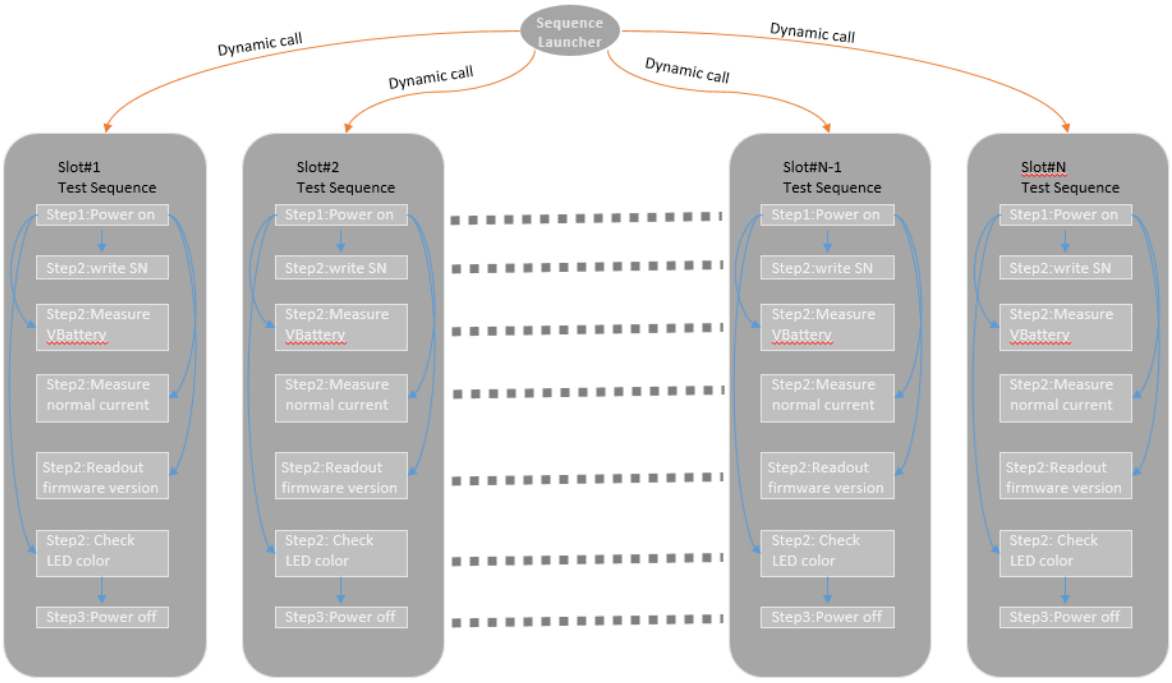


Figure 5

4.4 Automatically updates Kazam and Project folders

The platform has automatic update feature, the developer can upload the latest Kazam to AWS cloud, and then the factory's server will download it automatically, when running the Kazam.exe, Kazam will check the factory's update server automatically, if there is an update, it automatically updates the latest code.

Refer to section 8.1 for detailed parameter setting.

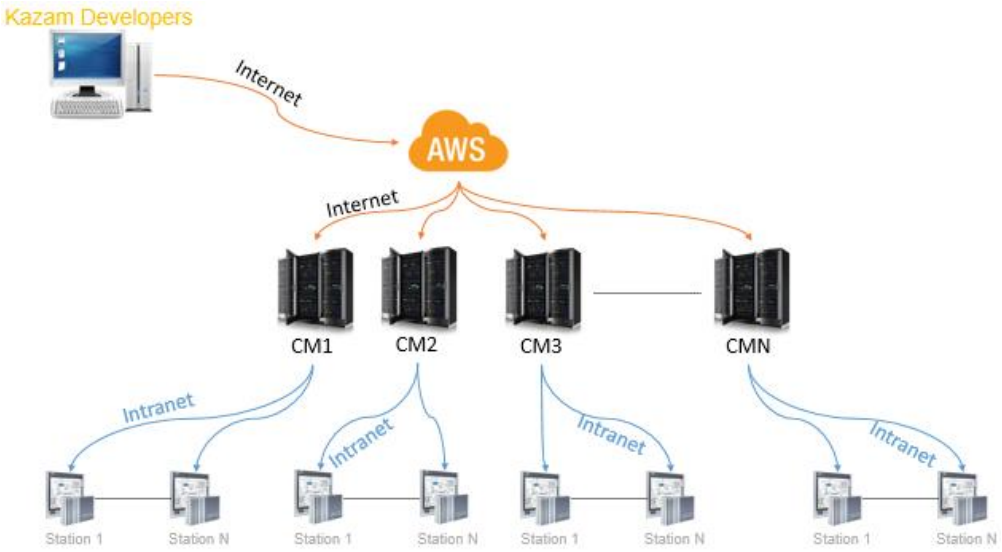


Figure 6

5. GUI Introduce

Figure 7 is the panel operator GUI, it used in production for panel test. It can easy to set the number of the panel. Supports up to 32 DUTs.

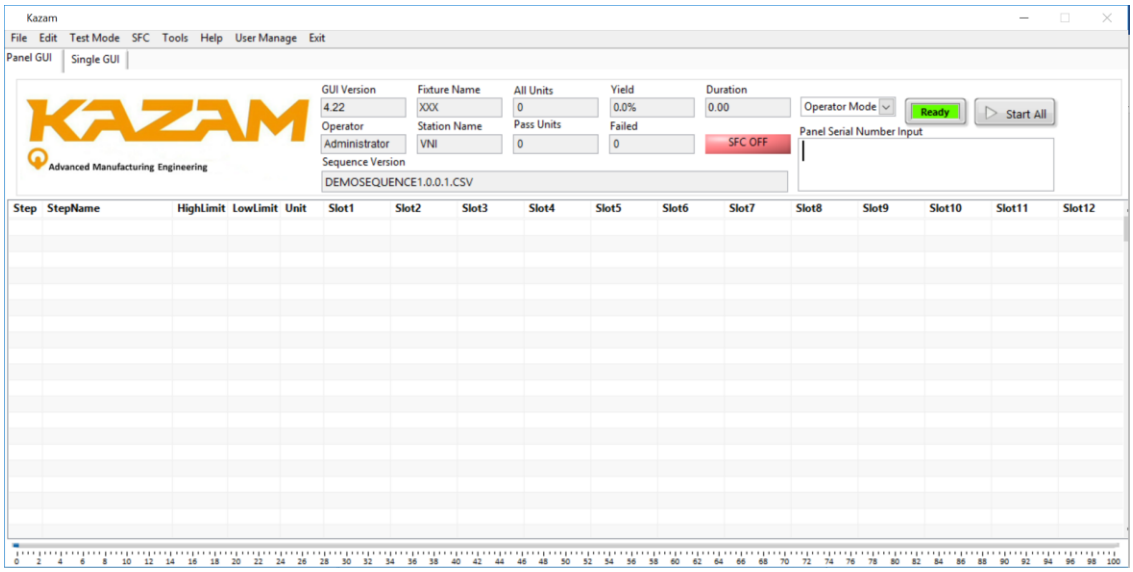


Figure 7

Figure 8 is the panel single GUI, it is used for debugging or maintenance.

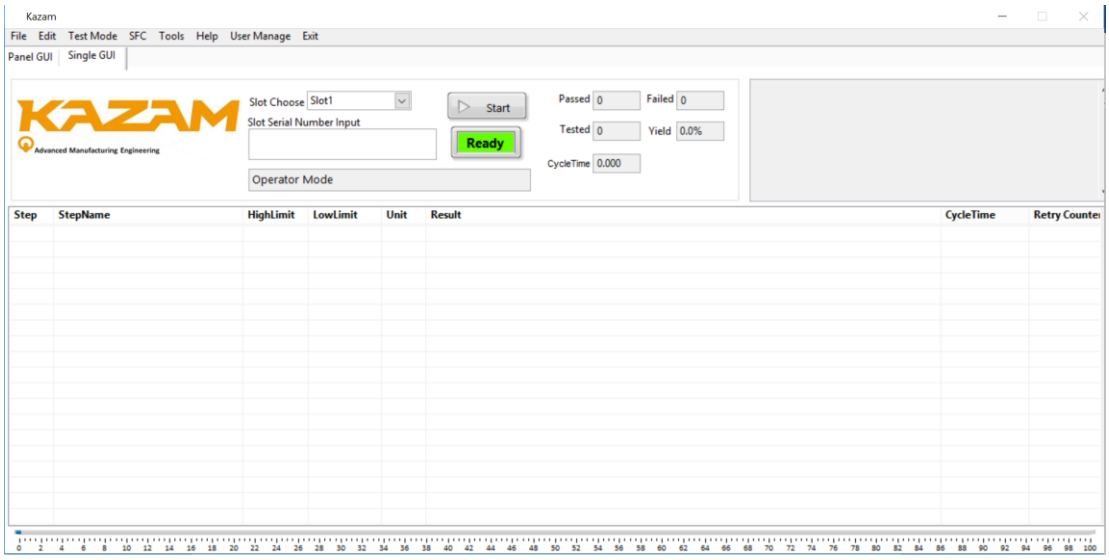


Figure 8

Figure 9 is the multiple single GUI, it is used when each slot needs to run independently.

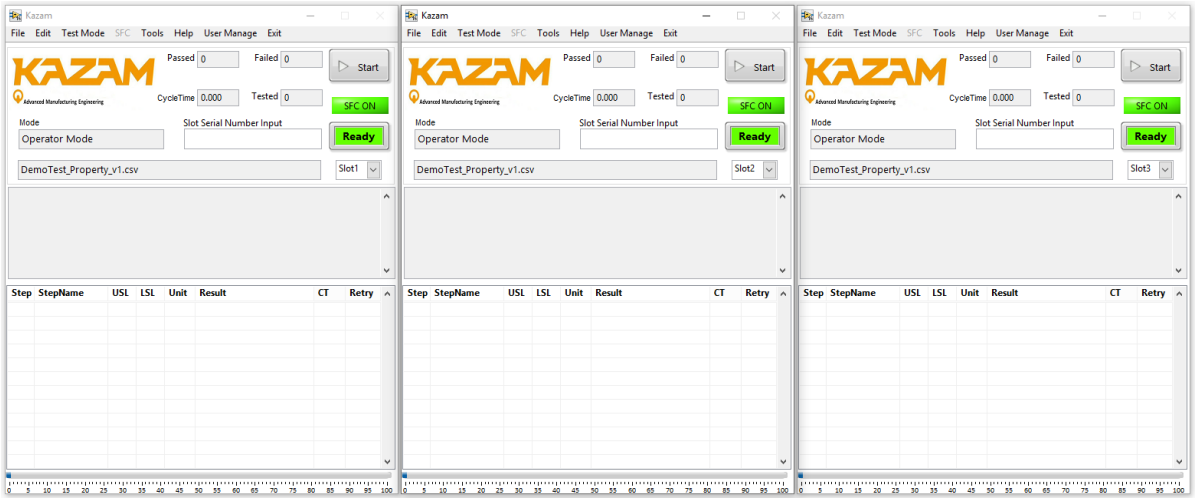


Figure 9

Figure 10 is the actuator GUI, it can be used when multiple actuators run in parallel and all the results of the actuators are displayed at the same time. And the developer can show their front panel of driver vi or test vi on the actuator GUI, the maximum size displayed is 760x600. More details please reference 6.2.5 TestPlan and Hardware configuration

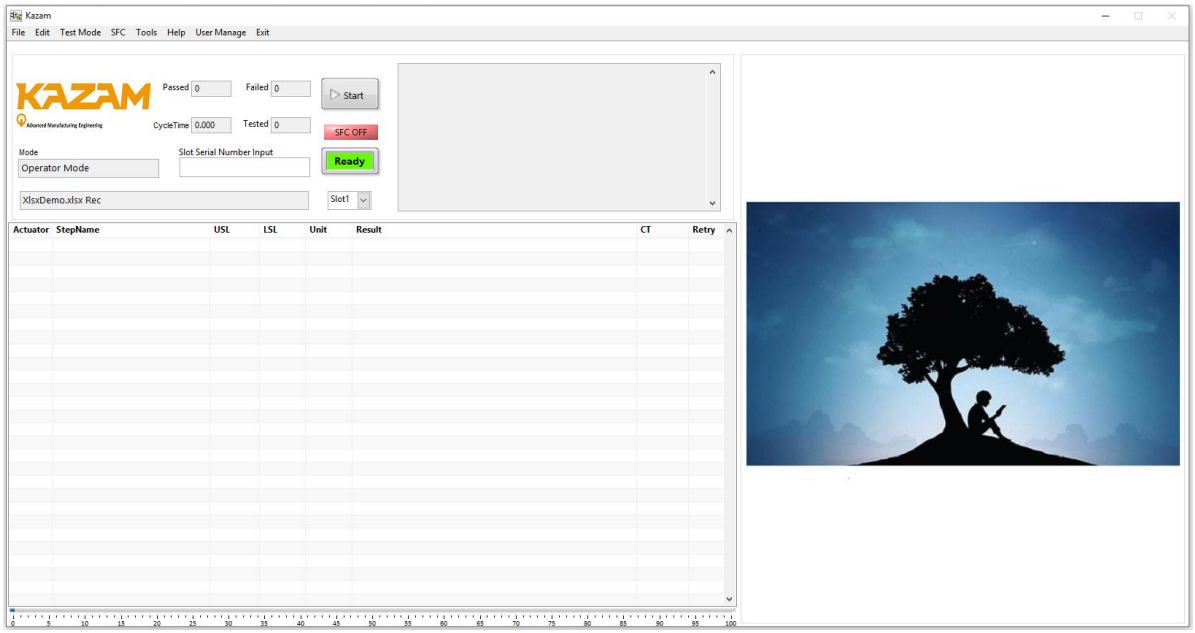


Figure 10

Figure 11 is the multiple single GUI, it is used when more than six slots are running independently.

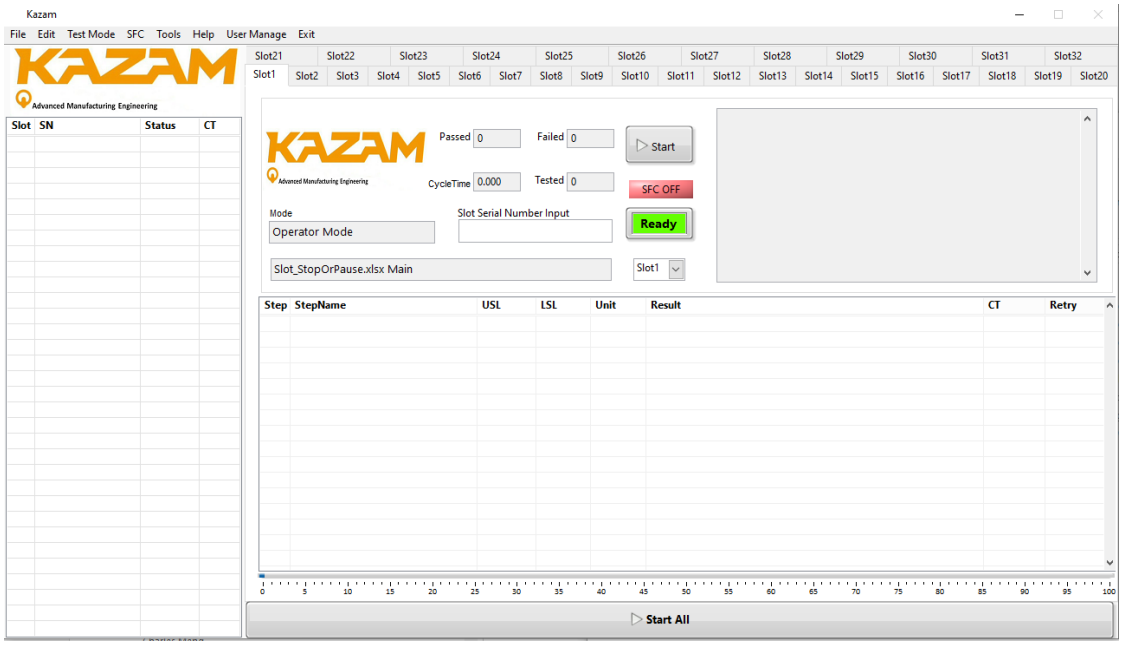


Figure 11

All GUIs support right-click shortcut menus, users can quickly locate the detailed log of the specified steps using the right shortcut menu in the log window. Figure 12 shows how to use the shortcut menu

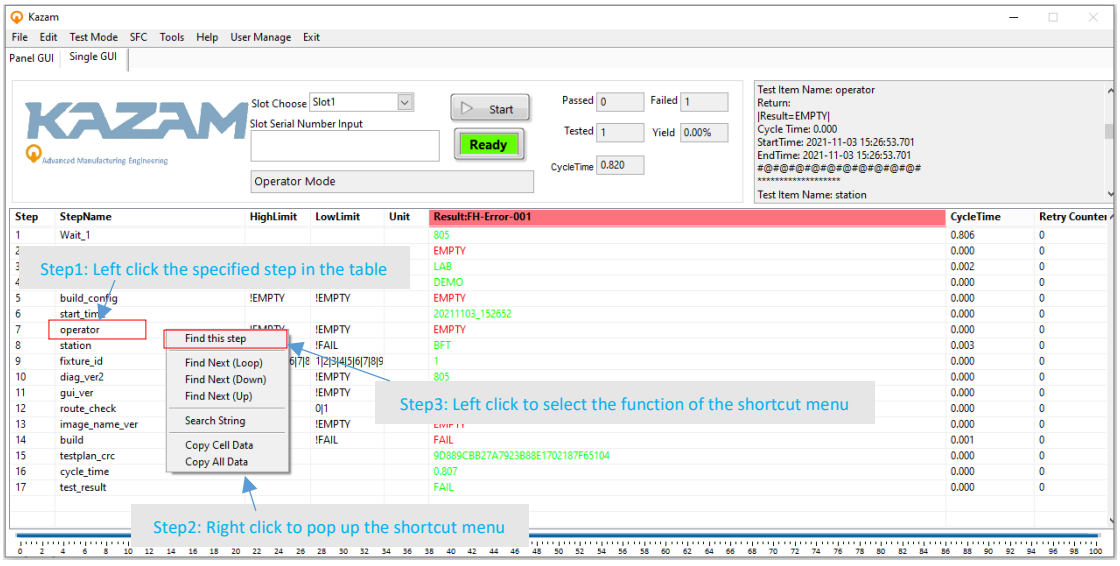


Figure 12

6. How to develop new projects

6.1 Kazam folder

We will release the Shell folder to supplier or CM, as Figure 13, it will have the following file or folder.

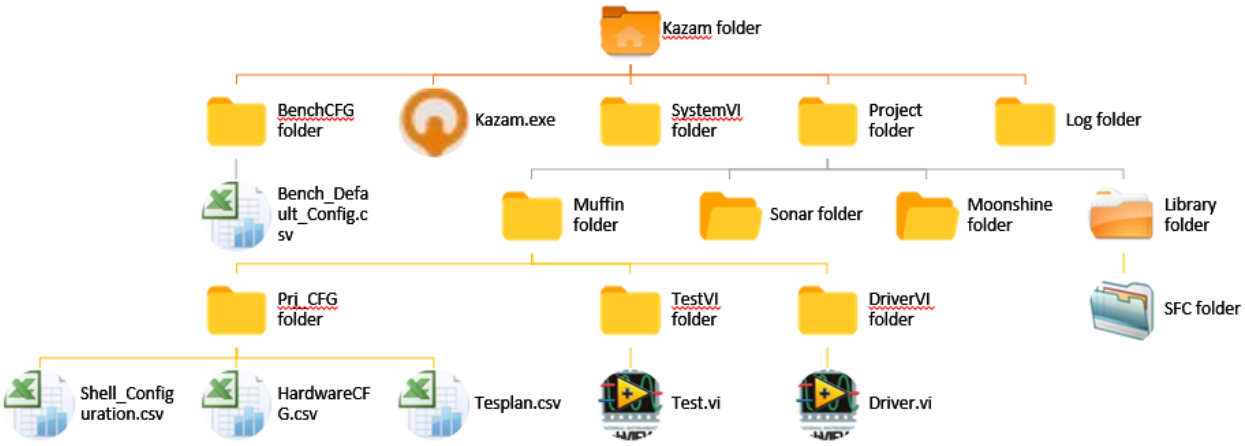


Figure 13

BenchCFG folder: it includes the bench configuration files.

LOG folder: it includes all the log files

SystemVI folder: it includes all the system vi files

Project folder: it includes all the project folder, each project have one folder to manager the vi, TestPlan, shell configuration file and hardware configuration file. And there is library folder in project folder, it is use for common library vi, exe and dll.

SFC folder: it includes all the SFC vi, the developer must to put all the SFC vi into this folder.

Kazam.exe: it is the platform's start-up entrance, double click to start our software

6.2 Project folder

We will release the project folder to developer, it as the Figure 14. In this project folder, it will including the Labview project, the developer can follow the project to develop test vi and driver vi.

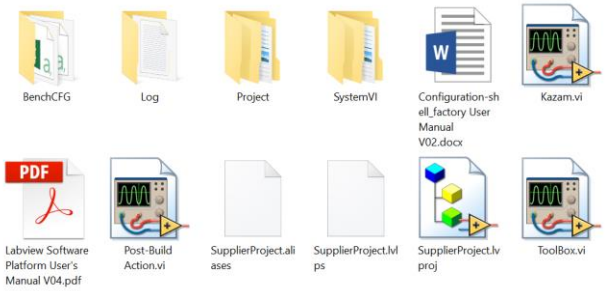


Figure 14

Figure 15 is the LabVIEW project, after the developers complete development, must compile the files.

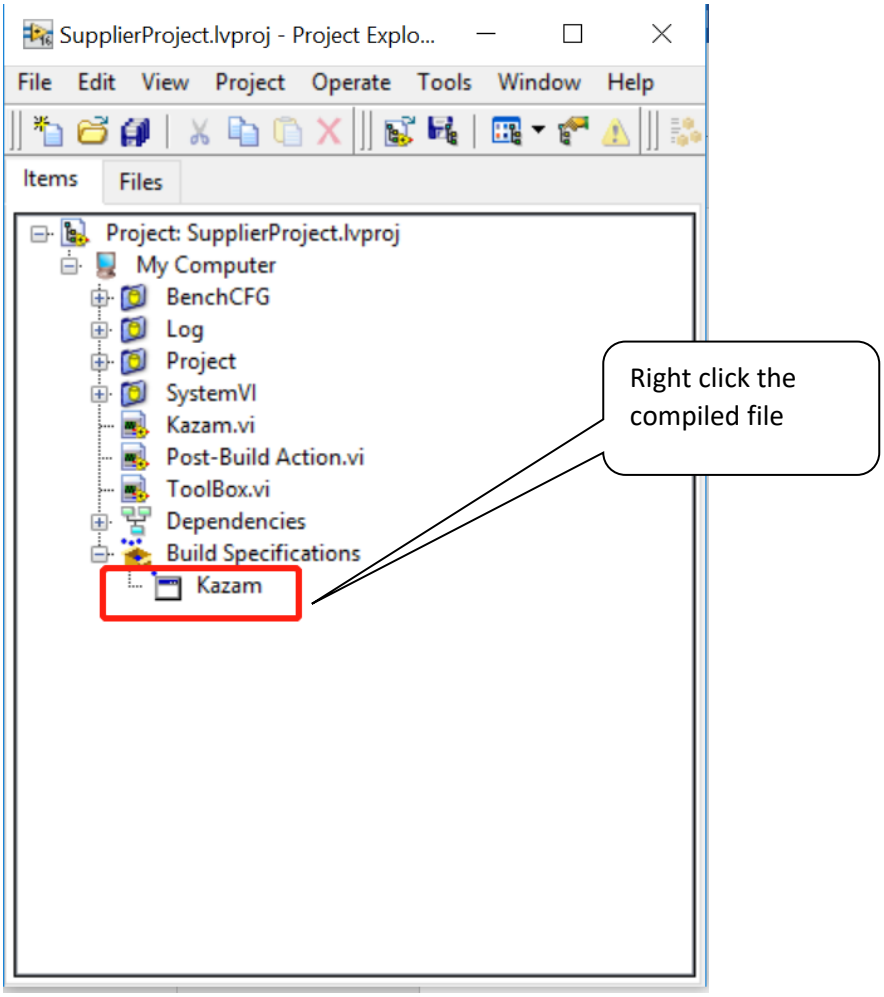


Figure 15

6.2.1 Driver VI

In the DriverVI folder, there is the driver vi template, its name is DRV_template.vi, its from panel as Figure 16. The developer can change or modify the front panel.

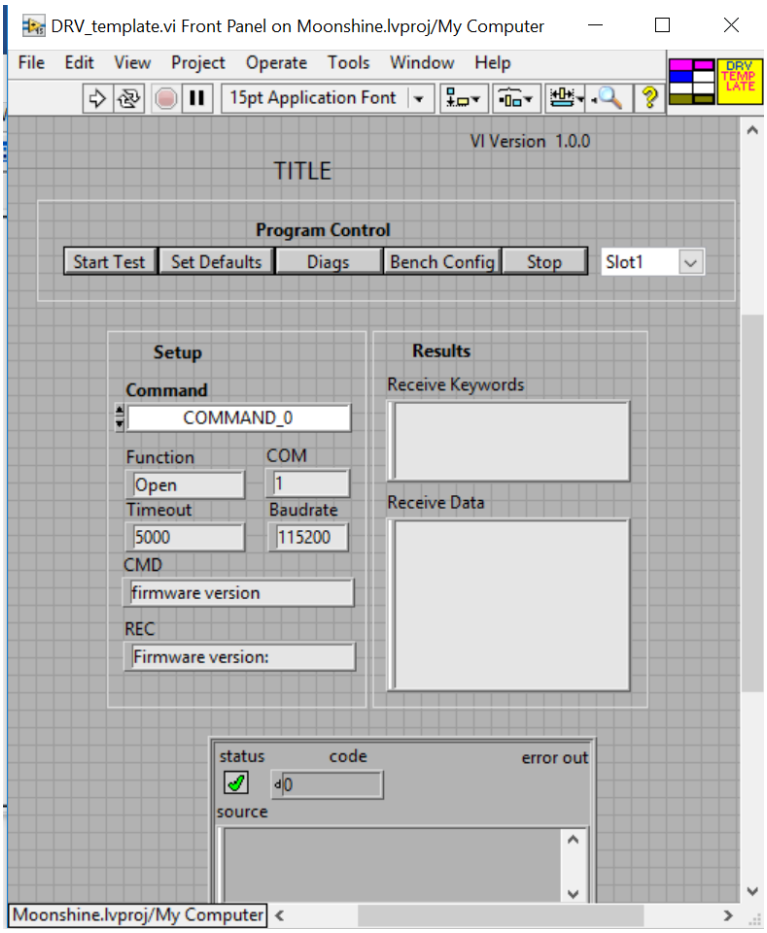


Figure 16

Figure 17 is the back panel of the DRV_template.vi, the developer can add their code in the “TEST” case or modify the template’s code.

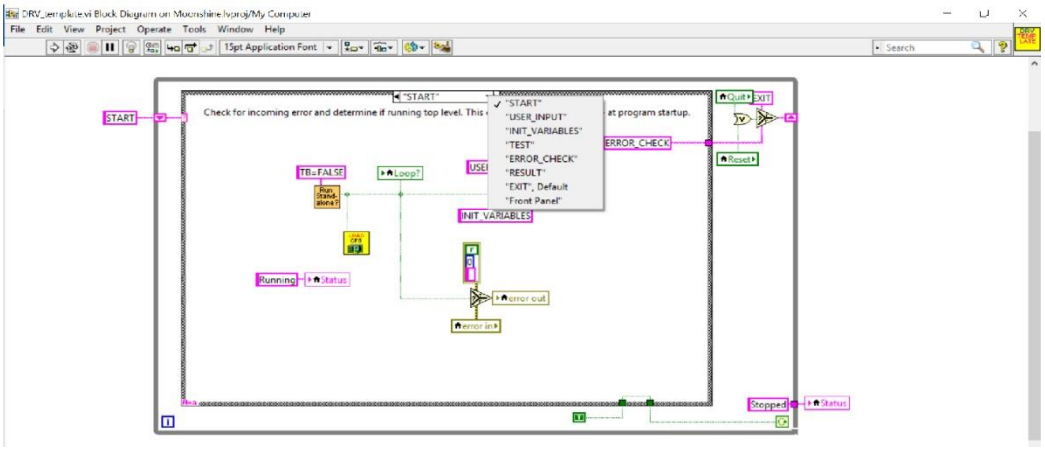


Figure 17

Figure 18 is the input and output interface of the DRV_template.vi, it is the default interface and cannot be changed.

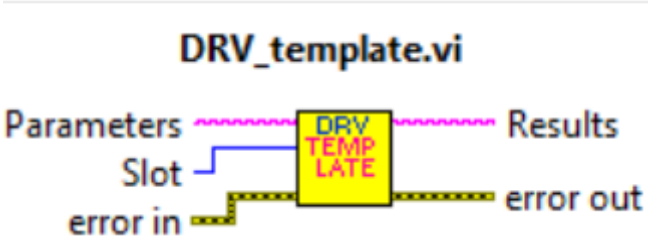


Figure 18

6.2.2 Test VI

In the TestVI folder, there is the test vi template, its name is TST_template.vi, its from panel as Figure 19. The developer can change or modify the front panel.

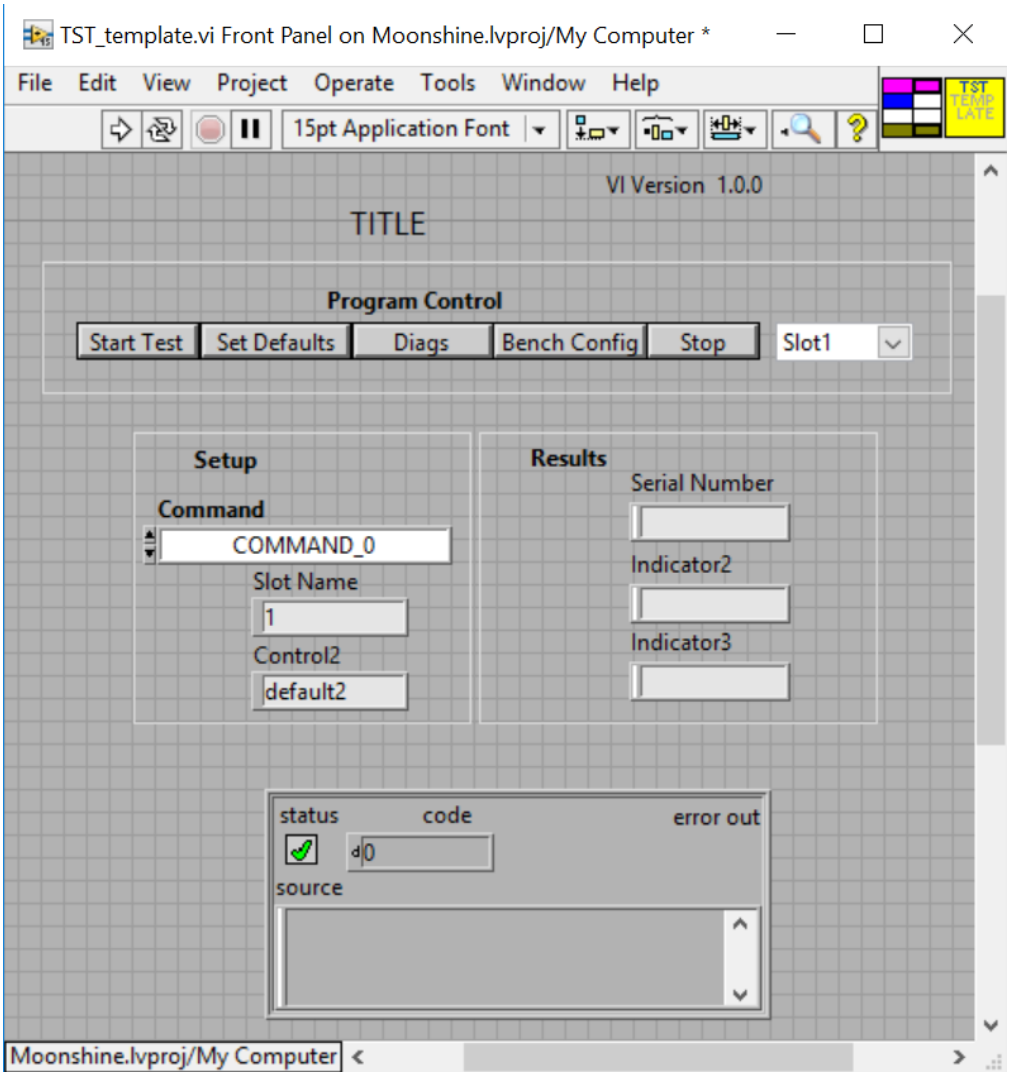


Figure 19

Figure 20 is the back panel of the TST_template.vi, the developer can add their code in the “TEST” case or modify the template’s code.

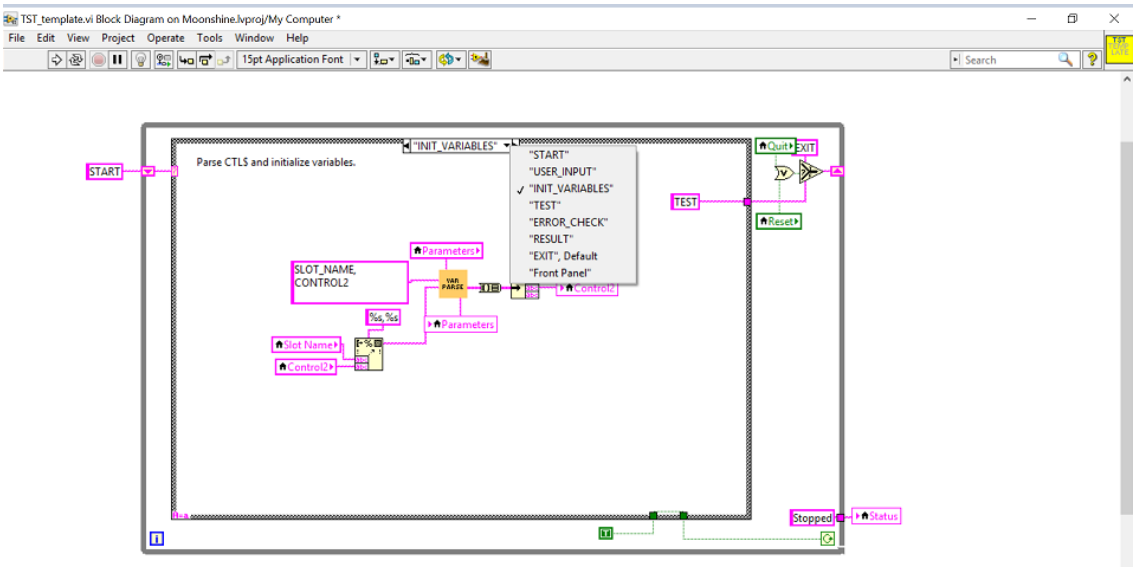


Figure 20

Figure 21 is the input and output interface of the TST_template.vi, it is the default interface and can not be changed.

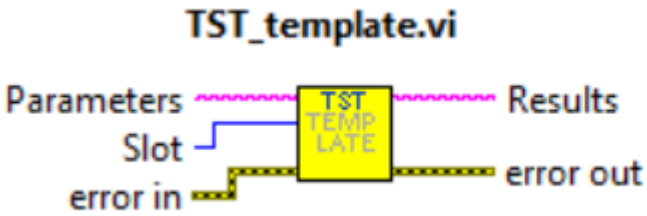


Figure 21

6.2.3 Log VI

In the SystemVI folder, there is the log template, its name is log_Template.vi, and there are two global variables vi, as Figure 22.

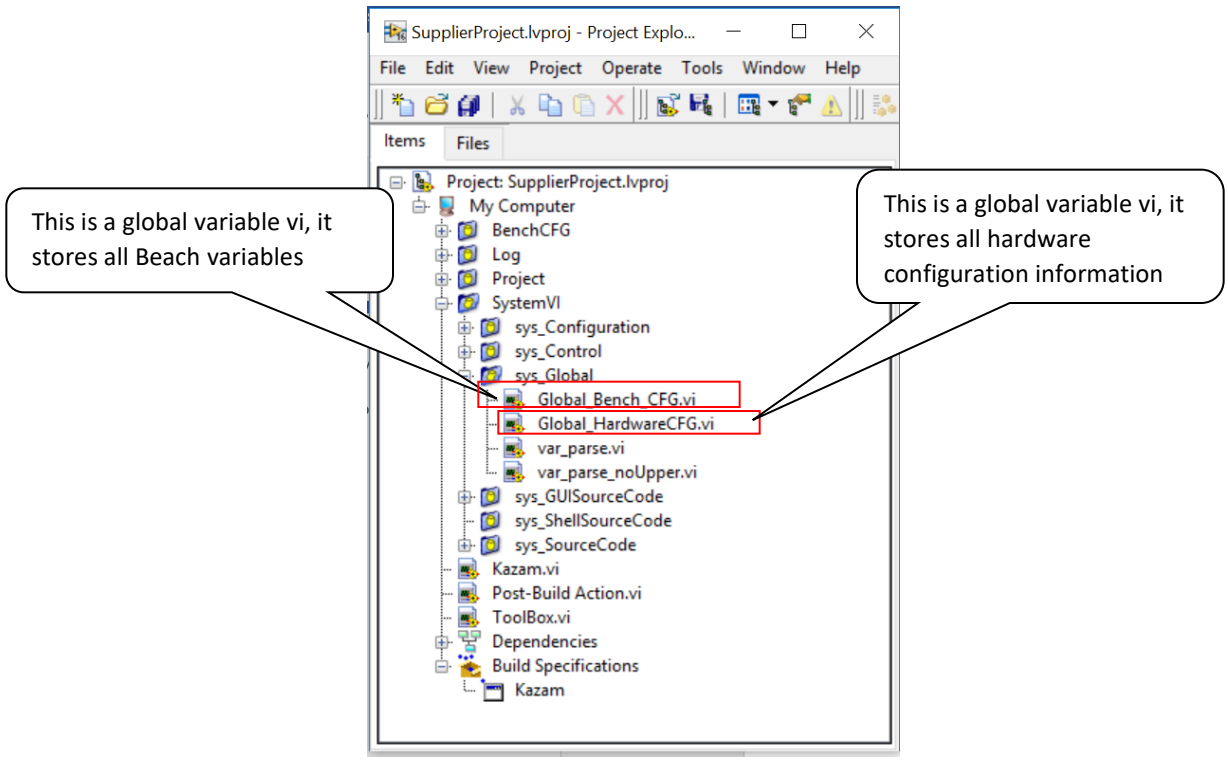


Figure 22

Figure 23 to Figure 29 is the back panel of the log_template.vi, here are some introduction, the developers can customize special log files according to their needs.

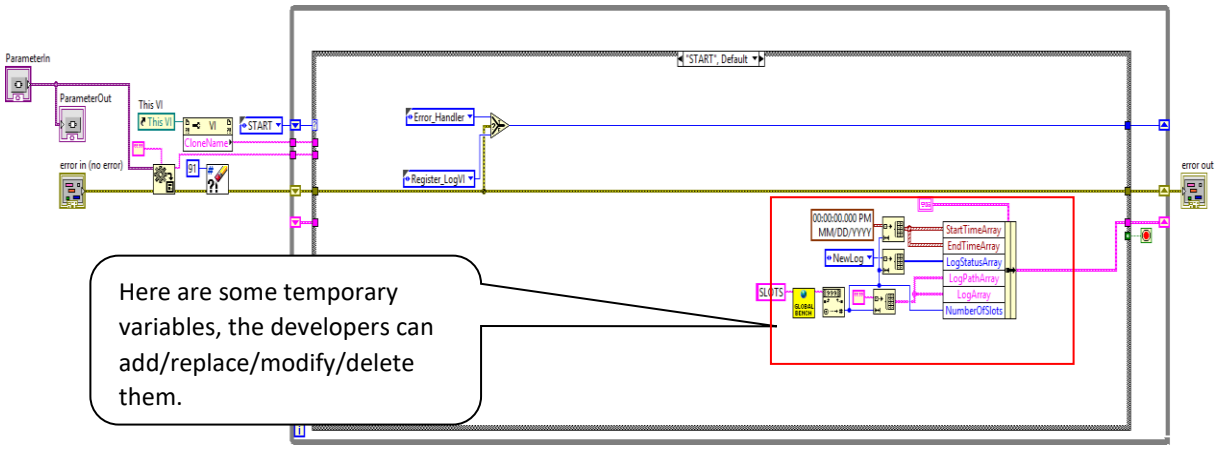


Figure 23

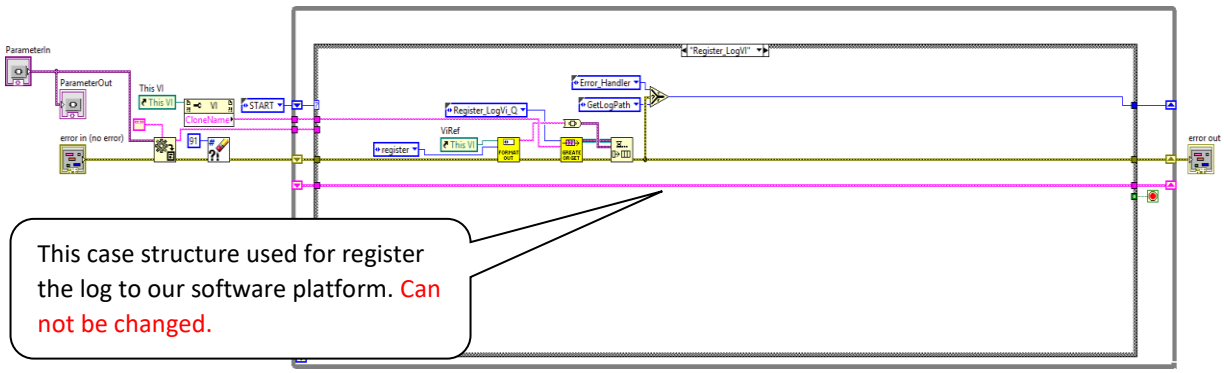


Figure 24

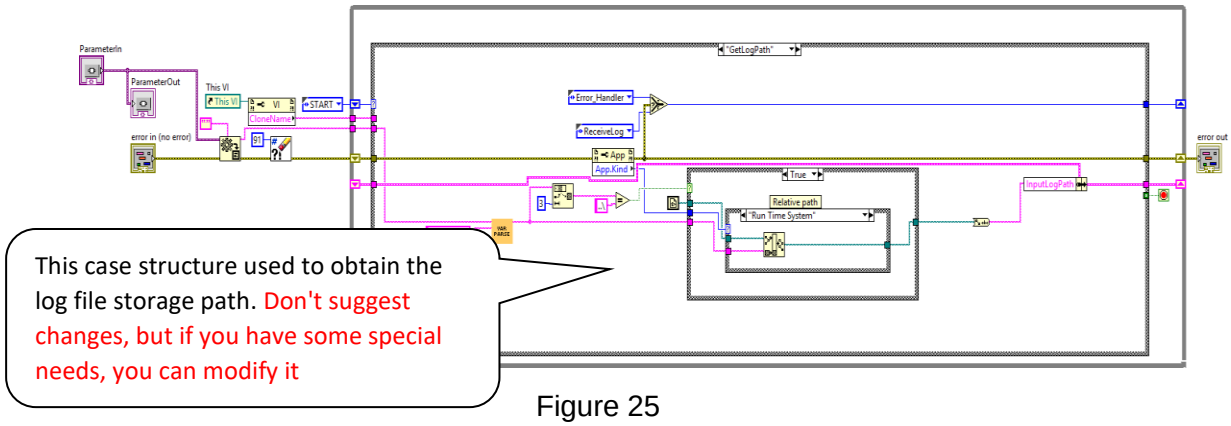


Figure 25

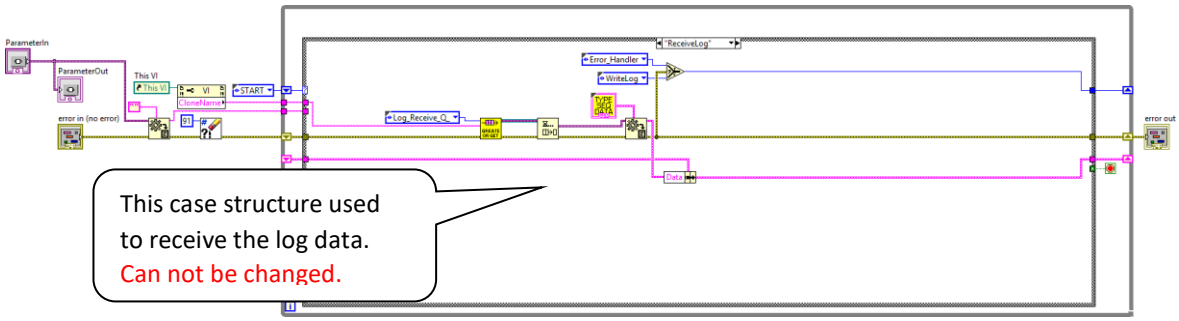


Figure 26

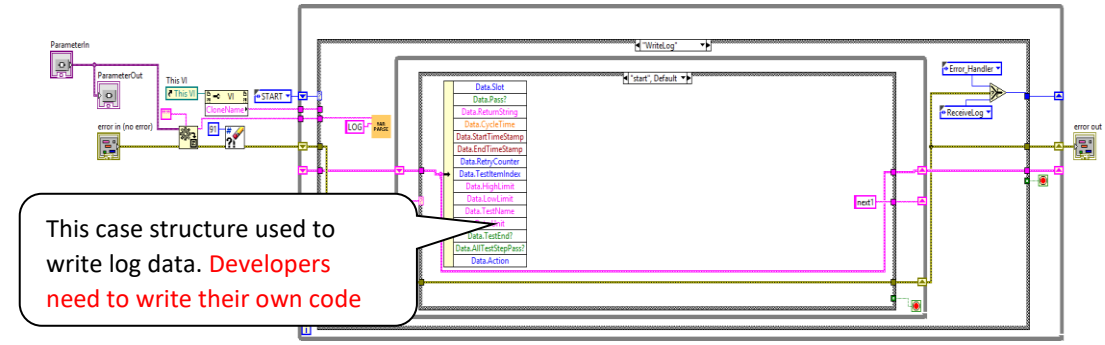


Figure 27

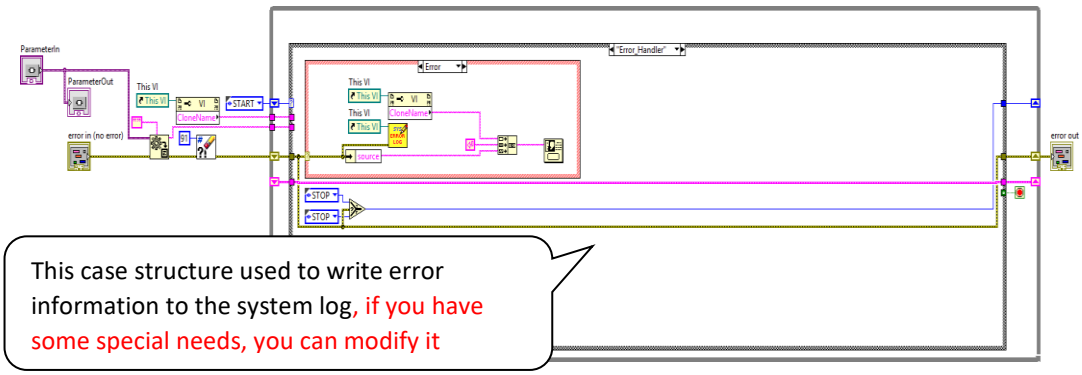


Figure 28

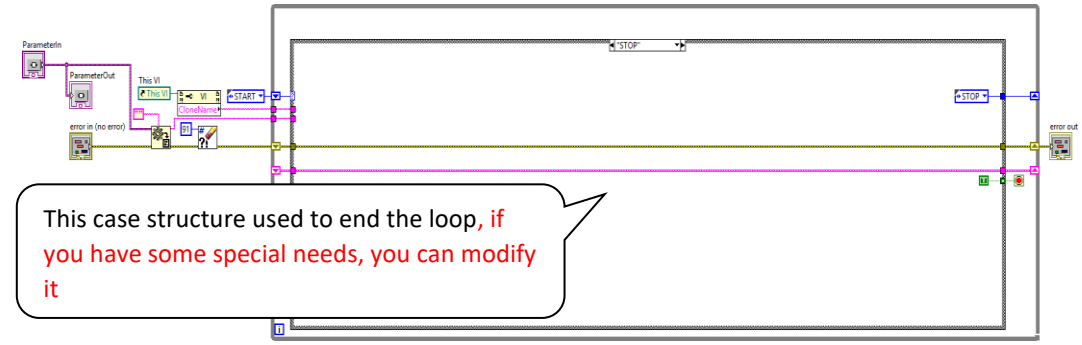


Figure 29

6.2.4 System configuration

In the BenchCFG folder, here are two files, as Figure 30.

Bench_Default_Config.csv, it is used for storing bench information, please do not rename it. And there are two kinds of Bench_Default_Config.csv files, one is used to start Kazam.exe from local PC, the other is used to start Kazam.exe from the server

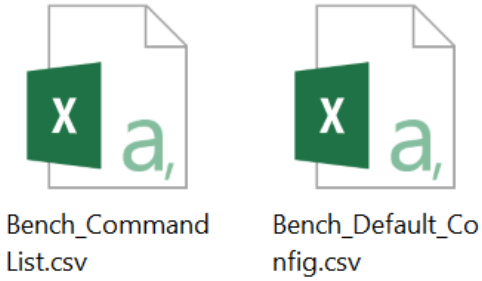
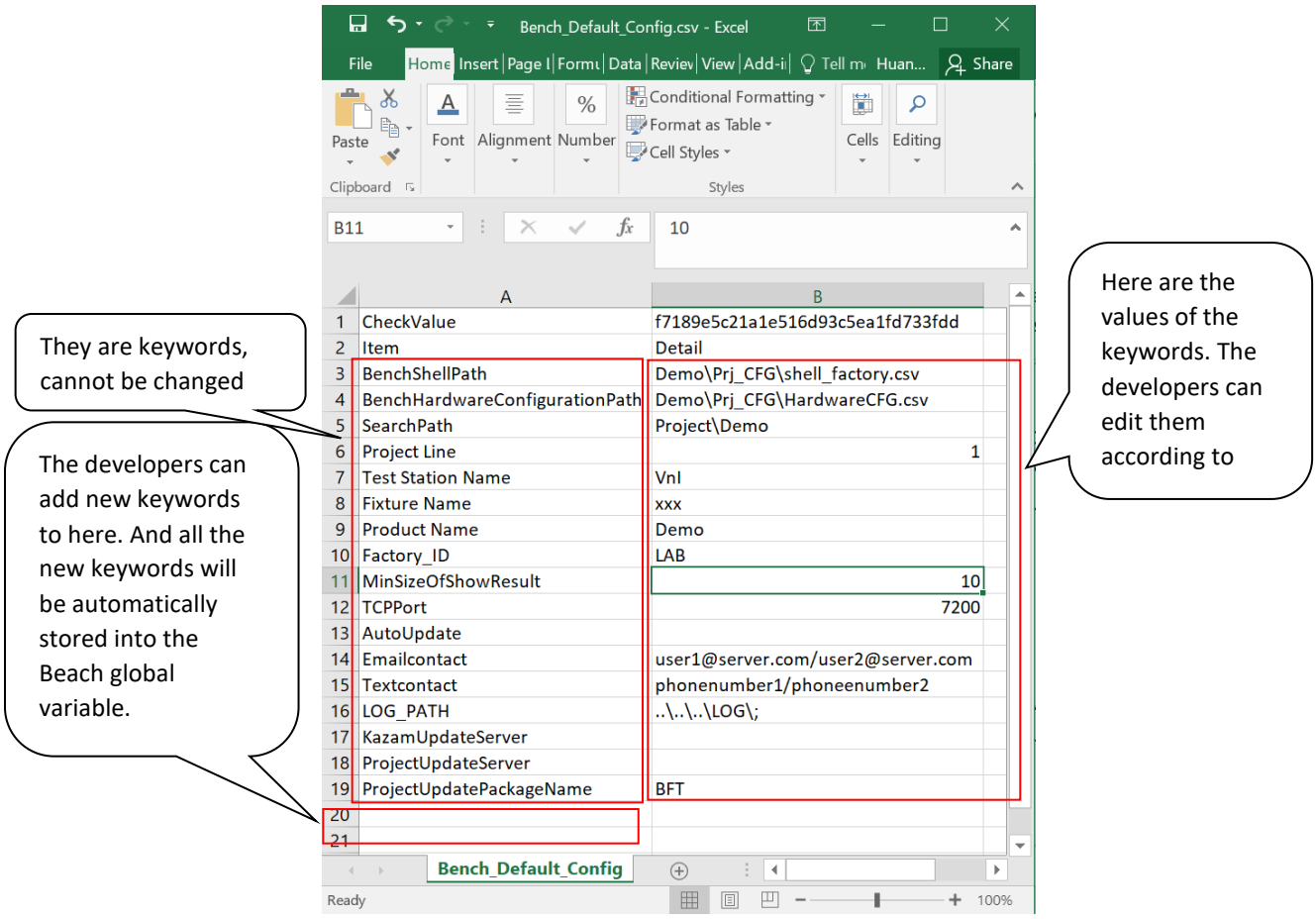


Figure 30

Figure 31 is the bench configuration file, it is used to start Kazam.exe from local PC, in this file, have some keywords, they can not be changed.



	A	B
1	CheckValue	f7189e5c21a1e516d93c5ea1fd733fdd
2	Item	Detail
3	BenchShellPath	Demo\Prj_CFG\shell_factory.csv
4	BenchHardwareConfigurationPath	Demo\Prj_CFG\HardwareCFG.csv
5	SearchPath	Project\Demo
6	Project Line	1
7	Test Station Name	VnI
8	Fixture Name	xxx
9	Product Name	Demo
10	Factory_ID	LAB
11	MinSizeOfShowResult	10
12	TCPPort	7200
13	AutoUpdate	
14	Emailcontact	user1@server.com/user2@server.com
15	Textcontact	phonenumber1/phoneenumber2
16	LOG_PATH	..\..\..\LOG\;
17	KazamUpdateServer	
18	ProjectUpdateServer	
19	ProjectUpdatePackageName	BFT
20		
21		

Figure 31

The value of “BenchShellPath” keyword is “Demo\Prj_CFG\shell_factory.csv”, it means that the Kazam.exe will run the “..\Project\ Demo\Prj_CFG\shell_factory.csv” file first.

Figure 32 is another bench configuration file, it is used to start Kazam.exe from the server, the developers can use this format of bench configuration file to centralize management of Kazam. It can use computer name or MAC address to specify the BenchCFG parameters of Kazam for each Kazam’s shortcut. If there are several network cards on a PC, the developer can use any MAC address or the computer name to specify BenchCFG parameter

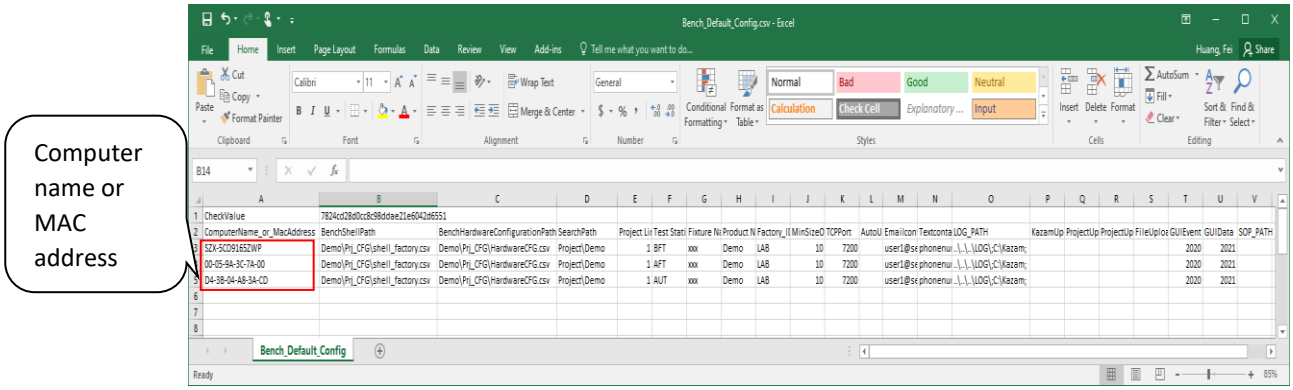


Figure 32

In the shell_factory.csv file, the developers can add/replace/reduce/modify the function of the software platform.

Note: shell_factory.csv, it is the configuration file of Kazam.exe, the developers need to edit it according to their own needs.

As Figure 33, it is a demo shell configuration, it has four log functions, one GUI and it redefine the number of the slot. Here's the details of this script:

Resource_Name: Sequence_config

VI_NAME: sys_SequenceCFGReadout.vi

CONTROL_STRING Keywords: (Red is the keyword, blue is the value of the keyword)

TestPlan= DemoSequence1.0.0.1.csv

WaitUntilDone=True

This step call the sys_SequenceCFGReadout.vi from the "..\..\..\SystemVI\" path, it will load the TestPlan form the "..\Project\xxx\DemoSequence1.0.0.1.csv" path, this path is configured by the Bench_Default_Config.csv file. The TestPlan name is "DemoSequence1.0.0.1.csv", and here is a keyword, it is called "TestPlan".

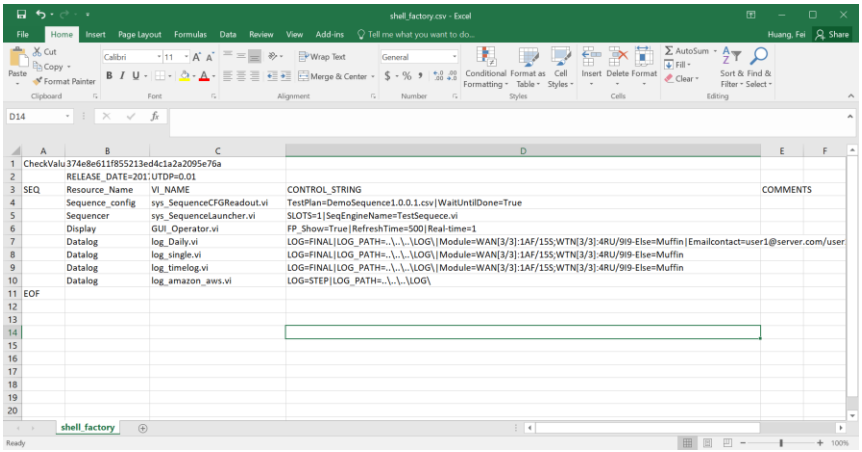
"TestPlan" is the default keyword for all slots, if define "TestPlan1= Seq_Slot1.csv", it means that slot 1 will use this "Seq_Slot1.csv" sequence.

If define "TestPlan2= Seq_Slot2.csv", it means that slot 2 will use this "Seq_Slot2.csv" sequence.

And so on.

For example:

TestPlan= DemoSequence1.0.0.1.csv| TestPlan1= Seq_Slot1.csv| TestPlan2= Seq_Slot2.csv



	A	B	C	D	E	F
1	CheckValu	374e8e611f855213ed4c1a2a2095e76a				
2		RELEASE_DATE=201;UTDP=0.01				
3	SEQ	Resource_Name	VI_NAME	CONTROL_STRING	COMMENTS	
4		Sequence_config	sys_SequenceCFGReadout.vi	TestPlan=DemoSequence1.0.0.1.csv|WaitUntilDone=True		
5		Sequencer	sys_SequenceLauncher.vi	SLOTS=1|SeqEngineName=TestSequece.vi		
6		Display	GUI_Operator.vi	FP_Show=True|RefreshTime=500|Real-time=1		
7		Datalog	log_Daily.vi	LOG=FINAL|LOG_PATH=..\..\..\LOG|Module=WAN[3/3]:1AF/15S;WTN[3/3]:4RU/9I9-Else=Muffin|Emailcontact=user1@server.com/user		
8		Datalog	log_single.vi	LOG=FINAL|LOG_PATH=..\..\..\LOG|Module=WAN[3/3]:1AF/15S;WTN[3/3]:4RU/9I9-Else=Muffin		
9		Datalog	log_timelog.vi	LOG=FINAL|LOG_PATH=..\..\..\LOG|Module=WAN[3/3]:1AF/15S;WTN[3/3]:4RU/9I9-Else=Muffin		
10		Datalog	log_amazon_aws.vi	LOG=STEP|LOG_PATH=..\..\..\LOG\		
11	EOF					

Figure 33

6.2.5 TestPlan and Hardware configuration

In the Prj_CFG folder, here are test plan, Kazam configuration and hardware configuration, the developers can store different TestPlan in this folder. As Figure 34, here are five TestPlans and one hardware configuration file.

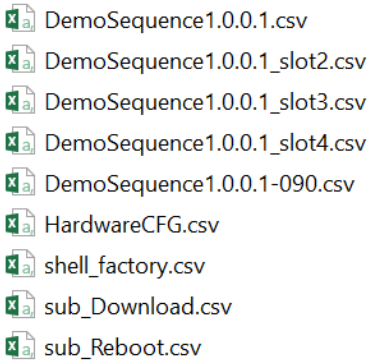


Figure 34

6.2.5.1 TestPlan

Kazam can support loading XLSX files as TestPlan, the format is as below:

TestPlan=Demo.xlsx Abc Abc is the tab name of xlsx file, the tag names are case-sensitive

Figure 35 is a demo TestPlan, there have some keywords, and the following is a detailed explanation:

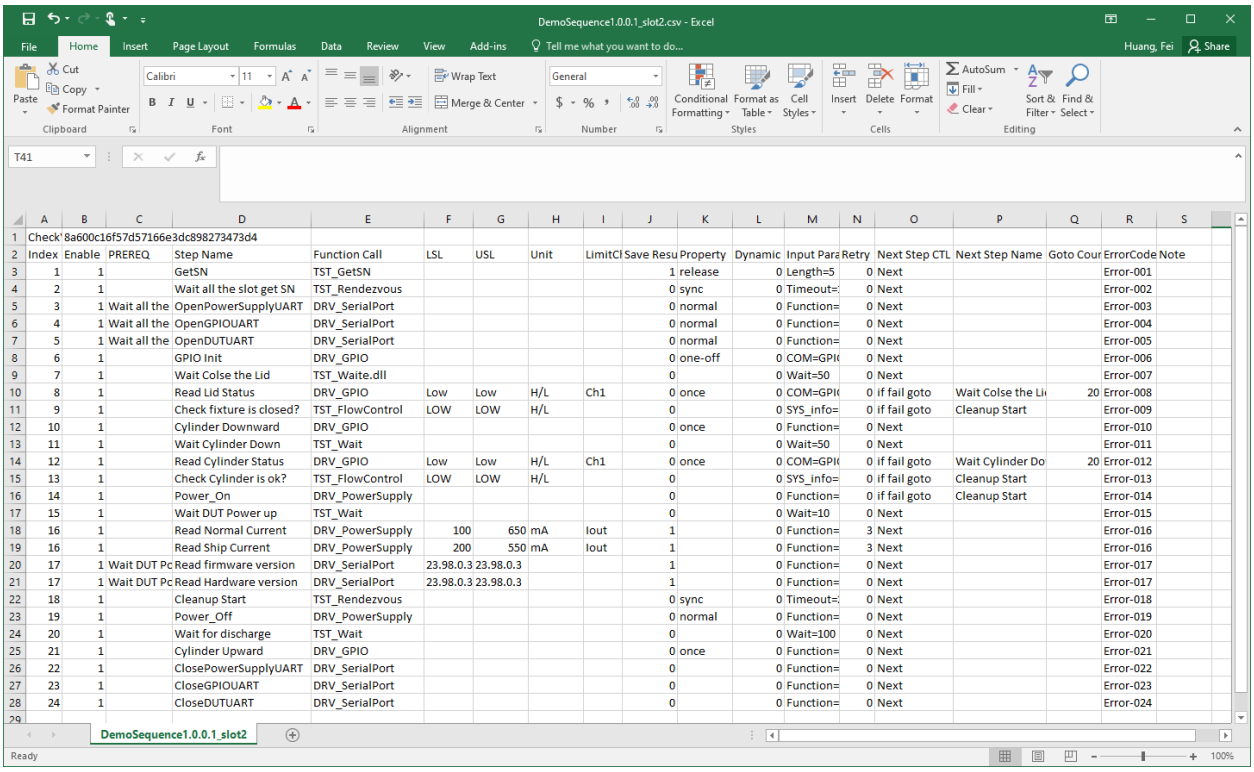
Keyword	Description	Example
Index	Step index, it is an integer range 1 to infinity F is a keyword for fixed header sequence, for example F1,F2,F3……	
Enable	It controls whether running this test step, its value is 0 , 1 ,SFC=ON, SFC=OFF, Loop=ON and Loop=OFF, SFC, LOOP, SFCOFF, LOOPOFF If the value is 1, the step will be running If the value is 0, the step will not be running If the value is SFC=ON, when the SFC is OFF, the step will not be running If the value is SFC=OFF, when the SFC is ON, the step will not be running If the value is Loop=ON, When Kazam runs in loop mode, the step will be running If the value is Loop=OFF, When Kazam runs in operator mode, the step will be running If the value is SFC, when the SFC is OFF, the step will not be running If the value is SFCOFF, when the SFC is ON, the step will not be running If the value is LOOP, When Kazam runs in loop mode, the step will be running If the value is LOOPOFF, When Kazam runs in operator mode, the step will be running "|" can be used as or condition And the developers can customize enabling parameters in Enable column in Testplan file, and modify its default status in the shell configuration file with the EnableParameter keyword. Please refer to the parameter description of sys_sequencecfgreadout.vi in section 8.2.2 Other parameter	SFC=ON|Loop=OFF SFC=OFF|Loop=OFF SFC=ON|Loop=ON SFC=OFF|Loop=ON Loop=ON Loop=OFF SFC=ON SFC=OFF SFC|LOOPOFF|HVT HVT is the developer's own definition
PREREQ	The prerequisite column is used to set the steps to run in parallel. Its value is the name of the previous step of this step	The prerequisite of “OpenPowerSupplyUART” is “Wait all the slot get SN”
RunInParallel	Its value is 0, 1 or empty. Empty and 0 mean that this test step is a blocking step. 1 means that this test step is a non-blocking	

	step and when using this feature, process control syntax only "Next" is valid. This step will launch, and continue with the next step without waiting for a return.	
Step Name	It is the name of this step. All the test name must be unique	GetSN
Function Call	The platform can call dynamic link libraries, executables, python script, batch file and C# source code directly If there is no suffix, it means that will call VI; If this is empty, the result of the previous step is automatically retrieved.	TST_Wait TST_Wait.dll -------6.5.1 EXE/Python/Bat file support TST_Wait.exe -------6.5.1 EXE/Python/Bat file support TST_Wait.py -------6.5.1 EXE/Python/Bat file support TST_Wait.pyc -------6.5.1 EXE/Python/Bat file support TST_Wait.bat -------6.5.1 EXE/Python/Bat file support TST_Wait.cs -------6.5.3 C# source code support
LSL	Low limit, its value can be decimal, hexadecimal, float and string If only one number is defined in LSL, it means that the test result needs to be greater than or equal to LSL If LSL is the same as USL, it means that the test results must be the same as USL and LSL "|" can be used as or condition, for example, LSL is "123|456|789", USL is "123|456|789", if the result is 123 or 456 or 789, the test result will pass, otherwise it will fail "!" can be used as Not condition, for example, LSL is "!EMPTY", USL is "!EMPTY", if the result is not empty, the test result will pass, otherwise it will fail, another example, LSL is "!abc", USL is "!abc", if the result is not “abc”, the test result will pass, otherwise it will fail, Note: when using OR and NOT expressions, the values of USL and LSL must be the same	
USL	High limit, its value can be decimal, hexadecimal, float and string If only one number is defined in USL, it means that the test result should be less than or equal to USL If USL is the same as LSL, it means that the test results must be the same as USL and LSL "|" can be used as or condition, for example, LSL is "123|456|789", USL is "123|456|789", if the result is 123 or 456 or 789, the test result will pass, otherwise it will fail Note: when using or expressions, the values of USL and LSL must be the same	
Unit	The limit of the unit, it will be displayed in the GUI	
LimitCheck	If the step has multiple results, this keyword will tell the platform which result is you needs.	If the multiple result is “Vout=3.8|Iout=110”, the “LimitCheck” is “Vout”,it means that this step result is “Result=3.8”

Save Result	If the value is 1, the result of this step will be stored in the daily log; If the value is 0 or empty, the result of this step will not be stored in the daily log;	
Casesensitive	If the value is "Yes", the result output and global variable will be case sensitive If the value is empty or "NO", the result output and global variable will not be case sensitive	
Property	➢ Property=normal, it means that when this test vi is called, this test vi will run in parallel ➢ Property=Sync, it means that when this test vi is called, this test vi will wait for the other test VI and then start together. ➢ Property=One-off, it means that this test vi is executed only once, this property commonly used to initialize instrument ➢ Property=Share, it means that this test vi is executed one by one, this property commonly used to share resource, such as instrument ➢ Property=Release, it means that when this test vi is called, one of this test vi will run and get some information and then send the information to other slots, such as get the panel SN ➢ Property=Once, it means that one of this clone test vi is executed, the other clone will receive the result, this property commonly used to control fixture's lid or other. ➢ Property= Transmit, it means that when this slot's vi runs complete, this slot's vi will send some information to other slot, such as a start signal. The receive step name must be specified in the "input parameter" for this step. The keyword is "REC_Step", for example, REC_Step=Slot2WaitForStart;Slot2WaitForStart; Slot2WaitForStart and Slot2WaitForStart are the test step name ➢ Property= Receive, it means that before this VI runs, it will wait for information from the transmitter VI.	
Dynamic	If the value is 1, this test step will be dynamic call; If the value is 0 or empty, this test step will be preload;	
Input Parameter	The parameter of this step. The format is as below: Keyword=value "|" is the delimiter	Function=Open|COM=PSCOM|Baudrate=115200|ShowFP=YES

	The developer can use "ShowFP" keyword to show their front panel of driver vi or test vi on the actuator GUI, the maximum size displayed is 760x600 The value of "ShowFP" is YES or NO, its default value is NO "Result" keyword is used to assign a value to the current step, for example, "Result=123", it means that the result of this step is 123, another example, "Result=<<Slots_info.SN>>", it means that the result of this step is the value of Slots_info.SN	
Retry	If the value is 0 or empty, this step is failed, this step does not retry; If the value is greater than or equal to 1, this step is failed, this step will retry;	
Next Step CTL	➢ Next, it means that when this test step is completed, the next will be the next step ➢ if fail goto, it means that when this test step is failed, the next will be run the specified step ➢ if pass goto, it means that when this test step is passed, the next will be run the specified step ➢ goto, it means that when this test step is completed, the next will be run the specified step ➢ force pass, it means that when this test step is completed, forced this step pass ➢ force fail, it means that when this test step is completed, forced this step fail ➢ Stop if fail, it means that when this test step is failed, the test sequence will be stopped ➢ If…else, it means that this step is not a test item, its result will not record to the final result, it is a true or false step. ➢ Switch...case, it means that this step is not a test item, its result will return the next step name, and it is a true or false step. When Kazam receives the next step name, kazam will jump to this step ➢ Pause Slot, it means that after running this step, the specified slot is paused, use SlotNum to set the specified slot in the "Input Parameter" ➢ Continue Slot, it means that after running this step, the specified slot will continue to run, use SlotNum to set the specified slot in the "Input Parameter" ➢ Stop Slot, it means that after running this step, the specified slot will stop running, use SlotNum to set the specified slot in the "Input Parameter"	SlotNum=1;2;3 , it's means you specify slot 1, slot2 and slot3 SlotNum=Other, it's going to pause/continue/stop all the other slots except for itself SlotNum=All, it's going to pause/continue/stop all the slots
Next Step Name	The name of the next step which will be running after current step There are three kind of expressions:	When we use the "If fail goto" to control the step flow, Kazam can handle more jump conditions. TestEnd---it means that if this step fail, it will go to "TestEnd"

	➢ Test name of the next step, for example: TestEnd ➢ Jump by number of cycles, for example: RandomNum|Fail:TestEnd|2:GetResult default:Random|Fail:TestEnd|3:GetResult "Fail" and "default" are the keywords "2" and "3" are cycles number ➢ True or false jumps, for example: Read firmware version|cleanup	RandomNum|Fail:TestEnd|3:GetResult---it means that if this step fail, the first time fail will go to "RandomNum", the second time fail will go to "RandomNum", the third time fail will go to "GetResult", if the goto count is reached, this step is still failed, it will go to "TestEnd" default:Random|Fail:TestEnd|3:GetResult---it means that if this step fail, the first time fail will go to "Random", the second time fail will go to "Random", the third time fail will go to "GetResult", if the goto count is reached, this step is still failed, it will go to "TestEnd" When we use the "If…else" to control the step flow, we will use "|" to separate the true step and false step, for example: Read firmware version|cleanup
Goto Counter	The number of goto jump statement If the value is empty, it means that the step will always jump	
ErrorHandle	The developer can define the error handler for each test step The error handling entry format is defined as follows: ErrorCode:Test Step Name If the error handler is empty, the test will be stopped If the ErrorCode is "default", the test will go to the default step, for example: default:TestEnd	0:EH_0|1:EH_1|2:EH_2|3:EH_3|4:EH_4|default:TestEnd
ErrorCode	If this step fails, will be recorded the error code into the log	
Note	Developers can according to their own need to edit some notes	



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Check	8a600c16f57d57166e3dc898273473d4																	
2	Index	Enable	PREREQ	Step Name	Function Call	LSL	USL	Unit	LimitCl	Save Resu	Property	Dynamic	Input Para	Retry	Next Step CTL	Next Step Name	Goto Cour	ErrorCode	Note
3	1	1		GetSN	TST_GetSN					1	release	0	Length=5	0	Next			Error-001	
4	2	1		Wait all the slot get SN	TST_Rendezvous					0	sync	0	Timeout=	0	Next			Error-002	
5	3	1	Wait all the	OpenPowerSupplyUART	DRV_SerialPort					0	normal	0	Function=	0	Next			Error-003	
6	4	1	Wait all the	OpenGPIOUART	DRV_SerialPort					0	normal	0	Function=	0	Next			Error-004	
7	5	1	Wait all the	OpenDUTUART	DRV_SerialPort					0	normal	0	Function=	0	Next			Error-005	
8	6	1		GPIO Init	DRV_GPIO					0	one-off	0	COM=GPI	0	Next			Error-006	
9	7	1		Wait Colse the Lid	TST_Waite.dll					0		0	Wait=50	0	Next			Error-007	
10	8	1		Read Lid Status	DRV_GPIO	Low	Low	H/L	Ch1	0	once	0	COM=GPI	0	if fail goto	Wait Colse the Li	20	Error-008	
11	9	1		Check fixture is closed?	TST_FlowControl	LOW	LOW	H/L		0		0	SYS_info=	0	if fail goto	Cleanup Start		Error-009	
12	10	1		Cylinder Downward	DRV_GPIO					0	once	0	Function=	0	Next			Error-010	
13	11	1		Wait Cylinder Down	TST_Wait					0		0	Wait=50	0	Next			Error-011	
14	12	1		Read Cylinder Status	DRV_GPIO	Low	Low	H/L	Ch1	0	once	0	COM=GPI	0	if fail goto	Wait Cylinder Do	20	Error-012	
15	13	1		Check Cylinder is ok?	TST_FlowControl	LOW	LOW	H/L		0		0	SYS_info=	0	if fail goto	Cleanup Start		Error-013	
16	14	1		Power_On	DRV_PowerSupply					0		0	Function=	0	if fail goto	Cleanup Start		Error-014	
17	15	1		Wait DUT Power up	TST_Wait					0		0	Wait=10	0	Next			Error-015	
18	16	1		Read Normal Current	DRV_PowerSupply	100	650	mA	Iout	1		0	Function=	3	Next			Error-016	
19	16	1		Read Ship Current	DRV_PowerSupply	200	550	mA	Iout	1		0	Function=	3	Next			Error-016	
20	17	1	Wait DUT Pc	Read firmware version	DRV_SerialPort	23.98.0.3	23.98.0.3			1		0	Function=	0	Next			Error-017	
21	17	1	Wait DUT Pc	Read Hardware version	DRV_SerialPort	23.98.0.3	23.98.0.3			1		0	Function=	0	Next			Error-017	
22	18	1		Cleanup Start	TST_Rendezvous					0	sync	0	Timeout=	0	Next			Error-018	
23	19	1		Power_Off	DRV_PowerSupply					0	normal	0	Function=	0	Next			Error-019	
24	20	1		Wait for discharge	TST_Wait					0		0	Wait=100	0	Next			Error-020	
25	21	1		Cylinder Upward	DRV_GPIO					0	once	0	Function=	0	Next			Error-021	
26	22	1		ClosePowerSupplyUART	DRV_SerialPort					0		0	Function=	0	Next			Error-022	
27	23	1		CloseGPIOUART	DRV_SerialPort					0		0	Function=	0	Next			Error-023	
28	24	1		CloseDUTUART	DRV_SerialPort					0		0	Function=	0	Next			Error-024	

Figure 35

6.2.5.2 Sub TestPlan

The developer can define the sub TestPlan to replace the repeat test step. There are two ways to call the sub testplan.

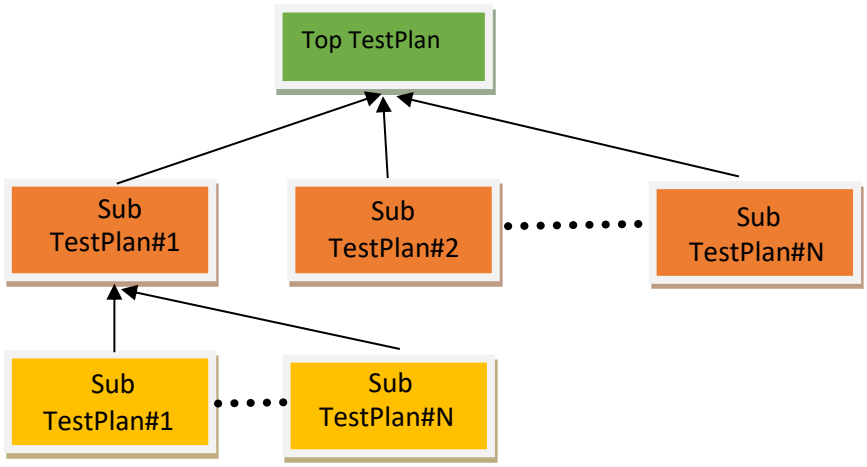


Figure 36

As Figure 36, the developer can use the %xxxxxx% format to tell the platform that it is the sub testplan, xxxxxx is the name of sub tesplan, "%" is the separator.

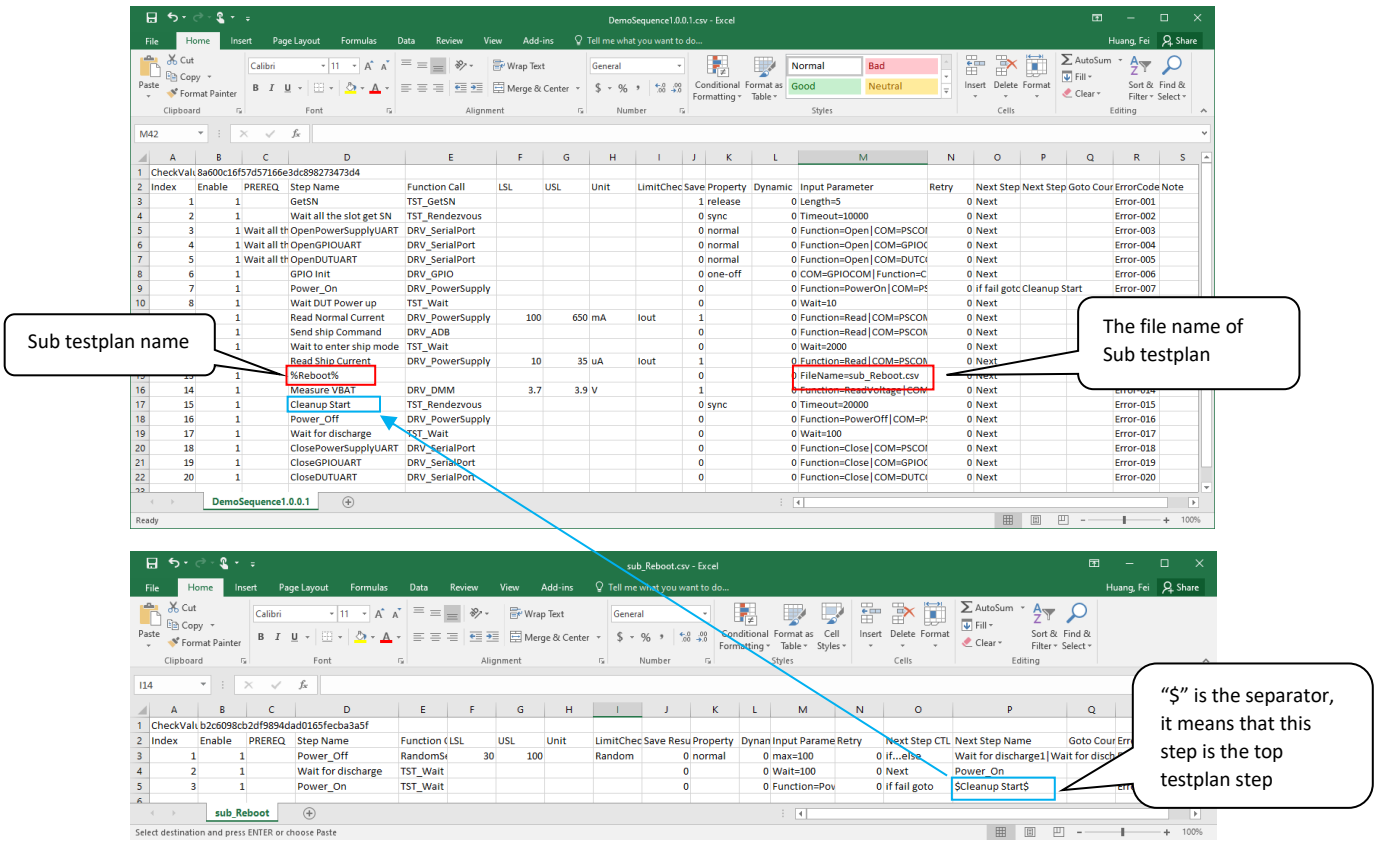


Figure 37

As Figure 38, the sub testplan can be loaded in the ‘Function Call’ field. If the format of sub testplan is xlsx, the developer must specify the name of the sheet to be called. If the format of sub testplan is csv, the developers only need to fill in the name of the calling CSV file

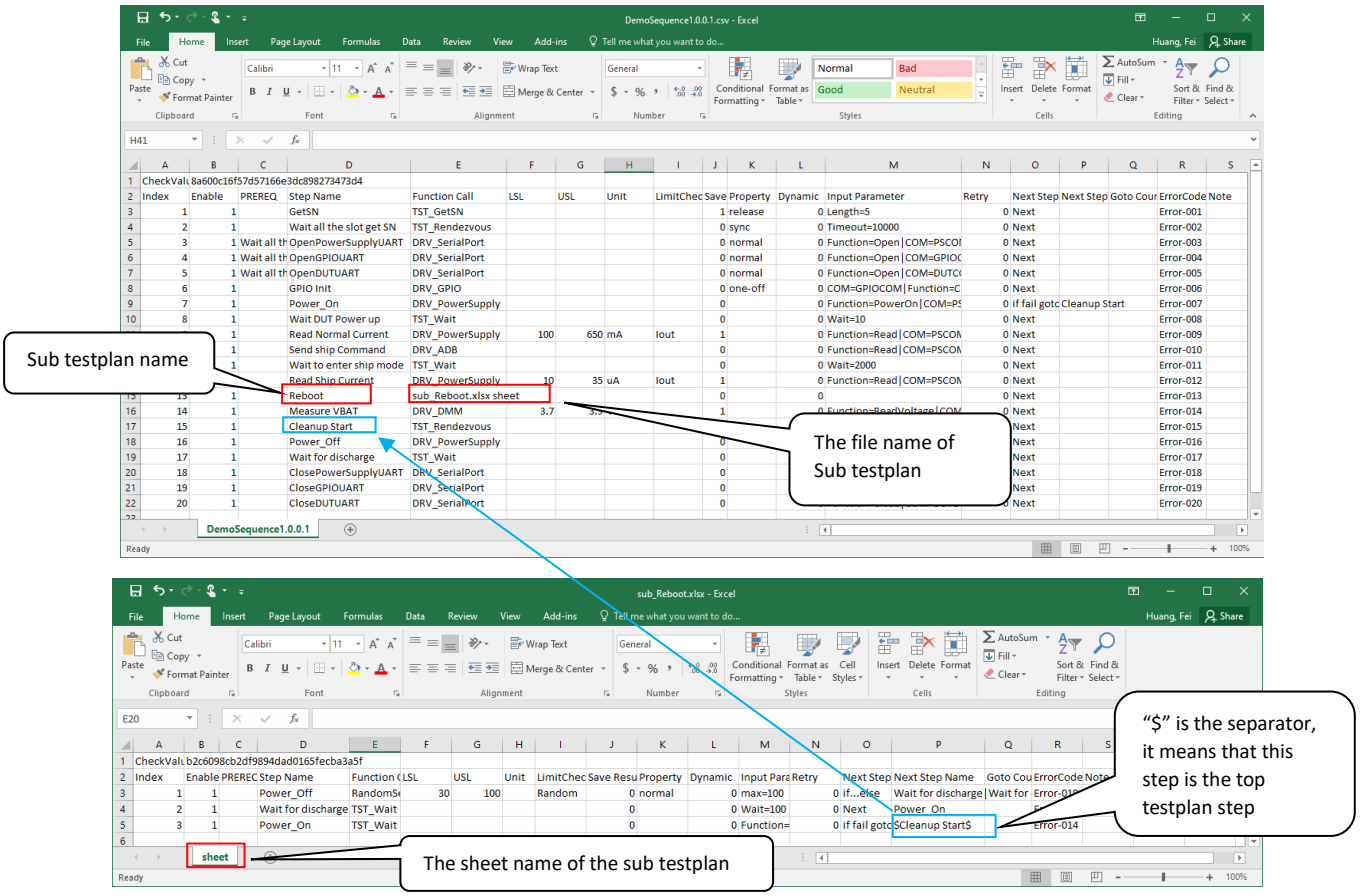


Figure 38

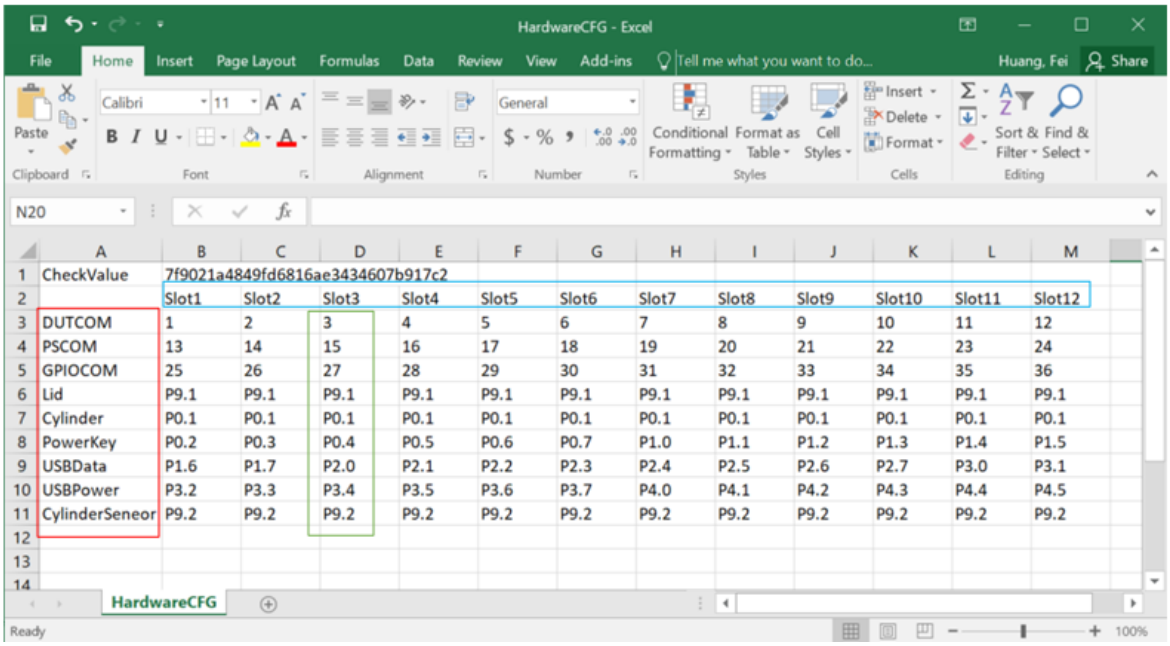
6.2.5.3 Hardware configuration

Figure 39 is a demo hardware configuration file.

In the red box is the name of the resource, the developers can customize.

In the blue box is the slot name, they are the keywords cannot be changed.

In the green box is the value of the resource, the developers can customize.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	CheckValue	7f9021a4849fd6816ae3434607b917c2											
2		Slot1	Slot2	Slot3	Slot4	Slot5	Slot6	Slot7	Slot8	Slot9	Slot10	Slot11	Slot12
3	DUTCOM	1	2	3	4	5	6	7	8	9	10	11	12
4	PSCOM	13	14	15	16	17	18	19	20	21	22	23	24
5	GPIOCOM	25	26	27	28	29	30	31	32	33	34	35	36
6	Lid	P9.1	P9.1	P9.1	P9.1	P9.1	P9.1	P9.1	P9.1	P9.1	P9.1	P9.1	P9.1
7	Cylinder	P0.1	P0.1	P0.1	P0.1	P0.1	P0.1	P0.1	P0.1	P0.1	P0.1	P0.1	P0.1
8	PowerKey	P0.2	P0.3	P0.4	P0.5	P0.6	P0.7	P1.0	P1.1	P1.2	P1.3	P1.4	P1.5
9	USBData	P1.6	P1.7	P2.0	P2.1	P2.2	P2.3	P2.4	P2.5	P2.6	P2.7	P3.0	P3.1
10	USBPower	P3.2	P3.3	P3.4	P3.5	P3.6	P3.7	P4.0	P4.1	P4.2	P4.3	P4.4	P4.5
11	CylinderSeneor	P9.2	P9.2	P9.2	P9.2	P9.2	P9.2	P9.2	P9.2	P9.2	P9.2	P9.2	P9.2
12													
13													
14													

Figure 39

6.2.6 Reset Plan

The reset plan is a TestPlan to restore the automation equipment to its original state, it means that when an exception occurs and the developer wants to reset the equipment to its original correct state, the developer can use this feature. This feature only supports running in debug mode, as shown in Figure 40, developers can enable this feature in the menu, after the reset plan is loaded into slot1, click the start button on the GUI to execute the reset plan

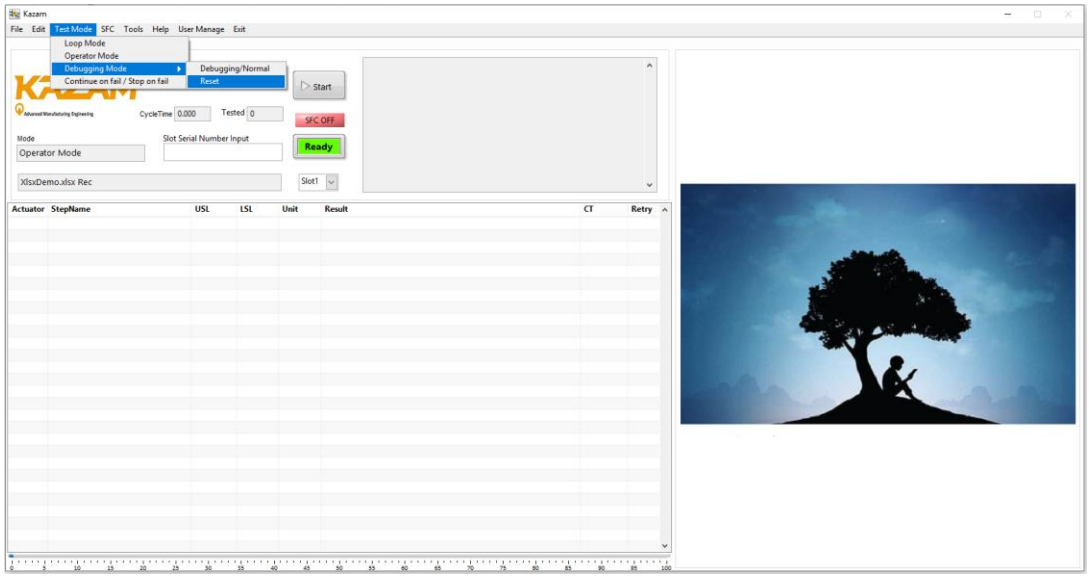


Figure 40

The reset plan is defined using the "ResetPlan" keyword in shell configuration file.

6.2.7 Global variable

In the TestPlan, the developers can define four kind of global variable.

- System global variable, it can be defining as below:

  Store=XXX>SYS_info.YYY

  Store is the keyword, cannot be changed.

  XXX is the keyword of the test step result, the developers can customize.

  SYS_info is the keyword, cannot be changed.

  YYY is the name of the global variable, the developers can customize.

  For example, if the result of test step is “Ch1=Low”, the developers want to store this information, it is defined as follows:

  Store=Ch1>SYS_info.LidStatus

  The developers can store multiple variables into the system global variable, the format is as follows:

  Store=XXX>SYS_info.YYY;ZZZ>SYS_info.NNN;TTT>SYS_info.QQQ

- Slot global variable, the storage space of each slot is separate, it can be defining as below:

  Store=XXX>Slots_info.YYY

  The default global store is ‘Slots_info’. The above can also be represented by

  Store=XXX>.YYY which functions the same as Store=XXX>Slots_info.YYY

  Store is the keyword, cannot be changed.

  XXX is the keyword of the test step result, the developers can customize.

  Slots_info is the keyword, cannot be changed.

  YYY is the name of the global variable, the developers can customize.

  For example, if the result of test step is “serialnumber=123456”, the developers want to store this information, it is defined as follows:

  Store= serialnumber >Slots_info.SN

  The developers can store multiple variables into the system global variable, the format is as follows:

  Store=XXX>Slots_info.YYY;ZZZ>SYS_info.NNN;TTT>SYS_info.QQQ

or

Store=XXX>.YYY;ZZZ>SYS_info.NNN;TTT>SYS_info.QQQ

- How to obtain the global variable value?

  1. The developers need to define as below, it means telling the platform which the variable you want

     SYS_info=LidStatus%

  2. In the vi, the developers need to search the keyword of "SYS_info.LidStatus" at the input parameter

  The developers can also get multiple variable values, the format is as follows:

  Slots_info=SN%CylinderStatus%|SYS_info=PanelSN%DUTCom%

- How to use the global variables in test plan directly?

  The "Test Name", "Input Parameter", "LSL", "USL", "Unit" and "ErrorCode" can contain global variable in the TestPlan file, and use the << and >> symbol to mark as follows:

  << Slots_info.SN >> or << SYS_info.PanelSN >>

  or

  << .SN >> or << SYS_info.PanelSN >>

  For example, if the "Input Parameter" is CMD=idme serial <<Slots_info.SN>>, the value of Slots_info.SN is P012345678, so the "Input Parameter" will be replaced by CMD=idme serial P012345678.

- Reserved variable name

  PanelSN is the reserved name of system global variable, the developers can get the panel serial number from this global variable

  SFC is the reserved name of system global variable, the developers can get the status of SFC from this global variable, the value of SFC is ON and OFF

  SN is the reserved name of slot global variable, the developers can get the slot serial number from this global variable, and it can modified by developers during the testing.

  Operator is the reserved name of system global variable, the developers can get the operator ID from this global variable, and it can modified by developers during the testing.

Diag_Ver is the reserved name of slot global variable, the developers can get the slot diag version from this global variable, and it can modified by developers during the testing.

Build_Config is the reserved name of slot global variable, the developers can get the slot build config information from this global variable, and it can modified by developers during the testing. It may be need to read from SFC server and then write into this global variable. It is Diag software version.

RunMode is the reserved name of slot global variable, the developers can get the slot running mode information from this global variable. There are only two modes, operator mode and loop mode, for example: RunMode=LOOP or RunMode=OPERATOR

Image_Name_Ver is the reserved name of slot global variable, it can modified by developers during the testing. It is Device software image bundle name and version.

Amazon_HW_Ver is the reserved name of slot global variable, it can modified by developers during the testing. It is Hardware version e.g. HVT2.

Route_check is the reserved name of slot global variable, it can modified by developers during the testing. It is used to determine whether DUT should be tested at the current test station, If DUT is ready for testing on current station, its value must be set to 1. Otherwise, its value must be set to 0. Figure 41 shows a SMT test flow, there are three stations, and the test flow of DUT1 is ISP -> BFT -> BRF, so the value of Route_check is 1 for DUT1. The test flow of DUT2 is ISP -> BRF, maybe due to the operator's wrong operation, BFT was skipped, and so when DUT2 tries to start testing at BRF, the test sequence should fail and prompt the operator that DUT2 is not ready to test at BRF station, because it skipped the BFT station, the value of Route_check should be set to 0.

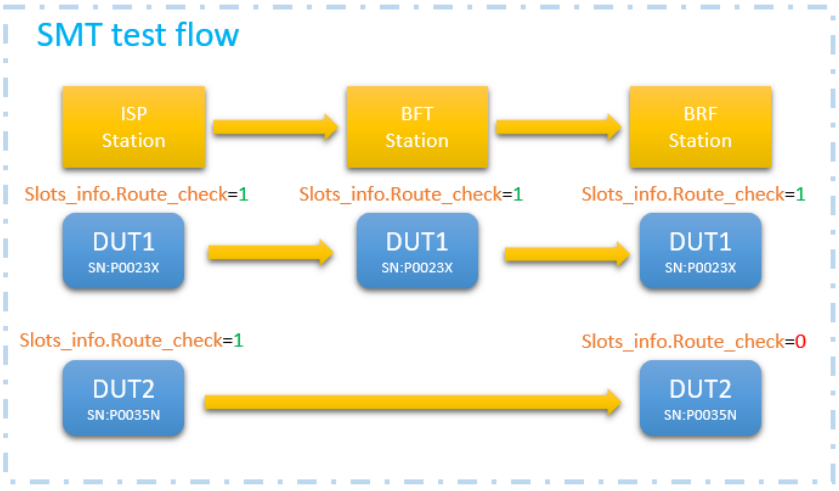


Figure 41

TestPlan_CRC is the reserved name of slot global variable, its value is MD5 value of the TestPlan, it is read-only

TestPlan_ip is the reserved name of slot global variable, its value is IP percentage of the TestPlan, it is read-only

Current_CT is the reserved name of slot global variable, its value is the time when Kazam ran to this step, it is read-only

OverallResult is the reserved name of slot global variable, its value is the overall result before this step, it is read-only, and it is used to get the status of the current test

Result.XXX is the reserved name of slot global variable, XXX is the step name, the developer need to replace it with the real step name, it can be used to get the results of a specified step

Value.XXX is the reserved name of slot global variable, XXX is the step name, the developer need to replace it with the real step name, it can be used to get the return string of a specified step

- Hardware global variable, this variable can be used to get the information defined by the developer in the hardware configuration file, it is read-only. It can be defining as below:

  HD_info.XXX

  HD_info is the keyword, cannot be changed.

  XXX is the name of the global variable, it should be defined in the hardware configuration file, the developers can customize.

  For example, the developer defined the value of DUTCOM in Slot1 as 3 in HardwareCFG file, if there is the "Function=Open |COM=<<HD_info.DUTCOM>> |BaudRate=115200" in the “input parameter” of testplan, Kazam will replace <<HD_info.DUTCOM>> to 3, so the “input parameter” of testplan will be changed to "Function=Open |COM=3 |BaudRate=115200" when the driver/test vi received it.

  And using hardware variables to fill "Test Vi Name", it can assign a different driver VI to each slot. The following is example.

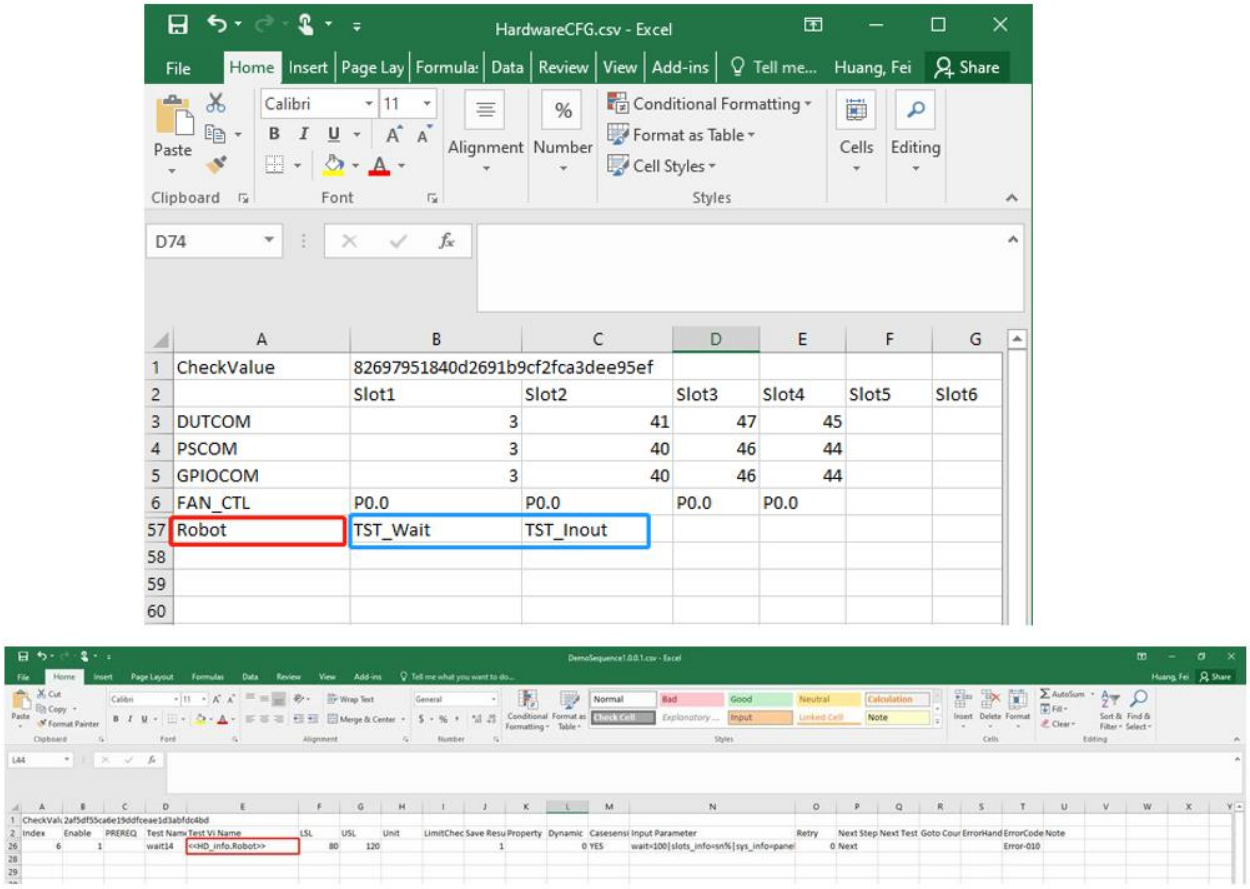


Figure 42

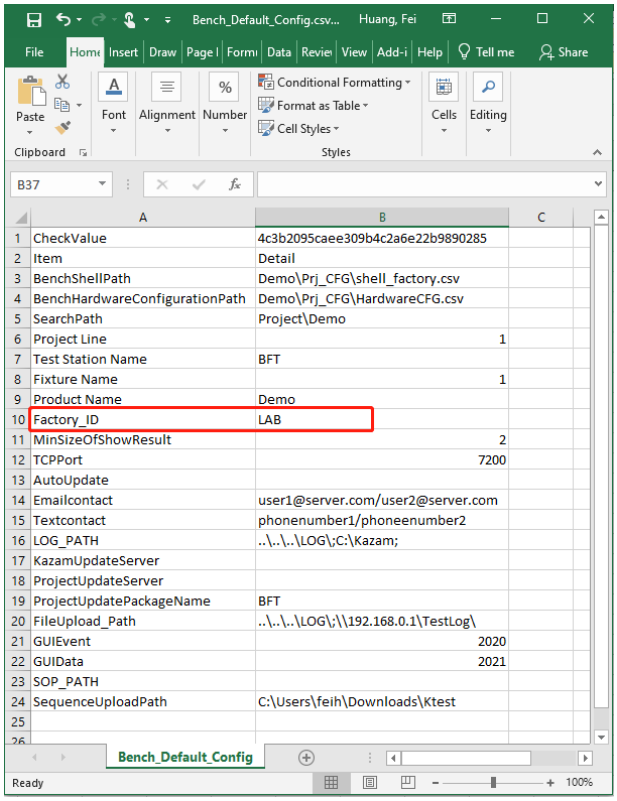
- Bench configuration global variable, this variable can be used to get the information defined by the developer in the bench configuration file, it is read-only. It can be defining as below:

  Bench_CFG.XXX

  Bench_CFG is the keyword, cannot be changed.

  XXX is the name of the global variable, it should be defined in the bench configuration file, the developers can customize.

  For example, the developer defined the value of Factory_ID in Bench_Default_Config.csv file, if there is the "Function=GetName|FactoryName=<<Bench_CFG.Factory_ID>>" in the “input parameter” of testplan, Kazam will replace << Bench_CFG.Factory_ID >> to LAB, so the “input parameter” of testplan will be changed to "Function=GetName|FactoryName=LAB" when the driver/test vi received it.



	A	B	C
1	CheckValue	4c3b2095caee309b4c2a6e22b9890285	
2	Item	Detail	
3	BenchShellPath	Demo\Prj_CFG\shell_factory.csv	
4	BenchHardwareConfigurationPath	Demo\Prj_CFG\HardwareCFG.csv	
5	SearchPath	Project\Demo	
6	Project Line	1	
7	Test Station Name	BFT	
8	Fixture Name	1	
9	Product Name	Demo	
10	Factory_ID	LAB	
11	MinSizeOfShowResult	2	
12	TCPPort	7200	
13	AutoUpdate		
14	Emailcontact	user1@server.com/user2@server.com	
15	Textcontact	phonenumber1/phoneenumber2	
16	LOG_PATH	..\..\..\LOG\;C:\Kazam;	
17	KazamUpdateServer		
18	ProjectUpdateServer		
19	ProjectUpdatePackageName	BFT	
20	FileUpload_Path	..\..\..\LOG\;\\192.168.0.1\TestLog\	
21	GUIEvent	2020	
22	GUIData	2021	
23	SOP_PATH		
24	SequenceUploadPath	C:\Users\feih\Downloads\Ktest	
25			

Figure 43

6.3 Development steps

1. Develop the driver vi, using the DRV_template.vi
2. Develop the test vi, using the TST_template.vi
3. Develop the log vi, using the log_Template.vi (If developers need special log)
4. Edit hardware configuration file
5. Edit TestPlan
6. Edit shell configuration file
7. Edit bench configuration file
8. Preview your project before build, if there are any errors here, it will tell you, as Figure 44
9. Build the Labview project, as Figure 45
10. Select the source folder, it is the compilation path for the supplier project, as Figure 46
11. Select the target folder, it is the path of Kazam.exe folder, as Figure 47

Note: All CSV and XLSX files need to be closed before compiling, such as TestPlan, shell configuration file, etc

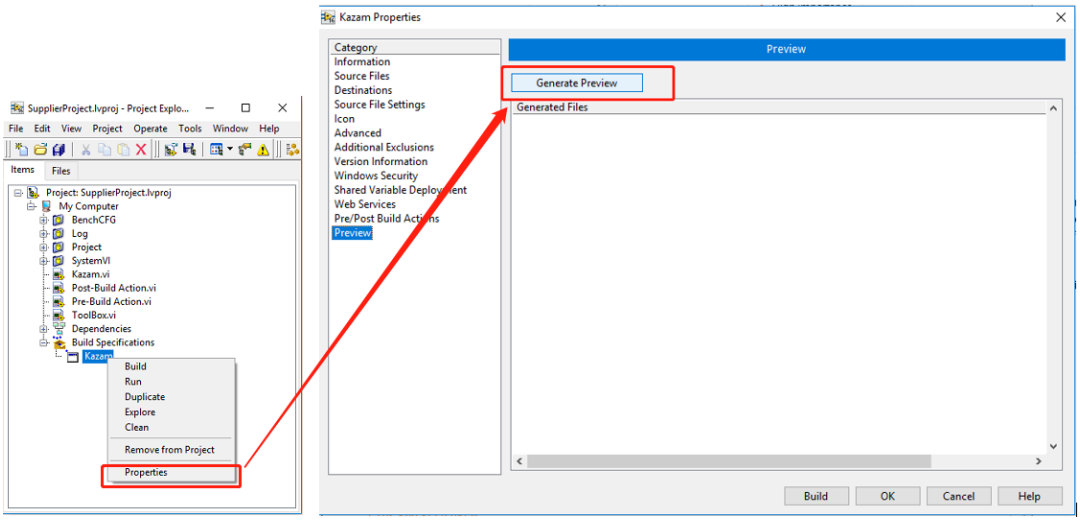


Figure 44

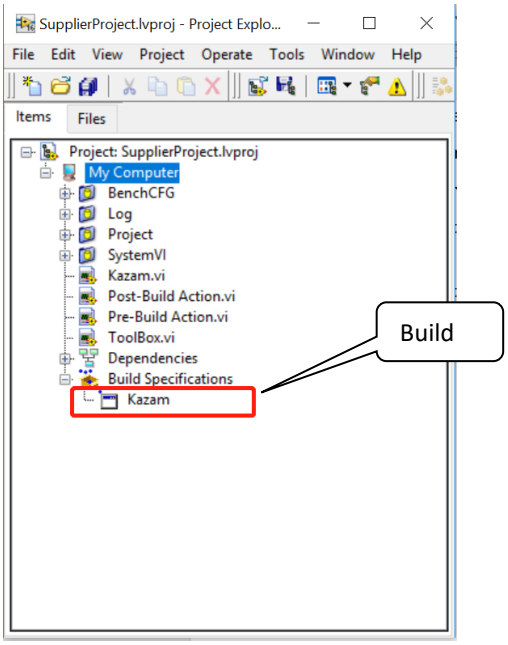


Figure 45

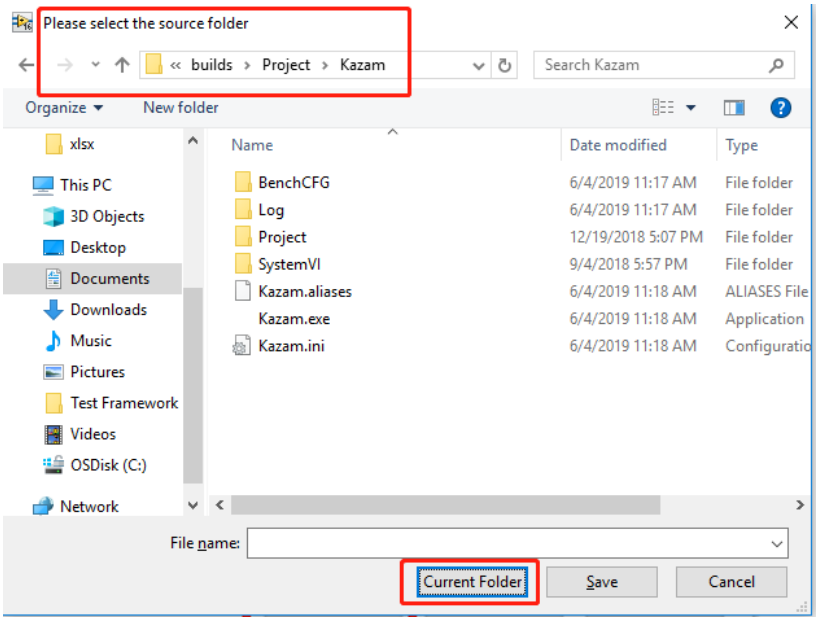


Figure 46

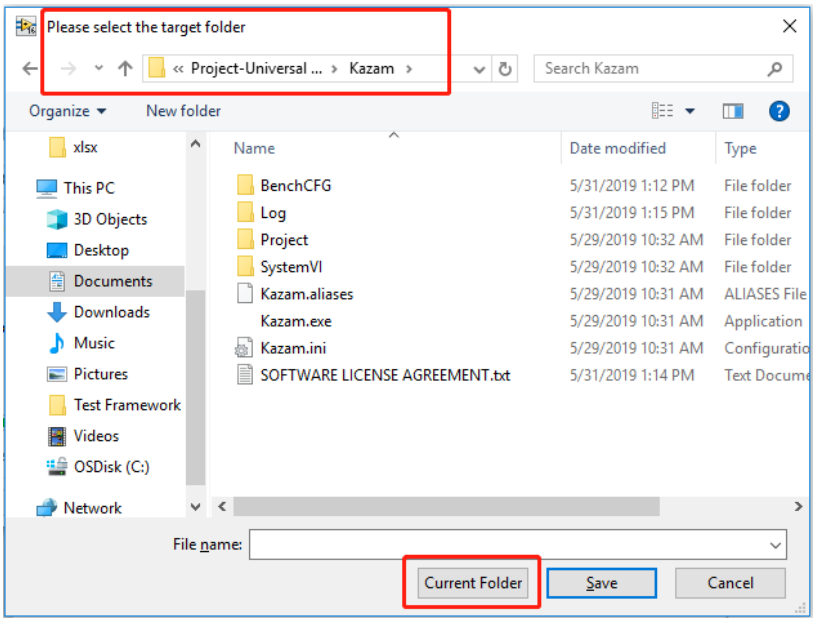

Please select the target folder
« Project-Universal ... › Kazam ›
Search Kazam
Organize
New folder
xlsx
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Test Framework
Videos
OSDisk (C:)
Network
Name
Date modified
Type
BenchCFG
5/31/2019 1:12 PM
File folder
Log
5/31/2019 1:15 PM
File folder
Project
5/29/2019 10:32 AM
File folder
SystemVI
5/29/2019 10:32 AM
File folder
Kazam.aliases
5/29/2019 10:31 AM
ALIASES File
Kazam.exe
5/29/2019 10:31 AM
Application
Kazam.ini
5/29/2019 10:31 AM
Configuratio
SOFTWARE LICENSE AGREEMENT.txt
5/31/2019 1:14 PM
Text Docume
File name:
Current Folder
Save
Cancel

Figure 47

6.4 Labview Run-time Version

The developers need to download the “LabVIEW Run-Time Engine 2016 - (64-bit) – Windows” from the following website:

http://www.ni.com/download/labview-run-time-engine-2016/6067/en/

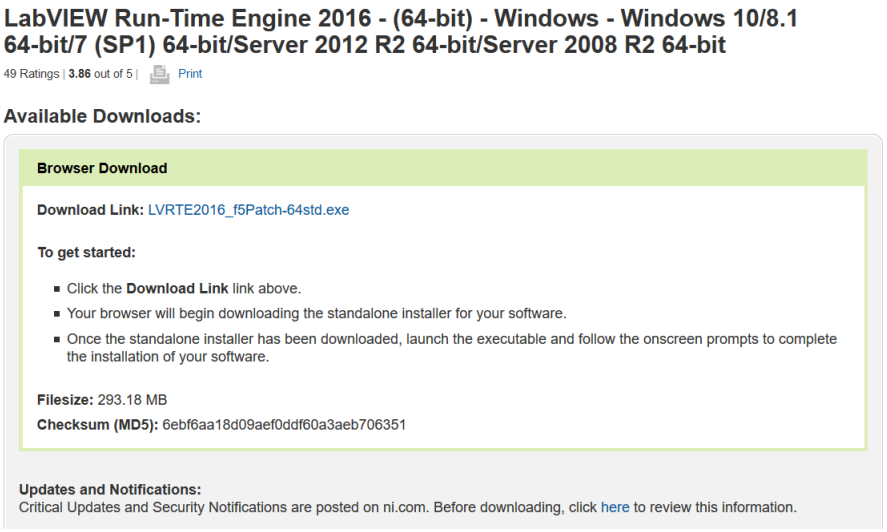


Figure 48

6.5 Other language or EXE/Python/Bat file support

6.5.1 EXE/Python/Bat file support

The platform uses the Windows command line to call the executive file, py, pyc and bat file. Kazam will call it in the following format:

cmd /c python \found_path_2_file\testfunction.py %args%|Timeout=30000|WorkingDir= found_path_2_file|wait4return=1

cmd /c python \found_path_2_file\testfunction.pyc %args%|Timeout=30000|WorkingDir= found_path_2_file|wait4return=1

cmd /c python \found_path_2_file\testfunction.exe %args%|Timeout=30000|WorkingDir= found_path_2_file|wait4return=1

cmd /c python \found_path_2_file\testfunction.bat %args%|Timeout=30000|WorkingDir= found_path_2_file|wait4return=1

%args% is used to set parameters, and it can be wrote in the input parameters, its keyword is args, for example: args= --port com4 --mode 0

Timeout is used to set a timeout to wait for a return, the default value is 30000 millisecond, and it can be overridden in the input parameters

WorkingDir is used to set the script path, the default path is the path of the project folder, and it can be overridden in the input parameters

wait4return is used to set whether to wait for a return, if this step is a non-blocking step, the default value is 0, if this step is a blocking step, the default value is 1, 1 means that it will wait for return, 0 means that it will not wait for return, and it can be overridden in the input parameters

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	CheckVali	792a7bc52dcbed1585412999ecd5cf07																			
2	Index	Enable	PREREQ	RunInPara	Test Nam	Test Vi Name	LSL	USL	Unit	LimitChec	Save Resu	Property	Dynamic	Casesensi	Input Parameter	Retry	Next Step	Next Test	Goto Cour	ErrorCode	Note
3	1	1			Delay_1	TST_delay.pyc					0	normal	0	Yes	args=Delay=1000	0	next			Error-001	
4	1	1			Delay_2	TST_delay.py					0	normal	0	Yes	args=Delay=1000|WorkingDir=D:\Kazam|wait4return=1	0	next			Error-002	
5	1	1			Wait_1	TST_Wait.exe					0	normal	0	Yes	args=Wait=1000	0	next			Error-003	
6	1	1			Wait_2	TST_Wait.bat					0	normal	0	Yes	args=Wait=1000|Timeout=30000|wait4return=1	0	next			Error-004	

Figure 49

As Figure 49, this is demo testplan, the developers can directly call exe, python, bat files in "Test Vi Name", and set their parameters in "Input Parameter"

6.5.2 C/C++ DLL support

The platform can call the C or C++ DLL directly. The DLL function define as below:

```
CStr AME_Function(const CStr ParametersIn);
```

AME_Function is the function name, it can't be changed, and the type of function return value is string, the format is same as executive file return value.

const CStr ParametersIn is the input of the function, the parameter name can't be changed, and the type of input parameters is string, the format is same as executive file return value

6.5.3 C# source code support

The platform can call the C# source code in testplan directly. And the developer can use Microsoft Visual Studio to debug their source code directly. The following is a C# code demo:

```
using System;
/*this name can be changed as you defined*/
namespace AAA
{
  /*Class name cannot be changed*/
  public class AME_TestScripture
  {
  /*Function name cannot be changed*/
    public string AME_Function(string para1, int para2)
    {
      string feedback = "";
      try
      {
          feedback ="Result="+para1+"|para2=" + para2.ToString();
      }
      catch (Exception ex)
      {
         throw(ex);
      }
      return feedback;
    }
  }
}
```

AAA is the namespace, it can be changed by the developer.

AME_TestScripture is the class name, it cannot be changed.

AME_Function is the function name, it cannot be changed. Its parameters can be defined by developer, Kazam supports common parameter types, such as string, integer, floating point, etc.

para1 and para2 are the input parameters of AME_Function, the developer can add, delete, and modify the input parameter by themselves. And the corresponding parameters must be defined in the “input parameter” of the testplan. Figure 49 is a demo, the testplan calls a c# soure code, and there are two parameters in the “input parameter” of the testplan, Kazam will automatically find para1 and para2 and assign the function parameters to AME_Function.

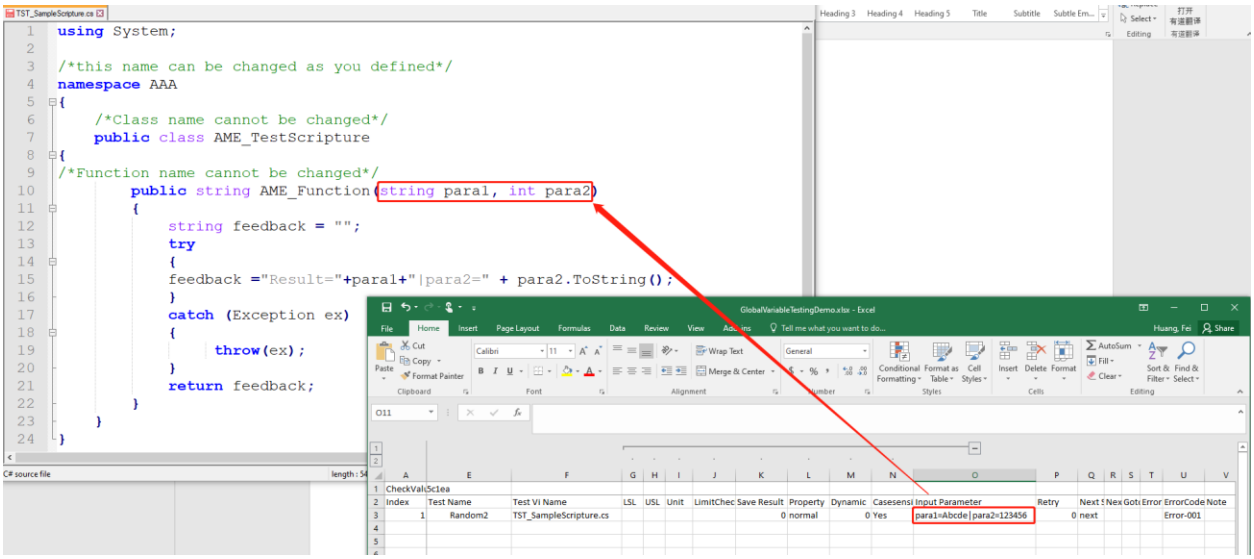

TST_SampleScripture.cs
using System;
/*this name can be changed as you defined*/
namespace AAA
{
/*Class name cannot be changed*/
public class AME_TestScripture
{
/*Function name cannot be changed*/
public string AME_Function(string para1, int para2)
{
string feedback = "";
try
{
feedback ="Result="+para1+"|para2=" + para2.ToString();
}
catch (Exception ex)
{
throw(ex);
}
return feedback;
}
}
}
C# source file
GlobalVariableTestingDemo.xlsx - Excel
CheckVal:Sc1ea
Index
Test Name
Test VI Name
Input Parameter
Retry
Random2
TST_SampleScripture.cs
para1=Abcde|para2=123456
Error-001

Figure 50

7. Data log

The platform contains three types of standard log formats, single log, time log and daily log. As mentioned in the previous chapter, the platform supports custom log formats, the developers can follow the log interface specification to develop their own log format.

7.1 Single Log

Single log is used to record information for all steps, it is in TXT format. The following is an example of a single log:

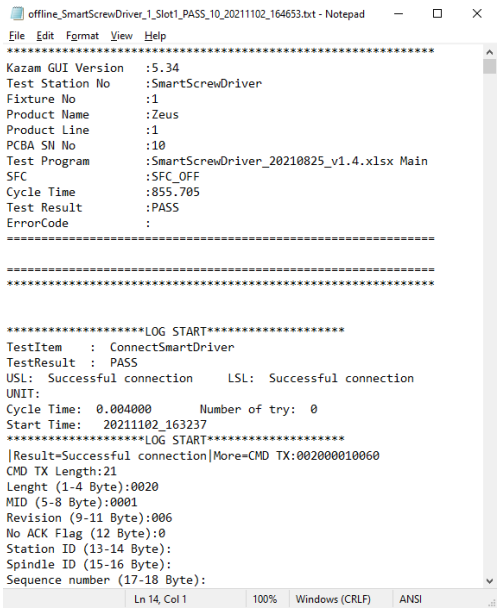


Figure 51

7.2 Time Log

Time log is used to record cycle time for all steps, it is in csv format. The following is an example of a time log:

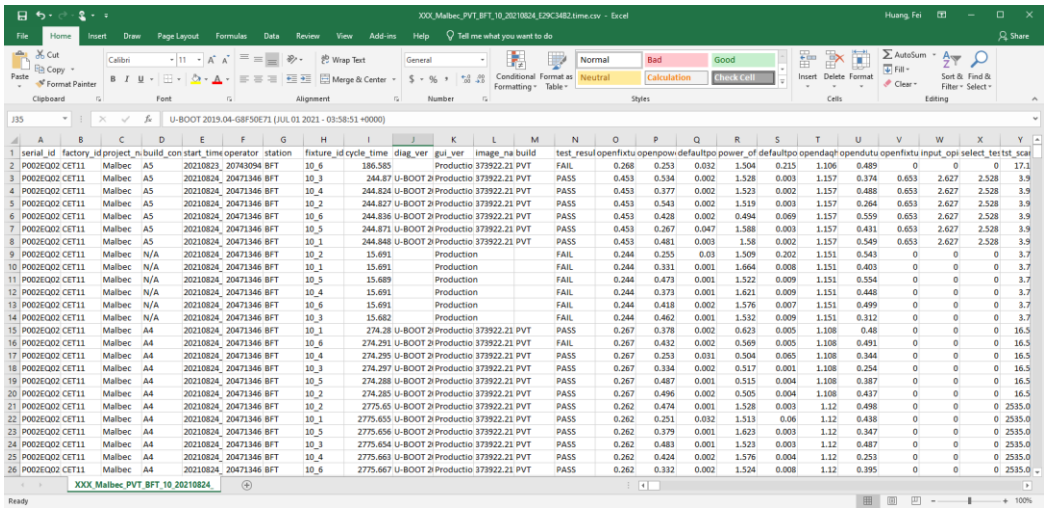


Figure 52

7.3 Daily Log

Daily log is used to record all data in a day, it is in csv format. Figure 53 is an example of a daily log, the part marked by the red box is called the fixed header.

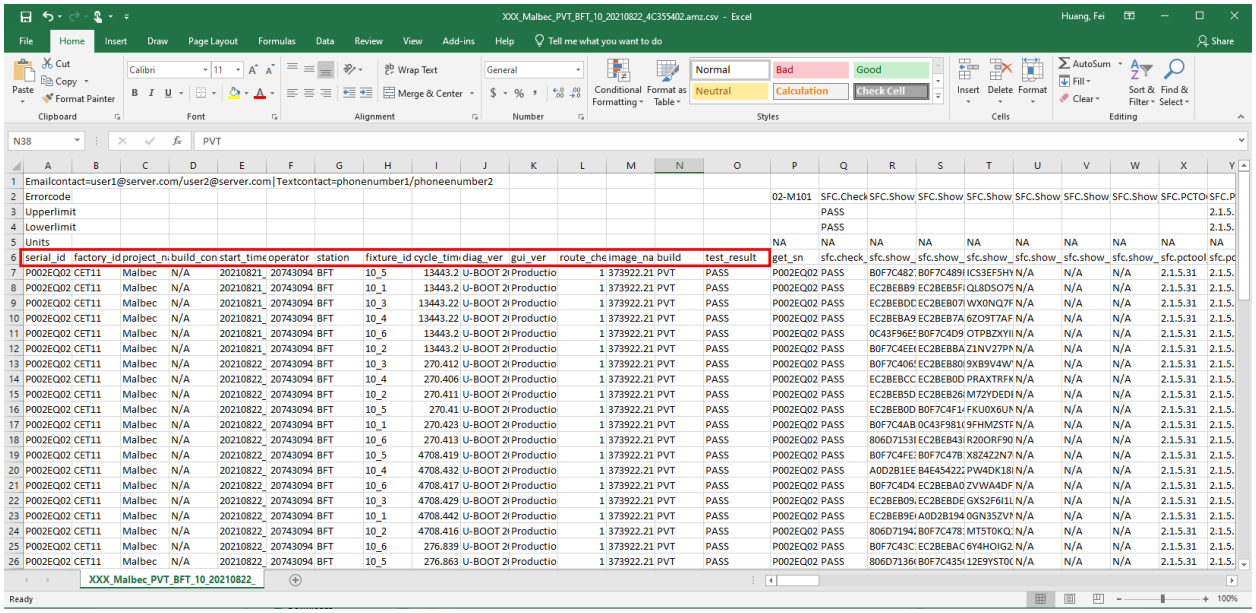


Figure 53

7.3.1 Fixed Header

The fixed header can be customized by the developer, and it can be defined in shell configuration file, Figure 54 shows how to define the developer's own fixed header file, it is similar to define a testplan.

FixedHeader is the keyword for the default fixed header sequence, if define "FixedHeader1= BBB.csv", it means that slot 1 will use this "BBB.csv" as its fixed header sequence

If define "FixedHeader2= CCC.csv", it means that slot 2 will use this "CCC.csv" as its fixed header sequence

And so on.

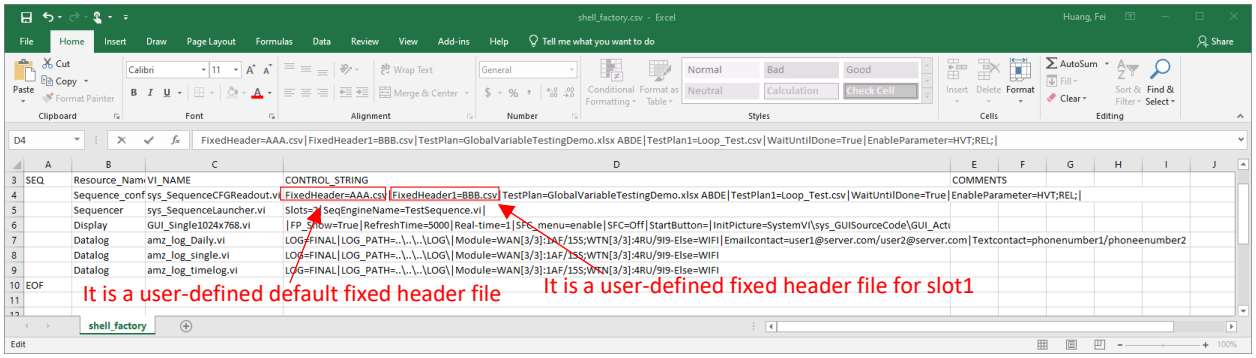
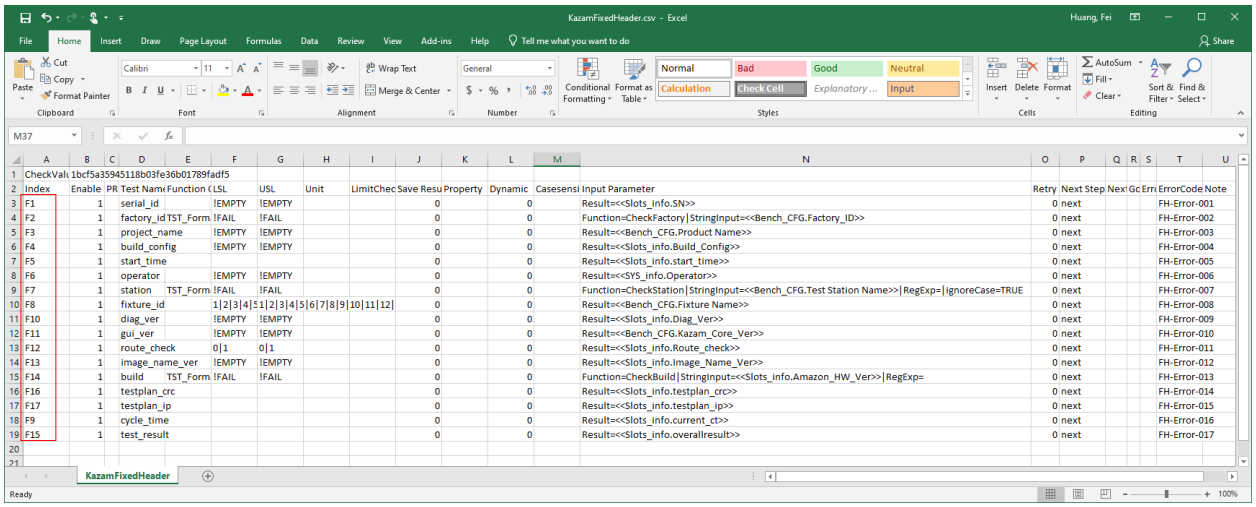


Figure 54

If the developer does not define the fixed header file in shell configuration file, Kazam will automatically call the default fixed header file for each slot, its default path is Kazam\SystemVI\sys_Configuration\KazamFixedHeader.csv

Figure 55 shows the default fixed header sequence, its format is the same as other sequences, the only difference is the index definition, the platform uses F to identify that this row is a fixed header row, F1 means that the name of this step is the name of the first fixed column in daily log, F2 means that the name of this step is the name of the second fixed column in daily log, F3 means that the name of this step is the name of the third fixed column in daily log ,and so on.



Index	Enable	PR	Test Nam	Function (	LSL	USL	Unit	LimitChec	Save Resu	Property	Dynamic	Casesensi	Input Parameter	Retry	Next Step	Next	Gc	Err	ErrorCode	Note
CheckVal	1bcf5a35945118b03fe36b01789fadf5																			
F1	1		serial_id		!EMPTY	!EMPTY			0		0		Result=<<Slots_info.SN>>	0	next				FH-Error-001	
F2	1		factory_id	TST_Form	!FAIL	!FAIL			0		0		Function=CheckFactory|StringInput=<<Bench_CFG.Factory_ID>>	0	next				FH-Error-002	
F3	1		project_name		!EMPTY	!EMPTY			0		0		Result=<<Bench_CFG.Product Name>>	0	next				FH-Error-003	
F4	1		build_config		!EMPTY	!EMPTY			0		0		Result=<<Slots_info.Build_Config>>	0	next				FH-Error-004	
F5	1		start_time						0		0		Result=<<Slots_info.start_time>>	0	next				FH-Error-005	
F6	1		operator		!EMPTY	!EMPTY			0		0		Result=<<SYS_info.Operator>>	0	next				FH-Error-006	
F7	1		station	TST_Form	!FAIL	!FAIL			0		0		Function=CheckStation|StringInput=<<Bench_CFG.Test Station Name>>|RegExp=|IgnoreCase=TRUE	0	next				FH-Error-007	
F8	1		fixture_id		1|2|3|4|5	1|2|3|4|5|6|7|8|9|10|11|12			0		0		Result=<<Bench_CFG.Fixture Name>>	0	next				FH-Error-008	
F10	1		diag_ver		!EMPTY	!EMPTY			0		0		Result=<<Slots_info.Diag_Ver>>	0	next				FH-Error-009	
F11	1		gui_ver		!EMPTY	!EMPTY			0		0		Result=<<Bench_CFG.Kazam_Core_Ver>>	0	next				FH-Error-010	
F12	1		route_check		0|1	0|1			0		0		Result=<<Slots_info.Route_check>>	0	next				FH-Error-011	
F13	1		image_name_ver		!EMPTY	!EMPTY			0		0		Result=<<Slots_info.Image_Name_Ver>>	0	next				FH-Error-012	
F14	1		build	TST_Form	!FAIL	!FAIL			0		0		Function=CheckBuild|StringInput=<<Slots_info.Amazon_HW_Ver>>|RegExp=	0	next				FH-Error-013	
F16	1		testplan_crc						0		0		Result=<<Slots_info.testplan_crc>>	0	next				FH-Error-014	
F17	1		testplan_ip						0		0		Result=<<Slots_info.testplan_ip>>	0	next				FH-Error-015	
F9	1		cycle_time						0		0		Result=<<Slots_info.current_ct>>	0	next				FH-Error-016	
F15	1		test_result						0		0		Result=<<Slots_info.overallresult>>	0	next				FH-Error-017	

Figure 55

8. External API

The platform communicates with external software using TCP/IP, The default port is 7200, and it can be modified by developer in Bench_Default_Config.csv file. Kazam work in server mode, external software need to access using TCP clients.

8.1 Command summary

The command format is as follows:

CMD=XXX| RequiredParameter=YYY|OptionalParameter=ZZZ\r\n

CMD is keyword of an external command, it can't be changed.

XXX is a command, can find them in the command columns of the table below.

RequiredParameter is a must keyword, can find them in the required parameters columns of the table below, some commands do not require this parameter.

YYY is the value of the required parameter.

OptionalParameter is the keyword of optional parameter, the developers can decide whether to need them.

ZZZ is the value of the optional parameter.

\r\n means CR (carriage return) followed by a LF (linefeed), it is end char.

Note1: All commands and parameters are not case sensitive.

Note2: There is already a vi to refer, and its name is DRV_EAPI.vi

Command	Required parameters	Optional Parameter	Description
StartAll		PanelSN	Start all the tests
StartOne	Slot	SN	Start the test of one slot
SetMode	Mode=Operator Mode=Loop		Set the test mode
SetSFC	SFC=ON SFC=OFF		Set the SFC
SetLoopCounter	LoopCounter=100		Set the number of loops.

ForceExit			Forced exit Kazam.exe
ReadSlotResults	Slot		Read all the results of a slot and Kazam running status
ReadStepResults	Slot, TestName		Read the result of the specified step and Kazam running status

For example:

External software send the following commands:

CMD= StartAll|PanelSN=30975

If no error, the platform will return back “Successful”

If there is error, the platform will return back the error information

7.1.1 Commands detail description

StartAll: This command will make all the slot to begin testing.

External software send the following commands:

CMD= StartAll|PanelSN=30975

If no error, the platform will return back “Successful”

If there is error, the platform will return back the error information

PanelSN is used to tell the platform the panel serial number.

-

StartOne: This command will make one slot to begin testing.

External software send the following commands:

CMD= StartOne|Slot=1|SN=12345678

If no error, the platform will return back “Successful”

If there is error, the platform will return back the error information

Slot is used to tell the platform which slot is selected. Its range is from 1 to 32.

SN is used to tell the platform the serial number of this slot.

-

SetMode: This command sets the running mode of the platform, the loop mode, or the operator mode.

External software send the following commands:

CMD= SetMode|Mode=Operator

If no error, the platform will return back “Successful”

If there is error, the platform will return back the error information

Mode is used to tell the platform which mode is selected. Its value is loop or operator.

SetSFC: This command will set the state of the SFC, online or offline.

External software send the following commands:

CMD= SetSFC|SFC=ON

If no error, the platform will return back “Successful”

If there is error, the platform will return back the error information

SFC is used to tell the platform whether to test online or offline. Its value is ON or OFF.

SetLoopCounter: This command will set the number of loop.

External software send the following commands:

CMD= SetLoopCounter|LoopCounter=100

If no error, the platform will return back “Successful”

If there is error, the platform will return back the error information

LoopCounter is used to tell the platform how many times it needs to loop. Its value is -1 or 1~ 2147483648, “-1” means forever.

ForceExit: This command is used to force an exit from the platform.

External software send the following commands:

CMD=ForceExit

This command does not return a value.

-

ReadSlotResults: This command will read all the results of a slot and Kazam running status.

External software send the following commands:

CMD= ReadSlotResults|Slot=1

If no error, the platform will return back all the results of this slot, and it is JSON string, the developer can use “Unflatten From JSON” (

) to parse it, or use other methods, more details, please reference Kazam\Project\Library\EAPI\EAPI_GetTestResults.vi

If there is error, the platform will return back the error information

Slot is used to tell the platform which slot is selected. Its range is from 1 to 32.

-

ReadStepResults: This command will read the result of the specified step and Kazam running status.

External software send the following commands:

CMD= ReadStepResults|Slot=1|TestName=WaitGetSN

If no error, the platform will return back all the results of this slot, and it is JSON string, the developer can use “Unflatten From JSON” (

) to parse it, or use other methods, more details, please reference Kazam\Project\Library\EAPI\EAPI_GetStepResults.vi

If there is error, the platform will return back the error information

Slot is used to tell the platform which slot is selected. Its range is from 1 to 32.

TestName is used to tell the platform which test step is selected.

9. Important parameters

9.1 Bench_Default_Config.csv

Some important parameters are predefined in the Bench_Default_Config.csv file

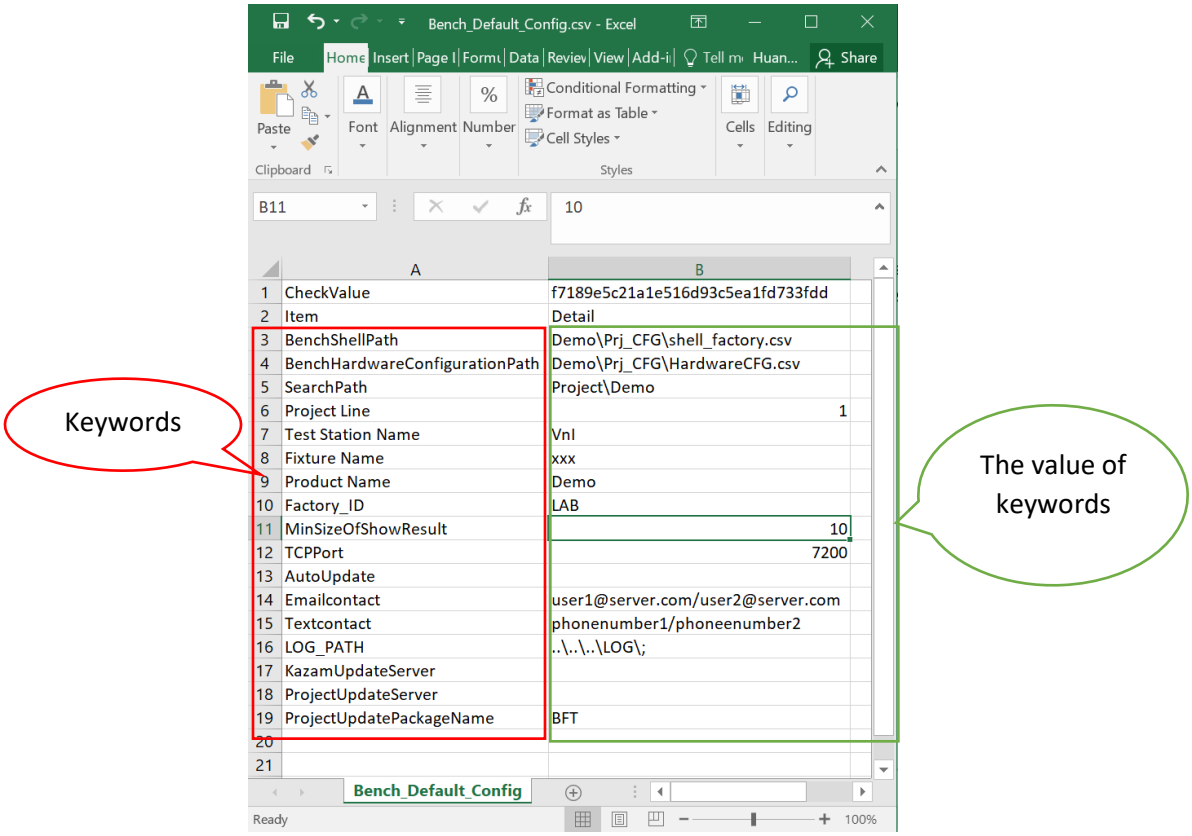


Figure 56

Project Line: This is used to define the name of project line. It will be used to define the file name of log.

AutoUpdate: This is used to set whether to automatically update the project folder. Its value is ENABLE or disable.

Note: Kazam will automatically check the “CheckValue” of each file when enabling the project folder to update

ProjectUpdateServer: This is used to define the project server path, when Kazam is running in exe mode, it automatically checks to see if there is an update in the update server.

ProjectUpdatePackageName: This is used to define the name of update package, when Kazam is running in exe mode, it automatically checks the name of update package to see if there is an update in the update server.

KazamUpdateServer: This is used to define the kazam server path, when Kazam is running in exe mode, it automatically checks to see if there is an update in the update server.

Emailcontact: It is used to set the mailbox for the contact. It will be writhed into the daily log.

Textcontact: It is used to set the phone number for the contact. It will be writhed into the daily log.

LOG_PATH: This is used to set the storage path for log. It can set multiple paths, and isolate each path with a semicolon.

BenchShellPath: It is used to define the shell configuration file path.

BenchHardwareConfigurationPath: It is used to define the hardware configuration file path.

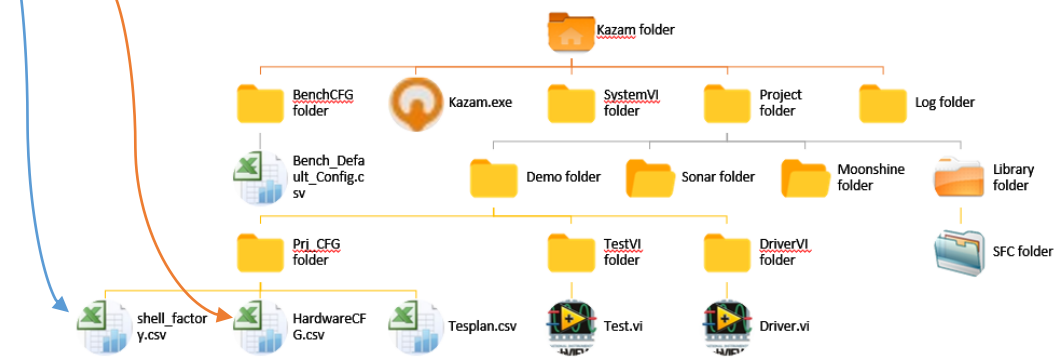


Figure 57

SearchPath: It is used to define search paths, it can be multiple paths, it allows developer to enter both the absolute path and the relative path, and isolate each path with a semicolon, for example:

SearchPath = Project\Demo;C:\kazam;\\192.168.0.32\AFT\Code;

In this case, Kazma will first search for the "Project\Demo" path under the kazam.exe path, if kazam find a specific test plan, kazam will stop the search, otherwise will continue to search next path. The second step, kazam will start from the first search path search the driver and test vi.

Test Station Name: This is used to define the type of test station, such as AFT, VnI, MT, PT, and so on. It will be writhed into the daily log

Fixture Name: This is used to define the fixture name, such as BFT01, BFT02, AFT01, and so on. It will be writhed into the daily log and will be used to define the file name of log.

Product Name: This is used to define the product name, such as Moonshine, Muffin, Sonar, and so on. It will be writhed into the daily log and will be used to define the file name of log.

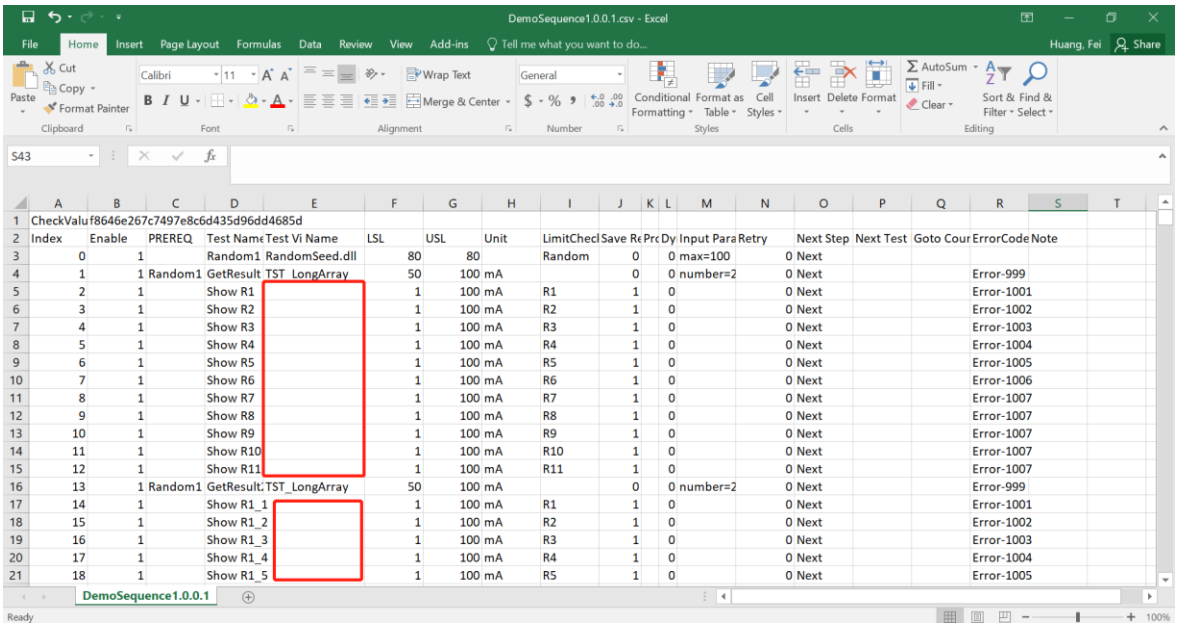
Factory_ID: This is used to define the factory name. Maybe in the future will be new rules. It will be writhed into the daily log.

MinSizeOfShowResult: This is used to define the length of show results. As Figure 58, there are 11 results need to show on GUI. So the length of show results is 11.

If the developer wants to control the “next step” of each show result, then MinSizeOfShowResult needs to be set to greater than 11, such as 12.

If the developer does not want to control the "next step" of each show result, the MinSizeOfShowResult needs to be set to less than 11, such as 10.

MinSizeOfShowResult is a very important parameter, please be careful to set it, otherwise the test time may become very long



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	CheckValu	f8646e267c7497e8c6d435d96dd4685d																		
2	Index	Enable	PREREQ	Test Name	Test Vi Name	LSL	USL	Unit	LimitChecl	Save Re	Pre	Dy	Input Para	Retry	Next Step	Next Test	Goto Cour	ErrorCode	Note	
3	0	1		Random1	RandomSeed.dll	80	80		Random	0	0		max=100	0	Next					
4	1	1	Random1	GetResult	TST_LongArray	50	100	mA		0	0		number=2	0	Next			Error-999		
5	2	1		Show R1		1	100	mA	R1	1	0			0	Next			Error-1001		
6	3	1		Show R2		1	100	mA	R2	1	0			0	Next			Error-1002		
7	4	1		Show R3		1	100	mA	R3	1	0			0	Next			Error-1003		
8	5	1		Show R4		1	100	mA	R4	1	0			0	Next			Error-1004		
9	6	1		Show R5		1	100	mA	R5	1	0			0	Next			Error-1005		
10	7	1		Show R6		1	100	mA	R6	1	0			0	Next			Error-1006		
11	8	1		Show R7		1	100	mA	R7	1	0			0	Next			Error-1007		
12	9	1		Show R8		1	100	mA	R8	1	0			0	Next			Error-1007		
13	10	1		Show R9		1	100	mA	R9	1	0			0	Next			Error-1007		
14	11	1		Show R10		1	100	mA	R10	1	0			0	Next			Error-1007		
15	12	1		Show R11		1	100	mA	R11	1	0			0	Next			Error-1007		
16	13	1	Random1	GetResult	TST_LongArray	50	100	mA		0	0		number=2	0	Next			Error-999		
17	14	1		Show R1_1		1	100	mA	R1	1	0			0	Next			Error-1001		
18	15	1		Show R1_2		1	100	mA	R2	1	0			0	Next			Error-1002		
19	16	1		Show R1_3		1	100	mA	R3	1	0			0	Next			Error-1003		
20	17	1		Show R1_4		1	100	mA	R4	1	0			0	Next			Error-1004		
21	18	1		Show R1_5		1	100	mA	R5	1	0			0	Next			Error-1005		

Figure 58

TCPPort: This is used to define the TCP port. Its default port is 7200.

9.2 Kazam configuration file

Kazam configuration file is used to define the feature of Kazam, such as what log vi is used, what GUI is used, and so on. As Figure 59, it is a Kazam configuration file, see below for parameter definitions:

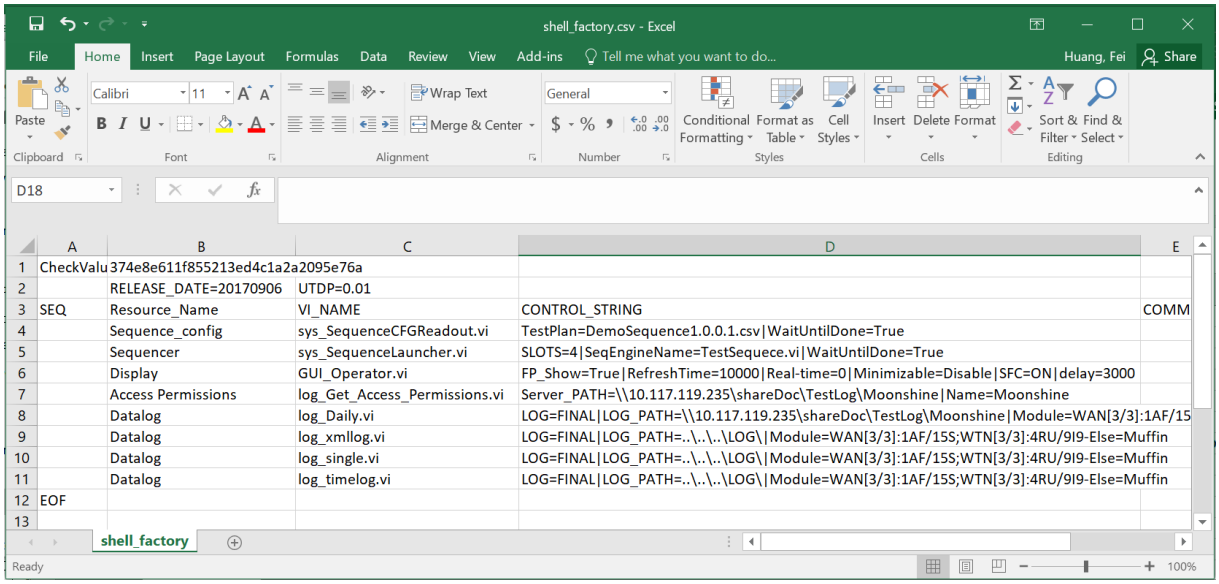


Figure 59

8.2.1 Common parameter

Common parameters can be used everywhere in Kazam configuration file.

Common parameter name	Description	Example
WaitUntilDone	It is used to tell the platform to wait for the completion of this vi execution WaitUntilDone=True, it means that the platform need to wait for the completion of this vi execution Empty, it means that no need to wait for execution to end	WaitUntilDone=True

8.2.2 Other parameter

For each function, the developer can define their parameter. Below is the parameters that the platform has defined

sys_SequenceLauncher.vi parameter name	Description	Example
SLOTS	It is used to tell the platform how many slots will be used	SLOTS=4
SeqEngineName	It can't be modified	SeqEngineName=TestSequence.vi

sys_SequenceCFGReadout.vi parameter name	Description	Example
TestPlan	It is used to tell the platform which testplan you want to use. "TestPlan" is the default keyword for all slots	TestPlan=DemoSequence1.0.0.1.csv
TestPlan1	It is used to tell the platform that the slot1 will use this testplan.	TestPlan1=Seq_Slot1.csv TestPlan1=Seq_Slot1.xlsx Sheet1
TestPlan2	It is used to tell the platform that the slot2 will use this testplan.	TestPlan2=Seq_Slot2.csv TestPlan2=Seq_Slot2.xlsx myTab
.....	.....	.....
TestPlan32	It is used to tell the platform that the slot32 will use this testplan.	TestPlan32=Seq_Slot32.csv
ResetPlan	It is used to tell the platform which reset plan you want to use.	ResetPlan=Seq_reset.csv
EnableParameter	It is used to set the enabling parameters that need to be enabled by default	EnableParameter=HVT;DVT;

GUI_Operator.vi parameter name	Description	Example
FP_Show	It is used to tell the platform that this GUI needs to be displayed	FP_Show=True
RefreshTime	It is used to set the refresh time of GUI, unit is a millisecond The refresh time must set to 10 seconds for mass production	RefreshTime=10000
Real-time	It is used to set whether test results are displayed in real time, its value is 0 or 1 The Real-time must set to 0 for mass production	Real-time=0
Minimizable	It is used to disable the GUI minimized, its value is Disable or Enable	Minimizable=Enable
SFC	It is used to tell the platform whether the SFC is ON or OFF by default	SFC=OFF
delay	In loop mode, it is used to set the delay between this run and the next	delay=3000

AutoScrolling	It is used to enable automatic scrolling, its value is Disable or Enable	AutoScrolling=Enable
ToolPath	It is used to specify the path of the developer's own tool vi	ToolPath=Demo\TesterVI\TST_tool.vi ToolPath=C:\Tools\TST_tool.vi
SFC_Menu	It is used to disabled the SFC menu, its value is Disable or Enable, default is enable	SFC_Menu=Enable
StartButton	It is used to disabled the start button, its value is Disable or Enable, default is enable	StartButton=Enable
DebuggingGUIPath	It is used to specify the path of the debugging tool vi	DebuggingGUIPath=DebuggingTool.vi DebuggingGUIPath=C:\Tools\DebuggingTool.vi
DebuggingToolCFG	It is used to specify the path of the debugging configuration file	DebuggingToolCFG= DebuggingToolCFG.xlsx DebuggingToolCFG =C:\Tools\ DebuggingToolCFG.xlsx

GUI_Single.vi GUI_Single1024x768.vi GUI_Single512x768.vi parameter name	Description	Example
FP_Show	It is used to tell the platform that this GUI needs to be displayed	FP_Show=True
RefreshTime	It is used to set the refresh time of GUI, unit is a millisecond The refresh time must set to 10 seconds for mass production	RefreshTime=10000
Real-time	It is used to set whether test results are displayed in real time, its value is 0 or 1 The Real-time must set to 0 for mass production	Real-time=0
Minimizable	It is used to disable the GUI minimized, its value is Disable or Enable	Minimizable=Enable
SFC	It is used to tell the platform whether the SFC is ON or OFF by default	SFC=OFF
delay	In loop mode, it is used to set the delay between this run and the next	delay=3000
SlotPosition	It is used to tell the platform the slot number for this GUI, its value is 1,2,3,…32	SlotPosition=1

AutoScrolling	It is used to enable automatic scrolling, its value is Disable or Enable	AutoScrolling=Enable
ToolPath	It is used to specify the path of the developer's own tool vi	ToolPath=Demo\TesterVI\TST_tool.vi ToolPath=C:\Tools\TST_tool.vi
SFC_Menu	It is used to disabled the SFC menu, its value is Disable or Enable, default is enable	SFC_Menu=Enable
StartButton	It is used to disabled the start button, its value is Disable or Enable, default is enable	StartButton= Enable
DebuggingGUIPath	It is used to specify the path of the debugging tool vi	DebuggingGUIPath=DebuggingTool.vi DebuggingGUIPath=C:\Tools\DebuggingTool.vi
DebuggingToolCFG	It is used to specify the path of the debugging configuration file	DebuggingToolCFG= DebuggingToolCFG.xlsx DebuggingToolCFG =C:\Tools\ DebuggingToolCFG.xlsx

GUI_MultiSingle.vi parameter name	Description	Example
FP_Show	It is used to tell the platform that this GUI needs to be displayed	FP_Show=True
RefreshTime	It is used to set the refresh time of GUI, unit is a millisecond The refresh time must set to 10 seconds for mass production	RefreshTime=10000
Real-time	It is used to set whether test results are displayed in real time, its value is 0 or 1 The Real-time must set to 0 for mass production	Real-time=0
SFC	It is used to tell the platform whether the SFC is ON or OFF by default	SFC=OFF
AutoScrolling	It is used to enable automatic scrolling, its value is Disable or Enable	AutoScrolling=Enable
ToolPath	It is used to specify the path of the developer's own tool vi	ToolPath=Demo\TesterVI\TST_tool.vi ToolPath=C:\Tools\TST_tool.vi
DebuggingGUIPath	It is used to specify the path of the debugging tool vi	DebuggingGUIPath=DebuggingTool.vi DebuggingGUIPath=C:\Tools\DebuggingTool.vi
DebuggingToolCFG	It is used to specify the path of the debugging configuration file	DebuggingToolCFG= DebuggingToolCFG.xlsx DebuggingToolCFG =C:\Tools\ DebuggingToolCFG.xlsx

GUI_Actuator1620x800.vi parameter name	Description	Example
FP_Show	It is used to tell the platform that this GUI needs to be displayed	FP_Show=True
RefreshTime	It is used to set the refresh time of GUI, unit is a millisecond The refresh time must set to 10 seconds for mass production	RefreshTime=10000
Real-time	It is used to set whether test results are displayed in real time, its value is 0 or 1 The Real-time must set to 0 for mass production	Real-time=0
Minimizable	It is used to disable the GUI minimized, its value is Disable or Enable	Minimizable=Enable
SFC	It is used to tell the platform whether the SFC is ON or OFF by default	SFC=OFF
delay	In loop mode, it is used to set the delay between this run and the next	delay=3000
SlotPosition	It is used to tell the platform which slot details information you want to print in the GUI, its value is 1,2,3,…32	SlotPosition=1
AutoScrolling	It is used to enable automatic scrolling, its value is Disable or Enable	AutoScrolling=Enable
ToolPath	It is used to specify the path of the developer's own tool vi	ToolPath=Demo\TesterVI\TST_tool.vi ToolPath=C:\Tools\TST_tool.vi
SFC_Menu	It is used to disabled the SFC menu, its value is Disable or Enable, default is enable	SFC_Menu=Enable
InitPicture	It is used to set the default image	InitPicture=SystemVI\sys_GUISourceCode\GUI_Actuator \GUI_Actuator_InitPicture.vi InitPicture=C:\Tools\ InitPicture.vi
StartButton	It is used to disabled the start button, its value is Disable or Enable, default is enable	StartButton=Enable
DebuggingGUIPath	It is used to specify the path of the debugging tool vi	DebuggingGUIPath=DebuggingTool.vi DebuggingGUIPath=C:\Tools\DebuggingTool.vi
DebuggingToolCFG	It is used to specify the path of the debugging configuration file	DebuggingToolCFG= DebuggingToolCFG.xlsx

		DebuggingToolCFG =C:\Tools\ DebuggingToolCFG.xlsx

amz_log_Daily.vi parameter name	Description	Example
LOG_PATH	It is used to set the log store paths It can be multiple paths, separated by semicolons	LOG_PATH=\\10.117.119.235\Moonshine\PVT\SMT\BFT\BFT03;..\..\..\LOG\
Module	It is used to define how to distinguish the different modules from serial number. Its format is as follows: **Module=ModuleName[Offset to search string/Length of special string]: Special string** For more module name, user can use ';' to separate different module, and use '–Else=**ModuleName**' for the rest	Module=WAN[3/3]:1AF/15S/1EA/1EQ/1F4/1F1/;WTN[3/3]:4RU/9I9-Else=WIFI

amz_log_single.vi parameter name	Description	Example
LOG_PATH	It is used to set the log store paths It can be multiple paths, separated by semicolons	LOG_PATH=\\10.117.119.235\Moonshine\PVT\SMT\BFT\BFT03;..\..\..\LOG\
Module	It is used to define how to distinguish the different modules from serial number. Its format is as follows: **Module=ModuleName[Offset to search string/Length of special string]: Special string** For more module name, user can use ';' to separate different module, and use '–Else=**ModuleName**' for the rest	Module=WAN[3/3]:1AF/15S/1EA/1EQ/1F4/1F1/;WTN[3/3]:4RU/9I9-Else=WIFI
LOG	It is used to set the logging mode, its value is STEP or Final, its default value is Final. If LOG=STEP, the single log vi is able to record the log line by line If LOG=Final, the single log vi will record the log after the test is completed	LOG=STEP

ZIPTXT	It is used to enable the log file compression feature, its value is TRUE or FALSE, its default value is TRUE	ZIPTXT=FALSE

amz_log_timelog.vi parameter name	Description	Example
LOG_PATH	It is used to set the log store paths It can be multiple paths, separated by semicolons	LOG_PATH=\\10.117.119.235\Moonshine\PVT\SMT\BFT\BFT03;..\..\..\LOG\
Module	It is used to define how to distinguish the different modules from serial number. Its format is as follows: **Module=ModuleName[Offset to search string/Length of special string]: Special string** For more module name, user can use ';' to separate different module, and use '–Else=**ModuleName**' for the rest	Module=WAN[3/3]:1AF/15S/1EA/1EQ/1F4/1F1/;WTN[3/3]:4RU/9I9-Else=WIFI

amz_FileSync.vi parameter name	Description	Example
exePath	It is used to specify the path of exe, its default value is "C:\Windows\System32\Robocopy.exe"	exePath=C:\Windows\System32\Robocopy.exe
WorkingDirectory	It is used to specify the working directory, its default value is "C:\Windows\System32"	WorkingDirectory=C:\Windows\System32
Source	It is used to define the source directory that needs to be synchronized, it is the absolute path.	Source= C:\Kazam\Log
Dest	It is used to define the destination directory	Dest=\\10.117.119.235\Log
Options	It is used to define the optional parameters of Robocopy.exe, its default value is "/mot:1 /mon:1 /e"	Options=/mot:1 /mon:1 /e
Log	It is used to define whether to generate log or not. Its value is enable or disable, its default value is disable	Log=enable

amz_RetentionPolicy.vi parameter name	Description	Example
Interval	It is used to define the interval between scanning file information, its unit is second, its default value is 3600	Interval=3600
days	It is used to define how many days to retain, and delete files older than n day, its unit is day. It can be a number less than 1, such as 0.1 day. Its default value is 14	days=14
Source	It is used to define the source directory that needs to be deleted, it is the absolute path.	Source= C:\Kazam\Log
Recursive	It is used to define whether to delete subdirectory files, its value is true or false and the default vale is true	Recursive=True
Log	It is used to define whether to generate log or not. Its value is enable or disable, its default value is disable	Log=enable

amz_Plug-in_SyncSystemTime.vi only sync up the time with server one time when Kazam boot up.

amz_Plug-in_SyncSystemTime.vi parameter name	Description	Example
Server_PATH	It is used to set the time server path	Server_PATH=\\10.117.119.235

log_SFC.vi need to develop by CM or supplier, and this vi need to save to the SFC folder, the developer need to create a new folder for their factory.

log_SFC.vi parameter name	Description	Example
LOG_PATH	It is used to set the log store paths It can be multiple paths, separated by semicolons	LOG_PATH=\\10.117.119.235\Moonshine\PVT\SMT\BFT\BFT03;..\..\..\LOG\
VI_PATH	It is used to set the log vi path	VI_PATH=..\Project\SFC\Compal\BFT\

10. Sequence Producer

The sequence producer is a component of Kazam that is used to generate new sequences and develop unified debugging tools. Its architecture is as follows:

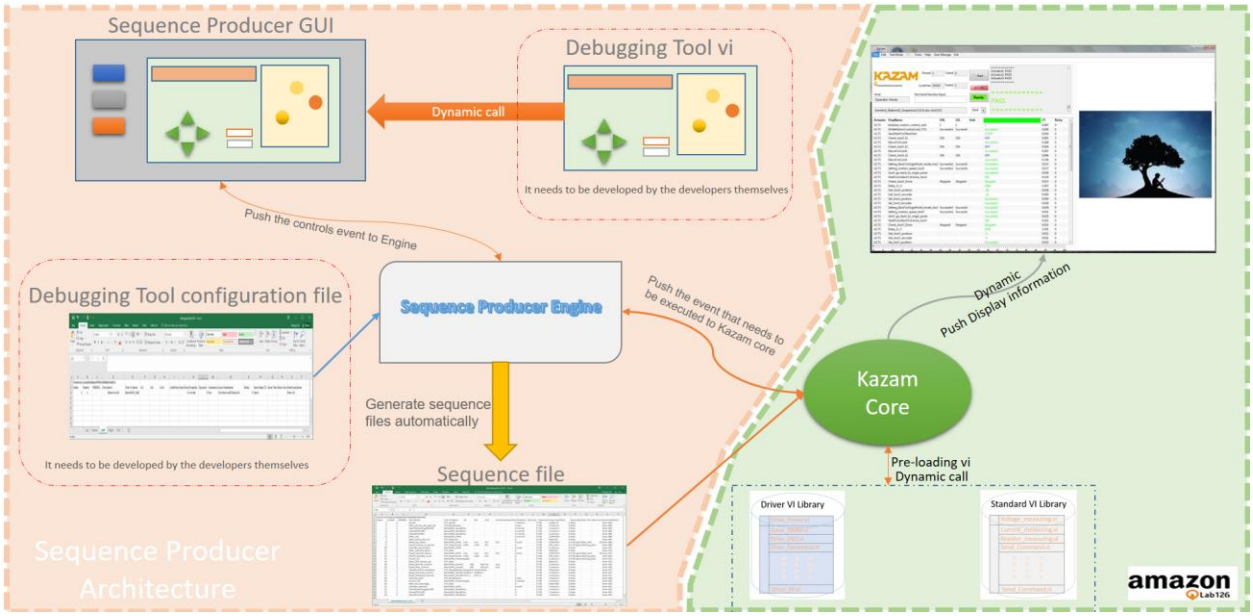


Figure 60

If the developer has completed the development of drive vi and test vi, then the developer only needs to complete the following two steps to complete the sequence producer development:

1, Create a debug tool profile in XLSX format

2, Develop the debugging tool GUI

9.1 Debugging Tool configuration file

The debugging tool configuration file in XLSX format, its contents the same as a normal TestPlan. The developer needs to manually create their own sequence on each sheet, and the sheet name will bind to the control in the debugging tool GUI.

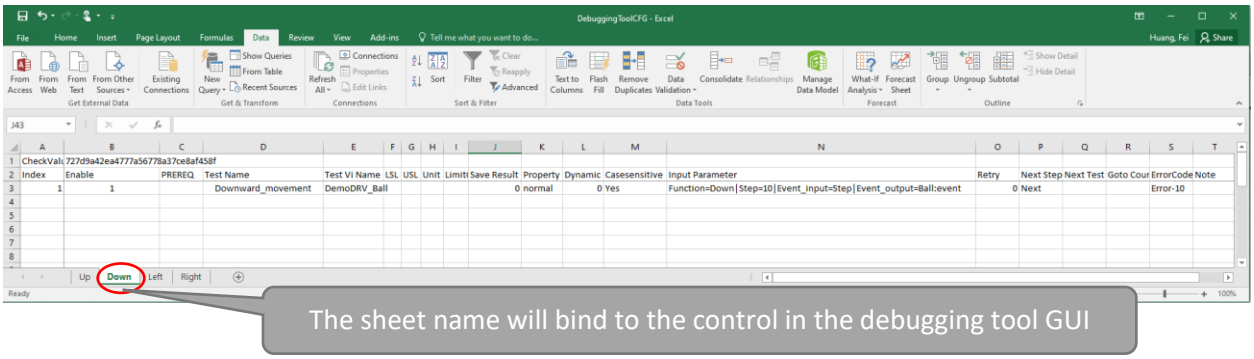


Figure 61

The developers can use the following two keywords to define input and output controls in “input parameters”:

Event_input=Name1;Name2;…Name N

Event_input is the keyword, cannot be changed.

Name1/Name2/Name N is the control’s label, the developers can customize, and it must have the same name as the control in the debugging tool GUI, it has to be unique.

Event_output=Name:event or Event_output=Name

Event_output is the keyword, cannot be changed.

Name is the control’s label, the developers can customize, and it must have the same name as the control in the debugging tool GUI, it has to be unique.

event is the type that defines the output control, and there are two types of output: value output and event output, the default is value output, which need not be specified.

9.2 Debugging Tool GUI

The developers need to develop their own debugging tool GUI, the controls name in this GUI are automatically bound to the sheet name in the configuration file, so if the developer needs to bind a control, the label name of the control must be matched to the sheet name of the configuration file. If the control does not need to be bound, it can name the control whatever you want.

Here is an example of a debugging tool GUI:

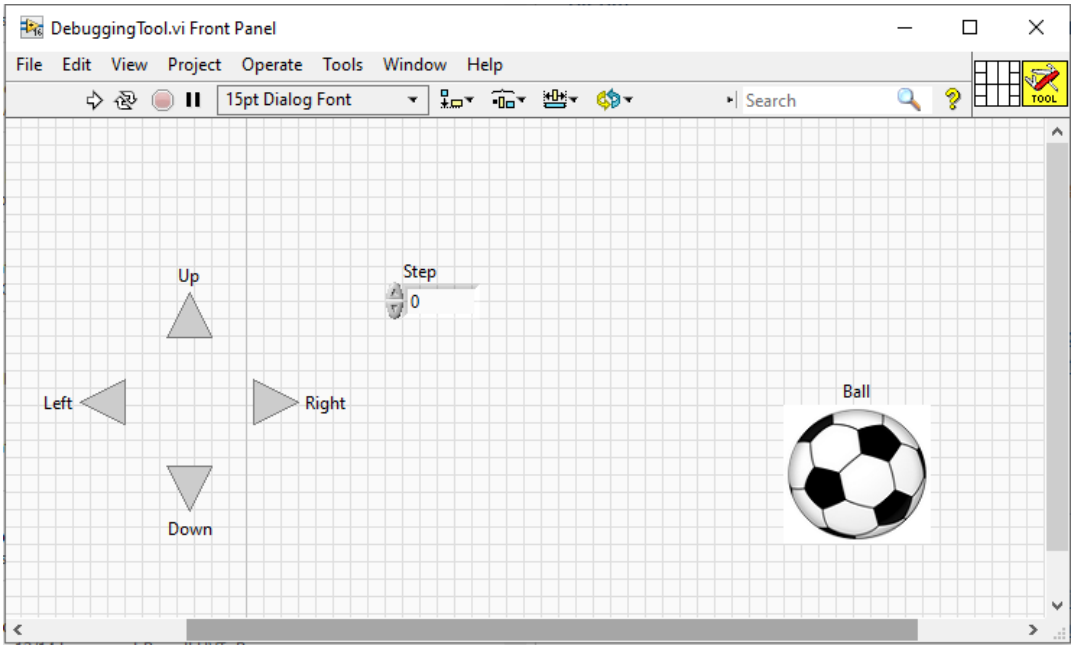


Figure 62

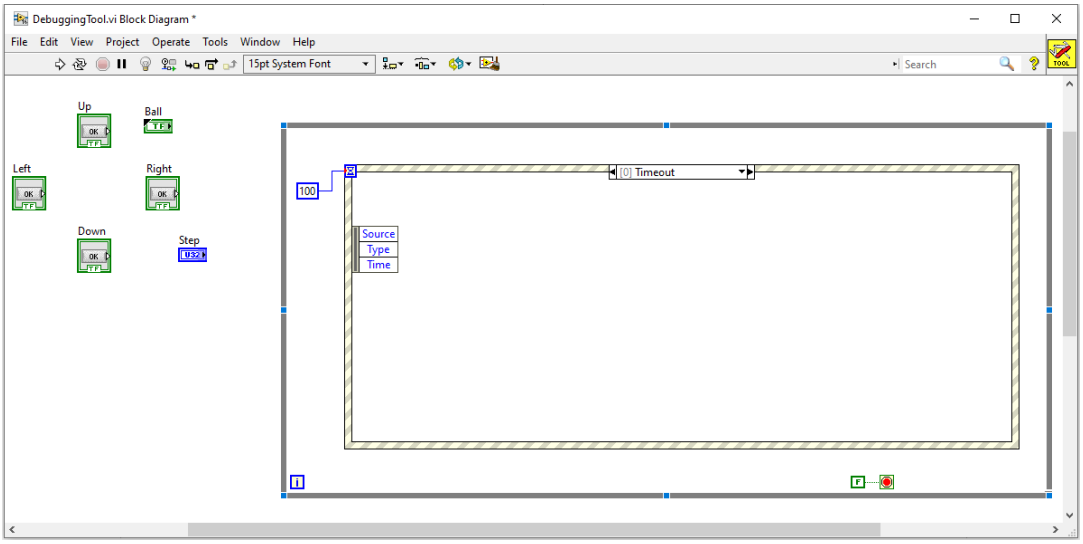


Figure 63

From Figure 62, there are only six controls are placed in the front panel. They are "up", "down", "right", "left", "Step" and "Ball".

From Figure 63, the "up", "down", "right", "left", "Step" and "Ball" are not connected anywhere. And

There is a blank event structure in the while loop, it ensures that this vi is always running when it is called.

9.3 How to configure sequence producer

The developer must place the debugging tools configuration file and debugging tools GUI in the project folder, Figure 64 shows the directory structure.

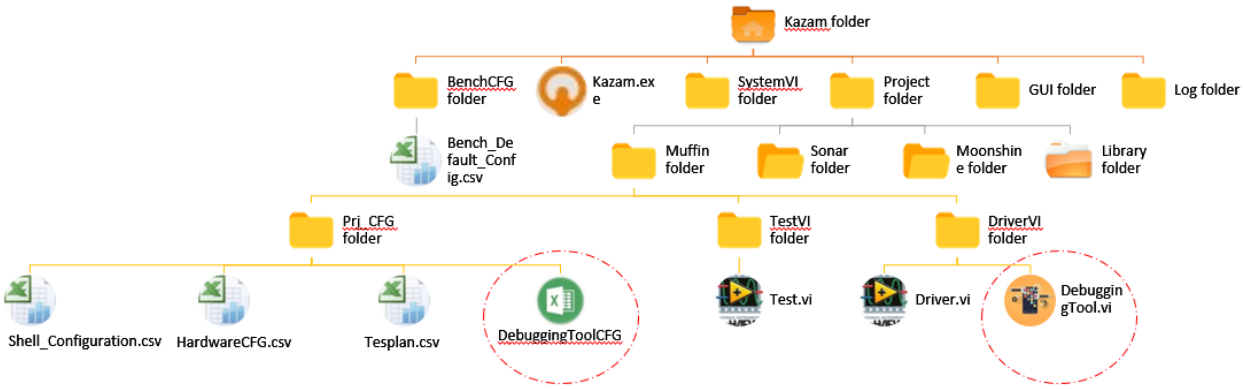


Figure 64

The developers can configure different debugging tools and their corresponding configuration files in each shell configuration file. Refer to 8.2.2 Other parameter for a description of display parameters

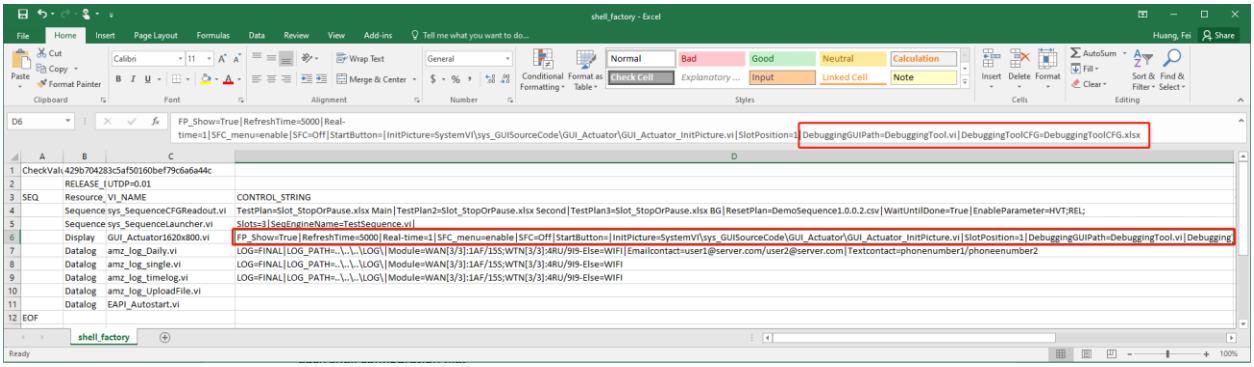


Figure 65

The user can open the sequence producer from the menu, Tool>Sequence Producer, it shows as Figure 67

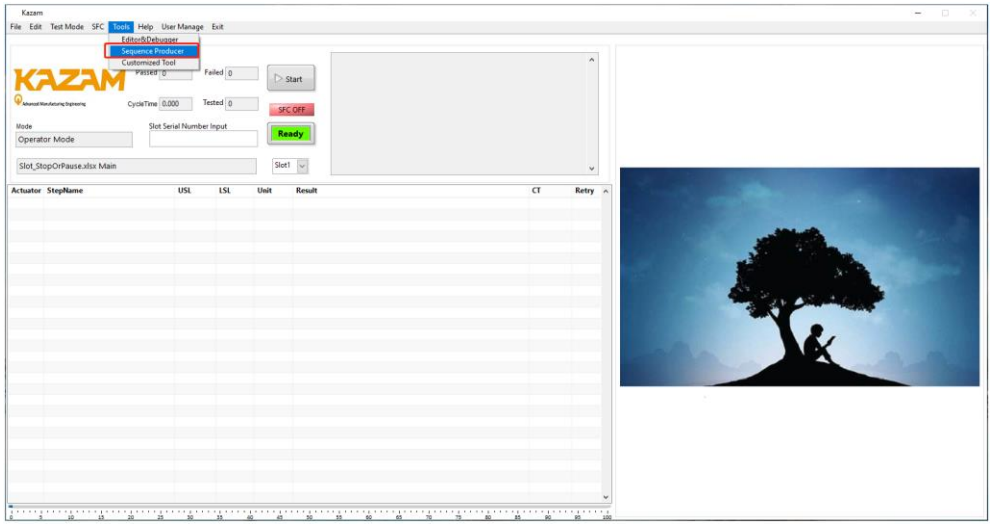


Figure 66

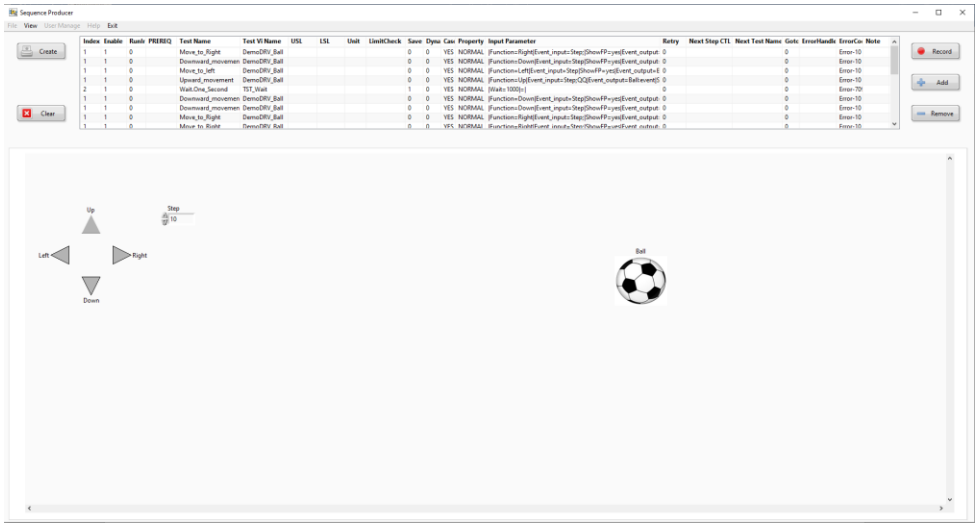


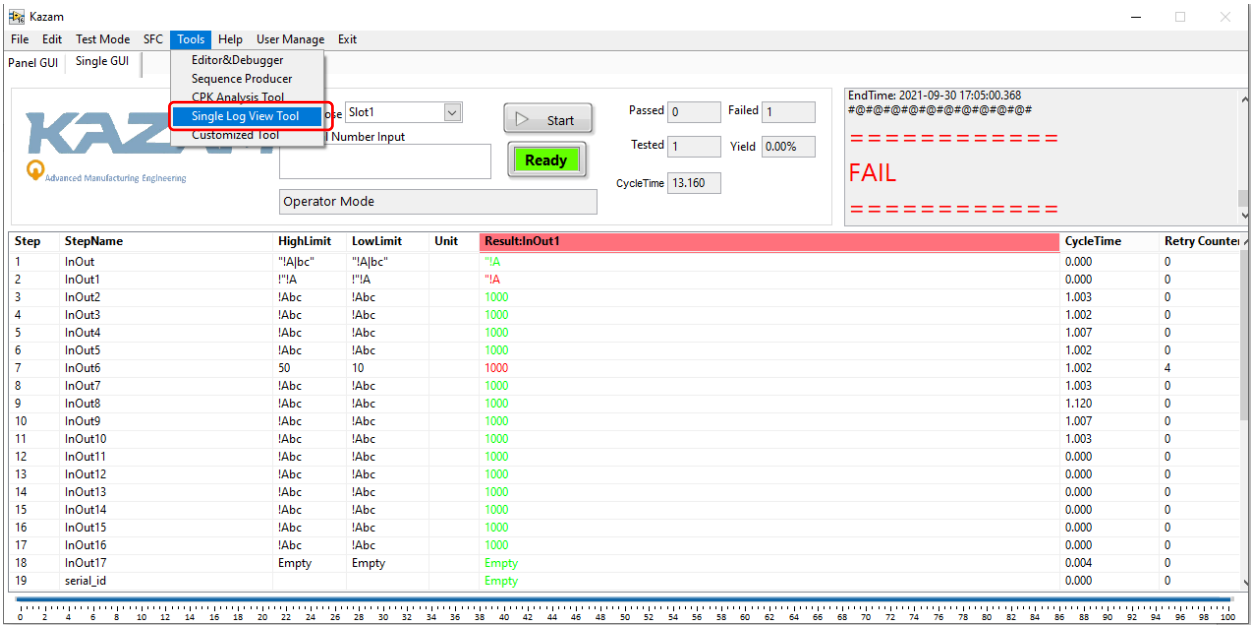
Figure 67

11. Kazam Tools

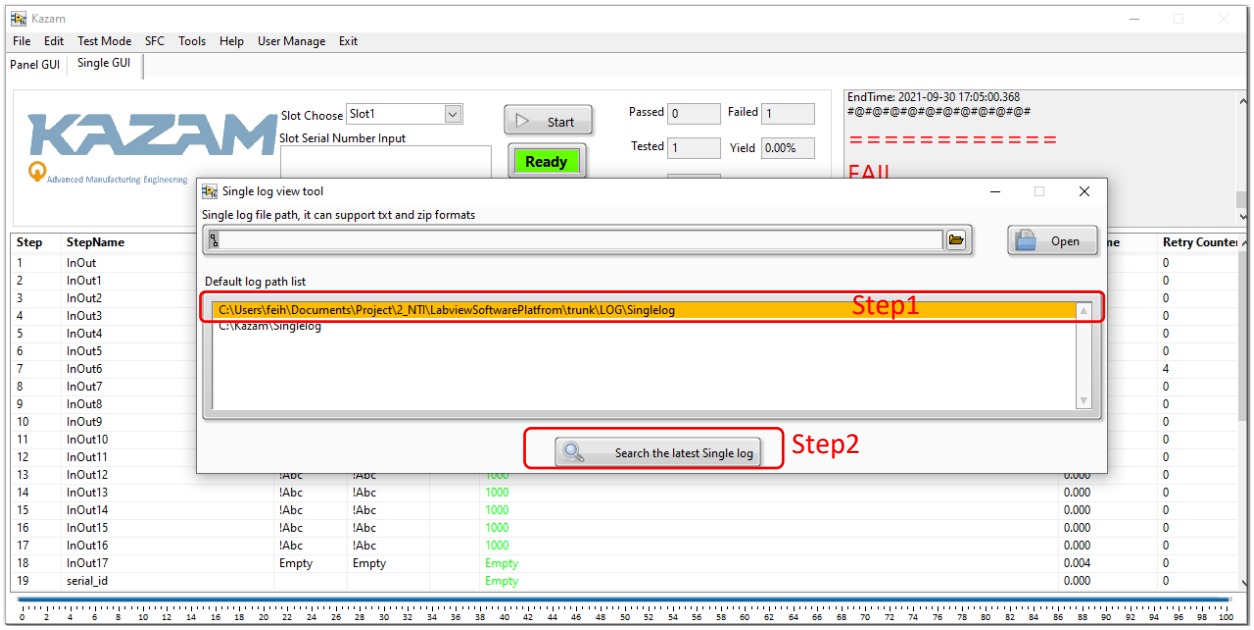
11.1 Single log browser

The single log browser can more efficiently view logs on the production line, it can support txt and zip formats, and support parallel browsing of multiple logs. The following is the instructions:

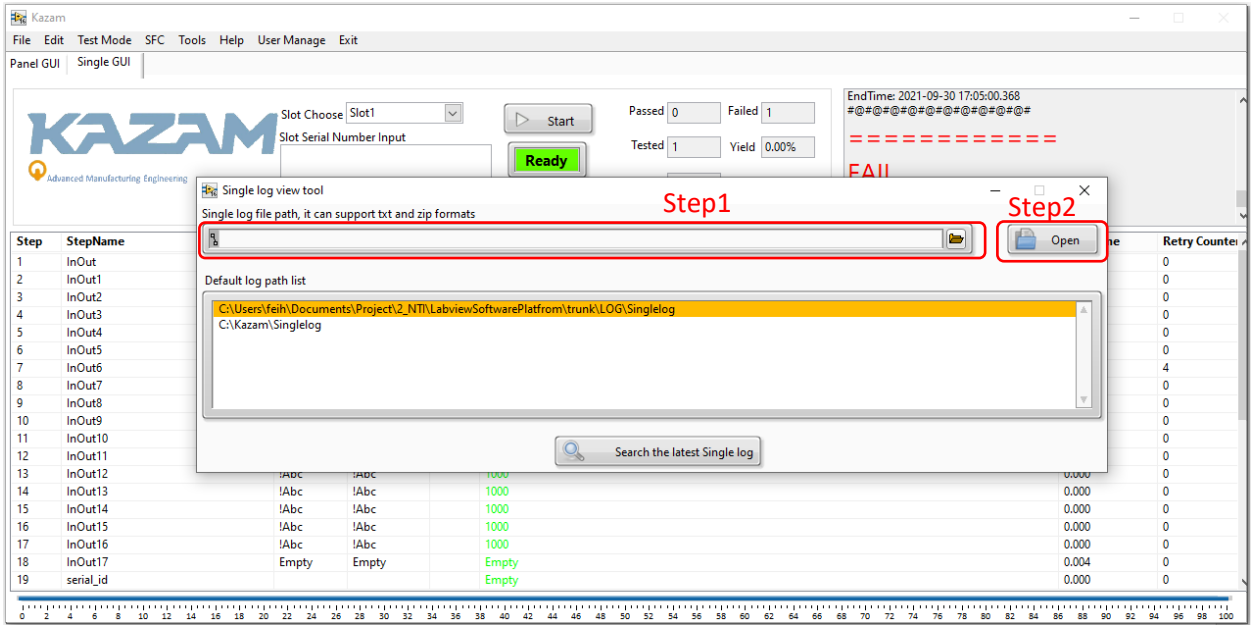
1, Open the log viewing tool from the menu



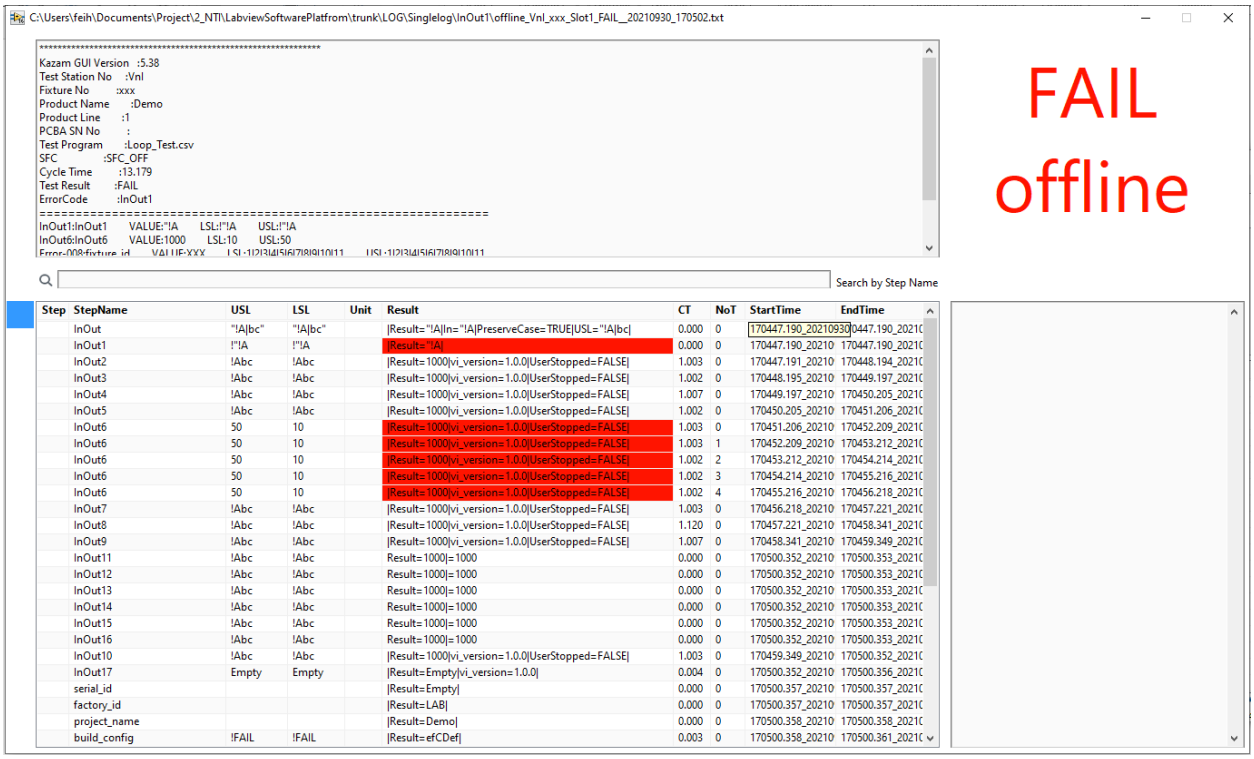
2, Select the default path to open the latest single log



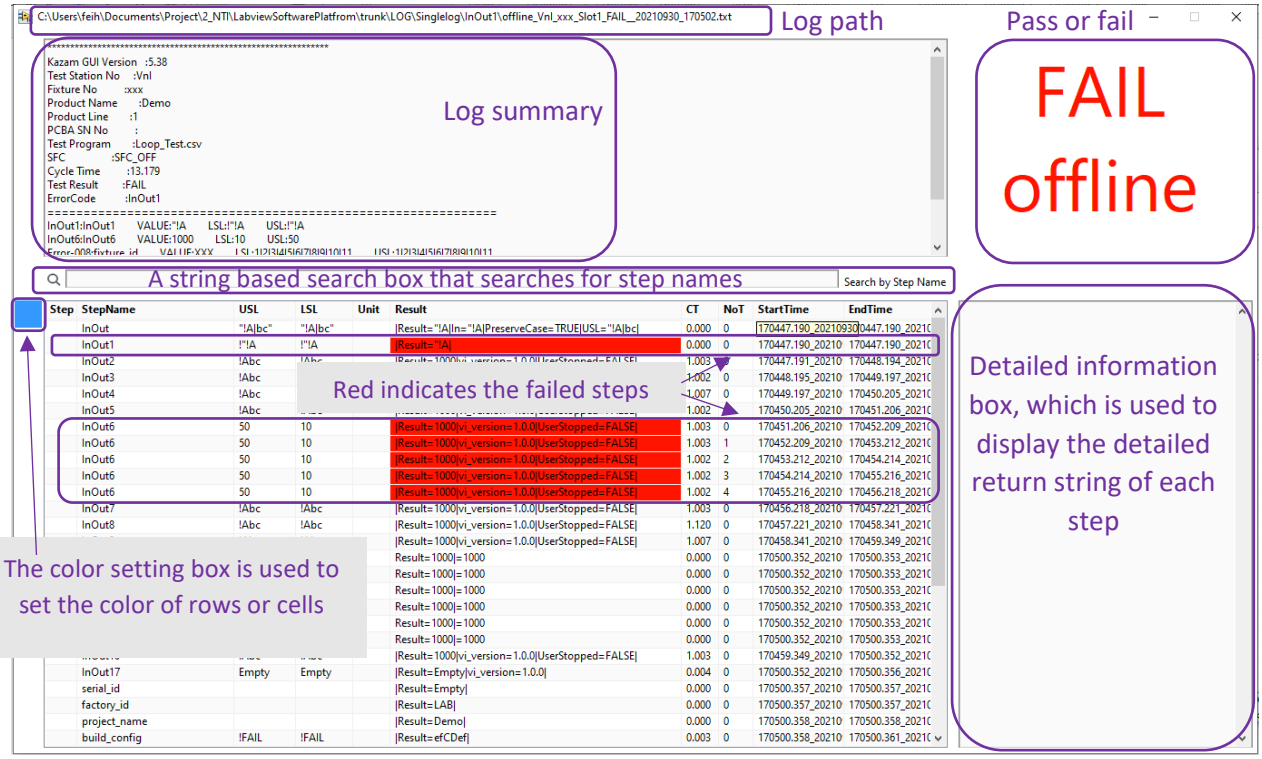
3, Select a specified single log file, it can support txt and zip formats



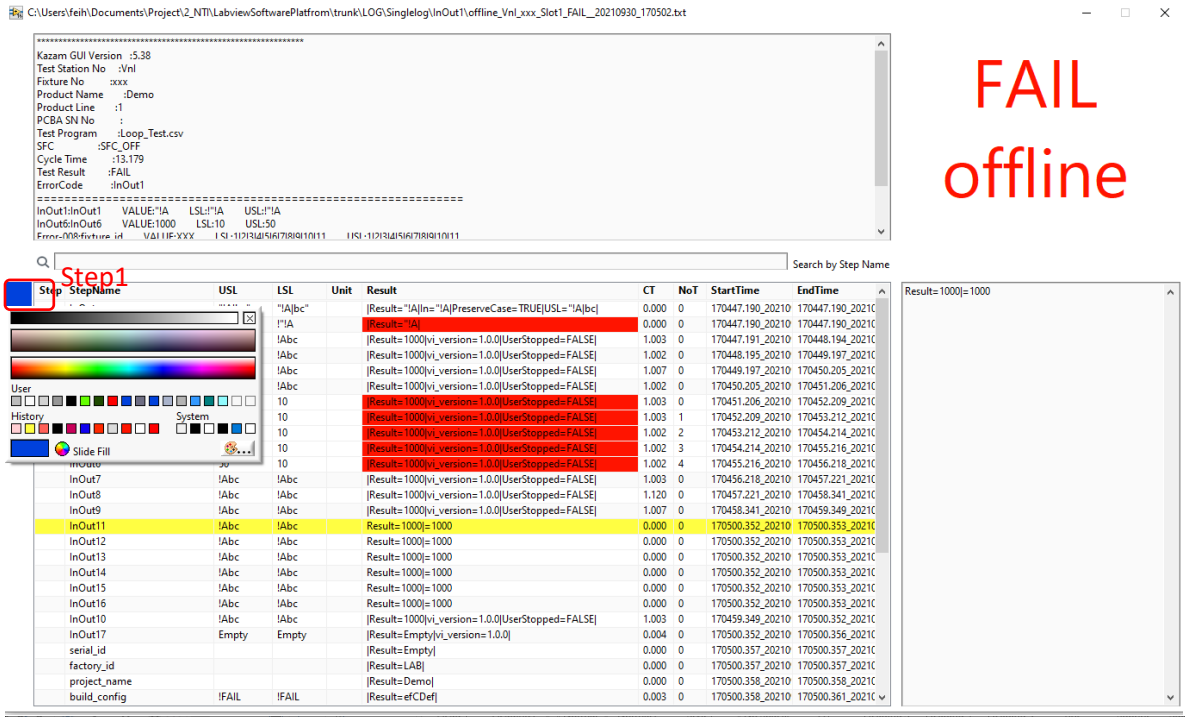
4, A single log browser will be opened

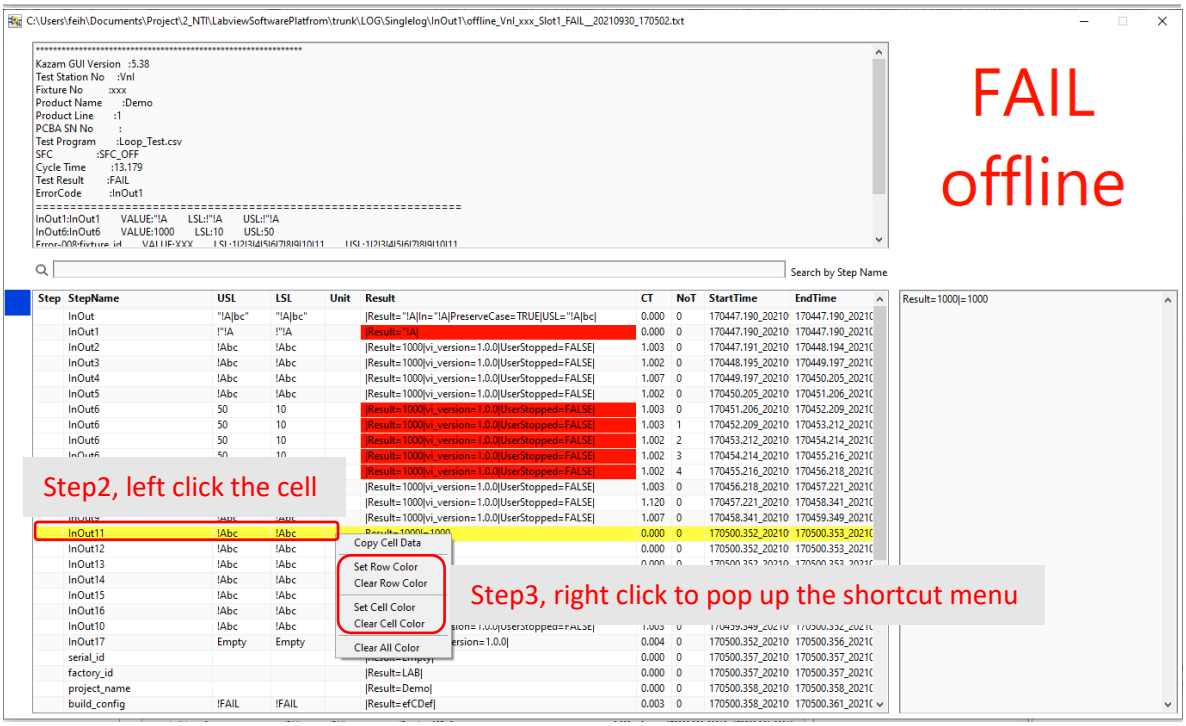


5, The following is an introduction to the log browser

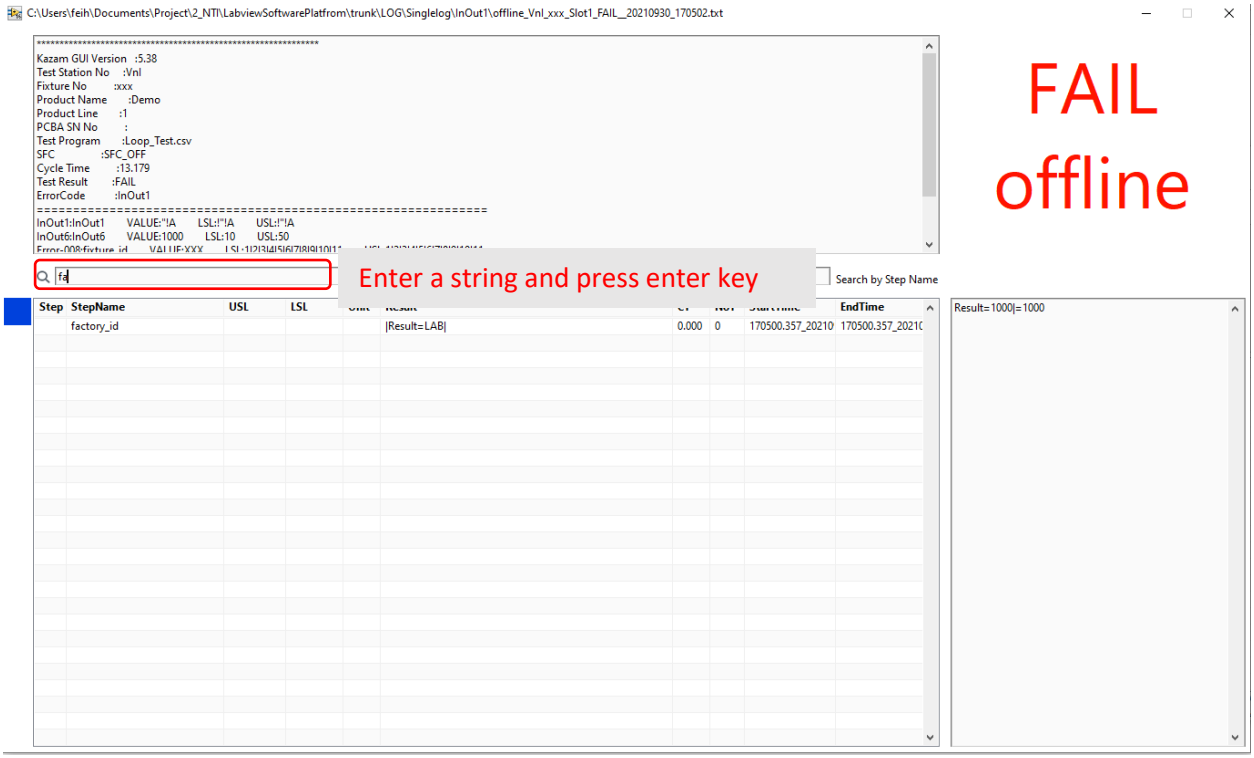


6, The following describes how to set/clear row or cell colors





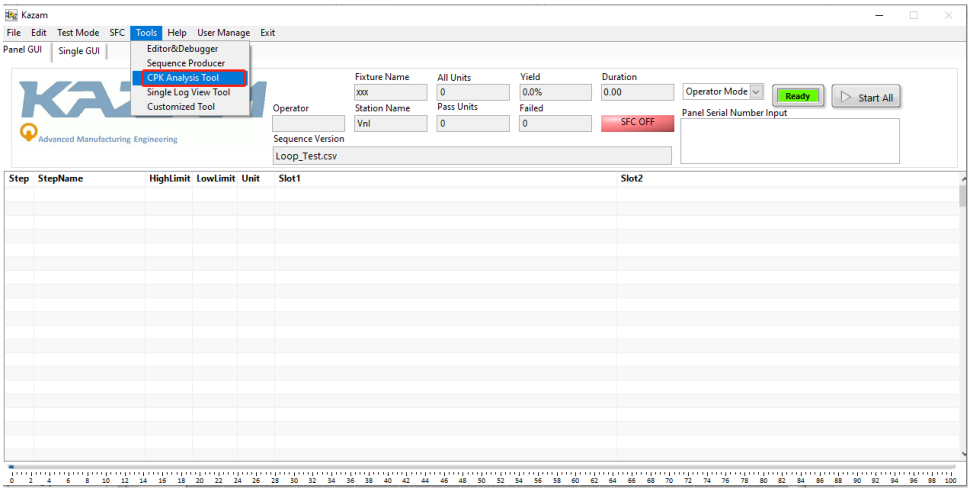
7, The user can use the search string feature to search for steps that contain this string in step name



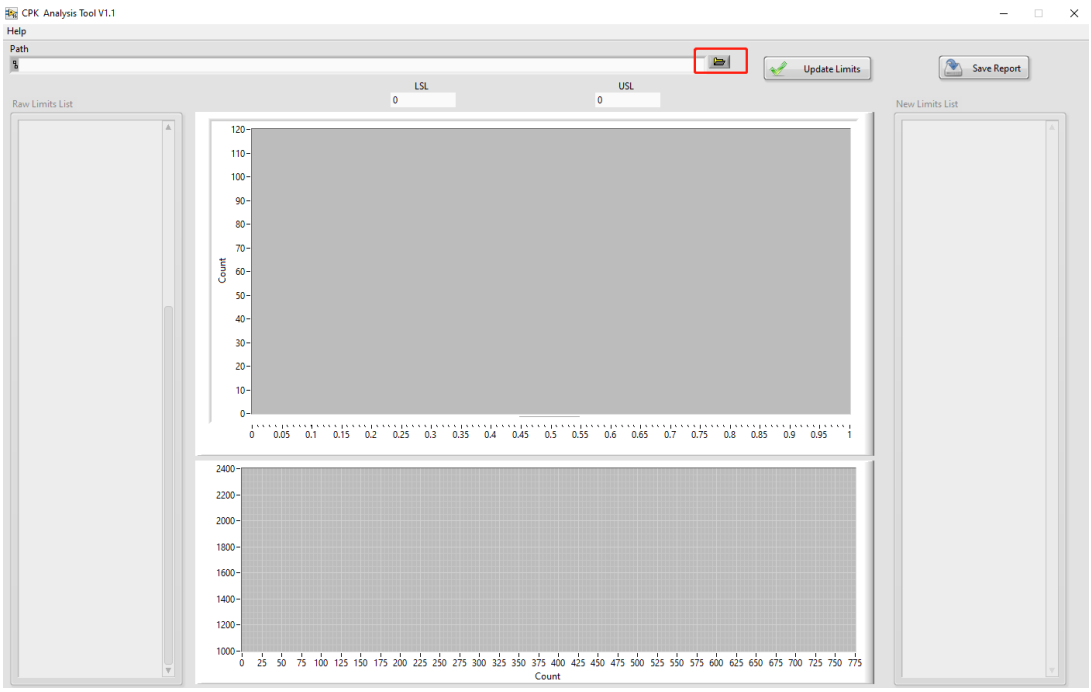
11.2 CPK Analysis Tool

The CPK analysis tool can help users quickly generate histograms, it can support direct import of daily logs to analyze CPK. And the users can also use it to generate a new limits adjustment suggestion report.

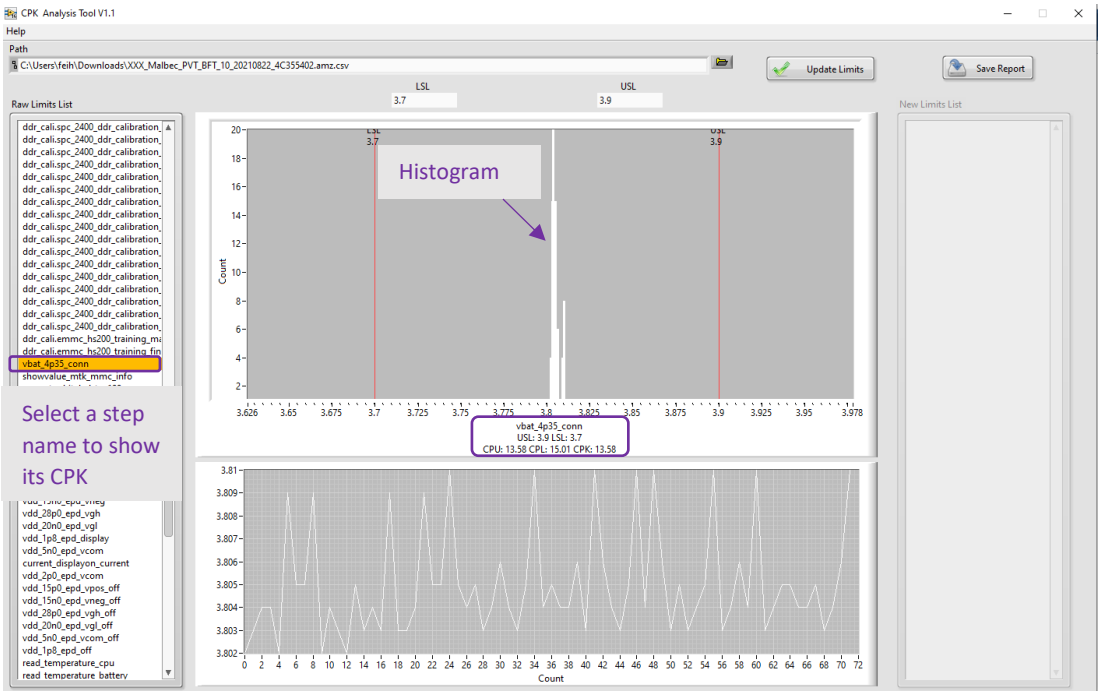
1, Open the CPK analysis tool from the menu



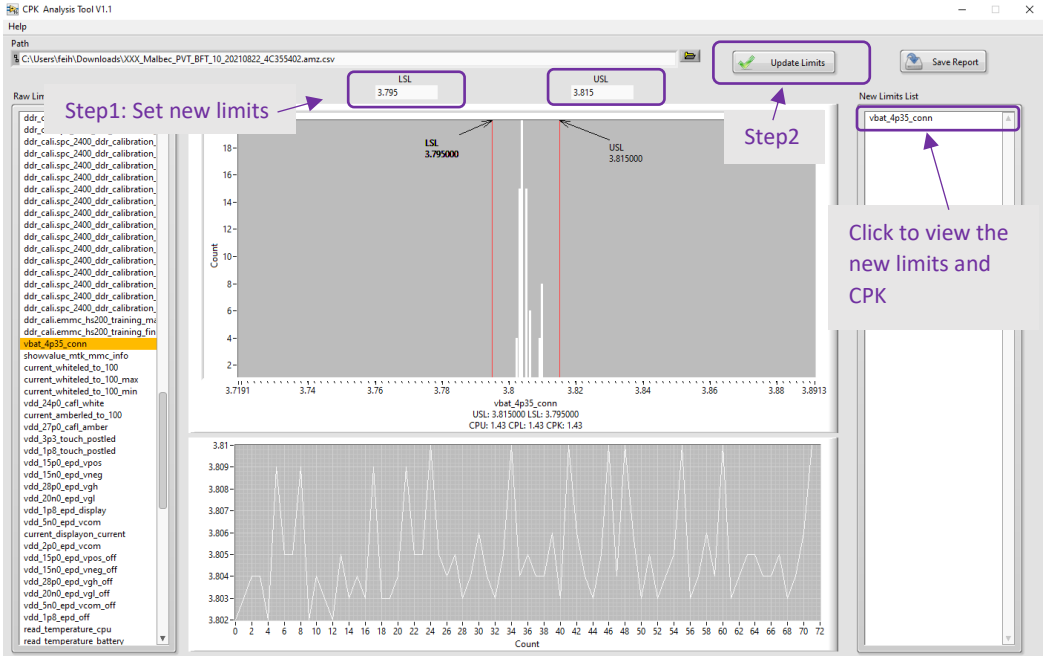
2, Select a daily log



3, Select a step name to show its CPK and histogram



4, The users can adjust limits according to the following steps



5, Click save report button, CPK tool will automatically generate CPK report to the same level directory of log file, and users can open folders under the same level directory to find reports in HTML format