Test Engine Instruction

Version	Date	Author	Comment
0.1	16/Jan/06	IvanGan	Initial Draft
0.2	16/Jan/26	IvanGan	Add load flow, GH part and Additional part

1. Instruction

Test Engine execute command from REQ. It can control HW, as Relay, measure, dmm, frequency, etc.
Engie will load libSocjetDev.dylib to creat TCP connecting with ARM though port 7601 to communicate with DUT and set ZMQ service.
Also, there is a real-time current record from ARM though port 7603 with ArmDL.lua to call dynamic library ibArmDL.dylib;
Get air pressure from server from tcp://169.254.99.65:6661
For Sip Manual fixture, it can get SN from socket with IASocketLua.lua to call libIASocket.dylib which is located /usr/local/lib

2. Load Flow

1st: load config table, normally it is /config/socket_zmq.lua
2nd: Set REP server
3rd: HW ARM tcpip connect(port:7600), and set HW PUB server, normally, console/fixture/socket_pub.lua
4th: load libSocketDev.dylib to connect DUT(port:7601) and set DUT PUB
5th: Read calibration factor from EEPROM and update to HWIO table
6th: Load each block which defined in function_table.la
7th: Poller start to wait for RPC command and PUB heartbeat

3. Calibration Factor Update

Engine will read factor from EEPROM and save data to cvs file, origin ARM data will save too.
Now, all factor has been write to two different areas of EEPROM, when reading factor, Engine need to read the two areas and verify the data be same or not.

* updateVTFactor()
This function will update factor to HWIO measure table
* updateVoltFactor()
Power supply factor, including BATT, VDDMAIN, PP_RECT and Eload will be updated
* updateCurrFactor()
Current measure factor will be updated
* Update_Datalogger_Factor()
Real-time current logger and current measure with datalogger will be updated
* Update_DMM_Low_Curr_Factor(),
DMM low current (100uA) measurement factor will be updated
* Update_DMM_High_Curr_Factor()
DMM low current (2mA) measurement factor will be updated
* ReadDateFromARM(),
Read the date and version of Calibration
* CheckBoardSN()
After done calibration, each board must not be changed, so SN of each board has be written to EEPROM. Engine will read SN from board and check it matched or not.

4. Function Instruction

For all function instruction, please refer file “TestEngineRPC.pages”, Here just instruct parts of the functions.
All functions which can be called by sequencer or debug panel are defined in “function_teble.lua”.

Callback Part

1). start_test

When sequencer start test, sequencer will publish “sequencer start” message and then it will call this function, It unnecessary to define this in test plan.
In this function, we will set the default status of fixture to wait for testing.
Also, it will return the log file path to sequencer for pudding(Send the log to DataBase)

For the project IA760, IA473 and IA173, ARM real-time current record is requirement, so ARM datalogger will be start to record.
For sip socket, SN will be read form eeprom.

** note **
For the publish message of sequencer, please refer the file “PublisherSpec.pages” and “SequencerRPC.pages” for RPC message.

2). end_test

After test finished, end_test will be called by sequencer.
For the project IA760, IA473 and IA173, datalogger will be closed and LED will be turned on.

DUT Part

1). diags

Send diags command to DUT.
Normally, it will wait for response including “] :-)”, but if the command is or contain string “sleep”, “playaudio”, it will send command without waiting.

2). parse

Parse the response of DUT with pattern file.
execute step:

a). Engine will search the temple table, if there is no key named pattern file, then engine will parse temple file which is named with diags command, the extension is "txt".
it will assert when failed parse temple file
b). Get the table value of key named pattern file in temple table, then get match format from value table with key be param1.
c). if there is no match format, it will assert.
d). return the match result.

Parse pattern file.

Engine will replace {{xxxx}} with .- in pattern file. So please make sure the format of pattern file is same with response from DUT. Even if there is miss a space, it will make parse fail.

e.g.

If the last diags command is "chipid", then the pattern file is "chipped.txt" which is located in "LuaDriver/Driver/Match/"

chipid.txt

```
Chip  ID: {{chipid}} Version: {{Version}}
Die   ID: {{dieid}}
Fuse  ID: {{fuseid}}
ECID    : {{ecid}}
Raw ECID: {{raw_ecid}}
```

After pattern file parse, we will get a table in Lua as following(Discard the time stamp):

```
TemplatePatternTable = {
        chipid.txt = {
                chip   id = "Chip  ID: (.-) ",
                Version = "Chip  ID: .- Version:(.-)[\r\n]+",
                Die   ID = "Chip  ID: .- Version:.-[\r\n]+ Die   ID: (.-)[\r\n]+",
                Fuse  ID = "Chip  ID: .- Version:.-[\r\n]+ Die   ID: .-[\r\n]+ Fuse  ID: (.-)[\r\n]
+",
                ECID = "Chip  ID: .- Version:.-[\r\n]+ Die   ID: .-[\r\n]+ Fuse  ID: .-[\r\n]+
ECID    : (.-)[\r\n]+",
                Raw ECID = "Chip  ID: .- Version:.-[\r\n]+ Die   ID: .-[\r\n]+ Fuse  ID: .-[\r\n]
+ ECID    : .-[\r\n]+ Raw ECID: (.-)[\r\n]+",
        }
}
```

---

3). smokey

smokey is a special diags function. it will add string "smokeyshell -f nandfs:\\AppleInternal\\Diags\\Scripts\\X527\\" before param1 as a new command and then call diags function to send this command to DUT.

4). calculate

calculate will return result of formulate form param1, the form should be LUA.

In csv test plan, param1 would be: *if {{eeprom}}=={{mlbsn}} then return “PASS” else return “FAIL” end*.
Sequencer will replace value to the key 0f {{xxxx}}, then the string would be:*if “ABCDEFG”==“ABCDEFGH” then return “PASS” else return “FAIL” end*
and send the string to Engine, engine will do this Lua string and return result “FAIL” to Sequencer.

** note **
For more detail info on {{}}, please refer the document of “TestSequenceSpec.pages”

Fixture Part

All of the function so support by ARM. For more detail info, please refer the file “ARM Board Command List_V1.18_neal.xls” and “IA473 ARM Board Communication Protocal V0.6.docx”

GroundHog Part

Station info will be got from /vault/data_collection/test_station_config/gh_station_info.json by libGhInfo.dylib

1). station

It will return Station type. e.g. : PREFCT, CombineFCT, Gatekeeper …

2). fixturetype

It will show PANEL fixture or SIP fixture. e.g. : PANELFIXTURE, SIPFIXTURE

3). fixturename

It will return string combined by production name and fixture type. e.g. : IA173_PANELFIXTURE

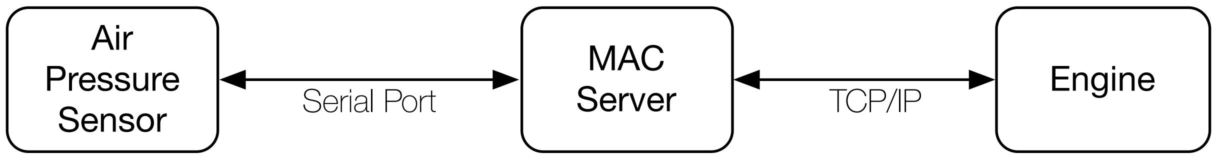
4). channel

It will return slot ID. e.g. : 0,1,2…

Additional Part

1). reference

reference is used to get AIR Pressure from server which will get value from sensor though Serial port. So if user want to use this function, please setup the instrument and run the server.



When call this function, command “read\r\n” will send to server and value will return.