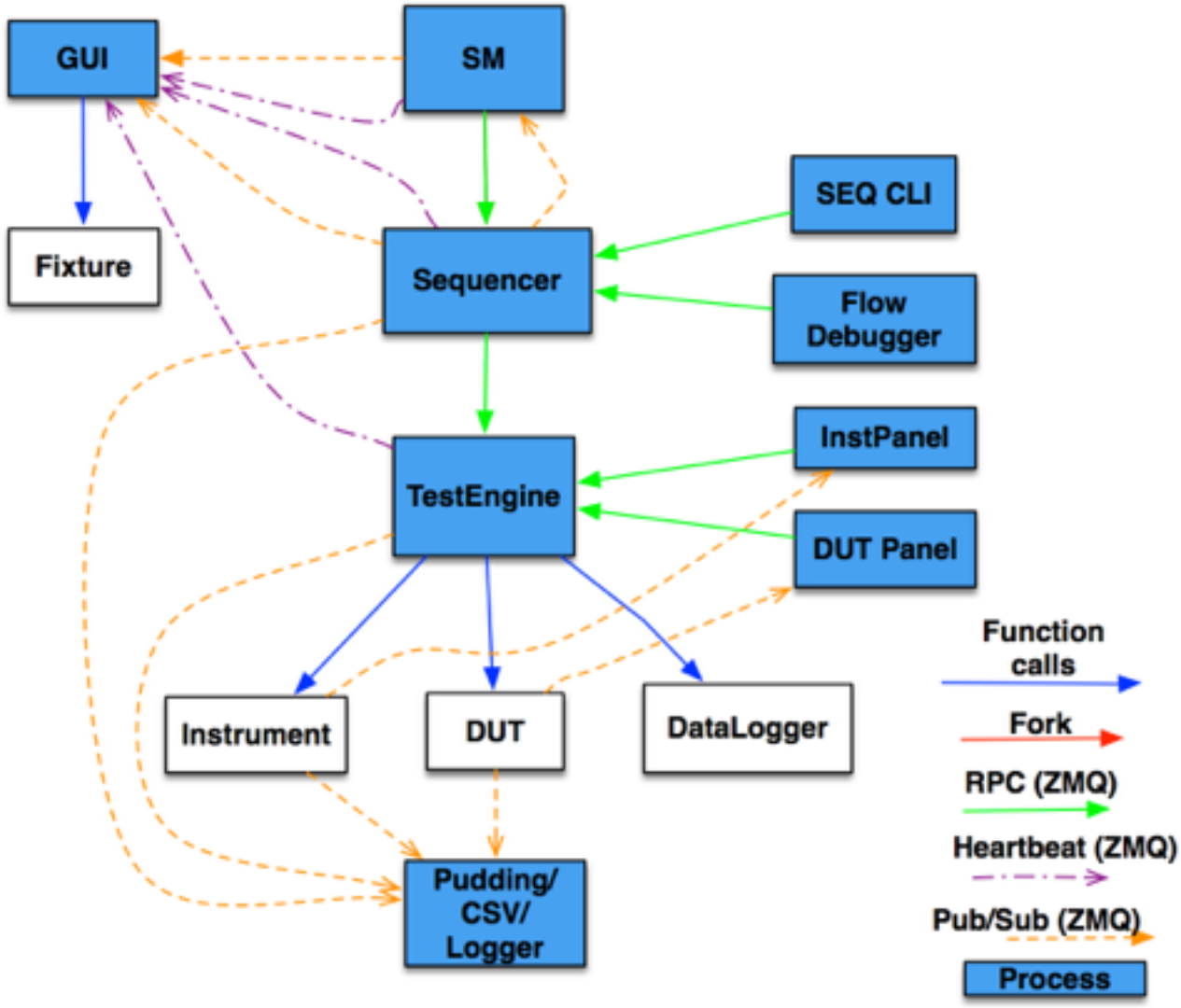
TM Instruction

Test Platform Based On ZMQ

Version	Date	Author	Comment
0.1	16/Jan/06	IvanGan	Intial Draft
0.2	16/Jan/25	IvanGan	Add Global Function Instruction

TM Architecture

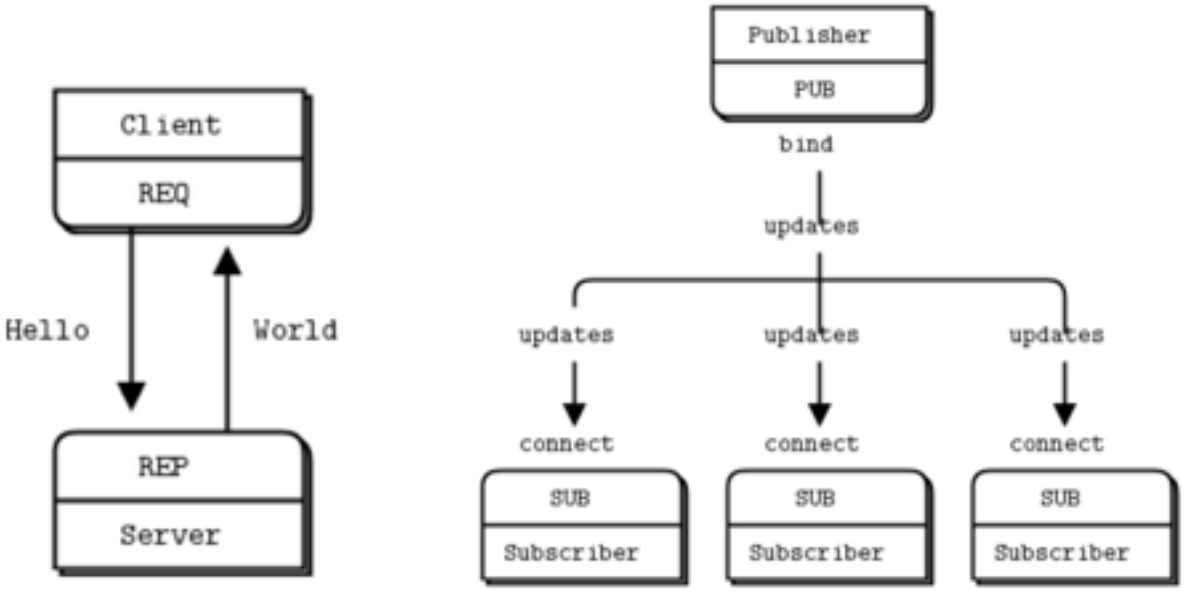
TM includes 8 parts, Sequencer, Test Engine, State Machine, data logger, Pudding, Debug Panel or CLI, GUI and fixture controller. Data transmit between each part is based on ZMQ.



ZMQ

Distributed Messaging.

Here is the basic use, Request-Reply and Publish-Subscribe, for more detail, you can get it from http://www.http://zeromq.org/



File	Content
ZMQ Interface.numbers	Define the port for each function

Sequencer

Sequencer controls test flow defined by sequence file (test-plan). The name of the sequence file must be of the format [name]__[version].csv. Name is the sequence name. The version is the sequence version. The name and version will show up in GUI and be reported to PDCA. The separator is double underscore. It's OK to have single underscore in the sequence name but not double underscore.

It's OK if there is a dot in the version. For instance: short_plan__1.0.csv.

Sequencer located: /python_sequencer/x527/sequencer/sequencer.py

Pudding: /python_sequencer/x527/loggers/logger.py

File	Content
TestSequenceSpec.pages	Test Sequence file define, user need to following the instruction to set test plan format
TestEngineRPC.pages	Here list the function which can be supported by engine. Please make sure your function used in test-plan be listed here
SequencerRPC.pages	Data format for Request-Reply of ZMQ, This is command to operate Sequencer
PublisherSpec.pages	Data format for Publish-Subscribe of ZMQ. It publishes the statue of test item.

Test Engine

Test engine will take action after get REQ from Sequencer or Debug panel. Normally, test engine will send command to ARM or DUT to do test.

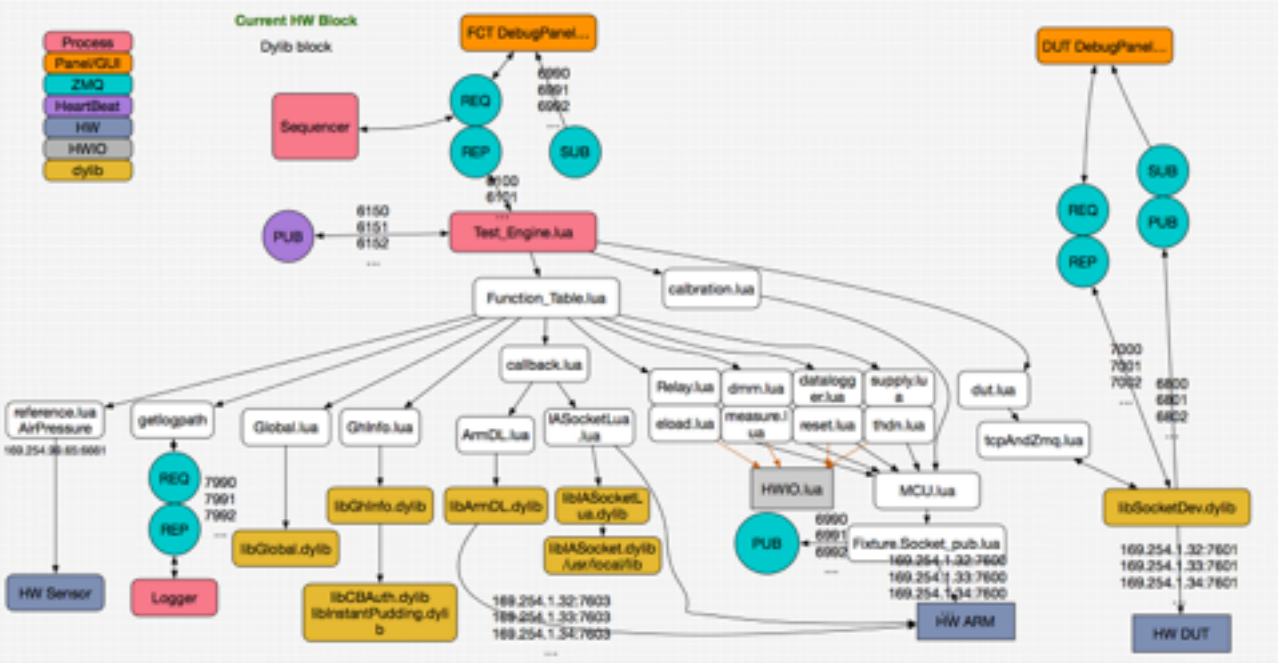
Record real-time current from FPGA.

Get Air pressure from server.

Read SN from socket in Sip manual fixture.

…

Engine located: /LuaDriver/Driver/Test_Engine_C_Zmq.lua



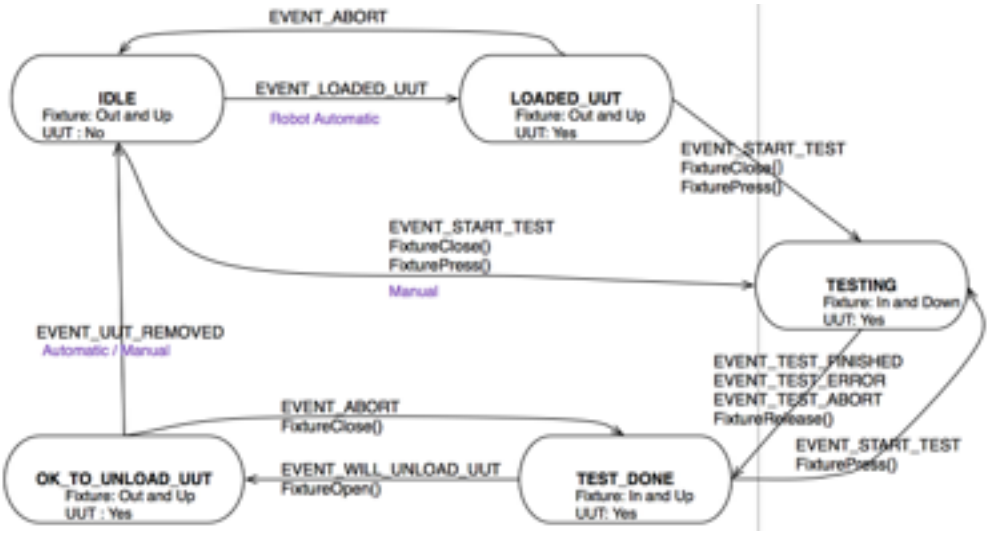
File	Content
TestEngineInstruction.pages	Define the function and usage

State Machine

State machine observe testing state, Only one state at a time, the state can change from one to another when initiated by a triggering event or condition. Following is current state machine. It will monitor REP and SUB ports to switch state.

For SIP automatic, Panel automation, SN will be get from GUI, then send to Sequencer while send command to sequencer.

State Machine located: /LuaDriver/StateMachine/StateMachine.lua



File	Content
StateMachineRPC.pages	Define data format for State Machine

Data Logger

Data logger monitor Sequencer, HW and DUT PUB to save log, including Flow log, Flow_Plain log, UART log and cvs log.

For UART log, FPGA add a time stamp before each line of DUT output. Data logger will convert the stamp from UTC time.

The other's is will add time stamp with local time.

Located: /LuaDriver/datalog/datalog.lua

Running Environment

Lua and Python running time are necessary.

Editor	Version
Lua	5.1.4
Python	2.7.0

Also there are some plug-ins are required:

Editor	Plug-in	Usage	install
Lua	dkjson	json data parse	luarocks install dkjson
	bit32	bit operation	luarocks install bit32
	llthread	thread manager	luarocks install llthread
	lfs	lua file system	luarocks install luafilesystem
	luasoket	lua tcp socket	luarocks install luasocket
	penlight		luarocks install penlight
	lzmq	lua zmq	luarocks install lzmq
Python	zmq	python zmq	easyinstall pyzmq

Also you can copy these file from tm_platform_v2/environment to the related folder.

Global Function

Here list the global function for Test Engine, user can call it directly.

Function	Usage
Now()	Get current time stamp
bit_and(x,y)	bit operation
bit_or(x,y)	bit operation
bit_not(x)	bit operation
bit_xor(x)	bit operation
bit_nor(x)	bit operation
BitSelect(x,y)	bit operation
set_bit(x,y)	bit operation
clr_bit(x,y)	bit operation
get_bit(x,y)	bit operation
LockInstrucment("xx")	lock (Different thread lock in same process)
UnlockInstrucment("xx")	unlock
floatConversion(x)	float convey to byte
floatToNumber(x)	byte convert to float
floatToFixed(x)	float convert to fixed
fixedToNumber(x)	fixed to float

For more detail info, please refer the following file:

File	Content
function list of Global.pages	Global functions instruction