```
#-------------------------------------------------------------------------------
# Name:        module2
# Purpose:      Use S-parameter data to de-embed test fixture characteristics.
#               Extract Tx Signal data for specification validation.
#               Uses PyVisa networking to communicate with measurement instruments.
#
# Author:      RAWINTER
#
# Created:     07/09/2014
# Copyright:   (c) RAWINTER 2014
# Licence:     <your licence>
#-------------------------------------------------------------------------------
#!/usr/bin/env python

def main():
    pass

if __name__ == '__main__':
    main()

# import python system modules
import sys, visa, time, datetime, string, subprocess, decimal, math, os

# Parse required command line arguments
if len(sys.argv) == 8:
    SN =      sys.argv[1]
    ID =      sys.argv[2]
    maskrev = sys.argv[3]
    skew =    sys.argv[4]
    temp =    sys.argv[5]
    IDV =     sys.argv[6]
    pattern = sys.argv[7]
else:
    print "Usage 'ProgramName.py SN ID Maskrev Skew Temp IDV pattern'\n"
    sys.exit(1)

sys.path.append("C:\PythonSV\misc\VRG\JCK")
sys.path.append("U:\FNV\IODV_Data")
sys.path.append("U:\FNV\PythonScripts")

# import user defined programs, modules
#import dummy
from Agilent_RF_Switch_Module_JCK import RF_Switch_Channel_Select
from DPOJET_FNV2 import DPOJET_JITTER_setup

# User Vars
datadir = "U:\FNV\PythonScripts"
scopepath = "C:\Autotest\FNV"
testname = "TX"
devname = "JCK"
formatstring =
"%s,%s,\"%s\",%s,%s,%s,%s,%s,%s,VMSEDQ%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s\n"
volts = "typ"
mathfunc1="arbflt1(ch1)"
mathfunc2="Math1-MEAN(Math1)"
filterpath="C:\Users\Public\Tektronix\TekApplications\SDLA\output filters\\"
filterfile="sdlatp1.flt"

if "k" in pattern:
```

```
    patlen = 2
    pattern = "clock"
else:
    patlen = 128
    pattern = "prbs7"

now = datetime.datetime.now()
timestamp = now.strftime("%Y%m%d%H%M%S")
#print "timestamp=", timestamp

# Connect to remote instruments
jbert = visa.instrument("TCPIP0::chipset4903b1.amr.corp.intel.com::inst0::INSTR")
scope = visa.instrument("TCPIP0::JF04EOSC0718.amr.corp.intel.com::inst0::INSTR")

# set-up scope measurements
scope.write("RECALL:SETUP \"JCK_IODV_TX_1.set\"")   # recall scope set-up for TX
waveforms

# open 2 file types. One for waveform, second for scope measurement parameters
filenamedata =
"%s\%s_%s_%s_%s_%s_%s_%s_%s_%s.csv"%(datadir,devname,testname,timestamp,maskrev,skew,
volts,SN,temp,pattern)
f = open(filenamedata, "w", 0)

for lane in range(0,32):
    filenamewfm1 =
"%s\%s_%s_%s_%s_%s_%s_%s_%s_%s_ch1_DQ%s"%(scopepath,devname,testname,timestamp,maskre
v,skew,volts,SN,temp,pattern,lane+32)
    filenamewfm2 =
"%s\%s_%s_%s_%s_%s_%s_%s_%s_%s_ch2_DQ%s"%(scopepath,devname,testname,timestamp,maskre
v,skew,volts,SN,temp,pattern,lane+32)
    filenamewfm3 =
"%s\%s_%s_%s_%s_%s_%s_%s_%s_%s_ch3_DQ%s"%(scopepath,devname,testname,timestamp,maskre
v,skew,volts,SN,temp,pattern,lane+32)
    filenamewfm4 =
"%s\%s_%s_%s_%s_%s_%s_%s_%s_%s_ch4_DQ%s"%(scopepath,devname,testname,timestamp,maskre
v,skew,volts,SN,temp,pattern,lane+32)

    RF_Switch_Channel_Select(lane)
    scope.write(":CLEAR ALL")
    scope.write(":SELECT:CH1 ON")
    scope.write(":SELECT:MATH1 ON")
    scope.write(":SELECT:MATH2 ON")

#    scope.write("%s %s"%(":C1:VERT:SCA",0.125))
    scope.write("%s %s"%(":MATH1:VERT:SCA",0.125))
    scope.write("%s \"%s\""%(":MATH1:DEFINE",mathfunc1))
    scope.write("%s \"%s%s\""%(":MATHARBFLT:FILEPATH",filterpath,filterfile))
    scope.write("%s %s"%(":MATH2:VERT:SCA",0.125))
    scope.write("%s \"%s\""%(":MATH2:DEFINE",mathfunc2))

    scope.write(":ACQ:STOPA SEQ")   # Acquire one trace and stop.
    scope.write(":ACQ:STATE RUN")
    time.sleep(8)

    scope.write("%s %s,\"%s\""%(":SAVE:WAVEFORM","MATH2",filenamewfm1))  # save the
Math2 waveform for Ron Edsall's EyeDiagram program processing
    scope.write(":SELECT:MATH2 OFF")   # Dump MATH2 waveform
    scope.write(":SELECT:MATH1 ON")    # Select MATH1 waveform
```

```
    scope.write("%s %s"%(":MEASU:IMM:SOU1", "MATH1"))  # measure values in the MATH1
(De-embedded) waveform.

    scope.write("%s %s"%(":MEASU:IMM:TYP", "RIS"))
    time.sleep(3)
    rise = scope.ask(":MEASU:IMM:VAL?")
    print "rise time=",rise

    scope.write("%s %s"%(":MEASU:IMM:TYP", "FREQ"))
    freq = scope.ask(":MEASU:IMM:VAL?")
    print "freq=",freq

    scope.write("%s %s"%(":MEASU:IMM:TYP", "MINI"))
    mini = scope.ask(":MEASU:IMM:VAL?")
    print "mini=",mini

    scope.write("%s %s"%(":MEASU:IMM:TYP", "MAX"))
    maxi = scope.ask(":MEASU:IMM:VAL?")
    print "maxi=",maxi

    pk2pk = float(maxi) - float(mini)
    print "pk2pk=",pk2pk

    scope.write("%s %s"%(":MEASU:IMM:TYP", "LOW"))
    low = scope.ask(":MEASU:IMM:VAL?")
    print "low=",low

    scope.write("%s %s"%(":MEASU:IMM:TYP", "HIGH"))
    high = scope.ask(":MEASU:IMM:VAL?")
    print "high=",high

    amp = float(high) - float(low)
    print "amp=",amp

    scope.write("%s %s"%(":MEASU:IMM:TYP", "MEAN"))
    mean = scope.ask(":MEASU:IMM:VAL?")
    print "mean=",mean

    scope.write("%s %s"%(":MEASU:IMM:TYP", "RMS"))
    rms = scope.ask(":MEASU:IMM:VAL?")
    print "rms=",rms

    Jit=DPOJET_JITTER_setup("dummy.wfm","MATH1",patlen)
    print "Jit =", Jit


f.write(formatstring%(testname,devname,timestamp,maskrev,skew,volts,ID,SN,IDV,lane+32
,pattern,1,rise,freq,mini,maxi,pk2pk,low,high,amp,mean,rms,Jit[0],Jit[1],Jit[2],Jit[3
],Jit[4],Jit[5],Jit[6],Jit[7],Jit[8],Jit[9]))

f.close() # close output file
RF_Switch_Channel_Select(0)

print "program end.\n"
sys.exit(0)
```