Links to the following programs:

```
#!/usr/bin/env python
__author__='acduroy'

#**********************************************************************
# Program Name          :  DDNFIO                          (MAIN-MENU)
# Description           : This script will fetch parameters from the end-user, wherein the
#                       : provided information will be used to auto-execute Flexible Input Output
#                       : (FIO) using the test.fio jobspec
# Usage                 :  ./DDNFIO
#**********************************************************************
import os
import sys
import subprocess
import re
import fileinput
import shutil
from string import split

#**************** Global Variables Declaration and Initialization *****************
global ZERO
global ONE
global VALID
global INVALID
global EMPTY
global DEVICES
global CAPACITY
global BLOCKSIZE
global IODEPTH
global RUNTIME

ZERO = 0
ONE = 1
EMPTY = ""
VALID = True
INVALID = False
DEVICES = ('sda', 'sdb')
CAPACITY = 20
```

```
BLOCKSIZE = ('16M', '32M')
IODEPTH = 4
RUNTIME = 86400

#gdevice = DEVICES
#gcapacity = CAPACITY
#gblocksize = BLOCKSIZE
#giodepth = IODEPTH
#gruntime = RUNTIME
#gparameters = (DEVICES,CAPACITY,BLOCKSIZE,IODEPTH,RUNTIME)

#**********************Start of Functions Declarations******************************

def pyfio_create_drivelist():
    #********** Look for all drives attached to the controller *****
    os.system('lsscsi | grep disk  >> mydrivelist.txt')
    return

def pyfio_delete_drivelist():
    #********** Delete the drive list ************
    os.system('rm -f mydrivelist.txt')
    return

def goto(linenum):
    global line
    line = linenum
    return

def pyfio_clear():
    subprocess.call ("clear",shell=True)
    return

def pyfio_create_statparam():
    #***** Create a file to be pass to dstat *******
    print("creating a new file")
    try:
        file=open("myparam.dat",'a')
        file.close()
    except:
        print("error occurred")
        sys.exit(0)
    return

def pyfio_delete_statparam():
    os.system('rm -f myparam.dat')
    return
```

```
def pyfio_send_param(pass_data):
    try:
        with open('myparam.dat','w') as out_file:
            out_file.write(pass_data)
    except:
        print ("error occurred in sending data to myparam.dat file")
    return

#*********** function to search for devices in the file ************
def pyfio_search_devices():

            #*** assign the word to look at on string variable 'search'***
    search2 = "/dev/"
    search1 = "disk"

            #*** declare an empty list to hold for all the devices found***
    devices = []
    with open('mydrivelist.txt', 'r') as f:
        for line in f:
            if search1 in line:
                for part in line.split():
                    if search2 in part:
                        part_slice = part[5:8]
                        devices.append(part_slice)

                         #***** return a list of the disk/s found *********
        f.close()
        return devices

def pyfio_update_file(jobfile):

            #print("*** %s workload will be used !!! ***"%jobfile)
    try:
        for line in fileinput.input(["test.dat"], inplace=True):
            line = line.replace(";filename=/dev/${d2}", "filename=/dev/${d2}")
            sys.stdout.write(line)
    except:
        print("error encountered, can't open the file")

def pyfio_get_IOtest_info():
    #******* local variable declaration and initialization *******
    n = ONE
    c = ZERO
    response = True
    device_selected = []
    selectdev = []
    devname = []
    #*************************************************************

    selectdev = pyfio_select_devices()
    home = selectdev[1]
    device_selected = selectdev[0]

            #*********** go to main menu ? **********
    if home == True:
        c = 20
        bs1 = '16M'
        bs2 = '32M'
        q = 4
        t = 86400
        devname = str(pyfio_search_devices())
        devnumber = len(devname)-1
        param_list = [devnumber,devname,c,bs1,bs2,q,t]
        return param_list

    # ***** I/O size / capacity ********
    c = pyfio_select_capacity()

    # ********** I/O depth *************
    print (" ")
    q = pyfio_select_iodepth()

    # ********** Block size ************
    print (" ")
    #bs1 = str(raw_input("Enter first range of block sizes [1-2M], or e to escape: "))
    bs1 = pyfio_select_blocksize()
    bs2 = str(raw_input("Enter second range of block sixes [1-2M], or e to escape: "))

    # ********* I/O engine type ********
    #print (" ")
    #ioengine = int(raw_input("Enter I/O engine [1-memory map, 2-read/write, 3-splice, 4-async io, 5-
syslet, 6-scsi generic]: "))
    # ************ I/O type ************
    #print (" ")
    #iotype = int(raw_input("Enter I/O type [1-buffered io, 2-direct/raw io]: "))
    # ************ Test Time ***********
    print (" ")
    t = pyfio_select_testlength()
    # ************ Consolidate all data entered ***********
    otherParam = [c,bs1,bs2,q,t]
    param_list = device_selected + otherParam
    param_list.insert(0,len(device_selected))
    return param_list
```

```
def pyfio_select_devices():
    #Variable name definition
    #'allDisk' - number of device choices
    #'numDiskToTest' - copy of allDisk
    #'pd' - physical device available
    #'pdt' - content copies of pd
    #'diskToTest[]'list of devices to test
    global gdevices
    global gparameters
    global ghome
    n = 0
    c = 0
    m = 0
    diskToTest = []
    olddev = []
    newdev = []
    #get the list of physical disk/s selected and display
    gparameters = pyfio_get_param()
    #gdevices = gparameters[0]
    ghome = gparameters[5]
    #******** get all physical disk 'pd' to run fio test *********
    pd = pyfio_search_devices()
    if not ghome == True:
        gdevices = gparameters[0]
        olddev = gdevices
        for i in range(0,len(gdevices)):
            dev_index = pd.index(gdevices[i])
            del pd[dev_index]
    allDisk = len(pd)
    os.system('clear')
    # ******** display list of physical drives found *********
    print (" ")
    print ("%s Physical Disk/s Available:" %allDisk )
    for x in pd:
        print "#%d->"%(n+1),x
        n = n + 1
    numDiskToTest = allDisk
    pdt = pd[:]
    invalid = True
    while invalid:
        instr = "\nSelect the disk/s to test (1-" + str(numDiskToTest) + "]\n[0] to select all, [e] to escape:  "
        try:
            WhatDiskToTest = raw_input(instr)
            if (WhatDiskToTest.upper()=='E'):
                invalid = False
            elif (0<=int(WhatDiskToTest)<=numDiskToTest):
                invalid = False
            else:
                invalid = True
                raise ValueError()
        except ValueError:
                print ("invalid option, you needed to type 0,1,... %d" %numDiskToTest)
                invalid = True
        else:
            ghome = False
            pyfio_update_param(7,ghome)
            print("You just selected %s "%WhatDiskToTest)
            #******** Valid data entered ************
            if (WhatDiskToTest.upper()=='E'):
                print "Now exiting the drive selection"
                invalid = True
                if ghome == False:
                    newdev = olddev + newdev

                break
            elif int(WhatDiskToTest) == 0:
                diskToTest = pyfio_search_devices()
                print "All disk/s were selected to test by FIO, exiting now..."
                invalid = True
                break
            else:
                #******** Valid integers only for Device choices **********
                diskToTest.append(pd[int(WhatDiskToTest)-1])
                pyfio_update_param(1,diskToTest)
                param = pyfio_get_param()
                newdev = param[0]
                del pdt[int(WhatDiskToTest)-1]
                numDiskToTest = numDiskToTest - 1
                #****** check if only one disk left  ********
                if numDiskToTest == 0:
                    n = ZERO
                    c = ZERO
                    print "**** Exiting now, no more disk/s to select *****"
                    for n in range(0,len(diskToTest)):
                        print "d%d="%c, diskToTest[n]
                        c = c + 1
                    newdev = newdev + olddev
                    #invalid = False
                    break
                #****** if more than one disk left, continue to pick *******
                else:
                    print "Choose the remaining disk(s) below to run FIO:"
                    i = ZERO
                    for x in pdt:
                    print "#%d->"%(i+1),pdt[i]
                    i = i + 1
                #diskToTest = pdt
                invalid = True
                #print "\nYou selected the following disk/s %s"%pdt

    #********************* end of while loop ****************************
    if (len(diskToTest) == 0):
        #if no disk selected use old devices selected
        print "No device selected in the list !!!"
    else:
        diskToTest = newdev
        print ("selected disk to test at the return of function %s"%diskToTest)
    return diskToTest

def pyfio_deselect_devices():
     #define global variables to use
     global gdevices
     global gparameters
     global ghome
     count = 0
     devremove = []
     #get the list of physical disk/s selected and display
     gparameters = pyfio_get_param()
     gdevices = gparameters[0]
     ghome = gparameters[5]
     pyfio_clear()
     print "*** Current list of drive/s that will run FIO test ***"
     #print "* De-select a drive in the list that will not to run FIO *"
     for i in range(0,len(gdevices)):
         count = count + 1
         print "#%d-> "%count,gdevices[i]
     count = 0
     #prompt user to select disk/s to remove from the list
     undone = True
     while undone:
         strmsg = "\nDe-select a drive, choose a number[1-%d], or [e] to escape: "%len(gdevices)
         dsel = raw_input(strmsg)
         try:
             if dsel.upper() == 'E':
                 undone = False
             elif (1<=int(dsel)<=len(gdevices)):
                 undone = False
             else:
                 raise ValueError()
         except ValueError:
             print "invalid input, pls. re-try again..."
```

```
            undone = True
        else:
            #change status of ghome flag
            pyfio_update_param(7,False)
            if dsel.upper() == 'E':
                print "Now exiting the selection"
                break
            else:
                #devremove = gdevices[dsel-1]
                del gdevices[int(dsel)-1]
                if len(gdevices) == 0:
                    print "No more on the list, exiting deselection now"
                    break
                invalid = True
                while(invalid):
                    dselmore =  raw_input("De-select more disk/s? [Y/N]: ")
                    if (dselmore.upper() == 'Y') or (dselmore.upper() == 'N'):
                        invalid = False
                        if dselmore.upper() == 'Y':
                            count = 0
                            for i in range(0,len(gdevices)):
                                count = count + 1
                                print "#%d-> "%count, gdevices[i]
                            undone = True
                        else:
                            print "The following disk/s deselected to run FIO %s "%gdevices
                            undone = False
                            break
                    else:
                        print "invalid entry, pls re-try again"
                        invalid = True
                        undone = True
    return gdevices

def pyfio_select_capacity():
    done = False
    while not done:
        try:
            print (" ")
            c = raw_input("Enter IO size/capacity in percentage [1-100'%'], or [e] to escape: ")
            if c.upper()=='E':
                c = 'exit'
            elif (0 <= int(c) <= 100):
                done = True
            else:
                done = False
                raise ValueError()
        except ValueError:
            print "invalid option, you needed to type a 1, 2,3 ...,100"
            done = False
        else:
            if c == 'exit':
                print "no capacity was entered, now exiting ..."
                c = 'NONE'
                return c
                break
            print ("Your choice is %d percent of the LUN capacity" %int(c))
            return c
            break

def pyfio_select_iodepth():
    done = False
    while not done:
        try:
            print (" ")
            q = raw_input("Enter IO depth [1-256], or [e] to escape: ")
            if q.upper()=='E':
                q = 'exit'
                done = True
            elif (0 <= int(q) <= 1000000):
                done = True
            else:
                done = False
                raise ValueError()

        except ValueError:
            print "invalid option, you needed to type a 1, 2,3 ...,1,000,000"
            done = False
        else:
            if q == 'exit':
                print "no IO depth was entered, now exiting ..."
                q = 'NONE'
                return q
                break
            print "Your choice is", q
            return q
            break


def pyfio_select_blocksize():
    done = False
    while not done:
        try:
            #print (" ")
            bs1 = raw_input("Enter a number [1,2,3...] and then \nfollowed with a valid letter [k,m,g]; \nor
press [e] to exit: ")
            if not bs1.isdigit():
                if not bs1:
                    bs1 = 'empty'
                if  (bs1[len(bs1)-1] == 'M') or (bs1[len(bs1)-1] == 'm'):
                    print ("Got a valid value ")
                elif (bs1[len(bs1)-1] == 'K') or (bs1[len(bs1)-1] == 'k'):
                    print ("Got a valid value")
                elif (bs1[len(bs1)-1] == 'G') or (bs1[len(bs1)-1] == 'g'):
                    print ("Got a valid value")
                elif (bs1[len(bs1)-1] == 'E') or (bs1[len(bs1)-1] == 'e'):
                    bs1 = 'exit'
                #elif (bs1(len(bs1)-1) == EMPTY):
                #    print ("Default value to be used")
                else:
                    print ("Pls use valid suffix letter only 'k' or 'm' or 'g'; retry again")
                    done = False
                    raise ValueError()
        except ValueError:
            print "invalid option, you needed to type a 1, 2,3 ...,1,000,000"
            done = False
        else:
            if bs1 == 'exit':
                print "no blocksize was entered, now exiting ..."
                bs1='NONE'
                return bs1
                break
            if bs1 == 'empty':
                bs1 = '16M'
            print "Your choice is", bs1
            done = True
            break
    return bs1

def pyfio_select_testtype():
    return

def pyfio_select_testdata():
    return

#********** function to select test length **************
def pyfio_select_testlength():
    timevalue=0
    msg = str(raw_input("Enter Test Length: [1-9] and [s,m,h,d]: "))
    #*** seperate numeral and char in time value ***
    r=re.compile("([0-9]+)([a-zA-Z]+)")
    m=r.match(msg)
    num=int(m.group(1))
    #let=m.group(2)
    #******* Compute for the time value in seconds *********
    if (m.group(2)=="s"):
        timevalue = num
    if (m.group(2)=="m"):
        timevalue = num * 60
        print(timevalue)
    if (m.group(2)=="h"):
        timevalue = num * 3600
    if (m.group(2)=="d"):
        timevalue = num * 216000
    return timevalue

def pyfio_display_usage():
    msg='lsscsi'
    print msg
    return

def debug_function():
    pyfio_delete_drivelist()
    pyfio_create_drivelist()
    pyfio_search_devices()
    return

#*********** added functions as of 5-12-2016 ************
def disable_alldisk(jfile):
    try:
        first = "filename=/dev/"
        second = ";filename=/dev/"
        for line in fileinput.input([jfile], inplace=True):
            if "filename" in line:
                if not ";filename" in line:
                    line=line.replace(first,second)
            sys.stdout.write(line)
    except:
        print("error at disable all disk function")

def update_numdisk(jobfile, diskNum):
    try:
        string1 = ";filename=/dev/${" + diskNum + "}"
        string2 = "filename=/dev/${" + diskNum + "}"
        for line in fileinput.input([jobfile], inplace=True):
            line = line.replace(string1,string2)
            sys.stdout.write(line)
    except:
        print("error encountered, can't open the file")

def find_txt(tofile, stext):
    try:
        found = False
        if stext in open(tofile).read():
            found = True
        return found
    except:
        print("error in search text in file encountered, can't open a file")

def insert_newline(fname_in,intext,stext):

    count = 0
    txt = fname_in
    tmptxt = txt + '.txt.tmp'
    foo1 = stext
    intext = intext + "\n"
    with open(tmptxt, 'w') as outfile:
        with open(txt, 'r') as infile:
            flag = 0
            for line in infile:
                if not foo1 in line and flag == 0:
                    outfile.write(line)
                    continue
                if (foo1 in line) and (flag == 0):
                    flag = 1
                    outfile.write(line)
                    continue
                if foo1 in line and flag == 1:
                    outfile.write(line)
                    continue
                if not foo1 in line and flag == 1:
                    outfile.write(intext)
                    outfile.write(line)
                    flag = 2
                    continue
                if not foo1 in line and flag == 2:
                    outfile.write(line)
                    continue

    shutil.move(tmptxt, txt)


def update_file_param(fname, param, value):
    try:
        if param == "filename":
            i=0
            for num in range(0,len(value)):
                valname="d" + str(num+1)
                f=fname
                valfound = find_txt(f, valname)
                if valfound == True:
                    update_numdisk(fname,valname)
                    i=i+1
                else:
                    s_text="filename=/dev/${d" + str(num+1) + "}"
                    s_pos=param
                    insert_newline(fname, s_text, s_pos)
    except:
        print("error encountered at insert file, can't open a file")

#******* End of functions added on 5-12-2016 ****************

def pyfio_default_display():
    return


def pyfio_getinfo_display(flag,parval):
    n = ZERO
    idx = ZERO
    dt = EMPTY
    devices = []
    #********** clear display ************
    pyfio_clear()
    #********** use default values **************
    if flag == 1:
        msg = "Default Parameters"
    else:
        msg = "Entered Parameters"
    devices = parval[0]
    if devices == []:
        devices = pyfio_search_devices()
        parval = parval.insert(0,devices)
    selectdrives = list(devices)
    qtydrives = len(selectdrives)
    #parval_bsr = list(parval[idx+1])
    for x in range(0,qtydrives):
       #devices.append(parval[x+1])
       n = n + 1
       dt = dt + "d" + str(x+1) + "=" + str(devices[n-1]) + " "
    c = 'c=' + str(parval[1]) + '%'
    bsr = parval[3]
    bs = 'bs=' + str(bsr[0]) + '-' + str(bsr[1])
    q = 'q=' + str(parval[2])
    T = 't=' + str(parval[4])
    test_data = dt + ' ' + c + ' ' + bs + ' ' + q + ' ' + T

    #******** printout all parameters ************
    print (" ")
    print (" ")
    print ("***************************************************")
    print ("       Welcome to FIO for FA Test Usage \n")
    print ("***************************************************")
    print ("**** %s ******"%msg)
    print ("Number of HDD(s) to test: %s"%qtydrives)
    print ("Disk/s to test:  %s "%dt)
    print ("Block Size: %s "%bs)
    print ("Depth: %s" %q)
    print ("Capacity: %s" %c)
    print ("Test time(sec): %s" %T)
    print ("********************************")
    return (test_data)

def default_param():
    CAPACITY = 20
    IODEPTH = 4
    BLOCKSIZE = ('16M', '32M')
    RUNTIME = 86400
    DEVICES = pyfio_search_devices()
    param_values = (DEVICES,CAPACITY,IODEPTH,BLOCKSIZE,RUNTIME)
    sdev_new = DEVICES
    cap_new = CAPACITY
    iod_new = IODEPTH
    bs_new = BLOCKSIZE
    tlen_new = RUNTIME
    home_new = True
    param_values = (sdev_new,cap_new,iod_new,bs_new,tlen_new, home_new)
    return param_values

def pyfio_get_param():
    global gdevices
    global gcapacity
    global giodepth
    global gblocksize
    global gruntime
    global ghome
    global gparameters
    gparameters = (gdevices, gcapacity,giodepth, gblocksize, gruntime, ghome)
    return gparameters
```

```
def pyfio_update_param(num,parval):
    global gdevices
    global gcapacity
    global giodepth
    global gblocksize
    global gruntime
    global ghome
    global gparameters

    if num == 1:
        gdevices = parval
    if num == 2:
        gdevices = parval
    if num == 3:
       gcapacity = parval

    if num == 4:
        giodepth = parval

    if num == 5:
        gblocksize = parval

    if num == 6:
        gruntime = parval
    if num == 7:
        ghome = parval

    if num == 0:
        gdevices = parval[0]
        gcapacity = parval[1]
        giodepth = parval[2]
        gblocksize = parval[3]
        gruntime = parval[4]
        ghome = parval[5]
    gparameters = (gdevices, gcapacity, giodepth, gblocksize, gruntime, ghome)
    return

def main():
    #********* local init variables **************
    hf = ONE
    done = False
    invalid = True
    flagmsg = "Default"
    bs = []
    #********** Search for mydrivelist ************
    drivelist_path = "/tools/fio-1.58/mydrivelist.txt"
    #drivelist_path = "/mnt/sdb/Data Integrity Test/fio/fio-1.58/mydrivelist.txt"
```

```
if (os.path.isfile(drivelist_path)):
    print ("file exist")
    pyfio_delete_drivelist()
     #********* create drive list text **************
else:
    print("file not found")
pyfio_create_drivelist()
#********* default parameter values **********
param_values = default_param()
pyfio_update_param(0,param_values)
#********* create myparam.dat for dstat *********
#pyfio_create_statparam()
#********* Clear  screen  **********
#subprocess.call ("clear",shell=True)
pyfio_clear()
#********** display defaulted parameters ***************
#default_param = (sdev,cap,iod,bs1,bs2,tlen)
param_display = pyfio_getinfo_display(hf,param_values)
while not done:
    try:
        print (" ")
        ans = raw_input("FIO will use the above %s values? [Y|N] or 'e' to exit: "%flagmsg)
        if (ans.upper() == 'Y') or (ans.upper() == 'N') or  (ans.upper() == 'E'):
            done = True
        else:
           done = False
           raise ValueError()

    except ValueError:
        print "invalid value, pls retry again ..."
        done = False
    else:
        if ans.upper() == 'E':
            done == True
            print "FIO test was aborted..."
            pyfio_delete_drivelist()
            exit(0)
            continue
        #****** if answer = Yes, use default values and run fio *****
        if ans.upper() == 'Y':
            done = True
            break
        #****** if answer = No, change paramaters *******
        elif ans.upper() == 'N':
            done = False
            flag = 0
            invalid = True
        #******* initialize global parameter values ********
        #initvalues = default_param()
        #pyfio_update_param(0,initvalues)

        while invalid:
            sdev = []
            try:
                print "\n*** Selection of parameter to update ***"
                print "\n[1] select device to test\n[2] de-select device to test \n[3] capacity\n[4] IO
depth\n[5] block size\n[6] test time"
                entry = raw_input("\nEnter a number (1-6) of parameter: or [e] to escape: ")
                if (entry.upper()=='E'):
                    entry = 'exit'
                    invalid = False
                elif (1<=int(entry)<=6):
                    invalid = False
                else:
                    raise ValueError()
            except ValueError:
                print "invalid value, pls retry again..."
                invalid = True
            else:
                if entry == 'exit':
                    pyfio_clear()
                    hf = ONE
                    flagmsg = "New"
                    #****** clear the contents of sdev ******
                    #sdev = []
                    #***************************************
                    param_values = default_param()
                    #print param_values
                    #ans = raw_input("breakpoint #1")
                    break
                elif int(entry) == 1:
                    sdev = pyfio_select_devices()
                    if sdev == []:
                        pyfio_update_param(0,default_param())
                        param_values = pyfio_get_param()
                    else:
                        pyfio_update_param(1,sdev)
                        param_values = pyfio_get_param()
                    invalid = True
                    hf = ZERO
                    flagmsg = "New"
                    break
                elif int(entry) == 2:
                    ddev = pyfio_deselect_devices()
        if ddev == []:
            pyfio_update_param(0,default_param())
            param_values = pyfio_get_param()
        else:
            pyfio_update_param(2,ddev)
            param_values = pyfio_get_param()
        invalid = True
        hf = ZERO
        flagmsg = "New"
        break
    elif int(entry) == 3:
        #sdev = []
        cap = pyfio_select_capacity()
        if cap == 'NONE':
            pyfio_update_param(0,default_param())
            param_values = pyfio_get_param()
        else:
            pyfio_update_param(3,cap)
            param_values = pyfio_get_param()
        invalid = True
        hf = ZERO
        flagmsg = "New"
        break
    elif int(entry) == 4:
        iod = pyfio_select_iodepth()
        if iod == 'NONE':
            pyfio_update_param(0, default_param())
            param_values = pyfio_get_param()
        else:
            pyfio_update_param(4,iod)
            param_values = pyfio_get_param()
        invalid = True
        hf = ZERO
        flagmsg = "New"
        break
    elif int(entry) == 5:
        print ("\nEnter 1st block size value")
        bs1 = pyfio_select_blocksize()
        if bs1 == "0" or bs1==" ":
            bs1='16M'
        print("\nEnter 2nd block size value")
        bs2 = pyfio_select_blocksize()
        if bs2 == "0" or bs2=="":
            bs2='32M'
        bs = (bs1, bs2)
        if bs == " ":
            pyfio_update_param(0, default_param())
```

```
                        param_values = pyfio_get_param()
                    else:
                        pyfio_update_param(5,bs)
                        param_values = pyfio_get_param()
                    invalid = True
                    hf = ZERO
                    flagmsg = "New"
                    break
                elif int(entry) == 6:
                    tlen = pyfio_select_testlength()
                    if tlen == 0:
                        pyfio_update_param(0, default_param())
                        param_values = pyfio_get_param()
                    else:
                        pyfio_update_param(6,tlen)
                        param_values = pyfio_get_param()
                    invalid_entry = True
                    hf = ZERO
                    flagmsg = "New"
                    break
                else:
                    invalid = True
                hf = ZERO
                #newparam = []
                #del param_values[:]
                #del new_param[:]

            #param_values = (sdev_new,cap_new_new,iod_new, bs_new,tlen_new)
            param_display = pyfio_getinfo_display(hf,param_values)
            done = False
            invalid = True
        else:
            done = True

    param_values = " "
done = False
#******* get the final user entered parameters ********
parvalues = pyfio_get_param()
#******** send param to myparam.dat ********
#pyfio_send_param(param_values[0])
#******** update parameters *********
jobfile="test.fio"
disable_alldisk(jobfile)
#******** update param ********
update_file_param(jobfile,'filename',parvalues[0])
#******** choose workload file to execute *******
print("\n*** '%s' workload will be used!!! *** "%jobfile)
```

```
    #********* use test.fio as jobspec ************
    exec_fio_job = " ./fio " + jobfile
    command = param_display + '' + exec_fio_job
    #*********** execute fio program  *************
    subprocess.call(command,shell=True)
    #********** delete myparam.dat *********
    #pyfio_delete_statparam()
    pyfio_delete_drivelist()
    print(command)
    print("success !!!")
    return
main()
#debug_function()
```

(MAIN-MENU)

(MAIN-MENU)

```
/*********************************************************
/* Program Name         : myscsi.c
/* Author               : acduroy
/* Description          : a low-level program to send SCSI command to a SCSI device
/*                      : using the SG_IO header interface
/* Usage                : ./myscsi
/*********************************************************

#include <stdio.h>
#include <sys/ioctl.h>    // Need ioctl()
#include <scsi/sg.h>      // Need sg_io_hdr_t interface
#include <string.h>
#include <iostream>
#include <stdlib.h>       // Need to system()
#include <unistd.h>       // Need getopt()
#include <ctype.h>        // Need isprintf()


/* CONSTANTS declaration */
#define OPTIONS "r:i:s:"// option letters
                                // option preceded by ':', arg is required
                                // option preceded by "::" , arg is optional
                                // option without either ':' or '::', arg not required

bool roption, ioption, soption;   // Program flags
char *rarg = NULL;        // roption argument
char *iarg = NULL;        // ioption argument
char *sarg = NULL;        // soption argument

/* global struct to store return data from a scsi cmd */
typedef struct SCSI_data {
        unsigned char data[1024];
        unsigned char raw_sens[252];
        unsigned char sense_key;
        unsigned char additional_sense_code;
        unsigned char additional_sense_qualifier;
        unsigned char additional_sense_length;
        unsigned char sense_data_descriptors[10][244];
        int result;
} SCSI_data;

/* global struct to store scsi cmd */
typedef struct SCSI_cmd {
        int sg_fd;
        unsigned char cmdblk[32];
        int cmdblklength;
        int allocation_length;
        int xfer;
        int timeout;
} SCSI_cmd;

SCSI_data send_scsicmd(SCSI_cmd cmdobject)
{
        int k;
        unsigned char inqBuff[cmdobject.allocation_length];
        unsigned char sense_buffer[252];
        SCSI_data output_data;
        sg_io_hdr_t  io_hdr;

        /* Prepare INQUIRY command */
        memset(&io_hdr,0,sizeof(sg_io_hdr_t));
        io_hdr.interface_id = 'S';
        io_hdr.cmd_len = cmdobject.cmdblklength;;
        io_hdr.mx_sb_len = sizeof(sense_buffer);
        io_hdr.dxfer_direction = cmdobject.xfer;
        io_hdr.dxfer_len = cmdobject.allocation_length;
        io_hdr.dxferp = inqBuff;
        io_hdr.cmdp = cmdobject.cmdblk;
        io_hdr.sbp = sense_buffer;
        io_hdr.timeout = cmdobject.timeout;

        if (ioctl(cmdobject.sg_fd, SG_IO, &io_hdr) < 0)
        {
                output_data.result=1;
                if (io_hdr.sb_len_wr > 0)
                {
                        printf("INQUIRY sense data:");
                        for (k=0; k<io_hdr.sb_len_wr; ++k)
                        {
                                if ((k>0) && (0==(k%10)))
                                        printf("\n");
                                printf("0x%02x", sense_buffer[k]);
                        }
                        printf("\n");
                }

                if (io_hdr.masked_status)
                        printf("INQUIRY SCSI status=0x%x\n", io_hdr.status);

                if (io_hdr.host_status)
                        printf("INQUIRY host_status=0x%x\n", io_hdr.host_status);

                if (io_hdr.driver_status)
```

```
                        printf("INQUIRY driver_status=0x%x\n", io_hdr.driver_status);

            }
            else { /* assume INQUIRY response is present */
                        output_data.result=0;
                        for (k=0; k<cmdobject.allocation_length; k++){
                                    output_data.data[k]=inqBuff[k];
                        }
            }
            return output_data;
}

/* Function read capacity */
int read_capacity(char *drivename)
{
            FILE *driveptr = fopen(drivename, "r");


        SCSI_data scsi_data_read_capacity;
        SCSI_cmd scsi_read_capacity;

        scsi_read_capacity.sg_fd=fileno(driveptr);
        scsi_read_capacity.cmdblk[0] = 0x9e;
        scsi_read_capacity.cmdblk[1] = 0x10;
        scsi_read_capacity.cmdblk[13] = 32;
        scsi_read_capacity.cmdblklength = 16;
        scsi_read_capacity.xfer = SG_DXFER_FROM_DEV;
        scsi_read_capacity.allocation_length = 32;
        scsi_read_capacity.timeout = 1000;

        scsi_data_read_capacity = send_scsicmd(scsi_read_capacity);

        if (scsi_data_read_capacity.result==0){
            printf("  capacity in blocks: %02x%02x%02x%02x%02x%02x%02x%02x\n",
                scsi_data_read_capacity.data[0],
                scsi_data_read_capacity.data[1],
                scsi_data_read_capacity.data[2],
                scsi_data_read_capacity.data[3],
                scsi_data_read_capacity.data[4],
                scsi_data_read_capacity.data[5],
                scsi_data_read_capacity.data[6],
                scsi_data_read_capacity.data[7]);
            printf("          blocksize: %02x%02x%02x%02x\n",
                scsi_data_read_capacity.data[8],
                scsi_data_read_capacity.data[9],
                scsi_data_read_capacity.data[10],
                scsi_data_read_capacity.data[11]);
```

```
              printf("     protection type: %02x\n",
                     scsi_data_read_capacity.data[12]);
       }
       fclose(driveptr);

            return 0;
}

/* Funtion inquiry */
int inquiry(char *drivename)
{
            int n = 0;            // counter for loop
            unsigned char *ser_num = NULL;            // serial number

            /* Define scsi cmd and data buffer length */
            const int INQ_CMD_LEN =  6;
            const int INQ_DATA_LEN = 252;

            /* Open the device File */
            FILE *driveptr = fopen(drivename, "r");
            /* Create an object of SCSI data and cmd structure  */
            SCSI_data scsi_data_inquiry;
            SCSI_cmd scsi_inquiry;

            /* Prepare the sg_io_hdr structure  */
            scsi_inquiry.sg_fd = fileno(driveptr);
            /* INQ(12) 6 byte command to get s/n */
            scsi_inquiry.cmdblk[0] = 0x12;
            scsi_inquiry.cmdblk[1] = 1;
            scsi_inquiry.cmdblk[2] = 0x80; // Page code = 80h
            scsi_inquiry.cmdblk[3] = INQ_DATA_LEN;
            scsi_inquiry.cmdblk[4] = 0;
            scsi_inquiry.cmdblk[5] = 0;

            scsi_inquiry.cmdblklength = 6;
            scsi_inquiry.xfer = SG_DXFER_FROM_DEV;
            scsi_inquiry.allocation_length = INQ_DATA_LEN;
            scsi_inquiry.timeout = 1000;

            /* Call the send_scsicmd function to pass the scsi_inquiry values */
            /* and then return the data result to scsi_data_inquiry */
            scsi_data_inquiry = send_scsicmd(scsi_inquiry);

            if (scsi_data_inquiry.result==0){

                        printf("*** Data Buffer after ioctl ****\n");
                  printf("  Serial Number: ");
```

```
                for(int i=0;i<INQ_DATA_LEN;i++)
                {
                        if((i>=10) && (i<=23))
                        {
                                ser_num = &(scsi_data_inquiry.data[i]);
                                printf("%hx",*ser_num);
                        }
                }
                printf("\n");

        }
        return 0;
}

/* Function smart */
int smart(char *drivename)
{
        printf("SMART function now running on %s ...\n", drivename);
        return 0;
}

/* Function instruc */
int instruc()
{
        printf("Options\n");
        printf("-r arg    -- option and arg\n");
        printf("-i arg    -- option and arg\n");
        printf("-s arg    -- option and arg\n");
        return 0;
}

/* Function lhdd */
int lhdd()
{
        printf("  List of drives found in the system:\n ");
        char *devicelist =  (char *)system("lsblk |grep -i disk");
        for(int i=0; i<3; i++)
        {
                printf((char *)devicelist[i]);
                printf("\n");
        }
        return  0;
}

/* Main function */
int main(int argc, char **argv)
{
        char c; // return by getopt()

        /* List of drives to choose  */
        lhdd();


        /* Get known options and any arguments */
        while((c=getopt(argc,argv,OPTIONS)) != -1)
                switch(c){
                        case 'r': // option for read capcity
                                roption = true;
                                rarg = optarg;
                                read_capacity(rarg);      // Call read_capacity function
                                break;
                        case 'i': // option for inquiry
                                ioption=true;
                                iarg=optarg;
                                inquiry(iarg);            // Call inquiry function
                                break;
                        case 's': // option for SMART information
                                soption=true;
                                sarg=optarg;
                                smart(sarg);              // Call smart function
                                break;
                        case '?':
                                instruc();                // Display instructions
                                //exit(1);                // End Program
                        default:
                                printf("Error in getopt() function");
                                //abort();        // ???
                }

        return 0;
}
```

(MAIN-MENU)

```
#!/bin/bash
__acduroy__
                                                (MAIN-MENU)
#***************************************************
# Program Name          : mylsscsi.sh
# Description           : A bash script to modify the Linux lsscsi utility.
#                       : To list  SCSI devices and their  attributes
# Usage                 :  ./mylsscsi.sh
#***************************************************

#declare global variables here
declare -ga diskname

echo
echo "   Running modified lsscsi utility !!!  "
echo

function getdevices(){
#*** declare variables ***
local -i nline      # total lines with 'disk' word content in lsscsi command
local -i ptrline    # pointer to the current line
local -i i          # number of disk/s found
local -a tmpdiskname    # array of disk devices

#*** count all disk devices in the system ***
nline=$(lsscsi |grep disk |grep 5:0 |wc -l)

#* for loop to populate array disk ***
if [ $nline -eq 1 ]
        then
                diskname[0]=$(lsscsi |grep disk |cut -c 59-61)
                i=1
        else
                nline=$nline-1
                for ((i=0; i <= $nline; i++))
                do
                        ptrline=i+1
                        device=$(lsscsi |grep disk |head -n $ptrline |tail -n 1 |cut -c 59-61)
                        tmpdiskname+=("$device")
                done
fi

#*** display all disk devices ***
#echo $i
#echo ${diskname[@]}
diskname="$(declare -p tmpdiskname)"
```

```
main
exit 1
}

function getserialnumber(){
serial=$(sg_inq /dev/$1 |grep -w "serial number:" |cut -c 21-43)
echo $serial
exit 1
}

function getcapacity(){
capacity=$(sg_readcap /dev/$1 |grep -w "Device size" |cut -c 16-60)
echo $capacity
exit 1
}


function main(){
local -i numdevices
local -i ptrnewline
local -i start


# *** program encountered fatal error  ***
if [ $? -ne 0 ]
then
        echo "$0: fatal error:" "$@" >&2
        exit 1
else
    #get number of physical disk(s) found in the system
    numdevices=$(lsscsi |grep disk |grep 5:0 |wc -l)
    echo $numdevices " Physical device(s) found !!!"

        eval "declare -a NEWLIST=${diskname#*=}"
        echo "Device Name of Physical Disk(s): " "${NEWLIST[@]}"

        if [ $numdevices -eq 1 ]
        then
                echo "sda"
                serialnumer=$(getserialnumber "sda")
                capacity=$(getcapacity "sda")
                lsscsicmd=$(lsscsi |grep disk)
                echo $lsscsicmd $serialnumber $capacity
        else
                start=1
        for disk in ${!NEWLIST[*]}  # ((i=0; i <= $start; i++))
                do
```

```
                        #echo "$disk: ${NEWLIST[$disk]}"
                        ptrnewline=i+1
                        serialnumber=$(getserialnumber ${NEWLIST[disk]})
                        capacity=$(getcapacity ${NEWLIST[disk]})
                        lsscsicmd=$(lsscsi |grep disk |head -n $start|tail -n 1)
                        echo $lsscsicmd $serialnumber $capacity
                        start=$start+1
                done
        fi
fi
exit 1
}

getdevices
```

(MAIN-MENU)

```
#!/bin/bash
__acduroy__
                                                  (MAIN-MENU)
#**************************************************************************************
# Program Name          : read_ss8462_ses_new.sh
# Description           : This bash script will fetch, display and monitor the SES enclosure status page 2
#                       : information using the Linux sg_ses utility. Any critical condition exist will
#                       : prompt the end-user a warning message and it will halt the program.
# Usage                 :  ./read_SS8462_ses_new
#****************************************

function select_command_option {
    #*** option to select command to run ***
    while [[ $# -gt 1 ]]
    do
        key="$1"
        #*** Select options -t for temp; -s for status
        case $key in
            #*** Get Enclosure Temp ***
            -t|--temp)
                # echo Display Enclosure Temperature
                SENSOR_NAME="$2"
                DEVICE_NAME="$3"
                read_enclopsure_temp
                shift  #pass argument or value
                ;;
            #*** Get Enclosure Overall Status ***
            -s|--status)
                # echo Display Enclosure Overall Status
                DEVICE_NAME="$2"
                EXPECTED_VALUE="$3"
                read_enclosure_page2
                shift  #pass argument or value
                ;;
            *)
         ;;
        esac
        shift #pass argument or value
    done
    echo element# = "${SENSOR_NAME}"
}

function select_option {
    OPTIND=1
    while getopts "ts:" opt
    do
        case "$opt" in
        t)
            SENSOR_NAME="$3"
            DEVICE_NAME="$2"
            ;;
        s)
            DEVICE_NAME="$2"
            EXPECTED_VALUE="$3"
            ;;
        esac
    done
    shift $((OPTIND-1))
}

function read_enclosure_temp {
#* Usage: ./read_ss8460_temp -g element_type -o overtemp_setpoint -u undertemp_setpoint
#* Description: To get the temp of the enclosure and to set the over/under temp trigger
#* Date: 02-16-2017
#* Rev. 1

    echo "*Read enclosure temp and over-under temp status*"
    #*** Loop forever ***
    for ((;;))
    do
        #*** Variable Initialization ***
        temp=0
        element_status=0
        smart=0
        date
        #*** Selection of temp sensor elemet type ***
        #*** Get Overall Enclosure Temp and Status ***
        #temp[0]=$(sg_ses -p 0x02 -H -s /dev/$DEVICE_NAME |head -n 28 |tail -n 1 |cut -c 52-54)
        #element_status[0]=$(sg_ses -p 0x02 -H -s /dev/$DEVICE_NAME |head -n 28 |tail -n 1 |cut -c 55-
56)
        #*** Get Sensor 2 Temp and Status ***
        #temp[2]=$(sg_ses -p 0x02 -H -s /dev/$DEVICE_NAME |head -n 29 |tail -n 1|cut -c 26-28)
        #element_status[2]=$(sg_ses -p 0x02 -H -s /dev/$DEVICE_NAME |head -n 29 |tail -n 1|cut -c 29-
31)
        #*** Get Sensor 6 Temp and Status ***
        #temp[6]=$(sg_ses -p 0x02 -H -s /dev/$DEVICE_NAME |head -n 30 |tail -n 1|cut -c 26-28)
        #element_status[6]=$(sg_ses -p 0x02 -H -s /dev/$DEVICE_NAME |head -n 30 |tail -n 1 |cut -c 29-
31)
        #*** Get Sensor 7 Temp and Status ***
        #temp[7]=$(sg_ses -p 0x02 -H -s /dev/$DEVICE_NAME |head -n 30 |tail -n 1 |cut -c 35-37)
        #element_status[7]=$(sg_ses -p 0x02 -H -s /dev/$DEVICE_NAME |head -n 30 |tail -n 1 |cut -c 38-
40)
```

```
        smart[0]=$(smartctl -a /dev/sda |head -n 6 |tail -n 1 |cut -c 18-33)
        smart[1]=$(smartctl -a /dev/sda |head -n 69 |tail -n 1 |cut -c 38-40)
        #*** check for exit status ***
        if [ $? -eq 0 ]
        then
            #*** Display Enclosure Temperature ***
            #echo Overall Enclosure Temperature = ${temp[0]}  Status = ${element_status[0]}

            echo model no=${smart[0]}
            echo hdd temp=${smart[1]}
        else
            echo command error !!!
            exit 1

        fi

    done

#***** End of Function read_enclosure_temp *****
}


function read_enclosure_page2 {
    echo "*Read enclosure page 2*"
    var1=$2
    for ((;;))
    do
        date
        #*** preserve last reading ***
        var2=$var1
        #*** get the enclosure status ***
        var1=$(sg_ses -p 0x02 -H /dev/$1 |head -n 4|tail -n 1|cut -c 11-13)
        #*** check for exit status ***
        if [ $? -eq 0 ]
        then
            #****** compare to last reading ********
            if [ $var1 != $var2 ]
            then
                echo *****status changed******
                echo  current status="$var1", last="$var2"
                break
            fi
        else
            break
        fi
    done
```

```
}

function check {
    echo "checking function !!!"
}


function options_main {
    echo "Choose what enclosure's page  to read !!!"

    OPTIONS=("OverallStatus"  "DisplaySMARTinfo"  "Quit")

    select opt in "${OPTIONS[@]}"
    do
         case "$opt" in
            "OverallStatus")
               select_command_option
               read_enclosure_page2
               echo "I don't know what is happening here… !!!"
               ;;

           "DisplaySMARTinfo")
               #select_command_option
               read_enclosure_temp
               ;;

           "Quit")
               echo "Thanks you for using this program !!! bye . bye..."
                break
               ;;

           *)echo invalid option;;
        esac
    done
}


function main {
    #select_command_option
    options_main
    #read_enclosure_temp
    #exit

}

#***** call main function *****
main
```
(MAIN-MENU)

(MAIN-MENU)

```
;****************************************************************************************
;          Script Name      : _9900_DiskHostCache_Offline.ttl
;          Description      : Multiple iteration of Cache Diagnostic Test
;          Last updated     : 4/9/2015
;          Author           : Alec Duroy
;          Purpose          : For automation of FA evaluation process only
;          Platform         : S2A9900
;          Usage            : With the controller powered ON, execute the macro
;****************************************************************************************

timeout=0
count_cachetest_fail=0
count_cachetest=0
int2str valstr count_cachetest
int2str valstr2 count_cachetest_fail
call getInfo
setdlgpos 0 0

;***** routine of main loop *****
:loop_main_cachetest
call displayStr
pause 2
call run_HostStageBuffer_test
call check_hostcache_result
call displayStr
pause 2
call run_HostFlood_test
call check_hostcache_result
call displayStr
pause 2
call run_HostCache_test
call check_hostcache_result
call displayStr
pause 2
call run_DiskCache_test
call check_diskcache_result
call displayStr
pause 2
call count_cachetest_loops
;sendln 'shutdown restart=120'
;wait 'continue'
;sendln 'y'
;include '_9900_1X_restarts_counter(COM-1).ttl'
goto loop_main_cachetest

;***** count_cachetest number of loops *****
```

```
:count_cachetest_loops
count_cachetest=count_cachetest+1
if count_cachetest=valInt+1 goto complete_cachetest
return

;***** routine to check host cache test *****
:check_hostcache_result
; wait for startup string
wait 'slot 12 passed' 'Timeout: Diagnostic on host slot 12 did not respond'
closesbox
; 'passed'?
if result=1 goto check_more
; 'failed'?
if result=2 goto failTest
; 'failed'?
if result=3 goto failTest

:check_more
wait 'slot 34 passed' 'Timeout: Diagnostic on host slot 34 did not respond'
; 'passed'?
if result=1 goto passTest
; 'failed'?
if result=2 goto failTest
; 'failed'?
if result=3 goto failTest

:failTest
count_cachetest_fail=count_cachetest_fail+1
count_cachetest=count_cachetest+1
int2str valstr count_cachetest
int2str valstr2 count_cachetest_fail
call stopped_cachetest

:passTest
pause 2
return

;***** routine to check disk cache test *****
:check_diskcache_result
; wait for startup string
wait 'disk slot AB passed' 'ERRR INT_AB'
closesbox
; 'passed'?
if result=1 goto check_CD
; 'failed'?
if result=2 goto failTest_diskcache
```

```
:check_CD
wait 'disk slot CD passed' 'ERRR INT_CD'
if result=1 goto check_EF
if result=2 goto failTest_diskcache

:check_EF
wait 'disk slot EF passed' 'ERRR INT_EF'
if result=1 goto check_GH
if result=2 goto failTest_diskcache

:check_GH
wait 'disk slot GH passed' 'ERRR INT_GH'
if result=1 goto check_PS
if result=2 goto failTest_diskcache

:check_PS
wait 'disk slot PS passed' 'ERRR INT_PS'
if result=1 goto passTest_diskcache
if result=2 goto failTest_diskcache

:failTest_diskcache
count_cachetest_fail=count_cachetest_fail+1
count_cachetest=count_cachetest+1
int2str valstr count_cachetest
int2str valstr2 count_cachetest_fail
call stopped_cachetest

:passTest_diskcache
pause 2
return

;***** subroutine of host cache test *****
:run_HostCache_test
pause 2
sendln "diag hostcache"
wait 'WARNING'
pause 2
sendln "y"
return

;***** subroutine of host stage buffer test *****
:run_HostStageBuffer_test
pause 2
sendln "diag hoststagebuffer"
wait 'WARNING'
pause 2
sendln "y"
```

```
return

;***** subroutine of host flood test *****
:run_HostFlood_test
pause 2
sendln "diag hostflood"
wait 'WARNING'
pause 2
sendln "y"
return

;***** subroutine of disk cache test *****
:run_DiskCache_test
pause 2
sendln "diag diskcache"
wait 'WARNING'
pause 2
sendln "y"
return

;***** getting information from user subroutine *****
:getInfo
inputbox 'Enter Unit Serial Number:' 'COM-1 Multiple Run of Cache Diag '
SerialNum=inputstr
inputbox 'Enter Number of Test' 'COM-1 Multiple Run of Cache Diag '
NumLoops=inputstr
str2int valInt NumLoops
return

;***** display subroutine *****
:displayStr
int2str valstr count_cachetest
message='Cache Diag Test Loop Number = '
strconcat message valstr
;strconcat message ' Fail count_cachetest= '
;strconcat message valstr2
TitleStr='COM-1 Cache Diagnostic of '
strconcat TitleStr SerialNum
statusbox message TitleStr
return

;***** stopped subroutine *****
:stopped_cachetest
sendln "COMMENT MACRO HAS STOPPED. HOST AND DISK CACHE DIAG TEST FAILED"
setdlgpos 0 0
statusbox message 'COM-1 MACRO STOPPED'
pause 20
```

```
end

;***** complete subroutine *****
:complete_cachetest
pause 5
sendln "COMMENT UNIT HAS SUCCESSFULLY COMPLETED " NumLoops " HOST AND DISK CACHE
DIAGNOSTIC TEST. POWER DOWN AND MOVE TO NEXT TEST."
setdlgpos 0 0
TitleStr='COM-1 COMPLETED'
strconcat TitleStr NumLoops
strconcat TitleStr 'Multiple Iteration of Cache Diag Test'
statusbox message TitleStr
pause 20
end
```

(MAIN-MENU)

(go back to main menu)

```
;*********************************************************************************
;       Script Name     : _9900_DiskHostCache_MultipleDiagTest
;       Description     : Multiple iteration of Simultaneous Host and Disk Cache Diagnostic Test
;       Last updated    : 4/9/2015
;       Author          : Alec Duroy
;       Purpose         : For automation of FA evaluation Process Only
;       Platform        : S2A9900
;       Usage           : With the controller powered ON, execute the macro
;*********************************************************************************

timeout=0
count=0
addpost=0

:loop
setdlgpos 0 0
int2str valstr count
int2str valstr2 addpost
message='Successful restarts='
strconcat message valstr
strconcat message ' Additional POSTs='
strconcat message valstr2
statusbox message 'COM-1 Booting'

; wait for startup string
wait 'FACT_DIAG: Host and Disk Cache Tests Passed' 'ERRR'

closesbox

; 'passed'?
if result=1 goto countTest

; 'failed'?
if result=2 goto stopped

:countTest
count=count+1
if count=1001 goto complete
pause 20
goto runTest

:runTest
pause 20
sendln "t"
wait 'Prompt'
```

```
pause 5
sendln "5"
goto loop

:stopped
sendln "COMMENT MACRO HAS STOPPED. HOST AND DISK CACHE DIAG TEST FAILED"
setdlgpos 0 0
statusbox message 'COM-1 MACRO STOPPED'
pause 36000
end

:complete
pause 10
sendln "COMMENT UNIT HAS SUCCESSFULLY COMPLETED 1000 HOST AND DISK CACHE DIAG TEST.
POWER DOWN AND MOVE TO NEXT TEST."
setdlgpos 0 0
statusbox message 'COM-1 COMPLETED 1000 REBOOTS'
pause 36000
end
```

(MAIN-MENU)

(go back to main menu)

```
;*******************************************************************************
;         Script Name      : 9900_Variable-restarts_counter.ttl
;         Description      : Continuous power-on self-test; Multiple Iteration of Boot-up
;                          : Diagnostic Test
;         Last updated     : 4/9/2015
;         Author           : Alec Duroy
;         Purpose          : For Automation of Failure Analysis Evaluation Process Only
;         Platform         : S2A9900
;         Usage            : With the controller powered ON, execute the macro in tera-term
;*******************************************************************************
timeout=0
count=0
addpost=0
inputbox 'Serial Number' 'COM-1'
serialNum=inputstr
inputbox 'Number of loops' 'COM-1'
numLoops=inputstr
inputbox 'Duration in millisecond per cycle (min=20 max=120)' 'COM-1'
delayCount=inputstr

:loop
setdlgpos 0 0
int2str valstr count
int2str valstr2 addpost
str2int intValue numLoops
str2int intValue2 delayCount
message='Successful restarts='
strconcat message valstr
strconcat message ' Additional POSTs='
strconcat message valstr2
title_str='COM-1 Booting of '
strconcat title_str serialNum
statusbox message title_str

; wait for startup string
wait 'ECC Error bitmap' 'Bootup Diagnostics failed' 'Warning: DDR Error' 'failed DP memory diags' 'PCIe
diagnostic error' 'Timed Out. Hardware Failure' 'fatal exception' 'not responding to No-op' '8 valid data
channels' 'Initiating additional POST'

closesbox

; 'bitmap'?
if result=1 goto done
```

```
; 'failed'?
if result=2 goto done

; 'Error'?
if result=3 goto done

; 'diags'?
if result=4 goto done

; 'error'?
if result=5 goto done

; 'Failure'?
if result=6 goto stopped

; 'exception'?
if result=7 goto done

; 'No-op'?
if result=8 goto stopped

; 'channels'?
if result=9 goto login

;system does additional post
if result=10 addpost=addpost+1
goto loop
end

:login
sendln "login factory"
pause 1
sendln "Tested Qual1ty!"
pause 1
sendln "bootup bypass"
count=count+1
if count=intValue goto complete
pause 20
goto restart

:restart
pause 20
sendln "shutdown restart="delayCount
wait 'continue'
pause 5
sendln "y"
```

```
wait 'Saving system data'
goto loop

:done
pause 20
sendln "login factory"
pause 1
sendln "Tested Qual1ty!"
pause 1
sendln "bootup bypass"
pause 5
sendln "log exception"
pause 2
sendln "host status"
pause 2
sendln "faults"
pause 2
sendln "version avr"
pause 2
sendln "COMMENT MACRO HAS STOPPED. SCROLL UP AND LOOK FOR FAUILURES. IF THERE IS NO
FAILURE POWER CYCLE UNIT AND RESTART MACRO"
setdlgpos 0 0
title_str='COM-1 MACRO STOPPED - '
strconcat title_str serialNum
strconcat title_str ' FAILED POWER CYCLE TEST'
statusbox message title_str
pause 36000
end

:stopped
pause 60
wait '8 valid data channels'
pause 2
sendln "login factory"
pause 1
sendln "Tested Qual1ty!"
pause 1
sendln "bootup bypass"
pause 5
sendln "log exception"
pause 2
sendln "host status"
pause 2
sendln "faults"
pause 2
sendln "version avr"
pause 2
```

```
sendln "COMMENT MACRO HAS STOPPED. SCROLL UP AND LOOK FOR FAUILURES. IF THERE IS NO
FAILURE POWER CYCLE UNIT AND RESTART MACRO"
setdlgpos 0 0
title_str='COM-1 MACRO STOPPED - '
strconcat title_str serialNum
strconcat title_str ' FAILED POWER CYCLE TEST'
statusbox message title_str
pause 36000
end

:complete
pause 10
sendln "COMMENT UNIT HAS SUCCESSFULLY COMPLETED " numLoops " POWER CYCLES. POWER DOWN
AND MOVE TO NEXT TEST."
setdlgpos 0 0
message='COM-1 COMPLETED '
strconcat message numLoops
strconcat message ' REBOOTS'
title_str='COM-1 Booting of '
strconcat title_str serialNum
statusbox message title_str
pause 36000
end
```

(MAIN-MENU)