A Python Use Example in the CTA Hardware Development

Akira Okumura [a, b]

[a] Institute for Space-Earth Environmental Research, Nagoya University

[b] Max-Planck-Institut für Kernphysik

oxon@mac.com

@akira-okumura / GitHub

PyGamma15

Nov 20, 2015

Disclaimer

- I am not a Python expert
- I have never liked Python
- But I have been using Python since 2007 for data analysis and hardware development

Python Examples in the Lab: PySerial and PyVISA

© Tektronix

© Keithley

- Instruments in the lab
  - Digital oscilloscopes
  - Digital multimeters
  - Power supply units, etc.
- Need an automated measurement system
  - $> 10^3$ repeated measurements
  - Python scripts can log the instrument configurations
- Communication standards
  - RS232C (PySerial) – slow
  - GPIB (PyVISA) – fast
  - Ethernet (PyVISA) - faster

Before and After (in My Case)

2005

Had to use C and system calls
Not easy for young students

```
#include <termios.h>
#include <unistd.h>

struct termios  config;
if(!isatty(fd)) { ... error handling ... }
if(tcgetattr(fd, &config) < 0) { ... error handling ... }
config.c_iflag &= ~(IGNBRK | BRKINT | ICRNL |
                    INLCR | PARMRK | INPCK | ISTRIP | IXON);
config.c_oflag = 0;
config.c_lflag &= ~(ECHO | ECHONL | ICANON | IEXTEN | ISIG);
config.c_cflag &= ~(CSIZE | PARENB);
config.c_cflag |= CS8;
config.c_cc[VMIN]  = 1;
config.c_cc[VTIME] = 0;
if(cfsetispeed(&config, B9600) < 0 || cfsetospeed(&config, B9600) <
  ... error handling ...
  }
if(tcsetattr(fd, TCSAFLUSH, &config) < 0) { ... error handling ...
```

continues…

Code example taken from
https://en.wikibooks.org/wiki/Serial_Programming/termios

2007–

PySerial

```
>>> import serial
>>> ser = serial.Serial(0)  # open first serial port
>>> print ser.name          # check which port was really used
>>> ser.write("hello")      # write a string
>>> ser.close()             # close port
```

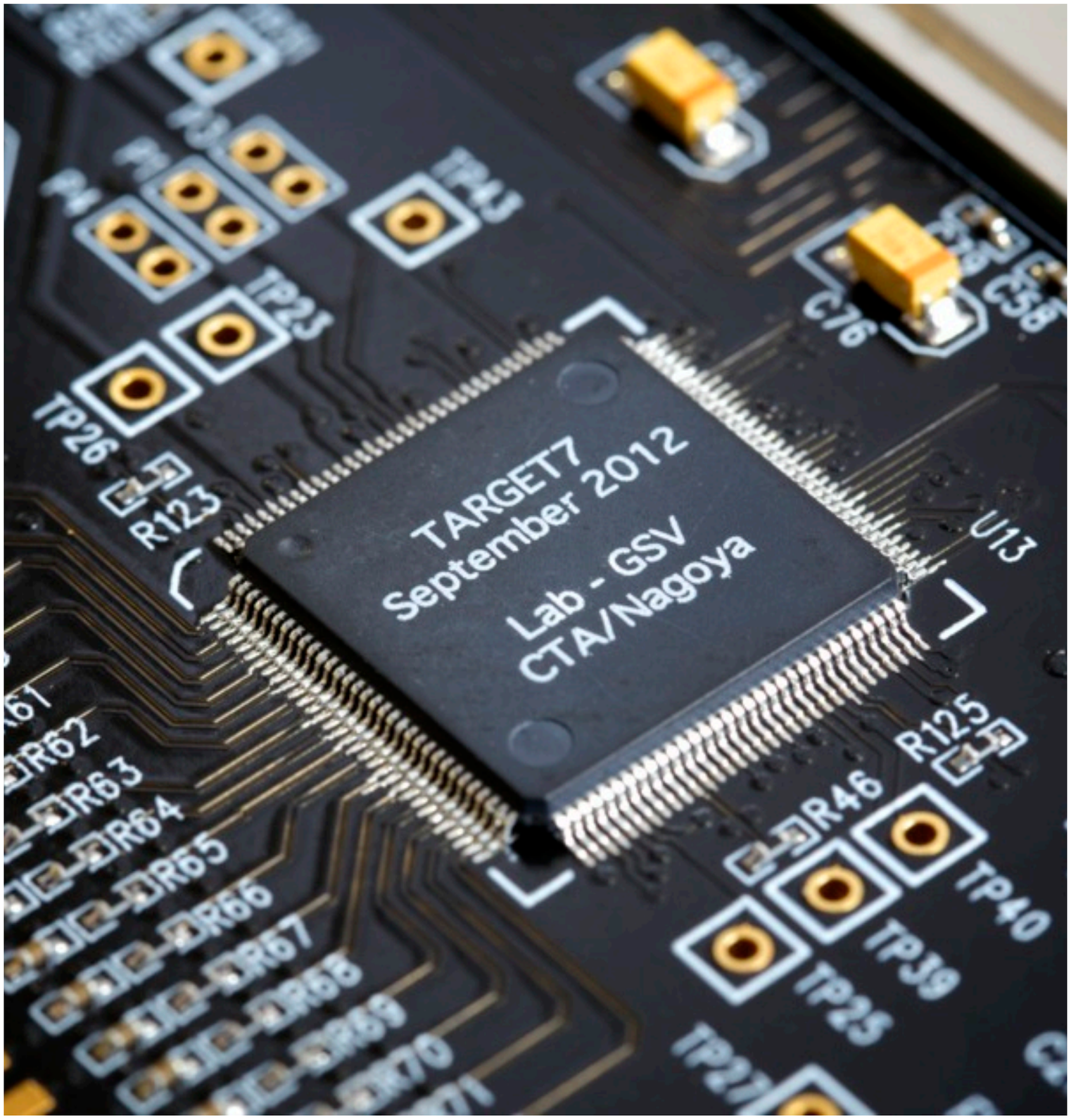
Code example taken from
https://pythonhosted.org/pyserial/shortintro.html

PyVISA

```
>>> import visa
>>> rm = visa.ResourceManager()
>>> rm.list_resources()
('ASRL1::INSTR', 'ASRL2::INSTR', 'GPIB0::12::INSTR')
>>> inst = rm.open_resource('GPIB0::12::INSTR')
>>> print(inst.query("*IDN?"))
```

Code example taken from
https://pyvisa.readthedocs.org/en/stable/

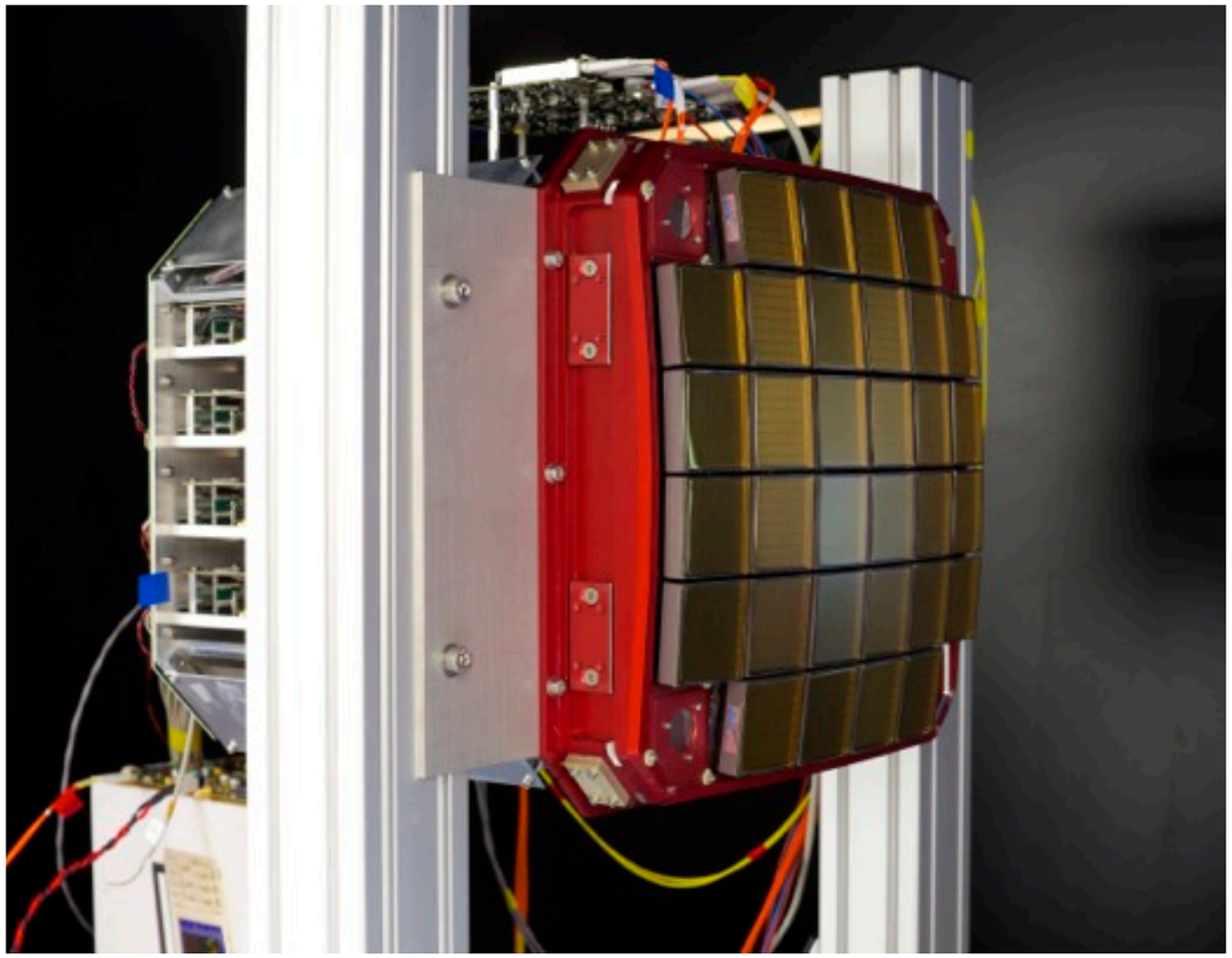
TARGET: Application Specific Integrated Circuit (ASIC) for CTA



- 1 GHz waveform sampling for fast Cherenkov flashes
- Long buffer of 16 µs for array triggering in > 1 km distance
- 16 channels, trigger
- Developed for SiPM-based CTA cameras
  - Schwarzschild-Couder (SC) Medium-Sized Telescopes (11,328 pix)
  - SC Small-Sized Telescopes (2,048 pix)

See Bechtol et al. (2012)

Prototype Camera (CHEC-M) and Camera Module



The prototype camera is being installed on a telescope in Paris!

Requirements for Our TARGET Software in the Lab

- Fast UDP (user data protocol) communication using socket and system calls
  - The core part must be written in **C or C++**
- Support three operating systems: Linux, OS X, and Windows(!)
  - Compilers are GCC, Clang, and Visual Studio
  - GNU make, autoconf, and automake cannot be used on Windows → **CMake**
  - I didn't want to write Windows specific multi-thread code
    → **C++11**, because a common library <thread> is supported
- DAQ must be as fast as we can implement (C/C++), but a scriptable language is desired for lab tests
  - Use **SWIG** to generate a Python wrapper module
  - Easy for summer students, engineers as well as young (< ~35 years old) researchers
  - Running a test program line by line makes electronics tests quite easy

libTARGET

- It ***was*** an open-source project hosted at http://sourceforge.net/projects/libtarget/ (old versions still exist there)
- For some reasons, the repository was migrated to the CTA SVN repository
  - Internal documents
  - Issue tracking (that should be hidden before publication)
  - LDAP user database
- You can get some hints from the old repository, while it is not an excellent tutorial (it's my first CMake/SWIG project)

Code Snippets (C++)

SocketReceiver.h

```
#ifndef TARGET_SOCKET_RECEIVER_H
#define TARGET_SOCKET_RECEIVER_H

#include <list>
#ifndef __CINT__
#include <thread>
#include <mutex>
#endif

#include "target/Types.h"

namespace TARGET {

class BaseInterface;

class SocketReceiver
{
private:
  std::list<uint8_t*>        fEventDataList;
  std::list<uint32_t>          fEventLengthList;
  BaseInterface*             fInterface;
  uint32_t                     fMaxEvents;
  std::list<ResponsePacket*> fResponseList;
  bool                       fStop;
#ifndef __CINT__
  std::mutex                 fMutex;
  std::thread                fThread;
#endif

  SocketReceiver(const SocketReceiver&);
  const SocketReceiver& operator=(const SocketReceiver&);

public:
  SocketReceiver(BaseInterface* baseInterface = 0);
  virtual ~SocketReceiver();

  bool             IsRunning() const {return fThread.joinable();}
  ResponsePacket*  ReadResponsePacket();
-:---  SocketReceiver.h    Top L1     SVN-9913   (C/l AC Abbrev)
Beginning of buffer
```

BaseInterface.cxx

```
  if(socket_buffer_size > 0){
    errno = 0;
 #ifdef _WIN32
    if(setsockopt(fSocket, SOL_SOCKET, SO_RCVBUF, (const char*)&socket_buffer_si
ze, sizeof(socket_buffer_size)) != 0){
 #else
    if(setsockopt(fSocket, SOL_SOCKET, SO_RCVBUF, &socket_buffer_size, sizeof(so
cket_buffer_size)) != 0){
 #endif
      char str[100];
      int err = errno;
      sprintf(str, "Cannot change the socket buffer size to %d bytes. %s.", sock
et_buffer_size, strerror(err));
      throw RuntimeError(str);
    } // if
  } // if

  memset(&fDestinationSocketAddress, 0, sizeof(fDestinationSocketAddress));
#ifdef _WIN32
  fDestinationSocketAddress.sin_addr.S_un.S_addr = inet_addr(dest_host);
#else
  fDestinationSocketAddress.sin_addr.s_addr = inet_addr(dest_host);
#endif
  fDestinationSocketAddress.sin_port = htons(dest_port);
  fDestinationSocketAddress.sin_family = AF_INET;

  memset(&fReceiveSocketAddress, 0, sizeof(fReceiveSocketAddress));
#ifdef _WIN32
  fReceiveSocketAddress.sin_addr.S_un.S_addr = inet_addr(receive_host);
#else
  fReceiveSocketAddress.sin_addr.s_addr = inet_addr(receive_host);
#endif
  fReceiveSocketAddress.sin_port = htons(receive_port);
  fReceiveSocketAddress.sin_family = AF_INET;

  errno = 0;
  int ret = bind(fSocket, (struct sockaddr*)&fReceiveSocketAddress, sizeof(fRece
-:---  BaseInterface.cxx   30% L109   SVN-10221   (C++/l AC Abbrev Isearch)
I-search: win32
```

Code Snippets (CMake)

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)

aux_source_directory(. SOURCES)

if("${CMAKE_CXX_COMPILER_ID}" STREQUAL "Clang")
  list(APPEND CMAKE_CXX_FLAGS "-std=c++11 -stdlib=libc++ -Wextra")
elseif("${CMAKE_CXX_COMPILER_ID}" STREQUAL "GNU")
  execute_process(COMMAND ${CMAKE_C_COMPILER} -dumpversion OUTPUT_VARIABLE GCC_V
ERSION)
  if(GCC_VERSION VERSION_GREATER 4.7 OR GCC_VERSION VERSION_EQUAL 4.7)
    list(APPEND CMAKE_CXX_FLAGS "-std=c++11 -Wextra")
  else()
    list(APPEND CMAKE_CXX_FLAGS "-std=c++0x -Wextra")
  endif()
endif()
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -O2")

if(FITS)
  list(APPEND EXTLIBS cfitsio)
endif()

if("${CMAKE_CXX_COMPILER_ID}" STREQUAL "MSVC")
  list(APPEND EXTLIBS ws2_32.lib)
endif()

add_library(TARGET SHARED ${SOURCES})

target_link_libraries(TARGET ${EXTLIBS})

if(WIN32)
# todo: I don't know how to install files under C:\Program Files (x86)\ ...
  install(TARGETS TARGET ARCHIVE DESTINATION lib COMPONENT library)
else()
  install(TARGETS TARGET LIBRARY DESTINATION lib COMPONENT library)
endif()

file(GLOB INCS "${TARGET_SOURCE_DIR}/inc/target/*h")
install(FILES ${INCS} DESTINATION include/target COMPONENT headers)
```

-:--- CMakeLists.txt Top L1 SVN-12573 (CMAKE)

CMakeLists.txt

```
if(PYTHON)
  message("Python support is added")
  find_package(SWIG REQUIRED)
  find_package(PythonLibs REQUIRED)

  include(${SWIG_USE_FILE})
  if(NUMPY)
    execute_process(COMMAND python -c "import numpy; print numpy.get_include()" ➥
➥OUTPUT_VARIABLE NUMPY_INCLUDE_DIR OUTPUT_STRIP_TRAILING_WHITESPACE)
    set(CMAKE_SWIG_FLAGS -DNUMPY -I${TARGET_SOURCE_DIR})
    message("${CMAKE_SWIG_FLAGS}")
    include_directories(${NUMPY_INCLUDE_DIR})
  endif()
  include_directories(${PYTHON_INCLUDE_DIRS})

  set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -D__STDC_FORMAT_MACROS")

  set_source_files_properties(target.i PROPERTIES CPLUSPLUS ON)
  swig_add_module(target python target.i ${SOURCES})
  swig_link_libraries(target ${PYTHON_LIBRARIES} ${EXTLIBS})

  execute_process(COMMAND python -c "from distutils.sysconfig import get_python_➥
➥lib; print get_python_lib()" OUTPUT_VARIABLE PYTHON_SITE_PACKAGES OUTPUT_STRIP_T➥
➥RAILING_WHITESPACE)

  install(TARGETS _target DESTINATION ${PYTHON_SITE_PACKAGES})
  install(FILES ${CMAKE_BINARY_DIR}/src/target.py DESTINATION ${PYTHON_SITE_PACK➥
➥AGES} COMPONENT pylibrary)
endif(PYTHON)
```

-:--- CMakeLists.txt Bot L75 SVN-12573 (CMAKE)

Code Snippets (SWIG)

target.i

```
// Use [] operator in Python
%extend TARGET::EventFile {
  FitsCardImage* __getitem__(const char* keyword) {
    return $self->GetCardImage(keyword);
  }
}

// Returns any type
%extend TARGET::FitsCardImage {
%pythoncode {
def GetValue(self):
    v = _target.FitsCardImage_GetValue(self)
    if v.IsBool():
        return v.AsBool()
    elif v.IsDouble():
        return v.AsDouble()
    elif v.IsFloat():
        return v.AsFloat()
    elif v.IsInt():
        return v.AsInt()
    elif v.IsLong64():
        return v.AsLong64()
    elif v.IsString():
        return v.AsString()
}
}

// todo: If the following line is uncommented, T5CameraModule::PopEvent() works
// without memory leak in Python. But then ToT5CameraModuleEvent does not work.
// Must be investigated...
// %newobject TARGET::T5CameraModule::PopEvent();

%newobject TARGET::EventFile::GetCardImage(const char* keyword);
%newobject TARGET::EventFile::GetT5EvalBoardEvent(uint32_t i);
%newobject TARGET::EventFile::GetT5CameraModuleEvent(uint32_t i);
%newobject TARGET::T2EvalBoardEvent::MakeGraph();
%newobject TARGET::T5EvalBoardEvent::MakeGraph();
%newobject TARGET::T5CameraModuleEvent::MakeGraph();
-:---  target.i       23% L58   SVN-14819  (C/l AC Abbrev)
```

Code Snippets (Python)

Init.py

```
import target
import time
import subprocess
import os

def FindModuleIP(iproot="192.168.0."):

    for ping in range(1,255):
        ip  = iproot + str(ping)
        result = subprocess.call(['ping', '-c', '1', '-W', '0.01','-q',ip])
    #print ping, result

    lines=os.popen('arp -a')
    for line in lines:
        if iproot in line and not 'ff:ff:ff:ff:ff:ff' in line and not 'incomplete' in line:
            unique_ip=line[13:16].split(')')[0]

    ipaddr=iproot+unique_ip

    return ipaddr

def ConnectModule(ip="192.168.0.173", buffersize=196724):
    """
    check/modify buffer size in OS and set accordingly
    """
    module = target.T5CameraModule()
    module.Open("0.0.0.0", 8106, ip, 8105, buffersize)
    module.SetTimeOut(1000)
    return module

def ConnectTesterBoard(ip="192.168.1.173", buffersize=196724):
    """
    check/modify buffer size in OS and set accordingly
    """
    tester = target.T5TesterBoard()
    tester.Open("0.0.0.0", 8107, ip, 8104, buffersize)
    tester.SetTimeOut(1000)
```

-:--- Init.py Top L1 SVN-7040 (Python AC)

testTrigger.py

```
    def setTriggerParams(self,pmtref4,thresh):

        T5_PMTref4= (target.T5_PMTref4_0, target.T5_PMTref4_1, target.T5_PMTref4_2, target.T5_PMTref4_3)
        T5_THResh = (target.T5_THResh_0,  target.T5_THResh_1,  target.T5_THResh_2,  target.T5_THResh_3)

        for asic in range(self.nasic):
            self.module.WriteTARGETRegister(1, 1, T5_PMTref4[asic] , pmtref4) # 0x20, 0x1D, 0x1A, 0x17
            self.bookkeeping.add_command("module.WriteTARGETRegister(%s,%s,%s,%s)"%(str(1),str(1),str(T5_PMTref4[asic]),str(pmtref4)))
            self.module.WriteTARGETRegister(1, 1, T5_THResh[asic]  , thresh) # 0x21, 0x1E, 0x1B, 0x18
            self.bookkeeping.add_command("module.WriteTARGETRegister(%s, %s,%s,%s)"%(str(1),str(1),str(T5_THResh[asic]),str(thresh)))


    def setFuncGen(self,inputf=1.e3,width=8.e-9,edge=5.e-9):

        self.inputf=inputf

        if isinstance(self.fg, Agilent33250A.Agilent33250A):
            self.fg.sendCmd("OUTP OFF")
            self.fg.sendCmd("BURS:STAT OFF")
            self.fg.sendCmd("FUNC PULS")
            self.fg.sendCmd("VOLT:UNIT VPP")
            self.fg.sendCmd("PULS:WIDT "+str(width))
            self.fg.sendCmd("PULS:TRAN "+str(edge))
            self.fg.sendCmd("FREQ "+str(int(inputf)))
        else:
            # Agilent 33521A
            self.fg.write("OUTP OFF")
            self.fg.write("BURS:STAT OFF")
            self.fg.write("FUNC PULS")
            self.fg.write("VOLT:UNIT VPP")
            self.fg.write("FUNC:PULS:WIDT "+str(width if width > 16e-9 else 16e-9
```

-:--- testTrigger.py 17% L77 SVN-7040 (Python AC)

Summary

- Using Python for hardware development is very successful as well as high-level data analysis in gamma-ray astronomy
- I have developed libTARGET for CTA hardware and it has been used in many lab testes at several institutions for three years
  - Different OSs: Mac, Linux, and Windows
  - Different languages: C++ and Python
- libTARGET will be replaced by more sophisticated libraries in very near future to improve the speed and to make it more C++11 like
- If you have any questions how to develop CMake/C++11/SWIG/Python projects, please feel free to contact me