Pydial

Contents

1 Introduction

"Pyserverdial" is a python framework that allow to send and analyse basic queries. Theses tests are in the *svn/calice/online-sw/trunk/pyserdiag* directory.

Clik on "Pyserverdial", "connect" to access all links one by one.
according to */calice@polcaldaq:.config/Calice/pyUSBSerialTests.conf*

- `echo` for testing send and reply
- `serial://COM1:115200` for RS232 on window
- `serial://dev:ttyS1` for RS232 on linux
- `serial://FTDI/DIF2` for USB link to DIF
- `serial:///dev/ttyUSB0:115200` for CCC via USB/RS232 apdapter
- `xmlrpc://` using RPC, the point to point remote procedure call (not used)
- `tcp://192.168.1.20:2300//` for CCC via comEth adapter
- `eth://1:2:3:4:5:6@nic4` for LDA on Windows
- `eth://1:2:3:4:5:6@eth6` for LDA on Linux

This script is the base of GUI interfaces for all the modules. For instance, "(send) load file" » "exemples" » "llrtests" » "GUI_DIF.py" will open the DIF tests popup.

1.1 Architecture

- Controler is defined in *main.py*.
- DUT means Device Under Test object
- SEQ is a text input parsed block
- BLOCK is an octets block
- DevAddr is a string to identify devices
- action_loader use a sandbox "calicediag" to run object code
- inspect: standard object introspection python module
- According to "help" menu entry doc parsing is using parameters names

1.2 Compilation

1.2.1 Ubuntu LTS Lucid 10.04

```
# aptitude install qt4-designer pyqt4-dev-tools python-dpkt python-pcapy
$ ./setup.py build

$install -D /usr/bin/python /opt/ubuntu/usr/bin/netraw_python
# aptitude install libcap2-bin
# setcap 'CAP_NET_RAW+eip CAP_NET_ADMIN+eip' /opt/ubuntu/usr/bin/netraw_python

# getcap /opt/ubuntu/usr/bin/netraw_python
netraw_python = cap_net_admin,cap_net_raw+eip

$ /opt/ubuntu/usr/bin/netraw_python gui.py
```

1.2.2 Windows XP

According to *svn/calice/online-sw/trunk/pyserdiag/README.txt*, the gui also compile on windows with:

- Python 2.6.6
- PyQT 4.8.2
- py2exe 0.6.9
- msvcp90.dll from VC++ 2008 Express Fr to `c:\python26\DLLs\`
- click on `setup`
- click on `gui`

For serial link (no need to recompile):

- Create COM1 from VirtualBox and add it ("ajout de matériel") into Windows
- PySerial 2.5
- Install Visual C++ 2008 Express FR (just to have a C compiler)
- Install WpdPcap 4.1.2 sources (winpcap dev kit) into `c:\devel\oss\`
- pcapy 0.10.8: commenter tous les `#ifdef HAVE_PCAP_SENDPACKET`
- `c:\python26\python.exe setup.py install`
- install dpkt 1.7
- remove *Build* and *Dist* directories. `win> setup.py` will all build into *dist/*. *dist/gui.exe* is a 12MB stand-alone executable.
- it only required *wpcap.dll* istalled by WinPcap 4.1.2 driver

1.2.3 Test on Windows XP

- Upgrade Windows to SP3
- Install WinPcap 4.1.2 driver
- Run the pyserdiag GUI

Note: libpcap doesn't run using some ethernet drivers.

2 LDA

2.1 LDA_version

Send a packet to get the LDA version: Ask for registers 300f, 300e, 3000, 3015, 3010, 1002, 1000, 100a, 100e, 1022 and 1024.

- Sub System Encoding: 0x00 / LDA Registers
- Operation Encoding: 0x02 / Read
- LDA_Modifier: 0000 / packet should not be sent down
- LDA_PktID: 0000

- LDA_DataLength: 000b / For Register operations on the LDA it is the total number of LDA Register Packets that follow.

```
0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0002
0x0010:  0000 0000 000b 300f 0000 0000 300e 0000
0x0020:  0000 3000 0000 0000 3015 0000 0000 3010
0x0030:  0000 0000 1002 0000 0000 1000 0000 0000
0x0040:  100a 0000 0000 100e 0000 0000 1022 0000
0x0050:  0000 1024 0000 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0004
0x0010:  0000 0000 000b 300f 0000 2302 300e 0000
0x0020:  2011 3000 0000 0000 3015 0000 0000 3010
0x0030:  0000 0000 1002 0000 0010 1000 0000 0010
0x0040:  100a 0000 5249 100e 0000 0000 1022 0000
0x0050:  0010 1024 0000 0001
```

2.2 LDA_get_DIF_status

Read the 19 registers from bank 1 (DIF bank) of the LDA: Ask for registers 1000, 1002, 1004, 1006, 1008, 100a, 100b, 100e, 1010, 1011, 1012, 1013, 1018, 1019, 101a, 101b, 1020, 1022 and 1024.

- Sub System Encoding: 0x00 / LDA Registers
- Operation Encoding: 0x02 / Read
- LDA_Modifier: 0000 / packet should not be sent down
- LDA_PktID: 0000
- LDA_DataLength: 0013 / For Register operations on the LDA it is the total number of LDA Register Packets that follow.

```
0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0002
0x0010:  0000 0000 0013 1000 ffff ffff 1002 ffff
0x0020:  ffff 1004 ffff ffff 1006 ffff ffff 1008
0x0030:  ffff ffff 100a ffff ffff 100b ffff ffff
0x0040:  100e ffff ffff 1010 ffff ffff 1011 ffff
0x0050:  ffff 1012 ffff ffff 1013 ffff ffff 1018
0x0060:  ffff ffff 1019 ffff ffff 101a ffff ffff
0x0070:  101b ffff ffff 1020 ffff ffff 1022 ffff
0x0080:  ffff 1024 ffff ffff

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0004
0x0010:  0000 0000 0013 1000 0000 0010 1002 0000
0x0020:  0010 1004 0000 0000 1006 0000 0000 1008
0x0030:  0000 0000 100a 0000 5000 100b 0000 0000
0x0040:  100e 0000 0000 1010 0000 0000 1011 0000
0x0050:  0000 1012 0000 0000 1013 0000 0000 1018
0x0060:  0000 0000 1019 0000 0000 101a 0000 0000
0x0070:  101b 0000 0000 1020 0000 0000 1022 0000
0x0080:  0010 1024 0000 0001
```

2.3 LDA_en_restart_links

Reset the LDA DIF links according to the mask: Write registers 1002, 1000, 1008, 4006, 4007 and 4008.

- Sub System Encoding: 0x00 / LDA Registers

- Operation Encoding: 0x01 / Write
- LDA_Modifier: 0000 / packet should not be sent down
- LDA_PktID: 0000
- LDA_DataLength: 0006 / For Register operations on the LDA it is the total number of LDA Register Packets that follow.

```
lda_out_mask  = 0x10
host_mac_addr = 00:22:19:00:3d:8b

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0001
0x0010:  0000 0000 0006 1002 0000 0010 1000 0000
0x0020:  0010 1008 0000 0010 4006 0000 3d8b 4007
0x0030:  0000 1900 4008 0000 0022

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0003
0x0010:  0000 0000 0006 1002 0000 0010 1000 0000
0x0020:  0010 1008 0000 0010 4006 0000 3d8b 4007
0x0030:  0000 1900 4008 0000 0022 0000
```

note: light the DCC green "link" led or the second upper DHCAL red led in case of direct LDA to DIF.

2.4 LDA_send_fast_command

"Send a fast command to the DIF(s)/DCC:

- Command Word: is set to a constant. Currently defined as 0xFA57.
- DIF_Link: 0010 / A mask which defines which port the command is for. A value of 0xFFFF would be used as a broadcast to all currently active DIF links.
- Comma: defines which comma character to use
- Data: defines which byte to send as the data byte.
- Parity: is a simple check (computed data).

```
lda_out_mask = 0x10
comma        = 0x7c
data         = 0x42

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c42 0015
```

2.5 LDA_dcm_relock

Reset the LDA DIF links according to the mask: Write lda out mask into register 1024 and then clean this register.

- Sub System Encoding: 0x00 / LDA Registers
- Operation Encoding: 0x01 / Write
- LDA_Modifier: 0000 / packet should not be sent down
- LDA_PktID: 0000

- LDA_DataLength: 0001 / For Register operations on the LDA it is the total number of LDA Register Packets that follow.

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0001
0x0010:  0000 0000 0001 1024 0000 0010

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0003 // Write ACK
0x0010:  0000 0000 0001 1024 0000 0010

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0001
0x0010:  0000 0000 0001 1024 0000 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0003
0x0010:  0000 0000 0001 1024 0000 0000
```

3 DCC

3.1 send_FC_DCC_reset

Send a fast command to reset the DCC (7c/10):

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c10 0035
```

3.2 send_FC_DCC_get_status

Send FCMD K28.3/D15.0 (aka. 7C/D15.0, DCC get status) and print out the whole DCC register page:

- Sub System Encoding: 0x01 / DIF Transport
- Operation Encoding: 0x09 / Not documented ?!
- LDA_Modifier: 0005 / packet receive from LDA port 5
- LDA_PktID: d00d ?!
- LDA_DataLength: 0001 / For DIF operations it is the Number of DIF Packets that follow.
- DIF_packettype: e000 / DCC identifier
- DIF_packetId: 7c2f
- DIF_type_modifier: 0f40
- DIF_data_length: 0035

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c0f 0015

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0109
0x0010:  0005 d00d 0001 00e0 2f7c 040f 3500 1807
0x0020:  0400 ff01 0000 0000 ff01 0000 0200 0000
0x0030:  0000 0400 0000 0000 0600 0000 0000 0800
```

```
0x0040:  4912 0000 0a00 490a 0000 0b00 0000 0000
0x0050:  0e00 0000 0000 1000 0000 0000 1100 0000
0x0060:  0000 1200 0000 0000 1800 0000 0000 1900
0x0070:  0000 0000 1a00 0000 0000 2000 0001 0000
0x0080:  2200 0100 0000 2400
```

note: the DIF and DCC inverts byte order: 0xabcd is sent as 0xcdba.

3.3 send_FC_DCC_init_links

Note: please remind that all the fast command are broadcasted to all DIFs connected to the DCC.

Send FCMD K28.3/D15.1 (aka. 7C/D15.1, DCC init links):

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c2f 0035
```

3.4 send_BT_DCC_get_status

Send BT "DCC get status" (dcc nibble 0x, type mod 2) and print out the whole DCC register page:

```
lda_out_link = 5

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0102
0x0010:  0005 0000 0008 00e0 5555 0200 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0109
0x0010:  0005 d00d 0001 00e0 2f7c 040f 3500 1807
0x0020:  0400 ff01 0000 0000 ff01 0000 0200 0000
0x0030:  0000 0400 0000 0000 0600 0000 0000 0800
0x0040:  4912 0000 0a00 490a 0000 0b00 0000 0000
0x0050:  0e00 0000 0000 1000 0000
```

3.5 send_BT_DCC_config_tx_rx

Configure TX/RX enable regs of DCC (channel bitmaps):

```
lda_out_link = 5
txen         = 0x1ff // dcc out mask enabling transmission
rxen         = 0x1ff // dcc out mask enabling reception

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 0014 00e0 5555 0100 0600 ff01
0x0020:  0000 0000 ff01 0000 0200
```

note: light the second upper DHCAL red leds.

3.6 send_BT_DCC_restart

Send restart:

```
lda_out_link = 5
restart_mask = 0x1ff

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 000e 00e0 5555 0100 0300 0000
0x0020:  0000 0800
```

3.7 send_BT_DCC_start_RTT

Start a new RTT measure session:

```
lda_out_link = 5
channels     = 0x113

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 0014 00e0 5555 0100 0600 1301
0x0020:  0000 0400 1301 0000 0800
```

3.8 send_BT_DCC_stop_RTT

Stop a RTT measure session:

```
lda_out_link = 5

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 000e 00e0 5555 0100 0300 0000
0x0020:  0000 0400
```

3.9 send_BT_DCC_relock_DCM

Write 1 to bit 1 of DCC's register 0x24:

```
lda_out_link = 5

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 000e 00e0 5555 0100 0300 0200
0x0020:  0000 2400
```

3.10 send_BT_DCC_register_blob

Send the given sequence of bytes to the DCC with the given type modifier (1 = status page write):

```
lda_out_link      = 5
dif_type_modifier = 0x1
ascii_bytes       = ff ff 00 00

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 000e 00e0 5555 0100 0300 ffff
0x0020:  ffff 0000
```

3.11 send_MULTI_BT_DCC_get_status

```
lda_out_link = 5
times        = 2

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0102
0x0010:  0005 0000 0008 00e0 5555 0200 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0109
0x0010:  0005 d00d 0001 00e0 2f7c 040f 3500 1807
0x0020:  0400 0100 0000 0000 0100 0000 0200 0100
0x0030:  0000 0400 0100 0000 0600 0100 0000 0800
0x0040:  0100 0000 0a00 0100 0000 0b00 0100 0000
0x0050:  0e00 0100 0000 1000 0100
```

```
0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0102
0x0010:  0005 0000 0008 00e0 5555 0200 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0109
0x0010:  0005 d00d 0001 00e0 2f7c 040f 3500 1807
0x0020:  0400 0100 0000 0000 0100 0000 0200 0100
0x0030:  0000 0400 0100 0000 0600 0100 0000 0800
0x0040:  0100 0000 0a00 0100 0000 0b00 0100 0000
0x0050:  0e00 0100 0000 1000 0100
```

4 DIF

4.1 send_FC_DIF_reset

Send FCMD K28.3/D1.1 (aka. 7C/21, reset DIF):

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c21 0015
```

note: This fast command is used to stop the rng test.

4.2 send_BT_DIF_register_state

Send a block transfer command to check if DIF are ready or not.
DIF reply with all registers at 0 if ready.

```
0x0000:  5e70 0cd2 777f 001b 217a bc85 0810 0102
0x0010:  0005 cba6 000a 0290 adde 1200 0100 0200

(DIF reply)
0x0000:  001b 217a bc85 5e70 0cd2 777f 0810 0109
0x0010:  0005 d00d 0001 0290 dada 1200 1000 0000
0x0020:  0000 0000 0000 0000 0000 0000 0000 0000
0x0030:  0000 0000 0000 0000 0000 0000 0000 cb91
```

4.3 send_BT_DIF_command

Send a block transfer command (DIF type modifier) + 1 data word:

```
lda_out_mask =
```

4.4 send_BT_DIF_write_debug_fifo

Write some data (sequence, starting at 1) to the DIF internal fifo:

```
lda_out_mask       = 0x10
dcc_pkttype_nibble = 9
num_bytes          = 24

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0000 0020 0190 5555 0a00 1800 0102
0x0020:  0304 0506 0708 090a 0b0c 0d0e 0f10 1112
0x0030:  1314 1516 1718
```

4.5 send_FC_DIF_read_debug_fifo

Send FCMD K28.3/D2.1 (aka. 7C/22, read debug fifo):
note: this fast command differ between firmware versions 1 and 2:

- Firmware v1

```
lda_out_link = 0x10
```

- Firmware v2

```
lda_out_link = 0x10
```

4.6 send_BT_DIF_read_debug_fifo

Send BT 0xe to read the contents of the DIF internal FIFO:

```
lda_out_mask       = 0x10
dcc_pkttype_nibble = 9

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0102
0x0010:  0005 0000 000a 1090 5555 0100 0100 0100

No reply... TODO
```

note: this doesn't works on firmware v2.

4.7 send_BT_DIF_write_RAM

Send BT 0xe to read the contents of the DIF internal FIFO:

```
lda_out_link =
```

4.8 send_BT_DIF_config_rndgen

Load Register bank 1 (DIF pseudo-random packet generator):

```
lda_out_link =
```

note:

- Configuration via `send_BT_DIF_config_rndgen` with "en" disabled, then click
- If num rndpkts set to 511 then generate an infinit stream of RNG packets
- **IMPORTANT**: inter rndpkt gap soit **must be** > 16
- Verification by `send_BT_DIF_read_bank` select_back set to 1 (cf map registers at Firmwares_DIF_de_test)
- Start with `send_BT_DIF_config_rndgen` with avec "en" enabled
- RAZ configuration: `send_BT_DIF_config_rndgen` with "en" disabled
- Infinit RNG packets stream should be stopped using `send_FC_DIF_reset`

4.9 send_BT_DIF_read_bank

Read DIF REG bank (BTCMD X0012):

```
lda_out_link =
```

4.10 send_BT_DIF_ignored

Read DIF REG bank (BTCMD Xff):

```
lda_out_link =
```

4.11 send_FC_DIF_Guillaume

Custom fast command:

```
lda_out_link =
```

4.12 send_BT_DIF_cmd_register

Send a block transfer command (DIF type modifier) + 1 data word:

```
lda_out_link =
```

5 Hight Level tests

5.1 Configure links

- `LDA_en_restart_links` light the DCC green "link" led or the second upper DHCAL red led in case of direct LDA to DIF.

```
0x0000:  5e70 0cd2 7968 0022 1900 3d8b 0810 0001
0x0010:  0000 0000 0006 1002 0000 0001 1000 0000
0x0020:  0001 1008 0000 0001 4006 0000 3d8b 4007
0x0030:  0000 1900 4008 0000 0022

0x0000:  0022 1900 3d8b 5e70 0cd2 7968 0810 0003
0x0010:  0000 0000 0006 1002 0000 0001 1000 0000
0x0020:  0001 1008 0000 0001 4006 0000 3d8b 4007
0x0030:  0000 1900 4008 0000 0022 0000
```

- `send_BT_DCC_config_tx_rx` light the second upper DHCAL red led.

```
0x0000:  5e70 0cd2 7968 0022 1900 3d8b 0810 0101
0x0010:  0001 0001 0014 00e0 5555 0100 0600 ff01
0x0020:  0000 0000 ff01 0000 0200
```

Note: provide a host mac address with "LDA_en_restart_links" if 2 setups share the same switch.

5.2 FIFO test

Status:

	ECALv1	ECALv2	DHCALv2
BT write	ok	ok	ok
FC read	data + echo	need flush at power on ; echo + data	data + echo
BT read	ok	KO	KO

note: We may prefer to use block transfer instead of fast command because we don't want the read orders arrive before the write orders.

- `LDA_en_restart_links`
- `send_BT_DCC_config_tx-rx`
- `send_BT_DIF_write_debug_fifo`

```
0x0000:  5e70 0cd2 7968 0022 1900 3d8b 0810 0101
0x0010:  0001 0000 0020 0110 5555 0a00 1800 0102
0x0020:  0304 0506 0708 090a 0b0c 0d0e 0f10 1112
0x0030:  1314 1516 1718
```

- `send_BT_DIF_read_debug_fifo`

```
0x0000:  5e70 0cd2 7968 0022 1900 3d8b 0810 0102
0x0010:  0001 0000 000a 0210 5555 0e00 0200 0100

0x0000:  0022 1900 3d8b 5e70 0cd2 7968 0810 0109
0x0010:  0001 d00d 0001 0112 0304 0506 0708 090a
0x0020:  0b0c 0d0e 0f10 1112 1314 1516 1718 f1da
0x0030:  33ee 0000 0000 0000 0000 0000
```

5.3 RNG test

Status:

	ECALv1	ECALv2	DHCALv2
hline config rng	ok	ok	ok
BT reset	ok	ok	KO

- `LDA_en_restart_links`
- `send_BT_DCC_config_tx-rx`
- `send_BT_DIF_config_rndgen`

```
0x0010:  0001 0000 0010 0150 ae60 1000 0400 4000
0x0020:  0100 0100 8000

replies:
0x0000:  ffff ffff ffff 5e70 0cd2 5cd6 0810 0109
0x0010:  0001 d00d 0001 0250 0400 0800 1000 2000
0x0020:  4000 8000 0001 0002 0004 0008 0010 0020
0x0030:  0040 0080 11a0 33e0 7760 eec0 cd21 9a43
0x0040:  3487 79ae e3fc d759 aeb3 4dc7 8b2e 165d
0x0050:  2cba 49d4 8308 0611 0c22 1844 3088 71b0
0x0060:  f3c0 f721 ee43 dc87 a9af 43ff 975e 2ebd
0x0070:  4dda 8b14 1629 2c52 58a4 a1e8 5371 a6e2
0x0080:  5d65 baca 6535 ca6a 94d5 390b 7216 e42c
0x0090:  c859 90b3 31c7 732e 4285
```

```
0x0000:  ffff ffff ffff 5e70 0cd2 5cd6 0810 0109
0x0010:  0001 d00d 0001 e65c ccb9 89d3 0307 060e
0x0020:  0c1c 1838 3070 60e0 d160 a2c1 5523 aa46
0x0030:  548d b9ba 63d5 d70a ae15 5c2b b856 70ad
0x0040:  f1fa f355 e6ab ddf7 ab4f 569f bd9e 6b9d
0x0050:  c79a 9f95 2f8b 4fb6 8fcc 0f39 1e72 3ce4
0x0060:  6968 d2d0 b501 6a03 d406 a80d 501b a036
0x0070:  406d 80da 1115 222a 4454 88a8 01f1 1342
0x0080:  2684 5da8 abf0 4741 8e82 0da5 0bea 0774
0x0090:  0ee8 0d70 1ae0 2560 b9441
```

5.4 Sending ASICs configuration

- LDA_en_restart_links
- send_BT_DCC_config_tx-rx
- send_BT_DIF_cmd_register
- test_lapp2
- test_lapp1

5.5 Read out from ASICs

- LDA_en_restart_links
- send_BT_DCC_config_tx-rx
- send_FC_start_acq
- send_trig_ext_cmd_register
- send_BT_start_RO_command