|  |
| --- |
|  |
| Phio Functions |
| Technical Note |
|  |
| **Giovanni SCOZZOLA** |
| **6/29/2017** |

1. Introduction

Linux Driver validation today is done with the help of unit test already existing. This unit tests are shell based script which load/unload Linux Driver that run/kill daemon for IPP then use a program called Phio to call driver functions.

It is important to analyze what is available and what is not to permit a script to use correct options.

1. Existing functions
   1. Option –m

Set/Get ranging mode:

* **mode value** 
  + 1 = standard
  + 2 = multi zone
  + 3 = autonomous
  + 4 = lite ranging
  1. Option –d

Set the device number to be used. Need to check if this is used for multiple device.

* 1. Option –M

Run a specific number of measurement.

For example:

${BINS}/phio -U=1 -m=1 -s -M=3 -S

In that command the -M specify to run 3 ranges, the –m=1 specify standard ranging.

If no value is given then only one ranging is done

* 1. Option –o

Get ROI values, a parameter will indicate how many ROIs we get.

-o --roi\_get no\_of\_roi

The ROI returned are less or equal to the max stored, so no\_of\_roi are returned if the no\_of\_roi < Max defined ROI else the max value will return.

* 1. Option –O

Set ROI values, values are given as list.

The following will set only one ROI with the coordinates x0 y0 x1 y1:

${BINS}/phio -O="6 9 9 6"

You can specify more ROI by using list:

${BINS}/phio -T=1 -O="0 15 7 8, 8 15 15 8, 0 7 7 0, 8 7 15 0"

To unset the ROI you can run:

${BINS}/phio -T=1 -O

< write here the x y coordinate >

* 1. Option –p

Set/Get the device polling delay in milliseconds.

Get: “-p”

Set: “–p=xxxx”

* 1. Option –P

Set the device polling delay to time\_ms and run a pause of the same time.

[-P --Pause time\_ms]

* 1. Option –r

Get range data (raw) repeat no\_of\_call time.

[-r [=no\_of\_call] --range [no\_of\_call]]

* 1. Option –s

Start ranging.

* 1. Option –S

Stop ranging.

* 1. Option –t

Set/Get timing budget.

-t [=time\_us]

* 1. Option –z

Get multi zone data.

-z [=no\_of\_call]

* 1. Option –Z

Poll multi zone ranging.

-Z no\_of\_range

1. Missing Functions

Usage: phio <option1> <option2> ....

[-r[=no\_of\_call] --range [no\_of\_call]] get range data (raw) repeat no\_of\_call time (override TBD)

-s --start start ranging

-S --stop stop ranging

-t [=time\_us] get timing budget [=set it]

-z [=no\_of\_call] get multi zone data

-Z no\_of\_range poll multi zone ranging

-R --Ref\_spad perform reference spad calibration

-X --Xtalk <method> perform crosstalk calibration using method

-F --oFfset <"mode distance reflectance"> perform offset calibration of type mode with target at given distance in mm and with given reflectance in percent

-c --cal <file> get calibration data and write them in given file name

-C --Cal <file> set calibrating data by reading given file name

-D --Distance[=distance\_mode] set/get distance mode

distance value 1 = short, 2 = medium, 3 = long

-E --xtalk\_Enable[=<0/1>] set/get crosstalk compensation enable

-U --oUtput[=output\_mode] set/get output mode

output mode value 1 = nearest, 2 = strongest

-f --force[=<0/1>] set/get force device on enable

-a --auto=["poll\_ms mode no\_target\_irq distance\_mode distance\_low distance\_high rate\_mode rate\_low rate\_high"] set/get autonomous configuration

-y --sysfs\_on turn on sysfs usage instead of ioctl

-Y --sysfs\_off turn off sysfs usage instead of ioctl

-T --offseT\_correction\_mode[=offset\_correction\_mode] set/get offset correction mode

offset correction mode value 1 = standard, 2 = per zone

-I --optical\_center get optical center offset in roi coordinate

-A --dmax\_reflectance[=reflectance] set/get reflectance used for dmax calculation in percent

-B --dmax\_mode[=dmax\_mode] set/get dmax mode

-w --wcal <file> get zone calibration data and write them in given file name

-W --Wcal <file> set zone calibrating data by reading given file name

-N --tuNing="<key> <value>" set low level layer parameter with key to given value

-g --tunings <file> get tunings values and write them in given file name

-G --tuninGs <file> set tunings values by reading given file name

-h --smudge[=0/1] set/get smudge correction enable

Examples:

- get current timing budget value

./phio -t

- take 10 measures in ranging mode

./phio -O -m=1 -s -Z=10 -S

- take 10 measures in multi zone mode with 4 zones

./phio -O="0 15 7 8, 8 15 15 8, 0 7 7 0, 8 7 15 0" -m=2 -s -Z=10 -S

- take 10 measures in lite ranging at 10 Hz and restore 60hz

./phio -t=100000 -O -m=4 -s -M=10 -S -t=16000

1. Test description
   1. Test list

List of test updated on version 12.8.0 of Linux driver

|  |  |
| --- | --- |
| Test Name | Description |
| autonomous\_low |  |
| crosstalk\_calibration |  |
| distance\_mode |  |
| dmax\_mode |  |
| dmax\_reflectance |  |
| force\_device\_on\_enable |  |
| get\_calibration\_data |  |
| lite |  |
| lite\_roi |  |
| load\_unload |  |
| multizone |  |
| multizone\_4x4 |  |
| offset\_calibration |  |
| offset\_calibration\_after\_ranging |  |
| offset\_calibration\_after\_ranging\_prerange\_only |  |
| offset\_calibration\_prerange\_only |  |
| offset\_calibration\_roi |  |
| offset\_calibration\_roi\_id |  |
| offset\_calibration\_roi\_set\_get |  |
| offset\_correction\_mode |  |
| optical\_offset |  |
| output\_mode |  |
| parameters |  |
| ranging |  |
| ranging\_multiple |  |
| ranging\_roi |  |
| ref\_spad\_calibration |  |
| set\_calibration\_data |  |
| set\_calibration\_data\_with\_ranging |  |
| set\_zone\_calibration\_data |  |
| smudge\_correction\_enable |  |
| sysfs\_autonomous\_config |  |
| sysfs\_calibration\_data |  |
| sysfs\_crosstalk\_enable |  |
| sysfs\_distance\_mode |  |
| sysfs\_dmax\_mode |  |
| sysfs\_dmax\_reflectance |  |
| sysfs\_enable\_debug |  |
| sysfs\_enable\_ps\_sensor |  |
| sysfs\_force\_device\_on\_enable |  |
| sysfs\_mode |  |
| sysfs\_offset\_correction\_mode |  |
| sysfs\_optical\_offset |  |
| sysfs\_output\_mode |  |
| sysfs\_poll\_delay\_ms |  |
| sysfs\_roi |  |
| sysfs\_smudge\_correction\_enable |  |
| sysfs\_timing\_budget |  |
| sysfs\_tuning |  |
| sysfs\_zone\_calibration\_data |  |
| tuning |  |

* 1. autonomous\_low

The following code is the sequence used for the test

|  |
| --- |
| cleanup  load  #trigger interrupt if higher  ${BINS}/phio -a="200 1 0 0 200 500 0 0.000000 0.000000"  ${BINS}/phio --mode=3 -t=100000  start  #after we got three we exit  ${BINS}/phio -M=3  stop  unload |

In this test the option “- - a” is used.

From the help we have:

-a --auto=["poll\_ms mode no\_target\_irq distance\_mode distance\_low distance\_high rate\_mode rate\_low rate\_high"] set/get autonomous configuration

In the above test we have:

|  |
| --- |
| full->pollingTimeInMs = 200  full->config.Rate.High = 0.0;  full->config.Rate.Low = 0.0;  full->config.DetectionMode = 1;  full->config.IntrNoTarget = 0;  full->config.Distance.CrossMode = 0;  full->config.Distance.High = 200;  full->config.Distance.Low = 500;  full->config.Rate.CrossMode = 0; |

phio --mode=3 -t=100000:

This will set device mode to VL53L1\_PRESETMODE\_AUTONOMOUS and option –t change the timing budget VL53L1\_TIMINGBUDGET\_PAR to 100000 micro seconds i.e. 100 ns.

The command “start” is inside function definition and correspond to:

|  |
| --- |
| function start {  ${BINS}/phio -s  } |

This will call stmvl53l1\_start() inside the Phio

${BINS}/phio -M=3:

This command will get only 3 ranging data, at the end this command exits. The next command do a stop ranging and then unload driver.

This command will wait until a new data is available, so if the threshold is not reached the program stuck without a timeout. A new version could be done with a timeout and test this condition.

This test will set the driver in autonomous mode with some thresholds then it gets three ranging data and then stop