API For Production Testing

Confidential

Revision 1 – January 2024

Date	Current Version	Edited by	Description
27th April 2023	1	S.Molin	
1st December 2023	2	S.Molin	SelfTest-API v3
22nd January 2024	3	P.Okumo	SelfTest-API v4

Contents

1 Introduction 3

2 Prerequisites 3

3 API content 3

3.1 Test Programs 3

3.2 Individual Commands 5

3.3 API commands 5

3.3.1 Useful commands in production testers 6

3.4 Additional arguments ..... **Error! Bookmark not defined.**

4 Storing Integral Values with API 7

4.1 Initialize Storage Parameters 7

4.2 Start Recording Integral Values 7

4.3 Set Press Point Event 7

4.4 Store Integral Values 8

4.5 Read Stored Integral Values 8

4.6 Storing Integral Values as a Flow Chart 10

Introduction

This page describes various things about the Application Interface (API) that provides a testing interface for production testing tools.

The API is based on the individual commands and test programs that can be run in a command prompt against a .exe program.

Prerequisites

Production Testing API is based on C++ scripts. The program is well integrated and as such that do not require any library or dependences.

API content

The first phase implementation of the API contains six-point press test for 12-piezo devices, and four-point press test for 10-piezo devices in addition to the individual self-test-commands, like read serial numbers and versions.

- This file describes the API content in html and pdf format.
- Release-folder contains API as an .exe program file.
- src/selftest_settings/Test_info contains the commands that have been implemented in the API.
- src/test_programs contain textual program is both windows batch and bash format for Linux environment.

The API can be used by running ".exe" program "SelfTest-API.exe" from the folder "Release". An example of running a part of the 6-point press test:

```
D:\Test\Selftest-API\src> Set-PressPoint1.sh / Set-PressPoint1.bat

D:\Test\Selftest-API\src> Return-PressIntegrals.sh / Return-PressIntegrals.bat
12      10      11      9       65535   1270    430     33521
```

Test Programs

Test programs are text files that can contain several individual self-test commands conforming a test sequence. They exist in the folder src\test_programs.

The command syntax, when running a test program, is:

SelfTest-API.exe <command**>**

E.g. running the 6-point press test, from src-folder:

```
6-Point-PressTest.sh / 6-Point-PressTest.bat
```

6-point-PressTest

- runs 6-point press test and prints the closest 4 piezo numbers and their integral values from each press point.

- targeted for 12-piezo devices testing.

4-point-PressTest

- runs 4-point press test and prints the closest 4 piezo numbers and their integral values from each press point.
- targeted for 10-piezo devices testing.

Set-PressPoint1 … 6

- Sets location parameters for an individual press point
- Can be used together with Return-PressIntegrals, when the tester synchronizes press event reading with the actuator function.

Return-PressIntegrals

- Prints the closest 4 piezo numbers and the integrals values of the set individual press location.

9-point-XY-Press-test

- runs 9-point press test and prints the event (press = 1, release = 0) and XY coordinate values
- targeted for any device.

Return-XY-Press

- Prints the event (press = 1, release = 0), X and Y coordinate values of one press event

Individual Commands

When using testing interface with individual commands, the command syntax is:

`SelfTest-API.exe [Command Name] <arguments>`

Here is an example of read_build_id-command execution with logfile as an argument:

```
D:\ SelfTest-API2\x64\Release>  .\SelfTest-API.exe read_build_id
Firmware Build ID: j2024-01-17-635 RDP2 E2
d52a17ce68c7326d6bc6f098836ec8704ffebae0
D:\Test\Selftest-API\src>
```

A command **`' SelfTest-API.exe /Usage'`** prints all available tests.

API commands

Test nbr	Test name	Return format	Printout rows	Encoding (LSB/MSB leading)	Response length	Response / functionality	Notes
1	change_parameters	2	1	dec	34	21 parameter values	Requires parameters: parameter number, new value
2	read_build_id	1	All	ascii	64	String	
3	read_controller_id	3	All	hex,LSB	3		
6	read_temperature	1	all	dec,signed	1	Celsius degrees	
7	read_vdd_voltage	2	all	dec	2	cV, cent voltage	
8	read_vreff_voltage	2	all	dec	2	raw value (12 bit), ASIC ref voltage	
10	read_booster_voltage	2	all	dec	2	raw value (12 bit), VHV voltage	
11	read_piezo_adc_voltages	2	1	dec	2 * piezo nbr	raw value (12 bit), ADC input voltage	
14	start_integral_storage	1	1	ascii	4	Pass / Fail	Activates recording of integral values
16	init_piezo_foil_test	1	all	pass/fail	0	Piezo polarization pulse	
25	play_haptics	1	all	pass/fail	1	Pass / Fail	Requires parameters: 13 + channels in a bit mask
27	disable_fw_haptics	1	all	pass/fail	1	Pass / Fail	
28	enable_fw_haptics	1	all	pass/fail	1	Pass / Fail	
29	store_integral_values	1	all	dec	1	1 / 0	Stores measured integral values per press point
30	read_stored_integrals	2	2,space	dec	48	Report of the latest stored integral values / Fail	The output contains headers and meta data at first
30	read_nvm_page	2	4,space	hex	2048	2 kB NVM chunk	
31	erase_nvm_page	1	all	pass/fail	1	Pass / Fail	
33	run_afpt	2	all,tab	dec	2 * 4 words	Returns closest 4 piezo numbers and their integral values, when touch pad pressed	Coordinate parameters (7 & 8) must be set with change_parameters, values as mm from top left corner
35	read_capsense_flag	1	all	bool	1	Capacitance sensor presence in I2C master bus	
36	read_bootloader_version	1	all	hex	20	String	
37	read_aito_controller_sn	1	all	ascii	32	String	

39	test_piezo_taus	2	4,space	dec	8 * piezo nbr	Charge, Discharge, MaxVoltage, ChargeRelation	
40	test_piezos	1	3,space	dec		Distortion, Capacitance, Charge(Stiffness) per each piezo	
41	test_booster	2	all,space	dec		Charging time constant, Discharging time constant, Max voltage (* 3.3 V * 95 / 4096), Low voltage charge energy, High voltage charge energy, Booster power coefficient, Booster current limit (* 125 mA)	
43	read_asic_serial_number	16	1	hex,LSB	32	String	
44	return_haptic_xy	1	all,space	dec	3 * multiple	Press/Release, X, Y; Continous until stopped with cntl-c	Continuous reading until interrupted by user
44	return_haptic_xy_1	1	all,space	dec	3	Press/Release, X, Y; One event	Returns one press event coordinates, long timeout
44	return_haptic_xy_2	1	all,space	dec	2 * 3	Press/Release, X, Y; Both press and release event	Returns press and release event coordinates, extra-long timeout
45	return_true	1	all	bool	1	True, sanity check	
46	return_false	1	all	bool	1	False, sanity check	
47	go_to_active_idle	1	all	dec	1	Wake-up reason, 1 = communication on I2C bus, 2 = piezo activity, 4 = I2C master (capsense)	
48	reset	0	0	0		Reset Aito Controller FW	

Useful commands in production testers

- read_aito_controller_sn
- read_bootloader_version
- read_build_id
- read_controller_id
- reset
- return_haptic_xy
- test_piezos
- test_booster
- run_afpt
- change_parameters
- start_integral_storage
- store_integral_values

Storing Integral Values with API

The measured integral values of 6-point press test can be stored into Aito firmware Non-Volatile memory (NVM) during the test procedure with specific SelfTest-commands.

This storage process has four different specific steps out of which the latter two need to be performed for each press point:

1. Initialize storage date
2. Start recording integral values
3. Set press point event just before executing run_afpt-command
4. Run store_integral_values-command right after executing run_afpt-command

When the integral values have been stored to NVM, they can be read with read_stored_integrals-command. With this command an application can verify that storing the values has been successful.

Initialize Storage Parameters

Initialize storage parameters consists of setting the date parameters for NVM storage. The date is set with `change_parameters` command and it needs to be done individually for the year, the month, and the day. The parameter indices are: 19 for a year, 20 for a month, and 21 for a day.

E.g., setting the date 21st November 2023, you need to run commands:

```
SelfTest-API.exe change_parameters 19 23
SelfTest-API.exe change_parameters 20 11
SelfTest-API.exe change_parameters 21 21
```

Note that setting the year is done without the century in the argument. As output change_parameters-command prints all (currently 21) available parameter values.

Start Recording Integral Values

Start recording integral values consists of executing `start_integral_storage`-command to inform Aito firmware that now it is time to record the press point results for NVM storage.

An execution example:

```
SelfTest-API.exe start_integral_storage
```

On success this command outputs “Pass” otherwise “Fail”.

Set Press Point Event

Setting press point event must be done for each press point, and it needs to be done before commanding `run_afpt`. This setting is also done with `change_parameters`-command, the parameter index is 18, and the value to be given is press point number + 192.

Example setting press point 1:

```
SelfTest-API.exe change_parameters 18 193
```

Example setting press point 6:

```
SelfTest-API.exe change_parameters 18 198
```

The tags 193 … 198 in NVM define that the following values represent the integral values measured during the final production test (AFPT) phase.

The tags 129 … 134 in NVM define that the values represent the integral values measured during the module integration production test (MIPT) phase.

Store Integral Values

After each press point is performed (i.e., `run_afpt` has returned the integral values), `store_integral_values`–command needs to be executed to inform Aito firmware to write the values to the Aito Controller's NVM memory. The firmware has recorded the integral values, so they are not sent with the command.

An example of storage-command:

```
SelfTest-API.exe store_integral_values
```

On success this command outputs "1" otherwise "0".

Read Stored Integral Values

The latest stored integral values can be read and displayed with `read_stored_integrals`-command. This command searches the latest stored values and prints the date and each press point with related piezo numbers and their integral values. With this command an application can verify that storing the values has been successful.

An example of read_stored_integrals-command:

```
SelfTest-API.exe read_stored_integrals
```

An example of the output of this command:

```
Date: 2023-11-22
Test Phase: Final
Press Point: 1
0009 12121
0010 13333
0011 9876
0012 8877
Press Point: 2
0001 7121
0002 8345
0010 9855
0012 8123

...

Press Point: 6
0003 7543
0004 8000
0005 9955
0007 9123

Press Point: 1
0009 0010 0011 0012
```

```
12121 13333 9876 8877
Press Point: 2
0001 0002 0010 0012
7121 8345 9855 8123

...

Press Point: 6
0003 0004 0005 0007
7543 8000 9955 12123
```

In case the reading fails, or the integral values are not stored this command outputs “Fail”.

Storing Integral Values as a Flow Chart