xtpl\_printer\_cmds

Table of Contents

[Revision History 3](#_Toc498958224)

[1. Signal related commands 4](#_Toc498958225)

[1.1 --pulse\_a 4](#_Toc498958226)

[1.2 --digital\_out\_ctrl 5](#_Toc498958227)

[1.3 --set\_adc\_a\_meas\_duration 6](#_Toc498958228)

[1.4 --get\_adc\_a\_result 7](#_Toc498958229)

[1.5 --signal\_form\_upload 8](#_Toc498958230)

[1.6 --signal\_parameters 9](#_Toc498958231)

[1.7 --signal\_start 10](#_Toc498958232)

[1.8 --signal\_stop 10](#_Toc498958233)

[Generate example signals 11](#_Toc498958234)

[2. Motors control related commands 13](#_Toc498958235)

[2.1 --smb\_get\_status 13](#_Toc498958236)

[3. System related commands 15](#_Toc498958237)

[4. Periphery related commands 16](#_Toc498958238)

[1.1 --temperature 16](#_Toc498958239)

# Revision History

|  |  |  |
| --- | --- | --- |
| Target Release | Description | Author |
| \_\_release\_20171013\_0900 | Add commands:  --pulse\_A  --digital\_out\_ctrl | Paweł Ządek |
| \_\_release\_20171106\_1300 | modified:  --pulse\_A -> --pulse\_a  Added:  --set\_adc\_a\_meas\_duration  --get\_adc\_a\_result | Paweł Ządek |
| \_\_release\_20171108\_1700 | Added commands:  --signal\_form\_upload  --signal\_parameters  --signal\_start  --signal\_stop  Modified:  --get\_adc\_a\_result  --pulse\_a (in section read status) | Paweł Ządek |
| \_\_release\_20171114\_1330  \_\_release\_20171117\_1300 | Added command:  --smb\_get\_status | Paweł Ządek |
| \_\_release\_20171120\_1608  **[current version]** | Deleted commands:  --signal\_user  --signal\_auto  --signal\_time  --current\_set  --current\_get  --start  --stop | Paweł Ządek |

# Signal related commands

## --pulse\_a

Command syntax:

*ics\_client.exe --dev\_xtpl\_printer IP TCP --pulse\_a P0 P1 P2 P3*

* P0: <0,1> represent subcommand for pulse\_a:
* P0 = 0: read status byte and average results of ADC0 and ADC1 channels.
* P1, P2, P3 are ignore
* P0 = 1: start pulse forming where:
  + P1: <0, 65535> represent desired DAC0 value
  + P2: <0, 65535> represent desired DAC1 value
  + P3: <0, 1677721000> represent pulse duration time parameter. To calculate pulse duration time use formula:

T = (P3\*10ns + 2uS)

Max pulse duration is restricted to 16.777s

Positive response syntax:

* For P0 = 1 (generate pulse):

*0. 0*

*I. OK*

* For P0 = 0 (read status and results):

*0. res0 res1 res2*

*I. Status: res0, Avg ADC0: res1, Avg ADC1: res2*

* Res0 – conversion status:

1 – we are in the middle of conversion results are corrupt

0 – conversion has been finished: Res1 and Res2 have valid values

* Res1 – Average of ADC0 conversions result
* Res2 – Average of ADC1 conversions result

Example scenario:

* Generate pulse on DAC0 (0.5V) and DAC1 (1V) for 102uS
* Read ADCs results, use user prescaler: 1.2345 for ADC0 and 0.001 for ADC1

ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --pulse\_a 1 0x7fff 0xffff 10000

ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --pulse\_a 0

## --digital\_out\_ctrl

Command syntax:

i*cs\_client.exe --dev\_xtpl\_printer IP TCP --digital\_out\_ctrl P0 P1*

* P0: <0x00, 0xFF> represent external relays coils value. Bits are corresponding to the relays coils: MSB: Relay7 (W7), MSB: Relay0 (W1). Logic “0” on corresponding bit will turn on relay coil.
* P1: <0x00, 0x03>. By changing this parameter user can change line-resistance (LR) according to below table:

|  |  |
| --- | --- |
| P1 | LR[kΩ] |
| 0x00 | 55 |
| 0x01 | 5 |
| 0x02 | 50 |
| 0x03 | 0 |

Positive response syntax:

*0. 0*

*I. OK*

Example scenario:

Switch on Relay0, Relay7 coils (coils of relays <1,6> switch off) and set line resistance to 50kΩ

ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 *--*digital\_out\_ctrl 0x7E 0x02

## --set\_adc\_a\_meas\_duration

Command syntax:

*ics\_client.exe --dev\_xtpl\_printer IP TCP --set\_adc\_a\_meas\_duration P0*

* P0: Represent amount of ADC conversions.

After invoking this commands ADC will start measurement. Period of measurement is equal to 1MHz. Measurement time can be calculated according to the following formula:

ADC results can be read-out via **--get\_adc\_a\_result** command.

Positive response syntax:

*0. 0*

*I. OK*

Negative response syntax:

*E. res0*

*I. ERROR res0*

## --get\_adc\_a\_result

Command syntax:

*ics\_client.exe --dev\_xtpl\_printer IP TCP --get\_adc\_a\_result*

Positive response syntax:

*0. res0 res1 res2 res3 res4 res5 res6*

*I. Status: res0, Avg ADC0: res1, AvgADC1: res2*

* Res0 – conversion status:

1 – we are in the middle of conversion results are corrupt

0 – conversion has been finished: Res1 and Res2 have valid values

* Res1 – Average of ADC0 conversions result
* Res2 – Average of ADC1 conversions result
* Res3 – Average of ADC0 conversion when potential on ADC0 is bigger or equal to potential on ADC1 (ADC0≥ADC1)
* Res4 – Average of ADC1 conversion when potential on ADC0 is bigger or equal to potential on ADC1 (ADC0≥ADC1)
* Res5 – Average of ADC0 conversion when potential on ADC0 is less than potential on ADC1 (ADC0<ADC1)
* Res6 – Average of ADC1 conversion when potential on ADC0 is less than potential on ADC1 (ADC0<ADC1)

Example scenario:

* Set ADC measurement to 1024us
* Get ADC result

*ics\_client.exe --dev\_xtpl\_printer IP TCP --set\_adc\_a\_meas\_duration 1024*

*ics\_client.exe --dev\_xtpl\_printer IP TCP --get\_adc\_a\_result*

## --signal\_form\_upload

Command syntax:

*ics\_client.exe --dev\_xtpl\_printer IP TCP --* signal\_form\_upload *P0 P1*

* P0: Represents system buffer number where user signal file was uploaded
* P1: Represents size of user signal in bytes

Positive response syntax:

*0. 0*

*I. OK*

Negative response syntax:

*E. res0*

*I. ERROR res0*

Where res0 is error code

## --signal\_parameters

Command syntax:

*ics\_client.exe --dev\_xtpl\_printer IP TCP --signal\_parameters P0 P1 P2 P3 P4 P5 P6 P7*

* P0: <0,1> represent signal mode
  + P0 = 0: user mode
  + ~~P0 = 1: auto mode~~ **currently not implemented**
* P1: <0, 65535> DAC minimum limiter. Generated signal will not go below this limiter. **Currently not implemented**
* P2: <0, 65535> DAC maximum limiter. Generated signal + DC offset will not exceed this limiter. **Currently not implemented**
* P3: <0, 1> DAC offset channel, DC offset can be add to every signal sample. By the P3 parameter value we can select DAC channel.

* P4: <0, 65535> offset data value, DC offset value can be calculated by the corresponding formula:
* P5: <0, 65535> represent signal multiplication factor. Amplitude on DAC0/1 can by calculated by the corresponding formula:
* P6: <1, 1024> decimation factor. By the changing this parameter we will increase frequency of generated signal. Frequency can by calculated by the formula:
* P7: <0, 4294967295> by changing this factor we are able to decrease frequency of generated signal. Frequency can by calculated by the corresponding formula:

Positive response syntax:

*0. 0*

*I. OK*

Negative response syntax:

*E. res0*

*I. ERROR res0*

## --signal\_start

Command syntax:

*ics\_client.exe --dev\_xtpl\_printer IP TCP --signal\_start*

After invoking this command signal will be generated on DAC0, DAC1.

**Note:** Signal must be previously uploaded (--signal\_form\_upload) to the board and configured by the --signal\_parameters command

Positive response syntax:

*0. 0*

*I. OK*

Negative response syntax:

*E. res0*

*I. ERROR res0*

Where res0 is error code

## --signal\_stop

Command syntax:

*ics\_client.exe --dev\_xtpl\_printer IP TCP --signal\_stop*

After invoking this command signal will stop being generate on DAC0, DAC1.

Positive response syntax:

*0. 0*

*I. OK*

Negative response syntax:

*E. res0*

*I. ERROR res0*

Where res0 is error code

## Generate example signals

Example scenario 1:

* Create new system buffer on server – with size 2048 Byte
* Upload user signal file named “sinus.sig” into new created buffer on server
* Upload user signal to XTPL021 board
* Set signal parameters:
  + User mode P0 = 0,
  + DAC minimum limiter P1 = 0,
  + DAC maximum limiter P2 = 0xffff
  + Assign offset for DAC1 channel P3 = 1;
  + Add 200mV Voltage offset to every sample of DAC1 channel P4 = 13100;
  + Divide initial amplitude of signal by 2, P5 = 32768
  + Set signal frequency to 100kHz (default frequency): P6 =1, P7 =0.
* Start generate user signal on board DACs

*ics\_client.exe --sys\_config\_buffer 127.0.0.1 42100 --config 1 2048*

*ics\_client.exe --sys\_client\_wr\_rd\_buffer 127.0.0.1 42100 --wr\_s 0 0 "sinus.sig"*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --user\_signal\_form\_upload 0 2048*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --signal\_parameters*

*0 0 65535 1 13100 32768 1 0*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --signal\_start*

As the result we will get following wave on DAC0/DAC1 outputs:

**Note:** File “sinus.sig” must be a binary file filled up with one signal period samples

(One samples shall have values in the range of: 0x0000 – 0xFFFF) ordered in big

endian style.

Sample signals file:



# Motors control related commands

1. --smb\_get\_status

Command Overview:

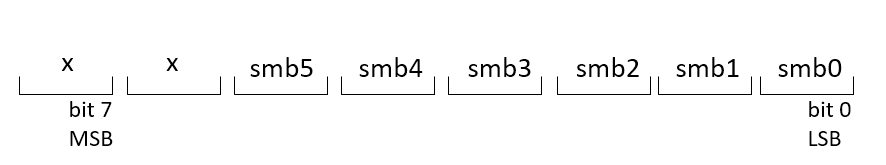
By invoking –smb\_get\_status command, with proper motor selector (mask), status

of selected motor drivers will be return.

Command syntax:

*ics\_client.exe --dev\_xtpl\_printer IP TCP --smb\_get\_status P0*

* P0: <0, 0x3F> represent smb driver selector/mask.



To read status from driver write logic “1” to corresponding bit (to read smb0 P0 =

0x01, to read smb0 & smb5 P0 = 0x21, to read all 6channels status P0 = 0x3f)

Positive response syntax:

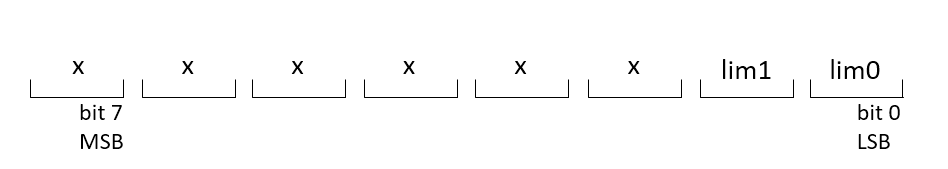
*0. res0 res1 res2 res3*

*I. SMBres0: Current Position: res1, Limiters: res2, State: res3*

**Note:**  numbers of return parameters are depends of numbers of smb drivers from

which status will be read

* res0 – smb driver number
* res1 – current position of res0 smb driver
* res2 – limiters status of res0 smb driver:



*(limiters return value 0 or 1 for asserted limiter is strictly depend on*

***--smb\_set\_limiters*** *command)*

* res3 – state of driver
  + 0 – driver is in idle state
  + 1 – driver is in the middle of move/processing

Negative response syntax:

*E. res0*

*I. ERROR res0*

Where res0 is error code

Example scenario:

(rest of the below command can be found in “Dokumentacja rozkazów

posuwu.docx” )

* Select SMB as motors driver algorithm for motor 0 and 5
* Activate/Enable limiters for smb0 and smb5
* Set current position of smb0 driver to 100000
* Set timing parameters for smb0
* Set maximum position of smb0 driver to 1000000
* Set destination for smb0 to 90000
* Start smb0 and smb5
* Get status of smb0

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --sel\_motor\_module 0x21*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --smb\_set\_limiters 0x21 0x30*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --smb\_set\_cur\_position 0x01 100000*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --smb\_set\_timing 0x01 1000 1000 10 2000*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --smb\_set\_max\_position 0x01 1000000*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --smb\_set\_dst\_position 0x01 90000 0*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --smb\_start 0x21*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --smb\_get\_status 0x21*

As result --smb\_get\_status command will return a set of parameter for two smb0

and smb5 drivers. Since only smb0 was configured to do some steps (configured

current position, destination position) values of smb5 position shall be equal to 0.

Expected result after calling –smb\_status 0x21 command:



# System related commands

# Periphery related commands

## --temperature

Command syntax:

i*cs\_client.exe --dev\_xtpl\_printer IP TCP --digital\_out\_ctrl P0 P1 P2 P3*

* P0: <0, 2> represent subcommand for temperature:
* P0 = 0: read current table temperature
* P0 = 1: set new table temperature pointed by the P1 parameter,
  + P1: <0, 100> desired table temperature: (0°C, 100°C>. For P1 = 0 controller will be switched off.
* P0 = 2: set new PID coefficients from P1, P2, P3 fields:
  + P1: <0, 65535> correspond to proportional (P). Physical value is calculated by dividing input value by 1000 factor.
  + P2: <0, 65535> is integral (I). Physical value is calculated by dividing input value by 10000 factor
  + P3: <0, 65535> is derivative (D). Physical value is calculated by dividing input value by 10000 factor

Positive response syntax:

*0. 0*

*I. OK*

Negative response syntax:

*E. res0*

*I. Peltier ctrl board error: res0*

Where res0 is error code:

* Res0 = 1: temperature sensor connection problem
* Res0 = 255:connection problem between XTPL021 and XTPL023 boards

Example scenario:

* Set PID coefficients to **P**:5, **I**:0,001, **D**:0.0002
* Set new table temperature to 50°C
* Check current table temperature
* Switch off heating/cooling

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --temperature 2 5000 10 2*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --temperature 1 50*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --temperature 0*

*ics\_client.exe --dev\_xtpl\_printer 127.0.0.1 42100 --temperature 1 0*