```
================================================================================
Swiss Server Client Command Protocol

2016-10-26  Peter S'heeren, Axiris
--------------------------------------------------------------------------------


1. Overview
-----------

This document summarizes the client command protocol of the Swiss Server.

The server is capable of accepting many client connections. For each client
connection, the server maintains a proper instance of the communications
protocol.
```

2. Communications Protocol

The communications protocol between server and client is composed of commands and responses. The client sends commands to the server, the server sends responses to the client.

The server returns a response for each command except for "close". When a command is prefixed with "norsp", the server doesn't return a response, even if the command contains errors.

A response may be without command; this is called an unsolicited response.

A subset of commands are executed asynchronously ("in the background") meaning their response comes back some time later, even after the server may have processed subsequent commands.

All commands and responses sent over the connection between client and server are encoded as ASCII characters.

All commands occupy one line that's concluded with an end-of-line marker. The server recognizes LF (10) and CR->LF (13->10).

A command comes with zero or more arguments. The response to a command carries one or more return values. Arguments and values can be numbers, labels, strings.

A number can be written in decimal, hexadecimal, or binary. The first digit must be decimal (0..9). An indicator at the end determines the radix. If no radix is specified, the server assumes a decimal value. The value may contain underscores to improve readability. Numbers are case-insensitive. Examples:

```
  Decimal:        1  10d  1_655_432  255D

  Hexidecimal:    0AFh  1234_eee_h  0AbbaH

  Binary:         1100b  11_0011_0010_B
```

Each number that is returned as part of a response is formatted according to a formatting style. You can change the formatting style with command "vfmts".

If a command argument is invalid or unexpected, the server responds with a failure message. A failure message may also occur on other occasions, like in the case of a hardware I/O error.

If the command is prefixed with an identifier, the response is prefixed with the same identifier. For example:

```
  Command:  id 10 ppw 121
  response: id 10 ppw ok
```

The # character starts a comment. A comment can occur at any point in a line. The server discards all comments.

3. Commands

Group	Command	Description
MCP23017	iop	Configure polling.
	iochg	Configure reporting of changes.
	ior	Read the state of one or all I/O pins.
	iow	Write the output state of one or all I/O pins.
	iod	Set the direction of one or all I/O pins.
	iopu	Control the pull-up resistor of one or all I/O pins.
PCF2129	rtcp	Configure polling.
	rtcchg	Configure reporting of changes.
	rtcr	Read date, time, other state.
	rtcw	Write date and time.
MAX11614	adcp	Configure polling.
	adcchg	Configure reporting of changes.
	adcr	Perform A/D conversion of all channels and report results. The results are also cached.
	adcrc	Read one or more cached results.
PCA9685	pwchg	Configure reporting of changes.
	pcr	Read one or more PWM channels.
	pcw	Write one or more PWM channels.
	ppr	Read the prescale value.
	ppw	Write the prescale value.
PCA9635	pchg	Configure reporting of changes.
	pr	Read one or more PWM channels.
	pw	Write one or more PWM channels.
PCF8574	ep	Configure polling.
	echg	Configure reporting of changes.
	er	Read the state of all I/O pins.
	ew	Write the state of all I/O pins.
RS-485	rser	Read all registers from EEPROM.
	rsew	Write all registers to EEPROM.
	rsrc	Clear all registers.

	rsrr	Read one or more registers.
	rste	Enable the TxD driver for the specified baud rate and word size.
	rstd	Disable the TxD driver.
Serial	serchg	Configure reporting of changes (incoming data).
	sercfg	Configure the serial interface and optionally enable the TxD driver in the RS-485 controller.
	serw	Write data bytes to the serial interface.
	serr	Read data bytes from the serial interface.
Servo PCA9685 required	svme	Enable servo mode.
	svmd	Disable servo mode.
	svcw	Write servo channel.
	svcr	Read servo channel.
	svmv	Initiate a servo movement.
Server	hi	Get hardware information (available components).
	vfmts	Set a specific value formatting style, or all styles at once.
	vfmtg	Get one or all specific value formatting styles.
	ver	Get version information.
	wait	Wait a period before accepting a new command.
	close	Close client connection.
	quit	Quit the server.

3.1. MCP23017 - 16-bit I/O Expander

Boards: Swiss Pi, Swiss Cape

3.1.1. iop

Set polling interval.

Command syntax:

```
  + [LABEL] =iop: Command designator.

  + Polling interval in milliseconds:
    [NUMBER] =0:   Disable polling.
             =1..: Enable polling using this interval.

  + Aim for precise intervals y/n:
    [NUMBER] =0: No. Intervals may be off several milliseconds.
             =1: Yes. This setting uses more processor time.

  + Grant ownership of the polling interval to this client y/n:
    [NUMBER] =0: No.
             =1: Yes. Polling is disabled when the client disconnects from the
                 server.
```

Response syntax:

```
  + [LABEL] =iop: Command designator.

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  iop 500 0 0
  Response: iop ok

    Poll every 500 milliseconds (twice a second), don't aim for precise interval
    and don't grant ownership of the polling interval.

  Command:  iop 0 0 0
  Response: iop ok

    Disable polling.
```

3.1.2. iochg

Configure reporting of changes.

Command syntax:

```
  + [LABEL] =iochg

  + [NUMBER] =0: Disable reporting.
             =1: Enable reporting. The server sends an unsolicited "ior"
                 response for each change in state.
```

Response syntax:

```
  + [LABEL] =iochg

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  iochg 1
  Response: iochg ok
```

3.1.3. ior

Read the state of one or all I/O pins.

Command syntax:

```
  + [LABEL] =ior

  + I/O pin selection:
    / + [LABEL]  =all:   All I/O lines.
    \ + [NUMBER] =0..15: This I/O line.
```

Response syntax:

```
  + [LABEL] =ior

  + - + [LABEL] =all
      + [NUMBER "ior-all-state"] =0..65535: Input state mask.
      + [NUMBER "ior-all-state"] =0..65535: Output state mask.
      + [NUMBER "ior-all-state"] =0..65535: Direction mask.
      + [NUMBER "ior-all-state"] =0..65535: Pull-up enabled mask.

    - + [NUMBER "ior-pin-index"] =0..15
      + [NUMBER "ior-pin-state"] =0..1: Input state.
      + [NUMBER "ior-pin-state"] =0..1: Output state.
      + [NUMBER "ior-pin-state"] =0..1: Direction.
      + [NUMBER "ior-pin-state"] =0..1: Pull-up enabled.

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  ior 4
  Response: ior 04 0 0 1 0

  Command:  ior all
  Response: ior all 0 0 65535 0
```

3.1.4. iow

Write the output state of one or all I/O pins.

Command syntax:

```
  + [LABEL] =iow

  + - + [LABEL]  =all: Write all I/O lines.
      + [NUMBER] =0..65535: Output state mask.

    \ + [NUMBER] =0..15: Write this I/O line.
      + [NUMBER] =0..1: Output state.
```

Response syntax:

```
  + [LABEL] =iow

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  iow 4 1
  Response: iow ok

  Command:  iow all 1111_0000_0000_0000_b
  Response: iow ok
```

3.1.5. iod

Set the direction of one or all I/O pins.

Command syntax:

```
  + [LABEL] =iod

  + - + [LABEL]  =all: Set the direction of all I/O lines.
      + [NUMBER] =0..65535: Direction state mask.

    \ + [NUMBER] =0..15: Set the direction of this I/O line.
      + [NUMBER] =0: Output direction.
                 =1: Input direction.
```

Response syntax:

```
  + [LABEL] =iod

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  iod 4 1
  Response: iod ok

    Set pin 4 as input.

  Command:  iod all 0FFF0h
  Response: iod ok

    Set pins 0-3 as output, pins 4-15 as input.
```

3.1.6. iopu

Control the pull-up resistor of one or all I/O pins.

Command syntax:

```
  + [LABEL] =iopu

  + - + [LABEL]  =all: Set the direction of all I/O lines.
      + [NUMBER] =0..65535: Pull-up state mask.

    \ + [NUMBER] =0..15: Set the direction of this I/O line.
      + [NUMBER] =0: Pull-up Disabled.
                 =1: Pull-up enabled.
```

Response syntax:

```
  + [LABEL] =iopu

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  iopu 4 0
  Response: iopu ok

  Command:  iopu all 100b
  Response: iopu ok
```

3.2. PCF2129A - Real-time Clock

Boards: Swiss Pi, Swiss Cape, AbioCard A, AbioCard B, AbioRTC, AbioWire, PiWire+

3.2.1. rtcp

Set polling interval.

Command syntax:

```
  + [LABEL] =rtcp

  + Polling interval in milliseconds:
    [NUMBER] =0:   Disable polling.
             =1..: Enable polling using this interval.

  + Aim for precise intervals y/n:
    [NUMBER] =0: No. Intervals may be off several milliseconds.
             =1: Yes. This setting uses more processor time.

  + Grant ownership of the polling interval to this client y/n:
    [NUMBER] =0: No.
             =1: Yes. Polling is disabled when the client disconnects from the
                 server.
```

Response syntax:

```
  + [LABEL] =rtcp

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  rtcp 250 0 1
  Response: rtcp ok
```

Poll RTC every 250 milliseconds (4 times per second), don't aim for precise interval, grant ownership of the polling interval to the client.

3.2.2. rtcchg

Configure reporting of changes.

Command syntax:

```
  + [LABEL] =rtcchg

  + [NUMBER] =0: Disable reporting.
             =1: Enable reporting. The server sends an unsolicited "rtcr"
                 response for each change in state.
```

Response syntax:

```
  + [LABEL] =rtcchg

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  rtcchg 1
  Response: rtcchg ok
```

3.2.3. rtcr

Read date, time, other state.

Command syntax:

```
  + [LABEL] =rtcr
```

Response syntax:

```
  + [LABEL] =rtcr

    / + [LABEL]  =fail
      + [STRING] Failure description.
  +

    \ + [NUMBER "rtc-flag"] PU =0..1: Power-up detected y/n.

      + [NUMBER "rtc-flag"] =0..1: Battery low detected y/n.

        / PU=1: Date and time are not available.
      +
        \ PU=0:
            + [NUMBER "rtc-year"]   =2000..2099: Year.
            + [NUMBER "rtc-month"]  =1..12: Month.
            + [NUMBER "rtc-day"]    =1..31: Day.
            + [NUMBER "rtc-hour"]   =0..23: Hour.
            + [NUMBER "rtc-minute"] =0..59: Minute.
            + [NUMBER "rtc-second"] =0..59: Second.
```

Examples:

```
  Command:  rtcr
  Response: rtcr 1 0

  Command:  rtcr
  Response: rtcr 0 0 2016 06 18 22 10 48
```

3.2.4. rtcw

Write date and time.

Command syntax:

```
  + [LABEL]  =rtcw
  + [NUMBER] =2000..2099: Year.
  + [NUMBER] =1..12: Month.
  + [NUMBER] =1..31: Day.
  + [NUMBER] =0..23: Hour.
  + [NUMBER] =0..59: Minute.
  + [NUMBER] =0..59: Second.
```

Response syntax:

```
  + [LABEL] =rtcw

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  rtcw 2015 12 15 18 40 55
  Response: rtcw ok
```

3.3. MAX11614EEE+ - 8-channel Analog-to-Digital Converter

Boards: Swiss Pi, Swiss Cape, AbioCard A, AbioCard B

3.3.1. adcp

Set polling interval.

Command syntax:

```
  + [LABEL] =adcp

  + Polling interval in milliseconds:
    [NUMBER] =0:   Disable polling.
             =1..: Enable polling using this interval.

  + Aim for precise intervals y/n:
    [NUMBER] =0: No. Intervals may be off several milliseconds.
             =1: Yes. This setting uses more processor time.

  + Grant ownership of the polling interval to this client y/n:
    [NUMBER] =0: No.
             =1: Yes. Polling is disabled when the client disconnects from the
                 server.
```

Response syntax:

```
  + [LABEL] =adcp

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  adcp 2000 0 0
  Response: adcp ok

    Perform A/D conversion every 2 seconds.
```

3.3.2. adcchg

Configure reporting of changes.

Command syntax:

```
  + [LABEL] =adcchg

  + [NUMBER] =0: Disable reporting.
             =1: Enable reporting. The server sends an unsolicited "adcr"
                 response for each change in state.
```

Response syntax:

```
  + [LABEL] =adcchg

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  adcchg 1
  Response: adcchg ok
```

3.3.3. adcr

Perform A/D conversion of all channels and report results. The results are also cached.

Command syntax:

```
  + [LABEL] =adcr
```

Response syntax:

```
  + [LABEL] =adcr

  + - + [NUMBER "adc-ch-val"] =0..4095: ADC value channel #0.
      + [NUMBER "adc-ch-val"] =0..4095: ADC value channel #1.
      + [NUMBER "adc-ch-val"] =0..4095: ADC value channel #2.
      + [NUMBER "adc-ch-val"] =0..4095: ADC value channel #3.
      + [NUMBER "adc-ch-val"] =0..4095: ADC value channel #4.
      + [NUMBER "adc-ch-val"] =0..4095: ADC value channel #5.
      + [NUMBER "adc-ch-val"] =0..4095: ADC value channel #6.
      + [NUMBER "adc-ch-val"] =0..4095: ADC value channel #7.

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  adcr
  Response: adcr 1256 0879 4005 1104 0005 0000 1324 0014
```

3.3.4. adcrc

Read one or more cached results. This command reads from the values stored during the last execution of the "adcr" command. The initial values in the cache are zeroes.

Command syntax:

```
  + [LABEL]  =adcrc
  + [NUMBER] =0..7: First channel.
  + [NUMBER] =1..8: Number of channels.
```

Response syntax:

```
  + [LABEL] =adcrc

  + - + [NUMBER "adc-ch-index"]     =0..7: First channel.
      + [NUMBER "adc-ch-index"] CNT =1..8: Number of channels.
      + Array of CNT elements:
          + [NUMBER "adc-ch-val"] =0..4095: ADC value channel.

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  adcrc 2 5
  Response: adcrc 2 5 4005 1104 0005 0000 1324
```

3.4. PCA9685 - 16-channel 12-bit PWM controller

Boards: Swiss Pi, Swiss Cape, AbioCard B

3.4.1. pwchg

Configure reporting of changes.

Command syntax:

```
  + [LABEL] =pwchg

  + [NUMBER] =0: Disable reporting.
             =1: Enable reporting of changes in state:
                 - Unsolicited "pcr" response for changes in the PWM channels.
                 - Unsolicited "ppr" response when the prescale value changes.
```

Response syntax:

```
  + [LABEL] =pwchg

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  pwchg 1
  Response: pwchg ok
```

3.4.2. pcr

Read one or more PWM channels.

The server always reads from cache. This cache is written on the following occasions:
- When the server connects to the PCA9685, it reads all PWM channels.
- When the server executes the "pcw" command successfully, it stores the values in the cache.

You can omit the second or both arguments to form shortened commands.

Command syntax:

```
  + [LABEL]  =pcr
  + [NUMBER] =0..15: First channel.
  + [NUMBER] =1..16: Number of channels.
```

Command syntax, shortened, read one channel:

```
  + [LABEL]  =pcr
  + [NUMBER] =0..15: Channel.
```

Command syntax, shortened, read all channels:

```
  + [LABEL] =pcr
```

Response syntax:

```
  + [LABEL] =pcr

  + - + [NUMBER "pcr-reg-index"]     =0..15: First channel.
      + [NUMBER "pcr-reg-index"] CNT =1..16: Number of channels.
      + Array of CNT elements:
          + [NUMBER "pcr-flag"] =0..1:    Always ON flag.
          + [NUMBER "pcr-pos"]  =0..4095: ON position.
          + [NUMBER "pcr-flag"] =0..1:    Always OFF flag.
          + [NUMBER "pcr-pos"]  =0..4095: OFF position.

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  pcr 0Ah 2
  Response: pcr 10 02 0 0000 1 0000 0 0000 1 0000

    Read channels 10..11.

  Command:  pcr 0Ah
  Response: pcr 10 02 0 0000 1 0000

    Read channel 10.
```

3.4.3. pcw

Write one or more PWM channels. The written values are cached in the server.

You can omit all arguments starting from the third position to form shortened commands. Omitted arguments are substituted with the last specified argument of the same type, or the default value if the type doesn't occur.

Command syntax:

```
  + [LABEL]      =pcw
  + [NUMBER]     =0..15: First channel.
  + [NUMBER] CNT =1..16: Number of channels.
  + Array of CNT elements:
      + [NUMBER] =0..1:    Always ON flag.  (default: 0)
      + [NUMBER] =0..4095: ON position.     (default: 0)
      + [NUMBER] =0..1:    Always OFF flag. (default: 1)
      + [NUMBER] =0..4095: OFF position.    (default: 0)
```

Response syntax:

```
  + [LABEL] =pcw

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  pcw 0 1 0 100 0 200
  Response: pcw ok

    Write (0;100;0;200) to PWM channel 0 (ON position 100, OFF position 200).

  Command:  pcw 1 2  0 100 0 200  0 150 0 250
  Response: pcw ok

    Write (0;100;0;200) to channel 1, (0;150;0;250) to channel 2.

  Command:  pcw 0 3 0 100 0 200
  Response: pcw ok

    Write (0;100;0;200) to PWM channels 0..2.

  Command:  pcw 1 4  0 100 0 200  0 150 0 250  0 200
  Response: pcw ok

    Write (0;100;0;200) to channel 1, (0;150;0;250) to channel 2, (0;200;0;250)
    to channels 3..4.

  Command:  pcw 4 8
  Response: pcw ok

    Write default values (0;0;1;0) to PWM channels 4..11.

  Command:  pcw 1 4  0 100
  Response: pcw ok

    Write (0;100;1;0) to channel 1..4. The latter two values (;;1;0) are default
    values.
```

3.4.4. ppr

Read the prescale value.

Command syntax:

```
  + [LABEL] =ppr
```

Response syntax:

```
  + [LABEL] =ppr

    / + [NUMBER "ppr-val"] =3..255: Prescale register value.
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  ppr
  Response: ppr 030
```

3.4.5. ppw

Write the prescale value.

Command syntax:

```
  + [LABEL]  =ppw
  + [NUMBER] =0..255: Prescale register value. See [1].

  [1] The PCA9685 replaces values 0..2 with 3.
```

Response syntax:

```
  + [LABEL] =ppw

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  ppw 121
  Response: ppw ok
```

3.5. PCA9635 - 16-channel 8-bit PWM controller

Boards: AbioCard A

3.5.1. pchg

Configure reporting of changes.

Command syntax:

```
  + [LABEL] =pchg

  + [NUMBER] =0: Disable reporting.
             =1: Enable reporting. The server sends an unsolicited "pr"
                 response for each change in state.
```

Response syntax:

```
  + [LABEL] =pchg

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  pchg 1
  Response: pchg ok
```

3.5.2. pr

Read one or more PWM channels.

The server always reads from cache. This cache is written on the following occasions:
- When the server connects to the PCA9635, it reads all PWM channels.
- When the server executes the "pw" command successfully, it stores the values in the cache.

You can omit the second or both arguments to form shortened commands.

Command syntax:

```
  + [LABEL]  =pr
  + [NUMBER] =0..16: First channel.
  + [NUMBER] =1..17: Number of channels.
```

Command syntax, shortened, read one channel:

```
  + [LABEL]  =pr
  + [NUMBER] =1..17: Channel.
```

Command syntax, shortened, read all channels:

```
  + [LABEL] =pr
```

Response syntax:

```
  + [LABEL] =pr

  + - + [NUMBER "pr-reg-index"]     =0..16: First channel.
      + [NUMBER "pr-reg-index"] CNT =1..17: Number of channels.
      + Array of CNT elements:
          + [NUMBER "pr-reg-val"] =0..255

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  pr 4 5
  Response: pr 04 05 000 000 000 000 000

    Read channels 4..8.

  Command:  pr 8
  Response: pr 08 01 000

    Read channel 8.
```

3.5.3. pw

Write one or more PWM channels. The written values are cached in the server.

You can omit all arguments starting from the third position to form shortened commands. Omitted arguments are substituted with the last specified argument, or the default value no previous argument was specified.

Command syntax:

```
  + [LABEL]      =pw
  + [NUMBER]     =0..16: First channel.
  + [NUMBER] CNT =1..17: Number of channels.
  + Array of CNT elements:
      + [NUMBER] =0..255.  (default: 0)
```

Response syntax:

```
  + [LABEL] =pw

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  pw 4 5  80 82 84 86 88
  Response: pw ok

    Write the specified to channels 4..8.

  Command:  pw 4 5
  Response: pw ok

    Write 0 to channels 4..8.

  Command:  pw 4 5  80 82
  Response: pw ok

    Write 80 to channel 4, write 82 to channels 5..8.
```

3.6. PCF8574 - 8-bit I/O Expander

Boards: AbioCard A, AbioCard B

3.6.1. ep

Set polling interval.

Command syntax:

```
  + [LABEL] =ep

  + Polling interval in milliseconds:
    [NUMBER] =0:   Disable polling.
             =1..: Enable polling using this interval.

  + Aim for precise intervals y/n:
    [NUMBER] =0: No. Intervals may be off several milliseconds.
             =1: Yes. This setting uses more processor time.

  + Grant ownership of the polling interval to this client y/n:
    [NUMBER] =0: No.
             =1: Yes. Polling is disabled when the client disconnects from the
                 server.
```

Response syntax:

```
  + [LABEL] =ep

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  ep 250 1 0
  Response: ep ok

    Poll every 250 milliseconds (4 times per second), aim for precise intervals.
```

3.6.2. echg

Configure reporting of changes.

Command syntax:

```
  + [LABEL] =echg

  + [NUMBER] =0: Disable reporting.
             =1: Enable reporting. The server sends an unsolicited "er"
                 response for each change in state.
```

Response syntax:

```
  + [LABEL] =echg

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  echg 1
  Response: echg ok
```

3.6.3. er

Read the state of all I/O pins.

Command syntax:

```
  + [LABEL] =er
```

Response syntax:

```
  + [LABEL] =er

  + - + [NUMBER "er-reg-val"] =0..255: State mask.

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  er
  Response: er 11001010b
```

3.6.4. ew

Write the state of all I/O pins.

Command syntax:

```
  + [LABEL]  =ew
  + [NUMBER] =0..255: State mask.
```

Response syntax:

```
  + [LABEL] =ew

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  ew 00101101b
  Response: ew ok
```

3.7. RS-485 Controller

This controller controls the direction of the half-duplex RS-485 transceiver chip. When the host sends a data word over the TxD line, the controller sets the transceiver in the TxD direction for the duration of the data word.

Boards: Swiss Pi, Swiss Cape

3.7.1. rser

Read all registers from EEPROM.

Command syntax:

```
  + [LABEL] =rser
```

Response syntax:

```
  + [LABEL] =rser

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  rser
  Response: rser ok
```

3.7.2. rsew

Write all registers to EEPROM.

Command syntax:

```
  + [LABEL] =rsew
```

Response syntax:

```
  + [LABEL] =rsew

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  rsew
  Response: rsew ok
```

3.7.3. rsrc

Clear all registers.

Command syntax:

```
  + [LABEL] =rsrc
```

Response syntax:

```
  + [LABEL] =rsrc

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  rsrc
  Response: rsrc ok
```

3.7.4. rsrr

Read one or more registers.

Command syntax:

```
  + [LABEL]  =rsrr
  + [NUMBER] =0..6: First register.
  + [NUMBER] =1..7: Number of registers.
```

Response syntax:

```
  + [LABEL] =rsrr

  + - + [NUMBER "rsrr-reg-index"]     =0..6: First register.
      + [NUMBER "rsrr-reg-index"] CNT =1..7: Number of registers.
      + Array of CNT elements:
          + [NUMBER "rsrr-reg-val"] =0..255

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  rsrr 0 7
  Response: rsrr 0 7 054h 001h 000h 000h 001h 000h 000h
```

3.7.5. rste

Enable the TxD driver for the specified baud rate and word size.

Command syntax:

```
  + [LABEL]  =rste
  + [NUMBER] =1..:   Baud rate.
  + [NUMBER] =1..20: Word bits. Sum of start bit, data bits, stop bits, parity
                     bit.
```

Response syntax:

```
  + [LABEL] =rste

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  rste 57600 10
  Response: rste ok

    57600 baud, 1 start bit, 8 data bits, 1 stop bit, no parity bit.
```

3.7.6. rstd

Disable the TxD driver.

Command syntax:

```
  + [LABEL] =rstd
```

Response syntax:

```
  + [LABEL] =rstd

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  rstd
  Response: rstd ok
```

3.8. Serial Interface

Boards: Swiss Pi, Swiss Cape

3.8.1. serchg

Configure reporting of changes (incoming data).

Command syntax:

```
  + [LABEL] =serchg

  + [NUMBER] =0: Disable reporting.
             =1: Enable reporting. The server sends an unsolicited "serr"
                 response when it has received data from the serial interface.
```

Response syntax:

```
  + [LABEL] =serchg

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  serchg 1
  Response: serchg ok
```

3.8.2. sercfg

Configure the serial interface and optionally enable the TxD driver in the RS-485 controller.

You can omit all arguments starting from data bits to form shortened commands. The server substitutes omitted arguments with their default value.

Command syntax:

```
  + [LABEL] =sercfg

  + Baud rate:
    [NUMBER] =1..

  + Data bits:
    [NUMBER] =1..16  (default: 8)

  + Stop bits:
    [NUMBER] =1..2  (default: 1)

  + Parity:
    [LABEL]  =none  (default)
             =even
             =odd

  + Set baud rate and enable TxD driver in RS-485 controller:
    [NUMBER] =0: No.  (default)
             =1: Yes.
```

Response syntax:

```
  + [LABEL] =sercfg

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  sercfg 57600 8 1 none 1
  Response: serchg ok

  Command:  sercfg 9600
  Response: serchg ok

  Command:  sercfg 115200 7
  Response: serchg ok
```

3.8.3. serw

Write data bytes to the serial interface.

Command syntax:

```
  + [LABEL] =serw

  + List of zero or more tokens:
      + [NUMBER] =0..255
      + [STRING] Text
```

Response syntax:

```
  + [LABEL] =serw

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  serw "ABCD"
  Response: serw ok

  Command:  serw 13 10 "A line of text" 13 10
  Response: serw ok
```

3.8.4. serr

Read data bytes from the serial interface.

You can omit the maximum number of bytes argument.

Command syntax:

```
  + [LABEL] =serr

  + Maximum number of bytes:
    [NUMBER] =0:   No maximum.  (default)
             =1..: Read this amount of data bytes at most.
```

Response syntax:

```
  + [LABEL] =serr

  + - + Array of received data bytes:
          + [NUMBER "serr-word"] =0..255

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  serr 40
  Response: serr

  Command:  serr
  Response: serr 065 066 067 068
```

3.9. Servo Mode

Boards: Swiss Pi, Swiss Cape, AbioCard B

3.9.1. svme

Enable servo mode.

The server starts executing pending "svmv" commands.

Command syntax:

```
  + [LABEL] =svme
```

Response syntax:

```
  + [LABEL] =svme

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  svme
  Response: svme ok
```

3.9.2. svmd

Disable servo mode.

Command syntax:

```
  + [LABEL] =svmd
```

Response syntax:

```
  + [LABEL] =svmd

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  svme
  Response: svme ok
```

3.9.3. svcw

Write servo channel.

Command syntax:

```
  + [LABEL] =svcw

  + Channel:
    [NUMBER] =0..15

  + Min. position as PWM ON time:
    [NUMBER] =0..4095: See [1].

  + Max. position as PWM ON time:
    [NUMBER] =0..4095: See [1].

  + Steps:
    [NUMBER] =0:        Don't use steps. Use PWM ON time to set position.
             =1..65535: The servo position ranges from 0 to this value.

  [1] When both values are zero, the channel is disabled. To enable, set values
      where min < max position.
```

Response syntax:

```
  + [LABEL] =svcw

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  svcw 0 118 515 1000
  Response: svcw ok

    Enable channel #0. ON time range 118..515. Set motor position with step
    values 0..1000 in command "svmv".

  Command:  svcw 1 125 500 0
  Response: svcw ok

    Enable channel #1. ON time range 125..500, No steps. Set motor position
    with ON time values 125..500 in command "svmv".

  Command:  svcw 2 0 0 0
  Response: svcw ok

    Disable channel #2.
```

3.9.4. svcr

Read servo channel.

Command syntax:

```
  + [LABEL] =svcr

  + Channel:
    [NUMBER] =0..15
```

Response syntax:

```
  + [LABEL] =svcr

  + - + Channel:
        [NUMBER] =0..15

      + Min. position as PWM ON time:
        [NUMBER] =0..4095

      + Max. position as PWM ON time:
        [NUMBER] =0..4095

      + Steps:
        [NUMBER] =0:        Don't use steps. Use PWM ON time to set position.
                 =1..65535: The servo position ranges from 0 to this value.

      + Current position as PWM ON time:
        [NUMBER] =0..4095: See [1].

      + Current step:
        [NUMBER] =0..: See [1].

      + Scheduled commands:
        [NUMBER] =0..

    \ + [LABEL]  =fail
      + [STRING] Failure description.

  [1] These values are meaningful (valid) from the moment when the server has
      begun executing the first "svmv" command for this channel.
```

Example:

```
  Command:  svcr 1
  Response: svcr 01 0118 0515 1000 0118 0000 0

  Command:  svcr 2
  Response: svcr 02 0000 0000 0000 0000 0000 0
```

3.9.5. svmv

Initiate a servo movement.

The command's effect depends on the type of servo motor:

* Positional rotation servo: the command changes the position of the servo motor.

* Continuous rotation servo: the command changes the speed of the servo motor. As such, it accelerates or deaccelerates the servo motor.

If non-zero, the delay specifies the point in time when the command must start. This point in time is defined as the previous starting point plus the delay. The starting point is either:
(a) the start of the previous command for this channel;
(b) the moment when servo mode was enabled if this is the first command since.

The delay is taken literally, meaning:
(a) If the command is sent on time, it'll be executed normally.
(b) If the command is sent too late, then:
    (b1) if the command specifies no period, it'll be executed normally.
    (b2) if the command specifies a period, it'll be executed for the remainder of the period.

As a result, you can send a sequence of commands with non-zero period and non-zero delay values, the result will be a well-timed sequence of movements.

Command syntax:

```
  + [LABEL] =svmv

  + Channel:
    [NUMBER] =0..15

  + Destination:
    [NUMBER] =0..: Either a step value or PWM ON time.

  + Period in milliseconds to move motor to destination position:
    [NUMBER] =0:        Immediately.
             =1..60000: Move over this period of time.

  + Delay in milliseconds before starting to move motor:
    [NUMBER] =0:        Immediately, no delay.
             =1..60000: Wait this period of time.
```

Response syntax:

```
  + [LABEL] =svmv

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

```
Example:

  Command:  svmv 0 600 0 0
  Response: svmv ok

    Move channel #0 immediately to step or PWM ON time 600, no delay.

  Command:  svmv 1 250 500 5000
  Response: svmv ok

    Move channel #1 to step or PWM ON time 500 over a period of 500 ms. Start
    the command after 5000 ms delay.
```

3.10. Server

3.10.1. hi

Get hardware information (available components).

Command syntax:

```
  + [LABEL] =hi
```

Response syntax:

```
  + [LABEL] =hi

  + - + [NUMBER] =0..1: Serial interface present y/n.
      + [NUMBER] =0..1: RS-485 controller present y/n.
      + [NUMBER] =0..1: Real-time clock present y/n.
      + [NUMBER] =0..1: 16-bit I/O expander present y/n.
      + [NUMBER] =0..1: A/D converter present y/n.
      + [NUMBER] =0..1: 16-channel 12-bit PWM controller present y/n.
      + [NUMBER] =0..1: 16-channel 8-bit PWM controller present y/n.
      + [NUMBER] =0..1: 8-bit I/O expander present y/n.

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  hi
  Response: hi 1 1 1 1 1 0 0
```

3.10.2. vfmts

Set a specific value formatting style, or all styles at once.

Command syntax:

```
  + [LABEL]  =vfmts

  + [STRING] Designator. "*" means all.

  + Radix:
    [LABEL]  =dec: Decimal.
             =hex: Hexadecimal.
             =bin: Binary.

  + Maximum number of digits:
    [NUMBER] =0:     No maximum, format number of digits as needed.
             =1..64: Format this number of digits at most.

  + Append radix character y/n:
    [NUMBER] =0: No.
             =1: Yes.

  + Leading zeroes y/n:
    [NUMBER] =0: No.
             =1: Yes.

  + Start with decimal digit y/n:
    [NUMBER] =0: No.
             =1: Yes.

  + Uppercase digits y/n:
    [NUMBER] =0: No.
             =1: Yes.

  + Uppercase radix character y/n:
    [NUMBER] =0: No.
             =1: Yes.
```

Response syntax:

```
  + [LABEL] =vfmts

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  vfmts "adc-ch-val" hex 4 1 1 1 1 0
  Response: vfmts ok

    Set formatting style of a specific value.

  Command:  vfmts "*" dec 0 0 0 0 0 0
  Response: vfmts ok

    Set one formatting style for all values.
```

3.10.3. vfmtg

Get one or all value formatting styles.

Command syntax:

```
  + [LABEL]  =vfmtg
  + [STRING] Designator. "*" means all and will produce a response for each
             formatting style.
```

Response syntax:

```
  + [LABEL] =vfmtg

  + - + Radix:
        [LABEL]  =dec: Decimal.
                 =hex: Hexadecimal.
                 =bin: Binary.

      + Maximum number of digits:
        [NUMBER] =0:     No maximum, format number of digits as needed.
                 =1..64: Format this number of digits at most.

      + Append radix character y/n:
        [NUMBER] =0: No.
                 =1: Yes.

      + Leading zeroes y/n:
        [NUMBER] =0: No.
                 =1: Yes.

      + Start with decimal digit y/n:
        [NUMBER] =0: No.
                 =1: Yes.

      + Uppercase digits y/n:
        [NUMBER] =0: No.
                 =1: Yes.

      + Uppercase radix character y/n:
        [NUMBER] =0: No.
                 =1: Yes.

    - + [LABEL] =ok

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  vfmtg "adc-ch-val"
  Response: vfmtg "adc-ch-val" hex 4 1 1 1 1 0
  Response: vfmtg ok
```

3.10.4. ver

Get version information.

Command syntax:

```
  + [LABEL] =ver
```

Response syntax:

```
  + [LABEL] =ver

  + - + [STRING] Version of the server.

    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  ver
  Response: ver "1.0.0"
```

3.10.5. wait

Wait a specified period before accepting a new command.

This command only affects processing of commands for the client connection, not any other client connection.

Command syntax:

```
  + [LABEL]  =wait
  + [NUMBER] =1..: Number of milliseconds to wait.
```

Response syntax:

```
  + [LABEL] =wait

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Examples:

```
  Command:  wait 5000
  Response: wait ok
```

3.10.6. close

Close client connection.

This command doesn't generate a response.

Command syntax:

```
  + [LABEL] =quit
```

Example:

```
  Command: close
```

3.10.7. quit

Quit the server.

Command syntax:

```
  + [LABEL] =quit
```

Response syntax:

```
  + [LABEL] =quit

    / + [LABEL] =ok
  +
    \ + [LABEL]  =fail
      + [STRING] Failure description.
```

Example:

```
  Command:  quit
  Response: quit ok
```

```
4. History
----------

2016-06-24  Peter S'heeren, Axiris

  * First release.

2016-09-03  Peter S'heeren, Axiris

  * Second release.

2016-10-26  Peter S'heeren, Axiris

  * Some clarifications in syntax description of iod and iopu.
  * Third release.
```