AxiCat Server Client Command Protocol

2016-11-06 Peter S'heeren, Axiris

1. Overview

This document summarizes the client command protocol of the AxiCat 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
Hexadecimal:	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 ior 4
response: id 10 ior 04 0 0 out
```

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
GPIO	ior	Read the state of a GPIO pin.
	iow	Write the output state of a GPIO pin.
	iod	Set the direction a GPIO pin.
I2C master	imss	Set speed of I2C bus, pick from predefined values.
	imsr	Set speed of I2C bus, raw register values.
	ime	Enable I2C master (join I2C bus as master).
	imd	Disable I2C master.
	imw	Initiate SLA+W transfer.
	imr	Initiate SLA+R transfer.
	imc	Cancel one or all I2C master transfers.
I2C slave	ise	Enable I2C slave (join I2C bus as slave).
	isd	Disable I2C slave.
	isw	Initiate write transfer to respond to SLA+R transfer.
	iswc	Cancel a specific or all write transfers.
	isr	Initiate read transfer to respond to SLA+W transfer.
	isrc	Cancel a specific or all read transfers.
SPI master	smss	Set speed of SPI bus, pick from predefined values.
	smsr	Set speed of SPI bus, raw register values.
	smsc	Set configuration.
	sme	Enable SPI master.
	smd	Disable SPI master.
	smt	Initiate SPI transfer.
	smc	Cancel a specific or all SPI transfers.
1-Wire master	ome	Enable 1-Wire master.
	omd	Disable 1-Wire master.
	omr	Initiate 1-Wire reset command.
	omt	Initiate 1-Wire touch bytes command.
	omb	Initiate 1-Wire touch bits command.
	omnf	Initiate enumerate first command.

	omnn	Initiate enumerate next command.
	omp	Initiate probe command.
	omc	Cancel a specific or all 1-Wire commands.
UART #0	u0sb	Set baudrate, pick from predefined values.
	u0sr	Set baudrate, raw register values.
	u0sd	Set number of data bits.
	u0ss	Set number of stop bits.
	u0st	Set Rx timeout.
	u0su	Set unsolicited Rx responses.
	u0e	Enable UART.
	u0d	Disable UART.
	u0w	Write data.
	u0r	Read data.
UART #1	u1sb	Set baudrate, pick from predefined values.
	u1sr	Set baudrate, raw register values.
	u1sd	Set number of data bits.
	u1ss	Set number of stop bits.
	u1st	Set Rx timeout.
	u1su	Set unsolicited Rx responses.
	u1e	Enable UART.
	u1d	Disable UART.
	u1w	Write data.
	u1r	Read data.
Server	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.

Commands are case-insensitive, i.e. "ior" and "IoR" are the same to the server.
The server always returns a lowercase response mnemonic.

The following sections describe the syntax of commands and responses. The syntax is presented as a simple flowchart that resolves to a stream of tokens. Possible tokens are numbers, strings, and labels. See above for more details.

The tokens are presented as follows:

```
Label:  + [LABEL]

String: + [STRING]

Number: + [NUMBER]
```

If a number is subject to a value formatting style, the token is presented as follows:

```
+ [NUMBER "..."]
```

where "..." is the identifier of the formatting style.

Tokens are appended with one or more equal signs with possible contents. For example:

```
Number: + [NUMBER] =0: Zero.
                   =1..: Non-zero.
```

Since this document is an ascii text file, the possibilities for displaying a flowchart are limited. The text uses the following conventions:

* A plus character designates to next token in the stream. Consecutive plus characters positioned in the same column (thus aligned vertically) link together consecutive tokens. For example:

  ```
  + [LABEL]  =ior
  + [NUMBER] =0..16: GPIO pin.
  ```

  This means "ior" followed by a value ranging from 0 to 16.

* A group of minus, slash and/or backslash characters positioned in the same column (thus aligned vertically) with a common plus character as ancestor: this construct denotes the spawning of multiple branches from a single point in the flowchart. A dot indicate an empty branch. For example:

  ```
  + [LABEL] =abc

  + - .

    - + [LABEL] =empty

    - + [NUMBER] =1..5

    \ + [LABEL] =tell
      + [STRING] Your description.
  ```

  Some valid token streams for this flowchart are:

  ```
  abc
  abc empty
  abc 5
  abc tell "something's happening"
  ```

* An array construct is denoted as in the following example:

  ```
  + Array of 0..65535 elements:
      + [NUMBER "imr-payload"] =0..255: Payload data byte.
  ```

```
  * Some parts of a flowchart are spun off and described in a separate section.
    A reference in made using curly brackets. For example:

    + {DATA-BYTES} Payload of 1..256 data bytes.

  * An arrow pointing to a label is used for branching to a specific point. For
    example, a series of data bytes:

    + P1

    + - + [NUMBER] =0..255
        + -> P1
      \ .

  * Description: sometimes it's simpler to describe (part of) a flowchart in
    plain text.

Each endpoint in the flowchart implies an end-of-line marker.
```

3.1. GPIO

Asynchronous transfers:

Command	Transfer Type	Result Types
ior	GPIO Read State	success cancel

Result	Description
success	The transfer has completed in the AxiCat. The response reports the outcome of the operation.
cancel	The transfer was cancelled before it was sent down to the AxiCat.

3.1.1. ior

Read the state of the specified GPIO pin.

Command syntax:

```
  + [LABEL]  =ior
  + [NUMBER] =0..16: GPIO pin.
```

Response syntax:

```
  + [LABEL] =ior

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

    \ + [NUMBER "ior-pin-index"] =0..16: GPIO pin.

      + - + [NUMBER "ior-pin-state"] =0..1: Input state.
          + [NUMBER "ior-pin-state"] =0..1: Output state.
          + [LABEL] =in:  Input direction.
                    =out: Output direction.

        - + [LABEL] =cancel
```

Example:

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

3.1.4. iow

Write the output state of the specified GPIO pin.

Command syntax:

```
  + [LABEL]  =iow
  + [NUMBER] =0..16: GPIO pin.
  + [NUMBER] =0..1: Output state.
```

Response syntax:

```
  + [LABEL] =iow

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

Example:

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

3.1.5. iod

Set the direction of the specified GPIO pin.

Command syntax:

```
  + [LABEL] =iod
  + [NUMBER] =0..16: GPIO pin.
    / + [NUMBER] =0: Output direction.
  +              =1: Input direction.
    \ + [STRING] =out: Output direction.
                 =in:  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.
```

3.2. I2C Master

3.2.1. imss

Set the speed of the I2C bus. The given value must match one of the predefined bus speeds defined in the AxiCat protocol (see axicat.h, AXICAT_TWI_SPEED_<n>).

Command syntax:

```
  + [LABEL] =imss

  + Bus speed (Hz):
    [NUMBER] =50000
             =100000
             =200000
             =400000
```

Response syntax:

```
  + [LABEL] =imss

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

Example:

```
  Command:  imss 50000
  Response: imss ok
```

3.2.2. imsr

Directly write to the speed registers of the I2C function (a.k.a. setting raw register values).

Refer to section “Two-wire Serial Interface” in the Atmega164A/324A/644A/1284 datasheet for more information.

Command syntax:

```
  + [LABEL] =imsr
  + [NUMBER] =0..255: TWBR register.
  + [NUMBER] =0..3: TWPS register.
```

Response syntax:

```
  + [LABEL] =imsr

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

Example:

```
  Command:  imsr 3Ah 1
  Response: imsr ok

    Set the bit rate register to 3Ah (58) and the prescaler register to 1,
    resulting in a bus speed of 25000 Hz.

  Command:  imsr 0 0
  Response: imsr ok

    Set the bit rate register to 00h and the prescaler register to 0, resulting
    in a bus speed of 750000 Hz. This is the maximum I2C speed.
```

3.2.3. ime

This command enables the I2C (TWI) master of the AxiCat.

Note that you can schedule transfers when the I2C master is disabled. They'll be executed as soon as the enable command has been processed.

Command syntax:

```
  + [LABEL] =ime
```

Response syntax:

```
  + [LABEL] =ime

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

Example:

```
  Command:  ime
  Response: ime ok
```

3.2.4. imd

This command disables the I2C (TWI) master of the AxiCat.

Command syntax:

```
  + [LABEL] =imd
```

Response syntax:

```
  + [LABEL] =imd

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

Example:

```
  Command:  imd
  Response: imd ok
```

All transfers being processed in the AxiCat will be completed. However, one or more transfers that are still scheduled in the application layer will be initiated in the AxiCat. The AxiCat will merely buffer these transfers, but for the application layer the transfers have been started.

If your intention is to cancel all master transfers and disable the I2C function, issue the following commands:

```
  Command:  imc
  Command:  imd
```

If you want to cancel all slave transfers as well:

```
  Command:  imc
  Command:  iscw
  Command:  iscr
  Command:  imd
```

The order is important. First request cancellation, then disable the I2C function. If you do it the other way around, the application layer may send one or more scheduled transfers to the AxiCat between "imd" and "imc", "iscw", "iscr".

3.2.4. I2C Master Transfers

Asynchronous transfers:

Command	Transfer Type	Result Types
imw	I2C Master SLA+W	success cancel skip bus arb
imr	I2C Master SLA+R	success cancel skip bus arb

Result	Description
success	The transfer has completed in the AxiCat. The response reports the outcome of the bus transaction.
cancel	The transfer was cancelled before it was sent down to the AxiCat.
skip	The transfer was sent down to the AxiCat. The AxiCat skipped the execution of the transfer.
bus	A bus error occured during the bus transaction.
arb	The master lost arbitration during the bus transaction.

```
3.2.4.1. imw
------------

Initiate an SLA+W transfer.

An SLA+W transfer without data payload is called a probe command. It's used for
detecting the presence of an I2C slave.

Command syntax:

  + [LABEL] =imw

  + [NUMBER] =0..127: Slave address.

  + {PAYLOAD-BYTES: CNT=0..65535} Payload of 0..65535 data bytes.

    / + [LABEL] =stop: Force STOP signal at the end of the transfer.
  + - + [LABEL] =rep: Force REPEATED START signal at the end of the transfer.
    \ . Let server choose between STOP and REPEATED START signal.

Response syntax:

  + [LABEL] =imw

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

    \ + [NUMBER "imw-slave-ad"] =0..127: Slave address.

      + - + [NUMBER "imw-xfrd"] =0..65535: Number of bytes transferred.

          + Response of the slave device. If no bytes were transferred, it's
            the slave's response to the SLA+W, else it's the slave's response
            to the last byte written by the master.
            [LABEL] =ack:  ACK response.
                    =nack: NACK response.

        - + [LABEL] =cancel

        - + [LABEL] =skip

        - + [LABEL] =bus

        - + [LABEL] =arb

Examples:

  Command:  imw 51h
  Response: imw 081 00000 nack

    Probing of I2C slave. The slave with the specified address doesn't respond
    so it's deemed not present on the I2C bus.

  Command:  imw 50h
  Response: imw 080 00000 ack

    The I2C slave responds with ACK, so it's present on the I2C bus.

  Command:  imw 80 41h 42h 43h 44h
  Response: imw 080 00004 ack

    Successful transmission of data bytes from AxiCat to I2C slave.
```

```
Command:  imw 80 "ABCD"
Response: imw 080 00002 nack
```

Partial transmission of data bytes from AxiCat to I2C slave.

```
Command:  imw 80 41h 42h "E" 65h
Response: imw 080 00000 nack
```

Failed transmission of data bytes to I2C slave. The slave may be absent, or it may respond with NACK deliberately.

```
3.2.4.2. imr
------------

Initiate an SLA+R transfer.

Note that the acknowlegde response is virtually always NACK.

Command syntax:

  + [LABEL] =imr

  + [NUMBER] =0..127: Slave address.

  + [NUMBER] =1..65535: Number of bytes to read.

    / + [LABEL] =stop: Force STOP signal at the end of the transfer.
  + - + [LABEL] =rep: Force REPEATED START signal at the end of the transfer.
    \ . Let server choose between STOP and REPEATED START signal.

Response syntax:

  + [LABEL] =imr

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

    \ + [NUMBER "imr-slave-ad"] =0..127: Slave address.

      + - + Array of 0..65535 elements:
              + [NUMBER "imr-payload"] =0..255: Payload data byte.

          + Last acknowledge response from either master or slave.
            [LABEL] =ack:  ACK response.
                    =nack: NACK response.

        - + [LABEL] =cancel

        - + [LABEL] =skip

        - + [LABEL] =bus

        - + [LABEL] =arb

Examples:

  Command:  imr 80 5
  Response: imr 080 nack

    Trying to read 5 bytes. The master received a NACK response while
    transmitting the slave address meaning the slave isn't present or
    deliberately returns a NACK response.

  Command:  imr 80 5
  Response: imr 080 146 008 004 146 008 nack

    The master reads 5 bytes from the slave. The master sent a NACK response
    after receiving the last data byte.
```

```
Command:  imr 80 5
Response: imr 080 146 ack
```

A partial data transfer has occured. The master sent an ACK response after receiving the second data byte. The transfer was aborted. The latter may happen if the I2C master is disabled during the execution of the I2C transfer.

3.2.4.3. imc

Request cancellation of a specific or all I2C master transfers.

Command syntax:

```
  + [LABEL] =imc

    / + [NUMBER] Command identifier of transfer to cancel.
  + - + [LABEL] =all: Cancel all transfers.
    \ . Cancel all transfers.
```

Response syntax:

```
  + [LABEL] =imc

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

Examples:

```
  Command:  imc
  Response: imc ok

  Command:  imc all
  Response: imc ok

  Command:  imc 40
  Response: imc ok
```

3.2.4.4. STOP and REPEATED START signals

If "rep" is specified, the server withholds scheduling of the transfer. As soon as a subsequent "imw" or "imr" command without "rep" comes in, the server will schedule all withheld transfers. Each transfers marked with "rep" will induce a REPEATED START signal for the following transfer. This way, the client can commit a list of transfers that the I2C master will execute without releasing the I2C bus.

If neither "rep" nor "stop" is specified, the AxiCat application layer will decide whether a STOP signal or REPEATED START signal will be generated at the end of the I2C transfer. If this is the last scheduled I2C transfer, the application layer concludes with a STOP signal, else it generates a REPEATED START signal and issues the next I2C transfer.

Example:

```
Command:  imw 81 2Ah rep
Command:  imr 81 2 rep
Command:  imw 81 00h rep
Command:  imr 81 12 stop
Response: imw 081 00001 ack
Response: imr 081 128 128 nack
Response: imw 081 00001 ack
Response: imr 081 128 128 000 003 000 000 000 000 000 000 000 000 nack
```

Initiate various I2C transfers in one block. When the server receives the last command, it schedules all four transfers at once. As a result, the AxiCat generates a REPEATED START signal between I2C transfers. The fourth transfer is explicitly concluded with a STOP signal.

Note that in case the I2C master looses arbitration, all remaining I2C transfers in the block will be skipped.

3.3. I2C Slave

3.3.1. ise

Enable the I2C slave function.

The I2C master must be enabled in order to enable the I2C slave.

When the I2C slave is enabled, it will respond to SLA+W and SLA+R packets carrying the specified slave address. When the slave is addressed, either a slave Tx transfer or a slave Rx transfer must have been scheduled to let the master continue the bus transaction. If the required transfer isn't scheduled, the I2C slave will lock up the I2C bus until the Slave Tx or Rx transfer is scheduled.

Note that you can schedule transfers when the I2C slave is disabled. They'll be executed as soon as the enable command has been processed.

Command syntax:

```
  + [LABEL] =ise

  + [NUMBER] =0..127: Slave address.

  + - + [LABEL] =gca: General call address is enabled.
    \ . General call address is disabled.
```

Response syntax:

```
  + [LABEL] =ise

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

Examples:

```
  Command:  ise 0A8h
  Response: ise ok

  Command:  ise 48h gca
  Response: ise ok
```

3.3.2. isd

Disable the I2C slave function.

Command syntax:

```
  + [LABEL] =isd
```

Response syntax:

```
  + [LABEL] =isd

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

Example:

```
  Command:  isd
  Response: isd ok
```

3.3.3. I2C Slave Write Transfers

Asynchronous transfers:

Command	Transfer Type	Result Types
isw	I2C Slave Tx	success cancel skip bus

Result	Description
success	The transfer has completed in the AxiCat. The response reports the outcome of the bus transaction.
cancel	The transfer was cancelled before it was sent down to the AxiCat.
skip	The transfer was sent down to the AxiCat. The AxiCat skipped the execution of the transfer.
bus	A bus error occured during the bus transaction.

```
3.3.3.1. isw
------------

Schedule a slave Tx transfer. When an I2C master addresses the slave with
SLA+R, it can read the given data bytes from the slave.

You can split up the data in chunks that form one logical data payload. This
features is useful when the slave can't produce all data bytes in time. When
"more" is specified and the I2C master has read data from the slave, the slave will
lock up the I2C bus until another slave Tx transfer is scheduled. Thus it's
important to schedule a slave Tx transfer without "more" at some point. In many
cases, you even don't have to split up the data payload so you'd never specify
"more".

Command syntax:

  + [LABEL] =isw

  + {PAYLOAD-BYTES: CNT=1..65535} Payload of 1..65535 data bytes.

  + - + [LABEL] =more
    \ .

Response syntax:

  + [LABEL] =isw

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

    \ + - + [NUMBER "isw-xfrd"] =0..65535: Number of bytes transferred.

          + Last response from I2C master:
            [LABEL] =ack:  ACK response.
                    =nack: NACK response.

        - + [LABEL] =cancel

        - + [LABEL] =skip

        - + [LABEL] =bus

Examples:

  Command:  isw "ABCD"
  Response: isw 00004 ack
```

3.3.3.2. iswc

Request cancellation of a specific or all slave Tx transfers.

Command syntax:

```
  + [LABEL] =iswc

    / + [NUMBER] Command identifier of transfer to cancel.
  + - + [LABEL] =all: Cancel all transfers.
    \ . Cancel all transfers.
```

Response syntax:

```
  + [LABEL] =iswc

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

Examples:

```
  Command:  iswc
  Response: iswc ok

  Command:  iswc all
  Response: iswc ok

  Command:  iswc 40
  Response: iswc ok
```

3.3.4. I2C Slave Read Transfers

Asynchronous transfers:

Command	Transfer Type	Result Types
isr	I2C Slave Rx	success cancel skip bus

Result	Description
success	The transfer has completed in the AxiCat. The response reports the outcome of the bus transaction.
cancel	The transfer was cancelled before it was sent down to the AxiCat.
skip	The transfer was sent down to the AxiCat. The AxiCat skipped the execution of the transfer.
bus	A bus error occured during the bus transaction.

```
3.3.4.1. isr
------------

Schedule a slave Rx transfer. When an I2C master addresses the slave with
SLA+W, it can write data bytes to the slave.

You can split up the data in chunks that form one logical data payload. This
features is useful when the slave can't handle all data bytes in time. When
"more" is specified and the I2C master has written data to the slave, the slave
will lock up the I2C bus until another slave Rx transfer is scheduled. Thus
it's important to schedule a slave Rx transfer without "more" at some point.

Command syntax:

  + [LABEL] =isr

  + [NUMBER] =1..65535: Number of bytes to read.

  + - + [LABEL] =more
    \ .

Response syntax:

  + [LABEL] =isr

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

    \ + - + Array of 1..65535 elements:
              + [NUMBER "isr-payload"] =0..255: Payload data byte.

        - + [LABEL] =cancel

        - + [LABEL] =skip

        - + [LABEL] =bus

Examples:

  Command:  isr 4
  Response: isr 001 000 025 240
```

3.3.4.2. isrc

Request cancellation of a specific or all slave Rx transfers.

Command syntax:

```
  + [LABEL] =isrc

    / + [NUMBER] Command identifier of transfer to cancel.
  + - + [LABEL] =all: Cancel all transfers.
    \ . Cancel all transfers.
```

Response syntax:

```
  + [LABEL] =isrc

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

Examples:

```
  Command:  isrc
  Response: isrc ok

  Command:  isrc all
  Response: isrc ok

  Command:  isrc 40
  Response: isrc ok
```

3.4. SPI Master

3.4.1. smss

Set the speed of the SPI bus. The given value must match one of the predefined bus speeds defined in the AxiCat protocol (see axicat.h, AXICAT_SPI_SPEED_<n>).

Command syntax:

```
  + [LABEL] =smss

  + Bus speed (Hz):
    [NUMBER] =750000
             =1500000
             =3000000
             =6000000
```

Response syntax:

```
  + [LABEL] =smss

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

Example:

```
  Command:  smss 750000
  Response: smss ok
```

Set the speed of the SPI bus to 750000 Hz.

3.4.2. smsr

Directly write to the speed registers of the SPI function (a.k.a. setting raw register values).

Refer to section “SPI – Serial Peripheral Interface” in the Atmega164A/324A/644A/1284 datasheet for more information.

Command syntax:

```
  + [LABEL] =smsr
  + [NUMBER] =0..3: CR register.
  + [NUMBER] =0..1: X2 register.
```

Response syntax:

```
  + [LABEL] =smsr

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

Example:

```
  Command:  smsr 2 1
  Response: smsr ok
```

3.4.3. smsc

Set configuration of the SPI bus.

Command syntax:

```
  + [LABEL] =smsc

  + Clock polarity:
    [NUMBER] =0: Clock is low when idle.
    [NUMBER] =1: Clock is high when idle.

  + Clock phase:
    [NUMBER] =0: Sample on leading edge.
    [NUMBER] =1: Sample on trailing edge.

  + Bit order:
    [NUMBER] =0: MSb->LSb.
    [NUMBER] =1: LSb->MSb.
```

Response syntax:

```
  + [LABEL] =smsc

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

Example:

```
  Command:  smsc 0 0 0
  Response: smsc ok
```

3.4.4. sme

This command enables the SPI master of the AxiCat.

Note that you can schedule transfers when the SPI master is disabled. They'll be executed as soon as the enable command has been processed.

Command syntax:

```
  + [LABEL] =sme
```

Response syntax:

```
  + [LABEL] =sme

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

Example:

```
  Command:  sme
  Response: sme ok
```

3.4.5. smd

This command disables the SPI master of the AxiCat.

All SPI master transfers being processed in the AxiCat will be completed. However, one or more transfers that are still scheduled in the application layer will be initiated in the AxiCat. The AxiCat will merely buffer these transfers, but for the application layer the transfers have been started.

If your intention is to cancel all SPI master transfers and disable the SPI function, issue the following commands:

```
Command: smc
Command: smd
```

The order is important. First request cancellation, then disable the SPI function. If you do it the other way around, the application layer may send one or more scheduled transfers to the AxiCat between "smd" and "smc".

Command syntax:

```
+ [LABEL] =smd
```

Response syntax:

```
+ [LABEL] =smd

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

Example:

```
Command:  smd
Response: smd ok
```

3.4.6. SPI Transfers

Asynchronous transfers:

Command	Transfer Type	Result Types
smt	SPI	success cancel skip

Result	Description
success	The transfer has completed in the AxiCat. The response reports the outcome of the bus transaction.
cancel	The transfer was cancelled before it was sent down to the AxiCat.
skip	The transfer was sent down to the AxiCat. The AxiCat skipped the execution of the transfer.

3.4.6.1. smt

This command schedules an SPI transfer.

The command and the response contain the same number of data bytes, unless the transfer was aborted. The latter may happen if the SPI function is disabled while the transfer is being executed.

Command syntax:

```
  + [LABEL] =smt

  + [NUMBER] =0..3: Slave select line.

  + {PAYLOAD-BYTES: CNT=1..65535} Payload of 1..65535 data bytes.
```

Response syntax:

```
  + [LABEL] =smt

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

    \ + [NUMBER "smt-ss"] =0..3: Slave select line.

      + - + Array of 0..65535 elements:
              + [NUMBER "smt-payload"] =0..255: Payload data byte.

            - + [LABEL] =cancel

            - + [LABEL] =skip
```

Examples:

```
  Command:  smt 2 255 255
  Response: smt 2 160 120
```

3.4.6.2. smc

Request cancellation of a specific or all SPI master transfers.

Command syntax:

```
  + [LABEL] =smc

    / + [NUMBER] Command identifier of transfer to cancel.
  + - + [LABEL] =all: Cancel all transfers.
    \ . Cancel all transfers.
```

Response syntax:

```
  + [LABEL] =smc

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

Examples:

```
  Command:  smc
  Response: smc ok

  Command:  smc all
  Response: smc ok

  Command:  smc 40
  Response: smc ok
```

3.5. 1-Wire Master

3.5.1. ome

This command enables the 1-Wire master of the AxiCat.

Note that you can schedule transfers when the 1-Wire master is disabled.
They'll be executed as soon as the enable command has been processed.

Command syntax:

```
  + [LABEL] =ome
```

Response syntax:

```
  + [LABEL] =ome

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

Example:

```
  Command:  ome
  Response: ome ok
```

3.5.2. omd

This command disables the 1-Wire master of the AxiCat.

Command syntax:

```
  + [LABEL] =omd
```

Response syntax:

```
  + [LABEL] =omd

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

Example:

```
  Command:  omd
  Response: omd ok
```

All 1-Wire master transfers being processed in the AxiCat will be completed. However, one or more transfers that are still scheduled in the application layer will be initiated in the AxiCat. The AxiCat will merely buffer these transfers, but for the application layer the transfers have been started.

If your intention is to cancel all 1-Wire master transfers and disable the 1-Wire function, issue the following commands:

```
  Command: omc
  Command: omd
```

The order is important. First request cancellation, then disable the 1-Wire function. If you do it the other way around, the application layer may send one or more scheduled transfers to the AxiCat between "omd" and "omc".

3.5.3. 1-Wire Transfers

Asynchronous transfers:

Command	Transfer Type	Result Types
omr	1-Wire Reset	success cancel skip
omt	1-Wire Touch Bytes	success cancel skip
omb	1-Wire Touch Bits	success cancel skip
omnf	1-Wire Enumerate First	success cancel skip
omnn	1-Wire Enumerate Next	success cancel skip
omp	1-Wire Probe	success cancel skip

Result	Description
success	The transfer has completed in the AxiCat. The response reports the outcome of the bus transaction.
cancel	The transfer was cancelled before it was sent down to the AxiCat.
skip	The transfer was sent down to the AxiCat. The AxiCat skipped the execution of the transfer.

```
3.5.3.1. omr
------------

This command schedules a 1-Wire reset.

Command syntax:

  + [LABEL] =omr

Response syntax:

  + [LABEL] =omr

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

    \ + - + [NUMBER] =0: No 1-Wire slaves present.
                     =1: One or more 1-Wire slaves are present.

              - + [LABEL] =cancel

              - + [LABEL] =skip

Examples:

  Command:  omr
  Response: omr 1

  Command:  omr
  Response: omr 0
```

3.5.3.2. omt

This command performs a 1-Wire touch bytes operation.

Under the surface, the command schedules a 1-Wire touch bits transfer. The data bits are specified as an array of bytes. As such, the command always transfers a multiple of 8 bits.

The command and the response contain the same number of data bytes, unless the transfer was aborted. The latter may happen if the 1-Wire function is disabled while the transfer is being executed.

Command syntax:

```
  + [LABEL] =omt

  + {PAYLOAD-BYTES: CNT=1..8191} Payload of 1..8191 data bits.
```

Response syntax:

```
  + [LABEL] =omt

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

    \ + - + Array of 0..8191 elements:
              + [NUMBER "omt-payload-byte"] =0..255: Payload data byte.

            - + [LABEL] =cancel

            - + [LABEL] =skip
```

Example:

```
  Command:  vfmts "omt-payload-byte" hex 3 1 1 0 1 0
  Command:  omr
  Command:  omt 055h 028h 0C6h 0D5h 00Ch 004h 000h 000h 033h 044h
  Command:  wait 750
  Command:  omr
  Command:  omt 055h 028h 0C6h 0D5h 00Ch 004h 000h 000h 033h
  Command:  omt 0BEh 0FFh 0FFh 0FFh 0FFh 0FFh 0FFh 0FFh 0FFh
  Response: vfmts ok
  Response: omr 1
  Response: omt 055h 028h 0C6h 0D5h 00Ch 004h 000h 000h 033h 044h
  Response: wait ok
  Response: omr 1
  Response: omt 055h 028h 0C6h 0D5h 00Ch 004h 000h 000h 033h
  Response: omt 0BEh 099h 001h 04Bh 046h 07Fh 0FFh 007h 010h 079h

    Temperature measurement using a DS18B20. The temperature value is returned
    as 099h 001h which translates to 25.6 degrees Celsius.
```

3.5.3.3. omb

This command schedules a 1-Wire touch bits transfer.

The command and the response contain the same number of data bits, unless the transfer was aborted. The latter may happen if the 1-Wire function is disabled while the transfer is being executed.

Command syntax:

```
  + [LABEL] =omb

  + {PAYLOAD-BITS} payload of 1..65535 data bits.
```

Response syntax:

```
  + [LABEL] =omb

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

    \ + - + Array of 0..65535 elements:
              + [NUMBER "omt-payload-bit"] =0..1: Payload data bit.

            - + [LABEL] =cancel

            - + [LABEL] =skip
```

Example:

```
  Command:  vfmts "omt-payload-byte" hex 3 1 1 0 1 0
  Command:  omr
  Command:  omt 55h 10h 17h 9Ah 0A4h 02h 08h 00h 0B1h 0B4h
  Command:  omb 1
  Response: vfmts ok
  Response: omr 1
  Response: omt 055h 010h 017h 09Ah 0A4h 002h 008h 000h 0B1h 0B4h
  Response: omb 0

    Read the power mode of a DS18S20 with ROM code 10-000802A49A17-B1.
```

3.5.3.4. omnf

The "omnf" command schedules the first iteration of a 1-Wire enumeration. The command initiates an enumeration procedure.

The command establishes a new enumeration context based on the specified search criteria, and executes the first iteration of the enumeration procedure. In other words, the command searches for the first 1-Wire slave.

"omnf" without parameters will start a search for all 1-Wire slaves that are visible on the 1-Wire bus. You can narrow down the enumeration by specifying any combination of the following search criteria:
* Alarm condition search: Only 1-Wire slaves that have an alarm condition set are enumerated.
* Family search: Only 1-Wire slaves bearing the specified family code are enumerated.
* DS2409 smart ON: If you choose this option, only 1-Wire slaves that reside behind the main or auxiliary port are enumerated. You have to choose between main and auxiliary port and you've to provide the ROM code of the target DS2409.

The DS2409 ROM code is made up of hexadecimal digits and one or two hyphens:
* The family code comes first, 00..FF hexadecimal.
* An hyphen follows.
* The serial number is specified as one or more hexadecimal digits.
* Optionally:
  * An hyphen.
  * Followed by a CRC value, 00..FF hexadecimal.

The server calculates the CRC value of the DS2409 ROM code. If a CRC value is specified, it overwrites the calculated value.

Command syntax:

```
  + [LABEL] =omnf

    / + [LABEL] =alarm
  +
    \ .

    / + [LABEL]  =family
  +   + [NUMBER] =0..255: Family code.
    \ .

    / + [LABEL]  =main
                 =aux
  +   + [STRING] DS2409 ROM code.

    \ .
```

Response syntax:

```
  + [LABEL] =omnf

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

    \ + - + [STRING] ROM code of enumerated 1-Wire slave.
        \ . No 1-Wire slave found.

              - + [LABEL] =cancel

              - + [LABEL] =skip
```

Examples:

```
  Command:  omnf
  Response: omnf
```

Start first iteration of an enumeration. No 1-Wire slaves were found.

```
  Command:  omnf
  Command:  omnn
  Command:  omnn
  Command:  omnn
  Command:  omnn
  Command:  omnn
  Command:  omnn
  Command:  omnn
  Command:  omnn
  Command:  omnn
  Response: omnf "20-00000014C3CF-0E"
  Response: omnn "10-000802A49A17-B1"
  Response: omnn "30-000012B5735B-F4"
  Response: omnn "42-00000038D0BE-A3"
  Response: omnn "22-0000003201DA-1C"
  Response: omnn "3A-000000052F6A-85"
  Response: omnn "01-000016707B5B-C5"
  Response: omnn
  Response: omnn
  Response: omnn
```

Schedule ten iterations of an enumeration. Seven 1-Wire slaves were found.

```
  Command:  omnf alarm
```

Start enumeration of 1-Wire slaves with an alarm condition set.

```
  Command:  omnf family 26h
```

Start enumeration of 1-Wire slaves with family code 26h (DS2438).

```
  Command:  omnf main "EF-15207BA3"
```

Start enumeration of 1-Wire slaves located behind the main port of the specified DS2409.

```
  Command:  omnf alarm family 29h aux "EF-15207BA3"
```

Start enumeration of 1-Wire slaves of family 29h (DS2408) with an alarm codition set and located behind the auxiliary port of the specified DS2409.

3.5.3.5. omnn

"omnn" continues from where the previous "omnf" or "omnn" command left and searches for the next 1-Wire slave, if any.

Note that the "omnf" and "omnn" response values are compatible. When processing programmatically, code can incorporate one routine for parsing "omnf" and "omnn" responses.

Command syntax:

```
  + [LABEL] =omnn
```

Response syntax:

```
  + [LABEL] =omnn

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

    \ + - + [STRING] ROM code of enumerated 1-Wire slave.
        \ . No 1-Wire slave found.

              - + [LABEL] =cancel

              - + [LABEL] =skip
```

Examples: see "omnf".

3.5.3.6. omp

This command schedules a 1-Wire probing command.

Command syntax:

```
  + [LABEL]  =omnp
  + [STRING] ROM code of 1-Wire slave.
```

Response syntax:

```
  + [LABEL] =omnp

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

    \ + - + [NUMBER] =0: 1-Wire slave not found.
                     =1: 1-Wire slave found.

              - + [LABEL] =cancel

              - + [LABEL] =skip
```

Examples:

```
  Command:  omp "3A-52F6A"
  Response: omp 0

  Command:  omp "20-14C3CF-0E"
  Response: omp 1
```

3.5.3.7. omc

Request cancellation of a specific or all 1-Wire master transfers.

Command syntax:

```
  + [LABEL] =omc

    / + [NUMBER] Command identifier of transfer to cancel.
  + - + [LABEL] =all: Cancel all transfers.
    \ . Cancel all transfers.
```

Response syntax:

```
  + [LABEL] =omc

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

Examples:

```
  Command:  omc
  Response: omc ok

  Command:  omc all
  Response: omc ok

  Command:  omc 40
  Response: omc ok
```

3.6. UART #0 and #1

3.6.1. u0sb, u1sb

Set the baud rate of the specified UART. The given value must match one of the predefined baud rates defined in the AxiCat protocol (see axicat.h, AXICAT_UART_BAUDRATE_<n>).

Command syntax:

```
  + [LABEL] =u0sb: UART #0.
            =u1sb: UART #1.

  + Baud rate:
    [NUMBER] =1200
             =2400
             =4800
             =9600
             =19200
             =38400
             =57600
             =115200
```

Response syntax:

```
  + [LABEL] =u0sb
            =u1sb

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

Example:

```
  Command:  u0sb 115200
  Response: u0sb ok
```

3.6.2. u0sr, u1sr

Directly write to the speed registers of the UART function (a.k.a. setting raw register values).

Refer to section “USART” in the Atmega164A/324A/644A/1284 datasheet for more information.

Command syntax:

```
  + [LABEL] =u0sr: UART #0.
            =u1sr: UART #1.

  + [NUMBER] =0..4095: UBRR register.

  + [NUMBER] =0..1: X2 register.
```

Response syntax:

```
  + [LABEL] =u0sr
            =u1sr

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

Example:

```
  Command:  u1sr 2499 0
  Response: u1sr ok

    Set baud rate of UART #1 to 300.
```

3.6.3. u0sd, u1sd

Set the number of bits per data word for the selected UART.

The given value must match one of the values defined in the AxiCat protocol (see axicat.h, AXICAT_UART_DATA_BITS_<n>).

Command syntax:

```
  + [LABEL] =u0sd: UART #0.
            =u1sd: UART #1.

  + Data bits:
    [NUMBER] =5..9
```

Response syntax:

```
  + [LABEL] =u0sd
            =u1sd

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

Example:

```
  Command:  u1sd 9
  Response: u1sd ok

    Set the data word size of UART #1 to 9 bits.
```

3.6.4. u0ss, u1ss

Set the number of bits per data word for the selected UART.

The given value must match one of the values defined in the AxiCat protocol (see axicat.h, AXICAT_UART_STOP_BITS_<n>).

Command syntax:

```
  + [LABEL] =u0ss: UART #0.
            =u1ss: UART #1.

  + Stop bits:
    [NUMBER] =1..2
```

Response syntax:

```
  + [LABEL] =u0ss
            =u1ss

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

Example:

```
  Command:  u1ss 2
  Response: u1ss ok

    Set 2 stop bits per data word for UART #1.
```

3.6.5. u0st, u1st

Set the timeout value in milliseconds for reporting received data. A value of zero means that the AxiCat will immediately report any received data words. A non-zero value programs a timeout for reporting all buffered data words since the last data word was received.

Note that the AxiCat has a limited buffer for receiving data words and will report the data words anyway when the buffer is full.

Command syntax:

```
  + [LABEL] =u0st: UART #0.
            =u1st: UART #1.

  + Timeout (milliseconds):
    [NUMBER] =       0: No Rx timeout, report immediately.
             =1..65535: Rx timeout for reporting received data words.
```

Response syntax:

```
  + [LABEL] =u0st
            =u1st

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

Example:

```
  Command:  u0st 100
  Response: u0st ok

    Set the Tx timeout to 100 milliseconds for UART #0.
```

3.6.6. u0su, u1su

Enable or disable reporting of of unsolicited Rx responses.

An unsolicited Rx response is formatted as a "u0r" or "u1r" response, depending on the originating UART.

The number of data bits determines the words size. If 9 data bits are selected, 2 data bytes for each word, ordered LSB->MSB, are reported by the AxiCat.

The maximum payload of an Rx response is currently defined as 256 data bytes (see axicatserver.h, RSP_UXR_MAX_CNT).

Command syntax:

```
  + [LABEL] =u0su: UART #0.
            =u1su: UART #1.

  + [NUMBER] =0: Disable unsolicited Rx responses.
             =1: Enable unsolicited Rx responses.
```

Response syntax:

```
  + [LABEL] =u0su
            =u1su

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

Example:

```
  Command:  u0su 1
  Response: u0su ok

    Enable unsolicited Rx responses for UART #0.
```

3.6.7. u0e, u1e

This command enables the specified UART.

Command syntax:

```
  + [LABEL] =u0e: UART #0.
            =u1e: UART #1.
```

Response syntax:

```
  + [LABEL] =u0e
            =u1e

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

Example:

```
  Command:  u0e
  Response: u0e ok
```

3.6.8. u0d, u1d

This command disables the specified UART.

Command syntax:

```
  + [LABEL] =u0d: UART #0.
            =u1d: UART #1.
```

Response syntax:

```
  + [LABEL] =u0d
            =u1d

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

Example:

```
  Command:  u0d
  Response: u0d ok
```

3.6.9. u0w, u1w

Transmit data words over the given UART's TXD line.

The number of data bits determines the words size. If 9 data bits are selected, you've to provide 2 data bytes for each word, in LSB->MSB order.

Command syntax:

```
  + [LABEL] =u0w: UART #0.
            =u1w: UART #1.

  + {PAYLOAD-BYTES: CNT=0..65535} payload of 0..65535 data bytes.
```

Response syntax:

```
  + [LABEL] =u0w
            =u1w

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

Example:

```
  Command:  u0w "Hello" 13 10
  Response: u0w ok
```

3.6.10. u0r, u1r

Read data from the specified UART.

The maximum payload of an Rx response is currently defined as 256 data bytes (see axicatserver.h, RSP_UXR_MAX_CNT).

The "u0r" and "u1r" responses also serve as unsolicited Rx responses.

Command syntax:

```
  + [LABEL] =u0r: UART #0.
            =u1r: UART #1.

    / + [NUMBER] =  0: Let server choose number of data bytes in payload.
  + - +          =1..: Propose maximum data bytes in payload.
    \ . Let server choose number of data bytes in payload.
```

Response syntax:

```
  + [LABEL] =u0r
            =u1r

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

    \ + - + Array of 0..65535 elements:
                / [NUMBER "u0r-word"] =0..255: Payload data bytes from UART #0.
              +
                \ [NUMBER "u1r-word"] =0..255: Payload data bytes from UART #1.
```

Examples:

```
  Command:  u0r
  Response: u0r

    Read data from UART #0. No data available.

  Command:  u0r
  Response: u0r 41h 42h 43h 44h

    Read data from UART #0. Four data bytes were available.

  Command:  u0r 10
  Response: u0r 41h 42h

    Read at most ten data byte from UART #0.
```

3.7. Server

3.7.1. 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 "omt-payload-byte" hex 2 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.
```

3.7.2. 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 "omt-payload-byte"
  Response: vfmtg "omt-payload-byte" hex 2 1 1 1 1 0
  Response: vfmtg ok
```

3.7.3. ver

Get version information.

Command syntax:

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

Response syntax:

```
  + [LABEL] =ver

  + - + [STRING] Version of the server.
      + [STRING] Version of the AxiCat.

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

Example:

```
  Command:  ver
  Response: ver "1.3.1" "AxiCat v1.3.0 2014-12-12"
```

3.7.4. 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
```

3.7.5. close

Close client connection.

This command doesn't generate a response.

Command syntax:

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

Example:

```
Command: close
```

3.7.6. 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
```

3.8. Additional Syntax Flowcharts

3.8.1. {PAYLOAD-BYTES} - Payload of Data Bytes

3.8.1.1. Usage

Parameters:

  CNT: Number expected of data bytes. This can be a range of numbers.

Example:

  + {PAYLOAD-BYTES: CNT=0..65535} Payload of 0..65535 data bytes.

3.8.1.2. Flowchart

Syntax:

```
  + P1

  + - + [NUMBER] =0..255
      + -> P1

    - + [STRING] ASCII string
      + -> P1

    \ .
```

Examples:

  45h 41h 1000_1010_b

  "A line of text" 13 10 0

3.8.2. {PAYLOAD-BITS} - Payload of Data Bits

3.8.2.1. Usage

Parameters:

  CNT: Number of expected data bits. This can be a range of numbers.

Example:

  + {PAYLOAD-BITS} Payload of 1..65535 data bits.

3.8.2.2. Flowchart

Syntax:

```
  + P1

  + - + [NUMBER] =0..1
      + -> P1

    - + [STRING] ASCII string containing characters '0', '1', and spaces.
      + -> P1

    \ .
```

Examples:

  1 1 0 0 1 1 1 0

  0b 01b 0d 01h "1111 0000 0110 1100"

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

2016-08-30  Peter S'heeren, Axiris

  * First release.

2016-11-06  Peter S'heeren, Axiris

  * Fixed typos.
  * Second release.
```