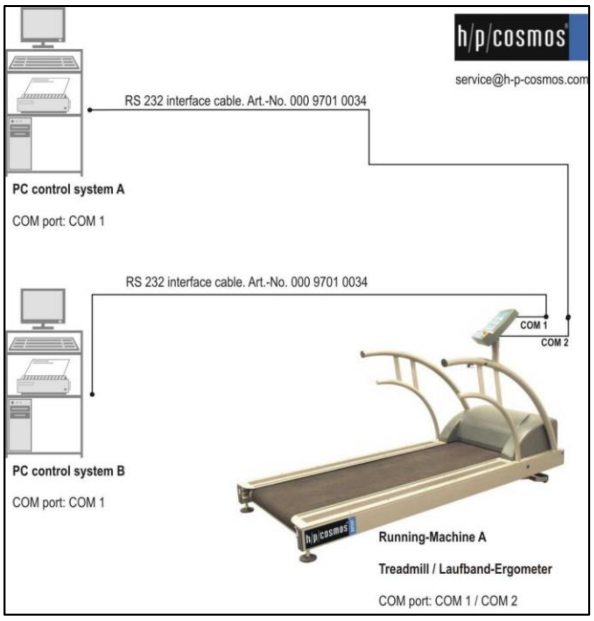


functional training
fitness & healthcare
gait training & gait analysis
sports science & athletics
motion analysis & biomechanics
rehab & locomotion therapy

h/p/cosmos® coscom® v4

treadmills & ergometers interface communication protocol

Basic Remote Service

Created for:
h/p/cosmos sports & medical gmbh
Am Sportplatz 8
DE 83365 Nussdorf-Traunstein
Germany
phone + 49 86 69 86 42 0
fax + 49 86 69 86 42 49
email@h-p-cosmos.com
www.h-p-cosmos.com

Author:
M. Sc. Andreas Feil,
Altotec Hard- und Software GmbH
Altofing 7, 83367 Petting / Germany

preliminary version / white paper
published: www.coscom.org
article #: cos100115v4
date: 2018-03-09

.

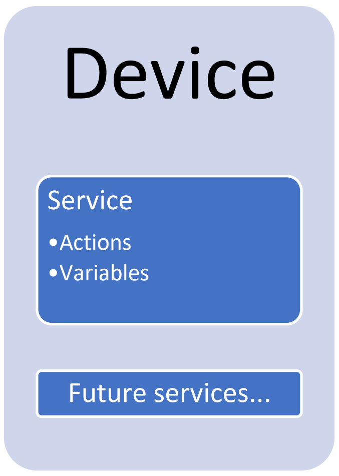
Table of contents

[1] Overview

h/p/cosmos coscom v4 defines an interface for basic device control. At the moment, the protocol is designed for five different device types (not all are available on the market right now):

- h/p/cosmos running machines (treadmill ergometers)
- h/p/cosmos ladder ergometers
- h/p/cosmos ergometers
- h/p/cosmos steppers
- h/p/cosmos cross trainers

The interface consists of one device which currently has one service (index 0). The service has several actions and variables with which the device can be controlled. In future, there could be an update which can add new services if necessary.



The protocol has a general request – response design. External software is always allowed to send a request but must wait for the corresponding answer before sending the next request (only one message can be pending). For active control see chapter safety considerations.

In order to get actual variable values, you can either query the latest value of a variable (so called polling) or get an automatic event message, when the variable value changes (see chapter eventing).

h/p/cosmos coscom v4 uses serial interface for communication. In order to distinguish between version 3 the baud rate is changed to 19200.

[2] Basic message format

h/p/cosmos coscom v4 is a UTF-8 based protocol. Every message starts with "*" followed by a character which indicates the message type:

- **"*A"** indicates an action request or response. Parameters are described with **"*I"** for input parameters followed by a parameter index and **":"** as delimiter for the parameter value. Output parameters follow the same scheme but with character **"O"**. Example request: `*A0s0*I0:<FirstParameterValue>*I1:<SecondParameterValue>*Y0:8D*Z`
- **"*Q"** followed by a variable index indicates a variable query request or response:
  - Request example for variable with index 0 on service index 0: `*Q0s0*Y0:4E*Z`
  - Response example for a variable with index 0 on service index 0: `*Q0s0:<CurrentValue>*Y0:E2*Z`
  - Querying variables which are not supported by current device type will result in an error response.
- **"*E"** indicates an event message. For further information see chapter "eventing".
- **"*R"** is a special response message this is sent when a wrong checksum was detected. See chapter error messages for details.

After the "main" character follows the service index **"s" + index** of service. Currently there is only one service defined, so this will always be "s0". A coscom v4 device will accept messages where no service index is defined (missing "s0"). If this is the case a default service index of 0 is assumed.

Every h/p/cosmos coscom v4 message ends with the **checksum extension followed by "*Z"**.

The **"*"** character is reserved as protocol delimiter therefore any "*" in an argument or variable value must be **escaped with "*X".** Example: If you want to send a message with "RequestControl" action which contains a "*" character, you would have to escape it:
`*A3s0*I0:My own text with a *X character*Y0:09*Z`

h/p/cosmos coscom v4 defines a **max. message length for incoming requests (external software to device) of 64 bytes and a max message length of 250 bytes for outgoing message**.

[3] Checksum extension

Every coscom v4 message must use the checksum extension. This is necessary because of RS232 support, where no data link layer exists.

request example: `*A2s4*I0:2*I1:3*I2:0`**`*Y0:73`**`*Z`
response example: `*A2s4`**`*Y0:44`**`*Z`

The calculation of the checksum is done by getting all UTF-8 bytes for the characters starting by the first '*' character of the message and ending before the message end "*Z". The bytes have to be added and computed modulo 256. The hexadecimal representation of the result is the actual checksum.

[4] Error messages

A h/p/cosmos device waits till it receives a "*Z" indicating that a whole coscom message was received. If this part of the message is disturbed the external software will run in a timeout.

If a h/p/cosmos device detects a "*Z" it parses the checksum of the received message. If no checksum is attached or an invalid checksum is detected it will send a general error response `*R1*F0:<ErrorNumber>*R1:<ErrorText>*Y0:84*Z`. The error description ("`*F1:Errortext`") is optional and can be left out.

If an h/p/cosmos device detects a wrong parameter (e. g. speed out of range) in an otherwise correct message it will send an error response message to the external software. An error message starts with the same Indices as the request but is followed by a "*F0:", an error number, "*F1:" and an error description. An example looks like this: `*A0s0*F0:123*F1:ErrorText*Y0:38*Z`

The error description ("`*F1:ErrorText`") is optional an can be left out.

[5] Eventing

For getting variable changes you can either use polling method be querying the current variable values. This is according to the general request -response design of the protocol. If you want to get automatic messages of the device, if variable values change, you can use so called eventing. This an extension of the protocol where the device sends messages without an explicit request. You can use action "SetEventMask" to control which variables should send automatic changes (default after switching device on: all off).

RS232 does not allow for discovering connection change therefore event mask could be set to any value when you are connecting. A precursor could have set it according to his needs. Best practice is to set event mask at connecting time to your needs (if you don't want any events, set all off.)

A service publishes changes to its variables by sending so called event messages. These messages contain the indices of one or more variables and the current value of those variables. There can be more than one value in a single event message, if more variables change at the same time.

An initial event message is sent when subscription changes by "SetEventMask". This event message contains the indices and values for all evented variables and allows the subscriber to initialize its model of the state of the service.

Event messages are tagged with an event key, which has one digit. The event key for a subscription is initialized to 0 when the device sends the initial event message. For each subsequent event message, the device increments the event key for a subscription, and includes that updated key in the event message. Event keys handle overflow and wrap the event key from 9 back to 1 (not 0). Subscribers must also handle this special case when the next event key is not an increment of the previous key.

To repair an event subscription, e.g., if a subscriber has missed one or more event messages, a subscriber must re-subscribe by invoking "SetEventMask" again. By doing so, the subscriber will get a new initial event message, and a new event key.

Following the general event syntax (for better reading the elements are put in separate lines):

*E**event key**s*service index*

*V**variable index**:*variable value*

other variables and their values go here, if any

*Y0:**xx***Z

An actual example message with event key **1** and two variable updates (Index **0** and **1**) of service **0** would look like this:

```
*E1s0*V0:<value1>*V1:<value2>*Y0:A9*Z
```

[6] Safety considerations

One goal was making h/p/cosmos coscom v4 more secure. h/p/cosmos coscom v3 has several safety features. But not all of them were mandatory. Over the last years regulations for medical devices have become more severe. In order to meet these concerns h/p/cosmos has changed some safety features and made them mandatory. Following the basic safety procedure of h/p/cosmos coscom v4 will be described.

[6A] GetDeviceInformation

In order to check if a h/p/cosmos coscom v4 device is present, you must use action “GetDeviceInformation”. You will get device type, the variant (distinguish between treadmill, ladder ergometer, bicycle and stepper), serial number and firmware version of the connected device.

[6B] Request control – fail-safe and stop state

An external device can send a h/p/cosmos coscom v4 message at any time. But before actively controlling the device by changing any load (speed, elevation, power, start or stop command) you first must request control allowance by using action “RequestControl”. This can be done anytime. After invoking this action, the user will see a message on the user terminal (if device hardware supports displaying text) where he must confirm external control allowance. If he accepts, variable “ControlAllowed” will change to “Allowed” if he declines the request, “NotAllowed” will be set. While the request is shown “RequestPending” will be set.

You can only actively control the device when “ControlAllowed” is set. Control is then generally allowed as long as control state “stopped” or “pause” is reached. This could be by pressing the stop button on the terminal, a failsafe, an error or sending stop action on coscom v4.

If the device supports control state “pause” external control will be revoked temporary until the user decides if he wants to continue or stop the current workout. This “pause” state is optional and depending on the actual device. “Pause” cannot be achieved due external command. Sending the “Stop” action will set control state to “stop”.

If variable “ControlAllowed” changes to “TemporaryNotAllowed” external software must pause current control profile. In this state no active commands are allowed and will result in an error response (error code: 133, “Control not allowed”). If the user continues workout, the state will change to “ControlAllowed” again and external software should continue its profile. Otherwise if

"NotAllowed" is set, you have to request control again before sending active controls. Sending request control while in "TemporaryNotAllowed" will show the user a message if he wants to allow control again.

If variable "ControlAllowed" changes to "NotAllowed" external software must cancel current control profile. In order to start controlling again, you have to repeat requesting control.

While external control is allowed, using Failsafe for detecting communication loss is mandatory. You have to invoke action "SetFailsafe" within every second while controlling the device. If the device does not receive the message within one second, the device will stop and control will be revoked.

This concept ensures that a stop state of the device is "detected" (by getting an error response to an active control request) by external software even if variable "ControlState" is not monitored.

[7] Feature Matrix

h/p/cosmos coscom v4 is designed to meet different device types. But not all device types support all defined actions and variables. Following matrix will address this issue:

Action/Device type	Treadmill	Ergometer	Ladder	Cross trainer	Stepper
GetDeviceInformation	x	x	x	x	x
SetEventMask	x	x	x	x	x
RequestControl	x	x	x	x	x
ResetFailsafe*	x	x	x	x	x
SetSpeed*	x		x		
GetSpeedRange	x		x		
GetAccelDecelRange	x		x		
HoldSpeed*	x		x		
SetElevation*	x				
SetElevationWithSpeed*	x				
GetElevationRange	x				

HoldElevation*	x				
SetPower*		x		x	x
SetTorque*		x		x	x
SetCadence*		x		x	
Start*	x	x	x	x	x
Stop*	x	x	x	x	x
SetPersonData*	x	x	x	x	x
GetPersonData	x	x	x	x	x
ResetCounterValues*	x	x	x	x	x
Beep*	x	x	x	x	x

** Using this action without external control permission will result in an error response (see chapter "Safety considerations").*

Variable/Device type	Treadmill	Ergometer	Ladder	Cross trainer	Stepper
ControlStatus (Index: 0)	x	x	x	x	x
ControlAllowed (Index: 1)	x	x	x	x	x
ActualSpeed (Index: 2)	x	x		x	
TargetSpeed (Index: 3)	x	x		x	
ActualElevation (Index: 4)	x				
TargetElevation (Index: 5)	x				
ActualPower (Index: 6)	x	x	x	x	x
TargetPower (Index: 7)		x		x	x
ActualTorque (Index: 18)		x		x	
TargetTorque (Index: 19)		x		x	
EnergyConsumption (Index: 8)	x	x	x	x	x
MET (Index: 9)	x	x	x	x	x
Time (Index: 10)	x	x	x	x	x
Distance (Index: 11)	x	x	x	x	
ActualCadence (Index: 12)		x		x	

TargetCadence (Index: 21)		x		x	
Height (Index: 13)	x		x		x
HeartRate (Index: 14)	x	x	x	x	x
RRInterval (Index: 15)	x	x	x	x	x
Errors (Index: 16)	x	x	x	x	x
StepHeight (Index: 20)			x		x

[8] Actions

Following you will find a short description with syntax sample of every h/p/cosmos coscom v4 action of basic remote service.

Note: If no other description is given, values are defined as floating point values expected with max. two decimal places and maximal value +/- 1.000.000. Integer values are defined as signed 32-bit values. String values max.

[8A] GetDeviceInformation

(Action index: 0) Gets basic device information for detecting if a h/p/cosmos coscom v4 device is available.

Parameters:

- *DeviceType* (direction out - index 0, max. length: 64 characters):
  e. g. urn:schemas-coscomorg:device:LadderErgometer:1
- *Variant (direction out*- index 1, digit): Number that indicates the actual variant type. Possible Values:
  - *0 = Treadmill Ergometer*
  - *1= Ladder*
  - *2 = Cross trainer*
  - *3 = Stepper*
  - *4 = Bicycle Ergometer*
- *Serial Number* (direction out – index 2, max. length: 50 characters): serial number string
- *FirmwareVersion* (direction out – index 3, max. length: 10 characters): Current firmware version

Syntax sample request: `*A0s0*Y0:3E*Z`
Syntax sample response: `*A0s0*O0:urn:schemas-coscom-org:device:Coscomv4Device:1*O1:1*O2:cos30007-01va06-0003*O3:1.0.0001*Y0:BF*Z`

[8B] SetEventMask

(Action index: 1) As described in chapter "eventing" with this method it is possible to enable or disable automatic value change transmission of the device. After switching on the device all events are disabled. Changing the event mask will lead to new initial events and new event index in the event messages (see chapter "eventing").

Parameters

- *EventMask* (direction in – index 0): a string consisting of 0 and 1 indicating if eventing for specific variables should be turned on. The position in the string determines the variable index. For example, the string 100101 enables eventing for variables with indices 0, 2 and 5. Leading 0 can be left out. The string starts with the highest variable index and ends with index 0. For variable indices see chapter variables. For variable indices which are not supported in the current device type changes to event mask will have no effect.

Syntax sample request: `*A1s0*I0:1001*Y0:DE*Z`
Syntax sample response: `*A1s0*Y0:3F*Z`

[8C] RequestControl

(Action index: 2) With this action external control can be requested. It must be done before actively controlling the device. For further information on controlling the device see chapter "Safety considerations".

Parameters

- *Message* (direction in – index 0, max. length: 45 characters): a string which will be shown to the user when he should accept external control. Because of different hardware types it is not guaranteed that the message is shown to the user. If no message should be shown, enter an empty string (request: `*A2s0*I0:*Y0:1D*Z`)

Syntax sample request: `*A2s0*I0:external software name wants to control*Y0:3D*Z`
Syntax sample response: `*A2s0*Y0:40*Z`

[8D] ResetFailsafe

(Action index: 3) This action **must** be invoked regularly within 1 second while external control is allowed. For more information see chapter "Safety considerations". If the action is invoked when controlling is not allowed an error response will be sent back from the device (see chapter "error messages").

Parameters

No parameters defined

Syntax sample request: `*A3s0*Y0:41*Z`
Syntax sample response: `*A3s0*Y0:41*Z`

[8E] SetSpeed

(Action index: 4) With this action you can set a new speed with given acceleration. Using this action without external control permission will result in an error response (see chapter "Safety considerations"). If speed or acceleration parameter is out of range, an error response will be sent.

Parameters

- *Speed* (direction in - index 0): The new target speed in m/s.
- *Acceleration* (direction in – index 1): The acceleration to be used with this command in m/s².

Syntax sample request (speed: 1.3 m/s, acceleration 0.2 m/s²):
`*A4s0*I0:1.30*I1:0.20*Y0:7F*Z`
Syntax sample response: `*A4s0*Y0:42*Z`

[8F] GetSpeedRange

(Action index: 5) Returns the current speed range of the device. The range can vary from its base value if it is additionally limited in the option settings of the device.

Parameters

- *MinimalSpeed* (direction out - index 0): Current minimal speed in m/s.
- *MaximalSpeed* (direction out – index 1): Current maximal speed in m/s.

Syntax sample request: `*A5s0*Y0:43*Z`
Syntax sample response (min 0 m/s, max 6.11 m/s): `*A5s0*O0:0.00*O1:6.11*Y0:8E*Z`

[8G] GetAccelDecelRange

(Action index: 6) Returns the current acceleration/deceleration range of the device. The range can vary from its base value if it is additionally limited in the option settings of the device.

Parameters

- *MinAcceleration* (direction out - index 0): Current minimal acceleration/deceleration in m/s².
- *MaxAcceleration* (direction out – index 1): Current maximal acceleration/deceleration in m/s².

Syntax sample request: `*A6s0*Y0:44*Z`
Syntax sample response (min 0.1 m/s², max: 0.6 m/s²): `*A6s0*O0:0.10*O1:0.60*Y0:8E*Z`

[8H] HoldSpeed

(Action index: 7) Can be used to hold the current speed at a constant level. Using this action without external control permission will result in an error response (see chapter "Safety considerations").

Parameters

No parameters defined

Syntax sample request: `*A7s0*Y0:45*Z`
Syntax sample response: `*A7s0*Y0:45*Z`

[8I] SetElevation

(Action index: 8) With this action you can set a new elevation. Using this action without external control permission will result in an error response (see chapter “Safety considerations”). If elevation parameter is out of range, an error response will be sent.

Parameters
- *Elevation* (direction in - index 0): The new target elevation in %.

Syntax sample request (elevation: 3.3 %): `*A8s0*I0:3.30*Y0:E7*Z`
Syntax sample response: `*A8s0*Y0:46*Z`

[8J] SetElevationWithSpeed

(Action index: 18) With this action you can set a new elevation with optional elevation speed. Using this action without external control permission will result in an error response (see chapter “Safety considerations”). If elevation parameter is out of range, an error response will be sent.

Parameters
- *Elevation* (direction in - index 0): The new target elevation in %.
- *ElevationSpeed* (direction in – index 1): The elevation speed to be used with this command in °/s. If the device does not support elevation speed, this parameter will be ignored. To use the default elevation speed, set this parameter to 0.

Syntax sample request (elevation: 3.3 %, speed: default):
`*A18s0*I0:3.30*I1:0*Y0:26*Z`
Syntax sample response: `*A18s0*Y0:77*Z`

[8K] GetElevationRange

(Action index: 9) Returns the current elevation range of the device. The range can vary from its base value if it is additionally limited in the option settings of the device.

Parameters
- *MinimalElevation* (direction out - index 0): Current minimal elevation in %.
- *MaximalElevation* (direction out – index 1): Current maximal elevation in %.

Syntax sample request: `*A9s0*Y0:47*Z`
Syntax sample response (min: 0 %, max: 22 %): `*A9s0*O0:0.00*O1:22.00*Y0:BE*Z`

[8L] HoldElevation

(Action index: 10) Can be used to hold the current elevation at a constant level. Using this action without external control permission will result in an error response (see chapter “Safety considerations”).

Parameters
No parameters defined

Syntax sample request: `*A10s0*Y0:6F*Z`
Syntax sample response: `*A10s0*Y0:6F*Z`

[8M] SetPower

(Action index: 11) Can be used to set a new target power. Using this action without external control permission will result in an error response (see chapter "Safety considerations").

Parameters
- *Power* (direction in - index 0, integer): New target power in watts.

Syntax sample request: `*A11s0*I0:50*Y0:B2*Z`
Syntax sample response: `*A11s0*Y0:70*Z`

[8N] SetTorque

(Action index: 16) Can be used to set a new target torque. Using this action without external control permission will result in an error response (see chapter "Safety considerations").

Parameters
- *Torque* (direction in - index 0): New target torque in nm.

Syntax sample request: `*A16s0*I0:6.50*Y0:1B*Z`
Syntax sample response: `*A16s0*Y0:75*Z`

[8O] SetCadence

(Action index: 17) Can be used to set a new target cadence. Using this action without external control permission will result in an error response (see chapter "Safety considerations").

Parameters
- *Cadence* (direction in - index 0, integer): New target cadence in rpm.

Syntax sample request: `*A17s0*I0:80*Y0:BB*Z`
Syntax sample response: `*A17s0*Y0:76*Z`

[8P] Start

(Action index: 12) Starts external control mode. Using this action without external control permission will result in an error response (see chapter "Safety considerations"). In order to start external control after getting permission you can either set a new target value (speed, elevation or power) or use this action to only switch or start the mode and not changing current target values. This can be helpful if you want to switch from a running device mode (e. g. cardio) to external control without setting a new target right now.

Parameters
No parameters defined

Syntax sample request: `*A12s0*Y0:71*Z`
Syntax sample response: `*A12s0*Y0:71*Z`

[8Q] Stop

(Action index: 13) Stops external control mode and the device. Using this action without external control permission will result in an error response (see chapter "Safety considerations). "ControlAllowed" will be revoked after using this action.

Parameters
No parameters defined

Syntax sample request: `*A13s0*Y0:72*Z`
Syntax sample response: `*A13s0*Y0:72*Z`

[8R] SetPersonData

(Action index: 14) Sets the current person data of the individual that is using the device. These values will affect energy consumption calculation and cardio default values. Using this action without external control permission will result in an error response (see chapter "Safety considerations").

Parameters

- *Gender* (direction in - index 0): 'F' = Female, 'M' = Male.
- *Age* (direction in - index 1, integer): age in years (1-150)
- *Height* (direction in - index 2, integer): 1 – 300 cm
- *Weight* (direction in – index 3): 1-300 kg

Syntax sample request (male, 26 years, 176 cm, 73 kg):
`*A14s0*I0:M*I1:26*I2:176*I3:73*Y0:AA*Z`
Syntax sample response: `*A14s0*Y0:73*Z`

[8S] GetPersonData

(Action index: 20) Gets the current person data of the individual that is using the device.

Parameters

- *Gender* (direction out - index 0): 'F' = Female, 'M' = Male.
- *Age* (direction out - index 1, integer): age in years (1-150)
- *Height* (direction out - index 2, integer): 1 – 300 cm
- *Weight* (direction out – index 3): 1-300 kg

Syntax sample request: `*A14s0*Y0:73*Z`
Syntax sample response (male, 26 years, 176 cm, 73 kg):
`*A14s0*O0:M*O1:26*O2:176*O3:73*Y0:C2*Z`

[8T] ResetCounterValues

(Action index: 15) Resets current parameters like time, distance, energy consumption, height. After using this action these parameters will be set to 0 but continue counting. Using this action without external control permission will result in an error response (see chapter "Safety considerations").

Parameters
No parameters defined

Syntax sample request: `*A15s0*Y0:74*Z`
Syntax sample response: `*A15s0*Y0:74*Z`

[8U] Beep

(Action index: 19) Gives external software a change to acoustical warn the user of load changes. You should use this action in order to warn users before changing any load. Using this action without external control permission will result in an error response (see chapter “Safety considerations”).

<u>Parameters</u>

- Duration (direction in – index 0, integer): Beep time in 1/100 seconds (0 to 2.55 s).

Syntax sample request: `*A19s0*I0:100*Y0:E6*Z`
Syntax sample response: `*A19s0*Y0:78*Z`

[9] Variables

h/p/cosmos coscom v4 defines one service with following variables:

Note: If no other description is given, values are defined as floating point values expected with max. two decimal places and maximal value +/- 1.000.000. Integer values are defined as signed 32-bit values. String values max.

[9A] ControlStatus

(Variable index: 0) Independent from which mode (Quickstart, Profile, Cardio, Extern, Test) is running this variable indicates the current control state. Possible values are:

0 = Stop (device is somewhere in menu)
1 = EmergencyStop
2 = Run
3 = Pause

Syntax sample query request: `*Q0s0*Y0:4E*Z`
Syntax sample query response (Run state): `*Q0s0:1*Y0:B9*Z`

[9B] ControlAllowed

(Variable index: 1) Value that indicates if external control is allowed right now. Possible values are:

- **0 = Allowed** (control can be done by external software)
- **1 = RequestPending** (User has yet to decide if external software is allowed)
- **2 = TemporaryNotAllowed** (external software is not allowed right now because running machine is in pause state. This state is between run and stop)
- **3 = NotAllowed** (control is not allowed)

Syntax sample query request: `*Q1s0*Y0:4F*Z`
Syntax sample query response (request pending): `*Q1s0:1*Y0:BA*Z`

[9C] ActualSpeed

(Variable index: 2) Gets the current speed value in m/s.

Syntax sample query request: `*Q2s0*Y0:50*Z`
Syntax sample query response: `*Q2s0:2.10*Y0:4B*Z`

[9D] TargetSpeed

(Variable index: 3) Gets the target speed value in m/s.

Syntax sample query request: `*Q3s0*Y0:51*Z`
Syntax sample query response: `*Q3s0:3.50*Y0:51*Z`

[9E] ActualElevation

(Variable index: 4) Gets the current elevation value in %.

Syntax sample query request: `*Q4s0*Y0:52*Z`
Syntax sample query response: `*Q4s0:3.50*Y0:52*Z`

[9F] TargetElevation

(Variable index: 5) Gets the target elevation value in %.

Syntax sample query request: `*Q5s0*Y0:53*Z`
Syntax sample query response: `*Q5s0:5.20*Y0:52*Z`

[9G] ActualPower

(Variable index: 6, integer) Gets the current power value in watts.

Syntax sample query request: `*Q6s0*Y0:54*Z`
Syntax sample query response: `*Q6s0:125*Y0:26*Z`

[9H] TargetPower

(Variable index: 7, integer) Gets the target power value in watts.

Syntax sample query request: `*Q7s0*Y0:55*Z`
Syntax sample query response: `*Q7s0:175*Y0:2C*Z`

[9I] ActualTorque

(Variable index: 18) Gets the current torque value in nm.

Syntax sample query request: `*Q18s0*Y0:87*Z`
Syntax sample query response: `*Q18s0:6.30*Y0:88*Z`

[9J] TargetTorque

(Variable index: 19) Gets the target torque value in nm.

Syntax sample query request: `*Q19s0*Y0:88*Z`
Syntax sample query response: `*Q19s0:6.50*Y0:8B*Z`

[9K] EnergyConsumption

(Variable index: 8) Gets the energy consumption of this workout in kj.

Syntax sample query request: `*Q8s0*Y0:56*Z`
Syntax sample query response: `*Q8s0:213.23*Y0:B9*Z`

[9L] MET

(Variable index: 9) Gets the current MET (metabolic equivalent) value of this workout.

Syntax sample query request: `*Q9s0*Y0:57*Z`
Syntax sample query response: `*Q9s0:85*Y0:FE*Z`

[9M] Time

(Variable index: 10, integer) Gets the current workout time in seconds.

Syntax sample query request: `*Q10s0*Y0:7F*Z`
Syntax sample query response: `*Q10s0:600*Y0:4F*Z`

[9N] Distance

(Variable index: 11) Gets the current distance value of this workout in meters.

Syntax sample query request: `*Q11s0*Y0:80*Z`
Syntax sample query response: `*Q11s0:10.25*Y0:B0*Z`

[9O] ActualCadence

(Variable index: 12, integer) Gets the current rotation frequency (RPM = rounds per minute).

Syntax sample query request: `*Q12s0*Y0:81*Z`
Syntax sample query response: `*Q12s0:80*Y0:23*Z`

[9P] Height

(Variable index: 13) Gets the current positive altitude difference in meters which has been gathered in this workout. In case of treadmill a negative value can be achieved if running direction is changed to backwards running.

Syntax sample query request: `*Q13s0*Y0:82*Z`
Syntax sample query response: `*Q13s0:235*Y0:56*Z`

[9Q] HeartRate

(Variable index: 14, integer) Gets the current heart rate value of the test person. If no heart rate is detected value will be 0.

Syntax sample query request: `*Q14s0*Y0:83*Z`
Syntax sample query response: `*Q14s0:140*Y0:52*Z`

[9R] RRInterval

(Variable index: 15, integer) Gets the latest RR-interval (beat-to-beat interval of the heart rate). This variable should be monitored using events in order to get all value changes. If RR-interval is not known (not supported or no heart rate available) the value will be 0.

Syntax sample query request: `*Q15s0*Y0:84*Z`
Syntax sample query response: `*Q15s0:862*Y0:5E*Z`

[9S] Errors

(Variable index: 16, semicolon separated list of errors) Gets the current error and service notes values of the device. If no error or service note is active, string will be empty. Errors start with "E" followed by three digits. Service notes start with "S" again followed by three digits e .g "E153" or "S102". Errors are defined in the user manual of the device.

Syntax sample query request: `*Q16s0*Y0:85*Z`
Syntax sample query response (errors 100 and 303): `*Q16s0:E100;E303*Y0:AB*Z`

[9T] StepHeight

(Variable index: 20) Gets the current step height in mm.

Syntax sample query request: `*Q20s0*Y0:80*Z`
Syntax sample query response: `*Q20s0:203*Y0:4F*Z`

[9U] TargetCadence

(Variable index: 21, integer) Gets the target rotation frequency (RPM = rounds per minute).

Syntax sample query request: `*Q21s0*Y0:81*Z`
Syntax sample query response: `*Q21s0:80*Y0:23*Z`

[10] Error Codes

coscom v4 defines following error codes:

Code	Description
111	Internal error prevents external command (like no connection to Frequency inverter)
112	Existing error prevents external command (device has some active error, see device manual for possible errors)
123	Invalid Protocol parameter (e. g. missing parameter in action message)
133	External command is not allowed. See chapter Request control for further information
134	Elevation system is not referenced yet
950	Wrong or missing checksum
999	Function not supported by current device. This could happen when using a function not supported for this device type (see chapter feature matrix).

[11] Extending h/p/cosmos coscom v4

If you want to extend coscom v4 with vendor specific extensions (new actions or variables) this must be placed in an extra service. In order to avoid collisions, the indices are assigned by h/p/cosmos. Please contact: andreas.feil@h-p-cosmos.com.

[12] Updates, downloads, simulators, regulatory affairs and publications

See website: http://www.coscom.org

Quality Assurance Agreements and Regulator Affairs Agreements must be agreed among all parties involved.

As h/p/cosmos at service@h-p-cosmos.com for a single page draft agreement.