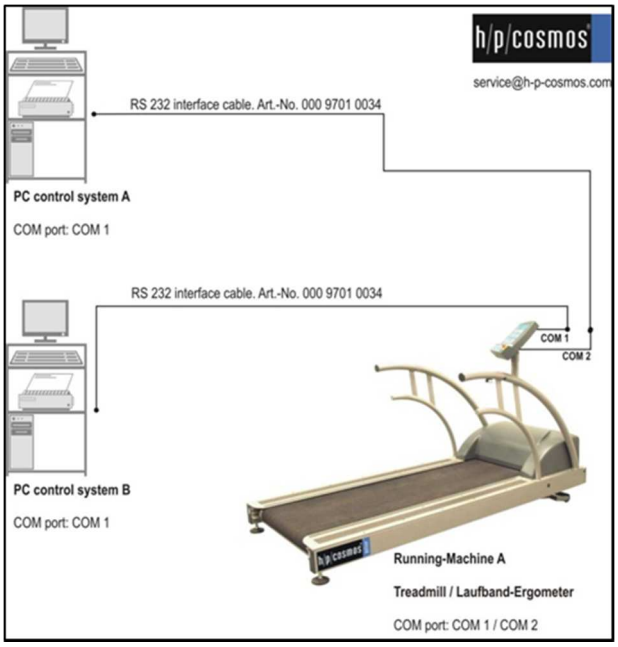


h/p/cosmos coscom® v3

device control
(example via ecg)

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
service@h-p-cosmos.com
www.h-p-cosmos.com

Author:
M. Sc. Andreas Feil, Altotec GmbH, Petting / Germany

Date: 2016-03-09

1 Content

2 Introduction

coscom® v3 offers a wide range of device control features which are all well described in the documents on www.coscom.org.
Since coscom® v3 offers an extensive and safe communication protocol to control h/p/cosmos running machines with many special feature and device diagnsiotic commands as well, the necessary documentation is quite comprehensive. This documentation may be overwhelming for some vendors like ECG software manufacturers, who basically only need commands like start/stop, status communication, increase/decrease speed and elevation and get a heart rate.

Therefore h/p/cosmos offers this pragmatical guide for minimal device control. It describes the needed commands for basic device control and the necessary model and syntax description of coscom v3. In most cases you will find short explanations and then references to the relevant documents in the already existing documentation. This document is not meant to be a complete guide. It can be seen as a short reference which will point you to the relevant sections of the full documentation.

coscom v3 is based on XDOP (Inde**x**ed **d**evice **o**bject **p**rotocol) therefore in this document you will find some references to XDOP protocol descriptions as well.

In order to operate coscom v3 safely you have to pay focus to some important safety features which are also explained in this document.

3 device model

coscom v3 defines a device model which encapsulates functionality in service objects and these services in a device object. You could compare the model to the object orientated programming model. There you have classes (coscom services) which encapsulate coherent methods (coscom actions) and properties (coscom variables). Finally the namespace (.NET) respectively package (Java) which contains these classes would be the device object.

coscom v3 currently defines eleven different services which divide functions in groups like controlling, measuring, firmware update etc. Basically you don't need to worry about the different services. The only difference in the control commands is the service index which will be used for a specific action depending in which service it is attached.

For a further description of the XDOP Device Model see the document "XDOP Device Architecture" section device model on page 2.

4 C# implementation

If you are using Microsoft .NET as your development platform you can use the h/p/cosmos .NET objects and controls in order to control running machines. In this case you don't need to know much about the protocol itself because the objects capsulate this from you.

See the tutorial documentation with sample projects for Microsoft's Visual Studio 2008 for a step to step sample on how to use the objects.

5 Syntax

coscom v3 is based on XDOP (Indexed Device Object Protocol). A complete protocol description can be found in the XDOP Device Architecture document.

Most used in XDOP are three different types of commands: Action, Queries and Events. Actions begin with an "*A", Queries with "*Q" and Events with "*E". All XDOP messages begin with an "*" followed by an upper case letter and end with an "*Z".
Actions can have parameters. Input parameters are defined with "*I" followed with the parameter index and an ":" as delimiter for the actual parameter value. An example would be "*A5s0d0*I0:10*Z" which invokes action with index 5 ("SetFailsafeTimeout") on service with index 0 on device with index 0. Note that coscom v3 allows to leave 0 indices of services and devices out (so the message "*A5*I0:10*Z" is equivalent in this sample). The input parameter with index 0 has the value 10 (in this case 1 second). Output parameters would follow the same scheme but with "*O".

coscom v3 defines only one device so the device index is always 0 which can be left out in the actual transmission.

In coscom v3 indices replace a longer textual description of action, variable, parameter and service names. You don't need to worry much about these indices. In the documentation you will find small syntax samples for all needed elements. For further sample messages at this point see document "coscom v3 protocol" section syntax samples.

6 Commands

Following you will find the basic commands a client will need in order to control an h/p/cosmos running machine using coscom v3.

6.A Polling vs. eventing

Variables like the actual speed can either be polled or received by a mechanism called eventing. Eventing means that the running machine will send variable changes through so called event messages to all subscribed clients.

Polling offers the advantage that all communication is initiated by the client. Like actions all variable queries will be done with a client request and a correspondent device response. This method will cause more traffic as values must be queried periodically but it keeps the parsing code simpler because the client knows which reply to expect.

Eventing on the other hand keeps communication to a minimum because values will only be transported when they change. But a client has to take more effort in writing the parsing code because he must be able to handle und parse event messages which can be send every time.

As many vendors of simple clients use the polling mechanism the following sections will only cover this method in order to get variable values. If you are interested in eventing see document "coscom v3 protocol" section "eventing". **Please note that your implementation has to ignore all XDOP messages which it doesn't recognize** (e. g. event messages which could be sent when a former application doesn't cancel event subscriptions properly). Therefore you have to ignore all messages that start with an unknown upper case letter (e. g. "*E", "*S", ...). Best way to achieve this is to ignore the rest of the unknown message until a message end ("*Z") is received.

6.B Opening a connection

coscom v3 is a self describing protocol. That means a device describes its functionality. In case of basic controlling you only need to worry about one description called device description. It contains the device type and nested service types. It's important to check for right versions of device and services by parsing the types and version. For further information see document "coscom v3 protocol" section "descriptions" and document "coscom v3 services notes" section "differences between h/p/cosmos devices". A more detailed description of all available descriptions is also found in the “XDOP DA 103.pdf” document.

NOTE: Do not use coscom v3 with a MCU4 device which has a firmware version between version MCU4 EPROM Firmware version 4.04.1 and 4.04.4. These versions contain implementation errors for coscom v3 features and must not be used with coscom v3. From MCU4 Firmware v 4.04.5.0025 the coscom v3 protocol can also be used, so please make sure the MCU4 Firmware was updated to 4.04.5 or higher. To check the version you can use action "GetFirmwareVersion" of service "GeneralConfig".

```
Request: *A1s0d0*Z
Response: *A1s0d0*O0:version*Z
```

Parameter "version": Returning parameter which contains the actual firmware version. It is structured like this:
Major version.subversion 1.subversion 2.Buildnumber.optional appendix
Examples: 4.04.1.0011, 4.04.1.0012.Test

MCU4 Firmware updates are available at h/p/cosmos headquarter through EPROM replacement. From MCU5 (built from November 2007 approx.) the h/p/cosmos treadmills have been equipped then with MCU5 where the firmware can be updated via RS232 interface. Please send an email to service@h-p-cosmos.com with the serial number of the treadmill based on the type plate and check the installed firmware version (will also be shown on the treadmill display for a few seconds after powering on the machine) and ask for details in case of any queries.

6.C Error Messages

coscom v3 has the ability to define errors. Every command to a device can respond in an error message from the device if the actual command couldn’t be processed (e. g. invalid parameter). You should always be able to process error responses from the device. For further details see document ""h/p/cosmos coscom v3 protocol.pdf" section "error responses".

6.D Get/Set speed

The action "SetDriveSpeed" offers a way to set a new speed with a given acceleration. You can either choose between a given acceleration index or an actual acceleration value in m/s². For further information see document ""h/p/cosmos coscom v3 protocol.pdf". *Note: You should query the actual device speed and acceleration ranges at connection establishment before setting a new speed in order to ensure that the new values are valid.* Otherwise you may get an error response (further details see document "20091022_cos100115_MCU5 device description with examples" on page 43).

To get the current speed you need to query the variable "Speed" in service "Measures".

Request: *Q2s2d0*Z
Response: *Q2s2:speedValue*Z

Parameter "speedValue": The actual speed value in m/s.

6.E Get/Set elevation

The action "SetElevatorGrade" offers a way to set a new elevator grade. This simple action has one input parameter "grade" with index 0. The value is the new final grade in %.

```
Request: *A3s4d0*I0:grade*Z
Response: *A3s4d0*Z
```

Note: You should query the actual elevator grade range at connection establishment before setting a new speed in order to ensure that the new values are valid. Otherwise you may get an error response (further details see document "20091022_cos100115_MCU5 device description with examples.pdf on page 44).

To get the current elevator grade you can query the variable "Grade" in service "Measures".

```
Request: *Q3s2d0*Z
Response: *Q3s2d0:actualGrade*Z
```

Parameter "actualGrade": The current grade in %.

6.F Increment, decrement and hold speed/elevation

Another way to change the current running machine control is to increment, decrement or hold the current speed or elevation. In this case you don't need to define a specific target value.
The used actions for changing the current speed are "IncSped" and "DecSpeed" which both have one parameter called acceleration (acceleration index):

```
IncSpeed Request: *A0s3d0*I0:accelIndex*Z
IncSpeed Response: *A0s3d0*Z

DecSpeed Request: *A1s3d0*I0:accelIndex*Z
DecSpeed Response: *A1s3d0*Z
```

Parameter accelIndex: Predefined running machine acceleration indices. 1 is the lowest acceleration, 7 the highest one. The allowed acceleration index range can be limited by running machine options (e. g. user option 27, 29 and admin option 14). In case of using an invalid index the nearest allowed one will be used.

Holding the current speed is done by using the action "HoldSpeed" of service "UserControl":

```
HoldSpeed Request: *A2s3d0*Z
HoldSpeed Response: *A2s3d0*Z
```

Changing the running machines current grade uses the actions "IncGrade", "DecGrade" and "HoldGrade" which all have no parameters.

```
IncGrade Request: *A3s3d0*Z
IncGrade Response: *A3s3d0*Z

DecGrade Request: *A4s3d0*Z
DecGrade Response: *A4s3d0*Z

HoldGrade Request: *A5s3d0*Z
HoldGrade Response: *A5s3d0*Z
```

6.G Start and Stop running machine

The action "StartManual" of service "UserControl" starts the running machine with the predefined start speed (running machine option). It also resets the counters for distance, energy and time.

```
Request: *A13s3d0*Z
Response: *A13s3d0*Z
```

Alternatively you can use the "SetSpeed" action to start the running machine although there will be no reset of time, energy and distance.

In order to stop the running machine you can use the action "Stop" of service UserControl. This action stops the running machine and the actual mode. It's the same as pressing the "Stop" key at the user terminal.

```
Request: *A9s3d0*Z
Response: *A9s3d0*Z
```

6.H Get Heart Rate

You can get the actual measured heart rate by using following query command:

```
Request: *Q4s2d0*Z
Response: *Q4s2d0:heartRateValue*Z
```

Parameter heartRateValue: the actual heart rate value in bpm.

6.I Getting distance, energy, MET, power and time

In addition to receiving speed, grade and heart rate you can get values for distance, energy, MET, power and time through service "Measures". All these parameters are defined as coscom v3 variables and can be queried any time. For further information on variable indices see document ""20091022_cos100115_MCU5 device description with examples" on page 74.

7 Safety features

In order to ensure a safe communication your implementation has to use the following safety features. Following is a small overview of the available safety features. ***For further information (and syntax samples) see document "h/p/cosmos coscom v3 protocol.pdf" section "Safety Features and Implementation notes".***

7.A Failsafe

The failsafe action is probably one of the most important safety features of the running machine. It sets a safety timeout in case the interface communication is interrupted (e. g. someone pulls the RS232 interface plug). If that timeout elapses the running machine will stop automatically.

7.B Stop-status

In case the running machine was stopped via STOP button on the running machine or via any other reason (for example after fall stop or after power off), the host program (for example stress test ECG) must realize this “stop status” and must terminate automatically any control functions of speed and elevation control of the running machine. Otherwise an automatic load profile (for example Naughton speed protocol) may still continue on the ECG and start the running belt automatically after time elapsed for the next stage. So a subject/patient could be surprised by unexpected acceleration or deceleration, could fall and suffer from serious injury.

7.C Checksum

coscom v3 defines an optional message checksum. The checksum is the last part of a coscom message and is indicated by the index Y0. It's necessary because coscom v3 is transported over RS232 which is not reliable. Therefore messages may be changed over transport which could result in false commands. h/p/cosmos coscom v3 does not use this checksum automatically. You have to enable this safety feature using the action "SetProtocolParams" of service "GeneralConfig".

```
Request: *A0s0d0*I0:processChecksum*I1:appendCRLF*I2:outputDescNames*Z
Response: *A0*Y0:9B*Z (response includes checksum and has left out 0 indices)
```

Parameters
processChecksum (Boolean)
Enables the processing of a 2 hex digit (modulo 256) checksum with element Y0.

appendCRLF (boolean)
Appends the characters CR and LF at the end of a message (after *Z).

outputDescNames (boolean)
Enables the output of the optional index names in the description lines.

7.D Warn users before changes

You should warn a user before automatic changes of speed, acceleration and elevation. You can do this by visual effects in your application or/and by using the buzzer of the running machine device. The action “Beep” in service Terminal is designed to activate the buzzer for a certain time.

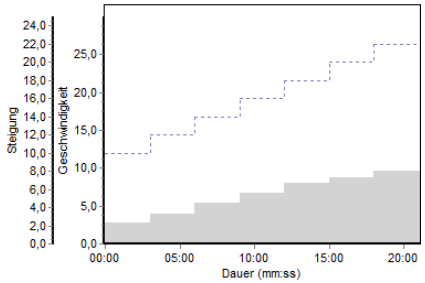
8 Test protocol

As every application is different there is no general way to check the correct protocol implementation. Therefore h/p/cosmos does not define a specific test protocol. In order to operate at a safe level your software should be implemented following this guide.

If h/p/cosmos should verify your implementation we would like to see your software test specification. In some cases we would like to analyze the communication between the running machine and your application and check if safety features, proper error message handling and connection establishment are implemented. If this should be necessary a demo version of your software will be needed.

9 Example Bruce protocol

Following a short sample protocol with seven steps should demonstrate the coscom v3 syntax.

Bruce protocol e.g. for ECG stress test  	Step	Duration	Speed	Elevation
	1	3:00 min	2.7 km/h	10 %
	2		4.0 km/h	12 %
	3		5.4 km/h	14 %
	4		6.7 km/h	16 %
	5		8.0 km/h	18 %
	6		8.8 km/h	20 %
	7		9.6 km/h	22 %

9.A Syntax

For a save operation of coscom v3 (using polling) the syntax for the above mentioned Bruce protocol could be:

Direction	Coscom v3	Description
To RM (running machine)	`*D0*Z`	Root Description Request

From RM	`*D0:`
	`0:=simple`
	`1:=1`
	`2:=1`
	`3:=255`
		Root Description Response
		Device Type is "urn:schemas-coscom-org:device:MCU5Treadmill:1"
		(optional description names are turned of as default)

	4:= 0:=0 1:=urn:schemas-coscom-org:device:MCU5Treadmill:1 2:=uuid:f1daec20-583c-11db-b0de-010087654321 *Z	
To	*A1*Z	Action GetFirmwareVersion
From	*A1*O0:1.06.4.0057*Z	Version: 1.06.4.0057
To	*A0*I0:1*I1:0*I2:1*Z	Action SetProtocolParams processChecksum: 1 appendCRLF: 0 outputDescNames: 1
From	*A0*Y0:9B*Z	SetProtocolParams answer with Checksum: 9B
To	*A5s4*Y0:47*Z	GetDriveAccelDecelRange
From	*A5s4*O0:1*O1:5*O2:0.046625*O3:2.377864*Y0:77*Z	GetDriveAccelDecelRange minIndex: 1 maxIndex: 5 minValue: 0.046625 m/s² maxValue: 2.377864 m/s²
To	*A4s4*Y0:46*Z	GetDriveSpeedRange
From	*A4s4*O0:0.000000*O1:6.111111*Y0:15*Z	GetDriveSpeedRange minSpeed: 0.000000 m/s maxSpeed: 6.111111 m/s
To	*A6s4*Y0:48*Z	GetElevatorGradeRange
From	*A6s4*O0:0.000000*O1:24.000000*Y0:41*Z	GetElevatorGradeRange minGrade: 0.000000 maxGrade: 24.000000
	Periodic setting of failsafe (within 1 second)	
To	*A5*I0:10*Y0:DE*Z	SetFailSafeTimeout newTimeout: 10
From	*A5*O1:0*Y0:B4*Z	SetFailSafeTimeout oldTimeout: 0 (First set)
To	*A5*I0:10*Y0:DE*Z	SetFailSafeTimeout newTimeout: 10
From	*A5*O1:10*Y0:E5*Z	SetFailSafeTimeout oldTimeout: 10 (must be 10 because it was set before)
	…	
	Start protocol with first step	
To	*Q1s3*Y0:52*Z	Query ControlStatus
From	*Q1s3:0*Y0:BC*Z	ControlStatus: 0 (Stop)
To	*A13s3*Y0:75*Z	StartManual request

From	*A13s3*Y0:75*Z	StartManual response
To	*A2s4*I0:0.75*I1:2*I2:0*Y0:0A*Z	SetDriveSpeed speed: 0.75 m/s accelIndex: 2 acceleration: 0
From	*A2s4*Y0:44*Z	SetDriveSpeed
To	*A3s4*I0:10*Y0:83*Z	SetElevatorGrade grade: 10 %
From	*A3s4*Y0:45*Z	SetElevatorGrade
To	*Q1s3*Y0:52*Z	Query ControlStatus
From	*Q1s3:1*Y0:BD*Z	ControlStatus: 1 (Run)
	Periodic query of control status and setting failsafe	
To	*Q1s3*Y0:52*Z	Query ControlStatus
From	*Q1s3:1*Y0:BD*Z	ControlStatus: 1 (Run)
To	*A5*I0:10*Y0:DE*Z	SetFailSafeTimeout newTimeout: 10
From	*A5*O1:10*Y0:E5*Z	SetFailSafeTimeout oldTimeout: 10
	Warn user before next step with a beep	
To	*A4s1*I0:100*Y0:B1*Z	Beep duration: 100 (1 second)
From	*A4s1*Y0:43*Z	Beep
	Set next step	
To	*A2s4*I0:1.11*I1:2*I2:0*Y0:01*Z	SetDriveSpeed speed: 1.11 m/s
From	*A2s4*Y0:44*Z	SetDriveSpeed
To	*A3s4*I0:12*Y0:85*Z	SetElevatorGrade grade: 12 %
From	*A3s4*Y0:45*Z	SetElevatorGrade
	Other steps here	
	Stop protocol	
	*A9s3*Y0:4A*Z	Stop
	*A9s3*Y0:4A*Z	Stop

9.B **Syntax Stop**

The syntax for the event, that somebody would stop the running machine with the STOP button on the terminal while the Bruce protocol still tries to control the running machine from PC or ECG:

	*Q1s3*Y0:52*Z	Query ControlStatus
	*Q1s3:0*Y0:BC*Z	ControlStatus: 0 (Stop)

9.C **Syntax Failsafe**

The syntax for the event, that the failsafe function stops the running machine automatically due to a broken interface cable during the Bruce protocol test: In this case you will not receive a reply for your coscom v3 requests until the interface cable is restored. After that the communication will show a stopped running machine due a failsafe timeout:

	*Q1s3*Y0:52*Z	Query ControlStatus
	*Q1s3:0*Y0:BC*Z	ControlStatus: 0 (Stop)
	*A5*I0:10*Y0:DE*Z	SetFailSafeTimeout newTimeout: 10
	*A5*O1:0*Y0:B4*Z	SetFailSafeTimeout oldTimeout: 0 (should have been 10 so failsafe must be occurred)

