h/p/cosmos coscom v3 services notes

basic device control

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 GmbH, Altofing 7, 83367 Petting / Germany

Date: 2009-09-24

1 Content

2 Introduction

This document gives an overview of the available coscom v3 services. You should be familiar with the coscom v3 protocol in order to fully understand the explanations. Please refer the following documents for an explanation of the coscom v3 protocol:

- *XDOP Device Architecture:* h/p/cosmos coscom v3 is based on XDOP (Indexed Device Object Protocol). This Document contains a complete protocol and device model description of XDOP.
- *h/p/cosmos v3 protocol – Implementation notes:* This document contains additional explanations of the h/p/cosmos coscom v3 protocol. It also contains a **list of important safety features** which should be considered when implementing coscom v3.
- *MCU4 respectively MCU5 device description with examples:* Contains a complete listing of all coscom v3 services, actions and variables. You will find a brief overview of all later mentioned actions and variables in this document.
- *h/p/cosmos discovery device description:* Contains a complete listing of all coscom v3 services, actions and variables of the h/p/cosmos discovery ergometer. Basicly the interface is almost the same as the one of the h/p/cosmos running machine devices.

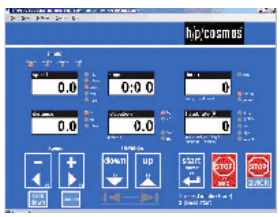
This document doesn't contain a formal description of all possible coscom v3 features and its protocol represantation. It concentrates on explaning basic coscom v3 features which can be used for a safe basic device control. Therefore only actions and variables which are needed for a basic control of a running machine are discussed.

3 The coscom protocol

The h/p/cosmos coscom interface protocol has it's origin in the year 1992 and has been developed for safe, reliable and advanced communication, control and links between different ergometers like running machines, treadmills, bicycle ergometers, ladder ergometers etc., as well as control and monitoring equipment like PC, EMG, ECG, EKG, ergospirometry, VO2max systems, metabolic carts, cardiopulmonary stress test systems, biomechanics and motion analysis systems, fitness and sports medical as well as lactate evaluation and analyzing software, etc.

4 Important Notes, Safety Precautions, Warnings

The coscom v3 .NET Controls can be used with h/p/cosmos running machines and OEM running machines which are equipped with MCU4 and MCU5 control board (UserTerminal).

			
h/p/cosmos mercury med	h/p/cosmos mercury lt med	h/p/cosmos para control® 4.0	MCU 5 control board

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 can be used, so please make sure the Firmware was updated this 4.04.5 or higher.

Pay attention to all safety instructions and warnings as well as the chapters “intended use” and “forbidden use” of the operation and service manual of the h/p/cosmos running machines.

Always activate “failsafe timeout mode”, so the running machines stops automatically in case of termination of the interface communication.

Always communicate the “status” mode of the running machine with any host programs.
In case the running machine was stopped via STOP button on the running machine or via any other reason (for example power failure for the running machine supply), the host program must realize this “stop status” and must terminate automatically any control functions of speed any elevation control of the running machine. If not, a user may be caught by surprise by an unexpected command from the host program, although the user pressed STOP on the running machine before. This may lead to serious accident.

For further safety features of coscom v3 and a small guide to a safe coscom v3 implementation refer the document “coscom v3 implementation notes”.

5 Differences between h/p/cosmos devices

Currently there are two different generations of h/p/cosmos running machine devices on the market (not including long used running machines with out of date MCU2 (Mikrocontroller Unit 2) control board). Regarding the coscom v3 protocol these generations are almost the same.

5.A Differences between MCU4 and MCU5 devices

These running machines differ in the type of control board they are using. MCU4 was produced till late 2008. After that the new MCU5 control board replaced the MCU4. Although the hardware is different there are only a few differences in the protocol:

- Device description type: The MCU5 has a different device description type than the MCU4. The device description type is a unique identifier which describs the actual type of device. If you want to search for a h/p/cosmos coscom v3 running machine you should query the device description and verify that the device type is either a MCU4 or MCU5 type string.
- Flash Update Service: Another difference is that the MCU5 has an additional service. This MCU5FlashUpdate Service is used the update the control board and only for internal use. Do not use this service in your own application.

5.B h/p/cosmos discovery device

h/p/cosmos also produces special products like the h/p/cosmos discovery. Among other things this ladder ergometer is used for firefighter training. The device is also MCU5 based and h/p/cosmos cosvom v3 is implemented. It has the same coscom services as the “normal” MCU5 running machine but due the different construction not all features are implemented. The device for example has no variable elevation. Therefore the SetGrade Action is not implemented and will respond with an error response message. In order to differentiate the machine type from the h/p/cosmos running machine types the h/p/cosmos discovery has also a different device type string in the coscom v3 device description.

If you want to search for all h/p/cosmos coscom v3 ergometer machines you should query the device description and verify that the device type is either MCU4, MCU5 running machine or MCU5 ladder ergometer.

6 coscom services

This section contains a description of the h/p/cosmos coscom v3 services which are needed for a basic control of a running machine device. Not all possible actions and variables are discussed. For a complete List of all actions and variables refer to the device description documents. You won't find any protocol examples in this document. For examples on how to build the correspondending coscom v3 messages for the elements refer the MCU4 respectively MCU5 device description documents. These documents contain protocol examples for all actions. For a basic coscom protocol description see the XDOP device architecture document.

6.A Service TreadmillGeneralConfig

This service can be used to get general information about the running machine and to make general configurations. It also contains an important safety feature of the device, the failsafe feature.

6.A1 Important actions

Following you will find some important actions of the service. The actual data type of arguments and variables is mentioned in Brackets. For a list of all available data types and there value ranges see the XDOP device architecture document.

▌GetFirmwareVersion

With this action you can query the firmware of the device. It is important to check which firmware version the controlling device has when errors occure. For that you should query the firmware version and quote it in error reports.

Action parameters:

version (string): 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

▌GetRTCTime

This action gets the time fields of the battery-buffered real time clock.

Action parameters:

sec (ui1): Returning parameter which contains the RTC seconds (0 – 59)
min (ui1): Returning parameter which contains the RTC minutes (0 – 59)
hrs (ui1): Returning parameter which contains the RTC hours (0 – 23)
day (ui1): Returning parameter which contains the RTC day (1 – 31)
month (ui1): Returning parameter which contains the RTC month (1 – 12)
year (ui2): Returning parameter which contains the RTC year (2000 – 2099)

▌GetSerialNumber

Gets the serial number of the device. It consists of 3 characters and an 8-digit number.
Please query the serialnumber of the device and quote it in error reports. The usual string representation is build like following example: abc-12345678.

Action parameters:

code1 (char): Returning parameter which indicates the first character

code2 (char): Returning parameter which indicates the second character
code3 (char): Returning parameter which indicates the third character
num8 (ui4): Returning parameter which indicates the 8-digit number

SetFailSafeTimeout

The failsafe action is a basic security feature 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.

You should constantly set this timeout in your application in order to check if a failsafe occurred.
Note: The failsafe will only occur when the running machine is actually in run mode. In selection mode you can set the timeout but it will not elapse till the running machine is started. If a failsafe elapses you have to set it again in order to restart the feature. It will not restart automaticly.

Action parameters:

newTimeout (byte): Sets the new timeout in 1/10 seconds (from 0 to 25.5 seconds). You should use a small timeout interval (<= 1 sec.) in order to quickly dectect an occurring failsafe.

oldTimeout (byte): Returning parameter that indicates the last set value of the timeout. This returning parameter is very useful because it indicates the state of the failsafe. If the fail safe wasn't set before it will be 0. If the fail safe was set before it should contain the previously set value, if it doesn't a failsafe occurred since you last set the time out.

6.B Service TreadmillDirectControl

The direct control service has actions to control the speed of the drive and the grade of the elevator directly. This is the most important service for basic device controlling.

6.B1 Important actions

Following you will find some important actions of the service.

GetDriveAccelDecelRange

This action gets the configured acceleration and deceleration range. For safety reasons these values can be limited. These values are for RS232 control only and can vary from the acceleration settings of the device itself. The actual allowed acceleration- and deceleration values for RS232 control depend on several user and administrator settings of the device. Please refer the h/p/cosmos running machine manual for a detailed options overview.

The acceleration value can be described in two different ways. You can either use one of the predefined acceleration level values or you can explicitly specify an extact acceleration value in m/s². Following acceleration level values exist:

1 = (Max. FU speed) / 131 s
2 = (Max. FU speed) / 65.5 s
3 = (Max. FU speed) / 32.8 s
4 = (Max. FU speed) / 16.0 s
5 = (Max. FU speed) / 8.0 s

6 = (Max. FU speed) / 5.0 s
7 = (Max. FU speed) / 3.0 s

The Max. FU speed is device dependend. Tbd (is max fu speed same as getdrivespeedrange value? Probably not)

Action parameters:
minIndex (ui1): Returning parameter which indicates the minimum acceleration/deceleration index value
maxIndex (ui1): Returning parameter which indicates the maximum acceleration/deceleration index value
minValue (r4): Returning parameter which indicates the minimum acceleration/deceleration value [m/s²]
maxValue (r4): Returning parameter which indicates the maximum acceleration/deceleration value [m/s²]

GetDriveSpeedRange

This action gets the configured speed range settings. In normal device configuration (forward running) this setting gets positive speed values. If the device is switched to reversed direction (backward running) the settings will get the maximal reserved speed.

Action parameters:

minSpeed (r4): Returning parameter which indicates the minimum speed for the current mode (forward or reverse running) in m/s
maxSpeed (r4): Returning parameter which indicates the maximum speed for the current mode (forward or reverse running) in m/s

GetDriveStatus

The status of the drive is very important for external control. You can query this information with this action or consuming automatic events with the variable “DriveStatus” of this service. Either way if you detect a “stopped” status you must terminate automatically any control functions of speed and elevation control of the running machine. If not, a user may be caught by surprise by an unexpected command from the external program, although the user pressed STOP on the running machine before. This may lead to serious accidents.

Action parameters:

driveStatus (ui1): Returning parameter that indicates the actual drive status. Possible values are:
0 = Stopped (RFR off)
1 = Constant speed
2 = Acceleration
3 = Deceleration
actualSpeed (r4): Returning parameter which holds the actual speed of the running machine in m/s
finalSpeed (r4): Returning parameter which holds the final speed which was set in m/s
acceleration (r4): Returning parameter which holds the current acceleration in m/s²

GetElevatorGradeRange

This action gets the configured grade range settings. The values are device and option dependend.

Action parameters:

minGrade (r4): Returning parameter which indicates the minimum grade in %
maxGrade (r4): Returning parameter which indicates the maximum grade in %

SetDriveSpeed

With this action a new speed and acceleration respectively deceleration can be set. You should query the actual device speed and acceleration ranges before, in order to verify that the desired values lie within the permitted range. Before making automatic speed changes you should display the user a note. Unexptected changes may catch the user by surprise which can lead to serious accidents.

Beside of settting a new speed value you must declare an acceleration respectively deceleration value. You can either use a predefined index value **or** an exact acceleration value in m/s². If you set an index value other than "0" the exact accerlation value will be ignored. On the other side if you want to use an exact value you must specify "0" for the index value.

Action parameters:

speed (r4): The new final Speed in m/s
accelIndex (ui1): The acceleration level index. Following values are possible:

0 = Use "acceleration" argument
1 = (Max. FU speed) / 131 s
2 = (Max. FU speed) / 65.5 s
3 = (Max. FU speed) / 32.8 s
4 = (Max. FU speed) / 16.0 s
5 = (Max. FU speed) / 8.0 s
6 = (Max. FU speed) / 5.0 s
7 = (Max. FU speed) / 3.0 s

acceleration (r4): The exact acceleration value in m/s². You can always specify positive values regardless of acceleration or deceleration.

Action errors:

Error code 831: Speed value is out of range. To avoid this error, query the allowed speed range with the action "GetDriveSpeedRange" before.

SetElevatorGrade

With this action a new elevation grade can be set. You should query the actual device elevation range before, in order to verify that the desired value lies within the permitted range. Before making automatic elevation changes you should display the user a note. Unexptected changes may catch the user by surprise which can lead to serious accidents.

Action parameters:

grade (r4): The new final Grade in %

Action errors:

Error code 832: Grade value is out of range. To avoid this error, query the allowed elevation range with the action "GetElevatorGradeRange" before.

6.B2 Important Variables

Following you will find some important variables of the service.

Acceleration
The Accerlation (type: r4) describes the changes of the running machine speed. Values greater 0 mean an actual accerleration in m/s², values lower then 0 desribe an actual deceleration in m/s².

FinalSpeed
The FinalSpeed (type: r4) describes the final speed value in m/s which was set either by an external software or on the running machine terminal.

ActualSpeed
The ActualSpeed (type: r4) variable describes the actual speed value of the device in m/s. This value can be different to the FinalSpeed value.

DriveStatus
The DriveStatus (type: ui1) variable describes the status of the drive. As mentioned above if you detect a "stopped" status you must terminate automatically any control functions of speed and elevation control of the running machine. If not, a user may be caught by surprise by an unexpected command from the external program, although the user pressed STOP on the running machine before. This may lead to serious accidents.
Following values are possible:

0 = Stopped (RFR off)
1 = Constant speed
2 = Acceleration
3 = Deceleration

6.C Service TreadmillMeasures

This service can be used to monitor parameters like distance, grade, speed, time and heartrate. It defines variables for these parameters which can easily monitored through coscom v3 eventing. In order to use this service properly examine action SetEventMask.

6.C1 Important actions

Following you will find some important actions of the service.

ResetDistance
This action enables you to reset the distance value to 0 meters. Similar actions exist for some other parameters like energy and time.

SetEventMask
This action controls the behavior of the service. Once eventing is enabled (event subscription for this service is send to the device) the service will send automatic changes for the defined variables. Because sending all variable changes can result in a flood of event messages (e. g. when the running machine speed changes fast many speed and possibly acceleration events will be sent) you can control the variables for which events should be sent. This comes in handy when you e. g. want to monitor the time variable but don't need all other variables. In this case you only enable eventing for this one variable and all other changes of speed, acceleration and so on will not be sent.

Action parameters:

eventMask (bin.hex): A bitmask that indicates the variables which should be evented. Following bits are available (Enabling all variables results in a bitmask of 0xFF):

time = 0x01
distance 0x02
speed: 0x04
grade 0x08
heart rate 0x10
power 0x20
energy 0x40
MET 0x80

6.C2 Important Variables

Following you will find some important variables of the service.

▌Variable Distance, Grade, Speed, HeartRate and Time

For each parameter a variable is defined. In order to get the actual values these variables can either explicitly be queried or variable events can be enabled. See action SetEventMask for further details on eventing these variables.

6.D Service TerminalLCD6x4Panel

This is a service to build a remote terminal with the same display and keyboard layout as the built-in running machine terminal. It contains the variables for the 6 displays and actions to control and monitor the key state of the 6 terminal keys. Because the description of the display variables are higly the same, only the LLDisplay Variable will be described. The layout of the built-in display looks like following schema:

- ULDisplay: Speed
- UMDisplay: Time
- URDisplay Index
- LLDisplay: Distance
- LMDisplay: Elevation
- LRDisplay: Heart rate

Note: Altough theses displays are generally bound to a specific value they can show other information like error messages or option settings. Therefore you must not swap the layout of this displays strings in your application.

6.D1 Important actions

Following you will find some important actions of the service.

▌Beep

This action switches the buzzer of the running machine terminal on. The beep lasts as long as the given duration argument. The buzzer will automaticly beep when a terminal key is pressed. You should use these feature to point out a new speed or elevation value to the user.

Action parameters:

duration (ui1): The beep time in 1/100 seconds (0 to 2.55 s)

GetDistanceUnit

This action gets the configured distance unit. There are three led groups of which you can query the selected Unit. These are Distance, Speed and Elevation. All actions are basicly the same why only this action will be explained. For unit values see the device description documentation document.

Action parameters:

unit (ui1): Returning parameter which holds the selected unit. Possible values are:
0 = km, 1 = miles, 2 = m

KeyDown

This action changes the key state of a terminal button to "key down" (key pressed).

Action parameters:
key (char): The key which is to be pressed: Possible values are:
+ = Minus, - = Plus, U = Up, D = Down, E = Start, S = Stop, F = Emergency Stop

KeyUp

This action changes the key state of a terminal button to "key up" (key released).

Action parameters:
key (char): The key which is to be pressed: Possible values are:
+ = Minus, - = Plus, U = Up, D = Down, E = Start, S = Stop, F = Emergency Stop

SetDistanceUnit

This action sets the distance unit temporarily. If the running machine is switched off or if the configuration mode is entered, then this setting is lost. There are three led groups of which you can set the selected Unit. These are Distance, Speed and Elevation. All actions are basicly the same why only this action will be explained. For unit values see the device description documentation document.

Action parameters:
unit (ui1): The unit which should be set. Possible values are:
0 = km, 1 = miles, 2 = m

Action Errors:

Error code 820: Invalid unit value. The given unit value is not possible.

6.D2 Important Variables

Following you will find some important variables of the service.

Variable LLDisplay

The LLDisplay (type: string) variable describes the content of the lower left LCD display of the built-in user terminal. Each state change sends an event, if the change rate isn't too high. This string has following attributes:

- the first 4 characters build the display string
- the fifth character (hexdecimal) describes the bit field for blinking digits. The bit field is:
  - 0x0 No blinking digits
  - 0x1 Left digit is blinking

  - 0x2 Middle left digit is blinking
  - 0x4 Middle right digit is blinking
  - 0x8 Right digit is blinking
- the sixth character (hexdecimal) describes the bit field for points between the digits
  - 0x0 No points
  - 0x1 Left point
  - 0x2 Middle point
  - 0x4 Right point
  - 0x8 Colon
- the seventh character (hexdecimal) describes the bit field for blinking points
  - 0x0 No blinking points
  - 0x1 Blinking left point
  - 0x2 Blinking middle point
  - 0x4 Blinking right point
  - 0x8 Blinking colon

Example: " 00040" shows the string " 00" with a right point and looks like " 0.0"

The expression blinking points or blinking digits describe if the digit/point should blink in the display. If set you can manage a synchron blinking to the original built-in terminal with the variable BlinkState. This variable switches between 0 = off and 1 = on. When you hide the according digits/points on value 0 (off) and showing them on 1 (on) you have a synchron blinking effect according to the built-in terminal.

6.E **Service DataLogger**

This service provides a configurable data logger. The output of the logger is evented. The service is not essential for a basic remote control but if you only want to log some basic parameter information you simply can use this service.

6.E1 **Important actions**

Following you will find some important actions of the service.

▌SetColumnDelimiter
Sets the column delimiter character. This character will be used to deliminate the columns which should be recorded in the SetRecordColumns action.

Action parameters:

delimiter (char): The delimiter character for delimiting the column values in the log record

▌SetRecordColumns
This action sets the columns for the log record. For a list of available columns (parameters) see the action parameter.

Action parameters:

columnList (string): A string with separated indexes of columns, which should be logged:

1 = Time
2 = Distance
3 = Speed

4 = Grade
5 = Heart Rate
6 = Power
7 = Energy
8 = MET

The standard delimiter is a semi-colon. This can be changed by using the SetColumnDelimiter action. In order to log e. g. the values for time, speed and grade the string should be like “1;3;4”.

SetRecordEventPeriod
This action sets the event period of the logger

Action parameters:

period (ui1): The event period (time between two log record outputs) in seconds

StartLogging
This action starts the logging mechanism.

StopLogging
This action stops the logging mechanism.

6.E2 Important Variables

Following you will find some important variables of the service.

Variable Record
The variable Record (type: string) contains a string with the with all configured column values.

6.F Service TreadmillUserContol

This service instantiates a higher level control interface in contrast to the terminal or direct control interface. It models mainly the user control concept of the user terminal. For example one can start a specific test, which is implemented in the running machine computer. You can use these actions in your application but they are not essential for a basic control. At this point only the main actions will be discussed. For further information see the device description documentation document.

6.F1 Important actions

Following you will find some important actions of the service.

IncSpeed
This action increases the running machine speed.

Action parameters:

accelIndex (ui1): Index (1 - 7) for an acceleration table with 7 values.
1 = lowest acceleration
...
7 = highest accelartion

DecSpeed

This action decreases the running machine speed.

Action parameters:

accelIndex (ui1): Index (1 - 7) for an acceleration table with 7 values.
1 = lowest acceleration
...
7 = highest accelartion

HoldSpeed

This action holds the actual speed (no acceleration or deceleration). No action parameters are defined.

IncGrade, DecGrade and HoldGrade

These actions enable you to control the elevator. The elevation level can be increased, decreased or set to hold the actual level. No arguments are defined for these actions.

Stop

The action stops the treadmill and the actual mode. It's the same as pressing the key "Stop" at the user terminal.

SetPersonData

This action sets all available person data. **Note:** These settings are temporal. When the running machine is in selection mode this parameters will be reset to default values within 30 seconds.

Action parameters:

sex (char): the gender of the person. 'F' = female, 'M' = male

age (ui1): the person age in years.

weight (ui1): the weight in kg.

size (ui1): the person size in cm.

Action errors:

Error code 814: Invalid person parameter. At least one of the parameters contains an invalid value.

GetPersonData

This action gets all available person data.

Action parameters:

sex (char): Returning parameter that indicates the gender of the person. ‘F’ = female, ‘M’ = male

age (ui1): Returning parameter that indicates the person age in years.

weight (ui1): Returning parameter that indicates the weight in kg.

size (ui1): Returning parameter that indicates the person size in cm.

6.G Service TreadmillOptions

This service is for intern use only. Do not use this service in you own applications.

6.H Service MCU5FlashUpdate (only MCU5)

This service is for intern use only. Do not use this service in you own applications.

6.I Service SunTechTangoStressBPBridge

This service manages a communication bridge to the SunTech Tango Blood Pressure device. If the Tango device is connected with one of the other serial ports of the running machine, it can be controlled with the actions of this service. See SunTech Tango manual for more details or contact h/p/cosmos. For a basic control this service is not needed.

6.J Service TreadmillProfileEditor

With the profile editor all stored profiles can be read and the user profiles can be edited and saved. With the basic control features shown so far you can easily implement your own profile control and only set the speed and elevation values of the device. Therefore this service is not needed for a basic control. For further information see the device description documentation document.

7 h/p/cosmos coscom v3 .NET Objects

You can either implement the protocol yourself or if you’re using the Microsoft .NET Framework you can use the h/p/cosmos coscom v3 .NET Objects. These class libraries implement the coscom v3 protocol and define a high level interface for the coscom v3 device and service model. You will find a method for every action that the h/p/cosmos running devices define. For further information see www.coscom.org.

8 h/pcosmos coscom v3 .NET Controls

If you want to rebuild the build-in control and you’re using the Microsoft .NET Framework you can use the h/p/cosmos coscom v3 .NET Controls. This class library contains controls for the 6 displays and user keys. For further information and a small sample project for visual studio visit www.coscom.org.

